DEA-C01 Q&As

SnowPro Advanced: Data Engineer Certification Exam

Pass Snowflake DEA-C01 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.geekcert.com/dea-c01.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Snowflake Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

A data engineer is using Amazon Athena to analyze sales data that is in Amazon S3. The data engineer writes a query to retrieve sales amounts for 2023 for several products from a table named sales_data. However, the query does not return results for all of the products that are in the sales_data table. The data engineer needs to troubleshoot the query to resolve the issue.

The data engineer\\'s original query is as follows:

SELECT product_name, sum(sales_amount)

FROM sales_data WHERE year = 2023

GROUP BY product_name

How should the data engineer modify the Athena query to meet these requirements?

A. Replace sum(sales amount) with count(*J for the aggregation.

B. Change WHERE year = 2023 to WHERE extractlyear FROM sales data) = 2023.

C. Add HAVING sumfsales amount) > 0 after the GROUP BY clause.

D. Remove the GROUP BY clause

Correct Answer: B

Explanation: The original query does not return results for all of the products because the year column in the sales_data table is not an integer, but a timestamp. Therefore, the WHERE clause does not filter the data correctly, and only returns

the products that have a null value for the year column. To fix this, the data engineer should use the extract function to extract the year from the timestamp and compare it with 2023. This way, the querywill return the correct results for all of

the products in the sales_data table. The other options are either incorrect or irrelevant, as they do not address the root cause of the issue. Replacing sum with count does not change the filtering condition, adding HAVING clause does not

affect the grouping logic, and removing the GROUP BY clause does not solve the problem of missing products. References:

Troubleshooting JSON queries - Amazon Athena (Section: JSON related errors) When I query a table in Amazon Athena, the TIMESTAMP result is empty (Section: Resolution)

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 7, page 197)

QUESTION 2

A company\\'s data engineer needs to optimize the performance of table SQL queries. The company stores data in an Amazon Redshift cluster. The data engineer cannot increase the size of the cluster because of budget constraints. The company stores the data in multiple tables and loads the data by using the EVEN distribution style. Some tables are hundreds of gigabytes in size. Other tables are less than 10 MB in size.

Which solution will meet these requirements?

A. Keep using the EVEN distribution style for all tables. Specify primary and foreign keys for all tables.

B. Use the ALL distribution style for large tables. Specify primary and foreign keys for all tables.

C. Use the ALL distribution style for rarely updated small tables. Specify primary and foreign keys for all tables.

D. Specify a combination of distribution, sort, and partition keys for all tables.

Correct Answer: C

Explanation: This solution meets the requirements of optimizing the performance of table SQL queries without increasing the size of the cluster. By using the ALL distribution style for rarely updated small tables, you can ensure that the entire table is copied to every node in the cluster, which eliminates the need for data redistribution during joins. This can improve query performance significantly, especially for frequently joined dimension tables. However, using the ALL distribution style also increases the storage space and the load time, so it is only suitable for small tables that are not updated frequently orextensively. By specifying primary and foreign keys for all tables, you can help the query optimizer to generate better query plans and avoid unnecessary scans or joins. You can also use the AUTO distribution style to let Amazon Redshift choose the optimal distribution style based on the table size and the query patterns. References: Choose the best distribution style Distribution styles Working with data distribution styles

QUESTION 3

A data engineer needs Amazon Athena queries to finish faster. The data engineer notices that all the files the Athena queries use are currently stored in uncompressed .csv format. The data engineer also notices that users perform most queries by selecting a specific column.

Which solution will MOST speed up the Athena query performance?

A. Change the data format from .csvto JSON format. Apply Snappy compression.

B. Compress the .csv files by using Snappy compression.

C. Change the data format from .csvto Apache Parquet. Apply Snappy compression.

D. Compress the .csv files by using gzjg compression.

Correct Answer: C

Explanation: Amazon Athena is a serverless interactive query service that allows you to analyze data in Amazon S3 using standard SQL. Athena supports 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, changing the data format from CSV to Parquet and applying Snappy compression will most speed up the Athena query performance. Parquet is a column- oriented format that allows Athena to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. Snappy is a compression algorithm that reduces the data size without compromising the query speed, as it is splittable and does not require decompression

before reading. This solution will also reduce the cost of Athena queries, as Athena charges based on the amount of data scanned from S3. The other options are not as effective as changing the data format to Parquet and applying Snappy compression. Changing the data format from CSV to JSON and applying Snappy compression will not improve the query performance significantly, as JSON is also a row- oriented format that does not support columnar access or encoding techniques. Compressing the CSV files by using Snappy compression will reduce the data size, but it will not improve the query performance significantly, as CSV is still a row-oriented format that does not support columnar access or encoding techniques. Compressing the CSV files by using gzjg compression will reduce the data size, but it willdegrade the query performance, as gzjg is not a splittable compression algorithm and requires decompression before reading. References: Amazon Athena Choosing the Right Data Format AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Analysis and Visualization, Section 5.1: Amazon Athena

QUESTION 4

A financial company wants to implement a data mesh. The data mesh must support centralized data governance, data analysis, and data access control. The company has decided to use AWS Glue for data catalogs and extract, transform, and load (ETL) operations.

