PassQueen
a.k.a dumpsqueen.com

AWS Certified Data Engineer Associate (DEA-C01)

Amazon AWS Data-Engineer-Associate-DEA-C01

Version Demo

Total Demo Questions: 30

Total Premium Questions: 306

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Data Ingestion and Transformation	132
Topic 2, Data Store Management	72
Topic 3, Data Operations and Support	43
Topic 4, Data Security and Governance	59
Total	306

QUESTION NO: 1

A company has a data lake in Amazon S3. The company collects AWS CloudTrail logs for multiple applications. The company stores the logs in the data lake, catalogs the logs in AWS Glue, and partitions the logs based on the year. The company uses Amazon Athena to analyze the logs.

Recently, customers reported that a query on one of the Athena tables did not return any data. A data engineer must resolve the issue.

Which combination of troubleshooting steps should the data engineer take? (Select TWO.)

A. Confirm that Athena is pointing to the correct Amazon S3 location.

B. Increase the query timeout duration.

C. Use the MSCK REPAIR TABLE command.

D. Restart Athena.

E. Delete and recreate the problematic Athena table.

ANSWER: A C

Explanation:

Confirming that Athena is pointing to the correct Amazon S3 location is an essential diagnostic step. Athena uses the table metadata stored in the AWS Glue Data Catalog, including the table `LOCATION`, to identify the S3 prefix that contains the records to query. If this location is incorrect, outdated, or references a prefix that does not contain the CloudTrail logs, Athena can complete a query successfully while returning zero rows. The engineer should compare the table location with the actual S3 data lake path and CloudTrail log layout.

Using `MSCK REPAIR TABLE` is appropriate because the table is partitioned by year. Athena queries partitioned data according to partition definitions in the AWS Glue Data Catalog. When year partitions are created in S3 but their metadata is absent from the catalog, Athena does not discover and scan those directories automatically. `MSCK REPAIR TABLE` checks the table's S3 location and adds compatible Hive-style partitions to the catalog, making the existing log data available to Athena queries. For partition projection or non-Hive-style layouts, partitions can instead be added explicitly, but repairing compatible partitions is the relevant troubleshooting action here. See the AWS documentation for MSCK REPAIR TABLE and partitioning data in Athena.

QUESTION NO: 2

A company wants to ingest streaming data into an Amazon Redshift data warehouse from an Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster. A data engineer needs to develop a solution that provides low data access time and that optimizes storage costs.

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

A. Create an external schema that maps to the MSK cluster. Create a materialized view that references the external schema to consume the streaming data from the MSK topic.

B. Develop an AWS Glue streaming extract, transform, and load (ETL) job to process the incoming data from Amazon MSK. Load the data into Amazon S3. Use Amazon Redshift Spectrum to read the data from Amazon S3.

C. Create an external schema that maps to the streaming data source. Create a new Amazon Redshift table that references the external schema.

D. Create an Amazon S3 bucket. Ingest the data from Amazon MSK. Create an event-driven AWS Lambda function to load the data from the S3 bucket to a new Amazon Redshift table.

ANSWER: A

Explanation:

Create an external schema that maps to the MSK cluster. Create a materialized view that references the external schema to consume the streaming data from the MSK topic. This uses Amazon Redshift streaming ingestion, a managed integration designed to ingest records directly from Amazon MSK into Redshift with low latency. The external schema defines the connection to the Kafka source, while the materialized view provides the Redshift object that consumes the topic data. Redshift incrementally ingests new records when the materialized view is refreshed, and the ingested data is then available for efficient local querying in the warehouse.

This native approach avoids operating separate streaming ETL infrastructure, staging data in an intermediate system, and building custom consumer or loading logic. It therefore minimizes pipeline components that would otherwise need to be deployed, scaled, scheduled, monitored, and maintained. It also avoids maintaining a duplicate S3 staging layer solely for warehouse ingestion, helping control storage use while retaining data in Redshift in a query-optimized form. AWS documents this external-schema and materialized-view pattern for streaming ingestion from Amazon MSK. See the Amazon Redshift streaming ingestion documentation and the Amazon MSK streaming ingestion tutorial.

QUESTION NO: 3

A company has a data pipeline that uses an Amazon RDS instance, AWS Glue jobs, and an Amazon S3 bucket. The RDS instance and AWS Glue jobs run in a private subnet of a VPC and in the same security group.

A use' made a change to the security group that prevents the AWS Glue jobs from connecting to the RDS instance. After the change, the security group contains a single rule that allows inbound SSH traffic from a specific IP address.

The company must resolve the connectivity issue. Which solution will meet this requirement?

A. Add an inbound rule that allows all TCP traffic on all TCP ports. Set the security group as the source.

B. Add an inbound rule that allows all TCP traffic on all UDP ports. Set the private IP address of the RDS instance as the source.

C. Add an inbound rule that allows all TCP traffic on all TCP ports. Set the DNS name of the RDS instance as the source.

D. Replace the source of the existing SSH rule with the private IP address of the RDS instance. Create an outbound rule with the same source, destination, and protocol as the inbound SSH rule.

ANSWER: A

Explanation:

Adding an inbound rule that allows all TCP traffic on all TCP ports with the security group itself as the source restores the self-referencing rule required for AWS Glue resources that access data stores within a VPC. AWS Glue creates and manages elastic network interfaces in the specified subnets when a job uses a VPC connection. The security group attached to those interfaces must permit the Glue job components to communicate with resources that use the same security group, including the Amazon RDS instance. A security-group source does this without depending on individual private IP addresses, which can change as Glue provisions and releases network interfaces.

AWS Glue documentation specifically states that a security group for a VPC connection needs an inbound rule that allows all TCP ports with the security group itself as the source. This is necessary for communication between AWS Glue components. Because the RDS instance and Glue jobs are associated with the same group, the self-reference allows the job's TCP connection to the database listener while supporting the required internal Glue networking. Security group rules are stateful, so return traffic for allowed connections is automatically permitted. AWS also supports security-group references as rule sources, allowing access based on group membership rather than fixed addresses. See AWS Glue VPC connection requirements and Amazon VPC security group rules.

QUESTION NO: 4

A company runs multiple applications on AWS. The company configured each application to output logs. The company wants to query and visualize the application logs in near real time.

Which solution will meet these requirements?

