**BIG QUERY**

The official BigQuery documentation is provided by Google Cloud and serves as a comprehensive guide to understanding and using BigQuery, a fully-managed, serverless data warehouse that enables scalable analysis over petabytes of data.

You can access the documentation here: [BigQuery Documentation](https://cloud.google.com/bigquery/docs)

**Key Sections in the Documentation:**

1. **Getting Started**:
   * Introduction to BigQuery.
   * Setting up your first project.
   * Running your first query.
2. **Core Concepts**:
   * Understanding datasets, tables, and schemas.
   * Querying data using SQL.
   * Managing storage and compute resources.
3. **Advanced Features**:
   * Partitioned and clustered tables.
   * User-defined functions (UDFs).
   * BigQuery ML for machine learning.
4. **Integration**:
   * Connecting BigQuery with other Google Cloud services.
   * Using APIs and client libraries.
   * Exporting and importing data.
5. **Pricing and Optimization**:
   * Understanding BigQuery's pricing model.
   * Best practices for cost optimization.
   * Query performance tuning.
6. **Security and Compliance**:
   * Managing access control with IAM.
   * Encryption and data protection.

**Introduction to BigQuery.**

1. **What Is BigQuery?**
   1. **BigQuery** is Google Cloud’s fully managed, serverless data warehouse designed for fast and scalable analytics. It allows you to run SQL queries on massive datasets—think terabytes in seconds and petabytes in minutes—without worrying about infrastructure.
2. **Key Features**
   1. **Serverless Architecture**: No need to manage servers or clusters. Google handles scaling and performance behind the scenes.
   2. **High-Speed Analytics**: Optimized for lightning-fast querying using columnar storage and distributed computing.
   3. **SQL-Based Interface**: Uses ANSI-compliant SQL, making it accessible to analysts and developers alike.
   4. **Real-Time Data Ingestion**: Supports streaming data for up-to-the-minute insights.
   5. **Built-In Machine Learning**: With BigQuery ML, you can train and deploy models directly using SQL.
   6. **Seamless Integration**: Works smoothly with other Google Cloud services like Cloud Storage, Dataflow, and Looker Studio.

### Why Use BigQuery?

### Ideal for analyzing large-scale structured and semi-structured data.

### Great for businesses needing real-time insights without the overhead of managing infrastructure.

### Scales automatically with your data and query needs.

### Offers flexible pricing models—pay-as-you-go or flat-rate.

**Core Concepts**:

1. **Understanding datasets, tables, and schemas.**
   * **Datasets**
     1. A **dataset** is a top-level container within a BigQuery project.
     2. It groups related tables and views, often by domain (e.g., sales\_data, marketing\_metrics).
     3. You can set access controls, labels, and geographic location at the dataset level.
     4. Every table or view must belong to a dataset.
     5. **Key Attributes of a Dataset**
        + **Dataset ID**: A unique name within your project (e.g., sales\_data\_2025).
        + **Location**: You choose a geographic region (like us-central1 or EU) when creating a dataset. This determines where your data is physically stored.
        + **Default Table Expiration**: Optional setting to automatically delete tables after a specified number of days.
        + **Encryption**: You can use Google-managed encryption or provide your own Customer-Managed Encryption Key (CMEK).
     6. **Access Control**
        + Datasets support **fine-grained IAM roles**:
          - BigQuery Data Viewer: Read-only access
          - BigQuery Data Editor: Can modify tables
          - BigQuery Admin: Full control
        + You can also share datasets with specific users or groups.
     7. **Data Retention & Time Travel**
        + BigQuery supports **time travel**, allowing you to access table data as it existed up to 7 days ago.
        + Datasets also have a **fail-safe period** for recovering deleted or modified data. But after time travel window ends, and it is also lasts for 7 days only but we need google customer support in this.
     8. **Storage Billing Models**
        + You can choose between:
          - **Logical storage billing**: Based on uncompressed data size. Useful for frequently updated datasets
          - **Physical storage billing**: Based on compressed data size. Useful for static datasets.
        + The billing model is set at the dataset level and can be changed (with a 14-day cool down).
        + BigQuery uses a columnar format called **Capacitor.**
     9. **External Datasets**
        + BigQuery supports **federated datasets** that link to external sources like:
          - Google Cloud Spanner
          - AWS Glue
        + These datasets don’t store data in BigQuery—they query it live from the source.
     10. **Limitations**
         + Dataset location is fixed once created.
         + All tables in a query must be in datasets located in the same region.
         + Dataset names must be unique within a project.
   * **Tables**
     1. A **table** is where your actual data lives—structured in rows and columns.
     2. Tables can be:
        + **Native** (stored in BigQuery)
        + **External** (linked from sources like Cloud Storage)
     3. BigQuery supports advanced table types like **partitioned** and **clustered** tables for performance optimization.
     4. **Table Schema**
        + Every table has a **schema** that defines:
          - **Column names** (e.g., customer\_id, purchase\_date)
          - **Data types** (STRING, FLOAT64, BOOLEAN, etc.)
          - **Modes**:

REQUIRED: Must have a value

NULLABLE: Can be empty

REPEATED: Can hold arrays of values

* + - * You can define the schema manually or let BigQuery auto-detect it when loading data
    1. **Types of Tables**

| **Table Type** | **Description** |
| --- | --- |
| **Standard Table** | Regular table stored in BigQuery’s columnar format |
| **External Table** | References data stored outside BigQuery (e.g., Cloud Storage, Google Drive) |
| **View** | A virtual table defined by a SQL query; doesn’t store data itself |
| **Materialized View** | A cached version of a view for faster querying |
| **Table Snapshot** | Read-only, point-in-time copy of a table |
| **Table Clone** | Lightweight, writable copy that stores only changes from the original |

* + 1. **Table Operations**
       - You can perform various operations on tables:
         * **Create**: Using Console, SQL (CREATE TABLE), CLI (bq), or API
         * **Load Data**: From CSV, JSON, Avro, Parquet, or via streaming
         * **Query**: Using SQL with powerful features like joins, nested fields, and UDFs
         * **Export**: To Cloud Storage in supported formats
         * **Copy**: Duplicate tables within or across datasets
         * **Delete**: Remove tables when no longer needed
    2. **Advanced Features**
       - **Partitioning**: Divide tables by date or integer range to improve performance and reduce cost
       - **Clustering**: Organize data within partitions based on column values for faster scans
       - **Time Travel**: Query historical versions of a table up to 7 days in the past
       - **Encryption**: Tables are encrypted at rest and in transit; CMEK is supported
    3. **Limitations & Quotas**
       - Max 50,000 tables per dataset (UI can display up to that)
       - Export destination must be Cloud Storage
       - Table names must be unique within a dataset
  + **Schemas**
    1. A **schema** defines the structure of a table: column names, data types, and modes.
    2. Each column has:
       - **Name** (e.g., customer\_id)
       - **Type** (e.g., STRING, INT64, DATE)
       - **Mode**:
         * NULLABLE: allows nulls
         * REQUIRED: must have a value
         * REPEATED: stores arrays of values
       - **Description** (Optional text to explain the column’s purpose(max 1024 chars))
       - **Default value** (can be a literal or function like current\_date(), etc..,)
    3. You can specify schemas manually or let BigQuery auto-detect them when loading data.
    4. **Supported Data Types**
       - BigQuery supports a wide range of types:
         * **Basic Types**: INT64, FLOAT64, NUMERIC, STRING, BOOL, BYTES
         * **Date/Time**: DATE, DATETIME, TIME, TIMESTAMP
         * **Complex Types**:

STRUCT: Nested fields (like a record)

