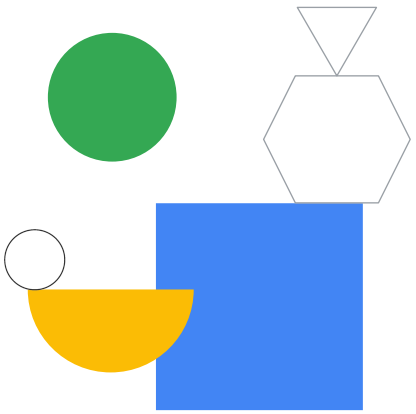
Google Cloud

Storing and Ingesting New Datasets



Agenda

01 Permanent Versus Temporary Data Tables

02 Ingesting New Datasets

- Demo: Querying External Data Sources from BigQuery
- Lab: Ingesting New Datasets into BigQuery

One of the building blocks of data analysis is creating SQL queries on raw data sources and then saving your results into new reporting tables that you can then query from. Here we'll cover the difference between permanent and temporary data tables and how to store the results of your queries.

Permanent Versus Temporary Tables

How to create a new permanent table

1. Write SQL query
2. Click **More > Query Settings**
3. Specify the **destination table** (can be existing)
4. Choose **Write Preference** (if table already exists)
5. Run query

If the destination table exists:

- Write if empty
- Append records
- Overwrite table

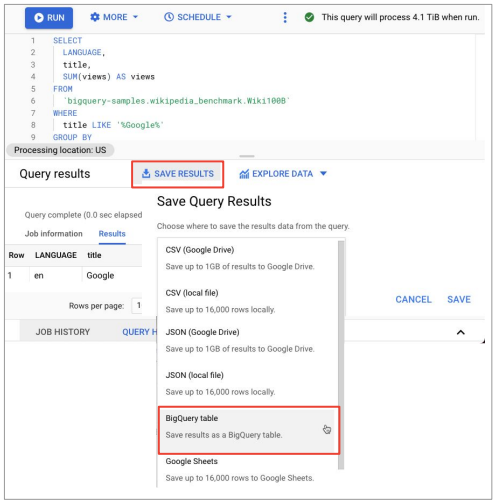
Query results are always saved to either a [temporary or permanent table]. You can choose whether to append or overwrite data in an existing table and whether to create a new table if none exists by that name.

Also: Alternately, if you forget to specify a destination table before running your query, you can copy the temporary table to a permanent table by clicking the Save as Table button in the results window.

More details:
https://cloud.google.com/bigquery/querying-data#queries

Forgot to specify a table? Store query results after running

- Use **SAVE RESULTS > BigQuery table.**
- **All query results are stored in tables** (regardless if you save as table)
- If you don't save as a permanent table, a temporary one is automatically created and saved for 24 hours
- Re-running the same query will likely hit the cached temporary table



All query results end up in a table (even if you don't explicitly create one)

If you don't create one, a temp one is created (per user, per project) and is saved for 24 hours. Same queries will use cached results when available assuming:

- Underlying table not modified
- No CURRENT_DATE() or similar function is used

See if it's used after execution time (will say cached result)

Can be disabled in Show Options

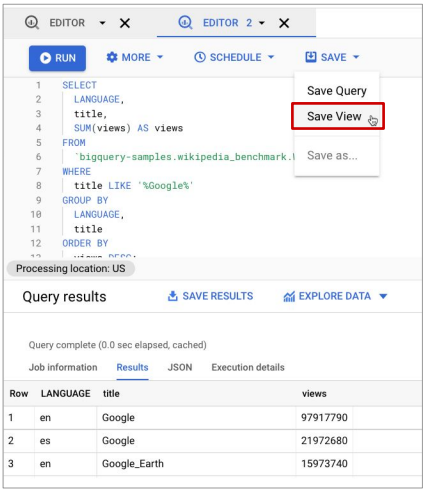
All query results are saved to a table

- All query results are saved to either a temporary or permanent table
- If you specify a destination table then that table becomes permanent otherwise it's a new temporary table
- Temporary tables are the basis of query cached results
- Temporary tables last 24 hours only

Alternately, if you forget to specify a destination table before running your query, you can copy the temporary table to a permanent table by clicking the Save as Table button in the results window.