A. Configure the applications to output logs to Amazon CloudWatch Logs log groups. Create an Amazon S3 bucket. Create an AWS Lambda function that runs on a schedule to export the required log groups to the S3 bucket. Use Amazon Athena to query the log data in the S3 bucket.

B. Create an Amazon OpenSearch Service domain. Configure the applications to output logs to Amazon CloudWatch Logs log groups. Create an OpenSearch Service subscription filter for each log group to stream the data to OpenSearch. Create the required queries and dashboards in OpenSearch Service to analyze and visualize the data.

C. Configure the applications to output logs to Amazon CloudWatch Logs log groups. Use CloudWatch log anomaly detection to query and visualize the log data.

D. Update the application code to send the log data to Amazon QuickSight by using Super-fast, Parallel, In-memory Calculation Engine (SPICE). Create the required analyses and dashboards in QuickSight.

E. Configure the applications to output logs to Amazon CloudWatch Logs log groups. Create an Amazon OpenSearch Service domain and use CloudWatch Logs subscription filters to invoke a Lambda function that streams the logs to OpenSearch. Create queries and dashboards in OpenSearch Dashboards.

ANSWER: E

Explanation:

Configure the applications to send logs to Amazon CloudWatch Logs, then use CloudWatch Logs subscription filters to invoke a Lambda function that streams the log events to Amazon OpenSearch Service. This architecture provides near-real-time processing because subscription filters forward matching log events as they are ingested by CloudWatch Logs. The Lambda function can decode the CloudWatch Logs payload, transform events if necessary, and index them into an OpenSearch domain. Once indexed, the logs are immediately available for full-text search, aggregation, and analysis. OpenSearch Dashboards provides the visualization layer, allowing the company to create operational dashboards for errors, request patterns, application latency, and other log-derived metrics across all applications. This is an AWS-supported integration pattern for centralized log analytics: CloudWatch Logs subscription filters invoke a processing function, and the function writes documents to OpenSearch. The subscription filter itself does not use OpenSearch Service as a direct destination; Lambda supplies the required delivery and indexing step. For implementation details, see Streaming CloudWatch Logs data to Amazon OpenSearch Service and the CloudWatch Logs subscription filters documentation.

QUESTION NO: 5

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: B D

Explanation:

Using Amazon S3 as a persistent data store provides a durable, highly available storage layer that is independent of the Amazon EMR cluster’s compute nodes. This architecture is a core EMR best practice for long-running analytics: Spark can read source data from and write results to S3, while the cluster can be resized, repaired, or replaced without making the data dependent on the lifecycle of local cluster storage. Separating persistent storage from compute improves operational reliability and permits the team to optimize the cluster’s compute capacity without retaining unnecessary storage solely for durability.

Using Graviton instances for core nodes and task nodes can reduce the cost of supported Spark workloads while preserving the required processing capability. Amazon EMR supports AWS Graviton instance families, and AWS identifies Graviton as a price-performance option for big-data workloads. Applying these instances to both core and task capacity extends the price-performance benefit across the nodes that run distributed Spark processing. The team should validate application dependencies and benchmark the selected Graviton instance family and size against its existing workload, then use the resulting capacity configuration to maintain the current performance target. See the Amazon EMR guidance on storage and file systems and supported instance types.

QUESTION NO: 6

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 bucket. Organize the data by year, month, and day.

B. Increase the AWS Glue instance size by scaling up the worker type.

C. Convert the AWS Glue schema to the DynamicFrame schema class.

D. Adjust AWS Glue job scheduling frequency so the jobs run half as many times each day.

E. Modify the 1AM role that grants access to AWS glue to grant access to all S3 features.

ANSWER: A B

Explanation:

Partition the data that is in the S3 bucket. Organize the data by year, month, and day. This creates a date-based S3 layout that enables AWS Glue Spark jobs to limit reads to the partitions relevant to a job run. As daily data accumulates, partition pruning prevents the job from scanning the entire historical dataset when processing a specific date range. This reduces S3 I/O, the amount of data shuffled and processed by Spark, job duration, and cost. AWS Glue supports partitioned S3 datasets through Data Catalog table partition metadata and predicate pushdown. See Managing partitions for ETL output in AWS Glue.

Increase the AWS Glue instance size by scaling up the worker type. AWS Glue worker types determine the compute, memory, and local disk resources available to each worker. Selecting a larger worker type gives resource-constrained Spark workloads additional memory and processing capacity, which can reduce disk spilling and allow transformations, joins, aggregations, and other parallel operations to complete more quickly. This is particularly useful after optimizing data layout, when job metrics indicate memory or executor resource pressure. AWS documents worker sizing and Spark tuning considerations in Tuning AWS Glue for Apache Spark jobs.

QUESTION NO: 7

A company uses Amazon S3 to store data and Amazon QuickSight to create visualizations.

The company has an S3 bucket in an AWS account named Hub-Account. The S3 bucket is encrypted by an AWS Key Management Service (AWS KMS) key. The company's QuickSight instance is in a separate account named BI-Account

The company updates the S3 bucket policy to grant access to the QuickSight service role. The company wants to enable cross-account access to allow QuickSight to interact with the S3 bucket. Which combination of steps will meet this requirement? (Select TWO.)

A. Use the existing AWS KMS key to encrypt connections from QuickSight to the S3 bucket.

B. Add the S3 bucket as a resource that the QuickSight service role can access.

C. Use AWS Resource Access Manager (AWS RAM) to share the S3 bucket with the BI-Account account.

D. Add an IAM policy to the QuickSight service role to give QuickSight access to the KMS key that encrypts the S3 bucket.

E. Add the KMS key as a resource that the QuickSight service role can access.

ANSWER: B E

Explanation:

QuickSight must be authorized within BI-Account to use the specific S3 bucket in Hub-Account. Adding the S3 bucket as a resource that the QuickSight service role can access updates the QuickSight-managed permissions for its service role, allowing QuickSight to make the required S3 requests. The bucket policy in Hub-Account supplies the corresponding resource-side authorization for this cross-account scenario. Together, the role permissions and bucket policy establish the permissions path QuickSight needs to read the data.

Because the S3 objects are encrypted with an AWS KMS key, authorization to read the bucket alone is insufficient. The QuickSight service role also needs permission to use the KMS key for decryption operations when it accesses encrypted objects. Adding the KMS key as a resource that the QuickSight service role can access configures the required QuickSight-side KMS access. The KMS key policy must also permit the role's use of the key, as KMS evaluates both IAM permissions and the key policy for cross-account access. This enables QuickSight to retrieve and decrypt the S3 data for analysis or SPICE ingestion. See Amazon QuickSight S3 data access settings and AWS KMS cross-account key access.

