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.certbus.com/dea-c01.html

100% Passing Guarantee
100% Money Back Assurance

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

QUESTION 1

A company uses Amazon Athena to run SQL queries for extract, transform, and load (ETL) tasks by using Create Table As Select (CTAS). The company must use Apache Spark instead of SQL to generate analytics.

Which solution will give the company the ability to use Spark to access Athena?

A. Athena query settings

B. Athena workgroup

C. Athena data source

D. Athena query editor

Correct Answer: C

Explanation: Athena data source is a solution that allows you to use Spark to access Athena by using the Athena JDBC driver and the Spark SQL interface. You can use the Athena data source to create Spark DataFrames from Athena tables, run SQL queries on the DataFrames, and write the results back to Athena. The Athena data source supports various data formats, such as CSV, JSON, ORC, and Parquet, and also supports partitioned and bucketed tables. The Athena data source is a cost-effective and scalable way to use Spark to access Athena, as it does not require any additional infrastructure or services, and you only pay for the data scanned by Athena. The other options are not solutions that give the company the ability to use Spark to access Athena. Option A, Athena query settings, is a feature that allows you to configure various parameters for your Athena queries, such as the output location, the encryption settings, the query timeout, and the workgroup. Option B, Athena workgroup, is a feature that allows you to isolate and manage your Athena queries and resources, such as the query history, the query notifications, the query concurrency, and the query cost. Option D, Athena query editor, is a feature that allows you to write and run SQL queries on Athena using the web console or the API. None of these options enable you to use Spark instead of SQL to generate analytics on Athena. References: Using Apache Spark in Amazon Athena Athena JDBC Driver Spark SQL Athena query settings [Athena workgroups] [Athena query editor]

QUESTION 2

A company receives a daily file that contains customer data in .xls format. The company stores the file in Amazon S3. The daily file is approximately 2 GB in size.

A data engineer concatenates the column in the file that contains customer first names and the column that contains customer last names. The data engineer needs to determine the number of distinct customers in the file.

Which solution will meet this requirement with the LEAST operational effort?

A. Create and run an Apache Spark job in an AWS Glue notebook. Configure the job to read the S3 file and calculate the number of distinct customers.

B. Create an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file. Run SQL queries from Amazon Athena to calculate the number of distinct customers.

C. Create and run an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.

D. Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers.

Correct Answer: D

Explanation: AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. You can use DataBrew to create recipes that define the steps to apply to your data, such as filtering, renaming, splitting, or aggregating columns. You can also use DataBrew to run jobs that execute the recipes on your data sources, such as Amazon S3, Amazon Redshift, or Amazon Aurora. DataBrew integrates with AWS Glue Data Catalog, which is a centralized metadata repository for your data assets1. The solution that meets the requirement with the least operational effort is to use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers. This solution has the following advantages: It does not require you to write any code, as DataBrew provides a graphical user interface that lets you explore, transform, and visualize your data. You can use DataBrewto concatenate the columns that contain customer first names and last names, and then use the COUNT_DISTINCT aggregate function to count the number of unique values in the resulting column2. It does not require you to provision, manage, or scale any servers, clusters, or notebooks, as DataBrew is a fully managed service that handles all the infrastructure for you. DataBrew can automatically scale up or down the compute resources based on the size and complexity of your data and recipes1. It does not require you to create or update any AWS Glue Data Catalog entries, as DataBrew can automatically create and register the data sources and targets in the Data Catalog. DataBrew can also use the existing Data Catalog entries to access the data in S3 or other sources3. Option A is incorrect because it suggests creating and running an Apache Spark job in an AWS Glue notebook. This solution has the following disadvantages: It requires you to write code, as AWS Glue notebooks are interactive development environments that allow you to write, test, and debug Apache Spark code using Python or Scala. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers. It requires you to provision and manage a development endpoint, which is a serverless Apache Spark environment that you can connect to your notebook. You need to specify the type and number of workers for your development endpoint, and monitor its status and metrics. It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code. Option B is incorrect because it suggests creating an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file, and running SQL queries from Amazon Athena to calculate the number of distinct customers. This solution has the following disadvantages: It requires you to create and run a crawler, which is a program that connects to your data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in the Data Catalog. You need to specify the data store, the IAM role, the schedule, and the output database for your crawler. It requires you to write SQL queries, as Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. You need to use Athena to concatenate the columns that contain customer first names and last names, and then use the COUNT(DISTINCT) aggregate function to count the number of unique values in the resulting column. Option C is incorrect because it suggests creating and running an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers. This solution has the following disadvantages: It requires you to write code, as Amazon EMR Serverless is a service that allows you to run Apache Spark jobs on AWS without provisioning or managing any infrastructure. You need to use the Spark SQL or the Spark DataFrame API to

read the S3 file and calculate the number of distinct customers. It requires you to create and manage an Amazon EMR Serverless cluster, which is a fully managed and scalable Spark environment that runs on AWS Fargate. You need to

specify the cluster name, the IAM role, the VPC, and the subnet for your cluster, and monitor its status and metrics.

