The Extract and Load Data Pipeline Pattern

In this module, you learn to ...

01 Explain the baseline extract and load architecture diagram.

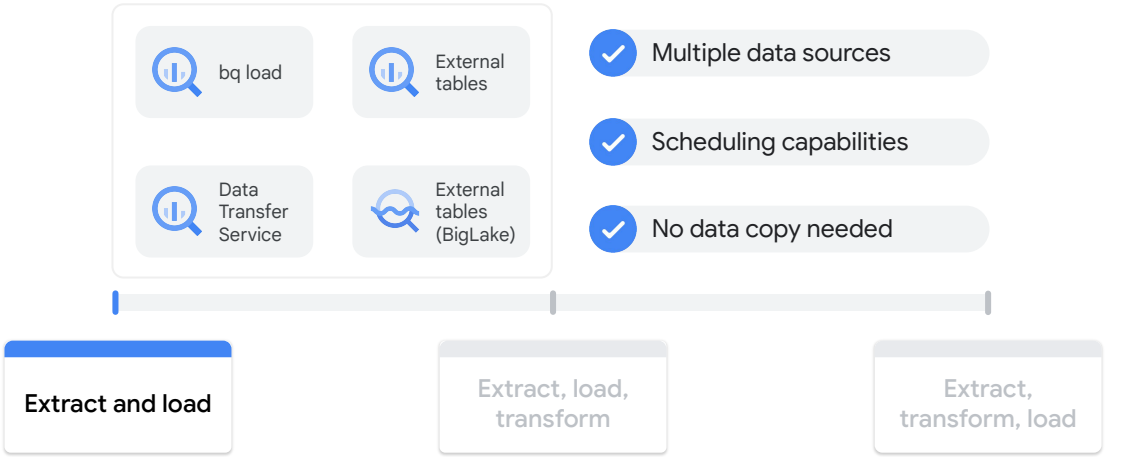
02 Understand the options of the **bq** command line tool.

03 Explain the functionality and use cases for the BigQuery Data Transfer Service.

04 Explain the functionality and use cases for BigLake as a non extract-load pattern.

In this module, first, you review the baseline extract and load architecture diagram. Second, you explore the options of the bq command line tool. Then, you review the functionality and use cases for the BigQuery Data Transfer Service. Finally, you look at the functionality and use cases for BigLake as a non extract-load pattern.

Extract and load ingests data into BigQuery without the need for transformation

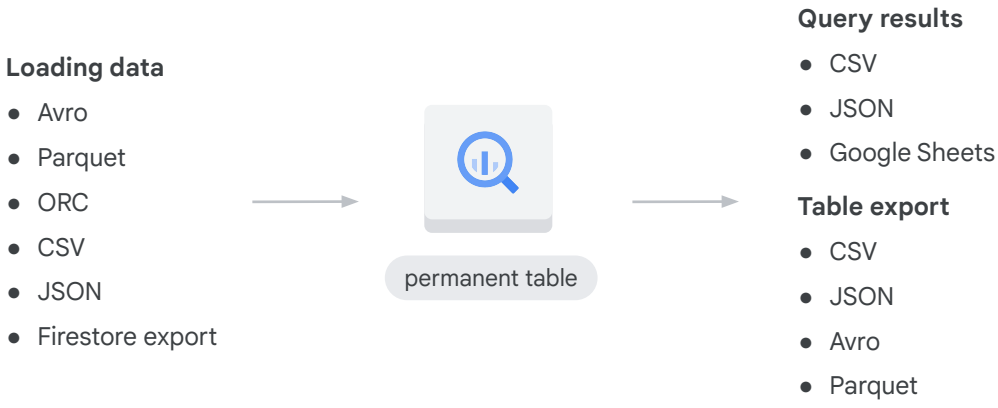


The extract and load data pipeline pattern focuses on the tools and options to bring data into BigQuery by eliminating the need for upfront transformation.

Extract and load greatly simplifies data ingestion into BigQuery. Extract and load leverages tools like `bq load` and Data Transfer Service to directly load data from various sources or uses external tables and BigLake tables to make data accessible via BigQuery.

This pattern also offers scheduling capabilities and eliminates the need for data copying, promoting efficiency in data pipelines.

