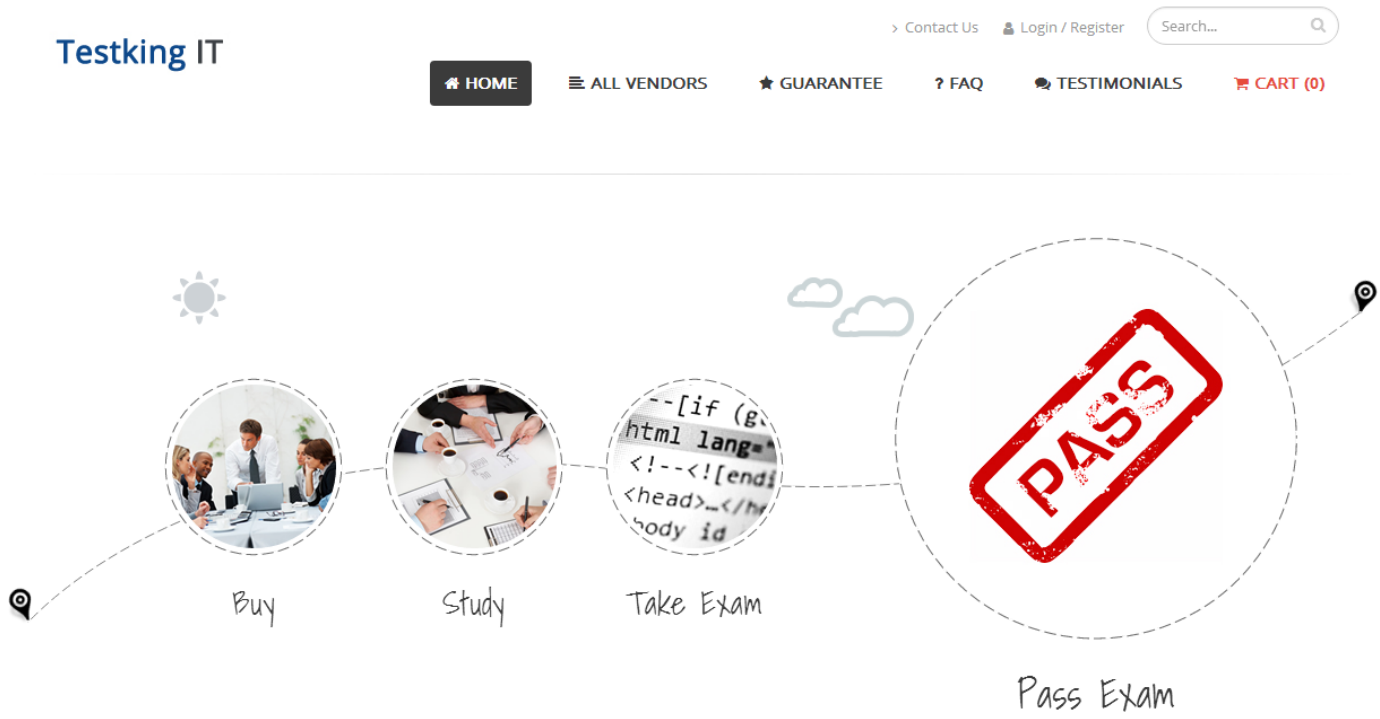
TestkingIT



Try **Desktop Test Engine** before you buy

We're not the only ones **happy** about TestKingsIT Practice Material ...

48236+ customers in 100+ countries use TestKingsIT Test Engine. Meet our customers.

iMessenger VOREED GetCustom JET ORANGE iCompany Paradoxx

http://www.testkingit.com/

Latest practice material - Exam Cram - TestKingIT

Exam : **Data-Engineer-Associate-KR**

Title : AWS Certified Data Engineer - Associate (DEA -C01) (Data-Engineer-Associate Korean Version)

Vendor : Amazon

Version : DEMO

QUESTION NO: 1
데이터 엔지니어는 AWS Glue를 사용하여 센서에서 스트리밍 데이터를 처리하는 추출, 변환 및 로드(ETL) 파이프라인을 구축하고 있습니다. 이 파이프라인은 데이터를 거의 실시간으로 Amazon S3 버킷으로 전송합니다. 또한 데이터 엔지니어는 변환 작업을 수행하고 수신 데이터를 Amazon RDS for PostgreSQL 데이터베이스에 저장된 메타데이터와 결합해야 합니다. 데이터 엔지니어는 결과를 Apache Parquet 형식으로 두 번째 S3 버킷에 다시 기록해야 합니다.
어떤 솔루션이 이러한 요구 사항을 충족할까요?
A. AWS Glue 스트리밍 작업과 AWS Glue Studio를 사용하여 변환을 수행하고 Parquet 형식으로 데이터를 씁니다.
B. AWS Glue 작업과 AWS Glue 데이터 카탈로그를 사용하여 Amazon S3 및 Amazon RDS의 데이터를 카탈로그화합니다. 변환 및 조인을 수행하고 Parquet 형식으로 출력을 작성하도록 작업을 구성합니다.
C. AWS Glue 대화형 세션을 사용하여 스트리밍 데이터를 처리하고 RDS 데이터베이스와 데이터를 조인합니다.
D. AWS Glue Python 셸 작업을 사용하여 데이터를 일괄 처리하는 Python 스크립트를 실행합니다. AWS Glue 북마크를 사용하여 처리된 파일을 추적합니다.
Answer: A
Explanation:
* AWS Glue streaming jobs are purpose-built for low-latency, near real-time ETL pipelines.
* Glue Studio allows visual job authoring and supports complex transformations, including joins with Amazon RDS databases, through JDBC connectors.
* Data can be written directly to S3 in Apache Parquet format using built-in AWS Glue support.
Options B and C are more suitable for batch or development/testing. Option D is for custom batch jobs, not real-time processing.
"You can use AWS Glue streaming jobs with Glue Studio to process streaming data, perform transformations, join with RDS data sources, and write results in formats like Parquet to S3."
Reference: AWS Glue Studio and Streaming Jobs - AWS Documentation

QUESTION NO: 2
데이터 엔지니어가 Amazon S3 버킷에서 데이터를 읽도록 AWS Glue 작업을 구성하고 있습니다. 데이터 엔지니어는 필요한 AWS Glue 연결 세부 정보와 관련 IAM 역할을 설정했습니다. 그러나 데이터 엔지니어가 AWS Glue 작업을 실행하려고 하면 데이터 엔지니어는 Amazon S3 VPC 게이트웨이 엔드포인트에 문제가 있음을 나타내는 오류 메시지를 수신합니다.
데이터 엔지니어는 오류를 해결하고 AWS Glue 작업을 S3 버킷에 연결해야 합니다.
이 요구 사항을 충족하는 솔루션은 무엇입니까?
A. Amazon S3 VPC 게이트웨이 엔드포인트로부터의 인바운드 트래픽을 허용하도록 AWS Glue 보안 그룹을 업데이트합니다.
B. S3 버킷에 액세스할 수 있는 AWS Glue 작업 권한을 명시적으로 부여하도록 S3 버킷 정책을 구성합니다.
C. AWS Glue 작업 코드를 검토하여 AWS Glue 연결 세부 정보에 정규화된 도메인 이름이 포함되어 있는지 확인합니다.

D. VPC의 라우팅 테이블에 Amazon S3 VPC 게이트웨이 엔드포인트에 대한 인바운드 및 아웃바운드 경로가 포함되어 있는지 확인합니다.

Answer: D

Explanation:

The error message indicates that the AWS Glue job cannot access the Amazon S3 bucket through the VPC endpoint. This could be because the VPC's route table does not have the necessary routes to direct the traffic to the endpoint. To fix this, the data engineer must verify that the route table has an entry for the Amazon S3 service prefix (com.amazonaws.region.s3) with the target as the VPC endpoint ID. This will allow the AWS Glue job to use the VPC endpoint to access the S3 bucket without going through the internet or a NAT gateway. For more information, see Gateway endpoints. References:

Troubleshoot the AWS Glue error "VPC S3 endpoint validation failed"

Amazon VPC endpoints for Amazon S3

[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

QUESTION NO: 3

전자상거래 회사가 AWS를 사용하여 온프레미스 환경에서 AWS 클라우드로 데이터 파이프라인을 마이그레이션하려고 합니다. 이 회사는 현재 온프레미스 환경에서 타사를 사용하여 데이터 수집 프로세스를 조율하고 있습니다.

이 회사는 서버를 관리할 필요가 없는 마이그레이션 솔루션을 원합니다. 이 솔루션은 Python 및 Bash 스크립트를 조정할 수 있어야 합니다. 이 솔루션은 회사가 코드를 리팩토링할 필요가 없어야 합니다.

어떤 솔루션이 운영 비용을 최소화하면서 이러한 요구 사항을 충족할 수 있을까요?

A. AWS 람다

B. Apache Airflow를 위한 Amazon 관리 워크플로(Amazon MWAA)

C. AWS Step Functions

D. AWS Glue

Answer: B

Explanation:

The ecommerce company wants to migrate its data pipelines into the AWS Cloud without managing servers, and the solution must orchestrate Python and Bash scripts without refactoring code. Amazon Managed Workflows for Apache Airflow (Amazon MWAA) is the most suitable solution for this scenario.

Option B: Amazon Managed Workflows for Apache Airflow (Amazon MWAA)MWAA is a managed orchestration service that supports Python and Bash scripts via Directed Acyclic Graphs (DAGs) for workflows. It is a serverless, managed version of Apache Airflow, which is commonly used for orchestrating complex data workflows, making it an ideal choice for migrating existing pipelines without refactoring. It supports Python, Bash, and other scripting languages, and the company would not need to manage the underlying infrastructure.

Other options:

AWS Lambda (Option A) is more suited for event-driven workflows but would require breaking down the pipeline into individual Lambda functions, which may require refactoring.

