**AWS Certified Data Analytics** - **Specialty Practice Questions**

**Requirement**: Share 10 DA Specialty practice questions.

**Important Note**: The practice questions should appropriately belong to DA Specialty in terms of exam objectives & difficulty level.

**Delivery Timeline**: April-4

**Question Response Types**

There are two types of questions:

* Multiple Choice Single Response – **1** correct answer **3** incorrect responses (distractors).
* Multiple Choice Multiple Response – **2** or more correct answers out of **5** or more options.

**Important Note**

* Do write Question Number for quick identification. Q# 1, Q# 2 …. & so on.
* Please mention Domain (based on DA Specialty exam blueprint), Topic & Sub-Topic (If Applicable) with every question.
* Note that we’re expecting standard scenario based questions & NOT straight-forward definition kind of questions.
* The options should not have any obviously incorrect option. We need to word the options so that all of them should appear correct for the students, but a subtle point should mark the correct answer without any ambiguity. So, one answer should be the best choice without any doubt.
* The answer / explanation section should contain explanations on why the answer is correct and others are incorrect. It should also contain the relevant resource link (for details) preferably from AWS documentation.
  + Example
    - Option A is incorrect because..
    - Option B is CORRECT because…
    - Option C is incorrect because..
    - Option D is incorrect because..
* Try to balance the domains based on weightage % defined in the exam blueprint.
* Any AWS service or feature must be approximately 6 months old to figure out in Practice Tests. Put a note in the explanation for any latest service or feature that might be on the borderline of appearing in the real exam.
* **Plagiarism** in any form - Question or in Explanation will be **rejected.** Questions & Explanations should reflect your own professional experience & AWS skills. Author’s who indulge in plagiarism will be **blacklisted** & dropped from our author’s list.
* The ownership of the questions once approved & published on Whizlabs LMS platform, lies solely with Whizlabs Software Pvt. Ltd. You can’t share or publish it elsewhere in any circumstances.

**Sample Format of** **Questions**

----------------------------------------------------------------------------------------------------------------------------

**Question​ ​:​** #

**Main​ ​Topic​ ​:​** < >

**Sub​ ​Topic​ ​:​** [optional]

**Domain:** < >

**Question text**:

<Scenario based. Should be clear in terms of requirements. No ambiguity. No duplicate options. In case of multiple answers, at the end, you should include the number of expected answers. e.g. (Select TWO) or (Select THREE) etc. For single answers this is NOT required>

1. Option A...
2. Option B...
3. Option C...
4. Option D...

**Answer:** A and C

**Explanation:**

**Option A is CORRECT because...**

**Option B is incorrect because...**

**Option C is CORRECT because...**

**Option D is incorrect because...**

[Insert the explanation in clear and lucid language here.]

**Diagram:** [Optional] [Insert the architectural or conceptual diagram here.]

**Reference:** [Insert the references here - which may include links to AWS Documentation, Blog, re:Invent video, Authority YouTube video].

----------------------------------------------------------------------------------------------------------------------------

**DA Specialty has 5 Domains**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Name of the Domain** | **Weight** | **Estimated No. of Questions**  (out of 65 As per weightage %) |
| 1 | Collection | 18% | 12 |
| 2 | Storage and Data Management | 22% | 14 |
| 3 | Processing | 24% | 15 |
| 4 | Analysis and Visualization | 18% | 12 |
| 5 | Security | 18% | 12 |

--------------------------------------Question Section Starts-----------------------------------------------------

Question: 1

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Determine the operational characteristics of a storage solution for analytics**

**Domain:** Storage and data management concerns

**Question text**:

You are a data scientist working for a financial services company that has several relational databases, data warehouses, and NoSQL databases that hold transactional information about their financial trades and operational activities. The company wants to manage their financial counterparty risk through using their real-time trading/operational data to perform risk analysis and build risk management dashboards.

You need to build a data repository that combines all of these disparate data sources so that your company can perform their Business Intelligence (BI) analysis work on the complete picture of their risk exposure.

What collection system best fits this use case?

1. Financial data sources data -> Kinesis Data Firehose -> S3 -> Glue -> S3 Data Lake -> Athena
2. Financial data sources data -> Kinesis Data Firehose -> Kinesis Data Analytics -> Kinesis Data Firehose -> Redshift -> QuickSight
3. Financial data sources data -> Database Migration Service -> S3 -> Glue -> S3 Data Lake -> Athena
4. Financial data sources data -> Kinesis Data Streams -> Kinesis Data Analytics -> S3 Data Lake -> QuickSight

**Answer:** C

**Explanation:**

Option A is incorrect. This data collection system architecture is best suited to batch consumption of stream data. You are trying to build a real-time financial risk management analytics collection architecture. You have several databases and data warehouses generating your data stream from their changed data. This approach is called ongoing replication or change data capture (CDC) within the Database Migration Service. A collection architecture using the Database Migration Service will be the most optimal for this use case.

Option B is incorrect. This data collection system architecture is suited to real-time consumption of data, but a collection architecture using the Database Migration Service would better fit this use case. You have several databases and data warehouses generating your data stream from their changed data. This approach is called ongoing replication or change data capture (CDC) within the Database Migration Service. A collection architecture using the Database Migration Service will be the most optimal for this use case.

Option C is correct. This type of data collection infrastructure is best used for streaming transactional data from existing relational data stores. You create a task within the Database Migration Service that collects ongoing changes within your various operational data stores, an approach called ongoing replication or change data capture (CDC). These changes are streamed to an S3 bucket where a Glue job is used to transform the data and move it to your S3 data lake.

