Big Query

Topics:

- ✓ What is Big Query and Why?
- ✓ Datatypes: Structured & Unstructured Data
- ✓ The Main Pillars/Features of Big Query
- ✓ Benefits, Pros & Cons, Use Cases of Big Query
- ✓ Real-World Examples in our Daily life
- ✓ Big Query Integrates and Supports to other services in the Google Cloud Platform
- ✓ Architecture and Flow
- ✓ Comparison of Big Query of GCP with Other Clouds like AWS, Azure
- ✓ Cost Optimization
- ✓ Concrete cost-optimization actions per platform (how to implement — step-by-step)
- ✓ Project on Big Query

Overview: BigQuery is **Google Cloud's fully managed, serverless data warehouse** designed for **large-scale analytics**.

- It allows you to store, query, and analyze petabytes of data using **standard SQL**.
- It's optimized for **real-time insights** and integrates deeply with other GCP services like Cloud Storage, Dataflow, Pub/Sub, and AI/ML tools.
- Think of it as **"Google Search for your data warehouse"** — extremely fast, scalable, and you pay only for what you query.
- Big Query is a data service with highly scalable serverless data warehouse component, so it means it's a data processing component where you can process your data, you can transform your data you can write all kind of complex SQL joint queries in big query and you can generate the reports so all ETL Transmissions you can able to do with respect to big query.
- Big Query is a **Serverless** so entire Infrastructure was maintained by Google Cloud.
- So Big Query works on **Scale out Functionality**, **available in all the regions**
- From Big Query you can **Auto Backup Feature** (It also supports Fault Tolerant applicable even a particular node goes down your data is not lost it is still there in a different node and if you made any changes in the table the changes will be **stored for seven days**.
- It also Supports **Programmatic Integrations like Java, Python, NodeJS, PHP, Ruby** etc.

➔**Example:** Suppose if you have **spreadsheets excels in Google drive** you have some data as a CSV file here so you can run a query from Big Query on top of this Google Drive without migrating the data from Google Drive to Big Query it's not required that is called avoiding the duplicating the data so you are not going to duplicate the data and that is also one main reason **Google Big Query act as a**

Data Lakes processing frameworks because in data lake only a component can able to do the compute only for the data stored in a different storage system. So Big Query is doing the same it is eligible to be a data lake kind of components. It also Supports other external files like ORC, Parquet, CSV, JSONL (Newline Delimited JSON), Avro, it and other open-source file formats.

Why BigQuery?

- **No infrastructure management** → Google handles servers, scaling, and replication.
- **Massive scalability** → Can run petabyte-scale queries in seconds.
- **SQL-based** → Analysts can use standard SQL.
- **Integrations** → Works seamlessly with GCP services (Cloud Storage, Pub/Sub, Dataflow) and BI tools (Looker Studio, Tableau, Power BI).

Main Value: Analyze huge datasets in near real-time without worrying about infrastructure or upfront costs.

Case Study Example – Spotify

- **Problem**: Needed real-time insights on billions of events (streams, likes, skips).
- **Solution**: Used BigQuery with Pub/Sub for streaming ingestion + Looker dashboards.
- **Result**:
  - Reduced query time from **hours → seconds**.
  - Enabled real-time artist dashboards for insights.
  - Optimized infra costs with on-demand query pricing.
- Another case: **Home Depot** uses BigQuery to analyze customer journey across website & stores in real-time.

Data Types: Structured & Unstructured Data

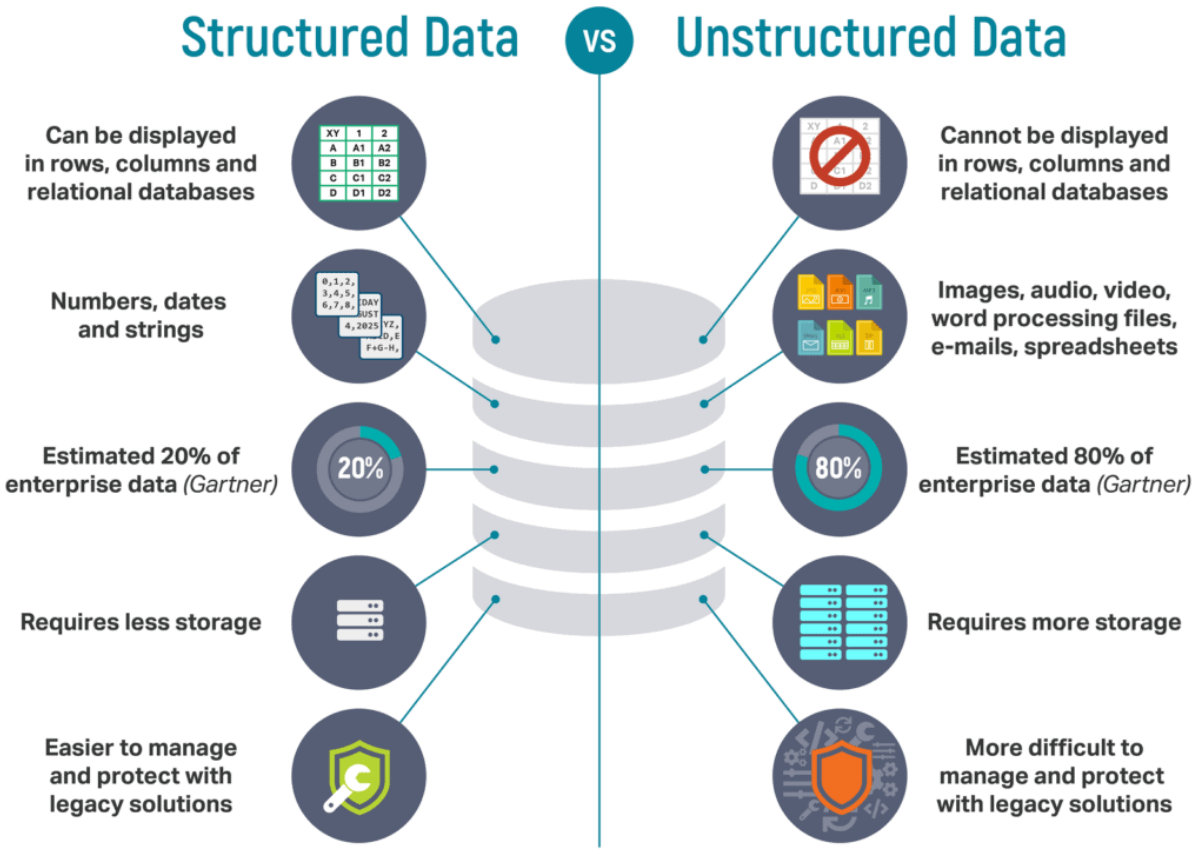
The Google storage and database services can be put into 2 categories:

1.) Structured Data

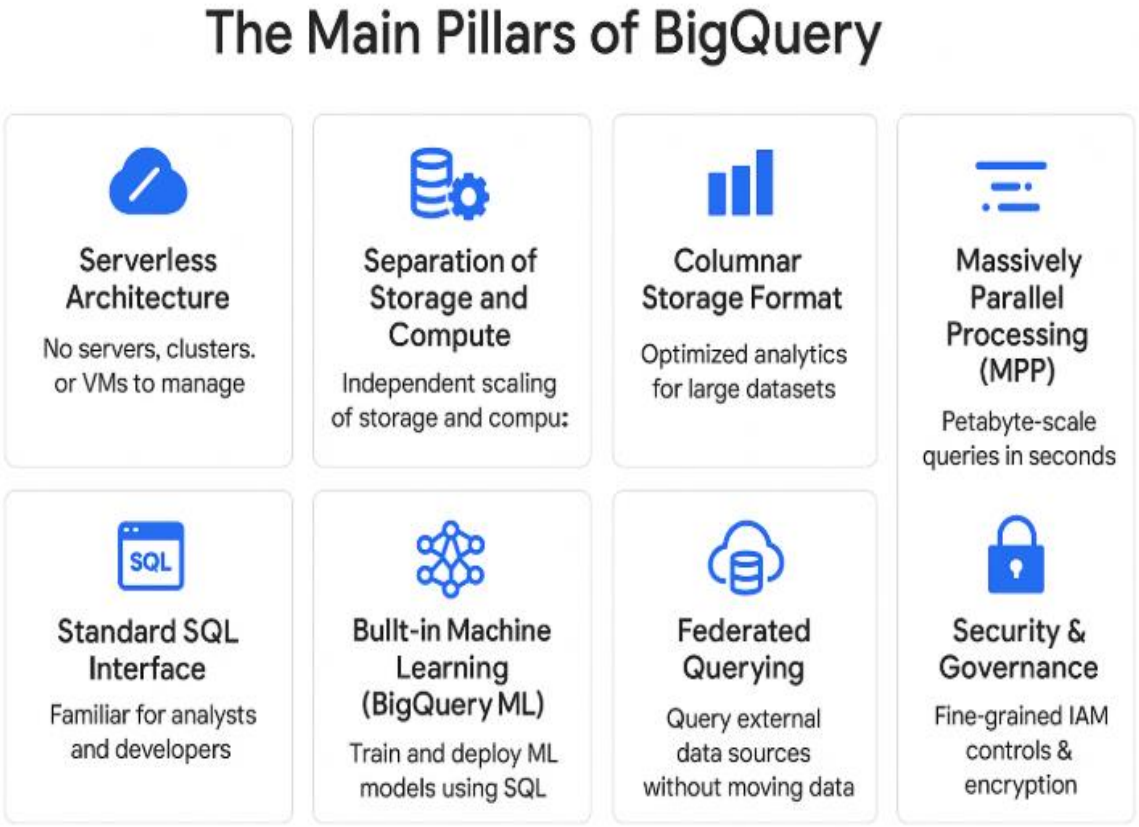
- If the data can be organized in a structural format like rows and columns then it is known as structured data. It comes in various sizes, latency, and cost based on the requirement. **Example: Financial data, logs, etc.**
- From the various offerings of Google Storage service, structured data can be stored in Cloud SQL, Cloud Spanner, Cloud Datastore, Cloud Bigtable, Cloud BigQuery, and Persistent disk.

