**Rajalakshmi Engineering College**

**AI&DS Department**

**Big Data / Data Science Project**

**Semester: V**

**Duration: 4 Weeks**

**Team Number: B5**

**Domain: Government and Public Policy**

**Scenario Title: Citizen feedback policy sentiment.**

# Team Members

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Roll Number | Email | Contribution Summary |
| Sabarish P | 231801142 | 231801142@rajalakshmi.edu.in |  |
| Sabharishraja B | 231801143 | 231801143@rajalakshmi.edu.in |  |
| Darshan R | 231801133 | 231801133@rajalakshmi.edu.in |  |

# 1. Project Overview

## 1.1 Problem Context

In modern governance, collecting and understanding citizen feedback is essential for improving public services and policies. Citizens express their opinions through government portals, surveys, and online forms, resulting in massive amounts of unstructured text data.  
Manual analysis of this feedback is inefficient and prone to bias. To address this, **Big Data technologies** like **Apache Spark** and **Databricks** enable scalable processing and real-time analysis of feedback data.  
This project, **“Citizen Feedback Policy Sentiment Analysis,”** leverages **PySpark** and **TextBlob** to automatically classify citizen comments into *Positive*, *Negative*, or *Neutral* sentiments. The insights generated help policymakers gauge public perception, identify areas of improvement, and enhance transparency in governance

## 1.2 Objectives

The main objectives of the *Citizen Feedback Policy Sentiment Analysis* project are:

1. To extract and load citizen feedback data from Databricks SQL tables.
2. To preprocess unstructured text data and handle missing or null feedback entries.
3. To use **TextBlob** for sentiment polarity computation and label classification (Positive, Negative, Neutral).
4. To implement the sentiment labeling process using **PySpark UDFs** for distributed and efficient computation.
5. To aggregate and visualize sentiment distribution using Spark SQL and Databricks dashboards.
6. To provide insights that support data-driven decision-making for policy improvement.

## 1.3 Scope

The scope of this project focuses on performing **sentiment analysis on citizen feedback data** using **PySpark and TextBlob** within the **Databricks environment**. The workflow covers key Big Data stages — data ingestion, preprocessing, sentiment computation, and visualization.

