Amazon

Exam Questions AWS-Certified-Data-Engineer-Associate

AWS Certified Data Engineer - Associate (DEA-C01)

NEW QUESTION 1
A data engineer needs to join data from multiple sources to perform a one-time analysis job. The data is stored in Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3.
Which solution will meet this requirement MOST cost-effectively?

A. Use an Amazon EMR provisioned cluster to read from all source
B. Use Apache Spark to join the data and perform the analysis.
C. Copy the data from DynamoDB, Amazon RDS, and Amazon Redshift into Amazon S3. Run Amazon Athena queries directly on the S3 files.
D. Use Amazon Athena Federated Query to join the data from all data sources.
E. Use Redshift Spectrum to query data from DynamoDB, Amazon RDS, and Amazon S3 directly from Redshift.

Answer: C

Explanation:
Amazon Athena Federated Query is a feature that allows you to query data from multiple sources using standard SQL. You can use Athena Federated Query to join data from Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3, as well as other data sources such as MongoDB, Apache HBase, and Apache Kafka1. Athena Federated Query is a serverless and interactive service, meaning you do not need to provision or manage any infrastructure, and you only pay for the amount of data scanned by your queries. Athena Federated Query is the most cost-effective solution for performing a one-time analysis job on data from multiple sources, as it eliminates the need to copy or move data, and allows you to query data directly from the source.
The other options are not as cost-effective as Athena Federated Query, as they involve additional steps or costs. Option A requires you to provision and pay for an Amazon EMR cluster, which can be expensive and time-consuming for a one-time job. Option B requires you to copy or move data from DynamoDB, RDS, and Redshift to S3, which can incur additional costs for data transfer and storage, and also introduce latency and complexity. Option D requires you to have an existing Redshift cluster, which can be costly and may not be necessary for a one-time job. Option D also does not supportquerying data from RDS directly, so you would need to use Redshift Federated Query to access RDS data, which adds another layer of complexity2. References:
? Amazon Athena Federated Query
? Redshift Spectrum vs Federated Query

NEW QUESTION 2
A company uses an Amazon QuickSight dashboard to monitor usage of one of the company's applications. The company uses AWS Glue jobs to process data for the dashboard. The company stores the data in a single Amazon S3 bucket. The company adds new data every day.
A data engineer discovers that dashboard queries are becoming slower over time. The data engineer determines that the root cause of the slowing queries is long-running AWS Glue jobs.
Which actions should the data engineer take to improve the performance of the AWS Glue jobs? (Choose two.)

A. Partition the data that is in the S3 bucke
B. Organize the data by year, month, and day.
C. Increase the AWS Glue instance size by scaling up the worker type.
D. Convert the AWS Glue schema to the DynamicFrame schema class.
E. Adjust AWS Glue job scheduling frequency so the jobs run half as many times each day.
F. Modify the 1AM role that grants access to AWS glue to grant access to all S3 features.

Answer: AB

Explanation:
Partitioning the data in the S3 bucket can improve the performance of AWS Glue jobs by reducing the amount of data that needs to be scanned and processed. By organizingthe data by year, month, and day, the AWS Glue job can use partition pruning to filter out irrelevant data and only read the data that matches the query criteria. This can speed up the data processing and reduce the cost of running the AWS Glue job. Increasing the AWS Glue instance size by scaling up the worker type can also improve the performance of AWS Glue jobs by providing more memory and CPU resources for the Spark execution engine. This can help the AWS Glue job handle larger data sets and complex transformations more efficiently. The other options are either incorrect or irrelevant, as they do not affect the performance of the AWS Glue jobs. Converting the AWS Glue schema to the DynamicFrame schema class does not improve the performance, but rather provides additional functionality and flexibility for data manipulation. Adjusting the AWS Glue job scheduling frequency does not improve the performance, but rather reduces the frequency of data updates. Modifying the IAM role that grants access to AWS Glue does not improve the performance, but rather affects the security and permissions of the AWS Glue service. References:
? Optimising Glue Scripts for Efficient Data Processing: Part 1 (Section: Partitioning Data in S3)
? Best practices to optimize cost and performance for AWS Glue streaming ETL jobs (Section: Development tools)
? Monitoring with AWS Glue job run insights (Section: Requirements)
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 133)

NEW QUESTION 3
A company stores datasets in JSON format and .csv format in an Amazon S3 bucket. The company has Amazon RDS for Microsoft SQL Server databases, Amazon DynamoDB tables that are in provisionedcapacity mode, and an Amazon Redshift cluster. A data engineering team must develop a solution that will give data scientists the ability to query all data sources by using syntax similar to SQL.
Which solution will meet these requirements with the LEAST operational overhead?

A. Use AWS Glue to crawl the data source
B. Store metadata in the AWS Glue Data Catalo
C. Use Amazon Athena to query the dat
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 Catalo
H. Use Redshift Spectrum to query the dat
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 Catalo
M. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv forma
N. Store the transformed data in an S3 bucke
O. Use Amazon Athena to query the original and transformed data from the S3 bucket.

