# Overview on Cloud BigQuery

- Storing and querying massive datasets can be time consuming and expensive without the right hardware and infrastructure.
- BigQuery is an enterprise data warehouse that solves this problem by enabling super-fast SQL queries using the processing power of Google's infrastructure.
- BigQuery is Google Cloud's fully managed, petabyte-scale and cost-effective analytics data warehouse that lets run analytics over vast amounts of data in near real time.
- BigQuery allows you to focus on analyzing data to find meaningful insights.
- BigQuery is a fully managed enterprise data warehouse that helps you manage and analyze your data with built-in features like machine learning, geospatial analysis, and business intelligence.
- BigQuery's serverless architecture lets you use SQL queries to answer your organization's biggest questions with zero infrastructure management.
- With BigQuery, there's no infrastructure to set up or manage, letting focus on finding meaningful insights using GoogleSQL.
    - There’s no need to build, deploy, or provision clusters.
    - No need to size VMs, storage, or hardware resources.
    - No need to setup disks, define replication, configure compression and encryption, or any other setup or configuration work
- BigQuery combines a cloud-based data warehouse and powerful analytic tools.

- Google BigQuery is built using BigTable on Google Cloud Platform.
- BigQuery is Google's fully managed, NoOps, low cost analytics database. It works great with all sizes of data, from a 100 row Excel spreadsheet to several Petabytes of data. You query terabytes in seconds and petabytes of data in minutes without having any infrastructure to manage or needing a database administrator.
- BigQuery uses SQL and can take advantage of the pay-as-you-go model.
- BigQuery is not solution for Transactional Data Operations. It's ideal for is for Analytical data operations.
- Query using
    - Standard SQL
    - legacy SQL
- BigQuery is super-fast and execute search on million of rows in seconds.
- BigQuery is great alternative of Apache Hive used in analytics.
- Apache Hive is a distributed, fault-tolerant data warehouse system that enables analytics at a massive scale and facilitates reading, writing, and managing petabytes of data residing in distributed storage using SQL.
    - Apache Hive is built on top of Apache Hadoop and supports storage on S3, ADLS, Google Storage etc though HDFS.
- BigQuery can load data from various sources.
    - CSV, JSON, Avro, SQL and many more
- Big Query can query from external data source.
    - Cloud storage, Cloud SQL, Big Table etc.,
- You access BigQuery through

    - Cloud Console
        - Using BigQuery in the Cloud Console will give you a visual interface to complete tasks like running queries, loading data, and exporting data.
    - **Bq** – command line tool
    - BigQuery RestAPI
    - Client library
        - written in C#, Go, Java, Node.js, PHP, **Python**, and Ruby
- There are also a variety of third-party tools that you can use to interact with BigQuery, such as visualizing the data or loading the data.
- BigQuery presents data in tables, rows, and columns and provides full support for database transaction semantics (ACID).
- BigQuery storage is automatically replicated across multiple locations to provide high availability.
- You can query data stored in BigQuery or run queries on data where it lives using external tables or federated queries including Cloud Storage, Bigtable, Spanner, or Google Sheets stored in Google Drive.
- Powerful tools like BigQuery ML and BI Engine let you analyze and understand that data.
- BigQuery can be used for **Storage, Analytics** and **Administration purposes**.
- BigQuery stores data using a **columnar storage** format that is optimized for analytical queries.
- BigQuery is Non-RDBMS column base DataBase in Google Cloud Infra. But like Relational database with SQL schema.

- Data analysts and data scientists can quickly query and filter large datasets, aggregate results, and perform complex operations without having to worry about setting up and managing servers.
- It comes in the form of a command line tool (pre installed in cloudshell) or a web console—both ready for managing and querying data housed in Google Cloud projects.

## Cloud BigQuery Hierarchy

BigQuery is structured as a hierarchy with 4 levels

### *Projects*

- Projects are the top-level containers that store the data
- Within the project, you can configure settings, permissions, and other metadata that describe your applications
- Each project has a name, ID, and number that you'll use as identifiers .
    - When billing is enabled, each project is associated with one billing account but multiple projects can be billed to the same account

### *Dataset*

- Dataset hold multiple tables, Within projects.
- Datasets allow you to organise and control access to your tables
- All tables must belong to a dataset. You must create a dataset before loading data into BigQuery

- You can configure permissions at the organization, project, and dataset level.