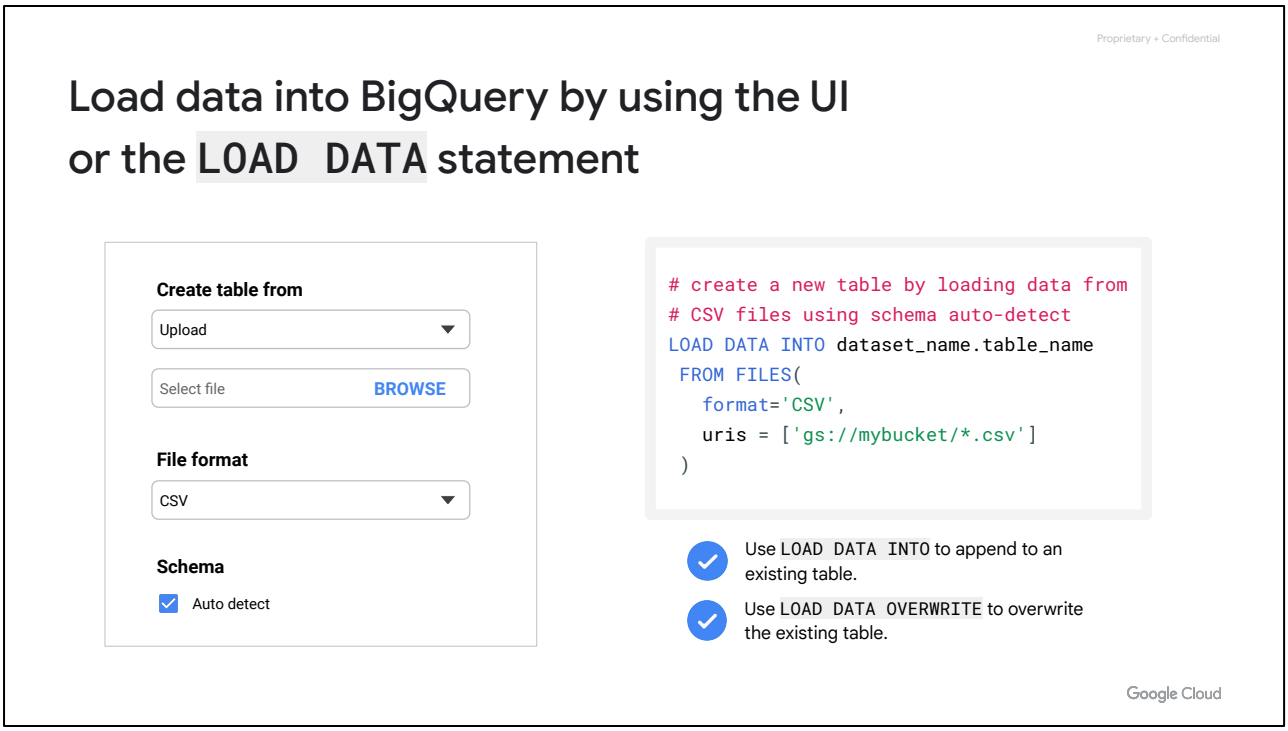
BigQuery supports importing from and exporting to multiple formats



BigQuery provides extensive flexibility in data handling.

It supports loading data from various formats like Avro, Parquet, ORC, CSV, JSON, as well as Google Cloud Firestore exports.

Similarly, you can export BigQuery artifacts, including query results and table data, into formats like CSV, JSON, Avro, and Parquet, facilitating easy integration with other tools and systems.



BigQuery offers two ways to load data: through its friendly user interface for file uploads or via the `LOAD DATA` SQL statement.

The UI simplifies the process, allowing you to select files, specify formats, and even auto-detect schema.

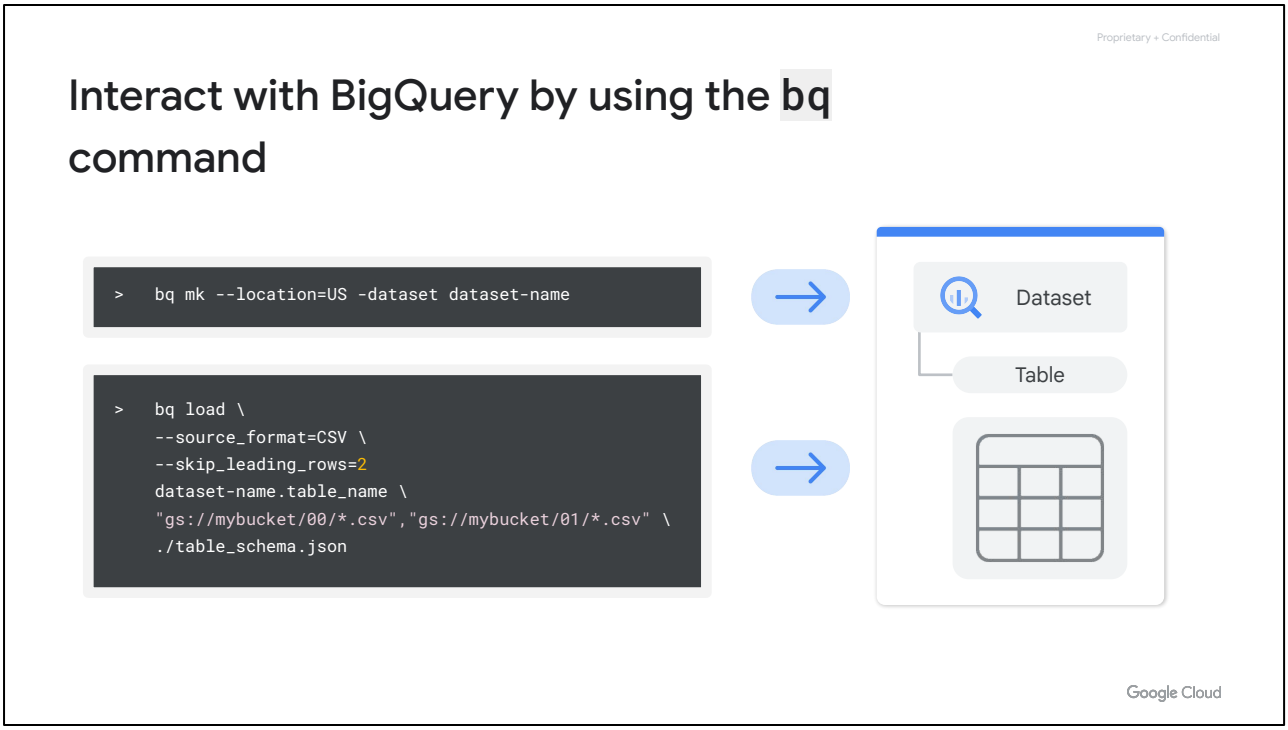
`LOAD DATA` provides more control, ideal for automation and appending or overwriting existing tables.

