Exam Questions AWS-Certified-Data-Engineer-Associate

AWS Certified Data Engineer - Associate (DEA-C01)

https://www.2passeasy.com/dumps/AWS-Certified-Data-Engineer-Associate/

NEW QUESTION 1
A company uses Amazon Athena for one-time queries against data that is in Amazon S3. The company has several use cases. The company must implement permission controls to separate query processes and access to query history among users, teams, and applications that are in the same AWS account.
Which solution will meet these requirements?

A. Create an S3 bucket for each use cas
B. Create an S3 bucket policy that grants permissions to appropriate individual IAM user
C. Apply the S3 bucket policy to the S3 bucket.
D. Create an Athena workgroup for each use cas
E. Apply tags to the workgrou
F. Create an 1AM policy that uses the tags to apply appropriate permissions to the workgroup.
G. Create an JAM role for each use cas
H. Assign appropriate permissions to the role for each use cas
I. Associate the role with Athena.
J. Create an AWS Glue Data Catalog resource policy that grants permissions to appropriate individual IAM users for each use cas
K. Apply the resource policy to the specific tables that Athena uses.

Answer: B

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

NEW QUESTION 2
A media company uses software as a service (SaaS) applications to gather data by using third-party tools. The company needs to store the data in an Amazon S3 bucket. The company will use Amazon Redshift to perform analytics based on the data.
Which AWS service or feature will meet these requirements with the LEAST operational overhead?

A. Amazon Managed Streaming for Apache Kafka (Amazon MSK)
B. Amazon AppFlow
C. AWS Glue Data Catalog
D. Amazon Kinesis

Answer: B

Explanation:
Amazon AppFlow is a fully managed integration service that enables you to securely transfer data between SaaS applications and AWS services like Amazon S3 and AmazonRedshift. Amazon AppFlow supports many SaaS applications as data sources and targets, and allows you to configure data flows with a few clicks. Amazon AppFlow also provides features such as data transformation, filtering, validation, and encryption to prepare and protect your data. Amazon AppFlow meets the requirements of the media company with the least operational overhead, as it eliminates the need to write code, manage infrastructure, or monitor data pipelines. References:
? Amazon AppFlow
? Amazon AppFlow | SaaS Integrations List
? Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions

NEW QUESTION 3
A company maintains an Amazon Redshift provisioned cluster that the company uses for extract, transform, and load (ETL) operations to support critical analysis tasks. A sales team within the company maintains a Redshift cluster that the sales team uses for business intelligence (BI) tasks.
The sales team recently requested access to the data that is in the ETL Redshift cluster so the team can perform weekly summary analysis tasks. The sales team needs to join data from the ETL cluster with data that is in the sales team's BI cluster.
The company needs a solution that will share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution must minimize usage of the computing resources of the ETL cluster.
Which solution will meet these requirements?

A. Set up the sales team Bl cluster asa consumer of the ETL cluster by using Redshift data sharing.
B. Create materialized views based on the sales team's requirement
C. Grant the sales team direct access to the ETL cluster.
D. Create database views based on the sales team's requirement
E. Grant the sales team direct access to the ETL cluster.
F. Unload a copy of the data from the ETL cluster to an Amazon S3 bucket every wee
G. Create an Amazon Redshift Spectrum table based on the content of the ETL cluster.

Answer: A

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

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

NEW QUESTION 4
A company uses AWS Step Functions to orchestrate a data pipeline. The pipeline consists of Amazon EMR jobs that ingest data from data sources and store the data in an Amazon S3 bucket. The pipeline also includes EMR jobs that load the data to Amazon Redshift.
The company's cloud infrastructure team manually built a Step Functions state machine. The cloud infrastructure team launched an EMR cluster into a VPC to support the EMR jobs. However, the deployed Step Functions state machine is not able to run the EMR jobs.
Which combination of steps should the company take to identify the reason the Step Functions state machine is not able to run the EMR jobs? (Choose two.)