### *Table*

- Each table must belong to dataset that hold actual data. It has Schema.
- Tables contains data in BigQuery
- Each table has a schema that describes the data contained in the table, including field names, types, and descriptions
- BigQuery supports the following table types:
    - **Native tables -** tables backed by native BigQuery storage
    - **External tables -** tables backed by storage external to BigQuery
    - **Views -** A view is a virtual table defined by a SQL query.
    - **Materialized Views -** materialized views are precomputed views that periodically cache the results of a query for increased performance and efficiency.

### *Jobs*

- manage asynchronous tasks, You can run multiple jobs concurrently.
- Jobs are objects that manage asynchronous tasks such as running queries, loading data, and exporting data
- Completed jobs are listed in the Jobs collection
- There are four types of jobs:
    - **Load -** load data into a table

    - **Query -** run a query against BigQuery data
    - **Extract -** export a BigQuery table to Google Cloud Storage
    - **Copy -** copy an existing table into another new or existing table
- Assign Role at the organization, project, and dataset level.

## Features of Cloud Bigquery

The major user-facing components

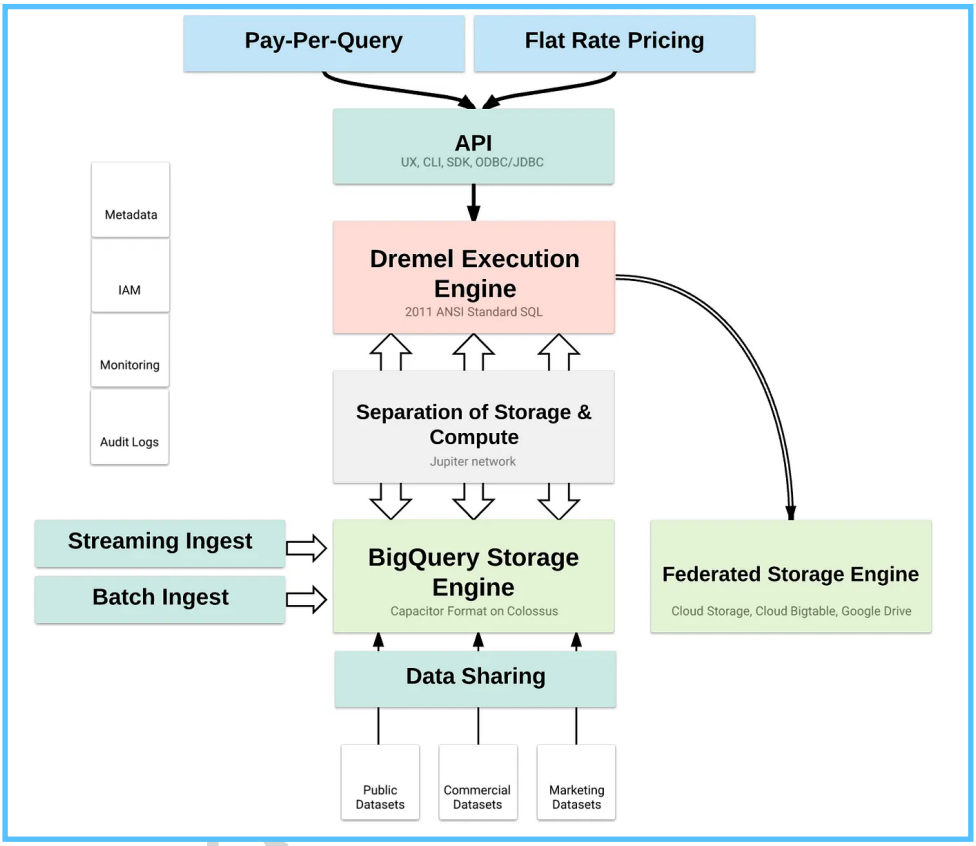
- Serverless Service Model
- Opinionated Storage Engine
- Dremel Execution Engine & Standard SQL
- Separation of Storage and Compute through Jupiter Network
- Enterprise-grade Data Sharing
- Public Datasets, Commercial Datasets, Marketing Datasets Free Pricing Tier
- Streaming Ingest
- Batch Ingest
- Federated Query Engine
- UX, CLI, SDK, ODBC/JDBC, API
- Pay-Per-Query AND Flat Rate Pricing
- IAM, Authentication & Audit Logs

### Serverless Insight

- BigQuery has a serverless architecture, which allows user to scale your analytics automatically.