Storing results in a view

- View = Saved SQL query (a virtual table)
- The underlying query is executed each time the view is accessed



A view is a virtual table defined by a SQL query. You can query views in BigQuery using the web UI, the command-line tool, or the API. You can also use a view as a data source for a visualization tool such as Google Data Studio.

BigQuery's views are logical views, not materialized views. Because views are not materialized, the query that defines the view is run each time the view is queried. Queries are billed according to the total amount of data in all table fields referenced directly or indirectly by the top-level query. For more information, see query pricing.

SQL queries used to define views are subject to the standard query quotas.

Limitations

BigQuery views are subject to the following limitations:

- You cannot run a BigQuery job that exports data from a view.
- You cannot use the TableDataList JSON API method to retrieve data from a view. For more information, see Tabledata: list.
- You cannot mix standard SQL and legacy SQL queries when using views. A standard SQL query cannot reference a view defined using legacy SQL syntax.
- The schemas of the underlying tables are stored with the view when the view is created. If columns are added, deleted, and so on after the view is created,

- the reported schema will be inaccurate until the view is updated. Even though the reported schema may be inaccurate, all submitted queries produce accurate results.
- You cannot update a legacy SQL view to standard SQL in the BigQuery web UI. You can change the SQL language using the command-line tool `bq update --view` command or by using the update or patch API methods.
- You cannot include a user-defined function in the SQL query that defines a view.
- You cannot reference a view in a wildcard table query.
- BigQuery supports up to four levels of nested views in legacy SQL. If there are more than four levels, an `INVALID_INPUT` error returns. In standard SQL, you are limited to 100 levels of nested views.
- You are limited to 1,000 authorized views per dataset.

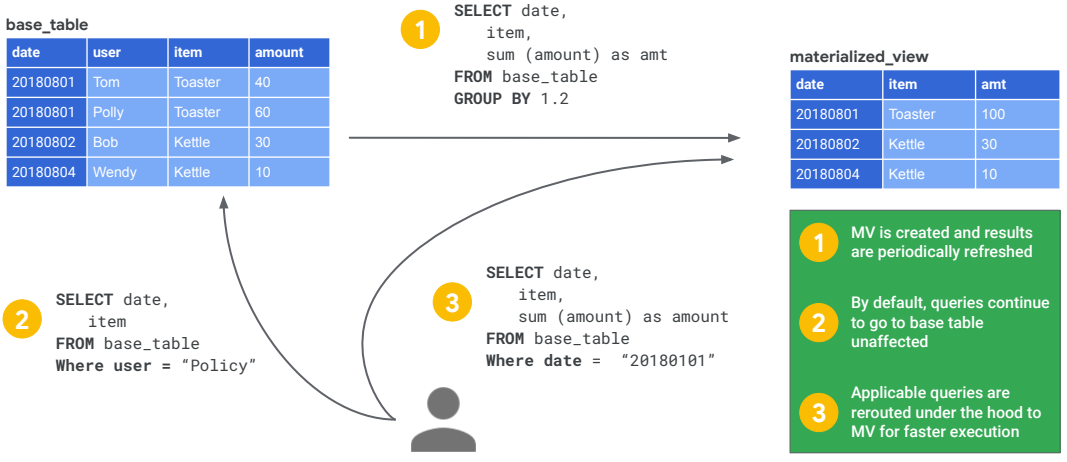
Tip: You can create tables and views using SQL

```
CREATE VIEW ecommerce.shirts_vw AS
SELECT DISTINCT
  v2ProductName
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE LOWER(v2ProductName) LIKE '%shirt%'
LIMIT 100
```

Use “OR REPLACE” or “IF NOT EXISTS” if view needs be overwritten or not

```
CREATE OR REPLACE VIEW ecommerce.shirts_vw AS
SELECT DISTINCT
  v2ProductName
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE LOWER(v2ProductName) LIKE '%shirt%'
LIMIT 100
```