It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in

your code.

References:

1: AWS Glue DataBrew - Features

2: Working with recipes - AWS Glue DataBrew

3: Working with data sources and data targets - AWS Glue DataBrew [4]: AWS Glue notebooks - AWS Glue [5]: Development endpoints - AWS Glue [6]: Populating the AWS Glue Data Catalog - AWS Glue [7]: Crawlers - AWS Glue

[8]: Amazon Athena - Features [9]: Amazon EMR Serverless - Features [10]: Creating an Amazon EMR Serverless cluster - Amazon EMR [11]: Using the AWS Glue Data Catalog with Amazon EMR Serverless - Amazon EMR

QUESTION 3

A data engineer must manage the ingestion of real-time streaming data into AWS. The data engineer wants to perform real-time analytics on the incoming streaming data by using time-based aggregations over a window of up to 30 minutes. The data engineer needs a solution that is highly fault tolerant.

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

A. Use an AWS Lambda function that includes both the business and the analytics logic to perform time-based aggregations over a window of up to 30 minutes for the data in Amazon Kinesis Data Streams.

B. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data that might occasionally contain duplicates by using multiple types of aggregations.

C. Use an AWS Lambda function that includes both the business and the analytics logic to perform aggregations for a tumbling window of up to 30 minutes, based on the event timestamp.

D. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data by using multiple types of aggregations to perform time- based analytics over a window of up to 30 minutes.

Correct Answer: A

Explanation: This solution meets the requirements of managing the ingestion of real-time streaming data into AWS and performing real-time analytics on the incoming streaming data with the least operational overhead. Amazon Managed Service for Apache Flink is a fully managed service that allows you to run Apache Flink applications without having to manage any infrastructure or clusters. Apache Flink is a framework for stateful stream processing that supports various types of aggregations, such as tumbling, sliding, and session windows, over streaming data. By using Amazon Managed Service for Apache Flink, you can easily connect to Amazon Kinesis Data Streams as the source and sink of your streaming data, and perform time-based analytics over a window of up to 30 minutes. This solution is also highly fault tolerant, as Amazon Managed Service for Apache Flink automatically scales, monitors, and restarts your Flink applications in case of failures. References: Amazon Managed Service for Apache Flink Apache Flink Window Aggregations in Flink

QUESTION 4

A security company stores IoT data that is in JSON format in an Amazon S3 bucket. The data structure can change when the company upgrades the IoT devices. The company wants to create a data catalog that includes the IoT data. The company\\'s analytics department will use the data catalog to index the data.

Which solution will meet these requirements MOST cost-effectively?

A. Create an AWS Glue Data Catalog. Configure an AWS Glue Schema Registry. Create a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

B. Create an Amazon Redshift provisioned cluster. Create an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3. Create Redshift stored procedures to load the data into Amazon Redshift.

C. Create an Amazon Athena workgroup. Explore the data that is in Amazon S3 by using Apache Spark through Athena. Provide the Athena workgroup schema and tables to the analytics department.

D. Create an AWS Glue Data Catalog. Configure an AWS Glue Schema Registry. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API. Create an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