## *Real-time analytics*

- BigQuery machine learning offers real-time analytics. This has a high-speed streaming insertion API.
- With real-time analytics, you can input your latest business data and analyze it immediately.




## *Logical data warehousing*

- User can process external data sources through BigQuery.
- It's a great way to help you to input all your data and process it without every duplicating your data.

## *Data transfer services*

- Service allow user to transfer your data from external sources automatically. Great tool to merge data from multiple tools at one place.

- Can transfer data from Google Marketing Platform, Google Ads, YouTube, Partner SaaS applications to BigQuery, Teradata, Amazon S3

### *High Availability*

- User can have multiple locations of storage and high availability for those locations.

### *Geo expansion*

- BigQuery machine learning gives you the option to control your geographic data. It only applies to the US, Asia, and Europe.

### *Automatic Backup and Easy Restore*

- Data is money, user don't want to lose it while trying to process it. With BigQuery, information is automatically replicated and stored.

## Efficient use of Bigquery admin panel

In BigQuery, you can run two types of queries:

- ***Interactive query jobs***, which are jobs that BigQuery runs on demand.
- ***Batch query jobs***, which are jobs that BigQuery waits to run until idle compute resources are available.

### *Using cached query results*

- BigQuery writes all query results to a table. The table is either explicitly identified by the user (a destination table), or it is a temporary, cached results table.

- If you run the exact same query again, BigQuery returns the results from the cached table, if it exists.
- Temporary, cached results tables are maintained per-user, per-project.
- There are no storage costs for cached query result tables, but if you write query results to a permanent table, you are charged for storing the data.
- All query results, including both interactive and batch queries, are cached in temporary tables for approximately 24 hours.

### *Limitations of Caching*

Using the query cache is subject to the following limitations:

- When you run a duplicate query, the query text must be the same as the original query.
- For query results to persist in a cached results table, the result set must be smaller than the maximum response size.
- You cannot target cached result tables with DML statements.
- Not cached for having streaming ingestion.
- Not cached if query runs against external sources like Cloud SQL or Storage.
- Not cached if used non-deterministic functions.
    - NOW()
    - CURRENT_TIMESTAMP()
    - CURRENT_USER()

### *Maximum bytes billed*

- Limits the bytes billed for this query. If this query will have bytes billed beyond this limit, the query will be failed (without incurring a charge).
- If not specified, the bytes billed will be set to the project default.

### *Session management*

- If you would like to capture a group of your SQL activities, create a BigQuery session.
- After creating a session, you can run interactive queries in your session until the session terminates.
- All queries in the session are run (processed) in the location where the session was created.

### *Schedule, Save and Share query*

- Schedule queries to run on a recurring basis.
- Scheduled queries must be written in GoogleSQL, which can include data definition language (DDL) and data manipulation language (DML) statements.
- When you create or update the schedule for a query, the scheduled time for the query is converted from your local time to UTC.
    - UTC is not affected by daylight saving time.
- Scheduled queries use features of BigQuery Data Transfer Service.

# Partitions and Clusters in BigQuery

- Partitioning and clustering are two very effective techniques to minimize query costs and increase performance (using fewer resources while improving speed).
- Limit the amount of data that needs to be read when running a query.
- We can use both techniques independently or combined to get optimal results.

## *Partitions*



- Table partitioning is a technique for splitting large tables into smaller ones. Here's an example of converting a classic table to a table partitioned by date.

