

DATABRICKS PROJECT REPORT

Project Title-

End-to-End ETL Pipeline Implementation and Data Analytics in Databricks for Retail Sales Insights.

Prepared by-

Shivam Kumar Jha

Date-

Table of Contents

1. **Introduction**
 - 1.1 Project Overview
 - 1.2 Dataset Description
2. **Setup and Initial Data Load**
 - 2.1 Task Overview
 - 2.2 Methodology
3. **Bronze Layer - Raw Data Ingestion**
 - 3.1 Task Overview
 - 3.2 Methodology
4. **Data Quality Checks**
 - 4.1 Task Overview
 - 4.2 Methodology
5. **Silver Layer - Data Transformation and Enrichment**
 - 5.1 Task Overview
 - 5.2 Methodology
6. **Gold Layer - Aggregation, Filtering, and Reporting**
 - 6.1 Task Overview
 - 6.2 Methodology
7. **Data Validation and Auditing**
 - 7.1 Task Overview
 - 7.2 Methodology
8. **Final Output, Reporting, and Dashboarding**
 - 8.1 Task Overview
 - 8.2 Methodology
 - 8.3 Dashboard
9. **Conclusion**
 - 9.1 Project Summary
 - 9.2 Future Work

1. Introduction

1.1 Project Overview

This project involved the implementation of a full ETL (Extract, Transform, Load) pipeline using Databricks. The objective was to upload and process a retail sales dataset, including orders, customers, order items and products, perform data ingestion of these CSV files, clean the data, apply transformations, and generate meaningful insights through data aggregation and reporting.

1.2 Dataset Description

Four datasets were used in this project:

- **Orders Dataset (orders.csv):** Contains details about customer orders, including order ID, order date, customer ID, order status, etc.
- **Customers Dataset (customers.csv):** Provides information about customers, including customer ID, customer name, and location details.
- **Order Items Dataset (order_items.csv):** Lists the individual items within each order, including order item ID, order ID, product ID, quantity, and price.
- **Products Dataset(products.csv):** Lists the name, and details of all the products like product id, product name and category.

Table ▾ +				
	Δ^B_C path	Δ^B_C name	1^2_3 size	1^2_3 modificationTime
1	dbfs:/FileStore/tables/customers.csv	customers.csv	36668	1723643535000
2	dbfs:/FileStore/tables/order_items.csv	order_items.csv	36421	1723643535000
3	dbfs:/FileStore/tables/orders.csv	orders.csv	12905	1723643534000
4	dbfs:/FileStore/tables/product.csv	product.csv	404	1723643534000

2. Setup and Initial Data Load

2.1 Task Overview

- **Objective:** Upload the dataset into Databricks and load it into a DataFrame.
- **Steps:**
 1. Logged into Databricks Community Edition.

2. Created a new notebook.
3. Uploaded the orders.csv, customers.csv, order_items.csv and product.csv datasets to Databricks using the DBFS(Databricks File System).
4. Loaded each dataset into separate DataFrames, ensuring proper parsing of data.

2.2 Methodology

- **DBFS:** The Databricks File System (DBFS) is a distributed file system that allows you to store and access files in Databricks. It abstracts storage, making it easy to manage data across different clusters and notebooks.

	^B _C path	^B _C name	¹ ₃ size	¹ ₃ modificationTime
1	dbfs:/FileStore/bronze/	bronze/	0	0
2	dbfs:/FileStore/gold/	gold/	0	0
3	dbfs:/FileStore/raw/	raw/	0	0
4	dbfs:/FileStore/silver/	silver/	0	0
5	dbfs:/FileStore/tables/	tables/	0	0

- **DataFrame Loading:** Used the spark.read.csv method to load the data into DataFrames. Ensured that all columns were correctly parsed, and appropriate data types were assigned.