Correct Answer: C

Explanation: The best solution to meet the requirements of creating a data catalog that includes the IoT data, and allowing the analytics department to index the data, most cost- effectively, is to create an Amazon Athena workgroup, explore the data that is in Amazon S3 by using Apache Spark through Athena, and provide the Athena workgroup schema and tables to the analytics department. Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python1. Amazon Athena also supports Apache Spark, an open-source distributed processing framework that can run large-scale data analytics applications across clusters of servers2. You can use Athena to run Spark code on data in Amazon S3 without having to set up, manage, or scale any infrastructure. You can also use Athena to create and manage external tables that point to your data in Amazon S3, and store them in an external data catalog, such as AWS Glue Data Catalog, Amazon Athena Data Catalog, or your own Apache Hive metastore3. You can create Athena workgroups to separate query execution and resource allocation based on different criteria, such as users, teams, or applications4. You can share the schemas and tables in your Athena workgroup with other users or applications, such as Amazon QuickSight, for data visualization and analysis5. Using Athena and Spark to create a data catalog and explore the IoT data in Amazon S3 is the most cost-effective solution, as you pay only for the queries you run or the compute you use, and you pay nothing when the service is idle1. You also save on the operational overhead and complexity of managing data warehouse infrastructure, as Athena and Spark are serverless and scalable. You can also benefit from the flexibility and performance of Athena and Spark, as they support various data formats, including JSON, and can handle schema changes and complex queries efficiently. Option A is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating a new AWS Glue workload to orchestrate theingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Glue Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components6. AWS Glue Schema Registry is a service that allows you to centrally store and manage the schemas of your streaming data in AWS Glue Data Catalog7. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores8. Amazon Redshift Serverless is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to run and scale analytics without having to manage data warehouse infrastructure9. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Glue charges you based on the compute time and the data processed by your ETL jobs8. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue and Amazon Redshift Serverless would introduce additional latency and complexity, as you would have to ingest the data from Amazon S3 to Amazon Redshift Serverless, and then query it from there, instead of querying it directly from Amazon S3 using Athena and Spark. Option B is not the best solution, as creating an Amazon Redshift provisioned cluster, creating an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3, and creating Redshift stored procedures to load the data into Amazon Redshift, would incur more costs and complexity than using Athena and Spark. Amazon Redshift provisioned clusters are clusters that you create and manage by specifying the number and type of nodes, and the amount of storage and compute capacity10. Amazon Redshift Spectrum is a feature of Amazon Redshift that allows you to query and join data across your data warehouse and your data lake using standard SQL11. Redshift stored procedures are SQL statements that you can define and store in Amazon Redshift, and then call them by using the CALL command12. While these features are powerful and useful for many data warehousing scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. Amazon Redshift provisioned clusters charge you based on the node type, the number of nodes, and the duration of the cluster10. Amazon Redshift Spectrum charges you based on the amount of data scanned by your queries11. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using Amazon Redshift provisioned clusters and Spectrum would introduce additional latency and complexity, as you would have to provision andmanage the cluster, create an external schema and database for the data in Amazon S3, and load the data into the cluster using stored procedures, instead of querying it directly from Amazon S3 using Athena and Spark. Option D is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating AWS Lambda user defined functions (UDFs) by using the Amazon

Redshift Data API, and creating an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers13. AWS Lambda UDFs are Lambda functions that you can invoke from within an Amazon Redshift query. Amazon Redshift Data API is a service that allows you to run SQL statements on Amazon Redshift clusters using HTTP requests, without needing a persistent connection. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost- effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Lambda charges you based on the number of requests and the duration of your functions13. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. AWS Step Functions charges you based on the number of state transitions in your workflows. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue, AWS Lambda, Amazon Redshift Data API, and AWS Step Functions would introduce additional latency and complexity, as you would have to create and invoke Lambda functions to ingest the data from Amazon S3 to Amazon Redshift Serverless using the Data API, and coordinate the ingestion process using Step Functions, instead of querying it directly from Amazon S3 using Athena and Spark. References: What is Amazon Athena? Apache Spark on Amazon Athena Creating tables, updating the schema, and adding new partitions in the Data Catalog from AWS Glue ETL jobs Managing Athena workgroups Using Amazon QuickSight to visualize data in Amazon Athena AWS Glue Data Catalog AWS Glue Schema Registry What is AWS Glue? Amazon Redshift Serverless Amazon Redshift provisioned clusters Querying external data using Amazon Redshift Spectrum Using stored procedures in Amazon Redshift What is AWS Lambda? [Creating and using AWS Lambda UDFs] [Using the Amazon Redshift Data API] [What is AWS Step Functions?] AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

---

QUESTION 5

A company maintains an Amazon Redshift provisioned cluster that the company uses for extract, transform, and load (ETL) operations to support critical analysis tasks. A sales team within the company maintains a Redshift cluster that the sales team uses for business intelligence (BI) tasks.

The sales team recently requested access to the data that is in the ETL Redshift cluster so the team can perform weekly summary analysis tasks. The sales team needs to join data from the ETL cluster with data that is in the sales team\\'s BI cluster.

The company needs a solution that will share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution must minimize usage of the computing resources of the ETL cluster.

Which solution will meet these requirements?

A. Set up the sales team Bl cluster asa consumer of the ETL cluster by using Redshift data sharing.

B. Create materialized views based on the sales team\\'s requirements. Grant the sales team direct access to the ETL cluster.

C. Create database views based on the sales team\\'s requirements. Grant the sales team direct access to the ETL cluster.

D. Unload a copy of the data from the ETL cluster to an Amazon S3 bucket every week. Create an Amazon Redshift Spectrum table based on the content of the ETL cluster.

