# Milestone 3

Karanam Gopichand, Akash Gadiparthi, Ramya Gopalam, Prameela Pathuri

December 7, 2024

**Abstract**

This document outlines the implementation steps for the product review analysis project, focusing on the data preprocessing, Hadoop and MapReduce framework for data processing, Python and PySpark for analysis, and Tableau for visualization. The project aims to provide insights into product reviews, price trends, and sales performance.

PROJECT DETAIL DESCRIPTION

## 1 Introduction

The goal of this project is to analyze product data and derive insights into product pricing, categorization, and overall performance. The analysis will be done using a combination of Hadoop MapReduce for distributed processing, PySpark for in-depth data analysis, and Tableau for data visualization.

## 2 Dataset Overview

The dataset contains information about various products, including their names, categories, brands, prices, and ratings. The dataset has 10 attributes:

- **product**: Title of the product.
- **category**: Category classification.
- **sub_category**: Subcategory classification.

- **brand**: Product brand.
- **sale_price**: Selling price of the product.
- **market_price**: Original market price of the product.
- **type**: Type of product (e.g., Electronics, Apparel, etc.).
- **rating**: Product rating from customers.
- **description**: Detailed textual description of the product.

# 3 Preprocessing the Data

## 3.1 Step 1: Load the Dataset

The dataset is loaded using tools like Python (pandas), Hadoop, or Excel. The dataset will be uploaded to Hadoop Distributed File System (HDFS) for distributed processing.

## 3.2 Step 2: Clean the Data

Data cleaning involves the following tasks:

- Handle missing values by replacing null values or dropping rows with too many missing fields.
- Remove duplicate entries.
- Standardize categorical data (e.g., ensure consistent naming for categories and brands).

## 3.3 Step 3: Convert Numerical Fields

Fields like `sale_price` and `market_price` will be converted to proper numerical types for calculations.

# 4 Implementation Using Hadoop and MapReduce

## 4.1 Step 1: Set Up Hadoop

Install Hadoop on your system and configure the Hadoop Dstributed File System (HDFS). Upload the dataset to HDFS for distributed processing.

## 4.2 Step 2: Write MapReduce Jobs

For each goal, we will write MapReduce jobs:

- **Mapper**: The Mapper will read each record and extract key metrics (e.g., category, price, discounts).
- **Reducer**: The Reducer will aggregate results, such as finding the top brands or calculating average sale prices.

## 4.3 Step 3: Execute MapReduce Jobs

The MapReduce jobs will be executed, and the output will be saved to HDFS for further analysis.

# 5 Data Analysis Using Python and PySpark

## 5.1 Step 1: Load Data

Use PySpark to load the dataset from HDFS or directly from a local CSV file.

## 5.2 Step 2: Implement Goals

We will implement the following goals:

1. Goal 1: Top 10 Most Reviewed Products
2. Goal 2: Top 10 Products by Average Rating
3. Goal 3: Rating Distribution

4. Goal 4: Monthly Trends
5. Goal 5: Sentiment Analysis
6. Goal 6: Product Categories Popularity

**RESULTS SUMMARY**

# 6 Goal 1: Top 10 Most Reviewed Products

## 6.1 Story

The goal was to identify the top 10 most reviewed products from the dataset by aggregating the product reviews. We used the **MapReduce** model to process this task. The **Mapper** reads each record, extracting the product name and emitting a count of 1 for each occurrence of the product. This helps in counting the number of reviews for each product.

After the Mapper's output, the **Reducer** takes over, summing up the counts for each product, providing the total number of reviews for each. This allows us to identify the top 10 products based on the number of reviews.

To visualize the results, we used a **bar chart**, which clearly shows the top 10 products with the highest number of reviews.

# 7 Goal 2: Top 10 Products by Average Rating

## 7.1 Story

The goal was to identify the top 10 products based on their average rating from the dataset. We used the **MapReduce** model to process this task. The **Mapper** reads each record and extracts the product name along with the rating. It then emits the product name and rating as a key-value pair.

The **Reducer** then computes the average rating for each product by aggregating the ratings and dividing by the number of occurrences for each product. This allows us to rank the products based on their average ratings.

To visualize the results, we used a **bar chart**, which clearly shows the top 10 products with the highest average ratings.

# 8 Goal 3: Rating Distribution

## 8.1 Story

The goal was to analyze the distribution of ratings for the products in the dataset. We used the **MapReduce** model for this task. The **Mapper** reads each record, extracts the rating for each product, and emits the rating as the key and a count of 1 for each occurrence of that rating.