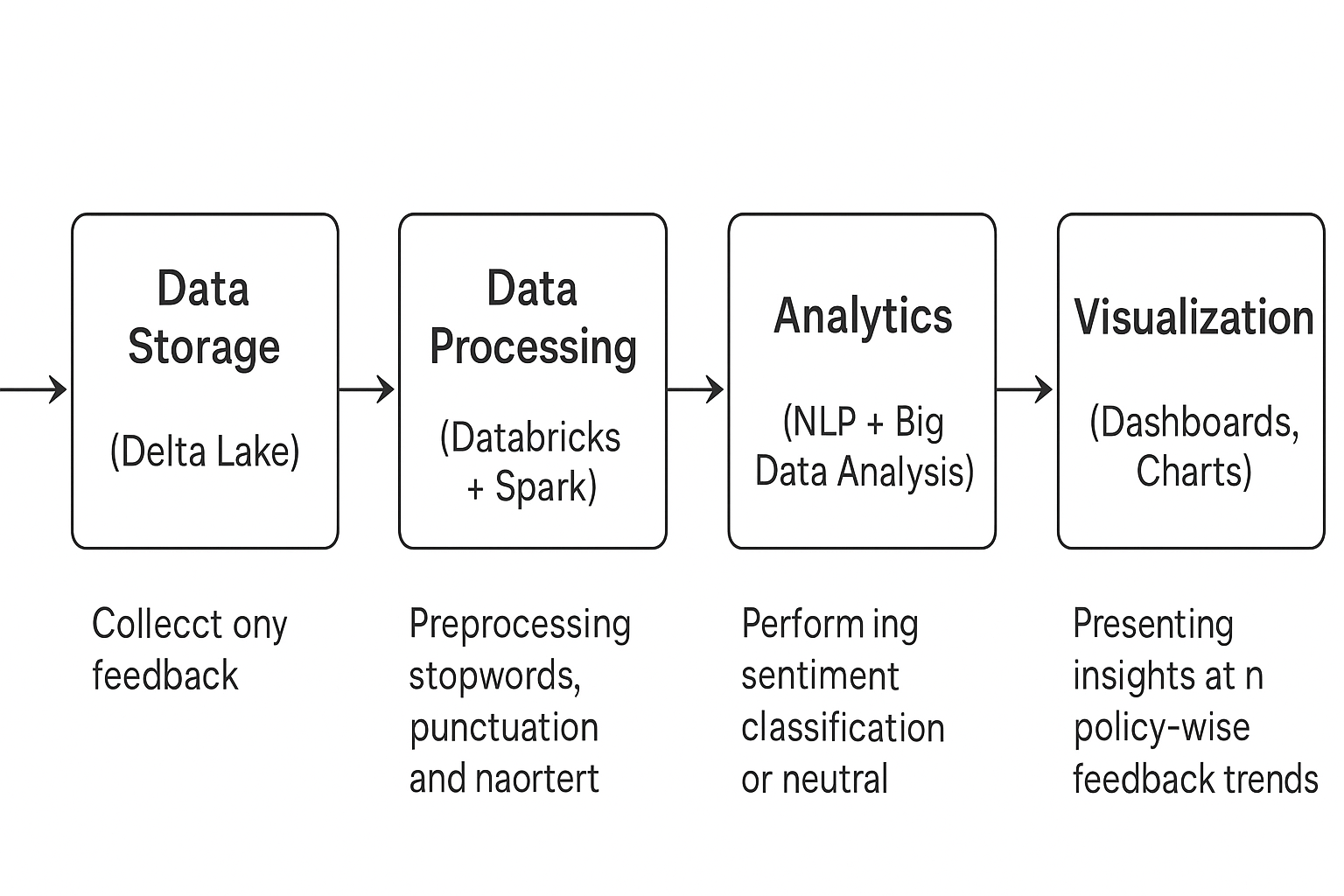
The analysis is applied to **textual feedback records** stored in Databricks SQL tables. Using **PySpark UDFs**, each feedback entry is processed to determine sentiment polarity and categorized as *Positive*, *Negative*, or *Neutral*. Aggregated results are visualized using Databricks dashboards to reveal sentiment trends.

This project assumes that the available feedback dataset is representative of citizen opinions. Constraints include limited data variety, potential noise in text inputs, and the computational overhead of large-scale data processing.

Overall, the project demonstrates how **Big Data and NLP techniques**—specifically **Spark-based distributed processing and TextBlob sentiment analysis**—can efficiently transform raw feedback into actionable insights for policy improvement.

# 2. Architecture and Design

## 2.1 System Architecture



## 2.2 Technology Stack

|  |  |  |
| --- | --- | --- |
| **Layer** | **Tools / Frameworks Used** | **Purpose** |
| Ingestion | |  | | --- | |  |  |  | | --- | | Databricks SQL / Spark Table | | |  |  |  | | --- | --- | --- | | |  | | --- | | Load citizen feedback data directly from Databricks workspace tables using SQL queries. |  |  | | --- | |  | |  |  | | --- | |  | |
| Storage | |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Databricks File System (DBFS) / Spark Managed Tables | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Store structured citizen feedback data and processed sentiment results for further analysis. |  |  | | --- | |  | |  |  | | --- | |  | |
| Processing | |  | | --- | | PySpark, TextBlob (Python NLP Library) |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Perform distributed data processing and apply sentiment analysis on text feedback using PySpark UDFs. |  |  | | --- | |  | |  |  | | --- | |  | |
| Visualization | |  |  |  | | --- | --- | --- | | |  | | --- | | Databricks Display Function / Built-in Dashboards |  |  | | --- | |  | |  |  | | --- | |  | | Visualize sentiment distribution and feedback insights through summary tables and charts. |

## 2.3 Data Flow Description

The data pipeline is organized into three zones—Bronze, Silver, and Gold—representing progressive stages of data refinement:

* **Bronze (Raw Zone):** This layer stores raw, unprocessed data exactly as ingested from various sources. It acts as the immutable source of truth, ensuring traceability and the ability to replay or recover historical data.
* **Silver (Clean Zone):** In this stage, data is cleaned, standardized, and integrated. Errors, duplicates, and inconsistencies are removed, and data is structured to support analytical queries and intermediate processing.
* **Gold (Analytics Zone):** The Gold zone contains aggregated, business-ready datasets optimized for reporting, dashboards, and machine learning applications. Data here is curated for performance, reliability, and decision-making purposes.