Option D is incorrect. Kinesis Data Analytics cannot write directly to S3; it only writes to a Kinesis data stream, a Kinesis Data Firehose delivery stream, or a Lambda function. Also, this collection architecture does not take advantage of the Database Migration Service ongoing replication or change data capture (CDC) technique.

**Reference:**

Please see the AWS Database Migration Service user guide titled **Creating Tasks for Ongoing Replication Using AWS DMS** (<https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Task.CDC.html>), the AWS Schema Conversion Tool user guide titled **What Is the AWS Schema Conversion Tool?** (<https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Welcome.html>),

the Amazon Kinesis Data Analytics for SQL Applications developer guide titled **Configuring Application Output**

(<https://docs.aws.amazon.com/kinesisanalytics/latest/dev/how-it-works-output.html>), the AWS Streaming Data page titled **What is Streaming Data?** (<https://aws.amazon.com/streaming-data/>), the **AWS Database Migration Service FAQs** (<https://aws.amazon.com/dms/faqs/>), the **Amazon Kinesis Data Analytics FAQs (**[**https://aws.amazon.com/kinesis/data-analytics/faqs/**](https://aws.amazon.com/kinesis/data-analytics/faqs/)), the **Amazon Kinesis Data Streams FAQs (**[**https://aws.amazon.com/kinesis/data-streams/faqs/**](https://aws.amazon.com/kinesis/data-streams/faqs/)), , the Amazon Kinesis Data Firehose developer guide titles **What is Amazon Kinesis Data Firehose? (**[**https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html#data-flow-diagrams**](https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html#data-flow-diagrams)), the AWS Glue developer guide titled **AWS Glue Concepts** (<https://docs.aws.amazon.com/glue/latest/dg/components-key-concepts.html>), and the **Amazon Kinesis Data Firehose FAQs (**[**https://aws.amazon.com/kinesis/data-firehose/faqs/**](https://aws.amazon.com/kinesis/data-firehose/faqs/))

Question: 2

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Select a collection system that handles the frequency, volume, and source of data**

**Domain:** Data collection systems

**Question text**:

You are a data scientist working on a project where you have two large tables (orders and products) that you need to load into Redshift from one of your S3 buckets. Your table files, which are both several million rows large, are currently on an EBS volume of one of your EC2 instances in a directory titled $HOME/myredshiftdata.

Since your table files are so large, what is the most efficient approach to first copy them to S3 from your EC2 instance?

1. Load your orders.tbl and products.tbl using the command: ‘aws s3 cp $HOME/myredshiftdata s3://dataanalytics/myredshiftdata --recursive’
2. Load your orders.tbl and products.tbl by first splitting each tbl file into smaller parts using the command: ‘split -d -l 5000000 -a 4 orders.tbl orders.tbl’ and ‘split -d -l 10000000 -a 4 products.tbl products.tbl’
3. Load your orders.tbl and products.tbl by first getting a count of the number of rows in each using the commands: ‘wc -l orders.tbl’ and ‘wc -l products.tbl’. Then splitting each tbl file into smaller parts using the command: ‘split -d -l # -a 4 orders.tbl orders.tbl’ and ‘split -d -l # -a 4 products.tbl products.tbl’ where # is replaced by the result of your wc command divided by 4.
4. Load your orders.tbl and products.tbl by first getting a count of the number of rows in each using the commands: ‘wc -l orders.tbl’ and ‘wc -l products.tbl’. Then splitting each tbl file into smaller parts using the command: ‘split -d -l # -a 4 orders.tbl orders.tbl-’ and ‘split -d -l # -a 4 products.tbl products.tbl-’ where # is replaced by the result of your wc command divided by 4.

**Answer:** D

**Explanation:**

Option A is incorrect because using the commands in this answer you don’t reduce the size of your tbl files before attempting to move them to S3. Also, when you attempt to move these files into Redshift from your S3 bucket the process will be less efficient because you haven’t split your files into more manageable sizes.

Option B is incorrect because when you attempt to split your files you haven’t determined the actual number of rows of each file. Therefore, your random selection of a split size will more than likely not be an efficient selection.

Option C is incorrect because your split command does not have a trailing ‘-’ at the end of the command. Therefore your smaller files on your S3 bucket will have names like ‘orders.tbl0001’ versus the more readable and manageable ‘orders.tbl-0001’ if you use a trailing ‘-’ in the split command.

Option D is correct because you have used the wc command to find the number of rows in each tbl file, and you have used the split command with the trailing ‘-’ to get the proper file name format on your S3 bucket in preparation for loading into Redshift.

**Reference:**

Please see the AWS Redshift Developer Guide titled **Tutorial: Loading Data from Amazon S3** (<https://docs.aws.amazon.com/redshift/latest/dg/tutorial-loading-data.html>), specifically step 2: Download the Data Files and Step 5: Run the Copy Commands where you’ll see that having the ‘-’ at the end of your split command will allow you to make your Redshift copy command more efficient.

Question: 3

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Determine appropriate data processing solution requirements**

**Domain:** Data processing solutions

**Question text**:

You are working on a project where you need to perform real-time analytics on your application server logs. Your application is split across several EC2 servers in an auto-scaling group and is behind an application load balancer as depicted in this diagram:

[](https://www.draw.io/?page-id=6_lzF5P4isDOIafqsJO-&scale=auto#G1GNODpLc0DVlcFq7mT79w1ym0bAFvR-pw)

You need to perform some transformation on the log data, such as joining rows of data, before you stream the data to your real-time dashboard.

