Download Google Associate-Data-Practitioner Exam Dumps Free

Shared by Oneill on 17-06-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

You are predicting customer churn for a subscription-based service. You have a 50 PB historical customer dataset in BigQuery that includes demographics, subscription information, and engagement metrics. You want to build a churn prediction model with minimal overhead. You want to follow the Google-recommended approach. What should you do?

Options:

A- Export the data from BigQuery to a local machine. Use scikit-learn in a Jupyter notebook to build the churn prediction model.
B- Use Dataproc to create a Spark cluster. Use the Spark MLlib within the cluster to build the churn prediction model.
C- Create a Looker dashboard that is connected to BigQuery. Use LookML to predict churn.
D- Use the BigQuery Python client library in a Jupyter notebook to query and preprocess the data in BigQuery. Use the CREATE MODEL statement in BigQueryML to train the churn prediction model.

Answer:

D

Explanation:

Using the BigQuery Python client library to query and preprocess data directly in BigQuery and then leveraging BigQueryML to train the churn prediction model is the Google-recommended approach for this scenario. BigQueryML allows you to build machine learning models directly within BigQuery using SQL, eliminating the need to export data or manage additional infrastructure. This minimizes overhead, scales effectively for a dataset as large as 50 PB, and simplifies the end-to-end process of building and training the churn prediction model.

Question 2

Question Type: MultipleChoice

Your team wants to create a monthly report to analyze inventory data that is updated daily. You need to aggregate the inventory counts by using only the most recent month of data, and save the results to be used in a Looker Studio dashboard. What should you do?

Options:

A- Create a materialized view in BigQuery that uses the SUM( ) function and the DATE_SUB( ) function.
B- Create a saved query in the BigQuery console that uses the SUM( ) function and the DATE_SUB( ) function. Re-run the saved query every month, and save the results to a BigQuery table.
C- Create a BigQuery table that uses the SUM( ) function and the _PARTITIONDATE filter.
D- Create a BigQuery table that uses the SUM( ) function and the DATE_DIFF( ) function.

Answer:

A

Explanation:

Creating a materialized view in BigQuery with the SUM() function and the DATE_SUB() function is the best approach. Materialized views allow you to pre-aggregate and cache query results, making them efficient for repeated access, such as monthly reporting. By using the DATE_SUB() function, you can filter the inventory data to include only the most recent month. This approach ensures that the aggregation is up-to-date with minimal latency and provides efficient integration with Looker Studio for dashboarding.

Question 3

Question Type: MultipleChoice

You have created a LookML model and dashboard that shows daily sales metrics for five regional managers to use. You want to ensure that the regional managers can only see sales metrics specific to their region. You need an easy-to-implement solution. What should you do?

Options:

A- Create a sales_region user attribute, and assign each manager's region as the value of their user attribute. Add an access_filter Explore filter on the region_name dimension by using the sales_region user attribute.
B- Create five different Explores with the sql_always_filter Explore filter applied on the region_name dimension. Set each region_name value to the corresponding region for each manager.

C- Create separate Looker dashboards for each regional manager. Set the default dashboard filter to the corresponding region for each manager.
D- Create separate Looker instances for each regional manager. Copy the LookML model and dashboard to each instance. Provision viewer access to the corresponding manager.

Answer:

A

Explanation:

Using a sales_region user attribute is the best solution because it allows you to dynamically filter data based on each manager's assigned region. By adding an access_filter Explore filter on the region_name dimension that references the sales_region user attribute, each manager sees only the sales metrics specific to their region. This approach is easy to implement, scalable, and avoids duplicating dashboards or Explores, making it both efficient and maintainable.

Question 4

Question Type: MultipleChoice

You have a Cloud SQL for PostgreSQL database that stores sensitive historical financial dat

a. You need to ensure that the data is uncorrupted and recoverable in the event that the primary region is destroyed. The data is valuable, so you need to prioritize recovery point objective (RPO) over recovery time objective (RTO). You want to recommend a solution that minimizes latency for primary read and write operations. What should you do?

Options:

A- Configure the Cloud SQL for PostgreSQL instance for regional availability (HA) with asynchronous replication to a secondary instance in a different region.
B- Configure the Cloud SQL for PostgreSQL instance for multi-region backup locations.
C- Configure the Cloud SQL for PostgreSQL instance for regional availability (HA). Back up the Cloud SQL for PostgreSQL database hourly to a Cloud Storage bucket in a different region.
D- Configure the Cloud SQL for PostgreSQL instance for regional availability (HA) with synchronous replication to a secondary instance in a different zone.

Answer:

D

Explanation:

Comprehensive and Detailed in Depth

Why D is correct:Synchronous replication ensures that data is written to both the primary and secondary instances at the same time, minimizing data loss (RPO).

Regional availability (HA) within different zones provides redundancy within the same region, minimizing latency.

Why other options are incorrect:A: Asynchronous replication has a potential for data loss.

B: Multiregion backups are for disaster recovery, not minimizing latency.

