Free Microsoft DP-800 Questions [2026]–Fully Updated

Our DP-800 training materials are regarded as the most excellent practice materials by authority. Our company is dedicated to researching, manufacturing, selling and service of the DP-800 study guide. Also, we have our own research center and experts team. So our products can quickly meet the new demands of customers. That is why our DP-800 Exam Questions are popular among candidates. we have strong strenght to support our DP-800 practice engine.

Microsoft DP-800 Exam Syllabus Topics:

Topic	Details
Topic 1	• 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.
Topic 2	• 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.
Topic 3	• 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.

>> DP-800 Latest Test Dumps <<

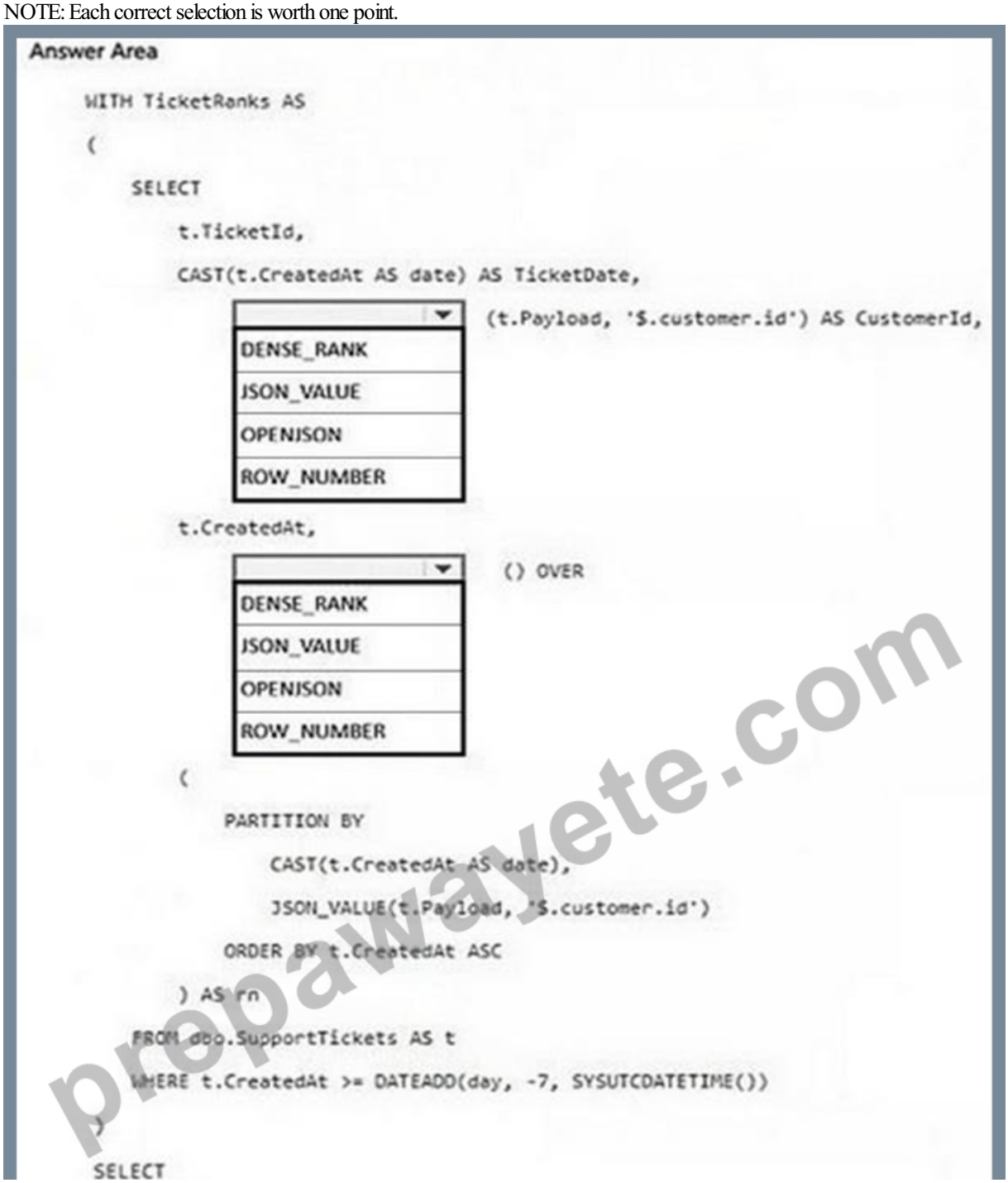
100% Pass Quiz Microsoft - DP-800 - Developing AI-Enabled Database Solutions –High-quality Latest Test Dumps

