CertyIQ

CertyIQ

Premium exam material

Get certification quickly with the CertyIQ Premium exam material.
Everything you need to prepare, learn & pass your certification exam easily. Lifetime free updates
First attempt guaranteed success.

https://www.CertyIQ.com

About CertyIQ

We here at CertyIQ eventually got enough of the industry's greedy exam paid for. Our team of IT professionals comes with years of experience in the IT industry Prior to training CertIQ we worked in test areas where we observed the horrors of the paywall exam preparation system.

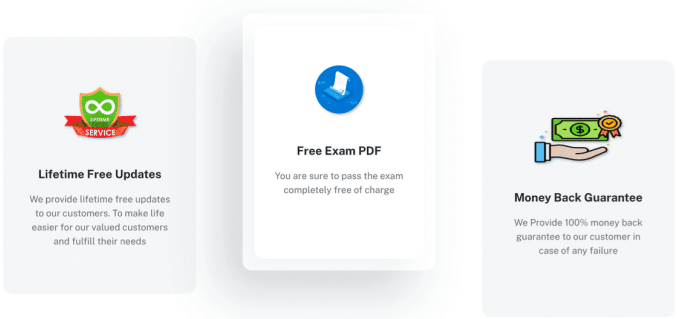
The misuse of the preparation system has left our team disillusioned. And for that reason, we decided it was time to make a difference. We had to make In this way, CertyIQ was created to provide quality materials without stealing from everyday people who are trying to make a living.

Doubt Support

We have developed a very scalable solution using which we are able to solve 400+ doubts every single day with an average rating of 4.8 out of 5.

https://www.certyiq.com

Mail us on - certyiqofficial@gmail.com



John

October 19, 2022

★★★★★

Thanks you so much for your help. I scored 972 in my exam today. More than 90% were from your PDFs!

Dana

September 04, 2022

★★★★★

Thanks a lot for this updated AZ-900 Q&A. I just passed my exam and got 974, I followed both of your Az-900 videos and the 6 PDF, the PDFs are very much valid, all answers are correct. Could you please create a similar video/PDF for DP900, your content/PDF's is really awesome. The team did a really good job. Thank You 😊.

Ahamed Shibly

2 months ago

★★★★★

Customer support is realy fast and helpful, I just finished my exam and this video along with the 6 PDF helped me pass! Definitely recommend getting the PDFs. Thank you!

October 22, 2022

★★★★★

Passed my exam today with 891 marks. Out of 52 questions, 51 were from certyiq PDFs including Contoso case study.
Thank You certyiq team!

Henry Rome

2 months ago

★★★★★

These questions are real and 100 % valid. Thank you so much for your efforts, also your 4 PDFs are awesome, I passed the DP900 exam on 1 Sept. With 968 marks. Thanks a lot, buddy!

Esmaria

2 months ago

★★★★★

Simple easy to understand explanations. To anyone out there wanting to write AZ900, I highly recommend 6 PDF's.Thank you so much, appreciate all your hard work in having such great content. Passed my exam Today - 3 September with 942 score.

Google

(Professional Data Engineer)

Professional Data Engineer on Google Cloud Platform

Total: **397 Questions**

Link: https://certyiq.com/papers/google/professional-data-engineer

Question: 1 CertyIQ

Your company built a TensorFlow neutral-network model with a large number of neurons and layers. The model fits well for the training data. However, when tested against new data, it performs poorly. What method can you employ to address this?

A. Threading
B. Serialization
C. Dropout Methods
D. Dimensionality Reduction

Answer: C

Explanation:

The scenario describes a classic overfitting problem in machine learning: a model performs well on training data but poorly on unseen data. The key to addressing overfitting is to introduce techniques that prevent the model from memorizing the training set's specific patterns and instead encourage it to generalize to new data.

Dropout methods are specifically designed to combat overfitting in neural networks. They work by randomly "dropping out" (deactivating) a proportion of neurons during each training iteration. This forces the network to learn redundant representations, preventing reliance on specific neurons and encouraging more robust feature extraction. The random deactivation prevents complex co-adaptations between neurons, promoting simpler models. Consequently, the model is less likely to memorize the training data and more capable of generalizing to unseen data. Therefore, option C, dropout methods, is the most effective choice to reduce overfitting.

Threading (option A) deals with parallelizing processes within a system, serialization (option B) involves converting an object into a data stream for storage or transmission, and dimensionality reduction (option D) aims to reduce the number of variables in a dataset. These options are not directly related to the problem of overfitting. They address other performance and data handling needs but not model generalization.

For further information:

Dropout: https://jmlr.org/papers/v15/srivastava14a.html (Original Dropout paper)
Overfitting and Regularization: https://developers.google.com/machine-learning/crash-course/generalization/peril-of-overfitting (Google's Machine Learning Crash Course)

Question: 2 CertyIQ

You are building a model to make clothing recommendations. You know a user's fashion preference is likely to change over time, so you build a data pipeline to stream new data back to the model as it becomes available. How should you use this data to train the model?

A. Continuously retrain the model on just the new data.
B. Continuously retrain the model on a combination of existing data and the new data.
C. Train on the existing data while using the new data as your test set.
D. Train on the new data while using the existing data as your test set.

Answer: B

Explanation:

The correct approach, option B, involves continuously retraining the model using a combination of both existing and new data. This strategy addresses the dynamic nature of user fashion preferences as indicated in

the scenario. Continuously retraining solely on new data (option A) would lead to catastrophic forgetting, where the model loses its knowledge of past preferences, impacting its overall performance. Treating new data as a test set (option C) or vice versa (option D) wouldn't allow the model to learn from the most recent user behavior and adapt over time. Combining existing and new data enables incremental learning and maintains the model's accuracy. It allows the model to retain knowledge of past trends while incorporating the latest shifts in fashion preferences. This avoids the model becoming overly sensitive to short-term, potentially volatile trends. This strategy aligns with the concept of continual learning in machine learning. It ensures the model remains relevant and accurate over time by incorporating up-to-date data. The implementation might involve regular updates, batch processing of new data, or using streaming data pipelines with retraining triggers. This retraining would likely require version control to track model changes and ensure reproducibility. Google Cloud Platform (GCP) offers services like Vertex AI and Dataflow that are well-suited for such iterative model training pipelines. https://cloud.google.com/vertex-ai and https://cloud.google.com/dataflow provide details about these services. These links provide resources to understand and apply these services for machine learning pipelines.

Question: 3 CertyIQ

You designed a database for patient records as a pilot project to cover a few hundred patients in three clinics. Your design used a single database table to represent all patients and their visits, and you used self-joins to generate reports. The server resource utilization was at 50%. Since then, the scope of the project has expanded. The database must now store 100 times more patient records. You can no longer run the reports, because they either take too long or they encounter errors with insufficient compute resources. How should you adjust the database design?

A. Add capacity (memory and disk space) to the database server by the order of 200.

B. Shard the tables into smaller ones based on date ranges, and only generate reports with prespecified date ranges.

C. Normalize the master patient-record table into the patient table and the visits table, and create other necessary tables to avoid self-join.

D. Partition the table into smaller tables, with one for each clinic. Run queries against the smaller table pairs, and use unions for consolidated reports.

Answer: C

Explanation:

The correct answer is **C. Normalize the master patient-record table into the patient table and the visits table, and create other necessary tables to avoid self-join.** Here's why:

Option A, adding massive resources, is a short-term fix and not scalable. Increasing resources by 200 times is costly and doesn't address the fundamental design problem. Option B, sharding by date, can improve query performance for specific date ranges but doesn't tackle the core issue of the inefficient data model. Option D, partitioning by clinic, also provides some performance improvement by focusing queries on specific clinics but will still struggle with the self-joins when generating cross-clinic reports and does not address the data model flaw.

Option C directly addresses the root cause: a poorly designed denormalized database. The original single table with patient and visit data leads to massive redundancy and slow queries. By normalizing the data into separate patient and visit tables, we eliminate redundant data and avoid the costly self-joins. The patient table would store patient-specific information, and the visit table would contain details of each visit, referencing the patient table through a foreign key relationship. This allows for more efficient storage and retrieval of data, including faster, less resource-intensive reporting. Normalizing the data reduces data duplication, and simplifies the overall structure, allowing for a much more scalable solution. Creating additional tables to support the patient-visit relationship will also improve query speed and resource usage.

Here are some authoritative links for further reading:

Database Normalization: https://en.wikipedia.org/wiki/Database_normalization
Database Design Principles: https://cloud.google.com/architecture/database-design-principles
Relational Databases on Google Cloud: https://cloud.google.com/sql/
Normalization for Scalability:https://www.geeksforgeeks.org/database-normalization-in-dbms/

In summary, while some performance enhancements might be gained by sharding or partitioning, only normalization provides a sustainable solution and efficient foundation for future growth and reduces data redundancy.

Question: 4 CertyIQ

You create an important report for your large team in Google Data Studio 360. The report uses Google BigQuery as its data source. You notice that visualizations are not showing data that is less than 1 hour old. What should you do?

A. Disable caching by editing the report settings.
B. Disable caching in BigQuery by editing table details.
C. Refresh your browser tab showing the visualizations.
D. Clear your browser history for the past hour then reload the tab showing the virtualizations.

Answer: A

Explanation:

The correct answer is **A. Disable caching by editing the report settings.**

Here's why: Google Data Studio 360, by default, caches data from its data sources, including BigQuery, to improve performance and reduce query costs. This caching mechanism means that when a report is loaded, Data Studio might not fetch the latest data from BigQuery, leading to a delay in seeing fresh results, as indicated by the visualizations not showing data less than 1 hour old.

Options B, C, and D are incorrect. Option B is incorrect because BigQuery caching does not affect Data Studio's own caching behavior. Option C, refreshing the browser tab, can help with minor browser issues but will still show the cached data from Data Studio, not the fresh data from BigQuery. Option D, clearing browser history, only impacts the client's local cache and does not interact with Data Studio's data caching layer.

Disabling caching in Data Studio will force it to query BigQuery each time a report is viewed, ensuring the visualizations reflect the most up-to-date data. This setting is typically found within the data source connection settings within the Data Studio report itself. While this might increase query costs and loading times, it is crucial when real-time data visibility is essential for reports.

It's important to note that disabling caching is often only necessary when data changes frequently and timely updates are critical. If data changes less frequently, or real-time data is not required, using caching may be a preferred approach.

Authoritative links for further research:

Data Studio Data Freshness: https://support.google.com/datastudio/answer/7020804?hl=en - This Google Support article provides detailed information on how Data Studio handles data freshness and caching.
BigQuery Caching: https://cloud.google.com/bigquery/docs/query-caching - This Google Cloud documentation explains BigQuery's query caching mechanism, which is separate from Data Studio's caching.

Question: 5 CertyIQ

An external customer provides you with a daily dump of data from their database. The data flows into Google Cloud Storage GCS as comma-separated values
(CSV) files. You want to analyze this data in Google BigQuery, but the data could have rows that are formatted incorrectly or corrupted. How should you build this pipeline?

A. Use federated data sources, and check data in the SQL query.

B. Enable BigQuery monitoring in Google Stackdriver and create an alert.

C. Import the data into BigQuery using the gcloud CLI and set max_bad_records to 0.

D. Run a Google Cloud Dataflow batch pipeline to import the data into BigQuery, and push errors to another dead-letter table for analysis.

Answer: D

Explanation:

The correct answer is D because it offers a robust and scalable solution for handling potentially corrupted data within a BigQuery data ingestion pipeline. Dataflow is designed for ETL (Extract, Transform, Load) operations, making it ideal for processing the CSV data before it lands in BigQuery. By using a Dataflow pipeline, we can implement error handling and data validation logic. The pipeline can be configured to parse each CSV row, identify invalid or corrupted records, and instead of failing the entire import, write the bad records to a separate "dead-letter" table. This allows for detailed analysis of data quality issues without disrupting the ingestion of valid data.

Option A (federated data sources) is not suitable because while it allows querying data residing in GCS directly, it doesn't address data cleansing or error handling. Queries will fail upon encountering incorrectly formatted rows.

Option B (BigQuery monitoring) is helpful for identifying issues post-ingestion but does not prevent bad data from entering BigQuery in the first place. It's reactive, not proactive.

Option C (gcloud CLI with max_bad_records=0) is the opposite of what we want. Setting max_bad_records=0 will cause the entire import to fail if even a single bad record is encountered.

Dataflow's ability to handle errors and route them to a dead-letter queue provides a graceful and informative mechanism for managing imperfect data sources. This approach minimizes data loss and allows for continuous loading of usable data. It provides a pathway for analyzing and rectifying data quality issues in the incoming CSV files.

For further research:

Google Cloud Dataflow: https://cloud.google.com/dataflow/docs
BigQuery data loading: https://cloud.google.com/bigquery/docs/loading-data
Dataflow Error Handling: https://cloud.google.com/dataflow/docs/guides/managing-your-pipeline#handling-errors

Question: 6 CertyIQ

Your weather app queries a database every 15 minutes to get the current temperature. The frontend is powered by Google App Engine and server millions of users. How should you design the frontend to respond to a database failure?

A. Issue a command to restart the database servers.

B. Retry the query with exponential backoff, up to a cap of 15 minutes.

C. Retry the query every second until it comes back online to minimize staleness of data.

D. Reduce the query frequency to once every hour until the database comes back online.

Answer: B

Explanation:

The correct answer is B, retrying the query with exponential backoff up to a 15-minute cap. This strategy addresses database failures gracefully while minimizing the load on the failing system and ensuring application responsiveness.

Option A, restarting database servers, is an operational task, not a frontend concern. The frontend should handle failure scenarios without needing to manage infrastructure. Option C, retrying every second, would create a "thundering herd" problem, overwhelming the failing database with excessive requests. This could exacerbate the issue and prolong the outage. Option D, reducing the frequency to once an hour, significantly degrades user experience by providing stale data for extended periods.

Exponential backoff, on the other hand, starts with short retry intervals (e.g., seconds) and increases them gradually. This avoids overloading the database when it's struggling, allowing it time to recover. Limiting the backoff to 15 minutes ensures the application doesn't retry indefinitely and that data becomes available within the expected refresh window of 15 minutes once the database recovers. This approach aligns with principles of fault tolerance and resilience in cloud applications. It minimizes the impact of a failure on the user experience, keeps resources in check, and is generally considered best practice for handling transient failures in distributed systems.

Here are authoritative resources for further reading:

Google Cloud's documentation on error handling and retry strategies: https://cloud.google.com/architecture/best-practices-for-handling-errors
AWS documentation on exponential backoff: https://aws.amazon.com/blogs/architecture/exponential-backoff-and-jitter/ (While AWS-specific, the principles apply broadly)
Microsoft's documentation on retry patterns: https://learn.microsoft.com/en-us/azure/architecture/patterns/retry (Again, while Azure-specific, the retry principles apply broadly)

Question: 7 CertyIQ

You are creating a model to predict housing prices. Due to budget constraints, you must run it on a single resource-constrained virtual machine. Which learning algorithm should you use?

A. Linear regression
B. Logistic classification
C. Recurrent neural network
D. Feedforward neural network

Answer: A

Explanation:

The correct answer is A, Linear Regression. Linear regression is computationally inexpensive, requiring minimal memory and processing power, making it suitable for resource-constrained environments like a single virtual machine with limited resources. It's a simple algorithm that establishes a linear relationship between input features and the target variable (housing price), making it efficient to train and deploy. Unlike complex models like Recurrent Neural Networks (RNNs) and Feedforward Neural Networks (FFNNs) which require significant computational resources for training and inference, linear regression can achieve acceptable performance without demanding a lot of resources. Logistic classification (B) is primarily used for binary

classification problems, not regression like predicting housing prices. Therefore, A, Linear Regression is the most appropriate algorithm given the budget and resource constraints. Its efficiency outweighs the potential gains of more complex models, especially in this situation.

For more information on linear regression and its resource efficiency, refer to these resources:

Google Cloud AI Platform Training Concepts: https://cloud.google.com/ai-platform/training/docs/concepts/training-overview (While not directly about algorithms, it discusses efficient training)
Scikit-learn's documentation on Linear Regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (Provides detailed insights into the algorithm and its practical implementation)
Understanding Resource Constraints in Machine Learning: https://towardsdatascience.com/machine-learning-with-limited-resources-5c65b7bb38f1 (General article on efficient ML strategies)

Question: 8 CertyIQ

You are building new real-time data warehouse for your company and will use Google BigQuery streaming inserts. There is no guarantee that data will only be sent in once but you do have a unique ID for each row of data and an event timestamp. You want to ensure that duplicates are not included while interactively querying data. Which query type should you use?

A. Include ORDER BY DESK on timestamp column and LIMIT to 1.
B. Use GROUP BY on the unique ID column and timestamp column and SUM on the values.
C. Use the LAG window function with PARTITION by unique ID along with WHERE LAG IS NOT NULL.
D. Use the ROW_NUMBER window function with PARTITION by unique ID along with WHERE row equals 1.

Answer: D

Explanation:

Here's a detailed justification for why option D is the correct choice for deduplicating data in BigQuery using streaming inserts, along with explanations of why other options are less suitable:

Why Option D is Correct: Using ROW_NUMBER()

Option D utilizes the ROW_NUMBER() window function, which is ideally suited for deduplication in this scenario. The core idea is to partition the data based on the unique ID column. This essentially groups all rows with the same unique ID together. Then, ROW_NUMBER() assigns a sequential number to each row within each partition, ordered by the timestamp (though not explicitly stated in the query, this is usually the intent). Since the question indicates an event timestamp, we use this to determine which record is the most recent, and thus, should be kept. By filtering the result using WHERE row equals 1, we are retaining only the first row (the latest record) for each unique ID. This effectively removes all but the first row, which corresponds to the most recent occurrence of that unique ID, thus eliminating duplicates. This method is efficient for interactive queries as it performs the deduplication as part of the query itself and is easy to implement.

Why Other Options Are Incorrect:

Option A: ORDER BY and LIMIT: This approach might retrieve a single row but it would not give you one unique row per unique ID. The ORDER BY clause applied to a timestamp, combined with LIMIT 1 across the entire dataset, would simply pick the most recent entry overall. It fails to address deduplication based on unique ID.
Option B: GROUP BY and SUM: While GROUP BY can help organize data, grouping by both unique ID and timestamp would not lead to deduplication; instead it will aggregate values based on those two fields. Then

using sum, we would be aggregating all the values together instead of filtering the rows. This option is appropriate for aggregation and not deduplication.
Option C: LAG with PARTITION BY: The LAG function retrieves data from a previous row in the partition. When paired with WHERE LAG IS NOT NULL, it essentially checks if a previous record exists. However, this does not remove duplicates. It can be used to identify if a previous record exists in each partition, but not filter out the duplicate records.

Cloud Computing Concepts:

Window Functions: ROW_NUMBER() is a window function, which performs calculations across a set of table rows that are related to the current row. This makes it effective for tasks like ranking, deduplication, and moving averages.
BigQuery Streaming Inserts: BigQuery supports streaming inserts for real-time data ingestion. Because of the near real-time nature, there is not an enforcement of uniqueness at the time of the insert, so developers must use queries such as this one to handle the duplicates in the data.
Data Warehousing: The core principle of data warehousing is to have a single source of truth. The goal of deduplication is to ensure that each unique record is represented only once and is an important aspect of ensuring that the data is correct.

Authoritative Links for Further Research:

BigQuery Window Functions: https://cloud.google.com/bigquery/docs/reference/standard-sql/window-functions
BigQuery Streaming Inserts: https://cloud.google.com/bigquery/streaming-data-into-bigquery
BigQuery Deduplication: https://towardsdatascience.com/how-to-deduplicate-data-in-google-bigquery-2905a00d918

Question: 9 CertyIQ

Your company is using WILDCARD tables to query data across multiple tables with similar names. The SQL statement is currently failing with the following error:

```
# Syntax error : Expected end of statement but got "-" at [4:11]
SELECT age
FROM
    bigquery-public-data.noaa_gsod.gsod
WHERE
    age != 99
    AND_TABLE_SUFFIX = '1929'
ORDER BY
    age DESC
```

Which table name will make the SQL statement work correctly?

A. 'bigquery-public-data.noaa_gsod.gsod'
B. bigquery-public-data.noaa_gsod.gsod*
C. 'bigquery-public-data.noaa_gsod.gsod'*
D. 'bigquery-public-data.noaa_gsod.gsod*`

Answer: D

Explanation:

Reference:

https://cloud.google.com/bigquery/docs/wildcard-tables
" target="_blank" style="word-break: break-all;">

Filtering selected tables using _TABLE_SUFFIX

To restrict a query so that it scans only a specified set of tables, use the `_TABLE_SUFFIX` pseudo column in a `WHERE` clause with a condition that is a constant expression.

The `_TABLE_SUFFIX` pseudo column contains the values matched by the table wildcard. For example, the previous sample query, which scans all tables from the 1940s, uses a table wildcard to represent the last digit of the year:

```
FROM
  `bigquery-public-data.noaa_gsod.gsod194*`
```

Question: 10 CertyIQ

Your company is in a highly regulated industry. One of your requirements is to ensure individual users have access only to the minimum amount of information required to do their jobs. You want to enforce this requirement with Google BigQuery. Which three approaches can you take? (Choose three.)

A. Disable writes to certain tables.
B. Restrict access to tables by role.
C. Ensure that the data is encrypted at all times.
D. Restrict BigQuery API access to approved users.
E. Segregate data across multiple tables or databases.
F. Use Google Stackdriver Audit Logging to determine policy violations.

Answer: BDE

Explanation:

The correct options for enforcing the principle of least privilege in BigQuery are **B. Restrict access to tables by role**, **D. Restrict BigQuery API access to approved users**, and **E. Segregate data across multiple tables or datasets.**

Option B aligns directly with role-based access control (RBAC), a fundamental security principle. By assigning roles with specific permissions (e.g., "read-only" on certain tables) to users, you grant only the necessary access for their function. https://cloud.google.com/bigquery/docs/access-control

Option D reinforces access control at the API level. By limiting which users can interact with the BigQuery API, you prevent unauthorized data access or manipulation, regardless of table permissions. This ensures that only validated application or service accounts can interact with BigQuery.
https://cloud.google.com/iam/docs/overview

Option E addresses data exposure by separating sensitive information into different tables or datasets. This allows you to grant granular permissions, ensuring that users only see the data necessary for their work. Different access control policies can then be applied to each table or dataset, minimizing the impact of any security breach. https://cloud.google.com/bigquery/docs/datasets

Option A, disabling writes, only addresses modification and does not restrict read access, which is essential

for least privilege. Option C, encryption, is important for data security but doesn't enforce access control based on user roles. Option F, auditing, helps identify violations but does not implement access control mechanisms. Together, options B, D, and E provide a comprehensive approach to limiting data exposure and implementing least privilege principles in BigQuery.

Question: 11 CertyIQ

You are designing a basket abandonment system for an ecommerce company. The system will send a message to a user based on these rules:
✑ No interaction by the user on the site for 1 hour
Has added more than $30 worth of products to the basket
✑ Has not completed a transaction
You use Google Cloud Dataflow to process the data and decide if a message should be sent. How should you design the pipeline?

A. Use a fixed-time window with a duration of 60 minutes.
B. Use a sliding time window with a duration of 60 minutes.
C. Use a session window with a gap time duration of 60 minutes.
D. Use a global window with a time based trigger with a delay of 60 minutes.

Answer: C

Explanation:

Use a session window with a gap time duration of 60 minutes.

Question: 12 CertyIQ

Your company handles data processing for a number of different clients. Each client prefers to use their own suite of analytics tools, with some allowing direct query access via Google BigQuery. You need to secure the data so that clients cannot see each other's data. You want to ensure appropriate access to the data.
Which three steps should you take? (Choose three.)

A. Load data into different partitions.
B. Load data into a different dataset for each client.
C. Put each client's BigQuery dataset into a different table.
D. Restrict a client's dataset to approved users.
E. Only allow a service account to access the datasets.
F. Use the appropriate identity and access management (IAM) roles for each client's users.

Answer: BDF

Explanation:

The correct answer is BDF because it directly addresses the requirements of data isolation and secure access. Loading data into different datasets for each client (B) is fundamental for achieving logical separation. This ensures that one client's data is completely isolated from another's at the BigQuery level. Restricting a client's dataset to approved users (D) using IAM policies is crucial for granting access only to authorized individuals. This prevents unauthorized access even if users exist within the same organization. Finally, leveraging appropriate IAM roles for each client's users (F) allows for granular control over what actions specific users can perform within their designated datasets. This follows the principle of least privilege, granting only necessary permissions to each user. Options A and C are incorrect because they don't provide

the necessary data isolation or are inefficient for organizing and securing data for multiple clients. Option E, while a security practice, isn't sufficient on its own. Service accounts would need further restriction through IAM to limit access to specific datasets, making option F more directly relevant to the given scenario.

