

**ETL Pipeline and KPI Reporting Using BigQuery and DBT**

July 02, 2025

**Table of Contents**

Contents

[1. Introduction 3](#_Toc202374799)

[2. Background and Objective 3](#_Toc202374800)

[3. Methodology 4](#_Toc202374801)

[3.1 Environment Setup: 4](#_Toc202374802)

[3.2 Data Cleaning and Transformation: 4](#_Toc202374803)

[3.3 Design: 5](#_Toc202374804)

[3.4 Query Building and Analysis: 6](#_Toc202374805)

[3.5 Dimension and Fact Tables: 6](#_Toc202374806)

[3.6 Key Performance Indicators: 8](#_Toc202374807)

[4 Recommended Actions: 10](#_Toc202374808)

[5 Conclusion: 11](#_Toc202374809)

# Introduction

This report documents the design and development of an end-to-end ETL pipeline and data warehousing project using the Brazilian E-Commerce Dataset on BigQuery. The project follows the Medallion Architecture (Bronze, Silver, and Gold layers) and applies a Star Schema model to build analytical datasets. The main objective is to transform raw data into structured dimensional models and derive Key Performance Indicators (KPIs) to drive meaningful business decisions.

# Background and Objective

The primary aim of this project is to ingest, clean, transform, and analyze e-commerce data collected from a public Kaggle dataset. The final dataset supports insightful KPI generation and that can be further used for visualization in BI tools.

**Key objectives included:**

* 1. **Medallion Architecture:**

The project implements a three-tiered Medallion Architecture in BigQuery: Bronze for raw ingestion, Silver for cleaned and transformed data, and Gold for analytics-ready dimension, fact tables, and KPI views. This layered structure ensures modularity, scalability, and clear data lineage throughout the pipeline.

* 1. **Clean and Standardize raw Data:**

Raw datasets were cleaned by handling nulls, removing duplicates, and standardizing values such as state names and timestamp formats to ensure consistency and reliability.

* 1. **Dimension and fact tables:**

Dimension tables were created with unique primary keys to describe customers, orders, products, locations, and reviews. A central fact table was built to capture transactional metrics and establish foreign key relationships for efficient analytical querying.

* 1. **Design KPI Queries:**

Key Performance Indicator (KPI) queries were written in BigQuery to track business metrics like total revenue, top categories, and customer distribution. These queries support real-time insights in BI tools and data-driven decision-making.

* 1. **Technology Stack:**

The project utilizes Google BigQuery for cloud data warehousing and DBT for transformation and model management. Python was used for initial data uploads, while SQL powered all transformation and analysis logic.

* 1. **Data Source:**

The dataset consists of 9 CSV files representing customers, orders, payments, reviews, products, sellers, geolocation data, and category translations. These were uploaded and processed into BigQuery.

# Methodology

The development of fact and dimension tables in BigQuery followed a structured ETL process based on the Medallion Architecture, consisting of bronze, silver, and gold layers. Raw CSV files were ingested into the bronze layer, then transformed and cleaned using SQL and DBT models in the silver layer. Finally, Star Schema-based dimension and fact tables, along with KPI views, were created in the gold layer to support scalable analytics and dashboarding.

