Big Query

24 June 2021 20:34

Create External Table

bq mkdef --autodetect --source_format=CSV "gs://project_data_sara061/user_appointment_data.csv" > myschema
bq mk --external_table_definition=myschema ic-int-sandbox-saraswata:test_dataset.user_appointment_data

Big Table Import

Temorary External table

```
bq query \
--project_id=ic-int-sandbox-saraswata \
--use_legacy_sql=false \
--external_table_definition=test-table::bigtable_schema \
'SELECT
 *
FROM
  test-table'
```

External table

```
bq mk \
--location=us-east1 \
--external_table_definition=bigtable_schema \
ic-int-sandbox-saraswata:test_dataset.bigtable_data
```

Query with dry run to estimate cost

```
bq query \
--use_legacy_sql=false \
--dry_run \
'SELECT
  COUNTRY,
  AIRPORT,
  IATA
FROM
  `project_id`.dataset.airports
LIMIT
  1000'
```

Key Points

Latency and Throughput

1. Access at dataset(regional and multi-regional, location set only at create time, query can use multiple dataset at same time), table, column (policy tag) and row level (policy tag)
2. Policy tag example: (You must be in group:high-access to see the columns containing TYPE_SSN)
3. Table type native, external(permanent(IAM access can be granted) and temporary), views
4. View, no dml only read-only no export job
5. Materialized view are cached and updated when base table is updated. It is very useful when there is so much computation and scan to result a small dataset. It can be used for BI.
6. External DataSource supported (Cloud storage (json, csv, parquet, avro,OCR), BigTable, CloudSQL, Google Drive), must be in same location. External table has some limitation (less consistency, query performance is slow, caching not supported, cannot predict amount of data to be processed,). Table metadata and schema is stored in bigquery storage. Good for short term less frequently accessed data.
7. For federated query, query involving more than one datasource we can use connection api and EXTERNAL_QUERY function.

Limit

8. Streaming Insert
    a. Maximum byte per second: 1GB per project (without insertId), with insertid 100MB
    b. Maximum rows per second per project: 500,000 (for eu and us) and for other location 100,000 (with insertId)
    c. Maximum rows per table : 100,000 (with InsertId)
    d. Maximum row size : 10MB
9. Maximum row size 1MB for csv and 2MB for JSON
10. Schema auto detection for table, column type cannot be changed once defined.
11. *Keep 7 Day history of data for restoration
12. Partition by partition key (by ingestion time(when record is created) , time-unit column in table and integer column)
13. *Can save query result as table in any dataset
14. Authorized view for providing access to query result to a user in different dataset without giving them the access for base table.
15. Schema auto-detect at load time from external datasource.
16. Can set dataset, table and partition expiration, this reduce cost and increase performance as less scan required
17. Primitive role owner(can create dataset, modify any dataset), editor(can create dataset, table) and viewer(can view ant table data, run query job) have access to big query
18. Bigquery user (can run bigquery job, list own job, can see metadata, create and list dataset)
19. Bigquery.jobuser(can create job)
20. Bigquery.dataviewer(can query table, list table and dataset)
21. Bigquery.admin all access

Consistency

22. We can use insertId during streaming insert to avoid duplicate row (for upto 1 min)
23. Clustering columns keep similar data together along with order
24. Use –dry-run to estimate query cost by byte scanned
25. Streaming insert cost and batch free
26. Can see all query in query history
27. Encrypt all data in rest (using default kek, can also use CMEK) and transit
28. Integrate with dlp
29. Older data which not frequently accessed should be stored in bigquery rather than exporting to other storage to get the benefit of long term storage
30. Storage set (group of capacitor file(Table are store encrypted durable in custom format)) contain (path to storage, commit timestamp, partition key, UUID, state (pending/committed/garbage)). Query only performed on committed storage set. Storage set can span across partition and clustering sort storage set by cluster key

for efficient query.On update bigquery rewrite the storage set and set obsolete data in garbage state. Storage set keep historical data up to 7 days.

Costing

31. If you want to directly stream data into BQ tables that has a cost. Currently the price for streaming is $0.01 per 200 MB (June 2018), so around $50 for 1TB. Batch Load is free.
32. Standard storage cost of bigquery is 0.02$/GB/Month for long term(if table not updated for 90days) cost reduce by 50% I.e, 0.01$/GB/Month. Doing in update operation reset the 90 days for the table where select operation does not affect long term storage.
33. If query run on a project with dataset from different project. Current project is billed for query job and other project billed for storage.
34. Custom quota can be set to fail the query if daily limit exceed for project or user to save cost
35. Partitioning, clustering and expiration reduce cost
36. Month and Yearly reservation can switch from o-demand to flat rate pricing model and reduce cost.
37. Bigquery data transfer service load data from campaign manager, youtube, s3, gcs, google play, google ads, teradata
38. We can write data to bigquery using cloud dataflow pipeline
39. Slot are computational capacity to execute a query as per project quota there can be 2000 concurrent slot to query 100gb data
40. Streaming Quota(maximum byte 1gb /sec without insert id, 100mb/sec with insert id, maximum row /sec/project 100,000 with insert id, for eu an us its 500,000)

Access Control

41. For running queries bigquery job user role is sufficient, bigquery user will provide some extra access like creating dataset which need to be taken care of.
42. A big query dataset can be shared with a user or a group of different project.
43. Long term storage with data not edited for 90 days lower cost

Pricing

On-Demand

Active Storage: Active storage includes any table or table partition that has been modified in the last 90 days.
See more details here.

Long Term Storage: Long-term storage includes any table or table partition that has not been modified for 90 consecutive days
See more details here.

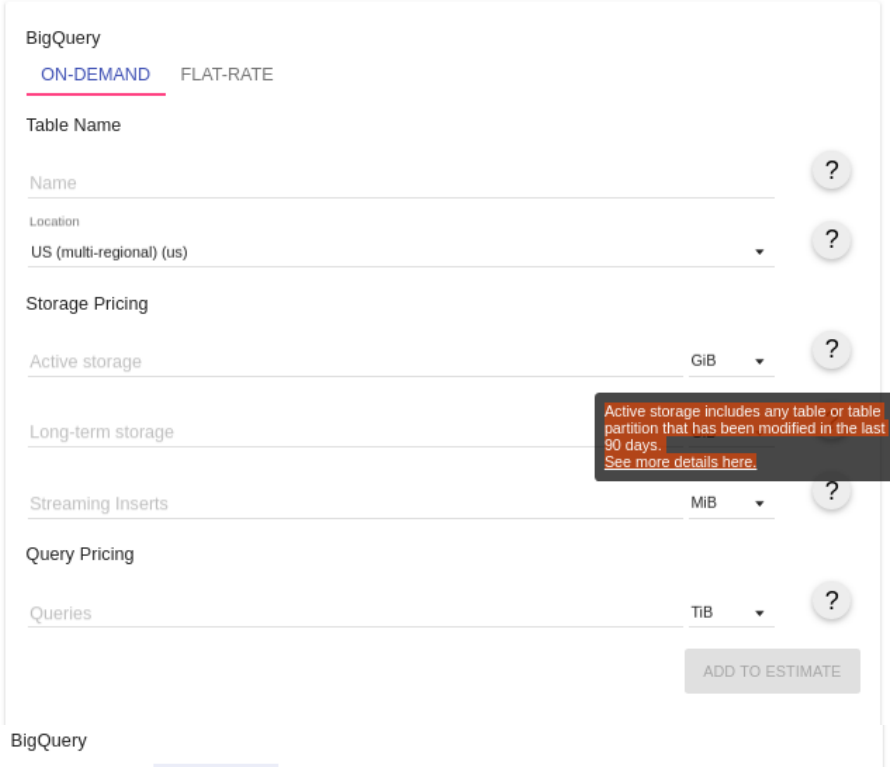
Streaming Insert: Number of Streaming Inserts, priced in megabytes per month.
See more details here.

Query Pricing: BigQuery charges you only for what you consume, per data processed. The first 1 TB of data processed per month is free of charge. You are not charged for queries that return an error.
See more details here.