QUESTION NO: 8

A company uses AWS Lake Formation to govern a data lake that contains hundreds of AWS Glue Data Catalog tables. The company must grant access to groups of tables based on business domains. The company wants to avoid managing permissions separately for each table.

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

A. Create LF-Tags that represent the business domains.

B. Associate the LF-Tags with the relevant databases and tables. Grant LF-Tag-based permissions to the required principals.

C. Create an IAM user for each table and grant the user Amazon S3 full access.

D. Create a separate Amazon S3 bucket for every business domain.

E. Use AWS Glue crawlers to apply IAM policies to each table.

ANSWER: A B

Explanation:

Create AWS Lake Formation tags that represent the business domains. LF-Tags provide a tag-based authorization model for data lake resources. A tag can use a key and value pair, such as `domain=finance` or `domain=marketing`, to classify resources according to the company's governance model.

Associate the LF-Tags with the relevant databases and tables, and grant LF-Tag-based permissions to the required principals. Lake Formation evaluates the tags associated with resources when a principal requests access. This enables a single tag-based permission grant to apply to all matching tables, including tables that are tagged later, which reduces the administration required for a large and growing catalog. See Lake Formation tag-based access control.

QUESTION NO: 9

A retail company stores order information in an Amazon Aurora table named Orders. The company needs to create operational reports from the Orders table with minimal latency. The Orders table contains billions of rows, and over 100,000 transactions can occur each second.

A marketing team needs to join the Orders data with an Amazon Redshift table named Campaigns in the marketing team ' s data warehouse. The operational Aurora database must not be affected.

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

A. Use AWS Database Migration Service (AWS DMS) Serverless to replicate the Orders table to Amazon Redshift. Create a materialized view in Amazon Redshift to join with the Campaigns table.

B. Use the Aurora zero-ETL integration with Amazon Redshift to replicate the Orders table. Create a materialized view in Amazon Redshift to join with the Campaigns table.

C. Use AWS Glue to replicate the Orders table to Amazon Redshift. Create a materialized view in Amazon Redshift to join with the Campaigns table.

D. Use federated queries to query the Orders table directly from Aurora. Create a materialized view in Amazon Redshift to join with the Campaigns table.

ANSWER: B

Explanation:

Use the Aurora zero-ETL integration with Amazon Redshift to replicate the Orders table. Create a materialized view in Amazon Redshift to join with the Campaigns table. Aurora zero-ETL integration is designed to make data from an Aurora DB cluster available in Amazon Redshift for near-real-time analytics without requiring a custom extract, transform, and load process. Changes made to the Aurora Orders table are replicated continuously to Redshift, allowing the reporting environment to remain current while isolating analytical queries from the high-throughput operational database. This separation is especially important when the source table has billions of records and receives more than 100,000 transactions per second.

After replication, both Orders and Campaigns are available in Amazon Redshift, so the marketing team can perform the join entirely within its data warehouse. A Redshift materialized view can persist the joined result and be refreshed as needed, reducing compute for repeated operational-report queries and improving response times. Because AWS manages the ongoing Aurora-to-Redshift replication, this approach provides the required low-latency reporting architecture with minimal operational administration. Aurora zero-ETL integration is specifically intended to eliminate the need to build and manage complex data pipelines for near-real-time analytics. For implementation details, see the Amazon Aurora zero-ETL integrations documentation and the Amazon Redshift materialized views documentation.

QUESTION NO: 10

A mobile gaming company wants to capture data from its gaming app. The company wants to make the data available to three internal consumers of the data. The data records are approximately 20 KB in size.

The company wants to achieve optimal throughput from each device that runs the gaming app. Additionally, the company wants to develop an application to process data streams. The streamprocessing application must have dedicated throughput for each internal consumer.

Which solution will meet these requirements?

A. Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Use the enhanced fan-out feature with a registered consumer for each internal consumer.

B. Configure the mobile app to call the PutRecordBatch API operation to send data to Amazon Data Firehose. Submit an AWS Support case to turn on dedicated throughput for the company's AWS account. Allow each internal consumer to access the stream.

C. Configure the mobile app to use the Amazon Kinesis Producer Library (KPL) to send data to Amazon Data Firehose. Use the enhanced fan-out feature with a stream for each internal consumer.

D. Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Host the stream-processing application for each internal consumer on Amazon EC2 instances. Configure auto scaling for the EC2 instances.

ANSWER: A

Explanation:

Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Use enhanced fan-out with a registered consumer for each internal consumer. This design supports both efficient producer ingestion and independent, predictable consumer performance. `PutRecords` batches multiple records into one request, reducing per-request overhead for a mobile producer and allowing it to ingest records efficiently. The approximately 20 KB event size is comfortably within the maximum size allowed for an individual Kinesis Data Streams record.

Amazon Kinesis Data Streams is designed to retain and distribute the same real-time stream to multiple applications. Enhanced fan-out registers each consuming application as a stream consumer. Each registered consumer receives dedicated read throughput of up to 2 MB/second per shard and uses push-based delivery, so each internal consumer can process the stream without sharing standard polling read throughput with the others. This directly satisfies the requirement for dedicated throughput per internal consumer while preserving a single source stream for the gaming events.

AWS documents enhanced fan-out behavior and per-consumer dedicated throughput in the Amazon Kinesis Data Streams enhanced fan-out documentation. Batching behavior and limits for producer requests are described in the PutRecords API reference.

QUESTION NO: 11

A telecommunications company collects network usage data throughout each day at a rate of several thousand data points each second. The company runs an application to process the usage data in real time. The company aggregates and stores the data in an Amazon Aurora DB instance.

Sudden drops in network usage usually indicate a network outage. The company must be able to identify sudden drops in network usage so the company can take immediate remedial actions.

Which solution will meet this requirement with the LEAST latency?

A. Create an AWS Lambda function to query Aurora for drops in network usage. Use Amazon EventBridge to automatically invoke the Lambda function every minute.

B. Modify the processing application to publish the data to an Amazon Kinesis data stream. Create an Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) application to detect drops in network usage.

C. Replace the Aurora database with an Amazon DynamoDB table. Create an AWS Lambda function to query the DynamoDB table for drops in network usage every minute. Use DynamoDB Accelerator (DAX) between the processing application and DynamoDB table.