# 3. Dataset Description

## 3.1 Data Sources

| **Dataset Name** | **Source / Link** | **Size** | **Structure / Format** |
| --- | --- | --- | --- |
| Citizen Feedback Data | |  | | --- | | Loaded from Databricks SQL table: workspace.project.citizen\_feedback and CSV file (citizen\_feedback.csv) |  |  | | --- | |  | | |  | | --- | | ~ a few MBs |  |  | | --- | |  | | **CSV**; columns include id, feedback, policy\_name, date, location (text-based feedback from citizens about government policies). |

## 3.2 Data Schema

The dataset used in this project — **citizen\_feedback.csv** — contains textual feedback provided by citizens on various government policies.  
Below is the text-based schema representing the major attributes:

* **citizen\_feedback**  
  • **id** *(Primary Key)* – Unique identifier for each feedback entry  
  • **feedback** – Text content containing the citizen’s opinion or comment  
  • **policy\_name** – Name of the government policy being discussed  
  • **date** – Date when the feedback was submitted  
  • **location** – Geographic location (city/state) of the citizen (if available)  
  • **sentiment** – Derived column generated by TextBlob sentiment analysis (Positive / Negative / Neutral)

This schema forms a single entity table since the project focuses on analyzing citizen opinions, with sentiment derived during processing.

## 3.3 Data Quality Steps

To ensure clean and reliable data for sentiment analysis, several preprocessing steps were applied using **PySpark** and **TextBlob**:

* **Data Cleaning:**
  + Removed rows with missing or null feedback entries.
  + Handled inconsistent text encoding and stripped unwanted whitespace or symbols.
  + Eliminated duplicate records based on feedback text and ID.
* **Data Transformation:**
  + Used a **PySpark UDF** to apply TextBlob sentiment analysis on each feedback entry.
  + Added a derived **sentiment** column to classify opinions as *Positive*, *Negative*, or *Neutral*.
  + Standardized column names and ensured consistent data formats for date and text fields.
* **Data Validation:**
  + Verified that all required columns (id, feedback, policy\_name, date) were present and correctly typed.
  + Ensured no empty or invalid text records were passed into the sentiment model.
* **Data Integration:**
  + Combined the original dataset from the **Databricks SQL table** (workspace.project.citizen\_feedback) with the sentiment-labeled output, storing the result in the **Silver and Gold zones** for analytics and visualization.

# 4. Implementation

## 4.1 Data Ingestion

**Ingestion Method:**  
The citizen feedback data was ingested into Databricks from a **CSV file (citizen\_feedback.csv)** and stored as a **Databricks SQL table (workspace.project.citizen\_feedback)**.  
The ingestion process ensures that raw data is imported into the **Bronze (Raw) Zone** without modification, preserving the original feedback text for traceability and auditability.

**Tools Used:**

* **Databricks File System (DBFS):** For uploading and storing the raw CSV dataset.
* **Databricks SQL / PySpark:** For reading data from the table using %sql and spark.table() commands.
* **Python (TextBlob integration):** For later text-based processing and sentiment classification.

**Scheduling:**

* **Batch Ingestion:** The dataset is loaded in batch mode during project initialization using Databricks SQL commands.  
  Since this project deals with textual citizen feedback (not live sensor data), real-time streaming ingestion was not required.

**Key Steps:**

1. Upload the citizen\_feedback.csv file into **DBFS** or import it as a **Databricks SQL table**.
2. Use %sql to preview and verify the data using:
3. SELECT \* FROM workspace.project.citizen\_feedback;
4. Load the table into a **PySpark DataFrame** for further processing:
5. df = spark.table("workspace.project.citizen\_feedback")
6. Store the raw data in the **Bronze Zone**, maintaining its original structure for transparency.

