End-to-End Retail Sales Analytics Pipeline Using Big Query, GCS, SQL Transformations & Power BI Dashboards

Submitted by:

P. Sai Sumeeth

CONTENTS

1. Introduction
2. Business Problem
   Objective of the Project
   Tools & Technologies Used (GCS, Big Query, SQL, Power BI)
3. Data Engineering Pipeline (GCS → Big Query → Power BI)
4. Raw Data Upload to Google Cloud Storage
   Creating External Tables in Big Query
   Creating the Cleaned sales_clean Table
5. Data Transformations and Derived Columns
   Creating Analytical Views
6. Connecting Big Query to Power BI
7. Loading Data Models into Power BI
8. DAX Measures & calculated fiels
9. Error Handling & Challenges
10. Power BI Sign-In Issues
    Data Loading Failures from Big Query
    Materialized Views Not Supported
    Dashboard Visual Errors & Fixes
11. Error & Resolution

1.Introduction

The Retail Sales Analytics project is designed to build a complete end-to-end data pipeline that transforms raw retail data into meaningful business insights. The project leverages cloud-based storage, scalable data warehousing, SQL transformations, and interactive dashboards to provide a comprehensive view of sales, customer behavior, category performance, and revenue trends.

In today's data-driven retail environment, businesses must analyze large volumes of transactional data to understand how customers shop, which products drive sales, when revenue peaks, and where operational improvements can be made. This project replicates a real-world cloud analytics workflow, starting from raw file ingestion into Google Cloud Storage (GCS), processing and modeling the data in BigQuery, creating analytical views, and finally visualizing the insights through Power BI dashboards.

The complete solution demonstrates the full lifecycle of a modern analytics system—from data acquisition and transformation to visualization and decision-making. It showcases not only BI capabilities but also strong data engineering and cloud analytics principles.

---

2. Business Problem

Retail businesses generate high volumes of sales transactions every day, but raw transactional data alone does not provide actionable insights. The data is often unstructured, repetitive, and not ready for analysis. Key challenges for the retail business include:

- Identifying revenue trends and seasonality
- Understanding customer purchase behavior and value segments
- Analyzing product category performance
- Recognizing best-selling months, peak days, and slow periods
- Forecasting future sales trends
- Providing management with a unified view of business performance

Without a centralized analytics system, business stakeholders struggle to answer critical questions such as:

- *Which categories perform well and why?*
- *Which customers contribute the most revenue?*
- *What buying patterns exist across genders and age groups?*
- *How does revenue fluctuate across time?*
- *Which months require aggressive marketing strategies?*

The goal of this project is to solve these challenges by creating a complete analytics pipeline that delivers clean, transformed, and insightful data through interactive dashboards.

---

3. Objective of the Project

The main objective of this project is to build a fully automated, scalable retail analytics ecosystem using Google Cloud and Power BI. The specific goals include:

✓ 1. Build an ELT Pipeline

- Upload raw retail data into Google Cloud Storage
- Create external and cleaned tables in BigQuery
- Perform transformations to make data analytics-ready

✓ 2. Develop Analytical Data Models

- Create views for category analysis, customer insights, revenue trends, and forecasts
- Clean and enrich the dataset with calculated fields like month names, year, day, and total amounts

✓ 3. Create Interactive Power BI Dashboards

Build 7 fully functional dashboards that analyze:

- Overall sales performance
- Product category insights
- Customer behavior and spending patterns
- Sales trends over months and years
- Revenue forecasting using moving averages

- Customer segmentation and demographic influence
- Category performance across age groups and genders

✓ 4. Provide Business Insights

Enable stakeholders to answer key strategic questions such as:

- Who are the top customers?
- Which categories generate the most revenue?
- What trends exist across different age groups?
- Which months have highest/lowest sales?
- How stable or volatile is the revenue stream?
- What is the expected direction of future sales?

✓ 5. Ensure Scalability and Real-World Implementation

Use cloud solutions (GCS + BigQuery) to create a pipeline that can handle larger datasets and refresh dashboards efficiently—mirroring real enterprise environments.

---

4. Tools & Technologies Used

This project uses a combination of cloud storage, data warehousing, SQL modeling, and business intelligence tools.

1. Google Cloud Storage (GCS)
   - Used to store raw CSV files
   - Acts as the landing zone for ingested data
   - Supports external table creation in BigQuery

2. BigQuery (Google Cloud Data Warehouse)
   - Used to create external tables from GCS
   - Performed ELT transformations using SQL
   - Created cleaned data model (sales_clean)
   - Built analytical views for dashboards
   - Offers high performance for large datasets

3. SQL (BigQuery Standard SQL)

- Used for cleaning data
- Creating derived fields (year, month, day, month-name)
- Aggregation logic for category and customer views
- Calculated metrics like total revenue, quantity sold, AOV

4. Power BI Desktop

- Connected directly to BigQuery
- Loaded tables and views for reporting
- Built 7 interactive dashboards
- Created DAX measures for KPIs, moving averages, and segmentation
- Designed charts, slicers, and visual layouts

5. DAX (Data Analysis Expressions)

Used in Power BI to create measures such as:

- Total Revenue
- Total Orders
- Average Order Value
- Moving Averages (MA7, MA30)
- Customer segmentation logic

6. Google Cloud IAM Permissions

- Ensured proper access to BigQuery and GCS
- Allowed Power BI authentication

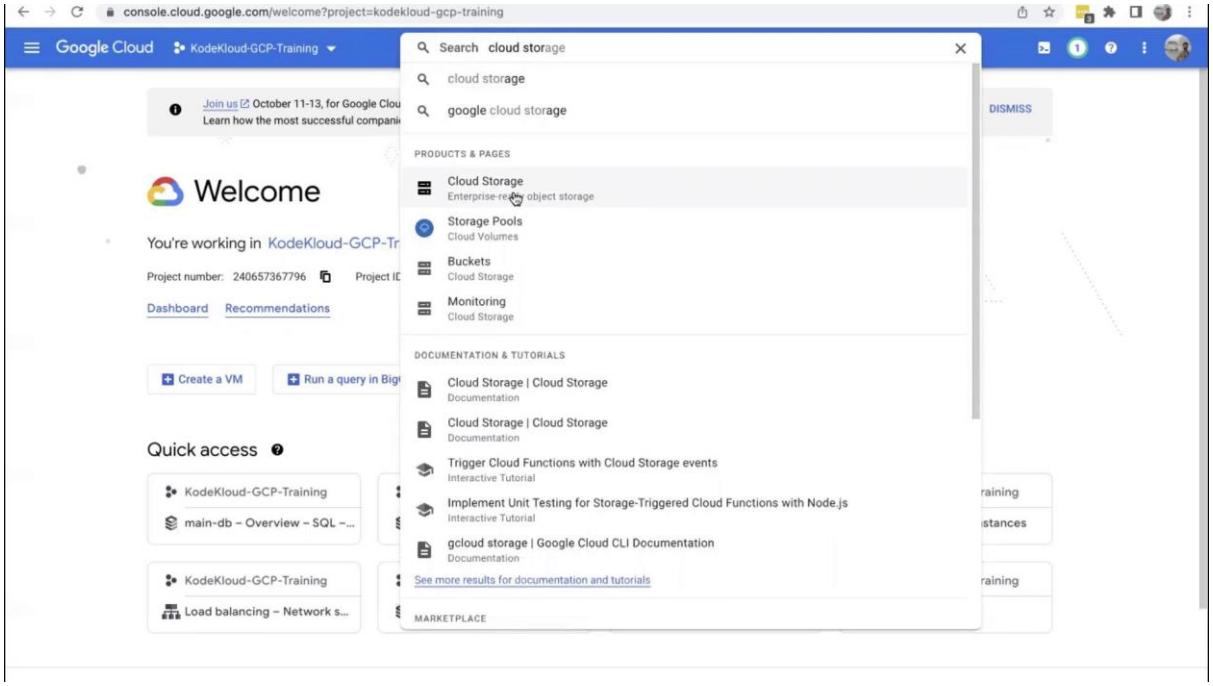
BigQuery to Power BI ETL & Reporting Pipeline

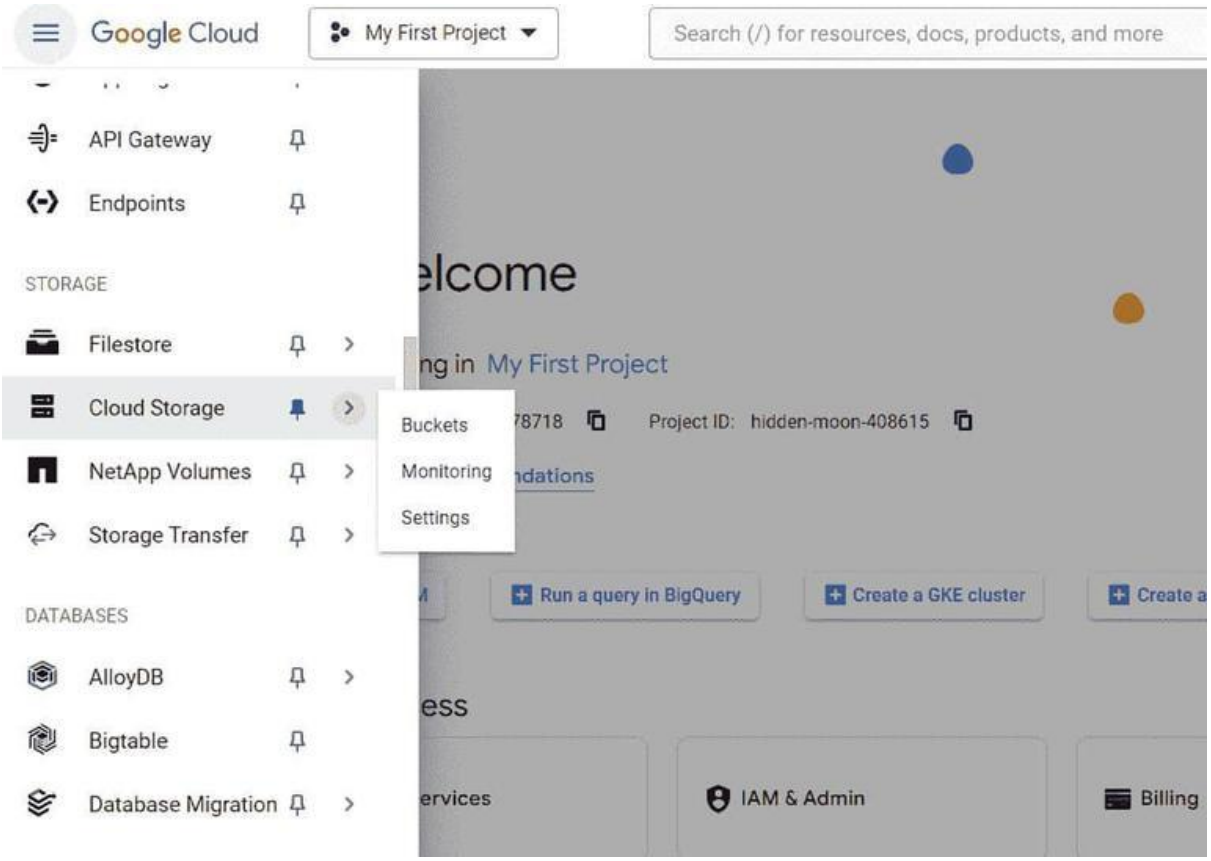
This document explains the end-to-end process used to load retail sales data from Google Cloud Storage (GCS) into BigQuery, transform it using SQL, create analytical views, and then connect and load the processed data into Power BI for reporting and dashboard creation.
Every step is part of a proper ELT Pipeline (Extract → Load → Transform) and reflects real industry practice.

