



Free Amazon DEA-C01 Practice Questions

Shared by Conner on 12-08-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page



Question 1

Question Type: MultipleChoice

A financial services company stores financial data in Amazon Redshift. A data engineer wants to run real-time queries on the financial data to support a web-based trading application. The data engineer wants to run the queries from within the trading application.

Which solution will meet these requirements with the LEAST operational overhead?

Options:

- A- Establish WebSocket connections to Amazon Redshift.
- B- Use the Amazon Redshift Data API.
- C- Set up Java Database Connectivity (JDBC) connections to Amazon Redshift.
- D- Store frequently accessed data in Amazon S3. Use Amazon S3 Select to run the queries.

Answer:

B

Explanation:

The Amazon Redshift Data API is a built-in feature that allows you to run SQL queries on Amazon Redshift data with web services-based applications, such as AWS Lambda, Amazon SageMaker notebooks, and AWS Cloud9. The Data API does not require a persistent connection to your database, and it provides a secure HTTP endpoint and integration with AWS SDKs. You can use the endpoint to run SQL statements without managing connections. The Data API also supports both Amazon Redshift provisioned clusters and Redshift Serverless workgroups. The Data API is the best solution for running real-time queries on the financial data from within the trading application, as it has the least operational overhead compared to the other options.

Option A is not the best solution, as establishing WebSocket connections to Amazon Redshift would require more configuration and maintenance than using the Data API. WebSocket connections are also not supported by Amazon Redshift clusters or serverless workgroups.

Option C is not the best solution, as setting up JDBC connections to Amazon Redshift would also require more configuration and maintenance than using the Data API. JDBC connections are also not supported by Redshift Serverless workgroups.

Option D is not the best solution, as storing frequently accessed data in Amazon S3 and using Amazon S3 Select to run the queries would introduce additional latency and complexity than using the Data API. Amazon S3 Select is also not optimized for real-time queries, as it scans the entire object before returning the results. Reference:

Using the Amazon Redshift Data API

Calling the Data API

Amazon Redshift Data API Reference

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Question 2

Question Type: MultipleChoice

A company uploads .csv files to an Amazon S3 bucket. The company's data platform team has set up an AWS Glue crawler to perform data discovery and to create the tables and schemas.

An AWS Glue job writes processed data from the tables to an Amazon Redshift database. The AWS Glue job handles column mapping and creates the Amazon Redshift tables in the Redshift database appropriately.

If the company reruns the AWS Glue job for any reason, duplicate records are introduced into the Amazon Redshift tables. The company needs a solution that will update the Redshift tables without duplicates.

Which solution will meet these requirements?

Options:

- A- Modify the AWS Glue job to copy the rows into a staging Redshift table. Add SQL commands to update the existing rows with new values from the staging Redshift table.
- B- Modify the AWS Glue job to load the previously inserted data into a MySQL database. Perform an upsert operation in the MySQL database. Copy the results to the Amazon Redshift tables.
- C- Use Apache Spark's DataFrame dropDuplicates() API to eliminate duplicates. Write the data to the Redshift tables.
- D- Use the AWS Glue ResolveChoice built-in transform to select the value of the column from the most recent record.

Answer:

D

Question 3

Question Type: MultipleChoice

A company is planning to upgrade its Amazon Elastic Block Store (Amazon EBS) General Purpose SSD storage from gp2 to gp3. The company wants to prevent any interruptions in its Amazon EC2 instances that will cause data loss during the migration to the upgraded storage.

Which solution will meet these requirements with the LEAST operational overhead?

Options:

- A- Create snapshots of the gp2 volumes. Create new gp3 volumes from the snapshots. Attach the new gp3 volumes to the EC2 instances.
- B- Create new gp3 volumes. Gradually transfer the data to the new gp3 volumes. When the transfer is complete, mount the new gp3 volumes to the EC2 instances to replace the gp2 volumes.
- C- Change the volume type of the existing gp2 volumes to gp3. Enter new values for volume size, IOPS, and throughput.
- D- Use AWS DataSync to create new gp3 volumes. Transfer the data from the original gp2 volumes to the new gp3 volumes.

Answer:

C

Explanation:

Changing the volume type of the existing gp2 volumes to gp3 is the easiest and fastest way to migrate to the new storage type without any downtime or data loss. You can use the AWS Management Console, the AWS CLI, or the Amazon EC2 API to modify the volume type, size, IOPS, and throughput of your gp2 volumes. The modification takes effect immediately, and you can monitor the progress of the modification using CloudWatch. The other options are either more complex or require additional steps, such as creating snapshots, transferring data, or attaching new volumes, which can increase the operational overhead and the risk of errors. Reference:

Migrating Amazon EBS volumes from gp2 to gp3 and save up to 20% on costs(Section: How to migrate from gp2 to gp3)

Switching from gp2 Volumes to gp3 Volumes to Lower AWS EBS Costs(Section: How to Switch from GP2 Volumes to GP3 Volumes)

Modifying the volume type, IOPS, or size of an EBS volume - Amazon Elastic Compute Cloud(Section: Modifying the volume type)

Question 4

Question Type: MultipleChoice

A company is building an analytics solution. The solution uses Amazon S3 for data lake storage and Amazon Redshift for a data warehouse. The company wants to use Amazon Redshift Spectrum to query the data that is in Amazon S3.