2.) Unstructured Data

- It is a sequence of bytes that could be from a video, image, or document. The data is stored as objects in buckets and no insight can be gained from unstructured data.
- Google Cloud Storage and Cloud Firestore are used to store unstructured data in the Google Cloud Platform.



The Main Pillars/Features of BigQuery



◈ 1. Serverless Architecture
- No servers, clusters, or VMs to manage.
- Google handles scaling, patching, availability, and performance tuning.
- You focus only on queries and data, not infrastructure.
- Result → Lower ops burden, faster deployment.

◈ 2. Separation of Storage and Compute
- Storage (data at rest) and Compute (query execution) are independent.
- Storage uses Colossus (Google's distributed storage system).
- Compute uses Dremel execution engine.
- Storage + Compute = Colossus + Dremel
- This allows you to scale storage infinitely without scaling compute, and vice versa.
- Result → Cost efficiency & flexibility.

◈ 3. Columnar Storage Format
- Data stored in columnar format (Capacitor engine).
- Only the required columns are scanned during a query.
- Greatly reduces I/O → faster queries & lower costs.
- Result → Optimized analytics for large datasets.

◈ 4. Massively Parallel Processing (MPP)
- Queries are split into smaller tasks and executed in parallel across thousands of nodes.
- Based on Google's Dremel technology (the same system that powers Google Search indexing).
- Result → Petabyte-scale queries in seconds.

◈ 5. Standard SQL Interface
- Supports ANSI SQL (familiar for analysts & developers).
- Handles nested and repeated data (good for JSON/NoSQL-like structures).
- Supports User-Defined Functions (UDFs) in SQL or JavaScript.
- Result → Low learning curve, wide adoption.

◈ 6. Built-in Machine Learning (BigQuery ML)
- Train and deploy ML models directly inside BigQuery using SQL.
- No need to export data to TensorFlow or scikit-learn.
- Supports regression, classification, clustering, recommendation, time-series forecasting, etc.
- Result → Democratizes ML for data analysts.

◈ 7. Federated Querying
- Query external data sources without moving data.
- Examples:
    - Cloud Storage (CSV, Parquet, Avro, ORC, JSON)
    - Google Sheets
    - Cloud SQL / Bigtable

- Result → Avoids unnecessary ETL pipelines.

🔷 8. Security & Governance
- Fine-grained IAM controls → dataset/table/column/row-level.
- Data encrypted at rest & in transit by default.
- Compliance with HIPAA, GDPR, SOC, ISO.
- Result → Enterprise-ready for sensitive data.

Summary (Easy to Remember)

The 8 Pillars of BigQuery are:

1. **Serverless (no infra management)**
2. **Separation of Storage & Compute**
3. **Columnar Storage (Capacitor)**
4. **Massively Parallel Processing (Dremel)**
5. **Standard SQL Support**
6. **Built-in ML (BigQuery ML)**
7. **Federated Querying**
8. **Security & Governance**

Benefits of BigQuery

- **Speed**: Interactive queries over TBs or PBs of data.
- **Ease of Use**: SQL interface, no infrastructure management.
- **Integration**: Works well with GCP ecosystem (Dataflow, Pub/Sub, AI Platform).
- **Scalability**: Scales automatically without provisioning.
- **Cost Model**: Pay per query or flat-rate; no upfront hardware cost.
- **Machine Learning at Scale**: Run ML directly on data.

Pros & Cons of BigQuery

Pros

1. **Serverless** → no ops overhead.
2. **High-speed analytics** → seconds for massive datasets.
3. **Highly Scalable** → supports terabytes to petabytes.
4. **Integration with AI/ML** → run models inside SQL.
5. **Both Batch and Streaming data Ingestion**
6. **Supports all the Regions**
7. **Automated data transfer**
8. **Access Control**
9. **Security & compliance** → enterprise-ready.
10. **Pay-as-you-go** → cost-efficient for unpredictable workloads.

Cons

1. **Cost unpredictability** → large ad-hoc queries can become expensive.
2. **Query limits** → daily quota limits may apply in free tier.
3. **No indexes** → unlike RDBMS, tuning options are limited.
4. **Latency for streaming inserts** → near real-time, but not true sub-second.
5. **Vendor lock-in** → migrating out of BigQuery can be costly.

BigQuery Use Cases

1. Business Intelligence (BI) & Analytics

- **Scenario:** A retail company wants to analyze sales data across 100+ stores. They need insights like "Top-selling products per region" or "Customer buying patterns."
- **How BigQuery Helps:**
  - Stores terabytes of POS (point of sale) data.
  - Runs SQL queries in seconds (without managing servers).
  - Connects to **Looker Studio**, Tableau, or Power BI for dashboards.

Real-world: Walmart and Target use data warehouses like BigQuery to optimize stock levels and detect seasonal trends.

2. Real-Time Analytics

- **Scenario:** A fintech company monitors **fraudulent transactions** in real time.
- **How BigQuery Helps:**
  - Streaming data from Pub/Sub or Kafka into BigQuery.
  - Analyze transactions with ML models on-the-fly.
  - Detects anomalies within seconds.

Real-world: Payment gateways like Stripe use real-time data analysis to flag suspicious activity instantly.

3. Log & Event Analytics

- **Scenario:** A SaaS company wants to analyze user logs to detect system errors, usage spikes, or customer churn signals.
- **How BigQuery Helps:**
  - Collects logs from **Cloud Logging** or **Google Analytics 360**.
  - Query petabytes of event data fast.
  - Identifies slow API endpoints, popular app features, etc.

Real-world: Netflix analyses streaming logs to optimize recommendations and server load.

4. Customer 360 & Personalization

- **Scenario:** An e-commerce business wants a **360-degree customer view**: purchases, clicks, support tickets.
- **How BigQuery Helps:**
  - Combines structured + semi-structured data (JSON, logs, DB exports).
  - Builds predictive models (using **BigQuery ML**) for churn, next best product, or lifetime value.
  - Feeds insights into marketing automation systems.

Real-world: Amazon uses big data warehouses to personalize recommendations.

5. IoT Data Analytics

- **Scenario:** A smart city project collects **sensor data** (traffic, weather, pollution) every second.
- **How BigQuery Helps:**
  - Ingests continuous IoT data via Pub/Sub → BigQuery.
  - Runs time-series analytics & predictions.
  - Provides dashboards for city planners.

Real-world: Tesla processes sensor data from cars to optimize performance and detect

failures.

6. Machine Learning (BigQuery ML)

- **Scenario:** A telecom wants to **predict customer churn** without exporting data to a separate ML platform.
- **How BigQuery Helps:**
  - Train ML models (regression, clustering, forecasting) directly in SQL.
  - No need for Python/ML engineers.
  - Deploy predictive models into business workflows.

Real-world: Telecoms predict churn & banks detect fraud directly inside BigQuery.

7. Data Lake + Data Warehouse Hybrid

- **Scenario:** A media company stores videos, clickstream, and user metadata in **Cloud Storage (data lake)** but needs analytics.
- **How BigQuery Helps:**
  - Directly queries data from Cloud Storage & external sources (federated queries).
  - No need to copy all raw data into the warehouse.
  - Saves cost + improves agility.

Real-world: Spotify uses similar architectures to analyze music streaming data.

8. A retail chain like **Walmart** or an e-commerce platform like **Flipkart** could use this setup to:

- Identify **top-selling products** per region.
- Detect **seasonal sales patterns** (Diwali, Christmas, etc.).
- Find **VIP customers** and run loyalty campaigns.

Optimize **supply chain & pricing strategies**.

Real-World Examples in our Daily life:

More Advanced and industry-specific use cases of BigQuery.

🛒 Retail & E-Commerce

- **Dynamic Pricing Optimization**
  - Adjust product prices in real-time based on demand, stock, or competitor data.
  - Example: Flipkart/Amazon change prices dynamically using massive historical + real-time sales data in their warehouses.
- **Recommendation Engines**
  - Analyze customer behaviour (clicks, searches, purchases) to suggest products.
  - Example: Myntra & Amazon use data warehouses like BigQuery to feed recommendation models.

💳 Banking & FinTech

- **Fraud Detection & Risk Analysis**
  - Detect unusual transaction patterns using real-time queries.
  - Example: Mastercard & Visa use anomaly detection pipelines on petabytes of

data.

- **Credit Scoring**
  - Process historical customer loan/payment records, transaction logs, and demographic data.
  - Predict customer creditworthiness using **BigQuery ML**.

🚑 Healthcare & Life Sciences

- **Genomic Data Analysis**
  - Store and analyze petabytes of DNA sequencing data.
  - Example: Google Cloud has partnerships with biotech firms for DNA pattern recognition.
- **Patient 360 & Predictive Healthcare**
  - Combine EHR (Electronic Health Records), wearable data, and lab results to predict health risks.
  - Example: Hospitals predict readmission risk and chronic illness trends.

🎬 Media & Entertainment

- **Content Recommendation & Personalization**
  - Streaming platforms analyze **watch history, pause/skip behaviour, and ratings**.
  - Example: Netflix & YouTube optimize "What to watch next" with BigQuery + ML.
- **Ad Revenue Optimization**
  - Track billions of ad impressions, clicks, and conversions.
  - Publishers use BigQuery for **real-time bidding insights**.

🚗 Transportation & Smart Mobility

- **Fleet Management & Predictive Maintenance**
  - IoT sensors in cars/planes stream telemetry data.
  - Predict component failure before breakdowns.
  - Example: Airlines use predictive maintenance to reduce downtime.
- **Route Optimization & Traffic Prediction**
  - Combine GPS, IoT, and weather data.
  - Example: Uber, Ola, and Google Maps use data warehouses for route optimization.

🏭 Manufacturing

- **Supply Chain Optimization**
  - Track inventory levels, production cycles, vendor lead times.
  - Predict shortages and optimize logistics.
- **Defect Detection**
  - Store quality inspection logs, sensor data, and production line metrics.
  - Identify defect patterns quickly.

🌍 Public Sector / Smart Cities

- **Urban Planning & Traffic Analytics**

- Process data from sensors, cameras, and GPS devices.
- Example: Cities optimize traffic signals and reduce congestion.