C: Hourly backups do not provide the lowest possible RPO.

Cloud SQL high availability: https://cloud.google.com/sql/docs/postgres/high-availability

Cloud SQL backups: https://cloud.google.com/sql/docs/postgres/backup-restore

Question 5

Question Type: MultipleChoice

You need to design a data pipeline to process large volumes of raw server log data stored in Cloud Storage. The data needs to be cleaned, transformed, and aggregated before being loaded into BigQuery for analysis. The transformation involves complex data manipulation using Spark scripts that your team developed. You need to implement a solution that leverages your team's existing skillset, processes data at scale, and minimizes cost. What should you do?

Options:

A- Use Dataflow with a custom template for the transformation logic.
B- Use Cloud Data Fusion to visually design and manage the pipeline.
C- Use Dataform to define the transformations in SQLX.
D- Use Dataproc to run the transformations on a cluster.

Answer:

D

Explanation:

Comprehensive and Detailed In-Depth

The pipeline must handle large-scale log processing with existing Spark scripts, prioritizing skillset reuse, scalability, and cost. Let's break it down:

Option A: Dataflow uses Apache Beam, not Spark, requiring script rewrites (losing skillset leverage). Custom templates scale well but increase development cost and effort.

Option B: Cloud Data Fusion is a visual ETL tool, not Spark-based. It doesn't reuse existing scripts, requiring redesign, and is less cost-efficient for complex, code-driven transformations.

Option C: Dataform uses SQLX for BigQuery ELT, not Spark. It's unsuitable for pre-load transformations of raw logs and doesn't leverage Spark skills.

Option D: Dataproc runs Spark natively, allowing direct use of your team's scripts. It scales for large datasets (ephemeral clusters minimize cost) and integrates with Cloud Storage and BigQuery seamlessly. Why D is Best: Dataproc is Google's managed Spark platform, ideal for large-scale, script-based processing. For example, a script cleaning logs (e.g., parsing, deduplicating) runs as-is on a cluster, writing results to BigQuery via the Spark BigQuery Connector. Cost is minimized with preemptible VMs or auto-scaling clusters. It's the most practical fit for your team's expertise and requirements. Extract from Google Documentation: From 'Dataproc Overview' (https://cloud.google.com/dataproc/docs): 'Dataproc is a managed Spark and Hadoop service that lets you run existing Spark scripts to process large-scale data from Cloud Storage, with cost-effective scaling and integration to BigQuery for analysis.' Reference: Google Cloud Documentation - 'Dataproc' (https://cloud.google.com/dataproc).

Why D is Best: Dataproc is Google's managed Spark platform, ideal for large-scale, script-based processing. For example, a script cleaning logs (e.g., parsing, deduplicating) runs as-is on a cluster, writing results to BigQuery via the Spark BigQuery Connector. Cost is minimized with preemptible VMs or auto-scaling clusters. It's the most practical fit for your team's expertise and requirements.

Extract from Google Documentation: From 'Dataproc Overview' (https://cloud.google.com/dataproc/docs): 'Dataproc is a managed Spark and Hadoop service that lets you run existing Spark scripts to process large-scale data from Cloud Storage, with cost-effective scaling and integration to BigQuery for analysis.'

Option D: Dataproc runs Spark natively, allowing direct use of your team's scripts. It scales for large datasets (ephemeral clusters minimize cost) and integrates with Cloud Storage and BigQuery seamlessly. Why D is Best: Dataproc is Google's managed Spark platform, ideal for large-scale, script-based processing. For example, a script cleaning logs (e.g., parsing, deduplicating) runs as-is on a cluster, writing results to BigQuery via the Spark BigQuery Connector. Cost is minimized with preemptible VMs or auto-scaling clusters. It's the most practical fit for your team's expertise and requirements. Extract from Google Documentation: From 'Dataproc Overview' (https://cloud.google.com/dataproc/docs): 'Dataproc is a managed Spark and Hadoop service that lets you run existing Spark scripts to process large-scale data

from Cloud Storage, with cost-effective scaling and integration to BigQuery for analysis.'
Reference: Google Cloud Documentation - 'Dataproc' (https://cloud.google.com/dataproc).

Question 6

Question Type: MultipleChoice

You are designing a BigQuery data warehouse with a team of experienced SQL developers. You need to recommend a cost-effective, fully-managed, serverless solution to build ELT processes with SQL pipelines. Your solution must include source code control, environment parameterization, and data quality checks. What should you do?

Options:

A- Use Cloud Data Fusion to visually design and manage the pipelines.
B- Use Dataform to build, orchestrate, and monitor the pipelines.
C- Use Dataproc to run MapReduce jobs for distributed data processing.
D- Use Cloud Composer to orchestrate and run data workflows.

Answer:

B

Explanation:

Comprehensive and Detailed In-Depth

The solution must support SQL-based ELT, be serverless and cost-effective, and include advanced features like version control and quality checks. Let's dive in:

Option A: Cloud Data Fusion is a visual ETL tool, not SQL-centric (uses plugins), and isn't fully serverless (requires instance management). It lacks native source code control and parameterization.

