Free Questions for DP-420 by certsdeals

Shared by Turner on 06-06-2022

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

You have an Azure Cosmos DB Core (SQL) API account that is used by 10 web apps.

You need to analyze the data stored in the account by using Apache Spark to create machine learning models. The solution must NOT affect the performance of the web apps.

Which two actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

Options:

A- In an Apache Spark pool in Azure Synapse, create a table that uses cosmos.olap as the data source.

B- Create a private endpoint connection to the account.

C- In an Azure Synapse Analytics serverless SQL pool, create a view that uses OPENROWSET and the CosmosDB provider.

D- Enable Azure Synapse Link for the account and Analytical store on the container.

E- In an Apache Spark pool in Azure Synapse, create a table that uses cosmos.oltp as the data source.

Answer:

A, D

Explanation:

https://github.com/microsoft/MCW-Cosmos-DB-Real-Time-Advanced-Analytics/blob/main/Hands-on%20lab/HOL%20step-by%20step%20-%20Cosmos%20DB%20real-time%20advanced%20analytics.md

Question 2

Question Type: MultipleChoice

You have a database in an Azure Cosmos DB Core (SQL) API account.

You need to create an Azure function that will access the database to retrieve records based on a variable named accountnumber. The solution must protect against SQL injection attacks.

How should you define the command statement in the function?

Options:

A- cmd = 'SELECT * FROM Persons p

WHERE p.accountnumber = 'accountnumber''

B- cmd = 'SELECT * FROM Persons p
WHERE p.accountnumber = LIKE @accountnumber'

C- cmd = 'SELECT * FROM Persons p
WHERE p.accountnumber = @accountnumber'

D- cmd = 'SELECT * FROM Persons p
WHERE p.accountnumber = '' + accountnumber + '''

Answer:

C

Explanation:

Azure Cosmos DB supports queries with parameters expressed by the familiar @ notation. Parameterized SQL provides robust handling and escaping of user input, and prevents accidental exposure of data through SQL injection.

For example, you can write a query that takes lastName and address.state as parameters, and execute it for various values of lastName and address.state based on user input.

SELECT *

FROM Families f

WHERE f.lastName = @lastName AND f.address.state = @addressState

Question 3

Question Type: MultipleChoice

You have a database in an Azure Cosmos DB Core (SQL) API account. The database is backed up every two hours.

You need to implement a solution that supports point-in-time restore.

What should you do first?

Options:

A- Enable Continuous Backup for the account.

B- Configure the Backup & Restore settings for the account.

C- Create a new account that has a periodic backup policy.

D- Configure the Point In Time Restore settings for the account.

Answer:

A

Question 4

Question Type: MultipleChoice

You have an Azure Cosmos DB Core (SQL) API account.

You configure the diagnostic settings to send all log information to a Log Analytics workspace.

You need to identify when the provisioned request units per second (RU/s) for resources within the account were modified.

You write the following query.

AzureDiagnostics

| where Category == "ControlPlaneRequests"

What should you include in the query?

Options:

A- | where OperationName startswith 'AccountUpdateStart'

B- | where OperationName startswith 'SqlContainersDelete'

C- | where OperationName startswith 'MongoCollectionsThroughputUpdate'

D- | where OperationName startswith 'SqlContainersThroughputUpdate'

Answer:

A

Explanation:

The following are the operation names in diagnostic logs for different operations:

RegionAddStart, RegionAddComplete

RegionRemoveStart, RegionRemoveComplete

AccountDeleteStart, AccountDeleteComplete

RegionFailoverStart, RegionFailoverComplete

AccountCreateStart, AccountCreateComplete

AccountUpdateStart, AccountUpdateComplete

VirtualNetworkDeleteStart, VirtualNetworkDeleteComplete

DiagnosticLogUpdateStart, DiagnosticLogUpdateComplete

Question 5

Question Type: MultipleChoice

You have a container named container1 in an Azure Cosmos DB Core (SQL) API account.

You need to provide a user named User1 with the ability to insert items into container1 by using role-based access control (RBAC). The solution must use the principle of least privilege.

Which roles should you assign to User1?

Options:

A- CosmosDB Operator only

B- DocumentDB Account Contributor and Cosmos DB Built-in Data Contributor

C- DocumentDB Account Contributor only

D- Cosmos DB Built-in Data Contributor only

Answer:

A

Explanation:

Cosmos DB Operator: Can provision Azure Cosmos accounts, databases, and containers. Cannot access any data or use Data Explorer.

Incorrect Answers:

B: DocumentDB Account Contributor can manage Azure Cosmos DB accounts. Azure Cosmos DB is formerly known as DocumentDB.

C: DocumentDB Account Contributor: Can manage Azure Cosmos DB accounts.

Question 6

Question Type: MultipleChoice

You are implementing an Azure Data Factory data flow that will use an Azure Cosmos DB (SQL API) sink to write a dataset. The data flow will use 2,000 Apache Spark partitions.

You need to ensure that the ingestion from each Spark partition is balanced to optimize throughput.

Which sink setting should you configure?