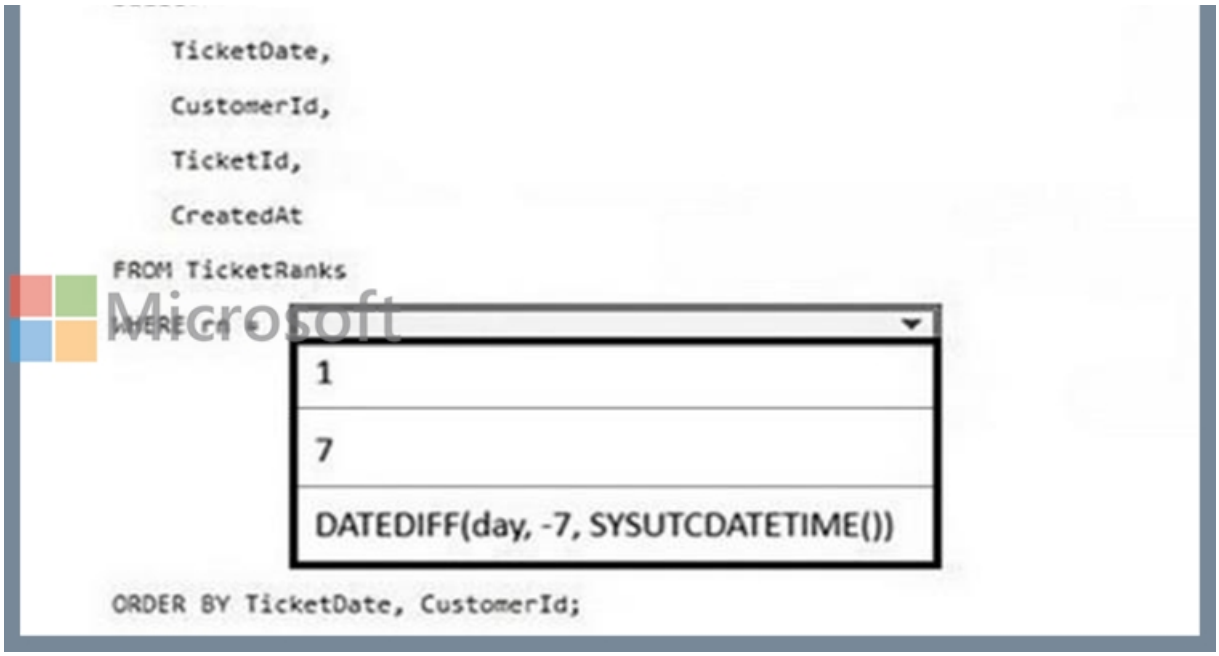
Our company employs the first-rate expert team which is superior to others both at home and abroad. Our experts team includes the experts who develop and research the DP-800 study materials for many years and enjoy the great fame among the industry, the

senior lecturers who boost plenty of experiences in the information about the exam and published authors who have done a deep research of the DP-800 Study Materials and whose articles are highly authorized. They provide strong backing to the compiling of the DP-800 study materials and reliable exam materials resources. They compile each answer and question carefully.

Microsoft Developing AI-Enabled Database Solutions Sample Questions (Q82-Q87):