A. Use AWS CloudFormation to automate the Step Functions state machine deploymen
B. Create a step to pause the state machine during the EMR jobs that fai
C. Configure the step to wait for a human user to send approval through an email messag
D. Include details of the EMR task in the email message for further analysis.
E. Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR job
F. Verify that the Step Functions state machine code also includes IAM permissions to access the Amazon S3 buckets that the EMR jobs us
G. Use Access Analyzer for S3 to check the S3 access properties.
H. Check for entries in Amazon CloudWatch for the newly created EMR cluste
I. Change the AWS Step Functions state machine code to use Amazon EMR on EK
J. Change the IAM access policies and the security group configuration for the Step Functions state machine code to reflect inclusion of Amazon Elastic Kubernetes Service (Amazon EKS).
K. Query the flow logs for the VP
L. Determine whether the traffic that originates from the EMR cluster can successfully reach the data provider
M. Determine whether any security group that might be attached to the Amazon EMR cluster allows connections to the data source servers on the informed ports.
N. Check the retry scenarios that the company configured for the EMR job
O. Increase the number of seconds in the interval between each EMR tas
P. Validate that each fallback state has the appropriate catch for each decision stat
Q. Configure an Amazon Simple Notification Service (Amazon SNS) topic to store the error messages.

Answer: BD

Explanation:
To identify the reason why the Step Functions state machine is not able to run the EMR jobs, the company should take the following steps:
? Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR jobs. The state machine code should have an IAM role that allows it to invoke the EMR APIs, such as RunJobFlow, AddJobFlowSteps, and DescribeStep. The state machine code should also have IAM permissions to access the Amazon S3 buckets that the EMR jobs use as input and output locations. The company can use Access Analyzer for S3 to check the access policies and permissions of the S3 buckets12. Therefore, option B is correct.
? Query the flow logs for the VPC. The flow logs can provide information about the network traffic to and from the EMR cluster that is launched in the VPC. The company can use the flow logs to determine whether the traffic that originates from the EMR cluster can successfully reach the data providers, such as Amazon RDS, Amazon Redshift, or other external sources. The company can also determine whether any security group that might be attached to the EMR cluster allows connections to the data source servers on the informed ports. The company can use Amazon VPC Flow Logs or Amazon CloudWatch Logs Insights to query the flow logs3 . Therefore, option D is correct.
Option A is incorrect because it suggests using AWS CloudFormation to automate the Step Functions state machine deployment. While this is a good practice to ensure consistency and repeatability of the deployment, it does not help to identify the reasonwhy the state machine is not able to run the EMR jobs. Moreover, creating a step to pause the state machine during the EMR jobs that fail and wait for a human user to send approval through an email message is not a reliable way to troubleshoot the issue. The company should use the Step Functions console or API to monitor the execution history and status of the state machine, and use Amazon CloudWatch to view the logs and metrics of the EMR jobs . Option C is incorrect because it suggests changing the AWS Step Functions state machine code to use Amazon EMR on EKS. Amazon EMR on EKS is a service that allows you to run EMR jobs on Amazon Elastic Kubernetes Service (Amazon EKS) clusters. While this service has some benefits, such as lower cost and faster execution time, it does not support all the features and integrations that EMR on EC2 does, such as EMR Notebooks, EMR Studio, and EMRFS. Therefore, changing the state machine code to use EMR on EKS may not be compatible with the existing data pipeline and may introduce new issues. Option E is incorrect because it suggests checking the retry scenarios that the company configured for the EMR jobs. While this is a good practice to handle transient failures and errors, it does not help to identify the root cause of why the state machine is not able to run the EMR jobs. Moreover, increasing the number of seconds in the interval between each EMR task may not improve the success rate of the jobs, and may increase the execution time and cost of the state machine. Configuring an Amazon SNS topic to store the error messages may help to notify the company of any failures, but it does not provide enough information to troubleshoot the issue.
References:
? 1: Manage an Amazon EMR Job - AWS Step Functions
? 2: Access Analyzer for S3 - Amazon Simple Storage Service
? 3: Working with Amazon EMR and VPC Flow Logs - Amazon EMR
? [4]: Analyzing VPC Flow Logs with Amazon CloudWatch Logs Insights - Amazon Virtual Private Cloud
? [5]: Monitor AWS Step Functions - AWS Step Functions
? [6]: Monitor Amazon EMR clusters - Amazon EMR
? [7]: Amazon EMR on Amazon EKS - Amazon EMR

NEW QUESTION 5
A data engineer is configuring Amazon SageMaker Studio to use AWS Glue interactive sessions to prepare data for machine learning (ML) models.
The data engineer receives an access denied error when the data engineer tries to prepare the data by using SageMaker Studio.
Which change should the engineer make to gain access to SageMaker Studio?