Flat Rate

Spot Price: A BigQuery Slot is a proprietary measure of capacity. You can choose a number of slots between 100 and 200000. It is a combination of CPU, memory, and networking resources. It also includes a number of supporting technologies and sub-services.

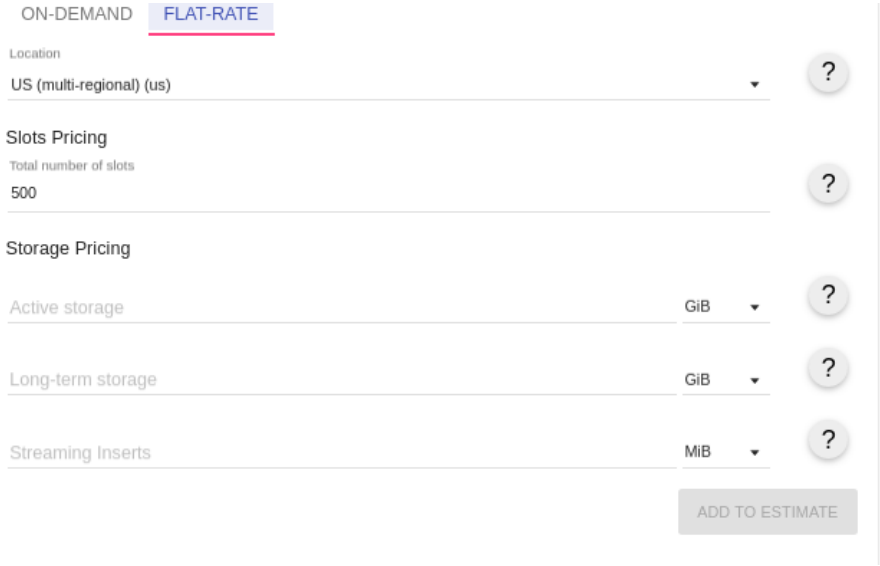
Cost Control using custom quota



If you have multiple BigQuery projects and users, you can manage costs by requesting a custom quota that specifies a limit on the amount of query data processed per day.

Creating a custom quota on query data allows you to control costs at the project-level or at the user-level.

- Project-level custom quotas limit the aggregate usage of all users in that project.
- User-level custom quotas are separately applied to all users and service accounts within a project.



BigQuery Features

BigQuery - Datawarehouse

- **Exabyte scale modern Datawarehousing** solution from GCP
  - Relational database (SQL, schema, consistency etc)
    - Use **SQL-like commands** to query massive datasets
  - Traditional (Storage + Compute) + Modern (Realtime + Serverless)
- When we are talking about a Datawarehouse, **importing and exporting data (and formats) becomes very important:**
  - Load data from a **variety of sources, incl. streaming data**
    - Variety of import formats - CSV/JSON/Avro/Parquet/ORC/Datastore backup
  - Export to Cloud Storage (long term storage) & Data Studio (visualization)
    - Formats - CSV/JSON (with Gzip compression), Avro (with deflate or snappy compression)
- Automatically expire data (**Configurable Table Expiration**)
- Query **external data sources** without storing data in BigQuery
  - Cloud Storage, Cloud SQL, BigTable, Google Drive
  - Use **Permanent or Temporary** external tables

Accessing and querying data

BigQuery - Accessing and Querying Data

- **Access databases using:**
  - Cloud Console
  - bq command-line tool (NOT gcloud)
  - BigQuery Rest API OR
  - HBase API based libraries (Java, .NET & Python)
- (Remember) BigQuery queries **can be expensive** as you are

running them on large data sets!

- (BEST PRACTICE) **Estimate BigQuery queries before running**:
  - **1:** Use UI(console)/bq(--dry-run) - Get scanned data volume (estimate)
  - **2:** Use Pricing Calculator: Find price for scanning 1 MB data. Calculate cost.

Partitioning and clustering

Partitioning and Clustering BigQuery Tables - Use Case

- You pay for BigQuery queries by the amount of data scanned
- How do you reduce your costs of querying BigQuery and improve performance?
- Scenario: Imagine a Questions table with millions of rows
  - You want to find all questions asked between a date range (date between `2022-10-02` and `2028-10-02`) belonging to a specific category
    - If you have a single questions table you need to scan all the rows
      - **Partitioning** - Divide table into multiple segments (example: by date)
      - **Clustering** - Group related data (example: by category)

Questions

Date	Question	Category
2025-10-02	Question Detail ...	GCP
2025-10-02	Question Detail ...	AWS
2025-10-02	Question Detail ...	GCP
2025-10-03	Question Detail ...	Azure
2025-10-03	Question Detail ...	GCP
2025-10-03	Question Detail ...	Azure

Partitioning and Clustering BigQuery Tables

- **Partitioning**: Table is divided into segments
  - Makes it easy to manage and query your data
  - Improves performance and reduces costs
  - Partition based on Ingestion time (arrival time) OR a column (TIMESTAMP, DATE, or DATETIME , or INTEGER)
  - (DEFAULT) All partitions will share same schema as table
  - Allows you to expire (delete) parts of table data easily (partition_expiration_days)
- **Clustering**: Organize table data based on the contents of one or more columns
  - Goal: colocate related data and eliminate scans of unnecessary data
  - Avoid creating too many small partitions (of less than 1 GB). In those cases, prefer Clustering.

Questions_2025_10_02

Date	Question	Category
2025-10-02	Question Detail ...	AWS
2025-10-02	Question Detail ...	GCP
2025-10-02	Question Detail ...	GCP

Questions_2025_10_03

Date	Question	Category
2025-10-03	Question Detail ...	Azure
2025-10-03	Question Detail ...	Azure
2025-10-03	Question Detail ...	GCP

Partitioning and Clustering BigQuery Tables - Syntax

```
CREATE TABLE `my_data_set.questions_partitioned_and_clustered`
...
...
PARTITIONED BY
    DATE(created_date)
    CLUSTER BY category
...
OPTIONS (
  expiration_timestamp=TIMESTAMP "2025-01-01 00:00:00 UTC",
  partition_expiration_days=7
)
```

Expiring data in bigquery

Expiring Data in BigQuery

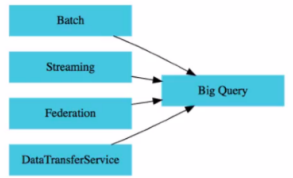
```
CREATE SCHEMA mydataset
OPTIONS(
  default_table_expiration_days=3.75
  )

ALTER TABLE mydataset.mytable
SET OPTIONS (
  expiration_timestamp=TIMESTAMP "2025-01-01 00:00:00 UTC",
  partition_expiration_days=7
)
```

- You pay for data stored in BigQuery:
  - How can you automatically delete (expire) data which is not needed?
- Big Query Hierarchy : Data Set > Table > Partitions
  - You can configure expiration at each level
    - Configure default table expiration (default_table_expiration_days) for datasets
    - Configure expiration time (expiration_timestamp) for tables
    - Configure partition expiration (partition_expiration_days) for partitioned tables
- Best Practice: Expire Tables and Partitions you are NOT using!

Importing data to BigQuery

Importing Data into BigQuery

- **Batch Import** (FREE):
  - Import from Cloud Storage and local files
  - Import after processing by Cloud Dataflow and Cloud Dataproc
- **Streaming Import** ($$$$):
  - From Cloud Pub/Sub, Streaming Inserts
  - Import after processing by Cloud Dataflow and Cloud Dataproc
- **Federated Queries**: Query **external data**
  - Cloud Storage, Cloud SQL, BigTable, Google Drive
- **BigQuery Data Transfer Service**: Import from
  - Google SaaS apps (Google Ads, Cloud Storage etc)
  - External cloud storage providers - Amazon S3
  - Data warehouses - Teradata, Amazon Redshift



Streaming data into BigQuery

Streaming Data into BigQuery

- Loading data in bulk is free **but streaming data is NOT FREE**
  - AND there are a lot of limitations (Use with caution!)
- Streaming data can contain duplicates. How can you avoid duplicates?
  - Add **insertId** with each streaming insert:
    - insertId is used to provide best effort de-duplication (for up to one minute)
      - For strict de-duplication and transactions, try Google Cloud Datastore
- There are **strict streaming quotas** with BigQuery:
  - IF you are NOT populating `insertId`:
    - Maximum bytes per second - 1 GB per second, per project (REMEMBER per project - NOT per table)
  - ELSE (i.e. you are using `insertId`)

- Maximum rows per second per project
  - US and EU multi-regions: 500,000, Other locations: 100,000
  - per table limitation: 100,000
- Maximum bytes per second: 100 MB

- If you have streaming of millions of rows per second, prefer BigTable!