Introduction to materialized views



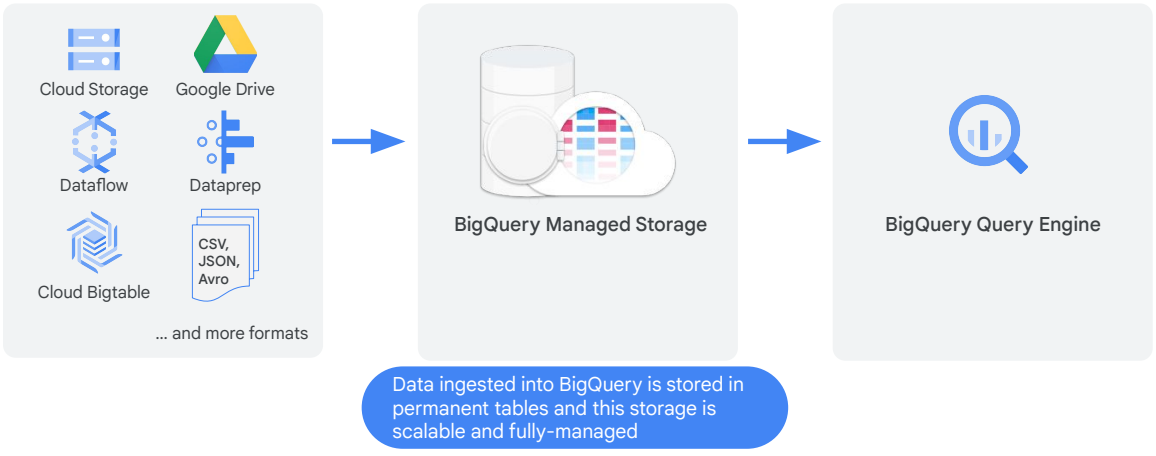
In BigQuery, materialized views are precomputed views that periodically cache the results of a query for increased performance and efficiency. BigQuery leverages precomputed results from materialized views and whenever possible reads only delta changes from the base table to compute up-to-date results. 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.

Further information on materialized views:
https://cloud.google.com/bigquery/docs/materialized-views-intro

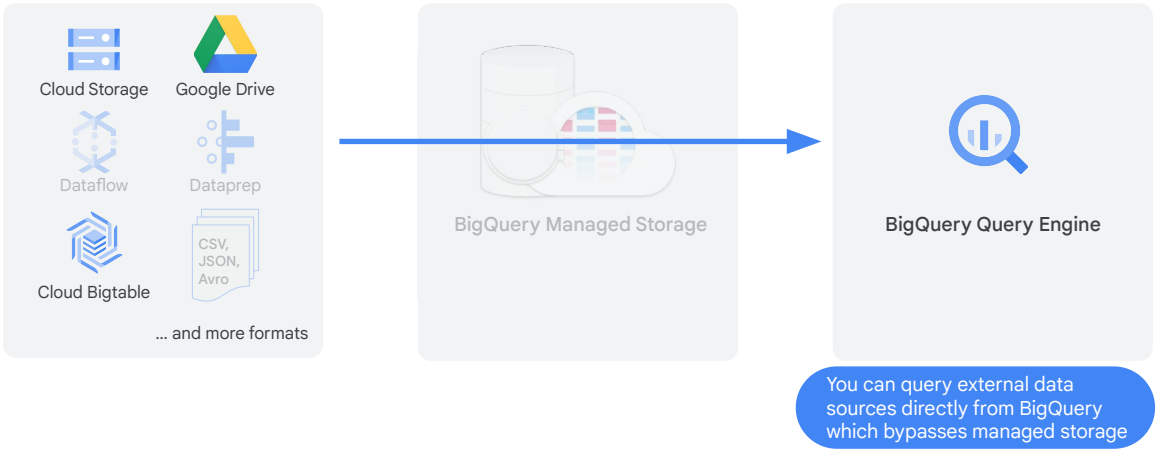
Ingesting New Datasets

Ingest data permanently into BigQuery from a variety of formats



BigQuery can ingest datasets from a variety of different formats. Once inside BigQuery native storage, it is fully managed by the BigQuery team here at Google (replication, backups, scaling out size, and more).