What is the most efficient and performant solution to aggregate your application logs?

1. Install the Kinesis Agent on your application servers to watch your logs and use Kinesis Data Firehose to stream the logs directly to S3. Use Kinesis Data Analytics queries to build your real-time analytics dashboard.
2. Write a Kinesis Data Streams producer application that reads the application logs and pushes the data directly into your Kinesis data stream. Use Kinesis Data Analytics queries to build your real-time analytics dashboard.
3. Install the Kinesis Agent on your application servers to watch your logs and ingest the log data. Write a Kinesis Data Analytics application that reads the application log data from the agent, performs the required transformations, and pushes the data into your Kinesis data output stream. Use Kinesis Data Analytics queries to build your real-time analytics dashboard.
4. Use a CloudWatch dashboard that uses your application’s CloudWatch logs as the data source.

**Answer:** C

**Explanation:**

Option A is incorrect because with this approach you don’t have a capability to perform the required transformations. You could write a lambda function to perform the transformations but the answer doesn’t specify these details.

Option B is incorrect because the answer is missing the Kinesis Agent part of the solution. You could write your Kinesis producer application to read the application log files, but using the Kinesis Agent is much more efficient.

Option C is correct. The Kinesis Agent ingests the application log data, the Kinesis Analytics application transforms the data, and Kinesis Analytics queries are used to build your dashboard.

Option D is incorrect since while a CloudWatch dashboard could be used to build this solution simply, it lacks the real-time capability. CloudWatch high-resolution metrics log in intervals of 1 second, 5 seconds, 10 seconds, 30 seconds, or multiples of 60 seconds. Also, this solution lacks the ability to perform the required transformations of the log data.

**Reference:**

Please see the **Amazon CloudWatch FAQs** (<https://aws.amazon.com/cloudwatch/faqs/>), the Amazon Kinesis Data Firehose Developer Guide titled **Amazon Kinesis Data Firehose Data Transformation** (<https://docs.aws.amazon.com/firehose/latest/dev/data-transformation.html>), the AWS blog titled **Implement Serverless Log Analytics Using Kinesis Analytics** (<https://aws.amazon.com/blogs/big-data/implement-serverless-log-analytics-using-amazon-kinesis-analytics/>), and the **Amazon Kinesis Data Streams overview page** (<https://aws.amazon.com/kinesis/data-streams/>)

Question: 4

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Design a solution for transforming and preparing data for analysis**

**Domain:** Data processing solutions

**Question text**:

You are a data scientist on a team where you are responsible for ingesting IoT streamed data into a data lake for use in an EMR cluster. The data in the data lake will be used to allow your company to do business intelligence analytics on the IoT data. Due to the large amount of data being streamed to your application you will need to compress the data on the fly as you process it into your EMR cluster.

How would you most efficiently collect the data from your IoT devices?

1. Use the Kinesis REST API to get the IoT device records from the IoT devices and stream the data to your data lake through Kinesis Data Streams, then use Apache DistCp to move the data from S3 to your EMR cluster.
2. Use the AWS IoT service to get the device data from the IoT devices, use Kinesis Data Firehose to stream the data to your data lake, then use S3DistCp to move the data from S3 to your EMR cluster.
3. Use the Kinesis Producer Library to create a Kinesis producer application that reads the data from the IoT devices and stream the data to your data lake through Kinesis Data Streams, then use S3DistCp to move the data from S3 to your EMR cluster.
4. Use the Kinesis Client Library to get the device data from the IoT devices, use Kinesis Data Firehose to stream the data to your data lake, then use Apache DistCp to move the data from S3 to your EMR cluster.

**Answer:** B

**Explanation:**

Option A is incorrect because the Kinesis REST API is not the most efficient way to gather the IoT device data from your set of devices. Also, Apache DistCp does not offer the compression option that S3DistCp offers.

Option B is correct. The AWS IoT service ingests the device data, Kinesis Data Firehose streams the data to your S3 data lake, then the S3DistCp command is used to compress and move the data inno your EMR cluster

Option C is incorrect. The Kinesis Producer Library is not the most efficient way to gather the IoT device data from your set of devices.

Option D is incorrect. The Kinesis Client Library is used to consume Kinesis Stream data, not to produce data for consumption into the data stream. Also, Apache DistCp does not offer the compression option that S3DistCp offers.

**Reference:**

Please see the **AWS IoT overview page** (<https://aws.amazon.com/iot/>), the Amazon Premium Support Knowledge Center article titled **How can I copy large amounts of data from Amazon S3 into HDFS on my Amazon EMR cluster?**

(<https://aws.amazon.com/premiumsupport/knowledge-center/copy-s3-hdfs-emr/>), the Amazon EMR Release Guide titled **S3DistCp (s3-dist-cp)**

(<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/UsingEMR_s3distcp.html>), the AWS Big Data blog titled **Seven Tips for Using S3DistCp on Amazon EMR to Move Data Efficiently Between HDFS and Amazon S3** (<https://aws.amazon.com/blogs/big-data/seven-tips-for-using-s3distcp-on-amazon-emr-to-move-data-efficiently-between-hdfs-and-amazon-s3/>), and the AWS Solutions page titled **Real-Time IoT Device Monitoring with Kinesis Data Analytics** (<https://aws.amazon.com/solutions/real-time-iot-device-monitoring-with-kinesis/>)

Question: 5

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Select a collection system that handles the frequency, volume, and source of data**

**Domain:** Data collection systems

**Question text**:

You are a data scientist working for a rental car company that has fleets of rental cars across the globe. Each car is equipped with IoT sensors that report important information about the car’s functioning, location, service levels, mileage, etc.