Correct Answer: A

Explanation: Redshift data sharing is a feature that enables you to share live data across different Redshift clusters without the need to copy or move data. Data sharing provides secure and governed access to data, while preserving the performance and concurrency benefits of Redshift. By setting up the sales team BI cluster as a consumer of the ETL cluster, the company can share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution also minimizes the usage of the computing resources of the ETL cluster, as the data sharing does not consume any storage space or compute resources from the producer cluster. The other options are either not feasible or not efficient. Creating materialized views or database views would require the sales team to have direct access to the ETL cluster, which could interfere with the critical analysis tasks. Unloading a copy of the data from the ETL cluster to anAmazon S3 bucket every week would introduce additional latency and cost, as well as create data inconsistency issues. References: Sharing data across Amazon Redshift clusters AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.2: Amazon Redshift

QUESTION 6

A data engineer uses Amazon Redshift to run resource-intensive analytics processes once every month. Every month, the data engineer creates a new Redshift provisioned cluster. The data engineer deletes the Redshift provisioned cluster after the analytics processes are complete every month. Before the data engineer deletes the cluster each month, the data engineer unloads backup data from the cluster to an Amazon S3 bucket.

The data engineer needs a solution to run the monthly analytics processes that does not require the data engineer to manage the infrastructure manually.

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

A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month.

B. Use Amazon Redshift Serverless to automatically process the analytics workload.

C. Use the AWS CLI to automatically process the analytics workload.

D. Use AWS CloudFormation templates to automatically process the analytics workload.

Correct Answer: B

Explanation: Amazon Redshift Serverless is a new feature of Amazon Redshift that enables you to run SQL queries on data in Amazon S3 without provisioning or managing any clusters. You can use Amazon Redshift Serverless to

automatically process the analytics workload, as it scales up and down the compute resources based on the query demand, and charges you only for the resources consumed. This solution will meet the requirements with the least operational

overhead, as it does not require the data engineer to create, delete, pause, or resume any Redshift clusters, or to manage any infrastructure manually. You can use the Amazon Redshift Data API to run queries from the AWS CLI, AWS SDK,

or AWS Lambda functions12.

The other options are not optimal for the following reasons:

A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month. This option is not recommended, as it would still require the data engineer to create and delete a new Redshift provisioned cluster every month, which can incur additional costs and time. Moreover, this option would require the data engineer to use Amazon Step Functions to orchestrate the workflow of pausing and resuming the cluster, which can add complexity and overhead. C. Use the AWS CLI to automatically process the analytics workload. This option is vague and does not specify how the AWS CLI is used to process the analytics workload. The AWS CLI can be used to run queries on data in Amazon S3 using Amazon Redshift Serverless, Amazon Athena, or Amazon EMR, but each of these services has different features and benefits. Moreover, this option does not address the requirement of not managing the infrastructure manually, as the data engineer may still need to provision and configure some resources, such as Amazon EMR clusters or Amazon Athena workgroups. D. Use AWS CloudFormation templates to automatically process the analytics workload. This option is also vague and does not specify how AWS CloudFormation templates are used to process the analytics workload. AWS CloudFormation is a service that lets you model and provision AWS resources using templates. You can use AWS CloudFormation templates to create and delete a Redshift provisioned cluster every month, or to create and configure other AWS resources, such as Amazon EMR, Amazon Athena, or Amazon Redshift Serverless. However, this option does not address the requirement of not managing the infrastructure manually, as the data engineer may still need to write and maintain the AWS CloudFormation templates, and to monitor the status and performance of the resources. References:

1: Amazon Redshift Serverless

2: Amazon Redshift Data API : Amazon Step Functions : AWS CLI : AWS CloudFormation

---

QUESTION 7

A company needs to partition the Amazon S3 storage that the company uses for a data lake. The partitioning will use a path of the S3 object keys in the following format: s3://bucket/prefix/year=2023/month=01/day=01.

A data engineer must ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket.

Which solution will meet these requirements with the LEAST latency?

A. Schedule an AWS Glue crawler to run every morning.

B. Manually run the AWS Glue CreatePartition API twice each day.

C. Use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call.

D. Run the MSCK REPAIR TABLE command from the AWS Glue console.

Correct Answer: C

Explanation: The best solution to ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket with the least latency is to use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call. This way, the Data Catalog is updated as soon as new data is written to S3, and the partition information is immediately available for querying by other services. The Boto3 AWS Glue create partition API call allows you to create a new partition in the Data Catalog by specifying the table name, the database name, and the partition values1. You can use this API call in your code that writes data to S3, such as a Python script or an AWS Glue ETL job, to create a partition for each new S3 object key that matches the partitioning scheme. Option A is not the best solution, as scheduling an AWS Glue crawler to run every morning would introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog2. Crawlers can be scheduled to run periodically, such as daily or hourly, but they cannot runcontinuously or in real-time. Therefore, using a crawler to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency. Option B is not the best solution, as manually running the AWS Glue CreatePartition API twice each day would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. Moreover, manually running the API would require more operational overhead and human intervention than using code that writes data to S3 to invoke the API automatically. Option D is not the best solution, as running the MSCK REPAIR TABLE command from the AWS Glue console would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. The MSCK REPAIR TABLE command is a SQL command that you can run in the AWS Glue console to add partitions to the Data Catalog based on the S3 object keys that match the partitioning scheme3. However, this command is not meant to be run frequently or in real-time, as it can take a long time to scan the entire S3 bucket and add the partitions. Therefore, using this command to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency. References: AWS Glue CreatePartition API Populating the AWS Glue Data Catalog MSCK REPAIR TABLE Command AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