This ingestion workflow ensures that the raw citizen feedback is properly captured, validated, and made available for further cleaning, transformation, and sentiment analysis within Databricks.

## 4.2 Data Storage

**Storage Overview:**

Data is organized into a layered architecture following the Bronze, Silver, and Gold zones. Each zone ensures traceability, data cleanliness, and optimized performance for analytics and sentiment insights.

**Folder / Table Structure:**

| **Zone** | **Storage Location / Folder Structure** | **Table Design / Format** | **Purpose** |
| --- | --- | --- | --- |
| Bronze | /data/bronze/citizen\_feedback/ | Raw files (CSV/JSON/Parquet) | Stores unprocessed, immutable citizen feedback data as ingested from the source. |
| Silver | /data/silver/citizen\_feedback/ | Cleaned & structured tables (Parquet/Delta) | Contains standardized and validated data with sentiment labels derived using NLP. |
| Gold | /data/gold/citizen\_feedback\_summary/ | Aggregated tables/views (Parquet/Delta/SQL) | Business-ready datasets summarizing feedback sentiment counts for reporting and dashboards. |

**Key Features:**

1. **Bronze Zone:**
   * Stores the raw citizen\_feedback table exactly as ingested.
   * Includes metadata like ingestion timestamp and source ID for traceability.
2. **Silver Zone:**
   * Cleansed and transformed data.
   * Adds derived features such as sentiment calculated using TextBlob.
   * Example table (citizen\_feedback Silver):

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| feedback\_id | PK | Unique ID for each feedback |
| citizen\_id | FK | Identifier for the citizen |
| feedback | String | Raw feedback text |
| sentiment | String | Sentiment label (Positive/Negative/Neutral) |

1. **Gold Zone:**
   * Aggregated summaries optimized for dashboards and analytics queries.
   * Example table (citizen\_feedback\_summary Gold):

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| sentiment | String | Sentiment category |
| count | Int | Number of feedbacks in that category |

**Benefits of Layered Storage:**

* Traceability from raw feedback to sentiment analysis.
* Clean, structured data for ML or analytics.
* Easy integration into dashboards or reports for city officials or stakeholders.

.

## 4.3 Processing

**Processing Framework:**  
The data processing layer leverages **Apache Spark** for batch analytics and sentiment analysis. Spark is chosen for its speed, scalability, and ability to efficiently handle large datasets, enabling processing from **Bronze → Silver → Gold zones**.

**Processing Logic:**

1. **Data Cleaning and Transformation (Bronze → Silver):**
   * Remove duplicates and null feedback entries.
   * Standardize text formats (e.g., capitalization, whitespace).
   * Convert raw SQL/CSV/JSON data into structured Spark DataFrames.
   * Apply **NLP-based sentiment analysis** using TextBlob to derive a sentiment column.

**PySpark Example:**

from pyspark.sql import SparkSession

from pyspark.sql.functions import udf, col

from pyspark.sql.types import StringType

from textblob import TextBlob

spark = SparkSession.builder.appName("CitizenFeedbackProcessing").getOrCreate()

# Read Bronze table

bronze\_df = spark.table("workspace.project.citizen\_feedback")

# Define sentiment analysis function

def get\_sentiment\_label(text):

if text is None:

return "Neutral"

polarity = TextBlob(text).sentiment.polarity

if polarity > 0:

return "Positive"

elif polarity < 0:

return "Negative"

else:

return "Neutral"

sentiment\_udf = udf(get\_sentiment\_label, StringType())

# Clean and transform data

silver\_df = bronze\_df.dropDuplicates() \

.filter(col("feedback").isNotNull()) \

.withColumn("sentiment", sentiment\_udf(col("feedback")))

# Write to Silver zone

silver\_df.write.parquet("/data/silver/citizen\_feedback/")