Classic Table

| date | user_id | country | name |
|---|---|---|---|
| 2021-10 | 2 | US | Stephan |
| 2021-08 | 3 | FR | Carlo |
| 2017-05 | 6 | US | Douglas |
| 2021-10 | 1 | FR | Romain |
| 2021-08 | 4 | IT | Laure |
| 2021-08 | 5 | GB | Laura |
| 2017-05 | 1 | FR | Romain |

→

Partitioned Table

| date | user_id | country | name |
|---|---|---|---|
| 2017-05 | 1 | FR | Romain |
| 2017-05 | 6 | US | Douglas |

| date | user_id | country | name |
|---|---|---|---|
| 2021-08 | 4 | IT | Laure |
| 2021-08 | 3 | FR | Carlo |
| 2021-08 | 5 | GB | Laura |

| date | user_id | country | name |
|---|---|---|---|
| 2021-10 | 1 | FR | Romain |
| 2021-10 | 2 | US | Stephan |

- BigQuery will store separately the different partitions at a physical level (meaning the data will be stored on different servers).

- When you partition a table and then execute a query, it is BigQuery that determines which partition to access and minimizes the data that must be read.
- You partition tables by specifying a partition column which is used to segment the table.
- You can create a partitioned table based on a column, also known as a partitioning key.
- In BigQuery, you can partition your table using different keys:
  - **Time-unit column -** Tables are partitioned based on a time value such as timestamps or dates.
  - **Ingestion time -** Tables are partitioned based on the timestamp when BigQuery ingests the data.
  - **Integer range -** Tables are partitioned based on a number.
- BigQuery has a limit of 4,000 partitions per table.

### *Ingestion time partitioning*

- When you create a table partitioned by ingestion time, BigQuery automatically assigns rows to partitions based on the time when BigQuery ingests the data.
- You can choose hourly, daily, monthly, or yearly granularity for the partitions.
  - **Daily partitioning** is the default partitioning type.
  - **Hourly partitioning** if your tables have a high volume of data that spans a short date range.

- **Monthly or yearly partitioning** if your tables have a relatively small amount of data for each day, but span a wide date range.

- Partitions boundaries are based on UTC time.

### *Time-unit column partitioning*

- You can partition a table on a **DATE,TIMESTAMP, or DATETIME** column in the table.
- For **TIMESTAMP and DATETIME** columns, the partitions can have either hourly, daily, monthly, or yearly granularity.
- For **DATE** columns, the partitions can have daily, monthly, or yearly granularity.
- Partitions boundaries are based on UTC time.

| Column value | Partition (monthly) |
|---|---|
| `DATETIME("2019-01-01")` | `201901` |
| `DATETIME("2019-01-15")` | `201901` |
| `DATETIME("2019-04-30")` | `201904` |

- In addition, two special partitions are created:
  - **__NULL__**: Contains rows with NULL values in the partitioning column.
  - **__UNPARTITIONED__**: Contains rows where the value of the partitioning column is earlier than 1960-01-01 or later than 2159-12-31.

### *Integer range partitioning*

- You can partition a table based on ranges of values in a specific INTEGER column.
- To create an integer-range partitioned table, you provide:
  - The partitioning column.
  - The starting value for range partitioning (inclusive).

| Argument | Value |
|---|---|
| column name | `customer_id` |
| start | 0 |
| end | 100 |
| interval | 10 |

  - The ending value for range partitioning (exclusive).
  - The interval of each range within the partition.
- The mode of the column can be required or nullable but it cannot be repeated.

# Clustered Tables in BigQuery

- Clustering in BigQuery will help us to keep data that is similar closer together, allowing a query to scan fewer data.
- Clustered tables in BigQuery have a **user-defined column** sort order using clustered columns, which sorts storage blocks based on the values.
- It can improve query performance and reduce query costs.

**Clustered Tables**

Orders table
Not clustered

| Order_Date | Country | Status |
|---|---|---|
| 2022-08-02 | US | Shipped |
| 2022-08-04 | JP | Shipped |
| 2022-08-05 | UK | Canceled |
| 2022-08-06 | KE | Shipped |
| 2022-08-02 | KE | Canceled |
| 2022-08-05 | US | Processing |
| 2022-08-04 | JP | Processing |
| 2022-08-04 | KE | Shipped |
| 2022-08-06 | UK | Canceled |
| 2022-08-02 | UK | Processing |
| 2022-08-05 | JP | Canceled |
| 2022-08-06 | UK | Processing |
| 2022-08-05 | US | Shipped |
| 2022-08-06 | JP | Processing |
| 2022-08-02 | KE | Shipped |
| 2022-08-04 | US | Shipped |