3. Bronze Layer - Raw Data Ingestion

3.1 Task Overview

- **Objective:** Ingest the raw data into the Bronze layer, handling schema evolution and semi-structured data.
- **Steps:**
 1. Loaded the dataframes from raw data folder using spark.read.csv.
 2. Defined the structure and schema of all the columns of dataset
 3. Checked for any null or duplicated values.
 4. Added the current timestamp column.
 5. Saved the DataFrames as Delta tables in the Bronze layer.
 6. Implemented schema enforcement using Delta Lake to ensure data integrity during ingestion.

3.2 Methodology

- **Schema Enforcement:** Ensured that the schema was strictly enforced during the ingestion process. This step involved verifying column data types and ensuring no critical data was missing.

```
from pyspark.sql.types import IntegerType, BooleanType, StringType, DoubleType, StructField, StructType, DateType
schema_fields=StructType(fields=[StructField('id', IntegerType(), False),
                                   StructField('customername', StringType(), True),
                                   StructField('state', StringType(), True),
                                   StructField('city', StringType(), True),
                                   StructField('created_on', DateType(), True),
                                   StructField('date_of_birth', DateType(), True),
                                   StructField('updated_on', DateType(), True),
                                   StructField('email', StringType(), True),
                                   StructField('gender', StringType(), True)
                                ])

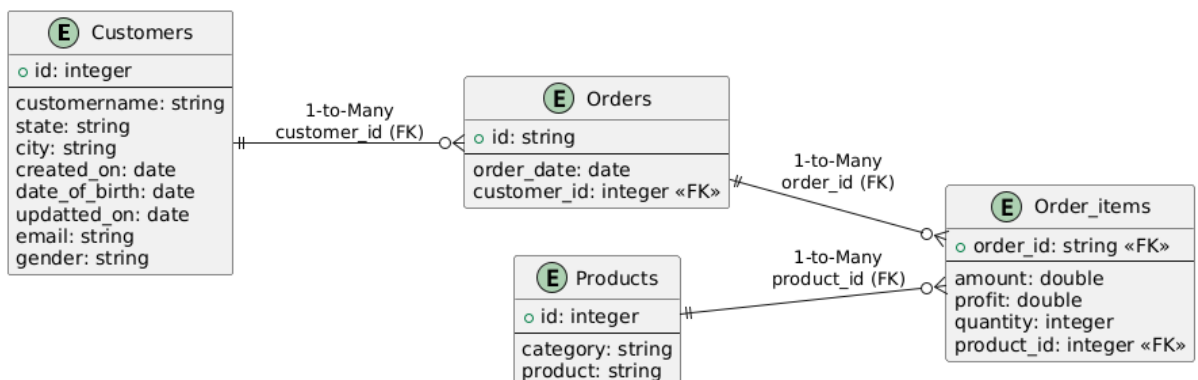
```

- **Data Lineage:** Added Created at column using current_timestamp().
- **Writing in bronze layer as delta tables:** Used spar.write.format('delta') to write the well defined datasets into the bronze directory.

```
customers_df.write.mode("overwrite").format("delta").option("overwriteSchema", "true").save(f'{bronze_path}/customers')

```


- **Bronze Table Structure:** Saved the data in the following paths:



○ /bronze/orders

	A ^B C id	A ^B C customername	A ^B C state	A ^B C city	A ^B C created_on	A ^B C date_of bi
1	267	Mala Pratap	Madhya Pradesh	Indore	2018-12-06	1983-11-04
2	59	Anudeep	Madhya Pradesh	Indore	2018-08-26	1978-09-09
3	273	Shakshi Sagar	Nagaland	Kohima	2018-04-17	1996-11-06
4	116	Ekta Chauhan	Madhya Pradesh	Indore	2018-06-28	1987-04-20
5	92	Bhutekar	Madhya Pradesh	Indore	2019-01-04	1989-10-08
6	48	Anjali Juneja	Delhi	Delhi	2019-02-01	1987-11-24
7	49	Anjali Juneja	Haryana	Chandigarh	2018-05-23	1979-05-28
8	266	Mala Bagga	Himachal Prade...	Simla	2018-11-03	1999-05-07

○ /bronze/customers

	A_C^B id	 order_date	1_3^2 customer_id
1	B-25709	2018-07-01	1
2	B-26081	2019-03-22	2
3	B-26018	2019-02-14	2
4	B-25608	2018-04-08	2
5	B-25893	2018-12-04	3

