Reliable Microsoft - DP-800 - New Developing AI-Enabled Database Solutions Exam Practice

At this moment, our company has been regarded as the best retailer of the DP-800 study materials. We are responsible for every customer. Your satisfactions on our DP-800 exam braindumps are our great motivation. In addition, all people have the right to enjoy our good pre-sale and after sale service on our DP-800 training guide. We warmly welcome every customer to select our DP-800 learning questions.

Microsoft DP-800 Exam Syllabus Topics:

Topic	Details
Topic 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.
Topic 2	• 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 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.

>> New DP-800 Exam Practice <<

Pass Guaranteed Accurate Microsoft - DP-800 - New Developing AI-Enabled Database Solutions Exam Practice

Do you want to catch up with the trend in the IT industry? Being certified by Microsoft DP-800 exam certification means a large possibility of success. While our DP-800 exam targeted training will help you step ahead of others. The valid DP-800 study practice will make your thoughts more clear, and you will have the ability to deal with problem in the practical application. Then, passing the DP-800 Actual Test is an easy and simple thing. If you still have some doubts, please download BraindumpsVCE DP-800 free demo for a try. You will be surprised.

Microsoft Developing AI-Enabled Database Solutions Sample Questions (Q77-Q82):

NEW QUESTION # 77

Which tool helps orchestrate data pipelines for AI workflows?

- A. Azure Data Factory
- B. Azure Monitor
- C. Power BI
- D. Azure DevOps

Answer: A

Explanation:
Azure Data Factory automates data movement and transformation pipelines.

NEW QUESTION # 78
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an SDK-style SQL database project stored in a Git repository. The project targets an Azure SQL database.
The CI build fails with unresolved reference errors when the project references system objects.
You need to update the SQL database project to ensure that dotnet build validates successfully by including the correct system objects in the database model for Azure SQL Database.
Solution: Add the Microsoft.SqlServer.Dacpacs.Azure.Master NuGet package to the project.
Does this meet the goal?

- A. Yes
- B. No

Answer: A

Explanation:
Correct:
* Add the Microsoft.SqlServer.Dacpacs.Azure.Master NuGet package to the project.
To resolve system reference errors in an SDK-style SQL project targeting Azure SQL Database, you need to add a reference to the Microsoft.SqlServer.Dacpacs.Azure.Master NuGet package.
In your .sqlproj file, include the following item group:
<ItemGroup>
<PackageReference Include="Microsoft.SqlServer.Dacpacs.Azure.Master" Version="1.60.0" />
</ItemGroup>
Why this works:
System Objects: Standard SDK-style projects don't automatically include system views (like sys.database_principals or sys.dm_db_resource_stats). This package provides the necessary metadata for the compiler.
Azure Specifics: It includes Azure-only system objects that aren't present in the standard master database dacpac used for on-premises SQL Server.
CI/CD Friendly: Since it is a NuGet package, the dotnet build command will automatically restore it during the CI process without requiring manual file paths or local installations of Visual Studio.
Incorrect:
* Add an artifact reference to the Azure SQL Database master.dacpac file.
* Add the Microsoft.SqlServer.Dacpacs.Master NuGet package to the project.
Reference:
https://learn.microsoft.com/en-us/sql/tools/sql-database-projects/concepts/system-objects