## Environment Setup:

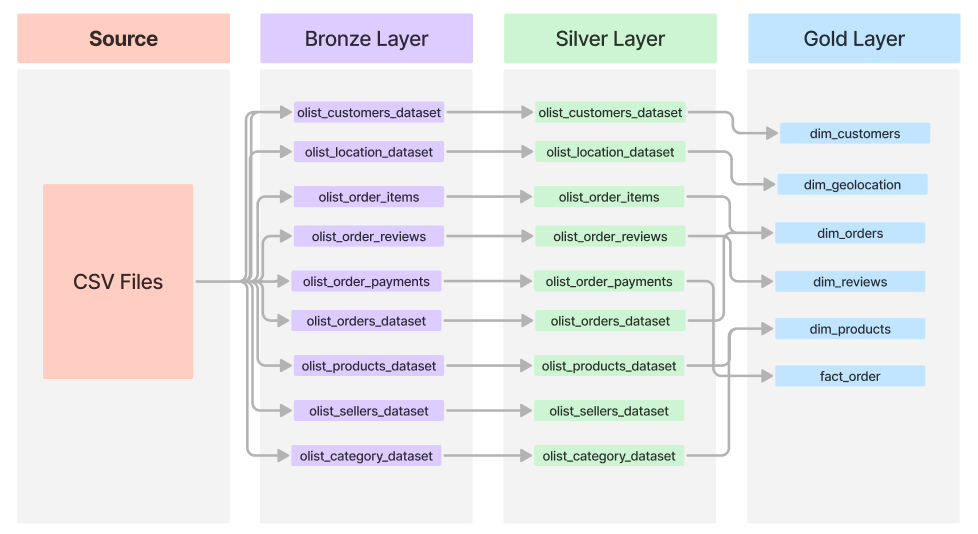
* + 1. **BigQuery and DBT Initialization**: A Google Cloud project was initialized to use BigQuery as the central data warehouse. Datasets for the Bronze, Silver, and Gold layers were structured to support the Medallion Architecture. DBT (Data Build Tool) was configured locally to manage model creation, run SQL transformations, and maintain version-controlled, modular ETL pipelines within the BigQuery environment
    2. **Data Upload & Configuration:** Raw CSV files from the Brazilian E-Commerce dataset were uploaded into the bronzelayer using the BigQuery UI and Python scripts. Schemas were auto-detected or manually defined to ensure correct data types during ingestion.

## Data Cleaning and Transformation:

* + 1. **Missing Value Handling**: Nulls were identified and treated appropriately (e.g., removal, replacement with average of columns).
    2. **Data Type Standardization**: Date fields, numeric fields, and IDs were standardized in proper formatting for consistency of the overall data.
    3. **Primary and Foreign Keys**: Referential integrity and relationships between were identified between dimension and fact tables.
    4. **Duplicate Removal**: Unnecessary duplicate records were identified and removed.

## Design:

* + 1. **Star Schema Implementation:** Data was modeled into a star schema, which is widely used in data warehousing for its simplicity and performance. In this model, a central fact table stores transactional data. This design facilitates easier aggregation, filtering, and slicing of data for analytical purposes.
    2. **Medallion Architecture:** Medallion architecture is followed throughout this project. Data is first loaded in bronze layer. Then after transformation it is pushed in silver layer then finally into gold layer for making dimension tables, fact table and KPI’s.



* + 1. **Logical Key Design and Denormalization Strategy:** Primary and foreign key relationships were logically enforced in the data model design. While BigQuery does not support physical constraints, each dimension table was built with a clearly defined primary key, and these were referenced as foreign keys in the central fact table (fact\_orders). Additionally, denormalization was applied where needed (e.g., enriching dim\_reviews with product category names) to minimize joins at query time, improving performance for KPI views and dashboards.

## Query Building and Analysis:

* + 1. **Creation of layers:** Three layers are used to contain all of the data. Bronze layer was created using python script while other layers i.e. silver and gold layers are created by DBT itself.

DATASET\_ID = "bronze"

credentials = service\_account.Credentials.from\_service\_account\_file(KEY\_FILE)

client = bigquery.Client(credentials=credentials, project=PROJECT\_ID)

dataset\_ref = f"{PROJECT\_ID}.{DATASET\_ID}"

dataset = bigquery.Dataset(dataset\_ref)

dataset.location = "US"

client.create\_dataset(dataset, exists\_ok=True)

print("Layer 'bronze' created.")

Figure 3.4.1 (Script for making Bronze Layer)

* + 1. **Transformation Steps:** The transformation logic focused on cleaning and enriching the data for analytical readiness in the gold layer. This included handling missing values, standardizing categorical fields like state names, and splitting timestamp fields into separate date and time columns. Joins were performed between fact and dimension tables to bring in contextual data such as customer location, product category, and review scores.