AWS Step Functions (Option C) is good for orchestration but lacks native support for Python and Bash without using Lambda functions, and it may require code changes.

AWS Glue (Option D) is an ETL service primarily for data transformation and not suitable for

orchestrating general scripts without modification.
References:
Amazon Managed Workflows for Apache Airflow (MWAA) Documentation

QUESTION NO: 4
애플리케이션에서 관리형 런타임으로 구성된 AWS Lambda 함수를 사용합니다. 이 Lambda 함수는 기본 Amazon CloudWatch Logs 로그 그룹에 로그를 성공적으로 기록합니다. 데이터 엔지니어는 애플리케이션 로그에는 ERROR 레벨 로그만, 시스템 로그에는 WARN 레벨 로그만 표시하도록 로깅 동작을 변경하려고 합니다.
어떤 솔루션이 이러한 요구 사항을 충족할까요?
A. Lambda 실행 역할에 추가 권한을 추가합니다.
B. Lambda 함수 코드에서 로그 레벨을 ERROR로 설정합니다.
C. Lambda 함수가 JSON 로그 형식을 사용하도록 구성합니다.
D. Lambda 함수가 사용자 지정 로그 그룹으로 로그를 전송하도록 구성합니다.
Answer: C
Explanation:
Option C is correct because AWS Lambda's advanced logging controls support separate application log level and system log level filtering, but AWS documentation states that for Lambda to filter application logs according to their log level, the function must use JSON formatted logs. AWS also documents that in the advanced logging configuration you can choose a log level such as ERROR for application logs and WARN for system logs. Therefore, configuring the function to use JSON log format is the necessary step that enables the required log-level filtering behavior.
Option A is irrelevant because the function already writes logs successfully. Option B alone is insufficient because changing code-level logging does not configure Lambda's separate system log filtering behavior, and Lambda's application log filtering relies on structured JSON logs. Option D changes the log destination but does not implement the required filtering levels. The official Lambda documentation makes clear that JSON log format is the enabling configuration for this feature, so that is the correct answer.

QUESTION NO: 5
회사는 Amazon S3 버킷에 JSON 형식과 .csv 형식으로 데이터 세트를 저장합니다. 이 회사는 Microsoft SQL Server 데이터베이스용 Amazon RDS, 프로비저닝된 용량 모드의 Amazon DynamoDB 테이블, Amazon Redshift 클러스터를 보유하고 있습니다. 데이터 엔지니어링 팀은 데이터 과학자가 SQL과 유사한 구문을 사용하여 모든 데이터 소스를 쿼리할 수 있는 기능을 제공하는 솔루션을 개발해야 합니다.
최소한의 운영 오버헤드로 이러한 요구 사항을 충족하는 솔루션은 무엇입니까?
A. AWS Glue를 사용하여 데이터 소스를 크롤링합니다. AWS Glue 데이터 카탈로그에 메타데이터를 저장합니다. Amazon Athena를 사용하여 데이터를 쿼리합니다. 구조화된 데이터 소스에는 SQL을 사용합니다. JSON 형식으로 저장된 데이터에는 PartiQL을 사용합니다.
B. AWS Glue를 사용하여 데이터 소스를 크롤링합니다. AWS Glue 데이터 카탈로그에 메타데이터를 저장합니다. Redshift Spectrum을 사용하여 데이터를 쿼리합니다. 구조화된 데이터 소스에는 SQL을 사용합니다. JSON 형식으로 저장된 데이터에는 PartiQL을 사용합니다.

C. AWS Glue를 사용하여 데이터 소스를 크롤링합니다. AWS Glue 데이터 카탈로그에 메타데이터를 저장합니다. AWS Glue 작업을 사용하여 JSON 형식의 데이터를 Apache Parquet 또는 .csv 형식으로 변환합니다. 변환된 데이터를 S3 버킷에 저장합니다. Amazon Athena를 사용하여 S3 버킷에서 원본 데이터와 변환된 데이터를 쿼리합니다.

D. AWS Lake Formation을 사용하여 데이터 레이크를 생성합니다. Lake Formation 작업을 사용하여 모든 데이터 소스의 데이터를 Apache Parquet 형식으로 변환합니다. 변환된 데이터를 S3 버킷에 저장합니다. Amazon Athena 또는 Redshift Spectrum을 사용하여 데이터를 쿼리합니다.

Answer: A

Explanation:

The best solution to meet the requirements of giving data scientists the ability to query all data sources by using syntax similar to SQL with the least operational overhead is to use AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use Amazon Athena to query the data, use SQL for structured data sources, and use PartiQL for data that is stored in JSON format.

AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores1. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog2. The Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components3. You can use AWS Glue to crawl the data sources, such as Amazon S3, Amazon RDS for Microsoft SQL Server, and Amazon DynamoDB, and store the metadata in the Data Catalog.

Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python4. Amazon Athena also supports PartiQL, a SQL-compatible query language that lets you query, insert, update, and delete data from semi-structured and nested data, such as JSON. You can use Amazon Athena to query the data from the Data Catalog using SQL for structured data sources, such as .csv files and relational databases, and PartiQL for data that is stored in JSON format. You can also use Athena to query data from other data sources, such as Amazon Redshift, using federated queries.

Using AWS Glue and Amazon Athena to query all data sources by using syntax similar to SQL is the least operational overhead solution, as you do not need to provision, manage, or scale any infrastructure, and you pay only for the resources you use. AWS Glue charges you based on the compute time and the data processed by your crawlers and ETL jobs1. Amazon Athena charges you based on the amount of data scanned by your queries. You can also reduce the cost and improve the performance of your queries by using compression, partitioning, and columnar formats for your data in Amazon S3.

Option B is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, and use Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena. Redshift Spectrum is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to query and join data across your data warehouse and your data lake using standard SQL. While Redshift Spectrum is powerful and useful for many data warehousing scenarios, it is not necessary or cost-effective for querying all data sources by using syntax similar to SQL.

Redshift Spectrum charges you based on the amount of data scanned by your queries, which is similar to Amazon Athena, but it also requires you to have an Amazon Redshift cluster, which charges you based on the node type, the number of nodes, and the duration of the cluster5. These costs can add up quickly, especially if you have large volumes of data and complex queries. Moreover, using Redshift Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, and create an external schema and database for the data in the Data Catalog, instead of querying it directly from Amazon Athena.
Option C is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format, store the transformed data in an S3 bucket, and use Amazon Athena to query the original and transformed data from the S3 bucket, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Glue jobs are ETL scripts that you can write in Python or Scala to transform your data and load it to your target data store. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes6. While using AWS Glue jobs and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to write, run, and monitor the ETL jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using AWS Glue jobs and Parquet would introduce additional latency, as you would have to wait for the ETL jobs to finish before querying the transformed data.
Option D is not the best solution, as using AWS Lake Formation to create a data lake, use Lake Formation jobs to transform the data from all data sources to Apache Parquet format, store the transformed data in an S3 bucket, and use Amazon Athena or Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Lake Formation is a service that helps you centrally govern, secure, and globally share data for analytics and machine learning7. Lake Formation jobs are ETL jobs that you can create and run using the Lake Formation console or API. While using Lake Formation and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to create, run, and monitor the Lake Formation jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using Lake Formation and Parquet would introduce additional latency, as you would have to wait for the Lake Formation jobs to finish before querying the transformed data.
Furthermore, using Redshift Spectrum to query the data would also incur the same costs and complexity as mentioned in option B.
References:
What is Amazon Athena?
Data Catalog and crawlers in AWS Glue
AWS Glue Data Catalog
Columnar Storage Formats
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
AWS Glue Schema Registry

What is AWS Glue?
Amazon Redshift Serverless
Amazon Redshift provisioned clusters
[Querying external data using Amazon Redshift Spectrum]
[Using stored procedures in Amazon Redshift]
[What is AWS Lambda?]
[PartiQL for Amazon Athena]
[Federated queries in Amazon Athena]
[Amazon Athena pricing]
[Top 10 performance tuning tips for Amazon Athena]
[AWS Glue ETL jobs]
[AWS Lake Formation jobs]

QUESTION NO: 6
한 회사가 Amazon Redshift에 데이터 웨어하우스를 두고 있습니다. 보안 규정을 준수하기 위해 회사는 데이터 웨어하우스에 대한 모든 사용자 활동과 연결 활동을 기록하고 저장해야 합니다.
어떤 솔루션이 이러한 요구 사항을 충족시킬까요?