---

1. Creating a Cloud Storage Bucket & Uploading the CSV File

The first step in the pipeline was to upload the raw dataset into Google Cloud Storage (GCS), where BigQuery can read it directly.



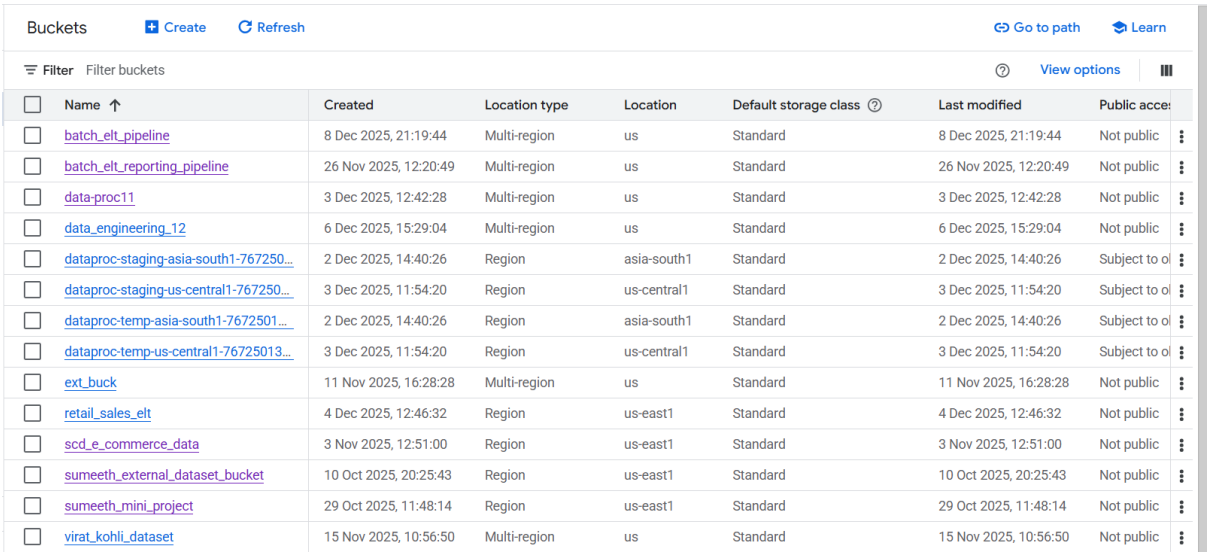


✓ Step Performed

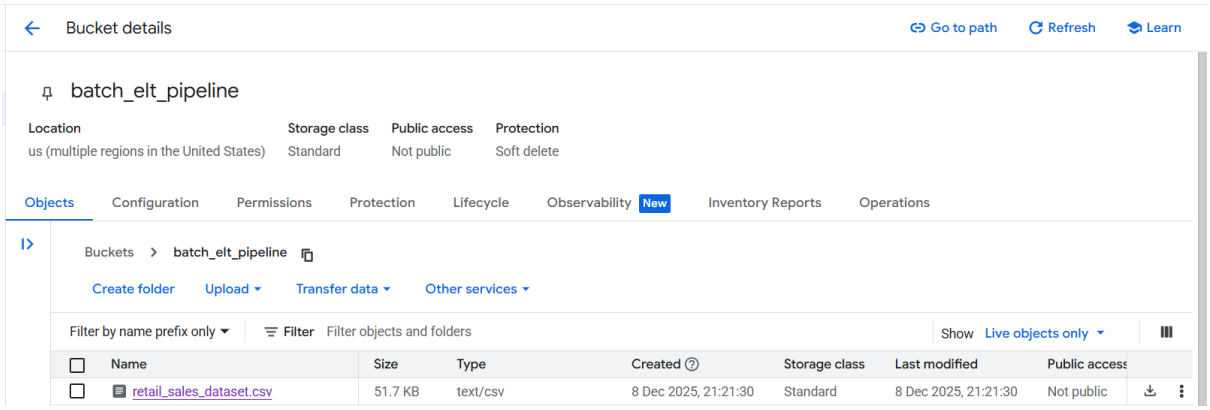
- Created a new bucket named retail_sales_elt
- Selected the region us-east1 (South Carolina)
- Chose Standard Storage Class (best for frequent access)
- Set public access to off and used soft-delete protection
- Uploaded the raw file powerbiproject_ETL_converted.csv

Why this step is important

Cloud Storage acts as the landing zone for your raw data.
BigQuery reads the file from here to create an external table without physically copying the data.



Uploading the file into the GCS bucket:



---

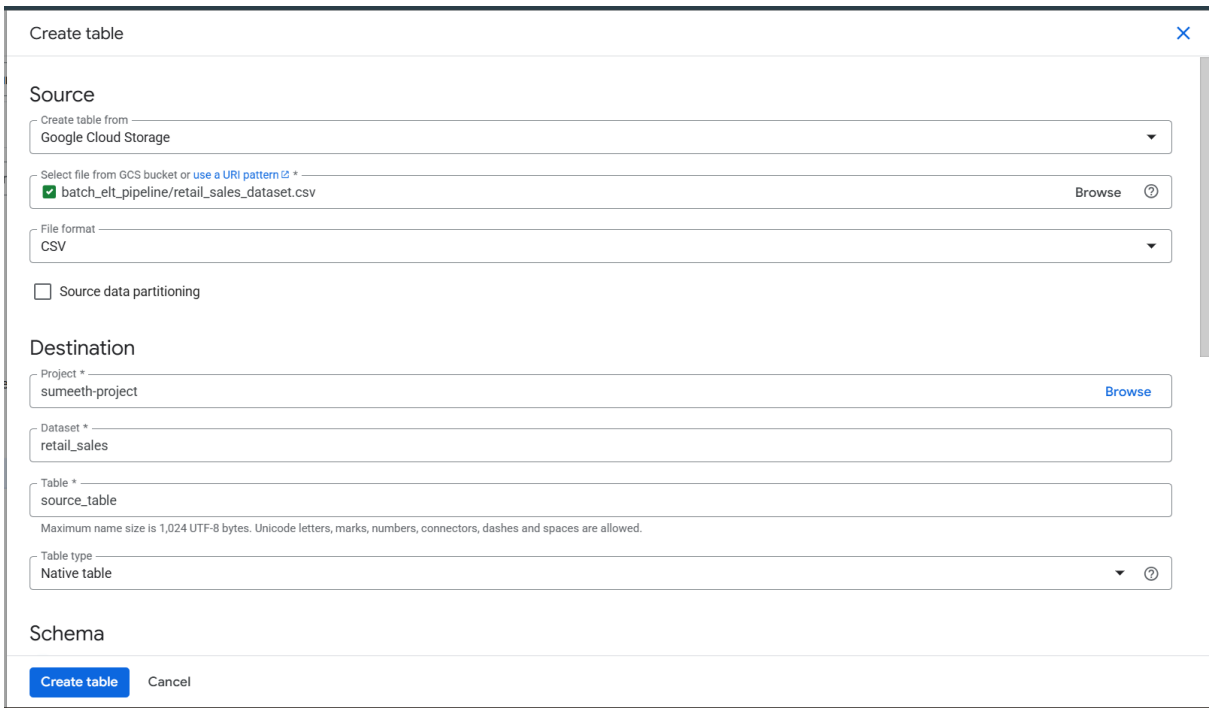
2. Creating an External Table in BigQuery (source_table)

Next, an external table was created in BigQuery referencing the file stored in GCS.

✓ Purpose of External Table

- External tables allow querying data directly from GCS
- No need to load/import data into BigQuery storage
- Perfect for the *staging* step of ELT workflows

This table was named:
retail_sales.source_table



This is the schema of our external table:

Schema | Details | Preview | Table explorer | Preview

Field name	Type	Mode	Desc
Transaction ID	INTEGER	NULLABLE	-
Date	DATE	NULLABLE	-
Customer ID	STRING	NULLABLE	-
Gender	STRING	NULLABLE	-
Age	INTEGER	NULLABLE	-
Product Category	STRING	NULLABLE	-
Quantity	INTEGER	NULLABLE	-
Price per Unit	INTEGER	NULLABLE	-
Total Amount	INTEGER	NULLABLE	-

Checking if the data is loaded correctly into the table:

Select * from `sumeeth-poject.retail_sales.source_table` limit 10;

Schema | Details | Preview | Table explorer Preview | Insights | Lineage | Data profile | Data Quality

Row	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quant
1	191	2023-10-18	CUST191	Male	64	Beauty	
2	204	2023-09-28	CUST204	Male	39	Beauty	
3	230	2023-04-23	CUST230	Male	54	Beauty	
4	232	2023-02-06	CUST232	Female	43	Beauty	
5	309	2023-12-23	CUST309	Female	26	Beauty	
6	310	2023-10-12	CUST310	Female	28	Beauty	
7	363	2023-06-03	CUST363	Male	64	Beauty	
8	371	2023-02-21	CUST371	Female	20	Beauty	
9	397	2023-03-10	CUST397	Female	30	Beauty	
10	454	2023-02-22	CUST454	Female	46	Beauty	
11	512	2023-11-07	CUST512	Female	57	Beauty	
12	739	2023-11-29	CUST739	Male	36	Beauty	