➢ **Environmental Monitoring**

- IoT sensors track air quality, water quality, noise pollution.
- Governments use real-time dashboards powered by BigQuery.

☑ Marketing & AdTech

➢ **Customer Segmentation**

- Cluster customers by behavior, spend, demographics.
- Use **BigQuery ML** for k-means clustering.

➢ **Campaign Performance Optimization**

- Analyze billions of ad clicks, impressions, conversions.
- Adjust campaigns in real-time for ROI.

Telecom

➢ **Network Performance Monitoring**

- Telecoms process logs from towers, routers, and devices.
- Predict outages or bottlenecks.

➢ **Churn Prediction**

- Use BigQuery ML to identify customers most likely to leave.
- Proactively offer discounts or personalized plans.

Energy & Utilities

➢ **Smart Grid Analytics**

- Real-time monitoring of electricity usage across millions of households.
- Optimize load balancing.

➢ **Renewable Energy Forecasting**

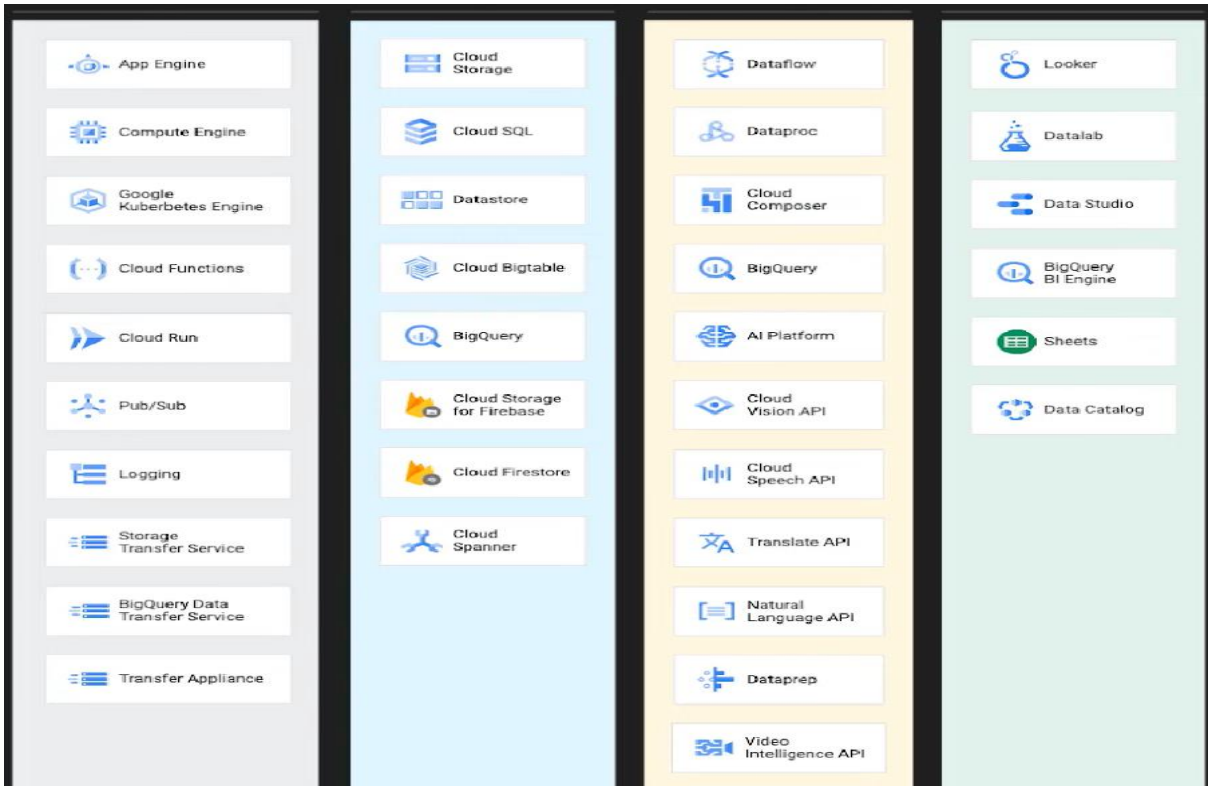
- Predict solar/wind output using IoT + weather data.
- Integrates with BigQuery for time-series forecasting.

Summary of Extra Use Cases

- Retail → pricing, recommendations
- FinTech → fraud detection, credit scoring
- Healthcare → genomics, patient 360
- Media → personalization, ad optimization
- Transport → fleet analytics, route optimization
- Manufacturing → supply chain, defect detection
- Smart Cities → traffic, environment
- Marketing → customer segmentation, campaigns
- Telecom → churn, network performance
- Energy → smart grids, renewable forecasting

Big Query Integrates and Supports to other services in the Google Cloud Platform:

Data Ingestion	Storage	Processing &Analytics	Visualization



- Google Drive (Spreadsheet Excels)
- Stack driver
- Cloud Dataproc
- Google Cloud Data Loss Prevention API
- Other Machine Learning API's
- AutoML
- Cloud PUB/SUB (Messaging Queue Service)
- Cloud Dataflow
- Data Studio
- Looker / Tableau also Cloud Functions
- Cloud AI Platform
- Cloud Scheduler and Cloud Functions
- Cloud Data Fusion
- Cloud Composer

Cost Optimization in BigQuery

Since BigQuery follows **on-demand pricing (per query, per TB processed)** or **flat-rate pricing**, optimization is **critical**.

◈ a) Partitioned Tables

- Partition by **date, timestamp, or integer column**.
- Queries scan only relevant partitions → **reduces scanned data size**.

◈ b) Clustered Tables

- Organize data by frequently filtered columns.
- Improves query performance and lowers scan costs.

◈ c) Avoid SELECT *

- Always **select specific columns** instead of SELECT *.

- Scans only necessary data.

🔹 **d) Materialized Views**

- Precompute results of expensive queries.
- Queries become faster and cheaper.

🔹 **e) Query Caching**

- Repeated queries within 24 hours return results from cache (free).

🔹 **f) Data Compression & Storage**

- Use compressed formats (Parquet, ORC, Avro) instead of raw CSV/JSON.
- Reduces storage and query costs.

🔹 **g) Batch Inserts instead of Streaming**

- Streaming costs $0.01 per 200 MB.
- Use batch loads (free) from Cloud Storage if real-time is not mandatory.

🔹 **h) Monitoring & Quota Controls**

- Set **cost controls & budgets** in GCP.
- Use **slot reservations** for predictable workloads.

🔹 **i) Flat-rate Pricing for Enterprise**

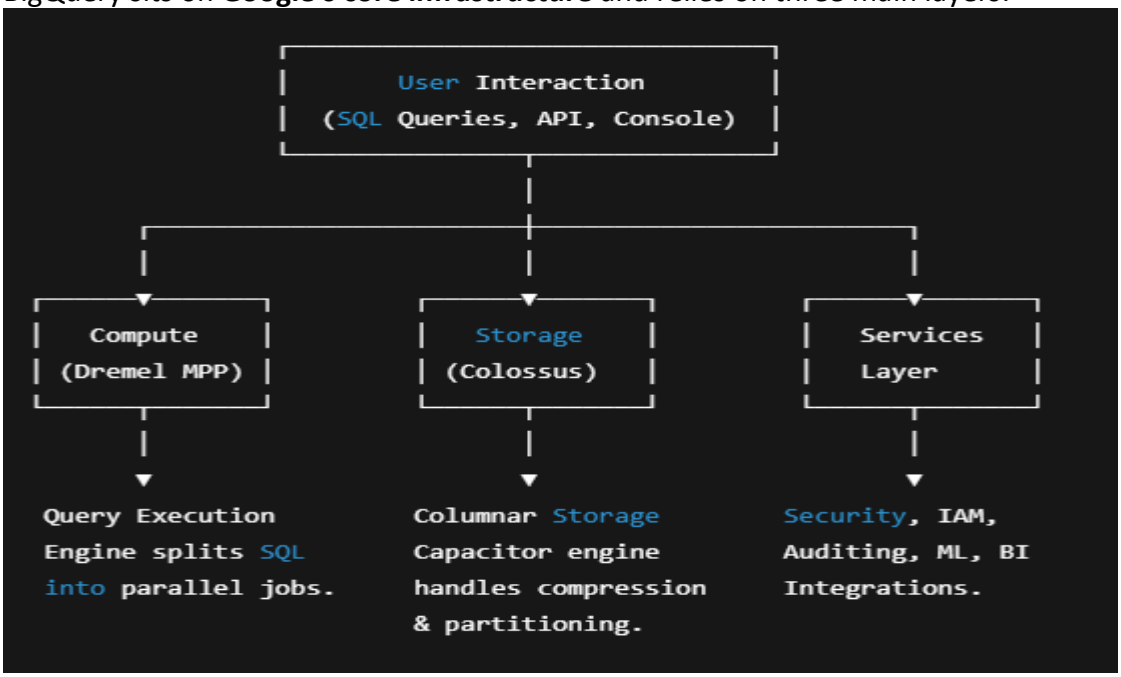
- If queries are very frequent, **flat-rate slots** provide predictable monthly cost.

Architecture:

1. **BigQuery Architecture Diagram**
2. **Detailed Explanation of Each Component**.

1. BigQuery Architecture (Conceptual View)

BigQuery sits on **Google's core infrastructure** and relies on three main layers:



2. Detailed Explanation of Each Layer

🔹 **A) User Interaction Layer**

- **Interfaces**:

- BigQuery Web UI (Cloud Console).
- REST API & Client Libraries (Python, Java, etc.).
- bq CLI tool.
- Integration with BI tools (Looker, Data Studio, Tableau, Power BI).

- **Workload**:
  - Analysts → Run SQL queries.
  - Engineers → Automate ETL via APIs.
  - ML teams → Use BigQuery ML for modelling.

🔷 B) Compute Layer (Dremel Execution Engine)

- Based on **Google's Dremel technology** (used internally for Google Search).
- Breaks down SQL queries into **multiple stages**.
- Executes tasks in **parallel across thousands of nodes** (Massively Parallel Processing).
- Automatically scales resources based on query complexity.
- **Query Jobs** are managed by **Borg (Google's cluster manager)**.