○ /bronze/prod

	1_3^2 id	A_C^B category	A_C^B product
1	1	Clothing	Hankerchief
2	2	Clothing	Kurti
3	3	Clothing	Leggings
4	4	Clothing	Saree
5	5	Clothing	Shirt
6	6	Clothing	Skirt
7	7	Clothing	Stole

○ /bronze/order_item

	A_C^B order_id	1.2 amount	1.2 profit	1_3^2 quantity	1_3^2 product_id
1	B-26010	18	2	3	1
2	B-25667	11	-2	4	1
3	B-26016	202	4	4	1
4	B-26018	61	8	4	1
5	B-26021	21	-12	3	1
6	B-26025	41	19	5	1

4. Data Quality Checks

4.1 Task Overview

- **Objective:** Implement data quality checks to identify and log issues before moving data to the Silver layer.
- **Steps:**
 1. Checked for null values, duplicates, and data type mismatches in each dataset.
 2. Logged any detected issues into a separate Delta table for auditing.

4.2 Methodology

- **Data Quality Checks:** Utilized PySpark functions such as `dropna()`, `dropDuplicates()`, and `cast()` to identify and handle potential issues in the data.

```
customers_df_cleaned=customers_df_cleaned.dropDuplicates()  
customers_df_cleaned: pyspark.sql.dataframe.DataFrame = [id: integer, customername: string ... 8
```

- **Null data replaced:** In the customers data, replaced the null values in the gender column with 'Unknown' and the email field null values is replaced with <username><id>@gmail.com

```
from pyspark.sql.functions import col,when,concat_ws,regexp_replace  
  
customers_df_cleaned = customers_df.withColumn(  
    "cleaned_name",  
    regexp_replace(col("customername"), " ", "") # Remove all spaces  
)  
customers_df_cleaned.withColumn(  
    "email",  
    when(col("email").isNull(), concat_ws("@", col("cleaned_name"),col('id'), lit("gmail.com"))).otherwise(col("email"))  
)  
customers_df_cleaned.drop("cleaned_name")
```

- **Auditing:** Created an audit Delta table to store records of data quality issues for transparency and tracking.

