Question #1 *Topic 1*

Which of the following describes the Spark driver?

A. The Spark driver is responsible for performing all execution in all execution modes – it is the entire Spark application.

B. The Spare driver is fault tolerant – if it fails, it will recover the entire Spark application.

C. The Spark driver is the coarsest level of the Spark execution hierarchy – it is synonymous with the Spark application.

D. The Spark driver is the program space in which the Spark application's main method runs coordinating the Spark entire application.

E. The Spark driver is horizontally scaled to increase overall processing throughput of a Spark application.

Suggested Answer: *D*

Community vote distribution

Saurabh_221b 2 months, 3 weeks ago

Selected Answer: D

The Spark driver runs the main method of the Spark application and coordinates the execution of tasks across the cluster.

upvoted 1 times

s127 6 months, 2 weeks ago

Selected Answer: D

spark is responsible for executing the main method of program and also collaborates with cluster manager and worker nodes as well.

upvoted 1 times

zic00 10 months, 2 weeks ago

Selected Answer: D

The Spark driver is responsible for orchestrating the execution of a Spark application, managing the SparkContext, and coordinating the execution of tasks across the Spark cluster. It does not perform the execution of tasks itself but rather schedules tasks on the worker nodes. The Spark driver is not fault-tolerant in the sense that if it fails, the entire Spark application usually fails. It also does not scale horizontally; only the executors (worker nodes) do that.

upvoted 1 times

Raheel_te 1 year ago

ignore my previous comment.

E is the answer as per the sample exam in Databricks site.

upvoted 1 times

Raheel_te 1 year ago

B is the answer as per the sample exam in Databricks site.

upvoted 1 times

SnData 1 year, 1 month ago

D is the answer

upvoted 1 times

Vikram1710 1 year, 2 months ago

Answer -D

upvoted 1 times

Pankaj_Shet 1 year, 8 months ago

Please let me know what are these dumps used for?

Scala - Spark or Python Spark?

upvoted 1 times

singh100 1 year, 11 months ago

Answer: D

Receives the user's code and breaks it into tasks for execution.

Orchestrates the execution plan and optimizes the Spark job.

Coordinates with cluster managers to allocate resources for tasks.

Collects and aggregates results from distributed workers.
Maintains the metadata and state of the Spark application during its execution.
upvoted 4 times

TmData 2 years ago

The correct answer is D. The Spark driver is the program space in which the Spark application's main method runs, coordinating the entire Spark application.

Explanation: The Spark driver is a program that runs the main method of a Spark application and coordinates the execution of the entire application. It is responsible for defining the SparkContext, which is the entry point for any Spark functionality. The driver program is responsible for dividing the Spark application into tasks, scheduling them on the cluster, and managing the overall execution. The driver communicates with the cluster manager to allocate resources and coordinate the distribution of tasks to the worker nodes. It also maintains the overall control and monitoring of the application. Horizontal scaling, fault tolerance, and execution modes are not directly related to the Spark driver.
upvoted 4 times

4be8126 2 years, 2 months ago

Selected Answer: D

D. The Spark driver is the program space in which the Spark application's main method runs coordinating the Spark entire application.

The Spark driver is responsible for coordinating the execution of a Spark application, and it runs in the program space where the Spark application's main method runs. It manages the scheduling, distribution, and monitoring of tasks across the cluster, and it communicates with the cluster manager to acquire resources and allocate them to the application. The driver also maintains the state of the application and collects results. It is not responsible for performing all execution in all execution modes, nor is it fault-tolerant or horizontally scalable.
upvoted 3 times

Question #2 *Topic 1*

Which of the following describes the relationship between nodes and executors?

A. Executors and nodes are not related.

B. Anode is a processing engine running on an executor.

C. An executor is a processing engine running on a node.

D. There are always the same number of executors and nodes.

E. There are always more nodes than executors.

Suggested Answer: *D*

Community vote distribution

C (100%)

TC007 Highly Voted 1 year, 9 months ago

Selected Answer: C

In a Spark cluster, each node typically has multiple executors, which are responsible for executing tasks on that node. An executor is a separate process that runs on a node and is responsible for executing tasks assigned to it by the driver. Therefore, a node can have multiple executors running on it. The number of executors on a node depends on the resources available on that node and the configuration settings of the Spark application. So, option C is the correct answer.

upvoted 7 times

Saurabh_221b Most Recent 2 months, 3 weeks ago

Selected Answer: C

Executors are the processing engines that run on nodes to execute tasks in a Spark application.

upvoted 1 times

SnData 7 months, 1 week ago

Executor is a process inside a node. Hence answer is C

upvoted 3 times

oda_rasheed_basha 9 months, 3 weeks ago

A) is C. nodes are the physical machines in the Spark cluster, while executors are the worker processes running on these nodes, responsible for executing tasks and processing data as part of a Spark application.

upvoted 3 times

Anweee 11 months ago

this is so wrong, it is option C

upvoted 1 times

Anweee 11 months ago

It is C, this solution is so wrong.

Executors are the processes running on the Nodes/Workers/Machines.

upvoted 1 times

amirshaz 11 months, 1 week ago

Selected Answer: C

C is correct

upvoted 2 times

mehroosali 1 year, 2 months ago

Selected Answer: C

C is correct.

upvoted 1 times

GeorgiT 1 year, 3 months ago

The correct answer is C. The other (currently selected as correct)answer D is also wrong the total number of nodes is always larger than the executor nodes by 1 sins we need one of the Cluster nodes to run the Driver.

upvoted 1 times

astone42 1 year, 4 months ago

Selected Answer: C

It's C

upvoted 1 times

Dgohel 1 year, 4 months ago

Which one to accept as an answer community answer or a Suggested answer from exam topics? Are the examtopics answers trustworthy?

upvoted 1 times

singh100 1 year, 5 months ago

C. Spark executors run on worker nodes and are responsible for executing tasks and storing intermediate data during data processing. The nodes represent the physical machines that provide computing resources to run the Spark application.

upvoted 1 times

Sandy544 1 year, 6 months ago

Selected Answer: C

An executor is a process that is launched for a Spark application on a worker node.

upvoted 1 times

TmData 1 year, 6 months ago

Selected Answer: C

The correct answer is C. An executor is a processing engine running on a node.

Explanation: In Apache Spark, a node refers to a physical or virtual machine in a cluster that is part of the Spark cluster. Each node can have one or more executors running on it. An executor is a Spark component responsible for executing tasks and storing data in memory or on disk. It is a worker process that runs on a node and performs the actual computation and data processing tasks assigned to it by the driver program. Executors are created and managed by the cluster manager, and they are responsible for executing the tasks and managing the data partitions assigned to them.

upvoted 1 times

evertonllins 1 year, 8 months ago

Selected Answer: C

The correct answer is C. The executor runs in a node.

upvoted 2 times

Question #3 *Topic 1*

Which of the following will occur if there are more slots than there are tasks?

A. The Spark job will likely not run as efficiently as possible.

B. The Spark application will fail – there must be at least as many tasks as there are slots.

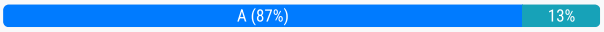
C. Some executors will shut down and allocate all slots on larger executors first.

D. More tasks will be automatically generated to ensure all slots are being used.

E. The Spark job will use just one single slot to perform all tasks.

Suggested Answer: *D*

Community vote distribution



TC007 Highly Voted 2 years, 2 months ago

Selected Answer: A

Slots are the basic unit of parallelism in Spark, and represent a unit of resource allocation on a single executor. If there are more slots than there are tasks, it means that some of the slots will be idle and not being used to execute any tasks, leading to inefficient resource utilization. In this scenario, the Spark job will likely not run as efficiently as possible. However, it is still possible for the Spark job to complete successfully. Therefore, option A is the correct answer.

upvoted 6 times

KM16494 Most Recent 2 months, 2 weeks ago

Selected Answer: A

A is correct

upvoted 1 times

zic00 10 months, 2 weeks ago

Selected Answer: A

If there are more slots (i.e., available cores) than tasks, some of the slots will remain idle, leading to underutilization of resources. This can result in less efficient execution because the available resources are not being fully utilized.

upvoted 1 times

Raheel_te 1 year ago

A is correct

upvoted 1 times

SnData 1 year, 1 month ago

Answer - A

upvoted 1 times

tzj_d 1 year, 3 months ago

Selected Answer: A

it is A

upvoted 1 times

zozoshanky 1 year, 6 months ago

C. Some executors will shut down and allocate all slots on larger executors first.

explanation :If there are more slots than there are tasks in Apache Spark, some executors may shut down, and the available slots will be allocated to larger executors first. This process is part of the dynamic resource allocation mechanism in Spark, where resources are adjusted based on the workload. It helps in efficient resource utilization by shutting down unnecessary executors and allocating resources to larger executors to perform tasks more efficiently.

upvoted 2 times

raghavendra516 11 months, 2 weeks ago

there is dynamic property : spark.dynamicAllocation.executorIdleTimeout which reallocate executors when they are idle

upvoted 1 times

knivesz 1 year, 6 months ago

Selected Answer: E

E , When there are more available slots than tasks, Spark will use a single slot to perform all tasks, which may result in inefficient use of resources.

upvoted 1 times

knivesz 1 year, 6 months ago

Selected Answer: E

E es Correct

upvoted 1 times

hua 1 year, 6 months ago

Selected Answer: A

A is correct

upvoted 1 times

astone42 1 year, 11 months ago

Selected Answer: A

A is correct

upvoted 1 times

singh100 1 year, 11 months ago

A. IF there are more slots than there are tasks, the extra slots will not be utilized, and they will remain idle, resulting in some resource waste. To maximize resource usage efficiency, it is essential to configure the cluster properly and adjust the number of tasks and slots based on the workload demands. Dynamic resource allocation features in cluster managers can also help improve resource utilization by adjusting the cluster size dynamically based on the task requirements.

upvoted 1 times

4be8126 2 years, 2 months ago

Selected Answer: A

A. The Spark job will likely not run as efficiently as possible.

In Spark, a slot represents a unit of processing capacity that an executor can offer to run a task. If there are more slots than there are tasks, some of the slots will remain unused, and the Spark job will likely not run as efficiently as possible. Spark automatically assigns tasks to slots, and if there are more slots than necessary, some of them may remain idle, resulting in wasted resources and slower job execution. However, the job will not fail as long as there are enough resources to execute the tasks, and Spark will not generate more tasks than needed. Also, executors will not shut down because there are unused slots. They will remain active until the end of the job or until explicitly terminated.

upvoted 4 times

Question #4 *Topic 1*

Which of the following is the most granular level of the Spark execution hierarchy?

A. Task

B. Executor

C. Node

D. Job

E. Slot

Suggested Answer: *A*

Community vote distribution

A (100%)

sekhargvvs 1 month, 3 weeks ago

Selected Answer: A

Task is the granular leve of spark hierarchy

upvoted 1 times

singh100 11 months ago

A. Task

upvoted 1 times

Mohitsain 1 year ago

Selected Answer: A

The correct answer is A. Task.

upvoted 1 times

TmData 1 year ago

Selected Answer: A

The correct answer is A. Task.

Explanation: In the Spark execution hierarchy, the most granular level is the task. A task represents a unit of work that is executed on a partitioned portion of the data in parallel. Tasks are created by the Spark driver program and assigned to individual executors for execution. Each task operates on a subset of the data and performs a specific operation defined by the Spark application, such as a transformation or an action.

upvoted 2 times

evertonllins 1 year, 2 months ago

Selected Answer: A

A is correct

upvoted 1 times

Question #5 *Topic 1*

Which of the following statements about Spark jobs is incorrect?

A. Jobs are broken down into stages.

B. There are multiple tasks within a single job when a DataFrame has more than one partition.

C. Jobs are collections of tasks that are divided up based on when an action is called.

D. There is no way to monitor the progress of a job.

E. Jobs are collections of tasks that are divided based on when language variables are defined.

Suggested Answer: *D*

Community vote distribution

D (100%)

4be8126 Highly Voted 1 year, 2 months ago

There are two incorrect answers in the original question.

Option D, "There is no way to monitor the progress of a job," is incorrect. As I mentioned earlier, Spark provides various tools and interfaces for monitoring the progress of a job, including the Spark UI, which provides real-time information about the job's stages, tasks, and resource utilization. Other tools, such as the Spark History Server, can be used to view completed job information.

Option E, "Jobs are collections of tasks that are divided based on when language variables are defined," is also incorrect. The division of tasks in a Spark job is not based on when language variables are defined, but rather based on when actions are called.

upvoted 7 times

sekhargvvs Most Recent 1 month, 3 weeks ago

Selected Answer: D

Spark UI we can monitor the job

upvoted 1 times

TmData 1 year ago

Selected Answer: D

The incorrect statement is:

D. There is no way to monitor the progress of a job.

Explanation: Spark provides several ways to monitor the progress of a job. The Spark UI (Web UI) provides a graphical interface to monitor the progress of Spark jobs, stages, tasks, and other relevant metrics. It displays information such as job status, task completion, execution time, and resource usage. Additionally, Spark provides programmatic APIs, such as the JobProgressListener interface, which allows developers to implement custom job progress monitoring logic within their Spark applications.

upvoted 3 times

Question #6 *Topic 1*

Which of the following operations is most likely to result in a shuffle?

A. DataFrame.join()

B. DataFrame.filter()

C. DataFrame.union()

D. DataFrame.where()

E. DataFrame.drop()

Suggested Answer: *A*

Community vote distribution

A (100%)

sekhargvvs 1 month, 3 weeks ago

Selected Answer: A

Join is a wide transformation results in a shuffle

upvoted 1 times

TmData 1 year ago

Selected Answer: A

The most likely operation to result in a shuffle is:

A. DataFrame.join()

Explanation: A shuffle operation in Spark involves redistributing and reorganizing data across partitions. It typically occurs when data needs to be rearranged or merged based on a specific key or condition. DataFrame joins involve combining two DataFrames based on a common key column, and this operation often requires data to be shuffled to ensure that matching records are located on the same executor or partition. The shuffle process involves exchanging data between nodes or executors in the cluster, which can incur significant data movement and network communication overhead.

upvoted 2 times

4be8126 1 year, 2 months ago

Selected Answer: A

The operation that is most likely to result in a shuffle is DataFrame.join().

Join operation requires data to be combined from two different sources based on a common key, and this typically involves a reorganization of the data such that the data with the same keys are co-located in the same executor. This process is known as a shuffle operation, which can be a performance-intensive operation, especially for large datasets.

The other DataFrame operations such as filter(), union(), where() or drop() do not require data to be shuffled across the nodes.

upvoted 2 times

Question #7 *Topic 1*

The default value of spark.sql.shuffle.partitions is 200. Which of the following describes what that means?

A. By default, all DataFrames in Spark will be spit to perfectly fill the memory of 200 executors.

B. By default, new DataFrames created by Spark will be split to perfectly fill the memory of 200 executors.

C. By default, Spark will only read the first 200 partitions of DataFrames to improve speed.

D. By default, all DataFrames in Spark, including existing DataFrames, will be split into 200 unique segments for parallelization.

E. By default, DataFrames will be split into 200 unique partitions when data is being shuffled.

Suggested Answer: *E*

Community vote distribution

E (100%)

singh100 11 months ago

E is correct.

upvoted 1 times

TmData 1 year ago

Selected Answer: E

The correct answer is E. By default, DataFrames will be split into 200 unique partitions when data is being shuffled.

Explanation: The spark.sql.shuffle.partitions configuration parameter in Spark determines the number of partitions to use when shuffling data. When a shuffle operation occurs, such as during DataFrame joins or aggregations, data needs to be redistributed across partitions based on a specific key. The spark.sql.shuffle.partitions value defines the default number of partitions to be used during such shuffling operations.

upvoted 3 times

sumand 1 year ago

E. By default, DataFrames will be split into 200 unique partitions when data is being shuffled.

The spark.sql.shuffle.partitions configuration parameter determines the number of partitions that are used when shuffling data for joins or aggregations. The default value is 200, which means that by default, when a shuffle operation occurs, the data will be divided into 200 partitions. This allows the tasks to be distributed across the cluster and processed in parallel, improving performance.

However, the optimal number of shuffle partitions depends on the specific details of your cluster and data. If the number is too small, then each partition will be large, and the tasks may take a long time to run. If the number is too large, then there will be many small tasks, and the overhead of scheduling and processing all these tasks can degrade performance. Therefore, tuning this parameter to match your specific use case can help optimize the performance of your Spark applications.

upvoted 2 times

Question #8 *Topic 1*

Which of the following is the most complete description of lazy evaluation?

A. None of these options describe lazy evaluation

B. A process is lazily evaluated if its execution does not start until it is put into action by some type of trigger

C. A process is lazily evaluated if its execution does not start until it is forced to display a result to the user

D. A process is lazily evaluated if its execution does not start until it reaches a specified date and time

E. A process is lazily evaluated if its execution does not start until it is finished compiling

Suggested Answer: *B*

Community vote distribution

B (100%)

TmData Highly Voted 1 year ago

Selected Answer: B

The most complete description of lazy evaluation is:

B. A process is lazily evaluated if its execution does not start until it is put into action by some type of trigger.

Explanation: Lazy evaluation is a programming language feature that delays the evaluation of expressions or computations until the result is actually needed or requested. In a lazily evaluated system, expressions are not immediately executed or evaluated when they are defined or assigned. Instead, the evaluation is deferred until the value is needed by another part of the program or when an action is triggered.

upvoted 5 times

4be8126 Most Recent 1 year, 2 months ago

B. A process is lazily evaluated if its execution does not start until it is put into action by some type of trigger.

Lazy evaluation is a programming paradigm that defers the evaluation of an expression until its value is needed. In other words, lazy evaluation delays the computation of a value until the value is actually required by the program. This is in contrast to eager evaluation, where expressions are evaluated as soon as they are bound to a variable.

In Spark, lazy evaluation is used extensively to optimize the execution of complex data processing pipelines. When a user creates a series of operations to transform a dataset, Spark does not immediately execute those operations. Instead, it builds a logical plan that represents the operations as a set of transformations on the original dataset. The actual execution of the plan is deferred until the user requests the result by triggering an action, such as writing the result to disk or displaying it on the screen. By using lazy evaluation, Spark can optimize the execution plan and avoid unnecessary computations, resulting in faster processing times and more efficient use of cluster resources.

upvoted 2 times

Question #9 *Topic 1*

Which of the following DataFrame operations is classified as an action?

A. DataFrame.drop()

B. DataFrame.coalesce()

C. DataFrame.take()

D. DataFrame.join()

E. DataFrame.filter()

Suggested Answer: *C*

Community vote distribution

C (100%)

TmData Highly Voted 2 years ago

Selected Answer: C

The DataFrame operation classified as an action is:

C. DataFrame.take()

Explanation: In Spark, actions are operations that trigger the execution of transformations on a DataFrame and return results or side effects. Actions are evaluated eagerly, meaning they initiate the execution of the computation plan built by transformations. Among the options provided, DataFrame.take() is an action because it returns an array with the first n elements from the DataFrame as an array. It triggers the execution of any pending transformations and collects the resulting data.

upvoted 5 times

Becida Most Recent 9 months ago

Selected Answer: C

It returns first n rows of an array

upvoted 1 times

SonicBoom10C9 2 years, 1 month ago

Selected Answer: C

DataFrame.take(num: int) → List[pyspark.sql.types.Row]

All the other functions return a dataframe again, which is defined as a transformation. An action returns a result of a computation, which take() does.

upvoted 3 times

Question #10 *Topic 1*

Which of the following DataFrame operations is classified as a wide transformation?

A. DataFrame.filter()

B. DataFrame.join()

C. DataFrame.select()

D. DataFrame.drop()

E. DataFrame.union()

Suggested Answer: *B*

Community vote distribution

B (100%)

4be8126 Highly Voted 2 years, 1 month ago

Selected Answer: B

B. DataFrame.join() is classified as a wide transformation, as it shuffles the data across the network to perform the join operation.

upvoted 5 times

Becida Most Recent 9 months ago

Selected Answer: B

Join takes different partitions and combines into single dataframe.

upvoted 1 times

TmData 2 years ago

Selected Answer: B

The DataFrame operation classified as a wide transformation is:

B. DataFrame.join()

Explanation: In Spark, transformations are operations on DataFrames that create a new DataFrame from an existing one. Wide transformations involve shuffling or redistributing data across partitions and typically require data movement across the network. Among the options provided, DataFrame.join() is a wide transformation because it involves combining two DataFrames based on a common key column, which often requires shuffling and redistributing the data across partitions.

upvoted 3 times

SonicBoom10C9 2 years, 1 month ago

Selected Answer: B

None of the other options are wide transformations, they are narrow (logically, they modify the length of a dataframe). Only a join can force shuffling of data between horizontally scaled partitions.

upvoted 3 times

Question #11 *Topic 1*

Which of the following describes the difference between cluster and client execution modes?

A. The cluster execution mode runs the driver on a worker node within a cluster, while the client execution mode runs the driver on the client machine (also known as a gateway machine or edge node).

B. The cluster execution mode is run on a local cluster, while the client execution mode is run in the cloud.

C. The cluster execution mode distributes executors across worker nodes in a cluster, while the client execution mode runs a Spark job entirely on one client machine.

D. The cluster execution mode runs the driver on the cluster machine (also known as a gateway machine or edge node), while the client execution mode runs the driver on a worker node within a cluster.

E. The cluster execution mode distributes executors across worker nodes in a cluster, while the client execution mode submits a Spark job from a remote machine to be run on a remote, unconfigurable cluster.

Suggested Answer: *A*

zozoshanky Highly Voted 11 months ago

Explanation:
In cluster mode, the driver runs on the master node, while in client mode, the driver runs on a virtual machine in the cloud.

This is wrong, since execution modes do not specify whether workloads are run in the cloud or on-premise.

In cluster mode, each node will launch its own executor, while in client mode, executors will exclusively run on the client machine.

Wrong, since in both cases executors run on worker nodes.

In cluster mode, the driver runs on the edge node, while the client mode runs the driver in a worker node.

Wrong C in cluster mode, the driver runs on a worker node. In client mode, the driver runs on the client machine.

In client mode, the cluster manager runs on the same host as the driver, while in cluster mode, the cluster manager runs on a separate node.

No. In both modes, the cluster manager is typically on a separate node C not on the same host as the driver. It only runs on the same host as the driver in local execution mode. More info: Learning Spark, 2nd Edition, Chapter 1, and Spark: The Definitive Guide, Chapter 15. ()
upvoted 5 times

Tedet Most Recent 2 weeks, 4 days ago

Selected Answer: A
In Apache Spark, the difference between cluster and client execution modes mainly relates to where the driver program runs:

Client mode: The driver runs on the machine that submitted the application (often called the client, edge node, or gateway node). The executors run on the cluster's worker nodes.

Cluster mode: The driver runs inside the cluster, typically on one of the worker nodes. This is useful for production deployments because the driver is decoupled from the submitting client.
upvoted 1 times

Question #12 *Topic 1*

Which of the following statements about Spark's stability is incorrect?