P. Use AWS Lake Formation to create a data lak
Q. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet forma
R. Store the transformed data in an S3 bucke
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 RedshiftSpectrum to query the data, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Lake Formation is a service that helps you centrally govern, secure, and globally share data for analytics and machine learning7. Lake Formation jobs are ETL jobs that you can create and run using the Lake Formation console or API. While using Lake Formation and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to create, run, and monitor the Lake Formation jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using Lake Formation and Parquet would introduce additional latency, as you would have to wait for the Lake Formation jobs to finish before querying the transformed data. Furthermore, using Redshift Spectrum to query the data would also incur the same costs and complexity as mentioned in option B. References:
? What is Amazon Athena?
? Data Catalog and crawlers in AWS Glue
? AWS Glue Data Catalog
? Columnar Storage Formats
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
? AWS Glue Schema Registry
? What is AWS Glue?
? Amazon Redshift Serverless
? Amazon Redshift provisioned clusters
? [Querying external data using Amazon Redshift Spectrum]
? [Using stored procedures in Amazon Redshift]
? [What is AWS Lambda?]
? [PartiQL for Amazon Athena]
? [Federated queries in Amazon Athena]
? [Amazon Athena pricing]
? [Top 10 performance tuning tips for Amazon Athena]
? [AWS Glue ETL jobs]
? [AWS Lake Formation jobs]

NEW QUESTION 4
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 orextensively. By specifying primary and foreign keys for all tables, you can help the query optimizer to generate better query plans and avoid unnecessary scans or joins. You can also use the AUTO distribution style to let Amazon Redshift choose the optimal distribution style based on the table size and the query patterns. References:
? Choose the best distribution style
? Distribution styles
? Working with data distribution styles

NEW QUESTION 5
A company uses Amazon S3 to store semi-structured data in a transactional data lake. Some of the data files are small, but other data files are tens of terabytes. A data engineer must perform a change data capture (CDC) operation to identify changed data from the data source. The data source sends a full snapshot as a JSON file every day and ingests the changed data into the data lake.
Which solution will capture the changed data MOST cost-effectively?

A. Create an AWS Lambda function to identify the changes between the previous data and the current dat
B. Configure the Lambda function to ingest the changes into the data lake.
C. Ingest the data into Amazon RDS for MySQ
D. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.
E. Use an open source data lake format to merge the data source with the S3 data lake to insert the new data and update the existing data.
F. Ingest the data into an Amazon Aurora MySQL DB instance that runs Aurora Serverles
G. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.

Answer: C

Explanation:
An open source data lake format, such as Apache Parquet, Apache ORC, or Delta Lake, is a cost-effective way to perform a change data capture (CDC) operation on semi-structured data stored in Amazon S3. An open source data lake format allows you to query data directly from S3 using standard SQL, without the need to move or copy data to another service. An open source data lake format also supports schema evolution, meaning it can handle changes in the data structure over time. An open source data lake format also supports upserts, meaning it can insert new data and update existing data in the same operation, using a merge command. This way, you can efficiently capture the changes from the data source and apply them to the S3 data lake, without duplicating or losing any data. The other options are not as cost-effective as using an open source data lake format, as they involve additional steps or costs. Option A requires you to create and maintain an AWS Lambda function, which can be complex and error-prone. AWS Lambda also has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the CDC operation. Option B and D require you to ingest the data into a relational database service, such as Amazon RDS or Amazon Aurora, which can be expensive and unnecessary for semi-structured data. AWS Database Migration Service (AWS DMS) can write the changed data to the data lake, but it alsocharges you for the data replication and transfer. Additionally, AWS DMS does not support JSON as a source data type, so you would need to convert the data to a supported format before using AWS DMS. References:
? What is a data lake?
? Choosing a data format for your data lake
? Using the MERGE INTO command in Delta Lake
? [AWS Lambda quotas]
? [AWS Database Migration Service quotas]

NEW QUESTION 6
A company is planning to upgrade its Amazon Elastic Block Store (Amazon EBS) General Purpose SSD storage from gp2 to gp3. The company wants to prevent any interruptions in its Amazon EC2 instances that will cause data loss during the migration to the upgraded storage.
Which solution will meet these requirements with the LEAST operational overhead?

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

Answer: C

Explanation:
Changing the volume type of the existing gp2 volumes to gp3 is the easiest and fastest way to migrate to the new storage type without any downtime or data loss. You can use the AWS Management Console, the AWS CLI, or the Amazon EC2 API to modify the volume type, size, IOPS, and throughput of your gp2 volumes. The modification takes effect immediately, and you can monitor the progress of the modification using CloudWatch. The other options are either more complex or require additional steps, such as creating snapshots, transferring data, or attaching new volumes, which can increase the operational overhead and the risk of errors. References:
? Migrating Amazon EBS volumes from gp2 to gp3 and save up to 20% on
costs (Section: How to migrate from gp2 to gp3)
? Switching from gp2 Volumes to gp3 Volumes to Lower AWS EBS Costs (Section: How to Switch from GP2 Volumes to GP3 Volumes)
? Modifying the volume type, IOPS, or size of an EBS volume - Amazon Elastic Compute Cloud (Section: Modifying the volume type)

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 thesame 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 to the permissions-related errors? (Choose two.)