Relevant links for further research:

Google Cloud IAM Overview: https://cloud.google.com/iam/docs/overview
BigQuery Access Control: https://cloud.google.com/bigquery/docs/access-control-overview
BigQuery Datasets: https://cloud.google.com/bigquery/docs/datasets

Question: 13 CertyIQ

You want to process payment transactions in a point-of-sale application that will run on Google Cloud Platform. Your user base could grow exponentially, but you do not want to manage infrastructure scaling.
Which Google database service should you use?

A. Cloud SQL
B. BigQuery
C. Cloud Bigtable
D. Cloud Datastore

Answer: D

Explanation:

The correct answer is **D. Cloud Datastore**. Here's why:

Cloud Datastore is a fully managed, schemaless, NoSQL document database designed for automatic scaling and high availability. This directly addresses the requirement of not wanting to manage infrastructure scaling as the user base grows exponentially. Its serverless nature means Google handles the underlying infrastructure management, allowing you to focus on application development. Unlike relational databases, Datastore's schemaless nature accommodates evolving data structures easily, which is often beneficial in rapidly changing applications.

Cloud SQL (A) is a managed relational database service, while useful, requires more management regarding scaling and doesn't scale as seamlessly as Datastore. BigQuery (B) is an analytical data warehouse primarily designed for large-scale data analysis, not for transactional processing within a point-of-sale system. Cloud Bigtable (C) is a high-performance, scalable NoSQL wide-column store ideal for large datasets and high-throughput applications but is more complex to manage and not the most suitable for document-style transactions.

Therefore, Cloud Datastore's combination of automatic scaling, high availability, and a simple data model make it the most appropriate choice for the given scenario, where the user base is expected to grow dramatically without manual infrastructure intervention.

Authoritative Links:

Cloud Datastore Overview: https://cloud.google.com/datastore/docs/concepts/overview
Cloud Datastore vs. Cloud SQL: https://cloud.google.com/datastore/docs/concepts/choosing-database
Cloud Datastore Scalability: https://cloud.google.com/datastore/docs/concepts/scaling-and-planning

Question: 14 CertyIQ

You want to use a database of information about tissue samples to classify future tissue samples as either normal

or mutated. You are evaluating an unsupervised anomaly detection method for classifying the tissue samples. Which two characteristic support this method? (Choose two.)

A. There are very few occurrences of mutations relative to normal samples.

B. There are roughly equal occurrences of both normal and mutated samples in the database.

C. You expect future mutations to have different features from the mutated samples in the database.

D. You expect future mutations to have similar features to the mutated samples in the database.

E. You already have labels for which samples are mutated and which are normal in the database.

Answer: AC

Explanation:

The correct answer is AC because unsupervised anomaly detection is most effective when dealing with imbalanced datasets and novel anomalies. Option A is correct because the scarcity of mutation samples relative to normal samples makes anomaly detection a suitable approach. Since normal samples are prevalent, the algorithm learns their patterns well, making deviations (mutations) easily identifiable as outliers. In contrast, Option B, with roughly equal samples, suggests a supervised learning scenario where you could train a model to directly classify normal and mutated samples. Option C aligns with the core principle of anomaly detection, where the model aims to identify data points that deviate from learned patterns. Anomaly detection is designed to detect novel, previously unseen, features unlike the seen mutated samples. Option D contradicts this principle; if future mutations had similar features, supervised classification would be more appropriate. Option E is incorrect because labeled data indicates supervised learning not unsupervised anomaly detection. Therefore, options A and C perfectly describe scenarios where an unsupervised approach makes sense for your specific classification task.

Supporting Concepts and Resources:

Unsupervised Learning: This technique learns from unlabeled data, discovering underlying patterns or structures. Anomaly detection is a form of unsupervised learning.

Google Cloud AI Platform: Unsupervised Learning

Anomaly Detection: Identifying data points that differ significantly from the majority of the dataset. Useful in situations with imbalanced datasets or novel anomalies.

Wikipedia: Anomaly Detection

Imbalanced Datasets: A class distribution where one class dominates the other. Anomaly detection can handle such situations.

Towards Data Science: Handling Imbalanced Datasets in Machine Learning

Question: 15 CertyIQ

You need to store and analyze social media postings in Google BigQuery at a rate of 10,000 messages per minute in near real-time. Initially, design the application to use streaming inserts for individual postings. Your application also performs data aggregations right after the streaming inserts. You discover that the queries after streaming inserts do not exhibit strong consistency, and reports from the queries might miss in-flight data. How can you adjust your application design?

A. Re-write the application to load accumulated data every 2 minutes.

B. Convert the streaming insert code to batch load for individual messages.

C. Load the original message to Google Cloud SQL, and export the table every hour to BigQuery via streaming inserts.

D. Estimate the average latency for data availability after streaming inserts, and always run queries after waiting twice as long.

Answer: D

Explanation:

The correct answer is **D. Estimate the average latency for data availability after streaming inserts, and always run queries after waiting twice as long.**

Here's why:

BigQuery streaming inserts offer near real-time data ingestion, but data isn't immediately available for querying due to internal processing. This "eventual consistency" means there's a latency period between insertion and query availability. Option D directly addresses this by acknowledging and accounting for this latency. By estimating the average delay and then waiting twice as long, the application increases the likelihood that all recently streamed data is available for consistent querying. This is a practical approach for handling eventual consistency in a system requiring near real-time analysis.

Option A, reloading data every 2 minutes, changes the architecture to a micro-batch approach, deviating from the initial near real-time requirement and potentially introducing complexity without addressing the core consistency issue within the micro-batches themselves. Option B, converting to batch loading for individual messages, is inefficient as it still doesn't guarantee consistency when doing analysis right after the load and defeats the purpose of streaming inserts. Option C, using Cloud SQL as an intermediate and hourly exports, introduces significant latency and overhead and is not suitable for near real-time analysis. It also shifts the initial goal from streaming to batch ingestion.

The chosen answer focuses on managing the inherent eventual consistency of BigQuery streaming inserts without fundamentally altering the streaming approach, ensuring near real-time analysis as much as possible. It represents a pragmatic solution that allows the application to remain true to the initial design while addressing the observed consistency problem.

For further reading on BigQuery streaming inserts and their consistency properties, consult the official Google Cloud documentation:

Streaming data into BigQuery: https://cloud.google.com/bigquery/docs/stream-data-into-bigquery
BigQuery Data consistency: https://cloud.google.com/bigquery/docs/data-consistency

Question: 16 CertyIQ

Your startup has never implemented a formal security policy. Currently, everyone in the company has access to the datasets stored in Google BigQuery. Teams have freedom to use the service as they see fit, and they have not documented their use cases. You have been asked to secure the data warehouse. You need to discover what everyone is doing. What should you do first?

A. Use Google Stackdriver Audit Logs to review data access.
B. Get the identity and access management IIAM) policy of each table
C. Use Stackdriver Monitoring to see the usage of BigQuery query slots.
D. Use the Google Cloud Billing API to see what account the warehouse is being billed to.

Answer: A

Explanation:

The most critical first step in securing the data warehouse, given the current situation of unrestricted access and undocumented usage, is to understand who is accessing what data and when. Option A, using Google Stackdriver Audit Logs to review data access, directly addresses this need. Audit logs capture detailed records of user actions, including data reads and modifications, providing a clear picture of current access

patterns. This insight is crucial for identifying potential security vulnerabilities and establishing a baseline for access control.

Option B, while important eventually, is premature. Obtaining the IAM policy of each table only shows who could access the data, not who is actually doing so. Similarly, option C (Stackdriver Monitoring) focuses on resource consumption (query slots) rather than user activity. Option D (Billing API) is useful for cost management, not access monitoring.

Therefore, auditing current access behavior is the logical priority before attempting to define access policies. This enables informed decision-making and minimizes disruption during the security implementation process. Using audit logs allows you to identify user roles, their usage patterns, and the specific datasets being accessed, which is fundamental for designing appropriate access control measures. Without this information, any security policy would be based on conjecture rather than evidence.

Relevant Google Cloud Documentation:

Cloud Logging (formerly Stackdriver Logging): https://cloud.google.com/logging - Provides comprehensive information on audit logs and their capabilities.
Audit logging: https://cloud.google.com/logging/docs/audit/ - Details different types of audit logs available on GCP.
BigQuery Audit Logs: https://cloud.google.com/bigquery/docs/audit-logs - Specific documentation on BigQuery's audit logging capabilities.

Question: 17 CertyIQ

Your company is migrating their 30-node Apache Hadoop cluster to the cloud. They want to re-use Hadoop jobs they have already created and minimize the management of the cluster as much as possible. They also want to be able to persist data beyond the life of the cluster. What should you do?

A. Create a Google Cloud Dataflow job to process the data.
B. Create a Google Cloud Dataproc cluster that uses persistent disks for HDFS.
C. Create a Hadoop cluster on Google Compute Engine that uses persistent disks.
D. Create a Cloud Dataproc cluster that uses the Google Cloud Storage connector.
E. Create a Hadoop cluster on Google Compute Engine that uses Local SSD disks.

Answer: D

Explanation:

The most appropriate solution is **D. Create a Cloud Dataproc cluster that uses the Google Cloud Storage connector.** Here's why:

Reusing Hadoop Jobs: Cloud Dataproc is a managed service designed to run Hadoop and Spark jobs, allowing you to leverage your existing Hadoop code with minimal modifications. This satisfies the requirement of reusing existing jobs.
Minimizing Cluster Management: Dataproc is a fully managed service, abstracting away much of the underlying infrastructure management. This means you don't have to deal with tasks like cluster setup, scaling, and patching, meeting the requirement to minimize management.
Persisting Data Beyond Cluster Lifespan: The Google Cloud Storage (GCS) connector integrates seamlessly with Dataproc. By storing data in GCS, data persists independently of the Dataproc cluster. This ensures data availability even if the cluster is stopped or scaled down.
Alternatives A, B, C, and E: Option A (Dataflow) requires rewriting Hadoop jobs, violating a key requirement. Option B (persistent disks on Dataproc) is not cost-effective and doesn't address data persistence beyond the cluster's lifespan. Option C (Hadoop on Compute Engine with persistent disks) introduces significant

management overhead. Option E (Hadoop on Compute Engine with local SSD) does not provide data persistence beyond the cluster's lifespan and is not cost-effective for data storage.
Cloud Dataproc Benefits: Dataproc enables faster cluster deployment, easier job submission, and automatic cluster scaling. It also integrates with other Google Cloud services.
Google Cloud Storage Benefits: GCS is a cost-effective, durable, and highly available object storage solution. It is the recommended storage solution for use with Dataproc when data persistence is needed.

Authoritative links:

Cloud Dataproc Documentation: https://cloud.google.com/dataproc/docs
Google Cloud Storage Documentation: https://cloud.google.com/storage/docs
Dataproc and Cloud Storage Connector:
https://cloud.google.com/dataproc/docs/concepts/connectors/cloud-storage

In conclusion, option D best aligns with the requirements for reusing existing Hadoop jobs, minimizing management, and ensuring persistent data storage beyond the cluster's lifespan through the use of managed Cloud Dataproc and the Cloud Storage connector.

Question: 18 CertyIQ

Business owners at your company have given you a database of bank transactions. Each row contains the user ID, transaction type, transaction location, and transaction amount. They ask you to investigate what type of machine learning can be applied to the data. Which three machine learning applications can you use? (Choose three.)

A. Supervised learning to determine which transactions are most likely to be fraudulent.
B. Unsupervised learning to determine which transactions are most likely to be fraudulent.
C. Clustering to divide the transactions into N categories based on feature similarity.
D. Supervised learning to predict the location of a transaction.
E. Reinforcement learning to predict the location of a transaction.
F. Unsupervised learning to predict the location of a transaction.

Answer: BCD

Explanation:

The correct machine learning applications for analyzing bank transaction data are B, C, and D. Option B, unsupervised learning, is suitable for detecting fraudulent transactions because fraud is often rare and unlabeled in datasets. Algorithms like anomaly detection can identify unusual patterns without needing predefined fraudulent examples. Option C, clustering, allows grouping similar transactions based on characteristics, potentially revealing distinct spending habits or geographic patterns that can inform business insights or fraud detection efforts. Option D, supervised learning, is applicable when the location of a transaction needs to be predicted based on other transaction features, provided labeled historical data with known locations is available.

Option A, supervised learning for fraud detection, might seem viable, but given that fraud is typically infrequent and difficult to label comprehensively, relying solely on supervised methods could be challenging and less effective initially than unsupervised approaches. Reinforcement learning (Option E) is not appropriate for this use case because it deals with learning through interaction with an environment, not directly from labeled or unlabeled historical data, and there is no environment for this learning task. Option F, using unsupervised learning to predict location is also not well-suited, since location prediction is inherently a supervised learning problem which requires labeled historical transaction data with known locations.

Unsupervised techniques reveal hidden structures in the data without labels, whereas supervised methods

learn from labeled data to make predictions. Clustering is an unsupervised technique specifically designed to group similar data points. These methods are all valuable for analyzing transaction data.

Authoritative Links for Further Research:

Google Cloud AI/ML Documentation: https://cloud.google.com/ai-platform/docs
Supervised Learning: https://developers.google.com/machine-learning/crash-course/ml-intro/supervised-learning
Unsupervised Learning: https://developers.google.com/machine-learning/crash-course/ml-intro/unsupervised-learning
Clustering: https://scikit-learn.org/stable/modules/clustering.html (Scikit-learn documentation, a popular Python library for ML)
Anomaly Detection: https://scikit-learn.org/stable/modules/outlier_detection.html (Scikit-learn documentation)

Question: 19 CertyIQ

Your company's on-premises Apache Hadoop servers are approaching end-of-life, and IT has decided to migrate the cluster to Google Cloud Dataproc. A like-for- like migration of the cluster would require 50 TB of Google Persistent Disk per node. The CIO is concerned about the cost of using that much block storage. You want to minimize the storage cost of the migration. What should you do?

A. Put the data into Google Cloud Storage.
B. Use preemptible virtual machines (VMs) for the Cloud Dataproc cluster.
C. Tune the Cloud Dataproc cluster so that there is just enough disk for all data.
D. Migrate some of the cold data into Google Cloud Storage, and keep only the hot data in Persistent Disk.

Answer: A

Explanation:

The optimal solution is to store the data in Google Cloud Storage (GCS) (Option A). Here's why: GCS is a cost-effective object storage service ideal for storing large datasets. Unlike Persistent Disk, which is block storage attached to VMs, GCS decouples storage from compute, leading to significant cost savings. Dataproc clusters can directly access data in GCS, allowing computation without needing large, expensive Persistent Disks on each node. This approach eliminates the need for 50 TB of disk space per node. Option B, using preemptible VMs, lowers compute costs but does not address the high storage costs. Option C, tuning disk space, doesn't fundamentally solve the issue of expensive block storage for large datasets. Option D, while helpful, doesn't go far enough; it is a partial implementation of option A. GCS offers a scalable and durable storage solution with various tiers to further optimize costs for different data access frequencies. In short, leveraging GCS for data storage significantly reduces infrastructure expenses compared to using large amounts of Persistent Disk.

Further Research:

Google Cloud Storage: https://cloud.google.com/storage
Google Cloud Dataproc: https://cloud.google.com/dataproc
Cost optimization with GCS: https://cloud.google.com/storage/docs/cost-optimization

Question: 20 CertyIQ

You work for a car manufacturer and have set up a data pipeline using Google Cloud Pub/Sub to capture anomalous sensor events. You are using a push subscription in Cloud Pub/Sub that calls a custom HTTPS endpoint

that you have created to take action of these anomalous events as they occur. Your custom HTTPS endpoint keeps getting an inordinate amount of duplicate messages. What is the most likely cause of these duplicate messages?

A. The message body for the sensor event is too large.
B. Your custom endpoint has an out-of-date SSL certificate.
C. The Cloud Pub/Sub topic has too many messages published to it.
D. Your custom endpoint is not acknowledging messages within the acknowledgement deadline.

Answer: D

Explanation:

The correct answer is **D: Your custom endpoint is not acknowledging messages within the acknowledgment deadline.** Here's why:

Cloud Pub/Sub's push subscriptions rely on acknowledgments (ACKs) from the subscriber endpoint to confirm successful message delivery. If a push endpoint fails to acknowledge a message within the configured acknowledgment deadline, Pub/Sub assumes the message wasn't processed and retries delivery. This retry mechanism results in duplicate messages being sent to the endpoint. Option A is incorrect as message size limits typically lead to message rejections, not duplicates. Option B is incorrect as SSL certificate issues would likely cause connection failures and no messages processed at all, not duplicates. Option C is also incorrect, as the number of messages on the topic doesn't directly cause duplicate deliveries to push subscriptions. The most common cause for the described duplicate message issue is the push endpoint failing to send an ACK within the deadline. It is critical to ensure that the endpoint can process the messages and respond with an acknowledgement within the Pub/Sub deadline to avoid duplicate messages. This behavior is part of the "at-least-once" delivery guarantee provided by Pub/Sub, which prioritizes reliability over absolute message uniqueness.

Authoritative Links:

Google Cloud Pub/Sub Push Subscriptions Documentation: https://cloud.google.com/pubsub/docs/push
Google Cloud Pub/Sub Acknowledgment and Ack Deadlines:
https://cloud.google.com/pubsub/docs/subscriber#ack_deadlines
Google Cloud Pub/Sub Delivery Guarantees:
https://cloud.google.com/pubsub/docs/overview#delivery_guarantees

Question: 21 CertyIQ

Your company uses a proprietary system to send inventory data every 6 hours to a data ingestion service in the cloud. Transmitted data includes a payload of several fields and the timestamp of the transmission. If there are any concerns about a transmission, the system re-transmits the data. How should you deduplicate the data most efficiency?

A. Assign global unique identifiers (GUID) to each data entry.
B. Compute the hash value of each data entry, and compare it with all historical data.
C. Store each data entry as the primary key in a separate database and apply an index.
D. Maintain a database table to store the hash value and other metadata for each data entry.

Answer: A

Explanation:

The most efficient way to deduplicate the inventory data is by assigning Global Unique Identifiers (GUIDs) to each entry (Option A). GUIDs, also known as UUIDs, are statistically unique 128-bit identifiers that can be

generated independently without requiring a central authority. This eliminates the need to compare new data with historical data, as each entry has a unique identifier. Upon data ingestion, checking for pre-existing GUIDs is a computationally lightweight operation. This approach is highly scalable and performs well even with large volumes of data, due to the minimal lookup complexity. Hash value comparison (Option B) is less efficient, requiring hashing each entry and comparing it to a potential large set of historical hashes, incurring significant computational cost and storage. Storing entire data entries as primary keys in a database with indexing (Option C) is computationally expensive and storage intensive, further complicated by the need for large tables and complicated indexing. Maintaining a database table storing hashes and metadata (Option D) involves the same computationally intensive comparison process like option B and also involves more data storage, thus it will be more complex and less efficient. GUIDs eliminate the need to search across historical datasets by assigning a unique identifier upfront. This translates into lower processing times and lower infrastructure costs for data deduplication. The process becomes simple: check if a GUID exists, if not, process the data, if yes, discard the data. Therefore, using GUIDs aligns with best practices in handling duplicate data in data engineering and efficient data ingestion, making Option A the most effective solution.

Authoritative Links for Further Research:

UUIDs/GUIDs: https://en.wikipedia.org/wiki/Universally_unique_identifier
Data Deduplication: https://cloud.google.com/architecture/data-deduplication-options
Scalable Data Processing: https://cloud.google.com/solutions/scalable-data-processing

Question: 22 CertyIQ

Your company has hired a new data scientist who wants to perform complicated analyses across very large datasets stored in Google Cloud Storage and in a
Cassandra cluster on Google Compute Engine. The scientist primarily wants to create labelled data sets for machine learning projects, along with some visualization tasks. She reports that her laptop is not powerful enough to perform her tasks and it is slowing her down. You want to help her perform her tasks.
What should you do?

A. Run a local version of Jupiter on the laptop.
B. Grant the user access to Google Cloud Shell.
C. Host a visualization tool on a VM on Google Compute Engine.
D. Deploy Google Cloud Datalab to a virtual machine (VM) on Google Compute Engine.

Answer: D

Explanation:

Here's a detailed justification for choosing option D, deploying Google Cloud Datalab, as the best solution:

The data scientist needs a powerful environment to analyze large datasets from both Google Cloud Storage and a Cassandra cluster, create labeled datasets for machine learning, and perform visualizations, all of which exceeds the capacity of their laptop. Option A, running a local Jupyter notebook, simply replicates the current underpowered setup. Option B, Google Cloud Shell, provides a command-line interface but lacks the necessary interactive notebook environment for data exploration and visualization. Option C, hosting a visualization tool on a Compute Engine VM, addresses a single aspect (visualization) but doesn't address the entire workflow involving data access, processing, and labeling.

Google Cloud Datalab (now superseded by Vertex AI Workbench, but Datalab concepts still apply) provides a fully managed and scalable Jupyter notebook environment hosted on a Google Compute Engine VM. This addresses the scientist's needs by offering: a powerful compute resource for processing large datasets, pre-installed data science libraries, direct access to Google Cloud Storage and other Google Cloud resources, support for various data connectors including Cassandra, and visualization capabilities within the notebook environment. It facilitates collaborative work and enables version control using Git. The data scientist can

easily perform interactive data exploration, analysis, and machine learning tasks without being constrained by local laptop limitations.

The solution offers scalability, allowing the data scientist to process data efficiently and meet their machine learning needs. It also handles the infrastructure management behind the scenes, allowing the scientist to focus on data analysis.

Relevant Links for further research:

Google Cloud Datalab (archived documentation): This provides an overview of the service and its functionalities. Note that Datalab has been replaced by Vertex AI Workbench.
Vertex AI Workbench: This is the successor to Datalab, offering similar capabilities with enhanced features. Understanding Workbench helps understand Datalab principles.
Jupyter Notebooks: This gives background on the interactive notebook environment used by Datalab and Vertex AI Workbench.

Question: 23 CertyIQ

You are deploying 10,000 new Internet of Things devices to collect temperature data in your warehouses globally. You need to process, store and analyze these very large datasets in real time. What should you do?

A. Send the data to Google Cloud Datastore and then export to BigQuery.

B. Send the data to Google Cloud Pub/Sub, stream Cloud Pub/Sub to Google Cloud Dataflow, and store the data in Google BigQuery.

C. Send the data to Cloud Storage and then spin up an Apache Hadoop cluster as needed in Google Cloud Dataproc whenever analysis is required.

D. Export logs in batch to Google Cloud Storage and then spin up a Google Cloud SQL instance, import the data from Cloud Storage, and run an analysis as needed.

Answer: B

Explanation:

Option B is the most appropriate solution for real-time processing, storage, and analysis of data from 10,000 IoT devices. Google Cloud Pub/Sub is a messaging service that allows for asynchronous data ingestion, handling a high volume of messages from various sources like IoT devices. This aligns with the need to manage streaming temperature data. Cloud Dataflow is a fully managed, serverless stream and batch data processing service, ideal for real-time analytics on the streaming Pub/Sub data. It can perform transformations and computations on the incoming data stream. Finally, BigQuery, a fully-managed data warehouse, provides scalable storage and enables rapid analysis of large datasets. The streamed and processed data from Dataflow can be stored directly in BigQuery for analysis, such as trend identification and anomaly detection. Option A suggests using Cloud Datastore, which is not optimized for the analytical workloads necessary for this scenario, and exporting to BigQuery introduces an unnecessary delay. Option C involves batch processing with Cloud Storage and Dataproc, which contradicts the real-time analysis requirement. Option D is entirely batch-oriented, using Cloud Storage, Cloud SQL, and import/analysis, thus, is unsuitable for real-time analysis.

Here are some authoritative links for further research:

Google Cloud Pub/Sub: https://cloud.google.com/pubsub/docs
Google Cloud Dataflow: https://cloud.google.com/dataflow/docs
Google BigQuery: https://cloud.google.com/bigquery/docs

Question: 24 CertyIQ

You have spent a few days loading data from comma-separated values (CSV) files into the Google BigQuery table CLICK_STREAM. The column DT stores the epoch time of click events. For convenience, you chose a simple schema where every field is treated as the STRING type. Now, you want to compute web session durations of users who visit your site, and you want to change its data type to the TIMESTAMP. You want to minimize the migration effort without making future queries computationally expensive. What should you do?