References:

https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-csv#loading_csv_data_into_a_table
https://cloud.google.com/bigquery/docs/reference/standard-sql/other-statements#load_data_statement

Instructor notes:

- Students already observed the UI example in Lab 1 (Loading data into BigQuery: https://www.cloudskillsboost.google/catalog_lab/2119)



The Cloud SDK's `bq` command offers a programmatic way to interact with BigQuery.

You can create BigQuery objects like datasets and tables with the familiar Linux-like command `bq mk`.

The `bq load` command efficiently loads data into BigQuery tables.

Key parameters for `bq load` include specifying the source format such as CSV, skipping header rows, and defining the target dataset and table.

You can load data from multiple files in Cloud Storage using wildcards and optionally provide a schema file for the table structure.

These options provide flexibility and control for loading data into BigQuery from various sources.

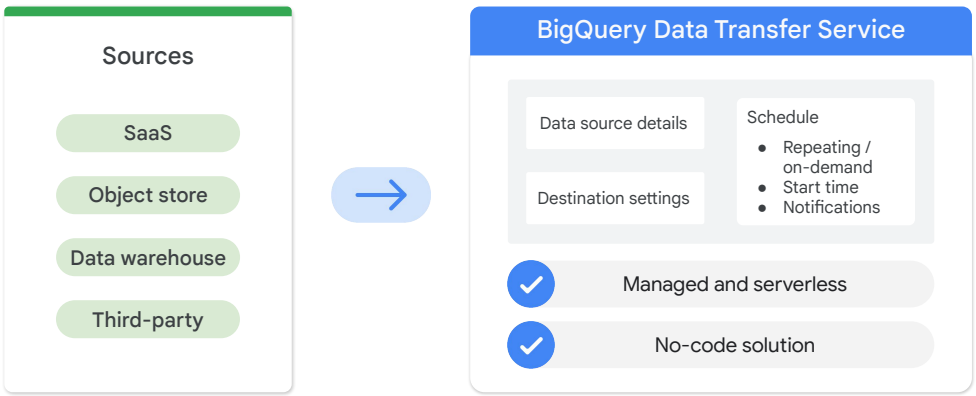
References:

https://cloud.google.com/bigquery/docs/datasets#bq
https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-csv#bq

Instructor notes:

- Students already observed the UI example in Lab 1 (Loading data into BigQuery: https://www.cloudskillsboost.google/catalog_lab/2119)

Use the BigQuery Data Transfer Service to load data from other structured data sources



The BigQuery Data Transfer Service enables seamless loading of structured data from diverse sources, like SaaS applications, object stores, and other data warehouses into BigQuery.

The service provides scheduling options for recurring or on-demand transfers, along with configuration options for data source details and destination settings.