A. Spark is designed to support the loss of any set of worker nodes.

B. Spark will rerun any failed tasks due to failed worker nodes.

C. Spark will recompute data cached on failed worker nodes.

D. Spark will spill data to disk if it does not fit in memory.

E. Spark will reassign the driver to a worker node if the driver's node fails.

Suggested Answer: *C*

Community vote distribution

E (100%)

TmData Highly Voted 2 years ago

Selected Answer: E

Option E is incorrect because the driver program in Spark is not reassigned to another worker node if the driver's node fails. The driver program is responsible for the coordination and control of the Spark application and runs on a separate machine, typically the client machine or cluster manager. If the driver's node fails, the Spark application as a whole may fail or need to be restarted, but the driver is not automatically reassigned to another worker node.

upvoted 6 times

Pazzobg Most Recent 1 month, 2 weeks ago

Selected Answer: E

Reassigning the work to another node is valid for the worker nodes, but in case the original Driver node fails, Spark does not reassign its work to another node making it a new Driver.

upvoted 1 times

zic00 10 months, 1 week ago

Selected Answer: E

In Spark, the driver node is crucial for orchestrating the execution of the Spark application. If the driver node fails, the Spark application fails. Spark does not automatically reassign the driver to another node if the driver fails. This would require the application to be restarted manually or through external high-availability mechanisms.

upvoted 2 times

YoSpark 11 months, 1 week ago

Considering the word "any set" looks like A is not correct either. What if all the worker nodes fail.

A- "Spark is designed to support the loss of any set of worker nodes."

upvoted 1 times

Sonu124 1 year, 9 months ago

Option E beacuse spark doesn't assiggned driver if faild

upvoted 2 times

TmData 2 years ago

Selected Answer: E

The incorrect statement about Spark's stability is:

E. Spark will reassign the driver to a worker node if the driver's node fails.

Explanation:

Option A is correct because Spark is designed to handle the failure of worker nodes. When a worker node fails, Spark redistributes the lost tasks to other available worker nodes to ensure fault tolerance.

Option C is correct because Spark is able to recompute data that was cached on failed worker nodes. Spark maintains lineage information about RDDs (Resilient Distributed Datasets), allowing it to reconstruct lost data partitions in case of failures.

upvoted 2 times

SonicBoom10C9 2 years, 1 month ago

Selected Answer: E

The driver is responsible for maintaining spark context. If it fails, there is no recourse. The driver can mitigate the failure of worker nodes through limited fault tolerance mechanisms.

upvoted 2 times

4be8126 2 years, 1 month ago

Selected Answer: E

All of the following statements about Spark's stability are correct except for:

E. Spark will reassign the driver to a worker node if the driver's node fails.

The driver is a special process in Spark that is responsible for coordinating tasks and executing the main program. If the driver fails, the entire Spark application fails and cannot be restarted. Therefore, Spark does not reassign the driver to a worker node if the driver's node fails.

upvoted 2 times

Indiee 2 years, 2 months ago

The E is only valid when spark-submit is in cluster modes

upvoted 2 times

raghavendra516 11 months, 2 weeks ago

And it also depened on Resource Manger of cluser on which spark is running.

upvoted 1 times

GuidoDC 2 years, 3 months ago

Selected Answer: E

If the driver node fails your cluster will fail. If the worker node fails, Databricks will spawn a new worker node to replace the failed node and resumes the workload.

upvoted 3 times

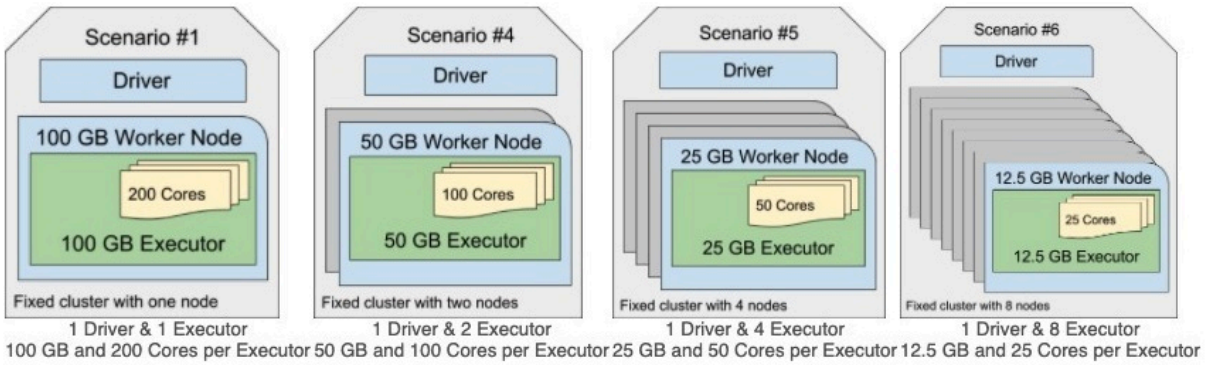
TC007 2 years, 2 months ago

If the node running the driver program fails, Spark's built-in fault-tolerance mechanism can reassign the driver program to run on another node.

upvoted 1 times

Question #13 *Topic 1*

Which of the following cluster configurations is most likely to experience an out-of-memory error in response to data skew in a single partition?



Note: each configuration has roughly the same compute power using 100 GB of RAM and 200 cores.

A. Scenario #4

B. Scenario #5

C. Scenario #6

D. More information is needed to determine an answer.

E. Scenario #1

Suggested Answer: *C*

Community vote distribution

C (100%)

TmData Highly Voted 1 year, 6 months ago

Selected Answer: C

The most likely scenario to experience an out-of-memory error in response to data skew in a single partition is:

C. Scenario #6: 12.5 GB Worker Node, 12.5 GB Executor. 1 Driver & 8 Executors.

Explanation:

Data skew refers to an uneven distribution of data across partitions. When there is significant skew in a single partition, it can lead to increased memory usage for that specific partition, potentially causing out-of-memory errors. The smaller the available memory per executor, the higher the likelihood of encountering such issues.

In this case, Scenario #6 has the smallest worker node and executor configuration, with only 12.5 GB of RAM available for each executor. With 8 executors, the total available memory is still 100 GB (similar to other scenarios), but the reduced memory per executor increases the risk of encountering out-of-memory errors when handling skewed data in a single partition.

upvoted 12 times

velle93 Most Recent 1 month, 3 weeks ago

Selected Answer: E

I think E scenario #1 as you have a single executor that is most likely to crash if you get a data skew in a partition this could lead to significant overhead going over the 100GB of RAM.

upvoted 2 times

azurearch 9 months, 3 weeks ago

D is correct. even though you have less executor memory in scenario 6, spark will still complete the process , it might take more time to do the shuffle neverthless.

upvoted 2 times

Mohitsain 1 year, 6 months ago

Selected Answer: C

This is the right answer.

upvoted 1 times

TmData 1 year, 6 months ago

Selected Answer: C

Option A, Scenario #4, has larger worker nodes and executors compared to Scenario #6, reducing the likelihood of encountering out-of-memory errors due to data skew.

Option B, Scenario #5, also has larger worker nodes and executors compared to Scenario #6, providing more memory per executor and reducing the risk of out-of-memory errors.

Option D states that more information is needed to determine an answer, but based on the available information, Scenario #6 is the most likely to experience out-of-memory errors due to data skew in a single partition.

Option E, Scenario #1, has larger worker nodes and executors compared to Scenario #6, reducing the likelihood of out-of-memory errors due to data skew.

upvoted 3 times

Indiee 1 year, 8 months ago

Data skew is when you have a few partitions oversized. But due to initial partitioning this large datasets needed to be processed by single threads so can cause OOM

upvoted 3 times

Dhruv_Ajmeri 1 year, 9 months ago

Please explain the answer!!

upvoted 1 times

Question #14 *Topic 1*

Of the following situations, in which will it be most advantageous to store DataFrame df at the MEMORY_AND_DISK storage level rather than the MEMORY_ONLY storage level?

A. When all of the computed data in DataFrame df can fit into memory.

B. When the memory is full and it's faster to recompute all the data in DataFrame df rather than read it from disk.

C. When it's faster to recompute all the data in DataFrame df that cannot fit into memory based on its logical plan rather than read it from disk.

D. When it's faster to read all the computed data in DataFrame df that cannot fit into memory from disk rather than recompute it based on its logical plan.

E. The storage level MENORY_ONLY will always be more advantageous because it's faster to read data from memory than it is to read data from disk.

Suggested Answer: *B*

Community vote distribution

D (100%)

sousouka Highly Voted 1 year, 3 months ago

D. When it's faster to read all the computed data in DataFrame df that cannot fit into memory from disk rather than recompute it based on its logical plan.

upvoted 9 times

ZSun Highly Voted 1 year ago

All other explanation is either wrong or misleading. To understand the question, you need to understand the difference between Memory_only and Memory_and_Disk

1. Memory_and_Disk, which is the default mode for cache ro persist. That means, if the data size is larger than the memory, it will store the extra data in disk. next time when we n eed to read data, we will read data firstly from memory, and then read from disk.

2. Memory_Only means, if the data size is larger than memory, it will not store the extra data. next time we read data, we will read from memory first and then recompute the extra data which cannot store in memory.

PS. Mr. 4be8126 is wrong about raising error when out of memory.

Therefore, the difference/balance between Memory_only and memory_and_disk lay in how they handle the extra data out of memory. which is option D, if it is faster to read data from disk is faster than recompute it, then memory_and_disk.

upvoted 7 times

newusername Most Recent 7 months, 3 weeks ago

Selected Answer: D

D is correct

upvoted 1 times

astone42 10 months, 3 weeks ago

Selected Answer: D

D is correct

upvoted 1 times

singh100 11 months ago

D. It is faster to read the computed data from disk instead of recomputing it based on its logical plan when the recomputation is costly and time-consuming.

upvoted 1 times

SonicBoom10C9 1 year, 1 month ago

Selected Answer: D

If it's faster to read from memory and can fit in, then there is no reason to use Memory_and_disk, Memory_only is sufficient. Also, if it's faster to compute than read from disk, that's what you would do. The only options is when it's too big to fit in memory and too expensive to recompute, so reading from disk (or rather caching from disk into memory on the fly) is faster.

upvoted 1 times

4be8126 1 year, 1 month ago

Selected Answer: D

The most advantageous situation to store a DataFrame at the MEMORY_AND_DISK storage level instead of the MEMORY_ONLY storage level is option D - when it's faster to read all the computed data in DataFrame df that cannot fit into memory from disk rather than recompute it based on its logical plan.

This is because the MEMORY_ONLY storage level only stores data in memory, which can result in an out-of-memory error if the data exceeds the available memory. On the other hand, the MEMORY_AND_DISK storage level will spill data to disk if there is not enough memory available, allowing more data to be processed without errors.

In situations where the computed data can fit entirely into memory, it is best to use the MEMORY_ONLY storage level as it will be faster than reading from disk. However, when there is not enough memory to store all the computed data, it may be necessary to use the MEMORY_AND_DISK storage level.

upvoted 1 times

sly75 1 year, 1 month ago

Yes but what about the link with the question ? I would say B too :)

upvoted 1 times

Indiee 1 year, 2 months ago

Answer is D. This is the whole idea behind caching

upvoted 2 times

Question #15 *Topic 1*

A Spark application has a 128 GB DataFrame A and a 1 GB DataFrame B. If a broadcast join were to be performed on these two DataFrames, which of the following describes which DataFrame should be broadcasted and why?

A. Either DataFrame can be broadcasted. Their results will be identical in result and efficiency.

B. DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of itself.

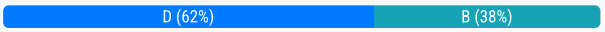
C. DataFrame A should be broadcasted because it is larger and will eliminate the need for the shuffling of DataFrame B.

D. DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of DataFrame A.

E. DataFrame A should be broadcasted because it is smaller and will eliminate the need for the shuffling of itself.

Suggested Answer: *D*

Community vote distribution



te1 4 months, 2 weeks ago

Selected Answer: B

Dataset B is smaller Dataset. It will be brodcasted to all worker nodes. So No shuffle for Dataset B.

upvoted 2 times

monibun 8 months, 3 weeks ago

Should be B: During the join, the intention of the shuffle would be to bring the same keys from both dataframes in same partition. Now, this would ideally require both of them to be shuffled. however, if smaller one is broadcasted, that would mean we have sent the entire smaller dataframe in each partition whereas the bigger one would still undergo a shuffle to get its similar keys in each partition. hence, the re-shuffle of just the smaller one is avoided.

upvoted 2 times

65bd33e 10 months, 3 weeks ago

Selected Answer: D

The correct answer is:

D. DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of DataFrame A.

Explanation:

In a broadcast join, the smaller DataFrame (in this case, DataFrame B, which is 1 GB) is broadcasted to all worker nodes. This allows the larger DataFrame (DataFrame A, which is 128 GB) to be joined without shuffling its data across the cluster, which would be computationally expensive. Broadcasting the smaller DataFrame reduces the amount of data that needs to be shuffled, improving the efficiency of the join operation.

upvoted 1 times

atulrao 11 months, 3 weeks ago

The correct answer is:
B. DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of itself.

Explanation:

In Spark, a broadcast join is a specific type of join where one DataFrame is sent to every node in the cluster to avoid the costly network shuffle that can occur with large datasets in regular joins.
Generally, the smaller DataFrame should be broadcasted to optimize performance. This is because broadcasting a smaller DataFrame requires less network bandwidth and memory usage across the cluster.
Broadcasting DataFrame B (the smaller DataFrame at 1 GB) means that each node will have a local copy of DataFrame B, allowing them to perform the join operation locally with their respective partitions of DataFrame A without needing to shuffle DataFrame B across the network.
This approach significantly reduces the amount of data that needs to be shuffled (since only DataFrame A is partitioned across the nodes), thereby improving the performance of the join operation.

upvoted 3 times

azurearch 1 year, 3 months ago

Correct answer is B. D is wrong. Being the larger dataset Dataframe A (128 GB) will get shuffled being the larger dataset. Dataframe A (1 GB) (if hint is specified in join), will be broadcasted hence it would not get shuffled.
upvoted 1 times

Ahlo 1 year, 4 months ago
answer D - With broadcast join, Spark broadcast the smaller DataFrame to all executors and the executor keeps this DataFrame in memory and the larger DataFrame is split and distributed across all executors so that Spark can perform a join without shuffling any data from the larger DataFrame as the data required for join colocated on every executor.
upvoted 1 times

mehroosali 1 year, 7 months ago
Selected Answer: B
It should really be B.
upvoted 3 times

thanab 1 year, 9 months ago
The correct answer is B. DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of itself. A broadcast join is an optimization technique in the Spark SQL engine that is used to join two DataFrames. This technique is ideal for joining a large DataFrame with a smaller one. With broadcast join, Spark broadcasts the smaller DataFrame to all executors and the executor keeps this DataFrame in memory. The larger DataFrame is split and distributed across all executors so that Spark can perform a join without shuffling any data from the larger DataFrame as the data required for join colocated on every executor.
upvoted 1 times

thanab 1 year, 9 months ago
Option D is incorrect because it states that DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of DataFrame A. However, broadcasting DataFrame B will not eliminate the need for shuffling DataFrame A. Instead, broadcasting DataFrame B will eliminate the need for shuffling itself. In a broadcast join, the smaller DataFrame is broadcasted to all executors and kept in memory. The larger DataFrame is split and distributed across all executors so that Spark can perform a join without shuffling any data from the larger DataFrame as the data required for join colocated on every executor.
upvoted 2 times

eendee 1 year, 10 months ago
Selected Answer: D
https://sparkbyexamples.com/spark/broadcast-join-in-spark/

Spark Broadcast Join is an important part of the Spark SQL execution engine, With broadcast join, Spark broadcast the smaller DataFrame to all executors and the executor keeps this DataFrame in memory and the larger DataFrame is split and distributed across all executors so that Spark can perform a join without shuffling any data from the larger DataFrame as the data required for join colocated on every executor.
upvoted 3 times

Diws 1 year, 11 months ago
D should be correct. Broadcast join happens on smaller DataFrame to prevent the shuffling of larger DataFrame.
upvoted 2 times

TmData 2 years ago
Selected Answer: D
Option A is incorrect because not both DataFrames can be broadcasted. Only one of the DataFrames should be broadcasted to minimize shuffling.

Option B is correct because DataFrame B is smaller and broadcasting it will eliminate the shuffling of DataFrame B, improving the join operation's efficiency.

Option C is incorrect because DataFrame A is larger and shuffling DataFrame B is not a concern in this scenario.

Option E is incorrect because DataFrame A is larger, and broadcasting it would not eliminate the shuffling of itself. The larger DataFrame typically undergoes shuffling in a broadcast join.

Therefore, the correct option is D.
upvoted 3 times

ZSun 2 years ago
Selected Answer: B
B is correct.
upvoted 2 times

4be8126 2 years, 1 month ago

Selected Answer: D

The correct answer is D. DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of DataFrame A.

A broadcast join is a technique where the smaller DataFrame is broadcast to all the worker nodes in the cluster, so that it can be joined with the larger DataFrame without requiring any shuffling of the larger DataFrame. This is generally more efficient than a shuffle join, which requires data to be shuffled across the network.

In this scenario, DataFrame B is much smaller than DataFrame A, so it is more efficient to broadcast DataFrame B to all worker nodes in the cluster. This will eliminate the need for shuffling of DataFrame A, making the join more efficient.

upvoted 1 times

Indiee 2 years, 2 months ago

All the ANS are incorrect. The DAG will perform a sort merge join instead of BCJ. The size of a DF needed to be 10MB max for broadcast else it will cause a network overload.

upvoted 2 times

Question #16 *Topic 1*

Which of the following operations can be used to create a new DataFrame that has 12 partitions from an original DataFrame df that has 8 partitions?

A. df.repartition(12)

B. df.cache()

C. df.partitionBy(1.5)

D. df.coalesce(12)

E. df.partitionBy(12)

Suggested Answer: *A*

Community vote distribution

A (100%)

4be8126 Highly Voted 1 year, 1 month ago

Selected Answer: A

The answer is A. The repartition operation can be used to increase or decrease the number of partitions in a DataFrame. In this case, the number of partitions is being increased from 8 to 12, so we can use the repartition operation with a partition count of 12: df.repartition(12).

Option B, df.cache(), is used to cache a DataFrame in memory for faster access, but it does not change the number of partitions.

Option C, df.partitionBy(1.5), is not a valid operation for partitioning a DataFrame.

Option D, df.coalesce(12), can be used to reduce the number of partitions in a DataFrame, but it cannot be used to increase the number of partitions beyond the current number.

Option E, df.partitionBy(12), is used to partition a DataFrame by a specific column or set of columns, but it does not change the number of partitions.

upvoted 6 times

NuclearGandhi Most Recent 7 months, 3 weeks ago

Selected Answer: A

nice explanation @4be8126

upvoted 1 times

TmData 1 year ago

Selected Answer: A

The operation that can be used to create a new DataFrame with 12 partitions from an original DataFrame df that has 8 partitions is:

D. df.coalesce(12)

Explanation:

The coalesce() operation in Spark is used to decrease the number of partitions in a DataFrame, and it can be used to create a new DataFrame with a specific number of partitions. In this case, calling df.coalesce(12) on the original DataFrame df with 8 partitions will create a new DataFrame with 12 partitions.

upvoted 1 times

SonicBoom10C9 1 year, 1 month ago

Selected Answer: A

Comprehensive explanation by 4be8126, only using this comment to vote A.

upvoted 1 times

Question #17 *Topic 1*

Which of the following object types cannot be contained within a column of a Spark DataFrame?

A. DataFrame

B. String

C. Array

D. null

E. Vector

Suggested Answer: *A*

Community vote distribution

A (100%)

singh100 11 months ago

A. Spark DataFrames do not directly support containing other DataFrames as columns. A DataFrame column can only have one of the supported data types, such as primitive types (e.g., IntegerType, StringType, DoubleType, etc.) or complex types (e.g., ArrayType, MapType, StructType, etc.), but it cannot contain an entire DataFrame as a column.

upvoted 3 times

TmData 1 year ago

Selected Answer: A

Spark DataFrames are designed to store structured data, where each column has a specific data type. While DataFrames can contain various data types such as strings (option B), arrays (option C), null values (option D), and vectors (option E), they cannot directly contain other DataFrames (option A) as a column.

upvoted 2 times

Question #18 *Topic 1*

Which of the following operations can be used to create a DataFrame with a subset of columns from DataFrame storesDF that are specified by name?

A. storesDF.subset()

B. storesDF.select()

C. storesDF.selectColumn()

D. storesDF.filter()

E. storesDF.drop()

Suggested Answer: *B*

Community vote distribution

B (100%)

4be8126 Highly Voted 2 years, 2 months ago

Selected Answer: B

The operation that can be used to create a DataFrame with a subset of columns from DataFrame storesDF that are specified by name is storesDF.select().

The select() operation allows you to specify the columns you want to keep in the resulting DataFrame by passing in the column names as arguments. For example, to create a new DataFrame that contains only the columns store_id and store_name from the storesDF

DataFrame, you can use the following code:

newDF = storesDF.select("store_id", "store_name")

upvoted 6 times

YoSpark Most Recent 11 months, 1 week ago

E.storesDF.drop() is also correct. It is just opposite of select. If you have a large number of columns you need to select but a few to drop to meet your requirements, then drop is easier than select.

upvoted 1 times

TmData 2 years ago

Selected Answer: B

The select() operation in Spark DataFrame allows you to specify the columns you want to include in the resulting DataFrame. You can provide column names as arguments to the select() operation to create a new DataFrame with only the specified columns.

upvoted 2 times

Question #19 *Topic 1*

The code block shown below contains an error. The code block is intended to return a DataFrame containing all columns from DataFrame storesDF except for column sqft and column customerSatisfaction. Identify the error.
Code block:
storesDF.drop(sqft, customerSatisfaction)

A. The drop() operation only works if one column name is called at a time – there should be two calls in succession like storesDF.drop("sqft").drop("customerSatisfaction").

B. The drop() operation only works if column names are wrapped inside the col() function like storesDF.drop(col(sqft), col(customerSatisfaction)).

C. There is no drop() operation for storesDF.

D. The sqft and customerSatisfaction column names should be quoted like "sqft" and "customerSatisfaction".

E. The sqft and customerSatisfaction column names should be subset from the DataFrame storesDF like storesDF."sqft" and storesDF."customerSatisfaction".

