Microsoft DP-800 Mock Exam

Shared by Barlow on 17-06-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

You have an Azure SQL database that contains a column named Notes.

A security review discovers that Notes contains sensitive data.

You need to ensure that the data is protected so that neither the stored values nor the query inputs reveal information about the actual data. The solution must prevent a user from inferring relationships or repetitions in the data based on the encrypted output

Which should you use?

Options:

A- Always Encrypted with secure enclaves
B- Always Encrypted with randomized encryption
C- row-level security <RLS)
D- Always Encrypted with deterministic encryption

Answer:

B

Explanation:

The requirement says the stored values and query inputs must both be protected, and users must not be able to infer relationships or repetitions in the data from the encrypted output. Microsoft documents that deterministic encryption always produces the same ciphertext for the same plaintext, which allows equality comparisons but also leaks patterns. By contrast, randomized encryption produces a different encrypted value each time for the same plaintext, which improves security and prevents pattern analysis based on repeated ciphertext values.

That makes randomized encryption the right choice here:

It protects data at rest and in transit/query parameters under Always Encrypted's client-side encryption model.

It prevents attackers from learning that the same plaintext value appears repeatedly, because repeated inputs do not produce repeated ciphertext.

Why the other options are wrong:

A . Always Encrypted with secure enclaves adds richer confidential query support, but the key

protection property the question is testing is the encryption type. The requirement to prevent inference from repeated ciphertext points specifically to randomized encryption.

C . RLS controls row access, not value confidentiality.

D . Deterministic encryption allows equality-based operations but leaks repetition patterns, which the question explicitly forbids.

Question 2

Question Type: MultipleChoice

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?

Options:

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-instruetions.md

Answer:

A

Explanation:

GitHub documents that repository custom instructions for Copilot Chat can be disabled for your own use in the editor settings, and that doing so does not affect other users. In VS Code, this is controlled through settings related to instruction files, where you can disable the use of repository instruction files for your own environment.

The other options are incorrect:

B is not a documented mechanism for disabling repository-level Copilot instructions.

C would remove the repository instruction file itself and therefore affect everyone using that repository, which violates the requirement.

Question 3

Question Type: MultipleChoice

Vou have an Azure SQL database named SalesDB that contains a table named dbo. Articles, dbo.Articles contains two million articles with embeddmgs. The articles are updated frequently throughout the day.

You query the embeddings by using VECTOR_SEARQi

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?

Options:

A- Modify the query to use VECTOR.DTSTANCF instead of VECTOR.SEARCK
B- enable change data capture (COC) on dbo.Articles and use an Azure Functions app to process CLX 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 EMBEDOINGS for each inserted or updated row.

Answer:

B

Explanation:

The correct answer is B because the problem is not the vector search operator itself. The problem is that embeddings are becoming stale when article content changes. Microsoft documents that change data capture (CDC) tracks insert, update, and delete operations on source tables, which makes it the right mechanism to identify only the rows that changed.

This also best satisfies the requirement to minimize CPU usage on SalesDB. With CDC, the database only records the row changes, and the embedding regeneration work can be moved to an external process such as an Azure Functions app. That avoids running embedding generation inline inside the database for every update and avoids repeatedly recalculating embeddings for unchanged rows. In contrast, an hourly full-table regeneration would be extremely wasteful on a table with two million frequently updated articles, and a trigger that calls embedding generation

per row would push expensive AI work into the transactional path of the database.

Option A is incorrect because changing from VECTOR_SEARCH to VECTOR_DISTANCE does not regenerate embeddings; it only changes the retrieval method. Microsoft states that VECTOR_SEARCH is the ANN search function, while VECTOR_DISTANCE performs exact distance calculation, so neither option addresses stale embedding data.

So the right design is:

use CDC to detect only changed articles,

process those changes outside the database,

regenerate embeddings only for changed rows,

write back the refreshed embeddings for current semantic search results.

Question 4

Question Type: MultipleChoice

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?