## Dimension and Fact Tables:

* + 1. **Dimension Tables:** Five dimension tables were created: dim\_products, dim\_customers, dim\_orders, dim\_reviews, and dim\_geolocation. Each table includes a clearly defined primary key—product\_id in dim\_products, customer\_id in dim\_customers, order\_id in dim\_orders, review\_id in dim\_reviews, and geolocation\_zip\_code\_prefix in dim\_geolocation. These primary keys ensure uniqueness and support reliable joins with the central fact table for efficient analytical queries.
    2. **Queries and Tables for Dimension Tables:** Following are the queries that are used to create the dimension tables which are dim\_products, dim\_customers, dim\_orders, dim\_reviews, and dim\_geolocation in the gold layer so that we can further make fact table from them in next step.

SELECT

  customer\_id,

  customer\_unique\_id,

  customer\_zip\_code\_prefix,

  customer\_city,

  customer\_state

FROM `e-commerce-etl.silver.olist\_customers\_dataset`

WHERE customer\_id IS NOT NULL

  AND customer\_unique\_id IS NOT NULL

  AND customer\_zip\_code\_prefix IS NOT NULL

  AND customer\_city IS NOT NULL

  AND customer\_state IS NOT NULL

Figure 3.5.2 (Creation of dim\_customers table)

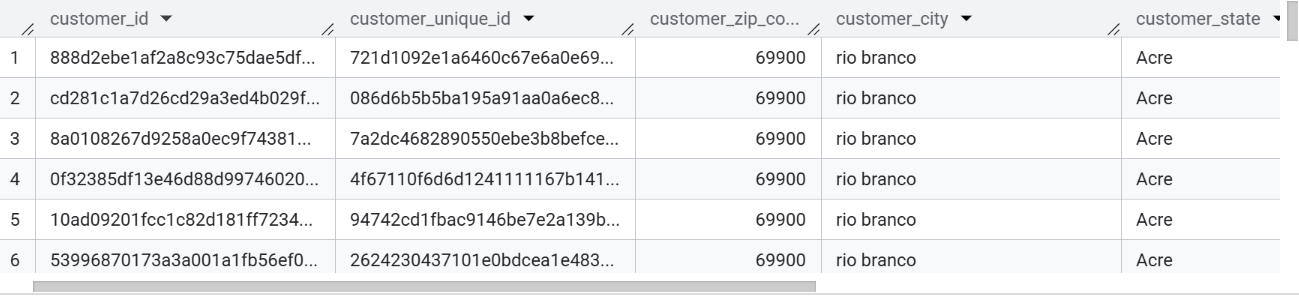


Figure 3.5.2 (Table of dim\_customers)

SELECT

  p.product\_id,

  COALESCE(t.product\_category\_name\_english, p.product\_category\_name) AS product\_category\_name,

  p.product\_weight\_g,

  p.product\_length\_cm,

  p.product\_height\_cm,

  p.product\_width\_cm

FROM `e-commerce-etl.silver.olist\_products\_dataset` p

LEFT JOIN `e-commerce-etl.silver.product\_category\_name\_translation` t

  ON p.product\_category\_name = t.product\_category\_name

WHERE p.product\_id IS NOT NULL

Figure 3.5.2 (Creation of dim\_products table)

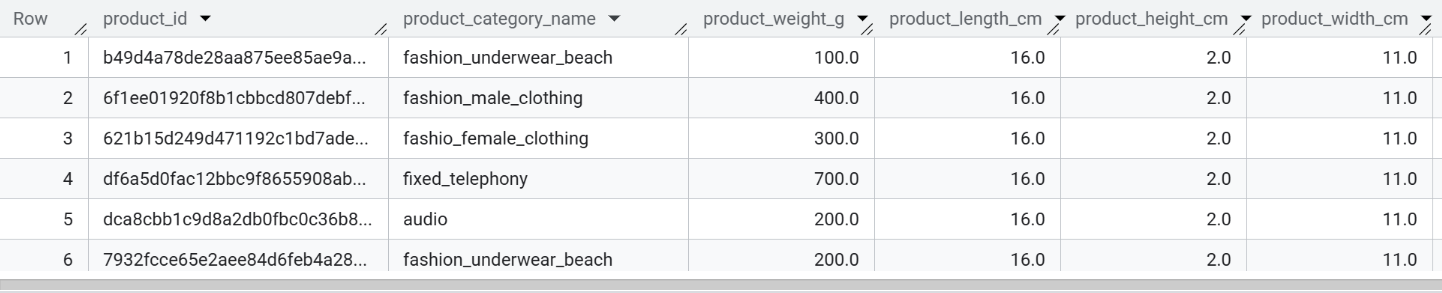


Figure 3.5.2 (Table of dim\_products)