ARRAY: Repeated values

GEOGRAPHY: Spatial data

JSON: Semi-structured data

RANGE: Date/time ranges

1. **Querying data using SQL.**
   * SELECT state, COUNT(\*) FROM `my\_dataset.customers` GROUP BY state;
2. **Managing Storage and Compute resources**
   * **Storage Management in BigQuery**
     1. **Types of storage**
        + **Managed Storage**: BigQuery stores your datasets and tables natively.
        + **External Storage**: You can query data directly from GCS, Cloud SQL, or other sources using federated queries
     2. **Storage Features**
        + **Columnar Format**: Internally uses Capacitor format for faster column-based retrieval.
        + **Partitioning**: Breaks data into date/time or integer ranges to improve performance and reduce cost.
        + **Clustering**: Organizes data within partitions based on selected columns to make filters efficient.
   * **Compute Resource Management in BigQuery**
     1. **Types of Compute models**
        + **On-demand :** pay per TB of data processed.
        + **Flat-rate :** Purchase slots for dedicated capacity.
     2. **How Compute Works**
        + BigQuery uses **slots**, which are units of computational capacity.
        + Queries are **parallelized** across slots, making them scalable and fast.
        + Complex queries might require more slots, hence higher cost unless tuned.

**Advanced Features**:

1. **Partitioned and clustered tables.**
   1. Partitioned and clustered tables are a powerhouse combo for optimizing performance and reducing query cost in BigQuery.
   2. **Partitioned tables**
      1. Partitioning divides a table into segments based on a column’s value—commonly a date or timestamp—so queries only scan relevant partitions rather than the entire table.
   3. **Types of Partitioning**

| **Partition Type** | **Description** | **Example Column** |
| --- | --- | --- |
| **Ingestion Time** | Automatically uses the \_PARTITIONTIME pseudo-column | Newly ingested logs |
| **Column-based** | Partitions by a specific DATE or TIMESTAMP column | event\_timestamp |
| **Integer Range** | Partitions by ranges like user\_id, region\_id, etc. | customer\_id |

* 1. **Clustered Tables**
     1. Clustering organizes data within each partition based on column values. Think of it like sorting data for faster lookups.
     2. **Benefits**
        1. Speeds up queries with filters and aggregations
        2. Efficient storage layout
        3. Works best with high-cardinality columns (e.g. user\_id, product\_id)

**What Are User-Defined Functions (UDFs)**

1. UDFs allow you to create **custom functions** in SQL using either **JavaScript** or **SQL syntax**, extending BigQuery’s native functionality with personalized logic.
2. They’re useful when:
   1. Built-in functions can’t handle your custom logic.
   2. You need reusable transformations across queries.
   3. You’re cleaning, normalizing, or validating data in complex ways.
3. **Types of UDFs in BQ**
   1. **JavaScript UDFs**
   2. **SQL UDFs**
4. **TEMP vs PERMANENT UDFs**

| **Type** | **Description** | **Use Case** |
| --- | --- | --- |
| **TEMP** | Defined in-session and used within one query | Quick experiments or ad hoc logic |
| **PERMANENT** | Stored in dataset and reusable across queries | Production-grade workflows |

**BigQuery ML for machine learning**

1. BigQuery ML lets you run machine learning models **directly inside BigQuery using SQL.**
2. BigQuery ML (BQML) enables you to:
   1. **Create, train, evaluate, and predict with ML models using SQL.**
   2. Avoid exporting data to external tools—everything stays within BigQuery.
   3. Use it with massive datasets that benefit from Google's distributed architecture.
3. **Supported Models**

| **Model Type** | **Use Case** | **Example Scenario** |
| --- | --- | --- |
| Linear & logistic regression | Predict numeric or binary outcomes | Forecast insurance premiums or churn |
| K-means clustering | Group similar data points | Segment customers or transactions |
| Time series forecasting | Predict future trends | Sales or traffic forecasting |
| Matrix factorization | Recommendation systems | Suggest policies based on preferences |
| Deep neural networks (DNN) | Complex patterns and interactions | Image/text feature extraction |
| AutoML & XGBoost | Advanced and automated model tuning | Insurance claim risk modeling |