A. Amazon S3 버킷을 만듭니다. Amazon Redshift 클러스터에 대한 로깅을 활성화합니다. 로깅 구성에서 로그를 저장할 S3 버킷을 지정합니다.

B. Amazon Elastic File System(Amazon EFS) 파일 시스템을 만듭니다. Amazon Redshift 클러스터에 대한 로깅을 활성화합니다. EFS 파일 시스템에 로그를 씁니다.

C. Amazon Aurora MySQL 데이터베이스를 만듭니다. Amazon Redshift 클러스터에 대한 로깅을 활성화합니다. Aurora MySQL 데이터베이스의 테이블에 로그를 씁니다.

D. Amazon Elastic Block Store(Amazon EBS) 볼륨을 생성합니다. Amazon Redshift 클러스터에 대한 로깅을 활성화합니다. EBS 볼륨에 로그를 씁니다.

Answer: A
Explanation:
Problem Analysis:
The company must log all user activities and connection activities in Amazon Redshift for security compliance.
Key Considerations:
Redshift supports audit logging, which can be configured to write logs to an S3 bucket.
S3 provides durable, scalable, and cost-effective storage for logs.
Solution Analysis:
Option A: S3 for Logging
Standard approach for storing Redshift logs.
Easy to set up and manage with minimal cost.
Option B: Amazon EFS
EFS is unnecessary for this use case and less cost-efficient than S3.
Option C: Aurora MySQL
Using a database to store logs increases complexity and cost.
Option D: EBS Volume
EBS is not a scalable option for log storage compared to S3.
Final Recommendation:

Enable Redshift audit logging and specify an S3 bucket as the destination.
Amazon Redshift Audit Logging
Storing Logs in Amazon S3

QUESTION NO: 7
데이터 엔지니어가 Amazon QuickSight를 사용하여 여러 AWS 리전에서 발생하는 회사 매출을 보고하는 대시보드를 구축하고 있습니다. 이 엔지니어는 시각화에서 표시되는 드릴다운 수준과 관계없이 모든 리전의 총 매출을 대시보드에 표시하기를 원합니다.
어떤 솔루션이 이러한 요구 사항을 충족할까요?
A. 테이블 계산을 생성합니다.
B. 간단한 계산 필드를 생성합니다.
C. 레벨 인식 계산 - 집계(LAC-A) 함수를 생성합니다.
D. 레벨 인식 계산 윈도우(LAC-W) 함수를 생성합니다.
Answer: C
Explanation:
Option C (LAC-A) is the correct choice because the requirement is to always show the Region-level total even when the visual is drilled down into lower levels (for example, country # city # store). A simple calculated field (Option B) is computed at the row level and then aggregated by the visual, so it will change as the drill-down changes the grain. A table calculation (Option A) is evaluated based on the current visual layout and can vary with the fields placed in the visual, which makes it unreliable for enforcing a fixed "Region total regardless of drill-down."
A level-aware calculation (aggregate) is specifically intended to "lock" an aggregation to a chosen dimensional level (here: Region). That means you can compute revenue aggregated at the Region level and reuse that value across lower drill levels without it recalculating at city/store granularity. A window LAC (LAC-W) is primarily for windowed analytics (running totals, period-over-period, rank, moving averages) over a partition/order, not for enforcing a fixed dimensional aggregation level. Therefore, LAC-A best matches the requirement.

QUESTION NO: 8
데이터 엔지니어는 온프레미스 데이터 센터에서 Amazon S3 버킷으로 5TB의 데이터를 안전하게 전송해야 합니다. 매일 약 5%의 데이터가 변경됩니다. 데이터 업데이트는 정기적으로 S3 버킷으로 확산되어야 합니다. 데이터에는 다양한 형식의 파일이 포함됩니다. 데이터 엔지니어는 전송 프로세스를 자동화해야 하며 프로세스가 정기적으로 실행되도록 예약해야 합니다.
데이터 엔지니어는 운영상 가장 효율적인 방식으로 데이터를 전송하기 위해 어떤 AWS 서비스를 사용해야 합니까?
A. AWS DataSync
B. AWS Glue
C. AWS Direct Connect
D. Amazon S3 전송 가속
Answer: A
Explanation:
AWS DataSync is an online data movement and discovery service that simplifies and accelerates data migrations to AWS as well as moving data to and from on-premises storage,

edge locations, other cloud providers, and AWS Storage services1. AWS DataSync can copy data to and from various sources and targets, including Amazon S3, and handle files in multiple formats. AWS DataSync also supports incremental transfers, meaning it can detect and copy only the changes to the data, reducing the amount of data transferred and improving the performance. AWS DataSync can automate and schedule the transfer process using triggers, and monitor the progress and status of the transfers using CloudWatch metrics and events1.
AWS DataSync is the most operationally efficient way to transfer the data in this scenario, as it meets all the requirements and offers a serverless and scalable solution. AWS Glue, AWS Direct Connect, and Amazon S3 Transfer Acceleration are not the best options for this scenario, as they have some limitations or drawbacks compared to AWS DataSync. AWS Glue is a serverless ETL service that can extract, transform, and load data from various sources to various targets, including Amazon S32. However, AWS Glue is not designed for large-scale data transfers, as it has some quotas and limits on the number and size of files it can process3.
AWS Glue also does not support incremental transfers, meaning it would have to copy the entire data set every time, which would be inefficient and costly.
AWS Direct Connect is a service that establishes a dedicated network connection between your on-premises data center and AWS, bypassing the public internet and improving the bandwidth and performance of the data transfer. However, AWS Direct Connect is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS DataSync, AWS Storage Gateway, or AWS CLI. AWS Direct Connect also has some hardware and location requirements, and charges you for the port hours and data transfer out of AWS.
Amazon S3 Transfer Acceleration is a feature that enables faster data transfers to Amazon S3 over long distances, using the AWS edge locations and optimized network paths. However, Amazon S3 Transfer Acceleration is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS CLI, AWS SDK, or third-party software. Amazon S3 Transfer Acceleration also charges you for the data transferred over the accelerated endpoints, and does not guarantee a performance improvement for every transfer, as it depends on various factors such as the network conditions, the distance, and the object size. References:
AWS DataSync
AWS Glue
AWS Glue quotas and limits
[AWS Direct Connect]
[Data transfer options for AWS Direct Connect]
[Amazon S3 Transfer Acceleration]
[Using Amazon S3 Transfer Acceleration]

QUESTION NO: 9
데이터 엔지니어는 Amazon Athena를 사용하여 Amazon S3에 있는 판매 데이터를 분석하고 있습니다. 데이터 엔지니어는 sales_data라는 테이블에서 여러 제품에 대한 2023년 판매량을 검색하는 쿼리를 작성합니다. 그러나 쿼리는 sales_data 테이블에 있는 모든 제품에 대한 결과를 반환하지 않습니다. 문제를 해결하려면 데이터 엔지니어가 쿼리 문제를 해결해야 합니다.

데이터 엔지니어의 원래 쿼리는 다음과 같습니다.
SELECT 제품명, 합계(매출_금액)
sales_data에서
연도 = 2023
GROUP BY 제품_이름
데이터 엔지니어는 이러한 요구 사항을 충족하기 위해 Athena 쿼리를 어떻게 수정해야 합니까?

A. 집계를 위해 sum(판매액)을 count(*J)로 바꿉니다.

B. WHERE 연도 = 2023을 WHERE extractlyear FROM 판매 데이터) = 2023으로 변경합니다.

C. GROUP BY 절 뒤에 HAVING sumfsales amount) > 0을 추가합니다.

D. GROUP BY 절을 제거합니다.

Answer: B

Explanation:
The original query does not return results for all of the products because the year column in the sales_data table is not an integer, but a timestamp. Therefore, the WHERE clause does not filter the data correctly, and only returns the products that have a null value for the year column. To fix this, the data engineer should use the extract function to extract the year from the timestamp and compare it with 2023. This way, the query will return the correct results for all of the products in the sales_data table. The other options are either incorrect or irrelevant, as they do not address the root cause of the issue. Replacing sum with count does not change the filtering condition, adding HAVING clause does not affect the grouping logic, and removing the GROUP BY clause does not solve the problem of missing products.
References:
Troubleshooting JSON queries - Amazon Athena (Section: JSON related errors) When I query a table in Amazon Athena, the TIMESTAMP result is empty (Section: Resolution) AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 7, page 197)

QUESTION NO: 10
금융 서비스 회사는 Amazon Redshift에 금융 데이터를 저장합니다. 데이터 엔지니어는 웹 기반 거래 애플리케이션을 지원하기 위해 금융 데이터에 대해 실시간 쿼리를 실행하려고 합니다. 데이터 엔지니어는 거래 애플리케이션 내에서 쿼리를 실행하려고 합니다.
최소한의 운영 오버헤드로 이러한 요구 사항을 충족하는 솔루션은 무엇입니까?

A. Amazon Redshift에 대한 WebSocket 연결을 설정합니다.

B. Amazon Redshift 데이터 API를 사용합니다.

C. Amazon Redshift에 대한 JDBC(Java Database Connectivity) 연결을 설정합니다.

D. 자주 액세스하는 데이터를 Amazon S3에 저장합니다. Amazon S3 Select를 사용하여 쿼리를 실행합니다.

Answer: B

Explanation:
The Amazon Redshift Data API is a built-in feature that allows you to run SQL queries on Amazon Redshift data with web services-based applications, such as AWS Lambda, Amazon SageMaker notebooks, and AWS Cloud9. The Data API does not require a persistent connection to your database, and it provides a secure HTTP endpoint and integration with AWS SDKs. You can use the endpoint to run SQL statements without managing connections.

The Data API also supports both Amazon Redshift provisioned clusters and Redshift Serverless workgroups. The Data API is the best solution for running real-time queries on the financial data from within the trading application, as it has the least operational overhead compared to the other options.
Option A is not the best solution, as establishing WebSocket connections to Amazon Redshift would require more configuration and maintenance than using the Data API. WebSocket connections are also not supported by Amazon Redshift clusters or serverless workgroups.
Option C is not the best solution, as setting up JDBC connections to Amazon Redshift would also require more configuration and maintenance than using the Data API. JDBC connections are also not supported by Redshift Serverless workgroups.
Option D is not the best solution, as storing frequently accessed data in Amazon S3 and using Amazon S3 Select to run the queries would introduce additional latency and complexity than using the Data API. Amazon S3 Select is also not optimized for real-time queries, as it scans the entire object before returning the results. References:
Using the Amazon Redshift Data API
Calling the Data API
Amazon Redshift Data API Reference
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