5. Silver Layer - Data Transformation and Enrichment

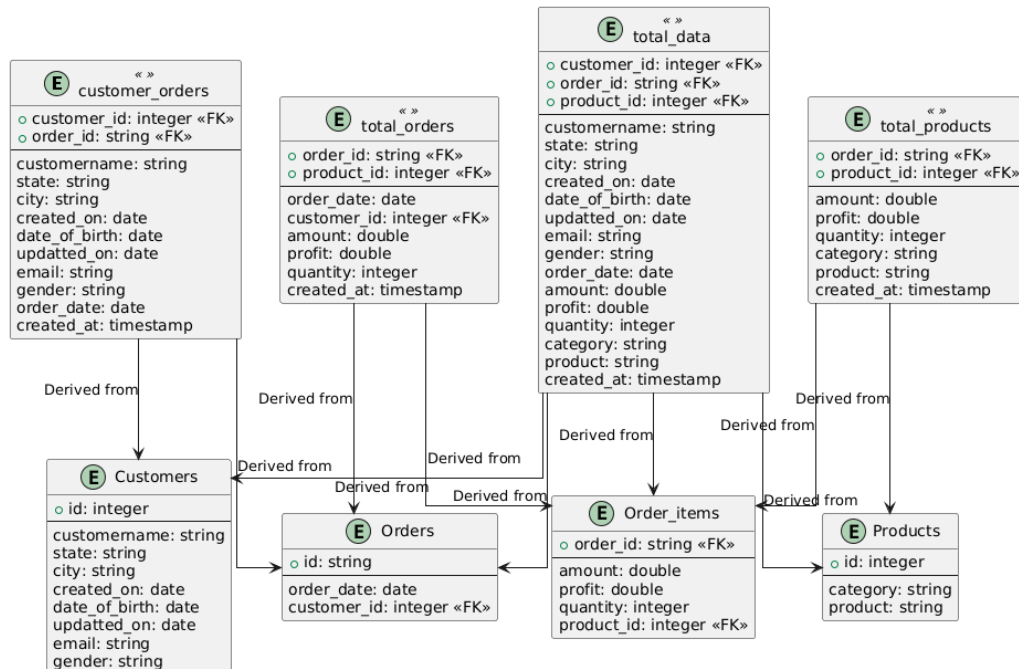
5.1 Task Overview

- **Objective:** Perform complex transformations, enrich the data and perform logical joins.
- **Steps:**
 1. Read data from the Bronze tables.

2. Cleaned the data and joined tables to create a comprehensive dataset, including customer order history.
3. Join the tables logically.
4. Saved the transformed data as Delta tables in the Silver layer.

5.2 Methodology

- **Data Cleaning:** Applied techniques such as handling missing values, correcting data formats, and standardizing categorical data.
- **Understand the relation:** Understood the relation between different tables and on which common columns they could be joined.
- **Table Joins:** Performed joins between orders, customers, products and order_items tables to create a unified view of customer orders.
- **Silver Table Structure:** Saved the data in the following paths:
 - /silver/customer_orders
 - /silver/total_orders
 - /silver/total_products
 - /silver/total_data



6. Gold Layer - Aggregation, Filtering, and Reporting

6.1 Task Overview

- **Objective:** Aggregate, filter, and prepare data for reporting and analytics.
- **Steps:**
 1. Read data from the Silver tables.
 2. Performed specific aggregations and calculations, such as total profit by product category, and total orders by state.
 3. Saved the result datasets as Delta tables optimized for reporting.

6.2 Methodology

- **Aggregation and Calculations:**
 - **Total Profit by Product Category:** Used grouping and aggregation functions to calculate total profits.
 - **Top 5 Customers by Spend:** Identified the top-spending customers using ordering and limiting functions.
- **Gold Table Structure:** Saved the aggregated data in the following paths:
 - /gold/amount_spent_per_city
 - /gold/avg_profit_per_category_
 - gold/avg_profit_per_city
 - /gold/customer_amount--
 - gold/customer_top5
 - gold/product_orders
 - gold/product_profit
 - gold/product_revenue
 - gold/state_orders
 - gold/total_orders_per_category

7. Data Validation and Auditing

7.1 Task Overview

- **Objective:** Implement validation and auditing mechanisms to ensure data integrity across the ETL pipeline.
- **Steps:**
 1. Implemented validation checks at each layer (Bronze, Silver, Gold).
 2. Created timestamp at each layer to implement data lineage.

3. Created audit logs to track data lineage and changes.
4. Generated reports on data quality and transformation steps.

7.2 Methodology

- **Validation Checks:** Ensured data consistency by validating data types, row counts, and key constraints across layers.
- **Audit Logs:** Used Delta Lake's built-in features and `current_timestamp` to track changes and data lineage.
- **Reporting:** Generated summary reports on the quality of the data, the transformations applied, and any anomalies detected.