D. Create an AWS Lambda function within the Database Activity Streams feature of Aurora to detect drops in network usage.

ANSWER: B

Explanation:

Modify the processing application to publish the data to an Amazon Kinesis data stream, then create an Amazon Managed Service for Apache Flink application to detect drops in network usage. This design evaluates the usage events continuously while they are flowing through the ingestion pipeline, rather than waiting for data to be stored and queried afterward. That minimizes the time between an actual reduction in traffic and detection of the outage condition.

Amazon Kinesis Data Streams is designed for real-time ingestion of high-volume event data and supports producers writing records continuously for downstream consumers. Amazon Managed Service for Apache Flink can consume that stream and perform stateful stream processing, including time-windowed aggregations. For example, the application can calculate usage totals over short rolling windows, compare the current rate with a recent baseline, and immediately emit an alert when the rate falls below the defined threshold. The processing application can continue storing aggregated results in Aurora independently, while outage detection remains on the low-latency streaming path. This separation ensures the detection workflow does not depend on database write completion, scheduled invocation intervals, or subsequent database queries. See the Amazon Kinesis Data Streams Developer Guide and the Amazon Managed Service for Apache Flink Developer Guide.

QUESTION NO: 12

A gaming company uses AWS Glue to perform read and write operations on Apache Iceberg tables for real-time streaming data. The data in the Iceberg tables is stored in Apache Parquet format. The company is experiencing slow query performance.

Which solutions will improve query performance? (Select TWO)

A. Use AWS Glue Data Catalog to generate column-level statistics for the Iceberg tables on a schedule.

B. Use AWS Glue Data Catalog to automatically compact the Iceberg tables.

C. Use AWS Glue Data Catalog to automatically optimize indexes for the Iceberg tables.

D. Use AWS Glue Data Catalog to enable copy-on-write for the Iceberg tables.

E. Use AWS Glue Data Catalog to generate views for the Iceberg tables.

ANSWER: A B

Explanation:

Use AWS Glue Data Catalog to generate column-level statistics for the Iceberg tables on a schedule. AWS Glue Data Catalog provides managed column statistics for Apache Iceberg tables. These statistics provide query engines with information such as the number of distinct values, null counts, minimum values, and maximum values. Current statistics help engines make better cost-based planning decisions, including determining efficient join strategies and reducing unnecessary data scans through predicate pruning. Because streaming ingestion continuously changes the table contents, scheduling statistics generation ensures that the metadata remains useful as new Parquet data files are written.

Use AWS Glue Data Catalog to automatically compact the Iceberg tables. Real-time streaming writes commonly produce a large number of small Parquet files. Query engines incur planning, metadata, and Amazon S3 request overhead when they must open and scan many small objects. The AWS Glue Data Catalog compaction optimizer rewrites these small files into fewer, appropriately sized files while preserving Iceberg table semantics. This reduces file-management overhead and generally improves read and scan efficiency. AWS documents these capabilities in its column statistics documentation and Data Catalog table optimizers documentation.

QUESTION NO: 13

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

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

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

A. Use a second Lambda function to invoke the first Lambda function based on Amazon CloudWatch events.

B. Use the Amazon Redshift Data API to publish an event to Amazon EventBridge. Configure an EventBridge rule to invoke the Lambda function.

C. Use the Amazon Redshift Data API to publish a message to an Amazon Simple Queue Service (Amazon SQS) queue. Configure the SQS queue to invoke the Lambda function.

D. Use a second Lambda function to invoke the first Lambda function based on AWS CloudTrail events.

ANSWER: B

Explanation:

Use the Amazon Redshift Data API to publish an event to Amazon EventBridge. Configure an EventBridge rule to invoke the Lambda function. This provides an event-driven way to update the DynamoDB tracking table when Redshift load activity completes. The Redshift Data API can execute SQL statements asynchronously, including statements used to load data into

Redshift tables. When the Data API `ExecuteStatement` request uses the `WithEvent` parameter, it sends a status event to the default Amazon EventBridge event bus after execution. The event contains statement execution information that can identify the completed operation and its status.

An EventBridge rule can match these Redshift Data API events and use the existing Lambda function as its target. Lambda then processes the event and writes the appropriate loaded, failed, or other status record to DynamoDB. This decouples the loading workflow from status persistence, avoids maintaining a persistent connection or polling Redshift for completion, and supports reliable operational tracking across multiple table-load statements. The EventBridge rule must have permission to invoke the Lambda function, which AWS configures when Lambda is selected as the rule target through supported configuration workflows. See the Redshift Data API ExecuteStatement reference and Amazon EventBridge targets documentation.

QUESTION NO: 14

A company stores sensitive customer files in an Amazon S3 bucket. A security team must record all object-level read and write activity in the bucket. The security team must retain the audit records in a separate centralized logging account.

Which combination of steps will meet these requirements? (Choose two.)

A. Configure a CloudTrail trail to log Amazon S3 data events for the bucket.

B. Configure the trail to deliver logs to an S3 bucket in the centralized logging account. Grant CloudTrail permission to write to the bucket.

C. Configure Amazon CloudWatch metrics for the bucket.

D. Enable Amazon S3 Transfer Acceleration on the bucket.

E. Create an AWS Glue crawler for the bucket every hour.

ANSWER: A B

Explanation:

Configure an AWS CloudTrail trail to log Amazon S3 data events for the bucket. CloudTrail management events do not record individual S3 object operations such as `GetObject`, `PutObject`, and `DeleteObject`. S3 data events provide the required object-level API activity, including the identity that made the request and details of the accessed object.

Configure the trail to deliver logs to an S3 bucket in the centralized logging account, and grant CloudTrail permission to write to that bucket. A centralized destination keeps audit records separate from the workload account. The destination bucket policy must allow the CloudTrail service principal to write the log objects and obtain the bucket ACL as required. See Logging data events with CloudTrail and Amazon S3 bucket policy for CloudTrail.

QUESTION NO: 15

A company receives marketing campaign data from a vendor. The company ingests the data into an Amazon S3 bucket every 40 to 60 minutes. The data is in CSV format. File sizes are between 100 KB and 300 KB.

A data engineer needs to set-up an extract, transform, and load (ETL) pipeline to upload the content of each file to Amazon Redshift.