Which actions will provide the FASTEST queries? (Select two.)

Options:

- A- Use gzip compression to compress individual files to sizes that are between 1 GB and 5 GB.
- B- Use a columnar storage file format.
- C- Partition the data based on the most common query predicates.
- D- Split the data into files that are less than 10 KB.
- E- Use file formats that are not

Answer:

B, C

Explanation:

Amazon Redshift Spectrum is a feature that allows you to run SQL queries directly against data in Amazon S3, without loading or transforming the data. Redshift Spectrum can query various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.

On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance.

Therefore, using a columnar storage file format, such as Parquet, will provide faster queries, as it allows Redshift Spectrum to scan only the relevant columns and skip the rest, reducing the

amount of data read from S3. Additionally, partitioning the data based on the most common query predicates, such as date, time, region, etc., will provide faster queries, as it allows Redshift Spectrum to prune the partitions that do not match the query criteria, reducing the amount of data scanned from S3. Partitioning also improves the performance of joins and aggregations, as it reduces data skew and shuffling.

The other options are not as effective as using a columnar storage file format and partitioning the data. Using gzip compression to compress individual files to sizes that are between 1 GB and 5 GB will reduce the data size, but it will not improve the query performance significantly, as gzip is not a splittable compression algorithm and requires decompression before reading. Splitting the data into files that are less than 10 KB will increase the number of files and the metadata overhead, which will degrade the query performance. Using file formats that are not supported by Redshift Spectrum, such as XML, will not work, as Redshift Spectrum will not be able to read or parse the data. Reference:

Amazon Redshift Spectrum

Choosing the Right Data Format

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Lakes and Data Warehouses, Section 4.3: Amazon Redshift Spectrum

Question 5

Question Type: MultipleChoice

A data engineer must build an extract, transform, and load (ETL) pipeline to process and load data from 10 source systems into 10 tables that are in an Amazon Redshift database. All the source systems generate .csv, JSON, or Apache Parquet files every 15 minutes. The source systems all deliver files into one Amazon S3 bucket. The file sizes range from 10 MB to 20 GB. The ETL pipeline must function correctly despite changes to the data schema.

Which data pipeline solutions will meet these requirements? (Select two.)

Options:

- A- Use an Amazon EventBridge rule to run an AWS Glue job every 15 minutes. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
- B- Use an Amazon EventBridge rule to invoke an AWS Glue workflow job every 15 minutes. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
- C- Configure an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket. Configure an AWS Glue job to process and load the data into the Amazon Redshift tables. Create a second Lambda function to run the AWS Glue job. Create an Amazon

EventBridge rule to invoke the second Lambda function when the AWS Glue crawler finishes running successfully.

D- Configure an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.

E- Configure an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket. Configure the AWS Glue job to read the files from the S3 bucket into an Apache Spark DataFrame. Configure the AWS Glue job to also put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream. Configure the delivery stream to load data into the Amazon Redshift tables.

Answer:

A, B

Explanation:

Using an Amazon EventBridge rule to run an AWS Glue job or invoke an AWS Glue workflow job every 15 minutes are two possible solutions that will meet the requirements. AWS Glue is a serverless ETL service that can process and load data from various sources to various targets, including Amazon Redshift. AWS Glue can handle different data formats, such as CSV, JSON, and Parquet, and also support schema evolution, meaning it can adapt to changes in the data schema over time. AWS Glue can also leverage Apache Spark to perform distributed processing and transformation of large datasets. AWS Glue integrates with Amazon EventBridge, which is a serverless event bus service that can trigger actions based on rules and schedules. By using an Amazon EventBridge rule, you can invoke an AWS Glue job or workflow every 15 minutes, and configure the job or workflow to run an AWS Glue crawler and then load the data into the Amazon Redshift tables. This way, you can build a cost-effective and scalable ETL pipeline that can handle data from 10 source systems and function correctly despite changes to the data schema.

The other options are not solutions that will meet the requirements. Option C, configuring an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket, and creating a second Lambda function to run the AWS Glue job, is not a feasible solution, as it would require a lot of Lambda invocations and coordination. AWS Lambda has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the ETL pipeline. Option D, configuring an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket, is not a necessary solution, as you can use an Amazon EventBridge rule to invoke the AWS Glue workflow directly, without the need for a Lambda function. Option E, configuring an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket, and configuring the AWS Glue job to put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream, is not a cost-effective solution, as it would incur additional costs for Lambda invocations and data delivery. Moreover, using Amazon Kinesis Data Firehose to load data into Amazon Redshift is not suitable for frequent and small batches of data, as it can cause performance issues and data fragmentation. Reference:

AWS Glue

Amazon EventBridge

Using AWS Glue to run ETL jobs against non-native JDBC data sources

[AWS Lambda quotas]

[Amazon Kinesis Data Firehose quotas]

Question 6

Question Type: MultipleChoice

A data engineer has a one-time task to read data from objects that are in Apache Parquet format in an Amazon S3 bucket. The data engineer needs to query only one column of the data.