You have been tasked with determining how rental efficiency has changed over time for fleets in certain cities across the US. This solution requires you to look back at several years of historical data collected by your IoT sensors.

Your management team wishes to perform Key Performance Indicator (KPI) analysis on the rental car data through visualization using business intelligence (BI) tools. They will use this analysis and visualization to make management decisions on how to keep their fleet of rental cars at optimum levels of service and use. They will also use the KPI analysis to assess the performance of their regional management teams for each city for which you collect data.

What collection system best fits this use case?

1. IoT device sensor data -> Kinesis Data Firehose -> S3 -> Glue -> S3 Data Lake -> Athena
2. IoT device sensor data -> Kinesis Data Firehose -> Kinesis Data Analytics -> Kinesis Data Firehose -> Redshift -> QuickSight
3. IoT device sensor data -> RDS -> Database Migration Service -> S3 -> Glue -> S3 Data Lake -> Athena
4. IoT device sensor data -> Kinesis Data Streams -> Kinesis Data Analytics -> S3 Data Lake -> QuickSight

**Answer:** A

**Explanation:**

Option A is correct. This data collection system architecture is best suited to batch consumption of stream data. Crawling the S3 data using Glue and then using a Glue job to write the data to an S3 data lake to then be queried by Athena allows you to produce aggregate data analytics. These data can help you build your KPI dashboard.

Option B is incorrect. This data collection system architecture is best suited to real-time consumption of data. Batch sensor data is better processed with a Glue ETL job versus a Kinesis Data Analytics application.

Option C is incorrect. This type of data collection infrastructure is best used for streaming transactional data from existing relational data stores. There is no need for an RDS instance in this data collection system since we can use a data lake to house the historical data and use Amazon Athena to query the data lake.

Option D is incorrect. Kinesis Data Analytics cannot write directly to S3; it only writes to a Kinesis data stream, a Kinesis Data Firehose delivery stream, or a Lambda function.

**Reference:**

Please see the Amazon Kinesis Data Analytics for SQL Applications developer guide titled **Configuring Application Output**

(<https://docs.aws.amazon.com/kinesisanalytics/latest/dev/how-it-works-output.html>), the AWS Streaming Data page titled **What is Streaming Data?** (<https://aws.amazon.com/streaming-data/>), the **AWS Database Migration Service FAQs** (<https://aws.amazon.com/dms/faqs/>), the **Amazon Kinesis Data Analytics FAQs (**[**https://aws.amazon.com/kinesis/data-analytics/faqs/**](https://aws.amazon.com/kinesis/data-analytics/faqs/)), the **Amazon Kinesis Data Streams FAQs (**[**https://aws.amazon.com/kinesis/data-streams/faqs/**](https://aws.amazon.com/kinesis/data-streams/faqs/)), , the Amazon Kinesis Data Firehose developer guide titles **What is Amazon Kinesis Data Firehose? (**[**https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html#data-flow-diagrams**](https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html#data-flow-diagrams)), the AWS Glue developer guide titled **AWS Glue Concepts** (<https://docs.aws.amazon.com/glue/latest/dg/components-key-concepts.html>), and the **Amazon Kinesis Data Firehose FAQs (**[**https://aws.amazon.com/kinesis/data-firehose/faqs/**](https://aws.amazon.com/kinesis/data-firehose/faqs/))

Question: 6

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Select a collection system that addresses the key properties of data, such as order, format, and compression**

**Domain:** Data collection systems

**Question text**:

You are a data scientist working for a mobile gaming company that is developing a new mobile gaming app that will need to handle thousands of messages per second arriving in your application data store. Due to the user interactivity of your game, all changes to the game datastore must be recorded with a before-change and after-change view of the data record. These data store changes will be used to deliver a near-real-time usage dashboard of the app for your management team.

What application collection system infrastructure best delivers these capabilities in the most performant and cost effective way?

1. Kinesis Firehose -> S3 -> EMR with Spark -> S3 -> Redshift -> QuickSight
2. DynamoDB -> DynamoDB Streams -> Lambda -> Kinesis Firehose -> Redshift -> QuickSight
3. Kinesis Firehose -> Aurora MySQL -> Lambda -> Kinesis Firehose -> Redshift -> QuickSight
4. Kinesis Data Streams -> Aurora MySQL -> Lambda->Kinesis Firehose -> Redshift -> QuickSight

**Answer:** B

**Explanation:**

Option A is incorrect because none of the collection systems listed easily allow for the before-change and after-change views of your applications data store changes. Also, there is no data store other than S3 in the listed collection system components. S3 is not the most cost effective data store for this type of application.

Option B is correct. Your application will write its game activity data to your DynamoDB table which will have DynamoDB streams enabled. DynamoDB Streams will record both the new and old (or before and after) images of any item in the DynamoDB table that is changed. Your Lambda function will be triggered by DynamoDB Streams. Your Lambda function will use the Firehose client to write to your Firehose stream. Firehose will stream your data to Redshift. Quicksite will visualize your data in near-real-time.

Option C is incorrect. Kinesis Firehose does not have the capability to write directly to Aurora. You would have to write your stream data to S3 then write a Lambda function, triggered on each write, to consume the data stream and then write the stream data to your Aurora data store. You could also use the Amazon Database Migration service to move your data from S3 to Aurora. Also, you would have to write custom code to record the before-change information.