👉 **Result:** Petabyte-scale queries in seconds without cluster management.

🔷 C) Storage Layer (Colossus + Capacitor)

- **Colossus** = Google's global distributed file system.
  - Provides **infinite scalability**.
  - Handles durability, availability, and replication.
- **Capacitor** = Columnar storage format for BigQuery.
  - Stores data in **columns, not rows** → optimized for analytics.
  - Uses compression & encoding → reduces storage and scan cost.
  - Supports **partitioning (by date/int/timestamp)** and **clustering (by column)**.

👉 **Result:** Faster queries, lower costs, and better resource utilization.

🔷 D) Services Layer (Control Plane & Add-ons)

This is what makes BigQuery **enterprise-ready**:

1. **Security & Governance**
   - IAM roles (dataset, table, row, column-level security).
   - Automatic encryption (in transit & at rest).
   - Audit logging.
2. **Machine Learning**
   - **BigQuery ML** allows training models directly with SQL.
   - Supports regression, classification, clustering, forecasting, TensorFlow exports.
3. **Federated Queries**
   - Query data in **Cloud Storage, Sheets, Cloud SQL, Bigtable** without ingestion.
4. **Streaming & Real-time**
   - Data can be **streamed in (via API/Pub&Sub)** → available for query within seconds.
5. **Backup & Restore**
   - Automatic replication & snapshots.

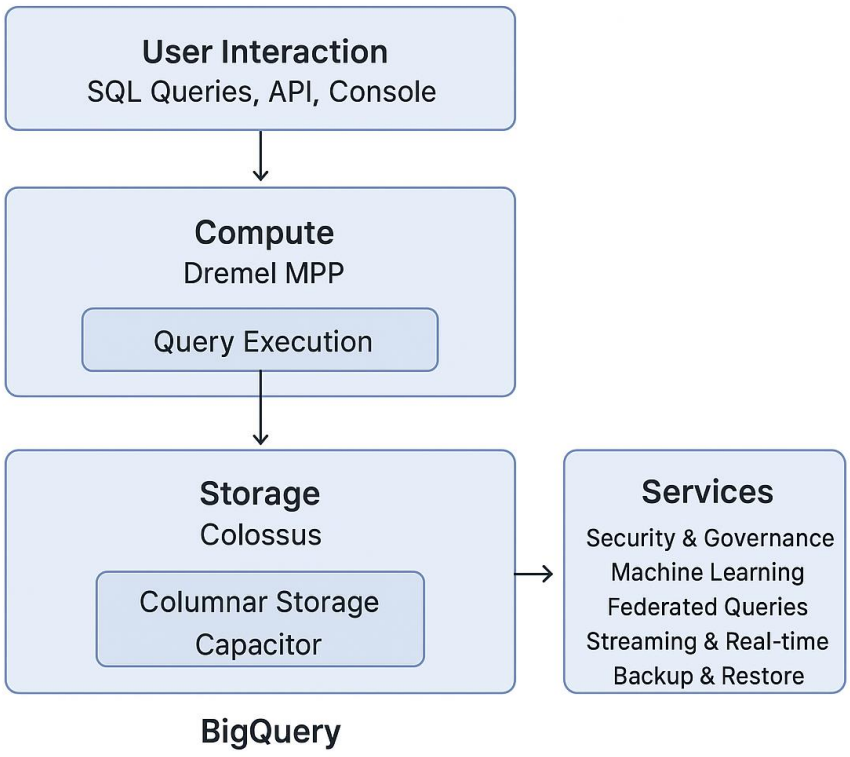
---

3. Data Flow Example

1. **Data Ingestion**
    - Batch Load: From Cloud Storage (CSV/JSON/Parquet).
    - Streaming Load: IoT/logs via Pub/Sub → BigQuery Streaming API.
    - Federated Query: Query directly from external sources (no ingestion).
2. **Storage**
    - Data stored in Colossus (global FS).
    - Organized in Capacitor columnar format → partitioned & clustered.
3. **Query Execution**
    - User submits SQL.
    - Query parsed & optimized.
    - Dremel engine splits into **sub-queries**.
    - Tasks run in parallel across nodes.
    - Results merged & returned.
4. **Output/Consumption**
    - Analysts → Looker/Data Studio dashboards.
    - ML → BigQuery ML models.
    - APIs → External applications.

Summary:

- **Storage**: Colossus + Capacitor → infinite scale, columnar.
- **Compute**: Dremel MPP → ultra-fast queries.
- **Control Plane/Services**: Security, ML, Federated queries, Streaming.
- **User Layer**: SQL, APIs, BI tools.



---

comparison of Google BigQuery vs AWS Redshift vs Azure Synapse Analytics

Their architectures, strengths/weaknesses, cost models, use-case fit, and what to choose when.

BigQuery Equivalents in Other Clouds

🔷 **In AWS**

- Equivalent service: **Amazon Redshift**
  - Purpose: Fully managed **cloud data warehouse**.
  - Similar to BigQuery but **cluster-based** (you manage node sizing/scaling).
  - Supports **SQL queries** over large datasets.
  - Integrates with **S3, Glue, SageMaker, QuickSight**.
  - Also has **Redshift Serverless** for BigQuery-like experience (no clusters).

👉 **BigQuery ≈ Redshift (AWS)**

🔷 **In Azure**

- Equivalent service: **Azure Synapse Analytics**
  - Purpose: Data warehouse + analytics service.
  - Two modes:
    - **Dedicated SQL Pools** (like Redshift clusters).
    - **Serverless SQL Pools** (like BigQuery on-demand pricing).
  - Integrates with **Power BI, Data Factory, Azure ML**.
  - Strong fit if you already use **SQL Server ecosystem**.

👉 **BigQuery ≈ Synapse (Azure)**

Comparison BigQuery vs Redshift vs Synapse

Aspect	BigQuery (GCP)	Amazon Redshift (AWS)	Azure Synapse Analytics (Microsoft Azure)
Architecture / Compute-Storage Separation	Fully serverless; storage and compute separated; Google handles scaling behind the scenes.	Uses clusters; newer generation nodes (RA3) also separate storage from compute to some extent, but you still provision clusters.	Offers both dedicated (provisioned) and serverless options; compute and storage can be scaled separately.
Management & Operations Effort	Low – minimal admin; no server management; scaling, optimizations handled by Google.	More hands-on: choosing node types, cluster sizing, tuning sort/dist keys, managing workloads.	Mid-level: if using serverless mode, less management; with provisioned/dedicated, more operations (provisioning, scaling, etc.).
Query Performance & Scalability	Very high; handles petabytes; good for high concurrency; performance	Also strong, especially for large, predictable	Good scalability; integrates big data processing (Spark), serverless SQL on data lake etc.; performance depends on whether dedicated

Aspect	BigQuery (GCP)	Amazon Redshift (AWS)	Azure Synapse Analytics (Microsoft Azure)
	optimized via “behind-the-scenes” features like cluster-by, partitioning, etc.	workloads; tuning (sort keys, distribution keys, materialized views) can make queries faster.	provisioning or serverless is used.
Support for Streaming / Near-Real-Time	Supports streaming inserts; near real-time data becoming available.	Supports streaming via AWS services (e.g. Kinesis + Firehose) etc. But often more indirect, not always “as simple” as BigQuery streaming interface.	Supports streaming and integration via Azure Stream Analytics, Spark, etc.; offers real-time ingestion options.
Data Types / Semi-Structured Support	Good support for JSON, nested/repeated fields; can query external data sources.	Also supports semi-structured (JSON, etc.), though sometimes performance penalties or need for particular formats / extra setup.	Supports structured, semi-structured; with Synapse, you can also query lake data, use Spark etc.
Machine Learning / Advanced Analytics	Built-in ML with BigQuery ML (train / predict from inside SQL).	Integrates well with AWS ML tools like SageMaker; sometimes need to move data.	Deep integration with Azure ML, Cognitive Services etc.; unified environment in Synapse Studio.
Pricing Model / Cost Predictability	On-demand (pay per query & storage), plus flat-rate pricing for committed use. Can have cost spikes for large ad-hoc queries.	Pay for clusters/nodes; can reserve instances; cost depends on cluster size & usage; you pay even if idle, unless you scale down.	Flexible: serverless SQL (which is pay per usage), or provisioned/dedicated with units (DWUs / capacities); you can optimize by pausing / scaling.
Integration / Ecosystem Fit	Excellent with GCP tools: Dataflow, Pub/Sub, GA, Looker, Data Studio etc. If you	Very good within AWS: S3, Glue, IAM, Lambda, etc. If your	Best if using Microsoft stack: Azure Data Lake, Power BI, Azure ML, Office365, etc. Synapse built to integrate

Aspect	BigQuery (GCP)	Amazon Redshift (AWS)	Azure Synapse Analytics (Microsoft Azure)
	are already using GCP, less friction.	data lake or workloads are in AWS, easier flow.	broadly on Azure.
Security, Compliance, Governance	Strong defaults: encryption at rest and in transit, IAM, auditing etc.	Also strong; many features; but sometimes more configuration required to meet best practices.	Also strong; fine-grained access, identity integration, etc. Azure tends to provide many enterprise tools out of the box.
Typical Use-Cases / When One Is Better	Best for: rapid prototyping, serverless analytics with variable loads; if you want minimal ops; bursts of usage; data already in GCP.	Best for: more stable, predictable heavy workloads; when you need tight control and optimizations; if your data ecosystem is AWS.	Best for: organizations invested in Microsoft/Azure; hybrid workloads; needing both data warehouse + big data + data lake + BI in unified platform.

Pros & Cons (Comparative)

Here are some of the trade-offs comparing them:

Platform	Key Strengths Compared to Others	Key Weaknesses / Drawbacks Compared to Others
BigQuery	+ Serverless: you don't need to manage clusters. + Very scalable and flexible; good performance for ad-hoc queries. + Built-in ML → reduces data movement. + Strong cost optimization features (partitioning, clustering, caching).	− Cost unpredictability: large ad-hoc queries can generate big query-cost bills. − Less control over hardware / fine-tuned performance knobs (compared to cluster-based ones). − Sometimes latency overhead for streaming / immediate consistency could be a factor. − If data/workloads are mostly on another cloud (AWS/Azure), moving data may incur networking cost or complexity.
Amazon Redshift	+ Good for analysts or teams who want fine control over compute resources. + Mature product; many tools,	− More administrative overhead: you need to size clusters, manage node types, tune keys/distributions.