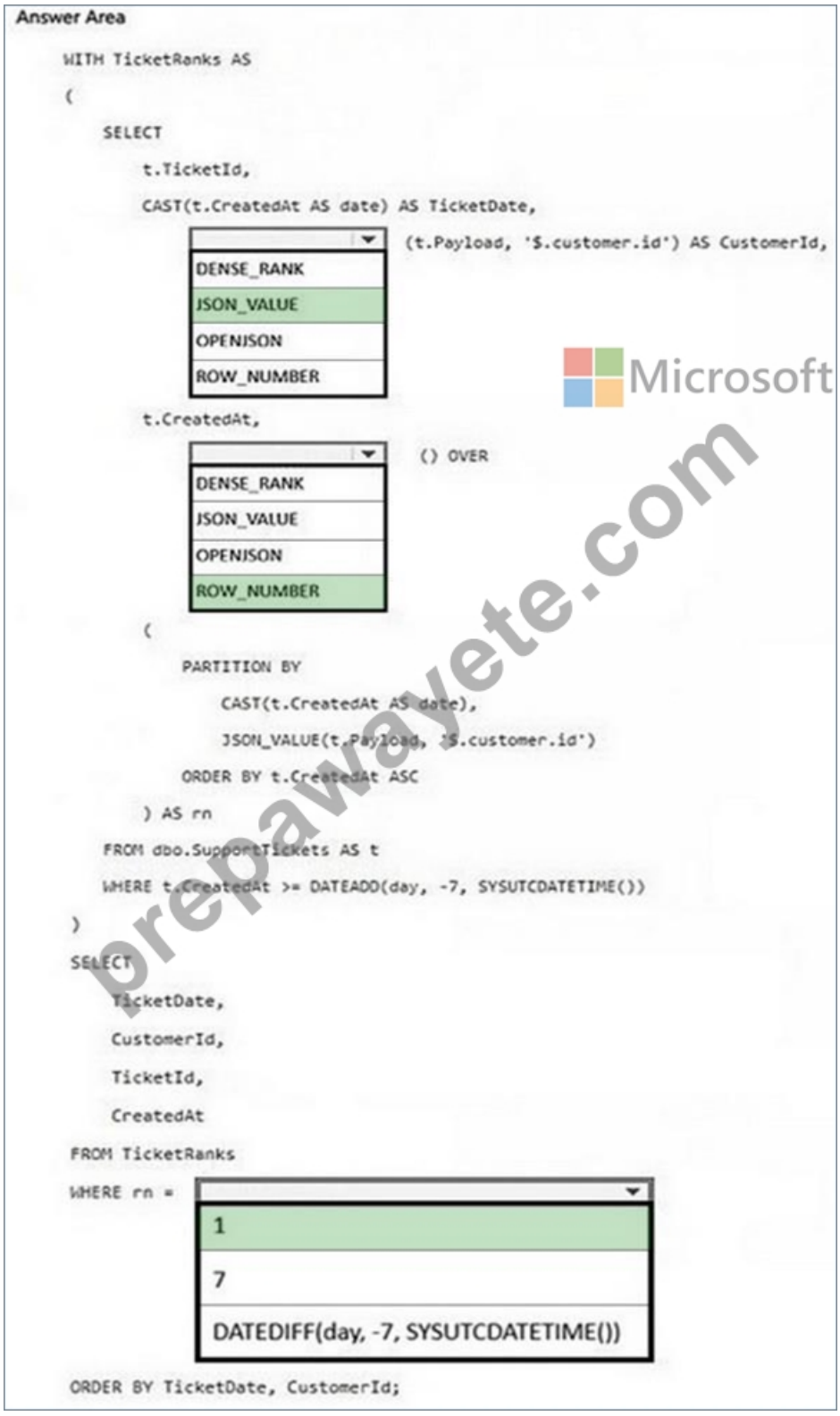
NEW QUESTION # 82
Hotspot Question
You have an Azure SQL database that contains a table named dbo.SupportTickets.
dbo.SupportTickets contains a JSON column named Payload and a datetime column CreatedAt.
You need to generate a report for the last seven days that meets the following requirements:
- Returns exactly one row per customer per day
- For each customer and day, returns the earliest ticket
- Includes the customer ID stored in Payload
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:

Explanation:



NEW QUESTION # 83
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
```

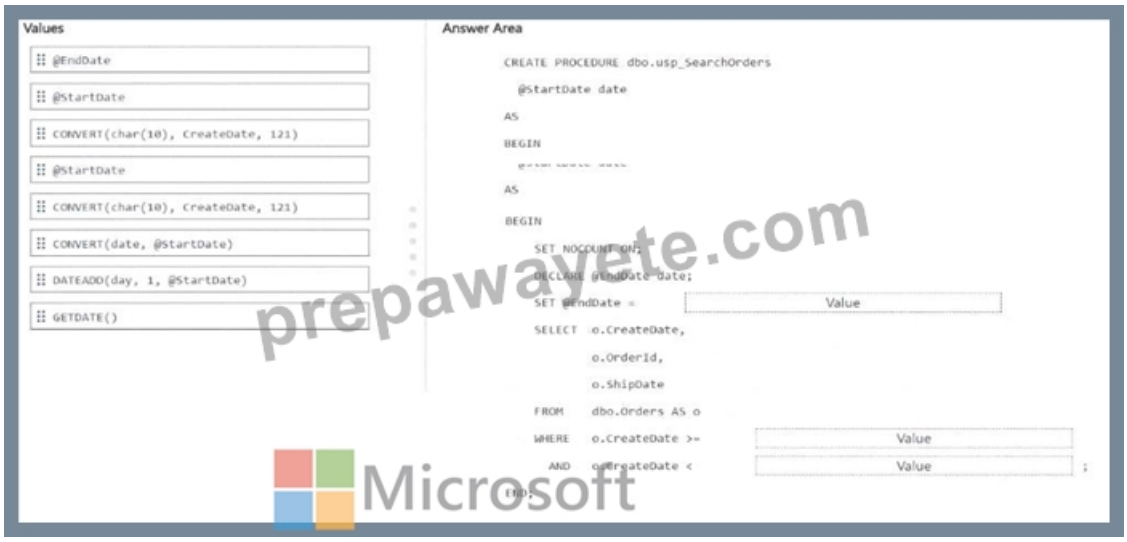

What should you insert at line 22?

- A. FOR XHL AUTO, TYPE, XML SCHEMA,
- B. FOR 3SON AUTO, IMCLUDE_MULL_VAIUES
- C. FOR JSON PATH, WITHOUT_ARRAY_WRAPPER
- D. FOR XHL PATH, INCLUDE_NULL_VAIUES

Answer: C

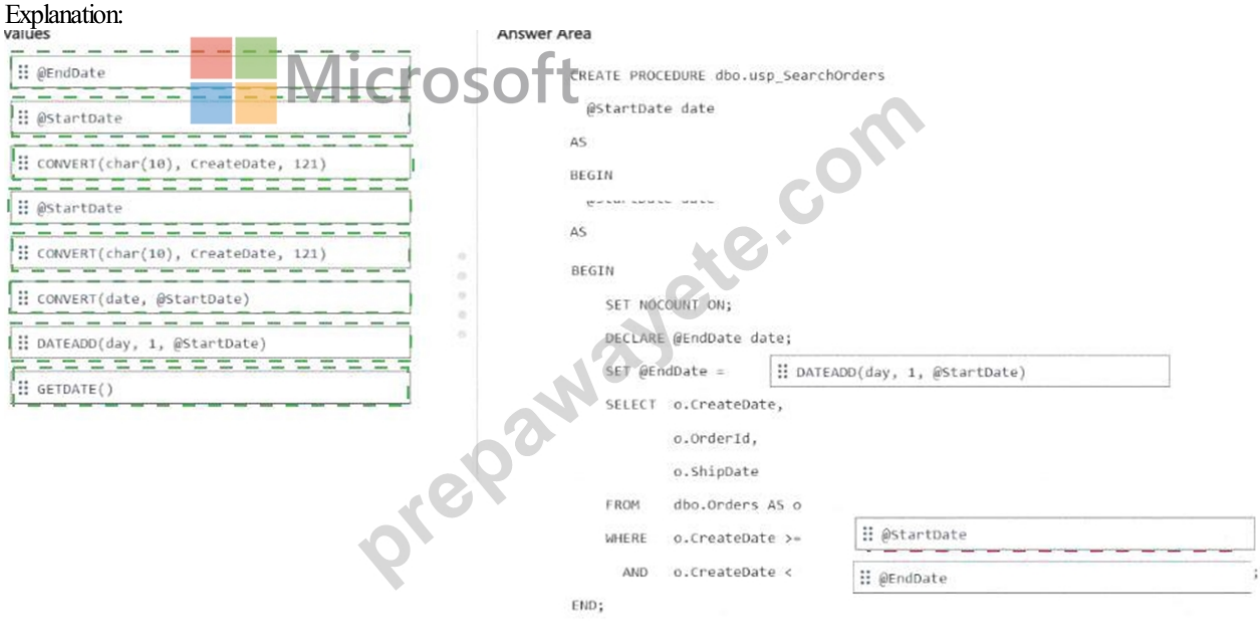
Explanation:
The correct insertion at line 22 is FOR JSON PATH, WITHOUT_ARRAY_WRAPPER .
The request body for the Azure OpenAI chat completions call must be a single JSON object containing the messages array with both the system/user content and the retrieved chunks. Microsoft documents that FOR JSON 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.

NEW QUESTION # 84
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.

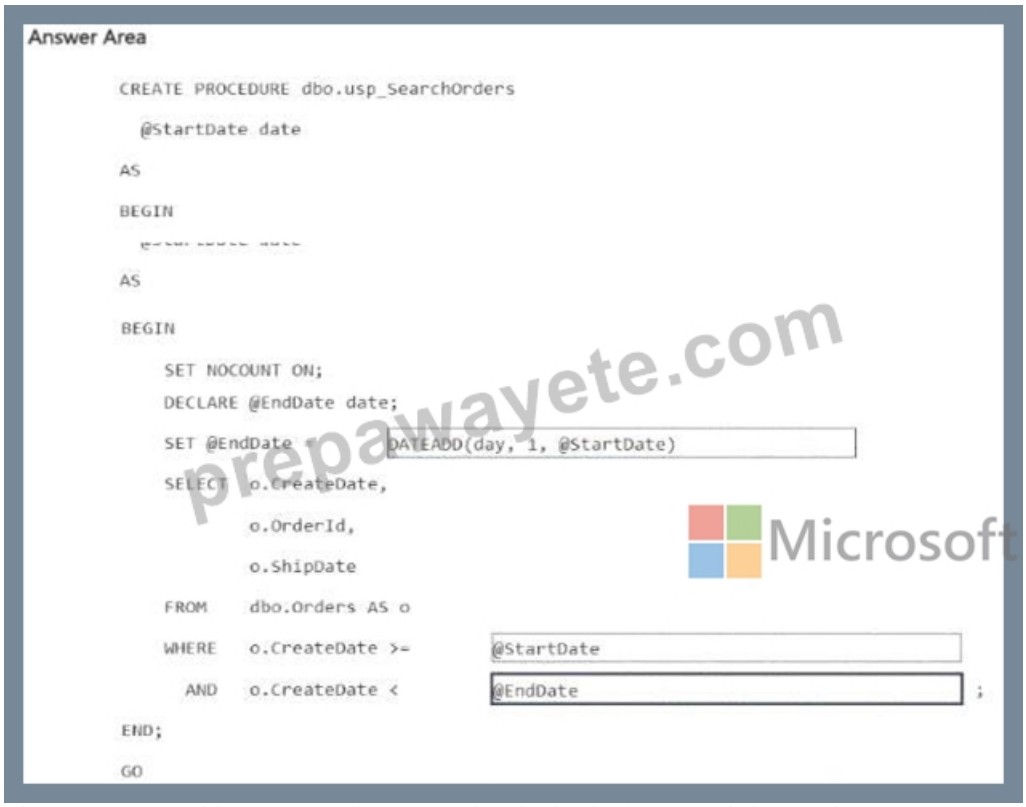


Answer:

Explanation:



Explanation:

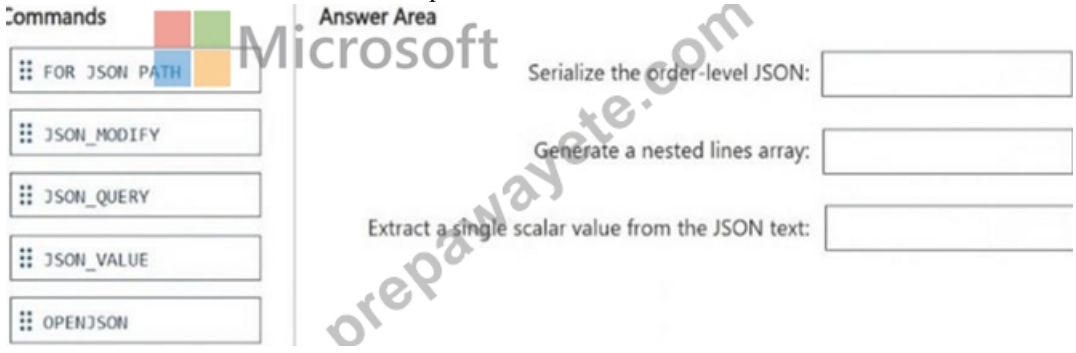


The correct SARGable pattern for filtering a single calendar day is to use a half-open date range :
o.CreateDate > = @StartDate
AND o.CreateDate < @EndDate
with:
SET @EndDate = DATEADD(day, 1, @StartDate)
This is the correct design because it keeps the function off the column and applies it only to the parameter.