Suggested Answer: *D*

Community vote distribution

D (100%)

4be8126 Highly Voted 1 year, 8 months ago

Selected Answer: D

The error in the code block is that the column names sqft and customerSatisfaction should be quoted, like "sqft" and "customerSatisfaction", since they are strings. The correct code block should be:

storesDF.drop("sqft", "customerSatisfaction")

Option D correctly identifies this error.
upvoted 5 times

ZSun 1 year, 6 months ago

The correct one is B:
storesDF.drop("sqft").drop("customerSatisfaction")
For D, it should be list of column name: storesDF.drop(["sqft", "customerSatisfaction"])
upvoted 1 times

ZSun 1 year, 6 months ago

The correct one is D, but my explanation is correct
upvoted 1 times

azurearch Most Recent 9 months, 3 weeks ago

sorry, Option D is correct
upvoted 1 times

azurearch 9 months, 3 weeks ago

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.drop.html option A is correct, drop expects only one argument, if its more than one, you would have to use as listofcols=['col1','col2'] and drop(*listofcols)
upvoted 1 times

zozoshanky 1 year, 5 months ago

D is correct,
df.drop('id','firstname').show() tested code
upvoted 1 times

TmData 1 year, 6 months ago

Selected Answer: D

When using the drop() operation in Spark DataFrame, the column names should be specified as strings and enclosed in quotes. In the given code block, the column names "sqft" and "customerSatisfaction" are not quoted, which results in a syntax error.

upvoted 1 times

Question #20 *Topic 1*

Which of the following code blocks returns a DataFrame containing only the rows from DataFrame storesDF where the value in column sqft is less than or equal to 25,000?

A. storesDF.filter("sqft" <= 25000)

B. storesDF.filter(sqft > 25000)

C. storesDF.where(storesDF[sqft] > 25000)

D. storesDF.where(sqft > 25000)

E. storesDF.filter(col("sqft") <= 25000)

Suggested Answer: *E*

Community vote distribution

E (100%)

4be8126 Highly Voted 2 years, 2 months ago

Selected Answer: E

The correct code block to return a DataFrame containing only the rows from DataFrame storesDF where the value in column sqft is less than or equal to 25,000 is:

storesDF.filter("sqft <= 25000")
Option A incorrectly uses the wrong syntax for the filter condition, it should be "sqft <= 25000" instead of "sqft" <= 25000.

Option B uses the wrong operator (greater than instead of less than or equal to) and also needs to quote the column name as a string.

Option C uses square brackets instead of quotes to reference the column name, and also uses the wrong operator.

Option D uses the correct operator but needs to quote the column name as a string.

Option E uses the correct syntax, but needs to pass the column name as a string instead of using col("sqft").

Therefore, the correct answer is E.

storesDF.filter(col("sqft") <= 25000)
upvoted 5 times

TmData Most Recent 2 years ago

Selected Answer: E

Option E, storesDF.filter(col("sqft") <= 25000), is the correct option. It uses the filter() operation with the condition col("sqft") <= 25000 to filter the rows where the value in the column sqft is less than or equal to 25,000.
upvoted 1 times

Question #21 *Topic 1*

Which of the following code blocks returns a DataFrame containing only the rows from DataFrame storesDF where the value in column sqft is less than or equal to 25,000 OR the value in column customerSatisfaction is greater than or equal to 30?

A. storesDF.filter(col("sqft") <= 25000 | col("customerSatisfaction") >= 30)

B. storesDF.filter(col("sqft") <= 25000 or col("customerSatisfaction") >= 30)

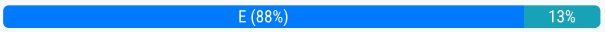
C. storesDF.filter(sqft <= 25000 or customerSatisfaction >= 30)

D. storesDF.filter(col(sqft) <= 25000 | col(customerSatisfaction) >= 30)

E. storesDF.filter((col("sqft") <= 25000) | (col("customerSatisfaction") >= 30))

Suggested Answer: *E*

Community vote distribution



jds0 11 months, 1 week ago

Selected Answer: E

Answer: E
I tried it with the code below, all other options raised an error:

```
# register UDF with udf function
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf
from pyspark.sql.types import IntegerType

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 3, 20000, "A"),
(1, 1, 50000, "A"),
(2, 2, 70000, "A"),
(3, 5, 10000, "B"),
(4, 4, 100000, "B"),
]
storesDF = spark.createDataFrame(data, ["storeID", "customerSatisfaction", "sqft", "division"])

try:
storesDF.filter(col("sqft") <= 25000 | col("customerSatisfaction") >= 30).show()
except Exception as e:
print(e)

try:
storesDF.filter((col("sqft") <= 25000) | (col("customerSatisfaction") >= 30)).show()
except Exception as e:
print(e)
```

upvoted 2 times

TmData 2 years ago

Selected Answer: E

Option E, storesDF.filter((col("sqft") <= 25000) | (col("customerSatisfaction") >= 30)), is the correct option. It uses the filter() operation with the conditions (col("sqft") <= 25000) | (col("customerSatisfaction") >= 30) to filter the rows where the value in column sqft is less than or equal to 25,000 OR the value in column customerSatisfaction is greater than or equal to 30.

upvoted 4 times

SonicBoom10C9 2 years, 1 month ago

Selected Answer: E

E has the right syntax, logic, operator and correct number of parentheses. All of the others falter in one of these respects.
upvoted 2 times

pierre_grns 2 years, 2 months ago

Selected Answer: A

Should be A. Tested it in communitity edition with 2 filters.
upvoted 1 times

pierre_grns 2 years, 2 months ago

sorry, we need 2 paranthesis indeed. So E !
upvoted 6 times

evertonllins 1 year, 10 months ago

Congrats man, not everyone goes back to tell they were wrong and corrects them selves. We need more people like this on this platform
upvoted 5 times

sly75 2 years, 1 month ago

Yes I agree, it's E
upvoted 2 times

4be8126 2 years, 2 months ago

The correct code block to return a DataFrame containing only the rows from DataFrame storesDF where the value in column sqft is less than or equal to 25,000 OR the value in column customerSatisfaction is greater than or equal to 30 is:

storesDF.filter((col("sqft") <= 25000) | (col("customerSatisfaction") >= 30))
Option A uses a single pipe (|) instead of the correct syntax of two vertical bars (||) to represent "OR" logic, and also uses the wrong syntax for column referencing.

Option B uses the correct or operator, but also uses the wrong syntax for column referencing.

Option C uses the correct operator and syntax for column referencing, but does not use the col() function to reference column names.

Option D uses the col() function, but also uses the wrong syntax for column referencing.

Option E uses the correct syntax for both column referencing and logical operator, and correctly specifies the parentheses to ensure the proper order of operations.

Therefore, the correct answer is E.

storesDF.filter((col("sqft") <= 25000) | (col("customerSatisfaction") >= 30))
upvoted 2 times

Question #22 *Topic 1*

Which of the following code blocks returns a new DataFrame from DataFrame storesDF where column storeId is of the type string?

A. storesDF.withColumn("storeId, cast(col("storeId"), StringType()))

B. storesDF.withColumn("storeId, col("storeId").cast(StringType()))

C. storesDF.withColumn("storeId, cast(storeId).as(StringType)

D. storesDF.withColumn("storeId, col(storeId).cast(StringType)

E. storesDF.withColumn("storeId, cast("storeId").as(StringType()))

Suggested Answer: *B*

Community vote distribution

B (100%)

jds0 11 months, 1 week ago

Selected Answer: B

Answer is B but with a typo:
See code below (Spark 3.5.1):

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, cast
from pyspark.sql.types import StringType

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 3, 20000, "A"),
(1, 1, 50000, "A"),
(2, 2, 70000, "A"),
]
storesDF = spark.createDataFrame(data, ["storeID", "customerSatisfaction", "sqft", "division"])

storesDF.withColumn("storeId", col("storeId").cast(StringType())).printSchema()
# root
# |-- storeId: string (nullable = true)
# |-- customerSatisfaction: long (nullable = true)
# |-- sqft: long (nullable = true)
# |-- division: string (nullable = true)
```

upvoted 1 times

DataEngine 1 year, 8 months ago

Anwer is B but it has a typo

upvoted 2 times

ZSun 2 years ago

cast is a method belongs to class pyspark.sql.column
therefore, A C E are wrong. it should be dataframe.column.cast() or col('col_name').cast()
B is correct, with small typo

upvoted 1 times

dduque10 2 years, 1 month ago

Selected Answer: B

All answers are wrong because the first argument does not have the closing quotes :D, apart from that, it is B

upvoted 3 times

4be8126 2 years, 2 months ago

Selected Answer: B

The correct code block to return a new DataFrame from DataFrame storesDF where column storeId is of the type string is:

storesDF.withColumn("storeId", col("storeId").cast(StringType()))
Option A has an extra quotation mark after "storeId" and is missing a closing parenthesis for the cast() function.

Option B correctly uses the cast() function to change the data type, but has a typo where "storeId" is repeated inside the string argument for the withColumn() function.

Option C is missing the col() function to reference the storeId column, and also has a typo with the closing parentheses for the cast() function.

Option D correctly references the storeId column using col(), but has a typo with the quotation marks and parentheses.

Option E has a syntax error where the cast() function is inside the quotation marks, and is also missing the col() function to reference the storeId column.

Therefore, the correct answer is B.

storesDF.withColumn("storeId", col("storeId").cast(StringType()))
upvoted 2 times

Question #23 *Topic 1*

Which of the following code blocks returns a new DataFrame with a new column employeesPerSqft that is the quotient of column numberOfEmployees and column sqft, both of which are from DataFrame storesDF? Note that column employeesPerSqft is not in the original DataFrame storesDF.

A. storesDF.withColumn("employeesPerSqft", col("numberOfEmployees") / col("sqft"))

B. storesDF.withColumn("employeesPerSqft", "numberOfEmployees" / "sqft")

C. storesDF.select("employeesPerSqft", "numberOfEmployees" / "sqft")

D. storesDF.select("employeesPerSqft", col("numberOfEmployees") / col("sqft"))

E. storesDF.withColumn(col("employeesPerSqft"), col("numberOfEmployees") / col("sqft"))

Suggested Answer: *A*

Community vote distribution

A (100%)

jds0 11 months, 1 week ago

Selected Answer: A

Answer: A
All other options do not work

Code:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 3, 20, "A"),
(1, 1, 50, "A"),
(2, 2, 70, "A"),
]
storesDF = spark.createDataFrame(data, ["storeID", "numberOfEmployees", "sqft", "division"])

storesDF.withColumn("employeesPerSqft", col("numberOfEmployees") / col("sqft")) # A.
```

upvoted 2 times

newusername 1 year, 9 months ago

Selected Answer: A

Test:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Initializing Spark session (if not already initialized)
spark = SparkSession.builder.appName("databricks_example").getOrCreate()

# Creating some synthetic data for storesDF
data = [
{"storeId": 1, "numberOfEmployees": 10, "sqft": 500},
{"storeId": 2, "numberOfEmployees": 15, "sqft": 750},
{"storeId": 3, "numberOfEmployees": 8, "sqft": 400}
]

storesDF = spark.createDataFrame(data)
```

```
# Option A:
try:
df_a = storesDF.withColumn("employeesPerSqft", col("numberOfEmployees") / col("sqft"))
df_a.show()
print("Option A works")
except Exception as e:
print("Option A doesn't work:", str(e))
```

upvoted 1 times

SonicBoom10C9 2 years, 1 month ago

Selected Answer: A

C, D are wrong as exmployeesPerSqft cannot be selected, it doesn't exist. Also, that is not proper select syntax anyway. B does not select existing columns using col(), and E refers to employeesPerSqft as an existing column; also, it cannot be the first argument for withColumn().

upvoted 2 times

4be8126 2 years, 2 months ago

storesDF.select("employeesPerSqft", col("numberOfEmployees") / col("sqft"))

This code block selects the columns "employeesPerSqft" and the quotient of "numberOfEmployees" and "sqft" from the DataFrame storesDF. However, since "employeesPerSqft" is not a column in the original storesDF, this code block would throw an error.

To create a new column "employeesPerSqft" in the resulting DataFrame, we need to use the withColumn() method instead of select(). Here's the corrected code block:

storesDF.withColumn("employeesPerSqft", col("numberOfEmployees") / col("sqft"))

This code block adds a new column "employeesPerSqft" to the storesDF DataFrame. The new column is created by dividing the values in column "numberOfEmployees" by the values in column "sqft".

upvoted 1 times

4be8126 2 years, 2 months ago

The correct code block to return a new DataFrame with a new column employeesPerSqft that is the quotient of column numberOfEmployees and column sqft from DataFrame storesDF is:

storesDF.withColumn("employeesPerSqft", col("numberOfEmployees") / col("sqft"))

Option A correctly uses the withColumn() function to create a new column employeesPerSqft by dividing column numberOfEmployees by column sqft.

Option B has a syntax error because it uses quotation marks to reference column names instead of the col() function.

Option C also has a syntax error because it uses quotation marks to reference column names instead of the col() function, and also uses the select() function instead of withColumn() to create a new column.

Option D correctly references column names using col() and uses the select() function to return a DataFrame with only the two selected columns.

Option E has a syntax error where col() is used as a first argument instead of a second argument for the withColumn() function.

Therefore, the correct answer is A.

storesDF.withColumn("employeesPerSqft", col("numberOfEmployees") / col("sqft"))

upvoted 1 times

Question #24 *Topic 1*

The code block shown below should return a new DataFrame from DataFrame storesDF where column modality is the constant string "PHYSICAL", Assume DataFrame storesDF is the only defined language variable. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.
Code block:
storesDF. _1_(_2_,_3_(_4_))

A. 1. withColumn
2. "modality"
3. col
4. "PHYSICAL"

B. 1. withColumn
2. "modality"
3. lit
4. PHYSICAL

C. 1. withColumn
2. "modality"
3. lit
4. "PHYSICAL"

D. 1. withColumn
2. "modality"
3. SrtringType
4. "PHYSICAL"

E. 1. newColumn
2. modality
3. SrtringType
4. PHYSICAL

Suggested Answer: *C*

Community vote distribution

C (100%)

jds0 11 months, 1 week ago

Selected Answer: C

Answer is C
Code:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import lit

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 3, 20, "A"),
(1, 1, 50, "A"),
(2, 2, 70, "A"),
]
storesDF = spark.createDataFrame(data, ["storeID", "numberOfEmployees", "sqft", "division"])

storesDF.withColumn("modality", lit("PHYSICAL"))
```

upvoted 1 times

newusername 1 year, 9 months ago

Selected Answer: C

Correct

upvoted 1 times

4be8126 2 years, 2 months ago

lit and col are two functions in PySpark that are used to create or reference columns in a DataFrame.

lit: This function is used to create a column with a literal value. It returns a Column expression of literal value. For example, lit(2) creates a Column with a value of 2. It can be useful when you want to add a new column to a DataFrame with a constant value for all rows.

col: This function is used to reference an existing column in a DataFrame. It returns a Column expression that represents a column. For example, col("age") returns a Column expression that represents the "age" column in a DataFrame. It can be useful when you want to select, filter or transform an existing column in a DataFrame.

In short, lit is used to create a new column with a constant value, while col is used to reference an existing column in a DataFrame.

upvoted 1 times

4be8126 2 years, 2 months ago

Selected Answer: C

Option C is the correct answer. Here's why:

The withColumn function is used to add a new column to the DataFrame based on an existing column or a constant value. The first blank (_1_) should be replaced with withColumn to indicate that we want to add a new column.

The second blank (_2_) should be replaced with the name of the column we want to add. In this case, we want to add a column called modality.

The third blank (_3_) should be replaced with a function that will create the values for the new column. In this case, we want to create a column that has the constant value "PHYSICAL". The lit function can be used to create a column with a literal value.

Finally, the fourth blank (_4_) should be replaced with the actual value we want to use for the new column. Since we want to use the string "PHYSICAL", it should be wrapped in quotation marks to indicate that it is a string.

Therefore, option C correctly fills in the blanks to give us the following code block:

storesDF.withColumn("modality", lit("PHYSICAL"))

upvoted 2 times

Question #25 *Topic 1*

Which of the following code blocks returns a DataFrame where column storeCategory from DataFrame storesDF is split at the underscore character into column storeValueCategory and column storeSizeCategory?

A sample of DataFrame storesDF is displayed below:

storeId	open	openDate	storeCategory
0	true	1100746394	VALUE_MEDIUM
1	true	944572255	MAINSTREAM_SMALL
2	false	925495628	PREMIUM_LARGE
3	true	1397353092	VALUE_MEDIUM
4	true	986505057	VALUE_LARGE
5	true	955988614	PREMIUM_LARGE
...	...	...	...

A. (storesDF.withColumn("storeValueCategory", split(col("storeCategory"), "_")[1])
.withColumn("storeSizeCategory", split(col("storeCategory"), "_")[2]))

B. (storesDF.withColumn("storeValueCategory", col("storeCategory").split("_")[0])
.withColumn("storeSizeCategory", col("storeCategory").split("_")[1]))

C. (storesDF.withColumn("storeValueCategory", split(col("storeCategory"), "_")[0])
.withColumn("storeSizeCategory", split(col("storeCategory"), "_")[1]))

D. (storesDF.withColumn("storeValueCategory", split("storeCategory", "_")[0])
.withColumn("storeSizeCategory", split("storeCategory", "_")[1]))

E. (storesDF.withColumn("storeValueCategory", col("storeCategory").split("_")[1])
.withColumn("storeSizeCategory", col("storeCategory").split("_")[2]))

Suggested Answer: *C*

Community vote distribution

C (100%)

ronfun Highly Voted 2 years, 2 months ago

Both C or D are correct. Function split accepts both col and str.
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.functions.split.html?highlight=split#pyspark.sql.functions.split

upvoted 8 times

NickWerbung 1 year, 12 months ago

Both C or D are correct!

upvoted 2 times

4be8126 2 years, 2 months ago

Option D is not correct because the split function should be used with the col function to split the values in a column. In option D, the split function is used with a string literal rather than a column, which will result in an error.

upvoted 3 times

bp_a_user Most Recent 5 months, 1 week ago

Selected Answer: C

Only C is correct
pyspark.sql.functions.split(str, pattern, limit=-1)
The exam is about Spark 3.0, the option to also use col is now existing (3.5, but not in 3.0)

upvoted 1 times

bp_a_user 5 months, 1 week ago

sry, only D is correct

upvoted 1 times

jds0 11 months, 1 week ago

Selected Answer: C

Both C or D work in Spark 3.5.1, but C is probably better for backward compatibility.
See code example below:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import split, col

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, True, 10020, "VALUE_MEDIUM"),
(1, True, 10050, "MAINSTREAM_SMALL"),
(2, False, 10070, "PREMIUM_LARGE"),
]
storesDF = spark.createDataFrame(data, ["storeID", "open", "openDate", "storeCategory"])

(storesDF.withColumn("storeValueCategory", split(col("storeCategory"), "_")[0]).withColumn("storeSizeCategory", split(col("storeCategory"), "_")
[1])).show()
(storesDF.withColumn("storeValueCategory", split("storeCategory", "_")[0]).withColumn("storeSizeCategory", split("storeCategory", "_")[1])).show()
```

upvoted 1 times

newusername 1 year, 9 months ago

Selected Answer: C

C
you can check, by running the code below:

```
from pyspark.sql import SparkSession

# Initialize Spark session
spark = SparkSession.builder.appName("split_test").getOrCreate()

# Create synthetic data
data = [
{"storeCategory": "value1_size1"},
{"storeCategory": "value2_size2"},
{"storeCategory": "value3_size3"},
]

storesDF = spark.createDataFrame(data)
storesDF.show()

from pyspark.sql.functions import split, col

# Option C


newDF = (storesDF.withColumn("storeValueCategory", split(col("storeCategory"), "_")[0])
.withColumn("storeSizeCategory", split(col("storeCategory"), "_")[1]))
newDF.show()
```

upvoted 2 times

zozoshanky 1 year, 11 months ago

c is correct

upvoted 1 times

4be8126 2 years, 2 months ago

Selected Answer: C

Option C returns a DataFrame where column storeCategory from DataFrame storesDF is split at the underscore character into column storeValueCategory and column storeSizeCategory.

The correct code is:

```
(storesDF.withColumn("storeValueCategory", split(col("storeCategory"), "_")[0])
.withColumn("storeSizeCategory", split(col("storeCategory"), "_")[1]))
```

Explanation:

split(col("storeCategory"), "_") splits the values in column storeCategory by the "_" character and returns an array of strings.

[0] gets the first element of the resulting array and assigns it to the new column storeValueCategory.

[1] gets the second element of the resulting array and assigns it to the new column storeSizeCategory.

withColumn is used to create the new columns and returns a new DataFrame.

upvoted 2 times

Question #26 *Topic 1*

Which of the following code blocks returns a new DataFrame where column productCategories only has one word per row, resulting in a DataFrame with many more rows than DataFrame storesDF?
A sample of storesDF is displayed below:

storeId	productCategories
0	[netus, pellentes...
1	[consequat enim,...
2	[massa, a, vitae,...
3	[aliquam, donec,...
4	[condimentum, fer...
5	[viverra habitan...
...	...

A. storesDF.withColumn("productCategories", explode(col("productCategories")))

B. storesDF.withColumn("productCategories", split(col("productCategories")))

C. storesDF.withColumn("productCategories", col("productCategories").explode())

D. storesDF.withColumn("productCategories", col("productCategories").split())

E. storesDF.withColumn("productCategories", explode("productCategories"))

Suggested Answer: *A*

Community vote distribution

A (88%) 13%

jds0 11 months, 1 week ago

Selected Answer: A

Both option A and E work with spark 3.5.1.
But A is better for backward compatibility.
See code example below:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, explode

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, ["value 1", "value 2", "value 3"]),
(1, ["value 1", "value 2", "value 3"]),
(2, ["value 1", "value 2", "value 3"]),
]
storesDF = spark.createDataFrame(data, ["storeID", "productCategories"])

storesDF.withColumn("productCategories", explode(col("productCategories"))).show() # A.
storesDF.withColumn("productCategories", explode("productCategories")).show() # E.
```

upvoted 1 times

bettermakeme 1 year, 2 months ago

A and E are correct

upvoted 1 times

arturffsi 1 year, 3 months ago

Selected Answer: E

Both A and E are correct according to the new version

upvoted 1 times

newusername 1 year, 9 months ago

Selected Answer: A

A is correct, use below code to test:

```
from pyspark.sql import SparkSession

# Initializing Spark session
spark = SparkSession.builder.appName("test").getOrCreate()

# 1. Creating DataFrame with an array column
data_array = [
(1, ["electronics", "clothes", "toys"]),
(2, ["groceries", "electronics"]),
(3, ["books", "clothes"]),
]

storesDF = spark.createDataFrame(data_array, ["ID", "productCategories"])
storesDF.show()

df_array = storesDF.withColumn("productCategories", explode(col("productCategories")))
df_array.show()
```

upvoted 3 times

newusername 1 year, 9 months ago

But E works as well, sadly. What has to be chosen then?

```
from pyspark.sql import SparkSession

# Initializing Spark session
spark = SparkSession.builder.appName("test").getOrCreate()

# 1. Creating DataFrame with an array column
data_array = [
(1, ["electronics", "clothes", "toys"]),
(2, ["groceries", "electronics"]),
(3, ["books", "clothes"]),
]

storesDF = spark.createDataFrame(data_array, ["ID", "productCategories"])
storesDF.show()

#df_array = storesDF.withColumn("productCategories", explode(col("productCategories")))
#df_array.show()


#check E
df_array = storesDF.withColumn("productCategories", explode("productCategories"))
df_array.show()
```

upvoted 2 times

newusername 1 year, 7 months ago

E for 3.0

upvoted 1 times

NickWerbung 1 year, 12 months ago

Both A and E are correct.

upvoted 4 times

mhaskins 2 years, 1 month ago

Selected Answer: A

While the Explode function allows for a str or Column input, this requires the col() wrapper because it is used in a withColumn() call, where the 2nd parameter requires the column object.