BigQuery Best Practice

Understanding BigQuery Best Practices

- **Estimate your queries** before running them
  - bq --dry_run flag or dryRun API parameter
- Use clustering and partitioning for your tables
- **Avoid streaming inserts** when possible
  - Loading data in bulk is free but streaming data is NOT FREE
  - Offers Best effort de-duplication (when you use `insertId`)
  - Remember Quota limits
- **Expire Data Automatically**:
  - Configure default table expiration (default_table_expiration_days) for datasets
  - Configure expiration time for tables
  - Configure partition expiration for partitioned tables

Understanding BigQuery Best Practices - 2

- Consider **Long-term storage** option
  - Long-term storage: Table in which data is NOT edited for 90 consecutive days
  - Lower Storage cost - Similar to Cloud Storage Nearline
- BigQuery is fast for **complex queries**:
  - BUT it is not as well optimized for narrow-range queries (Prefer Cloud Bigtable)
  - (REMEMBER) Too much complexity in setting up a query
- **Optimize BigQuery usage** using audit logs:
  - Analyze queries/jobs that were run earlier
  - Stream your audit logs (BigQueryAuditMetadata) to BigQuery
    - Understand usage patterns (query costs by user)
    - Optimize (visualize using Google Data Studio)

BigQuery ML

BigQuery ML lets you create and execute machine learning models in BigQuery using standard SQL queries. BigQuery ML democratizes machine learning by letting SQL practitioners build models using existing SQL tools and skills. BigQuery ML increases development speed by eliminating the need to move data.

A model in BigQuery ML represents what an ML system has learned from the training data.

BigQuery ML supports the following types of models:

- Linear regression
- Binary logistic regression
- Multiclass logistic regression
- multinomial classifier with a cross-entropy loss function.

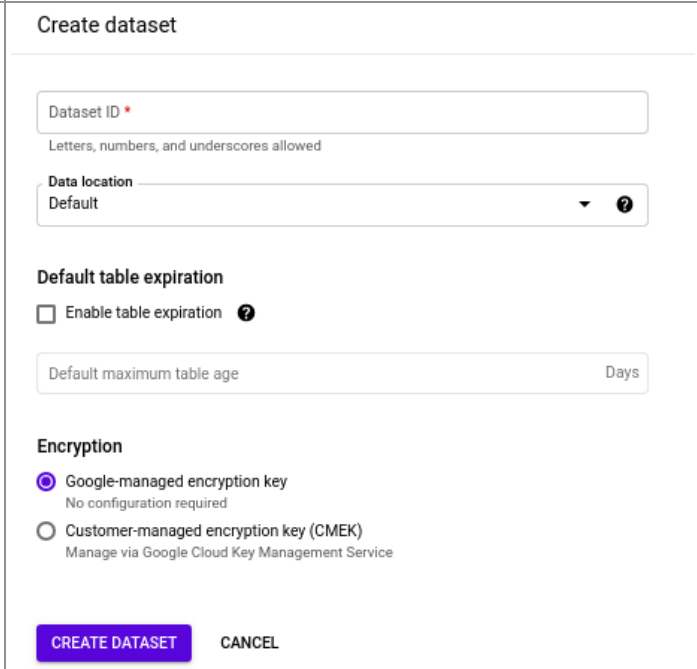
- K-means clustering
- Matrix Factorization
- Time series
- Boosted Tree for creating XGBoost based classification and regression models.
- Deep Neural Network (DNN)
- AutoML Tables
- TensorFlow model importing. This feature lets you create BigQuery ML models from previously trained TensorFlow models, then perform prediction in BigQuery ML.
- Autoencoder

BigQuery ML for the easy way, because data is already in BigQuery and it can host many kinds of models, even custom TensorFlow and Auto ML

Create Dataset

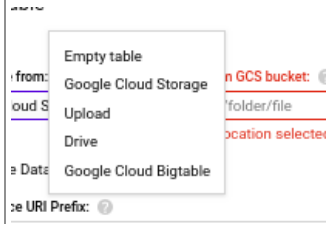
Data Location: If selected, determines location where your data is stored. The current default location is the US multi-region. All tables within this dataset will share this location. This cannot be changed later. Learn more

Enable Table Expiration: Any new table created in this dataset will be automatically deleted the specified number of days after creation. This is useful for temporary data which does not need to be preserved



Create Table

Table source type



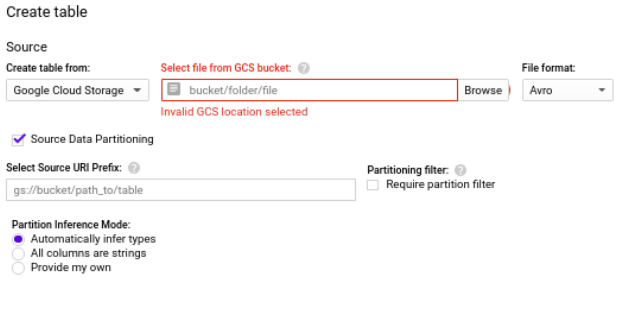


Table Type: Native tables are tables backed by native BigQuery storage. External tables are tables backed by storage external to BigQuery

External tables

An external table is a table that acts like a standard BigQuery table. The table metadata, including the table schema, is stored in BigQuery storage, but the data itself resides in the external source.

External tables can be temporary or permanent. A permanent external table is contained inside a dataset, and you manage it in the same way that you manage a standard BigQuery table. For example, you can view the table properties, set access controls, and so forth. You can query the table and join it with other tables.

You can use external tables with the following data sources:

- Bigtable
- Cloud Storage
- Drive

Hive partition for external GCS Bucket

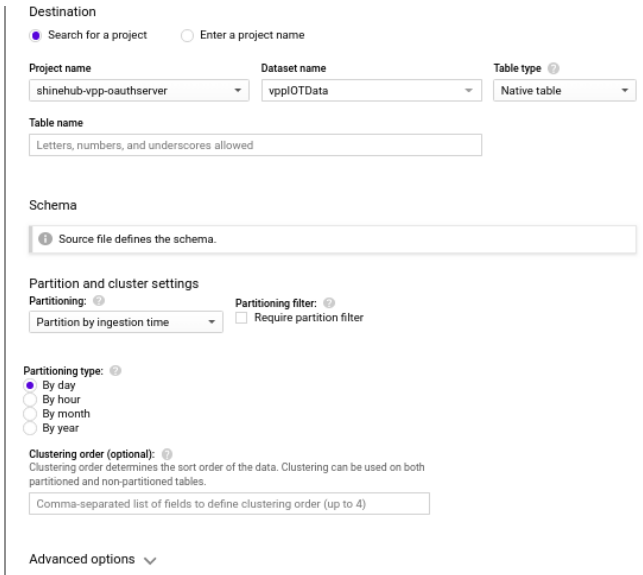
https://sourabhsjain.medium.com/querying-externally-partitioned-data-in-bigquery-97873fd157ac

Clustering Order

Clustering can be performed on multiple columns, where the order of the columns is important as it determines the sort order of the data

Partition Pruning

Partition pruning is an **essential performance feature for data warehouses**. In partition pruning, the optimizer analyzes FROM and WHERE clauses in SQL statements to eliminate unneeded partitions when building the partition access list.