A. Add the AWSGlueServiceRole managed policy to the data engineer's IAM user.
B. Add a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy.
C. Add the AmazonSageMakerFullAccess managed policy to the data engineer's IAM user.
D. Add a policy to the data engineer's IAM user that allows the sts:AddAssociation action for the AWS Glue and SageMaker service principals in the trust policy.

Answer: B

Explanation:
This solution meets the requirement of gaining access to SageMaker Studio to use AWS Glue interactive sessions. AWS Glue interactive sessions are a way to use AWS Glue DataBrew and AWS Glue Data Catalog from within SageMaker Studio. To use AWS Glue interactive sessions, the data engineer's IAM user needs to have permissions to assume the AWS Glue service role and the SageMaker execution role. By adding a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy, the data engineer can grant these permissions and avoid the access denied error. The other options are not sufficient or necessary to resolve the error. References:

? Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions
? Troubleshoot Errors - Amazon SageMaker
? AccessDeniedException on sagemaker:CreateDomain in AWS SageMaker Studio, despite having SageMakerFullAccess

NEW QUESTION 6
A data engineer needs to schedule a workflow that runs a set of AWS Glue jobs every day. The data engineer does not require the Glue jobs to run or finish at a specific time.
Which solution will run the Glue jobs in the MOST cost-effective way?

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

Answer: A

Explanation:
The FLEX execution class allows you to run AWS Glue jobs on spare compute capacity instead of dedicated hardware. This can reduce the cost of running non-urgent or non-time sensitive data integration workloads, such as testing and one-time data loads. The FLEX execution class is available for AWS Glue 3.0 Spark jobs. The other options are not as cost-effective as FLEX, because they either use dedicated resources (STANDARD) or do not affect the cost at all (Spot Instance type and GlueVersion). References:
? Introducing AWS Glue Flex jobs: Cost savings on ETL workloads
? Serverless Data Integration – AWS Glue Pricing
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 125)

NEW QUESTION 7
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 8
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 9
A company needs to set up a data catalog and metadata management for data sources that run in the AWS Cloud. The company will use the data catalog to maintain the metadata of all the objects that are in a set of data stores. The data stores include structured sources such as Amazon RDS and Amazon Redshift. The data stores also include semistructured sources such as JSON files and .xml files that are stored in Amazon S3.
The company needs a solution that will update the data catalog on a regular basis. The solution also must detect changes to the source metadata.
Which solution will meet these requirements with the LEAST operational overhead?

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

Answer: B

Explanation:
This solution will meet the requirements with the least operational overhead because it uses the AWS Glue Data Catalog as the central metadata repository for data sources that run in the AWS Cloud. The AWS Glue Data Catalog is a fully managed service that provides a unified view of your data assets across AWS and on-premises data sources. It stores the metadata of your data in tables, partitions, and columns, and enables you to access and query your data using various AWS services, such as Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can use AWS Glue crawlers to connect to multiple data stores, such as Amazon RDS, Amazon Redshift, and Amazon S3, and to update the Data Catalog with metadata changes. AWS Glue crawlers can automatically discover the schema and partition structure of your data, and create or update the corresponding tables in the Data Catalog. You can schedule the crawlers to run periodically to update the metadata catalog, and configure them to detect changes to the source metadata, such as new columns, tables, or partitions12.
The other options are not optimal for the following reasons:
? A. Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically. This option is not recommended, as it would require more operational overhead to create and manage an Amazon Aurora database as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? C. Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically. This option is also not recommended, as it would require more operational overhead to create and manage an Amazon DynamoDB table as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.
? D. Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog. This option is not optimal, as it would require more manual effort to extract the schema for Amazon RDS and Amazon Redshift sources, and to build the Data Catalog. This option would not take advantage of the AWS Glue crawlers' ability to automatically discover the schema and partition structure of your data from various data sources, and to create or update the corresponding tables in the Data Catalog.
References:
? 1: AWS Glue Data Catalog
? 2: AWS Glue Crawlers
? : Amazon Aurora
? : AWS Lambda
? : Amazon DynamoDB

NEW QUESTION 10
A company 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
A security company stores IoT data that is in JSON format in an Amazon S3 bucket. The data structure can change when the company upgrades the IoT devices. The company wants to create a data catalog that includes the IoT data. The company's analytics department will use the data catalog to index the data.
Which solution will meet these requirements MOST cost-effectively?