QUESTION 8

A company stores datasets in JSON format and .csv format in an Amazon S3 bucket. The company has Amazon RDS for Microsoft SQL Server databases, Amazon DynamoDB tables that are in provisionedcapacity mode, and an Amazon Redshift cluster. A data engineering team must develop a solution that will give data scientists the ability to query all data sources by using syntax similar to SQL.

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

A. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use Amazon Athena to query the data. Use SQL for structured data sources. Use PartiQL for data that is stored in JSON format.

B. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use Redshift Spectrum to query the data. Use SQL for structured data sources. Use PartiQL for data that is stored in JSON format.

C. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format. Store the transformed data in an S3 bucket. Use Amazon Athena to query the original and transformed data from the S3 bucket.

D. Use AWS Lake Formation to create a data lake. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet format. Store the transformed data in an S3 bucket. Use Amazon Athena or Redshift Spectrum to query the data.

Correct Answer: A

Explanation: The best solution to meet the requirements of giving data scientists the ability to query all data sources by using syntax similar to SQL with the least operational overhead is to use AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use Amazon Athena to query the data, use SQL for structured data sources, and use PartiQL for data that is stored in JSON format. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores1. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog2. The Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components3. You can use AWS Glue to crawl the data sources, such as Amazon S3, Amazon RDS for Microsoft SQL Server, and Amazon DynamoDB, and store the metadata in the Data Catalog. Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python4. Amazon Athena also supports PartiQL, a SQL-compatible query language that lets you query, insert, update, and delete data from semi-structured and nested data, such as JSON. You can use Amazon Athena to query the data from the Data Catalog using SQL for structured data sources, such as .csv files and relational databases, and PartiQL for data that is stored in JSON format. You can also use Athena to query data from other data sources, such as Amazon Redshift, using federated queries. Using AWS Glue and Amazon Athena to query all data sources by using syntax similar to SQL is the least operational overhead solution, as you do not need to provision, manage, or scale any infrastructure, and you pay only for the resources you use. AWS Glue charges you based on the compute time and the data processed by your crawlers and ETL jobs1. Amazon Athena charges you based on the amount of data scanned by your queries. You can also reduce the cost and improve the performance of your queries by using compression, partitioning, and columnar formats for your data in Amazon S3. Option B is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, and use Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena. Redshift Spectrum is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to query and join data across your data warehouse and your data lake using standard SQL. While Redshift Spectrum is powerful and useful for many data warehousing scenarios, it is not necessary or cost-effective for querying all data sources by using syntax similar to SQL. Redshift Spectrum charges you based on the amount of data scanned by your queries, which is similar to Amazon Athena, but it also requires you to have an Amazon Redshift cluster, which charges you based on the node type, the number of nodes, and the duration of the cluster5. These costs can add up quickly, especially if you have large volumes of data and complex queries. Moreover, using Redshift Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, and create an external schema and database for the data in the Data Catalog, instead of querying it directly from Amazon Athena. Option C is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format, store the

transformed data in an S3 bucket, and use Amazon Athena to query the original and transformed data from the S3 bucket, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Glue jobs are ETL scripts that

you can write in Python or Scala to transform your data and load it to your target data store. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be

scanned and providing efficient compression and encoding schemes6. While using AWS Glue jobs and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational

overhead of the data pipeline, as you would have to write, run, and monitor the ETL jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using AWS Glue jobs and Parquet would introduce additional latency,

as you would have to wait for the ETL jobs to finish before querying the transformed data.

Option D is not the best solution, as using AWS Lake Formation to create a data lake, use Lake Formation jobs to transform the data from all data sources to Apache Parquet format, store the transformed data in an S3 bucket, and use

Amazon Athena or RedshiftSpectrum to query the data, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Lake Formation is a service that helps you centrally govern, secure, and globally share data for

analytics and machine learning7. Lake Formation jobs are ETL jobs that you can create and run using the Lake Formation console or API. While using Lake Formation and Parquet can improve the performance and reduce the cost of your

queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to create, run, and monitor the Lake Formation jobs, and store the transformed data in a separate location in Amazon S3.

