

AWS-Certified-Data-Engineer-Associate Dumps

AWS Certified Data Engineer - Associate (DEA-C01)

<https://www.certleader.com/AWS-Certified-Data-Engineer-Associate-dumps.html>



NEW QUESTION 1

A company uses Amazon Athena for one-time queries against data that is in Amazon S3. The company has several use cases. The company must implement permission controls to separate query processes and access to query history among users, teams, and applications that are in the same AWS account. Which solution will meet these requirements?

- A. Create an S3 bucket for each use case
- B. Create an S3 bucket policy that grants permissions to appropriate individual IAM user
- C. Apply the S3 bucket policy to the S3 bucket.
- D. Create an Athena workgroup for each use case
- E. Apply tags to the workgroup
- F. Create an IAM policy that uses the tags to apply appropriate permissions to the workgroup.
- G. Create an IAM role for each use case
- H. Assign appropriate permissions to the role for each use case
- I. Associate the role with Athena.
- J. Create an AWS Glue Data Catalog resource policy that grants permissions to appropriate individual IAM users for each use case
- K. Apply the resource policy to the specific tables that Athena uses.

Answer: B

Explanation:

Athena workgroups are a way to isolate query execution and query history among users, teams, and applications that share the same AWS account. By creating a workgroup for each use case, the company can control the access and actions on the workgroup resource using resource-level IAM permissions or identity-based IAM policies. The company can also use tags to organize and identify the workgroups, and use them as conditions in the IAM policies to grant or deny permissions to the workgroup. This solution meets the requirements of separating query processes and access to query history among users, teams, and applications that are in the same AWS account. References:

? Athena Workgroups

? IAM policies for accessing workgroups

? Workgroup example policies

NEW QUESTION 2

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 provisioned capacity 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 source
- B. Store metadata in the AWS Glue Data Catalog
- C. Use Amazon Athena to query the data
- D. Use SQL for structured data source
- E. Use PartiQL for data that is stored in JSON format.
- F. Use AWS Glue to crawl the data source
- G. Store metadata in the AWS Glue Data Catalog
- H. Use Redshift Spectrum to query the data
- I. Use SQL for structured data source
- J. Use PartiQL for data that is stored in JSON format.
- K. Use AWS Glue to crawl the data source
- L. Store metadata in the AWS Glue Data Catalog
- M. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format
- N. Store the transformed data in an S3 bucket
- O. Use Amazon Athena to query the original and transformed data from the S3 bucket.
- P. Use AWS Lake Formation to create a data lake
- Q. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet format
- R. Store the transformed data in an S3 bucket
- S. Use Amazon Athena or Redshift Spectrum to query the data.

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 Redshift Spectrum 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]

NEW QUESTION 3

A company stores petabytes of data in thousands of Amazon S3 buckets in the S3 Standard storage class. The data supports analytics workloads that have unpredictable and variable data access patterns.

The company does not access some data for months. However, the company must be able to retrieve all data within milliseconds. The company needs to optimize S3 storage costs.

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

- A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost- optimized storage classe
- B. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classe
- C. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.
- D. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequentl
- E. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.
- F. Use S3 Intelligent-Tierin
- G. Activate the Deep Archive Access tier.
- H. Use S3 Intelligent-Tierin
- I. Use the default access tier.

Answer: D

Explanation:

S3 Intelligent-Tiering is a storage class that automatically moves objects between four access tiers based on the changing access patterns. The default access tier consists of two tiers: Frequent Access and Infrequent Access. Objects in the Frequent Access tier have the same performance and availability as S3 Standard, while objects in the Infrequent Access tier have the same performance and availability as S3 Standard-IA. S3 Intelligent-Tiering monitors the access patterns of each object and moves them between the tiers accordingly, without any operational overhead or retrieval fees. This solution can optimize S3 storage costs for data with unpredictable and variable access patterns, while ensuring millisecond latency for data retrieval. The other solutions are not optimal or relevant for this requirement. Using S3 Storage Lens standard metrics and activity metrics can provide insights into the storage usage and access patterns, but they do not automate the data movement between storage classes. Creating S3 Lifecycle policies for the S3 buckets can move objects to more cost-optimized storage classes, but they require manual configuration and maintenance, and they may incur retrieval fees for data that is accessed unexpectedly. Activating the Deep Archive Access tier for S3 Intelligent-Tiering can further reduce the storage costs for data that is rarely accessed, but it also increases the retrieval time to 12 hours, which does not meet the requirement of millisecond latency. References:

- ? S3 Intelligent-Tiering
- ? S3 Storage Lens
- ? S3 Lifecycle policies
- ? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 4

A data engineer needs to schedule a workflow that runs a set of AWS Glue jobs every day. The data engineer does not require the Glue jobs to run or finish at a specific time.

Which solution will run the Glue jobs in the MOST cost-effective way?

- A. Choose the FLEX execution class in the Glue job properties.
- B. Use the Spot Instance type in Glue job properties.
- C. Choose the STANDARD execution class in the Glue job properties.
- D. Choose the latest version in the GlueVersion field in the Glue job properties.

Answer: A

Explanation:

The FLEX execution class allows you to run AWS Glue jobs on spare compute capacity instead of dedicated hardware. This can reduce the cost of running non-urgent or non-time sensitive data integration workloads, such as testing and one-time data loads. The FLEX execution class is available for AWS Glue 3.0 Spark jobs. The other options are not as cost-effective as FLEX, because they either use dedicated resources (STANDARD) or do not affect the cost at all (Spot Instance type and GlueVersion). References:

? Introducing AWS Glue Flex jobs: Cost savings on ETL workloads

? Serverless Data Integration – AWS Glue Pricing

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 125)

NEW QUESTION 5

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 table
- B. Specify primary and foreign keys for all tables.
- C. Use the ALL distribution style for large table
- D. Specify primary and foreign keys for all tables.
- E. Use the ALL distribution style for rarely updated small table
- F. Specify primary and foreign keys for all tables.
- G. Specify a combination of distribution, sort, and partition keys for all tables.

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 or extensively. 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

NEW QUESTION 6

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.

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

NEW QUESTION 7

A company receives .csv files that contain physical address data. The data is in columns that have the following names: Door_No, Street_Name, City, and Zip_Code. The company wants to create a single column to store these values in the following format:

```
{
  "Door_No": "24",
  "Street_Name": "AAA street",
  "City": "BBB",
  "Zip_Code": "111111"
}
```

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

- A. Use AWS Glue DataBrew to read the file
- B. Use the NEST TO ARRAY transformation to create the new column.
- C. Use AWS Glue DataBrew to read the file
- D. Use the NEST TO MAP transformation to create the new column.
- E. Use AWS Glue DataBrew to read the file
- F. Use the PIVOT transformation to create the new column.
- G. Write a Lambda function in Python to read the file
- H. Use the Python data dictionary type to create the new column.

Answer: B

Explanation:

The NEST TO MAP transformation allows you to combine multiple columns into a single column that contains a JSON object with key-value pairs. This is the easiest way to achieve the desired format for the physical address data, as you can simply select the columns to nest and specify the keys for each column. The NEST TO ARRAY transformation creates a single column that contains an array of values, which is not the same as the JSON object format. The PIVOT transformation reshapes the data by creating new columns from unique values in a selected column, which is not applicable for this use case. Writing a Lambda function in Python requires more coding effort than using AWS Glue DataBrew, which provides a visual and interactive interface for data transformations.

