Question #1 *Topic 1*

HOTSPOT -

Case Study -
This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.
To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study -
To display the first question in this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Existing Environment -

Azure Environment -
Contoso has an Azure subscription in North Europe that contains the corporate infrastructure. The current infrastructure contains a Microsoft SQL Server 2017 database. The database contains the following tables.

Table names	Column names
CustomerFeedback	• FeedbackId (int) (primarykey) • FeedbackJson (nvarchar (max))
Fleets	• FleetId (int) (primarykey) • FleetName (nvarchar(100)) • Description (nvarchar(256))
MaintenanceEvents	• MaintenanceId (int) (primarykey) • VehicleId (int) • LastModifiedUTC (datetime2) • Description (nvarchar(256))
SupportTickets	• TicketId (int) (primarykey) • FleetId (int) • CreatedUtc (datetime2)
UserAccounts	• UserId (int) (primarykey) • UserPrincipalName (nvarchar(256)) • JobRole (nvarchar(256)) • StartDate (datetime2)
VehicleIncidentReports	• IncidentId (int) (primarykey) • VehicleId (int) • FleetId (int) • IncidentType (nvarchar(50)) • VehicleLocation (nvarchar(200)) • IncidentDescription (nvarchar(max)) • SeverityScore (int)
Vehicles	• VehicleId (int) (primarykey) • VIN ((nvarchar(50)) • VehicleDescription (nvarchar(256))
VehicleHealthSummary	• VehicleId (int) (primarykey) • FleetId (int) • Summary (nvarchar(2000)) • LastUpdatedUtc (datetime2) • EngineStatus [bit] • EngineStatusLastUpdatedUtc (datetime2) • BatteryHealth (int) • Embeddings (vector (1536))

The FeedbackJson column has a full-text index and stores JSON documents in the following format.

```
{
  "text": "The battery drains too fast when driving uphill.",
  "category": "Battery",
  "metadata": {
    "appVersion": "5.2.1",
    "device": "Android",
    "language": "en-GB"
  }
}
```

The support staff at Contoso never has the UNMASK permission.

Problem Statements -
Contoso is deploying a new Azure SQL database that will become the authoritative data store for the following:

AI workloads -

Vector search -

Modernized API access -
Retrieval Augmented Generation (RAG) pipelines
Sometimes the ingestion pipeline fails due to malformed JSON and duplicate payloads.
The engineers at Contoso report that the following dashboard query runs slowly.

```
SELECT VehicleId, LastUpdatedUtc, EngineStatus, BatteryHealth
FROM dbo.VehicleHealthSummary
WHERE FleetId = @FleetId
ORDER BY LastUpdatedUtc DESC;
```

You review the execution plan and discover that the plan shows a clustered index scan.
VehicleIncidentReports often contains details about the weather, traffic conditions, and location. Analysts report that it is difficult to find similar incidents based on these details.

Requirements -

Planned Changes -
Contoso wants to modernize Fleet Intelligence Platform to support AI-powered semantic search over incident reports.

Security Requirements -
Contoso identifies the following security requirements:
Restrict the support staff from viewing Personally Identifiable Information (PII) data, which is full email addresses and phone numbers.
Enforce row-level filtering so that analysts see only incidents for the fleets to which they are assigned. The analysts can be assigned to multiple fleets.
Database Performance and Requirements
Contoso identifies the following telemetry requirements:
Telemetry data must be stored in a partitioned table.
Telemetry data must provide predictable performance for ingestion and retention operations. latitude, longitude, and accuracy JSON properties must be filtered by using an index seek.
Contoso identifies the following maintenance data requirements:
Ensure that any changes to a row in the MaintenanceEvents table updates the corresponding value in the LastModifiedUtc column to the time of the change.
Avoid recursive updates.
AI Search, Embeddings, and Vector Indexing
Contoso plans to implement semantic search over incident data to meet the following requirements:
Embeddings must be stored in dedicated Azure SQL Database tables.
Embeddings must be generated from rich natural language fields.
Chunking must preserve semantic coherence.
Hybrid search must combine the following:

Vector similarity -

Keyword filtering or boosting -

Development Requirements -

The development team at Contoso will use Microsoft Visual Studio Code and GitHub Copilot and will retrieve live metadata from the databases.

Contoso identifies the following requirements for querying data in the FeedbackJson column of the CustomerFeedback table:

Extract the customer feedback text from the JSON document.

Filter rows where the JSON text contains a keyword.

Calculate a fuzzy similarity score between the feedback text and a known issue description.

Order the results by similarity score, with the highest score first.

You need to meet the development requirements for the FeedbackJson column.

How should you complete the Transact-SQL query? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Answer Area

```
SELECT
  f.FeedbackId,
  f.VehicleId,
  [                                                    ] ,
  EDIT_DISTANCE_SIMILARITY(
    JSON_VALUE(f.FeedbackJson, '$.text'),
    @KnownIssueDescription
  ) AS SimilarityScore
FROM
  dbo.CustomerFeedback f
WHERE
  [                                                    ]
ORDER BY
  [                                                    ]
DESC;
```

- CONTAINS(FeedbackJson, @Keyword)
- EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.details.comment'), @Keyword) < 3
- EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.text'), @Keyword) < 3
- JSON_QUERY(f.FeedbackJson, '$.text', @KnownIssueDescription) AS FeedbackText
- JSON_VALUE(f.FeedbackJson, '$.text') AS FeedbackText
- SimilarityScore

Suggested Answer:

Answer Area

```
SELECT
  f.FeedbackId,
  f.VehicleId,
  [ CONTAINS(FeedbackJson, @Keyword) | EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.details.comment'), @Keyword) < 3 | EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.text'), @Keyword) < 3 | JSON_QUERY(f.FeedbackJson, '$.text', @KnownIssueDescription) AS FeedbackText | JSON_VALUE(f.FeedbackJson, '$.text') AS FeedbackText | SimilarityScore ],
  EDIT_DISTANCE_SIMILARITY(
    JSON_VALUE(f.FeedbackJson, '$.text'),
    @KnownIssueDescription
  ) AS SimilarityScore
FROM
  dbo.CustomerFeedback f
WHERE
  [ CONTAINS(FeedbackJson, @Keyword) | EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.details.comment'), @Keyword) < 3 | EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.text'), @Keyword) < 3 | JSON_QUERY(f.FeedbackJson, '$.text', @KnownIssueDescription) AS FeedbackText | JSON_VALUE(f.FeedbackJson, '$.text') AS FeedbackText | SimilarityScore ]
ORDER BY
  [ CONTAINS(FeedbackJson, @Keyword) | EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.details.comment'), @Keyword) < 3 | EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.text'), @Keyword) < 3 | JSON_QUERY(f.FeedbackJson, '$.text', @KnownIssueDescription) AS FeedbackText | JSON_VALUE(f.FeedbackJson, '$.text') AS FeedbackText | SimilarityScore ]
DESC;
```

Currently there are no comments in this discussion, be the first to comment!

Question #2 *Topic 1*

DRAG DROP -

You have an Azure SQL database that contains a table named dbo.Orders.

You have an application that calls a stored procedure named dbo.usp_CreateOrder to insert rows into dbo.Orders.

When an insert fails, the application receives inconsistent error details.

You need to implement error handling to ensure that any failures inside the procedure abort the transaction and return a consistent error to the caller.

How should you complete the stored procedure? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.