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

A. Create an AWS Lambda function that connects to Amazon Redshift and runs a COPY command. Use Amazon EventBridge to invoke the Lambda function based on an Amazon S3 upload trigger.

B. Create an Amazon Data Firehose stream. Configure the stream to use an AWS Lambda function as a source to pull data from the S3 bucket. Set Amazon Redshift as the destination.

C. Use Amazon Redshift Spectrum to query the S3 bucket. Configure an AWS Glue Crawler for the S3 bucket to update metadata in an AWS Glue Data Catalog.

D. Creates an AWS Database Migration Service (AWS DMS) task. Specify an appropriate data schema to migrate. Specify the appropriate type of migration to use.

ANSWER: A

Explanation:

Create an AWS Lambda function that connects to Amazon Redshift and runs a COPY command. Use Amazon EventBridge to invoke the Lambda function based on an Amazon S3 upload trigger. This is an event-driven, serverless design that fits the arrival pattern: each independently delivered CSV file can cause processing without requiring a continuously running ingestion application, polling process, or schedule to manage. Amazon S3 can send object-created events to Amazon EventBridge, which can then route matching events to a Lambda target. The Lambda function can use the object bucket and key from the event to submit a Redshift COPY command for that newly delivered file.

Amazon Redshift COPY is the native bulk-load command for data stored in Amazon S3 and supports CSV input. The Lambda execution role can be configured with the permissions needed to invoke Redshift and read the S3 object, while Amazon Redshift uses an attached IAM role to retrieve the source data. This separation keeps the workflow managed and scalable while ensuring each arrival is processed promptly. For small files arriving only every 40 to 60 minutes, the absence of persistent infrastructure provides the least operational overhead. See Amazon S3 EventBridge integration and Amazon Redshift COPY command.

QUESTION NO: 16

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 deployment. Create a step to pause the state machine during the EMR jobs that fail. Configure the step to wait for a human user to send approval through an email message. Include details of the EMR task in the email message for further analysis.

B. Verify that the Step Functions execution role has all IAM permissions that are necessary to create and run the EMR jobs. Verify that the EMR job roles have IAM permissions to access the Amazon S3 buckets that the EMR jobs use. Use Access Analyzer for S3 to check the S3 access properties.

C. Check for entries in Amazon CloudWatch for the newly created EMR cluster. Change the AWS Step Functions state machine code to use Amazon EMR on EKS. 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).

D. Query the flow logs for the VPC. Determine whether the traffic that originates from the EMR cluster can successfully reach the data providers. 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.

E. Check the retry scenarios that the company configured for the EMR jobs. Increase the number of seconds in the interval between each EMR task. Validate that each fallback state has the appropriate catch for each decision state. Configure an Amazon Simple Notification Service (Amazon SNS) topic to store the error messages.

ANSWER: B D

Explanation:

Verifying the IAM permissions required to create and run the EMR jobs is essential because AWS Step Functions uses its execution role when it invokes the Amazon EMR optimized integration APIs, such as creating a cluster, adding steps, or describing step status. The execution role must authorize the specific EMR actions used by the workflow and, where required, allow `iam:PassRole` for the EMR service and EC2 instance-profile roles. Separately, the EMR runtime roles need

appropriate access to the S3 locations containing application scripts, input data, and output data. Reviewing S3 bucket access properties with IAM Access Analyzer can help identify access configurations that prevent the EMR workload from using those locations. AWS documents the required Step Functions permissions for EMR integrations at Integrating Step Functions with Amazon EMR.

Querying VPC Flow Logs is also an appropriate diagnostic step because the EMR cluster runs within the VPC and its jobs must establish network connectivity to source systems and, for loading operations, to Amazon Redshift. Flow-log records show whether relevant traffic was accepted or rejected. This provides evidence for investigating security-group rules, network ACLs, route tables, DNS resolution, and required service endpoints. Confirming that the cluster security groups permit traffic to data-source servers on the required ports directly validates the network path on which the EMR jobs depend. See VPC Flow Logs documentation.

QUESTION NO: 17

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

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

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

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

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

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

ANSWER: A

Explanation:

Use an AWS Step Functions workflow that includes a state machine. Configure the state machine to run the Lambda function and then the AWS Glue job. AWS Step Functions is a fully managed serverless workflow orchestration service that is designed to coordinate AWS services without requiring the data engineer to provision or operate orchestration infrastructure. A state machine explicitly models the required sequence: invoke the Lambda function first, then start and wait for completion of the AWS Glue job. Step Functions provides direct, optimized integrations for both services, including the `glue:startJobRun.sync` integration pattern, which lets the workflow wait for the Glue job result before proceeding.

This approach has minimal operational overhead because AWS manages the availability, scaling, workflow execution infrastructure, and execution history. The state machine also supplies built-in retry policies, catch/error-handling paths, timeouts, logging integration, and visual execution monitoring. These features allow the pipeline to be operated and troubleshot through managed AWS service capabilities instead of custom code or self-managed scheduling infrastructure. The workflow can be defined declaratively using Amazon States Language and secured through an IAM execution role that grants only the required Lambda invocation and Glue job permissions. See the AWS Step Functions Developer Guide and the Step Functions AWS Glue integration documentation.

QUESTION NO: 18

A company stores customer transaction data in an Amazon S3 data lake. Multiple data engineering teams use Amazon Athena and AWS Glue jobs to read and write the data. The company needs support for ACID transactions, schema evolution, and consistent reads when concurrent jobs modify tables.

Which combination of steps will meet these requirements? (Choose two.)

A. Create the data lake tables as Apache Iceberg tables.

B. Configure the AWS Glue Data Catalog as the Iceberg catalog.

C. Store each table as unpartitioned CSV files in Amazon S3.

D. Configure S3 Versioning as the table transaction mechanism.

E. Use an AWS Glue crawler before every Athena query.

ANSWER: A B

Explanation:

Creating the tables as Apache Iceberg tables provides the required transactional table capabilities on Amazon S3. Iceberg maintains table metadata and snapshots that allow supported engines to perform atomic data operations and consistent reads. Iceberg also supports schema evolution, so columns can be added, removed, or changed through supported operations without requiring consumers to manually manage individual S3 objects.

Using the AWS Glue Data Catalog as the Iceberg catalog provides a managed central catalog for the table metadata. Athena and AWS Glue can use this catalog to locate Iceberg table metadata and coordinate supported table operations. This allows the teams to continue using their existing lake storage and analytics services while gaining transactional table behavior. See Using Apache Iceberg tables in Athena and Using the Iceberg framework in AWS Glue.