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.withColumn.html?highlight=withcolumn#pyspark.sql.DataFrame.withColumn

upvoted 2 times

4be8126 2 years, 2 months ago

Selected Answer: A

Option A is correct: storesDF.withColumn("productCategories", explode(col("productCategories"))).

Explanation:

The explode function is used to transform a column of arrays or maps into multiple rows, one for each element in the array or map. In this case, productCategories is a column with arrays of strings.

The withColumn function is used to add a new column or update an existing column. The first argument is the name of the new or existing column, and the second argument is the expression that defines the values for the column.

upvoted 1 times

Question #27 *Topic 1*

Which of the following code blocks returns a new DataFrame with column storeDescription where the pattern "Description: " has been removed from the beginning of column storeDescription in DataFrame storesDF?

A sample of DataFrame storesDF is below:

storeId	storeDescription
0	Description: ultr...
1	Description: sagi...
2	Description: port...
3	Description: tris...
4	Description: ulla...
...	

A. storesDF.withColumn("storeDescription", regexp_replace(col("storeDescription"), "^Description: "))

B. storesDF.withColumn("storeDescription", col("storeDescription").regexp_replace("^Description: ", ""))

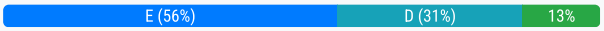
C. storesDF.withColumn("storeDescription", regexp_extract(col("storeDescription"), "^Description: ", ""))

D. storesDF.withColumn("storeDescription", regexp_replace("storeDescription", "^Description: ", ""))

E. storesDF.withColumn("storeDescription", regexp_replace(col("storeDescription"), "^Description: ", ""))

Suggested Answer: *E*

Community vote distribution



jds0 11 months, 1 week ago

Selected Answer: E

Both D and E work with Spark 3.5.1 but E is better for backward compatibility

See code below:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, regexp_replace

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, "Description: Store 0"),
(1, "Description: Store 1"),
(2, "Description: Store 2"),
]
storesDF = spark.createDataFrame(data, ["storeID", "StoreDescription"])

storesDF.withColumn("storeDescription", regexp_replace(col("StoreDescription"), "Description: ", "")).show()
storesDF.withColumn("storeDescription", regexp_replace("StoreDescription", "Description: ", "")).show()
```

upvoted 1 times

arturffsi 1 year, 3 months ago

Both D and E are correct according to the new version

upvoted 1 times

azure_bimonster 1 year, 4 months ago

Selected Answer: E

E is most likely correct in this scenario

upvoted 1 times

newusername 1 year, 9 months ago

Both work:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import regexp_replace,regexp_extract, col
spark = SparkSession.builder.appName("test").getOrCreate()

data = [
(1, "Description: This is a tech store. Description: This"),
(2, "Description: This is a grocery store."),
(3, "Description: This is a book store."),
]
storesDF = spark.createDataFrame(data, ["storeID", "storeDescription"])
storesDF.show(truncate=False)

#Case D
print ("Case D")
storesDF = storesDF.withColumn("storeDescription", regexp_replace("storeDescription", "^Description: ", ""))
storesDF.show(truncate=False)


#Case E
print ("Case E")
storesDF = storesDF.withColumn("storeDescription", regexp_replace(col("storeDescription"), "^Description: ", ""))
storesDF.show(truncate=False)
```

upvoted 3 times

Dgohel 1 year, 11 months ago

regexp_replace(str, regexp, rep [, position])

This is what Databricks documentation says. You guys can debate between D and E but actually question clearly says to remove from the begging of the string. And if you take answer D it takes whole only one constant string “storeDescription” to match pattern and will return empty string after Description for each row.

So if you have debate between D, E then E is the correct answer.

upvoted 2 times

zozoshanky 1 year, 11 months ago

E is the answer tested

upvoted 2 times

NickWerbung 1 year, 12 months ago

Both D and E are correct.

upvoted 1 times

SonicBoom10C9 2 years, 1 month ago

Selected Answer: E

It's between D and E, and D is wrong as there is no replacement string expression (which is a required argument/parameter). Thus, E wins as the correct option.

upvoted 1 times

ZSun 2 years ago

this is completely wrong explanation. Both D and E has replacement expression, the only difference is how they call the replaced column. Both D and E are correct, but D works for Pyspark 2.0. D and E both work Pyspark 3.0+. Period!

upvoted 7 times

ZSun 2 years ago

I think what you really mean, "there is no replacement string expression", is for option A.
The only difference between A and E, is about the claim of replacement string expression

upvoted 1 times

sly75 2 years, 1 month ago

Selected Answer: E

Correct answer is E indeed
- According to the pyspark doc, the syntax is regexp_replace(str, pattern, replacement)
-> it means that it's not a function of the column object

- storeDescription is a String field

https://spark.apache.org/docs/3.0.0/api/python/pyspark.sql.html#pyspark.sql.functions.regexp_replace
upvoted 2 times

pierre_grns 2 years, 2 months ago

Selected Answer: D

Correct answer is D.
First, regexp_replace/regexp_extract are from sql.functions. They cannot be applied directly after a column Object => B is incorrect.
Second, regexp_replace/regexp_extract accept a STRING Object as a first argument to specify the column. Check the documentation there :
https://spark.apache.org/docs/3.0.0/api/python/pyspark.sql.html#module-pyspark.sql.functions => A, C, E are incorrects.
upvoted 2 times

sly75 2 years, 1 month ago

Almost right but it's not about "String object" but "String value". So the correct answer is indeed the answer E ;)
upvoted 2 times

4be8126 2 years, 2 months ago

Selected Answer: E

The correct answer is option E: storesDF.withColumn("storeDescription", regexp_replace(col("storeDescription"), "^Description: ", "")).

This code block uses the withColumn() function to create a new column called storeDescription. It uses the regexp_replace() function to replace the pattern "^Description: " at the beginning of the string in the storeDescription column with an empty string. This effectively removes the pattern from the beginning of the string in each row of the column.
upvoted 4 times

4be8126 2 years, 2 months ago

The correct code block that returns a new DataFrame with column storeDescription where the pattern "Description: " has been removed from the beginning of column storeDescription in DataFrame storesDF is:

A. storesDF.withColumn("storeDescription", regexp_replace(col("storeDescription"), "^Description: "))

This code uses the regexp_replace function to replace the pattern "^Description: " (which matches the string "Description: " at the beginning of the string) with an empty string in the column storeDescription. The resulting DataFrame will have the modified storeDescription column.

Option B has a syntax error because the regexp_replace function should be called on the column using the dot notation instead of passing it as the second argument.

Option C uses the regexp_extract function, which extracts a substring matching a regular expression pattern. It doesn't remove the pattern from the string.

Option D has a syntax error because the column name is not wrapped in the col function.

Option E is the same as option A, except that it uses the col function unnecessarily.
upvoted 1 times

4be8126 2 years, 2 months ago

Selected Answer: A

Option A is correct: storesDF.withColumn("productCategories", explode(col("productCategories"))).

Explanation:

The explode function is used to transform a column of arrays or maps into multiple rows, one for each element in the array or map. In this case, productCategories is a column with arrays of strings.

The withColumn function is used to add a new column or update an existing column. The first argument is the name of the new or existing column, and the second argument is the expression that defines the values for the column.
upvoted 1 times

sly75 2 years, 1 month ago

You got the wrong question :°
upvoted 2 times

ronfun 2 years, 2 months ago

Both D and E are correct answer.

upvoted 2 times

TC007 2 years, 2 months ago

Selected Answer: D

This should actually be D sorry for the wrong answer. refer to this, https://sparkbyexamples.com/pyspark/pyspark-replace-column-values/

upvoted 3 times

TC007 2 years, 2 months ago

Selected Answer: A

The regexp_replace function is used to remove the pattern "Description: " from the beginning of the column storeDescription. The ^ symbol indicates the beginning of the string, and the pattern "Description: " is replaced with an empty string. This results in a new DataFrame with column storeDescription where the pattern "Description: " has been removed from the beginning of each cell in that column.

upvoted 1 times

4be8126 2 years, 2 months ago

Option A is incorrect because the regexp_replace function requires two arguments: the column to be transformed and the regular expression pattern to be replaced. In the given code block, only the regular expression pattern is provided, but not the column to be transformed.

The correct syntax to use regexp_replace on a DataFrame column is regexp_replace(col(column_name), pattern, replacement), where col(column_name) specifies the DataFrame column to be transformed, pattern specifies the regular expression pattern to be replaced, and replacement specifies the new string to replace the matched pattern.

Therefore, the correct code block to remove the pattern "Description: " from the beginning of the storeDescription column in DataFrame storesDF is:

storesDF.withColumn("storeDescription", regexp_replace(col("storeDescription"), "^Description: ", ""))

upvoted 2 times

Question #28 *Topic 1*

Which of the following code blocks returns a new DataFrame where column division from DataFrame storesDF has been replaced and renamed to column state and column managerName from DataFrame storesDF has been replaced and renamed to column managerFullName?

A. (storesDF.withColumnRenamed(["division", "state"], ["managerName", "managerFullName"])

B. (storesDF.withColumn("state", col("division"))
.withColumn("managerFullName", col("managerName")))

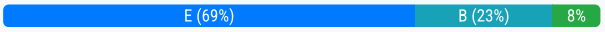
C. (storesDF.withColumn("state", "division")
.withColumn("managerFullName", "managerName"))

D. (storesDF.withColumnRenamed("state", "division")
.withColumnRenamed("managerFullName", "managerName"))

E. (storesDF.withColumnRenamed("division", "state")
.withColumnRenamed("managerName", "managerFullName"))

Suggested Answer: *E*

Community vote distribution



4be8126 Highly Voted 1 year, 8 months ago

Selected Answer: E

Option E is the correct answer. The withColumnRenamed function renames an existing column, whereas withColumn creates a new column. To replace the "division" column with a new column "state" and rename the "managerName" column to "managerFullName", we need to use withColumnRenamed. So option E is correct, where we first rename "division" to "state" and then rename "managerName" to "managerFullName".

upvoted 7 times

azure_bimonster Most Recent 10 months, 3 weeks ago

Selected Answer: E

E is the right answer for this question

upvoted 3 times

evertonllins 1 year, 4 months ago

Selected Answer: B

If we take in consideration this part of the text .-> "has been REPLACED and renamed to..." it means that the columnis not only renames but replaved with the contents of the other clumn. In this case the righr answer is B.

Do you guys thin this insterpretation is correct? Thanks for the feedback.

upvoted 3 times

zozoshanky 1 year, 5 months ago

D is right

upvoted 1 times

NickWerbung 1 year, 6 months ago

DataFrame.withColumnRenamed(existing, new)

upvoted 2 times

Mohitsain 1 year, 6 months ago

Selected Answer: D

this is the right answer.

upvoted 1 times

Question #29 *Topic 1*

The code block shown contains an error. The code block is intended to return a new DataFrame where column sqft from DataFrame storesDF has had its missing values replaced with the value 30,000. Identify the error.
A sample of DataFrame storesDF is displayed below:

storeId	sqft
0	43161
1	51200
2	null
3	78367
4	null
...	...

Code block:
storesDF.na.fill(30000, col("sqft"))

A. The argument to the subset parameter of fill() should be a string column name or a list of string column names rather than a Column object.

B. The na.fill() operation does not work and should be replaced by the dropna() operation.

C. he argument to the subset parameter of fill() should be a the numerical position of the column rather than a Column object.

D. The na.fill() operation does not work and should be replaced by the nafill() operation.

E. The na.fill() operation does not work and should be replaced by the fillna() operation.

Suggested Answer: *A*

Community vote distribution

A (80%) E (20%)

ZSun Highly Voted 2 years ago

Correct anwser is A.
even for most updated version, spark 3.4. na.fill() still functioning, it is an alias of fillna()
Mr. 4be8126 , □□□□□□□□□
upvoted 7 times

jds0 Most Recent 11 months, 1 week ago

Selected Answer: A

The most correct answer seems to be A:

Code below with Spark 3.5.1.

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.errors import PySparkTypeError

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 43161),
(1, 51200),
(2, None),
(3, 78367),
(4, None),
]
storesDF = spark.createDataFrame(data, ["storeID", "sqft"])

# storesDF.show()
```

```
try:
storesDF.na.fill(30000, col("sqft"))
except PySparkTypeError as e:
print(e)

storesDF.na.fill(30000, "sqft").show()
storesDF.na.fill(30000, ["sqft"]).show()
storesDF.fillna(30000, ["sqft"]).show()
storesDF.fillna(30000, "sqft").show()
```

upvoted 2 times

azure_bimonster 1 year, 4 months ago

Selected Answer: A

We don't need any replacement here. A would be correct.

In PySpark both fillna() and fill() are used to replace missing or null values of a DataFrame. Functionally they both perform same. One can choose either of these based on preference. These are used mainly for handling missing data in PySpark.

upvoted 1 times

4be8126 2 years, 2 months ago

The correct answer is either A or E, depending on the version of Spark being used.

In Spark 2.x, the correct method to replace missing values is na.fill(). Option A is correct in Spark 2.x, as it correctly specifies the column to apply the fill operation to using a Column object.

However, in Spark 3.x, the method has been renamed to fillna(). Therefore, in Spark 3.x, the correct answer is E, as it uses the correct method name.

Both A and E accomplish the same task of replacing missing values in the sqft column with 30,000, so either answer can be considered correct depending on the version of Spark being used.

upvoted 2 times

peekaboo15 2 years, 2 months ago

Selected Answer: A

the answer should be A. See this link for reference
https://sparkbyexamples.com/pyspark/pyspark-fillna-fill-replace-null-values/

upvoted 2 times

TC007 2 years, 2 months ago

Selected Answer: E

The error in the code block is that the method na.fill() should be replaced by fillna() to fill the missing values in the column "sqft" with the value 30,000.

upvoted 1 times

Question #30 *Topic 1*

Which of the following operations fails to return a DataFrame with no duplicate rows?

A. DataFrame.dropDuplicates()

B. DataFrame.distinct()

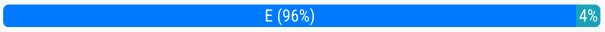
C. DataFrame.drop_duplicates()

D. DataFrame.drop_duplicates(subset = None)

E. DataFrame.drop_duplicates(subset = "all")

Suggested Answer: *A*

Community vote distribution



4be8126 Highly Voted 2 years, 2 months ago

Selected Answer: E

A. DataFrame.dropDuplicates(): This method returns a new DataFrame with distinct rows based on all columns. It should return a DataFrame with no duplicate rows.

B. DataFrame.distinct(): This method returns a new DataFrame with distinct rows based on all columns. It should also return a DataFrame with no duplicate rows.

C. DataFrame.drop_duplicates(): This is an alias for DataFrame.dropDuplicates(). It should also return a DataFrame with no duplicate rows.

D. DataFrame.drop_duplicates(subset=None): This method returns a new DataFrame with distinct rows based on all columns. It should return a DataFrame with no duplicate rows.

E. DataFrame.drop_duplicates(subset="all"): This method attempts to drop duplicates based on all columns but returns an error, because "all" is not a valid argument for the subset parameter. So this operation fails to return a DataFrame with no duplicate rows.

Therefore, the correct answer is E.

upvoted 10 times

TC007 Highly Voted 2 years, 3 months ago

Selected Answer: E

Option E is incorrect as "all" is not a valid value for the subset parameter in the drop_duplicates() method. The correct value should be a column name or a list of column names to be used as the subset to identify duplicate rows.

All other options (A, B, C, and D) can be used to return a DataFrame with no duplicate rows. The dropDuplicates(), distinct(), and drop_duplicates() methods are all equivalent and return a new DataFrame with distinct rows. The drop_duplicates() method also accepts a subset parameter to specify the columns to use for identifying duplicates, and when the subset parameter is not specified, all columns are used. Therefore, both option A and C are valid, and option D is also valid as it is equivalent to drop_duplicates() with no subset parameter.

upvoted 10 times

smd_ 11 months ago

bro the question asks (no duplicate rows) ,that means the correct answer should be able to return rows with duplication. and (E) does that so. Focus on question.

upvoted 1 times

jds0 Most Recent 11 months, 1 week ago

Selected Answer: E

It's E, see code below:

```
# Drop duplicates in a DataFrame
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
```

```
from pyspark.errors import PySparkTypeError

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 43161),
(0, 43161),
(1, 51200),
(2, None),
(2, None),
(3, 78367),
(4, None),
]
storesDF = spark.createDataFrame(data, ["storeID", "sqft"])

try:
storesDF.dropDuplicates().show()
except PySparkTypeError as e:
print(e)

try:
storesDF.distinct().show()
except PySparkTypeError as e:
print(e)

try:
storesDF.drop_duplicates().show()
except PySparkTypeError as e:
print(e)

try:
storesDF.drop_duplicates(subset=None).show()
except PySparkTypeError as e:
print(e)