Moreover, using Lake Formation and Parquet would introduce additional latency, as you would have to wait for the Lake Formation jobs to finish before querying the transformed data. Furthermore, using Redshift Spectrum to query the data

would also incur the same costs and complexity as mentioned in option B. References:

What is Amazon Athena?

Data Catalog and crawlers in AWS Glue

AWS Glue Data Catalog

Columnar Storage Formats

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide AWS Glue Schema Registry

What is AWS Glue?

Amazon Redshift Serverless

Amazon Redshift provisioned clusters

[Querying external data using Amazon Redshift Spectrum] [Using stored procedures in Amazon Redshift]

[What is AWS Lambda?]

[PartiQL for Amazon Athena]

[Federated queries in Amazon Athena]

[Amazon Athena pricing]

[Top 10 performance tuning tips for Amazon Athena] [AWS Glue ETL jobs]

[AWS Lake Formation jobs]

QUESTION 9

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 10

A company has used an Amazon Redshift table that is named Orders for 6 months. The company performs weekly updates and deletes on the table. The table has an interleaved sort key on a column that contains AWS Regions.

The company wants to reclaim disk space so that the company will not run out of storage space. The company also wants to analyze the sort key column.

Which Amazon Redshift command will meet these requirements?

A. VACUUM FULL Orders

B. VACUUM DELETE ONLY Orders

C. VACUUM REINDEX Orders

D. VACUUM SORT ONLY Orders

Correct Answer: C

Amazon Redshift is a fully managed, petabyte-scale data warehouse service that enables fast and cost-effective analysis of large volumes of data. Amazon Redshift uses columnar storage, compression, and zone maps to optimize the storage and performance of data. However, over time, as data is inserted, updated, or deleted, the physical storage of data can become fragmented, resulting in wasted disk space and degraded query performance. To address this issue, Amazon Redshift provides the VACUUM command, which reclaims disk space and resorts rows in either a specified table or all tables in the current schema1. The VACUUM command has four options: FULL, DELETE ONLY, SORT ONLY, and REINDEX. The option that best meets the requirements of the question is VACUUM REINDEX, which re-sorts the rows in a table that has an interleaved sort key and rewritesthe table to a new location on disk. An interleaved sort key is a type of sort key that gives equal weight to each column in the sort key, and stores the rows in a way that optimizes the performance of queries that filter by multiple columns in the sort key. However, as data is added or changed, the interleaved sort order can become skewed, resulting in suboptimal query performance. The VACUUM REINDEX option restores the optimal interleaved sort order and reclaims disk space by removing deleted rows. This option also analyzes the sort key column and updates the table statistics, which are used by the query optimizer to generate the most efficient query execution plan23. The other options are not optimal for the following reasons:

A. VACUUM FULL Orders. This option reclaims disk space by removing deleted rows and resorts the entire table. However, this option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved

sort order. Moreover, this option is the most resource-intensive and time-consuming, as it rewrites the entire table to a new location on disk. B. VACUUM DELETE ONLY Orders. This option reclaims disk space by removing deleted rows, but

does not resort the table. This option is not suitable for tables that have any sort key, as it does not improve the query performance by restoring the sort order. Moreover, this option does not analyze the sort key column and update the table

statistics.

D. VACUUM SORT ONLY Orders. This option resorts the entire table, but does not reclaim disk space by removing deleted rows. This option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal

interleaved sort order. Moreover, this option does not analyze the sort key column and update the table statistics.

References:

1: Amazon Redshift VACUUM

2: Amazon Redshift Interleaved Sorting

3: Amazon Redshift ANALYZE

QUESTION 11

A company uses Amazon Redshift for its data warehouse. The company must automate refresh schedules for Amazon Redshift materialized views.

Which solution will meet this requirement with the LEAST effort?

A. Use Apache Airflow to refresh the materialized views.

B. Use an AWS Lambda user-defined function (UDF) within Amazon Redshift to refresh the materialized views.

C. Use the query editor v2 in Amazon Redshift to refresh the materialized views.

D. Use an AWS Glue workflow to refresh the materialized views.

Correct Answer: C

Explanation: The query editor v2 in Amazon Redshift is a web-based tool that allows users to run SQL queries and scripts on Amazon Redshift clusters. The query editor v2 supports creating and managing materialized views, which are precomputed results of a query that can improve the performance of subsequent queries. The query editor v2 also supports scheduling queries to run at specified intervals, which can be used to refresh materialized views automatically. This solution requires the least effort, as it does not involve any additional services, coding, or configuration. The other solutions are more complex and require more operational overhead. Apache Airflow is an open-source platform for orchestrating workflows, which can be used to refresh materialized views, but it requires setting up and managing an Airflow environment, creating DAGs (directed acyclic graphs) to define the workflows, and integrating with Amazon Redshift. AWS Lambda is a serverless compute service that can run code in response to events, which can be used to refresh materialized views, but it requires creating and deploying Lambda functions, defining UDFs within Amazon Redshift, and triggering the functions using events or schedules. AWS Glue is a fully managed ETL service that can run jobs to transform and load data, which can be used to refresh materialized views, but it requires creating and configuring Glue jobs, defining Glue workflows to orchestrate the jobs, and scheduling the workflows using triggers. References: Query editor V2 Working with materialized views Scheduling queries [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