Options:

A- a foreign key
B- a check constraint
C- a nonclustered index
D- a unique constraint

Answer:

B

Explanation:

The table definition allows Owner to be nullable because it was created as Owner nvarchar(100) without NOT NULL. Since the question asks what to add so that future rows must have a value, a check constraint such as CHECK (Owner IS NOT NULL) is the appropriate choice. Microsoft documents that check constraints validate future INSERT and UPDATE operations against the constraint condition.

The other options do not solve the requirement:

A foreign key enforces referential integrity, not non-null entry by itself.

A nonclustered index does not require values to be present.

A unique constraint prevents duplicate values but still does not serve as the right mechanism here for enforcing presence across future writes. Microsoft's constraint documentation also notes that primary-key columns are implicitly NOT NULL, which helps distinguish nullability enforcement from other constraint types.

Question 5

Question Type: MultipleChoice

You have a GitHub Actions workflow that builds and deploys an Azure SQL database. The schema is stored in a GitHub repository as an SDK-style SQL database project.

Following a code review, you discover that you need to generate a report that shows whether the production schema has diverged from the model in source control.

Which action should you add to the pipeline?

Options:

A- SqlPackage.exe /Action:DriftReport
B- SqlPackage.exe /Action:DeployReport
C- SqlPackage.exe /Action:Extract
D- SqlPackage.exe /Action:Script

Answer:

A

Explanation:

Microsoft documents that DriftReport creates an XML report showing changes that have been made to the registered database since it was last registered. That is the action intended to detect whether the production schema has diverged from the expected model baseline in your deployment workflow.

This is different from DeployReport, which shows the changes that would be made by a publish action. In other words:

DriftReport answers: Has the deployed database drifted from the registered state/model?

DeployReport answers: What changes would be applied if I published now?

The other options are not the right fit:

Extract creates a DACPAC from an existing database, not a drift analysis report.

Script generates a deployment script, not a schema-drift report.

So to generate a report that shows whether production has diverged from the model in source control, add:

SqlPackage.exe /Action:DriftReport

Question 6

Question Type: MultipleChoice

You have an Azure SQL database that contains tables named dbo.ProduetDocs and dbo.ProductuocsEnbeddings. dbo.ProductOocs contains product documentation and the following columns:

* Docld (int)

* Title (nvdrchdr(200))

* Body (nvarthar(max))

* LastHodified (datetime2)

The documentation is edited throughout the day. dbo.ProductDocsEabeddings contains the following columns:

* Dotid (int)

* ChunkOrder (int)

* ChunkText (nvarchar(aax))

* Embedding (vector(1536))

The current embedding pipeline runs once per night

Vou need to ensure that embeddings are updated every time the underlying documentation content changes The solution must NOT 'equire a nightly batch process.

What should you include in the solution?

Options:

A- fixed-size chunking
B- a smaller embedding model
C- table triggers
D- change tracking on dbo.ProductDocs

Answer:

D

Explanation:

The requirement is to ensure embeddings are updated every time the underlying content changes without relying on a nightly batch job. The right design is to enable change tracking on the source table so an external process can identify which rows changed and regenerate embeddings only for those rows. Microsoft documents that change detection mechanisms are used to pick up new and updated rows incrementally, which is the right pattern when you need near-continuous refresh instead of full nightly rebuilds.

This is better than:

A . fixed-size chunking, which affects chunk strategy but not change detection.

B . a smaller embedding model, which affects model cost/latency but not update triggering.

C . table triggers, which would push embedding-maintenance logic directly into write operations and is generally not the best design for AI-processing pipelines. The question specifically asks for a solution that replaces the nightly batch requirement, not one that performs heavyweight work inline during every transaction.

Question 7

Question Type: MultipleChoice

You have an Azure SQL database that contains a table named dbo.ManualChunks. dbo.HonualChunks 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 endpomt for chat completions. The request body must include both the user question and theretiieved chunks.