A. Delete the table CLICK_STREAM, and then re-create it such that the column DT is of the TIMESTAMP type. Reload the data.

B. Add a column TS of the TIMESTAMP type to the table CLICK_STREAM, and populate the numeric values from the column TS for each row. Reference the column TS instead of the column DT from now on.

C. Create a view CLICK_STREAM_V, where strings from the column DT are cast into TIMESTAMP values. Reference the view CLICK_STREAM_V instead of the table CLICK_STREAM from now on.

D. Add two columns to the table CLICK STREAM: TS of the TIMESTAMP type and IS_NEW of the BOOLEAN type. Reload all data in append mode. For each appended row, set the value of IS_NEW to true. For future queries, reference the column TS instead of the column DT, with the WHERE clause ensuring that the value of IS_NEW must be true.

E. Construct a query to return every row of the table CLICK_STREAM, while using the built-in function to cast strings from the column DT into TIMESTAMP values. Run the query into a destination table NEW_CLICK_STREAM, in which the column TS is the TIMESTAMP type. Reference the table NEW_CLICK_STREAM instead of the table CLICK_STREAM from now on. In the future, new data is loaded into the table NEW_CLICK_STREAM.

Answer: E

Explanation:

The most efficient approach to convert the DT column from STRING to TIMESTAMP in BigQuery while minimizing migration effort and computational overhead is option E. Creating a new table, NEW_CLICK_STREAM, with the correct TIMESTAMP data type using a query that casts the DT strings avoids altering the original table and preserves the original data for potential historical analysis. This avoids the complexity and downtime of deleting and recreating the entire table (A) or loading data into additional columns on the existing table(B, D). While creating a view (C) is a valid way to represent data with the correct data type, it performs the casting operation every time it's queried, adding computational overhead. Using a destination table with the correct data type is more computationally performant long term. Moreover, this approach aligns well with the practice of creating data pipelines. By making the new table the point of reference going forward, this avoids complex conditional clauses on older data, and keeps the processing simple. Moving forward new data will be loaded directly into the new schema. Option E strikes a balance between avoiding costly reloads and ensuring performance without unnecessary complexity or introducing performance bottlenecks. Furthermore, new data is loaded into the new table eliminating the need for complicated conditions to differentiate between existing data and new data.

Authoritative links:

BigQuery documentation on data type conversion:
https://cloud.google.com/bigquery/docs/reference/standard-sql/conversion_functions
BigQuery documentation on creating tables from queries: https://cloud.google.com/bigquery/docs/writing-results#writing_query_results_to_a_new_table

Question: 25 CertyIQ

You want to use Google Stackdriver Logging to monitor Google BigQuery usage. You need an instant notification to be sent to your monitoring tool when new data is appended to a certain table using an insert job, but you do not want to receive notifications for other tables. What should you do?

A. Make a call to the Stackdriver API to list all logs, and apply an advanced filter.

B. In the Stackdriver logging admin interface, and enable a log sink export to BigQuery.

C. In the Stackdriver logging admin interface, enable a log sink export to Google Cloud Pub/Sub, and subscribe to the topic from your monitoring tool.

D. Using the Stackdriver API, create a project sink with advanced log filter to export to Pub/Sub, and subscribe to the topic from your monitoring tool.

Answer: D

Explanation:

The correct answer is D because it leverages the appropriate Google Cloud tools for targeted log filtering and real-time notifications. Stackdriver (now Cloud Logging) provides powerful capabilities for managing and analyzing logs. To achieve the specific requirement of receiving notifications only for insert jobs into a particular BigQuery table, a project sink with an advanced log filter is needed.

Here's why the other options are unsuitable:

A: Listing all logs and then applying a filter on the client side is inefficient and not real-time. This approach is not suitable for instant notifications.
B: Exporting all logs to BigQuery would incur unnecessary storage costs and introduce complexity when filtering for the specific table and insert jobs, which does not satisfy the requirement of instant notification to monitoring tools.
C: While exporting to Pub/Sub is a good first step, filtering and scoping log messages is needed via log sink with advanced filtering.

Option D, by using the Cloud Logging API to create a project sink:

1. **Targeted Filtering:** It allows defining a specific filter (an advanced filter using the Cloud Logging query language) to isolate log entries related to BigQuery insert jobs on a designated table. This ensures no other table updates cause notifications, satisfying the question's requirement.
2. **Real-time Export:** The project sink can export filtered logs to Cloud Pub/Sub, a messaging service, in real-time.
3. **Real-time Notification:** The monitoring tool can subscribe to the Pub/Sub topic, allowing it to receive instant notifications whenever a matching log entry is generated.

This approach combines the filtering capability of Cloud Logging sinks with the real-time notification ability of Pub/Sub, providing an efficient and targeted solution.

Relevant Links for Further Research:

Cloud Logging Overview: https://cloud.google.com/logging/docs/overview
Cloud Logging Sinks: https://cloud.google.com/logging/docs/export/configure_export
Cloud Logging Advanced Filters: https://cloud.google.com/logging/docs/view/advanced-queries
Cloud Pub/Sub: https://cloud.google.com/pubsub/docs

Question: 26 CertyIQ

You are working on a sensitive project involving private user data. You have set up a project on Google Cloud Platform to house your work internally. An external consultant is going to assist with coding a complex transformation in a Google Cloud Dataflow pipeline for your project. How should you maintain users' privacy?

A. Grant the consultant the Viewer role on the project.

B. Grant the consultant the Cloud Dataflow Developer role on the project.

C. Create a service account and allow the consultant to log on with it.

D. Create an anonymized sample of the data for the consultant to work with in a different project.

Answer: D

Explanation:

The most secure approach to protect user privacy when working with an external consultant is option D, creating an anonymized sample dataset. Granting the consultant direct access to the production project (options A, B, C) presents significant privacy risks, even with restricted roles. Option A, Viewer role, while read-only, still exposes the consultant to potentially sensitive metadata and schema details. Option B, Cloud Dataflow Developer, gives write access to Dataflow pipelines and indirectly to data, which is unacceptable for sensitive data. Option C, a shared service account, blurs the audit trail and doesn't prevent the consultant from accessing production data. Anonymized data in a separate project (D) completely removes the privacy risk. The consultant can still develop and test the Dataflow transformation, as it is a scaled-down and de-identified version of the original data. This approach ensures compliance with data privacy regulations, prevents unintended data exposure, and allows the consultant to work effectively without compromising user privacy. This methodology adheres to the principle of least privilege, ensuring data exposure is minimized. https://cloud.google.com/data-loss-prevention/docs/concepts-deid-strategies and https://cloud.google.com/docs/security/best-practices/data-privacy-best-practices provide further guidance on data anonymization and privacy practices on Google Cloud.

Question: 27 CertyIQ

You are building a model to predict whether or not it will rain on a given day. You have thousands of input features and want to see if you can improve training speed by removing some features while having a minimum effect on model accuracy. What can you do?

A. Eliminate features that are highly correlated to the output labels.
B. Combine highly co-dependent features into one representative feature.
C. Instead of feeding in each feature individually, average their values in batches of 3.
D. Remove the features that have null values for more than 50% of the training records.

Answer: B

Explanation:

The most effective strategy to reduce training time while minimizing accuracy loss when dealing with numerous input features, some of which are highly correlated, is to combine those highly co-dependent features into a single, representative feature (option B). This process, often referred to as feature engineering or feature aggregation, reduces the dimensionality of the input data. High dimensionality can increase training complexity and time due to the curse of dimensionality, where models struggle to generalize from sparsely populated high-dimensional spaces. Option A, eliminating features correlated with the output, is counterproductive as those features are likely crucial for accurate prediction. Averaging features in batches (option C) doesn't address the core issue of feature redundancy. Removing features based on missing values exceeding 50% (option D) can eliminate potentially valuable information if those missing values don't consistently bias the model. Option B is the optimal choice as it addresses redundancy while retaining information, thus leading to faster training without a significant loss in accuracy. Feature aggregation techniques include Principal Component Analysis (PCA), which creates new, uncorrelated features by combining existing ones, and simple averaging or weighted averaging of related features. These can substantially speed up training by reducing the number of parameters the model needs to learn, while keeping key information that allows high model performance.

Supporting Links:

Feature Selection: https://scikit-learn.org/stable/modules/feature_selection.html (Though not GCP specific, it covers key concepts applicable to any machine learning project)
Curse of Dimensionality: https://en.wikipedia.org/wiki/Curse_of_dimensionality
Feature Engineering: https://developers.google.com/machine-learning/guides/feature-engineering (Google's own ML guide)
Principal Component Analysis (PCA): https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-50b11811d191 (Provides in-depth explanation of PCA, a common feature aggregation method.)

Question: 28 CertyIQ

Your company is performing data preprocessing for a learning algorithm in Google Cloud Dataflow. Numerous data logs are being are being generated during this step, and the team wants to analyze them. Due to the dynamic nature of the campaign, the data is growing exponentially every hour.
The data scientists have written the following code to read the data for a new key features in the logs.

```
BigQueryIO.Read
     .named("ReadLogData")
     .from("clouddataflow-readonly:samples.log_data")
```

You want to improve the performance of this data read. What should you do?

A. Specify the TableReference object in the code.
B. Use .fromQuery operation to read specific fields from the table.
C. Use of both the Google BigQuery TableSchema and TableFieldSchema classes.
D. Call a transform that returns TableRow objects, where each element in the PCollection represents a single row in the table.

Answer: B

Explanation:

Use .from Query operation to read specific fields from the table. Big Query IO. read. from() directly reads the whole table from Big Query. This function exports the whole table to temporary files in Google Cloud Storage, where it will later be read from. This requires almost no computation, as it only performs an export job, and later Dataflow reads from GCS (not from Big Query).

Big Query IO .read .from Query() executes a query and then reads the results received after the query execution. Therefore, this function is more time-consuming, given that it requires that a query is first executed (which will incur in the corresponding economic and computational costs).

Question: 29 CertyIQ

Your company is streaming real-time sensor data from their factory floor into Bigtable and they have noticed extremely poor performance. How should the row key be redesigned to improve Bigtable performance on queries that populate real-time dashboards?

A. Use a row key of the form <timestamp>.
B. Use a row key of the form <sensorid>.
C. Use a row key of the form <timestamp>#<sensorid>.

D. Use a row key of the form >#<sensorid>#<timestamp>.

Answer: D

Explanation:

Here's a detailed justification for why option D is the most suitable row key design for improving Bigtable performance in this scenario:

Bigtable, a NoSQL wide-column store, is optimized for reads based on the lexicographical order of row keys. Using a timestamp as the sole row key (option A) leads to a "hotspotting" problem. Since timestamps are constantly increasing, all new writes would target a single server, creating a bottleneck and causing poor performance. Similarly, using only the sensor ID (option B) would concentrate writes for the same sensor on the same server. Option C, #, also doesn't fully mitigate hotspots since the timestamp is still the primary sorting key.

Option D, <sensorid>#<timestamp>, offers the best approach by distributing writes across multiple servers based on sensor IDs, preventing hotspotting. The sensor ID provides initial distribution and timestamp ensures a logical order and efficient retrieval of time series data. When retrieving sensor data for dashboards, the queries can leverage a prefix scan on sensor IDs, which is highly efficient. Using a timestamp within the row key also supports efficient querying for a specific time range for a given sensor. This design balances write distribution with query efficiency, therefore improving overall Bigtable performance. By properly designing the row key to consider the access patterns, you can significantly reduce latency and improve resource utilization.

Relevant Links:

Google Cloud Bigtable Schema Design: https://cloud.google.com/bigtable/docs/schema-design
Bigtable Key Design Best Practices: https://cloud.google.com/blog/products/data-analytics/cloud-bigtable-key-visualizer-lets-you-optimize-performance
Bigtable Performance Considerations: https://cloud.google.com/bigtable/docs/performance

Question: 30 CertyIQ

Your company's customer and order databases are often under heavy load. This makes performing analytics against them difficult without harming operations.
The databases are in a MySQL cluster, with nightly backups taken using mysqldump. You want to perform analytics with minimal impact on operations. What should you do?

A. Add a node to the MySQL cluster and build an OLAP cube there.
B. Use an ETL tool to load the data from MySQL into Google BigQuery.
C. Connect an on-premises Apache Hadoop cluster to MySQL and perform ETL.
D. Mount the backups to Google Cloud SQL, and then process the data using Google Cloud Dataproc.

Answer: B

Explanation:

The optimal solution is to use an ETL tool to load data from MySQL into Google BigQuery (Option B). This approach addresses the core issue: performing analytics without impacting the operational MySQL database. BigQuery is a fully managed, serverless data warehouse designed for large-scale analytics. By moving data to BigQuery, you isolate analytical workloads from the transactional MySQL cluster, preventing performance degradation during periods of heavy load. ETL tools like Dataflow or Cloud Data Fusion can efficiently extract data from MySQL, transform it as needed, and load it into BigQuery. This process ensures that analytical queries run against a data warehouse optimized for such operations, rather than the operational database.

Option A, adding a node to MySQL and building an OLAP cube there, would likely exacerbate the load issues as it still places analytical processing on the same infrastructure. Option C introduces unnecessary complexity by involving on-premises resources and a Hadoop cluster. Option D, mounting backups to Cloud SQL, adds latency with restoration and doesn't provide the analytical performance of BigQuery. Hence, B provides the best balance of scalability, performance, and operational safety, aligning with best practices for separating OLTP and OLAP workloads on cloud platforms.

Further Research:

BigQuery Overview: https://cloud.google.com/bigquery/docs/introduction
ETL on Google Cloud: https://cloud.google.com/solutions/etl
OLTP vs OLAP: https://www.ibm.com/topics/olap-vs-oltp

Question: 31 CertyIQ

You have Google Cloud Dataflow streaming pipeline running with a Google Cloud Pub/Sub subscription as the source. You need to make an update to the code that will make the new Cloud Dataflow pipeline incompatible with the current version. You do not want to lose any data when making this update. What should you do?

A. Update the current pipeline and use the drain flag.
B. Update the current pipeline and provide the transform mapping JSON object.
C. Create a new pipeline that has the same Cloud Pub/Sub subscription and cancel the old pipeline.
D. Create a new pipeline that has a new Cloud Pub/Sub subscription and cancel the old pipeline.

Answer: A

Explanation:

The correct answer is A: Update the current pipeline and use the drain flag. Here's why:

When updating a streaming Dataflow pipeline that introduces incompatibility, a simple update without proper handling can lead to data loss. The drain flag is crucial in this scenario. When a pipeline is drained, Dataflow stops accepting new data, but it continues processing all data already in-flight through the existing pipeline. This ensures that no messages from the Pub/Sub subscription are lost or discarded during the update process. It allows the current version to finish processing all the messages it has already consumed. After the drain operation completes, the pipeline can be safely updated with the new, incompatible code. Once the new version is deployed, it can start consuming from the existing Pub/Sub subscription without any data loss. Options C and D involve creating new pipelines and potentially cancelling the old one, which might cause some messages that are still in transit to be lost. Option B is incorrect because transform mapping is used during pipeline updates when there are schema changes between versions but is not applicable here for incompatible changes requiring draining. The drain mechanism is designed for graceful shutdowns, guaranteeing data safety. This is a standard practice in streaming data processing to avoid data loss during upgrades.https://cloud.google.com/dataflow/docs/guides/updating-a-pipelinehttps://cloud.google.com/dataflow/docs/concepts/pipeline-lifecycle

Question: 32 CertyIQ

Your company is running their first dynamic campaign, serving different offers by analyzing real-time data during the holiday season. The data scientists are collecting terabytes of data that rapidly grows every hour during their 30-day campaign. They are using Google Cloud Dataflow to preprocess the data and collect the feature (signals) data that is needed for the machine learning model in Google Cloud Bigtable. The team is observing suboptimal performance with reads and writes of their initial load of 10 TB of data. They want to improve this performance while minimizing cost. What should they do?

A. Redefine the schema by evenly distributing reads and writes across the row space of the table.

B. The performance issue should be resolved over time as the site of the BigDate cluster is increased.

C. Redesign the schema to use a single row key to identify values that need to be updated frequently in the cluster.

D. Redesign the schema to use row keys based on numeric IDs that increase sequentially per user viewing the offers.

Answer: A

Explanation:

The correct answer is **A. Redefine the schema by evenly distributing reads and writes across the row space of the table.**

Here's why:

Bigtable performance heavily relies on efficient row key design. When reads and writes are concentrated in a small range of row keys (a "hotspot"), the load isn't distributed across the cluster's tablets, leading to performance bottlenecks. This scenario often arises from sequentially increasing IDs or timestamp-based row keys, which results in all new data landing on the same tablet until it splits. Option A directly addresses this issue by advocating for a schema that distributes reads and writes evenly. This can be achieved through strategies like:

Hashing: Applying a hash function to user IDs or other identifying data to spread writes across the row space.
Salting: Adding a prefix to row keys that distributes them over different tablet servers.

Option B is incorrect because merely increasing cluster size won't fix a poorly designed schema; it simply buys time until the bigger cluster also bottlenecks in the same way. Option C promotes using a single row key for frequent updates, which is the opposite of good Bigtable schema design; it will only exacerbate the hot-spotting issue. Option D is also incorrect because sequentially increasing user IDs will cause the same hot-spotting problem. A good row key design needs to consider both reads and writes, and distribute the data evenly across the entire cluster.

In summary, a well-designed Bigtable schema requires distributing data across the row space. Hashing or salting row keys is crucial for achieving optimal read and write performance.

Authoritative Links:

Google Cloud Bigtable Schema Design: https://cloud.google.com/bigtable/docs/schema-design
Bigtable Performance Considerations: https://cloud.google.com/bigtable/docs/performance
Key Visualizer in Bigtable: https://cloud.google.com/bigtable/docs/key-visualizer

Question: 33 CertyIQ

Your software uses a simple JSON format for all messages. These messages are published to Google Cloud Pub/Sub, then processed with Google Cloud
Dataflow to create a real-time dashboard for the CFO. During testing, you notice that some messages are missing in the dashboard. You check the logs, and all messages are being published to Cloud Pub/Sub successfully. What should you do next?

A. Check the dashboard application to see if it is not displaying correctly.

B. Run a fixed dataset through the Cloud Dataflow pipeline and analyze the output.

C. Use Google Stackdriver Monitoring on Cloud Pub/Sub to find the missing messages.

D. Switch Cloud Dataflow to pull messages from Cloud Pub/Sub instead of Cloud Pub/Sub pushing messages to Cloud Dataflow.

Answer: B

Explanation:

The correct answer is **B. Run a fixed dataset through the Cloud Dataflow pipeline and analyze the output.**

Here's the justification:

The problem states messages are successfully published to Pub/Sub, but some are missing in the final dashboard. This indicates a problem after Pub/Sub. Option A, while potentially a cause, doesn't directly address a pipeline issue. Option C is not ideal because it doesn't provide an immediate root cause; it's more for monitoring system health rather than pinpointing data transformation issues within Dataflow. Option D isn't a solution but a different way to integrate, which doesn't directly solve the issue at hand if messages are getting lost.

Running a fixed dataset (a known set of messages) through Dataflow allows you to isolate the problem. You can analyze the output at each stage of the pipeline to see exactly where and why messages are being dropped or transformed incorrectly. This approach facilitates debugging and provides actionable insights into Dataflow pipeline logic. It allows you to verify if the Dataflow transformations are properly parsing the JSON messages, and handling any edge cases. If a known message disappears, you've narrowed down your search to the Dataflow code. This method is far more efficient for diagnosing data processing errors than methods based on monitoring raw messages, as the issue appears to lie in the data pipeline logic. This approach leverages Dataflow's ability to ingest and process the known dataset, allowing a step-by-step analysis for data lineage and pinpointing transformation errors.

Further research:

Google Cloud Dataflow Debugging: https://cloud.google.com/dataflow/docs/guides/debugging-pipelines This official Google documentation provides guidance on debugging Dataflow pipelines, including testing with fixed datasets.
Dataflow Basics: https://cloud.google.com/dataflow/docs/concepts/dataflow-model Provides a strong base on how Dataflow works.
Pub/Sub Monitoring: https://cloud.google.com/pubsub/docs/monitoring Details how to monitor Pub/Sub topics, while useful it is less relevant to the core issue here.

Question: 34 CertyIQ

Flowlogistic Case Study -

Company Overview -
Flowlogistic is a leading logistics and supply chain provider. They help businesses throughout the world manage their resources and transport them to their final destination. The company has grown rapidly, expanding their offerings to include rail, truck, aircraft, and oceanic shipping.

Company Background -
The company started as a regional trucking company, and then expanded into other logistics market. Because they have not updated their infrastructure, managing and tracking orders and shipments has become a bottleneck. To improve operations, Flowlogistic developed proprietary technology for tracking shipments in real time at the parcel level. However, they are unable to deploy it because their technology stack, based on Apache Kafka, cannot support the processing volume. In addition, Flowlogistic wants to further analyze their orders and shipments to determine how best to deploy their resources.

Solution Concept -
Flowlogistic wants to implement two concepts using the cloud:
✑ Use their proprietary technology in a real-time inventory-tracking system that indicates the location of their loads
✑ Perform analytics on all their orders and shipment logs, which contain both structured and unstructured data, to determine how best to deploy resources, which markets to expand info. They also want to use predictive analytics

to learn earlier when a shipment will be delayed.

Existing Technical Environment -
Flowlogistic architecture resides in a single data center:
✑ Databases
8 physical servers in 2 clusters
- SQL Server `" user data, inventory, static data
3 physical servers
- Cassandra `" metadata, tracking messages
10 Kafka servers `" tracking message aggregation and batch insert
✑ Application servers `" customer front end, middleware for order/customs
60 virtual machines across 20 physical servers
- Tomcat `" Java services
- Nginx `" static content
- Batch servers
✑ Storage appliances
- iSCSI for virtual machine (VM) hosts
- Fibre Channel storage area network (FC SAN) `" SQL server storage
- Network-attached storage (NAS) image storage, logs, backups
✑ 10 Apache Hadoop /Spark servers
- Core Data Lake
- Data analysis workloads
✑ 20 miscellaneous servers
- Jenkins, monitoring, bastion hosts,

Business Requirements -
Build a reliable and reproducible environment with scaled panty of production.
✑ Aggregate data in a centralized Data Lake for analysis
✑ Use historical data to perform predictive analytics on future shipments
✑ Accurately track every shipment worldwide using proprietary technology
✑ Improve business agility and speed of innovation through rapid provisioning of new resources
✑ Analyze and optimize architecture for performance in the cloud
✑ Migrate fully to the cloud if all other requirements are met

Technical Requirements -
✑ Handle both streaming and batch data
✑ Migrate existing Hadoop workloads
✑ Ensure architecture is scalable and elastic to meet the changing demands of the company.
✑ Use managed services whenever possible
✑ Encrypt data flight and at rest
✑ Connect a VPN between the production data center and cloud environment

SEO Statement -
We have grown so quickly that our inability to upgrade our infrastructure is really hampering further growth and efficiency. We are efficient at moving shipments around the world, but we are inefficient at moving data around. We need to organize our information so we can more easily understand where our customers are and what they are shipping.

CTO Statement -
IT has never been a priority for us, so as our data has grown, we have not invested enough in our technology. I have a good staff to manage IT, but they are so busy managing our infrastructure that I cannot get them to do the things that really matter, such as organizing our data, building the analytics, and figuring out how to implement the CFO' s tracking technology.

CFO Statement -
Part of our competitive advantage is that we penalize ourselves for late shipments and deliveries. Knowing where out shipments are at all times has a direct correlation to our bottom line and profitability. Additionally, I don't want to commit capital to building out a server environment.
Flowlogistic wants to use Google BigQuery as their primary analysis system, but they still have Apache Hadoop and Spark workloads that they cannot move to
BigQuery. Flowlogistic does not know how to store the data that is common to both workloads. What should they do?

A. Store the common data in BigQuery as partitioned tables.

B. Store the common data in BigQuery and expose authorized views.

C. Store the common data encoded as Avro in Google Cloud Storage.

D. Store he common data in the HDFS storage for a Google Cloud Dataproc cluster.

Answer: C

Explanation:

avro data can be accessed by spark as well

Question: 35 CertyIQ

Flowlogistic Case Study -

Company Overview -
Flowlogistic is a leading logistics and supply chain provider. They help businesses throughout the world manage their resources and transport them to their final destination. The company has grown rapidly, expanding their offerings to include rail, truck, aircraft, and oceanic shipping.