try:
storesDF.drop_duplicates(subset="all").show()
except PySparkTypeError as e:
print(e)
```

upvoted 2 times

dbdantas 1 year, 2 months ago

Selected Answer: E

the answer is E

upvoted 1 times

dbdantas 1 year, 2 months ago

Selected Answer: E

E

PySparkTypeError: [NOT_LIST_OR_TUPLE] Argument `subset` should be a list or tuple, got str.

upvoted 1 times

azurearch 1 year, 3 months ago

DataFrame.drop_duplicates(subset = "all") - this is specific to pandas

upvoted 1 times

azurearch 1 year, 3 months ago

Option E . df.drop_duplicates(subset = "all") returns error

SparkTypeError: [NOT_LIST_OR_TUPLE] Argument `subset` should be a list or tuple, got str.

upvoted 2 times

cookiemonster42 1 year, 11 months ago

Selected Answer: B

B is the right one, as TC007 said, the argument for drop_duplicates is a subset of columns:

DataFrame.dropDuplicates(subset: Optional[List[str]] = None) → pyspark.sql.dataframe.DataFrame[source]
Return a new DataFrame with duplicate rows removed, optionally only considering certain columns.

For a static batch DataFrame, it just drops duplicate rows. For a streaming DataFrame, it will keep all data across triggers as intermediate state to drop duplicates rows. You can use withWatermark() to limit how late the duplicate data can be and the system will accordingly limit the state. In addition, data older than watermark will be dropped to avoid any possibility of duplicates.

drop_duplicates() is an alias for dropDuplicates().

Parameters
subsetList of column names, optional
List of columns to use for duplicate comparison (default All columns).

upvoted 1 times

cookiemonster42 1 year, 11 months ago

OMG, I got it all wrong, the answer is E :)

upvoted 3 times

ItsAB 1 year, 11 months ago

the correct answer is E

upvoted 1 times

Question #31 *Topic 1*

Which of the following code blocks will most quickly return an approximation for the number of distinct values in column division in DataFrame storesDF?

A. storesDF.agg(approx_count_distinct(col("division")).alias("divisionDistinct"))

B. storesDF.agg(approx_count_distinct(col("division"), 0.01).alias("divisionDistinct"))

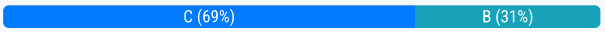
C. storesDF.agg(approx_count_distinct(col("division"), 0.15).alias("divisionDistinct"))

D. storesDF.agg(approx_count_distinct(col("division"), 0.0).alias("divisionDistinct"))

E. storesDF.agg(approx_count_distinct(col("division"), 0.05).alias("divisionDistinct"))

Suggested Answer: *A*

Community vote distribution



4be8126 Highly Voted 2 years, 2 months ago

Selected Answer: B

To quickly return an approximation for the number of distinct values in column division in DataFrame storesDF, the most efficient code block to use would be:

B. storesDF.agg(approx_count_distinct(col("division"), 0.01).alias("divisionDistinct"))

Using the approx_count_distinct() function allows for an approximate count of the distinct values in the column without scanning the entire DataFrame. The second parameter passed to the function is the maximum estimation error allowed, which in this case is set to 0.01. This is a trade-off between the accuracy of the estimate and the computational cost. Option C may still be efficient but with a larger estimation error of 0.15. Option A and D are not correct as they do not specify the estimation error, which means that the function would use the default value of 0.05. Option E specifies an estimation error of 0.05, but a smaller error of 0.01 is a better choice for a more accurate estimate with less computational cost.

upvoted 5 times

carlosmps 1 year ago

But your answer contradicts the question, they only ask you for the fastest way, while the error value is closer to zero, then it will take more time and resources. 0.15>0.01, that means that option C will be faster, it will have more errors, but it will be the fastest.

upvoted 3 times

ZSun 2 years ago

I see you reply in a lot of question, barely correct.
bro, you need to stop comment wrong information here.
This question only ask for efficiency, no need to balance between accuracy and efficiency.
Stop posting ChatGPT answer here

upvoted 19 times

outwalker 1 year, 7 months ago

I noticed the same thing with this ID - bro has confidence, I have to triple make sure because he keeps answering wrong thus creating doubts in my head.

upvoted 1 times

oussa_ama Most Recent 10 months, 2 weeks ago

Selected Answer: C

C is correct because it will provide the fastest approximate count with a standard deviation of 0.15

upvoted 3 times

zozoshanky 1 year, 6 months ago

B. storesDF.agg(approx_count_distinct(col("division"), 0.01).alias("divisionDistinct"))

Explanation:

approx_count_distinct(col("division"), 0.01): This uses the approx_count_distinct function to approximate the number of distinct values in the

"division" column with a relative error of 1%. The smaller the relative error, the more accurate the approximation, but it may require more resources.
.alias("divisionDistinct"): This renames the result column to "divisionDistinct" for better readability.
So, the correct answer is:

B. storesDF.agg(approx_count_distinct(col("division"), 0.01).alias("divisionDistinct"))
upvoted 1 times

smd_ 1 year ago
C is the correct answer.
C. storesDF.agg(approx_count_distinct(col("division"), 0.15).alias("divisionDistinct"))

This option uses the largest rsd value (0.15), which means it prioritizes speed over accuracy. the smaller the rsd, the more accurate the result, but the longer it might take to compute. Conversely, a larger rsd value provides a faster result with less accuracy.
upvoted 1 times

carlosmps 1 year ago
But your answer contradicts the question, they only ask you for the fastest way, while the error value is closer to zero, then it will take more time and resources. 0.15>0.01, that means that option C will be faster, it will have more errors, but it will be the fastest.
upvoted 1 times

thanab 1 year, 9 months ago
C
The code block that will most quickly return an approximation for the number of distinct values in column `division` in DataFrame `storesDF` is **C**, `storesDF.agg(approx_count_distinct(col("division"), 0.15).alias("divisionDistinct"))`. The `approx_count_distinct` function can be used to quickly estimate the number of distinct values in a column by using a probabilistic data structure. The second parameter of the `approx_count_distinct` function specifies the maximum estimation error allowed, with a smaller value resulting in a more accurate but slower estimation. In this case, an error of 0.15 is specified, which will result in a faster but less accurate estimation than the other options.
upvoted 4 times

cookiemonster42 1 year, 11 months ago
Selected Answer: C
C - the less accurate the calculation, the faster it is
upvoted 4 times

singh100 1 year, 11 months ago
A. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.functions.approx_count_distinct.html
upvoted 3 times

SonicBoom10C9 2 years, 1 month ago
Selected Answer: C
While not an option I would use, the question says most quickly (relatively), and this will be the fastest. Note that a 15% error is too high.
upvoted 3 times

TC007 2 years, 2 months ago
Selected Answer: C
The higher the relative error parameter, the less accurate and faster. The lower the relative error parameter, the more accurate and slower.
upvoted 2 times

Question #32 *Topic 1*

The code block shown below contains an error. The code block is intended to return a new DataFrame with the mean of column sqft from DataFrame storesDF in column sqftMean. Identify the error.
Code block:
storesDF.agg(mean("sqft").alias("sqftMean"))

A. The argument to the mean() operation should be a Column abject rather than a string column name.

B. The argument to the mean() operation should not be quoted.

C. The mean() operation is not a standalone function – it's a method of the Column object.

D. The agg() operation is not appropriate here – the withColumn() operation should be used instead.

E. The only way to compute a mean of a column is with the mean() method from a DataFrame.

Suggested Answer: *A*

Community vote distribution

4be8126 Highly Voted 2 years, 2 months ago

Selected Answer: E

The code block shown is correct and should return a new DataFrame with the mean of column sqft from DataFrame storesDF in column sqftMean. Therefore, the answer is E - none of the options identify a valid error in the code block.

Here's an explanation for each option:

A. The argument to the mean() operation can be either a Column object or a string column name, so there is no error in using a string column name in this case.

E. This option is incorrect because the code block shown is a valid way to compute the mean of a column using PySpark. Another way to compute the mean of a column is with the mean() method from a DataFrame, but that doesn't mean the code block shown is invalid.

upvoted 7 times

newusername 1 year, 7 months ago

wrong! A

upvoted 3 times

NirajBhise Most Recent 1 month, 2 weeks ago

Selected Answer: A

The function mean() is part of pyspark.sql.functions, and it expects a Column object, not a string.

upvoted 1 times

sofiess 8 months, 2 weeks ago

The mean() function expects a Column object as an argument, which can be created using col("sqft"). Simply passing the column name as a string will result in an error.

upvoted 2 times

DanYanez 8 months, 3 weeks ago

The correct answer is A. The argument to the mean() operation should be a Column object rather than a string column name.

In Spark DataFrames, the mean() function takes a Column object as its argument, not a string column name. To create a Column object from a string column name, you can use the col() function.

upvoted 1 times

ajayrtk 1 year, 3 months ago

The error in the code is A. The argument to the mean() operation should be a Column object rather than a string column name.
In the provided code block, "sqft" is passed as a string column name to the mean() function. However, the correct approach is to use a Column object. This can be achieved by referencing the column using the storesDF DataFrame and the col() function. Here's the corrected code:
storesDF.agg(mean(col("sqft")).alias("sqftMean"))

upvoted 2 times

azurearch 1 year, 3 months ago

from pyspark.sql.functions import col, mean

students =[
{'rollno':'001','name':'sravan','sqft':23, 'height':5.79,'weight':67,'address':'guntur'},
{'rollno':'002','name':'ojaswi','sqft':16, 'height':3.79,'weight':34,'address':'hyd'}]
storesDF = spark.createDataFrame(students)
storesDF.agg(mean('sqft').alias('sqftMean')).show()

this works as well! not sure which one is wrong then

upvoted 3 times

azure_bimonster 1 year, 4 months ago

Selected Answer: A

A is most like correct here

upvoted 2 times

Saurabh_prep 1 year, 6 months ago

Selected Answer: A

A) should be the one considering databricks practice pdf. mean() function should take col object as input.

upvoted 1 times

outwalker 1 year, 7 months ago

it appears that there might be some flexibility in how the mean function can be used with either a string column name or a col() function. However, the most accurate and recommended approach is to use the col() function to create a Column object explicitly.

With this in mind, the best choice is:

A. The argument to the mean() operation should be a Column object rather than a string column name. The mean function takes a Column object as an argument, not a string column name. To fix the error, the code block should be rewritten as storesDF.agg(mean(col("sqft")).alias("sqftMean")), where the col function is used to create a Column object from the string column name "sqft".

While there might be situations where using a string column name works, following the standard practice of creating a Column object with col() ensures compatibility and clarity in code.

upvoted 1 times

juliom6 1 year, 8 months ago

Selected Answer: A

Correct answer is A:

from pyspark.sql.functions import col, mean

students =[
{'rollno':'001','name':'sravan','sqft':23, 'height':5.79,'weight':67,'address':'guntur'},
{'rollno':'002','name':'ojaswi','sqft':16, 'height':3.79,'weight':34,'address':'hyd'}]
storesDF = spark.createDataFrame(students)
storesDF.agg(mean(col('sqft')).alias('sqftMean')).show()

upvoted 3 times

juadaves 1 year, 8 months ago

D
withColumn() for new calculated column.

upvoted 1 times

thanab 1 year, 9 months ago

A.
A
The error in the code block is **A**, the argument to the `mean` operation should be a Column object rather than a string column name. The `mean` function takes a Column object as an argument, not a string column name. To fix the error, the code block should be rewritten as `storesDF.agg(mean(col("sqft")).alias("sqftMean"))`, where the `col` function is used to create a Column object from the string column name `"sqft"`.

Here is the correct code
storesDF.agg(mean(col("sqft")).alias("sqftMean"))
upvoted 2 times

juadaves 1 year, 8 months ago
storesDF.agg(mean("Value").alias("sqftMean")).show() it works
upvoted 1 times

halouanne 1 year, 10 months ago
The correct answer is:

B. The argument to the mean() operation should not be quoted.

In the context of Apache Spark, the mean function takes a column name as its argument. Therefore, you would write it without quotes. The corrected code line would look something like this:
upvoted 1 times

cookiemonster42 1 year, 10 months ago
Selected Answer: A
There's a similar question in the official Databricks samples and the right answer there is:
Code block:
storesDF.__1__(__2__(__3__).alias("sqftMean"))
A.
1. agg
2. mean
3. col("sqft")

If we stick to this logic, the answer is A.
upvoted 3 times

zozoshanky 1 year, 11 months ago
df.agg(mean("amountpaid").alias("amountpaid")).show()
df.agg(mean(col("amountpaid")).alias("sqftMean")).show(). Both produces the result
upvoted 1 times

Mohitsain 2 years ago
Selected Answer: D
agg is not required here.
upvoted 3 times

Question #33 *Topic 1*

Which of the following operations can be used to return the number of rows in a DataFrame?

A. DataFrame.numberOfRows()

B. DataFrame.n()

C. DataFrame.sum()

D. DataFrame.count()

E. DataFrame.countDistinct()

Suggested Answer: *D*

Community vote distribution

D (100%)

jds0 11 months, 1 week ago

Selected Answer: D

Option D is correct: DataFrame.count()

upvoted 1 times

outwalker 1 year, 7 months ago

The operation that can be used to return the number of rows in a DataFrame is:

D. DataFrame.count()

The count() method in Spark DataFrame returns the number of rows in the DataFrame, and it is the standard way to determine the row count. Options A, B, C, and E are not valid methods for counting the number of rows in a DataFrame.

upvoted 2 times

Question #34 *Topic 1*

Which of the following operations returns a GroupedData object?

A. DataFrame.GroupBy()

B. DataFrame.cubed()

C. DataFrame.group()

D. DataFrame.groupBy()

E. DataFrame.grouping_id()

Suggested Answer: *D*

Community vote distribution

D (100%)

azure_bimonster 10 months, 3 weeks ago

Selected Answer: D

.groupBy() is correct one

upvoted 2 times

outwalker 1 year, 1 month ago

D. DataFrame.groupBy()

The groupBy() method is used to group the DataFrame based on one or more columns, and it returns a GroupedData object, which can then be used to perform various aggregation operations on the grouped data. Options A, B, C, and E do not return a GroupedData object.

upvoted 2 times

Question #35 *Topic 1*

Which of the following code blocks returns a collection of summary statistics for all columns in DataFrame storesDF?

A. storesDF.summary("mean")

B. storesDF.describe(all = True)

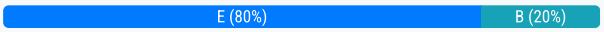
C. storesDF.describe("all")

D. storesDF.summary("all")

E. storesDF.describe()

Suggested Answer: *E*

Community vote distribution



NirajBhise 1 month, 2 weeks ago

Selected Answer: E

Column names or list of names is optional. If no columns specified then the function works on all columns.

upvoted 1 times

jds0 11 months, 1 week ago

Selected Answer: E

E is the right option.

See code below with Spark 3.5.1

```
# Summary statistics of a DataFrame
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.errors import PySparkTypeError

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 43161),
(1, 51200),
(2, None),
(3, 78367),
(4, None),
]
storesDF = spark.createDataFrame(data, ["storeID", "sqft"])

try:
storesDF.summary("mean").show()
except Exception as e:
print(e)

try:
storesDF.describe(all = True).show()
except Exception as e:
print(e)

try:
storesDF.describe("all").show()
except Exception as e:
print(e)
```

```
try:
storesDF.summary("all").show()
except Exception as e:
print(e)

try:
storesDF.describe().show()
except Exception as e:
print(e)
```

upvoted 2 times

dbdantas 1 year, 2 months ago

Selected Answer: E

E is the correct one

upvoted 1 times

azure_bimonster 1 year, 4 months ago

Selected Answer: E

E would be correct here

upvoted 1 times

mahmoud_salah30 1 year, 6 months ago

tested e is the right answer

upvoted 2 times

souha_axa 1 year, 10 months ago

E is the correct answer

upvoted 1 times

cookiemonster42 1 year, 11 months ago

Selected Answer: E

check the documentation, mates. both methods receive names of columns as arguments, so E is correct!

upvoted 1 times

zozoshanky 1 year, 11 months ago

E is correct, it's giving the output.

upvoted 2 times

zozoshanky 1 year, 11 months ago

B is correct. On running the last option it gives error.

TypeError: describe() got an unexpected keyword argument 'all'

upvoted 1 times

cookiemonster42 1 year, 11 months ago

checked it, it gave me the right result, so E is the one

upvoted 3 times

4be8126 2 years, 2 months ago

Selected Answer: B

The answer is B.

Explanation: The describe() method in DataFrame returns a DataFrame with summary statistics for all numeric columns in the input DataFrame. By default, only the count, mean, standard deviation, minimum, and maximum values are returned, but additional statistics can be specified with the percentiles parameter. Setting the all parameter to True will include non-numeric columns in the output as well. Therefore, option B is the correct answer.

Option A is not correct, as the summary() method only returns summary statistics for the specified column(s) and is not a valid option for returning summary statistics for all columns in the DataFrame.

Option C is not correct, as the describe() method does not have an "all" option.

Option D is also not correct, as the summary() method only returns summary statistics for the specified column(s) and does not have an "all" option.

Option E is not incorrect, but it does not specify whether to include non-numeric columns in the output. Therefore, option B is a better answer.
upvoted 1 times

ZSun 2 years ago
Did you really try this in pyspark, or look up the document?
TypeError: describe() got an unexpected keyword argument 'all'
upvoted 6 times

8605246 2 years ago
describe() is correct
upvoted 5 times

Deuterium 1 year, 11 months ago
Is you answer from Chat GPT ?
upvoted 1 times

cookiemonster42 1 year, 11 months ago
even chat gpt says E is the correct one :)
upvoted 3 times

juadaves 1 year, 8 months ago
TypeError Traceback (most recent call last)
<ipython-input-34-5077330dead7> in <cell line: 1>()
----> 1 storesDF.describe(all = True)

TypeError: DataFrame.describe() got an unexpected keyword argument 'all'
upvoted 1 times

Question #36 *Topic 1*

Which of the following code blocks fails to return a DataFrame reverse sorted alphabetically based on column division?

A. storesDF.orderBy("division", ascending – False)

B. storesDF.orderBy(["division"], ascending = [0])

C. storesDF.orderBy(col("division").asc())

D. storesDF.sort("division", ascending – False)

E. storesDF.sort(desc("division"))

Suggested Answer: *C*

Community vote distribution

C (100%)

58470e1 7 months, 2 weeks ago

Selected Answer: E

doesn't reverse sorted alphabetically mean descending????

upvoted 2 times

jds0 11 months, 1 week ago

Selected Answer: C

C is the right answer.

See code below with Spark 3.5.1

```
# Sort a DataFrame by a column in reverse alphabetical order
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, asc, desc
from pyspark.errors import PySparkTypeError

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 43161, "A"),
(1, 51200, "A"),
(2, None, "B"),
(3, 78367, "B"),
(4, None, "C"),
]
storesDF = spark.createDataFrame(data, ["storeID", "sqft", "division"])

storesDF.orderBy("division", ascending = False).show()
storesDF.orderBy(["division"], ascending = [0]).show()
storesDF.orderBy(col("division").asc()).show()
storesDF.sort("division", ascending = False).show()
storesDF.sort(desc("division")).show()
```

upvoted 1 times

dbdantas 1 year, 2 months ago

Selected Answer: C

C is the correct answer

upvoted 2 times

azurearch 1 year, 3 months ago

C is the right answer because it returns the dataframe in ascending order.

upvoted 2 times

mahmoud_salah30 1 year, 6 months ago

b i tesetd ut
upvoted 1 times

vishnuas 1 year, 6 months ago
C. It is the only option "not returning" the dataframe in descending(reverse) order.
All other formats are returning the descending order.
In Oprtion E, if we import the desc function,. it will not throw error and will return the dataframe in descending order.
upvoted 1 times

outwalker 1 year, 7 months ago
E. storesDF.sort(desc("division"))

Option E correctly uses the desc function to specify the descending order for sorting. Thank you for providing additional information and clarification.
upvoted 1 times

singh100 1 year, 11 months ago
Option A and D is giving errors. ~ cannot be used in ascending. Right way is to use ascending=False. Most relevant option is C which is sorting the data in ascending order , Option A, D have some typos it should be = instead of ~.
upvoted 1 times

singh100 1 year, 11 months ago
C is the answer. Only C will make the data in ascending order. Tested the code.
upvoted 3 times

zozoshanky 1 year, 11 months ago
E is right
upvoted 3 times

ItsAB 1 year, 11 months ago
It's C: storesDF.orderBy(col("division").asc()) => storesDF.orderBy(col("division").desc())
upvoted 1 times

lakhan0309 1 year, 11 months ago
Option E is right answer
upvoted 2 times

Question #37 *Topic 1*

Which of the following code blocks returns a 15 percent sample of rows from DataFrame storesDF without replacement?

A. storesDF.sample(fraction = 0.10)

B. storesDF.sampleBy(fraction = 0.15)

C. storesDF.sample(True, fraction = 0.10)

D. storesDF.sample()

E. storesDF.sample(fraction = 0.15)

Suggested Answer: *E*

Community vote distribution

E (100%)

4be8126 8 months, 1 week ago

Selected Answer: E

The answer is E.

Option A returns a 10% sample, not a 15% sample as requested.

Option B is incorrect because sampleBy() is used to perform stratified sampling based on a column's values.

Option C is incorrect because the first argument should be set to False to prevent sampling with replacement.

Option D is incorrect because the sample() method without arguments will return a 50% sample of the DataFrame.

Option E is the correct answer as it returns a sample of 15% of the DataFrame without replacement.

upvoted 3 times

Question #38 *Topic 1*

Which of the following code blocks returns all the rows from DataFrame storesDF?

A. storesDF.head()

B. storesDF.collect()

C. storesDF.count()

D. storesDF.take()

E. storesDF.show()

Suggested Answer: *B*

Community vote distribution

B (100%)

4be8126 Highly Voted 1 year, 8 months ago

Selected Answer: B

Answer: B

Explanation:

head() returns the first n rows of the DataFrame. By default, it returns the first 5 rows.
collect() returns an array of Row objects that represent the entire DataFrame.
count() returns the number of rows in the DataFrame.
take(n) returns the first n rows of the DataFrame as an array of Row objects.
show() prints the first 20 rows of the DataFrame in a tabular form.
Only collect() returns all the rows from the DataFrame.

upvoted 6 times

mahmoud_salah30 Most Recent 1 year ago

b correct

upvoted 1 times

Question #39 *Topic 1*

Which of the following code blocks applies the function assessPerformance() to each row of DataFrame storesDF?

A. [assessPerformance(row) for row in storesDF.take(3)]

B. [assessPerformance() for row in storesDF]

C. storesDF.collect().apply(lambda: assessPerformance)

D. [assessPerformance(row) for row in storesDF.collect()]

E. [assessPerformance(row) for row in storesDF]

Suggested Answer: *D*

Community vote distribution

D (100%)

jds0 11 months, 1 week ago

Selected Answer: D

Option D is correct.
See example code below:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 43161, "A"),
(1, 51200, "A"),
(2, None, "B"),
(3, 78367, "B"),
(4, None, "C"),
]
storesDF = spark.createDataFrame(data, ["storeID", "sqft", "division"])

def myFunction(row):
return row[0]

[myFunction(row) for row in storesDF.collect()]
```

upvoted 2 times

ZSun 2 years ago

There are many way to apply a function to dataframe.
1. apply, as shown in option D. but it should be apply(assessPerformance)
2. list comprehension: for row in df.collect()
3. foreach
4. map, but for RDD majorly

upvoted 1 times

4be8126 2 years, 2 months ago

Selected Answer: D

The correct answer is D.

Explanation:

Option A uses the take() method to extract three rows from the DataFrame, but it applies the assessPerformance() function to each row outside of the DataFrame context.

Option B attempts to apply the assessPerformance() function to each row, but it doesn't reference the row object in any way.

Option C tries to apply the assessPerformance() function to the entire DataFrame but does so using an incorrect syntax.

Option D correctly applies the assessPerformance() function to each row of the DataFrame using a list comprehension over the result of the collect() method.

Option E is similar to D, but it will iterate over rows individually instead of using the collect() method to retrieve all rows at once. While this is still a valid approach, it may be less efficient.

upvoted 2 times

Question #40 *Topic 1*

The code block shown below contains an error. The code block is intended to print the schema of DataFrame storesDF. Identify the error.
Code block:
storesDF.printSchema

A. There is no printSchema member of DataFrame – schema and the print() function should be used instead.

B. The entire line needs to be a string – it should be wrapped by str().

C. There is no printSchema member of DataFrame – the getSchema() operation should be used instead.

D. There is no printSchema member of DataFrame – the schema() operation should be used instead.

E. The printSchema member of DataFrame is an operation and needs to be followed by parentheses.

Suggested Answer: *E*

Community vote distribution

jds0 11 months, 1 week ago

Selected Answer: E

Option E is correct.
Code example below:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, asc, desc
from pyspark.errors import PySparkTypeError

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 43161, "A"),
(1, 51200, "A"),
(2, None, "B"),
(3, 78367, "B"),
(4, None, "C"),
]
storesDF = spark.createDataFrame(data, ["storeID", "sqft", "division"])

storesDF.printSchema()
# root
# |-- storeID: long (nullable = true)
# |-- sqft: long (nullable = true)
# |-- division: string (nullable = true)
```

upvoted 1 times

4be8126 2 years, 2 months ago

Selected Answer: E

E. The printSchema member of DataFrame is an operation and needs to be followed by parentheses.

The correct code block should be storesDF.printSchema() with parentheses to indicate that it's a method call.

upvoted 3 times

Question #41 *Topic 1*

The code block shown below should create and register a SQL UDF named "ASSESS_PERFORMANCE" using the Python function assessPerformance() and apply it to column customerSatisfaction in table stores. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.
Code block:
spark._1_._2_(_3_, _4_)
spark.sql("SELECT customerSatisfaction, _5_(customerSatisfaction) AS result FROM stores")

A. 1. udf
2. register
3. "ASSESS_PERFORMANCE"
4. assessPerformance
5. ASSESS_PERFORMANCE

B. 1. udf
2. register
3. assessPerformance
4. "ASSESS_PERFORMANCE"
5. "ASSESS_PERFORMANCE"

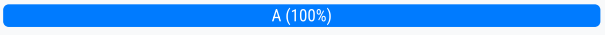
C. 1. udf
2. register
3."ASSESS_PERFORMANCE"
4. assessPerformance
5. "ASSESS_PERFORMANCE"

D. 1. register
2. udf
3. "ASSESS_PERFORMANCE"
4. assessPerformance
5. "ASSESS_PERFORMANCE"

E. 1. udf
2. register
3. ASSESS_PERFORMANCE
4. assessPerformance
5. ASSESS_PERFORMANCE

Suggested Answer: *A*

Community vote distribution



jds0 11 months, 1 week ago

Selected Answer: A

Answer: A
See code example below with Spark 3.5.1:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 43161, "A"),
(1, 51200, "A"),
(3, 78367, "B"),
]
storesDF = spark.createDataFrame(data, ["storeID", "sqft", "division"])
```

```
def assess_performance(x):
return "Large" if x > 50000 else "Small"

spark.udf.register("ASSESS_PERFORMANCE", assess_performance, "STRING")

storesDF.createOrReplaceTempView("stores")

df = spark.sql("SELECT StoreID, ASSESS_PERFORMANCE(sqft) AS performance FROM stores")
df.show()
```

upvoted 2 times

azurearch 1 year, 3 months ago

```
def assessperformance():
return 'Good'

spark.udf.register("assessperformance",assessperformance)
df = spark.sql("SELECT assessperformance()")
df.show()
```

A

upvoted 2 times

4be8126 2 years, 2 months ago

Selected Answer: A

Answer: A

Explanation:

udf: create a user-defined function (UDF) in PySpark
register: register the UDF with Spark so it can be used in SQL queries
"ASSESS_PERFORMANCE": name the UDF "ASSESS_PERFORMANCE"
assessPerformance: specify the Python function to use for the UDF
ASSESS_PERFORMANCE: use the registered UDF in the SQL query to apply the assessPerformance() function to the customerSatisfaction column.

upvoted 2 times

Question #42 *Topic 1*

The code block shown below contains an error. The code block is intended to create a Python UDF assessPerformanceUDF() using the integer-returning Python function assessPerformance() and apply it to column customerSatisfaction in DataFrame storesDF. Identify the error.
Code block:
assessPerformanceUDF – udf(assessPerformance)
storesDF.withColumn("result", assessPerformanceUDF(col("customerSatisfaction")))

A. The assessPerformance() operation is not properly registered as a UDF.

B. The withColumn() operation is not appropriate here – UDFs should be applied by iterating over rows instead.

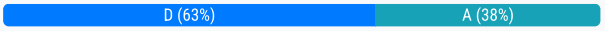
C. UDFs can only be applied vie SQL and not through the DataFrame API.

D. The return type of the assessPerformanceUDF() is not specified in the udf() operation.

E. The assessPerformance() operation should be used on column customerSatisfaction rather than the assessPerformanceUDF() operation.