SELECT

  geolocation\_zip\_code\_prefix,

  ANY\_VALUE(geolocation\_city) AS geolocation\_city,

  ANY\_VALUE(geolocation\_state) AS geolocation\_state,

  ROUND(AVG(geolocation\_lat), 6) AS avg\_latitude,

  ROUND(AVG(geolocation\_lng), 6) AS avg\_longitude

FROM `e-commerce-etl.silver.olist\_geolocation\_dataset`

WHERE geolocation\_zip\_code\_prefix IS NOT NULL

  AND geolocation\_city IS NOT NULL

  AND geolocation\_state IS NOT NULL

  AND geolocation\_lat IS NOT NULL

  AND geolocation\_lng IS NOT NULL

GROUP BY geolocation\_zip\_code\_prefix

Figure 3.5.2 (Creation of dim\_geolocation table)

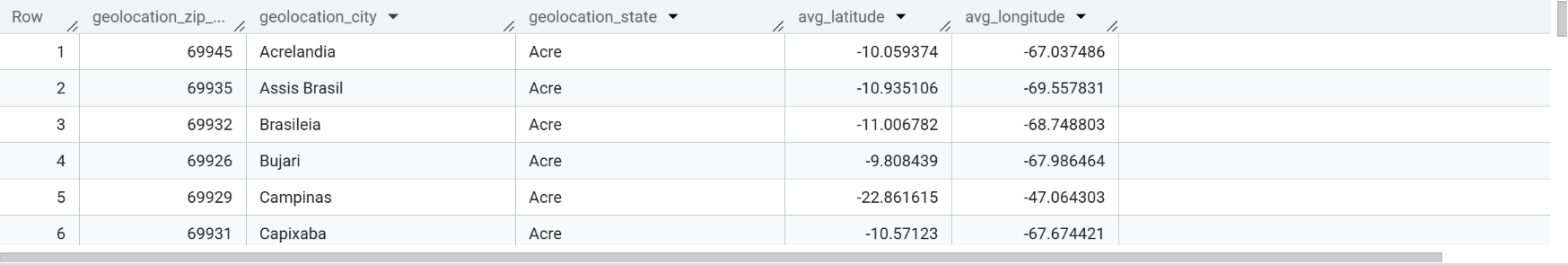


Figure 3.5.2 (Table of dim\_geolocation)

SELECT

  order\_id,

  customer\_id,

  order\_status,

  order\_purchase\_timestamp,

  order\_purchase\_time,

  order\_approved\_at,

  order\_delivered\_carrier\_date,

  order\_delivered\_customer\_date,

  order\_estimated\_delivery\_date

FROM `e-commerce-etl.silver.olist\_orders\_dataset`

WHERE order\_id IS NOT NULL SELECT

  r.review\_id,

  r.order\_id,

   -- Use English category name if available

  COALESCE(t.product\_category\_name\_english, p.product\_category\_name) AS product\_category\_name,

  r.review\_score

FROM `e-commerce-etl.silver.olist\_order\_reviews\_dataset` r

JOIN `e-commerce-etl.silver.olist\_order\_items\_dataset` i

  ON r.order\_id = i.order\_id

JOIN `e-commerce-etl.silver.olist\_products\_dataset` p

  ON i.product\_id = p.product\_id

LEFT JOIN `e-commerce-etl.silver.product\_category\_name\_translation` t

  ON p.product\_category\_name = t.product\_category\_name

WHERE r.review\_id IS NOT NULL

  AND r.order\_id IS NOT NULL

  AND r.review\_score IS NOT NULL

Figure 3.5.2 (Creation of dim\_order table)