Company Background -
The company started as a regional trucking company, and then expanded into other logistics market. Because they have not updated their infrastructure, managing and tracking orders and shipments has become a bottleneck. To improve operations, Flowlogistic developed proprietary technology for tracking shipments in real time at the parcel level. However, they are unable to deploy it because their technology stack, based on Apache Kafka, cannot support the processing volume. In addition, Flowlogistic wants to further analyze their orders and shipments to determine how best to deploy their resources.

Solution Concept -
Flowlogistic wants to implement two concepts using the cloud:
✑ Use their proprietary technology in a real-time inventory-tracking system that indicates the location of their loads
✑ Perform analytics on all their orders and shipment logs, which contain both structured and unstructured data, to determine how best to deploy resources, which markets to expand info. They also want to use predictive analytics to learn earlier when a shipment will be delayed.

Existing Technical Environment -
Flowlogistic architecture resides in a single data center:
✑ Databases
8 physical servers in 2 clusters
- SQL Server `" user data, inventory, static data
3 physical servers
- Cassandra `" metadata, tracking messages
10 Kafka servers `" tracking message aggregation and batch insert
✑ Application servers `" customer front end, middleware for order/customs
60 virtual machines across 20 physical servers
- Tomcat `" Java services
- Nginx `" static content
- Batch servers
✑ Storage appliances
- iSCSI for virtual machine (VM) hosts
- Fibre Channel storage area network (FC SAN) `" SQL server storage
- Network-attached storage (NAS) image storage, logs, backups
✑ 10 Apache Hadoop /Spark servers
- Core Data Lake
- Data analysis workloads
✑ 20 miscellaneous servers
- Jenkins, monitoring, bastion hosts,

Business Requirements -
✑ Build a reliable and reproducible environment with scaled panty of production.
✑ Aggregate data in a centralized Data Lake for analysis
✑ Use historical data to perform predictive analytics on future shipments
✑ Accurately track every shipment worldwide using proprietary technology

✑ Improve business agility and speed of innovation through rapid provisioning of new resources
✑ Analyze and optimize architecture for performance in the cloud
✑ Migrate fully to the cloud if all other requirements are met

Technical Requirements -
✑ Handle both streaming and batch data
✑ Migrate existing Hadoop workloads
✑ Ensure architecture is scalable and elastic to meet the changing demands of the company.
✑ Use managed services whenever possible
✑ Encrypt data flight and at rest
✑ Connect a VPN between the production data center and cloud environment

SEO Statement -
We have grown so quickly that our inability to upgrade our infrastructure is really hampering further growth and efficiency. We are efficient at moving shipments around the world, but we are inefficient at moving data around. We need to organize our information so we can more easily understand where our customers are and what they are shipping.

CTO Statement -
IT has never been a priority for us, so as our data has grown, we have not invested enough in our technology. I have a good staff to manage IT, but they are so busy managing our infrastructure that I cannot get them to do the things that really matter, such as organizing our data, building the analytics, and figuring out how to implement the CFO' s tracking technology.

CFO Statement -
Part of our competitive advantage is that we penalize ourselves for late shipments and deliveries. Knowing where out shipments are at all times has a direct correlation to our bottom line and profitability. Additionally, I don't want to commit capital to building out a server environment.
Flowlogistic's management has determined that the current Apache Kafka servers cannot handle the data volume for their real-time inventory tracking system.
You need to build a new system on Google Cloud Platform (GCP) that will feed the proprietary tracking software. The system must be able to ingest data from a variety of global sources, process and query in real-time, and store the data reliably. Which combination of GCP products should you choose?

A. Cloud Pub/Sub, Cloud Dataflow, and Cloud Storage
B. Cloud Pub/Sub, Cloud Dataflow, and Local SSD
C. Cloud Pub/Sub, Cloud SQL, and Cloud Storage
D. Cloud Load Balancing, Cloud Dataflow, and Cloud Storage

Answer: A

Explanation:

The correct answer is **A. Cloud Pub/Sub, Cloud Dataflow, and Cloud Storage**. Here's why:

Flowlogistic needs a system that can handle real-time ingestion, processing, and storage of tracking data from various global sources. Let's break down how each chosen service addresses specific needs:

Cloud Pub/Sub: This is a fully managed, scalable messaging service that enables real-time data ingestion from global sources. It decouples the data producers (tracking devices) from the data consumers (processing pipelines). Pub/Sub is designed to handle high-throughput streaming data, which is perfect for the real-time tracking use case, replacing their inadequate Kafka setup. https://cloud.google.com/pubsub/docs/overview

Cloud Dataflow: This is a serverless data processing service capable of both stream and batch processing. It's ideal for real-time processing of the tracking data coming from Pub/Sub. Dataflow can perform transformations, aggregations, and enrichments on the data stream before storing it. https://cloud.google.com/dataflow/docs/concepts/what-is-dataflow

Cloud Storage: This is a highly scalable, durable, and cost-effective object storage service. It serves as a reliable repository for storing the processed tracking data after Dataflow has completed its work. This will provide a centralized data lake for analytics and predictive modeling, addressing another business

requirement mentioned in the case study. https://cloud.google.com/storage/docs/overview

Option B uses Local SSD which is not suitable for data lake storage. It is ephemeral and associated with a specific virtual machine. Option C proposes Cloud SQL, which is a relational database, less suitable for a time series or message type data. Cloud Load Balancing in option D is good for distributing web traffic, not for data ingestion.Therefore, **Cloud Pub/Sub, Cloud Dataflow, and Cloud Storage** is the optimal combination for meeting Flowlogistic's real-time tracking system requirements.

Question: 36 CertyIQ

Flowlogistic Case Study -

Company Overview -
Flowlogistic is a leading logistics and supply chain provider. They help businesses throughout the world manage their resources and transport them to their final destination. The company has grown rapidly, expanding their offerings to include rail, truck, aircraft, and oceanic shipping.

Company Background -
The company started as a regional trucking company, and then expanded into other logistics market. Because they have not updated their infrastructure, managing and tracking orders and shipments has become a bottleneck. To improve operations, Flowlogistic developed proprietary technology for tracking shipments in real time at the parcel level. However, they are unable to deploy it because their technology stack, based on Apache Kafka, cannot support the processing volume. In addition, Flowlogistic wants to further analyze their orders and shipments to determine how best to deploy their resources.

Solution Concept -
Flowlogistic wants to implement two concepts using the cloud:
Use their proprietary technology in a real-time inventory-tracking system that indicates the location of their loads
▪
✑ Perform analytics on all their orders and shipment logs, which contain both structured and unstructured data, to determine how best to deploy resources, which markets to expand info. They also want to use predictive analytics to learn earlier when a shipment will be delayed.

Existing Technical Environment -
Flowlogistic architecture resides in a single data center:
✑ Databases
8 physical servers in 2 clusters
- SQL Server `" user data, inventory, static data
3 physical servers
- Cassandra `" metadata, tracking messages
10 Kafka servers `" tracking message aggregation and batch insert
✑ Application servers `" customer front end, middleware for order/customs
60 virtual machines across 20 physical servers
- Tomcat `" Java services
- Nginx `" static content
- Batch servers
✑ Storage appliances
- iSCSI for virtual machine (VM) hosts
- Fibre Channel storage area network (FC SAN) `" SQL server storage
- Network-attached storage (NAS) image storage, logs, backups
✑ 10 Apache Hadoop /Spark servers
- Core Data Lake
- Data analysis workloads
✑ 20 miscellaneous servers
- Jenkins, monitoring, bastion hosts,

Business Requirements -
✑ Build a reliable and reproducible environment with scaled panty of production.
✑ Aggregate data in a centralized Data Lake for analysis
✑ Use historical data to perform predictive analytics on future shipments
✑ Accurately track every shipment worldwide using proprietary technology
✑ Improve business agility and speed of innovation through rapid provisioning of new resources

- Analyze and optimize architecture for performance in the cloud
- Migrate fully to the cloud if all other requirements are met

Technical Requirements -
Handle both streaming and batch data

- Migrate existing Hadoop workloads
- Ensure architecture is scalable and elastic to meet the changing demands of the company.
- Use managed services whenever possible
- Encrypt data flight and at rest
- Connect a VPN between the production data center and cloud environment

SEO Statement -
We have grown so quickly that our inability to upgrade our infrastructure is really hampering further growth and efficiency. We are efficient at moving shipments around the world, but we are inefficient at moving data around. We need to organize our information so we can more easily understand where our customers are and what they are shipping.

CTO Statement -
IT has never been a priority for us, so as our data has grown, we have not invested enough in our technology. I have a good staff to manage IT, but they are so busy managing our infrastructure that I cannot get them to do the things that really matter, such as organizing our data, building the analytics, and figuring out how to implement the CFO' s tracking technology.

CFO Statement -
Part of our competitive advantage is that we penalize ourselves for late shipments and deliveries. Knowing where out shipments are at all times has a direct correlation to our bottom line and profitability. Additionally, I don't want to commit capital to building out a server environment.
Flowlogistic's CEO wants to gain rapid insight into their customer base so his sales team can be better informed in the field. This team is not very technical, so they've purchased a visualization tool to simplify the creation of BigQuery reports. However, they've been overwhelmed by all the data in the table, and are spending a lot of money on queries trying to find the data they need. You want to solve their problem in the most cost-effective way. What should you do?

A. Export the data into a Google Sheet for virtualization.

B. Create an additional table with only the necessary columns.

C. Create a view on the table to present to the virtualization tool.

D. Create identity and access management (IAM) roles on the appropriate columns, so only they appear in a query.

Answer: C

Explanation:

A logical view can be created with only the required columns which is required for visualization. B is not the right option as you will create a table and make it static. What happens when the original data is updated. This new table will not have the latest data and hence view is the best possible option here.

Question: 37 CertyIQ

Flowlogistic Case Study -

Company Overview -
Flowlogistic is a leading logistics and supply chain provider. They help businesses throughout the world manage their resources and transport them to their final destination. The company has grown rapidly, expanding their offerings to include rail, truck, aircraft, and oceanic shipping.

Company Background -
The company started as a regional trucking company, and then expanded into other logistics market. Because they have not updated their infrastructure, managing and tracking orders and shipments has become a bottleneck. To improve operations, Flowlogistic developed proprietary technology for tracking shipments in real time at the

parcel level. However, they are unable to deploy it because their technology stack, based on Apache Kafka, cannot support the processing volume. In addition, Flowlogistic wants to further analyze their orders and shipments to determine how best to deploy their resources.

Solution Concept -
Flowlogistic wants to implement two concepts using the cloud:
✑ Use their proprietary technology in a real-time inventory-tracking system that indicates the location of their loads
✑ Perform analytics on all their orders and shipment logs, which contain both structured and unstructured data, to determine how best to deploy resources, which markets to expand info. They also want to use predictive analytics to learn earlier when a shipment will be delayed.

Existing Technical Environment -
Flowlogistic architecture resides in a single data center:
✑ Databases
8 physical servers in 2 clusters
- SQL Server `" user data, inventory, static data
3 physical servers
- Cassandra `" metadata, tracking messages
10 Kafka servers `" tracking message aggregation and batch insert
✑ Application servers `" customer front end, middleware for order/customs
60 virtual machines across 20 physical servers
- Tomcat `" Java services
- Nginx `" static content
- Batch servers
✑ Storage appliances
- iSCSI for virtual machine (VM) hosts
- Fibre Channel storage area network (FC SAN) `" SQL server storage
- Network-attached storage (NAS) image storage, logs, backups
✑ 10 Apache Hadoop /Spark servers
- Core Data Lake
- Data analysis workloads
✑ 20 miscellaneous servers
- Jenkins, monitoring, bastion hosts,

Business Requirements -
✑ Build a reliable and reproducible environment with scaled panty of production.
✑ Aggregate data in a centralized Data Lake for analysis
✑ Use historical data to perform predictive analytics on future shipments
✑ Accurately track every shipment worldwide using proprietary technology
✑ Improve business agility and speed of innovation through rapid provisioning of new resources
✑ Analyze and optimize architecture for performance in the cloud
✑ Migrate fully to the cloud if all other requirements are met

Technical Requirements -
✑ Handle both streaming and batch data
✑ Migrate existing Hadoop workloads
✑ Ensure architecture is scalable and elastic to meet the changing demands of the company.
✑ Use managed services whenever possible
✑ Encrypt data flight and at rest
✑ Connect a VPN between the production data center and cloud environment

SEO Statement -
We have grown so quickly that our inability to upgrade our infrastructure is really hampering further growth and efficiency. We are efficient at moving shipments around the world, but we are inefficient at moving data around. We need to organize our information so we can more easily understand where our customers are and what they are shipping.

CTO Statement -
IT has never been a priority for us, so as our data has grown, we have not invested enough in our technology. I have a good staff to manage IT, but they are so busy managing our infrastructure that I cannot get them to do the things that really matter, such as organizing our data, building the analytics, and figuring out how to implement the CFO' s tracking technology.

CFO Statement -
Part of our competitive advantage is that we penalize ourselves for late shipments and deliveries. Knowing where

out shipments are at all times has a direct correlation to our bottom line and profitability. Additionally, I don't want to commit capital to building out a server environment.
Flowlogistic is rolling out their real-time inventory tracking system. The tracking devices will all send package-tracking messages, which will now go to a single
Google Cloud Pub/Sub topic instead of the Apache Kafka cluster. A subscriber application will then process the messages for real-time reporting and store them in
Google BigQuery for historical analysis. You want to ensure the package data can be analyzed over time.
Which approach should you take?

A. Attach the timestamp on each message in the Cloud Pub/Sub subscriber application as they are received.

B. Attach the timestamp and Package ID on the outbound message from each publisher device as they are sent to Clod Pub/Sub.

C. Use the NOW () function in BigQuery to record the event's time.

D. Use the automatically generated timestamp from Cloud Pub/Sub to order the data.

Answer: B

Explanation:

Okay, let's break down why option B is the correct approach for capturing timestamps in this scenario, and why the others are less suitable.

Justification for Option B (Correct):

Accuracy and Precision: Option B advocates attaching the timestamp at the source (the tracking device) along with the Package ID. This ensures the most accurate reflection of when the event (package movement) actually occurred. This is vital for reliable historical analysis as timestamps added later may not reflect the actual time of the event, which can be due to delays in the system.
Consistent Time Source: Each tracking device has its own clock. If the timestamp is added at the publisher (tracking device) then differences in time among devices can be addressed at the source.
Data Integrity: Embedding the Package ID alongside the timestamp guarantees that each time record is uniquely identified and connected to the correct package. This is crucial for precise tracking and analysis.
Immutable Data: Once the timestamp is attached at the source and sent via pub/sub, the data is considered more immutable. This prevents alterations or inconsistencies during later stages of processing.
Flexibility: This method enables using timestamps for various purposes beyond just ordering, such as calculating transit times or identifying anomalies.

Why Other Options Are Not Ideal:

Option A (Timestamp in Subscriber Application): Adding the timestamp within the subscriber application introduces delays based on network latency, processing time, and the subscriber's clock. This creates inaccuracies and defeats the purpose of a reliable timeline. Also, the publisher event time cannot be properly captured by the subscriber application.
Option C (BigQuery NOW() function): Using BigQuery's NOW() function only captures the time the record is inserted into BigQuery, not when the event actually occurred, making it unsuitable for historical analysis and package transit time calculations.
Option D (Cloud Pub/Sub Timestamp): While Cloud Pub/Sub automatically generates timestamps when a message is received, these timestamps reflect when the message hit the Pub/Sub service and not when the tracking device generated the event. Thus it does not capture the required event timeline accurately. While these automatically generated timestamps can be used for ordering data received by pub/sub, the question specifically asks about tracking over time and not ordering messages within pub/sub.

In Summary:

The most reliable and accurate way to ensure that package data can be analyzed over time is to attach the timestamp, along with the package ID at the publisher source (tracking device). This approach ensures integrity, accuracy, and proper data management and facilitates various analytics operations. This approach

adheres to the principle of capturing the most accurate representation of events as close to their source as possible.

Authoritative Links:

Cloud Pub/Sub Message Attributes: https://cloud.google.com/pubsub/docs/publisher#message_attributes - Learn how message attributes can be used to carry metadata like timestamps.
Google BigQuery Data Types: https://cloud.google.com/bigquery/docs/reference/standard-sql/data-types - Details on how BigQuery stores and handles different data types, including timestamps.
Best Practices for Time Series Data: Although not specifically from Google Cloud, understanding time-series data management is valuable, resources like "Designing Data-Intensive Applications" by Martin Kleppmann can be useful (https://www.oreilly.com/library/view/designing-data-intensive-applications/9781491903063/)

Question: 38 CertyIQ

MJTelco Case Study -

Company Overview -
MJTelco is a startup that plans to build networks in rapidly growing, underserved markets around the world. The company has patents for innovative optical communications hardware. Based on these patents, they can create many reliable, high-speed backbone links with inexpensive hardware.

Company Background -
Founded by experienced telecom executives, MJTelco uses technologies originally developed to overcome communications challenges in space. Fundamental to their operation, they need to create a distributed data infrastructure that drives real-time analysis and incorporates machine learning to continuously optimize their topologies. Because their hardware is inexpensive, they plan to overdeploy the network allowing them to account for the impact of dynamic regional politics on location availability and cost.
Their management and operations teams are situated all around the globe creating many-to-many relationship between data consumers and provides in their system. After careful consideration, they decided public cloud is the perfect environment to support their needs.

Solution Concept -
MJTelco is running a successful proof-of-concept (PoC) project in its labs. They have two primary needs:
✑ Scale and harden their PoC to support significantly more data flows generated when they ramp to more than 50,000 installations.
✑ Refine their machine-learning cycles to verify and improve the dynamic models they use to control topology definition.
MJTelco will also use three separate operating environments `" development/test, staging, and production `" to meet the needs of running experiments, deploying new features, and serving production customers.

Business Requirements -
✑ Scale up their production environment with minimal cost, instantiating resources when and where needed in an unpredictable, distributed telecom user community.
✑ Ensure security of their proprietary data to protect their leading-edge machine learning and analysis.
✑ Provide reliable and timely access to data for analysis from distributed research workers
✑ Maintain isolated environments that support rapid iteration of their machine-learning models without affecting their customers.

Technical Requirements -
✑ Ensure secure and efficient transport and storage of telemetry data
✑ Rapidly scale instances to support between 10,000 and 100,000 data providers with multiple flows each.
✑ Allow analysis and presentation against data tables tracking up to 2 years of data storing approximately 100m records/day
✑ Support rapid iteration of monitoring infrastructure focused on awareness of data pipeline problems both in telemetry flows and in production learning cycles.

CEO Statement -
Our business model relies on our patents, analytics and dynamic machine learning. Our inexpensive hardware is organized to be highly reliable, which gives us cost advantages. We need to quickly stabilize our large distributed data pipelines to meet our reliability and capacity commitments.

CTO Statement -
Our public cloud services must operate as advertised. We need resources that scale and keep our data secure. We also need environments in which our data scientists can carefully study and quickly adapt our models. Because we rely on automation to process our data, we also need our development and test environments to work as we iterate.

CFO Statement -
The project is too large for us to maintain the hardware and software required for the data and analysis. Also, we cannot afford to staff an operations team to monitor so many data feeds, so we will rely on automation and infrastructure. Google Cloud's machine learning will allow our quantitative researchers to work on our high-value problems instead of problems with our data pipelines.
MJTelco's Google Cloud Dataflow pipeline is now ready to start receiving data from the 50,000 installations. You want to allow Cloud Dataflow to scale its compute power up as required. Which Cloud Dataflow pipeline configuration setting should you update?

A. The zone
B. The number of workers
C. The disk size per worker
D. The maximum number of workers

Answer: D

Explanation:

The correct answer is **D. The maximum number of workers**.

Here's a detailed justification:

MJTelco needs its Cloud Dataflow pipeline to automatically scale its compute resources based on the incoming data volume from its 50,000 installations. They need the pipeline to be able to handle spikes in data flow without manual intervention. The crucial concept is autoscaling, and within Cloud Dataflow, the number of workers is the primary factor for horizontal scaling of resources.

Option A: The zone specifies the geographical location of resources. While important, it doesn't directly impact the dynamic scaling of compute resources based on data volume.
Option B: The number of workers refers to the initial number of workers requested when the pipeline starts. It doesn't enable automatic scaling.
Option C: The disk size per worker relates to storage per worker, which doesn't control the number of workers or compute resources for processing.

By setting a **maximum number of workers (D)**, Cloud Dataflow is authorized to automatically increase the number of worker instances as the data load increases, up to the configured limit. Conversely, when the data flow decreases, Dataflow can reduce the number of workers to optimize costs. This dynamic scaling aligns perfectly with MJTelco's need to handle fluctuating data volumes from a large number of providers, especially as they grow beyond 50,000 installations. This configuration lets Dataflow intelligently manage resource allocation, fulfilling their requirement for scalability and reliability while ensuring efficient resource use. It enables them to meet their reliability and capacity commitments and avoid manual intervention for resource scaling.

Here are authoritative links for further research:

Cloud Dataflow Autoscaling: https://cloud.google.com/dataflow/docs/concepts/autoscaling
Cloud Dataflow Pipeline Options: https://cloud.google.com/dataflow/docs/reference/pipeline-options
Autoscaling Concepts: https://cloud.google.com/compute/docs/autoscaler/

Question: 39 CertyIQ

MJTelco Case Study -

Company Overview -
MJTelco is a startup that plans to build networks in rapidly growing, underserved markets around the world. The company has patents for innovative optical communications hardware. Based on these patents, they can create many reliable, high-speed backbone links with inexpensive hardware.

Company Background -
Founded by experienced telecom executives, MJTelco uses technologies originally developed to overcome communications challenges in space. Fundamental to their operation, they need to create a distributed data infrastructure that drives real-time analysis and incorporates machine learning to continuously optimize their topologies. Because their hardware is inexpensive, they plan to overdeploy the network allowing them to account for the impact of dynamic regional politics on location availability and cost.
Their management and operations teams are situated all around the globe creating many-to-many relationship between data consumers and provides in their system. After careful consideration, they decided public cloud is the perfect environment to support their needs.

Solution Concept -
MJTelco is running a successful proof-of-concept (PoC) project in its labs. They have two primary needs:
✑ Scale and harden their PoC to support significantly more data flows generated when they ramp to more than 50,000 installations.
✑ Refine their machine-learning cycles to verify and improve the dynamic models they use to control topology definition.
MJTelco will also use three separate operating environments `" development/test, staging, and production `" to meet the needs of running experiments, deploying new features, and serving production customers.

Business Requirements -
✑ Scale up their production environment with minimal cost, instantiating resources when and where needed in an unpredictable, distributed telecom user community.
✑ Ensure security of their proprietary data to protect their leading-edge machine learning and analysis.
✑ Provide reliable and timely access to data for analysis from distributed research workers
Maintain isolated environments that support rapid iteration of their machine-learning models without affecting their customers.
▪

Technical Requirements -
✑ Ensure secure and efficient transport and storage of telemetry data
✑ Rapidly scale instances to support between 10,000 and 100,000 data providers with multiple flows each.
✑ Allow analysis and presentation against data tables tracking up to 2 years of data storing approximately 100m records/day
✑ Support rapid iteration of monitoring infrastructure focused on awareness of data pipeline problems both in telemetry flows and in production learning cycles.

CEO Statement -
Our business model relies on our patents, analytics and dynamic machine learning. Our inexpensive hardware is organized to be highly reliable, which gives us cost advantages. We need to quickly stabilize our large distributed data pipelines to meet our reliability and capacity commitments.

CTO Statement -
Our public cloud services must operate as advertised. We need resources that scale and keep our data secure. We also need environments in which our data scientists can carefully study and quickly adapt our models. Because we rely on automation to process our data, we also need our development and test environments to work as we iterate.

CFO Statement -
The project is too large for us to maintain the hardware and software required for the data and analysis. Also, we cannot afford to staff an operations team to monitor so many data feeds, so we will rely on automation and infrastructure. Google Cloud's machine learning will allow our quantitative researchers to work on our high-value problems instead of problems with our data pipelines.
You need to compose visualizations for operations teams with the following requirements:
✑ The report must include telemetry data from all 50,000 installations for the most resent 6 weeks (sampling once every minute).
✑ The report must not be more than 3 hours delayed from live data.
✑ The actionable report should only show suboptimal links.
✑ Most suboptimal links should be sorted to the top.
✑ Suboptimal links can be grouped and filtered by regional geography.
✑ User response time to load the report must be <5 seconds.