A. Create an AWS Glue Data Catalo
B. Configure an AWS Glue Schema Registr
C. Create a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.
D. Create an Amazon Redshift provisioned cluste
E. Create an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3. Create Redshift stored procedures to load the data into Amazon Redshift.
F. Create an Amazon Athena workgrou
G. Explore the data that is in Amazon S3 by using Apache Spark through Athen
H. Provide the Athena workgroup schema and tables to the analytics department.
I. Create an AWS Glue Data Catalo
J. Configure an AWS Glue Schema Registr
K. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data AP
L. Create an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

Answer: C

Explanation:
The best solution to meet the requirements of creating a data catalog that includes the IoT data, and allowing the analytics department to index the data, most cost- effectively, is to create an Amazon Athena workgroup, explore the data that is in Amazon S3 by using Apache Spark through Athena, and provide the Athena workgroup schema and tables to the analytics department.
Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python1. Amazon Athena also supports Apache Spark, an open-source distributed processing framework that can run large-scale data analytics applications across clusters of servers2. You can use Athena to run Spark code on data in Amazon S3 without having to set up, manage, or scale any infrastructure. You can also use Athena to create and manage external tables that point to your data in Amazon S3, and store them in an external data catalog, such as AWS Glue Data Catalog, Amazon Athena Data Catalog, or your own Apache Hive metastore3. You can create Athena workgroups to separate query execution and resource allocation based on different criteria, such as users, teams, or applications4. You can share the schemas and tables in your Athena workgroup with other users or applications, such as Amazon QuickSight, for data visualization and analysis5.
Using Athena and Spark to create a data catalog and explore the IoT data in Amazon S3 is the most cost-effective solution, as you pay only for the queries you run or the compute you use, and you pay nothing when the service is idle1. You also save on the operational overhead and complexity of managing data warehouse