QUESTION 12

A data engineer must orchestrate a data pipeline that consists of one AWS Lambda function and one AWS Glue job. The solution must integrate with AWS services.

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

A. Use an AWS Step Functions workflow that includes a state machine. Configure the state machine to run the Lambda function and then the AWS Glue job.

B. Use an Apache Airflow workflow that is deployed on an Amazon EC2 instance. Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

C. Use an AWS Glue workflow to run the Lambda function and then the AWS Glue job.

D. Use an Apache Airflow workflow that is deployed on Amazon Elastic Kubernetes Service (Amazon EKS). Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

Correct Answer: A

Explanation: AWS Step Functions is a service that allows you to coordinate multiple AWS services into serverless workflows. You can use Step Functions to create state machines that define the sequence and logic of the tasks in your workflow. Step Functions supports various types of tasks, such as Lambda functions, AWS Glue jobs, Amazon EMR clusters, Amazon ECS tasks, etc. You can use Step Functions to monitor and troubleshoot your workflows, as well as to handle errors and retries. Using an AWS Step Functions workflow that includes a state machine to run the Lambda function and then the AWS Glue job will meet the requirements with the least management overhead, as it leverages the serverless and managed capabilities of Step Functions. You do not need to write any code to orchestrate the tasks in your workflow, as you can use the Step Functions console or the AWS Serverless Application Model (AWS SAM) to define and deploy your state machine. You also do not need to provision or manage any servers or clusters, as Step Functions scales automatically based on the demand. The other options are not as efficient as using an AWS Step Functions workflow. Using an Apache Airflow workflow that is deployed on an Amazon EC2 instance or on Amazon Elastic Kubernetes Service (Amazon EKS) will require more management overhead, as you will need to provision, configure, and maintain the EC2 instance or the EKS cluster, as well as the Airflow components. You will also need to write and maintain the Airflow DAGs to orchestrate the tasks in your workflow. Using an AWS Glue workflow to run the Lambda function and then the AWS Glue job will not work, as AWS Glue workflows only support AWS Glue jobs and crawlers as tasks, not Lambda functions. References: AWS Step Functions AWS Glue AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.3: AWS Step Functions

QUESTION 13

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 14

A company stores daily records of the financial performance of investment portfolios in .csv format in an Amazon S3 bucket. A data engineer uses AWS Glue crawlers to crawl the S3 data.

The data engineer must make the S3 data accessible daily in the AWS Glue Data Catalog.

Which solution will meet these requirements?

A. Create an IAM role that includes the AmazonS3FullAccess policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler\\'s data store. Create a daily schedule to run the crawler. Configure the output destination to a new path in the existing S3 bucket.

B. Create an IAM role that includes the AWSGlueServiceRole policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler\\'s data store. Create a daily schedule to run the crawler. Specify a database name for the output.

C. Create an IAM role that includes the AmazonS3FullAccess policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler\\'s data store. Allocate data processing units (DPUs) to run the crawler every day. Specify a database name for the output.

D. Create an IAM role that includes the AWSGlueServiceRole policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler\\'s data store. Allocate data processing units (DPUs) to run the crawler every day. Configure the output destination to a new path in the existing S3 bucket.

Correct Answer: B

Explanation: To make the S3 data accessible daily in the AWS Glue Data Catalog, the data engineer needs to create a crawler that can crawl the S3 data and write the metadata to the Data Catalog. The crawler also needs to run on a daily

schedule to keep the Data Catalog updated with the latest data. Therefore, the solution must include the following steps:

Create an IAM role that has the necessary permissions to access the S3 data and the Data Catalog. The AWSGlueServiceRole policy is a managed policy that grants these permissions1.

Associate the role with the crawler.

Specify the S3 bucket path of the source data as the crawler\\'s data store. The crawler will scan the data and infer the schema and format2. Create a daily schedule to run the crawler. The crawler will run at the specified time every day and

update the Data Catalog with any changes in the data3. Specify a database name for the output. The crawler will create or update a table in the Data Catalog under the specified database. The table will contain the metadata about the data in

the S3 bucket, such as the location, schema, and classification.