Values

- BEGIN CATCH
- IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
- RAISERROR('CreateOrder failed', 16, 1)
- ROLLBACK TRANSACTION
- SET @OrderId = SCOPE_IDENTITY()
- THROW

Answer Area

```
CREATE OR ALTER PROCEDURE dbo.usp_CreateOrder
    @CustomerId int,
    @Amount decimal(10,2),
    @OrderId int OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION;
        INSERT INTO dbo.Orders(CustomerId, Amount, CreatedAt)
        VALUES (@CustomerId, @Amount, SYSUTCDATETIME());
        [                                        ] ;
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        [                                        ] ;
        THROW;
    END CATCH
END
```

Suggested Answer:

Answer Area

```
CREATE OR ALTER PROCEDURE dbo.usp_CreateOrder
    @CustomerId int,
    @Amount decimal(10,2),
    @OrderId int OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION;
        INSERT INTO dbo.Orders(CustomerId, Amount, CreatedAt)
        VALUES (@CustomerId, @Amount, SYSUTCDATETIME());
        SET @OrderId = SCOPE_IDENTITY() ;
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
        THROW;
    END CATCH
END
```

Currently there are no comments in this discussion, be the first to comment!

Question #3 *Topic 1*

Your team is developing an Azure SQL dataset solution from a locally cloned GitHub repository by using Microsoft Visual Studio Code and GitHub Copilot Chat.
You need to disable the GitHub Copilot repository-level instructions for yourself without affecting other users.
What should you do?

A. From Visual Studio Code, modify your GitHub Copilot Chat user settings.

B. Add a --debug flag when you start the GitHub Copilot Chat extension.

C. Delete .github/copilot-instructions.md.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #4 *Topic 1*

You have an Azure SQL database that contains the following SQL graph tables:

A NODE table named dbo.Person -

An EDGE table named dbo.Knows -
Each row in dbo.Person contains the following columns:
PersonID (int)
DisplayName (nvarchar(100))
You need to use a MATCH operator and exactly two directed Knows relationships to return the PersonID and DisplayName of people that are reachable from the person identified by an input parameter named @StartPersonId.
Which Transact-SQL query should you use?

A.
```
SELECT p2.PersonId, @StartPersonId
FROM dbo.Person AS p1, dbo.Knows AS k1, dbo.Person AS p2, dbo.Knows AS k2, dbo.Person AS p3
WHERE p1.DisplayName = p2.DisplayName
  AND MATCH(p1-(k1)->p2-(k2)->p3);
```

B.
```
SELECT p3.PersonId, p3.DisplayName FROM dbo.Person AS p1
JOIN dbo.Knows AS k1 ON 1 = 1
JOIN dbo.Person AS p2 ON 1 = 1
JOIN dbo.Knows AS k2 ON 1 = 1
JOIN dbo.Person AS p3 ON 1 = 1
WHERE p1.PersonId = @StartPersonId
  AND MATCH(p3<-(k2)-p2<-(k1)-p1);
```

C.
```
SELECT p3.PersonId, p3.DisplayName
FROM dbo.Person AS p1, dbo.Knows AS k1, dbo.Person AS p2, dbo.Knows AS k2, dbo.Person AS p3
WHERE p1.PersonId = @StartPersonId
  AND MATCH(p1-(k1)->p2)   AND MATCH(p2-(k2)->p3);
```

D.
```
SELECT p3.PersonId, p3.DisplayName
FROM dbo.Person AS p1, dbo.Knows AS k1, dbo.Person AS p2, dbo.Knows AS k2, dbo.Person AS p3
WHERE p1.PersonId = @StartPersonId
  AND MATCH(p1-(k1)->p2-(k2)->p3);
```

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #5 *Topic 1*

You have a SQL database in Microsoft Fabric that contains a column named Payload. Payload stores customer data in JSON documents that have the following format.

```
{
"date": "2020-01-25",
"customer_email": "user@contoso.com",
...
}
```

Data analysis shows that some customers have subaddressing in their email address, for example, user1+promo@contoso.com.

You need to return a normalized email value that removes the subaddressing, for example, user1 +promo@contoso.com must be normalized to user1@contoso.com.

Which Transact-SQL expression should you use?

A. REGEXP_REPLACE(JSON_VALUE(Payload, '$.customer_email'), '\+.*$', '')

B. REGEXP_SUBSTR(JSON_VALUE(Payload, '$.customer_email'), '^[^+]+@.*$=')

C. REGEXP_REPLACE(JSON_VALUE(Payload, '$.customer_email'), '\+.*@', '@')

D. REGEXP_REPLACE(JSON_VALUE(Payload, '$.customer_email'), '\+.*', '')

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #6 *Topic 1*

DRAG DROP -

You have an Azure SQL database that contains a table named dbo.Orders. dbo.Orders contains a column named CreateDate that stores order creation dates.

You need to create a stored procedure that filters Orders by CreateDate for a single calendar day. The solution must be SARGable.

How should you complete the Transact-SQL code? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.

Values

- @EndDate
- @StartDate
- CONVERT(char(10), CreateDate, 121)
- CONVERT(date, @StartDate)
- DATEADD(day, 1, @StartDate)
- GETDATE()

Answer Area

```
CREATE PROCEDURE dbo.usp_SearchOrders
  @StartDate date
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @EndDate date;
    SET @EndDate =
    SELECT  o.CreateDate,
            o.OrderId,
            o.ShipDate
    FROM    dbo.Orders AS o
    WHERE   o.CreateDate >=
      AND   o.CreateDate <
END;
GO
```

Suggested Answer:

Answer Area

```
CREATE PROCEDURE dbo.usp_SearchOrders
  @StartDate date
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @EndDate date;
    SET @EndDate =  DATEADD(day, 1, @StartDate)
    SELECT  o.CreateDate,
            o.OrderId,
            o.ShipDate
    FROM    dbo.Orders AS o
    WHERE   o.CreateDate >=  @StartDate
      AND   o.CreateDate <  @EndDate ;
END;
GO
```

Currently there are no comments in this discussion, be the first to comment!

Question #7 *Topic 1*

You have an Azure SQL database.
You need to create a scalar user-defined function (UDF) that returns the number of whole years between an input parameter named @OrderDate and the current date/time as a single positive integer. The function must be created in Azure SQL Database.
You write the following code.

```
01 CREATE FUNCTION dbo.ufnYearsSinceOrder (@OrderDate datetime2)
02 RETURNS int
03 AS
04 BEGIN
05
06 END
```

What should you insert at line 05?

A. RETURN DATEDIFF(year, GETDATE(), @OrderDate);

B. DATEDIFF(month, @orderdate, GETDATE()) / 12

C. DATEPART(year, GETDATE()) - DATEPART(year, @orderdate)

D. RETURN DATEDIFF(year, @OrderDate, GETDATE());

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #8 *Topic 1*

You have an Azure SQL database.
You deploy Data API builder (DAB) to Azure Container Apps by using the mcr.microsoft.com/azure-databases/data-api-builder:latest image.
You have the following Container Apps secrets:
MSSQL_CONNECTION_STRING that maps to the SQL connection string
DAB_CONFIG_BASE64 that maps to the DAB configuration
You need to initialize the DAB configuration to read the SQL connection string.
Which command should you run?

A. dab init --database-type mssql --connection-string “secretref:DAB_CONFIG_BASE64” --host-mode Production --config dab-config.json

B. dab init --database-type mssql --connection-string “@env(‘MSSQL_CONNECTION_STRING’)” --host-mode Production --config dab-config.json

C. dab init --database-type mssql --connection-string “secretref:mssql-connection-string” --host-mode Production --config dab-config.json

D. dab init --database-type mssql --connection-string “@env(‘DAB_CONFIG_BASE64’)” --host-mode Production --config dab-config.json

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #9 *Topic 1*

You have a SQL database in Microsoft Fabric that contains a nvarchar (max) column named MessageText. An ID is always contained within the first paragraph of MessageText.
You need to write a Transact-SQL query that uses REGEXP_SUBSTR to extract the ID from MessageText.
What should you include in the query?