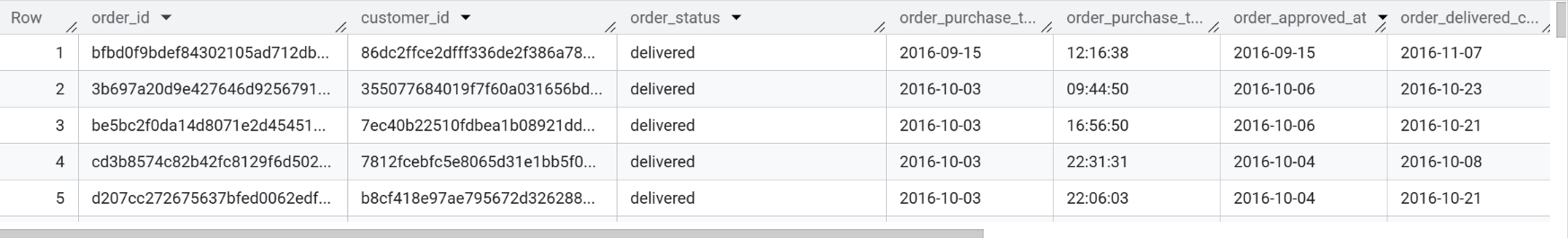


Figure 3.5.2 (Table of dim\_orders)

SELECT

  r.review\_id,

  r.order\_id,

   -- Use English category name if available

  COALESCE(t.product\_category\_name\_english, p.product\_category\_name) AS product\_category\_name,

  r.review\_score

FROM `e-commerce-etl.silver.olist\_order\_reviews\_dataset` r

JOIN `e-commerce-etl.silver.olist\_order\_items\_dataset` i

  ON r.order\_id = i.order\_id

JOIN `e-commerce-etl.silver.olist\_products\_dataset` p

  ON i.product\_id = p.product\_id

LEFT JOIN `e-commerce-etl.silver.product\_category\_name\_translation` t

  ON p.product\_category\_name = t.product\_category\_name

WHERE r.review\_id IS NOT NULL

  AND r.order\_id IS NOT NULL

  AND r.review\_score IS NOT NULL

Figure 3.5.2 (Creation of dim\_reviews table)

Figure 3.5.2 (Creation of dim\_reviews)

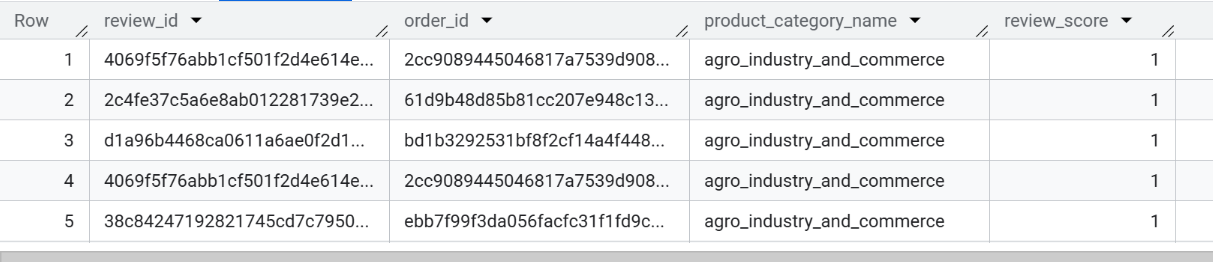


Figure 3.5.2 (Table of dim\_reviews)

* + 1. **Fact Table:** A table named as fact\_orders was created with five different primary keys acting as foreign keys referenced from dimension tables.
    2. **Queries and Tables for Fact Tables:** Each dimension table was created using DBT (Data build Tool) statements models to directly in BigQuery. These queries involved selecting relevant columns from the silver layer, applying transformations such as standardization, filtering out nulls, and performing joins where necessary (e.g., product category translations). The resulting tables form the foundational structure of the gold layer’s star schema.