3. Here we will be checking our data is cleanzed/not cleanzed by checking the data

-- checking for NULL values

```
SELECT
  COUNTIF(`Transaction ID` IS NULL) AS Null_TransactionID,
  COUNTIF(Date IS NULL) AS Null_Date,
  COUNTIF(`Customer ID` IS NULL) AS Null_CustomerID,
  COUNTIF(Gender IS NULL) AS Null_Gender,
  COUNTIF(Age IS NULL) AS Null_Age,
  COUNTIF(`Product Category` IS NULL) AS Null_ProductCategory,
  COUNTIF(Quantity IS NULL) AS Null_Quantity,
  COUNTIF(`Price per Unit` IS NULL) AS Null_PricePerUnit,
  COUNTIF(`Total Amount` IS NULL) AS Null_TotalAmount
FROM sumeeth-project.retail_sales.source_table;



HAVING dup_count > 1;
```

```sql
-- checking for NULL values
SELECT
  COUNTIF(`Transaction ID` IS NULL) AS Null_TransactionID,
  COUNTIF(Date IS NULL) AS Null_Date,
  COUNTIF(`Customer ID` IS NULL) AS Null_CustomerID,
  COUNTIF(Gender IS NULL) AS Null_Gender,
  COUNTIF(Age IS NULL) AS Null_Age,
  COUNTIF(`Product Category` IS NULL) AS Null_ProductCategory,
  COUNTIF(Quantity IS NULL) AS Null_Quantity,
  COUNTIF(`Price per Unit` IS NULL) AS Null_PricePerUnit,
  COUNTIF(`Total Amount` IS NULL) AS Null_TotalAmount
FROM sumeeth-project.retail_sales.source_table;
```

Query completed

Using on-demand processing quota

Query results

Save results | Open in

Job information | Results | Visualisation | JSON | Execution details | Execution graph

ull_Date	Null_CustomerID	Null_Gender	Null_Age	Null_ProductCate...	Null_Quantity	Null_PricePerUnit	Null_TotalAmount
0	0	0	0	0	0	0	0

-- checking for empty strings

SELECT *

FROM retail_sales.source_table

WHERE TRIM(Gender) = ''

  OR TRIM(`Product Category`) = '';

```sql
-- checking for empty strings
SELECT *
FROM retail_sales.source_table
WHERE TRIM(Gender) = ''
   OR TRIM(`Product Category`) = '';
```

Query completed

Using on-demand processing quota

Query results

Save results | Open in

Job information | Results | Visualisation | JSON | Execution details | Execution graph

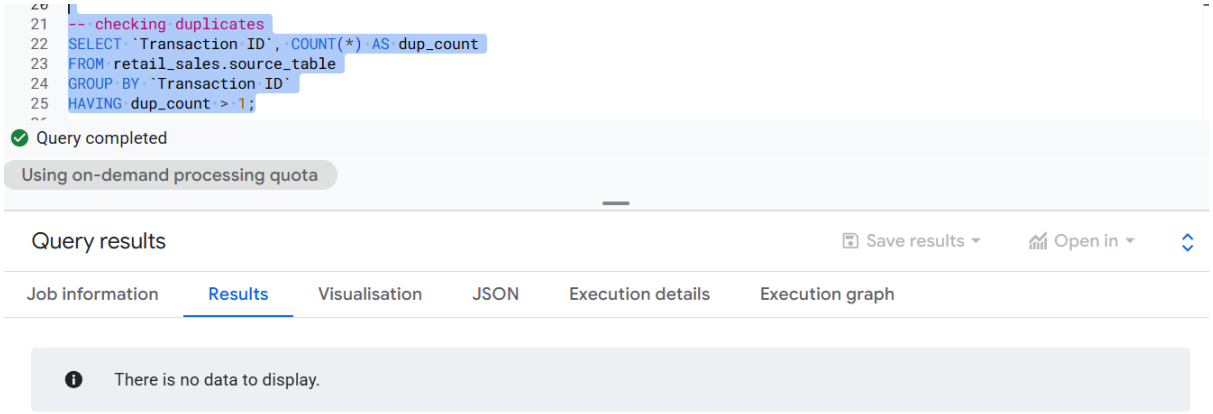
There is no data to display.

-- checking duplicates

SELECT `Transaction ID`, COUNT(*) AS dup_count

FROM retail_sales.source_table

GROUP BY `Transaction ID`



-- Check Negative or Zero Values

```
SELECT *
FROM retail_sales.source_table
WHERE SAFE_CAST(Quantity AS INT64) <= 0
  OR SAFE_CAST(`Price per Unit` AS FLOAT64) <= 0
  OR SAFE_CAST(`Total Amount` AS FLOAT64) <= 0;
```

-- checking category consistency

```
SELECT DISTINCT `Product Category`
FROM retail_sales.source_table
ORDER BY 1;
```

```sql
-- checking category consistency
SELECT DISTINCT `Product Category`
FROM retail_sales.source_table
ORDER BY 1;
```

Query completed

Using on-demand processing quota

Query results

Save results | Open in

Job information | Results | Visualisation | JSON | Execution details | Execution graph

There is no data to display.

-- Check for duplicate Customer IDs

```sql
SELECT
 `Customer ID`,
 COUNT(*) AS Num_Transactions
FROM retail_sales.source_table
GROUP BY `Customer ID`
HAVING COUNT(*) > 1
ORDER BY Num_Transactions DESC;
```

```sql
-- Check for duplicate Customer IDs
SELECT
  `Customer ID`,
  COUNT(*) AS Num_Transactions
FROM retail_sales.source_table
GROUP BY `Customer ID`
HAVING COUNT(*) > 1
ORDER BY Num_Transactions DESC;
```

This script will process 244.84 KB when run.

Using on-demand processing quota

Query results

Save results | Open in

Job information | Results | Visualisation | JSON | Execution details | Execution graph

There is no data to display.

-- checking if total amount<0 and date>current_date

```sql
SELECT *
FROM retail_sales.source_table
WHERE
    `Total Amount` < 0        -- negative amount
```

OR Date > CURRENT_DATE(); -- future date

```
49  -- checking if total amount<0 and date>current_dae
50  SELECT *
51  FROM retail_sales.source_table
52  WHERE
53      `Total Amount` < 0          -- negative amount
54      OR Date > CURRENT_DATE();   -- future date
```

This script will process 317.52 KB when run.

Using on-demand processing quota

Query results Save results Open in

Job information | Results | Visualisation | JSON | Execution details | Execution graph

There is no data to display.

-- check if Customer IDs are missing, empty, or zero

SELECT *

FROM `sumeeth-project.retail_sales.source_table`

WHERE

`Customer ID` IS NULL

OR TRIM(`Customer ID`) = ''

OR `Customer ID` = '0'; -- only if IDs should never be 0

```
56  -- check if Customer IDs are missing, empty, or zero
57  SELECT *
58  FROM `sumeeth-project.retail_sales.source_table`
59  WHERE
60      `Customer ID` IS NULL
61      OR TRIM(`Customer ID`) = ''
62      OR `Customer ID` = '0';   -- only if IDs should never be 0
63
```

Using on-demand processing quota

Query results Save results Open in

Job information | Results | Visualisation | JSON | Execution details | Execution graph

There is no data to display.

4.Your SQL transformation query:

After creating the external table, a new cleaned and transformed table sales_clean was created using SQL.

```
CREATE OR REPLACE TABLE `sumeeth-project.retail_sales.staging_table` AS
SELECT
  ROW_NUMBER() OVER(ORDER BY Date, `Transaction ID`) AS Row_ID,

  -- ORIGINAL COLUMNS
  `Transaction ID` AS Transaction_ID,
  Date,
  `Customer ID` AS Customer_ID,
  INITCAP(TRIM(Gender)) AS Gender,
  Age,
  INITCAP(TRIM(`Product Category`)) AS Product_Category,
  Quantity,
  `Price per Unit` AS Price_per_Unit,
  `Total Amount` AS Total_Amount,

  -- DATE INTELLIGENCE
  EXTRACT(YEAR FROM Date) AS Year,
  EXTRACT(QUARTER FROM Date) AS Quarter,
  CONCAT('Q', EXTRACT(QUARTER FROM Date)) AS Quarter_Name,
  EXTRACT(MONTH FROM Date) AS Month,
  FORMAT_DATE('%B', Date) AS Month_Name,
  EXTRACT(WEEK FROM Date) AS Week_Number,
  EXTRACT(DAY FROM Date) AS Day,
  FORMAT_DATE('%A', Date) AS Day_Name,
  FORMAT_DATE('%Y-%m', Date) AS Year_Month,
```

```
-- CUSTOMER SEGMENTATION
CASE
  WHEN Age BETWEEN 13 AND 17 THEN 'Teen'
  WHEN Age BETWEEN 18 AND 25 THEN 'Young Adult'
  WHEN Age BETWEEN 26 AND 35 THEN 'Adult'
  WHEN Age BETWEEN 36 AND 50 THEN 'Middle Age'
  ELSE 'Senior'
END AS Age_Group,

-- PRODUCT SEGMENTATION
CASE
  WHEN LOWER(`Product Category`) LIKE '%electronics%' THEN 'Electronics'
  WHEN LOWER(`Product Category`) LIKE '%beauty%' THEN 'Beauty'
  WHEN LOWER(`Product Category`) LIKE '%clothing%' THEN 'Clothing'
  ELSE 'Other'
END AS Product_Group,

-- VALUE & SALES METRICS
SAFE_DIVIDE(`Total Amount`, Quantity) AS Revenue_Per_Unit,

CASE
  WHEN `Total Amount` < 500 THEN 'Small'
  WHEN `Total Amount` BETWEEN 500 AND 1399 THEN 'Medium'
  ELSE 'Large'
END AS Transaction_Size,

CASE
  WHEN Quantity <= 1 THEN 'Low Qty'
  WHEN Quantity BETWEEN 2 AND 3 THEN 'Medium Qty'
  ELSE 'High Qty'
```

```
  END AS Quantity_Segment,

  CASE
    WHEN `Price per Unit` < 100 THEN 'Low Price'
    WHEN `Price per Unit` BETWEEN 100 AND 299 THEN 'Medium Price'
    ELSE 'High Price'
  END AS Price_Segment,

  CASE
    WHEN `Total Amount` BETWEEN 1500 AND 2000 THEN 'High Value'
    WHEN `Total Amount` BETWEEN 500 AND 1499 THEN 'Mid Value'
    ELSE 'Low Value'
  END AS Value_Segment,

  -- DAY TYPE
  CASE
    WHEN FORMAT_DATE('%A', Date) IN ('Saturday', 'Sunday') THEN 'Weekend'
    ELSE 'Weekday'
  END AS Day_Type

FROM `sumeeth-project.retail_sales.source_table`;
```

This is how our schema looks like

☐	Date	DATE	NULLABLE	-	-	-	-
☐	Customer_ID	STRING	NULLABLE	-	-	-	-
☐	Gender	STRING	NULLABLE	-	-	-	-
☐	Age	INTEGER	NULLABLE	-	-	-	-
☐	Product_Category	STRING	NULLABLE	-	-	-	-
☐	Quantity	INTEGER	NULLABLE	-	-	-	-
☐	Price_per_Unit	INTEGER	NULLABLE	-	-	-	-
☐	Total_Amount	INTEGER	NULLABLE	-	-	-	-
☐	Year	INTEGER	NULLABLE	-	-	-	-
☐	Quarter	INTEGER	NULLABLE	-	-	-	-
☐	Quarter_Name	STRING	NULLABLE	-	-	-	-
☐	Month	INTEGER	NULLABLE	-	-	-	-
☐	Month_Name	STRING	NULLABLE	-	-	-	-
☐	Week_Number	INTEGER	NULLABLE	-	-	-	-
☐	Day	INTEGER	NULLABLE	-	-	-	-
☐	Day_Name	STRING	NULLABLE	-	-	-	-
☐	Year_Month	STRING	NULLABLE	-	-	-	-
☐	Age_Group	STRING	NULLABLE	-	-	-	-
☐	Product_Group	STRING	NULLABLE	-	-	-	-
☐	Revenue_Per_Unit	FLOAT	NULLABLE	-	-	-	-
☐	Transaction_Size	STRING	NULLABLE	-	-	-	-
☐	Quantity_Segment	STRING	NULLABLE	-	-	-	-
☐	Price_Segment	STRING	NULLABLE	-	-	-	-
☐	Value_Segment	STRING	NULLABLE	-	-	-	-
☐	Day_Type	STRING	NULLABLE	-	-	-	-

FROM `retail_sales.staging_table`;

✓ Transformations performed

- Standardized Customer_ID to uppercase
- Cleaned spacing and capitalized Gender and Product Category
- Added Order_Year, Order_Month, Order_Day, Order_Month_Name
- Calculated a new column: Calculated_Total_Amount
- Cleaned raw data into analytics-friendly format

Why this is important

This table becomes your central fact table, used by reports, dashboards, and views.