When we define schema we should define each column and datatype

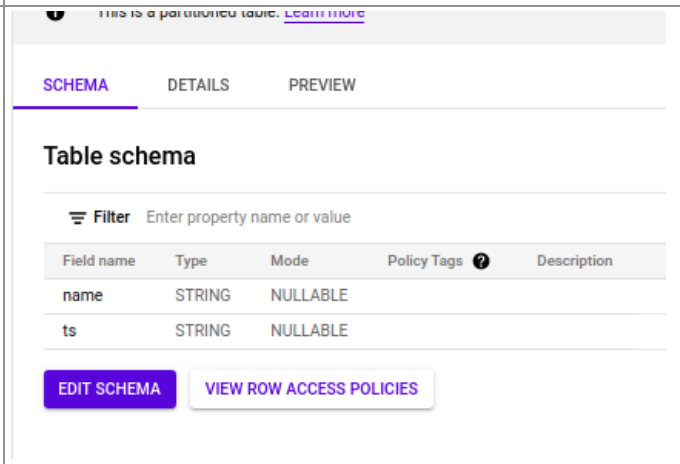
When you load Avro, Parquet, ORC, Firestore export files, or Datastore export files, the schema is automatically retrieved from the self-describing source data.

You can specify a table's schema in the following ways:

- Manually specify the schema:
  - Using the Cloud Console.
  - Inline using the bq command-line tool.
- Create a schema file in JSON format.
- Call the jobs.insert method and configure the schema property in the load job configuration.
- Call the tables.insert method and configure the schema in the table resource using the schema property.

Create a JSON schema file

```
[
  {
    "description": "quarter",
    "mode": "REQUIRED",
```



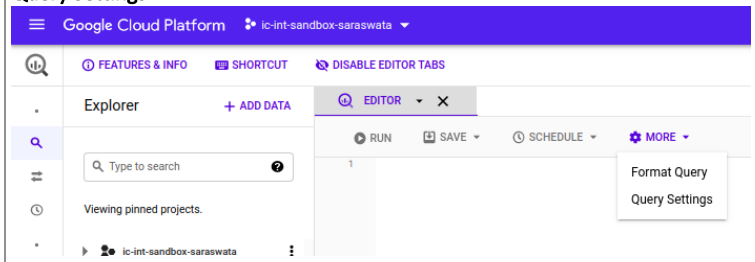
```
    "name": "qtr",
    "type": "STRING"
  },
  {
    "description": "sales representative",
    "mode": "NULLABLE",
    "name": "rep",
    "type": "STRING"
  },
  {
    "description": "total sales",
    "mode": "NULLABLE",
    "name": "sales",
    "type": "FLOAT"
  }
]
```

Using JSON schema file

```
bq --location=location load \
--source_format=format \
project_id:dataset.table \
path_to_data_file \
path_to_schema_file
```

```
bq load --source_format=CSV mydataset.mytable ./myfile.csv ./myschema.json
```

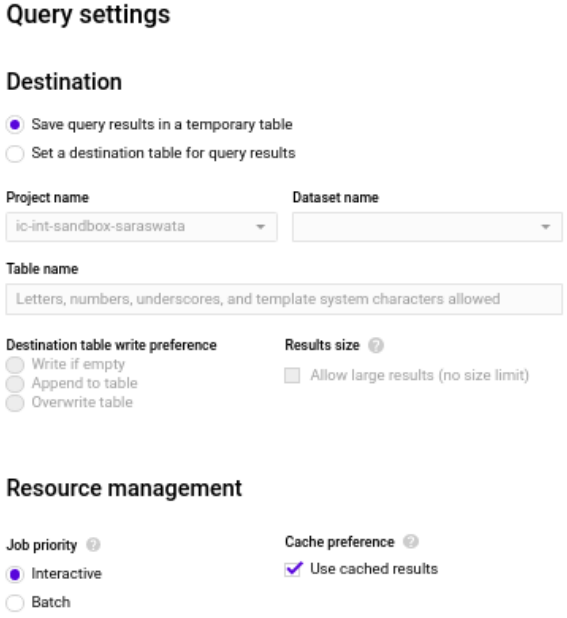
Query Settings



Cloud dataflow sql using big query

Cloud Dataflow SQL lets you use SQL queries to develop and run Dataflow jobs from the BigQuery web UI. You can join streams (such as Pub/Sub) and snapshotted datasets (such as BigQuery tables and Cloud Storage filesets); query your streams or static datasets with SQL by associating schemas with objects, such as tables, Cloud Storage filesets and Pub/Sub topics; and write your results into a BigQuery table for analysis and dashboarding.

Materialized View



In BigQuery, materialized views are precomputed views that periodically cache the results of a query for increased performance and efficiency. BigQuery leverages precomputed results from materialized views and whenever possible reads only delta changes from the base table to compute up-to-date results. Materialized views can be queried directly or can be used by the BigQuery optimizer to process queries to the base tables.

Queries that use materialized views are generally faster and consume fewer resources than queries that retrieve the same data only from the base table. Materialized views can significantly improve the performance of workloads that have the characteristic of common and repeated queries.

Steps to create materialized view

Data definition language (DDL) statements allow you to create and modify tables and views using standard SQL query syntax.

See more on Using data definition language statements.

To create a materialized view in the Cloud Console by using a DDL statement:

1. In the Cloud Console, go to the BigQuery page.

   Go to BigQuery

2. Click **Compose new query**.
3. Type your `CREATE MATERIALIZED VIEW` DDL statement into the **Query editor** text area.

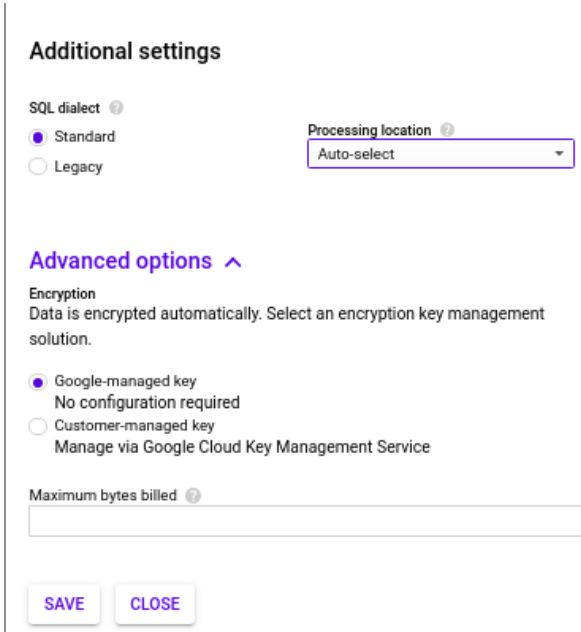
```
CREATE MATERIALIZED VIEW  project-id.my_dataset.my_mv_table
AS SELECT product_id, SUM(clicks) AS sum_clicks
FROM  project-id.my_dataset.my_base_table
GROUP BY 1
```

   where:

   - *project-id* is your project ID.
   - *my_dataset* is the ID of a dataset in your project.
   - *my_mv_table* is the ID of the materialized view that you're creating.
   - *my_base_table* is the ID of a table in your dataset that serves as the base table for your materialized view.
   - `product_id` is a column from the base table.
   - `clicks` is a column from the base table.
   - `sum_clicks` is a column in the materialized view that you are creating.

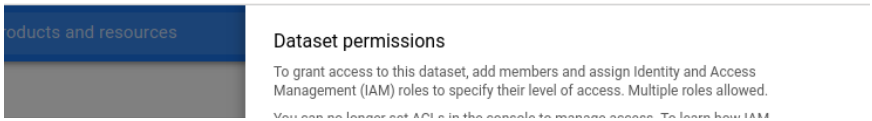
4. Click **Run**.

Unless you disable automatic refreshes, BigQuery starts an asynchronous full refresh for the materialized view. The query might return success right away, but the initial refresh might still be running. When the materialized view is successfully created, it appears in the **Datasets** pane.