SELECT

  o.order\_id,

  o.customer\_id,

  c.customer\_unique\_id,

  c.customer\_zip\_code\_prefix AS geolocation\_zip\_code\_prefix,

  r.review\_id,

  i.product\_id,

  COALESCE(t.product\_category\_name\_english, p.product\_category\_name) AS product\_category\_name,

  r.review\_score,

  i.price,

  i.freight\_value,

  pmt.total\_payment\_value

FROM `e-commerce-etl.silver.olist\_orders\_dataset` o

JOIN `e-commerce-etl.silver.olist\_customers\_dataset` c

  ON o.customer\_id = c.customer\_id

JOIN `e-commerce-etl.silver.olist\_order\_items\_dataset` i

  ON o.order\_id = i.order\_id

JOIN `e-commerce-etl.silver.olist\_products\_dataset` p

  ON i.product\_id = p.product\_id

LEFT JOIN `e-commerce-etl.silver.product\_category\_name\_translation` t

  ON p.product\_category\_name = t.product\_category\_name

LEFT JOIN (

  SELECT

    order\_id,

    SUM(payment\_value) AS total\_payment\_value

  FROM `e-commerce-etl.silver.olist\_order\_payments\_dataset`

  GROUP BY order\_id

) pmt ON o.order\_id = pmt.order\_id

-- Join reviews

LEFT JOIN `e-commerce-etl.silver.olist\_order\_reviews\_dataset` r

  ON o.order\_id = r.order\_id

WHERE

  o.order\_id IS NOT NULL

  AND o.customer\_id IS NOT NULL

  AND i.product\_id IS NOT NULL

Figure 3.5.4 (Creation of fact\_orders table)



Figure 3.5.4 (Table of fact\_sales)

## Key Performance Indicators:

* + 1. **Orders By State:** This KPI is created as a view that calculates the total number of orders placed from each Brazilian state by aggregating transactional data based on the customer's location. It helps identify states with high purchase activity, enabling strategies such as targeted marketing, regional inventory planning, and performance benchmarking across geographical areas.

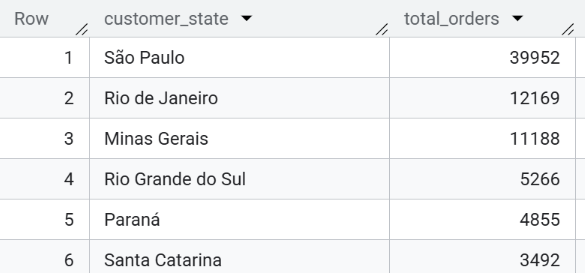
****

Figure 3.6.1 (KPI of Orders by State)

* + 1. **Revenue for each Category:** This KPI calculates total revenue by product category, helping identify the highest-earning categories for better product and marketing focus.

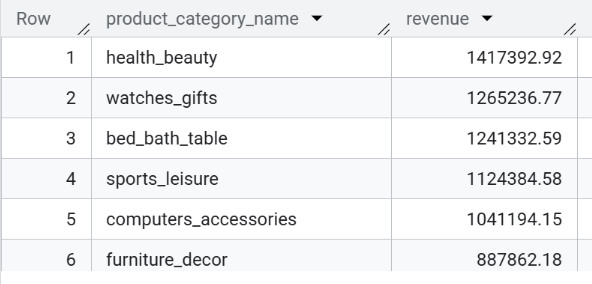


Figure 3.6.2 (KPI of Revenue for each Category)

* + 1. **Review for each Category:** This KPI lists down all of the categories against average ratings.