References:

? 7 most common data preparation transformations in AWS Glue DataBrew (Section: Nesting and unnesting columns)

? NEST TO MAP - AWS Glue DataBrew (Section: Syntax)

NEW QUESTION 8

A data engineer needs to use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket. When the data engineer connects to the QuickSight dashboard, the data engineer receives an error message that indicates insufficient permissions.

Which factors could cause the permissions-related errors? (Choose two.)

- A. There is no connection between QuickSight and Athena.
- B. The Athena tables are not cataloged.
- C. QuickSight does not have access to the S3 bucket.
- D. QuickSight does not have access to decrypt S3 data.
- E. There is no IAM role assigned to QuickSight.

Answer: CD

Explanation:

QuickSight does not have access to the S3 bucket and QuickSight does not have access to decrypt S3 data are two possible factors that could cause the permissions-related errors. Amazon QuickSight is a business intelligence service that allows you to create and share interactive dashboards based on various data sources, including Amazon Athena. Amazon Athena is a serverless query service that allows you to analyze data stored in Amazon S3 using standard SQL. To use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket, you need to grant QuickSight access to both Athena and S3, as well as any encryption keys that are used to encrypt the S3 data. If QuickSight does not have access to the S3 bucket or the encryption keys, it will not be able to read the data from Athena and display it on the dashboard, resulting in an error message that indicates insufficient permissions.

The other options are not factors that could cause the permissions-related errors. Option A, there is no connection between QuickSight and Athena, is not a factor, as QuickSight supports Athena as a native data source, and you can easily create a connection between them using the QuickSight console or the API. Option B, the Athena tables are not cataloged, is not a factor, as QuickSight can automatically discover the Athena tables that are cataloged in the AWS Glue Data Catalog, and you can also manually specify the Athena tables that are not cataloged. Option E, there is no IAM role assigned to QuickSight, is not a factor, as QuickSight requires an IAM role to access any AWS data sources, including Athena and S3, and you can create and assign an IAM role to QuickSight using the QuickSight console or the API. References:

? Using Amazon Athena as a Data Source

? Granting Amazon QuickSight Access to AWS Resources

? Encrypting Data at Rest in Amazon S3

NEW QUESTION 9

A manufacturing company wants to collect data from sensors. A data engineer needs to implement a solution that ingests sensor data in near real time.

The solution must store the data to a persistent data store. The solution must store the data in nested JSON format. The company must have the ability to query from the data store with a latency of less than 10 milliseconds.

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

- A. Use a self-hosted Apache Kafka cluster to capture the sensor data
- B. Store the data in Amazon S3 for querying.
- C. Use AWS Lambda to process the sensor data
- D. Store the data in Amazon S3 for querying.
- E. Use Amazon Kinesis Data Streams to capture the sensor data
- F. Store the data in Amazon DynamoDB for querying.
- G. Use Amazon Simple Queue Service (Amazon SQS) to buffer incoming sensor data
- H. Use AWS Glue to store the data in Amazon RDS for querying.

Answer: C

Explanation:

Amazon Kinesis Data Streams is a service that enables you to collect, process, and analyze streaming data in real time. You can use Kinesis Data Streams to capture sensor data from various sources, such as IoT devices, web applications, or mobile apps. You can create data streams that can scale up to handle any amount of data from thousands of producers. You can also use the Kinesis Client Library (KCL) or the Kinesis Data Streams API to write applications that process and analyze the data in the streams¹. Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. You can use DynamoDB to store the sensor data in nested JSON format, as DynamoDB supports document data types, such as lists and maps. You can also use DynamoDB to query the data with a latency of less than 10 milliseconds, as DynamoDB offers single-digit millisecond performance for any scale of data. You can use the DynamoDB API or the AWS SDKs to perform queries on the data, such as using key-value lookups, scans, or queries².

The solution that meets the requirements with the least operational overhead is to use Amazon Kinesis Data Streams to capture the sensor data and store the data in Amazon DynamoDB for querying. This solution has the following advantages:

? It does not require you to provision, manage, or scale any servers, clusters, or queues, as Kinesis Data Streams and DynamoDB are fully managed services that handle all the infrastructure for you. This reduces the operational complexity and cost of running your solution.

? It allows you to ingest sensor data in near real time, as Kinesis Data Streams can capture data records as they are produced and deliver them to your applications within seconds. You can also use Kinesis Data Firehose to load the data from the streams to DynamoDB automatically and continuously³.

? It allows you to store the data in nested JSON format, as DynamoDB supports document data types, such as lists and maps. You can also use DynamoDB Streams to capture changes in the data and trigger actions, such as sending notifications or updating other databases.

? It allows you to query the data with a latency of less than 10 milliseconds, as DynamoDB offers single-digit millisecond performance for any scale of data. You can also use DynamoDB Accelerator (DAX) to improve the read performance by caching frequently accessed data.

Option A is incorrect because it suggests using a self-hosted Apache Kafka cluster to capture the sensor data and store the data in Amazon S3 for querying. This solution has the following disadvantages:

? It requires you to provision, manage, and scale your own Kafka cluster, either on EC2 instances or on-premises servers. This increases the operational complexity and cost of running your solution.

? It does not allow you to query the data with a latency of less than 10 milliseconds, as Amazon S3 is an object storage service that is not optimized for low-latency queries. You need to use another service, such as Amazon Athena or Amazon Redshift Spectrum, to query the data in S3, which may incur additional costs and latency.

Option B is incorrect because it suggests using AWS Lambda to process the sensor data and store the data in Amazon S3 for querying. This solution has the following disadvantages:

? It does not allow you to ingest sensor data in near real time, as Lambda is a serverless compute service that runs code in response to events. You need to use another service, such as API Gateway or Kinesis Data Streams, to trigger Lambda functions with sensor data, which may add extra latency and complexity to your solution.

? It does not allow you to query the data with a latency of less than 10 milliseconds, as Amazon S3 is an object storage service that is not optimized for low-latency queries. You need to use another service, such as Amazon Athena or Amazon Redshift Spectrum, to query the data in S3, which may incur additional costs and latency.

Option D is incorrect because it suggests using Amazon Simple Queue Service (Amazon SQS) to buffer incoming sensor data and use AWS Glue to store the data in Amazon RDS for querying. This solution has the following disadvantages:

? It does not allow you to ingest sensor data in near real time, as Amazon SQS is a message queue service that delivers messages in a best-effort manner. You need to use another service, such as Lambda or EC2, to poll the messages from the queue and process them, which may add extra latency and complexity to your solution.

? It does not allow you to store the data in nested JSON format, as Amazon RDS is a relational database service that supports structured data types, such as tables and columns. You need to use another service, such as AWS Glue, to transform the data from JSON to relational format, which may add extra cost and overhead to your solution.