Dataset level permission (ACL)

Click on share data

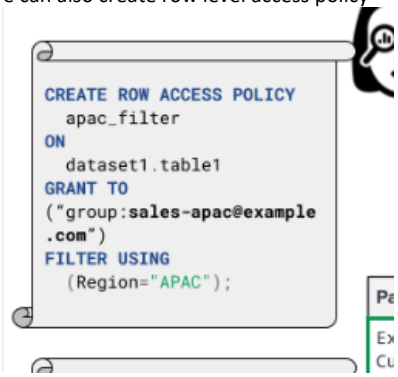


CREATE TABLE SHARE DATASET

Fine Grain Access Control

Core Concepts | BigQuery: Controlling Access

1. Dataset Level Permission by sharing dataset to a user and giving him **bigquery data viewer** role
1. Table Level Permission by sharing table to a user and giving him **bigquery data viewer** role.
2. Store the request of query as permanent table and grant access. This is point in time snapshot new rows are not available. (derived table)
3. Schedule query can be run the sync the data in point 3 but data may get out of sync in between.
4. We can create authorized view to create the view of table this will be always in sync and can grant access to a user to this view.We can save the result of query as a view in the dataset for analyst. Then we can give this view access to datasource used in query from source dataset using authorization view tab.
5. We can provide column level access using policy tag in data catalog section (tags are inside taxonomy), then we can provide access to tag with fine grained reader role to a user. This tag then can be used in any column in the schema of table to restrict access to column.
6. We can also create row level access policy



```
CREATE ROW ACCESS POLICY
  apac_filter
ON
  dataset1.table1
GRANT TO
("group:sales-apac@example
.com")
FILTER USING
  (Region="APAC");
```

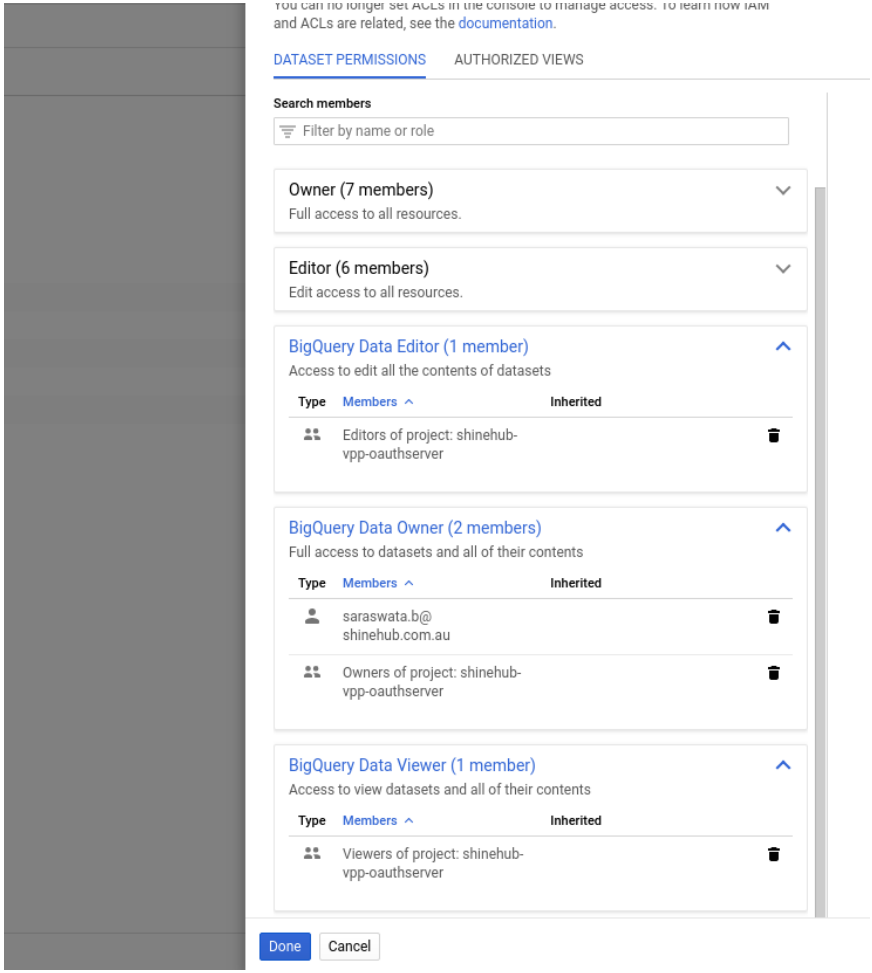
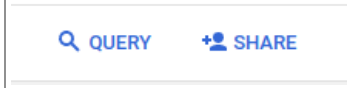


Table level Access Control

Share

Click on share



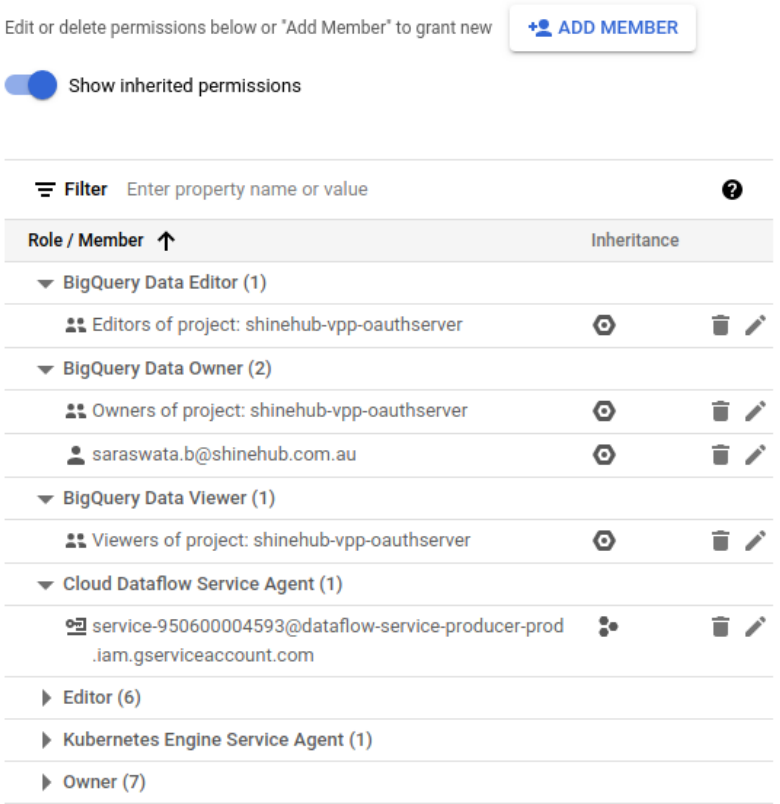
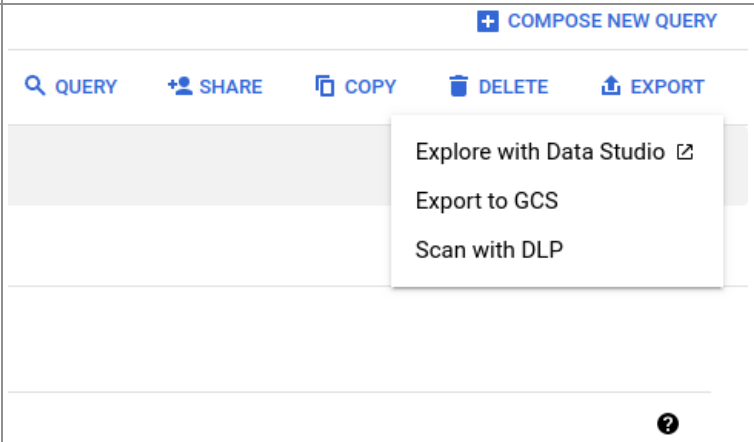