8. Final Output, Reporting, and Dashboarding

8.1 Task Overview

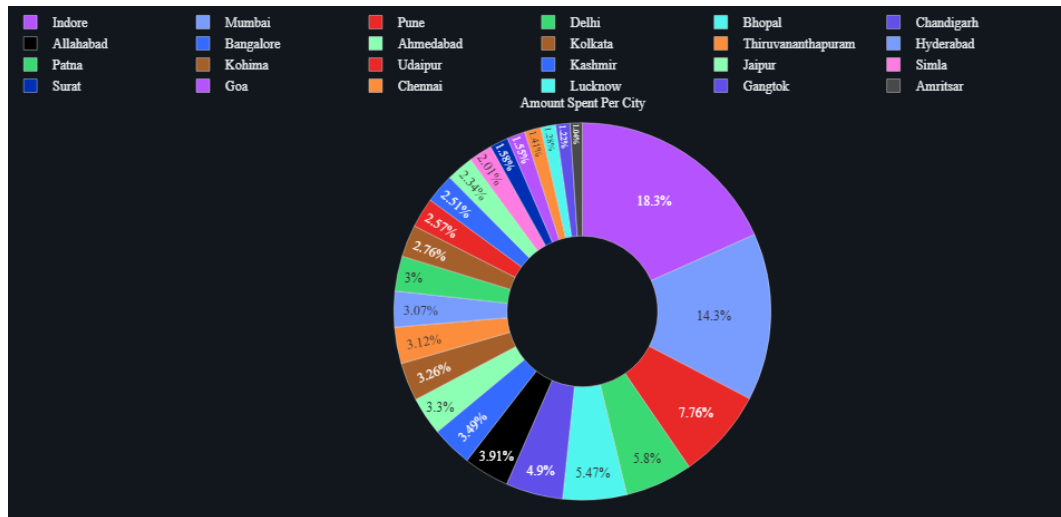
- **Objective:** Generate insights and visualizations using the Gold layer data.
- **Steps:**
 1. Created visualizations using Databricks' built-in visualization tools.
 2. Built a dashboard to display key metrics, such as total revenue and top customers.
 3. Prepared a final report documenting the ETL pipeline, data transformations, and insights.

8.2 Methodology

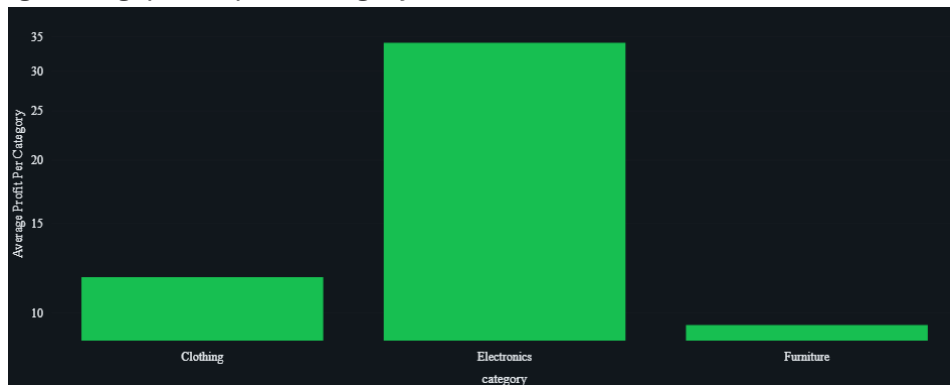
- **Visualization Tools:** Utilized Databricks' built-in visualization tools to create insightful charts and graphs.
- **Dashboard:** Displayed key business metrics such as total sales, average order value, and customer lifetime value.