A. Apply STRING_ESCAPE(MessageText, 'json') before calling REGEXP_SUBSTR.

B. Cast MessageText to nvarchar (4000) before calling REGEXP_SUBSTR.

C. Add a COLLATE Latin1_General_CS_AS clause to MessageText before calling REGEXP_SUBSTR.

D. Run TRY_CONVERT(varchar(max), MessageText) before calling REGEXP_SUBSTR.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #10 *Topic 1*

You have an Azure SQL database that contains database-level Data Definition Language (DDL) triggers, including a trigger named ddl_Audit.
You need to prevent ddl_Audit from firing during the next deployment. The trigger object must remain in place.
Which Transact-SQL statement should you use?

A. ALTER TRIGGER

B. ALTER DATABASE

C. ALTER SERVER AUDIT SPECIFICATION

D. DISABLE TRIGGER

E. ALTER DATABASE AUDIT SPECIFICATION

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #11 *Topic 1*

Your development team uses GitHub Copilot Chat in Microsoft SQL Server Management Studio (SSMS) to generate and run Transact-SQL queries against an Azure SQL database named DB1. DB1 contains tables that store sensitive customer data.
You need to ensure that any Transact-SQL queries that run from GitHub Copilot Chat in SSMS are restricted by the same permissions as the developer's database login.
What prevents the GitHub Copilot Chat-run queries from accessing data beyond the developer's access?

A. GitHub Copilot Chat runs queries in a read-only sandbox that is isolated from production database permissions.

B. GitHub Copilot Chat runs queries by using the developer's database identity and permissions.

C. GitHub Copilot Chat filters query results on the client side to remove rows the developer is unauthorized to see.

D. GitHub Copilot Chat uses different row-level security (RLS) policies than the developer.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #12 *Topic 1*

You have an Azure SQL database named AdventureWorksDB that contains a table named dbo.Employee.
You have a C# Azure Functions app that uses an HTTP-triggered function with an Azure SQL input binding to query dbo.Employee.
You are adding a second function that will react to row changes in dbo.Employee and write structured logs.
You need to configure AdventureWorksDB and the app to meet the following requirements:
Changes to dbo.Employee must trigger the new function within five seconds.
Each invocation must process no more than 100 changes.
Which two database configurations should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Create an AFTER trigger on dbo.Employee for Data Manipulation Language (DML).

B. Set Sql_Trigger_MaxBatchSize to 100.

C. Enable change tracking on the dbo.Employee table.

D. Enable change tracking at the database level.

E. Set Sql_Trigger_PollingIntervalMs to 5000.

F. Enable change data capture (CDC) for dbo.Employee table changes.

Suggested Answer: *CD*

Currently there are no comments in this discussion, be the first to comment!

Question #13 *Topic 1*

DRAG DROP -
You have a Microsoft SQL Server 2025 database that contains a table named dbo.CustomerMessages. dbo.CustomerMessages contains two columns named MessageID (int) and MessageRaw (nvarchar(max)).
MessageRaw can contain a phone number in multiple formats, and some rows do NOT contain a phone number.

You need to write a single SELECT query that meets the following requirements:
The query must return MessageID, RawNumber, DigitsOnly, and PhoneStatus.
RawNumber must contain the first substring that matches a phone-number pattern, or NULL if no match exists.

DigitsOnly must remove all non-digit characters from RawNumber, or return NULL.
PhoneStatus must return valid when a phone number exists in MessageRaw, otherwise return Missing.

How should you complete the Transact-SQL query? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.

Suggested Answer:

Currently there are no comments in this discussion, be the first to comment!

Question #14 *Topic 1*

You have an Azure SQL database that contains a table named Rooms. Rooms was created by using the following Transact-SQL statement.

```
CREATE TABLE Rooms
(
    RoomID int PRIMARY KEY,
    Owner nvarchar(100),
    Capacity int
);
```

You discover that some records in the Rooms table contain NULL values for the Owner field.

You need to ensure that all future records have a value for the Owner field.
What should you add?

A. a foreign key

B. a check constraint

C. a nonclustered index

D. a unique constraint

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

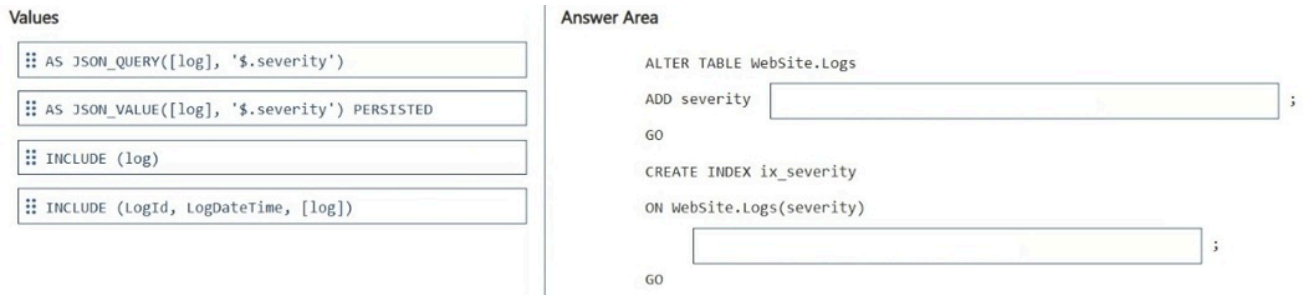
Question #15 *Topic 1*

DRAG DROP -
You have a SQL database in Microsoft Fabric that contains a table named WebSite.Logs. WebSite.Logs stores application telemetry data. WebSite.Logs contains a nvarchar (max) column named log that stores JSON documents.

You have a daily report that filters by the $.severity JSON property and returns LogId, LogDateTime, and log. The report frequently causes full table scans.
You need to modify WebSite.Logs to support efficient filtering by $.severity and avoid key lookups for the columns returned by the report.

How should you complete the Transact-SQL code to avoid full table scans? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.



Suggested Answer:

Currently there are no comments in this discussion, be the first to comment!

Question #1 *Topic 2*

HOTSPOT -

Case Study -
This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study -
To display the first question in this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Existing Environment -

Azure Environment -

Contoso has an Azure subscription in North Europe that contains the corporate infrastructure. The current infrastructure contains a Microsoft SQL Server 2017 database. The database contains the following tables.

Table names	Column names
CustomerFeedback	• FeedbackId (int) (primarykey) • FeedbackJson (nvarchar (max))
Fleets	• FleetId (int) (primarykey) • FleetName (nvarchar(100)) • Description (nvarchar(256))
MaintenanceEvents	• MaintenanceId (int) (primarykey) • VehicleId (int) • LastModifiedUTC (datetime2) • Description (nvarchar(256))
SupportTickets	• TicketId (int) (primarykey) • FleetId (int) • CreatedUtc (datetime2)
UserAccounts	• UserId (int) (primarykey) • UserPrincipalName (nvarchar(256)) • JobRole (nvarchar(256)) • StartDate (datetime2)
VehicleIncidentReports	• IncidentId (int) (primarykey) • VehicleId (int) • FleetId (int) • IncidentType (nvarchar(50)) • VehicleLocation (nvarchar(200)) • IncidentDescription (nvarchar(max)) • SeverityScore (int)
Vehicles	• VehicleId (int) (primarykey) • VIN ((nvarchar(50)) • VehicleDescription (nvarchar(256))
VehicleHealthSummary	• VehicleId (int) (primarykey) • FleetId (int) • Summary (nvarchar(2000)) • LastUpdatedUtc (datetime2) • EngineStatus [bit] • EngineStatusLastUpdatedUtc (datetime2) • BatteryHealth (int) • Embeddings (vector (1536))

The FeedbackJson column has a full-text index and stores JSON documents in the following format.

```
{
  "text": "The battery drains too fast when driving uphill.",
  "category": "Battery",
  "metadata": {
    "appVersion": "5.2.1",
    "device": "Android",
    "language": "en-GB"
  }
}
```

The support staff at Contoso never has the UNMASK permission.