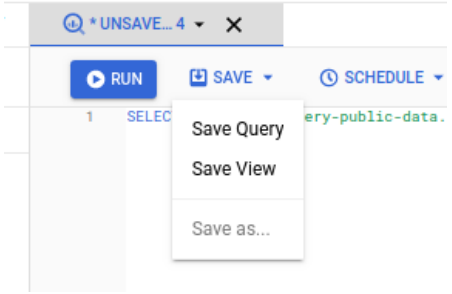
Table Export Option



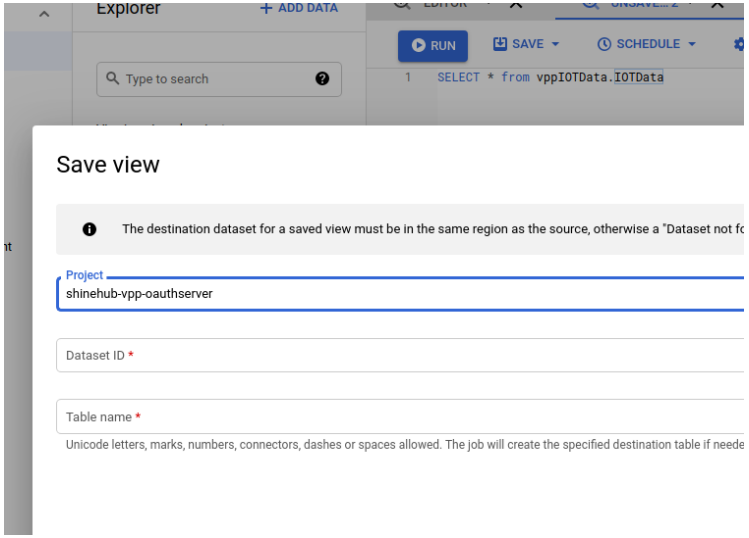
Authorized View

Giving a view access to a dataset is also known as creating an authorized view in BigQuery. An authorized view lets you share query results with particular users and groups without giving them access to the underlying tables. You can also use the view's SQL query to restrict the columns (fields) the users are able to query. In this tutorial, you create an authorized view.

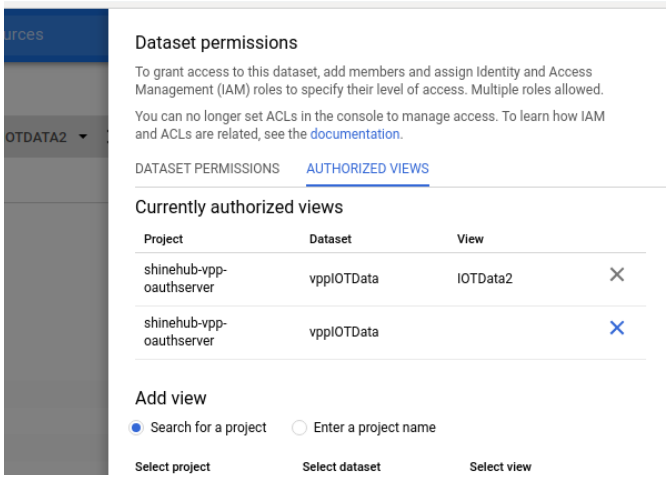
1. **Create Dataset view from query**

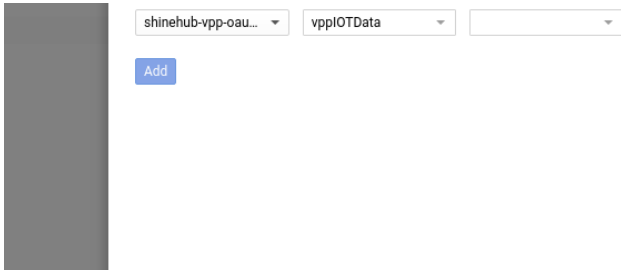


2.



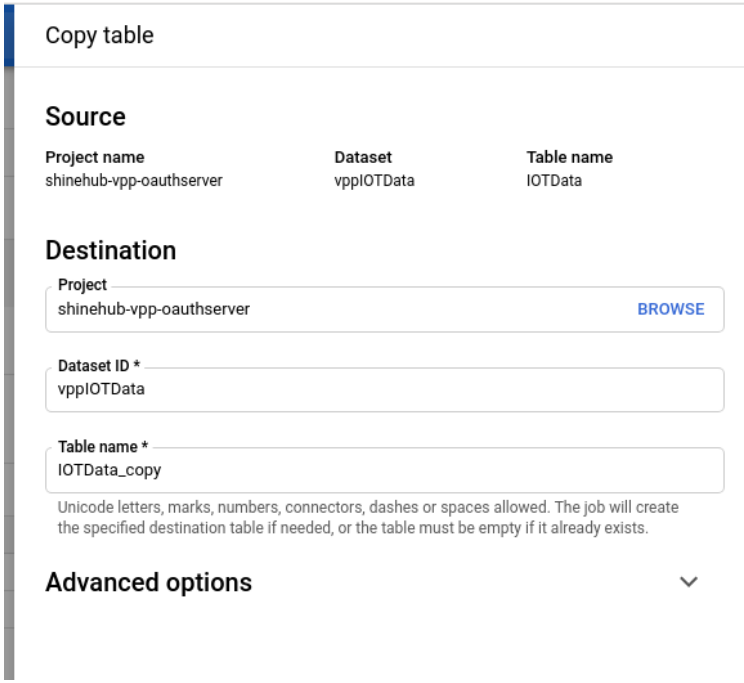
3. **Authorize the view to access source data set**