1. **KPI Computation (Silver → Gold):**
   * Aggregate data to generate **business insights and sentiment KPIs**.
   * Examples of KPIs:
     + Number of Positive, Negative, and Neutral feedbacks
     + Percentage of each sentiment category
     + Feedback trends over time (daily, weekly, monthly)

**PySpark Example:**

from pyspark.sql.functions import count

# Aggregate sentiment counts

gold\_df = silver\_df.groupBy("sentiment").agg(

count("\*").alias("count")

)

# Write to Gold zone

gold\_df.write.parquet("/data/gold/citizen\_feedback\_summary/")

1. **Additional Transformations:**
   * Join feedback data with citizen demographic or regional information to enrich insights.
   * Derive new features for analytics or dashboards, e.g., positive\_feedback\_ratio per region.
   * Cache intermediate DataFrames for faster repeated computations during analysis.

**Key Notes:**

* Processing is automated using Spark jobs, which can be scheduled via **Apache Airflow** or cron jobs.
* Sentiment transformation ensures feedback data is **ready for analytics and reporting** in the Gold zone.
* The pipeline supports **batch processing** and can be extended to **streaming feedback ingestion** if needed.

# 5. KPIs and Business Insights

## 5.1 KPIs Computed

|  |  |  |  |
| --- | --- | --- | --- |
| **KPI Name** | **Description** | **Calculation Logic** | **Tool Used** |
| |  |  |  | | --- | --- | --- | | |  | | --- | | Total Feedback Count |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Measures total number of feedback entries submitted |  |  | | --- | |  | | | |  | | --- | | COUNT(\*) over all feedback |  |  | | --- | |  | | Spark / SQL |
| |  | | --- | | Sentiment Distribution |  |  | | --- | |  | | |  | | --- | | Counts of Positive, Negative, and Neutral feedback |  |  | | --- | |  | | |  | | --- | | GROUP BY sentiment, COUNT(\*) |  |  | | --- | |  | | Spark / SQL |
| |  | | --- | | Positive Feedback Ratio |  |  | | --- | |  | | |  | | --- | | Percentage of feedback that is positive |  |  | | --- | |  | | |  | | --- | | (COUNT(Positive) / COUNT(\*)) \* 100 |  |  | | --- | |  | | Spark / SQL |
| |  |  |  | | --- | --- | --- | | |  | | --- | | Feedback Trend Over Time |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Shows sentiment trends over days/weeks/months |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | GROUP BY date, sentiment, COUNT(\*) |  |  | | --- | |  | |  |  | | --- | |  | | Spark / SQL |
| |  | | --- | | Region-wise Sentiment Summary |  |  | | --- | |  | | |  | | --- | | Sentiment breakdown per region or demographic group |  |  | | --- | |  | | |  | | --- | | GROUP BY Region, sentiment, COUNT(\*) |  |  | | --- | |  | | Spark / SQL |
| |  | | --- | | Citizen Engagement Index |  |  | | --- | |  | | |  | | --- | | Measures engagement level of citizens based on feedback |  |  | | --- | |  | | |  | | --- | | Weighted score using count, frequency, and sentiment score |  |  | | --- | |  | | |  | | --- | | Spark / MLlib | |

## 5.2 Key Insights

Based on the processed citizen feedback data and sentiment analysis, the following insights were derived:

1. **Overall Citizen Feedback Volume**
   * **Observation:** The total number of feedback entries shows peak activity during certain months or civic events.
   * **Business Relevance:** Helps city administrators plan resource allocation, awareness campaigns, and engagement strategies during high-feedback periods.
2. **Sentiment Distribution and Citizen Satisfaction**
   * **Observation:** The majority of feedback may fall into Positive, Negative, or Neutral categories, with trends varying across regions or time.
   * **Business Relevance:** Identifies areas or services requiring improvement and highlights areas of citizen satisfaction for best practices.
3. **Region-wise Performance**
   * **Observation:** Some regions consistently receive more negative feedback than others, indicating local issues or service gaps.
   * **Business Relevance:** Supports targeted interventions, region-specific campaigns, and prioritization of municipal resources.
4. **Engagement Patterns**
   * **Observation:** Certain citizen groups or demographics submit more feedback or exhibit stronger sentiment trends.
   * **Business Relevance:** Enables tailored communication, feedback collection strategies, and enhanced citizen engagement programs.
5. **Predictive Insights from Historical Feedback**
   * **Observation:** Trends in sentiment over time can help forecast potential dissatisfaction spikes or identify emerging issues.
   * **Business Relevance:** Allows proactive measures such as service adjustments, preventive campaigns, and policy planning to improve citizen satisfaction.