QUESTION NO: 19

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

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

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

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

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

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

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

ANSWER: D

Explanation:

Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers. AWS Glue DataBrew is a fully managed visual data-preparation service that is well suited to this task because the source is a file in Amazon S3 and the required logic is a straightforward transformation followed by an aggregate calculation. The engineer can create a DataBrew dataset for the Excel file, use recipe steps to concatenate the first-name and last-name fields into a single customer identifier, and apply an aggregate operation that calculates the number of unique values in that derived field. The recipe can then be run as a managed DataBrew job for each daily delivery. This avoids developing, testing, deploying, and operating custom Spark code or query infrastructure. DataBrew is intended to let analysts and data engineers clean, transform, and prepare data through an interactive interface while retaining reusable, repeatable recipe definitions. Its managed job execution and recipe-based workflow provide the lowest operational burden for this file-oriented transformation and distinct-count requirement. See What is AWS Glue DataBrew? and the AWS Glue DataBrew recipe actions reference.

QUESTION NO: 20

A company currently uses a provisioned Amazon EMR cluster that includes general purpose Amazon EC2 instances. The EMR cluster uses EMR managed scaling between one to five task nodes for the company ' s long-running Apache Spark extract, transform, and load (ETL) job. The company runs the ETL job every day.

When the company runs the ETL job, the EMR cluster quickly scales up to five nodes. The EMR cluster often reaches maximum CPU usage, but the memory usage remains under 30%.

The company wants to modify the EMR cluster configuration to reduce the EMR costs to run the daily ETL job.

Which solution will meet these requirements MOST cost-effectively?

A. Increase the maximum number of task nodes for EMR managed scaling to 10.

B. Change the task node type from general purpose EC2 instances to memory optimized EC2 instances.

C. Switch the task node type from general purpose EC2 instances to compute optimized EC2 instances.

D. Reduce the scaling cooldown period for the provisioned EMR cluster.

ANSWER: C

Explanation:

Switch the task node type from general purpose EC2 instances to compute optimized EC2 instances because the observed utilization shows that this Spark ETL workload is CPU-bound: CPU is consistently fully utilized while memory consumption stays below 30%. Compute optimized Amazon EC2 families provide a higher vCPU-to-memory ratio than general purpose families. This better aligns the capacity purchased for the task nodes with the resource that limits job performance, avoiding payment for comparatively unused memory capacity. For distributed Spark processing, task nodes supply elastic executor capacity to process ETL work, making them an appropriate target for this instance-family adjustment. Amazon EMR supports selecting EC2 instance types according to a cluster's workload requirements, and AWS identifies compute optimized instances as suitable for compute-intensive uses such as batch processing and distributed analytics. Because the cluster repeatedly scales to its configured task-node maximum, choosing a more CPU-focused instance type can deliver the necessary compute capacity more efficiently and lower the daily job's infrastructure cost. The change preserves EMR managed scaling while making each scaled-out task node a closer fit for the measured workload profile. See the AWS guidance on planning Amazon EMR instance types and the Amazon EC2 compute optimized instance overview.

QUESTION NO: 21

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: B D

Explanation:

Using Amazon S3 as a persistent data store separates durable data from the EMR cluster's compute and local storage. This is an EMR best practice for reliable, long-running analytics: Spark jobs can read and write durable input and output data in S3, while the cluster can be resized, recovered, or replaced without making HDFS the sole location of business-critical data.

Amazon S3 is designed for high durability and availability, and EMR integrates with it through EMRFS. This architecture also avoids paying to retain oversized cluster-attached storage simply to preserve data. See the Amazon EMR file systems documentation.

Using Graviton instances for core nodes and task nodes can lower the compute cost of a provisioned Spark cluster while retaining the performance needed for scale-out data processing, provided the workload and its dependencies are compatible with the ARM64 architecture. AWS Graviton processors are intended to provide improved price performance for supported workloads, making them a strong fit for sustained EMR processing where instance costs accumulate over time. Core nodes can continue to provide HDFS and YARN capacity, while task nodes supply additional compute capacity for Spark execution. AWS includes Graviton-based instance families in its EMR instance-planning guidance: Amazon EMR instance planning guidelines.

QUESTION NO: 22

A company runs an extract, transform, and load (ETL) job in AWS Glue. The job processes personally identifiable information (PII) data and writes logs to an Amazon CloudWatch Logs log group. A data engineer needs to mask PII data in the CloudWatch Logs log group.

Which solution will meet these requirements?

A. Attach an AWS Glue security configuration to the ETL job.

B. Configure a data protection policy. Attach the policy to the CloudWatch log group.

C. Run an Amazon Macie sensitive data discovery job.

D. Call AWS Glue sensitive data detection APIs in the ETL job.

ANSWER: B

Explanation:

Configure a data protection policy and attach it to the CloudWatch Logs log group. CloudWatch Logs data protection policies are designed specifically to find sensitive information in log events and mask it when the events are viewed. The policy can use AWS managed data identifiers for common sensitive-data types, including personally identifiable information, and can also use custom data identifiers when needed. This lets the data engineer apply protection at the destination where the AWS Glue job's operational output is stored, without needing to modify the ETL logic solely to redact each log message.

A data protection policy includes a statement that identifies sensitive data and specifies the audit and de-identification operations. After it is applied to a log group, matching values in new log events are masked by CloudWatch Logs. Users do not see the unmasked values unless their IAM permissions explicitly allow unmasking, such as through the `logs:Unmask` permission. This provides centralized, native log-data protection while retaining the usefulness of the logs for troubleshooting and monitoring. AWS documents log-group data protection policies and their managed identifiers at Help protect sensitive log data with masking and documents the API for applying the policy at PutDataProtectionPolicy.

QUESTION NO: 23

A data engineer must orchestrate a series of Amazon Athena queries that will run every day. Each query can run for more than 15 minutes.

Which combination of steps will meet these requirements MOST cost-effectively? (Choose two.)

A. Use an AWS Lambda function and the Athena Boto3 client start_query_execution API call to invoke the Athena queries programmatically.

B. Create an AWS Step Functions workflow that invokes the Lambda function, uses a Wait state between Athena get_query_execution status checks, and invokes the next query after the current query finishes successfully.