QUESTION NO: 11
한 회사가 Amazon Redshift 테이블에 민감한 데이터를 저장합니다. 회사는 특정 사용자에게 민감한 데이터에 접근할 수 있는 권한을 부여해야 하며, 데이터에 중복을 생성해서는 안 됩니다.
고객 지원 담당자는 민감한 데이터의 마지막 네 글자를 볼 수 있어야 합니다. 감사 담당자는 민감한 데이터의 전체 값을 볼 수 있어야 합니다. 다른 사용자는 민감한 정보에 접근할 수 없습니다.
어떤 솔루션이 이러한 요구 사항을 충족시킬까요?
A. 각 사용자 역할에 따라 액세스를 허용하는 동적 데이터 마스킹 정책을 생성합니다. 특정 액세스 권한을 가진 IAM 역할을 생성합니다. 민감한 데이터가 포함된 열에 마스킹 정책을 연결합니다.
B. Redshift 클러스터에서 메타데이터 보안을 활성화합니다. 고객 지원 사용자와 감사 사용자에 대한 IAM 사용자와 IAM 역할을 생성합니다. IAM 사용자와 IAM 역할에 Redshift 클러스터의 메타데이터를 볼 수 있는 권한을 부여합니다.
C. 각 사용자 역할에 따라 액세스를 허용하는 행 수준 보안 정책을 생성합니다. 특정 액세스 권한을 가진 IAM 역할을 생성합니다. 보안 정책을 테이블에 연결합니다.
D. 민감한 데이터를 삭제하고 새 Redshift 테이블에 데이터를 로드하기 위한 AWS Glue 작업을 생성합니다.
Answer: A
Explanation:
Option A is correct because Amazon Redshift dynamic data masking is designed to hide, obfuscate, or partially reveal sensitive values at query time without duplicating the underlying data. AWS states that masking policies can be attached to one or more columns and can be applied differently to certain users or roles. AWS also provides examples where different roles see different versions of the same column value, which matches the requirement that customer support users see only the last four characters while audit users can see the full

value.
Option B is incorrect because metadata security controls metadata visibility, not column-value masking.
Option C is incorrect because row-level security filters which rows are visible; it does not selectively mask part of a column value. Option D would create a second table and therefore duplicates data, which the question forbids. Dynamic data masking is the AWS-native feature that satisfies role-based partial exposure of a single sensitive column with no data duplication.

QUESTION NO: 12
데이터 엔지니어는 10개의 소스 시스템에서 Amazon Redshift 데이터베이스에 있는 10개의 테이블로 데이터를 처리하고 로드하기 위해 ETL(추출, 변환 및 로드) 파이프라인을 구축해야 합니다. 모든 소스 시스템은 15분마다 .csv, JSON 또는 Apache Parquet 파일을 생성합니다. 소스 시스템은 모두 파일을 하나의 Amazon S3 버킷으로 전달합니다. 파일 크기는 10MB에서 20GB까지입니다. ETL 파이프라인은 데이터 스키마 변경에도 불구하고 올바르게 작동해야 합니다.
이러한 요구 사항을 충족하는 데이터 파이프라인 솔루션은 무엇인가요? (2개를 선택하세요.)
A. Amazon EventBridge 규칙을 사용하여 15분마다 AWS Glue 작업을 실행합니다. 데이터를 처리하고 Amazon Redshift 테이블에 로드하도록 AWS Glue 작업을 구성합니다.
B. Amazon EventBridge 규칙을 사용하여 15분마다 AWS Glue 워크플로 작업을 호출합니다. AWS Glue 크롤러를 실행한 다음 크롤러 실행이 성공적으로 완료되면 AWS Glue 작업을 실행하는 온디맨드 트리거를 갖도록 AWS Glue 워크플로를 구성합니다. 데이터를 처리하고 Amazon Redshift 테이블에 로드하도록 AWS Glue 작업을 구성합니다.
C. 파일이 S3 버킷에 로드될 때 AWS Glue 크롤러를 호출하도록 AWS Lambda 함수를 구성합니다. 데이터를 처리하고 Amazon Redshift 테이블에 로드하도록 AWS Glue 작업을 구성합니다.
AWS Glue 작업을 실행하기 위한 두 번째 Lambda 함수를 생성합니다. AWS Glue 크롤러 실행이 성공적으로 완료되면 두 번째 Lambda 함수를 호출하는 Amazon EventBridge 규칙을 생성합니다.
D. 파일이 S3 버킷에 로드될 때 AWS Glue 워크플로를 호출하도록 AWS Lambda 함수를 구성합니다. AWS Glue 크롤러를 실행한 다음 크롤러 실행이 성공적으로 완료되면 AWS Glue 작업을 실행하는 온디맨드 트리거를 갖도록 AWS Glue 워크플로를 구성합니다. 데이터를 처리하고 Amazon Redshift 테이블에 로드하도록 AWS Glue 작업을 구성합니다.
E. 파일이 S3 버킷에 로드될 때 AWS Glue 작업을 호출하도록 AWS Lambda 함수를 구성합니다. S3 버킷의 파일을 Apache Spark DataFrame으로 읽도록 AWS Glue 작업을 구성합니다. DataFrame의 더 작은 파티션을 Amazon Kinesis Data Firehose 전송 스트림에 배치하도록 AWS Glue 작업을 구성합니다. Amazon Redshift 테이블에 데이터를 로드하도록 전송 스트림을 구성합니다.
Answer: A B
Explanation:
Using an Amazon EventBridge rule to run an AWS Glue job or invoke an AWS Glue workflow job every 15 minutes are two possible solutions that will meet the requirements. AWS Glue is a serverless ETL service that can process and load data from various sources to various targets, including Amazon Redshift. AWS Glue can handle different data formats, such as CSV, JSON, and Parquet, and also support schema evolution, meaning it can adapt to

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

QUESTION NO: 13
어떤 회사에서 애플리케이션과 온프레미스 Apache Kafka 서버를 AWS로 마이그레이션하려고 합니다. 애플리케이션은 온프레미스 Oracle 데이터베이스가 Kafka 서버로 보내는 증분 업데이트를 처리합니다. 이 회사는 리팩터링 전략 대신 리플랫폼 마이그레이션 전략을 사용하려고 합니다.
이러한 요구 사항을 가장 적은 관리 비용으로 충족할 수 있는 솔루션은 무엇일까요?
A. Amazon Kinesis 데이터 스트림
B. Apache Kafka(Amazon MSK)용 Amazon Managed Streaming 프로비저닝 클러스터
C. Amazon 데이터 파이어호스
D. Apache Kafka(Amazon MSK) Serverless용 Amazon Managed Streaming
Answer: D
Explanation:
Problem Analysis:
The company needs to migrate both an application and an on-premises Apache Kafka server to AWS.
Incremental updates from an on-premises Oracle database are processed by Kafka.

The solution must follow a replatform migration strategy, prioritizing minimal changes and low management overhead.
Key Considerations:
Replatform Strategy: This approach keeps the application and architecture as close to the original as possible, reducing the need for refactoring.
The solution must provide a managed Kafka service to minimize operational burden.
Low overhead solutions like serverless services are preferred.
Solution Analysis:
Option A: Kinesis Data Streams
Kinesis Data Streams is an AWS-native streaming service but is not a direct substitute for Kafka.
This option would require significant application refactoring, which does not align with the replatform strategy.
Option B: MSK Provisioned Cluster
Managed Kafka service with fully configurable clusters.
Provides the same Kafka APIs but requires cluster management (e.g., scaling, patching), increasing management overhead.
Option C: Amazon Kinesis Data Firehose
Kinesis Data Firehose is designed for data delivery rather than real-time streaming and processing.
Not suitable for Kafka-based applications.
Option D: MSK Serverless
MSK Serverless eliminates the need for cluster management while maintaining compatibility with Kafka APIs.
Automatically scales based on workload, reducing operational overhead.
Ideal for replatform migrations, as it requires minimal changes to the application.
Final Recommendation:
Amazon MSK Serverless is the best solution for migrating the Kafka server and application with minimal changes and the least management overhead.
Amazon MSK Serverless Overview
Comparison of Amazon MSK and Kinesis

QUESTION NO: 14
이 회사의 데이터 처리 파이프라인은 AWS Glue 작업과 AWS Glue 데이터 카탈로그를 사용합니다. 모든 AWS Glue 작업은 프라이빗 서브넷 내의 사용자 지정 VPC에서 실행되어야 합니다. 이 회사는 외부 연결을 지원하기 위해 NAT 게이트웨이를 사용합니다.
데이터 엔지니어가 AWS Glue를 사용하여 온프레미스 PostgreSQL 데이터베이스에서 Amazon S3로 데이터를 마이그레이션해야 합니다. 현재 AWS와 온프레미스 환경 간에는 네트워크 연결이 없습니다.
하지만 데이터 엔지니어는 사용자 지정 VPC에서 트래픽을 허용하도록 온프레미스 데이터베이스를 업데이트했습니다.
어떤 솔루션이 이러한 요구 사항을 충족할까요?
A. 데이터베이스 JDBC URL, 사용자 이름 및 비밀번호를 사용하여 AWS Glue에서 JDBC 연결을 생성합니다.
B. AWS Glue에서 온프레미스 데이터베이스에 대한 SASL(Simple Authentication and

Security Layer) 연결을 생성합니다.
C. AWS Glue에서 TCP 트래픽을 허용하는 보안 그룹을 사용하여 JDBC 연결을 생성합니다.
D. Amazon S3에 저장된 JDBC 드라이버를 사용하는 AWS Glue에서 JDBC 연결을 생성합니다. AWS Secrets Manager에서 데이터베이스 URL, 사용자 이름 및 암호를 검색합니다.
Answer: D
Explanation:
When AWS Glue jobs run inside a private subnet, they must use secure and supported methods to access external databases. AWS Glue supports JDBC connections to on-premises databases, but best practices require secure credential management and explicit driver configuration.
Using a JDBC driver stored in Amazon S3 allows Glue to connect to PostgreSQL without relying on default drivers. Storing credentials in AWS Secrets Manager eliminates hard-coded credentials and enables secure rotation, aligning with AWS security best practices.
Simply specifying credentials inline is less secure and not recommended. SASL connections are not supported for PostgreSQL JDBC connections. Security groups alone do not establish connectivity or authentication.
Therefore, Option D is the correct and production-grade solution.