Which approach meets the requirements?

A. Load the data into Google Sheets, use formulas to calculate a metric, and use filters/sorting to show only suboptimal links in a table.

B. Load the data into Google BigQuery tables, write Google Apps Script that queries the data, calculates the metric, and shows only suboptimal rows in a table in Google Sheets.

C. Load the data into Google Cloud Datastore tables, write a Google App Engine Application that queries all rows, applies a function to derive the metric, and then renders results in a table using the Google charts and visualization API.

D. Load the data into Google BigQuery tables, write a Google Data Studio 360 report that connects to your data, calculates a metric, and then uses a filter expression to show only suboptimal rows in a table.

Answer: D

Explanation:

1. DataStudio and BQ are the simpliest way to do it
2. They also can activate BI Engine feature to improve the response time.

Question: 40 CertyIQ

MJTelco Case Study -

Company Overview -
MJTelco is a startup that plans to build networks in rapidly growing, underserved markets around the world. The company has patents for innovative optical communications hardware. Based on these patents, they can create many reliable, high-speed backbone links with inexpensive hardware.

Company Background -
Founded by experienced telecom executives, MJTelco uses technologies originally developed to overcome communications challenges in space. Fundamental to their operation, they need to create a distributed data infrastructure that drives real-time analysis and incorporates machine learning to continuously optimize their topologies. Because their hardware is inexpensive, they plan to overdeploy the network allowing them to account for the impact of dynamic regional politics on location availability and cost.
Their management and operations teams are situated all around the globe creating many-to-many relationship between data consumers and provides in their system. After careful consideration, they decided public cloud is the perfect environment to support their needs.

Solution Concept -
MJTelco is running a successful proof-of-concept (PoC) project in its labs. They have two primary needs:
- Scale and harden their PoC to support significantly more data flows generated when they ramp to more than 50,000 installations.
- Refine their machine-learning cycles to verify and improve the dynamic models they use to control topology definition.

MJTelco will also use three separate operating environments `" development/test, staging, and production `" to meet the needs of running experiments, deploying new features, and serving production customers.

Business Requirements -
- Scale up their production environment with minimal cost, instantiating resources when and where needed in an unpredictable, distributed telecom user community.
- Ensure security of their proprietary data to protect their leading-edge machine learning and analysis.

Provide reliable and timely access to data for analysis from distributed research workers
- Maintain isolated environments that support rapid iteration of their machine-learning models without affecting their customers.

Technical Requirements -
- Ensure secure and efficient transport and storage of telemetry data
- Rapidly scale instances to support between 10,000 and 100,000 data providers with multiple flows each.
- Allow analysis and presentation against data tables tracking up to 2 years of data storing approximately 100m records/day
- Support rapid iteration of monitoring infrastructure focused on awareness of data pipeline problems both in telemetry flows and in production learning cycles.

CEO Statement -
Our business model relies on our patents, analytics and dynamic machine learning. Our inexpensive hardware is organized to be highly reliable, which gives us cost advantages. We need to quickly stabilize our large distributed data pipelines to meet our reliability and capacity commitments.

CTO Statement -
Our public cloud services must operate as advertised. We need resources that scale and keep our data secure. We also need environments in which our data scientists can carefully study and quickly adapt our models. Because we rely on automation to process our data, we also need our development and test environments to work as we iterate.

CFO Statement -
The project is too large for us to maintain the hardware and software required for the data and analysis. Also, we cannot afford to staff an operations team to monitor so many data feeds, so we will rely on automation and infrastructure. Google Cloud's machine learning will allow our quantitative researchers to work on our high-value problems instead of problems with our data pipelines.
You create a new report for your large team in Google Data Studio 360. The report uses Google BigQuery as its data source. It is company policy to ensure employees can view only the data associated with their region, so you create and populate a table for each region. You need to enforce the regional access policy to the data.
Which two actions should you take? (Choose two.)

A. Ensure all the tables are included in global dataset.
B. Ensure each table is included in a dataset for a region.
C. Adjust the settings for each table to allow a related region-based security group view access.
D. Adjust the settings for each view to allow a related region-based security group view access.
E. Adjust the settings for each dataset to allow a related region-based security group view access.

Answer: BE

Explanation:

Even if now BQ offers table level access control, since the number of tables can be expected to be high, controlling access at the dataset level would ease operations. That is why I would still go for E instead of C.

Question: 41 CertyIQ

MJTelco Case Study -

Company Overview -
MJTelco is a startup that plans to build networks in rapidly growing, underserved markets around the world. The company has patents for innovative optical communications hardware. Based on these patents, they can create many reliable, high-speed backbone links with inexpensive hardware.

Company Background -
Founded by experienced telecom executives, MJTelco uses technologies originally developed to overcome communications challenges in space. Fundamental to their operation, they need to create a distributed data infrastructure that drives real-time analysis and incorporates machine learning to continuously optimize their topologies. Because their hardware is inexpensive, they plan to overdeploy the network allowing them to account for the impact of dynamic regional politics on location availability and cost.
Their management and operations teams are situated all around the globe creating many-to-many relationship between data consumers and provides in their system. After careful consideration, they decided public cloud is the perfect environment to support their needs.

Solution Concept -
MJTelco is running a successful proof-of-concept (PoC) project in its labs. They have two primary needs:
✑ Scale and harden their PoC to support significantly more data flows generated when they ramp to more than 50,000 installations.
✑ Refine their machine-learning cycles to verify and improve the dynamic models they use to control topology definition.

MJTelco will also use three separate operating environments `" development/test, staging, and production `" to meet the needs of running experiments, deploying new features, and serving production customers.

Business Requirements -
✑ Scale up their production environment with minimal cost, instantiating resources when and where needed in an unpredictable, distributed telecom user community.
✑ Ensure security of their proprietary data to protect their leading-edge machine learning and analysis.
✑ Provide reliable and timely access to data for analysis from distributed research workers
✑ Maintain isolated environments that support rapid iteration of their machine-learning models without affecting their customers.

Technical Requirements -
Ensure secure and efficient transport and storage of telemetry data
▪
✑ Rapidly scale instances to support between 10,000 and 100,000 data providers with multiple flows each.
✑ Allow analysis and presentation against data tables tracking up to 2 years of data storing approximately 100m records/day
✑ Support rapid iteration of monitoring infrastructure focused on awareness of data pipeline problems both in telemetry flows and in production learning cycles.

CEO Statement -
Our business model relies on our patents, analytics and dynamic machine learning. Our inexpensive hardware is organized to be highly reliable, which gives us cost advantages. We need to quickly stabilize our large distributed data pipelines to meet our reliability and capacity commitments.

CTO Statement -
Our public cloud services must operate as advertised. We need resources that scale and keep our data secure. We also need environments in which our data scientists can carefully study and quickly adapt our models. Because we rely on automation to process our data, we also need our development and test environments to work as we iterate.

CFO Statement -
The project is too large for us to maintain the hardware and software required for the data and analysis. Also, we cannot afford to staff an operations team to monitor so many data feeds, so we will rely on automation and infrastructure. Google Cloud's machine learning will allow our quantitative researchers to work on our high-value problems instead of problems with our data pipelines.
MJTelco needs you to create a schema in Google Bigtable that will allow for the historical analysis of the last 2 years of records. Each record that comes in is sent every 15 minutes, and contains a unique identifier of the device and a data record. The most common query is for all the data for a given device for a given day.
Which schema should you use?

A. Rowkey: date#device_id Column data: data_point
B. Rowkey: date Column data: device_id, data_point
C. Rowkey: device_id Column data: date, data_point
D. Rowkey: data_point Column data: device_id, date
E. Rowkey: date#data_point Column data: device_id

Answer: D

Explanation:

rowkey be Device_Id+Date(reverse)

Question: 42 CertyIQ

Your company has recently grown rapidly and now ingesting data at a significantly higher rate than it was previously. You manage the daily batch MapReduce analytics jobs in Apache Hadoop. However, the recent increase in data has meant the batch jobs are falling behind. You were asked to recommend ways the development team could increase the responsiveness of the analytics without increasing costs. What should you recommend they do?

A. Rewrite the job in Pig.

B. Rewrite the job in Apache Spark.

C. Increase the size of the Hadoop cluster.

D. Decrease the size of the Hadoop cluster but also rewrite the job in Hive.

Answer: B

Explanation:

Rewriting the MapReduce job in Apache Spark (Option B) is the most cost-effective solution for improving responsiveness. Here's why:

Spark's in-memory processing capabilities offer significantly faster performance compared to Hadoop's disk-based MapReduce. Spark can store intermediate data in memory, reducing the need for costly disk I/O operations that slow down Hadoop jobs. This leads to quicker processing times for the same amount of data.

While increasing the Hadoop cluster size (Option C) might seem like a direct solution, it involves higher infrastructure costs (more nodes, storage, and network resources). Given the constraint of not increasing costs, this is not the best approach.

Rewriting the job in Pig (Option A) or Hive (Option D) might simplify development, but they ultimately translate to MapReduce jobs under the hood. These high-level languages don't inherently improve performance beyond what can be achieved with optimized MapReduce code. Furthermore, rewriting in Hive and decreasing the cluster size is counterintuitive if the goal is to increase responsiveness. Reducing the cluster size while simply changing the query language exacerbates the problem.

Spark's ability to handle data transformations efficiently, especially in a distributed environment, makes it ideal for large datasets. It also supports various data formats and integrations with other data processing tools. The performance gain from using Spark can outweigh the development effort required to rewrite the existing MapReduce job, while keeping costs constant. Spark's DataFrame API, in particular, provides an abstraction layer that enables optimization for processing, further improving overall speed.

Apache Spark DocumentationUnderstanding Spark Performance

Question: 43 CertyIQ

You work for a large fast food restaurant chain with over 400,000 employees. You store employee information in Google BigQuery in a Users table consisting of a FirstName field and a LastName field. A member of IT is building an application and asks you to modify the schema and data in BigQuery so the application can query a FullName field consisting of the value of the FirstName field concatenated with a space, followed by the value of the LastName field for each employee. How can you make that data available while minimizing cost?

A. Create a view in BigQuery that concatenates the FirstName and LastName field values to produce the FullName.

B. Add a new column called FullName to the Users table. Run an UPDATE statement that updates the FullName column for each user with the concatenation of the FirstName and LastName values.

C. Create a Google Cloud Dataflow job that queries BigQuery for the entire Users table, concatenates the FirstName value and LastName value for each user, and loads the proper values for FirstName, LastName, and FullName into a new table in BigQuery.

D. Use BigQuery to export the data for the table to a CSV file. Create a Google Cloud Dataproc job to process the CSV file and output a new CSV file containing the proper values for FirstName, LastName and FullName. Run a BigQuery load job to load the new CSV file into BigQuery.

Answer: A

Explanation:

The best approach is to create a BigQuery view that concatenates the FirstName and LastName fields. This minimizes cost because a view doesn't store any data itself. It's a virtual table defined by a query. When the application queries the view, the concatenation happens on-the-fly, using BigQuery's compute resources only when the query is executed.

Option B, adding a new column and updating the table, incurs storage costs for the new column and processing costs for the UPDATE statement. While it might seem efficient in terms of query performance, it adds unnecessary storage and modifies the existing table structure.

Option C, using Dataflow, involves significant overhead. It requires setting up and running a Dataflow pipeline, which consumes more resources and thus costs more than a simple view. It also creates a new table, increasing storage costs. Dataflow is better suited for complex data transformations and real-time processing, not a simple concatenation.

Option D, exporting to CSV, using Dataproc, and reloading, is the most expensive and complex option. It involves exporting the data, setting up a Dataproc cluster, writing and executing a Spark job (or similar) for processing, and then reloading the data into BigQuery. This incurs costs for storage, Dataproc cluster resources, and BigQuery load jobs. It's also unnecessary given the simplicity of the required transformation.

Views provide a cost-effective and efficient way to present transformed data without storing redundant information. In this scenario, where a simple concatenation is needed, a BigQuery view is the ideal solution.

Further Reading:

BigQuery Views: https://cloud.google.com/bigquery/docs/views
BigQuery Pricing: https://cloud.google.com/bigquery/pricing

Question: 44 CertyIQ

You are deploying a new storage system for your mobile application, which is a media streaming service. You decide the best fit is Google Cloud Datastore. You have entities with multiple properties, some of which can take on multiple values. For example, in the entity 'Movie' the property 'actors' and the property 'tags' have multiple values but the property 'date released' does not. A typical query would ask for all movies with actor=<actorname> ordered by date_released or all movies with tag=Comedy ordered by date_released. How should you avoid a combinatorial explosion in the number of indexes?

A. Manually configure the index in your index config as follows:

```
Indexes:
-kind: Movie
    Properties:
    -name: actors
    name: date_released
-kind: Movie
    Properties:
    -name: tags
    name: date_released
```

B. Manually configure the index in your index config as follows:

```
Indexes:
    -kind: Movie
        Properties:
        -name: actors
        -name: tags
-name: date_published
```

C. Set the following in your entity options: exclude_from_indexes = 'actors, tags'

D. Set the following in your entity options: exclude_from_indexes = 'date_published'

Answer: A

Explanation:

From Google cloud documentation

The rows of an index table are sorted first by ancestor and then by property values, in the order specified in the index definition. The perfect index for a query, which allows the query to be executed most efficiently, is defined on the following properties, in order:

Properties used in equality filters

Property used in an inequality filter (of which there can be no more than one)

Properties used in sort orders

Properties used in projections (that are not already included in sort orders)

Question: 45 CertyIQ

You work for a manufacturing plant that batches application log files together into a single log file once a day at

2:00 AM. You have written a Google Cloud
Dataflow job to process that log file. You need to make sure the log file in processed once per day as inexpensively as possible. What should you do?

A. Change the processing job to use Google Cloud Dataproc instead.

B. Manually start the Cloud Dataflow job each morning when you get into the office.

C. Create a cron job with Google App Engine Cron Service to run the Cloud Dataflow job.

D. Configure the Cloud Dataflow job as a streaming job so that it processes the log data immediately.

Answer: C

Explanation:

The correct answer is **C. Create a cron job with Google App Engine Cron Service to run the Cloud Dataflow job.** Here's why:

The problem requires processing a daily batch log file at a specific time (2:00 AM) in the most cost-effective way. Option A, using Google Cloud Dataproc, is less suitable for this scenario. Dataproc is designed for large-scale data processing using Hadoop and Spark, adding unnecessary complexity and cost for a simple daily batch job. Option B, manually starting the job, is inefficient and prone to human error, making it unreliable. Option D, configuring the job as a streaming job, is inappropriate as the data arrives in a batch once a day, not as a continuous stream.

Google App Engine Cron Service, as suggested in option C, perfectly addresses the need. It allows scheduling tasks to run automatically at specific times. You can configure a cron job to trigger your Cloud Dataflow job every day at 2:00 AM, ensuring timely and automatic execution. This method leverages a fully managed serverless service, reducing operational overhead and cost compared to maintaining a dedicated compute instance. Moreover, cron jobs are designed for such scheduled executions, making them a perfect fit for this requirement. This approach ensures cost-effectiveness as you only pay for the cron service and Dataflow execution during its scheduled run, without having a compute instance running continuously. This adheres to the principle of least cost while ensuring timely job execution.

For further information on Google App Engine Cron Service, please refer to:
https://cloud.google.com/appengine/docs/standard/python/config/cron and
https://cloud.google.com/scheduler/docsand for Cloud Dataflow: https://cloud.google.com/dataflow/docs/

Question: 46 CertyIQ

You work for an economic consulting firm that helps companies identify economic trends as they happen. As part of your analysis, you use Google BigQuery to correlate customer data with the average prices of the 100 most common goods sold, including bread, gasoline, milk, and others. The average prices of these goods are updated every 30 minutes. You want to make sure this data stays up to date so you can combine it with other data in BigQuery as cheaply as possible.
What should you do?

A. Load the data every 30 minutes into a new partitioned table in BigQuery.

B. Store and update the data in a regional Google Cloud Storage bucket and create a federated data source in BigQuery

C. Store the data in Google Cloud Datastore. Use Google Cloud Dataflow to query BigQuery and combine the data programmatically with the data stored in Cloud Datastore

D. Store the data in a file in a regional Google Cloud Storage bucket. Use Cloud Dataflow to query BigQuery and combine the data programmatically with the data stored in Google Cloud Storage.

Answer: B

Explanation:

The correct answer is **B. Store and update the data in a regional Google Cloud Storage bucket and create a federated data source in BigQuery.**

Here's why:

Cost-Effectiveness: The prompt explicitly states the need to combine data in BigQuery as cheaply as possible. Storing the frequently updated price data in Google Cloud Storage (GCS) and then querying it as an external table using BigQuery Federated Queries is generally more cost-effective than loading the data into BigQuery itself every 30 minutes. BigQuery charges for storage and queries, and frequent loading incurs both costs.

Near Real-Time Access: Federated queries allow you to query data directly from GCS without loading it into BigQuery. This allows for near real-time access to the price data, which is crucial for identifying economic trends "as they happen."

Simplicity: This approach avoids the complexity of data processing pipelines involving Cloud Dataflow (options C and D). While Dataflow is powerful, it's overkill for simply combining data from GCS with BigQuery. It also reduces operational overhead.

Partitioning in BigQuery (Option A): While partitioning could help query performance if the data was already in BigQuery, loading data every 30 minutes into a new partitioned table is inefficient and would lead to a proliferation of tables and significant storage and query costs. Furthermore, partitioning will not be triggered by data changes.

Cloud Datastore (Option C): Cloud Datastore is a NoSQL database. While it can store frequently updated data, querying Datastore with BigQuery using Dataflow adds complexity and cost compared to a federated query with GCS. Datastore is also not designed for analytical queries like those you'd typically perform in BigQuery for economic trend analysis.

Cloud Dataflow with GCS (Option D): Similar to option C, using Dataflow to combine GCS data with BigQuery adds unnecessary complexity and cost. Dataflow is suitable for complex data transformations, but not for directly querying data stored in GCS and combining it.

In Summary: Option B offers a cost-effective, near real-time, and simple solution that meets the stated requirements. It leverages the strengths of both GCS and BigQuery without introducing unnecessary complexity.

Authoritative Links:

BigQuery Federated Queries: https://cloud.google.com/bigquery/docs/external-data-cloud-storage
BigQuery Pricing: https://cloud.google.com/bigquery/pricing
Google Cloud Storage Pricing: https://cloud.google.com/storage/pricing

Question: 47 CertyIQ

You are designing the database schema for a machine learning-based food ordering service that will predict what users want to eat. Here is some of the information you need to store:
- The user profile: What the user likes and doesn't like to eat
- The user account information: Name, address, preferred meal times
- The order information: When orders are made, from where, to whom

The database will be used to store all the transactional data of the product. You want to optimize the data schema. Which Google Cloud Platform product should you use?

A. BigQuery
B. Cloud SQL

C. Cloud Bigtable

D. Cloud Datastore

Answer: A

Explanation:

The most appropriate Google Cloud Platform product for this scenario is BigQuery (A). The question describes a machine learning-based food ordering service that requires analysis of user profiles, account information, and order details. This inherently suggests a need for analytical capabilities and the ability to perform complex queries across large datasets. BigQuery is a fully-managed, serverless data warehouse designed for precisely these purposes. It excels at handling massive datasets, offering SQL-based querying, and enabling integration with machine learning services for model training and prediction. Unlike Cloud SQL (B), a relational database service, BigQuery isn't designed for transactional data, but rather for analytical workloads. Cloud Bigtable (C), a NoSQL wide-column store, and Cloud Datastore (D), a NoSQL document database, are better suited for high-throughput, low-latency operations rather than complex analytical processing. While these NoSQL options could store the data, they would be less efficient for analytical tasks needed to predict user preferences. BigQuery facilitates efficient data warehousing and allows for optimized analysis which is essential for the machine learning model. Furthermore, BigQuery's serverless architecture simplifies infrastructure management, allowing the focus to remain on developing and training the prediction model.

Authoritative Links:

BigQuery Documentation: https://cloud.google.com/bigquery/docs
Choosing a storage option: https://cloud.google.com/learn/what-cloud-storage-option

Question: 48 CertyIQ

Your company is loading comma-separated values (CSV) files into Google BigQuery. The data is fully imported successfully; however, the imported data is not matching byte-to-byte to the source file. What is the most likely cause of this problem?

A. The CSV data loaded in BigQuery is not flagged as CSV.

B. The CSV data has invalid rows that were skipped on import.

C. The CSV data loaded in BigQuery is not using BigQuery's default encoding.

D. The CSV data has not gone through an ETL phase before loading into BigQuery.

Answer: C

Explanation:

The correct answer is C, "The CSV data loaded in BigQuery is not using BigQuery's default encoding." Here's why:

BigQuery, like many data processing systems, defaults to UTF-8 encoding for text data. CSV files, however, can use various encodings (e.g., ASCII, Latin-1, UTF-16, etc.). If the source CSV file uses an encoding other than UTF-8, and this is not specified during the BigQuery load process, BigQuery will interpret the data incorrectly. This misinterpretation leads to byte-level differences compared to the original file, even if the data visually appears to load without error. The issue isn't about the data being identified as CSV (option A) as BigQuery knows this by its loading process. Skipped rows (option B), while a valid concern, don't typically change the byte-to-byte representation of successfully imported rows. Similarly, lack of ETL (option D) may affect data transformation but does not alter fundamental byte representation if the loading process works correctly. Therefore, the encoding mismatch is the prime suspect when byte-level discrepancies exist despite successful loading. Correctly specifying the source encoding using the encoding option during a load job is

crucial for ensuring data integrity and accurate byte representation in BigQuery. This ensures that character representations in the CSV files are correctly translated within BigQuery.

Further information on how to specify encoding can be found in the official Google Cloud documentation:

Loading data from Cloud Storage
BigQuery data types for how encoding impacts data representation.

Question: 49 CertyIQ

Your company produces 20,000 files every hour. Each data file is formatted as a comma separated values (CSV) file that is less than 4 KB. All files must be ingested on Google Cloud Platform before they can be processed. Your company site has a 200 ms latency to Google Cloud, and your Internet connection bandwidth is limited as 50 Mbps. You currently deploy a secure FTP (SFTP) server on a virtual machine in Google Compute Engine as the data ingestion point. A local SFTP client runs on a dedicated machine to transmit the CSV files as is. The goal is to make reports with data from the previous day available to the executives by 10:00 a.m. each day. This design is barely able to keep up with the current volume, even though the bandwidth utilization is rather low.
You are told that due to seasonality, your company expects the number of files to double for the next three months. Which two actions should you take? (Choose two.)

A. Introduce data compression for each file to increase the rate file of file transfer.

B. Contact your internet service provider (ISP) to increase your maximum bandwidth to at least 100 Mbps.

C. Redesign the data ingestion process to use gsutil tool to send the CSV files to a storage bucket in parallel.

D. Assemble 1,000 files into a tape archive (TAR) file. Transmit the TAR files instead, and disassemble the CSV files in the cloud upon receiving them.

E. Create an S3-compatible storage endpoint in your network, and use Google Cloud Storage Transfer Service to transfer on-premises data to the designated storage bucket.

Answer: CD

Explanation:

The correct actions are **C and D**.

Here's why:

C. Redesign the data ingestion process to use gsutil tool to send the CSV files to a storage bucket in parallel.

Problem: The current SFTP approach is likely bottlenecked by the latency and sequential file transfers, even with low bandwidth utilization. SFTP inherently performs one-to-one transfers, leading to inefficiency with many small files.
Solution: gsutil is a command-line tool that leverages Google Cloud Storage's (GCS) capabilities for parallel uploads. This allows for multiple files to be transferred concurrently, significantly reducing the overall transfer time. Parallel uploads minimize the impact of network latency by keeping the connection consistently utilized.
Efficiency: This approach aligns with best practices for ingesting large amounts of data into GCS.
Documentation: https://cloud.google.com/storage/docs/gsutil

D. Assemble 1,000 files into a tape archive (TAR) file. Transmit the TAR files instead, and disassemble the CSV files in the cloud upon receiving them.

Problem: 20,000 small files/hour creates significant overhead due to the initiation of individual network connections and request handshakes.
Solution: By combining files into a smaller number of larger TAR files, you reduce the number of individual network transfers, diminishing overhead and optimizing throughput. The slight overhead of packing and

unpacking is much lower than the network transfer overhead with large number of files.
Efficiency: Moving larger files reduces overheads related to network requests, enhancing efficiency during transfer.
Optimization: This consolidates numerous small file operations into few large ones.
Context: This practice is very common when dealing with transferring large number of small files across a network, be it on-prem or cloud.

