DP-800試験復習、DP-800模擬モード

JpexamのMicrosoftのDP-800問題集を買う前に、一部の問題と解答を無料に試用することができます。そうすると、JpexamのMicrosoftのDP-800トレーニング資料の品質をよく知っています。JpexamのMicrosoftのDP-800問題集は絶対あなたの最良の選択です。

Microsoft DP-800 認定試験の出題範囲:

トピック	出題範囲
トピック 1	• Secure, optimize, and deploy database solutions: This domain focuses on implementing data security measures like encryption, masking, and row-level security, optimizing query performance, managing CI • CD pipelines using SQL Database Projects, and integrating SQL solutions with Azure services including Data API builder and monitoring tools.
トピック 2	• Design and develop database solutions: This domain covers designing and building database objects such as tables, views, functions, stored procedures, and triggers, along with writing advanced T-SQL code and leveraging AI-assisted tools like GitHub Copilot and MCP for SQL development.
トピック 3	• Implement AI capabilities in database solutions: This domain covers designing and managing external AI models and embeddings, implementing full-text, semantic vector, and hybrid search strategies, and building retrieval-augmented generation (RAG) solutions that connect database outputs with language models.

>> DP-800試験復習 <<

DP-800試験の準備方法｜有効的なDP-800試験復習試験｜一番優秀なDeveloping AI-Enabled Database Solutions模擬モード

JpexamがMicrosoftのDP-800トレーニング資料を利用してから試験に合格することがとてもたやすことになって、これは今までがないことです。これは試験に合格した受験生の一人が言ったのです。Jpexamが提供したMicrosoftのDP-800トレーニング資料はあなたの雑然とした考えを整理できます。そうしたらあなたは心理的なストレスを減らせるだけでなく、気楽に試験に受かることもできます。Jpexamには一部の問題と解答を無料に提供して差し上げますから、もし私の話を信じないのなら、試用版を使ってみてください。利用してみたら効果があるかどうか自分でよく知っているようになります。あなたに絶対向いていると信じていますよ。

Microsoft Developing AI-Enabled Database Solutions 認定 DP-800 試験問題 (Q36-Q41):

質問 # 36
You have an Azure SQL database that stores order data. A reporting query aggregates monthly revenue per customer runs frequently.

You need to reduce how long it takes to retrieve the calculated values. The solution must NOT alter any underlying table structure. What should you do?

- A. Create a view by using ORDER BY without TOP. and then create a unique clustered index on the view.
- B. Create a view without using with SCHEMABHIDING, and then create a nonclustered index on the view.
- C. Create a view by using WITH SCHEHABINDING, include COUNT_BIG(*). and then create a unique clustered index on the view.
- D. Create a view by using GROUP BV. and then create a unique clustered index on the view.

正解：C

解説：
To speed up repeated aggregate retrieval without changing base-table structure, the right pattern is an indexed view . Microsoft requires that an indexed view be created with WITH SCHEMABINDING , and if the view uses GROUP BY , it must also include COUNT_BIG(*) . After that, the first index on the view must be a unique clustered index .
The other options fail Microsoft's indexed-view rules:
* A is invalid because ORDER BY is not allowed in the indexed-view definition.
* B is invalid because indexed views require WITH SCHEMABINDING , and the first index cannot just be a nonclustered index.
* C is incomplete because a grouped indexed view must include COUNT_BIG(*) .

質問 # 37
You need to recommend a solution to lesolve the slow dashboard query issue. What should you recommend?

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

正解：D

解説：
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.

質問 # 38
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 nvarehar(iMx) column named log that stores JSON documents You have a daily report that filters by the $.severity JSON property and returns Logld. 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.