Problem Statements -
Contoso is deploying a new Azure SQL database that will become the authoritative data store for the following:

AI workloads -

Vector search -

Modernized API access -
Retrieval Augmented Generation (RAG) pipelines

Sometimes the ingestion pipeline fails due to malformed JSON and duplicate payloads.

The engineers at Contoso report that the following dashboard query runs slowly.

```
SELECT VehicleId, LastUpdatedUtc, EngineStatus, BatteryHealth
FROM dbo.VehicleHealthSummary
WHERE FleetId = @FleetId
ORDER BY LastUpdatedUtc DESC;
```

You review the execution plan and discover that the plan shows a clustered index scan.
VehicleIncidentReports often contains details about the weather, traffic conditions, and location. Analysts report that it is difficult to find similar incidents based on these details.

Requirements -

Planned Changes -

Contoso wants to modernize Fleet Intelligence Platform to support AI-powered semantic search over incident reports.

Security Requirements -
Contoso identifies the following security requirements:
Restrict the support staff from viewing Personally Identifiable Information (PII) data, which is full email addresses and phone numbers.

Enforce row-level filtering so that analysts see only incidents for the fleets to which they are assigned. The analysts can be assigned to multiple fleets.

Database Performance and Requirements
Contoso identifies the following telemetry requirements:
Telemetry data must be stored in a partitioned table.

Telemetry data must provide predictable performance for ingestion and retention operations. latitude, longitude, and accuracy JSON properties must be filtered by using an index seek.
Contoso identifies the following maintenance data requirements:

Ensure that any changes to a row in the MaintenanceEvents table updates the corresponding value in the LastModifiedUtc column to the time of the change.

Avoid recursive updates.

AI Search, Embeddings, and Vector Indexing
Contoso plans to implement semantic search over incident data to meet the following requirements:
Embeddings must be stored in dedicated Azure SQL Database tables.

Embeddings must be generated from rich natural language fields.

Chunking must preserve semantic coherence.
Hybrid search must combine the following:

Vector similarity -

Keyword filtering or boosting -

Development Requirements -
The development team at Contoso will use Microsoft Visual Studio Code and GitHub Copilot and will retrieve live metadata from the databases.
Contoso identifies the following requirements for querying data in the FeedbackJson column of the CustomerFeedback table:

Extract the customer feedback text from the JSON document.
Filter rows where the JSON text contains a keyword.

Calculate a fuzzy similarity score between the feedback text and a known issue description.
Order the results by similarity score, with the highest score first.
You need to create a table in the database to store the telemetry data.
You have the following Transact-SQL code.

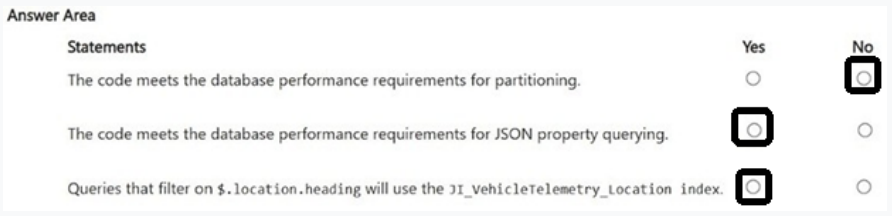
```
CREATE TABLE dbo.VehicleTelemetry
(
  TelemetryId BIGINT IDENTITY(1,1) NOT NULL,
  VehicleId NVARCHAR(50) NOT NULL,
  TelemetryTimeUtc DATETIME2(3) NOT NULL,
  BatteryPercent TINYINT NULL,
  SpeedKmh SMALLINT NULL,
  LocationJson JSON NULL,
  ErrorCodesJson JSON NULL,
  RawPayload NVARCHAR(MAX) NULL,
  SysStartTime DATETIME2(7) GENERATED ALWAYS AS ROW START NOT NULL,
  SysEndTime DATETIME2(7) GENERATED ALWAYS AS ROW END NOT NULL,
  PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime),
  CONSTRAINT PK_VehicleTelemetry PRIMARY KEY CLUSTERED (TelemetryId)
)
WITH
(
  SYSTEM_VERSIONING = ON
  (
    HISTORY_TABLE = dbo.VehicleTelemetryHistory
  )
);
GO
CREATE INDEX IX_VehicleTelemetry_Time ON dbo.VehicleTelemetry (TelemetryTimeUtc);
CREATE JSON INDEX JI_VehicleTelemetry_Location ON dbo.VehicleTelemetry (LocationJson) FOR
(
  '$.location. latitude', '$.location. longitude',
  '$.location. accuracy'
);
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.
NOTE: Each correct selection is worth one point.

Answer Area

Statements	Yes	No
The code meets the database performance requirements for partitioning.	○	○
The code meets the database performance requirements for JSON property querying.	○	○
Queries that filter on `$.location.heading` will use the `JI_VehicleTelemetry_Location` index.	○	○

Suggested Answer:



Currently there are no comments in this discussion, be the first to comment!

Question #2 *Topic 2*

HOTSPOT -

Case Study -
This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study -
To display the first question in this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Existing Environment -

Azure Environment -
Contoso has an Azure subscription in North Europe that contains the corporate infrastructure. The current infrastructure contains a Microsoft SQL Server 2017 database. The database contains the following tables.

Table names	Column names
CustomerFeedback	• FeedbackId (int) (primarykey) • FeedbackJson (nvarchar (max))
Fleets	• FleetId (int) (primarykey) • FleetName (nvarchar(100)) • Description (nvarchar(256))
MaintenanceEvents	• MaintenanceId (int) (primarykey) • VehicleId (int) • LastModifiedUTC (datetime2) • Description (nvarchar(256))
SupportTickets	• TicketId (int) (primarykey) • FleetId (int) • CreatedUtc (datetime2)
UserAccounts	• UserId (int) (primarykey) • UserPrincipalName (nvarchar(256)) • JobRole (nvarchar(256)) • StartDate (datetime2)
VehicleIncidentReports	• IncidentId (int) (primarykey) • VehicleId (int) • FleetId (int) • IncidentType (nvarchar(50)) • VehicleLocation (nvarchar(200)) • IncidentDescription (nvarchar(max)) • SeverityScore (int)
Vehicles	• VehicleId (int) (primarykey) • VIN ((nvarchar(50)) • VehicleDescription (nvarchar(256))
VehicleHealthSummary	• VehicleId (int) (primarykey) • FleetId (int) • Summary (nvarchar(2000)) • LastUpdatedUtc (datetime2) • EngineStatus [bit] • EngineStatusLastUpdatedUtc (datetime2) • BatteryHealth (int) • Embeddings (vector (1536))

The FeedbackJson column has a full-text index and stores JSON documents in the following format.

```
{
  "text": "The battery drains too fast when driving uphill.",
  "category": "Battery",
  "metadata": {
    "appVersion": "5.2.1",
    "device": "Android",
    "language": "en-GB"
  }
}
```

The support staff at Contoso never has the UNMASK permission.

Problem Statements -
Contoso is deploying a new Azure SQL database that will become the authoritative data store for the following:

AI workloads -

Vector search -

Modernized API access -
Retrieval Augmented Generation (RAG) pipelines
Sometimes the ingestion pipeline fails due to malformed JSON and duplicate payloads.

The engineers at Contoso report that the following dashboard query runs slowly.