Option D is incorrect. Kinesis Data Streams does not have the capability to write directly to Aurora. You would have to write a Kinesis consumer client using the Kinesis Consumer Library (KCL) to consume the data stream and then write the stream data to your Aurora data store. Also, you would have to write custom code to record the before-change information.

**Reference:**

Please see the Amazon DynamoDB developer guide titled **Capturing Table Activity with DynamoDB Streams**

(<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html#Streams.Processing>), the Medium.com article titled **Data Transfer Dynamodb to Redshift**

(<https://medium.com/@ananthsrinivas/data-transfer-dynamodb-to-redshift-5424d7fdf673>), the **Amazon Redshift overview page** (<https://aws.amazon.com/redshift/>), the AWS Database blog titled **Stream data into an Aurora PostgreSQL Database using AWS DMS and Amazon Kinesis Data Firehose** (<https://aws.amazon.com/blogs/database/stream-data-into-an-aurora-postgresql-database-using-aws-dms-and-amazon-kinesis-data-firehose/>), the AWS Database blog titled **Capturing Data Changes in Amazon Aurora Using AWS Lambda**

(<https://aws.amazon.com/blogs/database/capturing-data-changes-in-amazon-aurora-using-aws-lambda/>), the **Kinesis Data Firehose overview page** (<https://aws.amazon.com/kinesis/data-firehose/>), and the **Kinesis Data Streams overview page** (<https://aws.amazon.com/kinesis/data-streams/>)

Question: 7

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Determine data access and retrieval patterns**

**Domain:** Storage and data management concerns

**Question text**:

You are a data scientist working for an online retail electronics chain. Their website receives very heavy traffic during certain months of the year, but these heavy traffic periods fluctuate over time. Your firm wants to get a better understanding of these patterns. Therefore, they have decided to build a traffic prediction machine learning model based on click-stream data.

Your task is to capture the click-stream data and store it in S3 for use as training and inference data in the machine learning model. You have built a streaming data capture system using Kinesis Data Streams and its Kinesis Producer Library (KPL) for your click-stream data capture component. You are using collection batching in your KPL code to improve performance of your collection system. Exception and failure handling is very important to your collection process, since losing click-stream data will compromise the integrity of your machine learning model data.

How can you best handle failures in your KPL component?

1. For each record processed by your KPL component trigger a Lambda function that ensures proper sequencing of the records processed
2. Kinesis Data Streams synchronously replicates your data across three availability zones. Take advantage of this to recover from failed record processing with retry logic.
3. With the KPL PutRecords operation, if a put fails, the record is automatically put back into the KPL buffer and retried.
4. With the KPL PutRecords operation, if a put fails, the record is automatically rolled back, giving you the option to use retry logic in your KPL code.

**Answer:** C

**Explanation:**

Option A is incorrect because this implementation would be very inefficient. Also, you would be writing logic that the KPL gives you in its PutRecords operation.

Option B is incorrect. While Kinesis Data Streams does synchronously replicate your data across three availability zones, this capability would not give you the opportunity to recover from failed record puts into the stream since the failed records would not be replicated across the three availability zones.

Option C is correct. You would use the Kinesis Producer Library (KPL) PutRecords method in your KPL code to send click-stream records into your Kinesis Data Streams stream. The KPL PutRecords automatically adds any failed records back into the KPL buffer so it can be retried.

Option D is incorrect. The KPL PutRecords automatically adds any failed records back into the KPL buffer so it can be retried. You don’t need to implement retry logic in your code since the failed record is placed back into the KPL buffer. Your normal buffer processing logic will process the KPL buffer data without changes needed for retry.

**Reference:**

Please see the Amazon Kinesis Streams developer guide titled **KPL Key Concepts**

(<https://docs.aws.amazon.com/streams/latest/dev/kinesis-kpl-concepts.html>), the Amazon Kinesis Streams developer guide titled **Developing Producers Using the Amazon Kinesis Producer Library** (<https://docs.aws.amazon.com/streams/latest/dev/developing-producers-with-kpl.html>), the Amazon Kinesis Streams developer guide titled **KPL Retries and Rate Limiting** (<https://docs.aws.amazon.com/streams/latest/dev/kinesis-producer-adv-retries-rate-limiting.html>), the AWS Big Data blog titled **Implementing Efficient and Reliable Producers with the Amazon Kinesis Producer Library**

(<https://aws.amazon.com/blogs/big-data/implementing-efficient-and-reliable-producers-with-the-amazon-kinesis-producer-library/>), the **AWS Real-time Analytics on AWS overview page** (<https://aws.amazon.com/big-data/real-time-analytics-featured-partners/>), and the AWS Big Data blog titled **Create real-time clickstream sessions and run analytics with Amazon Kinesis Data Analytics, AWS Glue, and Amazon Athena (**[**https://aws.amazon.com/blogs/big-data/create-real-time-clickstream-sessions-and-run-analytics-with-amazon-kinesis-data-analytics-aws-glue-and-amazon-athena/**](https://aws.amazon.com/blogs/big-data/create-real-time-clickstream-sessions-and-run-analytics-with-amazon-kinesis-data-analytics-aws-glue-and-amazon-athena/))

Question: 8

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Determine the operational characteristics of an analysis and visualization solution**

**Domain:** Analysis and visualization of analytical data

**Question text**:

You are a data scientist working for a large city that has implemented an electric scooter ride sharing system. Each electric scooter is equipped with IoT sensors that report the scooter’s location, whether it is currently rented out, current renter, battery level, speed of travel, etc.