Option B: Dataform is a serverless, SQL-based ELT platform for BigQuery. It uses SQLX scripts, integrates with Git for version control, supports environment variables (parameterization), and offers assertions for data quality---all meeting the requirements cost-effectively.

Option C: Dataproc is for Spark/MapReduce, not SQL ELT, and requires cluster management, contradicting serverless and cost goals.

Option D: Cloud Composer orchestrates workflows (Python DAGs), not SQL pipelines natively. It's

managed but not optimized for ELT within BigQuery alone. Why B is Best: Dataform leverages your team's SQL skills, runs in BigQuery (no extra infrastructure), and provides Git integration (e.g., GitHub), parameterization (e.g., DECLARE env STRING DEFAULT 'prod';), and quality checks (e.g., assert 'no_nulls' AS SELECT COUNT(*) FROM table WHERE col IS NULL). It's the perfect fit. Extract from Google Documentation: From 'Dataform Overview' (https://cloud.google.com/dataform/docs): 'Dataform is a fully managed, serverless solution for building SQL-based ELT pipelines in BigQuery, with built-in Git version control, environment parameterization, and data quality assertions for robust data warehouse management.' Reference: Google Cloud Documentation - 'Dataform' (https://cloud.google.com/dataform).

Why B is Best: Dataform leverages your team's SQL skills, runs in BigQuery (no extra infrastructure), and provides Git integration (e.g., GitHub), parameterization (e.g., DECLARE env STRING DEFAULT 'prod';), and quality checks (e.g., assert 'no_nulls' AS SELECT COUNT(*) FROM table WHERE col IS NULL). It's the perfect fit.

Extract from Google Documentation: From 'Dataform Overview' (https://cloud.google.com/dataform/docs): 'Dataform is a fully managed, serverless solution for building SQL-based ELT pipelines in BigQuery, with built-in Git version control, environment parameterization, and data quality assertions for robust data warehouse management.'

Option D: Cloud Composer orchestrates workflows (Python DAGs), not SQL pipelines natively. It's managed but not optimized for ELT within BigQuery alone. Why B is Best: Dataform leverages your team's SQL skills, runs in BigQuery (no extra infrastructure), and provides Git integration (e.g., GitHub), parameterization (e.g., DECLARE env STRING DEFAULT 'prod';), and quality checks (e.g., assert 'no_nulls' AS SELECT COUNT(*) FROM table WHERE col IS NULL). It's the perfect fit. Extract from Google Documentation: From 'Dataform Overview' (https://cloud.google.com/dataform/docs): 'Dataform is a fully managed, serverless solution for building SQL-based ELT pipelines in BigQuery, with built-in Git version control, environment parameterization, and data quality assertions for robust data warehouse management.' Reference: Google Cloud Documentation - 'Dataform' (https://cloud.google.com/dataform).

Question 7

Question Type: MultipleChoice

Your company is migrating their batch transformation pipelines to Google Cloud. You need to choose a solution that supports programmatic transformations using only SQL. You also want the technology to support Git integration for version control of your pipelines. What should you do?

Options:

A- Use Cloud Data Fusion pipelines.

B- Use Dataform workflows.
C- Use Dataflow pipelines.
D- Use Cloud Composer operators.

Answer:

B

Explanation:

Dataform workflows are the ideal solution for migrating batch transformation pipelines to Google Cloud when you want to perform programmatic transformations using only SQL. Dataform allows you to define SQL-based workflows for data transformations and supports Git integration for version control, enabling collaboration and version tracking of your pipelines. This approach is purpose-built for SQL-driven data pipeline management and aligns perfectly with your requirements.

The solution must use SQL for transformations and integrate with Git for version control, focusing on batch pipelines. Let's evaluate:

Option A: Cloud Data Fusion uses a visual UI with plugins, not SQL-only transformations. It lacks native Git integration (requires external tools), missing a key requirement.

Option B: Dataform is a SQL-based workflow tool for BigQuery transformations, defining pipelines as SQLX scripts. It integrates natively with Git for version control, supporting batch ELT processes with minimal overhead.

Option C: Cloud Composer uses Python DAGs and operators, not SQL-only transformations. Git is possible but not intrinsic to its workflow design.

Option D: Dataflow uses Apache Beam (Python/Java), not SQL, and lacks built-in Git support for pipeline definitions. Why B is Best: Dataform is purpose-built for SQL-driven ELT in BigQuery, with Git integration baked in (e.g., GitHub sync). It's serverless, aligns with batch migration, and simplifies pipeline management. Extract from Google Documentation: From 'Dataform Overview' (https://cloud.google.com/dataform/docs): 'Dataform lets you define SQL-based transformation workflows for BigQuery, with native Git integration for version control, enabling teams to manage batch data pipelines programmatically and collaboratively.' Reference: Google Cloud Documentation - 'Dataform' (https://cloud.google.com/dataform).