Free Microsoft DP-800 Practice Questions

Shared by Chambers on 13-08-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: Hotspot

Case Study: Contoso

Existing 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)
VehicleHealthSummary	• IncidentId (int) (primarykey) • VehicleId (int) • 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.

Requirements

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

* Al 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 VehicleTd, Lastupdatedutc, EngineStatus, BatteryHealth FROM dbo.VehicleHealthSumary where fleetld - gFleetld ORDER BV 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.

Planned Changes

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

Security 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 LastModif reduce column to the time of the change.

* Avoid recursive updates.

AI Search, Embedding's, and Vector indexing

The development learn 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 customer-Feedback table:

* Extract the customer feedback text from the JSON document.

* Filter rows where the JSON text contains a keyword.

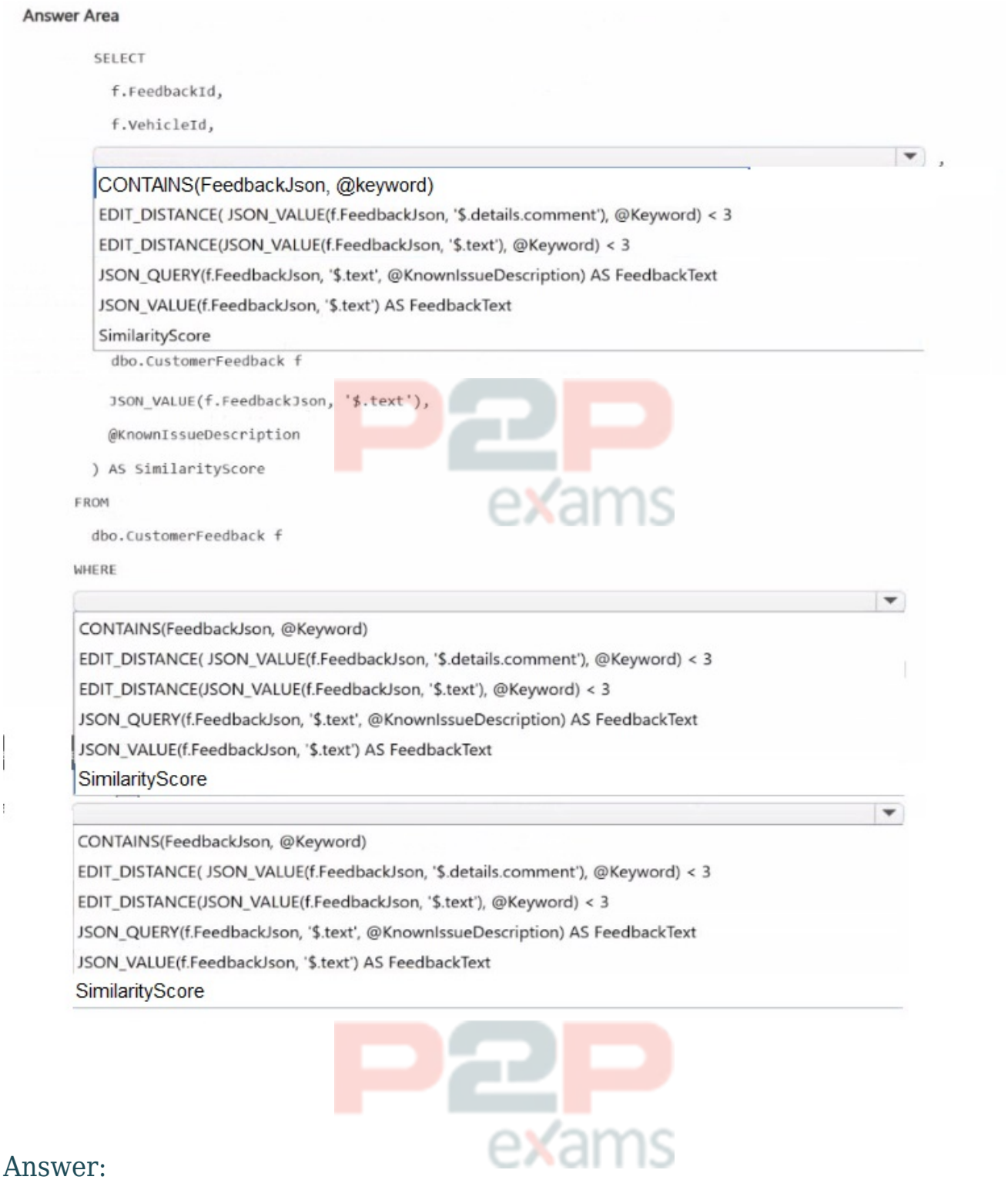
* Calculate a fuzzy similarity score between the feedback text and a known issue description.