---

3. Creating Analytical Views in BigQuery

Creating a materialized view

```
create or replace materialized view retail_sales.mv_target
as select * from retail_sales.target_table;
```

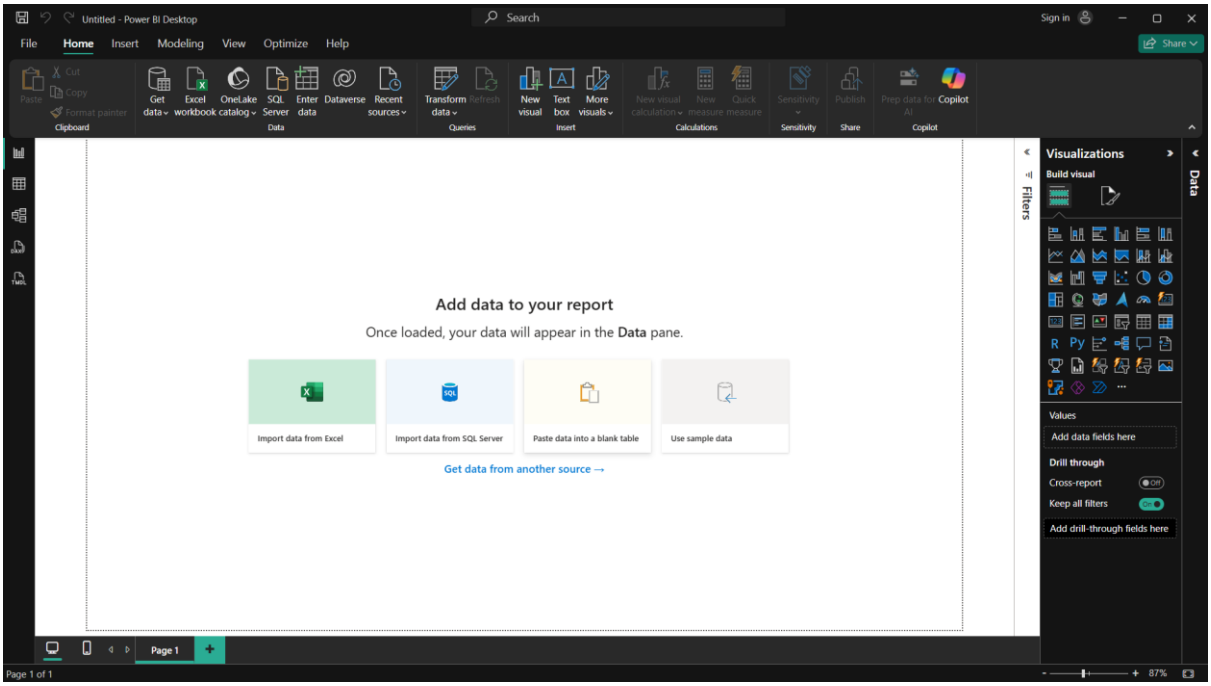
☐	Date	DATE	NULLABLE	-	-	-	-
☐	Customer_ID	STRING	NULLABLE	-	-	-	-
☐	Gender	STRING	NULLABLE	-	-	-	-
☐	Age	INTEGER	NULLABLE	-	-	-	-
☐	Product_Category	STRING	NULLABLE	-	-	-	-
☐	Quantity	INTEGER	NULLABLE	-	-	-	-
☐	Price_per_Unit	INTEGER	NULLABLE	-	-	-	-
☐	Total_Amount	INTEGER	NULLABLE	-	-	-	-
☐	Year	INTEGER	NULLABLE	-	-	-	-
☐	Quarter	INTEGER	NULLABLE	-	-	-	-
☐	Quarter_Name	STRING	NULLABLE	-	-	-	-
☐	Month	INTEGER	NULLABLE	-	-	-	-
☐	Month_Name	STRING	NULLABLE	-	-	-	-
☐	Week_Number	INTEGER	NULLABLE	-	-	-	-
☐	Day	INTEGER	NULLABLE	-	-	-	-
☐	Day_Name	STRING	NULLABLE	-	-	-	-
☐	Year_Month	STRING	NULLABLE	-	-	-	-
☐	Age_Group	STRING	NULLABLE	-	-	-	-
☐	Product_Group	STRING	NULLABLE	-	-	-	-
☐	Revenue_Per_Unit	FLOAT	NULLABLE	-	-	-	-
☐	Transaction_Size	STRING	NULLABLE	-	-	-	-
☐	Quantity_Segment	STRING	NULLABLE	-	-	-	-
☐	Price_Segment	STRING	NULLABLE	-	-	-	-
☐	Value_Segment	STRING	NULLABLE	-	-	-	-
☐	Day_Type	STRING	NULLABLE	-	-	-	-

4. Connecting BigQuery to Power BI

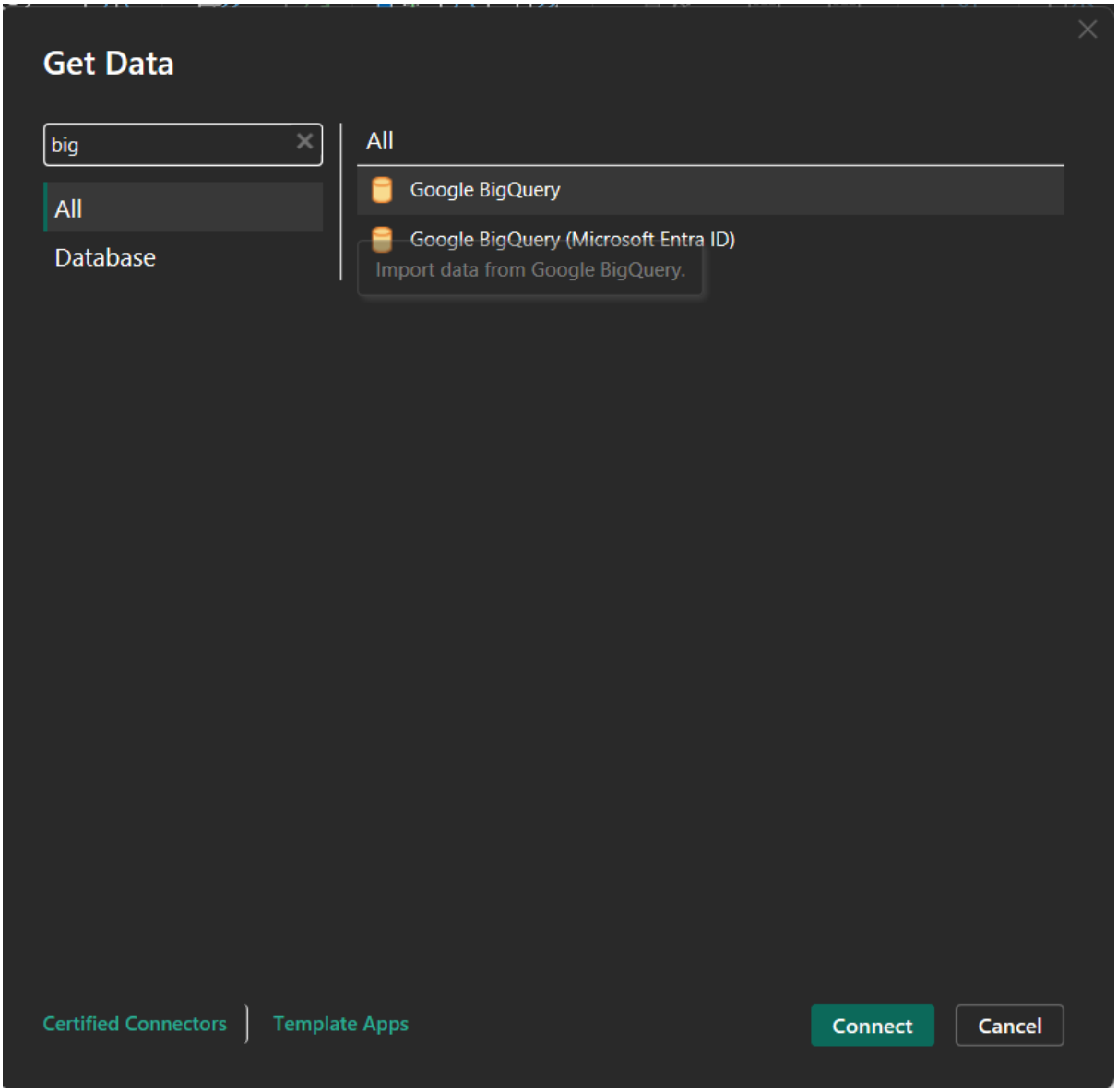
After preparing tables and views in BigQuery, you connected them to Power BI to build dashboards.