8.3 Dashboard-

- /gold/amount_spent_per_city



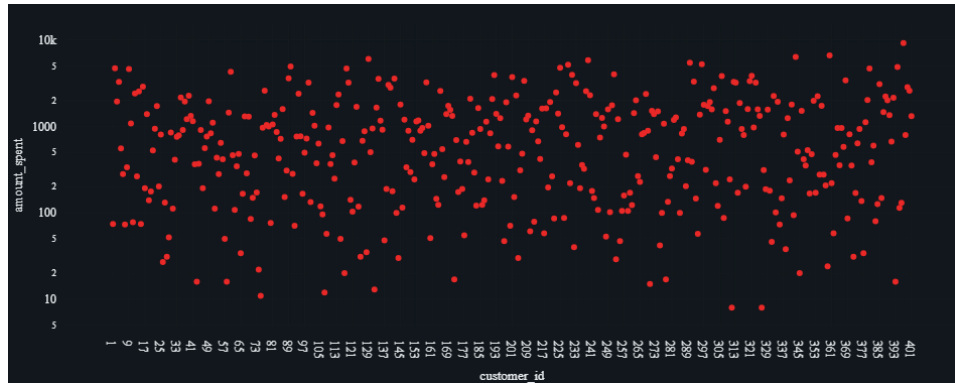
- /gold/avg_profit_per_category_



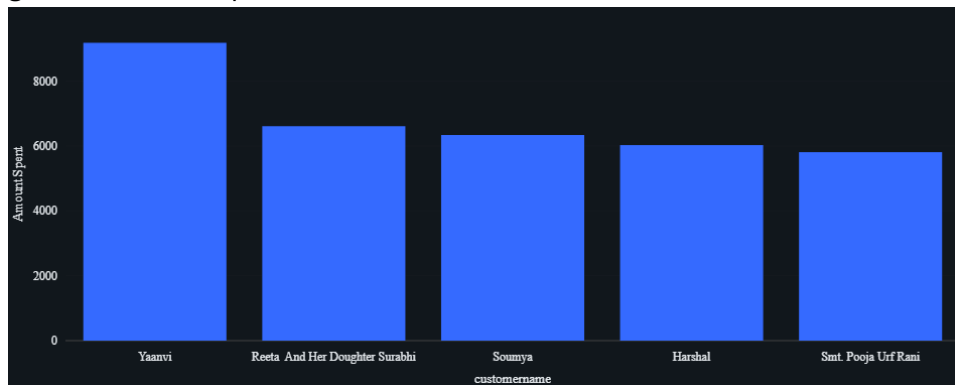
- gold/avg_profit_per_city



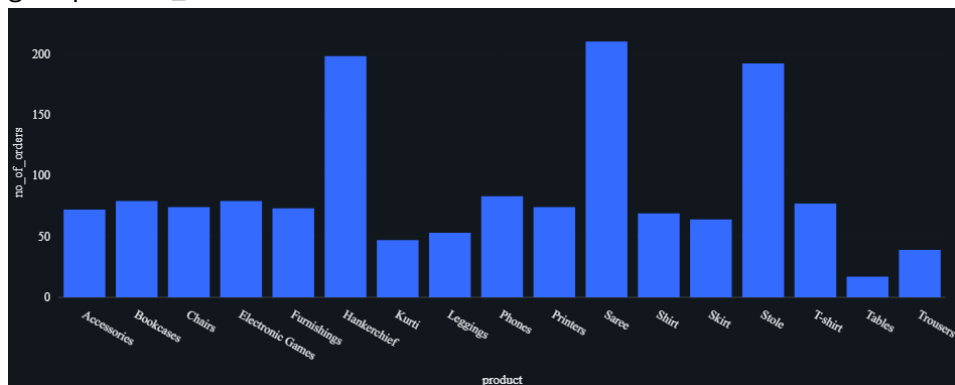
- /gold/customer_amount--



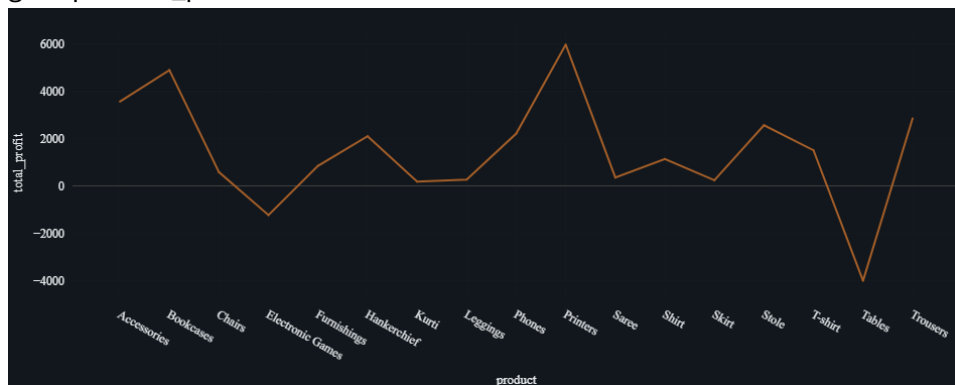
- gold/customer_top5



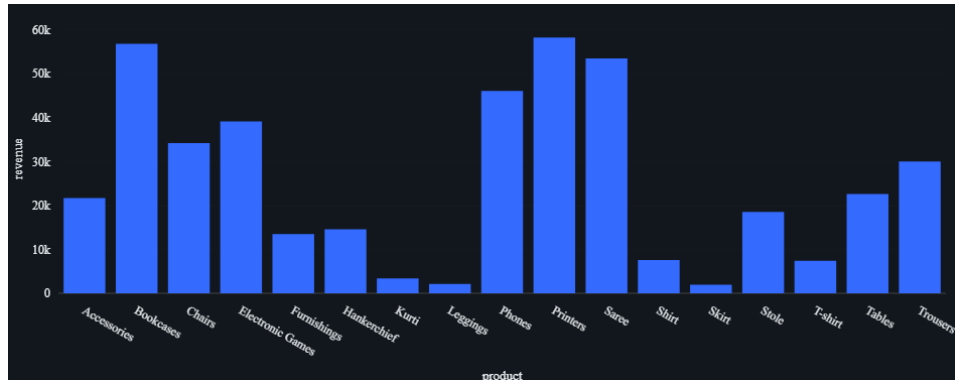
- gold/product_orders



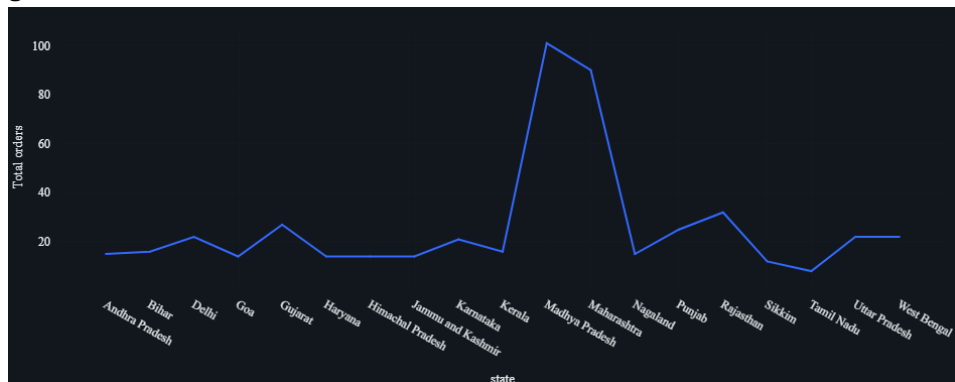
- gold/product_profit



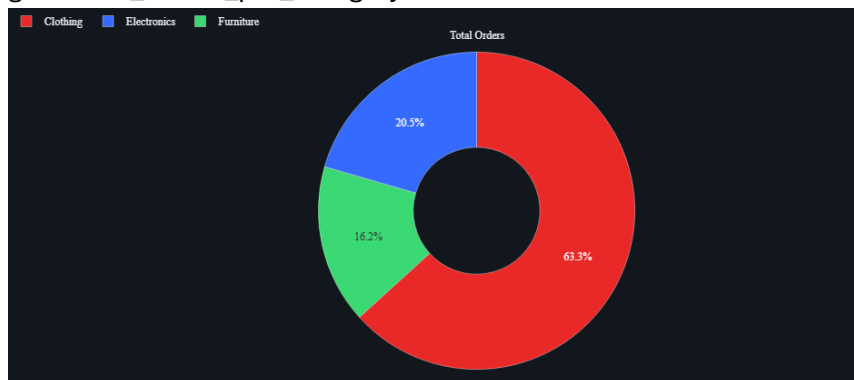
- gold/product_revenue



- gold/state_orders



- gold/total_orders_per_category



9. Conclusions

9.1 Project Summary

This project successfully implemented an end-to-end ETL pipeline in Databricks, from raw data ingestion to the creation of a comprehensive reporting layer. The pipeline maintained data integrity through rigorous quality checks and provided valuable business insights through data aggregation and reporting.

9.2 Future Work

Potential improvements could include:

- Automating the pipeline using azure data factory tools for regular data updates.
 - Creating pipeline for the streaming data to regularly updating the database.
 - Using of Unity catalog to store data more securely .
 - Integrating additional data sources for richer analytics.
 - Implementing machine learning models for predictive insights based on the Gold layer data.
-