A. There is no connection between QuickSgqht and Athena.
B. The Athena tables are not cataloged.
C. QuickSiqht does not have access to the S3 bucket.
D. QuickSight does not have access to decrypt S3 data.
E. There is no 1AM role assigned to QuickSiqht.

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 dat
B. Store the data in Amazon S3 for querying.
C. Use AWS Lambda to process the sensor dat
D. Store the data in Amazon S3 for querying.
E. Use Amazon Kinesis Data Streams to capture the sensor dat
F. Store the data in Amazon DynamoDB for querying.
G. Use Amazon Simple Queue Service(Amazon SQS) to buffer incomingsensor dat
H. Use AWS Glue to store thedata 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 capturechanges 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 uses Amazon Athena to run SQL queries for extract, transform, and load (ETL) tasks by using Create Table As Select (CTAS). The company must use Apache Spark instead of SQL to generate analytics.
Which solution will give the company the ability to use Spark to access Athena?

A. Athena query settings
B. Athena workgroup
C. Athena data source
D. Athena query editor

Answer: C

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

? Athena query settings
? [Athena workgroups]
? [Athena query editor]

NEW QUESTION 10
During a security review, a company identified a vulnerability in an AWS Glue job. The company discovered that credentials to access an Amazon Redshift cluster were hard coded in the job script.
A data engineer must remediate the security vulnerability in the AWS Glue job. The solution must securely store the credentials.
Which combination of steps should the data engineer take to meet these requirements? (Choose two.)

A. Store the credentials in the AWS Glue job parameters.
B. Store the credentials in a configuration file that is in an Amazon S3 bucket.
C. Access the credentials from a configuration file that is in an Amazon S3 bucket by using the AWS Glue job.
D. Store the credentials in AWS Secrets Manager.
E. Grant the AWS Glue job 1AM role access to the stored credentials.

Answer: DE

Explanation:
AWS Secrets Manager is a service that allows you to securely store and manage secrets, such as database credentials, API keys, passwords, etc. You can use Secrets Manager to encrypt, rotate, and audit your secrets, as well as to control access to them using fine-grained policies. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue 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). Storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials will meet the requirements, as it will remediate the security vulnerability in the AWS Glue job and securely store the credentials. By using AWS Secrets Manager, you can avoid hard coding the credentials in the job script, which is a bad practice that exposes the credentials to unauthorized access or leakage. Instead, you can store the credentials as a secret in Secrets Manager and reference the secret name or ARN in the job script. You can also use Secrets Manager to encrypt thecredentials using AWS Key Management Service (AWS KMS), rotate the credentials automatically or on demand, and monitor the access to the credentials using AWS CloudTrail. By granting the AWS Glue job 1AM role access to the stored credentials, you can use the principle of least privilege to ensure that only the AWS Glue job can retrieve the credentials from Secrets Manager. You can also use resource-based or tag-based policies to further restrict the access to the credentials.
The other options are not as secure as storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials. Storing the credentials in the AWS Glue job parameters will not remediate the security vulnerability, as the job parameters are still visible in the AWS Glue console and API. Storing the credentials in a configuration file that is in an Amazon S3 bucket and accessing the credentials from the configuration file by using the AWS Glue job will not be as secure as using Secrets Manager, as the configuration file may not be encrypted or rotated, and the access to the file may not be audited or controlled. References:
? AWS Secrets Manager
? AWS Glue
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

NEW QUESTION 15
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 datase
B. Create an AWS Lambda transform function to identify the PI
C. Use an AWS SDK to obfuscate the PI
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 PI
F. Obfuscate the PI
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 PI
I. Create a rule in AWS Glue Data Quality to obfuscate the PI
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 DynamoD
L. Create an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the dat
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 identifyand 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 uses Amazon RDS for MySQL as the database for a critical application. The database workload is mostly writes, with a small number of reads.
A data engineer notices that the CPU utilization of the DB instance is very high. The high CPU utilization is slowing down the application. The data engineer must reduce the CPU utilization of the DB Instance.
Which actions should the data engineer take to meet this requirement? (Choose two.)

A. Use the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilizatio
B. Optimize the problematic queries.
C. Modify the database schema to include additional tables and indexes.
D. Reboot the RDS DB instance once each week.
E. Upgrade to a larger instance size.
F. Implement caching to reduce the database query load.