References:

? 1: Amazon Kinesis Data Streams - Features

? 2: Amazon DynamoDB - Features

? 3: Loading Streaming Data into Amazon DynamoDB - Amazon Kinesis Data Firehose

? [4]: Capturing Table Activity with DynamoDB Streams - Amazon DynamoDB

? [5]: Amazon DynamoDB Accelerator (DAX) - Features

? [6]: Amazon S3 - Features

? [7]: AWS Lambda - Features

? [8]: Amazon Simple Queue Service - Features

? [9]: Amazon Relational Database Service - Features

? [10]: Working with JSON in Amazon RDS - Amazon Relational Database Service

? [11]: AWS Glue - Features

NEW QUESTION 10

A company needs to set up a data catalog and metadata management for data sources that run in the AWS Cloud. The company will use the data catalog to maintain the metadata of all the objects that are in a set of data stores. The data stores include structured sources such as Amazon RDS and Amazon Redshift. The data stores also include semistructured sources such as JSON files and .xml files that are stored in Amazon S3.

The company needs a solution that will update the data catalog on a regular basis. The solution also must detect changes to the source metadata.

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

- A. Use Amazon Aurora as the data catalog
- B. Create AWS Lambda functions that will connect to the data catalog
- C. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog
- D. Schedule the Lambda functions to run periodically.
- E. Use the AWS Glue Data Catalog as the central metadata repository
- F. Use AWS Glue crawlers to connect to multiple data stores and to update the Data Catalog with metadata change
- G. Schedule the crawlers to run periodically to update the metadata catalog.
- H. Use Amazon DynamoDB as the data catalog
- I. Create AWS Lambda functions that will connect to the data catalog
- J. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog
- K. Schedule the Lambda functions to run periodically.
- L. Use the AWS Glue Data Catalog as the central metadata repository
- M. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog
- N. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog.

Answer: B

Explanation:

This solution will meet the requirements with the least operational overhead because it uses the AWS Glue Data Catalog as the central metadata repository for

data sources that run in the AWS Cloud. The AWS Glue Data Catalog is a fully managed service that provides a unified view of your data assets across AWS and on-premises data sources. It stores the metadata of your data in tables, partitions, and columns, and enables you to access and query your data using various AWS services, such as Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can use AWS Glue crawlers to connect to multiple data stores, such as Amazon RDS, Amazon Redshift, and Amazon S3, and to update the Data Catalog with metadata changes. AWS Glue crawlers can automatically discover the schema and partition structure of your data, and create or update the corresponding tables in the Data Catalog. You can schedule the crawlers to run periodically to update the metadata catalog, and configure them to detect changes to the source metadata, such as new columns, tables, or partitions¹².

The other options are not optimal for the following reasons:

? A. Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically. This option is not recommended, as it would require more operational overhead to create and manage an Amazon Aurora database as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? C. Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically. This option is also not recommended, as it would require more operational overhead to create and manage an Amazon DynamoDB table as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? D. Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog. This option is not optimal, as it would require more manual effort to extract the schema for Amazon RDS and Amazon Redshift sources, and to build the Data Catalog. This option would not take advantage of the AWS Glue crawlers' ability to automatically discover the schema and partition structure of your data from various data sources, and to create or update the corresponding tables in the Data Catalog.

References:

? 1: AWS Glue Data Catalog

? 2: AWS Glue Crawlers

? : Amazon Aurora

? : AWS Lambda

? : Amazon DynamoDB

NEW QUESTION 10

A company is migrating its database servers from Amazon EC2 instances that run Microsoft SQL Server to Amazon RDS for Microsoft SQL Server DB instances. The company's analytics team must export large data elements every day until the migration is complete. The data elements are the result of SQL joins across multiple tables. The data must be in Apache Parquet format. The analytics team must store the data in Amazon S3.

Which solution will meet these requirements in the MOST operationally efficient way?

- A. Create a view in the EC2 instance-based SQL Server databases that contains the required data element
- B. Create an AWS Glue job that selects the data directly from the view and transfers the data in Parquet format to an S3 bucket
- C. Schedule the AWS Glue job to run every day.
- D. Schedule SQL Server Agent to run a daily SQL query that selects the desired data elements from the EC2 instance-based SQL Server database
- E. Configure the query to direct the output .csv objects to an S3 bucket
- F. Create an S3 event that invokes an AWS Lambda function to transform the output format from .csv to Parquet.
- G. Use a SQL query to create a view in the EC2 instance-based SQL Server databases that contains the required data element
- H. Create and run an AWS Glue crawler to read the view
- I. Create an AWS Glue job that retrieves the data and transfers the data in Parquet format to an S3 bucket
- J. Schedule the AWS Glue job to run every day.
- K. Create an AWS Lambda function that queries the EC2 instance-based databases by using Java Database Connectivity (JDBC). Configure the Lambda function to retrieve the required data, transform the data into Parquet format, and transfer the data into an S3 bucket
- L. Use Amazon EventBridge to schedule the Lambda function to run every day.

Answer: A

Explanation:

Option A is the most operationally efficient way to meet the requirements because it minimizes the number of steps and services involved in the data export process. AWS Glue is a fully managed service that can extract, transform, and load (ETL) data from various sources to various destinations, including Amazon S3. AWS Glue can also convert data to different formats, such as Parquet, which is a columnar storage format that is optimized for analytics. By creating a view in the SQL Server databases that contains the required data elements, the AWS Glue job can select the data directly from the view without having to perform any joins or transformations on the source data. The AWS Glue job can then transfer the data in Parquet format to an S3 bucket and run on a daily schedule.

Option B is not operationally efficient because it involves multiple steps and services to export the data. SQL Server Agent is a tool that can run scheduled tasks on SQL Server databases, such as executing SQL queries. However, SQL Server Agent cannot directly export data to S3, so the query output must be saved as .csv objects on the EC2 instance. Then, an S3 event must be configured to trigger an AWS Lambda function that can transform the .csv objects to Parquet format and upload them to S3. This option adds complexity and latency to the data export process and requires additional resources and configuration.

Option C is not operationally efficient because it introduces an unnecessary step of running an AWS Glue crawler to read the view. 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. However, in this scenario, the schema and format of the data elements are already known and fixed, so there is no need to run a crawler to discover them. The AWS Glue job can directly select the data from the view without using the Data Catalog. Running a crawler adds extra time and cost to the data export process.

Option D is not operationally efficient because it requires custom code and configuration to query the databases and transform the data. An AWS Lambda function is a service that can run code in response to events or triggers, such as Amazon EventBridge. Amazon EventBridge is a service that can connect applications and services with event sources, such as schedules, and route them to targets, such as Lambda functions. However, in this scenario, using a Lambda function to query the databases and transform the data is not the best option because it requires writing and maintaining code that uses JDBC to connect to the SQL Server databases, retrieve the required data, convert the data to Parquet format, and transfer the data to S3. This option also has limitations on the execution time, memory, and concurrency of the Lambda function, which may affect the performance and reliability of the data export process.

References:

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

? AWS Glue Documentation

? Working with Views in AWS Glue

? Converting to Columnar Formats

NEW QUESTION 11

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.

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 run continuously 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

NEW QUESTION 16

A data engineer must use AWS services to ingest a dataset into an Amazon S3 data lake. The data engineer profiles the dataset and discovers that the dataset contains personally identifiable information (PII). The data engineer must implement a solution to profile the dataset and obfuscate the PII. Which solution will meet this requirement with the LEAST operational effort?