* 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:

See the Answer in the Premium Version!

Question 2

Question Type: MultipleChoice

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.

Options:

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.

Answer:

D, E

Explanation:

The two correct actions are D and E because the ingestion failures are caused by malformed JSON and duplicate payloads, and these two controls address those two problems directly. Microsoft's JSON documentation states that SQL Server and Azure SQL support validating JSON with ISJSON, and Microsoft specifically recommends using a CHECK constraint to ensure JSON text stored in a column is properly formatted.

For the duplicate-payload issue, creating a unique index on a hash of the payload is the appropriate design. Microsoft documents using hashing functions such as HASHBYTES to hash column values, and SQL Server allows a deterministic computed column to be used as a key column in a UNIQUE constraint or unique index. That makes a persisted hash-based computed column plus a unique index a practical and exam-consistent way to reject duplicate payloads efficiently.

The other options do not solve the stated root causes:

Snapshot isolation addresses concurrency behavior, not malformed JSON or duplicate payload detection.

A trigger to rewrite malformed JSON is not the right integrity control and is brittle.

Foreign key constraints enforce referential integrity, not JSON validity or duplicate-payload prevention

Question 3

Question Type: MultipleChoice

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

You have a GitHub Actions workflow that already runs dotnet build and produces a database artifact.

You need to add a deployment step that publishes the dacpac file to an Azure SQL database by using the secrets stored in GitHub repository secrets

What should you include in the workflow?

A)

```
env:
  SQL_CONNECTION_STRING: Server=tcp:myserver.database.windows.net;
...
steps:
- name: Publish
  uses: azure/sql-action@v2
  with:
    action: publish
    path: bin/Debug/db1.dacpac
    connection-string: ${{ env.SQL_CONNECTION_STRING }}
```

B)

```
- name: Publish
  uses: azure/sql-action@v2
  with:
    action: extract
    path: bin/Debug/db1.dacpac
    connection-string: ${{ secrets.SQL_CONNECTION_STRING }}
```

C)

```
- name: Publish
  uses: azure/sql-action@v2
  with:
    action: publish
    path: bin/Debug/db1.dacpac
    connection-string: ${{ secrets.SQL_CONNECTION_STRING }}
```

D)

```
- name: Publish
  run: |
    dotnet build db1.sqlproj \
      /p:TargetConnectionString="${{ secrets.SQL_CONNECTION_STRING }}"
```

Options:

A- Option A
B- Option B
C- Option C

D- Option D

Answer:

C

Explanation:

The correct workflow step is Option C because it uses the Azure SQL GitHub Action to publish a .dacpac file and reads the connection string from GitHub repository secrets, which is exactly what the requirement asks for. Microsoft's Azure SQL GitHub Actions guidance shows using azure/sql-action@v2 with a connection string stored in secrets and a DACPAC path for deployment.

The key parts that make C correct are:

uses: azure/sql-action@v2

action: publish

path: bin/Debug/db1.dacpac

connection-string: ${{ secrets.SQL_CONNECTION_STRING }}

That matches the documented publish pattern for deploying a DACPAC to Azure SQL Database from GitHub Actions. Microsoft and the Azure SQL action documentation both describe Publish as the deployment action for applying a DACPAC to a target database, while Extract is used to create a DACPAC from an existing database, not deploy one.

Why the other options are incorrect:

A uses an environment variable defined inline with a visible connection string rather than using GitHub repository secrets, which does not meet the requirement.

B uses action: extract, which would create a DACPAC from a database instead of publishing the existing DACPAC artifact.

D passes a target connection string to dotnet build, but the question says the workflow already runs dotnet build and produces a database artifact. The missing step is the deployment/publish step, not another build step. Microsoft's SQL project automation guidance separates build the DACPAC from publish the DACPAC.