Answer: AE

Explanation:
Amazon RDS is a fully managed service that provides relational databases in the cloud. Amazon RDS for MySQL is one of the supported database engines that you can use to run your applications. Amazon RDS provides various features and tools to monitor and optimize the performance of your DB instances, such as Performance Insights, Enhanced Monitoring, CloudWatch metrics and alarms, etc.
Using the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization and optimizing the problematic queries will help reduce the CPU utilization of the DB instance. Performance Insights is a feature that allows you to analyze the load on your DB instance and determine what is causing performance issues. Performance Insights collects, analyzes, and displays database performance data using an interactive dashboard. You can use Performance Insights to identify the top SQL statements, hosts, users, or processes that are consuming the most CPU resources. You can also drill down into the details of each query and see the execution plan, wait events, locks, etc. By using Performance Insights, you can pinpoint the root cause of the high CPU utilization and optimize the queries accordingly. For example, you can rewrite the queries to make them more efficient, add or remove indexes, use prepared statements, etc.
Implementing caching to reduce the database query load will also help reduce the CPU utilization of the DB instance. Caching is a technique that allows you to store frequently accessed data in a fast and scalable storage layer, such as Amazon ElastiCache. By using caching, you can reduce the number of requests that hit your database, which in turn reduces the CPU load on your DB instance. Caching also improves the performance and availability of your application, as it reduces the latency and increases the throughput of your data access. You can use caching for various scenarios, such as storing session data, user preferences, application configuration, etc. You can also use caching for read-heavy workloads, such as displaying product details, recommendations, reviews, etc.
The other options are not as effective as using Performance Insights and caching. Modifying the database schema to include additional tables and indexes may or may not improve the CPU utilization, depending on the nature of the workload and the queries. Adding more tables and indexes may increase the complexity and overhead of the database, which may negatively affect the performance. Rebooting the RDS DB instance once each week will not reduce the CPU utilization, as it will not address the underlying cause of the high CPU load. Rebooting may also cause downtime and disruption to your application. Upgrading to a larger instance size may reduce the CPUutilization, but it will also increase the cost and complexity of your solution. Upgrading may also not be necessary if you can optimize the queries and reduce the database load by using caching. References:
? Amazon RDS
? Performance Insights
? Amazon ElastiCache
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 3: Data Storage and Management, Section 3.1: Amazon RDS

NEW QUESTION 21
A company uses Amazon Redshift for its data warehouse. The company must automate refresh schedules for Amazon Redshift materialized views.
Which solution will meet this requirement with the LEAST effort?

A. Use Apache Airflow to refresh the materialized views.
B. Use an AWS Lambda user-defined function (UDF) within Amazon Redshift to refresh the materialized views.
C. Use the query editor v2 in Amazon Redshift to refresh the materialized views.
D. Use an AWS Glue workflow to refresh the materialized views.

Answer: C

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

NEW QUESTION 23
A financial company wants to implement a data mesh. The data mesh must support centralized data governance, data analysis, and data access control. The company has decided to use AWS Glue for data catalogs and extract, transform, and load (ETL) operations.

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

A. Use Amazon Aurora for data storag
B. Use an Amazon Redshift provisioned cluster for data analysis.
C. Use Amazon S3 for data storag
D. Use Amazon Athena for data analysis.
E. Use AWS Glue DataBrewfor centralized data governance and access control.
F. Use Amazon RDS for data storag
G. Use Amazon EMR for data analysis.
H. Use AWS Lake Formation for centralized data governance and access control.

Answer: BE

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

NEW QUESTION 28
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 30
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 EventBridq
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) queu
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 35
A company receives a daily file that contains customer data in .xls format. The company stores the file in Amazon S3. The daily file is approximately 2 GB in size.
A data engineer concatenates the column in the file that contains customer first names and the column that contains customer last names. The data engineer needs to determine the number of distinct customers in the file.
Which solution will meet this requirement with the LEAST operational effort?

A. Create and run an Apache Spark job in an AWS Glue noteboo
B. Configure the job to read the S3 file and calculate the number of distinct customers.
C. Create an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 fil
D. Run SQL queries from Amazon Athena to calculate the number of distinct customers.
E. Create and run an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.
F. Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers.

Answer: D

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

NEW QUESTION 36
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 rewritesthe table to a new location on disk. An interleaved sort key is a type of sort key that gives equal weight to each column in the sort key, and stores the rows in a way that optimizes the performance of queries that filter by multiple columns in the sort key. However, as data is added or changed, the interleaved sort order can become skewed, resulting in suboptimal query performance. The VACUUM REINDEX option restores the optimal interleaved sort order and reclaims disk space by removing deleted rows. This option also analyzes the sort key column and updates the table statistics, which are used by the query optimizer to generate the most efficient query execution plan23.
The other options are not optimal for the following reasons:
? A. VACUUM FULL Orders. This option reclaims disk space by removing deleted rows and resorts the entire table. However, this option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved sort order. Moreover, this option is the most resource-intensive and time-consuming, as it rewrites the entire table to a new location on disk.
? B. VACUUM DELETE ONLY Orders. This option reclaims disk space by removing deleted rows, but does not resort the table. This option is not suitable for tables that have any sort key, as it does not improve the query performance by restoring the sort order. Moreover, this option does not analyze the sort key column and update the table statistics.
? D. VACUUM SORT ONLY Orders. This option resorts the entire table, but does not reclaim disk space by removing deleted rows. This option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved sort order. Moreover, this option does not analyze the sort key column and update the table statistics.
References:
? 1: Amazon Redshift VACUUM
? 2: Amazon Redshift Interleaved Sorting
? 3: Amazon Redshift ANALYZE

NEW QUESTION 38
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, suchas 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 40

A company is migrating on-premises workloads to AWS. The company wants to reduce overall operational overhead. The company also wants to explore serverless options.
The company's current workloads use Apache Pig, Apache Oozie, Apache Spark, Apache Hbase, and Apache Flink. The on-premises workloads process petabytes of data in seconds. The company must maintain similar or better performance after the migration to AWS.
Which extract, transform, and load (ETL) service will meet these requirements?