Options:

A- Throughput

B- Write throughput budget

C- Batch size

D- Collection action

Answer:

C

Explanation:

Batch size: An integer that represents how many objects are being written to Cosmos DB collection in each batch. Usually, starting with the default batch size is sufficient. To further tune this value, note:

Cosmos DB limits single request's size to 2MB. The formula is 'Request Size = Single Document Size * Batch Size'. If you hit error saying 'Request size is too large', reduce the batch size value.

The larger the batch size, the better throughput the service can achieve, while make sure you allocate enough RUs to empower your workload.

Incorrect Answers:

A: Throughput: Set an optional value for the number of RUs you'd like to apply to your CosmosDB collection for each execution of this data flow. Minimum is 400.

B: Write throughput budget: An integer that represents the RUs you want to allocate for this Data Flow write operation, out of the total throughput allocated to the collection.

D: Collection action: Determines whether to recreate the destination collection prior to writing.

None: No action will be done to the collection.

Recreate: The collection will get dropped and recreated

Question 7

Question Type: MultipleChoice

You need to configure an Apache Kafka instance to ingest data from an Azure Cosmos DB Core (SQL) API account. The data from a container named telemetry must be added to a Kafka topic named iot. The solution must store the data in a compact binary format.

Which three configuration items should you include in the solution? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

Options:

A- 'connector.class': 'com.azure.cosmos.kafka.connect.source.CosmosDBSourceConnector'

B- 'key.converter': 'org.apache.kafka.connect.json.JsonConverter'

C- 'key.converter': 'io.confluent.connect.avro.AvroConverter'

D- 'connect.cosmos.containers.topicmap': 'iot#telemetry'

E- 'connect.cosmos.containers.topicmap': 'iot'

F- 'connector.class': 'com.azure.cosmos.kafka.connect.source.CosmosDBSinkConnector'

Answer:

C, D, F

Explanation:

C: Avro is binary format, while JSON is text.

F: Kafka Connect for Azure Cosmos DB is a connector to read from and write data to Azure Cosmos DB. The Azure Cosmos DB sink connector allows you to export data from Apache Kafka topics to an Azure Cosmos DB database. The connector polls data from Kafka to write to containers in the database based on the topics subscription.

D: Create the Azure Cosmos DB sink connector in Kafka Connect. The following JSON body defines config for the sink connector.

Extract:

'connector.class': 'com.azure.cosmos.kafka.connect.sink.CosmosDBSinkConnector',

'key.converter': 'org.apache.kafka.connect.json.AvroConverter'

'connect.cosmos.containers.topicmap': 'hotels#kafka'

Incorrect Answers:

B: JSON is plain text.

Note, full example:

{

'name': 'cosmosdb-sink-connector',

'config': {

'connector.class': 'com.azure.cosmos.kafka.connect.sink.CosmosDBSinkConnector',

'tasks.max': '1',

'topics': [

'hotels'

],

'value.converter': 'org.apache.kafka.connect.json.AvroConverter',

'value.converter.schemas.enable': 'false',

'key.converter': 'org.apache.kafka.connect.json.AvroConverter',

'key.converter.schemas.enable': 'false',

'connect.cosmos.connection.endpoint': 'Error! Hyperlink reference not valid.',

'connect.cosmos.master.key': '<cosmosdbprimarykey>',

'connect.cosmos.databasename': 'kafkaconnect',

'connect.cosmos.containers.topicmap': 'hotels#kafka'

}

}

https://docs.microsoft.com/en-us/azure/cosmos-db/sql/kafka-connector-sink

https://www.confluent.io/blog/kafka-connect-deep-dive-converters-serialization-explained/

Question 8

Question Type: **MultipleChoice**

You plan to create an Azure Cosmos DB Core (SQL) API account that will use customer-managed keys stored in Azure Key Vault.

You need to configure an access policy in Key Vault to allow Azure Cosmos DB access to the keys.

Which three permissions should you enable in the access policy? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

Options:

A- Wrap Key

B- Get

C- List

D- Update

E- Sign

F- Verify

G- Unwrap Key

Answer:

A, B, G

Question 9

Question Type: MultipleChoice

You have the following query.

```
SELECT * FROM

WHERE c.sensor = "TEMP1"

AND c.value

AND c.timestamp >= 1619146031231
```

You need to recommend a composite index strategy that will minimize the request units (RUs) consumed by the query.

What should you recommend?

Options:

A- a composite index for (sensor ASC, value ASC) and a composite index for (sensor ASC, timestamp ASC)

B- a composite index for (sensor ASC, value ASC, timestamp ASC) and a composite index for (sensor DESC, value DESC, timestamp DESC)

C- a composite index for (value ASC, sensor ASC) and a composite index for (timestamp ASC, sensor ASC)

D- a composite index for (sensor ASC, value ASC, timestamp ASC)

Answer:

A

Explanation:

If a query has a filter with two or more properties, adding a composite index will improve performance.

Consider the following query:

SELECT * FROM c WHERE c.name = "Tim" and c.age > 18

In the absence of a composite index on (name ASC, and age ASC), we will utilize a range index for this query. We can improve the efficiency of this query by creating a composite index for name and age.

Queries with multiple equality filters and a maximum of one range filter (such as >,<, <=, >=, !=) will utilize the composite index.