Platform	Key Strengths Compared to Others	Key Weaknesses / Drawbacks Compared to Others
	optimizations. + Sometimes cost efficient if you reserve capacity and workloads are stable. + Good performance for well-tuned schemas.	− Scaling sometimes slower / more manual than serverless options. − If usage fluctuates a lot, might pay for idle capacity. − Some features like scaling or storage separation may lag behind competitors in flexibility.
Azure Synapse	+ Very flexible: supports both provisioned and serverless; integrates with big data tools (Spark etc.), BI (Power BI), ML. + Good for organizations already invested in Microsoft / Azure; strong enterprise features & security. + Good hybrid capability (data lakes + warehouse in one ecosystem).	− Complexity: because it's broad and supports many modes, sometimes more moving parts. − Cost management can be tricky depending on which mode (serverless vs provisioned) you use. − Performance tuning may require more effort. − Serverless modes sometimes have less predictable performance.

Cost Comparison / Cost-Efficiency Highlights

From comparisons and benchmarks:

- BigQuery tends to be more cost-efficient for **variable workloads**, burst usage, or when many ad-hoc queries are run, because you don't need to maintain provisioned clusters.
- Redshift can become cheaper when workloads are **predictable, large, and sustained**, because you can reserve capacity, benefit from economies of scale. But risk of paying for idle resources.
- Synapse's serverless / dedicated mix gives flexibility; for mixed workloads or where data lake + warehouse + BI + ML all come in, you may get better overall through integrated tooling. But sometimes "serverless SQL" costs can accumulate.

When to Choose Which — Decision Guidance

Here is a decision-flow / criteria you can use to pick among them, depending on what you have / need:

1. **Existing Cloud Ecosystem**
    - If most of your infrastructure/data is already in GCP → BigQuery is likely to have lower data transfer cost, better integration.
    - If AWS is your cloud of record → Redshift s simpler.
    - If Azure + Microsoft tools are heavily used (BI via Power BI, Azure ML, Azure Data Lake) → Synapse fits well.
2. **Workload Patterns (Stable vs Bursty)**
    - Bursty / variable workloads → serverless or auto-scaling (BigQuery or Synapse serverless); minimal ops overhead.
    - Stable, heavy load → provisioned compute with reserved instances /

dedicated capacities may be more cost effective (Redshift RA3, Synapse dedicated, or BigQuery flat-rate if usage high).

3. **Data Types / Real-Time Needs**
    - If you need real-time ingestion / streaming + fast analytics → BigQuery or Synapse might have simpler paths.
    - If mostly batch ETL and historical analytics, Redshift and Synapse work well.
4. **Control vs Simplicity**
    - Do you need fine control over performance tuning, clusters, hardware? Redshift offers more knobs.
    - If you prefer simplest, lower maintenance/ops, BigQuery or serverless Synapse.
5. **Cost Predictability vs Pay-As-You-Go**
    - If budget predictability is critical: reserved instances (Redshift), flat-rate commitments (BigQuery), provisioned capacity (Synapse) help.
    - If flexibility is more important: on-demand usage.
6. **Security, Compliance, & Governance Requirements**
    - All three are enterprise-grade, but depending on region, data residency, regulatory/compliance requirements, certain features (encryption, auditing, access control) differ and you need to check whether provider supports your requirements out-of-box or with extra configuration.

Summary Recommendation

- Use **BigQuery** if you want minimal operations, quick scaling, flexible cost (especially if your workload is variable or you have a lot of ad-hoc querying), and if you're in the GCP ecosystem.
- Go with **Redshift** if your usage is predictable, you need tight performance tuning, and you're embedded in AWS; also, if you can leverage reserved capacity to lower costs.
- Choose **Azure Synapse** if you want integration with Microsoft stack, need a mix of data warehouse + data lake + big data + BI + ML in one platform, and you may prefer the flexibility of both serverless and provisioned options.

Cost Optimization:

Reference: https://cloud.google.com/bigquery/pricing?hl=en (pricing)

sample cost-estimation / cost-comparison formula for your project (based on your data volumes, query patterns) across all three, so you can see which might be cheapest in your case **with** sample numbers

Assumptions (sample workload)

- **Stored data**: **10 TB** active in the data warehouse.
- **Monthly query volume**: **50 TB** of data processed by queries (on-demand model).
- **Streaming ingestion**: **200 GB / day** → **~6 TB / month** (used where streaming pricing applies).
- Currency: **USD**, region effects ignored (prices vary by region).

Sample Numbers: Pricing references (sources used)

- BigQuery on-demand query pricing: **~$5 per TB** (first 1 TB free). BigQuery storage: **~$0.02 / GB-month (active)**. BigQuery streaming insert pricing: **$0.01 per 200 MB (legacy) / Storage Write API $0.025 per GB**.

- Amazon Redshift (RA3 example) — node / cluster compute pricing examples and managed storage **~$0.024 / GB-month** for managed storage; node hourly pricing depends on node type (example ra3.xl shown in AWS docs).
- Azure Synapse serverless SQL pricing: **~$5 per TB** processed (serverless) and dedicated SQL pool billed by DWU/hr (if used).

Cost estimate — simple monthly totals

1) Google BigQuery (on-demand + streaming)

Rates used:

- Storage: **$0.02 / GB-month**.
- Queries: **$5 / TB processed (on-demand)**.
- Streaming inserts (legacy tabledata.insertAll): **$0.01 per 200 MB** (~$0.01 / 0.2 GB = $0.05/GB equivalent for legacy; Storage Write API newer price ~$0.025/GB). I'll calculate using the legacy streaming unit which is common in docs.

Math:

- Storage: 10 TB = 10,000 GB × $0.02 = **$200 / month**.
- Query processing: 50 TB × $5 = **$250 / month**.
- Streaming cost (6 TB/month ≈ 6,000 GB): 6,000 GB / 0.2 GB per unit = 30,000 units × $0.01 = **$300 / month**.

  Estimated BigQuery total ≈ $200 + $250 + $300 = $750 / month.

Notes / caveats: BigQuery also offers capacity (slot) pricing (flat-rate) that may be cheaper for heavy predictable workloads; streaming Storage Write API newer prices change the streaming total (Storage Write API: $0.025/GB with first 2 TB free). Use the exact streaming mechanism you plan to use to pick the right unit price.

---

2) Amazon Redshift (RA3 cluster example)

Rates used:

- Managed storage: **~$0.024 / GB-month** (RA3 uses managed storage billing).
- Compute: depends on node type & count. Example RA3.xl hourly was used in AWS docs examples (~$3.26/hr per node in an example). We'll show *two cluster sizing scenarios* to make the tradeoffs clear.

Math (Storage):

- Storage: 10 TB = 10,000 GB × $0.024 = **$240 / month**.

Math (Compute examples):

- **Small cluster (2 × ra3.xl)**: 2 × $3.26/hr × 730 hrs ≈ **$4,760 / month** (approx).
- **Medium cluster (4 × ra3.xl)**: 4 × $3.26/hr × 730 ≈ **$9,520 / month**.

Estimated Redshift total (example):

- Small cluster ≈ $4,760 (compute) + $240 (storage) = **≈ $5,000 / month**.
- Medium cluster ≈ $9,520 + $240 = **≈ $9,760 / month**.

Notes / caveats:

- Redshift cost is dominated by compute (cluster hours). If your usage is steady and high, reserved/commitment pricing (1yr/3yr) can cut this dramatically. If usage is light or intermittent you can scale down/pause clusters (some manual overhead) or use Redshift Serverless (different pricing model). Redshift Spectrum (querying S3) adds per-TB scanned fees ($5/TB) for external queries.

3) Azure Synapse Analytics (serverless SQL example)

Rates used:

- Serverless SQL: **~$5 / TB** processed (serverless SQL pool).
- Storage (ADLS Gen2) — approximated similar to $0.02/GB-month for parity with others (actual ADLS pricing may vary by region and tier).

Math:

- Storage: 10 TB ≈ **$200 / month** (using $0.02/GB for parity).
- Query processing (serverless): 50 TB × $5 = **$250 / month**.
  Estimated Synapse serverless total ≈ $450 / month.

Notes / caveats:

- If you choose **Dedicated SQL Pool** instead, cost is per DWU/hr and depends on how many DWUs you provision and how long they run — that can be more expensive than serverless for intermittent workloads, but cheaper/more predictable for heavy sustained workloads if you size/reserve appropriately.

Quick comparison of the sample totals

- **BigQuery (on-demand + streaming)** ≈ **$750 / month** (our sample).
- **Redshift (RA3 cluster)** ≈ **$5,000 → $9,700 / month** depending on cluster size (compute-heavy).
- **Azure Synapse (serverless)** ≈ **$450 / month** (serverless SQL + storage parity assumption).

Interpretation: with the chosen sample numbers and *on-demand/serverless* models, serverless Synapse and BigQuery look lower cost for the given volumes. Redshift's cluster pricing shows the classic tradeoff: high compute capacity (fast) but higher monthly cost unless you have stable, reserved workloads or can aggressively right-size and schedule the cluster.

Concrete cost-optimization actions per platform (how to implement — step-by-step)

BigQuery — practical ops to lower the $750 sample bill

1. **Partition by date** for large time series tables (only scan needed partitions).
   - CREATE TABLE dataset.tbl PARTITION BY DATE(event_date) AS SELECT ...
2. **Cluster** on frequently filtered columns (customer_id, region) to reduce scanned bytes.
3. **Avoid SELECT *****; use SELECT column_list.
4. **Materialized views** for repeated expensive aggregations.
   - CREATE MATERIALIZED VIEW dataset.mv AS SELECT ...
5. **Use batch loads from GCS** for high-volume ingestion instead of streaming if real-time is noncritical (batch loads = free). If you need streaming, prefer Storage Write API pricing when advantageous.
6. **Preview queries / dry-run**: use query dry-run to estimate bytes scanned.
   - In bq CLI: bq query --dry_run --format=prettyjson 'SELECT ...'
7. **Slot reservations** (flat-rate) if steady heavy query volume — can save vs on-demand.
8. **Automate lifecycle**: move old partitions to long-term storage (cost $0.01/GB) after 90 days.