A. AWS Glue
B. Amazon EMR
C. AWS Lambda
D. Amazon Redshift

Answer: A

Explanation:
AWS Glue is a fully managed serverless ETL service that can handle petabytes of data in seconds. AWS Glue can run Apache Spark and Apache Flink jobs without requiring any infrastructure provisioning or management. AWS Glue can also integrate with Apache Pig, Apache Oozie, and Apache Hbase using AWS Glue Data Catalog and AWS Glue workflows. AWS Glue can reduce the overall operational overhead by automating the data discovery, data preparation, and data loading processes. AWS Glue can also optimize the cost and performance of ETL jobs by using AWS Glue Job Bookmarking, AWS Glue Crawlers, and AWS Glue Schema Registry. References:
? AWS Glue
? AWS Glue Data Catalog
? AWS Glue Workflows
? [AWS Glue Job Bookmarking]
? [AWS Glue Crawlers]
? [AWS Glue Schema Registry]
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 45
A data engineer must build an extract, transform, and load (ETL) pipeline to process and load data from 10 source systems into 10 tables that are in an Amazon Redshift database. All the source systems generate .csv, JSON, or Apache Parquet files every 15 minutes. The source systems all deliver files into one Amazon S3 bucket. The file sizes range from 10 MB to 20 GB. The ETL pipeline must function correctly despite changes to the data schema.
Which data pipeline solutions will meet these requirements? (Choose two.)

A. Use an Amazon EventBridge rule to run an AWS Glue job every 15 minute
B. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
C. Use an Amazon EventBridge rule to invoke an AWS Glue workflow job every 15 minute
D. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfull
E. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
F. Configure an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucke
G. Configure an AWS Glue job to process and load the data into the Amazon Redshift table
H. Create a second Lambda function to run the AWS Glue jo
I. Create an Amazon EventBridge rule to invoke the second Lambda function when the AWS Glue crawler finishes running successfully.
J. Configure an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucke
K. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfull
L. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
M. Configure an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucke
N. Configure the AWS Glue job to read the files from the S3 bucket into an Apache Spark DataFram
O. Configure the AWS Glue job to also put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery strea
P. Configure the delivery stream to load data into the Amazon Redshift tables.

Answer: AB

Explanation:
Using an Amazon EventBridge rule to run an AWS Glue job or invoke an AWS Glue workflow job every 15 minutes are two possible solutions that will meet the requirements. AWS Glue is a serverless ETL service that can process and load data from various sources to various targets, including Amazon Redshift. AWS Glue can handle different data formats, such as CSV, JSON, and Parquet, and also support schema evolution, meaning it can adapt to changes in the data schema over time. AWS Glue can also leverage Apache Spark to perform distributed processing and transformation of large datasets. AWS Glue integrates with Amazon EventBridge, which is a serverless event bus service that can trigger actions based on rules and schedules. By using an Amazon EventBridge rule, you can invoke an AWS Glue job or workflow every 15 minutes, and configure the job or workflow to run an AWS Glue crawler and then load the data into the Amazon Redshift tables. This way, you can build a cost-effective and scalable ETL pipeline that can handle data from 10 source systems and function correctly despite changes to the data schema.
The other options are not solutions that will meet the requirements. Option C, configuring an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket, and creating a second Lambda function to run the AWS Glue job, is not a feasible solution, as it would require a lot of Lambda invocations andcoordination. AWS Lambda has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the ETL pipeline. Option D, configuring an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket, is not a necessary solution, as you can use an Amazon EventBridge rule to invoke the AWS Glue workflow directly, without the need for a Lambda function. Option E, configuring an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket, and configuring the AWS Glue job to put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream, is not a cost-effective solution, as it would incur additional costs for Lambda invocations and data delivery. Moreover, using Amazon Kinesis Data Firehose to load data into Amazon Redshift is not suitable for frequent and small batches of data, as it can cause performance issues and data fragmentation. References:
? AWS Glue
? Amazon EventBridge
? Using AWS Glue to run ETL jobs against non-native JDBC data sources
? [AWS Lambda quotas]
? [Amazon Kinesis Data Firehose quotas]

NEW QUESTION 49
A financial company wants to use Amazon Athena to run on-demand SQL queries on a petabyte-scale dataset to support a business intelligence (BI) application. An AWS Glue job that runs during non-business hours updates the dataset once every day. The BI application has a standard data refresh frequency of 1 hour to

comply with company policies.
A data engineer wants to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs.
Which solution will meet these requirements with the LEAST operational overhead?

A. Configure an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day
B. Use the query result reuse feature of Amazon Athena for the SQL queries.
C. Add an Amazon ElastiCache cluster between the BI application and Athena.
D. Change the format of the files that are in the dataset to Apache Parquet.