```
SELECT VehicleId, LastUpdatedUtc, EngineStatus, BatteryHealth
FROM dbo.VehicleHealthSummary
WHERE FleetId = @FleetId
ORDER BY LastUpdatedUtc DESC;
```

You review the execution plan and discover that the plan shows a clustered index scan.

VehicleIncidentReports often contains details about the weather, traffic conditions, and location. Analysts report that it is difficult to find similar incidents based on these details.

Requirements -

Planned Changes -
Contoso wants to modernize Fleet Intelligence Platform to support AI-powered semantic search over incident reports.

Security Requirements -
Contoso identifies the following security requirements:

Restrict the support staff from viewing Personally Identifiable Information (PII) data, which is full email addresses and phone numbers.
Enforce row-level filtering so that analysts see only incidents for the fleets to which they are assigned. The analysts can be assigned to multiple fleets.

Database Performance and Requirements
Contoso identifies the following telemetry requirements:
Telemetry data must be stored in a partitioned table.
Telemetry data must provide predictable performance for ingestion and retention operations.

latitude, longitude, and accuracy JSON properties must be filtered by using an index seek.

Contoso identifies the following maintenance data requirements:
Ensure that any changes to a row in the MaintenanceEvents table updates the corresponding value in the LastModifiedUtc column to the time of the change.

Avoid recursive updates.

AI Search, Embeddings, and Vector Indexing
Contoso plans to implement semantic search over incident data to meet the following requirements:

Embeddings must be stored in dedicated Azure SQL Database tables.
Embeddings must be generated from rich natural language fields.
Chunking must preserve semantic coherence.

Hybrid search must combine the following:

Vector similarity -

Keyword filtering or boosting -

Development Requirements -

The development team at Contoso will use Microsoft Visual Studio Code and GitHub Copilot and will retrieve live metadata from the databases.
Contoso identifies the following requirements for querying data in the FeedbackJson column of the CustomerFeedback table:
Extract the customer feedback text from the JSON document.
Filter rows where the JSON text contains a keyword.
Calculate a fuzzy similarity score between the feedback text and a known issue description.

Order the results by similarity score, with the highest score first.
You are creating a table that will store customer profiles.
You have the following Transact-SQL code.

```
CREATE TABLE dbo.CustomerProfiles
(
    CustomerId     BIGINT IDENTITY(1,1) PRIMARY KEY,
    FullName       NVARCHAR(200) MASKED WITH (FUNCTION = 'partial(1,"xxxx",1)'),
    EmailAddress   NVARCHAR(200) MASKED WITH (FUNCTION = 'email()'),
    PhoneNumber    NVARCHAR(50)  MASKED WITH (FUNCTION = 'default()'),
    RegionCode     NVARCHAR(10) NOT NULL
);
GO
CREATE FUNCTION dbo.fn_FilterByRegion(@RegionCode NVARCHAR(10))
RETURNS TABLE
AS
RETURN
(
    SELECT 1
    FROM dbo.UserRegionAccess ura
    WHERE ura.UserPrincipalName = SUSER_SNAME()
      AND ura.RegionCode = @RegionCode
);
GO
CREATE SECURITY POLICY CustomerRegionPolicy
ADD FILTER PREDICATE dbo.fn_FilterByRegion(RegionCode)
ON dbo.CustomerProfiles
WITH (STATE = ON);
GO
```

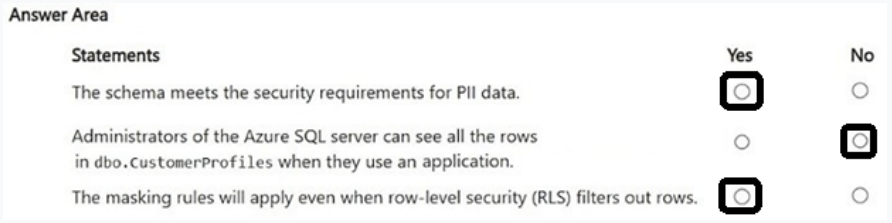
For each of the following statements, select Yes if the statement is true. Otherwise, select No.

NOTE: Each correct selection is worth one point.

Answer Area

Statements	Yes	No
The schema meets the security requirements for PII data.	○	○
Administrators of the Azure SQL server can see all the rows in `dbo.CustomerProfiles` when they use an application.	○	○
The masking rules will apply even when row-level security (RLS) filters out rows.	○	○

Suggested Answer:



Currently there are no comments in this discussion, be the first to comment!

Question #3 *Topic 2*

DRAG DROP -

Case Study -

This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.
At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study -
To display the first question in this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Existing Environment -

Azure Environment -
Contoso has an Azure subscription in North Europe that contains the corporate infrastructure. The current infrastructure contains a Microsoft SQL Server 2017 database. The database contains the following tables.

Table names	Column names
CustomerFeedback	• FeedbackId (int) (primarykey) • FeedbackJson (nvarchar (max))
Fleets	• FleetId (int) (primarykey) • FleetName (nvarchar(100)) • Description (nvarchar(256))
MaintenanceEvents	• MaintenanceId (int) (primarykey) • VehicleId (int) • LastModifiedUTC (datetime2) • Description (nvarchar(256))
SupportTickets	• TicketId (int) (primarykey) • FleetId (int) • CreatedUtc (datetime2)
UserAccounts	• UserId (int) (primarykey) • UserPrincipalName (nvarchar(256)) • JobRole (nvarchar(256)) • StartDate (datetime2)
VehicleIncidentReports	• IncidentId (int) (primarykey) • VehicleId (int) • FleetId (int) • IncidentType (nvarchar(50)) • VehicleLocation (nvarchar(200)) • IncidentDescription (nvarchar(max)) • SeverityScore (int)
Vehicles	• VehicleId (int) (primarykey) • VIN ((nvarchar(50)) • VehicleDescription (nvarchar(256))
VehicleHealthSummary	• VehicleId (int) (primarykey) • FleetId (int) • Summary (nvarchar(2000)) • LastUpdatedUtc (datetime2) • EngineStatus [bit] • EngineStatusLastUpdatedUtc (datetime2) • BatteryHealth (int) • Embeddings (vector (1536))

The FeedbackJson column has a full-text index and stores JSON documents in the following format.

```
{
  "text": "The battery drains too fast when driving uphill.",
  "category": "Battery",
  "metadata": {
    "appVersion": "5.2.1",
    "device": "Android",
    "language": "en-GB"
  }
}
```

The support staff at Contoso never has the UNMASK permission.

Problem Statements -
Contoso is deploying a new Azure SQL database that will become the authoritative data store for the following:

AI workloads -

Vector search -

Modernized API access -
Retrieval Augmented Generation (RAG) pipelines
Sometimes the ingestion pipeline fails due to malformed JSON and duplicate payloads.

The engineers at Contoso report that the following dashboard query runs slowly.