✓ Steps performed:

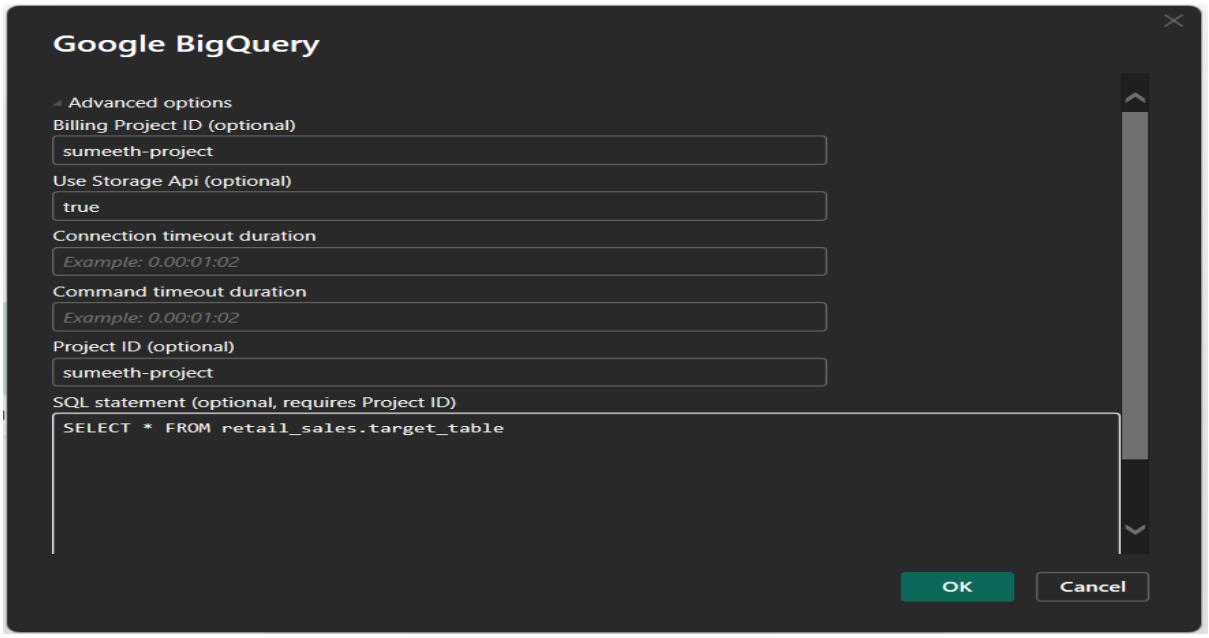
1. Open Power BI → Get Data



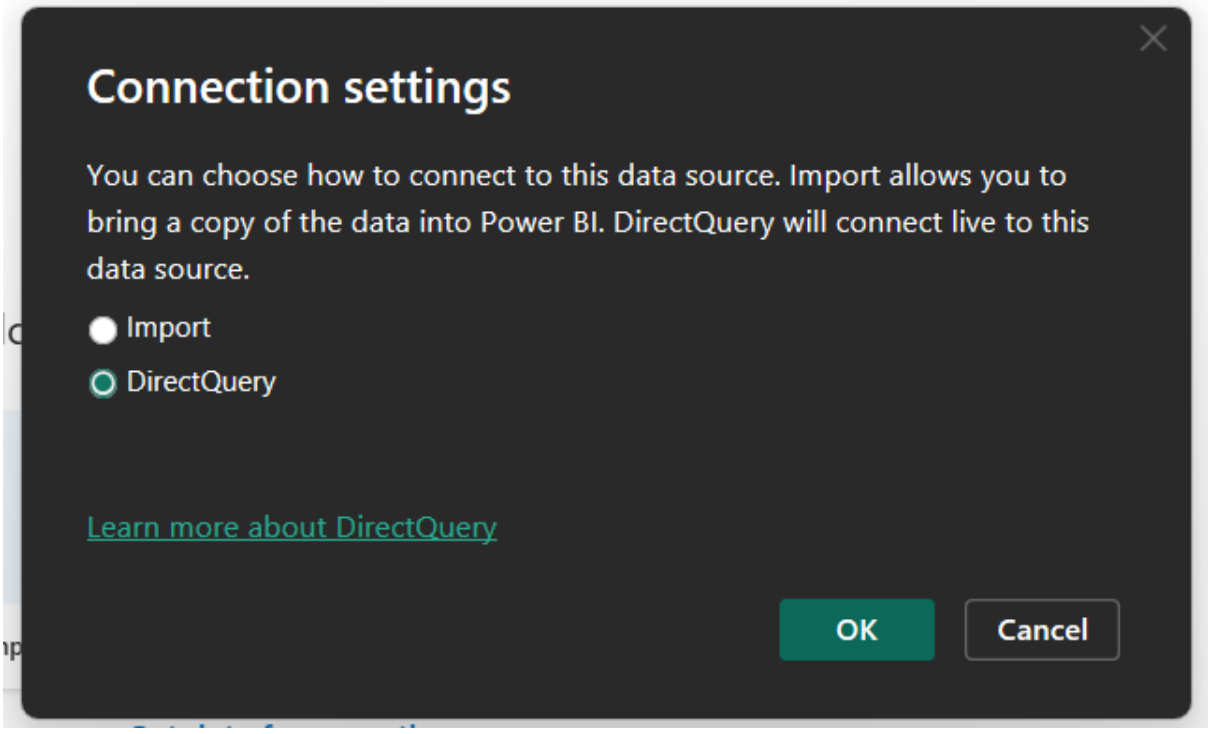
2. Chose Google BigQuery



3. Signed in with your Google Cloud account



4. Selected the project → dataset → views & tables



Why this step is important

Power BI directly communicates with BigQuery, allowing:

- Live data connections
- Automatic refresh
- Query folding (when possible)
- Secure enterprise-level analytics

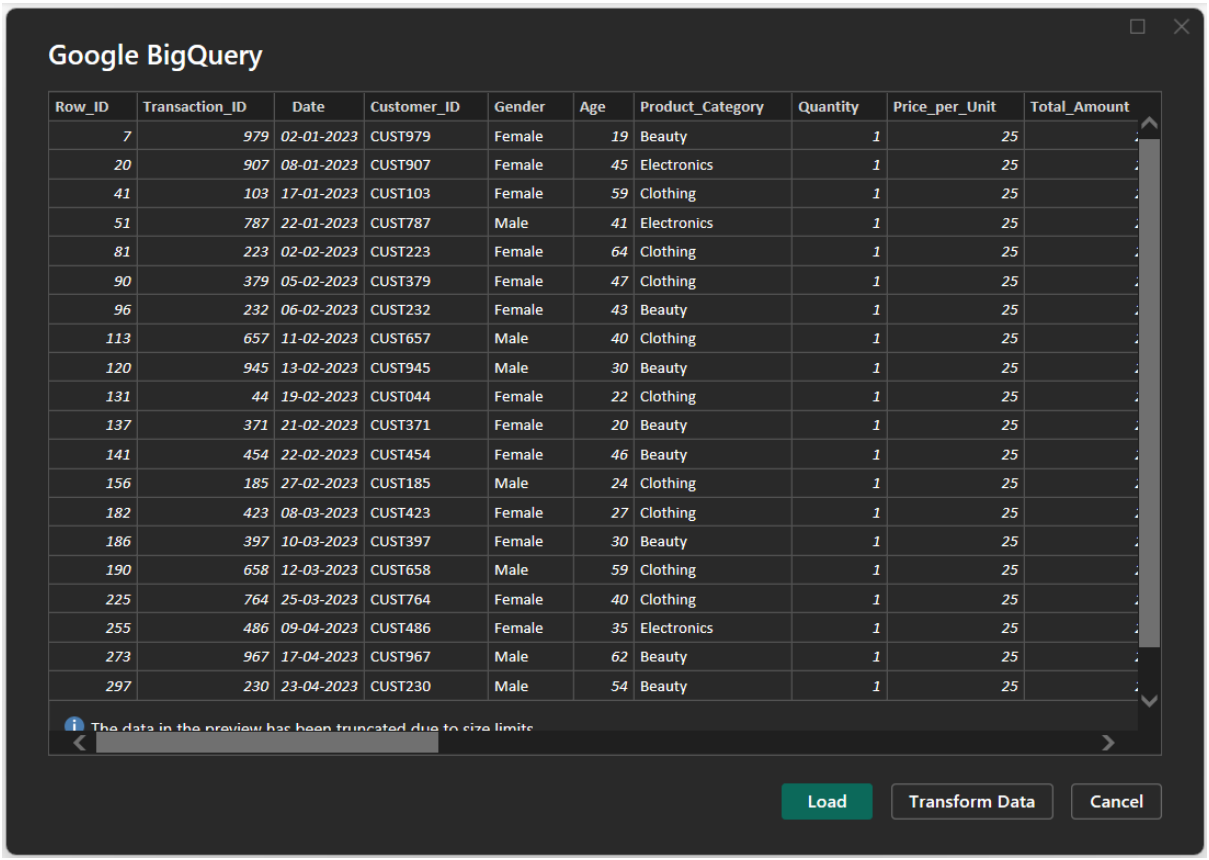
---

6. Data Loading in Power BI

Once the views were selected, Power BI loaded them into the model.

Power BI evaluates each view, converts SQL into queries, retrieves the final transformed structure, and makes it available for visuals.

This ensures that Power BI uses clean, pre-aggregated, and optimized data.



FINAL OUTPUT: Retail Sales ETL Reporting Pipeline

We successfully built an end-to-end ELT + Reporting Pipeline:

1. Data Storage
    - Raw CSV uploaded to GCS bucket
2. Data Loading
    - External table in BigQuery referencing GCS
3. Data Transformation
    - sales_clean table with cleaned & enriched fields
4. Data Modeling
    - Analytical views created for category, customer, daily, gender, and monthly insights
5. Data Visualization
    - Power BI connected directly to BigQuery
    - Views loaded into Power BI
    - Dashboards built on top of transformed data

DAX Measures & Calculated Fields

Table Used: RETAIL_SALES

This section documents all the **DAX measures and calculated fields** created in Power BI for the Retail Sales Analytics project. These measures form the analytical foundation for KPIs, trend analysis, customer segmentation, and forecasting.