正解：

解説：
Explanation:
The correct way to avoid full table scans here is to add a computed column that extracts the JSON scalar property with JSON_VALUE , and then create a nonclustered index on that computed column with the report's returned columns in the INCLUDE list. Microsoft's JSON indexing guidance specifically recommends creating a computed column that exposes the JSON property you filter on, using the same expression as in the query, and then indexing that computed column.
So the computed column must be:
AS JSON_VALUE([log], ' $.severity ') PERSISTED
This is correct because $.severity is a scalar JSON value, so JSON_VALUE is the proper function.
JSON_QUERY would be for extracting an object or array, not a scalar property. Microsoft also notes that persisted computed columns can improve access speed for JSON-derived values.
The index should then include:
INCLUDE (LogId, LogDateTime, [log])
That is the right covering strategy because the report filters by severity but returns LogId, LogDateTime, and log. Microsoft's guidance on included columns explains that nonkey included columns let a nonclustered index cover more queries and reduce extra lookups to the base table.
So the completed code is:
ALTER TABLE WebSite.Logs
ADD severity AS JSON_VALUE([log], ' $.severity ') PERSISTED;
GO
CREATE INDEX ix_severity
ON WebSite.Logs(severity)
INCLUDE (LogId, LogDateTime, [log]);
GO

質問 # 39
You have a SQL database in Microsoft Fabric that contains a table named dbo.Orders, dbo.Orders has a clustered index, contains three years of data, and is partitioned by a column named OrderDate by month.
You need to remove all the rows for the oldest month. The solution must minimize the impact on other queries that access the data in dbo.orders.
Solution; Identify the partition scheme (or the oldest month, and then run the following Transact-SQL statement.
ALTER TABLE dbo.Orders
DROP PARTITION SCHEME (partition_scheme_name);
Does this meet the goal?

- A. No
- B. Yes

正解：A

解説：
This also does not meet the goal. DROP PARTITION SCHEME removes the partition scheme object from the database; it is not the command used to remove just the rows for the oldest month from a partitioned table.
Microsoft's DROP PARTITION SCHEME documentation is explicit that the statement removes the partition scheme itself.
For removing only the oldest month's rows with minimal impact, Microsoft points to partition-level maintenance operations such as truncating a single partition on a partitioned table. That targets only the needed data subset and is more efficient for retention workloads.

質問 # 40
You need to create a table in the database to store the telemetry data. You have the following Transact-SQL code.

正解：

解説：

Explanation:

The first statement is No . The requirement says telemetry data must be stored in a partitioned table to provide predictable performance for ingestion and retention operations. However, the shown CREATE TABLE statement does not define a partition function or partition scheme, and the table is created with a regular clustered primary key on TelemetryId. Microsoft's partitioning guidance states that creating a partitioned table requires a partition function , a partition scheme , and creating the table or index on that partition scheme using a partitioning column. None of that appears in the code, so the table is not partitioned.

The second statement is Yes . The code creates a JSON index named JI_VehicleTelemetry_Location on LocationJson for these specific JSON paths: $.location.latitude, $.location.longitude, and $.location.accuracy.

That matches the requirement that those JSON properties must be filterable by using an index seek .

Microsoft documents that JSON indexing is used to optimize filtering and sorting on JSON properties, and the index only helps for the properties included in the index definition.

The third statement is No . The JSON index is defined only for latitude, longitude, and accuracy. A query filtering on $.location.heading references a different path that is not included in the index definition, so that query would not use JI_VehicleTelemetry_Location for that predicate. JSON indexes are path-specific; they do not automatically cover unrelated properties in the same JSON document.

質問 # 41

.....

現在、試験銀行がシミュレーションテストを提供するような統合システムを持っていることはほとんどありません。 DP-800学習ツールについて学習した後、実際のDP-800試験を刺激することの重要性が徐々に認識されます。この機能により、練習システムがどのように動作するかを簡単に把握し、DP-800試験に関する中核的な知識を得ることができます。さらに、実際の試験環境にいるときは、質問への回答の速度と品質を制御し、エクササイズの良い習慣を身に付けることを学ぶことができます。そのため、DP-800試験に合格できます。