Question 4

Question Type: MultipleChoice

You need to generate embeddings to resolve the issues identified by the analysts. Which column should you use?

Options:

A- vehicleLocation
B- incidentDescrlption
C- incidentType
D- SeverityScore

Answer:

B

Explanation:

The correct column to use for generating embeddings is incidentDescrlption because embeddings are intended to represent the semantic meaning of rich textual content, not simple categorical, numeric, or location-only values. Microsoft's DP-800 study guide explicitly includes skills such as identifying which columns to include in embeddings, generating embeddings, and implementing semantic vector search for scenarios where users need to find similar records based on meaning rather than exact matches.

In this scenario, analysts report that it is difficult to find similar incidents based on details such as weather, traffic conditions, and location. Those are descriptive context elements that are typically captured in a free-text incident description field. An embedding generated from incidentDescrlption can encode the semantic relationships among these narrative details, making it suitable for similarity search, semantic search, and RAG retrieval. Microsoft documentation on vectors and embeddings explains that embeddings are generated from text data and then stored for vector search to find semantically related items.

The other options are weaker choices:

vehicleLocation is too narrow and usually better handled with geospatial filtering, not embeddings.

incidentType is likely categorical and too low in semantic richness.

SeverityScore is numeric and not appropriate as the primary source for semantic embeddings.

Microsoft also notes that when multiple useful attributes exist, you can either embed each text column separately or concatenate relevant text fields into one textual representation before generating the embedding. But among the options given, the best and most exam-aligned answer is the textual narrative column: incidentDescrlption.

Question 5

Question Type: MultipleChoice

You have an Azure SQL database That contains a table named dbo.Products, dbo.Products contains three columns named Embedding Category, and Price. The Embedding column is defined as VECTOR(1536).

You use Ai_GENERME_EMBEDOINGS and VECTOR_SEARCH to support semantic search and apply additional filters on two columns named Category and Price.

You plan to change the embedding model from text-embedding-ada-002 to text-embedding-3-smalL Existing rows already contain embeddings in the Embedding column.

You need to implement the model change. Applications must be able to use VECTOR_SEARCH without runtime errors.

What should you do first?

Options:

A- Regenerate embeddings for the existing rows.
B- Normalize the vector lengths before storing new embeddings.
C- Convert the Embedding column to nvacchar(mex).
D- Create a vector index on dbo.Products.Embedding.

Answer:

A

Explanation:

When you change embedding models, the stored vectors should be treated as belonging to a different embedding space unless you intentionally keep the entire corpus consistent. Microsoft's vector guidance notes that when most or all embeddings are replaced with fresh embeddings from a new model, the recommended practice is to reload the new embeddings and, for large-scale replacement scenarios, consider dropping and recreating the vector index afterward so search quality remains predictable.

This question also says applications must continue to use VECTOR_SEARCH without runtime errors. VECTOR_SEARCH requires compatible vector dimensions, and the vector column already exists. Azure OpenAI documentation shows that text-embedding-ada-002 is fixed at 1536 dimensions and text-embedding-3-small supports up to 1536 dimensions. That means the migration can remain compatible with a VECTOR(1536) column, but the right implementation step is still to re-embed the existing rows so the table does not contain a mixed corpus produced by different models.

The other options are not appropriate:

B normalization does not solve a model migration problem.

C converting the vector column to nvarchar(max) would break vector-native search design.

D a vector index improves performance, but it does not migrate old embeddings to the new model.

Question 6

Question Type: MultipleChoice

You have an Azure SQL table that contains the following data.

FaqId	ProductName	Question	Answer
1	Laptop Pro	Does it support USB-C charging?	Yes, it supports USB-C charging.
2	Tablet X	Does it support a stylus?	It supports the Active Pen stylus.

You need to retrieve data to be used as context for a large language model (LLM). The solution must minimize token usage.

Which formal should you use to send the data to the LLM?