infrastructure, as Athena and Spark are serverless and scalable. You can also benefit from the flexibility and performance of Athena and Spark, as they support various data formats, including JSON, and can handle schema changes and complex queries efficiently.
Option A is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating a new AWS Glue workload to orchestrate theingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Glue Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components6. AWS Glue Schema Registry is a service that allows you to centrally store and manage the schemas of your streaming data in AWS Glue Data Catalog7. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores8. Amazon Redshift Serverless is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to run and scale analytics without having to manage data warehouse infrastructure9. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Glue charges you based on the compute time and the data processed by your ETL jobs8. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue and Amazon Redshift Serverless would introduce additional latency and complexity, as you would have to ingest the data from Amazon S3 to Amazon Redshift Serverless, and then query it from there, instead of querying it directly from Amazon S3 using Athena and Spark.
Option B is not the best solution, as creating an Amazon Redshift provisioned cluster, creating an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3, and creating Redshift stored procedures to load the data into Amazon Redshift, would incur more costs and complexity than using Athena and Spark. Amazon Redshift provisioned clusters are clusters that you create and manage by specifying the number and type of nodes, and the amount of storage and compute capacity10. Amazon Redshift Spectrum is a feature of Amazon Redshift that allows you to query and join data across your data warehouse and your data lake using standard SQL11. Redshift stored procedures are SQL statements that you can define and store in Amazon Redshift, and then call them by using the CALL command12. While these features are powerful and useful for many data warehousing scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. Amazon Redshift provisioned clusters charge you based on the node type, the number of nodes, and the duration of the cluster10. Amazon Redshift Spectrum charges you based on the amount of data scanned by your queries11. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using Amazon Redshift provisioned clusters and Spectrum would introduce additional latency and complexity, as you would have to provision andmanage the cluster, create an external schema and database for the data in Amazon S3, and load the data into the cluster using stored procedures, instead of querying it directly from Amazon S3 using Athena and Spark. Option D is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API, and creating an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers13. AWS Lambda UDFs are Lambda functions that you can invoke from within an Amazon Redshift query. Amazon Redshift Data API is a service that allows you to run SQL statements on Amazon Redshift clusters using HTTP requests, without needing a persistent connection. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost- effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Lambda charges you based on the number of requests and the duration of your functions13. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. AWS Step Functions charges you based on the number of state transitions in your workflows. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue, AWS Lambda, Amazon Redshift Data API, and AWS Step Functions would introduce additional latency and complexity, as you would have to create and invoke Lambda functions to ingest the data from Amazon S3 to Amazon Redshift Serverless using the Data API, and coordinate the ingestion process using Step Functions, instead of querying it directly from Amazon S3 using Athena and Spark. References:
? What is Amazon Athena?
? Apache Spark on Amazon Athena
? Creating tables, updating the schema, and adding new partitions in the Data Catalog from AWS Glue ETL jobs
? Managing Athena workgroups
? Using Amazon QuickSight to visualize data in Amazon Athena
? AWS Glue Data Catalog
? AWS Glue Schema Registry
? What is AWS Glue?
? Amazon Redshift Serverless
? Amazon Redshift provisioned clusters
? Querying external data using Amazon Redshift Spectrum
? Using stored procedures in Amazon Redshift
? What is AWS Lambda?
? [Creating and using AWS Lambda UDFs]
? [Using the Amazon Redshift Data API]
? [What is AWS Step Functions?]
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 13
A company needs to partition the Amazon S3 storage that the company uses for a data lake. The partitioning will use a path of the S3 object keys in the following format: s3://bucket/prefix/year=2023/month=01/day=01.
A data engineer must ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket.
Which solution will meet these requirements with the LEAST latency?

A. Schedule an AWS Glue crawler to run every morning.
B. Manually run the AWS Glue CreatePartition API twice each day.
C. Use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call.
D. Run the MSCK REPAIR TABLE command from the AWS Glue console.

Answer: C

Explanation:
The best solution to ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket with the least latency is to use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call. This way, the Data Catalog is updated as soon as new data is written to S3, and the partition information is immediately available for querying by other
services. The Boto3 AWS Glue create partition API call allows you to create a new partition in the Data Catalog by specifying the table name, the database name, and the partition values1. You can use this API call in your code that writes data to S3, such as a Python script or an AWS Glue ETL job, to create a partition for each new S3 object key that matches the partitioning scheme.
Option A is not the best solution, as scheduling an AWS Glue crawler to run every morning would introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog2. Crawlers can be scheduled to run periodically, such as daily or hourly, but they cannot runcontinuously or in real-time. Therefore, using a crawler to synchronize the Data Catalog with the S3 storage would not meet the

requirement of the least latency.
Option B is not the best solution, as manually running the AWS Glue CreatePartition API twice each day would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. Moreover, manually running the API would require more operational overhead and human intervention than using code that writes data to S3 to invoke the API automatically.
Option D is not the best solution, as running the MSCK REPAIR TABLE command from the AWS Glue console would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. The MSCK REPAIR TABLE command is a SQL command that you can run in the AWS Glue console to add partitions to the Data Catalog based on the S3 object keys that match the partitioning scheme3. However, this command is not meant to be run frequently or in real-time, as it can take a long time to scan the entire S3 bucket and add the partitions. Therefore, using this command to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency. References:
? AWS Glue CreatePartition API
? Populating the AWS Glue Data Catalog
? MSCK REPAIR TABLE Command
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 17
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 data engineer needs to build an extract, transform, and load (ETL) job. The ETL job will process daily incoming .csv files that users upload to an Amazon S3 bucket. The size of each S3 object is less than 100 MB.
Which solution will meet these requirements MOST cost-effectively?

A. Write a custom Python applicatio
B. Host the application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster.
C. Write a PySpark ETL scrip
D. Host the script on an Amazon EMR cluster.
E. Write an AWS Glue PySpark jo
F. Use Apache Spark to transform the data.
G. Write an AWS Glue Python shell jo
H. Use pandas to transform the data.