Orders table
Clustered by Country

| Order_Date | Country | Status |
|---|---|---|
| 2022-08-02 | US | Shipped |
| 2022-08-05 | US | Shipped |
| 2022-08-05 | US | Processing |
| 2022-08-04 | US | Shipped |
| 2022-08-04 | JP | Shipped |
| 2022-08-04 | JP | Processing |
| 2022-08-05 | JP | Canceled |
| 2022-08-06 | JP | Processing |
| 2022-08-05 | UK | Canceled |
| 2022-08-06 | UK | Canceled |
| 2022-08-06 | UK | Processing |
| 2022-08-02 | UK | Processing |
| 2022-08-06 | KE | Shipped |
| 2022-08-02 | KE | Canceled |
| 2022-08-04 | KE | Shipped |
| 2022-08-02 | KE | Shipped |

Orders table
Clustered by Country and Status

| Order_Date | Country | Status |
|---|---|---|
| 2022-08-05 | US | Processing |
| 2022-08-02 | US | Shipped |
| 2022-08-05 | US | Shipped |
| 2022-08-04 | US | Shipped |
| 2022-08-05 | JP | Canceled |
| 2022-08-04 | JP | Processing |
| 2022-08-06 | JP | Processing |
| 2022-08-04 | JP | Shipped |
| 2022-08-05 | UK | Canceled |
| 2022-08-06 | UK | Canceled |
| 2022-08-06 | UK | Processing |
| 2022-08-02 | UK | Processing |
| 2022-08-02 | KE | Canceled |
| 2022-08-06 | KE | Shipped |
| 2022-08-04 | KE | Shipped |
| 2022-08-02 | KE | Shipped |

## When to use clustering

- If your queries commonly **filter on particular columns** and have many distinct values (high cardinality).

## Cluster column types

Cluster columns must be top-level, non-repeated columns that are one of the following types:

- STRING

- INT64, NUMERIC and BIGNUMERIC
- DATE, DATETIME and TIMESTAMP
- BOOL
- GEOGRAPHY

## Cluster column ordering

- The query filter order must match the clustered column order or else it affects query performance.

## Limitations

- Only GoogleSQL is supported for querying clustered tables and for writing query results to clustered tables.
- You can only specify up to four clustering columns.
- When using STRING type columns for clustering, BigQuery uses only the first 1,024 characters to cluster the data.
- If you alter an existing non-clustered table to be clustered, the existing data is not clustered. Only new data is stored using the clustered columns and is subject to automatic reclustering.

# When to use Partitioning or clustering or both

## *When to use partitioning*

Consider partitioning a table in the following scenarios:

- To improve the query performance by only scanning a portion of a table.
- Your table operation exceeds a standard table quota.
- To determine query costs before a query runs.
- Partitioning your table results in an average partition size of at least 10 GB per partition.
- Maximum number of allowed partitions are 4000.
- Max number of partitions allowed by a single job is also 4000.
- No need of using additional filters on where clause using AND or OR operator.
- Max number of partitions modified per day is
  - Ingestion time - 5000
  - Column partitioned - 30,000
- Wanted any of the following partition-level management features:
  - Set a partition expiration time
  - Write data to a specific partition
  - Delete specific partitions

## *When to use clustering*

Clustering addresses how a table is **stored** so it's generally a good first option for improving query performance.

- If your queries commonly filter on particular columns.
- You need more granularity than partitioning allows.
- Your queries commonly use filters or aggregation against multiple columns.
- The cardinality of the number of values in a column or group of columns is large.
- You don't need strict cost estimates before query execution.
- Partitioning results in a small amount of data per partition (approximately less than 10 GB). Creating many small partitions increases the table's metadata, and can affect metadata access times when querying the table.
- Your DML operations frequently modify most partitions in the table.