- A. Use an Amazon Kinesis Data Firehose delivery stream to process the dataset
- B. Create an AWS Lambda transform function to identify the PII
- C. Use an AWS SDK to obfuscate the PII
- D. Set the S3 data lake as the target for the delivery stream.
- E. Use the Detect PII transform in AWS Glue Studio to identify the PII
- F. Obfuscate the PII
- G. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.
- H. Use the Detect PII transform in AWS Glue Studio to identify the PII
- I. Create a rule in AWS Glue Data Quality to obfuscate the PII
- J. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.
- K. Ingest the dataset into Amazon DynamoDB
- L. Create an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data
- M. Use the same Lambda function to ingest the data into the S3 data lake.

Answer: C

Explanation:

AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue Studio is a graphical interface that allows you to easily author, run, and monitor AWS Glue ETL jobs. AWS Glue Data Quality is a feature that enables you to validate, cleanse, and enrich your data using predefined or custom rules. AWS Step Functions is a service that allows you to coordinate multiple AWS services into serverless workflows.

Using the Detect PII transform in AWS Glue Studio, you can automatically identify and label the PII in your dataset, such as names, addresses, phone numbers, email addresses, etc. You can then create a rule in AWS Glue Data Quality to obfuscate the PII, such as masking, hashing, or replacing the values with dummy data. You can also use other rules to validate and cleanse your data, such as checking for null values, duplicates, outliers, etc. You can then use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake. You can use AWS Glue DataBrew to visually explore and transform the data, AWS Glue crawlers to discover and catalog the data, and AWS Glue jobs to load the data into the S3 data lake.

This solution will meet the requirement with the least operational effort, as it leverages the serverless and managed capabilities of AWS Glue, AWS Glue Studio, AWS Glue Data Quality, and AWS Step Functions. You do not need to write any code to identify or obfuscate the PII, as you can use the built-in transforms and rules in AWS Glue Studio and AWS Glue Data Quality. You also do not need to provision or manage any servers or clusters, as AWS Glue and AWS Step Functions scale automatically based on the demand. The other options are not as efficient as using the Detect PII transform in AWS Glue Studio, creating a rule in AWS Glue Data Quality, and using an AWS Step Functions state machine. Using an Amazon Kinesis Data Firehose delivery stream to process the dataset, creating an AWS Lambda transform function to identify the PII, using an AWS SDK to obfuscate the PII, and setting the S3 data lake as the target for the delivery stream will require more operational effort, as you will need to write and maintain code to identify and obfuscate the PII, as well as manage the Lambda function and its resources. Using the Detect PII transform in AWS Glue Studio to identify the PII, obfuscating the PII, and using an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake will not be as effective as creating a rule in AWS Glue Data Quality to obfuscate the PII, as you will need to manually obfuscate the PII after identifying it, which can be error-prone and time-consuming. Ingesting the dataset into Amazon DynamoDB, creating an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data, and using the same Lambda function to ingest the data into the S3 data lake will require more operational effort, as you will need to write and maintain code to identify and obfuscate the PII, as well as manage the Lambda function and its resources. You will also incur additional costs and complexity by using DynamoDB as an intermediate data store, which may not be necessary for your use case. References:

? AWS Glue

? AWS Glue Studio

? AWS Glue Data Quality
? [AWS Step Functions]
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

NEW QUESTION 18

A company has a frontend ReactJS website that uses Amazon API Gateway to invoke REST APIs. The APIs perform the functionality of the website. A data engineer needs to write a Python script that can be occasionally invoked through API Gateway. The code must return results to API Gateway. Which solution will meet these requirements with the LEAST operational overhead?

- A. Deploy a custom Python script on an Amazon Elastic Container Service (Amazon ECS) cluster.
- B. Create an AWS Lambda Python function with provisioned concurrency.
- C. Deploy a custom Python script that can integrate with API Gateway on Amazon Elastic Kubernetes Service (Amazon EKS).
- D. Create an AWS Lambda function
- E. Ensure that the function is warm by scheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by using mock events.

Answer: B

Explanation:

AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. You can use Lambda to create functions that perform custom logic and integrate with other AWS services, such as API Gateway. Lambda automatically scales your application by running code in response to each trigger. You pay only for the compute time you consume¹.

Amazon ECS is a fully managed container orchestration service that allows you to run and scale containerized applications on AWS. You can use ECS to deploy, manage, and scale Docker containers using either Amazon EC2 instances or AWS Fargate, a serverless compute engine for containers².

Amazon EKS is a fully managed Kubernetes service that allows you to run Kubernetes clusters on AWS without needing to install, operate, or maintain your own Kubernetes control plane. You can use EKS to deploy, manage, and scale containerized applications using Kubernetes on AWS³.

The solution that meets the requirements with the least operational overhead is to create an AWS Lambda Python function with provisioned concurrency. This solution has the following advantages:

? It does not require you to provision, manage, or scale any servers or clusters, as Lambda handles all the infrastructure for you. This reduces the operational complexity and cost of running your code.

? It allows you to write your Python script as a Lambda function and integrate it with API Gateway using a simple configuration. API Gateway can invoke your Lambda function synchronously or asynchronously, and return the results to the frontend website.

? It ensures that your Lambda function is ready to respond to API requests without any cold start delays, by using provisioned concurrency. Provisioned concurrency is a feature that keeps your function initialized and hyper-ready to respond in double-digit milliseconds. You can specify the number of concurrent executions that you want to provision for your function.

Option A is incorrect because it requires you to deploy a custom Python script on an Amazon ECS cluster. This solution has the following disadvantages:

? It requires you to provision, manage, and scale your own ECS cluster, either using EC2 instances or Fargate. This increases the operational complexity and cost of running your code.

? It requires you to package your Python script as a Docker container image and store it in a container registry, such as Amazon ECR or Docker Hub. This adds an extra step to your deployment process.

? It requires you to configure your ECS cluster to integrate with API Gateway, either using an Application Load Balancer or a Network Load Balancer. This adds another layer of complexity to your architecture.

Option C is incorrect because it requires you to deploy a custom Python script that can integrate with API Gateway on Amazon EKS. This solution has the following disadvantages:

? It requires you to provision, manage, and scale your own EKS cluster, either using EC2 instances or Fargate. This increases the operational complexity and cost of running your code.

? It requires you to package your Python script as a Docker container image and store it in a container registry, such as Amazon ECR or Docker Hub. This adds an extra step to your deployment process.

? It requires you to configure your EKS cluster to integrate with API Gateway, either using an Application Load Balancer, a Network Load Balancer, or a service of type LoadBalancer. This adds another layer of complexity to your architecture.

Option D is incorrect because it requires you to create an AWS Lambda function and ensure that the function is warm by scheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by using mock events. This solution has the following disadvantages:

? It does not guarantee that your Lambda function will always be warm, as Lambda may scale down your function if it does not receive any requests for a long period of time. This may cause cold start delays when your function is invoked by API Gateway.

? It incurs unnecessary costs, as you pay for the compute time of your Lambda function every time it is invoked by the EventBridge rule, even if it does not perform any useful work¹.