Why other options are not correct:

A. Introduce data compression: While compression can reduce the data volume, the files are already small (4KB). The gain will likely be minimal. The problem is more with the sheer number of files.
B. Increase bandwidth: While helpful, the main issue is the latency and the inefficient handling of multiple, small files through SFTP. Increased bandwidth alone won't solve the efficiency problems of handling a large volume of small files, and is not the most efficient way to solve the problem.
E. S3-compatible storage and transfer service: This solution introduces a new piece of infrastructure, adds complexity, and doesn't solve the core issue of sending many small files efficiently. Using gsutil directly to GCS is a better approach.

In summary, parallel uploads (C) and file consolidation (D) are the most effective actions for handling the increased file volume and meeting the reporting deadline given network latency and limited bandwidth.

Question: 50 CertyIQ

You are choosing a NoSQL database to handle telemetry data submitted from millions of Internet-of-Things (IoT) devices. The volume of data is growing at 100
TB per year, and each data entry has about 100 attributes. The data processing pipeline does not require atomicity, consistency, isolation, and durability (ACID).
However, high availability and low latency are required.
You need to analyze the data by querying against individual fields. Which three databases meet your requirements? (Choose three.)

A. Redis
B. HBase
C. MySQL
D. MongoDB
E. Cassandra
F. HDFS with Hive

Answer: BDE

Explanation:

The problem requires a NoSQL database for high-volume IoT telemetry data, prioritizing high availability and low latency while sacrificing ACID properties. The need to query against individual fields also dictates a specific type of NoSQL solution. Let's evaluate each option:

Redis (A): While Redis is a fast, in-memory data store, it primarily functions as a key-value or data structure server. It lacks the necessary features for querying 100 attributes per entry in an efficient manner, and is not designed for storing and retrieving data at the scale of 100TB per year. Thus, Redis is not a good choice.
HBase (B): HBase is a column-oriented, NoSQL database designed for handling large datasets, making it suitable for the scale requirements. It offers high availability and is optimized for low latency reads and writes. It's particularly well-suited for querying based on fields/columns, aligning with the requirements. [https://hbase.apache.org/]
MySQL (C): MySQL is a relational database (SQL) that enforces ACID properties, which the problem states are

not required. It might also struggle with the sheer volume of data and low latency writes demanded by the IoT data stream. Relational databases are not designed for such high scale as quickly as NoSQL databases.
MongoDB (D): MongoDB is a document-oriented database. It is capable of handling large data volumes, and its flexible schema makes it suitable for handling the 100 attributes per entry. Furthermore, MongoDB can provide high availability and low latency reads and writes, meeting the requirements.
[https://www.mongodb.com/]
Cassandra (E): Cassandra is another column-oriented NoSQL database that excels at handling large-scale data with high availability and low latency. It is built for write-heavy workloads, like telemetry data, and can efficiently query data using the fields defined within the columns, fitting the use case.
[https://cassandra.apache.org/]
HDFS with Hive (F): HDFS is a distributed file system designed for batch processing and large data storage, not real-time data access and low-latency reads. Hive is a query engine that operates on top of HDFS, further adding to the latency, and it's not well-suited for the low latency requirement of this use case.

Therefore, the three databases that best meet the requirements are **HBase (B), MongoDB (D), and Cassandra (E)**. They are NoSQL solutions that support high availability, low latency, large volumes of data, and efficient querying based on fields.

Question: 51 CertyIQ

You are training a spam classifier. You notice that you are overfitting the training data. Which three actions can you take to resolve this problem? (Choose three.)

A. Get more training examples
B. Reduce the number of training examples
C. Use a smaller set of features
D. Use a larger set of features
E. Increase the regularization parameters
F. Decrease the regularization parameters

Answer: ACE

Explanation:

Overfitting occurs when a model learns the training data too well, capturing noise and specific patterns that don't generalize to unseen data. This results in high accuracy on training data but poor performance on new data. To combat overfitting, we need to make the model less sensitive to the nuances of the training set and more focused on underlying, generalizable patterns.

Option A, **"Get more training examples,"** helps by providing the model with a wider variety of data points. This exposes the model to more diverse scenarios, making it less likely to memorize the specifics of a small training dataset and improving generalization. This aligns with the core principles of statistical learning, where more data typically leads to better model generalization.

Option B, **"Reduce the number of training examples,"** would worsen overfitting, therefore it's incorrect. Reducing training data makes it more likely for a model to memorize the noise of the limited dataset.

Option C, **"Use a smaller set of features,"** is effective in reducing overfitting. Models trained on datasets with too many features might learn relationships specific to those features in the training set that might not exist outside. Feature selection, or reducing the feature space, prevents the model from overfitting to irrelevant details, enabling it to focus on more important patterns.

Option D, **"Use a larger set of features,"** would worsen overfitting, therefore it's incorrect. Adding more

features increase the chances of the model learning the spurious correlations present in training data.

Option E, **"Increase the regularization parameters,"** is another method to mitigate overfitting. Regularization techniques, such as L1 or L2 regularization, add penalties to the model's complexity, discouraging it from learning overly intricate patterns that generalize poorly. Increasing regularization makes it harder to model noise or outliers. This concept is explained in detail on Google Cloud documentation related to ML model training.

Option F, **"Decrease the regularization parameters,"** would worsen overfitting, therefore it's incorrect.

In summary, by obtaining more data (A), simplifying the input features (C), and reducing model complexity through regularization (E), we force the model to focus on generalizable patterns rather than memorizing the training set, effectively reducing overfitting.

Authoritative Links:

Google Cloud AI Platform documentation on training ML models: https://cloud.google.com/ai-platform/docs/ml-solutions/overfitting
Google Machine Learning Crash Course on Regularization: https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/video-lecture

Question: 52 CertyIQ

You are implementing security best practices on your data pipeline. Currently, you are manually executing jobs as the Project Owner. You want to automate these jobs by taking nightly batch files containing non-public information from Google Cloud Storage, processing them with a Spark Scala job on a Google Cloud
Dataproc cluster, and depositing the results into Google BigQuery.
How should you securely run this workload?

A. Restrict the Google Cloud Storage bucket so only you can see the files

B. Grant the Project Owner role to a service account, and run the job with it

C. Use a service account with the ability to read the batch files and to write to BigQuery

D. Use a user account with the Project Viewer role on the Cloud Dataproc cluster to read the batch files and write to BigQuery

Answer: C

Explanation:

The correct answer is **C. Use a service account with the ability to read the batch files and to write to BigQuery.** Here's why:

Option C embodies the principle of least privilege, a core security best practice. Service accounts are designed for applications and services, not human users. By granting a service account only the necessary permissions (read access to Cloud Storage and write access to BigQuery), we minimize the risk associated with overly permissive access. If the service account is compromised, the potential damage is limited.

Option A, restricting the Cloud Storage bucket solely to the Project Owner, is insufficient. While good practice generally, this doesn't automate job execution. You would still have to manually initiate the Spark job, which contradicts the requirement for automated nightly runs.

Option B, granting the Project Owner role to a service account, is overly permissive. The Project Owner role provides broad access, far exceeding what's needed for the job, increasing the attack surface. It violates the principle of least privilege and poses a significant security risk.

Option D, using a user account, even with Project Viewer, is a poor security practice for automated processes.

User accounts are tied to individuals and not appropriate for application-level access, they also lack the security benefit of tightly scoped permissions.

Therefore, using a dedicated service account with specific, restricted permissions is the most secure and efficient way to automate the data pipeline. This follows cloud security best practices and minimizes the potential for unauthorized access or data breaches. This allows for automated operation and enhanced security.

For further research, refer to the Google Cloud documentation on:

Service Accounts: https://cloud.google.com/iam/docs/service-accounts
Principle of Least Privilege: https://cloud.google.com/security/gdpr/principles/least-privilege
IAM Best Practices: https://cloud.google.com/iam/docs/best-practices

Question: 53 CertyIQ

You are using Google BigQuery as your data warehouse. Your users report that the following simple query is running very slowly, no matter when they run the query:
SELECT country, state, city FROM [myproject:mydataset.mytable] GROUP BY country
You check the query plan for the query and see the following output in the Read section of Stage:1:

What is the most likely cause of the delay for this query?

A. Users are running too many concurrent queries in the system
B. The [myproject:mydataset.mytable] table has too many partitions
C. Either the state or the city columns in the [myproject:mydataset.mytable] table have too many NULL values
D. Most rows in the [myproject:mydataset.mytable] table have the same value in the country column, causing data skew

Answer: D

Explanation:

D Image says that average(dark) and maximum(light) have difference in few times, this it is a skew The color indicators show the relative timings for all steps across all stages. For example, the COMPUTE step of Stage 00 shows a bar whose shaded fraction is 21/30 since 30ms is the maximum time spent in a single step of any stage. The parallel input information shows that each stage required only a single worker, so there's no variance between average and slowest timings.

Reference:

https://cloud.google.com/bigquery/query-plan

https://cloud.google.com/bigquery/docs/best-practices-performance-patterns

Question: 54 CertyIQ

Your globally distributed auction application allows users to bid on items. Occasionally, users place identical bids at nearly identical times, and different application servers process those bids. Each bid event contains the item, amount, user, and timestamp. You want to collate those bid events into a single location in real time to determine which user bid first. What should you do?

A. Create a file on a shared file and have the application servers write all bid events to that file. Process the file with Apache Hadoop to identify which user bid first.

B. Have each application server write the bid events to Cloud Pub/Sub as they occur. Push the events from Cloud Pub/Sub to a custom endpoint that writes the bid event information into Cloud SQL.

C. Set up a MySQL database for each application server to write bid events into. Periodically query each of those distributed MySQL databases and update a master MySQL database with bid event information.

D. Have each application server write the bid events to Google Cloud Pub/Sub as they occur. Use a pull subscription to pull the bid events using Google Cloud Dataflow. Give the bid for each item to the user in the bid event that is processed first.

Answer: B

Explanation:

The optimal solution is **B**, utilizing Cloud Pub/Sub and Cloud SQL. Here's why:

Option **A** introduces a shared file system for bid events, which suffers from concurrency issues and potential data loss. Processing with Hadoop adds latency, hindering real-time analysis crucial for determining the first bid. File systems are not designed for high-throughput writes from multiple sources concurrently, making this solution unsuitable.

Option **C**, using distributed MySQL databases, introduces complexity in querying and synchronizing data. Regularly querying and updating a master database creates delays and is not ideal for real-time scenarios. Managing multiple MySQL instances and reconciliation is also operationally complex.

Option **B** leverages Cloud Pub/Sub, a messaging service that enables decoupled and scalable real-time data ingestion. This ensures that bid events are reliably captured as they occur. Dataflow offers a managed service for stream and batch data processing. Although Dataflow is mentioned in Option D, the critical difference is the data landing location. Option B uses a custom endpoint that lands into Cloud SQL, where data is stored in a structured manner. While both use Pub/Sub, option D implies processing within Dataflow itself, not landing the data for later analysis, which is needed in this scenario. Cloud SQL is a managed, relational database for structured data storage, making it perfect for quickly identifying the first bid. Push subscriptions in Pub/Sub ensure immediate event delivery to the custom endpoint, enabling timely analysis and resolution of concurrent bids. The architecture in Option B allows real-time analysis within milliseconds.

Here are some links for further research:

Cloud Pub/Sub: https://cloud.google.com/pubsub/docs
Cloud SQL: https://cloud.google.com/sql/docs
Cloud Dataflow: https://cloud.google.com/dataflow/docs
Push vs Pull Subscriptions in Pub/Sub: https://cloud.google.com/pubsub/docs/subscriber#push_and_pull

Therefore, option **B** provides a robust, scalable, and real-time solution for processing and analyzing bid events.

Question: 55 CertyIQ

Your organization has been collecting and analyzing data in Google BigQuery for 6 months. The majority of the data analyzed is placed in a time-partitioned table named events_partitioned. To reduce the cost of queries, your organization created a view called events, which queries only the last 14 days of data. The view is described in legacy SQL. Next month, existing applications will be connecting to BigQuery to read the events data via an ODBC connection. You need to ensure the applications can connect. Which two actions should you take? (Choose two.)

A. Create a new view over events using standard SQL

B. Create a new partitioned table using a standard SQL query

C. Create a new view over events_partitioned using standard SQL

D. Create a service account for the ODBC connection to use for authentication

E. Create a Google Cloud Identity and Access Management (Cloud IAM) role for the ODBC connection and shared events

Answer: CD

Explanation:

Here's a detailed justification for choosing options C and D as the correct actions:

Option C: Create a new view over events_partitioned using standard SQL.

The original view events uses legacy SQL, which is deprecated and not compatible with ODBC connections using the BigQuery API. To enable ODBC connectivity, you must rewrite the view in standard SQL. Creating a new view directly on the time-partitioned table events_partitioned ensures that the view will continue to only query the data needed by applying filters. This improves efficiency and reduces cost compared to querying the whole table. Also, standard SQL has enhanced functionality and compatibility and is the recommended query language for BigQuery.

Option D: Create a service account for the ODBC connection to use for authentication.

Applications connecting to BigQuery via ODBC require a mechanism to authenticate and authorize access. Service accounts provide a secure way to do this. Service accounts are special Google accounts that do not require human interaction and are designed for applications and automated processes to access Google Cloud resources. Instead of using individual user credentials for each application, you create one service account per application/system and grant that account the appropriate permissions to access the relevant BigQuery datasets. This is more secure and manageable than user-based authentication for systems.

Why other options are incorrect:

Option A: Creating a new view over events does not resolve the issue of legacy SQL. The core issue is not that events needs to be overwritten, it is that a legacy view is being used.

Option B: Creating a new partitioned table is unnecessary for the given scenario. The question indicates that events_partitioned is suitable and is already partitioned.

Option E: While creating a Cloud IAM role is necessary for granting permissions, it is done in addition to creating a service account and not in place of it. The service account is what is associated with the role for authentication purposes.

Authoritative Links:

BigQuery standard SQL: https://cloud.google.com/bigquery/docs/reference/standard-sql/
BigQuery ODBC driver: https://cloud.google.com/bigquery/docs/reference/odbc-jdbc-drivers
Service accounts: https://cloud.google.com/iam/docs/service-accounts
IAM roles: https://cloud.google.com/iam/docs/understanding-roles

Question: 56 CertyIQ

You have enabled the free integration between Firebase Analytics and Google BigQuery. Firebase now automatically creates a new table daily in BigQuery in the format app_events_YYYYMMDD. You want to query all of the tables for the past 30 days in legacy SQL. What should you do?

A. Use the TABLE_DATE_RANGE function
B. Use the WHERE_PARTITIONTIME pseudo column
C. Use WHERE date BETWEEN YYYY-MM-DD AND YYYY-MM-DD
D. Use SELECT IF.(date >= YYYY-MM-DD AND date <= YYYY-MM-DD

Answer: A

Explanation:

The correct answer is **A. Use the TABLE_DATE_RANGE function**.

Here's why:

Firebase Analytics integration with BigQuery automatically creates daily tables named app_events_YYYYMMDD. To query a range of these tables in Legacy SQL, you need a mechanism to address these date-suffixed table names programmatically. The TABLE_DATE_RANGE function in Legacy SQL is designed specifically for querying multiple tables that follow a date-based naming convention. It allows you to specify a table name prefix and a start and end date, enabling BigQuery to access only the relevant tables within the given date range. It's crucial for efficiency as it avoids unnecessary scanning of older, irrelevant data.

Options B, C, and D are not appropriate in Legacy SQL for querying date-suffixed tables created by Firebase Analytics integration:

B. Use the WHERE _PARTITIONTIME pseudo column: _PARTITIONTIME is used for querying partitioned tables, and while daily tables can be partitioned, Firebase Analytics tables aren't automatically configured as such and using it would only work on such tables. The provided context states the name format app_events_YYYYMMDD, implying separate tables instead of a single partitioned table.

C. Use WHERE date BETWEEN YYYY-MM-DD AND YYYY-MM-DD: This option is for filtering data within a single table based on a 'date' column. It doesn't address querying multiple tables based on date-suffixed names. It assumes a date column within the data, which is not part of Firebase Analytics' automatically created tables.

D. Use SELECT IF.(date >= YYYY-MM-DD AND date <= YYYY-MM-DD: This is an incomplete and syntactically incorrect approach in Legacy SQL and addresses filtering data within a single table, not querying multiple date-suffixed tables. The IF function requires a THEN and ELSE clause. Also it doesn't handle multiple tables.

Therefore, TABLE_DATE_RANGE is the only function specifically designed for querying a range of tables with date-based naming patterns in Legacy SQL, making it the most suitable solution for this scenario.

Reference:

Legacy SQL TABLE_DATE_RANGE Function (although Legacy SQL is deprecated, it's relevant to the question's constraints)
Firebase Analytics BigQuery Export

Question: 57 CertyIQ

Your company is currently setting up data pipelines for their campaign. For all the Google Cloud Pub/Sub streaming data, one of the important business requirements is to be able to periodically identify the inputs and their timings during their campaign. Engineers have decided to use windowing and transformation in Google Cloud Dataflow for this purpose. However, when testing this feature, they find that the Cloud Dataflow job fails for the all streaming insert. What is the most likely cause of this problem?

A. They have not assigned the timestamp, which causes the job to fail

B. They have not set the triggers to accommodate the data coming in late, which causes the job to fail

C. They have not applied a global windowing function, which causes the job to fail when the pipeline is created

D. They have not applied a non-global windowing function, which causes the job to fail when the pipeline is created

Answer: D

Explanation:

The most likely cause of the Cloud Dataflow job failure for streaming inserts when using windowing and transformation is the absence of a non-global windowing function. Here's why:

Dataflow pipelines, especially those processing unbounded (streaming) data, require a mechanism to group incoming elements into finite chunks for processing. This is the fundamental role of windowing. Without a windowing function, Dataflow doesn't know how to group elements for aggregations or other window-specific operations.

Global windows, while technically a windowing strategy, aggregate all data into a single window. This isn't useful for periodic analysis of streaming data because it waits for the entire stream to finish before processing. Streaming jobs are inherently endless, making global windows impractical in most streaming scenarios intended for real-time or near-real-time insights.

Dataflow throws an error if you try to use transformations that require windowing (like aggregations, joins, etc.) without explicitly defining a windowing strategy that breaks the stream into smaller chunks. The absence of a non-global windowing function means the pipeline tries to operate on the unbounded stream as if it were a single unit, which is incompatible with common windowed transformations. The error often arises when the pipeline is created because the validity of the windowing strategy is checked early in the process.

Options A and B are related to common challenges in streaming data processing, such as handling out-of-order data (late data) and timestamp extraction. But those are generally considered after the data has been properly windowed. While important for data accuracy and completeness, those problems would not produce this specific pipeline creation failure.

For further research, consult the official Google Cloud Dataflow documentation on windowing:

Dataflow Windowing: https://cloud.google.com/dataflow/docs/concepts/windowing
Apache Beam Windowing: https://beam.apache.org/documentation/programming-guide/#windowing
(Dataflow is based on the Apache Beam model)

These resources explain the role of windowing, different windowing types (fixed, sliding, session), and how to apply them in your Dataflow pipelines. They also cover timestamp assignment and late data handling.

Question: 58 CertyIQ

You architect a system to analyze seismic data. Your extract, transform, and load (ETL) process runs as a series of MapReduce jobs on an Apache Hadoop cluster. The ETL process takes days to process a data set because some steps are computationally expensive. Then you discover that a sensor calibration step has been omitted. How should you change your ETL process to carry out sensor calibration systematically in the future?

A. Modify the transformMapReduce jobs to apply sensor calibration before they do anything else.

B. Introduce a new MapReduce job to apply sensor calibration to raw data, and ensure all other MapReduce jobs are chained after this.

C. Add sensor calibration data to the output of the ETL process, and document that all users need to apply sensor calibration themselves.

D. Develop an algorithm through simulation to predict variance of data output from the last MapReduce job based on calibration factors, and apply the correction to all data.

Answer: B

Explanation:

The most effective approach to incorporate the missing sensor calibration step into the existing MapReduce-based ETL process is option B: introduce a new MapReduce job dedicated to sensor calibration and chain all subsequent MapReduce jobs after it.

Here's why:

Data Integrity and Consistency: By applying calibration to the raw data before any other transformations, you ensure that all subsequent processing steps operate on corrected, reliable data. This is crucial for accurate analysis and prevents the propagation of errors.
Centralized Calibration: Having a dedicated calibration job provides a centralized and maintainable location for the calibration logic. Changes to the calibration process only need to be made in one place, minimizing the risk of inconsistencies.
Clear Data Provenance: Introducing a separate job makes it clear that the data has been calibrated, improving data provenance and auditability.
Modularity and Reusability: A dedicated calibration job can be easily reused for future data sets or even for other analysis pipelines that use the same sensor data.
Maintain Existing Structure: Introducing a new MapReduce job is minimally invasive and preserves the existing structure of the ETL process.

Option A, modifying existing transform jobs, would scatter the calibration logic across multiple jobs, making maintenance and updates difficult. Option C, adding calibration data to the output and relying on users to apply the calibration themselves, is unacceptable because it introduces the potential for human error and inconsistencies. Data analysis should aim to produce clean, corrected data. Option D, predicting variance and applying a correction to the final output, is complex, may introduce inaccuracies, and doesn't address the fundamental problem of working with uncalibrated data from the start.

Relevant Cloud Computing Concepts:

Data Pipelines: This scenario focuses on designing and managing a data pipeline for ETL processing.
MapReduce: Understanding how MapReduce distributes data processing across a cluster is essential.
Data Quality: Sensor calibration directly impacts the quality of the input data.
Data Governance: A well-defined and auditable ETL process is important for data governance.

Authoritative Links for Further Research:

Google Cloud Dataflow: A fully-managed, scalable, and reliable service for building data pipelines. Though the question uses MapReduce on Hadoop, Dataflow is a modern alternative on Google Cloud.
https://cloud.google.com/dataflow/docs
Apache Hadoop MapReduce: The fundamental concept behind the original question's architecture.
https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

Question: 59 CertyIQ

An online retailer has built their current application on Google App Engine. A new initiative at the company mandates that they extend their application to allow their customers to transact directly via the application. They need to manage their shopping transactions and analyze combined data from multiple datasets using a business intelligence (BI) tool. They want to use only a single database for this purpose. Which Google Cloud database should they choose?

A. BigQuery
B. Cloud SQL
C. Cloud BigTable
D. Cloud Datastore

Answer: B

Explanation:

The correct answer is **B. Cloud SQL**. Here's why:

The scenario requires a database that supports transactional workloads (managing shopping transactions) and analytical workloads (BI using combined datasets). While BigQuery (A) excels at analytics, it's not designed for handling real-time transactional data from an online shopping cart. Cloud Bigtable (C) is a NoSQL database best suited for high-volume, low-latency data storage and retrieval, but it lacks strong transactional capabilities and complex querying required for BI. Cloud Datastore (D), now known as Firestore in Datastore mode, is a NoSQL document database suitable for application data but not ideal for complex analytics or transactional integrity required for financial transactions.

Cloud SQL, on the other hand, is a fully managed relational database service that offers MySQL, PostgreSQL, and SQL Server options. Relational databases are inherently designed for ACID (Atomicity, Consistency, Isolation, Durability) compliance, which is crucial for maintaining the integrity of financial transactions. Furthermore, relational databases readily integrate with BI tools through standard SQL queries, making them suitable for analyzing combined datasets. While it might require some schema design and optimization, Cloud SQL offers the best balance between transactional needs of an e-commerce platform built using Google App Engine and the analytical requirements of the business intelligence initiative. It provides familiar SQL interface and supports the necessary features for both transactional and analytical processing needed by the retailer. This single database approach reduces complexity and cost compared to managing separate transactional and analytical databases. Cloud SQL's scalability also makes it a suitable long-term solution as the business grows.

Authoritative Links:

Cloud SQL: https://cloud.google.com/sql
ACID Properties: https://en.wikipedia.org/wiki/ACID
BigQuery: https://cloud.google.com/bigquery

Question: 60 CertyIQ

You launched a new gaming app almost three years ago. You have been uploading log files from the previous day to a separate Google BigQuery table with the table name format LOGS_yyyymmdd. You have been using table wildcard functions to generate daily and monthly reports for all time ranges. Recently, you discovered that some queries that cover long date ranges are exceeding the limit of 1,000 tables and failing. How can you resolve this issue?