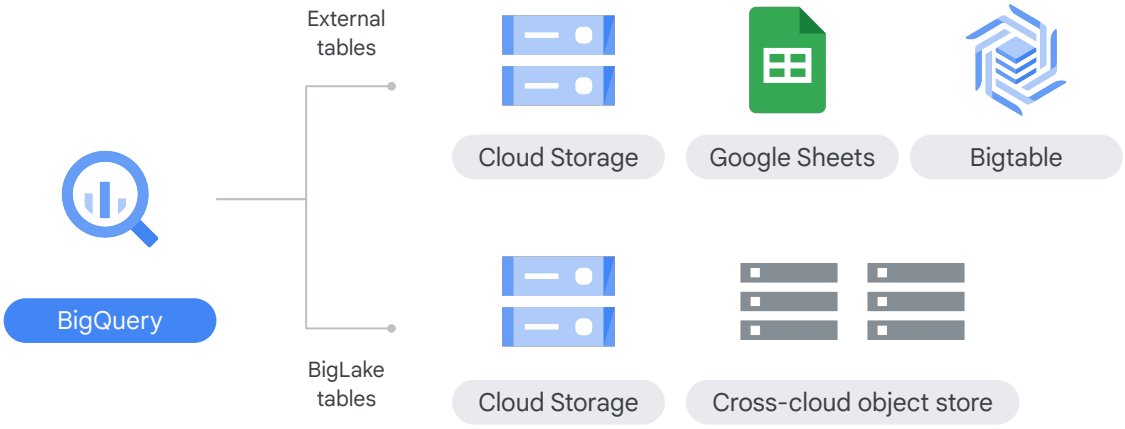
It is a managed and serverless solution, eliminating infrastructure management overhead.

In addition, its no-code approach simplifies data transfer setup and management.

References:

https://cloud.google.com/bigquery/docs/dts-locations

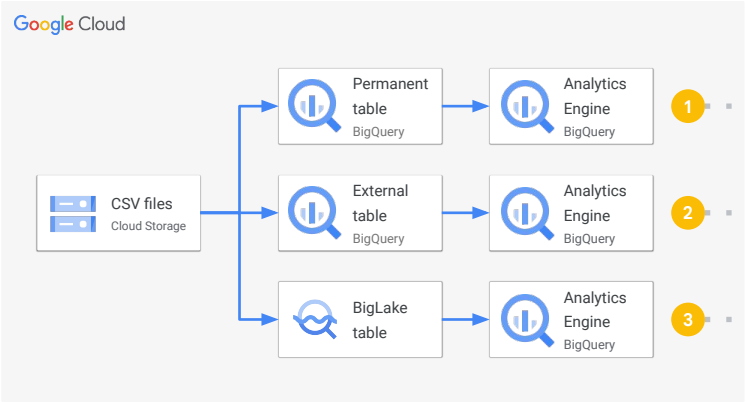
Your data isn't stored in BigQuery? You can still query it!



BigQuery's data access capabilities extend beyond its own storage. BigQuery allows you to query data residing in sources like Cloud Storage, Google Sheets, and Bigtable using external tables.

Additionally, BigLake tables provide a way to query data across Cloud Storage and even other cloud object stores, expanding BigQuery's reach and flexibility for data analysis.