QUESTION NO: 15
한 제조 회사는 운영 효율성을 모니터링하고 향상시키기 위해 공장 현장에서 센서 데이터를 수집합니다. 이 회사는 Amazon Kinesis Data Streams를 사용하여 센서가 수집하는 데이터를 데이터 스트림에 게시합니다. 그런 다음 Amazon Kinesis Data Firehose가 Amazon S3 버킷에 데이터를 씁니다.
회사는 제조 시설의 대형 화면에 운영 효율성을 실시간으로 표시해야 합니다.
가장 낮은 대기 시간으로 이러한 요구 사항을 충족하는 솔루션은 무엇입니까?
A. Apache Flink용 Amazon Managed Service(이전의 Amazon Kinesis Data Analytics)를 사용하여 센서 데이터를 처리합니다. Apache Flink용 커넥터를 사용하여 Amazon Timestream 데이터베이스에 데이터를 씁니다. Timestream 데이터베이스를 소스로 사용하여 Grafana 대시보드를 생성합니다.
B. 새 객체가 생성되면 AWS Lambda 함수에 알림을 보내도록 S3 버킷을 구성합니다. Lambda 함수를 사용하여 Amazon Aurora에 데이터를 게시합니다. Aurora를 소스로 사용하여 Amazon QuickSight 대시보드를 생성하십시오.
C. Apache Flink용 Amazon Managed Service(이전의 Amazon Kinesis Data Analytics)를 사용하여 센서 데이터를 처리합니다. 새로운 Data Firehose 전송 스트림을 생성하여 Amazon Timestream 데이터베이스에 직접 데이터를 게시합니다. Timestream 데이터베이스를 소스로 사용하여 Amazon QuickSight 대시보드를 생성합니다.
D. AWS Glue 북마크를 사용하여 S3 버킷에서 센서 데이터를 실시간으로 읽습니다. Amazon Timestream 데이터베이스에 데이터를 게시합니다. Timestream 데이터베이스를 소스로 사용하여 Grafana 대시보드를 생성합니다.
Answer: C
Explanation:
This solution will meet the requirements with the lowest latency because it uses Amazon Managed Service for Apache Flink to process the sensor data in real time and write it to Amazon Timestream, a fast, scalable, and serverless time series database. Amazon

Timestream is optimized for storing and analyzing time series data, such as sensor data, and can handle trillions of events per day with millisecond latency. By using Amazon Timestream as a source, you can create an Amazon QuickSight dashboard that displays a real-time view of operational efficiency on a large screen in the manufacturing facility. Amazon QuickSight is a fully managed business intelligence service that can connect to various data sources, including Amazon Timestream, and provide interactive visualizations and insights123.
The other options are not optimal for the following reasons:
A). Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Use a connector for Apache Flink to write data to an Amazon Timestream database.
Use the Timestream database as a source to create a Grafana dashboard. This option is similar to option C, but it uses Grafana instead of Amazon QuickSight to create the dashboard. Grafana is an open source visualization tool that can also connect to Amazon Timestream, but it requires additional steps to set up and configure, such as deploying a Grafana server on Amazon EC2, installing the Amazon Timestream plugin, and creating an IAM role for Grafana to access Timestream. These steps can increase the latency and complexity of the solution.
B). Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is created. Use the Lambda function to publish the data to Amazon Aurora. Use Aurora as a source to create an Amazon QuickSight dashboard. This option is not suitable for displaying a real-time view of operational efficiency, as it introduces unnecessary delays and costs in the data pipeline. First, the sensor data is written to an S3 bucket by Amazon Kinesis Data Firehose, which can have a buffering interval of up to 900 seconds.
Then, the S3 bucket sends a notification to a Lambda function, which can incur additional invocation and execution time. Finally, the Lambda function publishes the data to Amazon Aurora, a relational database that is not optimized for time series data and can have higher storage and performance costs than Amazon Timestream .
D). Use AWS Glue bookmarks to read sensor data from the S3 bucket in real time. Publish the data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is also not suitable for displaying a real-time view of operational efficiency, as it uses AWS Glue bookmarks to read sensor data from the S3 bucket. AWS Glue bookmarks are a feature that helps AWS Glue jobs and crawlers keep track of the data that has already been processed, so that they can resume from where they left off. However, AWS Glue jobs and crawlers are not designed for real-time data processing, as they can have a minimum frequency of 5 minutes and a variable start-up time. Moreover, this option also uses Grafana instead of Amazon QuickSight to create the dashboard, which can increase the latency and complexity of the solution .
1: Amazon Managed Streaming for Apache Flink
2: Amazon Timestream
3: Amazon QuickSight
Analyze data in Amazon Timestream using Grafana
Amazon Kinesis Data Firehose
Amazon Aurora
AWS Glue Bookmarks
AWS Glue Job and Crawler Scheduling

QUESTION NO: 16
한 회사가 Microsoft SQL Server를 실행하는 Amazon EC2 인스턴스에서 Microsoft SQL Server DB 인스턴스용 Amazon RDS로 데이터베이스 서버를 마이그레이션하고 있습니다. 회사의 분석 팀은 마이그레이션이 완료될 때까지 매일 대규모 데이터 요소를 내보내야 합니다. 데이터 요소는 여러 테이블에 대한 SQL 조인의 결과입니다. 데이터는 Apache Parquet 형식이어야 합니다. 분석 팀은 Amazon S3에 데이터를 저장해야 합니다.
어떤 솔루션이 운영상 가장 효율적인 방식으로 이러한 요구 사항을 충족합니까?
A. 필수 데이터 요소가 포함된 EC2 인스턴스 기반 SQL Server 데이터베이스에 뷰를 생성합니다.
보기에서 직접 데이터를 선택하고 Parquet 형식의 데이터를 S3 버킷으로 전송하는 AWS Glue 작업을 생성합니다. 매일 실행되도록 AWS Glue 작업을 예약합니다.
B. EC2 인스턴스 기반 SQL Server 데이터베이스에서 원하는 데이터 요소를 선택하는 일일 SQL 쿼리를 실행하도록 SQL Server 에이전트를 예약합니다. 출력 .csv 객체를 S3 버킷으로 보내도록 쿼리를 구성합니다. AWS Lambda 함수를 호출하여 출력 형식을 변환하는 S3 이벤트를 생성합니다.
.csv를 Parquet로 변환합니다.
C. SQL 쿼리를 사용하여 필요한 데이터 요소가 포함된 EC2 인스턴스 기반 SQL Server 데이터베이스에 보기를 생성합니다. 보기를 읽으려면 AWS Glue 크롤러를 생성하고 실행하세요. 데이터를 검색하고 Parquet 형식의 데이터를 S3 버킷으로 전송하는 AWS Glue 작업을 생성합니다. 매일 실행되도록 AWS Glue 작업을 예약합니다.
D. JDBC(Java Database Connectivity)를 사용하여 EC2 인스턴스 기반 데이터베이스를 쿼리하는 AWS Lambda 함수를 생성합니다. 필요한 데이터를 검색하고, 데이터를 Parquet 형식으로 변환하고, 데이터를 S3 버킷으로 전송하도록 Lambda 함수를 구성합니다. Amazon EventBridge를 사용하여 Lambda 함수가 매일 실행되도록 예약합니다.
Answer: A
Explanation:
Option A is the most operationally efficient way to meet the requirements because it minimizes the number of steps and services involved in the data export process. AWS Glue is a fully managed service that can extract, transform, and load (ETL) data from various sources to various destinations, including Amazon S3. AWS Glue can also convert data to different formats, such as Parquet, which is a columnar storage format that is optimized for analytics. By creating a view in the SQL Server databases that contains the required data elements, the AWS Glue job can select the data directly from the view without having to perform any joins or transformations on the source data. The AWS Glue job can then transfer the data in Parquet format to an S3 bucket and run on a daily schedule.
Option B is not operationally efficient because it involves multiple steps and services to export the data. SQL Server Agent is a tool that can run scheduled tasks on SQL Server databases, such as executing SQL queries.
However, SQL Server Agent cannot directly export data to S3, so the query output must be saved as .csv objects on the EC2 instance. Then, an S3 event must be configured to trigger an AWS Lambda function that can transform the .csv objects to Parquet format and upload them to S3. This option adds complexity and latency to the data export process and requires additional resources and configuration.
Option C is not operationally efficient because it introduces an unnecessary step of running an AWS Glue crawler to read the view. An AWS Glue crawler is a service that can scan data

sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. However, in this scenario, the schema and format of the data elements are already known and fixed, so there is no need to run a crawler to discover them. The AWS Glue job can directly select the data from the view without using the Data Catalog. Running a crawler adds extra time and cost to the data export process.
Option D is not operationally efficient because it requires custom code and configuration to query the databases and transform the data. An AWS Lambda function is a service that can run code in response to events or triggers, such as Amazon EventBridge. Amazon EventBridge is a service that can connect applications and services with event sources, such as schedules, and route them to targets, such as Lambda functions. However, in this scenario, using a Lambda function to query the databases and transform the data is not the best option because it requires writing and maintaining code that uses JDBC to connect to the SQL Server databases, retrieve the required data, convert the data to Parquet format, and transfer the data to S3.
This option also has limitations on the execution time, memory, and concurrency of the Lambda function, which may affect the performance and reliability of the data export process.
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
AWS Glue Documentation
Working with Views in AWS Glue
Converting to Columnar Formats