Redshift — practical ops to reduce cost and improve cost predictability

1. **Right-size nodes**: run performance tests and choose smallest RA3 node that meets latency SLOs. Use autoscaling where available.
2. **Use Spectrum** for colder data kept in S3 to avoid loading everything into cluster storage; but Spectrum adds per-TB scanned charges — structure queries to minimize scanned S3 bytes.
3. **Use sort & distribution keys** to avoid costly data shuffles and speed queries.
4. **Pause / snapshot clusters** if intermittent workloads (or use Redshift Serverless for ephemeral workloads).
5. **Reserved instances / savings plans** for predictable workloads — commit 1 or 3 years to reduce compute costs.

Azure Synapse — practical ops (serverless + dedicated)

1. **Choose serverless SQL** for ad-hoc/exploration; costs are per TB processed — use partition pruning and SELECT lists to limit scanned data.
2. **Use dedicated pools (DWUs) for stable heavy workloads** and pause/scale when not used. Estimate DWU hours required from sample query tests.
3. **Store cold data in ADLS Gen2** and query via serverless only when needed; format data as Parquet/ORC to reduce scanned bytes.
4. **Use result caching & materialized views** (where available) to reduce repeated compute.

Final recommendations (based on sample and typical decision criteria)

- If you want **minimal ops** and are already on GCP → **BigQuery**. For our sample, BigQuery ≈ **$750**/mo; if you reduce streaming or switch to batch loads, you can cut the bill further.
- If your workloads are **very steady & heavy**, and you can commit to reservations / reserved nodes — **Redshift** (or dedicated Synapse) can be cost-effective after commitments; but beware big baseline compute cost.
- If you prefer **serverless with low ops** and are on Azure or want cheaper serverless SQL for big data lake queries, **Azure Synapse serverless** can be lower cost (in our sample **~$450**/mo). Use Parquet/partitioning to optimize.

PROJECT

Retail & E-Commerce BigQuery project:

1. **Create datasets & tables** (upload CSVs, connect to Google Sheets)
2. **In the Datasets you can create tables from (**Google Cloud Storage, Upload from your own, Drive, Google Bigtable, Amazon S3, Azure Blob Storage, Existing table/view).
3. **Run ETL SQL scripts** (transformations, joins, aggregations).

Note: ETL stands for Extract, Transform, Load, a data pipeline process used to collect raw data from various sources, convert it into a structured format, and then load it into a target data repository like a data warehouse or data lake for

analysis and decision-making. This process involves gathering diverse data, cleaning and standardizing it according to business rules, and integrating it into a unified system to support business intelligence, data analytics, and machine learning.

The ETL process involves three main steps:

- Extract:

Data is retrieved from various source systems, which can include databases, applications, files, and cloud services.

- Transform:

The extracted data is then cleaned, standardized, validated, and transformed into a format suitable for analysis and storage, often using business rules to organize and structure the raw information.

- Load:

The transformed data is loaded into a central destination, such as a data warehouse or data lake, where it can be accessed for reporting, analysis, and other business purposes.

4. **Build Views & Materialized Views**.
5. **Connect BigQuery to Looker Studio** for dashboards.
6. **Explore Public Datasets** (Google provides free retail, weather, finance datasets).

Step-by-Step Process:

- Create a **new project**
- **Enable BigQuery**
    1. In the left menu → **BigQuery** → **Enable API**.
    2. You'll see the **BigQuery Console (SQL workspace)**.
- **Create a Dataset**
    1. In BigQuery → left panel → **Create Dataset**.
    2. Name it: retail_db.
    3. Location: **US (multi-region)** (cheaper for free tier).
- **Upload Sample Tables**

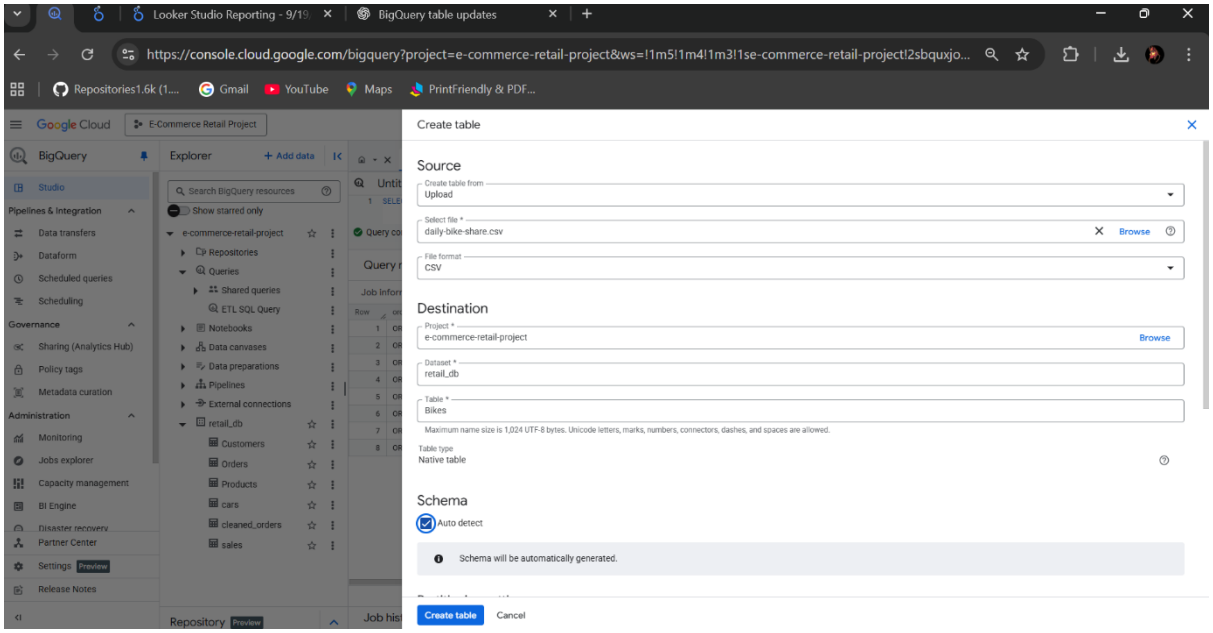
We'll upload the **CSV files** I gave you earlier:

1. customers.csv
2. products.csv
3. orders.csv

For each file:

1. In retail_db → click **Create Table**.
2. Source: **Upload → select CSV**.
3. Table name: customers (repeat for products, orders, cars, Bikes, etc).

4. Schema → choose **Auto Detect**.
5. Click **Create Table**.



➢ **Clean & Transform Data (ETL in BigQuery)**

Open **SQL Workspace** and run this ETL script:

```
-- Step 1: Cleaned Orders Table
CREATE OR REPLACE TABLE `e-commerce-retail-
project.retail_db.cleaned_orders` AS
SELECT
  order_id,
  customer_id,
  product_id,
  order_date,
  quantity,
  price,
  CASE
    WHEN LOWER(region) IN ('north', 'n') THEN 'North'
    WHEN LOWER(region) IN ('south', 's') THEN 'South'
    WHEN LOWER(region) IN ('east', 'e') THEN 'East'
    WHEN LOWER(region) IN ('west', 'w') THEN 'West'
    ELSE 'Unknown'
  END AS region
FROM `e-commerce-retail-project.retail_db.Orders`
WHERE order_id IS NOT NULL
  AND customer_id IS NOT NULL
  AND product_id IS NOT NULL;
```

```
-- Step 2: Sales Summary Fact Table
CREATE OR REPLACE TABLE `e-commerce-retail-
project.retail_db.sales_summary` AS
SELECT
  o.order_id,
  o.order_date,
  o.region,
  c.Customer_ID AS customer_id,
  c.Name AS customer_name,
  c.Country AS country,
  p.Product_ID AS product_id,
  p.Product_Name AS product_name,
  p.Category AS category,
  o.quantity,
  o.price,
  (o.quantity * o.price) AS total_amount
FROM `e-commerce-retail-project.retail_db.cleaned_orders` o
JOIN `e-commerce-retail-project.retail_db.Customers` c
  ON o.customer_id = c.Customer_ID
JOIN `e-commerce-retail-project.retail_db.Products` p
  ON o.product_id = p.Product_ID;

-- Step 3: Analytics Views

-- Total sales by region
CREATE OR REPLACE VIEW `e-commerce-retail-
project.retail_db.vw_sales_by_region` AS
SELECT region, SUM(total_amount) AS total_sales
FROM `e-commerce-retail-project.retail_db.sales_summary`
GROUP BY region;

-- Top 5 customers by spend
CREATE OR REPLACE VIEW `e-commerce-retail-
project.retail_db.vw_top_customers` AS
SELECT customer_name, SUM(total_amount) AS total_spend
FROM `e-commerce-retail-project.retail_db.sales_summary`
GROUP BY customer_name
ORDER BY total_spend DESC
LIMIT 5;
```

```
-- Monthly sales trend
CREATE OR REPLACE VIEW `e-commerce-retail-
project.retail_db.vw_monthly_sales` AS
SELECT
  FORMAT_DATE('%Y-%m', DATE(order_date)) AS month,
  SUM(total_amount) AS monthly_sales
FROM `e-commerce-retail-project.retail_db.sales_summary`
GROUP BY month
ORDER BY month;
```

Test: Run Analytics Queries

list, you can validate every aspect of your dataset (counts, distincts, joins, sales rollups).