Three ways of analyzing structured data in BigQuery

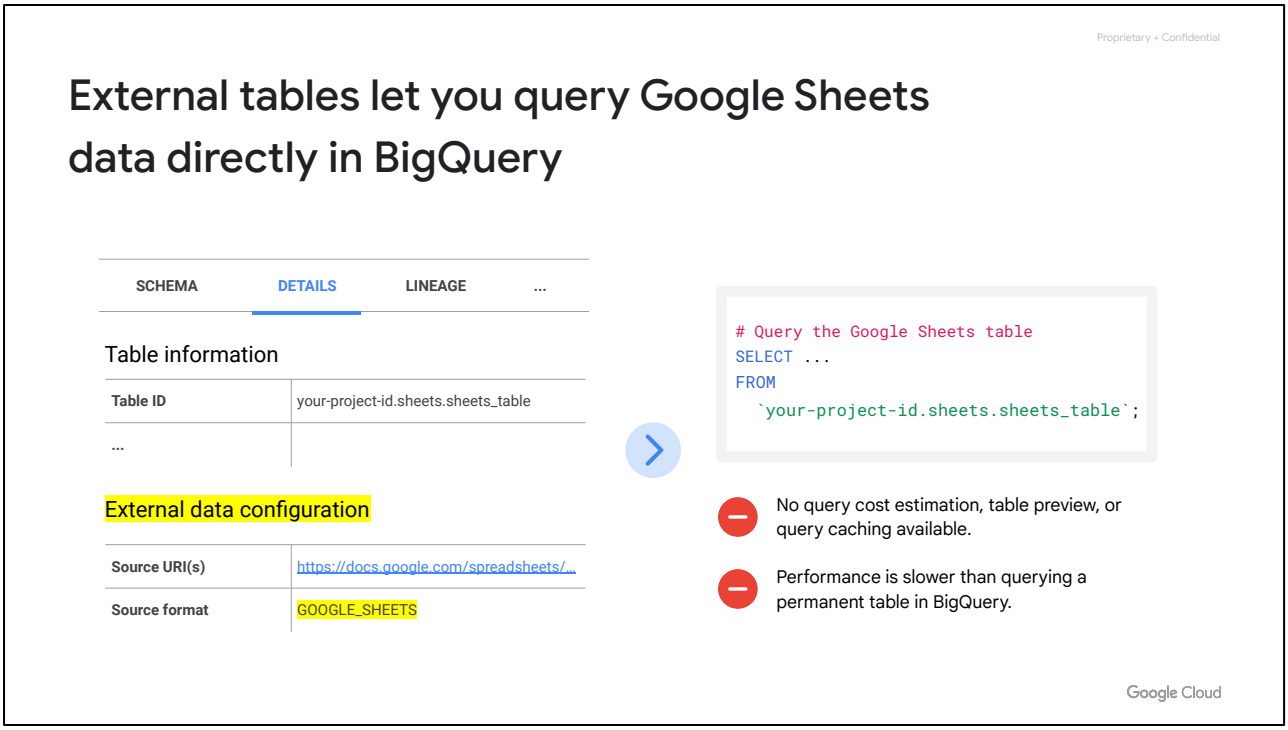


BigQuery offers flexibility in analyzing structured data.

You can load data into permanent BigQuery tables for high-performance analytics, but with data movement involved.

External tables allow you to query data directly in Cloud Storage without loading it into BigQuery, which is suitable for less frequent access.

BigLake tables provide the best of both worlds: high-performance analytics on data in Cloud Storage without the need to load it into BigQuery, and without data movement.



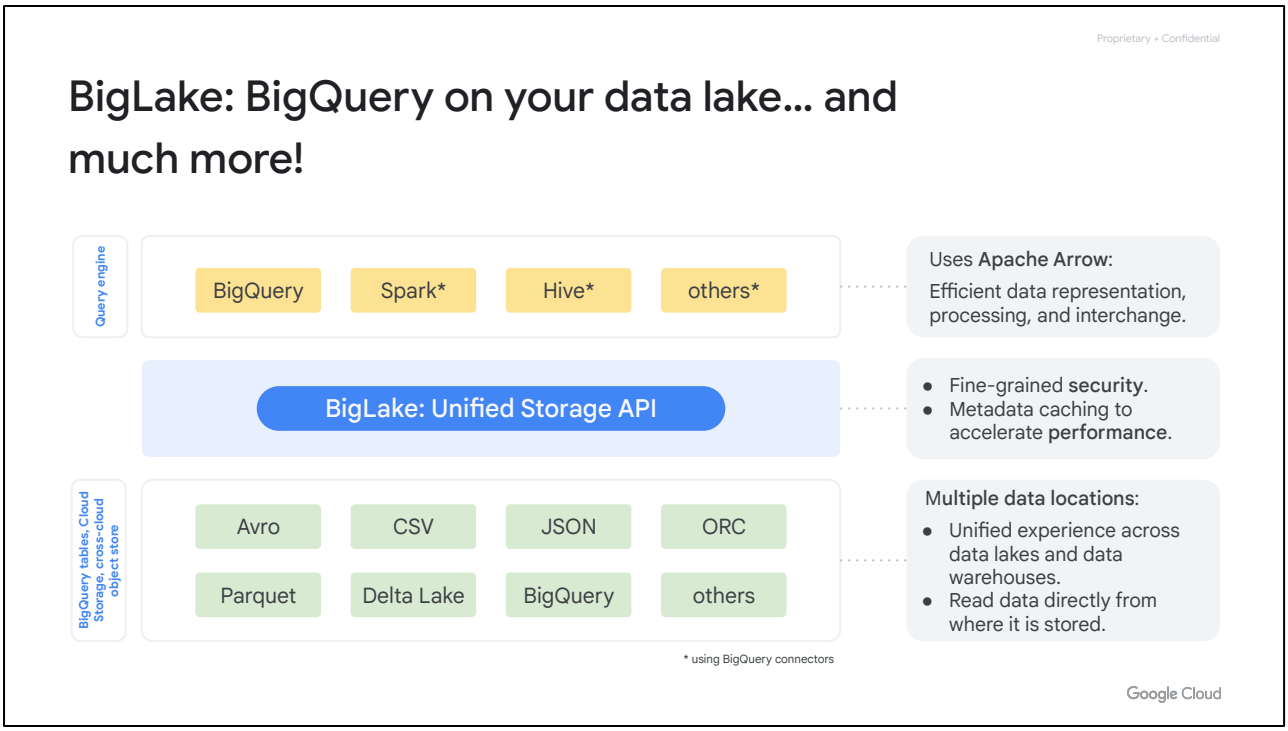
BigQuery external tables bridge the gap between Google Sheets and BigQuery, enabling direct querying of Sheets data within BigQuery.