### *Combining clustered and partitioned tables*

- First segment data into partitions, and then you cluster the data within each partition by the clustering columns.
- Partitioning enables each partition to be considered separately for eligibility for long term pricing.
- Need a strict query cost estimate before you run a query.
- Vol of data is very huge within partition.
- When you create a table that is clustered and partitioned, you can achieve more **finely grained sorting**, as the following diagram shows:

**Clustered and Partitioned Tables**

Orders table
Not Clustered; Not partitioned

| Order_Date | Country | Status |
|---|---|---|
| 2022-08-02 | US | Shipped |
| 2022-08-04 | JP | Shipped |
| 2022-08-05 | UK | Canceled |
| 2022-08-06 | KE | Shipped |
| 2022-08-02 | KE | Canceled |
| 2022-08-05 | US | Processing |
| 2022-08-04 | JP | Processing |
| 2022-08-04 | KE | Shipped |
| 2022-08-06 | UK | Canceled |
| 2022-08-02 | UK | Processing |
| 2022-08-05 | JP | Canceled |
| 2022-08-06 | UK | Processing |
| 2022-08-05 | US | Shipped |
| 2022-08-06 | JP | Processing |
| 2022-08-02 | KE | Shipped |
| 2022-08-04 | US | Shipped |

Orders table
Clustered by Country; Not partitioned

| Order_Date | Country | Status |
|---|---|---|
| 2022-08-04 | JP | Shipped |
| 2022-08-04 | JP | Processing |
| 2022-08-05 | JP | Canceled |
| 2022-08-06 | JP | Processing |
| 2022-08-06 | KE | Shipped |
| 2022-08-02 | KE | Canceled |
| 2022-08-04 | KE | Shipped |
| 2022-08-02 | KE | Shipped |
| 2022-08-05 | UK | Processing |
| 2022-08-06 | UK | Canceled |
| 2022-08-02 | UK | Canceled |
| 2022-08-06 | UK | Processing |
| 2022-08-02 | US | Shipped |
| 2022-08-05 | US | Processing |
| 2022-08-05 | US | Shipped |
| 2022-08-04 | US | Shipped |

Orders table
Clustered by Country; Partitioned by Order_Date (Daily)

| | Order_Date | Country | Status |
|---|---|---|---|
| **Partition:** 2022-08-02 | 2022-08-02 | KE | Shipped |
| | 2022-08-02 | KE | Canceled |
| **Clusters:** Country | 2022-08-02 | UK | Processing |
| | 2022-08-02 | US | Shipped |

| | Order_Date | Country | Status |
|---|---|---|---|
| **Partition:** 2022-08-04 | 2022-08-04 | JP | Shipped |
| | 2022-08-04 | JP | Processing |
| **Cluster:** Country | 2022-08-04 | KE | Shipped |
| | 2022-08-04 | US | Shipped |

| | Order_Date | Country | Status |
|---|---|---|---|
| **Partition:** 2022-08-05 | 2022-08-05 | JP | Canceled |
| | 2022-08-05 | UK | Canceled |
| **Cluster:** Country | 2022-08-05 | US | Shipped |
| | 2022-08-05 | US | Processing |

| | Order_Date | Country | Status |
|---|---|---|---|
| **Partition:** 2022-08-06 | 2022-08-06 | JP | Processing |
| | 2022-08-06 | KE | Shipped |
| **Cluster:** Country | 2022-08-06 | UK | Canceled |
| | 2022-08-06 | UK | Processing |

## External Data sources in BigQuery

- An external data source is a data source that you can query directly from BigQuery but the data is not stored in BigQuery storage.
- For example,
  - Data in a different Google Cloud database,
  - Files in Cloud Storage,
  - In a different cloud product altogether that you would like to analyze in BigQuery, but that you aren't prepared to migrate.
- BigQuery has two different mechanisms for querying external data:
  - External tables
  - Federated queries.

## External tables

- External tables are similar to standard BigQuery tables, in that these tables store their metadata and schema in BigQuery storage.
- However, their data resides in an external source.External tables are contained inside a dataset, and you manage them in the same way as a standard BigQuery table.
- There are three kinds of external tables:
    - BigLake tables
    - Object tables
    - Non-BigLake external tables

## BigLake tables

- BigLake tables let you query structured data in external data stores with access delegation.
- An external connection associated with a service account is used to connect to the data store.

## Object tables

- Object tables let you analyze unstructured data in Cloud Storage.
- An external connection associated with a service account is used to connect to Cloud Storage.