References:

? 1: AWS Lambda - Features

? 2: Amazon Elastic Container Service - Features

? 3: Amazon Elastic Kubernetes Service - Features

? [4]: Building API Gateway REST API with Lambda integration - Amazon API Gateway

? [5]: Improving latency with Provisioned Concurrency - AWS Lambda

? [6]: Integrating Amazon ECS with Amazon API Gateway - Amazon Elastic Container Service

? [7]: Integrating Amazon EKS with Amazon API Gateway - Amazon Elastic Kubernetes Service

? [8]: Managing concurrency for a Lambda function - AWS Lambda

NEW QUESTION 21

A data engineer needs to create an AWS Lambda function that converts the format of data from .csv to Apache Parquet. The Lambda function must run only if a user uploads a .csv file to an Amazon S3 bucket.

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

- A. Create an S3 event notification that has an event type of s3:ObjectCreated:*. Use a filter rule to generate notifications only when the suffix includes .cs
- B. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
- C. Create an S3 event notification that has an event type of s3:ObjectTagging:* for objects that have a tag set to .cs
- D. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
- E. Create an S3 event notification that has an event type of s3:*. Use a filter rule to generate notifications only when the suffix includes .cs
- F. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
- G. Create an S3 event notification that has an event type of s3:ObjectCreated:*. Use a filter rule to generate notifications only when the suffix includes .cs
- H. Set an Amazon Simple Notification Service (Amazon SNS) topic as the destination for the event notificatio
- I. Subscribe the Lambda function to the SNS topic.

Answer: A

Explanation:

Option A is the correct answer because it meets the requirements with the least operational overhead. Creating an S3 event notification that has an event type of s3:ObjectCreated:* will trigger the Lambda function whenever a new object is created in the S3 bucket. Using a filter rule to generate notifications only when the suffix includes .csv will ensure that the Lambda function only runs for .csv files. Setting the ARN of the Lambda function as the destination for the event notification will directly invoke the Lambda function without any additional steps.

Option B is incorrect because it requires the user to tag the objects with .csv, which adds an extra step and increases the operational overhead.

Option C is incorrect because it uses an event type of s3:*, which will trigger the Lambda function for any S3 event, not just object creation. This could result in unnecessary invocations and increased costs.

Option D is incorrect because it involves creating and subscribing to an SNS topic, which adds an extra layer of complexity and operational overhead.

References:

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.2: S3 Event Notifications and Lambda Functions, Pages 67-69

? Building Batch Data Analytics Solutions on AWS, Module 4: Data Transformation, Lesson 4.2: AWS Lambda, Pages 4-8

? AWS Documentation Overview, AWS Lambda Developer Guide, Working with AWS Lambda Functions, Configuring Function Triggers, Using AWS Lambda with Amazon S3, Pages 1-5

NEW QUESTION 26

A company loads transaction data for each day into Amazon Redshift tables at the end of each day. The company wants to have the ability to track which tables have been loaded and which tables still need to be loaded.

A data engineer wants to store the load statuses of Redshift tables in an Amazon DynamoDB table. The data engineer creates an AWS Lambda function to publish the details of the load statuses to DynamoDB.

How should the data engineer invoke the Lambda function to write load statuses to the DynamoDB table?

- A. Use a second Lambda function to invoke the first Lambda function based on Amazon CloudWatch events.
- B. Use the Amazon Redshift Data API to publish an event to Amazon EventBridge
- C. Configure an EventBridge rule to invoke the Lambda function.
- D. Use the Amazon Redshift Data API to publish a message to an Amazon Simple Queue Service (Amazon SQS) queue
- E. Configure the SQS queue to invoke the Lambda function.
- F. Use a second Lambda function to invoke the first Lambda function based on AWS CloudTrail events.

Answer: B

Explanation:

The Amazon Redshift Data API enables you to interact with your Amazon Redshift data warehouse in an easy and secure way. You can use the Data API to run SQL commands, such as loading data into tables, without requiring a persistent connection to the cluster. The Data API also integrates with Amazon EventBridge, which allows you to monitor the execution status of your SQL commands and trigger actions based on events. By using the Data API to publish an event to EventBridge, the data engineer can invoke the Lambda function that writes the load statuses to the DynamoDB table. This solution is scalable, reliable, and cost-effective. The other options are either not possible or not optimal. You cannot use a second Lambda function to invoke the first Lambda function based on CloudWatch or CloudTrail events, as these services do not capture the load status of Redshift tables. You can use the Data API to publish a message to an SQS queue, but this would require additional configuration and polling logic to invoke the Lambda function from the queue. This would also introduce additional latency and cost. References:

? Using the Amazon Redshift Data API

? Using Amazon EventBridge with Amazon Redshift

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.2: Amazon Redshift

NEW QUESTION 31

A company extracts approximately 1 TB of data every day from data sources such as SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. Some of the data sources have undefined data schemas or data schemas that change.

A data engineer must implement a solution that can detect the schema for these data sources. The solution must extract, transform, and load the data to an Amazon S3 bucket. The company has a service level agreement (SLA) to load the data into the S3 bucket within 15 minutes of data creation.

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

- A. Use Amazon EMR to detect the schema and to extract, transform, and load the data into the S3 bucket
- B. Create a pipeline in Apache Spark.
- C. Use AWS Glue to detect the schema and to extract, transform, and load the data into the S3 bucket
- D. Create a pipeline in Apache Spark.
- E. Create a PySpark program in AWS Lambda to extract, transform, and load the data into the S3 bucket.
- F. Create a stored procedure in Amazon Redshift to detect the schema and to extract, transform, and load the data into a Redshift Spectrum table
- G. Access the table from Amazon S3.

Answer: B

Explanation:

AWS Glue is a fully managed service that provides a serverless data integration platform. It can automatically discover and categorize data from various sources, including SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. It can also infer the schema of the data and store it in the AWS Glue Data Catalog, which is a central metadata repository. AWS Glue can then use the schema information to generate and run Apache Spark code to extract, transform, and load the data into an Amazon S3 bucket. AWS Glue can also monitor and optimize the performance and cost of the data pipeline, and handle any schema changes that may occur in the source data. AWS Glue can meet the SLA of loading the data into the S3 bucket within 15 minutes of data creation, as it can trigger the data pipeline based on events, schedules, or on-demand. AWS Glue has the least operational overhead among the options, as it does not require provisioning, configuring, or managing any servers or clusters. It also handles scaling, patching, and security automatically. References:

? AWS Glue

? [AWS Glue Data Catalog]

? [AWS Glue Developer Guide]

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

NEW QUESTION 32

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

Answer: C

Explanation:

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 rewrites the 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

NEW QUESTION 36

A company uses an on-premises Microsoft SQL Server database to store financial transaction data. The company migrates the transaction data from the on-premises database to AWS at the end of each month. The company has noticed that the cost to migrate data from the on-premises database to an Amazon RDS for SQL Server database has increased recently.

The company requires a cost-effective solution to migrate the data to AWS. The solution must cause minimal downtime for the applications that access the database.

Which AWS service should the company use to meet these requirements?