By specifying the Google Sheets URL and format, users can treat the sheet as a table in BigQuery, simplifying data analysis across platforms.

However, be aware that querying external tables may have limitations, like slower performance and the unavailability of cost estimation, table preview, and query caching.

Instructor Notes:

You can optionally show a demo here.
Demo steps are in the guide here:
https://docs.google.com/document/d/1hLoeHfGAnyp6tyjH3LtHsVg2rB_PhGYme47-BLJ5KqQ/edit?resourcekey=0-92Txnxw9r6ZzTPll4pGEzw&tab=t.0#heading=h.e5ktnr3aro38



BigLake extends BigQuery's capabilities, providing a unified interface to query data directly from your data lake and other sources without moving or copying it.

BigLake leverages Apache Arrow for efficient data handling and offers fine-grained security and metadata caching.

With BigLake, you can seamlessly access data across data lakes and data warehouses using familiar BigQuery tools.

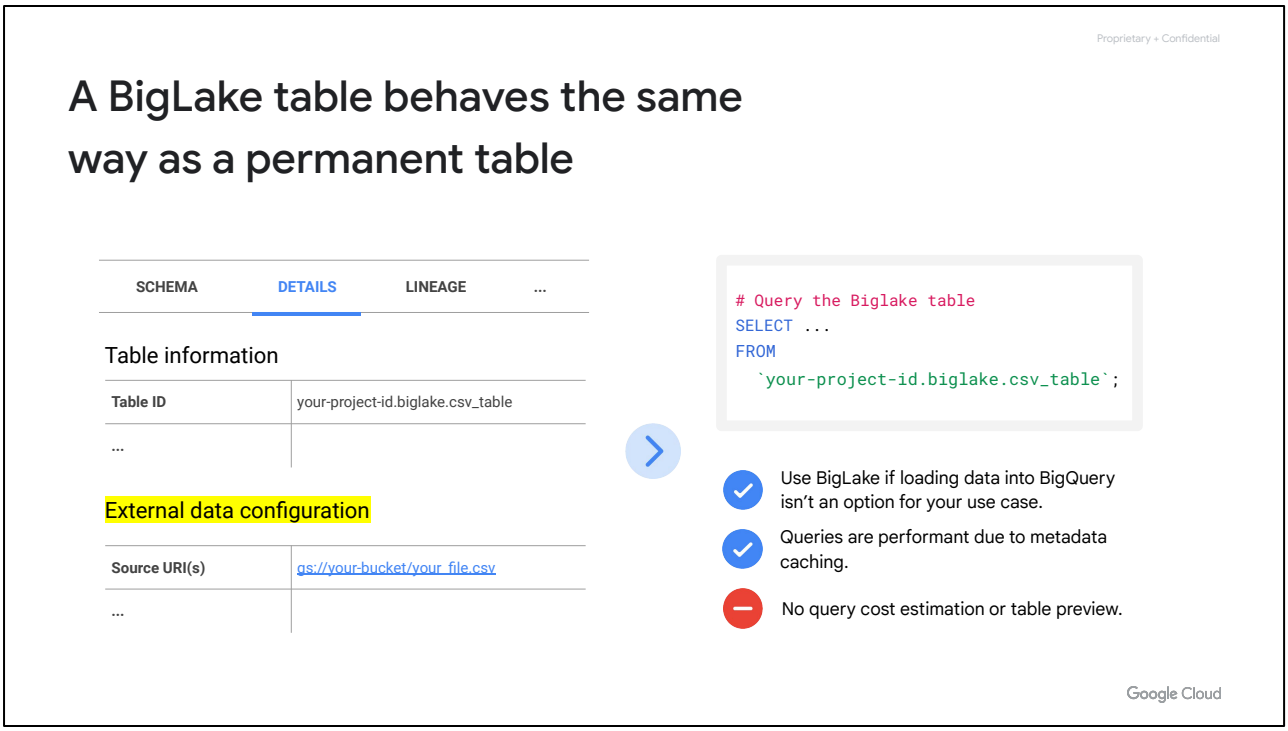
References:

https://cloud.google.com/bigquery/docs/biglake-intro#limitations
https://cloud.google.com/bigquery/docs/biglake-intro#connectors

Instructor notes:

BigLake is a storage layer / API that unifies access to different storage systems like BigQuery internal / managed storage, objects on Google Cloud Storage, and also AWS and Azure storage.

- motivation is that customers using Spark etc. wanted to query BQ data as well, but without moving/exporting it, so BigQuery Storage API was created
- now this API is extended to read data from other storage systems as well, not just BQ storage
- accessible through OSS tools
- will never copy or move data, data stays where it is
- create and work with tables pointing to other storage systems in BQ, have the same unified experience



BigLake tables provide a seamless querying experience, allowing you to interact with data stored in external sources like Cloud Storage just like you would with data in native BigQuery tables.