Option B is the only solution that includes all these steps. Therefore, option B is the correct answer.

Option A is incorrect because it configures the output destination to a new path in the existing S3 bucket. This is unnecessary and may cause confusion, as the crawler does not write any data to the S3 bucket, only metadata to the Data

Catalog. Option C is incorrect because it allocates data processing units (DPUs) to run the crawler every day. This is also unnecessary, as DPUs are only used for AWS Glue ETL jobs, not crawlers.

Option D is incorrect because it combines the errors of option A and C. It configures the output destination to a new path in the existing S3 bucket and allocates DPUs to run the crawler every day, both of which are irrelevant for the crawler.

References:

1: AWS managed (predefined) policies for AWS Glue - AWS Glue

2: Data Catalog and crawlers in AWS Glue - AWS Glue

3: Scheduling an AWS Glue crawler - AWS Glue

[4]: Parameters set on Data Catalog tables by crawler - AWS Glue [5]: AWS Glue pricing - Amazon Web Services (AWS)

QUESTION 15

A manufacturing company collects sensor data from its factory floor to monitor and enhance operational efficiency. The company uses Amazon Kinesis Data Streams to publish the data that the sensors collect to a data stream. Then Amazon Kinesis Data Firehose writes the data to an Amazon S3 bucket.

The company needs to display a real-time view of operational efficiency on a large screen in the manufacturing facility.

Which solution will meet these requirements with the LOWEST latency?

A. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Use a connector for Apache Flink to write data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard.

B. Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is created. Use the Lambda function to publish the data to Amazon Aurora. Use Aurora as a source to create an Amazon QuickSight dashboard.

C. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Create a new Data Firehose delivery stream to publish data directly to an Amazon Timestream database. Use the Timestream database as a source to create an Amazon QuickSight dashboard.

D. Use AWS Glue bookmarks to read sensor data from the S3 bucket in real time. Publish the data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard.

Correct Answer: C

Explanation: This solution will meet the requirements with the lowest latency because it uses Amazon Managed Service for Apache Flink to process the sensor data in real time and write it to Amazon Timestream, a fast, scalable, and

serverless time series database. Amazon Timestream is optimized for storing and analyzing time series data, such as sensor data, and can handle trillions of events per day with millisecond latency. By using AmazonTimestream as a source,

you can create an Amazon QuickSight dashboard that displays a real-time view of operational efficiency on a large screen in the manufacturing facility. Amazon QuickSight is a fully managed business intelligence service that can connect to

various data sources, including Amazon Timestream, and provide interactive visualizations and insights123.

The other options are not optimal for the following reasons:

A. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Use a connector for Apache Flink to write data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is similar to option C, but it uses Grafana instead of Amazon QuickSight to create the dashboard. Grafana is an open source visualization tool that can also connect to Amazon Timestream, but it requires additional steps to set up and configure, such as deploying a Grafana server on Amazon EC2, installing the Amazon Timestream plugin, and creating an IAM role for Grafana to access Timestream. These steps can increase the latency and complexity of the solution. B. Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is created. Use the Lambda function to publish the data to Amazon Aurora. Use Aurora as a source to create an Amazon QuickSight dashboard. This option is not suitable for displaying a real-time view of operational efficiency, as it introduces unnecessary delays and costs in the data pipeline. First, the sensor data is written to an S3 bucket by Amazon Kinesis Data Firehose, which can have a buffering interval of up to 900 seconds. Then, the S3 bucket sends a notification to a Lambda function, which can incur additional invocation and execution time. Finally, the Lambda function publishes the data to Amazon Aurora, a relational database that is not optimized for time series data and can have higher storage and performance costs than Amazon Timestream . D. Use AWS Glue bookmarks to read sensor data from the S3 bucket in real time. Publish the data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is also not suitable for displaying a real-time view of operational efficiency, as it uses AWS Glue bookmarks to read sensor data from the S3 bucket. AWS Glue bookmarks are a feature that helps AWS Glue jobs and crawlers keep track of the data that has already been processed, so that they can resume from where they left off. However, AWS Glue jobs and crawlers are not designed for real-time data processing, as they can have a minimum frequency of 5 minutes and a variable start-up time. Moreover, this option also uses Grafana instead of Amazon QuickSight to create the dashboard, which can increase the latency and complexity of the solution . References:

1: Amazon Managed Streaming for Apache Flink

2: Amazon Timestream

3: Amazon QuickSight : Analyze data in Amazon Timestream using Grafana : Amazon Kinesis Data Firehose : Amazon Aurora : AWS Glue Bookmarks : AWS Glue Job and Crawler Scheduling