Answer: D

Explanation:
AWS Glue is a fully managed serverless ETL service that can handle various data sources and formats, including .csv files in Amazon S3. AWS Glue provides two types of jobs: PySpark and Python shell. PySpark jobs use Apache Spark to process large-scale data in parallel, while Python shell jobs use Python scripts to process small-scale data in a single execution environment. For this requirement, a Python shell job is more suitable and cost-effective, as the size of each S3 object is less than 100 MB, which does not require distributed processing. A Python shell job can use pandas, a popular Python library fordata analysis, to transform the .csv data as needed. The other solutions are not optimal or relevant for this requirement. Writing a custom Python application and hosting it on an Amazon EKS cluster would require more effort and resources to set up and manage the Kubernetes environment, as well as to handle the data ingestion and transformation logic. Writing a PySpark ETL script and hosting it on an Amazon EMR cluster would also incur more costs and complexity to provision and configure the EMR cluster, as well as to use Apache Spark for processing small data files. Writing an AWS Glue PySpark job would also be less efficient and economical than a Python shell job, as it would involve unnecessary overhead and charges for using Apache Spark for small data files. References:
? AWS Glue
? Working with Python Shell Jobs
? pandas
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 21
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 22
A company is planning to migrate on-premises Apache Hadoop clusters to Amazon EMR. The company also needs to migrate a data catalog into a persistent storage solution.
The company currently stores the data catalog in an on-premises Apache Hive metastore on the Hadoop clusters. The company requires a serverless solution to migrate the data catalog.
Which solution will meet these requirements MOST cost-effectively?

A. Use AWS Database Migration Service (AWS DMS) to migrate the Hive metastore into Amazon S3. Configure AWS Glue Data Catalog to scan Amazon S3 to produce the data catalog.
B. Configure a Hive metastore in Amazon EM
C. Migrate the existing on-premises Hive metastore into Amazon EM
D. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.
E. Configure an external Hive metastore in Amazon EM
F. Migrate the existing on-premises Hive metastore into Amazon EM
G. Use Amazon Aurora MySQL to store the company's data catalog.
H. Configure a new Hive metastore in Amazon EM
I. Migrate the existing on-premises Hive metastore into Amazon EM
J. Use the new metastore as the company's data catalog.

Answer: A

Explanation:
AWS Database Migration Service (AWS DMS) is a service that helps you migrate databases to AWS quickly and securely. You can use AWS DMS to migrate the Hive metastore from the on-premises Hadoop clusters into Amazon S3, which is a highlyscalable, durable, and cost-effective object storage service. AWS Glue Data Catalog is a serverless, managed service that acts as a central metadata repository for your data assets. You can use AWS Glue Data Catalog to scan the Amazon S3 bucket that contains the migrated Hive metastore and create a data catalog that is compatible with Apache Hive and other AWS services. This solution meets the requirements of migrating the data catalog into a persistent storage solution and using a serverless solution. This solution is also the most cost-effective, as it does not incur any additional charges for running Amazon EMR or Amazon Aurora MySQL clusters. The other options are either not feasible or not optimal. Configuring a Hive metastore in Amazon EMR (option B) or an external Hive metastore in Amazon EMR (option C) would require running and maintaining Amazon EMR clusters, which would incur additional costs and complexity. Using Amazon Aurora MySQL to store the company's data catalog (option C) would also incur additional costs and complexity, as well as introduce compatibility issues with Apache Hive. Configuring a new Hive metastore in Amazon EMR (option D) would not migrate the existing data catalog, but create a new one, which would result in data loss and inconsistency. References:
? Using AWS Database Migration Service
? Populating the AWS Glue Data Catalog
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.2: AWS Glue Data Catalog

NEW QUESTION 27
A company extracts approximately 1 TB of data every day from data sources such as SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. Some of the data sources have undefined data schemas or data schemas that change.
A data engineer must implement a solution that can detect the schema for these data sources. The solution must extract, transform, and load the data to an Amazon S3 bucket. The company has a service level agreement (SLA) to load the data into the S3 bucket within 15 minutes of data creation.
Which solution will meet these requirements with the LEAST operational overhead?