BigQuery can query external data sources in Cloud Storage and Drive directly

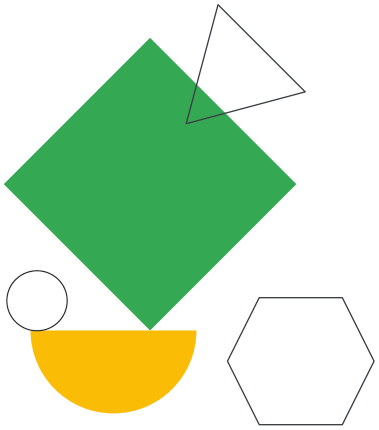


You also have the option of querying external data sources directly and bypassing BigQuery managed storage.

Demo

Querying External Data Sources from BigQuery

Live updates from Google Spreadsheets



Refer to https://github.com/GoogleCloudPlatform/training-data-analyst/tree/master/courses/data-to-insights/demos/external-data-query.sql

Pitfalls: Querying from external data sources directly

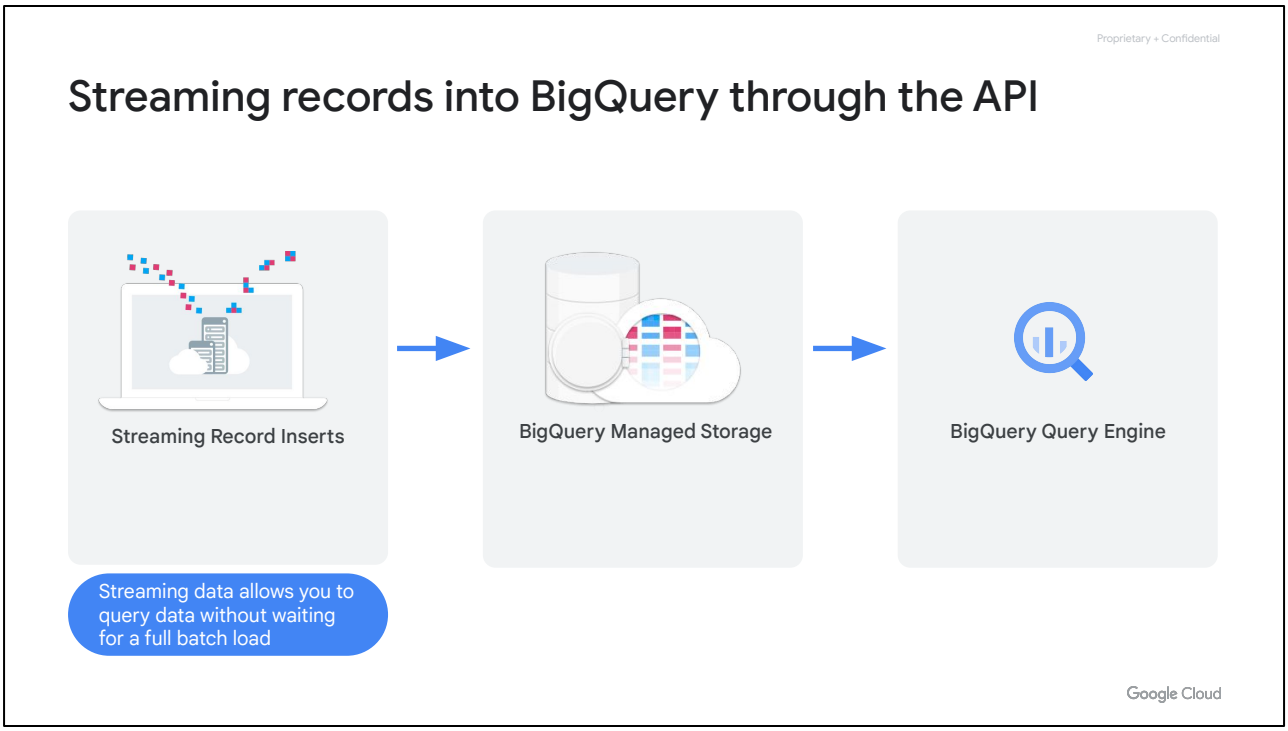
Limitations

- Strong performance disadvantages
- Data consistency not guaranteed
- Can't use table wildcards (cool feature we will introduce shortly)