```
SELECT VehicleId, LastUpdatedUtc, EngineStatus, BatteryHealth
FROM dbo.VehicleHealthSummary
WHERE FleetId = @FleetId
ORDER BY LastUpdatedUtc DESC;
```

You review the execution plan and discover that the plan shows a clustered index scan.

VehicleIncidentReports often contains details about the weather, traffic conditions, and location. Analysts report that it is difficult to find similar incidents based on these details.

Requirements -

Planned Changes -
Contoso wants to modernize Fleet Intelligence Platform to support AI-powered semantic search over incident reports.

Security Requirements -
Contoso identifies the following security requirements:

Restrict the support staff from viewing Personally Identifiable Information (PII) data, which is full email addresses and phone numbers.
Enforce row-level filtering so that analysts see only incidents for the fleets to which they are assigned. The analysts can be assigned to multiple fleets.
Database Performance and Requirements
Contoso identifies the following telemetry requirements:
Telemetry data must be stored in a partitioned table.

Telemetry data must provide predictable performance for ingestion and retention operations. latitude, longitude, and accuracy JSON properties must be filtered by using an index seek.

Contoso identifies the following maintenance data requirements:
Ensure that any changes to a row in the MaintenanceEvents table updates the corresponding value in the LastModifiedUtc column to the time of the change.

Avoid recursive updates.
AI Search, Embeddings, and Vector Indexing
Contoso plans to implement semantic search over incident data to meet the following requirements:

Embeddings must be stored in dedicated Azure SQL Database tables.
Embeddings must be generated from rich natural language fields.
Chunking must preserve semantic coherence.
Hybrid search must combine the following:

Vector similarity -

Keyword filtering or boosting -

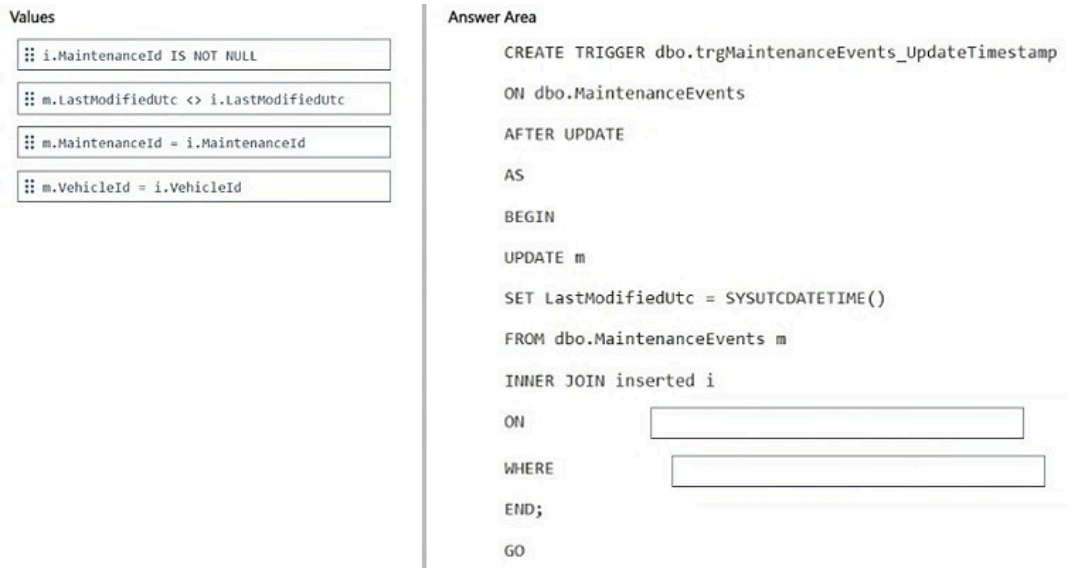
Development Requirements -
The development team at Contoso will use Microsoft Visual Studio Code and GitHub Copilot and will retrieve live metadata from the databases.
Contoso identifies the following requirements for querying data in the FeedbackJson column of the CustomerFeedback table:

Extract the customer feedback text from the JSON document.
Filter rows where the JSON text contains a keyword.
Calculate a fuzzy similarity score between the feedback text and a known issue description.

Order the results by similarity score, with the highest score first.
You need to meet the database performance requirements for maintenance data.

How should you complete the Transact-SQL code? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.



Suggested Answer:

Answer Area

```
CREATE TRIGGER dbo.trgMaintenanceEvents_UpdateTimestamp
ON dbo.MaintenanceEvents
AFTER UPDATE
AS
BEGIN
UPDATE m
SET LastModifiedUtc = SYSUTCDATETIME()
FROM dbo.MaintenanceEvents m
INNER JOIN inserted i
ON    m.VehicleId = i.VehicleId
WHERE    m.LastModifiedUtc <> i.LastModifiedUtc
END;
GO
```

Currently there are no comments in this discussion, be the first to comment!

Question #4 *Topic 2*

Case Study -
This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study -
To display the first question in this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Existing Environment -

Azure Environment -
Contoso has an Azure subscription in North Europe that contains the corporate infrastructure. The current infrastructure contains a Microsoft SQL Server 2017 database. The database contains the following tables.

Table names	Column names
CustomerFeedback	• FeedbackId (int) (primarykey) • FeedbackJson (nvarchar (max))
Fleets	• FleetId (int) (primarykey) • FleetName (nvarchar(100)) • Description (nvarchar(256))
MaintenanceEvents	• MaintenanceId (int) (primarykey) • VehicleId (int) • LastModifiedUTC (datetime2) • Description (nvarchar(256))
SupportTickets	• TicketId (int) (primarykey) • FleetId (int) • CreatedUtc (datetime2)
UserAccounts	• UserId (int) (primarykey) • UserPrincipalName (nvarchar(256)) • JobRole (nvarchar(256)) • StartDate (datetime2)
VehicleIncidentReports	• IncidentId (int) (primarykey) • VehicleId (int) • FleetId (int) • IncidentType (nvarchar(50)) • VehicleLocation (nvarchar(200)) • IncidentDescription (nvarchar(max)) • SeverityScore (int)
Vehicles	• VehicleId (int) (primarykey) • VIN ((nvarchar(50)) • VehicleDescription (nvarchar(256))
VehicleHealthSummary	• VehicleId (int) (primarykey) • FleetId (int) • Summary (nvarchar(2000)) • LastUpdatedUtc (datetime2) • EngineStatus [bit] • EngineStatusLastUpdatedUtc (datetime2) • BatteryHealth (int) • Embeddings (vector (1536))

The FeedbackJson column has a full-text index and stores JSON documents in the following format.

```
{
  "text": "The battery drains too fast when driving uphill.",
  "category": "Battery",
  "metadata": {
    "appVersion": "5.2.1",
    "device": "Android",
    "language": "en-GB"
  }
}
```

The support staff at Contoso never has the UNMASK permission.

Problem Statements -

Contoso is deploying a new Azure SQL database that will become the authoritative data store for the following:

AI workloads -

Vector search -

Modernized API access -
Retrieval Augmented Generation (RAG) pipelines
Sometimes the ingestion pipeline fails due to malformed JSON and duplicate payloads.
The engineers at Contoso report that the following dashboard query runs slowly.