Suggested Answer: *A*

Community vote distribution



ZSun Highly Voted 2 years ago
The right answer is D.
pyspark.sql.functions.udf(f=None, returnType=StringType)
The default return type is string, but this question requires integer returning.
so it should be D. "The return type of the assessPerformanceUDF() is not specified in the udf() operation."
upvoted 11 times

jds0 11 months, 1 week ago
Good explanation for Answer being D. Thank you!
upvoted 2 times

jds0 Most Recent 11 months, 1 week ago
Selected Answer: D
D is the right answer as otherwise the return type is the default StringType().
Test code below:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf
from pyspark.sql.types import IntegerType

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 3, "A"),
(1, 1, "A"),
(2, 2, "A"),
]
storesDF = spark.createDataFrame(data, ["storeID", "customerSatisfaction", "division"])

def assessPerformance(x):
return 1 if x > 3 else 0

print("IntegerType()")
assessPerformanceUDF = udf(assessPerformance, IntegerType())
df = storesDF.withColumn("result", assessPerformanceUDF(col("customerSatisfaction")))
df.printSchema()

print("Default")
```

assessPerformanceUDF = udf(assessPerformance)
df = storesDF.withColumn("result", assessPerformanceUDF(col("customerSatisfaction")))
df.printSchema()
upvoted 3 times

Raheel_te 1 year ago
correct answer is D
upvoted 1 times

juliom6 1 year, 8 months ago
Selected Answer: D
It is necessary to inform the return type as IntegerType().

from pyspark.sql.functions import udf, col
from pyspark.sql.types import IntegerType

storesDF = spark.createDataFrame([('1', '123'), ('2', '234')], ['id', 'customerSatisfaction'])
assessPerformance = lambda x: int(x)

assessPerformanceUDF = udf(assessPerformance, IntegerType())
storesDF.withColumn('result', assessPerformanceUDF(col('customerSatisfaction'))).printSchema()
upvoted 1 times

Singh_Sumit 1 year, 9 months ago
| 1. When `f` is a Python function:
|
| `returnType` defaults to string type and can be optionally specified. The produced
| object must match the specified type. In this case, this API works as if
| `register(name, f, returnType=StringType())`.
upvoted 2 times

thanab 1 year, 9 months ago
Selected Answer: D
The error in the code block is that the return type of the assessPerformanceUDF() is not specified in the udf() operation. In PySpark, when you register a Python function as a UDF, you should also specify the return type. This is important because Spark SQL needs to understand the return type to properly handle the UDF. Therefore, the correct answer is:
upvoted 2 times

cookiemonster42 1 year, 11 months ago
Selected Answer: D
if they mean that - is =, then we need a second parameter, the output type. so, D is the answe
upvoted 1 times

Deuterium 1 year, 11 months ago
Right answer is D, return type has to be specified into udf() or it will return StringType by default, the code should be :
function_UDF = udf(function, returnType=IntegerType())
upvoted 2 times

4be8126 2 years, 2 months ago
Selected Answer: A
The error in the code block is A. The function assessPerformance() needs to be passed as a parameter to the udf() operation in order to create a UDF from it. The correct code block should be:

assessPerformanceUDF = udf(assessPerformance)
storesDF.withColumn("result", assessPerformanceUDF(col(
upvoted 3 times

ZSun 2 years ago
what is the difference between your code and question itsefl?
assessPerformanceUDF – udf(assessPerformance)
assessPerformanceUDF = udf(assessPerformance)
changing "-" to "="?
upvoted 3 times

Question #43 *Topic 1*

The code block shown below contains an error. The code block is intended to use SQL to return a new DataFrame containing column storeId and column managerName from a table created from DataFrame storesDF. Identify the error.
Code block:
storesDF.createOrReplaceTempView("stores")
storesDF.sql("SELECT storeId, managerName FROM stores")

A. The createOrReplaceTempView() operation does not make a Dataframe accessible via SQL.

B. The sql() operation should be accessed via the spark variable rather than DataFrame storesDF.

C. There is the sql() operation in DataFrame storesDF. The operation query() should be used instead.

D. This cannot be accomplished using SQL – the DataFrame API should be used instead.

E. The createOrReplaceTempView() operation should be accessed via the spark variable rather than DataFrame storesDF.

Suggested Answer: *B*

Community vote distribution

B (100%)

jds0 11 months, 1 week ago

Selected Answer: B

B is correct. 'storeDF' has not attribute or method `sql`
Test code below:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [
(0, 3, "John"),
(1, 1, "Jane"),
(2, 2, "Jack"),
]
storesDF = spark.createDataFrame(data, ["storeID", "customerSatisfaction", "managerName"])

storesDF.createOrReplaceTempView("stores")
try:
storesDF.sql("SELECT storeId, managerName FROM stores")
except AttributeError as e:
print(e)
finally:
spark.sql("SELECT storeId, managerName FROM stores").show()
```

upvoted 2 times

juliom6 1 year, 8 months ago

Selected Answer: B

B is correct:

```
storesDF = spark.createDataFrame([('1', 'juan'), ('2', 'perez')], ['storeId', 'managerName'])
storesDF.createOrReplaceTempView("stores")
spark.sql("SELECT storeId, managerName FROM stores").show()
```

upvoted 2 times

4be8126 2 years, 2 months ago

Selected Answer: B

Option B is correct because the sql() function is not a method of a DataFrame object. It is actually a method of the SparkSession object spark. Therefore, the correct way to execute a SQL statement using Spark SQL is to call sql() on the SparkSession object as follows:

spark.sql("SELECT storeId, managerName FROM stores")

In the code block provided in the question, sql() is called on a DataFrame object, which will result in a DataFrame object without executing the SQL statement. Therefore, option B correctly identifies the error in the code block.

upvoted 2 times

Question #44 *Topic 1*

The code block shown below should create a single-column DataFrame from Python list years which is made up of integers. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.
Code block:
_1_._2_(_3_, _4_)

A. 1. spark
2. createDataFrame
3. years
4. IntegerType

B. 1. DataFrame
2. create
3. [years]
4. IntegerType

C. 1. spark
2. createDataFrame
3. [years]
4. IntegertType

D. 1. spark
2. createDataFrame
3. [years]
4. IntegertType()

E. 1. spark
2. createDataFrame
3. years
4. IntegertType()

Suggested Answer: *D*

Community vote distribution

E (100%)

peekaboo15 Highly Voted 2 years, 2 months ago

The answer should be E because Year is already a python list.

upvoted 10 times

zic00 Most Recent 10 months, 1 week ago

Selected Answer: E

it uses spark.createDataFrame correctly with the Python list years and the appropriate data type IntegerType(). All other options have errors either in syntax or the use of PySpark methods and types.

upvoted 1 times

jds0 11 months, 1 week ago

Selected Answer: E

E is the right answer
See code below:

```
# Create a DataFrame from a list of integers
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType

spark = SparkSession.builder.appName("MyApp").getOrCreate()

years = [2017, 2018, 2019]
df = spark.createDataFrame(years, IntegerType())
```

df.show()
df.printSchema()
upvoted 4 times

znets 1 year, 4 months ago

Selected Answer: E

E is the most suitable, but it also contains an error.

In PySpark, the correct class name for the integer data type is IntegerType (not "IntegertType").
upvoted 1 times

mahmoud_salah30 1 year, 6 months ago

e is the right
upvoted 1 times

juliom6 1 year, 8 months ago

Selected Answer: E

E is correct:

```
from pyspark.sql.types import IntegerType
years = [2023, 2024]
print(type(years))
storesDF = spark.createDataFrame(years, IntegerType())
storesDF.show()

<class 'list'>
+-----+
|value|
+-----+
| 2023|
| 2024|
+-----+
```
upvoted 1 times

juadaves 1 year, 8 months ago

D

```
from pyspark.sql.types import IntegerType
spark.createDataFrame([1991,2023],IntegerType()).show()

+-----+
|value|
+-----+
| 1991|
| 2023|
+-----+
```
upvoted 1 times

carlosmps 1 year ago

it's E. years is already a list
upvoted 2 times

thanab 1 year, 9 months ago

Selected Answer: E

1. spark
2. createDataFrame
3. years
4. IntegertType()
upvoted 1 times

cookiemonster42 1 year, 11 months ago

Selected Answer: E

if years is variable, it works, just tested it: years = [1, 3, 4, 5 , 9]
df7 = spark.createDataFrame(years, IntegerType())
df7.show()

this works as well: df7 = spark.createDataFrame([1, 3, 4, 5 , 9], IntegerType())
df7.show()

this won't work:
df7 = spark.createDataFrame([years], IntegerType())
df7.show()

so, the answer is E
upvoted 1 times

singh100 1 year, 11 months ago
E. D is giving an error .
upvoted 1 times

zozoshanky 1 year, 11 months ago
D throws a big error.
/usr/local/spark/python/pyspark/sql/types.py in verify_acceptable_types(obj)
1291 # subclass of them can not be fromInternal in JVM
1292 if type(obj) not in _acceptable_types[_type]:
-> 1293 raise TypeError(new_msg("%s can not accept object %r in type %s"
1294 % (dataType, obj, type(obj))))
1295

TypeError: field value: IntegerType can not accept object [1, 2, 3, 4, 5] in type <class 'list'>

E is correct answer

from pyspark.sql.types import IntegerType
a = [1,2,3,4,5]
spark.createDataFrame(a, IntegerType()).show()
upvoted 1 times

Indiee 2 years, 2 months ago
Two responses
1. D is an error. E will split the array into rows
2. spark.createDataFrame([arraryVar_name],ArrayType(IntegerType())) will store the whole array as a row
upvoted 2 times

Indiee 2 years, 2 months ago
Agreed
upvoted 1 times

Question #45 *Topic 1*

The code block shown below contains an error. The code block is intended to cache DataFrame storesDF only in Spark's memory and then return the number of rows in the cached DataFrame. Identify the error.
Code block:
storesDF.cache().count()

A. The cache() operation caches DataFrames at the MEMORY_AND_DISK level by default – the storage level must be specified to MEMORY_ONLY as an argument to cache().

B. The cache() operation caches DataFrames at the MEMORY_AND_DISK level by default – the storage level must be set via storesDF.storageLevel prior to calling cache().

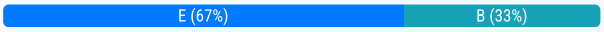
C. The storesDF DataFrame has not been checkpointed – it must have a checkpoint in order to be cached.

D. DataFrames themselves cannot be cached – DataFrame storesDF must be cached as a table.

E. The cache() operation can only cache DataFrames at the MEMORY_AND_DISK level (the default) – persist() should be used instead.

Suggested Answer: *B*

Community vote distribution



singh100 Highly Voted 1 year, 11 months ago
E is correct. You cannot set StorageLevel Memory_only with cache(), if memory available then it keeps everything into memory else it will spill to disk. To keep everything into Memory you need to use Persist() with Storage Level Memory only.
upvoted 5 times

Ryan2025 Most Recent 1 month, 1 week ago
Selected Answer: A
The cache() operation caches DataFrames at the MEMORY_AND_DISK level by default – the storage level must be specified to MEMORY_ONLY as an argument to cache().

This is correct with a slight nuance:
.cache() cannot accept arguments, so if you want to specify MEMORY_ONLY, you must use .persist(StorageLevel.MEMORY_ONLY).
upvoted 1 times

sofiess 8 months, 2 weeks ago
A because the default behavior of cache() is MEMORY_AND_DISK, and if you want MEMORY_ONLY, you must specify it explicitly
upvoted 1 times

azurearch 1 year, 3 months ago
E is wrong. The cache() operation can only cache DataFrames at the MEMORY_AND_DISK level (the default)
note the use of 'only' here, cache can also store in disk if required.

B is also wrong, there is no condition to set storagelevel prior to calling cache()

correct answer is A.
upvoted 1 times

juliom6 1 year, 8 months ago
Selected Answer: E
E is correct!

from pyspark.sql.types import IntegerType
from pyspark import StorageLevel

storesDF = spark.createDataFrame([2023, 2024], IntegerType())
print(storesDF.persist(StorageLevel.MEMORY_ONLY).storageLevel)
upvoted 1 times

juadaves 1 year, 8 months ago

E
cache() -> 'DataFrame' method of pyspark.sql.dataframe.DataFrame instance
Persists the :class:`DataFrame` with the default storage level (`MEMORY_AND_DISK`).

.. versionadded:: 1.3.0

.. versionchanged:: 3.4.0
Supports Spark Connect.

Notes

The default storage level has changed to `MEMORY_AND_DISK` to match Scala in 2.0.
upvoted 1 times

ItsAB 1 year, 11 months ago

there are two options here: B and E. Who chose B => you can't explicitly set the storage level, it's a read-only property, so the correct answer is E.
upvoted 1 times

Jtic 2 years, 1 month ago

Selected Answer: E

E

B. The cache() operation caches DataFrames at the MEMORY_AND_DISK level by default – the storage level must be set via storesDF.storageLevel prior to calling cache().

This option is incorrect. The storage level does not need to be set via storesDF.storageLevel prior to calling cache(). The cache() operation can be used directly on the DataFrame without explicitly setting the storage level.

E. The cache() operation can only cache DataFrames at the MEMORY_AND_DISK level (the default) – persist() should be used instead.

This option is the correct answer. The error in the code block is that the cache() operation is used instead of persist(). While cache() caches DataFrames at the default MEMORY_AND_DISK level, persist() provides more flexibility by allowing different storage levels to be specified, such as MEMORY_ONLY for caching only in memory. Therefore, persist() should be used instead of cache() to achieve the desired caching behavior.
upvoted 1 times

4be8126 2 years, 1 month ago

Selected Answer: B

B. The cache() operation caches DataFrames at the MEMORY_AND_DISK level by default – the storage level must be set via storesDF.storageLevel prior to calling cache().

The storage level of a DataFrame cache can be specified as an argument to the cache() operation, but if the storage level has not been specified, the default MEMORY_AND_DISK level is used. Therefore, option A is incorrect.

Option C is incorrect because caching and checkpointing are different operations in Spark. Caching stores a DataFrame in memory or on disk, while checkpointing saves a DataFrame to a reliable storage system like HDFS, which is necessary for iterative computations.

Option D is incorrect because DataFrames can be cached in memory or on disk using the cache() operation.

Option E is incorrect because cache() is the recommended method for caching DataFrames in Spark, and it supports caching at all storage levels, including MEMORY_ONLY. The persist() operation can be used to specify a storage level, but cache() is simpler and more commonly used.
upvoted 1 times

ZSun 2 years ago

Wrong explanation. you can call cache() or persist() without set storage level, it will use default Memoery_and_disk.
You clearly misunderstand the question itself. storesDF.cache().count() is a workable code, but fail the requirement. This is the issue.
The question asked "only in memory", that means, if the data size is out of the memory, i do not want to store it in disk, but rather recompute.
Therefore, you need to specifically set the storage level as "MEMORY ONLY".
A is correct
upvoted 3 times

peekaboo15 2 years, 2 months ago

The answer should be E. See this post for reference https://stackoverflow.com/questions/26870537/what-is-the-difference-between-cache-and-persist

upvoted 2 times

4be8126 2 years, 1 month ago

No, option E is incorrect. The cache() method is the appropriate method to cache a DataFrame in Spark's memory, and it can cache DataFrames at the MEMORY_ONLY level if that's what is desired. The persist() method is a more general-purpose method that allows the user to specify other storage levels (such as MEMORY_AND_DISK), but it is not required for this task.

upvoted 1 times

juadaves 1 year, 8 months ago

You should use storesDF.persist(StorageLevel.MEMORY_ONLY).count()

upvoted 1 times

Question #46 *Topic 1*

Which of the following operations can be used to return a new DataFrame from DataFrame storesDF without inducing a shuffle?

A. storesDF.intersect()

B. storesDF.repartition(1)

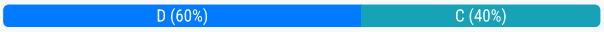
C. storesDF.union()

D. storesDF.coalesce(1)

E. storesDF.rdd.getNumPartitions()

Suggested Answer: *D*

Community vote distribution



mineoolee 6 months, 2 weeks ago

Selected Answer: D

C is not operation

upvoted 1 times

azurearch 9 months, 3 weeks ago

Though Union does not cause a shuffle, you need another dataframe to do union. in this question its limited to storesDF. coalesce(1) is the correct answer, as it does not cause shuffle rather combines multiple partitions into 1, i.e. reducing partitions = no shuffle.

execute storedDF.coalesce(1) and check DAG

upvoted 3 times

Ahlo 10 months, 1 week ago

Answer C : union

Narrow transformation - all transformation logic performed within one partition

Wide transformations - transformation during which is needed shuffle/exchange, distribution of data to other partitions

Union is narrow transaction

upvoted 1 times

newusername 1 year, 1 month ago

Selected Answer: C

union is the only operation from mentioned here that won't do shuffling. And as @ZSun mentioned, do not follow any of the 4be8126 answers, they are all blindly from GPT

upvoted 2 times

issibra 1 year, 4 months ago

C is the correct

coalesce may induce a partial shuffle

upvoted 1 times

ZSun 1 year, 6 months ago

I think this question contains error, it should not be which one without shuffle, it should be which one cause shuffle.

union is a narrow transformation, not causing shuffle.

coalesce simply combine partitions together into one, not shuffle them.

rdd.getNumPartitions just evaluate the number of partition of a dataframe, no shuffle.

even for repartition(1), since there is only one partition in the end, it also not causing shuffle, it simply combine all partition together.

Therefore, it should be A, this is the only one inducing a shuffle.

or, B C D E without inducing a shuffle

upvoted 3 times

ZSun 1 year, 6 months ago

The Answer is C. Union rather than coalesce.

Union is a narrow transformation. unlike wide transformationl, narrow transformation does not require shuffle.

Coalesce is wide transformation, combine multiple partition to smaller number of partition. Don't this process require shuffling partition together?

if you ask ChatGPT, it will tell you what 4be8126 comment.

upvoted 2 times

wlademaro 10 months, 3 weeks ago

The problem with Union answer is that it returns an error if we run it without arg.

upvoted 2 times

ZSun 1 year, 6 months ago

This is incorrect explanation, delete it

upvoted 6 times

4be8126 1 year, 7 months ago

Selected Answer: D

The correct answer is D. coalesce() can be used to return a new DataFrame with a reduced number of partitions, without inducing a shuffle.

A shuffle is an expensive operation that involves the redistribution of data across a cluster, so it's important to minimize its use whenever possible. In this case, repartition() and union() both involve shuffles, while intersect() returns only the common rows between two DataFrames, and rdd.getNumPartitions() returns the number of partitions in the RDD underlying the DataFrame.

upvoted 3 times

Question #47 *Topic 1*

The code block shown below contains an error. The code block is intended to return a new 12-partition DataFrame from the 8-partition DataFrame storesDF by inducing a shuffle. Identify the error.
Code block:
storesDF.coalesce(12)

A. The coalesce() operation cannot guarantee the number of target partitions – the repartition() operation should be used instead.

B. The coalesce() operation does not induce a shuffle and cannot increase the number of partitions – the repartition() operation should be used instead.

C. The coalesce() operation will only work if the DataFrame has been cached to memory – the repartition() operation should be used instead.

D. The coalesce() operation requires a column by which to partition rather than a number of partitions – the repartition() operation should be used instead.

E. The number of resulting partitions, 12, is not achievable for an 8-partition DataFrame.

Suggested Answer: *B*

Community vote distribution

B (100%)

Raju_Bhai 8 months, 3 weeks ago
with version 3.4.0,

df.repartition(12).coalesce(16).rdd.getNumPartitions() returns 12. it doesn't throw error, but only doesn't increase partition either
upvoted 1 times

4be8126 1 year, 1 month ago
Selected Answer: B
The correct answer is B.

The coalesce() operation can decrease the number of partitions but cannot increase the number of partitions. It also does not induce a shuffle, and is therefore more efficient when decreasing the number of partitions.

If the goal is to increase the number of partitions, repartition() should be used instead.
upvoted 3 times

Question #48 *Topic 1*

Which of the following Spark properties is used to configure whether DataFrame partitions that do not meet a minimum size threshold are automatically coalesced into larger partitions during a shuffle?

A. spark.sql.shuffle.partitions

B. spark.sql.autoBroadcastJoinThreshold

C. spark.sql.adaptive.skewJoin.enabled

D. spark.sql.inMemoryColumnarStorage.batchSize

E. spark.sql.adaptive.coalescePartitions.enabled

Suggested Answer: *E*

Community vote distribution

E (100%)

juliom6 7 months, 3 weeks ago

Selected Answer: E

https://spark.apache.org/docs/latest/sql-performance-tuning.html

spark.sql.adaptive.coalescePartitions.enabled: When true and spark.sql.adaptive.enabled is true, Spark will coalesce contiguous shuffle partitions according to the target size (specified by spark.sql.adaptive.advisoryPartitionSizeInBytes), to avoid too many small tasks.

upvoted 1 times

4be8126 1 year, 1 month ago

Selected Answer: E

The answer is E. spark.sql.adaptive.coalescePartitions.enabled is the Spark property used to configure whether DataFrame partitions that do not meet a minimum size threshold are automatically coalesced into larger partitions during a shuffle. When set to true, Spark automatically coalesces partitions that are smaller than the configured minimum size into larger partitions to optimize shuffles.

upvoted 3 times

Question #49 *Topic 1*

The code block shown below contains an error. The code block is intended to return a DataFrame containing a column openDateString, a string representation of Java's SimpleDateFormat. Identify the error.
Note that column openDate is of type integer and represents a date in the UNIX epoch format – the number of seconds since midnight on January 1st, 1970.
An example of Java's SimpleDateFormat is "Sunday, Dec 4, 2008 1:05 PM".
A sample of storesDF is displayed below:

storeId	openDate
0	1100746394
1	1474410343
2	1116610009
3	1180035265
4	1408024997
...	...

Code block:
storesDF.withColumn("openDateString", from_unixtime(col("openDate"), "EEE, MMM d, yyyy h:mm a", TimestampType()))

A. The from_unixtime() operation only accepts two parameters – the TimestampTime() arguments not necessary.

B. The from_unixtime() operation only works if column openDate is of type long rather than integer – column openDate must first be converted.

C. The second argument to from_unixtime() is not correct – it should be a variant of TimestampType() rather than a string.

D. The from_unixtime() operation automatically places the input column in java's SimpleDateFormat – there is no need for a second or third argument.

E. The column openDate must first be converted to a timestamp, and then the Date() function can be used to reformat to java's SimpleDateFormat.

Suggested Answer: *A*

Community vote distribution

juliom6 7 months, 2 weeks ago

Selected Answer: A

A is correct:

```
from pyspark.sql.functions import from_unixtime, col

storesDF = spark.createDataFrame([(0, 1100746394), (1, 1474410343)], ['storeId', 'openDate'])
storesDF = storesDF.withColumn("openDateString", from_unixtime(col("openDate"), "EEE, MMM d, yyyy h:mm a"))
display(storesDF)
```

upvoted 2 times

nicklasbekkevold 10 months, 1 week ago

Selected Answer: A

A is the right answer.