### *Non-BigLake external tables*

- Non-BigLake external tables let you query structured data which is present in external table and external data source.
- You can use non-BigLake external tables with the following data stores:
    - Cloud Bigtable
    - Cloud Storage
    - Google Drive

### *Federated queries*

- Federated queries let you send a query statement to Cloud Spanner or Cloud SQL databases and get the result back as a temporary table.
- Federated queries use the BigQuery Connection API to establish a connection with Spanner or Cloud SQL.

### *BigQuery Omni*

- With BigQuery Omni, you can run BigQuery analytics on data stored in Amazon Simple Storage Service (Amazon S3) or Azure Blob Storage using BigLake tables.
- Many organizations store data in multiple public clouds.

### *Limitations*

- BigQuery does not guarantee **data consistency** for external data tables.
- Query performance for external tables might be **slow** compared to querying data in a standard BigQuery table.

- **Cannot modify** external data tables using DML or other methods. External tables are read-only for BigQuery.
- Cannot reference an external table in a wildcard table query.
- External tables don't support clustering. They support partitioning in limited ways.
- BigQuery doesn't support the display of table storage statistics for external tables.
- Cannot run a BigQuery job that exports data from an external table.

## Big Query Pricing

- BigQuery is a serverless data analytics platform. You don't need to provision individual instances or virtual machines to use BigQuery.
- Instead, BigQuery automatically allocates computing resources as you need them.
- But You can reserve compute capacity ahead of time in the form of slots, which represent virtual CPUs.
- BigQuery pricing has two main components:
    - **Analysis pricing** is the cost to process queries, including SQL queries, user-defined functions, scripts, and certain data manipulation language (DML) and data definition language (DDL) statements that scan tables.
    - **Storage pricing** is the cost to store data that you load into BigQuery.
- **Analysis pricing models**
    - BigQuery offers a choice of two pricing models for running queries:

- **On-demand pricing**. With this pricing model, you are charged for the number of bytes processed by each query. The first 1 TB of query data processed per month is free.
- **Flat-rate pricing**. With this pricing model, you purchase slots, which are virtual CPUs. When you buy slots, you are buying dedicated processing capacity that you can use to run queries. Slots are available in the following commitment plans:
  - **Flex slots -** You commit to an initial 60 seconds.
  - **Monthly -** You commit to an initial 30 days.
  - **Annual -** You commit to 365 days.
- With monthly and annual plans, you receive a lower price in exchange for a longer-term capacity commitment.

### *Google BigQuery Pricing Model*

| Category | Price | Note |
|---|---|---|
| **Storage Cost** | $0.020 per GB per month | |
| **Query Cost** | $5 per TB | 1st TB per month is free |



Free usage is available for the below operations:

- Loading data (network pricing policy applicable in case of inter-region).
- Copying data.
- Exporting data.

- Deleting datasets.
- Metadata operations.
- Deleting tables, views, and partitions.

## *Big Query Slots*

- A BigQuery slot is a **unit of computational capacity** used to execute SQL, DDL, and DML statements in BigQuery.
- Pay-per-query pricing users may choose to take advantage of flat-rate pricing by buying BigQuery slot commitments.
- You can take advantage of BigQuery flat-rate by taking the following actions:
  - Purchase a commitment via 'Buy Slots'.
  - A 'default' reservation is created for you (optionally, you can create additional reservations).
- Assign your GCP projects, folders, or orgs to a reservation.
- Note - any projects/folders/orgs not assigned to a reservation will remain on on-demand billing.

## *Google BigQuery Storage Cost*

- **Active –** Monthly charge for stored data modified within 90 days.
- **Long-term –** Monthly charge for stored data that have not been modified within 90 days. This is usually lower than the earlier one.

## Optimization techniques in BigQuery

- Few best practices as part of cost optimization in bigquery are as follows.
  - Make use of **Caching**.
  - **Data minimization:** Avoid mindless data processing and only query the data you need.
  - **Partitioning:** Partition tables by date or other logical grouping to reduce the amount of data scanned during queries.
  - **Clustering:** Cluster tables by frequently accessed columns to reduce the amount of data scanned during queries.
  - **Joins:** Use INT64 data types in joins instead of STRING data types.
  - **Denormalization:** Use nested and repeated fields for hierarchical relationships that are frequently queried together.
  - **Operators:** Choose operators carefully and avoid non-cacheable results.
  - **Nested and repeated fields:** Use nested and repeated fields when relationships are hierarchical and frequently queried together.
  - **Preview the table:** Before querying the data from a table, check the size of the table and preview the table.
  - **Look at how much data is processed:** Before running a query, always look at how much data your query will process.