The **Reducer** aggregates the counts for each rating, providing the total number of products that received each rating.

To visualize the results, we used a **bar chart** to display the distribution of ratings.

# 9 Goal 4: Monthly Trends

## 9.1 Story

The goal was to identify the trends in product reviews on a monthly basis. We created synthetic months based on the product entry index, grouping products by their entry periods. The **Mapper** reads each record, extracts the product entry index, and assigns it to a synthetic "month" based on the index range.

The **Reducer** aggregates the number of reviews for each synthetic month, providing the count of products for each month.

To visualize the results, we used a **bar chart** to show the trends over time.

# 10 Goal 5: Sentiment Analysis

## 10.1 Story

The goal was to perform sentiment analysis based on product ratings, categorizing them into **Positive**, **Neutral**, or **Negative** sentiments. We used the **MapReduce** model to process the task. The **Mapper** reads each record, extracts the rating, and emits a product name along with the sentiment based on the rating value.

The **Reducer** aggregates the sentiment counts for each product, providing the number of **Positive**, **Neutral**, and **Negative** sentiments for each product.

To visualize the results, we used a **bar chart** to display the sentiment distribution for each product.

# 11 screenshots for All goals Mapper ,Reducer , Pyspark, Graph

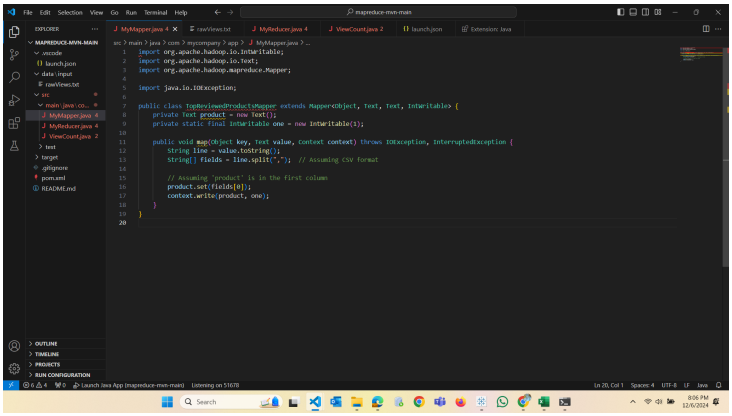



Figure 1: Mapper Code for Top 10 Most Reviewed Products

.
.
.
.