Function signature from the docs:
pyspark.sql.functions.from_unixtime(timestamp, format='uuuu-MM-dd HH:mm:ss')

upvoted 1 times

zozoshanky 11 months ago

A is also right.

upvoted 2 times

Jtic 1 year, 1 month ago

Selected Answer: B

B. The from_unixtime() operation only works if column openDate is of type long rather than integer - column openDate must first be converted.

This option is correct. The code block has an error because the from_unixtime() function expects the column openDate to be of type long, not integer. The column should be cast to long before applying the function.

upvoted 1 times

juliom6 7 months, 2 weeks ago

That not make sense, the code below works perfectly:

```
from pyspark.sql.functions import from_unixtime, col

storesDF = spark.createDataFrame([(0, 1100746394), (1, 1474410343)], ['storeId', 'openDate'])
storesDF = storesDF.withColumn('openDate', col('openDate').cast('integer'))
storesDF = storesDF.withColumn("openDateString", from_unixtime(col("openDate"), "EEE, MMM d, yyyy h:mm a"))
display(storesDF)
```

upvoted 1 times

ZSun 1 year ago

This is completely nonsense about long and integer.
long (or bigint): It is a 64-bit signed integer data type anging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
integer (or int): It is a 32-bit signed integer data ranging from -2,147,483,648 to 2,147,483,647

upvoted 3 times

Question #50 *Topic 1*

Which of the following code blocks returns a DataFrame containing a column dayOfYear, an integer representation of the day of the year from column openDate from DataFrame storesDF?
Note that column openDate is of type integer and represents a date in the UNIX epoch format – the number of seconds since midnight on January 1st, 1970.
A sample of storesDF is displayed below:

storeId	openDate
0	1100746394
1	1474410343
2	1116610009
3	1180035265
4	1408024997
...	...

A. (storesDF.withColumn("openTimestamp", col("openDate").cast("Timestamp"))
. withColumn("dayOfYear", dayofyear(col("openTimestamp"))))

B. storesDF.withColumn("dayOfYear", get dayofyear(col("openDate")))

C. storesDF.withColumn("dayOfYear", dayofyear(col("openDate")))

D. (storesDF.withColumn("openDateFormat", col("openDate").cast("Date"))
. withColumn("dayOfYear", dayofyear(col("openDateFormat"))))

E. storesDF.withColumn("dayOfYear", substr(col("openDate"), 4, 6))

Suggested Answer: *C*

Community vote distribution

A (100%)

juliom6 7 months, 2 weeks ago

Selected Answer: A

A is correct:

from pyspark.sql.functions import col, dayofyear

storesDF = spark.createDataFrame([(0, 1100746394), (1, 1474410343)], ['storeId', 'openDate'])
storesDF = (storesDF.withColumn("openTimestamp", col("openDate").cast("Timestamp")).withColumn("dayOfYear", dayofyear(col("openTimestamp"))))
display(storesDF)

upvoted 1 times

newusername 7 months, 3 weeks ago

Selected Answer: A

A is correct

upvoted 1 times

thanab 9 months, 2 weeks ago

Selected Answer: A

storesDF.withColumn("openTimestamp", col("openDate").cast("Timestamp")).withColumn("dayOfYear", dayofyear(col("openTimestamp")))

upvoted 2 times

singh100 11 months ago

A. dayofyear function in PySpark's functions module expects the column openDate to be of type timestamp rather than long.

upvoted 1 times

4be8126 1 year, 1 month ago

Selected Answer: A

The correct answer is C.

Option A is correct because it casts the openDate column to a timestamp using cast("Timestamp") and then uses the dayofyear function to extract the day of the year from the timestamp.

Option B is incorrect because it contains syntax errors, including the "get" keyword, which is not necessary or valid in this context.

Option C is close, but it does not cast the openDate column to a timestamp, which is necessary to use the dayofyear function.

Option D is incorrect because it converts column "openDate" to a date format, which is unnecessary for extracting the day of the year. Additionally, the dayofyear() function can be applied directly to the "openDate" column.

Option E is incorrect because it uses the substr() function to extract a substring from the "openDate" column, which does not correspond to the day of the year.

upvoted 1 times

peekaboo15 1 year, 2 months ago

The answer should be A. Unixtime should be cast to timestamp first

upvoted 2 times

Question #51 *Topic 1*

The code block shown below contains an error. The code block intended to return a new DataFrame that is the result of an inner join between DataFrame storesDF and DataFrame employeesDF on column storeId. Identify the error.
Code block:
StoresDF.join(employeesDF, "inner", "storeID")

A. The key column storeID needs to be wrapped in the col() operation.

B. The key column storeID needs to be in a list like ["storeID"].

C. The key column storeID needs to be specified in an expression of both DataFrame columns like storesDF.storeId == employeesDF.storeId.

D. There is no DataFrame.join() operation – DataFrame.merge() should be used instead.

E. The column key is the second parameter to join() and the type of join in the third parameter to join() – the second and third arguments should be switched.

Suggested Answer: *E*

Community vote distribution

E (100%)

newusername 7 months, 3 weeks ago

Selected Answer: E

there are diff methods to join dataframes, one of the:
joinedDF = StoresDF.join(employeesDF, "storeId", "inner")
upvoted 1 times

Jtic 1 year, 1 month ago

Selected Answer: E

E

storesDF.join(employeesDF, "storeID", "inner")

The column key is the second parameter to join() and the type of join is in the third parameter to join() – the second and third arguments should be switched
upvoted 3 times

Question #52 *Topic 1*

Which of the following operations can perform an outer join on two DataFrames?

A. DataFrame.crossJoin()

B. Standalone join() function

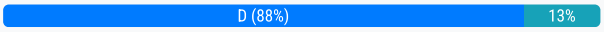
C. DataFrame.outerJoin()

D. DataFrame.join()

E. DataFrame.merge()

Suggested Answer: *D*

Community vote distribution



cookiemonster42 Highly Voted 1 year, 5 months ago

Selected Answer: D

D. result_df = df1.join(df2, on="id", how="outer")

upvoted 6 times

juliom6 Most Recent 1 year, 1 month ago

Selected Answer: D

D is correct.

There is no exists outerJoin() operation in pyspark.

upvoted 2 times

4be8126 1 year, 7 months ago

Selected Answer: C

The correct answer is C - DataFrame.outerJoin(). The outer join operation can be performed by specifying the join type as "outer" when calling the outerJoin() function on a DataFrame. The join() function in Spark only performs an inner join, while the merge() function is not a valid function in Spark SQL. The crossJoin() function performs a Cartesian product between two DataFrames, which is not an outer join.

upvoted 1 times

ZSun 1 year, 6 months ago

There is no outerjoin, bro!

only dataframe.join(how='outer')

upvoted 9 times

Seeker_thunder 1 year, 1 month ago

this guy always post wrong answers, sometime gpts as well. ignore his commnmets

upvoted 1 times

65bd33e 8 months, 1 week ago

Wrong answer, check documentation

upvoted 1 times

Question #53 *Topic 1*

Which of the following pairs of arguments cannot be used in DataFrame.join() to perform an inner join on two DataFrames, named and aliased with "a" and "b" respectively, to specify two key columns?

A. on = [a.column1 == b.column1, a.column2 == b.column2]

B. on = [col("column1"), col("column2")]

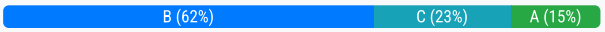
C. on = [col("a.column1") == col("b.column1"), col("a.column2") == col("b.column2")]

D. All of these options can be used to perform an inner join with two key columns.

E. on = ["column1", "column2"]

Suggested Answer: *D*

Community vote distribution



NirajBhise 1 month, 2 weeks ago

Selected Answer: A

The correct answer is: A

This is because the on parameter in DataFrame.join() expects either a string, list of strings, or a single expression. The correct way to specify multiple key columns for an inner join is to use a list of column names or column expressions.

Options B, C, and E are valid ways to specify the key columns for an inner join.

upvoted 1 times

azure_bimonster 10 months, 3 weeks ago

Selected Answer: B

B cannot be used as this seems ambiguous

upvoted 1 times

Gurdel 1 year ago

Selected Answer: B

B throws AnalysisException: [AMBIGUOUS_REFERENCE] Reference `column1` is ambiguous, could be: [`a`.`column1`, `b`.`column1`]

upvoted 1 times

juliom6 1 year, 1 month ago

Selected Answer: B

According to the following code, only response B returns an error. The key concept here is that dataframes must be "named" AND "aliased".

```
from pyspark.sql.functions import col

a = spark.createDataFrame([(1, 2), (3, 4)], ['column1', 'column2'])
b = spark.createDataFrame([(1, 2), (5, 6)], ['column1', 'column2'])

a = a.alias('a')
b = b.alias('b')

df = a.join(b, on = [a.column1 == b.column1, a.column2 == b.column2])
display(df)
# df = a.join(b, on = [col("column1"), col("column2")])
df = a.join(b, on = [col("a.column1") == col("b.column1"), col("a.column2") == col("b.column2")])
display(df)
df = a.join(b, on = ["column1", "column2"])
display(df)
```

upvoted 3 times

newusername 1 year, 1 month ago

Selected Answer: B

100% B

Below code to test:

```
dataA = [Row(column1=1, column2=2), Row(column1=2, column2=4), Row(column1=3, column2=6)]
dfA = spark.createDataFrame(dataA)
```

upvoted 3 times

newusername 1 year, 1 month ago

```
# Sample data for DataFrame 'b'
dataB = [Row(column1=1, column2=2), Row(column1=2, column2=5), Row(column1=3, column2=4)]
dfB = spark.createDataFrame(dataB)

# Alias DataFrames as 'a' and 'b'
a = dfA.alias("a")
b = dfB.alias("b")

a.show()
b.show()


#Option A
joinedDF_A = a.join(b, [a.column1 == b.column1, a.column2 == b.column2])
joinedDF_A.show()


#Option B
#joinedDF_B = a.join(b, [col("column1"), col("column2")])
#joinedDF_B.show()


#Option C
joinedDF_C = a.join(b, [col("a.column1") == col("b.column1"), col("a.column2") == col("b.column2")])
joinedDF_C.show()


#Option E
joinedDF_E = a.join(b, ["column1", "column2"])
joinedDF_E.show()
```

upvoted 3 times

juadaves 1 year, 2 months ago

I tried all of the options and I got 2 errors from:

B

AMBIGUOUS_REFERENCE] Reference `Category` is ambiguous, could be: [`Category`, `Category`]

C:

[UNRESOLVED_COLUMN.WITH_SUGGESTION] A column or function parameter with name `df_1`.`Category` cannot be resolved.

Did you mean one of the following? [`Category`, `Category`, `Truth`, `Truth`, `Value`].;

upvoted 2 times

Ahmadkt 1 year, 2 months ago

it's B, it seems you didn't do the alias

```
a = df1.alias("a")
b = df2.alias("b")
```

upvoted 1 times

Singh_Sumit 1 year, 3 months ago

from pyspark.sql.functions import col
df2.alias('a').join(df3.alias('b'),
[col("a.name") == col("b.name"), col("a.name") == col("b.name")],
'full_outer').select(df2['name'],'height','age').show()
It worked. so every answer is correct.
upvoted 1 times

cookiemonster42 1 year, 4 months ago

Selected Answer: C

should be C as in col() we specify only a column name as a string, not a dataframe
upvoted 3 times

Jtic 1 year, 7 months ago

Selected Answer: A

A. on = [a.column1 == b.column1, a.column2 == b.column2]
This option is valid and can be used to perform an inner join on two key columns. It specifies the key columns using the syntax a.column1 == b.column1 and a.column2 == b.column2.
upvoted 2 times

ZSun 1 year, 6 months ago

I think the question "which one cannot be used to perform inner join", is confusing,
Because only A works, the rest of answer is incorrect.
The question should be "which one can be used"
upvoted 2 times

Question #54 *Topic 1*

The below code block contains a logical error resulting in inefficiency. The code block is intended to efficiently perform a broadcast join of DataFrame storesDF and the much larger DataFrame employeesDF using key column storeId. Identify the logical error.
Code block:
storesDF.join(broadcast(employeesDF), "storeId")

A. The larger DataFrame employeesDF is being broadcasted rather than the smaller DataFrame storesDF.

B. There is never a need to call the broadcast() operation in Apache Spark 3.

C. The entire line of code should be wrapped in broadcast() rather than just DataFrame employeesDF.

D. The broadcast() operation will only perform a broadcast join if the Spark property spark.sql.autoBroadcastJoinThreshold is manually set.

E. Only one of the DataFrames is being broadcasted rather than both of the DataFrames.

Suggested Answer: *A*

Community vote distribution

A (100%)

juliom6 7 months, 2 weeks ago

Selected Answer: A

A si correct:

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.functions.broadcast.html

```
from pyspark.sql import types
from pyspark.sql.functions import broadcast

df = spark.createDataFrame([1, 2, 3, 3, 4], types.IntegerType())
df_small = spark.range(3)
df.join(broadcast(df_small), df.value == df_small.id).show()
```

upvoted 1 times

4be8126 1 year, 1 month ago

Selected Answer: A

The answer is A.

The logical error in the code block is that the larger DataFrame, employeesDF, is being broadcasted instead of the smaller DataFrame, storesDF. This defeats the purpose of a broadcast join, which is to optimize performance by broadcasting the smaller DataFrame to all the worker nodes, avoiding the need to shuffle data over the network.

To perform a broadcast join efficiently, the smaller DataFrame should be broadcasted, which in this case is storesDF. The corrected code should be:

```
broadcast(storesDF).join(employeesDF, "storeId")
```

upvoted 2 times

Question #55 *Topic 1*

The code block shown below contains an error. The code block is intended to return a new DataFrame that is the result of a cross join between DataFrame storesDF and DataFrame employeesDF. Identify the error.
Code block:
storesDF.join(employeesDF, "cross")

A. A cross join is not implemented by the DataFrame.join() operations – the standalone CrossJoin() operation should be used instead.

B. There is no direct cross join in Spark, but it can be implemented by performing an outer join on all columns of both DataFrames.

C. A cross join is not implemented by the DataFrame.join()operation – the DataFrame.crossJoin()operation should be used instead.

D. There is no key column specified – the key column "storeId" should be the second argument.

E. A cross join is not implemented by the DataFrame.join() operations – the standalone join() operation should be used instead.

Suggested Answer: *C*

Community vote distribution

C (55%) D (45%)

mineoolee 6 months, 2 weeks ago

Selected Answer: D

it is wokring
data = [
(0, 2, 1100746394),
(1, 2, 1474410343)
]

df = spark.createDataFrame(
data,
['storeId','a', 'openDate']
)

_data = [
('a', 2, 4444444444),
('c', 2, None),
('b', None, 2222222222)
]

_df = spark.createDataFrame(
_data,
['storeId','a', 'openDate']
)

df.join(_df, 'a', "cross").show()
upvoted 1 times

mineoolee 6 months, 2 weeks ago

also, df.join(_df, '"cross").show() is working
upvoted 1 times

Kalipe 5 months, 3 weeks ago

it's wrong, it doesn't work or you obviously haven't try it
upvoted 1 times

oussa_ama 10 months, 2 weeks ago

Selected Answer: C

Cross Join in PySpark: A cross join (also known as a Cartesian product) returns the Cartesian product of the two DataFrames, meaning every row from the first DataFrame is paired with every row from the second DataFrame. In PySpark, the crossJoin() method is used specifically for this type of join.

upvoted 2 times

65bd33e 10 months, 2 weeks ago

Selected Answer: C

The correct identification of the error is:

C. A cross join is not implemented by the DataFrame.join() operation – the DataFrame.crossJoin() operation should be used instead.

Explanation:

In Spark, to perform a cross join between two DataFrames, you should use the crossJoin() method, not the join() method with the "cross" argument.

upvoted 1 times

Ahlo 1 year, 4 months ago

Correct answer C

```
from pyspark.sql import Row
df = spark.createDataFrame(
[(14, "Tom"), (23, "Alice"), (16, "Bob")], ["age", "name"])
df2 = spark.createDataFrame(
[Row(height=80, name="Tom"), Row(height=85, name="Bob")])
df.crossJoin(df2.select("height")).select("age", "name", "height").show()
```

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.crossJoin.html

upvoted 2 times

azure_bimonster 1 year, 4 months ago

Selected Answer: D

D is the answer here as key is missing. As per syntax, key is needed.

upvoted 2 times

juliom6 1 year, 7 months ago

Selected Answer: C

C is correct.

```
# https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.crossJoin.html

a = spark.createDataFrame([(1, 2), (3, 4)], ['column1', 'column2'])
b = spark.createDataFrame([(5, 6), (7, 8)], ['column3', 'column4'])

df = a.crossJoin(b)
display(df)
```

upvoted 3 times

newusername 1 year, 7 months ago

Selected Answer: D

I know it looks confusing to have key column for cross join, but it ijoin method syntaxis:
https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.join.html

see example below :

```
dataA = [Row(column1=1, column2=2), Row(column1=2, column2=4), Row(column1=3, column2=6)]
dfA = spark.createDataFrame(dataA)

# Sample data for DataFrame 'b'
dataB = [Row(column1=1, column2=2), Row(column1=2, column2=5), Row(column1=3, column2=4)]
dfB = spark.createDataFrame(dataB)

joinedDF = dfA.join(dfB, on=None, how="cross")
joinedDF.show()
```

it is possible to do Cross join this way as well DataFrame.crossJoin() but answer C states that df.join () doesn't do cross, which is wrong.
upvoted 4 times

tmz1 5 months ago
Totally agree. The stament in answer C "A cross join is not implemented by the DataFrame.join()operation" is incorrect. It is implemented and I have tested it. Results below:
products_df = spark.table('products')
orders_df = spark.table('orders')
print(products_df.count()) -> 200
print(orders_df.count()) -> 2140
cross_joined_df = products_df.join(orders_df, None, "cross")
print(cross_joined_df.count()) -> 428000
upvoted 1 times

4be8126 2 years, 1 month ago
Selected Answer: C
C. A cross join is not implemented by the DataFrame.join()operation – the DataFrame.crossJoin()operation should be used instead.
upvoted 2 times

peekaboo15 2 years, 2 months ago
cross join doesn't need a key. Answer is C
upvoted 2 times

4be8126 2 years, 1 month ago
No, the issue is not that the key column is missing. In a cross join, there is no key column to join on. The correct answer is C: a cross join is not implemented by the DataFrame.join() operation – the DataFrame.crossJoin() operation should be used instead.
upvoted 1 times

ronfun 2 years, 2 months ago
Key is missing. Answer is D.
upvoted 4 times

4be8126 2 years, 1 month ago
No, the issue is not that the key column is missing. In a cross join, there is no key column to join on. The correct answer is C: a cross join is not implemented by the DataFrame.join() operation – the DataFrame.crossJoin() operation should be used instead.
upvoted 1 times

ZSun 2 years ago
completely wrong.
join(other, on=None, how=None)
Joins with another DataFrame, using the given join expression.
[source]
Parameters:
other – Right side of the join
on – a string for the join column name, a list of column names, a join expression (Column), or a list of Columns. If on is a string or a list of strings indicating the name of the join column(s), the column(s) must exist on both sides, and this performs an equi-join.
how – str, default inner. Must be one of: inner, cross, outer, full, fullouter, full_outer, left, leftouter, left_outer, right, rightouter, right_outer, semi, leftsemi, left_semi, anti, leftanti and left_anti.
upvoted 2 times

ZSun 2 years ago
you can specify cross in dataframe.join(how = 'cross')
the reason why this code block doesn't work, because the second parameter is on. You need to specify the key column and then use how = 'cross'.
otherwise, the function will regard 'cross' for 'on' instead of 'how'
upvoted 2 times

newusername 1 year, 7 months ago
ZSun is as always right. 4be8126 - it is not a problem to use gpt, but check its answers. Otherwise do not post it anywhere.
upvoted 1 times

Question #56 *Topic 1*

The code block shown below contains an error. The code block is intended to return a new DataFrame that is the result of a position-wise union between DataFrame storesDF and DataFrame acquiredStoresDF. Identify the error.
Code block:
storesDF.unionByName(acquiredStoresDF)

A. There is no DataFrame.unionByName() operation – the concat() operation should be used instead with both DataFrames as arguments.

B. There are no key columns specified – similar column names should be the second argument.

C. The DataFrame.unionByName() operation does not union DataFrames based on column position – it uses column name instead.

D. The unionByName() operation is a standalone operation rather than a method of DataFrame – it should have both DataFrames as arguments.

E. There are no column positions specified – the desired column positions should be the second argument.

Suggested Answer: *C*

Community vote distribution

C (100%)

juliom6 7 months, 2 weeks ago

Selected Answer: C

C is correct according to documentation:
https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.unionByName.html

"The difference between this function and union() is that this function resolves columns by name (not by position)"

upvoted 1 times

newusername 7 months, 3 weeks ago

Selected Answer: C

C is correct - https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.unionByName.html

upvoted 1 times

4be8126 1 year, 1 month ago

Selected Answer: C

The error in the code block is:

C. The DataFrame.unionByName() operation does not union DataFrames based on column position – it uses column name instead.

The unionByName() operation performs a position-wise union based on column names, not based on column positions. Therefore, the error in the code block is that the intended operation should be union(), which performs a position-wise union regardless of column names.

The correct code block to perform a position-wise union between DataFrame storesDF and DataFrame acquiredStoresDF would be:

storesDF.union(acquiredStoresDF)

upvoted 3 times

Question #57 *Topic 1*

Which of the following code blocks writes DataFrame storesDF to file path filePath as JSON?

A. storesDF.write.option("json").path(filePath)

B. storesDF.write.json(filePath)

C. storesDF.write.path(filePath)

D. storesDF.write(filePath)

E. storesDF.write().json(filePath)

Suggested Answer: *B*

Community vote distribution

B (100%)

juliom6 7 months, 2 weeks ago

Selected Answer: B

B is correct according documentation:

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameWriter.json.html

upvoted 1 times

4be8126 1 year, 1 month ago

Selected Answer: B

The correct answer is B.

Explanation:

The write method is used to write a DataFrame to a file system in various formats.
The json method specifies that the output format should be JSON.
The filePath argument specifies the location to write the output file.
Option A is incorrect because option requires a key-value pair (e.g., option("key", "value")).

Option C is incorrect because path is not a valid option for write.

Option D is incorrect because write method requires a format argument to specify the output format.

Option E is a valid option, but the parentheses after write are unnecessary.

upvoted 1 times

Question #58 *Topic 1*

In what order should the below lines of code be run in order to write DataFrame storesDF to file path filePath as parquet and partition by values in column division?

Lines of code:

1. .write() \
2. .partitionBy("division") \
3. .parquet(filePath)
4. .storesDF \
5. .repartition("division")
6. .write \
7. .path(filePath, "parquet")

A. 4, 1, 2, 3

B. 4, 1, 5, 7

C. 4, 6, 2, 3

D. 4, 1, 5, 3

E. 4, 6, 2, 7

Suggested Answer: *C*

Community vote distribution

C (100%)

juliom6 7 months, 2 weeks ago

Selected Answer: C

C is correct:

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameWriter.parquet.html

upvoted 1 times

newusername 7 months, 3 weeks ago

Selected Answer: C

Correct

upvoted 1 times

Question #59 *Topic 1*

The code block shown below contains an error. The code block intended to read a parquet at the file path filePath into a DataFrame. Identify the error.
Code block:
spark.read.load(filePath, source – "parquet")

A. There is no source parameter to the load() operation – the schema parameter should be used instead.

B. There is no load() operation – it should be parquet() instead.

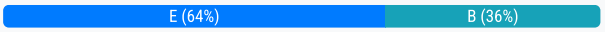
C. The spark.read operation should be followed by parentheses to return a DataFrameReader object.