Answer: B

Explanation:
The best solution to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs is to use the query result reuse feature of AmazonAthena for the SQL queries. This feature allows you to run the same query multiple times without incurring additional charges, as long as the underlying data has not changed and the query results are still in the query result location in Amazon S31. This feature is useful for scenarios where you have a petabyte-scale dataset that is updated infrequently, such as once a day, and you have a BI application that runs the same queries repeatedly, such as every hour. By using the query result reuse feature, you can reduce the amount of data scanned by your queries and save on the cost of running Athena. You can enable or disable this feature at the workgroup level or at the individual query level1.
Option A is not the best solution, as configuring an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon S3 Lifecycle policies are rules that you can define to automatically transition objects between different storage classes based on specified criteria, such as the age of the object2. S3 Glacier Deep Archive is the lowest-cost storage class in Amazon S3, designed
for long-term data archiving that is accessed once or twice in a year3. While moving data to S3 Glacier Deep Archive can reduce the storage cost, it would also increase the retrieval cost and latency, as it takes up to 12 hours to restore the data from S3 Glacier Deep Archive3. Moreover, Athena does not support querying data that is in S3 Glacier or S3 Glacier Deep Archive storage classes4. Therefore, using this option would not meet the requirements of running on-demand SQL queries on the dataset.
Option C is not the best solution, as adding an Amazon ElastiCache cluster between the BI application and Athena would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon ElastiCache is a service that offers fully managed in-memory data stores, such as Redis and Memcached, that can improve the performance and scalability of web applications by caching frequently accessed data. While using ElastiCache can reduce the latency and load on the BI application, it would not reduce the amount of data scanned by Athena, which is the main factor that determines the cost of running Athena. Moreover, using ElastiCache would introduce additional infrastructure costs and operational overhead, as you would have to provision, manage, and scale the ElastiCache cluster, and integrate it with the BI application and Athena. Option D is not the best solution, as changing the format of the files that are in the dataset to Apache Parquet would not cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs, but rather increase the complexity. 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 schemes. However,changing the format of the files that are in the dataset to Apache Parquet would require additional processing and transformation steps, such as using AWS Glue or Amazon EMR to convert the files from their original format to Parquet, and storing the converted files in a separate location in Amazon S3. This would increase the complexity and the operational overhead of the data pipeline, and also incur additional costs for using AWS Glue or Amazon EMR. References:
? Query result reuse
? Amazon S3 Lifecycle
? S3 Glacier Deep Archive
? Storage classes supported by Athena
? [What is Amazon ElastiCache?]
? [Amazon Athena pricing]
? [Columnar Storage Formats]
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 52
A data engineer uses Amazon Redshift to run resource-intensive analytics processes once every month. Every month, the data engineer creates a new Redshift provisioned cluster. The data engineer deletes the Redshift provisioned cluster after the analytics processes are complete every month. Before the data engineer deletes the cluster each month, the data engineer unloads backup data from the cluster to an Amazon S3 bucket.
The data engineer needs a solution to run the monthly analytics processes that does not require the data engineer to manage the infrastructure manually.
Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month.
B. Use Amazon Redshift Serverless to automatically process the analytics workload.
C. Use the AWS CLI to automatically process the analytics workload.
D. Use AWS CloudFormation templates to automatically process the analytics workload.

Answer: B

Explanation:
Amazon Redshift Serverless is a new feature of Amazon Redshift that enables you to run SQL queries on data in Amazon S3 without provisioning or managing any clusters. You can use Amazon Redshift Serverless to automatically process the analytics workload, as it scales up and down the compute resources based on the query demand, and charges you only for the resources consumed. This solution will meet the requirements with the least operational overhead, as it does not require the data engineer to create, delete, pause, or resume any Redshift clusters, or to manage any infrastructure manually. You can use the Amazon Redshift Data API to run queries from the AWS CLI, AWS SDK, or AWS Lambda functions12.
The other options are not optimal for the following reasons:
? A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month. This option is not recommended, as it would still require the data engineer to create and delete a new Redshift provisioned cluster every month, which can incur additional costs and time. Moreover, this option would require the data engineer to use Amazon Step Functions to orchestrate the workflow of pausing and resuming the cluster, which can add complexity and overhead.
? C. Use the AWS CLI to automatically process the analytics workload. This option is vague and does not specify how the AWS CLI is used to process the analytics workload. The AWS CLI can be used to run queries on data in Amazon S3 using Amazon Redshift Serverless, Amazon Athena, or Amazon EMR, but each of these services has different features and benefits. Moreover, this option does not address the requirement of not managing the infrastructure manually, as the data engineer may still need to provision and configure some resources, such as Amazon EMR clusters or Amazon Athena workgroups.
? D. Use AWS CloudFormation templates to automatically process the analytics workload. This option is also vague and does not specify how AWS CloudFormation templates are used to process the analytics workload. AWS CloudFormation is a service that lets you model and provision AWS resources using templates. You can use AWS CloudFormation templates to create and delete a Redshift provisioned cluster every month, or to create and configure other AWS resources, such as Amazon EMR, Amazon Athena, or Amazon Redshift Serverless. However, this option does not address the requirement of not managing the infrastructure manually, as the data engineer may still need to write and maintain the AWS CloudFormation templates, and to monitor the status and performance

of the resources.
References:
? 1: Amazon Redshift Serverless
? 2: Amazon Redshift Data API
? : Amazon Step Functions
? : AWS CLI
? : AWS CloudFormation

NEW QUESTION 55
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 sourc
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 sourc
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 60
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 63
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 ofcolumns. 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 67
A company has multiple applications that use datasets that are stored in an Amazon S3 bucket. The company has an ecommerce application that generates a dataset that contains personally identifiable information (PII). The company has an internal analytics application that does not require access to the PII.
To comply with regulations, the company must not share PII unnecessarily. A data engineer needs to implement a solution that with redact PII dynamically, based on the needs of each application that accesses the dataset.
Which solution will meet the requirements with the LEAST operational overhead?