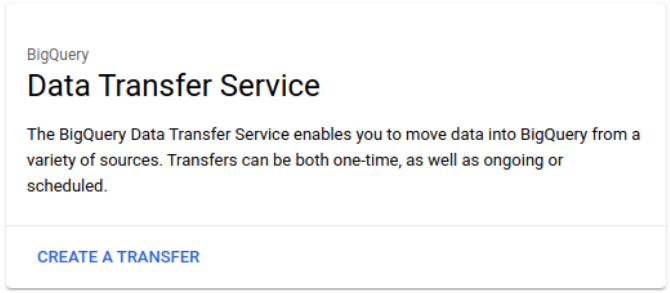


4. **In The IAM section of new dataset containing view provide access to user**

Copy Table



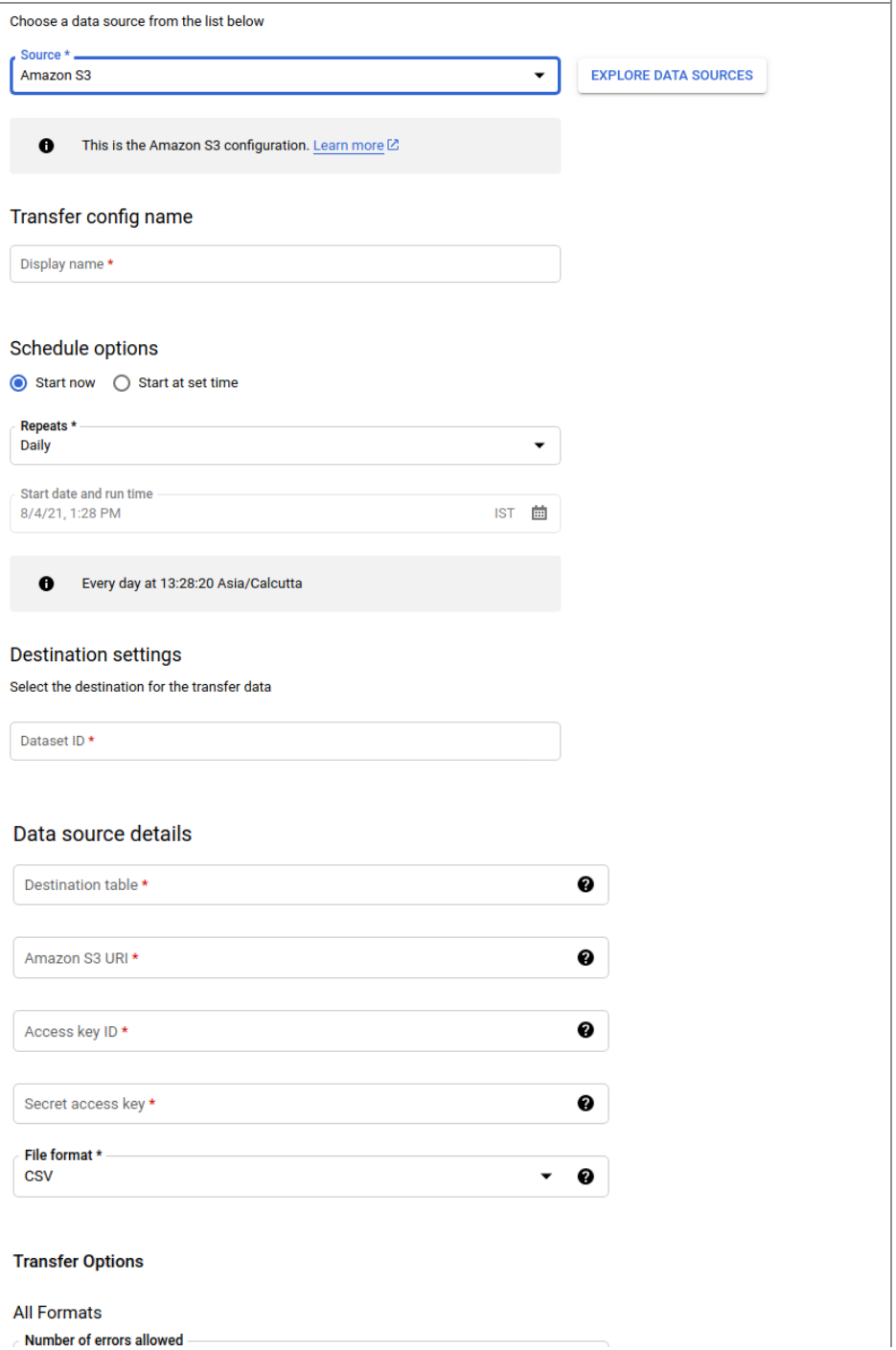
Data Transfer Service

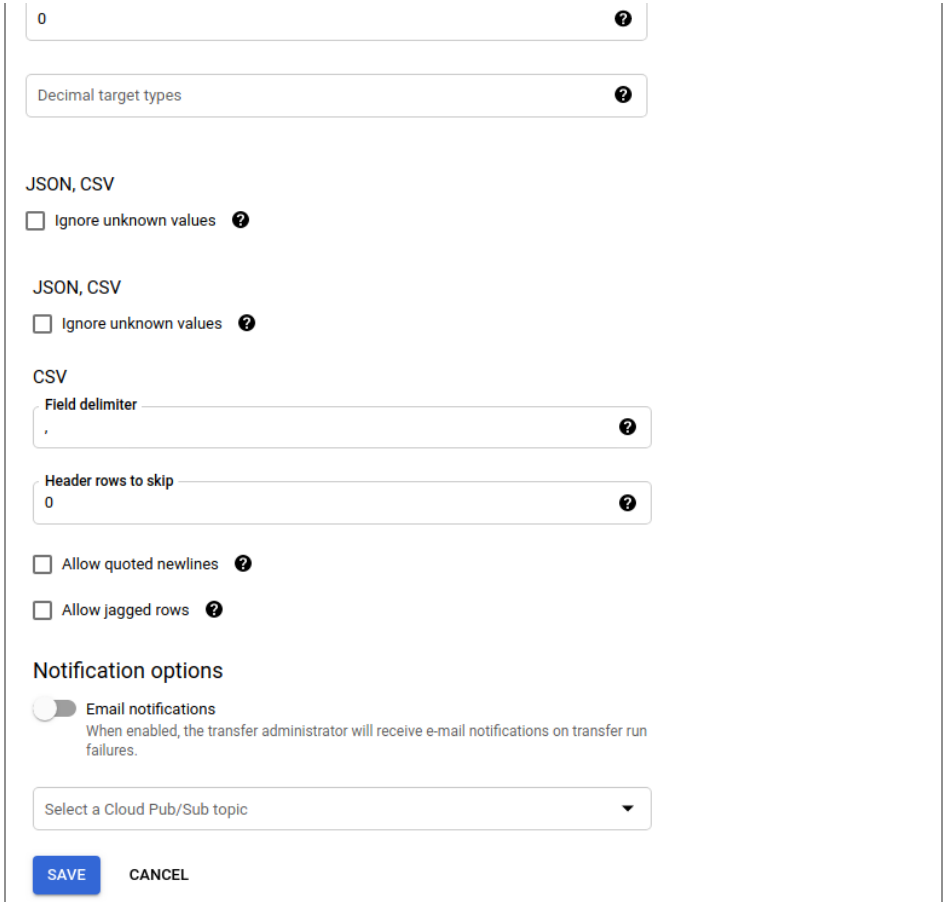


Create Transfer Option

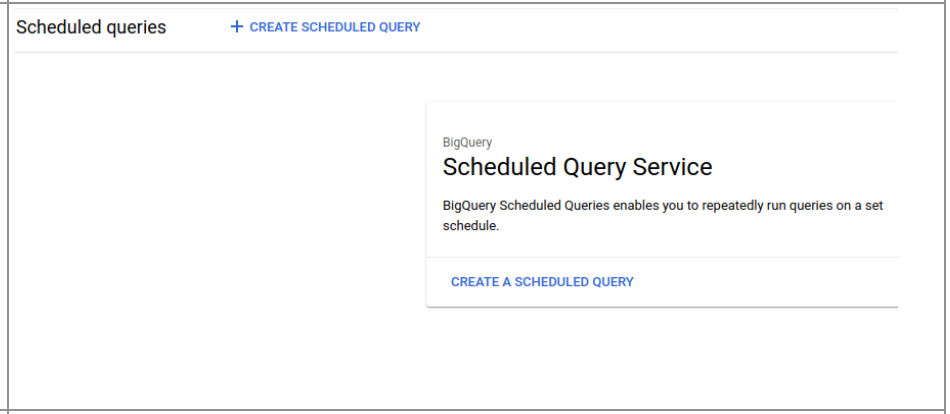
Source

Amazon S3
Campaign Manager (formerly DCM)
Dataset Copy
Google Ad Manager (formerly DFP)
Google Ads (formerly AdWords)
Google Cloud Storage
Google Merchant Center
Google Play
Migration: Redshift
Migration: Teradata
Search Ads 360 (formerly DoubleClick Search)
YouTube Channel
YouTube Content Owner