```
SELECT VehicleId, LastUpdatedUtc, EngineStatus, BatteryHealth
FROM dbo.VehicleHealthSummary
WHERE FleetId = @FleetId
ORDER BY LastUpdatedUtc DESC;
```

You review the execution plan and discover that the plan shows a clustered index scan.
VehicleIncidentReports often contains details about the weather, traffic conditions, and location. Analysts report that it is difficult to find similar incidents based on these details.

Requirements -

Planned Changes -
Contoso wants to modernize Fleet Intelligence Platform to support AI-powered semantic search over incident reports.

Security Requirements -
Contoso identifies the following security requirements:
Restrict the support staff from viewing Personally Identifiable Information (PII) data, which is full email addresses and phone numbers.
Enforce row-level filtering so that analysts see only incidents for the fleets to which they are assigned. The analysts can be assigned to multiple fleets.

Database Performance and Requirements
Contoso identifies the following telemetry requirements:
Telemetry data must be stored in a partitioned table.

Telemetry data must provide predictable performance for ingestion and retention operations. latitude, longitude, and accuracy JSON properties must be filtered by using an index seek.
Contoso identifies the following maintenance data requirements:
Ensure that any changes to a row in the MaintenanceEvents table updates the corresponding value in the LastModifiedUtc column to the time of the change.
Avoid recursive updates.

AI Search, Embeddings, and Vector Indexing

Contoso plans to implement semantic search over incident data to meet the following requirements:

Embeddings must be stored in dedicated Azure SQL Database tables.
Embeddings must be generated from rich natural language fields.
Chunking must preserve semantic coherence.
Hybrid search must combine the following:

Vector similarity -

Keyword filtering or boosting -

Development Requirements -
The development team at Contoso will use Microsoft Visual Studio Code and GitHub Copilot and will retrieve live metadata from the databases.

Contoso identifies the following requirements for querying data in the FeedbackJson column of the CustomerFeedback table:

Extract the customer feedback text from the JSON document.
Filter rows where the JSON text contains a keyword.

Calculate a fuzzy similarity score between the feedback text and a known issue description.
Order the results by similarity score, with the highest score first.

You need to recommend a solution to resolve the slow dashboard query issue.
What should you recommend?

A. Create a clustered index on LastUpdatedUtc.

B. On FleetId, create a nonclustered index that includes LastUpdatedUtc, EngineStatus, and BatteryHealth.

C. On LastUpdatedUtc, create a nonclustered index that includes FleetId.

D. On FleetId, create a filtered index where LastUpdatedUtc > DATEADD(DAY, -7, SYSUTCDATETIME()).

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #5 *Topic 2*

Case Study -
This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study -
To display the first question in this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Existing Environment -

Azure Environment -
Contoso has an Azure subscription in North Europe that contains the corporate infrastructure. The current infrastructure contains a Microsoft SQL Server 2017 database. The database contains the following tables.

Table names	Column names
CustomerFeedback	• FeedbackId (int) (primarykey) • FeedbackJson (nvarchar (max))
Fleets	• FleetId (int) (primarykey) • FleetName (nvarchar(100)) • Description (nvarchar(256))
MaintenanceEvents	• MaintenanceId (int) (primarykey) • VehicleId (int) • LastModifiedUTC (datetime2) • Description (nvarchar(256))
SupportTickets	• TicketId (int) (primarykey) • FleetId (int) • CreatedUtc (datetime2)
UserAccounts	• UserId (int) (primarykey) • UserPrincipalName (nvarchar(256)) • JobRole (nvarchar(256)) • StartDate (datetime2)
VehicleIncidentReports	• IncidentId (int) (primarykey) • VehicleId (int) • FleetId (int) • IncidentType (nvarchar(50)) • VehicleLocation (nvarchar(200)) • IncidentDescription (nvarchar(max)) • SeverityScore (int)
Vehicles	• VehicleId (int) (primarykey) • VIN ((nvarchar(50)) • VehicleDescription (nvarchar(256))
VehicleHealthSummary	• VehicleId (int) (primarykey) • FleetId (int) • Summary (nvarchar(2000)) • LastUpdatedUtc (datetime2) • EngineStatus [bit] • EngineStatusLastUpdatedUtc (datetime2) • BatteryHealth (int) • Embeddings (vector (1536))

The FeedbackJson column has a full-text index and stores JSON documents in the following format.

```
{
  "text": "The battery drains too fast when driving uphill.",
  "category": "Battery",
  "metadata": {
    "appVersion": "5.2.1",
    "device": "Android",
    "language": "en-GB"
  }
}
```

The support staff at Contoso never has the UNMASK permission.

Problem Statements -
Contoso is deploying a new Azure SQL database that will become the authoritative data store for the following:

AI workloads -

Vector search -

Modernized API access -
Retrieval Augmented Generation (RAG) pipelines
Sometimes the ingestion pipeline fails due to malformed JSON and duplicate payloads.
The engineers at Contoso report that the following dashboard query runs slowly.

```
SELECT VehicleId, LastUpdatedUtc, EngineStatus, BatteryHealth
FROM dbo.VehicleHealthSummary
WHERE FleetId = @FleetId
ORDER BY LastUpdatedUtc DESC;
```

You review the execution plan and discover that the plan shows a clustered index scan.
VehicleIncidentReports often contains details about the weather, traffic conditions, and location. Analysts report that it is difficult to find similar incidents based on these details.

Requirements -

Planned Changes -
Contoso wants to modernize Fleet Intelligence Platform to support AI-powered semantic search over incident reports.

Security Requirements -
Contoso identifies the following security requirements:

Restrict the support staff from viewing Personally Identifiable Information (PII) data, which is full email addresses and phone numbers.
Enforce row-level filtering so that analysts see only incidents for the fleets to which they are assigned. The analysts can be assigned to multiple fleets.

Database Performance and Requirements
Contoso identifies the following telemetry requirements:
Telemetry data must be stored in a partitioned table.
Telemetry data must provide predictable performance for ingestion and retention operations. latitude, longitude, and accuracy JSON properties must be filtered by using an index seek.

Contoso identifies the following maintenance data requirements:
Ensure that any changes to a row in the MaintenanceEvents table updates the corresponding value in the LastModifiedUtc column to the time of the change.
Avoid recursive updates.

AI Search, Embeddings, and Vector Indexing
Contoso plans to implement semantic search over incident data to meet the following requirements:
Embeddings must be stored in dedicated Azure SQL Database tables.

Embeddings must be generated from rich natural language fields.
Chunking must preserve semantic coherence.
Hybrid search must combine the following:

Vector similarity -

Keyword filtering or boosting -

Development Requirements -
The development team at Contoso will use Microsoft Visual Studio Code and GitHub Copilot and will retrieve live metadata from the databases.
Contoso identifies the following requirements for querying data in the FeedbackJson column of the CustomerFeedback table:
Extract the customer feedback text from the JSON document.
Filter rows where the JSON text contains a keyword.
Calculate a fuzzy similarity score between the feedback text and a known issue description.

Order the results by similarity score, with the highest score first.
You need to recommend a solution that will resolve the ingestion pipeline failure issues.
Which two actions should you recommend? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Enable snapshot isolation on the database.

B. Use a trigger to automatically rewrite malformed JSON.

C. Add foreign key constraints on the table.

D. Create a unique index on a hash of the payload.

E. Add a check constraint that validates the JSON structure.

Suggested Answer: *DE*

Currently there are no comments in this discussion, be the first to comment!

Question #6 *Topic 2*

You have a Microsoft SQL Server 2025 instance that has a managed identity enabled.
You have a database that contains a table named dbo.ManualChunks. dbo.ManualChunks contains product manuals.

A retrieval query already returns the top five matching chunks as nvarchar(max) text.
You need to call an Azure OpenAI REST endpoint for chat completions. The solution must provide the highest level of security.

You write the following Transact-SQL code.