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

Answer: B

Explanation:
AWS Glue is a fully managed service that provides a serverless data integration platform. It can automatically discover and categorize data from various sources, including SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. It can also infer the schema of the data and store it in the AWS Glue Data Catalog, which is a central metadata repository. AWS Glue can then use the schema information to generate and run Apache Spark code to extract, transform, and load the data into an Amazon S3 bucket. AWS Glue can also monitor and optimize the performance and cost of the data pipeline, and handle any schema changes that may occur in the source data. AWS Glue can meet the SLA of loading the data into the S3 bucket within 15 minutes of data creation, as it can trigger the data pipeline based on events, schedules, or on-demand. AWS Glue has the least operational overhead among the options, as it does not require provisioning, configuring, or managing any servers or clusters. It also handles scaling, patching, and security automatically. References:
? AWS Glue
? [AWS Glue Data Catalog]
? [AWS Glue Developer Guide]
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 31
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 34
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 37
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 40
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 44
A company wants to implement real-time analytics capabilities. The company wants to use Amazon Kinesis Data Streams and Amazon Redshift to ingest and process streaming data at the rate of several gigabytes per second. The company wants to derive near real-time insights by using existing business intelligence (BI) and analytics tools.
Which solution will meet these requirements with the LEAST operational overhead?

A. Use Kinesis Data Streams to stage data in Amazon S3. Use the COPY command to load data from Amazon S3 directly into Amazon Redshift to make the data immediately available for real-time analysis.
B. Access the data from Kinesis Data Streams by using SQL querie
C. Create materialized views directly on top of the strea
D. Refresh the materialized views regularly to query the most recent stream data.
E. Create an external schema in Amazon Redshift to map the data from Kinesis Data Streams to an Amazon Redshift objec
F. Create a materialized view to read data from the strea
G. Set the materialized view to auto refresh.
H. Connect Kinesis Data Streams to Amazon Kinesis Data Firehos
I. Use Kinesis Data Firehose to stage the data in Amazon S3. Use the COPY command to load the data from Amazon S3 to a table in Amazon Redshift.

Answer: C

Explanation:
This solution meets the requirements of implementing real-time analytics capabilities with the least operational overhead. By creating an external schema in

Amazon Redshift, you can access the data from Kinesis Data Streams using SQL queries without having to load the data into the cluster. By creating a materialized view on top of the stream, you can store the results of the query in the cluster and make them available for analysis. By setting the materialized view to auto refresh, you can ensure that the view is updated with the latest data from the stream at regular intervals. This way, you can derive near real-time insights by using existing BI and analytics tools. References:
? Amazon Redshift streaming ingestion
? Creating an external schema for Amazon Kinesis Data Streams
? Creating a materialized view for Amazon Kinesis Data Streams

NEW QUESTION 46
A company is planning to use a provisioned Amazon EMR cluster that runs Apache Spark jobs to perform big data analysis. The company requires high reliability. A big data team must follow best practices for running cost-optimized and long-running workloads on Amazon EMR. The team must find a solution that will maintain the company's current level of performance.
Which combination of resources will meet these requirements MOST cost-effectively? (Choose two.)

A. Use Hadoop Distributed File System (HDFS) as a persistent data store.
B. Use Amazon S3 as a persistent data store.
C. Use x86-based instances for core nodes and task nodes.
D. Use Graviton instances for core nodes and task nodes.
E. Use Spot Instances for all primary nodes.