## Use Cases of Google BigQuery

You can use Google BigQuery Data Warehouse in the following cases:

- Use it when you have queries that run more than five seconds in a relational database.
- The idea of BigQuery is **running complex analytical queries**, which means there is no point in running queries that are doing simple aggregation or filtering.
- BigQuery is suitable for “heavy” queries, those that operate using a big set of data.
- BigQuery is good for scenarios where **data does not change often** and you want to use the cache, as it has a built-in cache.
- You can also use BigQuery when you want to reduce the load on your relational database.
    - Analytical queries are “heavy” and overusing them under a relational database can lead to performance issues.
    - So, you could eventually be forced to think about scaling your server.
- Not Suitable for Data Aggregation queries.
- BigQuery doesn’t like joins, so you should merge your data into one table to get better execution time.
- BigQuery will just use cached results and will not try to execute the query again. Also, BigQuery is not charging money for cached queries.

### *When BigQuery should be used*

- When workload is analytical.When Data doesn't change in database, as bigquery use built –in cache.
- For complex query - When query takes more execution time.
- Off-load some workload from primary transaction DB. When you have large volume of data.
- No Join is preferred.
- When your data is denormalized.

## Other services in Cloud BigQuery

### *BI Engine*

- BigQuery BI Engine is a fast, in-memory analysis service that accelerates many SQL queries in BigQuery by intelligently caching the data you use most frequently.
- BI Engine can accelerate SQL queries from any source, including those written by data visualization tools, and can manage cached tables for on-going optimization.
- This lets you improve query performance without manual tuning or data tiering.
- BI Engine natively integrates any BI tool, including Looker Studio, Connected Sheets, or any third-party tool that connects to BigQuery.
- BI Engine provides the following advantages:
  - **BigQuery API:**

    - BI Engine directly integrates with the BigQuery API. Any BI solution or custom application that works with the BigQuery API through standard mechanisms such as REST or JDBC and ODBC drivers can use BI Engine without modification.

- **Vectorized runtime:**
    - With the BI Engine SQL interface, BI Engine introduces a more modern technique called *vectorized processing*.
    - Using vectorized processing in an execution engine makes more efficient use of modern CPU architecture, by operating on batches of data at a time.
    - BI Engine also uses advanced data encodings, specifically, dictionary run-length encoding, to further compress the data that's stored in the in-memory layer.
- **Seamless integration:**
    - BI Engine works with BigQuery features and metadata, including authorized views, column level security, and data masking.
- **Reservations:**
    - BI Engine reservations manage memory allocation at the project location level.
    - BI Engine caches specific columns or partitions that are queried, prioritizing those in tables marked as preferred.

### *Analytics Hub*

- Exchange data and analytics assets securely and efficiently.

- Analytics Hub provides an easy way to publish, discover and subscribe to share BigQuery datasets between users in your organization or other organizations.
- Once you create an exchange, you can invite others to publish or subscribe to data in the exchange.

### *Data Transfer Service*

- The BigQuery Data Transfer Service enables you to move data into BigQuery from a variety of sources.
- Transfers can be both one-time, as well as ongoing or scheduled.

### *Scheduled Query Service*

- BigQuery Scheduled Queries enables you to repeatedly run queries on a set schedule.

### *Dataform*

- Dataform helps analytics teams transform data in BigQuery using SQL.
- With Dataform, data engineers and data analysts can develop BigQuery table definitions following software engineering best practices like version control and orchestrate table updates without requiring dedicated infrastructure.

### *BigQuery Partner Center*

- Discover tools and services from partners and accelerate your workflow.

### *BigQuery migration assessment*

- Assessment helps you plan your data warehouse migration to BigQuery with a detailed view of your existing system and easy steps to efficiently migrate to BigQuery.

### *BigQuery SQL translation*

- The BigQuery SQL translation tool provides translations between various supported SQL dialects.