A. Create an S3 bucket policy to limit the access each application ha
B. Create multiple copies of the datase
C. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.
D. Create an S3 Object Lambda endpoin
E. Use the S3 Object Lambda endpoint to read data from the S3 bucke
F. Implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data.
G. Use AWS Glue to transform the data for each applicatio
H. Create multiple copies of the datase
I. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.
J. Create an API Gateway endpoint that has custom authorizer
K. Use the API Gateway endpoint to read data from the S3 bucke
L. Initiate a REST API call to dynamically redact PII based on the needs of each application that accesses the data.

Answer: B

Explanation:
Option B is the best solution to meet the requirements with the least operational overhead because S3 Object Lambda is a feature that allows you to add your own code to process data retrieved from S3 before returning it to an application. S3 Object Lambda works with S3 GET requests and can modify both the object metadata and the object data. By using S3 Object Lambda, you can implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data. This way, you can avoid creating and maintaining multiple copies of the dataset with different levels of redaction.
Option A is not a good solution because it involves creating and managing multiple copies of the dataset with different levels of redaction for each application. This option adds complexity and storage cost to the data protection process and requires additional resources and configuration. Moreover, S3 bucket policies cannot enforce fine-grained data access control at the row and column level, so they are not sufficient to redact PII.
Option C is not a good solution because it involves using AWS Glue to transform the data for each application. AWS Glue is a fully managed service that can extract, transform, and load (ETL) data from various sources to various destinations, including S3. AWS Glue can also convert data to different formats, such as Parquet, which is a columnar storage format that is optimized for analytics. However, in this scenario, using AWS Glue to redact PII is not the best option because it requires creating and maintaining multiple copies of the dataset with different levels of redaction for each application. This option also adds extra time and cost to the data protection process and requires additional resources and configuration.
Option D is not a good solution because it involves creating and configuring an API Gateway endpoint that has custom authorizers. API Gateway is a service that allows youto create, publish, maintain, monitor, and secure APIs at any scale. API Gateway can also integrate with other AWS services, such as Lambda, to provide custom logic for processing requests. However, in this scenario, using API Gateway to redact PII is not the best option because it requires writing and maintaining custom code and configuration for the API endpoint, the custom authorizers, and the REST API call. This option also adds complexity and latency to the data protection process and requires additional resources and configuration.
References:
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
? Introducing Amazon S3 Object Lambda – Use Your Code to Process Data as It Is Being Retrieved from S3
? Using Bucket Policies and User Policies - Amazon Simple Storage Service
? AWS Glue Documentation
? What is Amazon API Gateway? - Amazon API Gateway

NEW QUESTION 71
A data engineer runs Amazon Athena queries on data that is in an Amazon S3 bucket. The Athena queries use AWS Glue Data Catalog as a metadata table.
The data engineer notices that the Athena query plans are experiencing a performance bottleneck. The data engineer determines that the cause of the performance bottleneck is the large number of partitions that are in the S3 bucket. The data engineer must resolve the performance bottleneck and reduce Athena query planning time.
Which solutions will meet these requirements? (Choose two.)

A. Create an AWS Glue partition inde
B. Enable partition filtering.
C. Bucketthe data based on a column thatthe data have in common in a WHERE clause of the user query
D. Use Athena partition projection based on the S3 bucket prefix.
E. Transform the data that is in the S3 bucket to Apache Parquet format.
F. Use the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects.