```
01  CREATE DATABASE SCOPED CREDENTIAL AzureOpenAIHeaders
02
03  GO
04  CREATE OR ALTER PROCEDURE dbo.AskManuals
05    ...
06    SELECT @chunks =
07    ...
08    SET @payload =
09    ...
10    EXEC @retval = sys.sp_invoke_external_rest_endpoint
11      @url = N'https://<your-resource>.openai.azure.com/openai/deployments/<your-deployment>/chat/completions?api-version=2024-02-15-preview',
12      @method = N'POST',
13      @credential = N'AzureOpenAIHeaders',
14      @payload = @payload,
15      @response = @response OUTPUT;
16  END;
17  GO
```

What should you insert at line 02?

A.
```
WITH IDENTITY = 'HTTPEndpointQueryString',
SECRET = N'{"api-key":"<value>"}';
```

B.
```
WITH IDENTITY = 'Managed Identity',
SECRET = '{"resourceid":"<value>"}';
```

C.
```
WITH IDENTITY = 'Managed Identity',
SECRET = N'{"api-key":"<value>"}';
```

D.
```
WITH IDENTITY = 'HTTPEndpointHeaders',
SECRET = N'{"api-key":"<value>"}';
```

E.
```
WITH IDENTITY = 'AzureOpenAI',
SECRET = N'{"resourceid":"<value>"}';
```

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #7 *Topic 2*

You have an Azure SQL database named SalesDB that contains a table named dbo.Articles. dbo.Articles contains two million articles with embeddings. The articles are updated frequently throughout the day.
You query the embeddings by using VECTOR_SEARCH.
Users report that semantic search results do NOT reflect the updates until the following day.
You need to ensure that the embeddings are updated whenever the articles change. The solution must minimize CPU usage on SalesDB.
Which embedding maintenance method should you implement?

A. Modify the query to use VECTOR_DISTANCE instead of VECTOR_SEARCH.

B. Enable change data capture (CDC) on dbo.Articles and use an Azure Functions app to process CDC changes.

C. Run an hourly Transact-SQL job that regenerates embeddings for all the rows in dbo.Articles.

D. On dbo.Articles, create a trigger that calls AI_GENERATE_EMBEDDINGS for each inserted or updated row.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #8 *Topic 2*

You have a GitHub Codespaces environment that has GitHub Copilot Chat installed and is connected to a SQL database in Microsoft Fabric named DB1. DB1 contains tables named Sales.Orders and Sales.Customers.
You use GitHub Copilot Chat in the context of DB1.
A company policy prohibits sharing customer Personally Identifiable Information (PII), secrets, and query result sets with any AI service.
You need to use GitHub Copilot Chat to write and review Transact-SQL code for a new stored procedure that will join Sales.Orders to Sales.Customers and return customer names and email addresses. The solution must NOT share the actual data in the tables with GitHub Copilot Chat.
What should you do?

A. From Sales.Customers, paste several rows that include email addresses into a chat, so that GitHub Copilot Chat can infer edge cases.

B. Run a SELECT statement that returns customer names and email addresses and provide the result set to GitHub Copilot Chat so that GitHub Copilot Chat can generate the stored procedure.

C. Provide the database connection string to GitHub Copilot Chat so that GitHub Copilot Chat can validate the stored procedure.

D. Ask GitHub Copilot Chat to generate the stored procedure by using schema details only.

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #9 *Topic 2*

HOTSPOT -

You have an Azure subscription. The subscription contains an Azure SQL database named SalesDB and an Azure App Service app named sales-api. sales-api uses virtual network integration to a subnet named vnet-prod/subnet-app and reads from SalesDB.

You need to configure authentication and network access to meet the following requirements:

Ensure that sales-api connects to SalesDB by using passwordless authentication.

Ensure that all the database traffic remains within the subscription.

The solution must minimize administrative effort.

What should you configure? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Answer Area

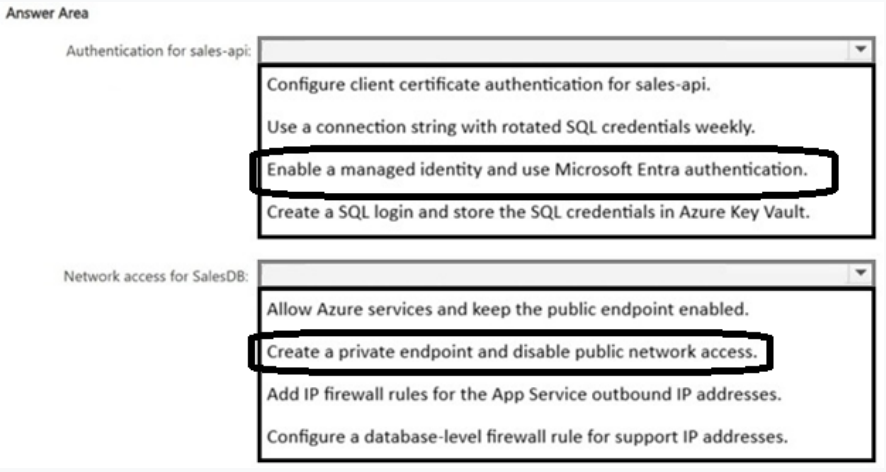
Authentication for sales-api:

- Configure client certificate authentication for sales-api.
- Use a connection string with rotated SQL credentials weekly.
- Enable a managed identity and use Microsoft Entra authentication.
- Create a SQL login and store the SQL credentials in Azure Key Vault.

Network access for SalesDB:

- Allow Azure services and keep the public endpoint enabled.
- Create a private endpoint and disable public network access.
- Add IP firewall rules for the App Service outbound IP addresses.
- Configure a database-level firewall rule for support IP addresses.

Suggested Answer:



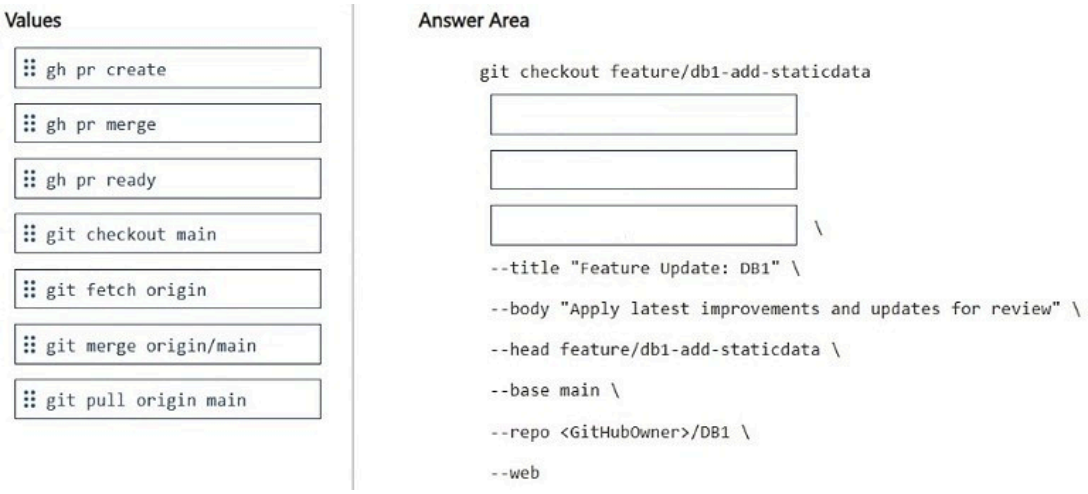
Currently there are no comments in this discussion, be the first to comment!

Question #10 *Topic 2*

DRAG DROP -
You have a database named DB1. The schema is stored in a GitHub repository as an SDK-style SQL database project.

You use a feature branch workflow to deploy changes to DB1.
You need to update the local feature branch with the latest changes to main, and then create a pull request to merge the feature branch into main for review.

How should you complete the GitHub CLI script? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Suggested Answer:

Currently there are no comments in this discussion, be the first to comment!

Question #11 *Topic 2*

You have an Azure SQL database that contains tables named dbo.ProductDocs and dbo.ProductDocsEmbeddings. dbo.ProductDocs contains product documentation and the following columns:
DocID (int)
Title (nvarchar(200))
Body (nvarchar(max))
LastModified (datetime2)
The documentation is edited throughout the day.
dbo.ProductDocsEmbeddings contains the following columns:
DocID (int)
ChunkOrder (int)
ChunkText (nvarchar(max))
Embedding (vector(1536))
The current embedding pipeline runs once per night.
You need to ensure that embeddings are updated every time the underlying documentation content changes. The solution must NOT require a nightly batch process.
What should you include in the solution?

A. fixed-size chunking

B. a smaller embedding model

C. table triggers

D. change tracking on dbo.ProductDocs

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!