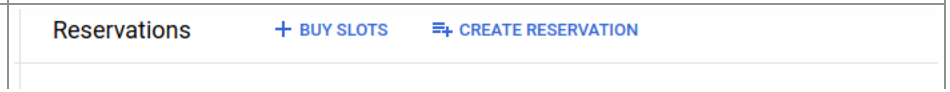




Schedule Query

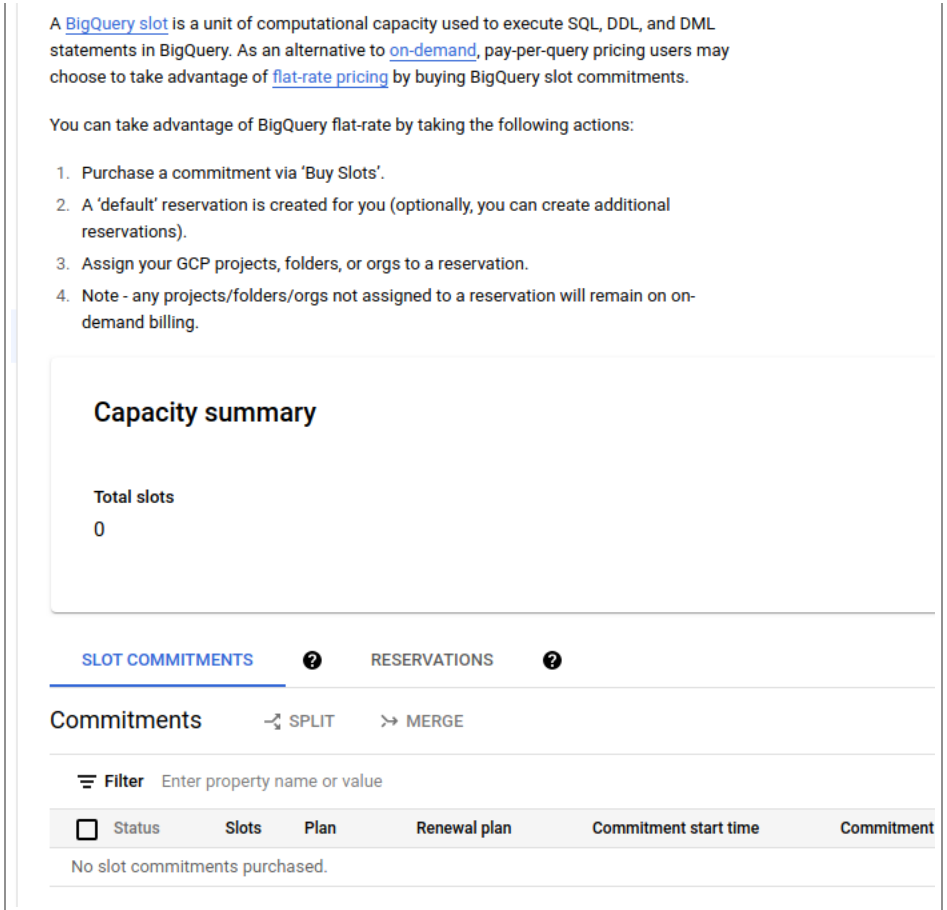


Capacity Management

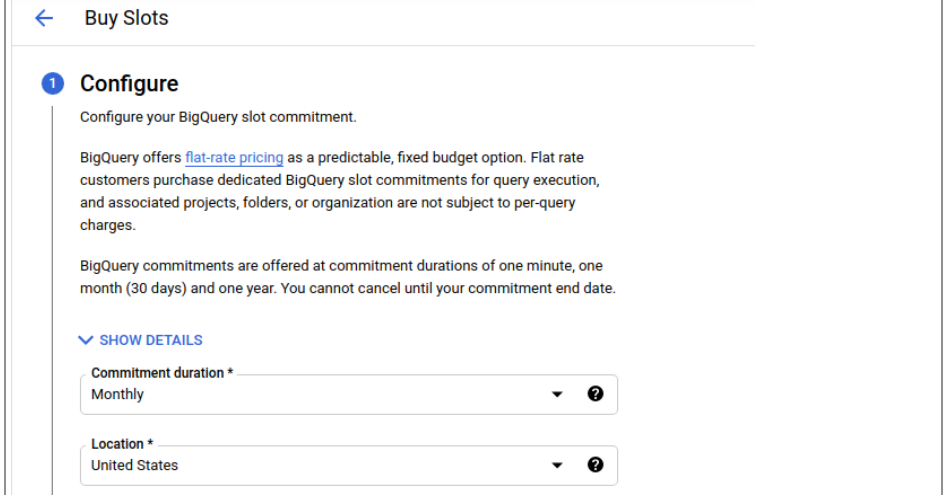


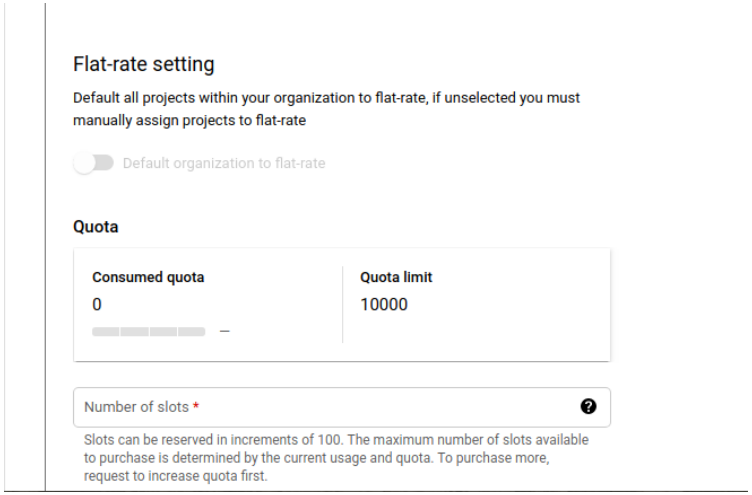
BigQuery offers flat-rate pricing for customers who prefer a stable cost for queries rather than paying the on-demand price per TB of data processed.

BigQuery offers flat-rate pricing for customers who prefer a stable monthly cost for queries rather than paying the on-demand price per TB of data processed. You enrol in flat-rate pricing by purchasing slot commitments, measured in BigQuery slots. Slot commitments start at 500 slots, and the price starts from $10000. Your queries consume this slot capacity, and you are not billed for bytes processed.

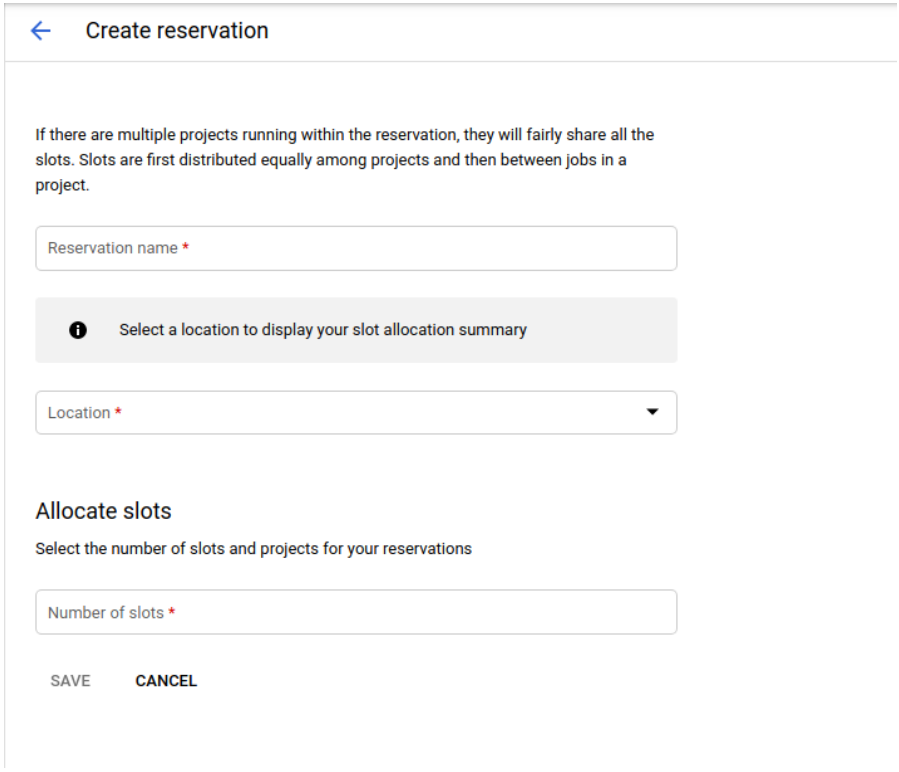


Buy Slot



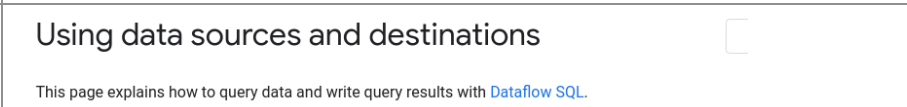


Slot Reservation



Data Flow SQL

1. For CSV file in cloud storage dataflow sql require that we don't specify column name in header.



2. Dataflow SQL natively only works when reading data from Pub/Sub topics, Cloud Storage file sets, and BigQuery tables.
3. You can make use of the Cloud Dataflow connector for Cloud Spanner (https://cloud.google.com/spanner/docs/dataflow-connector) and Dataflow Connector for Cloud Bigtable (https://cloud.google.com/bigtable/docs/hbase-dataflow-java) to retrieve data from these sources

Dataflow SQL can query the following sources:

- Streaming data from Pub/Sub topics
- Streaming and batch data from Cloud Storage filesets
- Batch data from BigQuery tables

Dataflow SQL can write query results to the following destinations:

- Pub/Sub topics
- BigQuery tables