You can use standard SQL queries to access and analyze the data within BigLake tables, including `SELECT` statements and joins.

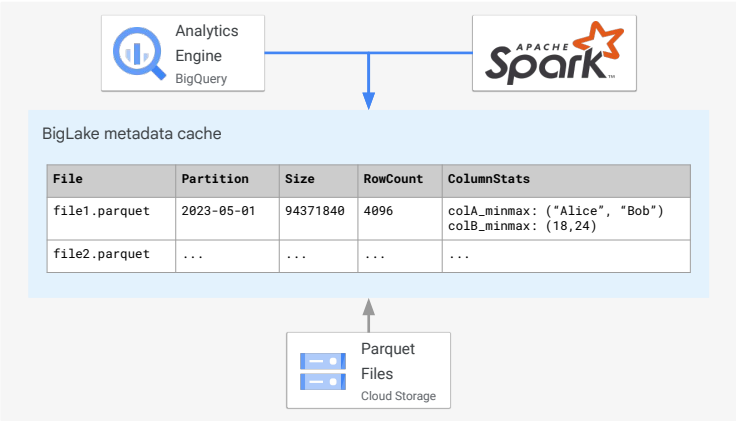
Behind the scenes, BigLake leverages metadata caching to enhance query performance, even though the data physically resides outside BigQuery.

However, some features like query cost estimation and table preview are not available for BigLake tables due to the external nature of the data.

Instructor Notes:

You can optionally show a demo here.
Demo steps are in the guide here:
https://docs.google.com/document/d/1hLoeHfGAnyp6tyjH3LtHsVg2rB_PhGYme47-BLJ5KqQ/edit?resourcekey=0-92Txnxw9r6ZzTPll4pGEzw&tab=t.0#heading=h.do0gj4ndllr9

BigLake uses metadata caching to accelerate performance



Advantages of BigQuery

- Skip object list
- Faster file and partition pruning
- Dynamic predicate pruning

Advantages of Spark

- Can read metadata statistics using the spark-bigquery connector.

Cache properties

- Staleness between 30 min and 7 days
- Automatic or manual refresh

Google Cloud

BigLake maintains a metadata cache. The cache stores details about external data. For example, it can contain details about Parquet files stored in Cloud Storage, such as file size, row count, and column statistics like minimum/maximum values.

This cache allows querying via BigQuery to skip listing all objects, prune files, and partitions faster, and enable dynamic predicate pushdown, resulting in improved query performance.

The cache allows querying by Spark to access metadata statistics that the Spark-BigQuery connector can leverage to speed up queries.

The metadata cache has configurable staleness from 30 minutes to 7 days and it can be refreshed automatically or manually.

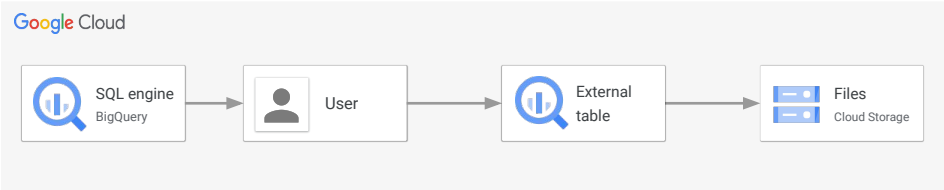
References:

https://cloud.google.com/blog/products/data-analytics/deep-dive-on-how-biglake-accelerates-query-performance
https://cloud.google.com/bigquery/docs/metadata-caching

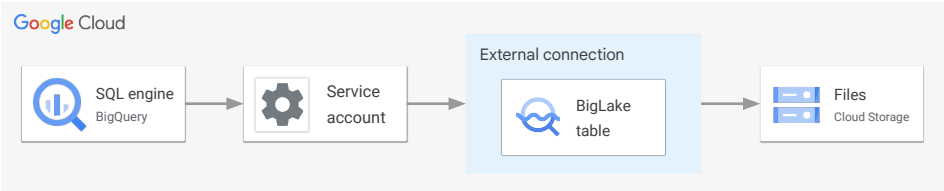
Instructor notes:

- partition pruning = skip files (partitions) not required, e.g. looking for DATE=2023-05-01, then skipping other partitions with different DATEs
- predicate pruning = filter data before reading it using metadata
- If you don't enable metadata caching, queries on the table must read the external data source to get object metadata which increases the query latency; listing millions of files from the external data source can take several minutes.

Compare security approaches: Non-native stored data



External tables: the user needs separate permissions to access the table and the data source.



BigLake: access is delegated using a service account. It decouples access to the table from the underlying data source.

External tables in BigQuery require users to have separate permissions for both the table itself and the underlying data source This can lead to more complex access management.