Answer: BD

Explanation:
The best combination of resources to meet the requirements of high reliability, cost-optimization, and performance for running Apache Spark jobs on Amazon EMR is to use Amazon S3 as a persistent data store and Graviton instances for core nodes and task nodes.
Amazon S3 is a highly durable, scalable, and secure object storage service that can store any amount of data for a variety of use cases, including big data analytics1. Amazon S3 is a better choice than HDFS as a persistent data store for Amazon EMR, as it decouples the storage from the compute layer, allowing for more flexibility and cost-efficiency. Amazon S3 also supports data encryption, versioning, lifecycle management, and cross-region replication1. Amazon EMR integrates seamlessly with Amazon S3, using EMR File System (EMRFS) to access data stored in Amazon S3 buckets2. EMRFS also supports consistent view, which enables Amazon EMR to provide read-after-write consistency for Amazon S3 objects that are accessed through EMRFS2.
Graviton instances are powered by Arm-based AWS Graviton2 processors that deliver up to 40% better price performance over comparable current generation x86-based instances3. Graviton instances are ideal for running workloads that are CPU-bound, memory-bound, or network-bound, such as big data analytics, web servers, and open- source databases3. Graviton instances are compatible with Amazon EMR, and can beused for both core nodes and task nodes. Core nodes are responsible for running the data processing frameworks, such as Apache Spark, and storing data in HDFS or the local file system. Task nodes are optional nodes that can be added to a cluster to increase the processing power and throughput. By using Graviton instances for both core nodes and task nodes, you can achieve higher performance and lower cost than using x86-based instances.
Using Spot Instances for all primary nodes is not a good option, as it can compromise the reliability and availability of the cluster. Spot Instances are spare EC2 instances that are available at up to 90% discount compared to On-Demand prices, but they can be interrupted by EC2 with a two-minute notice when EC2 needs the capacity back. Primary nodes are the nodes that run the cluster software, such as Hadoop, Spark, Hive, and Hue, and are essential for the cluster operation. If a primary node is interrupted by EC2, the cluster will fail or become unstable. Therefore, it is recommended to use On-Demand Instances or Reserved Instances for primary nodes, and use Spot Instances only for task nodes that can tolerate interruptions. References:
? Amazon S3 - Cloud Object Storage
? EMR File System (EMRFS)
? AWS Graviton2 Processor-Powered Amazon EC2 Instances
? [Plan and Configure EC2 Instances]
? [Amazon EC2 Spot Instances]
? [Best Practices for Amazon EMR]

NEW QUESTION 48
A data engineer needs to maintain a central metadata repository that users access through Amazon EMR and Amazon Athena queries. The repository needs to provide the schema and properties of many tables. Some of the metadata is stored in Apache Hive. The data engineer needs to import the metadata from Hive into the central metadata repository.
Which solution will meet these requirements with the LEAST development effort?

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

Answer: C

Explanation:
The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such as setting up Apache Ranger for security, managing a Hive metastore on an EMR cluster or an RDS instance, or migrating the metadata manually. References:
? Using the AWS Glue Data Catalog as the metastore for Hive (Section: Specifying
AWS Glue Data Catalog as the metastore)
? Metadata Management: Hive Metastore vs AWS Glue (Section: AWS Glue Data Catalog)
? AWS Glue Data Catalog support for Spark SQL jobs (Section: Importing metadata from an existing Hive metastore)
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 131)

NEW QUESTION 52
A data engineer needs to use AWS Step Functions to design an orchestration workflow. The workflow must parallel process a large collection of data files and apply a specific transformation to each file.
Which Step Functions state should the data engineer use to meet these requirements?

A. Parallel state

B. Choice state
C. Map state
D. Wait state

Answer: C

Explanation:
Option C is the correct answer because the Map state is designed to process a collection of data in parallel by applying the same transformation to each element. The Map state can invoke a nested workflow for each element, which can be another state machine ora Lambda function. The Map state will wait until all the parallel executions are completed before moving to the next state.
Option A is incorrect because the Parallel state is used to execute multiple branches of logic concurrently, not to process a collection of data. The Parallel state can have different branches with different logic and states, whereas the Map state has only one branch that is applied to each element of the collection.
Option B is incorrect because the Choice state is used to make decisions based on a comparison of a value to a set of rules. The Choice state does not process any data or invoke any nested workflows.
Option D is incorrect because the Wait state is used to delay the state machine from continuing for a specified time. The Wait state does not process any data or invoke any nested workflows.
References:
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Orchestration, Section 5.3: AWS Step Functions, Pages 131-132
? Building Batch Data Analytics Solutions on AWS, Module 5: Data Orchestration, Lesson 5.2: AWS Step Functions, Pages 9-10
? AWS Documentation Overview, AWS Step Functions Developer Guide, Step Functions Concepts, State Types, Map State, Pages 1-3

NEW QUESTION 57
......

THANKS FOR TRYING THE DEMO OF OUR PRODUCT