🔹 1. Total Revenue

Purpose:
Calculates the total sales revenue generated from all transactions.

```
Total Revenue =
SUM(RETAIL_SALES[Total_Amount])
```

🔹 2. Total Orders

Purpose:
Counts the total number of transactions placed by customers.

```
Total Orders =
COUNT(RETAIL_SALES[Transaction_ID])
```

🔹 3. Total Quantity

Purpose:
Calculates the total number of units sold.

```
Total Quantity =
SUM(RETAIL_SALES[Quantity])
```

🔹 4. Average Order Value (AOV)

Purpose:
Represents the average value of a customer order.

```
AOV =
DIVIDE(
[Total Revenue],
[Total Orders]
)
```

🔹 5. Avg Daily Revenue

Purpose:
Shows the average revenue generated per day.

```
Avg Daily Revenue =
AVERAGEX(
VALUES(RETAIL_SALES[Date]),
[Total Revenue]
)
```

🔹 6. Avg Quantity

Purpose:
Calculates the average quantity sold per transaction.

```
Avg Quantity =
AVERAGE(RETAIL_SALES[Quantity])
```

🔷 7. Monthly Revenue

Purpose:
Aggregates total revenue at a monthly level for trend and forecast analysis.

```
Monthly Revenue =
CALCULATE(
[Total Revenue],
VALUES(RETAIL_SALES[Year Month])
)
```

🔷 8. Moving Average – 7 Days

Purpose:
Smooths short-term daily revenue fluctuations.

```
Moving Avg 7 Days =
AVERAGEX(
DATESINPERIOD(
        RETAIL_SALES[Date],
        MAX(RETAIL_SALES[Date]),
        -7,
        DAY
),
[Total Revenue]
)
```

🔷 9. Moving Average – 30 Days

Purpose:
Displays long-term revenue trends.

```
Moving Avg 30 Days =
AVERAGEX(
DATESINPERIOD(
        RETAIL_SALES[Date],
        MAX(RETAIL_SALES[Date]),
        -30,
        DAY
),
```

```
[Total Revenue]
)
```

🔹 10. Best Day (Highest Revenue Day)

Purpose:
Identifies the day with the highest total revenue.

```
Best Day =
MAXX(
VALUES(RETAIL_SALES[Date]),
[Total Revenue]
)
```

🔹 11. Peak Orders Day

Purpose:
Finds the day with the highest number of orders.

```
Peak Orders Day =
MAXX(
VALUES(RETAIL_SALES[Date]),
COUNT(RETAIL_SALES[Transaction_ID])
)
```

🔹 12. Revenue per Unit

Purpose:
Calculates revenue generated per unit sold.

```
Revenue Per Unit =
DIVIDE(
[Total Revenue],
[Total Quantity]
)
```

🔹 13. Total Revenue per Customer

Purpose:
Shows how much revenue each customer contributes.

```
Total Revenue per Customer =
CALCULATE(
[Total Revenue],
ALLEXCEPT(RETAIL_SALES, RETAIL_SALES[Customer_ID])
```

```
)
```

🔹 14. Top Customer

Purpose:
Identifies the customer with the highest total spend.

```
Top Customer =
MAXX(
VALUES(RETAIL_SALES[Customer_ID]),
[Total Revenue per Customer]
)
```

🔹 15. Revenue (Selected Category)

Purpose:
Calculates revenue dynamically based on selected product category slicer.

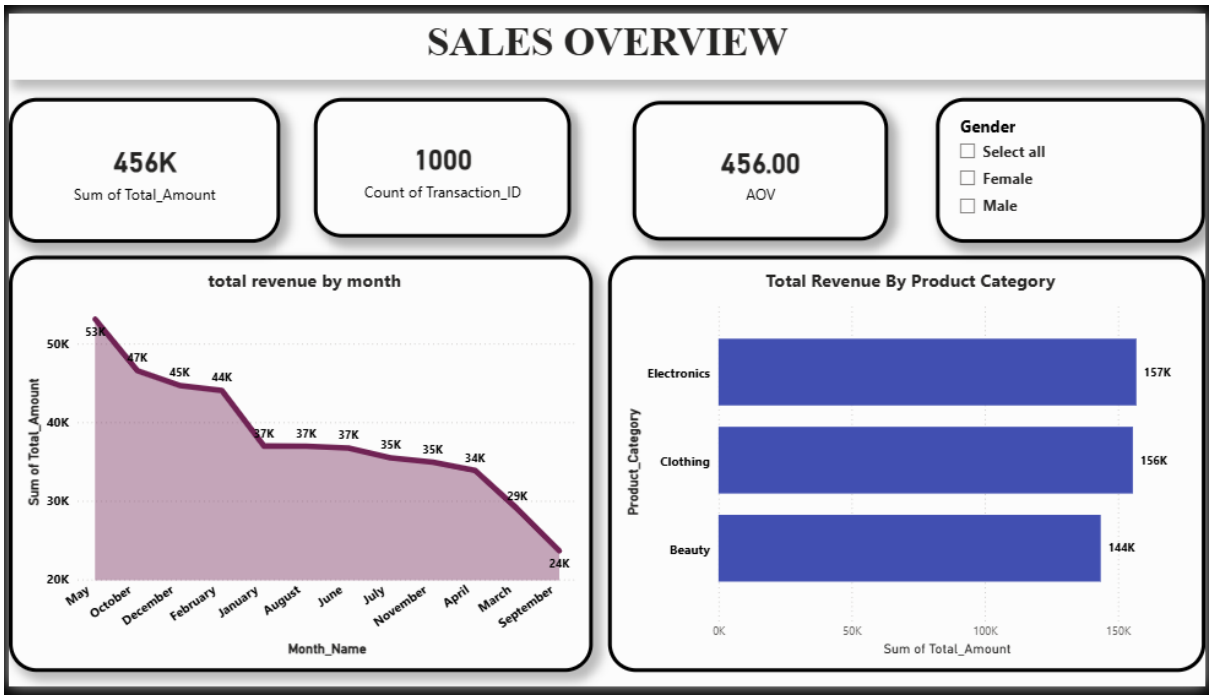
```
Revenue (Selected Category) =
CALCULATE(
[Total Revenue],
VALUES(RETAIL_SALES[Product_Category])
)
```

🔹 16. High / Mid / Low Value Sales

Purpose:
Segments customers based on spending value.

```
High Value Sales =
CALCULATE(
[Total Revenue],
RETAIL_SALES[Value_Segment] = "High"
)
Mid Value Sales =
CALCULATE(
[Total Revenue],
RETAIL_SALES[Value_Segment] = "Medium"
)
Low Value Sales =
CALCULATE(
[Total Revenue],
RETAIL_SALES[Value_Segment] = "Low"
)
```

PAGE 1 – Sales Overview Dashboard



This page serves as the executive summary of the entire retail business. It distills millions of transactional rows into a simple, intuitive view that leadership can understand at a glance.

KPI Cards – What They Tell Us

The KPI cards at the top (Total Revenue, Total Transactions, AOV) give a quick health check of the business.

- **456K Total Revenue** immediately indicates how much the business earned in the selected period.
- **1000 Transactions** show how active the store was.
- **AOV (Average Order Value = 456)** helps stakeholders understand customer spending habits—whether customers tend to make small purchases frequently or fewer but larger purchases.

These KPIs help in evaluating both sales performance and customer buying behavior in one quick snapshot.

Total Revenue by Month (Area Line Chart)

This visualization narrates the story of how revenue evolves over time.

- Months like **May and December** show peak sales, likely due to seasonal demand or promotions.
- Months like **September and April** show dips, indicating slow sales cycles.

Leadership uses this chart to:
✓ Identify seasonal peaks
✓ Plan inventory and marketing budgets
✓ Schedule promotional campaigns
✓ Understand natural demand patterns

Total Revenue by Product Category (Bar Chart)

This chart reveals which product categories—Electronics, Clothing, and Beauty—drive the highest revenue.

- **Electronics (157K)** holds a slight lead, showing it is the most value-contributing category.
- **Clothing (156K)** follows closely, showing stable customer demand.
- **Beauty (144K)** still performs strongly despite being a smaller category.

This helps companies decide where to invest more stock, which categories to expand, and how to design category-specific discounts.

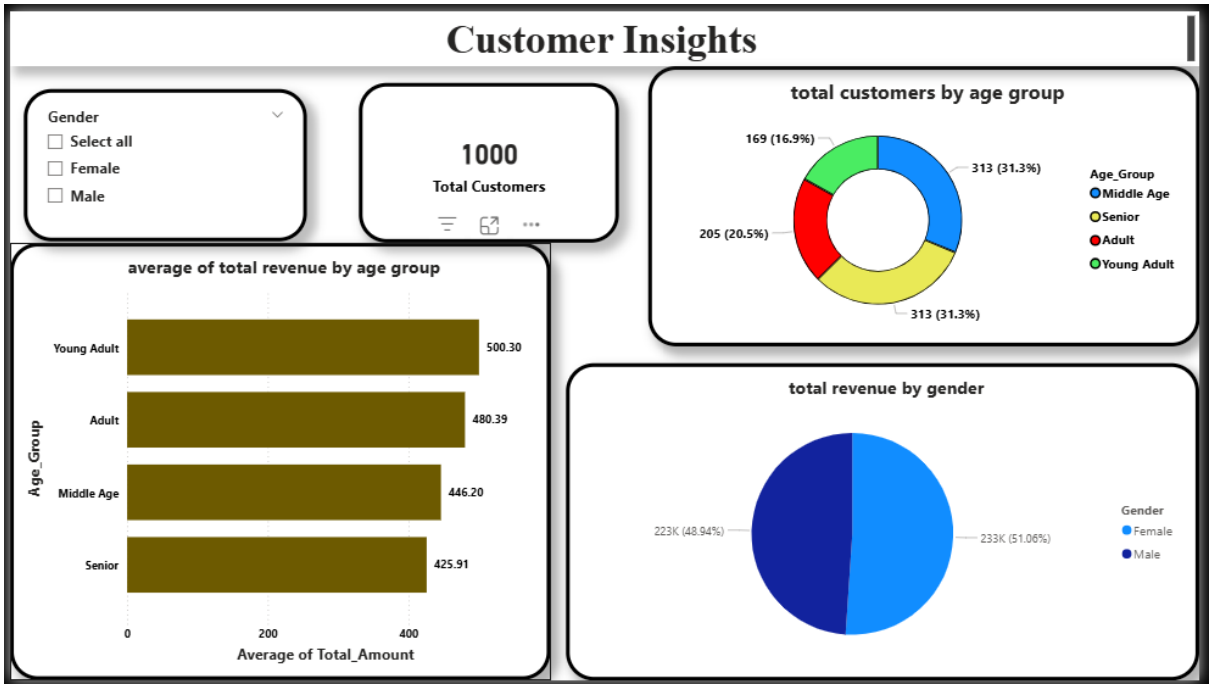
Gender Filter

A gender slicer allows management to explore spending behavior between **male and female customers**, enabling targeted marketing strategies.

Conclusion – Page 1

The Sales Overview dashboard provides an immediate, high-level understanding of organizational performance. By combining KPIs, time-series trends, and category-level comparison, this page equips decision-makers with a strategic snapshot that supports revenue evaluation, planning, and executive reporting.

PAGE 2 – Customer Insights Dashboard



This page focuses on understanding who the customers are, how they behave, and where revenue is coming from demographically.

Total Customers (KPI)

The total customer count (1000) reflects the customer base and helps measure growth in user acquisition.

Customer Distribution by Age Group (Donut Chart)

This visualization shows the demographic split across:

- Young Adult
- Adult
- Middle Age
- Senior

Young Adults and Adults each make up **31.3%** of customers, making them the dominant customer groups.

This insight guides:
✓ Targeted digital marketing
✓ Personalized campaigns
✓ Product assortment planning

Average Revenue by Age Group (Bar Chart)

This chart uncovers the **spending potential of each age group**.

- **Young Adults spend the most (500.30 avg.)**, showing high purchasing power.
- **Seniors spend the least**, indicating they may be more price-sensitive.

Business Impact:
📌 Higher-spending groups should receive premium product recommendations and loyalty incentives.
📌 Lower-spending groups may respond better to discount-oriented campaigns.