Now you can test:

```
SELECT * FROM `e-commerce-retail-project.retail_db.Customers` LIMIT 7;
SELECT * FROM `e-commerce-retail-project.retail_db.Orders` LIMIT 8;
SELECT * FROM `e-commerce-retail-project.retail_db.Products` LIMIT 8;

-- Count rows in each table
SELECT COUNT(*) AS total_orders FROM `e-commerce-retail-
project.retail_db.Orders`;
SELECT COUNT(*) AS total_customers FROM `e-commerce-retail-
project.retail_db.Customers`;
SELECT COUNT(*) AS total_products FROM `e-commerce-retail-
project.retail_db.Products`;

-- Distinct values
SELECT DISTINCT region FROM `e-commerce-retail-project.retail_db.Orders`;
SELECT DISTINCT country FROM `e-commerce-retail
project.retail_db.Customers`;
SELECT DISTINCT category FROM `e-commerce-retail-
project.retail_db.Products`;
```

Orders Analysis

```
-- Orders per region
SELECT region, COUNT(*) AS total_orders
FROM `e-commerce-retail-project.retail_db.cleaned_orders`
GROUP BY region
ORDER BY total_orders DESC;
```

```
-- Monthly order counts
SELECT FORMAT_DATE('%Y-%m', DATE(order_date)) AS month, COUNT(*) AS
order_count
FROM `e-commerce-retail-project.retail_db.Orders`
GROUP BY month
ORDER BY month;

-- Highest quantity order
SELECT order_id, customer_id, product_id, quantity
FROM `e-commerce-retail-project.retail_db.Orders`
ORDER BY quantity DESC
LIMIT 5;
```

Customers Analysis

```
-- Customers by country
SELECT country, COUNT(*) AS total_customers
FROM `e-commerce-retail-project.retail_db.Customers`
GROUP BY country
ORDER BY total_customers DESC;

-- Customers with most orders
SELECT c.Customer_ID, c.Name, COUNT(o.order_id) AS total_orders
FROM `e-commerce-retail-project.retail_db.Customers` c
JOIN `e-commerce-retail-project.retail_db.Orders` o
  ON c.Customer_ID = o.customer_id
GROUP BY c.Customer_ID, c.Name
ORDER BY total_orders DESC
LIMIT 10;
```

Products Analysis

```
-- Products by category
SELECT category, COUNT(*) AS total_products
FROM `e-commerce-retail-project.retail_db.Products`
GROUP BY category
ORDER BY total_products DESC;

-- Most expensive products
SELECT product_id, product_name, category, price
FROM `e-commerce-retail-project.retail_db.Products`
```

```
ORDER BY price DESC
LIMIT 10;
```

Sales Summary (Fact Table)

```
SELECT region, SUM(price) AS price
FROM `e-commerce-retail-project.retail_db.Orders`
GROUP BY region
ORDER BY price DESC;

-- Total sales
SELECT SUM(total_amount) AS total_sales
FROM `e-commerce-retail-project.retail_db.sales_summary`;

-- Sales by country
SELECT country, SUM(total_amount) AS total_sales
FROM `e-commerce-retail-project.retail_db.sales_summary`
GROUP BY country
ORDER BY total_sales DESC;

-- Sales by category
SELECT category, SUM(total_amount) AS total_sales
FROM `e-commerce-retail-project.retail_db.sales_summary`
GROUP BY category
ORDER BY total_sales DESC;

-- Top 10 products by sales
SELECT product_name, category, SUM(total_amount) AS total_sales
FROM `e-commerce-retail-project.retail_db.sales_summary`
GROUP BY product_name, category
ORDER BY total_sales DESC
LIMIT 10;

-- Top 10 customers by spend
SELECT customer_name, SUM(total_amount) AS total_spend
FROM `e-commerce-retail-project.retail_db.sales_summary`
GROUP BY customer_name
ORDER BY total_spend DESC
LIMIT 10;

-- Monthly sales trend
```

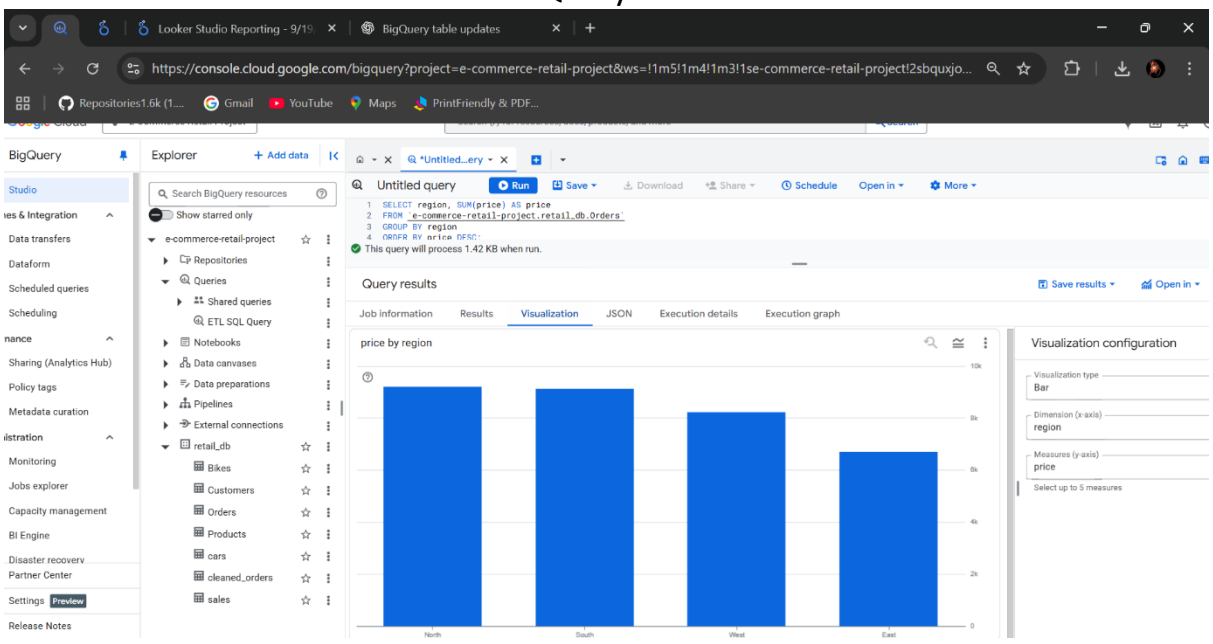
```
SELECT FORMAT_DATE('%Y-%m', DATE(order_date)) AS month,
SUM(total_amount) AS monthly_sales
FROM `e-commerce-retail-project.retail_db.sales_summary`
GROUP BY month
ORDER BY month;
```

Data Quality Checks

```
-- Null checks
SELECT COUNT(*) AS null_orders FROM `e-commerce-retail-project.retail_db.Orders` WHERE order_id IS NULL;
SELECT COUNT(*) AS null_customers FROM `e-commerce-retail-project.retail_db.Customers` WHERE Customer_ID IS NULL;
SELECT COUNT(*) AS null_products FROM `e-commerce-retail-project.retail_db.Products` WHERE Product_ID IS NULL;

-- Duplicate check (orders)
SELECT order_id, COUNT(*) AS dup_count
FROM `e-commerce-retail-project.retail_db.Orders`
GROUP BY order_id
HAVING dup_count > 1;
```

➔You can also Visualisation in the Query results.



- ➢ **Build Dashboard in Looker Studio (**raw data → ETL → summary tables → Looker Studio dashboards.)

1. Go to 👉 https://lookerstudio.google.com.

2. Click **Create → Data Source → Blank Report -> BigQuery**.
3. Select project → dataset (retail_db).
4. Choose tables/views (cars, Bikes, etc.).
5. Build charts:
    - Pie chart → Sales by Region
    - Bar chart → Top Customers
    - Line chart → Monthly Sales
6. You can also select Your template for Vizualizations.

Real-World Scenario (Retail & E-Commerce)

- **Retailers** use BigQuery to combine **sales transactions, customer profiles, and product catalogs**.
- Analytics like **top customers, best-selling products, seasonal trends** help drive:
    - Personalized marketing campaigns
    - Inventory optimization
    - Dynamic pricing

➔Looker Studio Dashboard for Retail & E-Commerce

Step 1: Prepare BigQuery Views

Make sure you already ran the ETL SQL from earlier and you have these views in BigQuery:

- vw_sales_by_region
- vw_top_customers
- vw_monthly_sales
- cars
- Bikes
- Sales

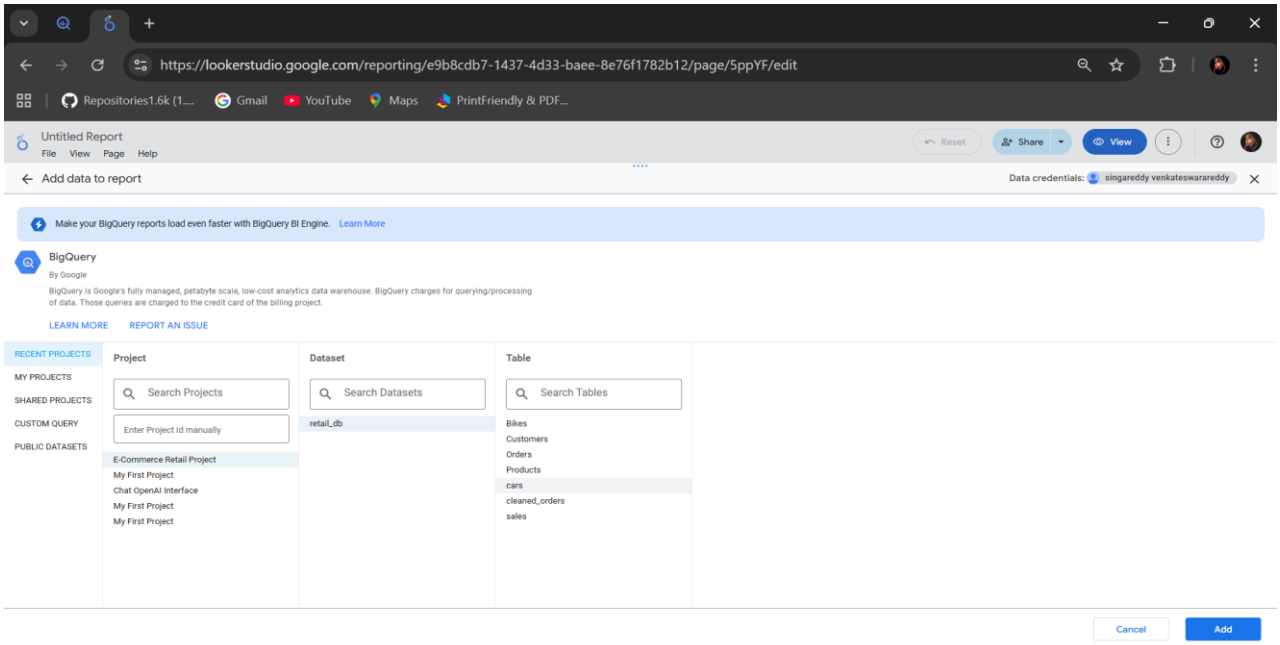
These will feed data directly into Looker Studio.