C. Use an AWS Glue Python shell job and the Athena Boto3 client start_query_execution API call to invoke the Athena queries programmatically.

D. Use an AWS Glue Python shell script to run a sleep timer that checks every 5 minutes to determine whether the current Athena query has finished running successfully. Configure the Python shell script to invoke the next query when the current query has finished running.

E. Use Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to orchestrate the Athena queries in AWS Batch.

ANSWER: A B

Explanation:

Using an AWS Lambda function and the Athena Boto3 client `start_query_execution` API call is appropriate because Athena query execution is asynchronous. The function submits a query and receives a query execution ID without remaining active for the entire query duration. This keeps the Lambda invocation short, avoids the 15-minute Lambda maximum timeout, and provides the identifier that the orchestration workflow needs to track each query. Athena documents `StartQueryExecution` as the operation that starts an asynchronous query execution.

A Step Functions workflow can then manage the sequence reliably. After the Lambda task submits an Athena query, the workflow can use a Wait state between status checks and invoke logic that calls Athena's `GetQueryExecution` API. Once the status indicates that the current query has completed successfully, the workflow proceeds to submit the next query. Step Functions preserves workflow state during waits, so no compute process must run continuously while an Athena query executes. This makes it well suited to coordinating queries that can exceed Lambda's runtime limit while maintaining ordered execution. The daily invocation can be initiated by an event-based schedule, while this workflow handles the query lifecycle. See Athena StartQueryExecution and AWS Step Functions Wait state.

QUESTION NO: 24

A company is designing a serverless data processing workflow in AWS Step Functions that involves multiple steps. The processing workflow ingests data from an external API, transforms the data by using multiple AWS Lambda functions, and loads the transformed data into Amazon DynamoDB.

The company needs the workflow to perform specific steps based on the content of the incoming data.

Which Step Functions state type should the company use to meet this requirement?

A. Parallel

B. Choice

C. Task

D. Map

ANSWER: B

Explanation:

The **Choice** state is designed for conditional branching in an AWS Step Functions state machine. It evaluates the workflow's JSON input against one or more rules and routes execution to the appropriate next state based on the condition that matches. This enables the workflow to inspect content received from the external API and select the required transformation path before writing the resulting data to DynamoDB.

A Choice state can evaluate values using JSONPath-based comparison operators, including string, numeric, Boolean, timestamp, and existence checks. It can also combine conditions with logical operations such as `And`, `Or`, and `Not`. For example, the state machine can direct one kind of incoming record to a particular Lambda transformation and route another kind to a different processing sequence. A default transition can be configured to handle input that does not match any specified rule, ensuring predictable workflow behavior. This keeps orchestration and decision logic declarative within the state machine definition rather than requiring a Lambda function solely to decide the next processing step.

AWS documents this conditional routing behavior in its Choice workflow state documentation and lists Choice among the supported Step Functions state types.

QUESTION NO: 25

A company uses Amazon Redshift for analytics. A large fact table is frequently joined to a small dimension table by using the customer_id column. The data engineer notices high data redistribution during the joins.

Which actions will reduce data movement during the joins? (Choose two.)

A. Set customer_id as the distribution key for both the fact table and the dimension table.

B. Use distribution style ALL for the small dimension table.

C. Use distribution style EVEN for both tables.

D. Set the same sort key on all columns in both tables.

E. Use distribution style ALL for the large fact table.

ANSWER: A B

Explanation:

Setting `customer_id` as the distribution key on both tables colocates rows that have the same customer ID on the same compute node. When the tables are joined on the common distribution key, Amazon Redshift can perform more of the join locally and can avoid redistributing matching rows across nodes. This is appropriate when both tables are large enough that distributing them by the join key is beneficial.

Using distribution style ALL for the small dimension table is also appropriate. Redshift replicates a table with distribution style ALL to every compute node. Each node can then join its local fact-table rows with a local copy of the dimension table, avoiding redistribution of the dimension table during the join. This strategy is best for a relatively small table that is joined frequently. See Amazon Redshift data distribution for query optimization.

QUESTION NO: 26

A company is building a data stream processing application. The application runs in an Amazon Elastic Kubernetes Service (Amazon EKS) cluster. The application stores processed data in an Amazon DynamoDB table.

The company needs the application containers in the EKS cluster to have secure access to the DynamoDB table. The company does not want to embed AWS credentials in the containers. Which solution will meet these requirements?

A. Store the AWS credentials in an Amazon S3 bucket. Grant the EKS containers access to the S3 bucket to retrieve the credentials.

B. Attach an IAM role to the EKS worker nodes. Grant the IAM role access to DynamoDUse the IAM role to set up IAM roles service accounts (IRSA) functionality.

C. Create an IAM user that has an access key to access the DynamoDB table. Use environment variables in the EKS containers to store the IAM user access key data.

D. Create an IAM user that has an access key to access the DynamoDB table. Use Kubernetes secrets that are mounted in a volume of the EKS cluster nodes to store the user access key data.

E. Configure IAM roles for service accounts (IRSA) for the application pods. Grant the associated IAM role access to the DynamoDB table.

ANSWER: E

Explanation:

Configure IAM roles for service accounts (IRSA) for the application pods. Grant the associated IAM role access to the DynamoDB table. This approach provides the Kubernetes workload with temporary AWS credentials without placing long-term access keys, secrets, or credentials in the container image, environment variables, or Kubernetes configuration. With IRSA, the EKS cluster uses an OpenID Connect (OIDC) identity provider and AWS Security Token Service (AWS STS) to let

a pod assume an IAM role associated with the Kubernetes service account that the pod uses. The application then obtains credentials automatically through the standard AWS SDK credential provider chain.

The IAM role should have a trust policy scoped to the EKS cluster's OIDC provider and the specific Kubernetes service account. Its permissions policy should follow least privilege by allowing only the DynamoDB API actions and table resources required by this stream-processing workload. This creates workload-level isolation: the application pods receive only their intended DynamoDB permissions, independently of other pods running on the same worker nodes. AWS documents IRSA as the mechanism for assigning AWS permissions directly to Kubernetes service accounts and workloads in Amazon EKS. See IAM roles for service accounts and Assign an IAM role to a Kubernetes service account.

QUESTION NO: 27

An airline company is collecting metrics about flight activities for analytics. The company is conducting a proof of concept (POC) test to show how analytics can provide insights that the company can use to increase on-time departures.