You have been tasked with determining scooter density of location throughout the city and redistributing scooters if some areas of the city are overpopulated with scooters while other areas are underpopulated. This solution requires real-time IoT data to be ingested into your data collection system.

Your management team wishes to perform real-time analysis on the scooter data through visualization using business intelligence (BI) tools. They will use this analysis and visualization to make management decisions on how to keep their fleet of scooters at optimum levels of service and use.

What collection system best fits this use case?

1. IoT device sensor data -> Kinesis Data Firehose -> S3 -> Glue -> S3 Data Lake -> Athena
2. IoT device sensor data -> Kinesis Data Firehose -> Kinesis Data Analytics -> Kinesis Data Firehose -> Redshift -> QuickSight
3. IoT device sensor data -> RDS -> Database Migration Service -> S3 -> Glue -> S3 Data Lake -> Athena
4. IoT device sensor data -> Kinesis Data Streams -> Kinesis Data Analytics -> S3 Data Lake -> QuickSight

**Answer:** B

**Explanation:**

Option A is incorrect. This data collection system architecture is better suited to batch consumption of stream data. Crawling the S3 data using Glue and then using a Glue job to write the data to an S3 data lake to then be queried by Athena would not allow you to produce real-time analytics. While Glue can process micro-batches, it does not handle streaming data.

Option B is correct. You can use a Kinesis Data Firehose stream to ingest the IoT data, then analyze and filter your data with Kinesis Data Analytics, then direct the analyzed data to another Kinesis Data Firehose stream to load the data into your data warehouse in RedShift. Finally, use QuickSight to produce your visualization and dashboard for your management team.

Option C is incorrect. This type of data collection infrastructure is best used for streaming transactional data from existing relational data stores. There is no need for an RDS instance in this data collection system since the data is transitory in nature.

Option D is incorrect. Kinesis Data Analytics cannot write directly to S3; it only writes to a Kinesis data stream, a Kinesis Data Firehose delivery stream, or a Lambda function.

**Reference:**

Please see the Amazon Kinesis Data Analytics for SQL Applications developer guide titled **Configuring Application Output**

(<https://docs.aws.amazon.com/kinesisanalytics/latest/dev/how-it-works-output.html>), the AWS Streaming Data page titled **What is Streaming Data?** (<https://aws.amazon.com/streaming-data/>), the **AWS Database Migration Service FAQs** (<https://aws.amazon.com/dms/faqs/>), the **Amazon Kinesis Data Analytics FAQs (**[**https://aws.amazon.com/kinesis/data-analytics/faqs/**](https://aws.amazon.com/kinesis/data-analytics/faqs/)), the **Amazon Kinesis Data Streams FAQs (**[**https://aws.amazon.com/kinesis/data-streams/faqs/**](https://aws.amazon.com/kinesis/data-streams/faqs/)), the Amazon Kinesis Data Firehose developer guide titles **What is Amazon Kinesis Data Firehose? (**[**https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html#data-flow-diagrams**](https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html#data-flow-diagrams)), the AWS Glue developer guide titled **AWS Glue Concepts** (<https://docs.aws.amazon.com/glue/latest/dg/components-key-concepts.html>), and the **Amazon Kinesis Data Firehose FAQs (**[**https://aws.amazon.com/kinesis/data-firehose/faqs/**](https://aws.amazon.com/kinesis/data-firehose/faqs/))

Question: 9

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Apply data governance and compliance controls**

**Domain:** Security of the data analysis system

**Question text**:

You are a data scientist working for a medical services company that has a suite of apps available for patients and their doctors to share their medical data. These apps are used to share patient details, MRI and XRAY images, appointment schedules, etc. Because of the importance of this data and its inherent Personally Identifiable Information (PII), your data collection system needs to be secure and the system cannot suffer lost data, process data out of order, or duplicate data.

Which data collection system(s) gives you the security and data integrity your requirements demand? (SELECT 2)

1. Apache Kafka/Amazon MSK
2. SQS (FIFO)
3. SQS (Standard)
4. Kinesis Data Firehose
5. Kinesis Data Streams
6. DynamoDB Streams

**Answers:** B, F

**Explanation:**

Option A is incorrect. Apache Kafka/Amazon MSK allows you to process streaming data. It guarantees the correct order of delivery of your data messages, but it uses the “at-least-once” delivery method. At-least-once delivery means that the message will not be lost, but the message may be delivered to a consumer more than once.

Option B is correct. SQS in the FIFO mode guarantees the correct order of delivery of your data messages and it uses the “exactly-once” delivery method. Exactly-once means that all messages will be delivered exactly one time. No message losses, no duplicate data.

Option C is incorrect. SQS in the Standard mode does not guarantee the correct order of delivery of your data messages and it uses the “at-least-once” delivery method. At-least-once delivery means that the message will not be lost, but the message may be delivered to a consumer more than once.

Option D is incorrect. Kinesis Data Firehose does not guarantee the correct order of delivery of your data messages and it uses the “at-least-once” delivery method. At-least-once delivery means that the message will not be lost, but the message may be delivered to a consumer more than once.

Option E is incorrect. Kinesis Data Streams guarantees the correct order of delivery of your data messages, but it uses the “at-least-once” delivery method. At-least-once delivery means that the message will not be lost, but the message may be delivered to a consumer more than once.