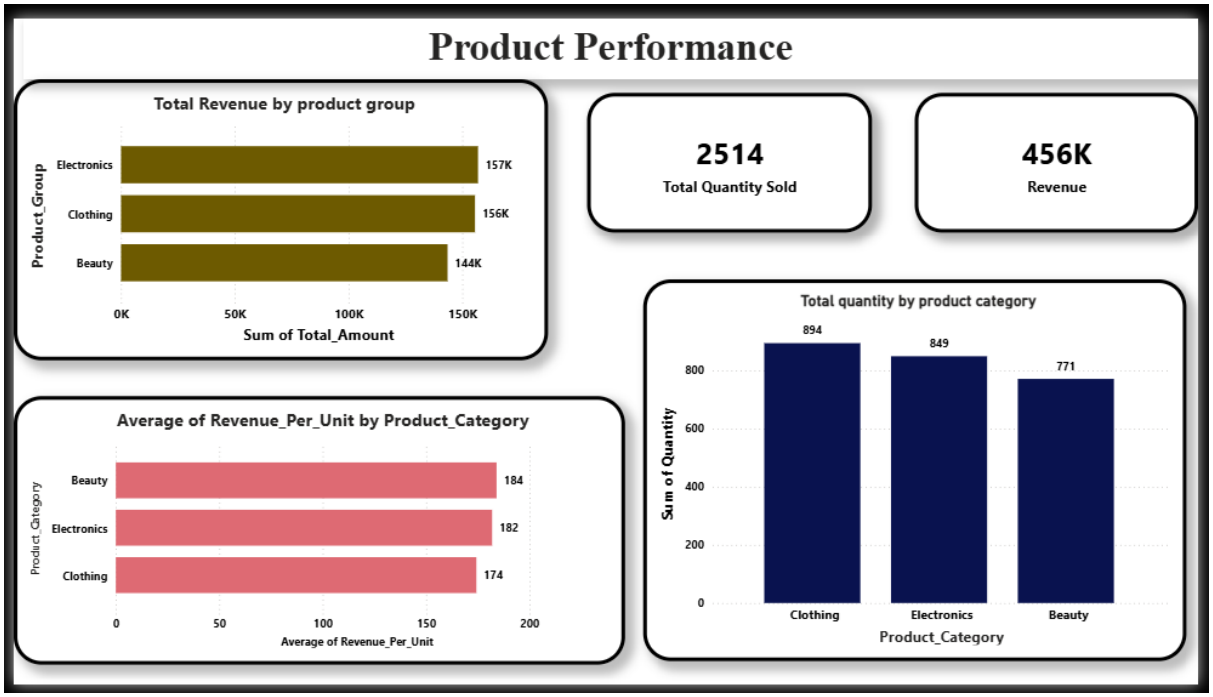
Total Revenue by Gender (Pie Chart)

The revenue split between genders helps identify loyal customer segments. Males and Females contribute almost equally—showing that both segments are important and should be marketed to accordingly.

Conclusion – Page 2

This dashboard provides a detailed understanding of customer demographics and spending patterns. By analyzing gender, age groups, and their purchasing behavior, businesses can develop targeted marketing strategies, optimize product offerings, and better understand the customer segments that drive revenue.

PAGE 3 – Product Performance Dashboard



This page tells the story of how different product categories contribute to both revenue and operational performance.

Total Revenue by Product Group (Bar Chart)

This highlights category-wise revenue distribution, showing which category is the backbone of the business.
Electronics and Clothing lead in revenue, providing the biggest share in overall profit.

Total Quantity Sold (KPI)

Total units sold (2514) help measure product movement and sales efficiency.

Revenue (KPI)

Reinforces total sales from Page 1 for cross-page consistency.

Total Quantity by Product Category (Bar Chart)

This shows which categories sell the highest *number* of items.
Even if a category earns less revenue, a high quantity sold means strong customer engagement.

Average Revenue per Unit by Product Category (Bar Chart)

This chart uncovers **pricing power** or **value perception** of each category.
Beauty products have the highest per-unit profitability.

Business leaders use this to:

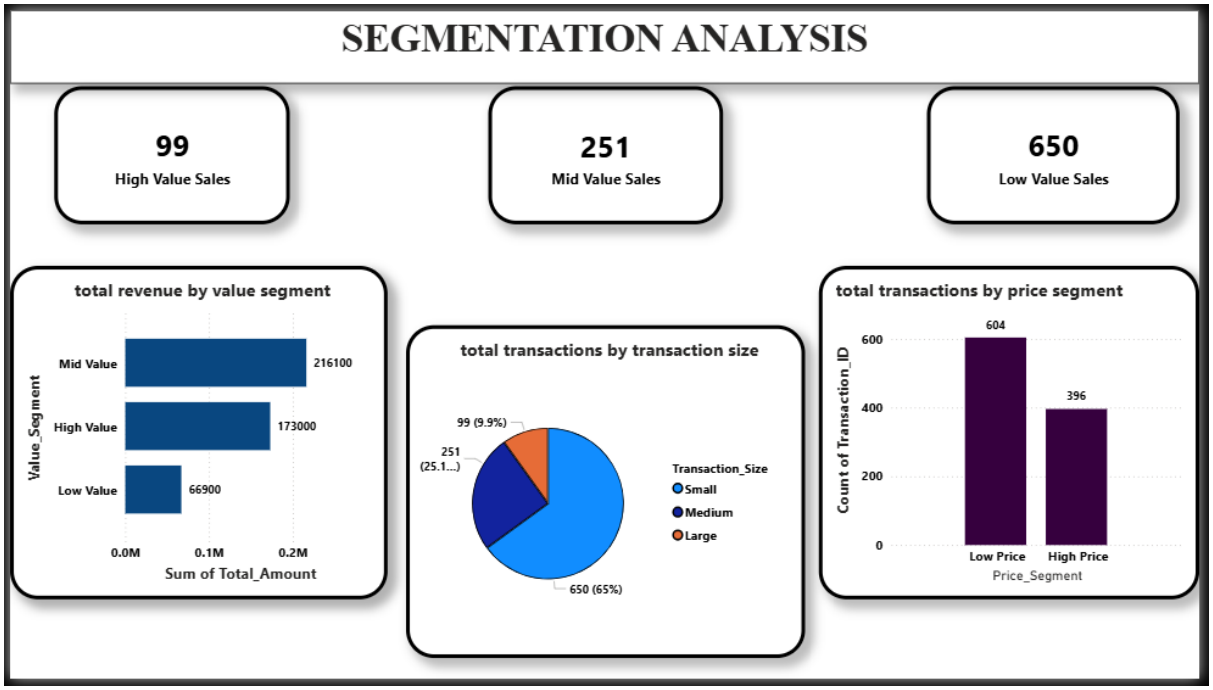
✓ Set pricing strategies

✓ Adjust discounts

✓ Identify premium-performing categories

The Product Performance dashboard highlights revenue strength, quantity movement, and pricing value across categories. These insights support product planning, pricing decisions, and inventory optimization to ensure the right products are prioritized for customer demand and profitability.

PAGE 4 – Segmentation Analysis Dashboard



This page breaks customers and transactions into meaningful segments that help the business understand high-value patterns.

Value Segment KPIs

- **99 High-Value Sales**
- **251 Mid-Value Sales**
- **650 Low-Value Sales**

This categorization helps in customer targeting and personalized offers.

Revenue by Value Segment (Bar Chart)

Mid-Value customers generate the highest revenue, suggesting:

- They purchase more frequently
- They may be the most loyal customer group

High-Value customers contribute strongly despite fewer transactions—ideal for premium campaigns.

Transactions by Transaction Size (Pie Chart)

Small transactions dominate (65%), showing customers prefer buying fewer items per purchase.

Transactions by Price Segment (Bar Chart)

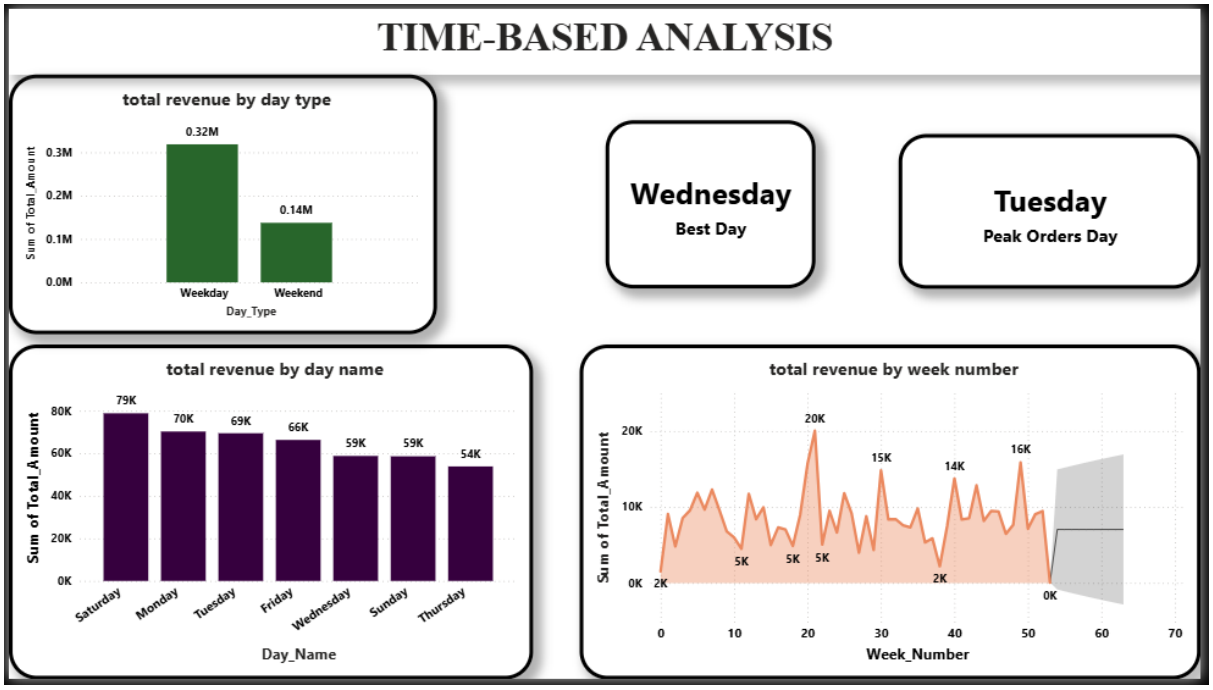
Low-price items sell more frequently, but high-price items generate large revenue chunks.

This is crucial for inventory planning and pricing decisions.

Conclusion – Page 4

This dashboard provides a multi-dimensional understanding of customer spending patterns. By analyzing value segments, transaction sizes, and price categories, businesses can create targeted promotions, adjust pricing strategies, and align product offerings with customer behavior.

PAGE 5 – Time-Based Analysis Dashboard



This page reveals the hidden story behind *when* customers prefer to shop.