The POC test uses objects in Amazon S3 that contain the metrics in .csv format. The POC test uses Amazon Athena to query the data. The data is partitioned in the S3 bucket by date.

As the amount of data increases, the company wants to optimize the storage solution to improve query performance.

Which combination of solutions will meet these requirements? (Choose two.)

A. Add a randomized string to the beginning of the keys in Amazon S3 to get more throughput across partitions.

B. Use an S3 bucket that is in the same account that uses Athena to query the data.

C. Use an S3 bucket that is in the same AWS Region where the company runs Athena queries.

D. Preprocess the .csv data to JSON format by fetching only the document keys that the query requires.

E. Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates.

ANSWER: C E

Explanation:

Use an S3 bucket that is in the same AWS Region where the company runs Athena queries. Athena queries data directly in Amazon S3, and the S3 data source must be in the same Region as the Athena query processing Region. Keeping the data and query workload colocated provides the supported architecture and avoids cross-Region data-access considerations as the analytics workload grows. The AWS documentation describes Athena as querying data in S3 and notes the Region requirements for data sources and query results.

Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates. Parquet is a compressed, columnar format, so Athena can read only the columns requested by a query rather than scanning every field in each CSV row. Parquet files also contain metadata that allows predicate pushdown: when a filter applies, Athena can avoid reading row groups whose statistics show that they cannot contain matching values. Together with the existing date partitioning, this reduces bytes scanned, lowers query cost, and improves performance as flight-metric data volume increases. AWS recommends columnar formats such as Parquet or ORC and predicate pushdown as Athena data-optimization techniques. See What is Amazon Athena? and Amazon Athena data optimization techniques.

QUESTION NO: 28

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 splittable.

ANSWER: B C

Explanation:

Use a columnar storage file format. and **Partition the data based on the most common query predicates.** provide the best query performance for Amazon Redshift Spectrum. Columnar formats, including Apache Parquet and ORC, organize values by column rather than by row. This allows Spectrum to read only the columns referenced by a query, reducing the volume of data retrieved from Amazon S3. These formats also support efficient compression, lowering storage I/O and improving scan performance for analytical workloads.

Partitioning external data on columns frequently used in query filters, such as event date, Region, or business unit, enables partition pruning. When a query includes a predicate on a partition column, Spectrum can identify the relevant S3 partition locations and avoid scanning data in unrelated partitions. This is especially important for large data lakes, where avoiding unnecessary S3 reads directly reduces query duration and resource consumption. AWS recommends combining columnar storage with partitioning aligned to common access patterns to minimize data scanned by Spectrum and maximize parallel query efficiency. See Amazon Redshift Spectrum performance best practices and Amazon Redshift Spectrum external tables.

QUESTION NO: 29

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

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

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

A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost-optimized storage classes. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classes. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.

B. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequently. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.

C. Use S3 Intelligent-Tiering. Activate the Deep Archive Access tier.

D. Use S3 Intelligent-Tiering. Use the default access tier.

ANSWER: D

Explanation:

Use S3 Intelligent-Tiering. Use the default access tier. S3 Intelligent-Tiering is designed for objects with unknown, changing, or unpredictable access patterns. It continuously monitors access at the object level and automatically moves eligible objects between the Frequent Access, Infrequent Access, and Archive Instant Access tiers as their access patterns change. These automatic movements remove the need to analyze access patterns, build and repeatedly tune lifecycle rules, or manage transitions separately across thousands of buckets.

The milliseconds retrieval requirement is met because all of the default automatic access tiers provide immediate, millisecond retrieval. Data that has not been accessed for an extended period can therefore be stored at a lower cost in an applicable Intelligent-Tiering tier while remaining immediately available to analytics workloads. The optional asynchronous archive tiers are not needed for this requirement. Intelligent-Tiering also automatically returns an object to the Frequent Access tier when it is accessed, ensuring that storage placement adapts as workload behavior changes. This provides cost optimization for petabyte-scale data while imposing minimal operational administration. See the Amazon S3 storage class overview and S3 Intelligent-Tiering documentation.

QUESTION NO: 30

A data engineer develops an AWS Glue Apache Spark ETL job to perform transformations on a dataset. When the data engineer runs the job, the job returns an error that reads, " No space left on device. "

The data engineer needs to identify the source of the error and provide a solution.

Which combinations of steps will meet this requirement MOST cost-effectively? (Select TWO.)

A. Scale out the workers vertically to address data skewness.

B. Use the Spark UI and AWS Glue metrics to monitor data skew in the Spark executors.

Monitor using Spark UI and Glue metrics to find skewed partitions or executor issues.

C. Scale out the number of workers horizontally to address data skewness.

D. Enable the --write-shuffle-files-to-s3 job parameter. Use the salting technique.

Use --write-shuffle-files-to-s3 to offload intermediate data to S3 instead of local disk, and apply salting to reduce skew.
“You can reduce the impact of data skew and large shuffle operations by monitoring with Spark UI and enabling the --write-shuffle-files-to-s3 option. Salting can help rebalance the skewed keys.”
– Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf
Scaling out workers (A, C) is more costly and less efficient if the root cause (skew) is not fixed.

E. Use error logs in Amazon CloudWatch to monitor data skew.

ANSWER: B D

Explanation:

“Use the Spark UI and AWS Glue metrics to monitor data skew in the Spark executors” is correct because local-disk exhaustion in a Spark ETL job commonly results from a shuffle operation that produces excessive intermediate data or from uneven key distribution that causes one or a few executors to process disproportionately large partitions. The AWS Glue Spark UI provides visibility into stages, task durations, shuffle read and write activity, and spill behavior, allowing the engineer to identify the stage and partitions responsible for the pressure. Glue observability metrics complement this investigation with job- and executor-level operational signals.

“Enable the --write-shuffle-files-to-s3 job parameter. Use the salting technique.” is correct because it addresses both the local-storage constraint and the skew condition without first increasing worker capacity. The `--write-shuffle-files-to-s3` setting stores shuffle files in Amazon S3 rather than relying solely on executor local disk, reducing the likelihood of disk-space failures during large shuffles. Salting distributes records associated with highly frequent keys across multiple partitions, balancing task workloads and reducing the oversized shuffle partitions that trigger excessive spill and storage use. This diagnostic-and-remediation combination is generally more cost-effective than scaling resources before identifying and correcting the underlying skew. See Monitoring AWS Glue jobs with the Spark UI and AWS Glue observability metrics.