Which combination of AWS services will implement a data mesh? (Choose two.)

A. Use Amazon Aurora for data storage. Use an Amazon Redshift provisioned cluster for data analysis.

B. Use Amazon S3 for data storage. Use Amazon Athena for data analysis.

C. Use AWS Glue DataBrewfor centralized data governance and access control.

D. Use Amazon RDS for data storage. Use Amazon EMR for data analysis.

E. Use AWS Lake Formation for centralized data governance and access control.

Correct Answer: BE

Explanation: A data mesh is an architectural framework that organizes data into domains and treats data as products that are owned and offered for consumption by different teams1. A data mesh requires a centralized layer for data governance and access control, as well as a distributed layer for data storage and analysis. AWS Glue can provide data catalogs and ETL operations for the data mesh, but it cannot provide data governance and access control by itself2. Therefore, the company needs to use another AWS service for this purpose. AWS Lake Formation is a service that allows you to create, secure, and manage data lakes on AWS3. It integrates with AWS Glue and other AWS services to provide centralized data governance and access control for the data mesh. Therefore, option E is correct. For data storage and analysis, the company can choose from different AWS services depending on their needs and preferences. However, one of the benefits of a data mesh is that it enables data to be stored and processed in a decoupled and scalable way1. Therefore, using serverless or managed services that can handle large volumes and varieties of data is preferable. Amazon S3 is a highly scalable, durable, and secure object storage service that can store any type of data. Amazon Athena is a serverless interactive query service that can analyze data in Amazon S3 using standard SQL. Therefore, option B is a good choice for data storage and analysis in a data mesh. Option A, C, and D are not optimal because they either use relational databases that are not suitable for storing diverse and unstructured data, or they require more management and provisioning than serverless services. References:

1: What is a Data Mesh? - Data Mesh Architecture Explained - AWS

2: AWS Glue - Developer Guide

3: AWS Lake Formation - Features [4]: Design a data mesh architecture using AWS Lake Formation and AWS Glue [5]: Amazon S3 - Features [6]: Amazon Athena - Features

QUESTION 5

A healthcare company uses Amazon Kinesis Data Streams to stream real-time health data from wearable devices, hospital equipment, and patient records.

A data engineer needs to find a solution to process the streaming data. The data engineer needs to store the data in an Amazon Redshift Serverless warehouse. The solution must support near real-time analytics of the streaming data and the previous day\\'s data.

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

A. Load data into Amazon Kinesis Data Firehose. Load the data into Amazon Redshift.

B. Use the streaming ingestion feature of Amazon Redshift.

C. Load the data into Amazon S3. Use the COPY command to load the data into Amazon Redshift.

D. Use the Amazon Aurora zero-ETL integration with Amazon Redshift.

Correct Answer: B

Explanation: The streaming ingestion feature of Amazon Redshift enables you to ingest data from streaming sources, such as Amazon Kinesis Data Streams, into Amazon Redshift tables in near real-time. You can use the streaming ingestion feature to process the streaming data from the wearable devices, hospital equipment, and patient records. The streaming ingestion feature also supports incremental updates, which means you can append new data or update existing data in the Amazon Redshift tables. This way, you can store the data in an Amazon Redshift Serverless warehouse and support near real-time analytics of the streaming data and the previous day\\'s data. This solution meets the requirements with the least operational overhead, as it does not require any additional services or components to ingest and process the streaming data. The other options are either not feasible or not optimal. Loading data into Amazon Kinesis Data Firehose and then into Amazon Redshift (option A) would introduce additional latency and cost, as well as require additional configuration and management. Loading data into Amazon S3 and then using the COPY command to load the data into Amazon Redshift (option C) would also introduce additional latency and cost, as well as require additional storage space and ETL logic. Using the Amazon Aurora zero-ETL integration with Amazon Redshift (option D) would not work, as it requires the data to be stored in Amazon Aurora first, which is not the case for the streaming data from the healthcare company. References: Using streaming ingestion with Amazon Redshift AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.5: Amazon Redshift Streaming Ingestion