Revenue by Day Type (Weekday vs Weekend)

Weekdays outperform weekends massively, proving that:

📌 Customers shop more during weekdays (office lunch breaks or evenings).

📌 Weekend campaigns may need reinforcement to increase sales.

Best Day & Peak Orders Day (Cards)

- **Wednesday** consistently shows the *highest* revenue.
- **Tuesday** has the most *orders*.

This helps businesses plan:

✓ Promotional emails

✓ Flash sales

✓ Inventory restocking

Revenue by Day Name (Bar Chart)

Visual proof of which days outperform others.
Saturday leads in some weeks—likely due to weekend shopping patterns.

Revenue by Week Number (Area Chart with Forecast)

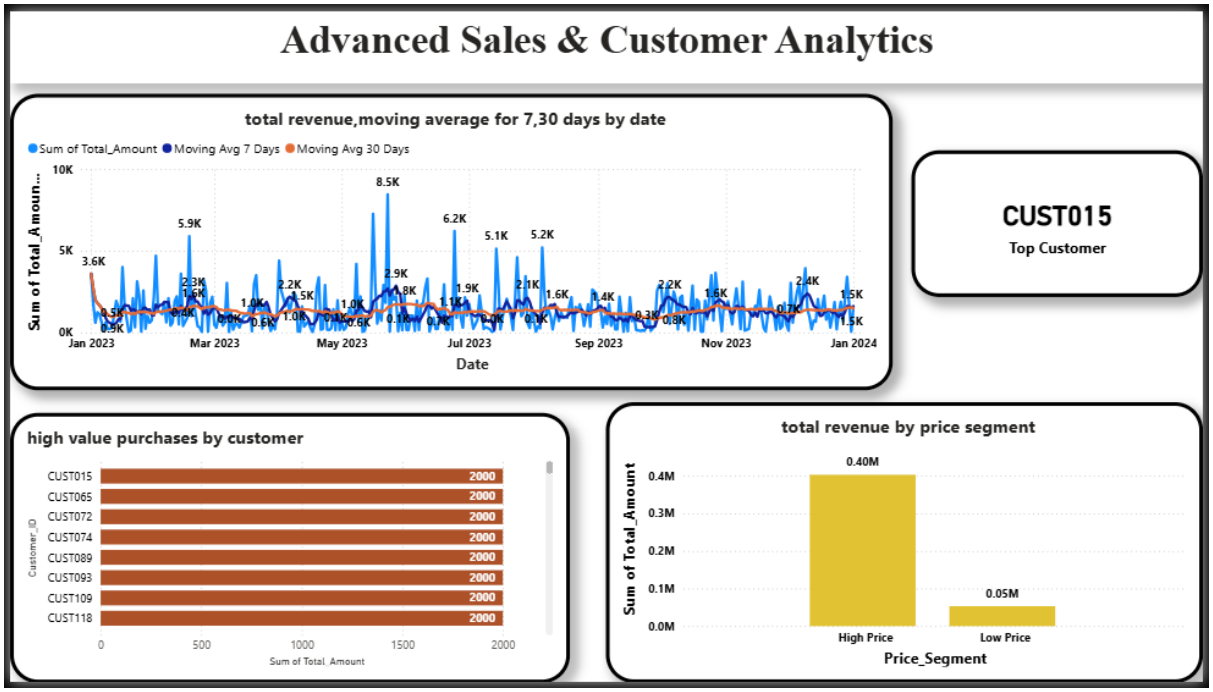
This chart shows weekly fluctuations alongside predicted future revenue.
It helps understand:

- Seasonal cycles
- Weeks with sudden spikes
- Inventory planning
- Cash flow estimation

Conclusion – Page 5

This dashboard identifies patterns in customer shopping behavior across days and weeks. It enables businesses to optimize staffing, schedule promotions, and anticipate demand with greater accuracy.

PAGE 6 – Advanced Sales & Customer Analytics Dashboard



Daily Revenue with Moving Averages (7 & 30 Days)

This is the most analytical chart so far.

- The **blue line** shows actual daily revenue.
- The **orange line** (7-day MA) smooths short-term fluctuations.
- The **red line** (30-day MA) reveals long-term trends.

This chart answers:
✓ Are sales stabilizing or declining?
✓ Are marketing campaigns causing short-term spikes?
✓ Is the business growing in the long run?

Top Customer Highlight (Card)

CUST015 is identified as the top customer—ideal for loyalty programs or personalized offers.

High Value Purchases by Customer

This bar chart reveals the most profitable customers who consistently spend high amounts.

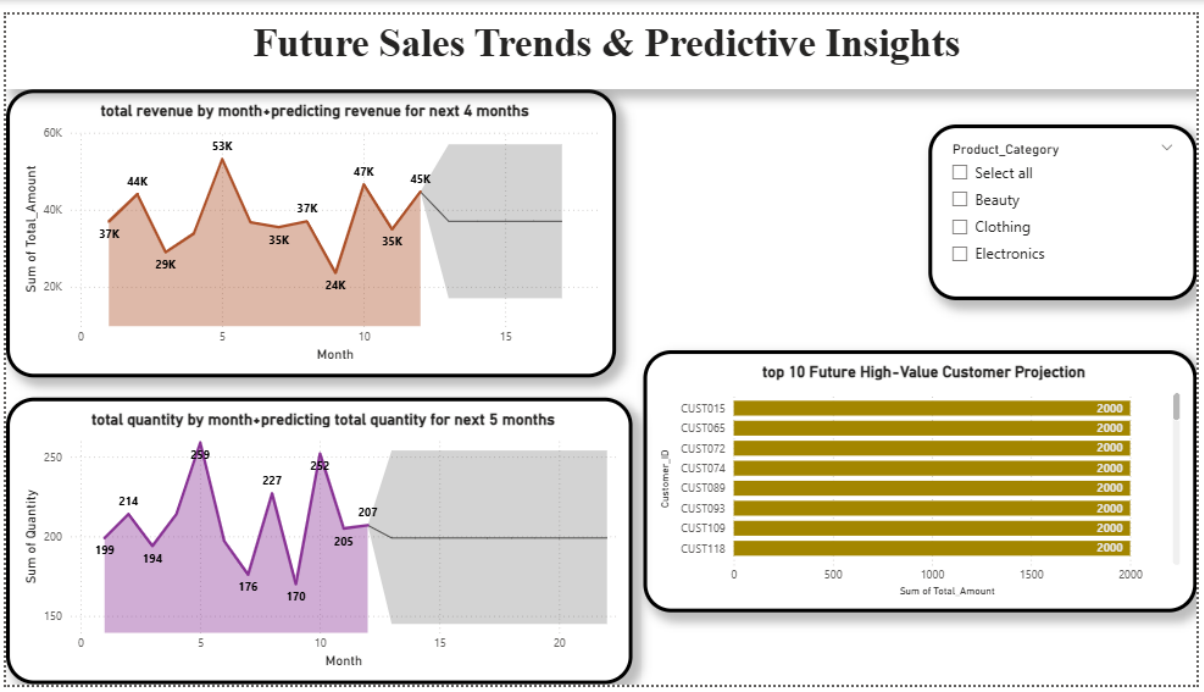
Revenue by Price Segment

Shows how pricing strategies impact total revenue.
High-price items generate maximum revenue despite lower transaction count.

Conclusion – Page 6

The Advanced Analytics dashboard provides forward-looking insights into customer value, revenue trends, and pricing behavior. By combining moving averages, customer value ranking, and price segmentation, this page helps businesses prepare long-term strategies and predictive planning.

PAGE 7 – Future Sales Trends & Predictive Insights



This page focuses on predicting future business outcomes using Power BI's forecasting engine.

Forecasted Revenue for Next 4 Months (Line Area Chart)

The chart combines historical monthly revenue with AI-powered future predictions.

Insights:

📌 Seasonal peaks likely to continue

📌 Revenue expected to stabilize around 45K–50K

📌 Businesses can prepare inventory accordingly

This helps leaders proactively plan:

✓ Budgets

✓ Staffing

✓ Inventory orders

✓ Marketing campaigns

Forecasted Quantity for Next 5 Months

This predictive chart shows how future sales volume may behave.

Key observations:

- Quantity is expected to fluctuate but remain stable.
- Helps in warehouse and supply-chain management.

Top 10 Predicted High-Value Customers

The chart forecasts which customers are likely to remain or become high-value in the future.

Business impact:
✓ Helps identify VIP customers
✓ Enables targeted retention campaigns
✓ Supports personalized product recommendations

ERRORS & SOLUTIONS

1. Power BI Sign-In & Connectivity Issues

Error: Power BI failed to connect to BigQuery due to authentication problems.
Solution:

- Enabled **BigQuery API** in Google Cloud Console
- Updated IAM permissions
- Signed in again using **OAuth Google account authentication**

---

2. Materialized Views Not Supported in Power BI

Error: Power BI DirectQuery cannot load Materialized Views from BigQuery.
Solution:

- Converted Materialized View into a **normal SQL table**
- Replaced the connection string with SELECT * FROM target_table

---

3. BigQuery Date Parsing Errors

Error: PARSE_DATE returned errors due to inconsistent date formats.
Solution:

- Used SAFE.PARSE_DATE to avoid query failure
- Standardized date formats while loading
- Filtered out invalid or future dates

---

4. Null or Missing Key Values

Error: Customer_ID, Quantity, or Total Amount contained null/invalid values.
Solution:

- Added validation checks in staging table
- Replaced nulls with defaults or filtered them out

- Applied IFNULL() or SAFE_CAST() to prevent errors

---

5. Incorrect Segmentation Logic

Error: CASE statements overlapped or categorized data incorrectly.
Solution:

- Rewrote segment logic with clear numeric boundaries
- Verified segment distribution before loading target table

---

6. Window Functions Failing in Staging Table

Error: ROW_NUMBER() failed due to ambiguous or incorrect ORDER BY fields.
Solution:

- Explicitly referenced ordered columns
- Ensured consistent data types before applying window functions

---

7. Power BI Visuals Not Displaying Data

Error: Some charts appeared blank or incomplete.
Solution:

- Corrected data types (e.g., INT instead of STRING)
- Ensured correct aggregation (SUM, COUNT, AVERAGE)
- Adjusted relationships between fields

---

8. Month Sorting Incorrect in Power BI

Error: Month_Name was alphabetically sorted instead of calendar order.
Solution:

- Added numeric Month column in BigQuery
- Set “Sort by Column → Month” in Power BI

---

9. Advanced SQL Query Issues in Power BI

Error: Dataset failed to load due to invalid SQL in Power BI's query box.
Solution:

- Used fully qualified BigQuery table name
- Tested SQL in BigQuery first before inserting into Power BI
- Removed unsupported SQL functions

---

10. Slow Performance with DirectQuery

Error: Dashboard visuals loaded slowly due to heavy queries.
Solution:

- Optimized queries in BigQuery
- Removed unnecessary columns from load
- Used filtered data instead of full dataset when possible