QUESTION NO: 17
한 회사가 Amazon Redshift를 사용하여 당일 주문 내역을 저장합니다. 이 회사는 이전 주문 데이터가 저장된 orders 테이블과 신규 또는 업데이트된 주문 기록을 저장하는 staging 테이블을 보유하고 있습니다. 회사는 orders 테이블에서 오래된 기록을 삭제하고 staging 테이블에서 최신 데이터를 orders 테이블에 삽입해야 합니다. 여러 하위 애플리케이션에서 orders 테이블의 최신 정보를 필요로 합니다.
어떤 솔루션이 이러한 요구 사항을 충족할까요?
A. Amazon Redshift Spectrum을 사용하여 주문 테이블에서 오래된 레코드를 삭제하고 스테이징 테이블의 레코드를 주문 테이블에 삽입합니다.
B. 주문 테이블과 스테이징 테이블을 Amazon S3에 언로드합니다. Amazon Athena를 사용하여 Amazon S3에서 오래된 주문 테이블 데이터를 삭제하고 새 스테이징 테이블 데이터를 삽입합니다. S3에 있는 주문 테이블을 Amazon Redshift의 주문 테이블로 복사합니다.
C. Amazon Athena 페더레이티드 쿼리를 사용하여 orders 테이블에서 오래된 레코드를 읽습니다. 오래된 레코드를 삭제하고 스테이징 테이블의 레코드를 orders 테이블에 삽입합니다.
D. Amazon Redshift 저장 프로시저를 작성하여 orders 테이블에서 오래된 레코드를 삭제하고 staging 테이블에서 새 레코드를 삽입하세요.
Answer: D
Explanation:
Option D is correct because Amazon Redshift stored procedures are designed to encapsulate a sequence of SQL statements and business logic inside the database. AWS documentation states that stored procedures are commonly used for data transformation,

data validation, and business-specific logic, and that they can combine multiple SQL steps into one procedure. AWS also documents the standard Redshift pattern for deleting stale rows and inserting fresh rows from a staging table, which is exactly the requirement here. Keeping the operation inside Redshift is the most direct way to maintain an up-to-date orders table for downstream consumers.
Option A is incorrect because Redshift Spectrum is for querying external data in S3, not for performing this in- place Redshift table-maintenance pattern. Option B adds unnecessary unload and reload steps, creating delay and operational complexity. Option C is also unsuitable because Athena federated queries are not the right mechanism for transactional maintenance of Redshift tables. The correct DEA-C01-style answer is to use Redshift-native procedural SQL to delete stale rows and insert current rows from staging.

QUESTION NO: 18
데이터 엔지니어가 AWS Glue Apache Spark 추출, 변환 및 로드(ETL) 작업을 구성하고 있습니다. 이 작업에는 크기가 같고 큰 두 개의 DataFrame을 정렬-병합 조인하는 작업이 포함되어 있습니다.
다음 오류로 인해 작업이 실패합니다: 장치에 남은 공간이 없습니다.
어떤 해결책이 오류를 해결할 수 있을까요?
A. AWS Glue Spark 셔플 관리자를 사용합니다.
B. 작업에 사용할 Amazon Elastic Block Store(Amazon EBS) 볼륨을 배포합니다.
C. 작업의 정렬 병합 조인을 브로드캐스트 조인으로 변환합니다.
D. DataFrames를 DynamicFrames로 변환하고 작업에서 DynamicFrame 조인을 수행합니다.
Answer: C
Explanation:
A sort-merge join generates large shuffle files, leading to "No space left on device" errors when both datasets are large. Using a broadcast join sends a smaller dataset to all executors, avoiding shuffle and disk I/O overhead.
"Broadcast joins reduce shuffle I/O by distributing the smaller dataset to all worker nodes, mitigating disk space and shuffle errors."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf This is the most cost-effective and direct fix for large shuffle-stage failures.

QUESTION NO: 19
데이터 엔지니어는 새로운 데이터 생산자를 AWS에 온보딩해야 합니다. 데이터 생산자는 데이터 제품을 AWS로 마이그레이션해야 합니다.
데이터 생산자는 비즈니스 애플리케이션을 지원하는 많은 데이터 파이프라인을 유지 관리합니다. 각 파이프라인에는 서비스 계정과 해당 자격 증명이 있어야 합니다. 데이터 엔지니어는 데이터 생산자의 온프레미스 데이터 센터에서 AWS로 보안 연결을 설정해야 합니다. 데이터 엔지니어는 퍼블릭 인터넷을 사용하여 온프레미스 데이터 센터에서 AWS로 데이터를 전송해서는 안 됩니다.
어떤 솔루션이 이러한 요구 사항을 충족시킬까요?
A. 새로운 데이터 생산자에게 Amazon Elastic Container Service(Amazon ECS)에 Amazon Machine Images(AMI)를 생성하여 애플리케이션의 코드 기반을 저장하도록 지시합니다. 온프레미스 데이터 센터에만 연결을 허용하는 퍼블릭 서브넷에 보안 그룹을 만듭니다.
B. 온프레미스 데이터 센터에 AWS Direct Connect 연결을 만듭니다. AWS Secrets

Manager에 서비스 계정 자격 증명을 저장합니다.
C. 퍼블릭 서브넷에 보안 그룹을 만듭니다. 데이터 생산자에 해당하는 CIDR 블록에서만 연결을 허용하도록 보안 그룹을 구성합니다. 만료 날짜가 하루인 미리 서명된 URL을 포함하는 Amazon S3 버킷을 만듭니다.
D. 온프레미스 데이터 센터에 AWS Direct Connect 연결을 만듭니다. 애플리케이션 키를 AWS Secrets Manager에 저장합니다. 만료 날짜가 하루인 재서명된 URL이 포함된 Amazon S3 버킷을 만듭니다.
Answer: B
Explanation:
For secure migration of data from an on-premises data center to AWS without using the public internet, AWS Direct Connect is the most secure and reliable method. Using Secrets Manager to store service account credentials ensures that the credentials are managed securely with automatic rotation.
AWS Direct Connect:
Direct Connect establishes a dedicated, private connection between the on-premises data center and AWS, avoiding the public internet. This is ideal for secure, high-speed data transfers.
Reference: AWS Direct Connect
AWS Secrets Manager:
Secrets Manager securely stores and rotates service account credentials, reducing operational overhead while ensuring security.
Reference: AWS Secrets Manager
Alternatives Considered:
A (ECS with security groups): This does not address the need for a secure, private connection from the on- premises data center.
C (Public subnet with presigned URLs): This involves using the public internet, which does not meet the requirement.
D (Direct Connect with presigned URLs): While Direct Connect is correct, presigned URLs with short expiration dates are unnecessary for this use case.
References:
AWS Direct Connect Documentation
AWS Secrets Manager Documentation

QUESTION NO: 20
한 회사가 중앙 거버넌스 계정이 있는 데이터 메시를 구현합니다. 회사는 거버넌스 계정의 모든 데이터를 카탈로그화해야 합니다. 거버넌스 계정은 AWS Lake Formation을 사용하여 데이터를 중앙에서 공유하고 액세스 권한을 부여합니다.
이 회사는 Amazon Redshift Serverless 테이블 그룹을 포함하는 새로운 데이터 제품을 만들었습니다. 데이터 엔지니어는 마케팅 팀과 데이터 제품을 공유해야 합니다. 마케팅 팀은 열의 하위 집합에만 액세스할 수 있어야 합니다. 데이터 엔지니어는 동일한 데이터 제품을 규정 준수 팀과 공유해야 합니다.
규정 준수 팀은 마케팅 팀이 액세스해야 하는 것과 다른 하위 집합의 열에 액세스해야 합니다.
이러한 요구 사항을 충족하기 위해 데이터 엔지니어는 어떤 단계 조합을 취해야 합니까? (2개를 선택하세요.)
A. 공유해야 하는 테이블의 뷰를 만듭니다. 필요한 열만 포함합니다.

B. 공유해야 하는 테이블이 포함된 Amazon Redshift 데이터를 생성합니다.
C. 마케팅팀 계정에서 Amazon Redshift 관리형 VPC 엔드포인트를 만듭니다. 마케팅팀에 뷰에 대한 액세스 권한을 부여합니다.
D. Amazon Redshift 데이터 공유를 거버넌스 계정의 Lake Formation 카탈로그에 공유합니다.
E. 마케팅 팀 계정의 Amazon Redshift Serverless 작업 그룹에 Amazon Redshift 데이터 공유를 공유합니다.
Answer: A E
Explanation:
The company is using a data mesh architecture with AWS Lake Formation for governance and needs to share specific subsets of data with different teams (marketing and compliance) using Amazon Redshift Serverless.
Option A: Create views of the tables that need to be shared. Include only the required columns.Creating views in Amazon Redshift that include only the necessary columns allows for fine-grained access control. This method ensures that each team has access to only the data they are authorized to view.
Option E: Share the Amazon Redshift data share to the Amazon Redshift Serverless workgroup in the marketing team ' s account.Amazon Redshift data sharing enables live access to data across Redshift clusters or Serverless workgroups. By sharing data with specific workgroups, you can ensure that the marketing team and compliance team each access the relevant subset of data based on the views created.
Option B (creating a Redshift data share) is close but does not address the fine-grained column-level access.
Option C (creating a managed VPC endpoint) is unnecessary for sharing data with specific teams.
Option D (sharing with the Lake Formation catalog) is incorrect because Redshift data shares do not integrate directly with Lake Formation catalogs; they are specific to Redshift workgroups.
References:
Amazon Redshift Data Sharing
AWS Lake Formation Documentation

QUESTION NO: 21
어떤 회사에는 개인 식별 정보(PIT) 데이터와 비 PII 데이터가 포함된 JSON 파일이 있습니다. 회사는 쿼리 및 분석을 위해 데이터를 제공해야 합니다. 개인 식별 정보가 아닌 데이터는 회사 내 모든 사람이 이용할 수 있어야 합니다. 개인 식별 정보는 제한된 직원 그룹에게만 제공되어야 합니다. 운영 오버헤드를 최소화하면서 이러한 요구 사항을 충족하는 솔루션은 무엇일까요?
A. JSON 파일을 Amazon S3 버킷에 저장합니다. AWS Glue를 구성하여 파일을 PII 데이터가 포함된 파일 하나와 PII가 아닌 데이터가 포함된 파일 하나로 분할합니다. 출력 파일은 별도의 S3 버킷에 저장합니다. 사용자 유형에 따라 버킷에 필요한 액세스 권한을 부여합니다.
B. JSON 파일을 Amazon S3 버킷에 저장합니다. Amazon Macie를 사용하여 PII 데이터를 식별하고 사용자 유형에 따라 액세스 권한을 부여합니다.
C. JSON 파일을 Amazon S3 버킷에 저장합니다. AWS Lake Formation에서 파일 스키마를 카탈로그화합니다. Lake Formation 권한을 사용하여 사용자 유형에 따라 필요한 데이터에 대한 액세스를 제공합니다.