Answer: AC

Explanation:
The best solutions to resolve the performance bottleneck and reduce Athena query planning time are to create an AWS Glue partition index and enable partition filtering, and to use Athena partition projection based on the S3 bucket prefix.
AWS Glue partition indexes are a feature that allows you to speed up query processing of highly partitioned tables cataloged in AWS Glue Data Catalog. Partition indexes are available for queries in Amazon EMR, Amazon Redshift Spectrum, and AWS Glue ETL jobs. Partition indexes are sublists of partition keys defined in the table. When you create a partition index, you specify a list of partition keys that already exist on a given table. AWS Glue then creates an index for the specified keys and stores it in the Data Catalog. When you run a query that filters on the partition keys, AWS Glue uses the partition index to quickly identify the relevant partitions without scanning the entiretable metadata. This reduces the query planning time and improves the query performance1.
Athena partition projection is a feature that allows you to speed up query processing of highly partitioned tables and automate partition management. In partition projection, Athena calculates partition values and locations using the table properties that you configure directly on your table in AWS Glue. The table properties allow Athena to 'project', or determine, the necessary partition information instead of having to do a more time- consuming metadata lookup in the AWS Glue Data Catalog. Because in-memory operations are often faster than remote operations, partition projection can reduce the runtime of queries against highly partitioned tables. Partition projection also automates partition management because it removes the need to manually create partitions in Athena, AWS Glue, or your external Hive metastore2.
Option B is not the best solution, as bucketing the data based on a column that the data have in common in a WHERE clause of the user query would not reduce the query planning time. Bucketing is a technique that divides data into buckets based on a hash function applied to a column. Bucketing can improve the performance of join queries by
reducing the amount of data that needs to be shuffled between nodes. However, bucketing does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario3.
Option D is not the best solution, as transforming the data that is in the S3 bucket to Apache Parquet format would not reduce the query planning time. 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 schemes. However, Parquet does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario4.
Option E is not the best solution, as using the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects would not reduce the query planning time. S3DistCP is a tool that can copy large amounts of data between Amazon S3 buckets or from HDFS to Amazon S3. S3DistCP can also aggregate smaller files into larger files to improve the performance of sequential access. However, S3DistCP does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario5. References:
? Improve query performance using AWS Glue partition indexes
? Partition projection with Amazon Athena
? Bucketing vs Partitioning
? Columnar Storage Formats
? S3DistCp
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 74
A company receives call logs as Amazon S3 objects that contain sensitive customer information. The company must protect the S3 objects by using encryption.
The company must also use encryption keys that only specific employees can access.
Which solution will meet these requirements with the LEAST effort?

A. Use an AWS CloudHSM cluster to store the encryption key
B. Configure the process that writes to Amazon S3 to make calls to CloudHSM to encrypt and decrypt the object
C. Deploy an IAM policy that restricts access to the CloudHSM cluster.
D. Use server-side encryption with customer-provided keys (SSE-C) to encrypt the objects that contain customer informatio
E. Restrict access to the keys that encrypt the objects.
F. Use server-side encryption with AWS KMS keys (SSE-KMS) to encrypt the objects that contain customer informatio
G. Configure an IAM policy that restricts access to the KMS keys that encrypt the objects.
H. Use server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt the objects that contain customer informatio
I. Configure an IAM policy that restricts access to the Amazon S3 managed keys that encrypt the objects.

Answer: C

Explanation:
Option C is the best solution to meet the requirements with the least effort because server-side encryption with AWS KMS keys (SSE-KMS) is a feature that allows you to encrypt data at rest in Amazon S3 using keys managed by AWS Key Management Service (AWS KMS). AWS KMS is a fully managed service that enables you to create and manage encryption keys for your AWS services and applications. AWS KMS also allows you to define granular access policies for your keys, such as who can use them to encrypt and decrypt data, and under what conditions. By using SSE-KMS, you canprotect your S3 objects by using encryption keys that only specific employees can access, without having to manage the encryption and decryption process yourself.
Option A is not a good solution because it involves using AWS CloudHSM, which is a service that provides hardware security modules (HSMs) in the AWS Cloud. AWS CloudHSM allows you to generate and use your own encryption keys on dedicated hardware that is compliant with various standards and regulations. However, AWS CloudHSM is not a fully managed service and requires more effort to set up and maintain than AWS KMS. Moreover, AWS CloudHSM does not integrate with Amazon S3, so you have to configure the process that writes to S3 to make calls to CloudHSM to encrypt and decrypt the objects, which adds complexity and latency to the data protection process. Option B is not a good solution because it involves using server-side encryption with customer-provided keys (SSE-C), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that you provide and manage yourself. SSE-C requires you to send your encryption key along with each request to upload or retrieve an object. However, SSE-C does not provide any mechanism to restrict access to the keys that encrypt the objects, so you have to implement your own key management and access control system, which adds more effort and risk to the data protection process.
Option D is not a good solution because it involves using server-side encryption with Amazon S3 managed keys (SSE-S3), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that are managed by Amazon S3. SSE-S3 automatically encrypts and decrypts your objects as they are uploaded and downloaded from S3. However, SSE-S3 does not allow you to control who can access the encryption keys or under what conditions. SSE-S3 uses a single encryption key for each S3 bucket, which is shared by all users who have access to the bucket. This means that you cannot restrict access to the keys that encrypt the objects by specific employees, which does not meet the requirements.
References:
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
? Protecting Data Using Server-Side Encryption with AWS KMS–Managed Encryption Keys (SSE-KMS) - Amazon Simple Storage Service
? What is AWS Key Management Service? - AWS Key Management Service
? What is AWS CloudHSM? - AWS CloudHSM
? Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C) - Amazon Simple Storage Service
? Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) - Amazon Simple Storage Service

NEW QUESTION 77
.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questons and Answers in PDF Format

AWS-Certified-Data-Engineer-Associate Practice Exam Features:

* AWS-Certified-Data-Engineer-Associate Questions and Answers Updated Frequently

* AWS-Certified-Data-Engineer-Associate Practice Questions Verified by Expert Senior Certified Staff

* AWS-Certified-Data-Engineer-Associate Most Realistic Questions that Guarantee you a Pass on Your FirstTry

* AWS-Certified-Data-Engineer-Associate Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year