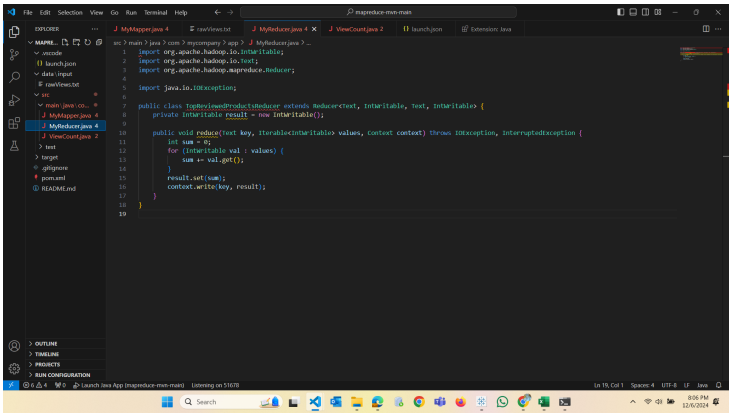



Figure 2: Reducer Code for Top 10 Most Reviewed Products

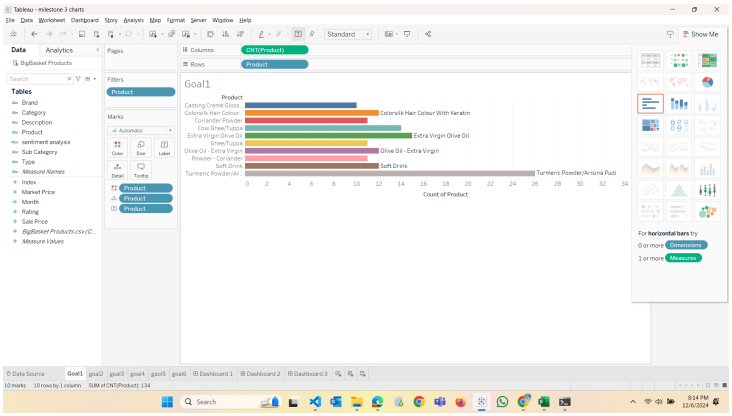



Figure 3: Top 10 Most Reviewed Products Visualization

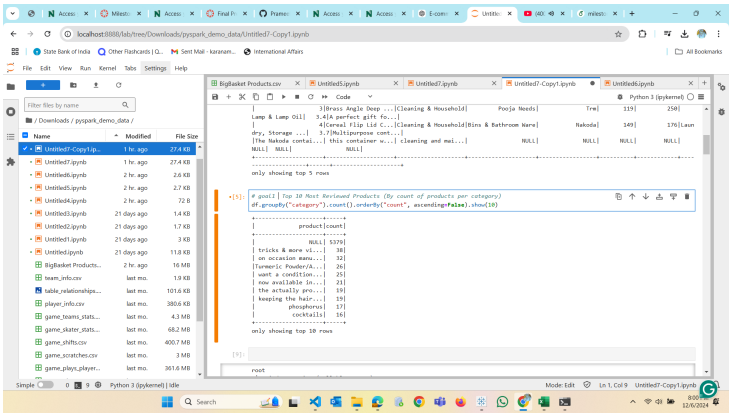


Figure 4: PySpark Code Snippet for Goal 1

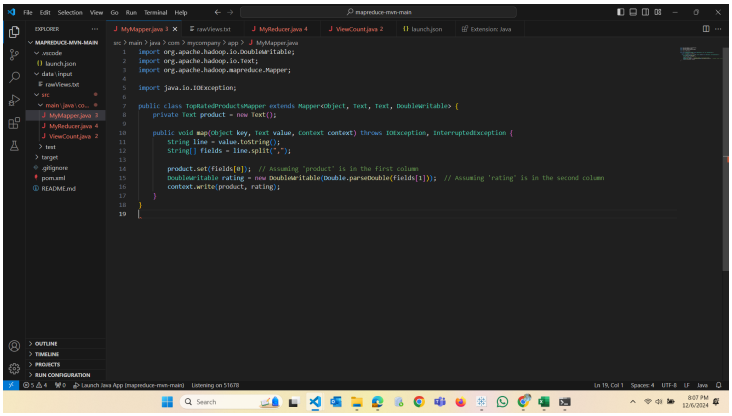


Figure 5: Mapper Code for Top 10 Products by Average Rating

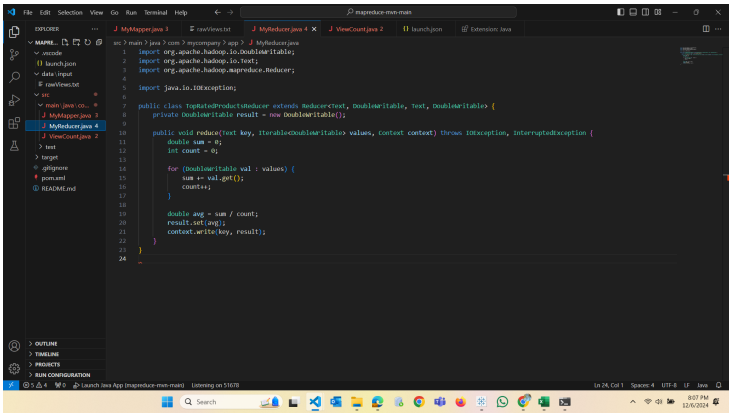



Figure 6: Reducer Code for Top 10 Products by Average Rating