Option F is correct. DynamoDB Streams guarantees the correct order of delivery of your data messages and it uses the “exactly-once” delivery method. Exactly-once means that all messages will be delivered exactly one time. No message losses, no duplicate data.

**Reference:**

Please see the **Amazon Managed Streaming for Apache Kafka (Amazon MSK)** overview page (<https://aws.amazon.com/msk/>), the Amazon Simple Queue Service developer guide titled **Amazon SQS Standard Queues** (<https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/standard-queues.html>), the Amazon Simple Queue Service developer guide titled **Amazon SQS FIFO (First-In-First-Out) Queues** (<https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html>), the Amazon DynamoDB developer guide titled **Capturing Table Activity with DynamoDB Streams** (<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html>), the Amazon Kinesis Data Streams developer guide titled **Handling Duplicate Records** (<https://docs.aws.amazon.com/streams/latest/dev/kinesis-record-processor-duplicates.html>), the **Amazon Kinesis Data Firehose FAQs** (<https://aws.amazon.com/kinesis/data-firehose/faqs/>), and the **Amazon Kinesis Data Streams FAQs** (<https://aws.amazon.com/kinesis/data-streams/faqs/>)

Question: 10

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Determine the operational characteristics of a storage solution for analytics**

**Domain:** Storage and data management concerns

**Question text**:

You work for a ski resort corporation. Your company is developing a lift ticket system for mobile devices that allows skiers and snowboarders to use their phone as their lift ticket. The ski resort corporation owns many resorts around the world. The lift ticketing system needs to handle users who move from resort to resort throughout any given time period. Resort customers can also purchase packages where they can ski or snowboard at a defined list (a subset of the total) of several different resorts across the globe as part of their package.

The storage system for the lift ticket mobile application has to handle large fluctuations in volume. The data collected from the devices and stored in the data store is small in size, but the system must provide the data at low latency and high throughput. It also has to authenticate users through their mobile device registered facial recognition service, so that users can’t share a lift ticket by sharing their mobile devices.

What storage system is the best fit for this system?

1. Neptune
2. RDS
3. DynamoDB
4. ElastiCache
5. Redshift
6. S3

**Answer:** C

**Explanation:**

Option A is incorrect. Neptune is a graph database engine optimized for storing billions of relationships and querying the graph data. Graph databases like Neptune are best leveraged for use cases like social networking, recommendation engines, and fraud detection, where you need to create relationships between data and quickly query these relationships. Your application is more operational in nature and therefore requires a database that fits that profile.

Option B is incorrect. While RDS is operational in nature, it is bounded by instance and storage size limits. Also, while offering a multi-availability zone (multi-AZ) capability, RDS does not scale globally as easily as DynamoDB. Therefore, DynamoDB is a better choice for your global availability requirements.

Option C is correct. DynamoDB offers single-digit millisecond latency at scale. It also scales horizontally for high performance at any size data store. Finally, DynamoDB offers global tables for multi-region replication of your data, which you’ll need for your globally dispersed user base and ski resort locations.

Option D is incorrect. ElastiCache is an in-memory caching system that, alone, would not have the persistence needed for your system.

Option E is incorrect. Redshift is a columnar storage database best used for data warehouse use cases. Since your application requires an operational data store, Redshift would not be the correct choice.

Option F is incorrect. S3 is used for structured and unstructured data. Querying S3 using Athena or Redshift Spectrum allow for relatively quick queries, but not fast enough for an operational application like your ski resort mobile application requirements.

**Reference:**

Please see the **Amazon DynamoDB FAQs** (<https://aws.amazon.com/dynamodb/faqs/>), the **Amazon Neptune overview** page (<https://aws.amazon.com/neptune/>), the Amazon DynamoDB developer guide titled **Global Tables: Multi-Region Replication with DynamoDB** (<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GlobalTables.html>), the **Amazon RDS FAQs** (<https://aws.amazon.com/rds/faqs/>), the **Amazon S3 FAQs** (<https://aws.amazon.com/s3/faqs/>), the **Amazon Redshift FAQs** (<https://aws.amazon.com/redshift/faqs/>), and the **Amazon ElastiCache FAQs** (<https://aws.amazon.com/elasticache/faqs/>)

Question: 11

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Determine data access and retrieval patterns**

**Domain:** Storage and data management concerns

**Question text**:

You work for a mobile gaming company that has developed a word puzzle game that allows multiple users to challenge each other to complete a crossword puzzle type of game board. This interactive game works on mobile devices and web browsers. You have a world-wide user base that can play against each other no matter where each player is located.

You now need to create a leaderboard component of the game architecture where players can look at the daily point leaders for the day, week, or other timeframes. Each time a player accumulates points, the points counter for that player needs to be updated in real-time. This leaderboard data is transient in that it only needs to be stored for a limited duration.

Which of the following architectures best suits your data access and retrieval patterns using the simplest, most efficient approach?

1. Data sources -> Kinesis Data Streams -> Spark Streaming on EMR -> ElastiCache Redis -> DynamoDB
2. Data Sources -> Kinesis Data Firehose -> S3 -> Athena
3. Data sources -> Kinesis Data Streams -> Spark Streaming on EMR -> ElastiCache Memcached -> DynamoDB
4. Data sources -> Kinesis Data Streams -> Spark Streaming on EMR -> ElastiCache Redis
5. Data sources -> Kinesis Data Firehose -> Spark Streaming on EMR -> ElastiCache Redis -> S3

**Answer:** D

**Explanation:**

Option A is incorrect. While Kinesis Data Streams is the appropriate streaming solution for gathering the streaming player data and loading it onto your EMR cluster, then using Spark Streaming to transform the data into a format that is efficiently stored in ElastiCache Redis. There is no need for DynamoDB based on your data access and retrieval patterns for your application since your leaderboard application data is transient.