- A. AWS Lambda
- B. AWS Database Migration Service (AWS DMS)
- C. AWS Direct Connect
- D. AWS DataSync

Answer: B

Explanation:

AWS Database Migration Service (AWS DMS) is a cloud service that makes it possible to migrate relational databases, data warehouses, NoSQL databases, and other types of data stores to AWS quickly, securely, and with minimal downtime and zero data loss¹. AWS DMS supports migration between 20-plus database and analytics engines, such as Microsoft SQL Server to Amazon RDS for SQL Server². AWS DMS takes over many of the difficult or tedious tasks involved in a migration project, such as capacity analysis, hardware and software procurement, installation and administration, testing and debugging, and ongoing replication and monitoring¹. AWS DMS is a cost-effective solution, as you only pay for the compute resources and additional log storage used during the migration process². AWS DMS is the best solution for the company to migrate the financial transaction data from the on-premises Microsoft SQL Server database to AWS, as it meets the requirements of minimal downtime, zero data loss, and low cost.

Option A is not the best solution, as AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers, but it does not provide any built-in features for database migration. You would have to write your own code to extract, transform, and load the data from the source to the target, which would increase the operational overhead and complexity.

Option C is not the best solution, as AWS Direct Connect is a service that establishes a dedicated network connection from your premises to AWS, but it does not provide any built-in features for database migration. You would still need to use another service or tool to perform the actual data transfer, which would increase the cost and complexity.

Option D is not the best solution, as AWS DataSync is a service that makes it easy to transfer data between on-premises storage systems and AWS storage services, such as Amazon S3, Amazon EFS, and Amazon FSx for Windows File Server, but it does not support Amazon RDS for SQL Server as a target. You would have to use another service or tool to migrate the data from Amazon S3 to Amazon RDS for SQL Server, which would increase the latency and complexity.

References:

? Database Migration - AWS Database Migration Service - AWS

? What is AWS Database Migration Service?

? AWS Database Migration Service Documentation

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

NEW QUESTION 41

A media company wants to improve a system that recommends media content to customer based on user behavior and preferences. To improve the recommendation system, the company needs to incorporate insights from third-party datasets into the company's existing analytics platform.

The company wants to minimize the effort and time required to incorporate third-party datasets.

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

- A. Use API calls to access and integrate third-party datasets from AWS Data Exchange.
- B. Use API calls to access and integrate third-party datasets from AWS
- C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories.
- D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR).

Answer: A

Explanation:

AWS Data Exchange is a service that makes it easy to find, subscribe to, and use third-party data in the cloud. It provides a secure and reliable way to access and integrate data from various sources, such as data providers, public datasets, or AWS services. Using AWS Data Exchange, you can browse and subscribe to data products that suit your needs, and then use API calls or the AWS Management Console to export the data to Amazon S3, where you can use it with your existing analytics platform. This solution minimizes the effort and time required to incorporate third-party datasets, as you do not need to set up and manage data pipelines, storage, or access controls. You also benefit from the data quality and freshness provided by the data providers, who can update their data products as frequently as needed¹².

The other options are not optimal for the following reasons:

? B. Use API calls to access and integrate third-party datasets from AWS. This option is vague and does not specify which AWS service or feature is used to access and integrate third-party datasets. AWS offers a variety of services and features that can help with data ingestion, processing, and analysis, but not all of them are suitable for the given scenario. For example, AWS Glue is a serverless data integration service that can help you discover, prepare, and combine data from various sources, but it requires you to create and run data extraction, transformation, and loading (ETL) jobs, which can add operational overhead³.

? C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories. This option is not feasible, as AWS CodeCommit is a source control service that hosts secure Git-based repositories, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams is a service that enables you to capture, process, and analyze data streams in real time, such as clickstream data, application logs, or IoT telemetry. It does not support accessing and integrating data from AWS CodeCommit repositories, which are meant for storing and managing code, not data .

? D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR). This option is also not feasible, as Amazon ECR is a fully managed container registry service that stores, manages, and deploys container images, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams does not support accessing and integrating data from Amazon ECR, which is meant for storing and managing container images, not data .

References:

? 1: AWS Data Exchange User Guide

? 2: AWS Data Exchange FAQs

? 3: AWS Glue Developer Guide

? : AWS CodeCommit User Guide

? : Amazon Kinesis Data Streams Developer Guide

? : Amazon Elastic Container Registry User Guide

? : Build a Continuous Delivery Pipeline for Your Container Images with Amazon ECR as Source

NEW QUESTION 44

A data engineering team is using an Amazon Redshift data warehouse for operational reporting. The team wants to prevent performance issues that might result from long- running queries. A data engineer must choose a system table in Amazon Redshift to record anomalies when a query optimizer identifies conditions that might indicate performance issues.

Which table views should the data engineer use to meet this requirement?

- A. STL USAGE CONTROL
- B. STL ALERT EVENT LOG
- C. STL QUERY METRICS
- D. STL PLAN INFO

Answer: B

Explanation:

The STL ALERT EVENT LOG table view records anomalies when the query optimizer identifies conditions that might indicate performance issues. These conditions include skewed data distribution, missing statistics, nested loop joins, and broadcasted data. The STL ALERT EVENT LOG table view can help the data engineer to identify and troubleshoot the root causes of performance issues and optimize the query execution plan. The other table views are not relevant for this requirement. STL USAGE CONTROL records the usage limits and quotas for Amazon Redshift resources. STL QUERY METRICS records the execution time and resource consumption of queries. STL PLAN INFO records the query execution plan and the steps involved in each query. References:

? STL ALERT EVENT LOG

? System Tables and Views

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

NEW QUESTION 47

A data engineer must ingest a source of structured data that is in .csv format into an Amazon S3 data lake. The .csv files contain 15 columns. Data analysts need to run Amazon Athena queries on one or two columns of the dataset. The data analysts rarely query the entire file.

Which solution will meet these requirements MOST cost-effectively?

- A. Use an AWS Glue PySpark job to ingest the source data into the data lake in .csv format.
- B. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source.
- C. Configure the job to ingest the data into the data lake in JSON format.
- D. Use an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format.
- E. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source.
- F. Configure the job to write the data into the data lake in Apache Parquet format.

Answer: D

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, creating an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source and writing the data into the data lake in Apache Parquet format will meet the requirements most cost-effectively. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue ETL jobs allow you to transform and load data from various sources into various targets, using either a graphical interface (AWS Glue Studio) or a code-based interface (AWS Glue console or AWS Glue API). By using AWS Glue ETL jobs, you can easily convert the data from CSV to Parquet format, without having to write or manage any code. 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. 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 cost-effective as creating an AWS Glue ETL job to write the data into the data lake in Parquet format. Using an AWS Glue PySpark job to ingest the source data into the data lake in .csv format will not improve the query performance or reduce the query cost, as .csv is a row-oriented format that does not support columnar access or compression. Creating an AWS Glue ETL job to ingest the data into the data lake in JSON format will not improve the query performance or reduce the query cost, as JSON is also a row-oriented format that does not support columnar access or compression. Using an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format will improve the query performance, as Avro is a column-oriented format that supports compression and encoding, but it will require more operational effort, as you will need to write and maintain PySpark code to convert the data from CSV to Avro format. References:

? Amazon Athena

? Choosing the Right Data Format

? AWS Glue

? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 5: Data Analysis and Visualization, Section 5.1: Amazon Athena

NEW QUESTION 48

A company needs to build a data lake in AWS. The company must provide row-level data access and column-level data access to specific teams. The teams will access the data by using Amazon Athena, Amazon Redshift Spectrum, and Apache Hive from Amazon EMR.