D. Amazon RDS PostgreSQL 데이터베이스 두 개를 생성합니다. PII 데이터와 비PII 데이터를 각 데이터베이스에 로드합니다. 사용자 유형에 따라 데이터베이스에 대한 접근 권한을 부여합니다.

Answer: C

QUESTION NO: 22

한 회사에서 분석 솔루션을 구축하고 있습니다. 이 솔루션은 데이터 레이크 스토리지로 Amazon S3를 사용하고 데이터 웨어하우스로 Amazon Redshift를 사용합니다. 회사는 Amazon Redshift Spectrum을 사용하여 Amazon S3에 있는 데이터를 쿼리하려고 합니다. 가장 빠른 쿼리를 제공하는 작업은 무엇입니까? (2개를 선택하세요.)

A. gzip 압축을 사용하여 개별 파일을 1GB에서 5GB 사이의 크기로 압축합니다.

B. 열 기반 저장 파일 형식을 사용합니다.

C. 가장 일반적인 쿼리 조건자를 기반으로 데이터를 분할합니다.

D. 데이터를 10KB 미만의 파일로 분할합니다.

E. 다음이 아닌 파일 형식을 사용합니다.

Answer: B C

Explanation:

Amazon Redshift Spectrum is a feature that allows you to run SQL queries directly against data in Amazon S3, without loading or transforming the data. Redshift Spectrum can query various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.

On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run- length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance.

Therefore, using a columnar storage file format, such as Parquet, will provide faster queries, as it allows Redshift Spectrum to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. Additionally, partitioning the data based on the most common query predicates, such as date, time, region, etc., will provide faster queries, as it allows Redshift Spectrum to prune the partitions that do not match the query criteria, reducing the amount of data scanned from S3. Partitioning also improves the performance of joins and aggregations, as it reduces data skew and shuffling.

The other options are not as effective as using a columnar storage file format and partitioning the data. Using gzip compression to compress individual files to sizes that are between 1 GB and 5 GB will reduce the data size, but it will not improve the query performance significantly,

as gzip is not a splittable compression algorithm and requires decompression before reading. Splitting the data into files that are less than 10 KB will increase the number of files and the metadata overhead, which will degrade the query performance. Using file formats that are not supported by Redshift Spectrum, such as XML, will not work, as Redshift Spectrum will not be able to read or parse the data. References:
Amazon Redshift Spectrum
Choosing the Right Data Format
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Lakes and Data Warehouses, Section 4.3: Amazon Redshift Spectrum

QUESTION NO: 23
데이터 엔지니어는 소매 주문을 처리하는 데이터 파이프라인의 성능을 최적화해야 합니다. 주문 데이터는 매일 Amazon S3 버킷에 수집됩니다.
데이터 엔지니어는 매주 한 번씩 쿼리를 실행하여 여러 날짜 범위의 주문 날짜를 기준으로 주문 데이터에서 지표를 추출합니다. 데이터 엔지니어는 데이터 양이 증가해도 쿼리 성능이 저하되지 않도록 하는 최적화 솔루션이 필요합니다.
A. 주문 날짜를 기준으로 데이터를 분할합니다. Amazon Athena를 사용하여 데이터를 쿼리합니다.
B. 주문 날짜를 기준으로 데이터를 분할합니다. Amazon Redshift를 사용하여 데이터를 쿼리합니다.
C. 로드 날짜를 기준으로 데이터를 분할합니다. Amazon EMR을 사용하여 데이터를 쿼리합니다.
D. 로드 날짜를 기준으로 데이터를 분할합니다. Amazon Aurora를 사용하여 데이터를 쿼리합니다.
Answer: A
Explanation:
For query workloads on S3 data that depend on date-based filters, partitioning by order date optimizes performance and cost because Athena reads only the relevant partitions. Athena scales automatically and doesn't degrade with increasing data size when partitions are managed efficiently.
"Partitioning data in Amazon S3 based on query predicates such as order date improves Athena query performance and reduces scanned data volume."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf This is the most cost-effective and scalable option for date-based queries.

QUESTION NO: 24
데이터 엔지니어는 Amazon S3 버킷에 있는 데이터에 대해 Amazon Athena 쿼리를 실행합니다. Athena 쿼리는 AWS Glue 데이터 카탈로그를 메타데이터 테이블로 사용합니다.
데이터 엔지니어는 Athena 쿼리 계획에 성능 병목 현상이 발생하고 있음을 발견했습니다. 데이터 엔지니어는 성능 병목 현상의 원인이 S3 버킷에 있는 많은 수의 파티션이라고 판단합니다. 데이터 엔지니어는 성능 병목 현상을 해결하고 Athena 쿼리 계획 시간을 줄여야 합니다.
어떤 솔루션이 이러한 요구 사항을 충족합니까? (2개를 선택하세요.)
A. AWS Glue 파티션 인덱스를 생성합니다. 파티션 필터링을 활성화합니다.
B. 사용자 쿼리의 WHERE 절에서 데이터가 공통으로 갖는 열을 기준으로 데이터를

버킷합니다.
C. S3 버킷 접두사를 기반으로 Athena 파티션 프로젝션을 사용합니다.
D. S3 버킷에 있는 데이터를 Apache Parquet 형식으로 변환합니다.
E. Amazon EMR S3DistCP 유틸리티를 사용하여 S3 버킷의 작은 객체를 더 큰 객체로 결합합니다.
Answer: A C
Explanation:
The best solutions to resolve the performance bottleneck and reduce Athena query planning time are to create an AWS Glue partition index and enable partition filtering, and to use Athena partition projection based on the S3 bucket prefix.
AWS Glue partition indexes are a feature that allows you to speed up query processing of highly partitioned tables cataloged in AWS Glue Data Catalog. Partition indexes are available for queries in Amazon EMR, Amazon Redshift Spectrum, and AWS Glue ETL jobs. Partition indexes are sublists of partition keys defined in the table. When you create a partition index, you specify a list of partition keys that already exist on a given table. AWS Glue then creates an index for the specified keys and stores it in the Data Catalog. When you run a query that filters on the partition keys, AWS Glue uses the partition index to quickly identify the relevant partitions without scanning the entire table metadata. This reduces the query planning time and improves the query performance1.
Athena partition projection is a feature that allows you to speed up query processing of highly partitioned tables and automate partition management. In partition projection, Athena calculates partition values and locations using the table properties that you configure directly on your table in AWS Glue. The table properties allow Athena to 'project', or determine, the necessary partition information instead of having to do a more time-consuming metadata lookup in the AWS Glue Data Catalog. Because in-memory operations are often faster than remote operations, partition projection can reduce the runtime of queries against highly partitioned tables. Partition projection also automates partition management because it removes the need to manually create partitions in Athena, AWS Glue, or your external Hive metastore2.
Option B is not the best solution, as bucketing the data based on a column that the data have in common in a WHERE clause of the user query would not reduce the query planning time. Bucketing is a technique that divides data into buckets based on a hash function applied to a column. Bucketing can improve the performance of join queries by reducing the amount of data that needs to be shuffled between nodes. However, bucketing does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario3.
Option D is not the best solution, as transforming the data that is in the S3 bucket to Apache Parquet format would not reduce the query planning time. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes. However, Parquet does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario4.
Option E is not the best solution, as using the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects would not reduce the query planning time. S3DistCP is a tool that can copy large amounts of data between Amazon S3 buckets or from

HDFS to Amazon S3. S3DistCP can also aggregate smaller files into larger files to improve the performance of sequential access. However, S3DistCP does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario5.
References:
Improve query performance using AWS Glue partition indexes
Partition projection with Amazon Athena
Bucketing vs Partitioning
Columnar Storage Formats
S3DistCp
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

QUESTION NO: 25
데이터 엔지니어가 Apache Spark를 사용하여 Amazon S3에 저장된 대규모 데이터 세트를 분석하는 Amazon Athena 노트북에서 쿼리 성능을 최적화하고 있습니다. 데이터는 분할되어 있으며, AWS Glue 크롤러가 분할된 데이터를 업데이트합니다.
데이터 엔지니어는 Athena 쿼리의 효율성을 개선하기 위해 스캔하는 데이터 양을 최소화하려고 합니다.
어떤 솔루션이 이러한 요구 사항을 충족시킬까요?
A. 쿼리에 파티션 필터를 적용합니다.
B. AWS Glue 크롤러 호출 빈도를 높여 데이터 카탈로그를 더 자주 업데이트합니다.
C. 중첩된 디렉토리 구조를 사용하여 Amazon S3에 있는 데이터를 구성합니다.
D. 자주 액세스되는 데이터에 대해 메모리 내 캐싱을 사용하도록 Spark를 구성합니다.
Answer: A