Option B is incorrect. Streaming your player data from Kinesis Data Firehose straight to S3 without any caching or transformation won’t give you your leaderboard functionality.

Option C is incorrect. While Kinesis Data Streams is the appropriate streaming solution for gathering the streaming player data and loading it onto your EMR cluster, then using Spark Streaming to transform the data into a format that is efficiently stored in ElastiCache. The Memcached version of ElastiCache does not allow you to easily implement the leaderboard functionality that ElastiCache Redis gives you. So this option is much less efficient.

Option D is correct. Kinesis Data Streams is the appropriate streaming solution for gathering the streaming player data and loading it onto your EMR cluster, then using Spark Streaming to transform the data into a format that is efficiently stored in ElastiCache Redis. You can use the Redis INCR and DECR functions to keep track of user points and the Redis Sorted Set data structure to maintain the leader list sorted by player. You can maintain your real-time ranked leader list by updating each user's score each time it changes.

Option E is incorrect. Based on your data access and retrieval patterns, there is no need for an S3 storage layer in this architecture.

**Reference:**

Please see the **Amazon ElastiCache for Redis overview page** (<https://aws.amazon.com/elasticache/redis/>), the **Amazon ElastiCache for Redis User Guide** (<https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/redis-ug.pdf>), the **RedisLabs Leaderboards page** (<https://redislabs.com/redis-enterprise/use-cases/leaderboards/>), the AWS Database Blog page titled **Build a real-time gaming leaderboard with Amazon ElastiCache for Redis** (<https://aws.amazon.com/blogs/database/building-a-real-time-gaming-leaderboard-with-amazon-elasticache-for-redis/>), and the Amazon ElastiCache for Redis user guide titled **Common ElastiCache Use Cases and How ElastiCache Can Help** (<https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/elasticache-use-cases.html>)

Question: 12

**Main​ ​Topic​ ​:​** Data Analytics

**Sub​ ​Topic​ ​:​ Define a data lifecycle based on usage patterns and business requirements**

**Domain:** Storage and data management concerns

**Question text**:

You work for a car manufacturer who has implemented many sensors into their vehicles such as GPS, lane-assist, braking-assist, temperature/humidity, etc. These cars continuously transmit their structured and unstructured sensor data. You need to build a data collection system to capture their data for use in ad-hoc analytics applications to understand the performance of the cars, the locations traveled to and from, the effectiveness of the lane and brake assist features, etc. You also need to filter and transform the sensor data depending on rules based on parameters such as temperature readings. The sensor data needs to be stored indefinitely, however you only wish to pay for the analytics processing when you use it.

Which of the following architectures best suits your data lifecycle and usage patterns using the simplest, most efficient approach?

1. Sensor data -> Kinesis Data Streams -> IoT Core -> S3 -> Athena
2. Sensor data -> Kinesis Data Firehose -> IoT Core -> S3 -> Athena
3. Sensor data -> Kinesis Data Streams -> IoT Core -> Kinesis Data Firehose -> RedShift -> QuickSight
4. Sensor data -> IoT Core -> S3 -> Athena
5. Sensor data -> Kinesis Data Firehose -> S3 -> Athena

**Answer:** D

**Explanation:**

Option A is incorrect. While Kinesis Data Streams can be used to ingest IoT sensor data, it is an unnecessary component in your data collection architecture since IoT Core can do the sensor data ingestion task.

Option B is incorrect. While Kinesis Data Firehose can be used to ingest IoT sensor data, it is an unnecessary component in your data collection architecture since IoT Core can do the sensor data ingestion task.

Option C is incorrect. This data collection architecture has unnecessary components. While Kinesis Data Streams can be used to ingest IoT sensor data, it is an unnecessary component in your data collection architecture since IoT Core can do the sensor data ingestion task. RedShift is not the optimal data store for your IoT sensor data in this scenario. RedShift is better suited for storing structured data, but you have both structured and unstructured data.

Option D is correct. The simplest data collection architecture that meets your data lifecycle and usage patterns uses IoT Core to ingest the sensor data. Also, IoT Core is used to run a rules-based filtering and transformation set of functions. IoT Core then streams the sensor data to S3 where you house your data lake. You then use Athena to run your ad-hoc queries on your sensor data, taking advantage of Athena’s serverless query service so that you only pay for the service when you use it.

Option E is incorrect. This data collection architecture gives you a simple process flow to get your sensor data into your S3 data lake. However, it lacks the rules-based filtering and transformation set of functions. You would have to implement these functions in a Lambda function, which would make this data collection architecture less efficient than using the IoT Core service to address this requirement.

**Reference:**

Please see the **AWS IoT Core overview page** (<https://aws.amazon.com/iot-core/>), the AWS Big Data blog titled **Integrating IoT Events into Your Analytic Platform**

(<https://aws.amazon.com/blogs/big-data/integrating-iot-events-into-your-analytic-platform/>), the blog titled **Athena Vs Redshift: An Amazonian Battle Or Performance And Scale** (<https://blog.panoply.io/an-amazonian-battle-comparing-athena-and-redshift>), and the **Amazon Athena overview page** (<https://aws.amazon.com/athena/>)