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

- A. Use Amazon S3 for data lake storag
- B. Use S3 access policies to restrict data access by rows and column
- C. Provide data access throughAmazon S3.
- D. Use Amazon S3 for data lake storag
- E. Use Apache Ranger through Amazon EMR to restrict data access byrows and column
- F. Providedata access by using Apache Pig.
- G. Use Amazon Redshift for data lake storag
- H. Use Redshift security policies to restrict data access byrows and column
- I. Provide data accessby usingApache Spark and Amazon Athena federated queries.
- J. UseAmazon S3 for data lake storag
- K. Use AWS Lake Formation to restrict data access by rows and column
- L. Provide data access through AWS Lake Formation.

Answer: D

Explanation:

Option D is the best solution to meet the requirements with the least operational overhead because AWS Lake Formation is a fully managed service that simplifies the process of building, securing, and managing data lakes. AWS Lake Formation allows you to define granular data access policies at the row and column level for different users and groups. AWS Lake Formation also integrates with Amazon Athena, Amazon Redshift Spectrum, and Apache Hive on Amazon EMR, enabling these services to access the data in the data lake through AWS Lake Formation.

Option A is not a good solution because S3 access policies cannot restrict data access by rows and columns. S3 access policies are based on the identity and permissions of the requester, the bucket and object ownership, and the object prefix and tags. S3 access policies cannot enforce fine-grained data access control at the row and column level. Option B is not a good solution because it involves using Apache Ranger and Apache Pig, which are not fully managed services and require additional configuration and maintenance. Apache Ranger is a framework that provides centralized security administration for data stored in Hadoop clusters, such as Amazon EMR. Apache Ranger can enforce row-level and column-level access policies for Apache Hive tables. However, Apache Ranger is not a native AWS service and requires manual installation and configuration on Amazon EMR clusters. Apache Pig is a platform that allows you to analyze large data sets using a high-level scripting language called Pig Latin. Apache Pig can access data stored in Amazon S3 and process it using Apache Hive. However,Apache Pig is not a native AWS service and requires manual installation and configuration on Amazon EMR clusters.

Option C is not a good solution because Amazon Redshift is not a suitable service for data lake storage. Amazon Redshift is a fully managed data warehouse service that allows you to run complex analytical queries using standard SQL. Amazon Redshift can enforce row- level and column-level access policies for different users and groups. However, Amazon Redshift is not designed to store and process large volumes of unstructured or semi- structured data, which are typical characteristics of data lakes. Amazon Redshift is also more expensive and less scalable than Amazon S3 for data lake storage.

References:

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

? What Is AWS Lake Formation? - AWS Lake Formation

? Using AWS Lake Formation with Amazon Athena - AWS Lake Formation

? Using AWS Lake Formation with Amazon Redshift Spectrum - AWS Lake Formation

? Using AWS Lake Formation with Apache Hive on Amazon EMR - AWS Lake Formation

? Using Bucket Policies and User Policies - Amazon Simple Storage Service

? Apache Ranger

? Apache Pig

? What Is Amazon Redshift? - Amazon Redshift

NEW QUESTION 53

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 Firehos
- B. Load the data into Amazon Redshift.
- C. Use the streaming ingestion feature of Amazon Redshift.
- D. Load the data into Amazon S3. Use the COPY command to load the data into Amazon Redshift.
- E. Use the Amazon Aurora zero-ETL integration with Amazon Redshift.

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

NEW QUESTION 57

A company uses Amazon RDS to store transactional data. The company runs an RDS DB instance in a private subnet. A developer wrote an AWS Lambda function with default settings to insert, update, or delete data in the DB instance.

The developer needs to give the Lambda function the ability to connect to the DB instance privately without using the public internet.

Which combination of steps will meet this requirement with the LEAST operational overhead? (Choose two.)

- A. Turn on the public access setting for the DB instance.
- B. Update the security group of the DB instance to allow only Lambda function invocations on the database port.
- C. Configure the Lambda function to run in the same subnet that the DB instance uses.
- D. Attach the same security group to the Lambda function and the DB instance.
- E. Include a self-referencing rule that allows access through the database port.
- F. Update the network ACL of the private subnet to include a self-referencing rule that allows access through the database port.

Answer: CD

Explanation:

To enable the Lambda function to connect to the RDS DB instance privately without using the public internet, the best combination of steps is to configure the Lambda function to run in the same subnet that the DB instance uses, and attach the same security group to the Lambda function and the DB instance. This way, the Lambda function and the DB instance can communicate within the same private network, and the security group can allow traffic between them on the database port. This solution has the least operational overhead, as it does not require any changes to the public access setting, the network ACL, or the security group of the DB instance.

The other options are not optimal for the following reasons:

? A. Turn on the public access setting for the DB instance. This option is not recommended, as it would expose the DB instance to the public internet, which can compromise the security and privacy of the data. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

? B. Update the security group of the DB instance to allow only Lambda function invocations on the database port. This option is not sufficient, as it would only modify the inbound rules of the security group of the DB instance, but not the outbound rules of the security group of the Lambda function. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

? E. Update the network ACL of the private subnet to include a self-referencing rule

that allows access through the database port. This option is not necessary, as the network ACL of the private subnet already allows all traffic within the subnet by default. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

References:

? 1: Connecting to an Amazon RDS DB instance

? 2: Configuring a Lambda function to access resources in a VPC

? 3: Working with security groups

? : Network ACLs

NEW QUESTION 59

A company currently stores all of its data in Amazon S3 by using the S3 Standard storage class.

A data engineer examined data access patterns to identify trends. During the first 6 months, most data files are accessed several times each day. Between 6 months and 2 years, most data files are accessed once or twice each month. After 2 years, data files are accessed only once or twice each year.

The data engineer needs to use an S3 Lifecycle policy to develop new data storage rules. The new storage solution must continue to provide high availability.

Which solution will meet these requirements in the MOST cost-effective way?

- A. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months.
- B. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.
- C. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months.
- D. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.
- E. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months.
- F. Transfer objects to S3 Glacier Deep Archive after 2 years.
- G. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months.
- H. Transfer objects to S3 Glacier Deep Archive after 2 years.

Answer: C

Explanation:

To achieve the most cost-effective storage solution, the data engineer needs to use an S3 Lifecycle policy that transitions objects to lower-cost storage classes based on their access patterns, and deletes them when they are no longer needed. The storage classes should also provide high availability, which means they should be resilient to the loss of data in a single Availability Zone¹. Therefore, the solution must include the following steps:

? Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. S3 Standard-IA is designed for data that is accessed less frequently, but requires rapid access when needed. It offers the same high durability, throughput, and low latency as S3 Standard, but with a lower storage cost and a retrieval fee².

Therefore, it is suitable for data files that are accessed once or twice each month. S3 Standard-IA also provides high availability, as it stores data redundantly across multiple Availability Zones¹.

? Transfer objects to S3 Glacier Deep Archive after 2 years. S3 Glacier Deep Archive is the lowest-cost storage class that offers secure and durable storage for data that is rarely accessed and can tolerate a 12-hour retrieval time. It is ideal for long-term archiving and digital preservation³. Therefore, it is suitable for data files that are accessed only once or twice each year. S3 Glacier Deep Archive also provides high availability, as it stores data across at least three geographically dispersed Availability Zones¹.