NEW QUESTION # 79
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: ${{ env.SQL_CONNECTION_STRING }}
```
- B.
```
- name: Publish
  run: |
    dotnet build db1.sqlproj \
      /p:TargetConnectionString="${{ 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
  uses: azure/sql-action@v2
  with:
    action: extract
    path: bin/Debug/db1.dacpac
    connection-string: ${{ secrets.SQL_CONNECTION_STRING }}
```

Answer: C

Explanation:
To deploy your .dacpac to Azure SQL using GitHub Actions, you should use the official azure/sql- action@v2. This action is designed specifically to take the output of your dotnet build and publish it.
Assuming your build step is already working, here is how you structure the deployment job.
GitHub Actions Workflow Snippet
Add this job to your .yml file. It depends on your build job (usually named build) and runs on a runner with the Azure CLI installed (like ubuntu-latest).
- name: Deploy SQL Schema
uses: azure/sql-action@v2
with:
connection-string: ${{ secrets.AZURE_SQL_CONNECTION_STRING }}
path: './bin/Release/netstandard2.1/YourDatabase.dacpac' # Path to your .dacpac action: 'publish' Incorrect:
[Not A]
Use connection-string: ${{ secrets.AZURE_SQL_CONNECTION_STRING }}
[Not B]
Action set to publish, not to extract.
[Not D]
Use azure/sql-action@v2.
Reference:
https://stackoverflow.com/questions/75490960/github-actions-dotnet-publish-specify-a-project- with-a-dot-in-the-name

NEW QUESTION # 80
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.

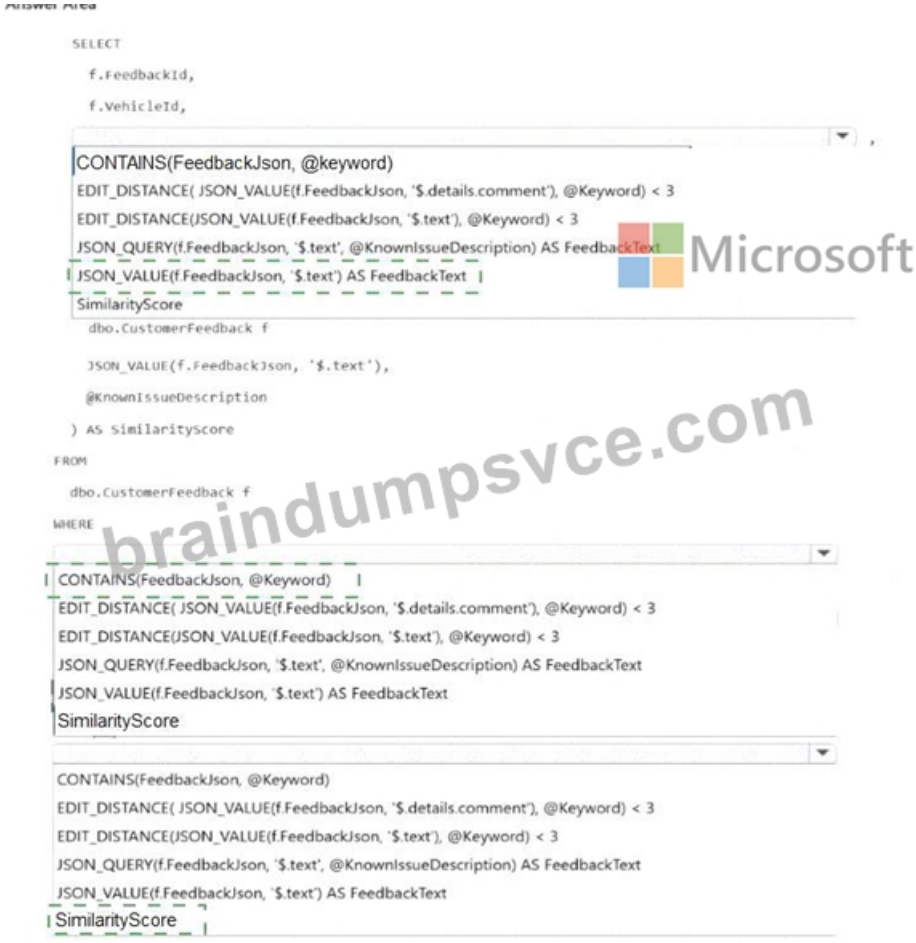
Answer Area

```
SELECT
  f.FeedbackId,
  f.VehicleId,
  [ ],
  dbo.CustomerFeedback f
  JSON_VALUE(f.FeedbackJson, '$.text'),
  @KnownIssueDescription
) AS SimilarityScore
FROM
  dbo.CustomerFeedback f
WHERE
  [ ]
  [ ]
```

- CONTAINS(FeedbackJson, @keyword)
- EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.details.comment'), @Keyword) < 3
- EDIT_DISTANCE(JSON_VALUE(f.FeedbackJson, '$.text'), @Keyword) < 3
- JSON_QUERY(f.FeedbackJson, '$.text', @KnownIssueDescription) AS FeedbackText
- JSON_VALUE(f.FeedbackJson, '$.text') AS FeedbackText
- SimilarityScore

Answer:

Explanation:



Explanation:
JSON_VALUE(f.FeedbackJson, ' $.text ') AS FeedbackText
CONTAINS(FeedbackJson, @Keyword)
SimilarityScore
These three selections are the correct way to complete the query because they align exactly with the stated requirements for the FeedbackJson column.
First, to extract the customer feedback text from the JSON document , the correct expression is JSON_VALUE(f.FeedbackJson, ' $.text ') AS FeedbackText . Microsoft documents that JSON_VALUE is used to extract a scalar value from JSON, while JSON_QUERY is used for returning an object or array .
Since $.text is the textual feedback string, JSON_VALUE is the correct function.
Second, to filter rows where the JSON text contains a keyword , the best choice is CONTAINS (FeedbackJson, @Keyword) . The scenario explicitly states that FeedbackJson already has a full-text index
, and Microsoft documents that CONTAINS is the full-text predicate used in the WHERE clause to search full-text indexed character data. That makes it more appropriate than using EDIT_DISTANCE for keyword filtering.
Third, to order the results by similarity score, highest first , the correct item is SimilarityScore in the ORDER BY clause, which would be paired with DESC in the query. This matches the requirement to sort by the computed fuzzy similarity value. The DP-800 study guide specifically includes writing queries that use fuzzy string matching functions such as EDIT_DISTANCE, which supports the earlier computed SimilarityScore expression in the query.

NEW QUESTION # 81
You have a SQL database in Microsoft Fabric that contains the following functions:
* A multi-statement table-valued function (TVF) named sales.mstvf_orderStatus() that returns order status information
* A scalar user-defined function (UOF) named dbo.ufn_GetlaxMultiplier(