A. Convert all daily log tables into date-partitioned tables
B. Convert the sharded tables into a single partitioned table
C. Enable query caching so you can cache data from previous months
D. Create separate views to cover each month, and query from these views

Answer: B

Explanation:

The correct answer is **B. Convert the sharded tables into a single partitioned table.**

Here's a detailed justification:

The primary problem is the limit of 1,000 tables that can be scanned within a single BigQuery query when using table wildcard functions. Currently, the log data is sharded into many small tables (LOGS_yyyymmdd). Wildcard tables are useful when dealing with lots of data, but querying over many such tables will easily hit

BigQuery's table limit.

Partitioning, on the other hand, addresses this limit directly. By converting all the daily sharded tables into a single date-partitioned table, you consolidate the data into one logical table. BigQuery then leverages the partitioning scheme (in this case, date) to only scan the relevant partitions that match the query's date range, dramatically reducing the number of tables scanned and circumventing the 1,000-table limit. This allows you to efficiently query large date ranges.

Option A is incorrect because converting all daily log tables into date-partitioned tables wouldn't solve the table count issue if you still maintained a multitude of separate partitioned tables. The benefit is gained by having a single partitioned table.

Option C, enabling query caching, will only help with repeated queries of the exact same data. It won't reduce the number of tables scanned in the initial query. Cache only helps with a repeated identical query. If the data changes or if even one parameter changes, cache will not be used. It does not address the underlying limit on table scanning.

Option D, creating separate views for each month, only adds another layer of abstraction and does not solve the table scanning issue. Each view still effectively queries potentially many tables within that month. This maintains the same limitation on table scanning that the current wildcard table setup suffers from. It also would be extremely tedious to maintain.

Therefore, converting the sharded tables into a single date-partitioned table is the optimal solution because it directly addresses the 1,000-table limit while maintaining efficient querying capabilities.

For further research:

BigQuery Partitioned Tables: https://cloud.google.com/bigquery/docs/partitioned-tables
BigQuery Table Wildcards: https://cloud.google.com/bigquery/docs/querying-wildcard-tables (Note the limitations mentioned regarding table limits)
BigQuery Query Caching: https://cloud.google.com/bigquery/docs/cached-queries

Question: 61 CertyIQ

Your analytics team wants to build a simple statistical model to determine which customers are most likely to work with your company again, based on a few different metrics. They want to run the model on Apache Spark, using data housed in Google Cloud Storage, and you have recommended using Google Cloud
Dataproc to execute this job. Testing has shown that this workload can run in approximately 30 minutes on a 15-node cluster, outputting the results into Google
BigQuery. The plan is to run this workload weekly. How should you optimize the cluster for cost?

A. Migrate the workload to Google Cloud Dataflow
B. Use pre-emptible virtual machines (VMs) for the cluster
C. Use a higher-memory node so that the job runs faster
D. Use SSDs on the worker nodes so that the job can run faster

Answer: B

Explanation:

The most cost-effective approach for a weekly, predictable Dataproc job is to utilize preemptible VMs (option B). Preemptible VMs offer a significant discount compared to regular VMs, making them ideal for fault-tolerant workloads like Spark jobs. Since the job runs weekly, a preemption event does not drastically disrupt the process as the job can be restarted. The 30-minute runtime on a 15-node cluster suggests a manageable duration for recovery if preemption occurs. Option A, migrating to Dataflow, introduces a different paradigm that might require a significant rewrite of existing Spark code and doesn't directly address cost optimization

within a Dataproc context. Options C and D, using higher-memory or SSDs, will likely increase the per-node cost, offsetting the potential runtime gains; this doesn't tackle the root of the cost optimization goal. Preemptible VMs, on the other hand, directly reduce the hourly cost. The nature of the workload makes it suitable for handling potential interruptions. Further research on preemptible VMs for Dataproc can be found here: https://cloud.google.com/dataproc/docs/concepts/preemptible-vms

Question: 62 CertyIQ

Your company receives both batch- and stream-based event data. You want to process the data using Google Cloud Dataflow over a predictable time period.
However, you realize that in some instances data can arrive late or out of order. How should you design your Cloud Dataflow pipeline to handle data that is late or out of order?

A. Set a single global window to capture all the data.

B. Set sliding windows to capture all the lagged data.

C. Use watermarks and timestamps to capture the lagged data.

D. Ensure every datasource type (stream or batch) has a timestamp, and use the timestamps to define the logic for lagged data.

Answer: C

Explanation:

The correct answer is C: Use watermarks and timestamps to capture the lagged data. This approach is fundamental to handling late or out-of-order data in streaming data processing with Dataflow.

Dataflow, unlike traditional batch processing, operates on unbounded data streams. Therefore, it needs mechanisms to determine when it has processed all data for a specific window or period, even when data arrives late. Watermarks provide this mechanism. A watermark is a system's understanding of when all data up to a certain point in time is expected to have arrived. It's an approximation, not a guarantee. Data arriving after the watermark for a window has passed is considered late data.

Timestamps are essential because Dataflow uses them to assign data to the correct processing window and to calculate the watermark. Each event in the stream must have an associated timestamp. If events do not have a timestamp, it's necessary to create or assign one during pipeline processing. By using timestamps and watermarks, Dataflow can handle out-of-order data efficiently by correctly grouping them into corresponding windows based on their timestamps, even if they arrive later than anticipated. This approach ensures data is processed correctly, even with inherent delays or irregularities in the data stream.

Option A (single global window) would capture all data but wouldn't allow for time-based analysis. Option B (sliding windows) is useful for continuous analytics over shifting time ranges, but does not inherently address the handling of late data. Option D mentions using timestamps, which is important but doesn't address how these timestamps relate to late data processing, and doesn't mention watermarks which is a core concept for handling late data. Option C incorporates both timestamps and watermarks which are critical when dealing with both batch and stream-based data sources to handle late-arriving data properly.

Therefore, using watermarks and timestamps allows Dataflow to manage late data correctly by knowing when to consider a window complete even with out-of-order and late data arrivals.

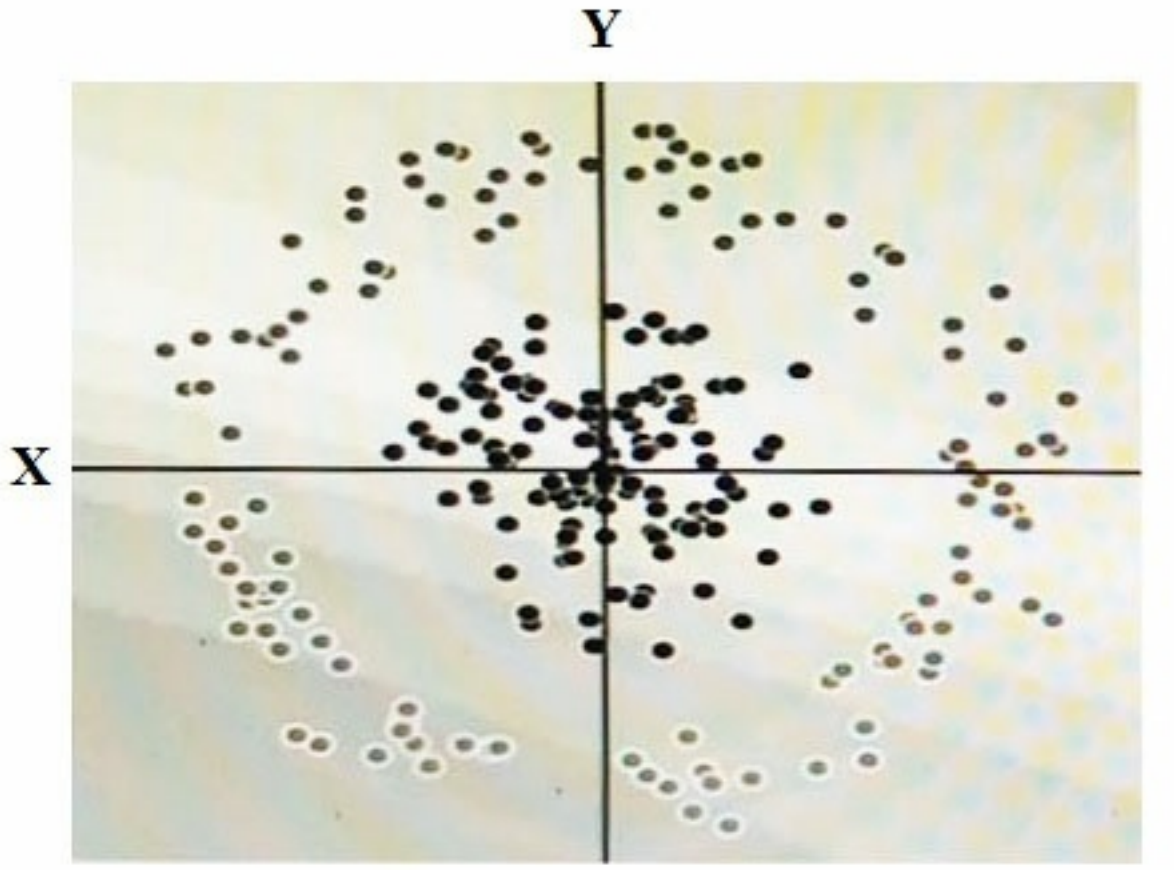
Authoritative Links:

Apache Beam Programming Guide - Watermarks and Late Data: This is an excellent resource detailing how watermarks function in Beam (Dataflow's programming model) and the concept of late data.
Google Cloud Dataflow documentation - Time and Windows: This is Google's Dataflow specific documentation

that discusses time and windowing, essential to handling data ordering and late data.

Question: 63 CertyIQ

You have some data, which is shown in the graphic below. The two dimensions are X and Y, and the shade of each dot represents what class it is. You want to classify this data accurately using a linear algorithm. To do this you need to add a synthetic feature. What should the value of that feature be?



A. X2+Y2
B. X2
C. Y2
D. cos(X)

Answer: A

Explanation:

For fitting a linear classifier when the data is in a circle use A.

Question: 64 CertyIQ

You are integrating one of your internal IT applications and Google BigQuery, so users can query BigQuery from the application's interface. You do not want individual users to authenticate to BigQuery and you do not want to give them access to the dataset. You need to securely access BigQuery from your IT application. What should you do?

A. Create groups for your users and give those groups access to the dataset
B. Integrate with a single sign-on (SSO) platform, and pass each user's credentials along with the query request
C. Create a service account and grant dataset access to that account. Use the service account's private key to

access the dataset

D. Create a dummy user and grant dataset access to that user. Store the username and password for that user in a file on the files system, and use those credentials to access the BigQuery dataset

Answer: C

Explanation:

The correct approach is to utilize a service account (Option C). Service accounts are non-human accounts specifically designed for applications and services to interact with Google Cloud resources. This addresses the requirement of not having individual users authenticate directly to BigQuery. By granting the service account access to the BigQuery dataset, the application can access the data without exposing user credentials or giving them direct data access. The application can use the service account's private key to authenticate and authorize API calls to BigQuery, as opposed to storing sensitive credentials like usernames and passwords. This method ensures that only the application, through the service account, has access, which is a best practice for security. Sharing credentials across multiple applications or using dummy accounts, as suggested in options B and D, introduces significant security risks. Using groups for user-based access, as in option A, defeats the requirement of not giving individual users access to the dataset. Therefore, the service account method, as described in option C, offers the most secure and appropriate solution for accessing BigQuery from an application without direct user authentication.

Authoritative links:

Service accounts: Official Google Cloud documentation on service accounts.
Best practices for using service accounts: Guidance on using service accounts securely and effectively.
BigQuery Authentication: Documentation on authenticating to BigQuery, covering service accounts and other methods.

Question: 65 CertyIQ

You are building a data pipeline on Google Cloud. You need to prepare data using a casual method for a machine-learning process. You want to support a logistic regression model. You also need to monitor and adjust for null values, which must remain real-valued and cannot be removed. What should you do?

A. Use Cloud Dataprep to find null values in sample source data. Convert all nulls to 'none' using a Cloud Dataproc job.

B. Use Cloud Dataprep to find null values in sample source data. Convert all nulls to 0 using a Cloud Dataprep job.

C. Use Cloud Dataflow to find null values in sample source data. Convert all nulls to 'none' using a Cloud Dataprep job.

D. Use Cloud Dataflow to find null values in sample source data. Convert all nulls to 0 using a custom script.

Answer: B

Explanation:

The correct answer is B. Here's why:

The problem requires handling null values in a dataset intended for a logistic regression model. Logistic regression, being a numerical algorithm, cannot directly process non-numeric values like "none". Therefore, converting null values to a string, like in options A and C, is not suitable.

Option D suggests using a custom script within Cloud Dataflow. While this approach is technically viable, it introduces unnecessary complexity compared to the simplicity offered by Cloud Dataprep, which is explicitly designed for casual data preparation tasks as the question requires. Dataprep's visual interface allows for

simpler and faster handling of common data transformations.

Option B leverages Cloud Dataprep's strength in data profiling to identify null values. Crucially, it also uses Dataprep's capability to easily convert these nulls to '0', fulfilling the need to retain real-valued data as required by the question. Substituting null values with 0 is a common, legitimate approach when working with numerical data for machine learning models, assuming that zero is a sensible placeholder in the context of the data. This makes the data immediately suitable for logistic regression without requiring code development.

Therefore, option B effectively addresses the problem's requirements, balancing efficacy with the desired ease of use in the preparation process.

Authoritative Links:

Cloud Dataprep Documentation: https://cloud.google.com/dataprep/docs - Provides comprehensive information on Dataprep's features.
Logistic Regression: A quick search for "logistic regression data preprocessing null values" will show the common practice of replacing nulls with reasonable numerical values.
Cloud Dataflow Documentation: https://cloud.google.com/dataflow/docs - Can be used as a reference for comparison but is not the optimal solution for the scenario presented.

Question: 66 CertyIQ

You set up a streaming data insert into a Redis cluster via a Kafka cluster. Both clusters are running on Compute Engine instances. You need to encrypt data at rest with encryption keys that you can create, rotate, and destroy as needed. What should you do?

A. Create a dedicated service account, and use encryption at rest to reference your data stored in your Compute Engine cluster instances as part of your API service calls.

B. Create encryption keys in Cloud Key Management Service. Use those keys to encrypt your data in all of the Compute Engine cluster instances.

C. Create encryption keys locally. Upload your encryption keys to Cloud Key Management Service. Use those keys to encrypt your data in all of the Compute Engine cluster instances.

D. Create encryption keys in Cloud Key Management Service. Reference those keys in your API service calls when accessing the data in your Compute Engine cluster instances.

Answer: B

Explanation:

The correct answer is **B. Create encryption keys in Cloud Key Management Service. Use those keys to encrypt your data in all of the Compute Engine cluster instances.**

Here's a breakdown of why:

The primary requirement is to encrypt data at rest in the Redis and Kafka clusters, which are hosted on Compute Engine instances. We also need full control over key creation, rotation, and destruction. Google Cloud Key Management Service (Cloud KMS) is the ideal service for this. Cloud KMS allows you to manage encryption keys centrally, enabling you to create, rotate, and destroy keys as required. Option B directly uses Cloud KMS to generate the necessary keys. These keys are then used to encrypt data stored on the Compute Engine instances, specifically the Redis and Kafka persistent storage. The encryption takes place at the storage layer of these instances.

Option A is incorrect because it focuses on encryption during API service calls. While important, this addresses encryption in transit, not at rest. Additionally, relying on the instances' default encryption is not the same as providing user-managed keys which are required in this scenario. Option C is not recommended because creating keys locally and uploading them to Cloud KMS introduces an unnecessary security risk and

complexity. Local key management can be cumbersome and vulnerable to compromise. Cloud KMS is designed to securely manage key generation from the start. Finally, option D incorrectly suggests encryption is handled through API calls. The goal here is data at rest encryption, and it should not be performed via API calls, but at the storage level of the Compute Engines.

By using Cloud KMS and applying encryption at the storage layer of the Compute Engines, we satisfy the need for user-controlled encryption at rest and gain the flexibility to manage keys securely and efficiently using a central service.

Relevant Cloud Computing Concepts:

Encryption at Rest: Protecting data when it's stored on a persistent medium (like a hard drive).
Cloud Key Management Service (Cloud KMS): A managed service for creating, using, rotating, and destroying cryptographic keys.
Compute Engine: Google Cloud's Infrastructure-as-a-Service, which provides virtual machines.

Authoritative Links for further research:

Cloud KMS Documentation
Encryption at rest in Compute Engine

Question: 67 CertyIQ

You are developing an application that uses a recommendation engine on Google Cloud. Your solution should display new videos to customers based on past views. Your solution needs to generate labels for the entities in videos that the customer has viewed. Your design must be able to provide very fast filtering suggestions based on data from other customer preferences on several TB of data. What should you do?

A. Build and train a complex classification model with Spark MLlib to generate labels and filter the results. Deploy the models using Cloud Dataproc. Call the model from your application.

B. Build and train a classification model with Spark MLlib to generate labels. Build and train a second classification model with Spark MLlib to filter results to match customer preferences. Deploy the models using Cloud Dataproc. Call the models from your application.

C. Build an application that calls the Cloud Video Intelligence API to generate labels. Store data in Cloud Bigtable, and filter the predicted labels to match the user's viewing history to generate preferences.

D. Build an application that calls the Cloud Video Intelligence API to generate labels. Store data in Cloud SQL, and join and filter the predicted labels to match the user's viewing history to generate preferences.

Answer: C

Explanation:

Option C is the most appropriate solution for building a recommendation engine that efficiently handles large datasets and provides fast filtering. Cloud Video Intelligence API readily extracts labels from video content, eliminating the need for custom model training for label generation, which is time-consuming and requires significant expertise (as suggested in options A and B). This accelerates development. Cloud Bigtable, a NoSQL wide-column database, is designed to handle massive amounts of data with low latency, making it ideal for storing and querying user viewing history and video labels at scale, crucial for fast filtering. Unlike Cloud SQL (as in option D) which is a relational database, Bigtable is specifically optimized for the high read/write throughput requirements of this recommendation use case. Bigtable's ability to filter based on column values directly matches the requirement to filter based on user preferences. Storing and joining the data in SQL is possible, but less performant when dealing with a TB of data and requiring low latency filtering. Option A and B propose using Spark MLlib and Cloud Dataproc. While powerful for batch processing and model training, they are not suitable for real-time label generation or filtering. Also, training and deploying multiple models adds unnecessary complexity to the solution. In essence, Cloud Video Intelligence API

provides the labelling capabilities, while Bigtable provides scalability and the low latency query mechanism for the vast amount of data.

Authoritative links:

Cloud Video Intelligence API: https://cloud.google.com/video-intelligence/docs
Cloud Bigtable: https://cloud.google.com/bigtable/docs
Comparison of Cloud SQL and Cloud Bigtable: https://cloud.google.com/learn/what-is-cloud-sql-vs-cloud-bigtable

Question: 68 CertyIQ

You are selecting services to write and transform JSON messages from Cloud Pub/Sub to BigQuery for a data pipeline on Google Cloud. You want to minimize service costs. You also want to monitor and accommodate input data volume that will vary in size with minimal manual intervention. What should you do?

A. Use Cloud Dataproc to run your transformations. Monitor CPU utilization for the cluster. Resize the number of worker nodes in your cluster via the command line.

B. Use Cloud Dataproc to run your transformations. Use the diagnose command to generate an operational output archive. Locate the bottleneck and adjust cluster resources.

C. Use Cloud Dataflow to run your transformations. Monitor the job system lag with Stackdriver. Use the default autoscaling setting for worker instances.

D. Use Cloud Dataflow to run your transformations. Monitor the total execution time for a sampling of jobs. Configure the job to use non-default Compute Engine machine types when needed.

Answer: C

Explanation:

Option C, using Cloud Dataflow, is the most suitable solution for transforming JSON messages from Cloud Pub/Sub to BigQuery while minimizing costs and adapting to varying data volumes. Cloud Dataflow is a fully managed, serverless data processing service designed for both batch and stream data pipelines. Its autoscaling feature, a key aspect of the default setting mentioned, automatically adjusts the number of worker instances based on the workload, eliminating the need for manual intervention and optimizing resource utilization. This reduces operational overhead and minimizes costs during periods of low data volume. Monitoring "system lag" in Stackdriver provides critical insight into the pipeline's health, enabling you to identify and address bottlenecks. Options A and B, using Cloud Dataproc, are less suitable for this specific use case. Dataproc is primarily designed for running Hadoop and Spark clusters, requiring manual cluster management and scaling, leading to increased operational burden and costs. While Dataproc offers cluster resizing, it involves manual intervention, contradicting the requirement for minimal manual effort. Option D while using Cloud Dataflow, focuses on execution time sampling which is not the most effective way to optimize a data pipeline. Autoscaling is Dataflow's strength and should be used. In summary, Dataflow's serverless nature, autoscaling capabilities, and integration with Stackdriver monitoring make it the ideal choice for a cost-effective and adaptable data pipeline, addressing the prompt’s requirements effectively.

Authoritative Links:

Cloud Dataflow: https://cloud.google.com/dataflow
Dataflow Autoscaling: https://cloud.google.com/dataflow/docs/guides/autoscaling
Cloud Pub/Sub: https://cloud.google.com/pubsub
BigQuery: https://cloud.google.com/bigquery
Stackdriver (Google Cloud Observability): https://cloud.google.com/stackdriver
Cloud Dataproc: https://cloud.google.com/dataproc

Question: 69 CertyIQ

Your infrastructure includes a set of YouTube channels. You have been tasked with creating a process for sending the YouTube channel data to Google Cloud for analysis. You want to design a solution that allows your world-wide marketing teams to perform ANSI SQL and other types of analysis on up-to-date YouTube channels log data. How should you set up the log data transfer into Google Cloud?

A. Use Storage Transfer Service to transfer the offsite backup files to a Cloud Storage Multi-Regional storage bucket as a final destination.

B. Use Storage Transfer Service to transfer the offsite backup files to a Cloud Storage Regional bucket as a final destination.

C. Use BigQuery Data Transfer Service to transfer the offsite backup files to a Cloud Storage Multi-Regional storage bucket as a final destination.

D. Use BigQuery Data Transfer Service to transfer the offsite backup files to a Cloud Storage Regional storage bucket as a final destination.

Answer: A

Explanation:

The correct answer is **A. Use Storage Transfer Service to transfer the offsite backup files to a Cloud Storage Multi-Regional storage bucket as a final destination.** Here's why:

The core requirement is to transfer YouTube channel data to Google Cloud for analysis, specifically enabling SQL queries and other analyses by marketing teams. The question mentions "offsite backup files," implying the data is already stored as files outside Google Cloud.

Storage Transfer Service is the ideal choice for moving large volumes of data from various sources, including other cloud storage providers or on-premises locations, into Google Cloud Storage. It's efficient and cost-effective for large-scale file transfers. BigQuery Data Transfer Service is designed for transferring data from SaaS applications (like Google Ads, YouTube Analytics, etc.) directly into BigQuery, not for moving existing backup files. Since the question describes backup files, Storage Transfer Service is more appropriate.

Multi-Regional Cloud Storage buckets offer higher availability and lower latency access globally, which is advantageous for teams worldwide needing to access and analyze the data. Regional buckets are tied to a single region, potentially creating performance issues for geographically dispersed teams. Hence, a multi-regional bucket aligns better with the stated requirement.

Therefore, Storage Transfer Service provides the necessary functionality for transferring the existing backup files, and a Multi-Regional Cloud Storage bucket ensures global availability for the marketing teams. Options C and D are incorrect because they suggest using BigQuery Data Transfer Service, which is unsuitable for transferring existing files. Option B is incorrect because regional storage would not align well with the requirements for a global marketing team.

Authoritative Links:

Storage Transfer Service Documentation: https://cloud.google.com/storage-transfer-service/docs
Cloud Storage Buckets Overview: https://cloud.google.com/storage/docs/buckets
BigQuery Data Transfer Service Documentation: https://cloud.google.com/bigquery/docs/transfer-service

Question: 70 CertyIQ

You are designing storage for very large text files for a data pipeline on Google Cloud. You want to support ANSI SQL queries. You also want to support compression and parallel load from the input locations using Google recommended practices. What should you do?

A. Transform text files to compressed Avro using Cloud Dataflow. Use BigQuery for storage and query.

B. Transform text files to compressed Avro using Cloud Dataflow. Use Cloud Storage and BigQuery permanent linked tables for query.

C. Compress text files to gzip using the Grid Computing Tools. Use BigQuery for storage and query.

D. Compress text files to gzip using the Grid Computing Tools. Use Cloud Storage, and then import into Cloud Bigtable for query.