BigLake tables offer a streamlined approach. Access is delegated through a service account, decoupling table access from the data source. This simplifies permission management and enhances security.

References:

https://cloud.google.com/bigquery/docs/biglake-intro#security_model
https://cloud.google.com/bigquery/docs/create-cloud-storage-table-biglake#set_up_access_control_policies

Instructor notes:

External table:

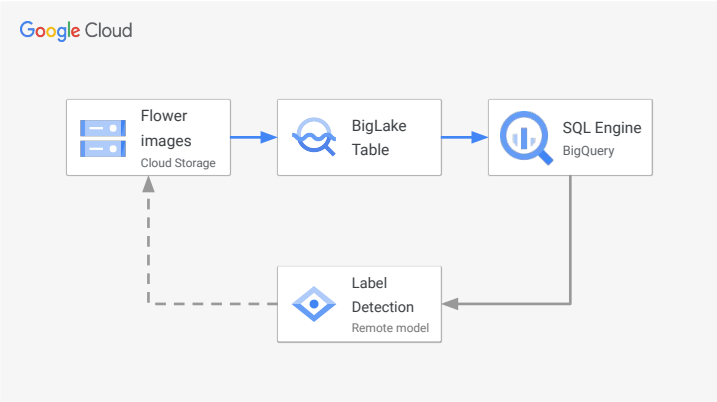
- User needs to have access to underlying data source, meaning no fine-grained security possible, since access e.g. to Cloud Storage object means being able to read the entire object or not, or e.g. Google Sheets read

- in the entire sheet or not.

BigLake Access delegation

- Decouples access to BigLake table from underlying data store, enables fine-grained security at table level.
- An external connection associated with a service account is used to connect to the data source. User needs to have access to External Connection, but not to underlying data source.
- Service Account needs to have access to underlying data store.

[Demo] Analyze an object table by using a remote model



```
gs://biglake-flowers/flower_photos/
        daisy/pic.jpg
```

```
"label_annotations": [{
  "description": "Flower",
  "score": 0.98062
}]
```


Instructor Notes:

Demo steps are in the guide here: https://docs.google.com/document/d/1hLoeHfGAnyp6tyjH3LtHsVg2rB_PhGYme47-BLJ5KqQ/edit?resourcekey=0-92Txnxw9r6ZzTPll4pGEzw&tab=t.0#heading=h.ke2n4yn06b26

Compare external and BigLake tables

	External tables	BigLake tables
Query Engine	BigQuery	BigQuery, others using BigQuery connector
Data types	• Parquet, Avro, JSON, ORC, CSV • Datastore export, Firestore export • Google Sheets	• Parquet, Avro, JSON, ORC, CSV • Apache Iceberg, Delta, Hudi • Hive partitioned • Any unstructured (via object tables)
Locations	• Cloud Storage • Bigtable • Google Drive	• Cloud Storage • Amazon S3 • Azure Blob Storage
Access pattern	Direct link to the source. The user needs to have access to the source	External connection with access delegation. The service account has access to the source.
Security	None	• Column-level security, data masking • Row-level security

In summary, both external and BigLake tables enable querying data residing outside of BigQuery, but BigLake offers broader capabilities.

BigLake supports a wider range of data formats and storage locations, including object stores across multiple cloud providers, and provides advanced security features like column-level and row-level security.

External tables are simpler to set up, but lack fine-grained security controls.

BigLake tables offer enhanced performance, security, and flexibility for querying external data, making them suitable for enterprise data lake use cases.

References:

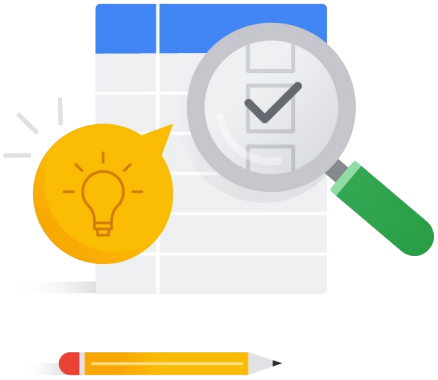
https://cloud.google.com/bigquery/docs/external-data-sources#external_data_source_feature_comparison

Lab: BigLake: Qwik Start

45 min

Learning objectives

- Create and view a connection resource.
- Set up access to a Cloud Storage data lake.
- Create a BigLake table.
- Query a BigLake table through BigQuery.
- Set up access control policies.
- Upgrade an external table to be a BigLake table.



In this lab, you use BigLake to connect to various external data sources.

You configure a connection resource and set up access to a Cloud Storage data lake.

You create and query a BigLake table and set up access control policies.

Finally, you upgrade an existing external table to be a BigLake table.

Lab URL: https://www.cloudskillsboost.google/catalog_lab/4896


Google Cloud