Limitations

External data source limitations include the following:

- BigQuery does not guarantee data consistency for external data sources. Changes to the underlying data while a query is running can result in unexpected behavior.
- Query performance for external data sources may not be as high as querying data in a native BigQuery table. If query speed is a priority, load the data into BigQuery instead of setting up an external data source. The performance of a query that includes an external data source depends on the external storage type. For example, querying data stored in Google Cloud Storage is faster than querying data stored in Google Drive. In general, query performance for external data sources should be equivalent to reading the data directly from the external storage.
- You cannot use the `TableDataList` JSON API method to retrieve data from tables that reside in an external data source. For more information, see Tabledata: list.
- You cannot run a BigQuery job that exports data from an external data source.
- You cannot reference an external data source in a wildcard table query.



Stream records into BigQuery

- Using the API tabledata().insertAll() method
- Max row size: 1MB
- Max throughput: 100,000 per second per project
  Max rows per request: 10,000 (batching recommended)

Event Logging

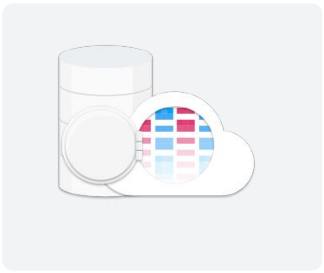
One example of high volume event logging is event tracking. Suppose you have a mobile app that tracks events. Your app, or mobile servers, could independently record user interactions or system errors and stream them into BigQuery. You could analyze this data to determine overall trends, such as areas of high interaction or problems, and monitor error conditions in real-time.

Real-time dashboards and queries

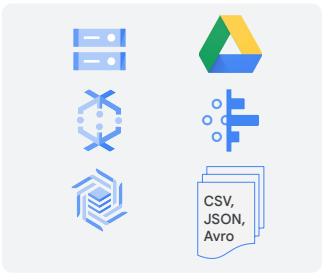
In certain situations, streaming data into BigQuery enables real-time analysis over transactional data. Since streaming data comes with a possibility of duplicated data, ensure that you have a primary, transactional data store outside of BigQuery.

https://cloud.google.com/bigquery/streaming-data-into-bigquery

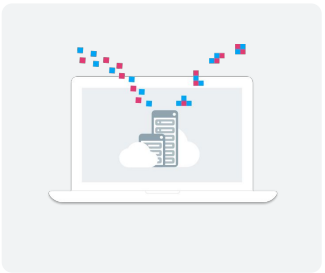
Summary: Ingest new datasets into BigQuery managed storage



Data stored permanently in BigQuery is fully-managed (performance, backups, redundancy).



Load new data from a variety of formats.



Setup streaming ingestion into BigQuery through APIs.

As you have seen, there are a variety of ways to get your new data into BigQuery. We covered loading and storing this data as new permanent tables (which have the benefit of being fully-managed). We also looked at querying external data directly and why this may be useful for one-time Extract Transform Load jobs. Lastly, you can stream individual records into BigQuery through an API.

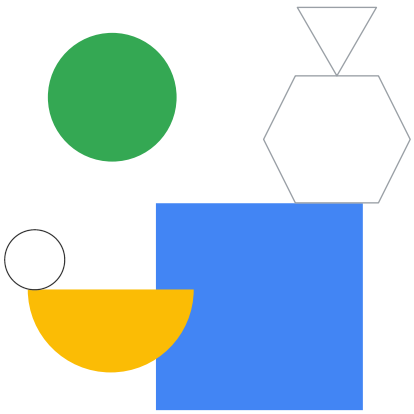
Next up is our lab where we will practice creating new datasets, loading external data into tables, and running queries on this new data.

Links:
Loading Data into BigQuery: https://cloud.google.com/bigquery/loading-data

Image (data blocks) cc0: https://cloud.google.com/data-transfer/

Lab Intro

Ingesting New Datasets into BigQuery



Lab objectives

01 Ingest a new Dataset from a CSV

02 Exploring newly loaded data with SQL

03 Ingest data from Google Cloud Storage

04 Ingest a new dataset from a Google Spreadsheet

05 Saving data to Google Sheets

06 External table performance and data quality consideration