You write the following Transact-SQL code.

```
01 CREATE DATABASE SCOPED CREDENTIAL AzureOpenAIHeaders
02 WITH IDENTITY = 'HTTPEndpointHeaders',
03   SECRET = N'{"api-key":"<YOUR_AZURE_OPENAI_API_KEY>"}';
04 GO
05 CREATE OR ALTER PROCEDURE dbo.AskManuals
06   . . .
07   SELECT @chunks =
08   (
09     SELECT TOP (5)
10       mc.ChunkText AS [text]
11     FROM dbo.ManualChunks AS mc
12     ORDER BY mc.Score DESC
13     FOR JSON PATH
14   );
15   SET @payload =
16   (
17     SELECT
18       'system' AS [messages[0].role],
19       'Use only the provided manual chunks.' AS [messages[0].content],
20       'user' AS [messages[1].role],
21       CONCAT(@question, CHAR(10), JSON_QUERY(@chunks)) AS [messages[1].content]
22
23   );
24
25   EXEC @retval =
26     . . .
27 END;
28 GO
```

What should you insert at line 22?

Options:

A- FOR XHL AUTO, TYPE, XML SCHEMA,

B- FOR 3SON AUTO, IMCLUDE_MULL_VAIUES

C- FOR XHL PATH, INCLUDE_NULL_VAIUES

D- FOR JSON PATH, WITHOUT_ARRAY_WRAPPER

Answer:

D

Explanation:

The correct insertion at line 22 is FOR JSON PATH, WITHOUT_ARRAY_WRAPPER.

The request body for the Azure OpenAI chat completions call must be a single JSON object containing the messages array with both the system/user content and the retrieved chunks. Microsoft documents that FOR JSON PATH is the preferred way to shape JSON output, especially when you want precise control over nested property names like messages[0].role and messages[1].content.

The key detail is WITHOUT_ARRAY_WRAPPER. By default, FOR JSON returns results enclosed in square brackets as a JSON array. Microsoft documents that WITHOUT_ARRAY_WRAPPER removes those brackets so a single JSON object is produced instead. That is exactly what is needed here for @payload, because the stored procedure is building one request body, not an array of request bodies.

Question 8

Question Type: MultipleChoice

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 0orderDate 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?

Options:

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());

Answer:

D

Explanation:

The correct answer is D because the scalar UDF must return the number of whole years from the input @OrderDate to the current date/time as a single positive integer. The correct DATEDIFF order is:

DATEDIFF(year, @OrderDate, GETDATE())

Microsoft documents that DATEDIFF(datepart, startdate, enddate) returns the count of specified datepart boundaries crossed between the start and end values. Since @OrderDate is the earlier date and GETDATE() is the later date, this ordering returns a positive result for past order dates.

The other choices are incorrect:

A reverses the arguments and would return a negative value for a past order date.

B is missing RETURN, and converting month difference to years by dividing by 12 is not the direct whole-year expression the question asks for.

C subtracts year parts only, which can be off around anniversary boundaries because it ignores whether the full year has actually elapsed.

So the correct insertion at line 05 is:

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

Question 9

Question Type: MultipleChoice

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-SG1 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)

Options:

A- Option A
B- Option B
C- Option C
D- Option D
E- Option E

Answer:

B

Explanation:

The correct answer is Option B because the requirement is to call an Azure OpenAI REST endpoint from SQL Server 2025 while providing the highest level of security, and the instance already has a managed identity enabled. For Microsoft's SQL AI features, the preferred secure pattern is to use a database scoped credential with IDENTITY = 'Managed Identity' instead of storing an API key. Microsoft documents that SQL Server 2025 supports managed identity for external AI endpoints, and for Azure OpenAI the credential secret uses the Cognitive Services resource identifier: {'resourceid':'https://cognitiveservices.azure.com'}.

So line 02 should be:

WITH IDENTITY = 'Managed Identity', SECRET = '{'resourceid':'https://cognitiveservices.azure.com'}';

Why the other options are incorrect:

A and D use HTTP header or query-string credentials with an API key, which is less secure than managed identity because a secret key must be stored and rotated manually. Microsoft recommends managed identity where supported to avoid embedded secrets.

C mixes Managed Identity with an api-key secret, which is not the correct pattern for Azure OpenAI managed-identity authentication.

E uses an invalid identity value for this scenario. The accepted credential identities for external REST endpoint calls include HTTPEndpointHeaders, HTTPEndpointQueryString, Managed Identity, and Shared Access Signature.

Because the endpoint is Azure OpenAI and the question explicitly asks for the highest security, managed identity with the Cognitive Services resource ID is the Microsoft-aligned answer.