? Delete objects when they are no longer needed. The data engineer can specify an expiration action in the S3 Lifecycle policy to delete objects after a certain period of time. This will reduce the storage cost and comply with any data retention policies.

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

Option A is incorrect because it transitions objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. S3 One Zone-IA is similar to S3 Standard-IA, but it stores data in a single Availability Zone. This means it has a lower availability and durability than S3 Standard-IA, and it is not resilient to the loss of data in a single Availability Zone¹. Therefore, it does not provide high availability as required.

Option B is incorrect because it transfers objects to S3 Glacier Flexible Retrieval after 2 years. S3 Glacier Flexible Retrieval is a storage class that offers secure and durable storage for data that is accessed infrequently and can tolerate a retrieval time of minutes to hours. It is more expensive than S3 Glacier Deep Archive, and it is not suitable for data that is accessed only once or twice each year³. Therefore, it is not the most cost-effective option.

Option D is incorrect because it combines the errors of option A and B. It transitions objects to S3 One Zone-IA after 6 months, which does not provide high availability, and it transfers objects to S3 Glacier Flexible Retrieval after 2 years, which is not the most cost-effective option.

References:

? 1: Amazon S3 storage classes - Amazon Simple Storage Service

? 2: Amazon S3 Standard-Infrequent Access (S3 Standard-IA) - Amazon Simple Storage Service

? 3: Amazon S3 Glacier and S3 Glacier Deep Archive - Amazon Simple Storage Service

? [4]: Expiring objects - Amazon Simple Storage Service

? [5]: Managing your storage lifecycle - Amazon Simple Storage Service

? [6]: Examples of S3 Lifecycle configuration - Amazon Simple Storage Service

? [7]: Amazon S3 Lifecycle further optimizes storage cost savings with new features

- What's New with AWS

NEW QUESTION 60

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? (Choose two.)

- 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: BC

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. References:

? 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

NEW QUESTION 62

A retail company has a customer data hub in an Amazon S3 bucket. Employees from many countries use the data hub to support company-wide analytics. A governance team must ensure that the company's data analysts can access data only for customers who are within the same country as the analysts.

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

- A. Create a separate table for each country's customer data
- B. Provide access to each analyst based on the country that the analyst serves.
- C. Register the S3 bucket as a data lake location in AWS Lake Formation
- D. Use the Lake Formation row-level security features to enforce the company's access policies.
- E. Move the data to AWS Regions that are close to the countries where the customers are
- F. Provide access to each analyst based on the country that the analyst serves.
- G. Load the data into Amazon Redshift
- H. Create a view for each country
- I. Create separate IAM roles for each country to provide access to data from each country
- J. Assign the appropriate roles to the analysts.

Answer: B

Explanation:

AWS Lake Formation is a service that allows you to easily set up, secure, and manage data lakes. One of the features of Lake Formation is row-level security, which enables you to control access to specific rows or columns of data based on the identity or role of the user. This feature is useful for scenarios where you need to restrict access to sensitive or regulated data, such as customer data from different countries. By registering the S3 bucket as a data lake location in Lake Formation, you can use the Lake Formation console or APIs to define and apply row-level security policies to the data in the bucket. You can also use Lake Formation blueprints to automate the ingestion and transformation of data from various sources into the data lake. This solution requires the least operational effort compared to the other options, as it does not involve creating or moving data, or managing multiple tables, views, or roles. References:

? AWS Lake Formation

? Row-Level Security

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Lakes and Data Warehouses, Section 4.2: AWS Lake Formation

NEW QUESTION 66

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 extract(year FROM sales_data) = 2023.
- C. Add HAVING sum(sales amount) > 0 after the GROUP BY clause.
- D. Remove the GROUP BY clause

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 query will 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)

NEW QUESTION 67

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?

- 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 object
- E. 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.

References:

? 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

NEW QUESTION 69

A company stores details about transactions in an Amazon S3 bucket. The company wants to log all writes to the S3 bucket into another S3 bucket that is in the same AWS Region.

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

- A. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function
- B. Program the Lambda function to write the event to Amazon Kinesis Data Firehose
- C. Configure Kinesis Data Firehose to write the event to the logs S3 bucket.
- D. Create a trail of management events in AWS CloudTrail
- E. Configure the trail to receive data from the transactions S3 bucket
- F. Specify an empty prefix and write-only event
- G. Specify the logs S3 bucket as the destination bucket.
- H. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function
- I. Program the Lambda function to write the events to the logs S3 bucket.
- J. Create a trail of data events in AWS CloudTrail
- K. Configure the trail to receive data from the transactions S3 bucket
- L. Specify an empty prefix and write-only event
- M. Specify the logs S3 bucket as the destination bucket.

Answer: D

Explanation:

This solution meets the requirement of logging all writes to the S3 bucket into another S3 bucket with the least operational effort. AWS CloudTrail is a service that records the API calls made to AWS services, including Amazon S3. By creating a trail of data events, you can capture the details of the requests that are made to the transactions S3 bucket, such as the requester, the time, the IP address, and the response elements. By specifying an empty prefix and write-only events, you can filter the data events to only include the ones that write to the bucket. By specifying the logs S3 bucket as the destination bucket, you can store the CloudTrail logs in another S3 bucket that is in the same AWS Region. This solution does not require any additional coding or configuration, and it is more scalable and reliable than using S3 Event Notifications and Lambda functions. References:

? Logging Amazon S3 API calls using AWS CloudTrail

? Creating a trail for data events

? Enabling Amazon S3 server access logging

NEW QUESTION 71

A data engineer needs to maintain a central metadata repository that users access through Amazon EMR and Amazon Athena queries. The repository needs to provide the schema and properties of many tables. Some of the metadata is stored in Apache Hive. The data engineer needs to import the metadata from Hive into the central metadata repository.

Which solution will meet these requirements with the LEAST development effort?

- A. Use Amazon EMR and Apache Ranger.
- B. Use a Hive metastore on an EMR cluster.
- C. Use the AWS Glue Data Catalog.
- D. Use a metastore on an Amazon RDS for MySQL DB instance.

Answer: C

Explanation:

The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such as setting up Apache Ranger for security, managing a Hive metastore on an EMR cluster or an RDS instance, or migrating the metadata manually. References:

? Using the AWS Glue Data Catalog as the metastore for Hive (Section: Specifying

AWS Glue Data Catalog as the metastore)

? Metadata Management: Hive Metastore vs AWS Glue (Section: AWS Glue Data Catalog)

? AWS Glue Data Catalog support for Spark SQL jobs (Section: Importing metadata from an existing Hive metastore)

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 131)

NEW QUESTION 72

.....

Thank You for Trying Our Product

* **100% Pass or Money Back**

All our products come with a 90-day Money Back Guarantee.

* **One year free update**

You can enjoy free update one year. 24x7 online support.

* **Trusted by Millions**

We currently serve more than 30,000,000 customers.

* **Shop Securely**

All transactions are protected by VeriSign!

100% Pass Your AWS-Certified-Data-Engineer-Associate Exam with Our Prep Materials Via below:

<https://www.certleader.com/AWS-Certified-Data-Engineer-Associate-dumps.html>