A)

```
[
  {
    "ProductName": "Laptop Pro",
    "Question": "Does it support USB-C charging?",
    "Answer": "Yes, it supports USB-C charging."
  },
  {
    "ProductName": "Tablet X",
    "Question": "Does it support a stylus?",
    "Answer": "It supports the Active Pen stylus."
  }
]
```

B)

```
[
  {
    "Prompt": "Yes, it supports USB-C charging."
  },
  {
    "Prompt": "It supports the Active Pen stylus."
  }
]
```

C)

```
[
  {

    "FaqId": 1,
    "Question": "Does it support USB-C charging?",
    "Answer": "Yes, it supports USB-C charging."
  },
  {
    "FaqId": 2,
    "Question": "Does it support a stylus?",
    "Answer": "It supports the Active Pen stylus."
  }
]
```

D)

```
[
  {
    "FaqId": 1,
    "ProductName": "Laptop Pro",
    "Question": "Does it support USB-C charging?",
    "Answer": "Yes, it supports USB-C charging."
  },
  {
    "FaqId": 2,
    "ProductName": "Tablet X",
    "Question": "Does it support a stylus?",
    "Answer": "It supports the Active Pen stylus."
  }
]
```

Options:

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

Answer:

A

Explanation:

The correct choice is Option A because it provides the relevant semantic context the LLM needs while avoiding an unnecessary field that would add tokens without improving answer quality.

For LLM grounding and RAG-style context, Microsoft guidance emphasizes mapping and sending the fields that contain text pertinent to the use case. In this FAQ scenario, the useful context is the ProductName, the Question, and the Answer. Those three fields help the model understand both the subject domain and the actual Q&A pair. By contrast, FaqId is just a technical identifier and generally adds no semantic value for response generation, so including

it wastes tokens.

That is why Option A is better than the others:

Option A keeps the meaningful text fields and removes the low-value identifier.

Option B is too minimal because it includes only the answer text as Prompt, which strips away the product and question context the LLM may need for accurate grounding.

Option C keeps FaqId but omits ProductName, which can be important disambiguating context.

Option D includes everything, but that does not minimize token usage because it keeps the unnecessary FaqId.

Question 7

Question Type: MultipleChoice

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?

Options:

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 (he 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.

Answer:

B

Explanation:

The correct answer is B. Microsoft's SSMS Copilot documentation states that queries from

Copilot in SSMS are executed under the context of the user's login and permissions, and that there are no separate permissions for Copilot in SSMS. That means Copilot-run Transact-SQL cannot access more data than the developer's own database principal is already allowed to access.

That is why the other options are incorrect:

A is wrong because Copilot does not use a separate read-only sandbox in place of database permissions.

C is wrong because enforcement is not a client-side filtering trick; it is enforced by the database security context of the current login.

D is wrong because Copilot does not apply a different RLS model from the developer; it simply runs under the same login context.

So the security boundary is the developer's existing database identity and permissions.

Question 8

Question Type: MultipleChoice

You have an Azure SQL database that supports a customer-facing API. The API calls a stored procedure named dbo.GetCustomerOrders thousands of times per hour.

After a deployment that updated indexes and statistics, users report that the API endpoint backed by dbo.Getcustomerorders is slower. In Query Store, the same query now has two persisted execution plans. During the last hour, the newer plan had a significantly higher average duration and CPU time than the older plan.

You need to restore the previous performance quickly, without changing the API code.

Which Transact-SQL command should you run?

Options:

A- EXEC sys.sp_query_store_set_hints
B- DBCC FREEPROCCACHE
C- EXEC sp_query_store_force_plan
D- ALTER DATABASE

Answer:

C

Explanation:

The scenario says Query Store already shows two persisted execution plans for the same query, and the older plan performed much better than the newer one during the last hour. Microsoft documents that sp_query_store_force_plan is used to force a particular plan for a particular query in Query Store. That makes it the fastest way to restore the previously good plan without changing application code, which is exactly what the question requires.

Why the other options are not the best fit:

sp_query_store_set_hints is for adding or updating Query Store hints to influence compilation behavior, but when you already know the exact older good plan, Microsoft points to plan forcing as the direct remedy.

DBCC FREEPROCCACHE clears cached plans broadly and is disruptive; it does not guarantee a return to the known good plan.

ALTER DATABASE is too general and does not directly restore the prior execution plan.