D. The filePath argument to the load() operation should be quoted.

E. There is no source parameter to the load() operation – it can be removed.

Suggested Answer: *B*

Community vote distribution



juliom6 7 months, 2 weeks ago

Selected Answer: E

E is correct. The "format" parameter should be used instead of "source" (default "parquet"):

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameReader.load.html

format: str, optional
optional string for format of the data source. Default to 'parquet'.

upvoted 1 times

newusername 7 months, 3 weeks ago

Selected Answer: E

I would go for E

upvoted 1 times

Singh_Sumit 9 months ago

spark.read.load(PARQUET_PATH,format='parquet')

Load is valid, if provided with format.

upvoted 1 times

Ram459 10 months, 2 weeks ago

Selected Answer: E

Intention is to read a parquet at the file path filePath into a DataFrame

upvoted 2 times

cookiemonster42 11 months ago

The parameters for load() function are: path, format, schema, **options
A. Overall it makes sense, but do we really need to use schema?
B. There is load operation, that's FALSE
C. read is used without parenthesis, FALSE
D. It should indeed, but there's no source parameter, FALSE
E. That's true, but we need to put quotes for the filePath, then it's FALSE

Makes it A, but the question is really strange and not clear.

upvoted 2 times

cookiemonster42 11 months ago

UPD - parquet already has schema in it, it's not needed, then, I don't know what the answer is then

upvoted 2 times

Larrave 1 year ago

Selected Answer: E

Answer should be E. Removing source and default is 'parquet' anyway. However, it is not ideal to use load, rather the respective method.

https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrameReader.load.html?highlight=datframereader%20load#pyspark.sql.DataFrameReader.load

upvoted 3 times

ZSun 1 year ago

1. pyspark.sql.SparkSession.read Returns a DataFrameReader
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.SparkSession.read.html#pyspark.sql.SparkSession.read
2. we check this DataFrameReader, it contains both "load" and "parquet" methods.
2.1. for load, load(path, format, schema)
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameReader.load.html#pyspark.sql.DataFrameReader.load
Therefore, the answer is A or E.
Typically parquet contains schema information.
I do not like this question, because if reading a parquet file, directly use spark.read.parquet()

upvoted 2 times

4be8126 1 year, 1 month ago

Selected Answer: B

The correct code block to read a parquet file would be

spark.read.parquet(filePath).

upvoted 4 times

tmz1 5 months ago

the statement "there is no load() operation" in answer B is clearly wrong as this operation exists in pyspark. E is correct answer - instead of source parameter, you should use format to achieve the same result

upvoted 1 times

Question #60 *Topic 1*

In what order should the below lines of code be run in order to read a JSON file at the file path filePath into a DataFrame with the specified schema schema?

Lines of code:

1. .json(filePath, schema = schema)
2. .storesDF
3. .spark \
4. .read() \
5. .read \
6. .json(filePath, format = schema)

A. 3, 5, 6

B. 2, 4, 1

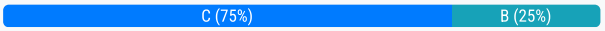
C. 3, 5, 1

D. 2, 5, 1

E. 3, 4, 1

Suggested Answer: *C -*

Community vote distribution



azure_bimonster 10 months, 3 weeks ago

Selected Answer: C

we use the following structure: spark.read.json(filePath, schema=schemaName)

upvoted 2 times

juliom6 1 year, 1 month ago

Selected Answer: C

C is correct:

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameReader.json.html

json function does not have a "format" parameter.

upvoted 2 times

ZSun 1 year, 6 months ago

storesDF = spark.read.json(filePath, schema = schema)

C

upvoted 3 times

Jtic 1 year, 7 months ago

Selected Answer: B

2. .storesDF: This line is unrelated to reading the JSON file and can be disregarded.

.read(): This line invokes the DataFrameReader's read() method to create a DataFrameReader object.

.json(filePath, schema=schema): This line uses the DataFrameReader object to read the JSON file at the specified filePath into a DataFrame with the provided schema.

upvoted 1 times

pnev 12 months ago

This is so wrong.. in order to read a table you need to use spark.read.json / parquet / Table.

upvoted 1 times

Question #61 *Topic 1*

Which of the following storage levels should be used to store as much data as possible in memory on two cluster nodes while storing any data that does not fit in memory on disk to be read in when needed?

A. MEMORY_ONLY_2

B. MEMORY_AND_DISK_SER

C. MEMORY_AND_DISK

D. MEMORY_AND_DISK_2

E. MEMORY_ONLY

Suggested Answer: *D*

Community vote distribution

D (100%)

58470e1 7 months, 2 weeks ago

D is correct.

upvoted 2 times

newusername 1 year, 7 months ago

Selected Answer: D

Correct is D

upvoted 2 times

juadaves 1 year, 8 months ago

B

https://stackoverflow.com/questions/30520428/what-is-the-difference-between-memory-only-and-memory-and-disk-caching-level-in

upvoted 1 times

newusername 1 year, 7 months ago

not sure how you came to the conclusion of B after that article, but it states clearly in the question "on two cluster nodes" ! So, the answer is D

upvoted 1 times

Question #62 *Topic 1*

Which of the following Spark properties is used to configure the maximum size of an automatically broadcasted DataFrame when performing a join?

A. spark.sql.broadcastTimeout

B. spark.sql.autoBroadcastJoinThreshold

C. spark.sql.shuffle.partitions

D. spark.sql.inMemoryColumnarStorage.batchSize

E. spark.sql.adaptive.skewedJoin.enabled

Suggested Answer: *B*

Community vote distribution

B (100%)

thanab 9 months, 2 weeks ago

Selected Answer: B

The correct answer is B. spark.sql.autoBroadcastJoinThreshold. This property in Apache Spark is used to configure the maximum size (in bytes) of a table that will be broadcast to all worker nodes when performing a join. If the size of the table is below this threshold, it will be broadcasted, which can significantly speed up join operations.

upvoted 1 times

Question #63 *Topic 1*

Which of the following Spark properties is used to configure whether skewed partitions are automatically detected and subdivided into smaller partitions when joining two DataFrames together?

A. spark.sql.adaptive.skewedJoin.enabled

B. spark.sql.adaptive.coalescePartitions.enable

C. spark.sql.adaptive.skewHints.enabled

D. spark.sql.shuffle.partitions

E. spark.sql.shuffle.skewHints.enabled

Suggested Answer: *B*

Community vote distribution

A (100%)

Larrave Highly Voted 1 year, 6 months ago

Selected Answer: A

Answer should be A, but the config is skewJoin not skewedJoin

upvoted 5 times

amirshaz Most Recent 11 months, 1 week ago

Selected Answer: A

A is correct

upvoted 1 times

thanab 1 year, 3 months ago

A

The Spark property used to configure whether skewed partitions are automatically detected and subdivided into smaller partitions when joining two DataFrames together is `spark.sql.adaptive.skewJoin.enabled`. This feature dynamically handles skew in sort-merge join by splitting (and replicating if needed) skewed tasks into roughly evenly sized tasks. It takes effect when both `spark.sql.adaptive.enabled` and `spark.sql.adaptive.skewJoin.enabled` configurations are enabled.

upvoted 3 times

Question #64 *Topic 1*

Which of the following statements about the Spark DataFrame is true?

A. Spark DataFrames are mutable unless they've been collected to the driver.

B. A Spark DataFrame is rarely used aside from the import and export of data.

C. Spark DataFrames cannot be distributed into partitions.

D. A Spark DataFrame is a tabular data structure that is the most common Structured API in Spark.

E. A Spark DataFrame is exactly the same as a data frame in Python or R.

Suggested Answer: *D*

Community vote distribution

D (100%)

azure_bimonster 10 months, 3 weeks ago

Selected Answer: D

D seems correct here

upvoted 1 times

Question #65 *Topic 1*

Which of the following operations can be used to return a new DataFrame from DataFrame storesDF without columns that are specified by name?

A. storesDF.filter()

B. storesDF.select()

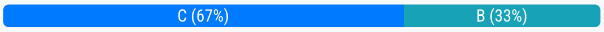
C. storesDF.drop()

D. storesDF.subset()

E. storesDF.dropColumn()

Suggested Answer: *C*

Community vote distribution



mehroosali 7 months, 3 weeks ago

Selected Answer: C

Its C not B

upvoted 1 times

newusername 7 months, 3 weeks ago

Selected Answer: C

you should be careful; with the question. The specified columns should not be in the new DF

therefore it is drop , you specify in drop what you want to remove and they won't be in the new DF

upvoted 1 times

cookiemonster42 10 months ago

Selected Answer: B

in select you can use * instead of the column names

upvoted 1 times

Question #66 *Topic 1*

Which of the following code blocks returns a DataFrame containing only the rows from DataFrame storesDF where the value in column sqft is less than or equal to 25,000?

A. storesDF.where(storesDF[sqft] > 25000)

B. storesDF.filter(sqft > 25000)

C. storesDF.filter("sqft" <= 25000)

D. storesDF.filter(col("sqft") <= 25000)

E. storesDF.where(sqft > 25000)

Suggested Answer: *D*

Community vote distribution

D (100%)

5523042 8 months, 1 week ago

Selected Answer: D

Answer is D

upvoted 1 times

Question #67 *Topic 1*

Which of the following code blocks returns a DataFrame containing only the rows from DataFrame storesDF where the value in column sqft is less than or equal to 25,000 OR the value in column customerSatisfaction is greater than or equal to 30?

A. storesDF.filter(col("sqft") <= 25000 and col("customerSatisfaction") >= 30)

B. storesDF.filter(col("sqft") <= 25000 | col("customerSatisfaction") >= 30)

C. storesDF.filter(col(sqft) <= 25000 or col(customerSatisfaction) >= 30)

D. storesDF.filter(sqft <= 25000 | customerSatisfaction >= 30)

E. storesDF.filter(col("sqft") <= 25000 or col("customerSatisfaction") >= 30)

Suggested Answer: *B*

Akash567890978 Highly Voted 12 months ago

I dont think even B is correct the conditions should be inside parenthesis as well

upvoted 10 times

JahanzebBehan Most Recent 6 months, 1 week ago

Selected Answer: B

Correct usage of col and "|"

upvoted 1 times

Question #68 *Topic 1*

The code block shown below should return a new DataFrame from DataFrame storesDF where column storeId is of the type string. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.

Code block:

storesDF.__1__("storeId", __2__("storeId").__3__(__4__)

A. 1. withColumn
2. col
3. cast
4. StringType()

B. 1. withColumn
2. cast
3. col
4. StringType()

C. 1. newColumn
2. col
3. cast
4. StringType()

D. 1. withColumn
2. cast
3. col
4. StringType

E. 1. withColumn
2. col
3. cast
4. StringType

Suggested Answer: *A*

Sowwy1 8 months, 4 weeks ago
should be A
upvoted 2 times

Question #69 *Topic 1*

Which of the following code blocks returns a new DataFrame from DataFrame storesDF where column modality is the constant string "PHYSICAL"? Assume DataFrame storesDF is the only defined language variable.

A. storesDF.withColumn("modality", lit(PHYSICAL))

B. storesDF.withColumn("modality", col("PHYSICAL"))

C. storesDF.withColumn("modality", lit("PHYSICAL"))

D. storesDF.withColumn("modality", StringType("PHYSICAL"))

E. storesDF.withColumn("modality", "PHYSICAL")

Suggested Answer: *C*

Sowwy1 8 months, 4 weeks ago

should be C

upvoted 2 times

Question #70 *Topic 1*

The code block shown below contains an error. The code block is intended to return a new DataFrame where column managerName from DataFrame storesDF is split at the space character into column managerFirstName and column managerLastName. Identify the error.

A sample of DataFrame storesDF is displayed below:

storeId	open	openDate	managerName
0	true	1100746394	Vulputate Curabitur
1	true	944572255	Tempor Augue
2	false	925495628	Aliquam Et
3	true	1397353092	Faucibus Orci
4	true	986505057	Sed Fermentum
...	...	...	...

Code block:

storesDF.withColumn("managerFirstName", col("managerName").split(" ").getItem(0))
.withColumn("managerLastName", col("managerName").split(" ").getItem(1))

A. The index values of 0 and 1 are not correct – they should be 1 and 2, respectively.

B. The index values of 0 and 1 should be provided as second arguments to the split() operation rather than indexing the result.

C. The split() operation comes from the imported functions object. It accepts a string column name and split character as arguments. It is not a method of a Column object.

D. The split() operation comes from the imported functions object. It accepts a Column object and split character as arguments. It is not a method of a Column object.

E. The withColumn operation cannot be called twice in a row.

Suggested Answer: *D*

Community vote distribution

C (100%)

tmz1 5 months ago

In my opinion both C and D are correct because split function from package pyspark.sql.functions accepts both column object or string as a first parameter

upvoted 2 times

Sowwy1 1 year, 2 months ago

D. The split() operation comes from the imported functions object. It accepts a Column object and split character as arguments. It is not a method of a Column object.

upvoted 1 times

Ahlo 1 year, 4 months ago

Answer C

pyspark.sql.functions provides a function split() to split DataFrame string Column into multiple columns.

https://sparkbyexamples.com/pyspark/pyspark-split-dataframe-column-into-multiple-columns/

upvoted 1 times

newusername 1 year, 7 months ago

Selected Answer: C

I think it is C

data = [

("John Smith",),

("Jane Doe",),

("Mike Johnson",)

```
]

df = spark.createDataFrame(data, ["managerName"])

df.show()

df = df.withColumn("managerFirstName", split(col("managerName"), " ").getItem(0)) \
.withColumn("managerLastName", split(col("managerName"), " ").getItem(1))

df.show()
```

upvoted 2 times

cd6a625 11 months, 3 weeks ago

in your example, your are using split(col("managerName"), ...) and not split("managerName", ...) <- means that answer is D

upvoted 1 times

zozoshanky 1 year, 11 months ago

Can be C as an answer too.

upvoted 1 times

cookiemonster42 1 year, 11 months ago

But you have to pass a column as an object, not a string. you have to use col() expression. So D is the right one.

upvoted 4 times

65bd33e 1 year, 1 month ago

Yes, I agree with you, D is correct we have to pass a column as an object

upvoted 1 times

Question #71 *Topic 1*

The code block shown below should return a new DataFrame where single quotes in column storeSlogan have been replaced with double quotes. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.

A sample of DataFrame storesDF is below:

storeId	storeSlogan
0	'consequat vitae ...
1	'aliquam at pelle...
2	'non ac leo phare...
3	'eget purus vel sed"
4	'vitae phasellus ...
...	...

Code block:

storesDF.__1__(__2__, __3__(__4__, __5__, __6__))

A. 1. withColumn
2. "storeSlogan"
3. regexp_extract
4. col("storeSlogan")
5. "\""
6. """

B. 1. newColumn
2. storeSlogan
3. regexp_extract
4. col(storeSlogan)
5. "\""
6. """

C. 1. withColumn
2. "storeSlogan"
3. regexp_replace
4. col("storeSlogan")
5. "\""
6. """

D. 1. withColumn
2. "storeSlogan"
3. regexp_replace
4. col("storeSlogan")
5. """
6. "\""

E. 1. withColumn
2. "storeSlogan"
3. regexp_extract
4. col("storeSlogan")
5. """
6. "\""

Suggested Answer: *D*

Community vote distribution

D (100%)

ARUNKUMARKRISHNASAMY 3 months, 3 weeks ago

Selected Answer: D

In Python, the expression "" (double quote inside single quotes) or \" (escaped double quote) both represent a double quote (").

In the correct answer D, the replacement part is written as:

regexp_replace(col("storeSlogan"), "", "\"")
The first argument ("storeSlogan") refers to the column to be modified.
The second argument ("") specifies the single quote character to be replaced.
The third argument ("\"" or "") represents the double quote character that will replace the single quote.
Since Python uses backslashes (\) to escape special characters, \" ensures that the double quote is correctly interpreted as part of the replacement string.

So, in short:
✓ ' " ' and ' \" ' are equivalent in this case, both representing a double quote.

upvoted 1 times

azure_bimonster 10 months, 3 weeks ago

Selected Answer: D

To me it would like this:

storesDF.withColumn("storeSlogan", regexp_replace(col("storeSlogan"), " ' ", "\""))

upvoted 1 times

cookiemonster42 1 year, 5 months ago

Selected Answer: D

D is correct, we need a replacement function with first argument ""

upvoted 2 times

Question #72 *Topic 1*

Which of the following code blocks returns a new DataFrame where column division from DataFrame storesDF has been replaced and renamed to column state and column managerName from DataFrame storesDF has been replaced and renamed to column managerFullName?

A. storesDF.withColumnRenamed("division", "state")
.withColumnRenamed("managerName", "managerFullName")

B. storesDF.withColumn("state", "division")
.withColumn("managerFullName", "managerName")

C. storesDF.withColumn("state", col("division"))
.withColumn("managerFullName", col("managerName"))

D. storesDF.withColumnRenamed(Seq("division", "state"), Seq("managerName", "managerFullName"))

E. storesDF.withColumnRenamed("state", "division")
.withColumnRenamed("managerFullName", "managerName")

Suggested Answer: *A*

Community vote distribution

A (100%)

azure_bimonster 10 months, 3 weeks ago

Selected Answer: A

A would be the right one here.

upvoted 1 times

zozoshanky 1 year, 5 months ago

B is the answer.

upvoted 1 times

newusername 1 year, 1 month ago

you are wrong! The answer is A

upvoted 1 times

Question #73 *Topic 1*

Which of the following code blocks returns a new DataFrame where column sqft from DataFrame storesDF has had its missing values replaced with the value 30,000?

A sample of DataFrame storesDF is below:

storeId	sqft
0	43161
1	51200
2	null
3	78367
4	null
...	...

A. storesDF.na.fill(30000, Seq("sqft"))

B. storesDF.nafill(30000, col("sqft"))

C. storesDF.na.fill(30000, col("sqft"))

D. storesDF.fillna(30000, col("sqft"))

E. storesDF.na.fill(30000, "sqft")

Suggested Answer: *E*

Community vote distribution

Samir_91 Highly Voted 6 months, 3 weeks ago

E is answer. It's tested.

A)AttributeError: module 'pyspark.sql.functions' has no attribute 'Seq'

B)AttributeError: 'DataFrame' object has no attribute 'nafill'

C)PySparkTypeError: [NOT_LIST_OR_TUPLE] Argument `subset` should be a list or tuple, got Column.

D)PySparkTypeError: [NOT_LIST_OR_TUPLE] Argument `subset` should be a list or tuple, got Column.

upvoted 5 times

Samir_91 Most Recent 6 months, 3 weeks ago

Selected Answer: E

E is answer

upvoted 3 times

hosniadel666 9 months, 1 week ago

Selected Answer: A

Check fill function at scala API docs

https://spark.apache.org/docs/3.0.0/api/scala/org/apache/spark/sql/DataFrameNaFunctions.html#fill(value:Long,cols:Array%5BString%5D):org.apache.spa

upvoted 1 times

azure_bimonster 10 months, 3 weeks ago

Selected Answer: C

To me C is likely correct, because we need to use col()

C. storesDF.na.fill(30000, col("sqft"))

upvoted 1 times

tmz1 5 months ago

it is wrong. na.fill function accepts str, list or tuple as the second argument, not column object

upvoted 1 times

learnsh1 10 months, 3 weeks ago

typo error I THINK 1st arg is col name right ?

upvoted 2 times

Question #74 *Topic 1*

Which of the following operations can be used to return a DataFrame with no duplicate rows? Please select the most complete answer.

A. DataFrame.distinct()

B. DataFrame.dropDuplicates() and DataFrame.distinct()

C. DataFrame.dropDuplicates()

D. DataFrame.drop_duplicates()

E. DataFrame.dropDuplicates(), DataFrame.distinct() and DataFrame.drop_duplicates()

Suggested Answer: *E*

Community vote distribution

E (100%)

Ahlo 10 months, 1 week ago

Answer E

drop_duplicates() is an alias for dropDuplicates() it also work in pyspark

upvoted 1 times

azure_bimonster 10 months, 3 weeks ago

Selected Answer: E

it asks "most complete" one, so E would be correct as all these three options would work in pyspark

upvoted 1 times

thanab 1 year, 3 months ago

B

The most complete answer is B. DataFrame.dropDuplicates() and DataFrame.distinct(). Both DataFrame.distinct() and DataFrame.dropDuplicates() methods in PySpark can be used to return a new DataFrame with duplicate rows removed. The DataFrame.drop_duplicates() method is used in pandas, not in PySpark.

upvoted 1 times

juadaves 1 year, 2 months ago

It should be E, drop_duplicates() works in pyspark too.

upvoted 1 times

Question #75 *Topic 1*

QUESTION NO: 75 -

Which of the following code blocks returns a DataFrame where column divisionDistinct is the approximate number of distinct values in column division from DataFrame storesDF?

A. storesDF.withColumn("divisionDistinct", approx_count_distinct(col("division")))

B. storesDF.agg(col("division").approx_count_distinct("divisionDistinct"))

C. storesDF.agg(approx_count_distinct(col("division")).alias("divisionDistinct"))

D. storesDF.withColumn("divisionDistinct", col("division").approx_count_distinct())

E. storesDF.agg(col("division").approx_count_distinct().alias("divisionDistinct"))

Suggested Answer: *C*

Sowwy1 9 months ago

I think it's C

https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.functions.approx_count_distinct.html

upvoted 2 times

Question #76 *Topic 1*

The code block shown below should return a new DataFrame with the mean of column sqft from DataFrame storesDF in column sqftMean. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.

Code block:

storesDF.__1__(__2__(__3__).alias("sqftMean"))

A. 1. agg
2. mean
3. col("sqft")

B. 1. withColumn
2. mean
3. col("sqft")

C. 1. agg
2. average
3. col("sqft")

D. 1. mean
2. col
3. "sqft"

E. 1. agg
2. mean
3. "sqft"

Suggested Answer: *A*

Community vote distribution

A (100%)

cookiemonster42 11 months ago

Selected Answer: A

A is the right one

upvoted 1 times

Question #77 *Topic 1*

Which of the following code blocks returns the number of rows in DataFrame storesDF for each unique value in column division?

A. storesDF.groupBy("division").agg(count())

B. storesDF.agg(groupBy("division").count())

C. storesDF.groupby.count("division")

D. storesDF.groupBy().count("division")

E. storesDF.groupBy("division").count()

Suggested Answer: *E*

Community vote distribution

E (100%)

azure_bimonster 10 months, 3 weeks ago

Selected Answer: E

E is the right answer here

upvoted 2 times