Step 2: Open Looker Studio

1. Go to 👉 https://lookerstudio.google.com.
2. Sign in with your Google account.
3. Click **Blank Report** (new dashboard).

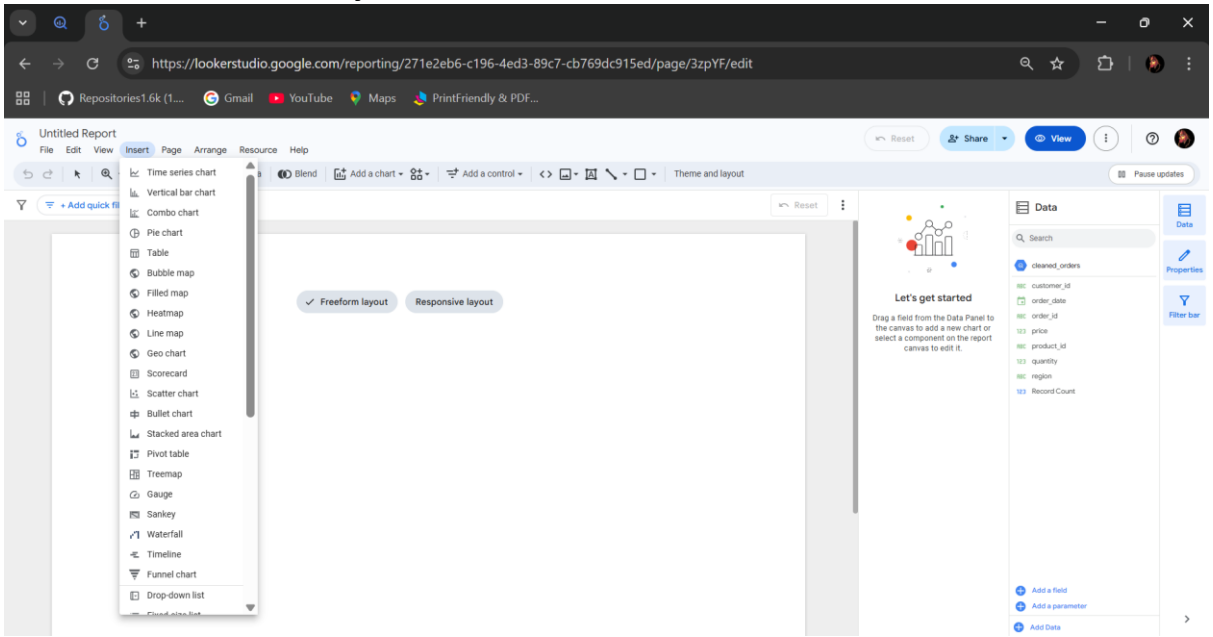
Step 3: Connect BigQuery

1. Click **Add Data → BigQuery**.
2. Choose your project → dataset (retail_db).
3. Select each view (cars, Bikes, sales, etc).
4. Click **Add to Report**.

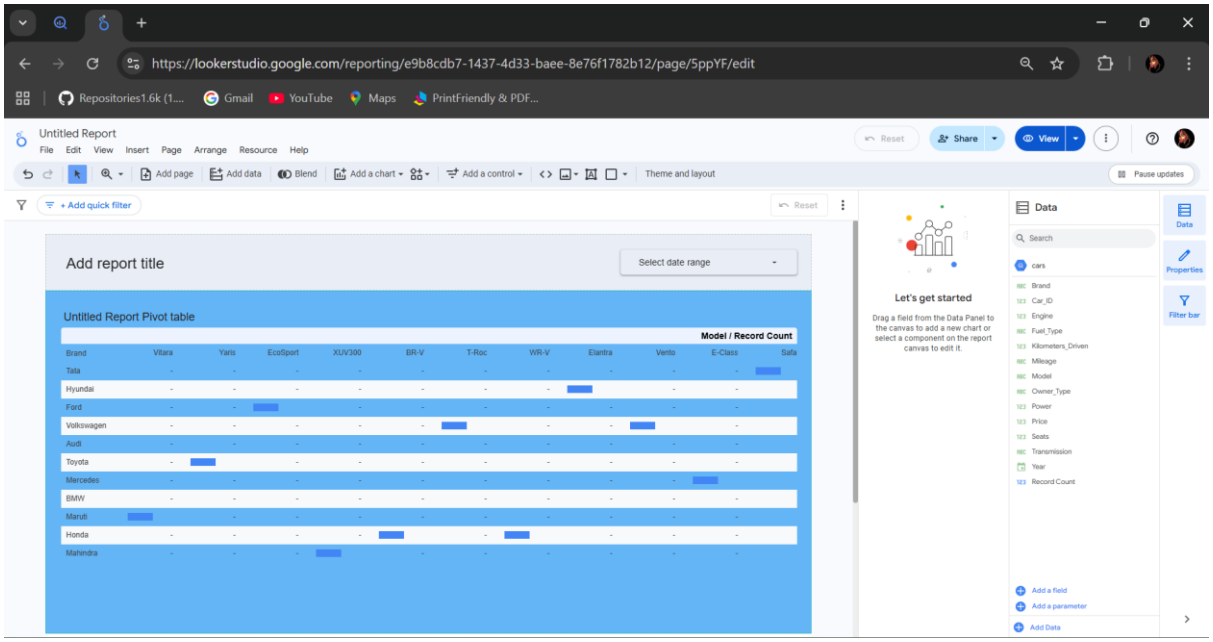


➔**Build Visualizations**

➔**Select the charts as you wish in the Looker Studio for better Visualizations.**



- **KPI Cards**: Total Sales, Total Customers, Top Category
- **Pie Chart**: Sales by Region
- **Bar Chart**: Top Customers
- **Line Chart**: Monthly Sales Trend
- **Date Range Picker**
- **Category Filter**
- **Colorful e-commerce theme** (bright colors, bold titles)



➔Customize Dashboard

- Add filters (e.g., by category or region).
- Add company logo & titles.
- Apply color themes (Looker Studio has ready-made palettes).

➔ Share the Dashboard

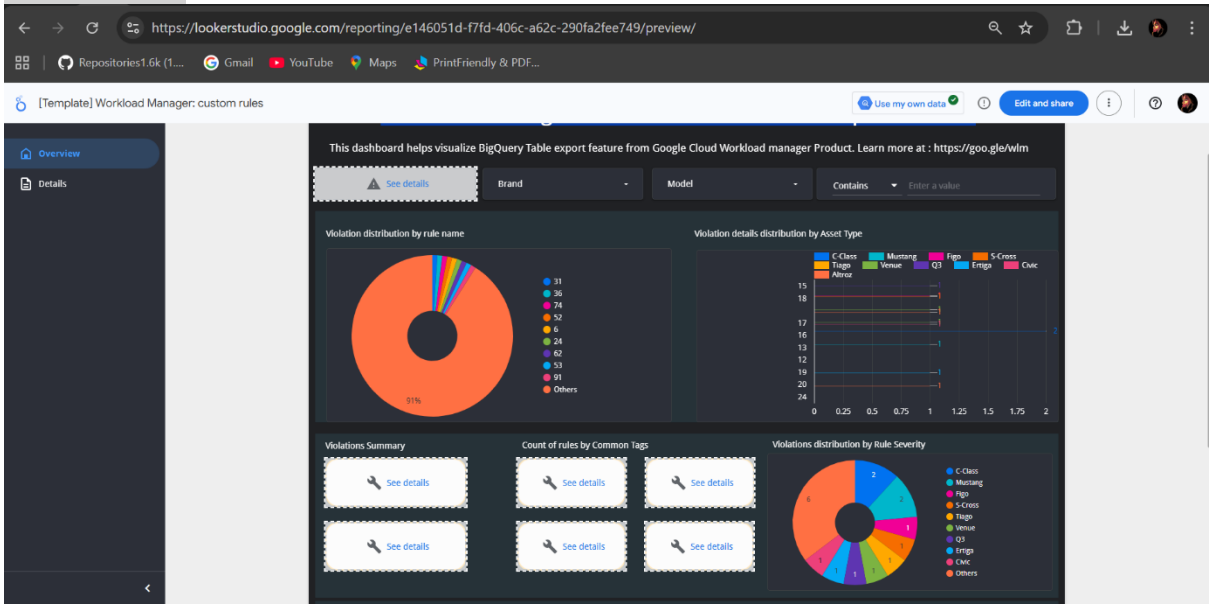
1. Click **Share → Manage Access**.
2. Choose “Anyone with link (Viewer)” if you want public access.
3. Copy & share the dashboard link with your team.

➔Real-World Impact

Retailers use such dashboards for:

- 📈 **Sales monitoring** (see real-time regional sales).
- 👥 **Customer insights** (top buyers, loyalty programs).
- 📦 **Product strategy** (category-wise sales trends).
- 💰 **Business planning** (monthly sales forecasting).

OUTPUT:


https://lookerstudio.google.com/reporting/e146051d-f7fd-406c-a62c-290fa2fee749/preview/
Repositories1.6k (1....
Gmail
YouTube
Maps
PrintFriendly & PDF...
[Template] Workload Manager: custom rules
Use my own data
Edit and share
Overview
Details
This dashboard helps visualize BigQuery Table export feature from Google Cloud Workload manager Product. Learn more at : https://goo.gle/wlm
See details
Brand
Model
Contains
Enter a value
Violation distribution by rule name
91%
31
36
74
52
6
24
62
53
91
Others
Violation details distribution by Asset Type
C-Class
Mustang
Figo
S-Cross
Tiago
Venue
Q3
Ertiga
Civic
Altroz
Violations Summary
See details
Count of rules by Common Tags
Violations distribution by Rule Severity