That allows SQL Server and Azure SQL to use an index on CreateDate efficiently, which is the key requirement for a SARGable predicate. Microsoft documents DATEADD as the standard function for adding one day to a date value, which makes it the right way to derive the exclusive upper boundary for the next day.
The incorrect choices are the ones that wrap CreateDate in CONVERT(...), because expressions like:
CONVERT(char(10), CreateDate, 121) = ...
make the predicate non-SARGable and typically prevent efficient seeks on an index over CreateDate.
So the completed procedure is:
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;

NEW QUESTION # 85
Drag and Drop Question
You have an Azure SQL database named SalesDB that contains tables named Sales.Orders and Sales.OrderLines. Both tables contain sales data.
You have a Retrieval Augmented Generation (RAG) service that queries SalesDB to retrieve order details and passes the results to a large language model (LLM) as JSON text. The following is a simple of the JSON.

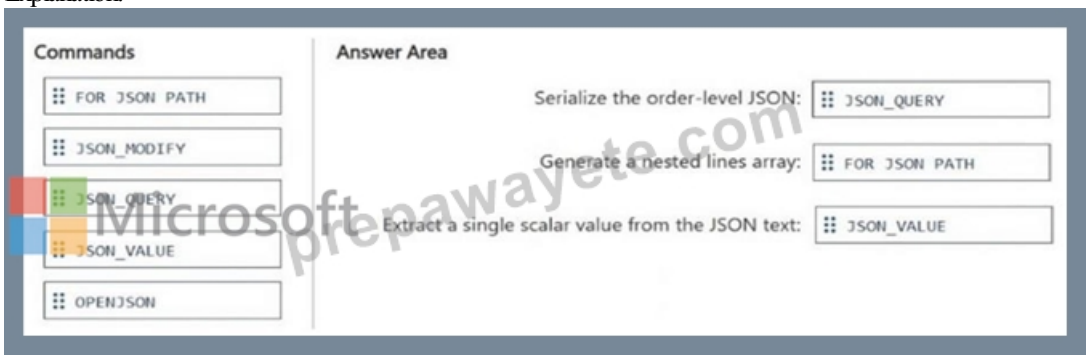
```
[
  {
    "orderHeaderId": 102348,
    "orderNumber": "SO-2026-000912",
    "orderDateUtc": "2026-01-28T14:22:09Z",
    "customerId": 77821,
    "currencyCode": "EUR",
    "orderTotal": 149.97,
    "lines": [
      {
        "lineNumber": 1,
        "productId": 5012,
        "sku": "CBL-USB-C-1M",
        "quantity": 2,
        "unitPrice": 19.99,
        "lineTotal": 39.98
      },
      {
        "lineNumber": 2,
        "productId": 8841,
        "sku": "HUB-USB-C-7P",
        "quantity": 1,
        "unitPrice": 109.99,
        "lineTotal": 109.99
      }
    ]
  }
]
```


You need to return one JSON document per order that includes the order header fields and an array of related order lines. The LLM must receive a single JSON array of orders, where each order contains a lines property that is a JSON array of line items.
Which Transact-SQL commands should you use to produce the required JSON shape from the relational tables? To answer, drag the appropriate commands to the correct operations. Each command 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:

Explanation:



NEW QUESTION # 86
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.
```
- name: Publish
  run: |
    dotnet build db1.sqlproj \
      /p:TargetConnectionString="${{ secrets.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.
```
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 }}
```
- D.
```
- name: Publish
  uses: azure/sql-action@v2
  with:
    action: publish
    path: bin/Debug/db1.dacpac
    connection-string: ${{ secrets.SQL_CONNECTION_STRING }}
```

Answer: D

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 .

NEW QUESTION # 87
.....