Answer: B

Explanation:

Option B is the most suitable choice because it leverages Google Cloud's recommended practices for handling large text files while enabling ANSI SQL queries. Cloud Dataflow is a robust service designed for data transformation, making it ideal for converting the raw text files into compressed Avro format. Avro is a row-based data serialization format that supports schema evolution and efficient compression, optimizing both storage and query performance. Storing the compressed Avro files in Cloud Storage provides cost-effective and scalable storage. BigQuery permanent linked tables then allow querying the data directly from Cloud Storage using ANSI SQL without the need to import it. This approach eliminates redundant storage and simplifies the overall architecture. Using Cloud Dataflow and BigQuery together aligns with Google's modern data processing best practices. Options A and C involve directly loading or using Gzip files with BigQuery, which while feasible is less optimized for query performance compared to using Avro files. Option D is unsuitable as Bigtable is not designed for analytical SQL queries, and it does not work well with large text files directly.

Authoritative Links for further research:

Cloud Dataflow: https://cloud.google.com/dataflow/docs
Avro format: https://avro.apache.org/
Cloud Storage: https://cloud.google.com/storage/docs
BigQuery Permanent Linked Tables: https://cloud.google.com/bigquery/docs/external-data-cloud-storage
BigQuery Best Practices: https://cloud.google.com/bigquery/docs/best-practices-performance

Question: 71 CertyIQ

You are developing an application on Google Cloud that will automatically generate subject labels for users' blog posts. You are under competitive pressure to add this feature quickly, and you have no additional developer resources. No one on your team has experience with machine learning. What should you do?

A. Call the Cloud Natural Language API from your application. Process the generated Entity Analysis as labels.

B. Call the Cloud Natural Language API from your application. Process the generated Sentiment Analysis as labels.

C. Build and train a text classification model using TensorFlow. Deploy the model using Cloud Machine Learning Engine. Call the model from your application and process the results as labels.

D. Build and train a text classification model using TensorFlow. Deploy the model using a Kubernetes Engine cluster. Call the model from your application and process the results as labels.

Answer: A

Explanation:

The correct answer is **A. Call the Cloud Natural Language API from your application. Process the generated Entity Analysis as labels.**

Here's why:

The scenario emphasizes speed and limited resources, especially the lack of ML expertise. Option A leverages

the pre-trained Cloud Natural Language API, a managed service. This eliminates the need to build, train, and deploy a custom ML model, which requires considerable time, effort, and expertise (as suggested in Options C and D). Entity analysis, a core feature of the Natural Language API, identifies key concepts (entities) within text, which directly translates to appropriate subject labels for blog posts. Sentiment analysis (Option B) focuses on the emotional tone of text, not relevant to subject labeling. Options C and D are not feasible due to the lack of internal machine learning knowledge, and using Kubernetes for option D further increases complexity. Utilizing an existing API like Cloud Natural Language allows for rapid implementation and avoids the overhead of managing infrastructure or model training, meeting the pressing need to add the feature quickly. Therefore, Cloud Natural Language API's Entity Analysis offers the most pragmatic and efficient approach.

Further research on Cloud Natural Language API and its capabilities can be found here:

Google Cloud Natural Language API Documentation: https://cloud.google.com/natural-language/docs
Entity Analysis documentation: https://cloud.google.com/natural-language/docs/analyzing-entities

Question: 72 CertyIQ

You are designing storage for 20 TB of text files as part of deploying a data pipeline on Google Cloud. Your input data is in CSV format. You want to minimize the cost of querying aggregate values for multiple users who will query the data in Cloud Storage with multiple engines. Which storage service and schema design should you use?

A. Use Cloud Bigtable for storage. Install the HBase shell on a Compute Engine instance to query the Cloud Bigtable data.
B. Use Cloud Bigtable for storage. Link as permanent tables in BigQuery for query.
C. Use Cloud Storage for storage. Link as permanent tables in BigQuery for query.
D. Use Cloud Storage for storage. Link as temporary tables in BigQuery for query.

Answer: C

Explanation:

The correct answer is C, using Cloud Storage for storage and linking it as permanent tables in BigQuery for querying.

Here's a detailed justification:

Cost Optimization: Cloud Storage is significantly cheaper than Cloud Bigtable for storing large volumes of data, especially text files like CSVs. Cloud Storage is designed for cost-effective object storage, making it ideal for data at rest. https://cloud.google.com/storage/pricing

Querying Aggregate Values: BigQuery is a fully managed, serverless data warehouse optimized for analyzing large datasets and performing aggregate queries. It's designed to handle complex analytical workloads efficiently. Cloud Bigtable is designed for low-latency reads/writes of large amounts of data and is not ideal for analytical queries. https://cloud.google.com/bigquery/docs

Multiple Users and Query Engines: BigQuery supports concurrent queries from multiple users and integrates seamlessly with various query engines (e.g., SQL, data science tools via APIs). This makes it ideal for serving multiple users with different analytical needs.

CSV Format: BigQuery can directly ingest and query CSV files stored in Cloud Storage. This eliminates the need for complex data transformations or schema migrations.

Permanent vs. Temporary Tables: Creating permanent tables in BigQuery provides a persistent, queryable dataset. Using temporary tables would require reloading the data each time, increasing cost and effort.

Permanent tables also enable better access control and data governance.

Cloud Bigtable Inappropriateness: Cloud Bigtable is a NoSQL database suitable for applications requiring very low latency reads and writes, such as real-time personalization or IoT data. It's not optimized for batch analytical queries or the CSV format. Using Cloud Bigtable and then querying it through BigQuery would add unnecessary complexity and cost.

HBase Shell Inefficiency: While HBase is an option for querying Cloud Bigtable, it's not suitable for ad-hoc aggregate queries from multiple users due to its complex setup and the nature of performing such analytical tasks on a NoSQL environment.

In summary, Cloud Storage offers the most cost-effective storage for 20 TB of text files. Linking this data as permanent tables in BigQuery allows multiple users to perform efficient aggregate queries with various query engines, fulfilling all requirements in the question. Bigtable would be overkill and more expensive for this scenario.

Question: 73 CertyIQ

You are designing storage for two relational tables that are part of a 10-TB database on Google Cloud. You want to support transactions that scale horizontally.
You also want to optimize data for range queries on non-key columns. What should you do?

A. Use Cloud SQL for storage. Add secondary indexes to support query patterns.
B. Use Cloud SQL for storage. Use Cloud Dataflow to transform data to support query patterns.
C. Use Cloud Spanner for storage. Add secondary indexes to support query patterns.
D. Use Cloud Spanner for storage. Use Cloud Dataflow to transform data to support query patterns.

Answer: C

Explanation:

The correct answer is C: "Use Cloud Spanner for storage. Add secondary indexes to support query patterns."

Here's why:

Horizontal Scalability and Transactions: Cloud Spanner is specifically designed to provide horizontally scalable, strongly consistent transactions. This is critical for a 10-TB database needing to scale without compromising data integrity. Cloud SQL, while offering transactions, generally faces more limitations in horizontal scaling for very large datasets compared to Spanner.

Range Queries on Non-Key Columns: Spanner supports secondary indexes, which allow you to optimize query performance on columns that are not part of the primary key. By creating secondary indexes on the non-key columns used in your range queries, you can significantly speed up those queries without needing to transform the data.

Why other options are not ideal:

Option A (Cloud SQL with secondary indexes): Cloud SQL's scalability limitations for a 10-TB database make it less suitable than Spanner. While Cloud SQL offers indexes, it may not scale as efficiently for large datasets and transaction volumes.
Option B (Cloud SQL with Cloud Dataflow): Using Cloud Dataflow to transform data to optimize for query patterns is more complex and costly than using Spanner with secondary indexes. It introduces an additional data processing step and potential latency. Dataflow is more relevant for ETL or complex transformations, not for optimizing basic query patterns on relational data.
Option D (Cloud Spanner with Cloud Dataflow): While Cloud Dataflow can be used with Spanner, it is not the

optimal solution for range queries on non-key columns. Spanner's secondary indexes offer a more direct and efficient solution for these types of queries. Using Dataflow would involve unnecessarily complex data transformation.

In summary, Cloud Spanner offers the necessary horizontal scalability and transaction support for a 10-TB database, and secondary indexes provide the most straightforward way to optimize range queries on non-key columns without requiring complex data transformations.

Authoritative Links:

Cloud Spanner Overview: https://cloud.google.com/spanner/docs/overview
Cloud Spanner Secondary Indexes: https://cloud.google.com/spanner/docs/secondary-indexes
Cloud SQL Overview: https://cloud.google.com/sql/docs/mysql/introduction
Cloud Dataflow Overview: https://cloud.google.com/dataflow/docs/concepts/what-is-dataflow

Question: 74 CertyIQ

Your financial services company is moving to cloud technology and wants to store 50 TB of financial time-series data in the cloud. This data is updated frequently and new data will be streaming in all the time. Your company also wants to move their existing Apache Hadoop jobs to the cloud to get insights into this data.
Which product should they use to store the data?

A. Cloud Bigtable
B. Google BigQuery
C. Google Cloud Storage
D. Google Cloud Datastore

Answer: A

Explanation:

The most appropriate choice for storing 50 TB of financial time-series data with frequent updates and continuous streaming, while also needing compatibility with Apache Hadoop jobs, is **Cloud Bigtable (A)**. Cloud Bigtable is a fully managed, scalable NoSQL database service designed for low-latency read and write access to large datasets. Its columnar storage model excels in handling time-series data, allowing for efficient storage and retrieval of data points across time. Furthermore, it's specifically built for high-throughput, which aligns with the requirement for continuous streaming. Crucially, Bigtable integrates well with the Hadoop ecosystem through its HBase compatibility, making it easy to migrate existing Hadoop jobs for data analysis. While BigQuery (B) is excellent for analytics, it's less suitable for transactional workloads with constant updates and streaming data. Cloud Storage (C) provides object storage, which isn't ideal for data that requires frequent updates, and Datastore (D) is better suited for application data, not large-scale time series data. Therefore, Bigtable's combination of high-throughput writes, low-latency reads, and Hadoop compatibility makes it the optimal solution for the company's needs.

Further research:

Cloud Bigtable Overview: https://cloud.google.com/bigtable/docs/overview
Cloud Bigtable for Time Series Data: https://cloud.google.com/bigtable/docs/time-series
HBase Compatibility in Bigtable: https://cloud.google.com/bigtable/docs/hbase/

Question: 75 CertyIQ

An organization maintains a Google BigQuery dataset that contains tables with user-level data. They want to expose aggregates of this data to other Google

Cloud projects, while still controlling access to the user-level data. Additionally, they need to minimize their overall storage cost and ensure the analysis cost for other projects is assigned to those projects. What should they do?

A. Create and share an authorized view that provides the aggregate results.

B. Create and share a new dataset and view that provides the aggregate results.

C. Create and share a new dataset and table that contains the aggregate results.

D. Create dataViewer Identity and Access Management (IAM) roles on the dataset to enable sharing.

Answer: A

Explanation:

The correct answer is A. Creating and sharing an authorized view is the most efficient and cost-effective method to achieve the described requirements. Authorized views in BigQuery allow you to share query results (aggregates in this case) without directly exposing the underlying user-level data. They operate like virtual tables; queries against the view are executed using the permissions of the view itself, not the user querying it. This enables secure sharing across projects. This approach minimizes storage costs because no new tables with aggregated data are created, hence no data duplication is involved. Since authorized views inherit billing from the querying project, query costs are correctly attributed to the consuming project, as opposed to the project where original data lies. Options B and C involve duplicating the data into new datasets/tables which can significantly increase storage costs and data management overhead. Option D would expose the entire dataset, including user-level data, which contradicts the core requirement of securing raw data.

Here are some authoritative links for further research:

BigQuery Authorized Views: https://cloud.google.com/bigquery/docs/share-access-views
BigQuery Cost Control: https://cloud.google.com/bigquery/docs/cost-control-guide
IAM for BigQuery: https://cloud.google.com/bigquery/docs/access-control

Question: 76 CertyIQ

Government regulations in your industry mandate that you have to maintain an auditable record of access to certain types of data. Assuming that all expiring logs will be archived correctly, where should you store data that is subject to that mandate?

A. Encrypted on Cloud Storage with user-supplied encryption keys. A separate decryption key will be given to each authorized user.

B. In a BigQuery dataset that is viewable only by authorized personnel, with the Data Access log used to provide the auditability.

C. In Cloud SQL, with separate database user names to each user. The Cloud SQL Admin activity logs will be used to provide the auditability.

D. In a bucket on Cloud Storage that is accessible only by an AppEngine service that collects user information and logs the access before providing a link to the bucket.

Answer: B

Explanation:

The most suitable option for storing auditable data, as mandated by regulations, is **B. In a BigQuery dataset that is viewable only by authorized personnel, with the Data Access log used to provide the auditability.**

Here's why: BigQuery is designed for analytical workloads and provides robust access control and audit logging features crucial for compliance. By restricting access to the dataset to only authorized personnel, you ensure data confidentiality. Furthermore, BigQuery's Data Access logs track who accessed what data and when, fulfilling the auditable record requirement. These logs are detailed, readily available, and designed

specifically for compliance and security auditing. Options A, C, and D fall short. While A offers encryption, it lacks the built-in audit logging necessary for compliance. Option C relies on Cloud SQL logs, which are less granular and geared towards administrative activities rather than data access auditing. Option D introduces unnecessary complexity by requiring an intermediate App Engine service, and the logging might not be as comprehensive as BigQuery's Data Access logs. BigQuery offers granular permission management using Identity and Access Management (IAM), ensuring only intended users can access specific parts of the data, enhancing control and accountability. This approach aligns with best practices for data governance and compliance in regulated industries.

Authoritative Links for further research:

BigQuery Security Overview: https://cloud.google.com/bigquery/docs/security-overview
BigQuery Audit Logging: https://cloud.google.com/bigquery/docs/audit-logs
Cloud IAM: https://cloud.google.com/iam/docs
Data Access Logs: https://cloud.google.com/logging/docs/audit/data-access

Question: 77 CertyIQ

Your neural network model is taking days to train. You want to increase the training speed. What can you do?

A. Subsample your test dataset.
B. Subsample your training dataset.
C. Increase the number of input features to your model.
D. Increase the number of layers in your neural network.

Answer: B

Explanation:

The correct answer is **B. Subsample your training dataset.** Here's a detailed justification:

Training a neural network, especially a large one, is computationally intensive. A major factor affecting training time is the size of the training dataset. A larger dataset means more data points to process during each training iteration (epoch), leading to longer training times. By subsampling, or reducing the size of the training dataset, we reduce the amount of computation needed per epoch. This results in faster training, although it might affect the model's generalization performance.

Subsampling doesn't directly impact the complexity of the model itself (number of layers or features). Options C and D, increasing input features or layers, would increase model complexity and increase training time. A test dataset is used for evaluation, not for training, so A would not affect training speed. Subsampling should be done carefully as a drastic reduction can lead to underfitting. It is a common technique used when facing training speed constraints. One might have to fine-tune the size of the subsampled dataset for the best trade-off between training speed and model quality.

Cloud platforms like Google Cloud Platform (GCP) often provide tools for data preprocessing and sampling, along with resources for parallelizing training on GPUs or TPUs, which will further reduce training time. Utilizing a managed service like Vertex AI will also manage the scaling of resources for you.

For further research on data sampling and model training on GCP:

Google Cloud Vertex AI: Data Sampling
Google Cloud Vertex AI: Model Training
Google Cloud AI Platform: Training Overview

Question: 78 CertyIQ

You are responsible for writing your company's ETL pipelines to run on an Apache Hadoop cluster. The pipeline will require some checkpointing and splitting pipelines. Which method should you use to write the pipelines?

A. PigLatin using Pig
B. HiveQL using Hive
C. Java using MapReduce
D. Python using MapReduce

Answer: A

Explanation:

The best choice for writing ETL pipelines requiring checkpointing and splitting on an Apache Hadoop cluster is **A. PigLatin using Pig**. Pig is a high-level data flow language specifically designed for parallel data processing on Hadoop. Its scripting nature simplifies complex transformations, while its built-in operators and UDF (User Defined Function) support allow for flexible checkpointing by saving intermediate data sets and splitting workflows through logical branching based on filtering results. These functionalities facilitate easier pipeline management, recovery, and more granular control over processing stages compared to the other options. While Hive (B) is suitable for SQL-like queries on structured data and MapReduce (C, D) provides low-level control, Pig's data flow approach and built-in operators better align with the specific needs of a pipeline requiring checkpointing and splitting. Additionally, Pig's relatively simple scripting compared to Java in MapReduce can reduce the development and debugging time.

Authoritative Resources:

Apache Pig Documentation: https://pig.apache.org/docs/r0.17.0/ - This is the official documentation for Apache Pig and can provide deeper insights into its features, data processing capabilities, and functionalities.
Hadoop: The Definitive Guide by Tom White: This book is an authoritative source on Hadoop and includes a detailed discussion of Pig's functionalities and its applications in ETL pipelines.

Question: 79 CertyIQ

Your company maintains a hybrid deployment with GCP, where analytics are performed on your anonymized customer data. The data are imported to Cloud
Storage from your data center through parallel uploads to a data transfer server running on GCP. Management informs you that the daily transfers take too long and have asked you to fix the problem. You want to maximize transfer speeds. Which action should you take?

A. Increase the CPU size on your server.
B. Increase the size of the Google Persistent Disk on your server.
C. Increase your network bandwidth from your datacenter to GCP.
D. Increase your network bandwidth from Compute Engine to Cloud Storage.

Answer: C

Explanation:

The correct answer is **C. Increase your network bandwidth from your datacenter to GCP.** Here's why: The bottleneck in this scenario is the speed at which data moves from the on-premises data center to Google Cloud Storage (GCS). The problem statement explicitly states that transfers are taking too long, indicating a limitation in the data ingestion pipeline. Since the data is being uploaded in parallel through a transfer server on GCP, the server's resources are less likely to be the primary constraint. Options A and B focus on increasing server resources, which address processing or storage bottlenecks. The key issue is the network

connection between the on-premises data center and GCP. Increasing the CPU size (A) would improve processing power but does not impact data transfer speeds from the source. Similarly, increasing the disk size (B) only changes storage capacity on the server and does not improve network throughput. Option D, while related to networking, addresses the connection between the transfer server and GCS, which is not where the primary bottleneck is occurring. The initial data transfer is from the data center to the transfer server. Therefore, the critical factor for improving transfer speed is the available network bandwidth connecting the on-premises network to Google's network. Increasing this bandwidth will allow for more data to be transferred simultaneously, reducing the overall time taken. This aligns with the fundamental principles of network performance, where bandwidth is the key limiting factor for data transfer speeds.

Authoritative Links:

Google Cloud Networking Overview: https://cloud.google.com/network-connectivity/docs/overview - This documentation provides a broad overview of Google Cloud networking, highlighting the importance of bandwidth and network capacity.
Hybrid Connectivity: https://cloud.google.com/hybrid-connectivity/docs - This resource specifically discusses options for connecting on-premises environments to Google Cloud, emphasizing the role of bandwidth.
Optimizing Storage Transfers : https://cloud.google.com/storage/docs/optimizing-transfers - This doc highlights best practices for optimizing the transfer of data to GCS, which includes adequate network bandwidth.

Question: 80 CertyIQ

MJTelco Case Study -

Company Overview -
MJTelco is a startup that plans to build networks in rapidly growing, underserved markets around the world. The company has patents for innovative optical communications hardware. Based on these patents, they can create many reliable, high-speed backbone links with inexpensive hardware.

Company Background -
Founded by experienced telecom executives, MJTelco uses technologies originally developed to overcome communications challenges in space. Fundamental to their operation, they need to create a distributed data infrastructure that drives real-time analysis and incorporates machine learning to continuously optimize their topologies. Because their hardware is inexpensive, they plan to overdeploy the network allowing them to account for the impact of dynamic regional politics on location availability and cost.
Their management and operations teams are situated all around the globe creating many-to-many relationship between data consumers and provides in their system. After careful consideration, they decided public cloud is the perfect environment to support their needs.

Solution Concept -
MJTelco is running a successful proof-of-concept (PoC) project in its labs. They have two primary needs:
✑ Scale and harden their PoC to support significantly more data flows generated when they ramp to more than 50,000 installations.
✑ Refine their machine-learning cycles to verify and improve the dynamic models they use to control topology definition.
MJTelco will also use three separate operating environments `" development/test, staging, and production `" to meet the needs of running experiments, deploying new features, and serving production customers.

Business Requirements -
✑ Scale up their production environment with minimal cost, instantiating resources when and where needed in an unpredictable, distributed telecom user community.
✑ Ensure security of their proprietary data to protect their leading-edge machine learning and analysis.
✑ Provide reliable and timely access to data for analysis from distributed research workers
✑ Maintain isolated environments that support rapid iteration of their machine-learning models without affecting their customers.

Technical Requirements -
Ensure secure and efficient transport and storage of telemetry data
Rapidly scale instances to support between 10,000 and 100,000 data providers with multiple flows each.

Allow analysis and presentation against data tables tracking up to 2 years of data storing approximately 100m records/day
Support rapid iteration of monitoring infrastructure focused on awareness of data pipeline problems both in telemetry flows and in production learning cycles.

CEO Statement -
Our business model relies on our patents, analytics and dynamic machine learning. Our inexpensive hardware is organized to be highly reliable, which gives us cost advantages. We need to quickly stabilize our large distributed data pipelines to meet our reliability and capacity commitments.

CTO Statement -
Our public cloud services must operate as advertised. We need resources that scale and keep our data secure. We also need environments in which our data scientists can carefully study and quickly adapt our models. Because we rely on automation to process our data, we also need our development and test environments to work as we iterate.

CFO Statement -
The project is too large for us to maintain the hardware and software required for the data and analysis. Also, we cannot afford to staff an operations team to monitor so many data feeds, so we will rely on automation and infrastructure. Google Cloud's machine learning will allow our quantitative researchers to work on our high-value problems instead of problems with our data pipelines.
MJTelco is building a custom interface to share data. They have these requirements:
1. They need to do aggregations over their petabyte-scale datasets.
2. They need to scan specific time range rows with a very fast response time (milliseconds).
Which combination of Google Cloud Platform products should you recommend?

A. Cloud Datastore and Cloud Bigtable
B. Cloud Bigtable and Cloud SQL
C. BigQuery and Cloud Bigtable
D. BigQuery and Cloud Storage

Answer: C

Explanation:

The correct answer is **C. BigQuery and Cloud Bigtable**. Here's why:

MJTelco has two primary data access needs: large-scale aggregations and fast, time-series-based retrieval. BigQuery is Google Cloud's fully-managed, serverless data warehouse, perfectly suited for running complex analytical queries on petabyte-scale datasets. It excels at aggregations and offers the ability to perform analysis on historical data, making it ideal for requirement 1. On the other hand, Cloud Bigtable is a NoSQL wide-column database ideal for storing and quickly retrieving large amounts of time-series data with low latency. This satisfies requirement 2, providing the required milliseconds response time for scanning specific time ranges.

Cloud Datastore (option A) is suitable for transactional data and not for large-scale aggregations. Cloud SQL (option B) is a relational database, which isn't optimal for high-throughput time-series data and struggles to match Bigtable's scale and speed for time-based queries. Cloud Storage (option D) provides object storage, not suited for the fast querying of specific rows in the required milliseconds as specified for use case #2. BigQuery on the other hand excels at large-scale aggregations (use case #1) and can manage petabyte-scale data as described. Thus, the combination of BigQuery and Cloud Bigtable efficiently addresses MJTelco's distinct data requirements. The choice between Bigtable and alternatives like Datastore revolves around the required read speed which big table delivers far better.

Authoritative Links:

BigQuery: https://cloud.google.com/bigquery/docs
Cloud Bigtable: https://cloud.google.com/bigtable/docs
Cloud Datastore: https://cloud.google.com/datastore/docs
Cloud SQL: https://cloud.google.com/sql/docs

Cloud Storage: https://cloud.google.com/storage/docs

Thank you

Thank you for being so interested in the premium exam material.
I'm glad to hear that you found it informative and helpful.

But Wait

I wanted to let you know that there is more content available in the full version.
The full paper contains additional sections and information that you may find helpful, and I encourage you to download it to get a more comprehensive and detailed view of all the subject matter.

Download Full Version Now

Future is Secured

100% Pass Guarantee

24/7 Customer Support

Mail us - certyiqofficial@gmail.com

Free Updates

Lifetime Free Updates!

Total: **397 Questions**

Link: https://certyiq.com/papers/google/professional-data-engineer