PassQueen
a.k.a dumpsqueen.com

Developing AI-Enabled Database Solutions

Microsoft DP-800

Version Demo

Total Demo Questions: 7

Total Premium Questions: 73

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Design and implement a vector search solution	16
Topic 2, Design and implement a RAG solution	11
Topic 3, Design and implement an AI agent	4
Topic 4, Mix Questions	42
Total	73

QUESTION NO: 1 - (SIMULATION)

You need to create a table in the database to store the telemetry data. You have the following Transact-SQL code.

ANSWER: See the explanation for the answer

Explanation:

Selections (in statement order):

No — The table is not partitioned. A clustered primary key alone does not partition a table; partitioning requires a partition function, a partition scheme, and creation of the table/index on that scheme using the partitioning column.
Yes — The JSON index `JI_VehicleTelemetry_Location` includes `$.location.latitude`, `$.location.longitude`, and `$.location.accuracy`. Predicates on these indexed JSON paths can be optimized with an index seek.
No — `$.location.heading` is not one of the paths included in `JI_VehicleTelemetry_Location`. Therefore, filtering by heading is not covered by that JSON index and cannot use it for an index seek.

QUESTION NO: 2

Your development team plans to connect GitHub Copilot Chat to an MCP server that can retrieve database metadata from an Azure SQL database. The team needs current schema information, but the MCP server must not expose customer rows, credentials, or unrestricted Transact-SQL execution.

Which two actions should you recommend? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. Use a dedicated database identity with only the permissions required for approved metadata.

B. Expose read-only MCP tools that return approved schema metadata rather than arbitrary query results.

C. Configure the MCP server to connect by using a sysadmin identity.

D. Expose a tool that executes any Transact-SQL statement supplied in chat.

E. Store the database connection string in a repository instruction file.

ANSWER: A B

Explanation:

The MCP server should connect by using a dedicated database identity that has only the permissions required to read approved metadata. This prevents the server from inheriting broad developer or application permissions. The server should also expose narrowly scoped, read-only metadata tools, such as tools that list approved schemas, tables, columns, and relationships. Tool inputs should not permit arbitrary Transact-SQL execution.

Using a sysadmin identity violates least-privilege requirements. Returning result sets from customer tables exposes data that the requirement explicitly excludes. Embedding a connection string in a repository instruction file would disclose a secret and does not constrain MCP tool behavior.

QUESTION NO: 3

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 processes 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. SetSql 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

ANSWER: C D

Explanation:

The Azure SQL trigger for Azure Functions is based on SQL Server change tracking. Therefore, change tracking must first be enabled for AdventureWorksDB itself, and then it must be enabled explicitly for the monitored `dbo.Employee` table. Database-level enablement establishes change tracking as a database feature; table-level enablement identifies the table whose insert, update, and delete activity the trigger can detect. These are complementary requirements, and both are required before an Azure SQL trigger can monitor changes to that table.

The stated five-second and 100-change limits do not change the required database-side setup. Those behaviors are controlled by Azure Functions SQL extension trigger configuration, such as polling interval and maximum batch-size settings, rather than by a different Azure SQL database feature. The SQL trigger's default maximum batch size is 100, while the polling interval can be configured by the application when needed. With change tracking correctly enabled at both scopes, the function runtime can maintain its change-tracking lease and invoke the function for detected changes. See Azure SQL trigger for Azure Functions and Enable and disable change tracking.

QUESTION NO: 4

You have an Azure SQL database that contains dbo.InventoryChunks. The table includes Embedding, WarehouseRegion, ProductCategory, and IsActive columns.

A RAG application must retrieve the most relevant active chunks for a selected warehouse region and product category. The application must not retrieve candidates from other regions or categories and then discard them after ranking.

What should you do?

A. Filter by IsActive, WarehouseRegion, and ProductCategory before ordering the eligible rows by vector distance.

B. Return the nearest vectors across all rows and remove unmatched regions and categories in the application.

C. Append WarehouseRegion and ProductCategory to the chunk text and omit structured predicates.

D. Create a full-text index on WarehouseRegion and ProductCategory.

ANSWER: A

Explanation:

Apply the structured predicates in the retrieval query before ranking the vectors by distance. This restricts the candidate set to active chunks in the requested warehouse region and product category, and then returns the nearest vectors within that permitted and relevant set.

Filtering after selecting the nearest vectors can leave too few results and can omit better matches that were outside the initial global top results. Embedding structured filter values does not reliably enforce a business filter, and full-text indexing does not enforce vector-retrieval scope.

QUESTION NO: 5

You are designing a Retrieval Augmented Generation (RAG) solution for an internal policy database. Each policy document is split into chunks and stored with an embedding in dbo.PolicyChunks.

Users must be able to verify a generated answer by opening the policy section from which the answer was derived.

What should you store with each embedding?

A. The source document identifier and the chunk location

B. The embedding model name and deployment name

C. The vector similarity score from the most recent search

D. The document title without a section identifier

ANSWER: A

Explanation:

Store the source document identifier and the chunk location, such as a section identifier or chunk ordinal, with each embedding. A vector search returns chunks, not only a generated answer. Persisting source metadata with each chunk enables the application to identify the originating policy document and the specific section that was retrieved. The application can then include a citation or link to that source in the generated response.

Storing only the embedding model name is useful for lifecycle management but does not identify the source content. A similarity score indicates retrieval relevance but cannot be used as a citation. Storing only the document title is insufficient when a document contains multiple sections with different guidance.

QUESTION NO: 6 - (HOTSPOT)

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:



Explanation:

The correct selections recognize that Dynamic Data Masking and row-level security address complementary parts of protecting customer-profile data. The schema marks FullName, EmailAddress, and PhoneNumber with masking functions. This causes users who do not have permission to view unmasked data to receive masked representations of those PII values in query results. Microsoft documents Dynamic Data Masking as a way to limit exposure of sensitive data for nonprivileged users; it is applied when data is presented by a query. See Dynamic Data Masking in Azure SQL Database.

At the same time, the security policy has an active filter predicate on dbo.CustomerProfiles. For each candidate row, the predicate checks dbo.UserRegionAccess for a record whose UserPrincipalName equals the current SUSER_SNAME() value

and whose RegionCode equals the row's RegionCode. Consequently, the application receives only customer-profile rows for regions to which its current principal has been assigned. This is the intended use of a row-level-security filter predicate: transparently restricting returned rows according to the executing user's context. Microsoft describes that behavior in Row-Level Security.

The selections are therefore **Yes** for the statement about meeting PII security requirements, followed by **No** for the statement that Azure SQL server administrators automatically see every row through the application, and **No** for the statement that masks apply to rows filtered away by RLS. An administrator may have the authority to administer or change a policy, but that authority is distinct from the result produced by the currently enabled predicate during an application query. Also, masking applies to values in rows returned by the query. Because an RLS-filtered row is not returned at all, it has no displayed values to mask.

QUESTION NO: 7 - (SIMULATION)

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.

ANSWER: See the explanation for the answer

Explanation:

Use a half-open date range so that no function is applied to `o.CreateDate`. The correct selections complete the procedure as follows:

```
SET @EndDate = DATEADD(day, 1, @StartDate);

WHERE o.CreateDate >= @StartDate
  AND o.CreateDate < @EndDate;
```

This returns all orders created on `@StartDate`, including values with a time portion, while remaining SARGable. Applying `DATEADD` to the parameter rather than converting or otherwise transforming the table column allows an index on `CreateDate` to be used efficiently.