Which solution will meet these requirements with the LEAST operational overhead?

Options:

- A- Configure an AWS Lambda function to load data from the S3 bucket into a pandas dataframe- Write a SQL SELECT statement on the dataframe to query the required column.
- B- Use S3 Select to write a SQL SELECT statement to retrieve the required column from the S3 objects.
- C- Prepare an AWS Glue DataBrew project to consume the S3 objects and to query the required column.
- D- Run an AWS Glue crawler on the S3 objects. Use a SQL SELECT statement in Amazon Athena to query the required column.

Answer:

B

Explanation:

Option B is the best solution to meet the requirements with the least operational overhead because S3 Select is a feature that allows you to retrieve only a subset of data from an S3 object by using simple SQL expressions. S3 Select works on objects stored in CSV, JSON, or Parquet format. By using S3 Select, you can avoid the need to download and process the entire S3 object, which reduces the amount of data transferred and the computation time. S3 Select is also easy to use and does not require any additional services or resources.

Option A is not a good solution because it involves writing custom code and configuring an AWS

Lambda function to load data from the S3 bucket into a pandas dataframe and query the required column. This option adds complexity and latency to the data retrieval process and requires additional resources and configuration. Moreover, AWS Lambda has limitations on the execution time, memory, and concurrency, which may affect the performance and reliability of the data retrieval process.

Option C is not a good solution because it involves creating and running an AWS Glue DataBrew project to consume the S3 objects and query the required column. AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. However, in this scenario, the data is already in Parquet format, which is a columnar storage format that is optimized for analytics. Therefore, there is no need to use AWS Glue DataBrew to prepare the data. Moreover, AWS Glue DataBrew adds extra time and cost to the data retrieval process and requires additional resources and configuration.

Option D is not a good solution because it involves running an AWS Glue crawler on the S3 objects and using a SQL SELECT statement in Amazon Athena to query the required column. An AWS Glue crawler is a service that can scan data sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. However, in this scenario, the schema and format of the data are already known and fixed, so there is no need to run a crawler to discover them. Moreover, running a crawler and using Amazon Athena adds extra time and cost to the data retrieval process and requires additional services and configuration.

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

S3 Select and Glacier Select - Amazon Simple Storage Service

AWS Lambda - FAQs

What Is AWS Glue DataBrew? - AWS Glue DataBrew

Populating the AWS Glue Data Catalog - AWS Glue

What is Amazon Athena? - Amazon Athena

Question 7

Question Type: MultipleChoice

A data engineer is building a data pipeline. A large data file is uploaded to an Amazon S3 bucket once each day at unpredictable times. An AWS Glue workflow uses hundreds of workers to process the file and load the data into Amazon Redshift. The company wants to process the file as quickly as possible.

Which solution will meet these requirements?

Options:

- A-** Create an on-demand AWS Glue trigger to start the workflow. Create an AWS Lambda function that runs every 15 minutes to check the S3 bucket for the daily file. Configure the function to start the AWS Glue workflow if the file is present.
- B-** Create an event-based AWS Glue trigger to start the workflow. Configure Amazon S3 to log events to AWS CloudTrail. Create a rule in Amazon EventBridge to forward PutObject events to the AWS Glue trigger.
- C-** Create a scheduled AWS Glue trigger to start the workflow. Create a cron job that runs the AWS Glue job every 15 minutes. Set up the AWS Glue job to check the S3 bucket for the daily file. Configure the job to stop if the file is not present.
- D-** Create an on-demand AWS Glue trigger to start the workflow. Create an AWS Database Migration Service (AWS DMS) migration task. Set the DMS source as the S3 bucket. Set the target endpoint as the AWS Glue workflow.

Answer:

B

Explanation:

The best solution for fast, event-driven processing of unpredictable file uploads is to use S3 event notifications, CloudTrail, and EventBridge to automatically trigger the AWS Glue workflow:

"You can configure S3 PutObject events to be captured by CloudTrail and forwarded through EventBridge to trigger an AWS Glue job or workflow. This allows Glue to begin processing as soon as the file arrives, with minimal latency."

-- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

This option provides the lowest latency and least manual overhead compared to polling or scheduling solutions.

To Get Premium Files for DEA-C01 Visit

<https://www.p2pexams.com/products/dea-c01>

For More Free Questions Visit

<https://www.p2pexams.com/amazon/pdf/dea-c01>

20%
DISCOUNT

P2P
exams