**Introduction to Databases and BigQueries**

**Session 2023-2027**

**Submitted by:**

FajarShahzad

**Supervised by:**

Ali Ahmed Qureshi

Mujahid

**Course:**

Data-Engineering

## 

Table of Content

[**Database and BigQuery 3**](#_heading=h.eep8w4gsj7gs)

[Introduction to Databases & BigQuery Setup 3](#_heading=h.jwhl9rigymk4)

[What is a Database? 3](#_heading=h.ghk11red04u3)

[What is a Data Warehouse? 3](#_heading=h.w7hhpqs1hzz1)

[Difference between Database and Data Warehouse: 3](#_heading=h.783h8h3zamsu)

[Google BigQuery Architecture 3](#_heading=h.anm5bnpo6mdd)

[Advantages of BigQuery over RDBMS 4](#_heading=h.pik1qo9q1xrd)

[Loading Data into BigQuery 4](#_heading=h.pr7kxwdvf23o)

[1. Using the UI: 4](#_heading=h.or2g73noqqdd)

[Datasets, Tables, and Views 4](#_heading=h.qrrycs7mnlvc)

[Federated vs Native Tables 4](#_heading=h.6gi7bjpbtsyx)

[IAM Roles in BigQuery 5](#_heading=h.8epb8dmm2bsd)

[SQL: Extracting Data 5](#_heading=h.dgfy4kbw30m)

[SELECT \* vs Column Names 5](#_heading=h.w832zfqi7ay3)

[Common SELECT Mistakes: 5](#_heading=h.lh2do3vwyf7m)

[Functions, Filtering & Subqueries 5](#_heading=h.wcrkajb1hmqo)

[Window Functions 7](#_heading=h.q4wwo3civqer)

[Indexes & Partitioning 7](#_heading=h.oqwnxwixi2m4)

[Normalization & Transactions 7](#_heading=h.23jro953t4ff)

[Introduction to NoSQL: MongoDB 8](#_heading=h.irjpv6fbmgol)

## 

# Database and BigQuery

## Introduction to Databases & BigQuery Setup

### What is a Database?

A database is an organized collection of data that can be accessed, managed, and updated. It is usually managed by a database management system (DBMS), like MySQL or PostgreSQL.

### What is a Data Warehouse?

A data warehouse is designed for analytical processing rather than transactional processing. It stores historical data and is optimized for reading large datasets efficiently.

### Difference between Database and Data Warehouse:

| **Feature** | **Database (RDBMS)** | **Data Warehouse (BigQuery)** |
| --- | --- | --- |
| **Purpose** | OLTP (Online Transactional) | OLAP (Online Analytical) |
| **Data Structure** | Normalized | Denormalized |
| **Query Type** | Insert/Update/Delete | Complex Analytics (SELECT-heavy) |
| **Performance Optimization** | Indexes | Partitioning & Clustering |

### Google BigQuery Architecture

* Serverless: No infrastructure management
* Separation of Storage & Compute
* Columnar Storage: Data is stored in columns to improve query performance
* Dremel Engine: Executes SQL queries quickly over petabytes of data
* Security & IAM: Integrated with Google Cloud IAM

### Advantages of BigQuery over RDBMS

* No server provisioning
* Handles petabyte-scale data efficiently
* Real-time analytics support
* Simple integration with GCS, Sheets, Looker
* Cost model supports both pay-as-you-go and flat-rate options

## Loading Data into BigQuery

### 1. Using the UI:

* Go to GCP Console > BigQuery > Create Table
* Choose source (e.g., GCS, local file)
* Define schema manually or auto-detect
* Click Create Table

## Datasets, Tables, and Views

* Dataset: Logical container for tables and views.
* Table: Actual storage structure where data resides.
* View: Virtual table created using SQL queries.

## Federated vs Native Tables

* Federated Table: Points to external sources like GCS, Google Sheets.
* Native Table: Stores data inside BigQuery.

## IAM Roles in BigQuery

* BigQuery Admin: Full access
* BigQuery Data Editor: Modify data
* BigQuery Job User: Run queries

## SQL: Extracting Data

**Note:Basic Queries are in my sql file**

### SELECT \* vs Column Names

* SELECT \*: retrieves all columns
* SELECT name, age: retrieves specified columns (faster, better practice)

### Common SELECT Mistakes:

* Forgetting WHERE clause
* Selecting unused columns
* Missing GROUP BY
* Incorrect function usage

## Functions, Filtering & Subqueries

**Scalar vs Aggregate Functions**

**Scalar Function:** Operates on each row individually.

**Aggregate Function:** Operates on multiple rows.

**WHERE vs HAVING:**

WHERE: Filters rows before aggregation.

HAVING: Filters groups after aggregation.

**Subquery:**

A query nested inside another.

**Correlated Subquery:**

Refers to the outer query’s columns.

**Nested Subqueries vs Joins:**

Subqueries simplify logic.

Joins are often more efficient for large data.

**Joins**

**Types of Joins:**

* **INNER JOIN:** Only matching rows
* **LEFT JOIN:** All left + matched right
* **RIGHT JOIN:** All right + matched left
* **FULL OUTER JOIN:** All rows from both tables
* **CROSS JOIN:** Cartesian product
* **SELF JOIN:** Join table with itself

**Common Join Mistakes:**

* Missing ON clause
* Using wrong column for join
* Ambiguous column names

**Performance:**

* Use indexes on join columns
* Reduce data with WHERE clauses before join

**GROUP BY vs DISTINCT:**

**DISTINCT** removes duplicates.

**GROUP BY** aggregates data.

## Window Functions

**What is a window function?**

Performs calculations across rows related to the current row.

Unlike GROUP BY, it doesn’t collapse rows.

## Indexes & Partitioning

**Purpose of Index**

Faster search, filtering, JOINs.

**BigQuery uses:**

**Partitioning:** splits data into segments (e.g., by date).

**Clustering:** sorts within partitions for better performance.

**Partitioning vs Clustering**

**Feature Partitioning Clustering**

Works on One column (usually DATE) Multiple columns

Purpose Reduce scanned data Organize within partitions

Used For Time-series, large datasets Faster filtering, GROUP BY

## Normalization & Transactions

**What is Normalization?**

Process of organizing data to reduce redundancy.

**Normal Forms:**

* **1NF:** No repeating groups
* **2NF:** Remove partial dependencies
* **3NF:** Remove transitive dependencies
* **BCNF:**Remove all anomalies caused by functional dependencies
* **4NF:**Remove all anomalies caused by multi-valued dependencies
* **5NF:**Remove all anomalies caused by join dependencies

**Anomalies in Non-Normalized Tables:**

**Insert anomaly:** Can’t insert without unrelated data

**Update anomaly:** Change in one place needs update everywhere

**Delete anomaly:** Deleting one item may delete related data

**ACID Transactions:**

* **Atomicity:** All or none
* **Consistency:** DB stays valid
* **Isolation:** No interference
* **Durability:** Changes persists

## Introduction to NoSQL: MongoDB

**What is NoSQL?**

* Schema-less, document-oriented, key-value or graph stores.
* Useful for unstructured or rapidly evolving data.

**MongoDB & CAP Theorem**

* Prioritizes Availability and Partition Tolerance.
* Can be eventually consistent