QUESTION NO: 26
회사에서 온프레미스 워크로드를 AWS로 마이그레이션하고 있습니다. 회사는 전반적인 운영 오버헤드를 줄이고 싶어합니다. 또한 회사는 서버리스 옵션도 살펴보고 싶어합니다.
회사의 현재 워크로드는 Apache Pig, Apache Oozie, Apache Spark, Apache Hbase 및 Apache Flink를 사용합니다. 온프레미스 워크로드는 페타바이트 규모의 데이터를 몇 초 만에 처리합니다. 회사는 AWS로 마이그레이션한 후에도 유사하거나 더 나은 성능을 유지해야 합니다.
이러한 요구 사항을 충족하는 ETL(추출, 변환 및 로드) 서비스는 무엇입니까?
A. AWS Glue
B. Amazon EMR
C. AWS 람다
D. Amazon Redshift
Answer: B
Explanation:
AWS Glue is a fully managed serverless ETL service that can handle petabytes of data in seconds. AWS Glue can run Apache Spark and Apache Flink jobs without requiring any infrastructure provisioning or management. AWS Glue can also integrate with Apache Pig, Apache Oozie, and Apache Hbase using AWS Glue Data Catalog and AWS Glue workflows. AWS Glue can reduce the overall operational overhead by automating the data discovery, data preparation, and data loading processes. AWS Glue can also optimize the cost and performance of ETL jobs by using AWS Glue Job Bookmarking, AWS Glue Crawlers, and

AWS Glue Schema Registry. References:
AWS Glue
AWS Glue Data Catalog
AWS Glue Workflows
[AWS Glue Job Bookmarking]
[AWS Glue Crawlers]
[AWS Glue Schema Registry]
[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

QUESTION NO: 27
보안 검토 중에 한 회사에서 AWS Glue 작업의 취약점을 식별했습니다. 회사는 Amazon Redshift 클러스터에 액세스하기 위한 자격 증명이 작업 스크립트에 하드 코딩되어 있음을 발견했습니다.
데이터 엔지니어는 AWS Glue 작업의 보안 취약성을 해결해야 합니다. 솔루션은 자격 증명을 안전하게 저장해야 합니다.
이러한 요구 사항을 충족하기 위해 데이터 엔지니어는 어떤 단계 조합을 수행해야 합니까? (2개를 선택하세요.)
A. AWS Glue 작업 매개변수에 자격 증명을 저장합니다.
B. Amazon S3 버킷에 있는 구성 파일에 자격 증명을 저장합니다.
C. AWS Glue 작업을 사용하여 Amazon S3 버킷에 있는 구성 파일에서 자격 증명에 액세스합니다.
D. AWS Secrets Manager에 자격 증명을 저장합니다.
E. AWS Glue 작업 1AM 역할에 저장된 자격 증명에 대한 액세스 권한을 부여합니다.
Answer: D E
Explanation:
AWS Secrets Manager is a service that allows you to securely store and manage secrets, such as database credentials, API keys, passwords, etc. You can use Secrets Manager to encrypt, rotate, and audit your secrets, as well as to control access to them using fine-grained policies. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue jobs allow you to transform and load data from various sources into various targets, using either a graphical interface (AWS Glue Studio) or a code-based interface (AWS Glue console or AWS Glue API).
Storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials will meet the requirements, as it will remediate the security vulnerability in the AWS Glue job and securely store the credentials. By using AWS Secrets Manager, you can avoid hard coding the credentials in the job script, which is a bad practice that exposes the credentials to unauthorized access or leakage. Instead, you can store the credentials as a secret in Secrets Manager and reference the secret name or ARN in the job script. You can also use Secrets Manager to encrypt the credentials using AWS Key Management Service (AWS KMS), rotate the credentials automatically or on demand, and monitor the access to the credentials using AWS CloudTrail. By granting the AWS Glue job 1AM role access to the stored credentials, you can use the principle of least privilege to ensure that only the AWS Glue job can retrieve the credentials from Secrets Manager. You can also use resource-based or tag-based policies to further restrict the access to the

credentials.
The other options are not as secure as storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials. Storing the credentials in the AWS Glue job parameters will not remediate the security vulnerability, as the job parameters are still visible in the AWS Glue console and API. Storing the credentials in a configuration file that is in an Amazon S3 bucket and accessing the credentials from the configuration file by using the AWS Glue job will not be as secure as using Secrets Manager, as the configuration file may not be encrypted or rotated, and the access to the file may not be audited or controlled. References:
AWS Secrets Manager
AWS Glue
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

QUESTION NO: 28
보안 회사는 JSON 형식의 IoT 데이터를 Amazon S3 버킷에 저장합니다. 회사가 IoT 장치를 업그레이드하면 데이터 구조가 변경될 수 있습니다. 회사는 IoT 데이터가 포함된 데이터 카탈로그를 생성하려고 합니다. 회사의 분석 부서는 데이터 카탈로그를 사용하여 데이터를 색인화합니다.
이러한 요구 사항을 가장 비용 효율적으로 충족하는 솔루션은 무엇입니까?
A. AWS Glue 데이터 카탈로그를 생성합니다. AWS Glue 스키마 레지스트리를 구성합니다. 분석 부서가 Amazon Redshift Serverless에 사용할 데이터 수집을 조정하기 위해 새로운 AWS Glue 워크로드를 생성합니다.
B. Amazon Redshift 프로비저닝된 클러스터를 생성합니다. 분석 부서가 Amazon S3에 있는 데이터를 탐색할 수 있도록 Amazon Redshift Spectrum 데이터베이스를 생성합니다. Amazon Redshift에 데이터를 로드하는 Redshift 저장 프로시저를 생성합니다.
C. Amazon Athena 작업 그룹을 생성합니다. Athena를 통해 Apache Spark를 사용하여 Amazon S3에 있는 데이터를 탐색합니다. Athena 작업 그룹 스키마와 테이블을 분석 부서에 제공합니다.
D. AWS Glue 데이터 카탈로그를 생성합니다. AWS Glue 스키마 레지스트리를 구성합니다. Amazon Redshift Data API를 사용하여 AWS Lambda 사용자 정의 함수(UDF)를 생성합니다. 분석 부서가 Amazon Redshift Serverless에 사용할 데이터 수집을 조율하기 위해 AWS Step Functions 작업을 생성합니다.
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 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 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 and manage the cluster, create an external schema and database for the data in Amazon S3, and load the data into the cluster using stored procedures, instead of querying it directly from Amazon S3 using Athena and Spark.
Option D is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API, and creating an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers13. AWS Lambda UDFs are Lambda functions that you can invoke from within an Amazon Redshift query. Amazon Redshift Data API is a service that allows you to run SQL statements on Amazon Redshift clusters using HTTP requests, without needing a persistent connection. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Lambda charges you based on the number of requests and the duration of your functions13. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. AWS Step Functions charges you based on the number of state transitions in your workflows. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue, AWS Lambda, Amazon Redshift Data API, and AWS Step Functions would introduce additional latency and complexity, as you would have to create and invoke Lambda functions to ingest the data from Amazon S3 to Amazon Redshift Serverless using the Data API, and coordinate the ingestion process using Step Functions, instead of querying it directly from Amazon S3 using Athena and Spark. References:
What is Amazon Athena?
Apache Spark on Amazon Athena
Creating tables, updating the schema, and adding new partitions in the Data Catalog from AWS Glue ETL jobs Managing Athena workgroups Using Amazon QuickSight to visualize data in Amazon Athena AWS Glue Data Catalog AWS Glue Schema Registry What is AWS Glue?
Amazon Redshift Serverless
Amazon Redshift provisioned clusters
Querying external data using Amazon Redshift Spectrum

Using stored procedures in Amazon Redshift
What is AWS Lambda?
[Creating and using AWS Lambda UDFs]
[Using the Amazon Redshift Data API]
[What is AWS Step Functions?]
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

QUESTION NO: 29
한 회사가 아마존 S3 버킷에 차원 테이블을 구축하려고 합니다. 해당 버킷에는 1천만 건의 레코드가 포함된 과거 데이터가 저장되어 있으며, 데이터 크기는 1TB입니다.
데이터 엔지니어는 기본 테이블에서 매일 최대 10,000개의 레코드 변경 사항을 업데이트할 수 있는 솔루션이 필요합니다.
어떤 솔루션이 가장 짧은 실행 시간으로 이 요구 사항을 충족할까요?

A. Amazon EMR에서 Apache Spark 작업을 개발하여 과거 데이터와 새로운 변경 사항을 읽어 두 개의 Spark DataFrame으로 저장하세요. Spark의 update 메서드를 사용하여 기본 테이블을 업데이트하세요.

B. AWS Glue Python 작업을 개발하여 과거 데이터와 새로운 변경 사항을 두 개의 Pandas DataFrame으로 읽어들이세요. Pandas의 update 메서드를 사용하여 기본 테이블을 업데이트하세요.

C. AWS Glue Apache Spark 작업을 개발하여 과거 데이터와 새로운 변경 사항을 두 개의 Spark DataFrame으로 읽어들이세요. Spark의 update 메서드를 사용하여 기본 테이블을 업데이트하세요.

D. 새로운 변경 사항을 Apache Spark DataFrame으로 읽어들이는 Amazon EMR 작업을 개발하세요. Apache Hudi 프레임워크를 사용하여 Amazon S3에 기본 테이블을 생성하고, Spark의 update 메서드를 사용하여 기본 테이블을 업데이트하세요.

Answer: D
Explanation:
Option D provides the lowest runtime because it uses a table format designed for efficient incremental upserts on Amazon S3, rather than repeatedly scanning and rewriting large portions of a 1 TB dataset. Although Spark on its own (Options A and C) can perform joins/merges, updating files stored in S3 typically requires expensive rewrites, especially as data grows. By contrast, Apache Hudi is purpose-built for maintaining large datasets on object storage with incremental updates, which directly fits "update up to 10,000 records every day" without reprocessing the full historical footprint.
For the compute layer, the document highlights that Amazon EMR provides a fully managed environment for running Apache Spark and other big data frameworks to process and analyze large datasets, making it appropriate for high-scale processing where performance matters. This is a better fit than using Pandas on 1 TB (Option B), which is not designed for distributed processing at that scale.
Therefore, combining EMR + Spark with an incremental storage framework (Hudi) is the most runtime- efficient approach for daily record-level updates on S3.