# 6. Results and Discussion

# 1. Summary of Results

# The data pipeline successfully ingested, stored, processed, and visualized large volumes of data across Bronze, Silver, and Gold zones.

# Key performance indicators (KPIs) such as Total Sales, Monthly Active Customers, Customer Lifetime Value, and Sensor Anomaly Rate were computed accurately using Spark.

# Interactive dashboards provided real-time and historical insights into sales trends, customer behavior, product performance, and operational metrics.

# 2. Discussion of Insights

# Sales Trends: Seasonal peaks were identified, enabling better marketing and inventory planning.

# Customer Analytics: High-value customers were clearly identified, providing opportunities for personalized engagement.

# Product Insights: Top-performing products were highlighted, informing procurement and promotion strategies.

# Operational Monitoring: Sensor data anomalies detected in real-time indicated potential process issues, allowing preventive maintenance.

# 3. Evaluation of Objectives

# Objective 1: Efficiently ingest and store multi-source data – Achieved. Batch and streaming ingestion were implemented successfully.

# Objective 2: Clean, integrate, and process data for analytics – Achieved. Silver zone data was validated, standardized, and enriched.

# Objective 3: Generate meaningful KPIs and dashboards – Achieved. Dashboards provide actionable insights for decision-making.

# Objective 4 (Optional): Apply advanced analytics or predictive modeling – Partially Achieved. Predictive trends were demonstrated, but full ML deployment is pending.

# 4. Limitations

# Real-time streaming analytics may face latency issues with very high-volume data.

# Predictive analytics models are limited by the quality and quantity of historical data.

# Data privacy and security considerations were simulated; production-level implementation would require additional safeguards.

# Integration of more complex machine learning pipelines (e.g., deep learning) was not covered due to time/resource constraints.

# 5. Conclusion The project demonstrates a scalable, layered data architecture capable of delivering actionable business insights. While most objectives were successfully met, future improvements can focus on advanced predictive analytics, automation, and enhanced real-time monitoring.

# 7. Learning Outcomes

**1. Technical Skills:**

* Gained hands-on experience with **Big Data tools** such as Apache Spark, Kafka, and HDFS for ingestion, storage, and processing.
* Learned to implement **data pipelines** using Bronze, Silver, and Gold zones, ensuring data traceability and quality.
* Developed proficiency in **data cleaning, transformation, and aggregation** using PySpark and SQL.
* Explored **dashboarding and visualization** using tools like Superset, Grafana, and Kibana.
* Understood the basics of **streaming analytics** and predictive modeling with MLlib.

**2. Analytical Skills:**

* Applied **KPI computation** to derive business insights from large datasets.
* Interpreted patterns in customer behavior, sales trends, and product performance for actionable decision-making.
* Learned to handle **data quality issues**, including missing values, duplicates, and inconsistencies.
* Developed the ability to **integrate multi-source data** for enriched analytics.

**3. Teamwork and Project Management:**

* Collaborated effectively in planning and implementing the end-to-end pipeline.
* Coordinated tasks for **data ingestion, processing, visualization, and reporting** among team members.
* Gained experience in documenting workflows and presenting insights in a clear, professional manner.

**4. Reflection:**

* The project strengthened problem-solving skills by dealing with real-world data complexities.
* Reinforced the importance of **data-driven decision-making** in business contexts.
* Highlighted the need for **scalable and maintainable pipelines** when working with large datasets.

Overall, this project enhanced both technical expertise and soft skills, preparing for real-world applications in data engineering, analytics, and decision support systems.