So the right Transact-SQL command is:

EXEC sp_query_store_force_plan

using the relevant @query_id and @plan_id from Query Store for the older, better-performing plan. Microsoft also notes that when a plan is forced, SQL Server tries to use that plan whenever it encounters the query again.

Question 9

Question Type: MultipleChoice

You need to design a generative Al solution that uses a Microsoft SOL Server 2025 database named DB1 as a data source. The solution must generate responses that meet the following requirements:

* Ait' grounded In the latest transactional and reference data stored in D61

* Do NOT require retraining or fine-tuning the language model when the data changes

* Can include citations or references to the source data used in the response

Which scenario is the best use case for implementing a Retrieval Augmented Generation (RAG) pattern? More than one answer choice may achieve the goal. Select the BEST answer

Options:

A- summarizing free-form user input text
B- training a custom language model on historical database data
C- answering user questions based on company-specific knowledge
D- generating marketing slogans based on user sentiment analysis

Answer:

C

Explanation:

The best use case for RAG is answering user questions based on company-specific knowledge. Microsoft defines RAG as a pattern that augments a language model with a retrieval system that provides grounding data at inference time, which is exactly what you need when responses must be based on the latest transactional and reference data, must avoid retraining/fine-tuning, and should be able to include citations or references to source data.

The other options do not fit as well:

summarizing free-form user input does not inherently require retrieval from DB1,

training a custom model contradicts the requirement to avoid retraining/fine-tuning,

generating marketing slogans is a creative generation task, not a grounding-and-citation scenario. RAG is specifically strong when answers must come from your organization's own changing knowledge.

Question 10

Question Type: MultipleChoice

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

Options:

A- Create a clustered index on Lastupdatedutc.
B- On Fleetid, create a nonclustered index that includes Lastupdatedutc. inginestatus, and BatteryHealth.
C- On Lastupdatedutc. create a nonclustered index that includes Fleetid.
D- On Fleetid, create a filtered index where lastupdatedutc > DATEADD(DAV, -7, SYSuTCOATETIME()).

Answer:

B

Explanation:

The best recommendation is B because the slow query filters on FleetId and returns LastUpdatedUtc, EngineStatus, and BatteryHealth. A nonclustered index with FleetId as the key column allows the optimizer to perform an index seek instead of a clustered index scan, and including the other selected columns makes the index covering, which reduces extra lookups and I/O. Microsoft's SQL Server indexing guidance states that a nonclustered index with included columns can significantly improve performance when all query columns are available in the index, because the optimizer can satisfy the query directly from the index.

The query is:

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

Among the given choices, FleetId is the most important search argument because it appears in the WHERE predicate. Microsoft's index design guidance recommends putting columns used for searching in the key and using nonkey included columns to cover the rest of the query efficiently.

Why the other options are weaker:

A is not appropriate because changing the clustered index to LastUpdatedUtc would not target the main filter predicate on FleetId, and a table can have only one clustered index.

C makes LastUpdatedUtc the key, which is poor for a query whose primary filter is FleetId.

D is not the right answer here because the query requirement does not specify only recent rows, and filtered indexes are meant for a well-defined subset; this option also uses a time-based expression that is not aligned to the stated query pattern.

Strictly speaking, the most optimal design for both filtering and ordering would usually be a composite key like (FleetId, LastUpdatedUtc), but since that is not one of the available options, B is the correct exam answer.

Question 11

Question Type: MultipleChoice

You need to recommend a solution for the development team to retrieve the live metadata. The solution must meet the development requirements.

What should you include in the recommendation?

Options:

A- Export the database schema as a .dacpac file and load the schema into a GitHub Copilot context window.
B- Add the schema to a GitHub Copilot instruction file.
C- Use an MCP server
D- Include the database project in the code repository.

Answer:

C

Explanation:

The best recommendation is to use an MCP server. In the official DP-800 study guide, Microsoft explicitly lists skills such as configuring Model Context Protocol (MCP) tool options in a GitHub Copilot session and connecting to MCP server endpoints, including Microsoft SQL Server and Fabric Lakehouse. That makes MCP the exam-aligned mechanism for enabling AI-assisted tools to work with live database context rather than static snapshots.