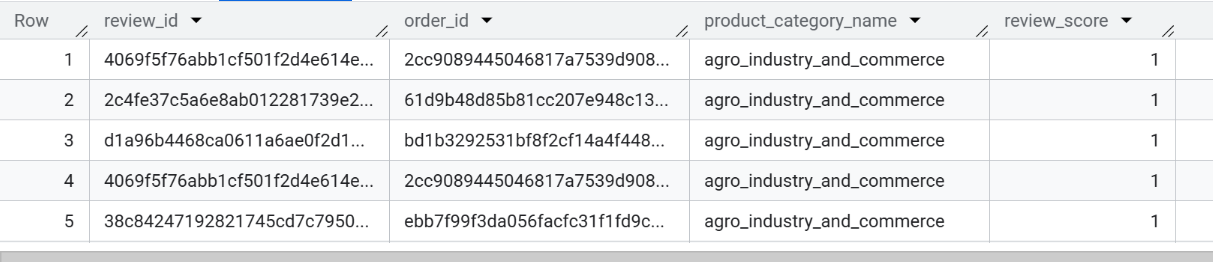
****

Figure 3.6.3 (KPI of Review for each Category)

* + 1. **Total Revenue:** This KPI measures the total revenue generated from all of the products collectively to be shown in the dashboard.

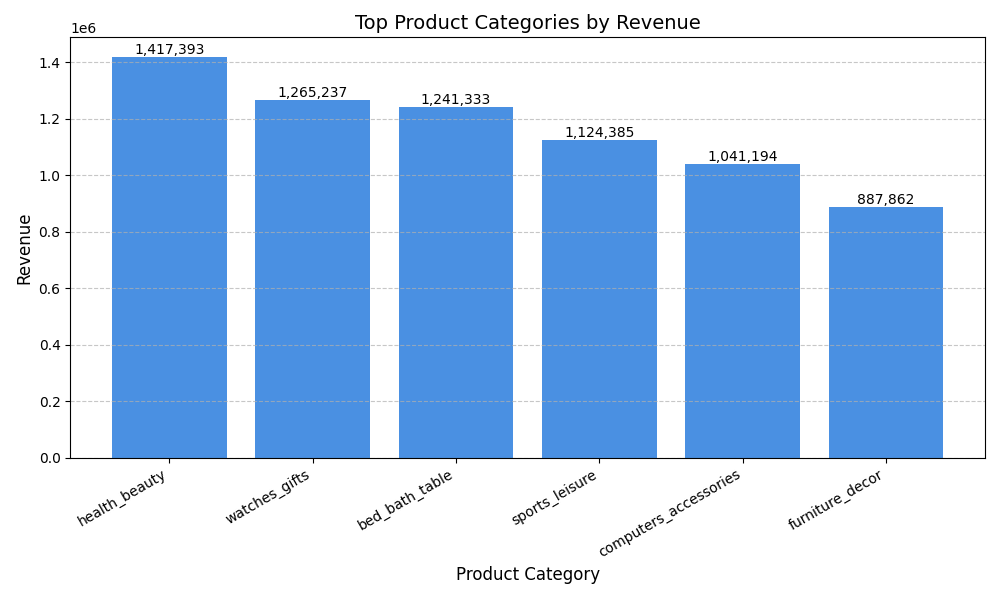
****

Figure 3.6.4 (KPI of Total Revenue)

# Recommended Actions:

**4.1** **Expand Best-Selling Categories:**

Based on the KPI tracking revenue by product category, the business can identify high-performing categories such as "health\_beauty" or "bed\_bath\_table." Expanding product offerings and promotions in these categories can help boost overall revenue and strengthen market share in areas with proven customer demand.



**4.2 Location-Based Targeting:**

Using the “Orders by State” KPI, the business can pinpoint underperforming states with lower order volumes. Targeted campaigns, region-specific promotions, or improved delivery logistics in these states can help drive engagement and increase order rates across underserved locations.

* 1. **Customer Experience Monitoring**:

By analyzing average review scores per product category, the business can monitor customer satisfaction and identify underperforming segments. Consistently low-rated categories may signal issues with product quality or service, enabling timely improvements to enhance overall customer experience and retention.

# Conclusion:

This ETL pipeline project successfully ingested, cleaned, and transformed Brazilian e-commerce data using BigQuery. The adoption of Medallion Architecture and Star Schema enabled a scalable and query-efficient model. KPIs were created as views and are ready to power dashboards and data apps. This system provides a strong foundation for ongoing analytics, forecasting, and strategic decision-making.