This also matches the stated development requirement: the team will use Visual Studio Code and GitHub Copilot and needs to retrieve live metadata from the databases. Microsoft's documentation for GitHub Copilot with the MSSQL extension explains that Copilot works with an active database connection, provides schema-aware suggestions, supports chatting with a connected database, and adapts responses based on the current database context. Microsoft also documents MCP as the standard way for AI tools to connect to external systems and data sources through discoverable tools and endpoints.

The other options do not satisfy the ''live metadata'' requirement as well:

A .dacpac is a point-in-time schema artifact, not live metadata.

A Copilot instruction file provides guidance, not live database discovery.

Including the database project in the repository helps source control and deployment, but it still does not provide live database metadata by itself.

Question 12

Question Type: MultipleChoice

You need to enable similarity search to provide the analysts with the ability to retrieve the most relevant health summary reports. The solution must minimize latency.

What should you include in the solution?

Options:

A- a computed column that manually compares vector values
B- a standard nonclustered index on the Fmbeddings (vector (1536)) column
C- a full-text index on the Fmbeddings (vector (1536)) column
D- a vector index on the Embedding* (vector (1536)) column

Answer:

D

Explanation:

The correct answer is D because the requirement is to enable similarity search over embedding vectors and to minimize latency. Microsoft documents that CREATE VECTOR INDEX is specifically used to create an index on vector data for approximate nearest neighbor (ANN) search, which is designed to accelerate vector similarity queries compared to exact k-nearest-neighbor scans.

This matches the scenario exactly. The VehicleHealthSummary table already includes an Embeddings (vector(1536)) column. In Microsoft SQL platforms, embeddings are stored in vector columns and queried for semantic similarity. To improve performance and reduce response time, Microsoft recommends a vector index, not a regular B-tree nonclustered index and not a full-text index. A vector index is purpose-built for finding the most similar vectors efficiently.

The other options are not appropriate:

A would require manual comparison logic and would increase latency rather than minimize it.

B is incorrect because a standard nonclustered index is not the index type used for vector similarity operations.

C is incorrect because full-text indexes are for textual token-based search, not numeric vector embeddings.

Microsoft's current documentation is explicit that vector indexes support approximate nearest neighbor search, and that the optimizer can use the ANN index automatically for vector queries. That is the exam-aligned design choice when the goal is fast retrieval of the most relevant health summary reports from an embeddings column.

Question 13

Question Type: MultipleChoice

You have an Azure SQL database that supports the OLTP workload of an order-processing application.

During a 10-minute incident window, you run a dynamic management view query and discover the following:

Session 72 is sleeping with open_transaction_count = 1.

Multiple other sessions show blocking_session_id = 72 in sys.dm_exec_requests.

sys.dm_exec_input_buffer(72, NULL) returns only BEGIN TRANSACTION UPDATE Sales.Orders.

Users report that updates to Sales.Orders intermittently time out during the incident window. The timeouts stop only after you manually terminate session 72.

What is a possible cause of the blocking?

Options:

A- A long-running SELECT statement is blocking writers.
B- Session 72 caused a deadlock.
C- An explicit transaction was started but not committed or rolled back.
D- A lock escalation occurred.

Answer:

C

Explanation:

The best explanation is an open explicit transaction. During the incident, session 72 was sleeping but still had open_transaction_count = 1, and sys.dm_exec_input_buffer(72, NULL) showed only BEGIN TRANSACTION UPDATE Sales.Orders. That pattern indicates the session executed an update inside an explicit transaction and then remained idle without committing or rolling back, while still holding locks. Other sessions showing blocking_session_id = 72 is the expected symptom of that situation. Microsoft explains that blocking occurs when one session holds a lock on a resource and another session requests a conflicting lock, and sleeping sessions can continue to block if they retain locks through an open transaction.

This also fits the observed behavior that the timeouts stopped only after session 72 was terminated. Killing the session would roll back the active transaction and release the locks, allowing waiting updates to continue. That is much more consistent with an uncommitted

transaction than with a deadlock, because deadlocks are normally detected and one session is chosen as the victim automatically rather than persisting until manual termination.

P2P exams

P2P exams

To Get Premium Files for DP-800 Visit

https://www.p2pexams.com/products/dp-800

For More Free Questions Visit

https://www.p2pexams.com/microsoft/pdf/dp-800