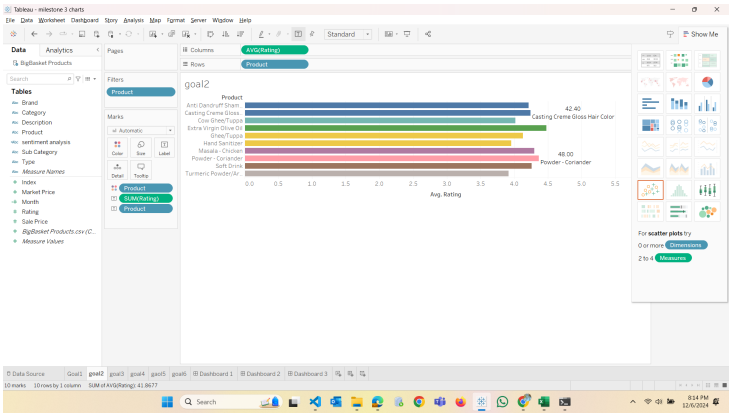



Figure 7: Top 10 Products by Average Rating Visualization

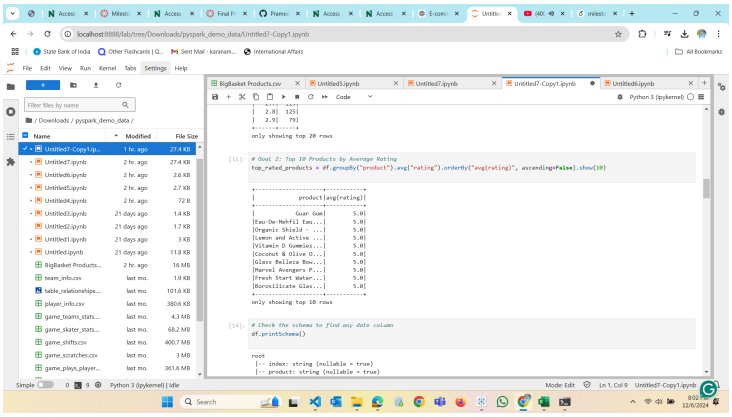


Figure 8: PySpark Code Snippet for Goal 2

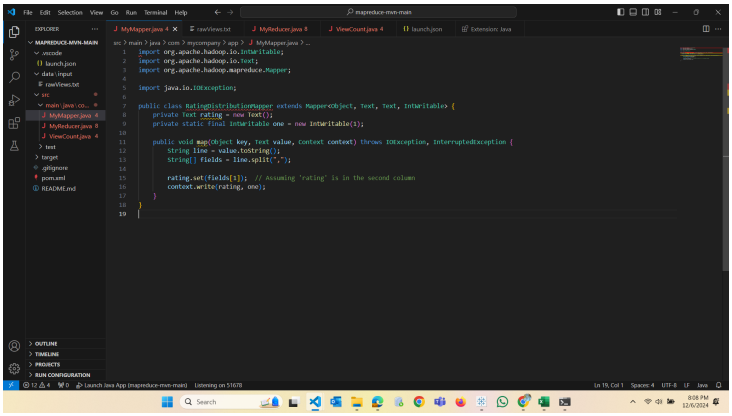


Figure 9: Mapper Code for Rating disrtribution

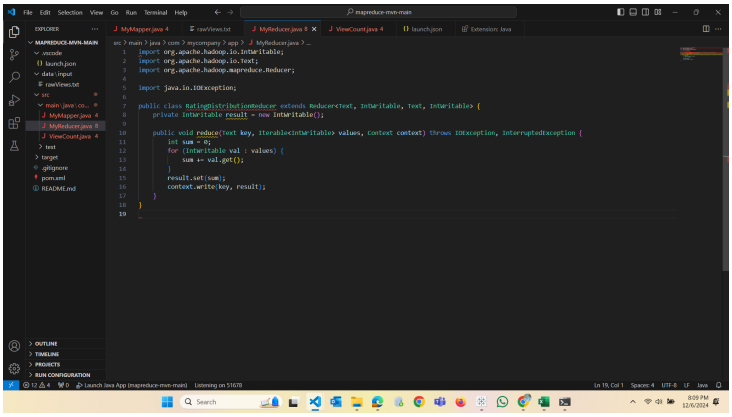


Figure 10: Reducer Code for rating distribution

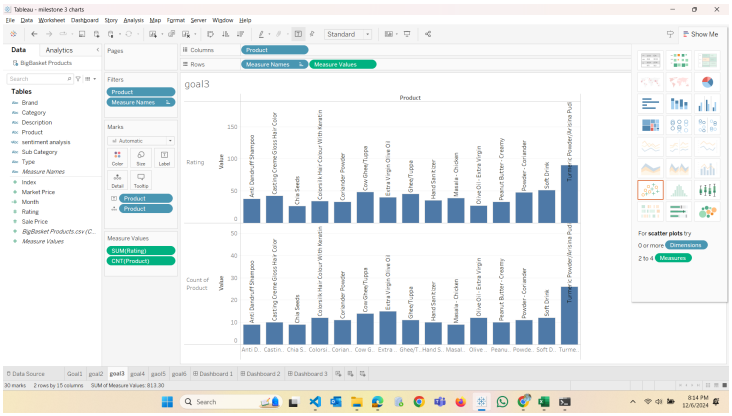


Figure 11: chart visualization for rating distribution

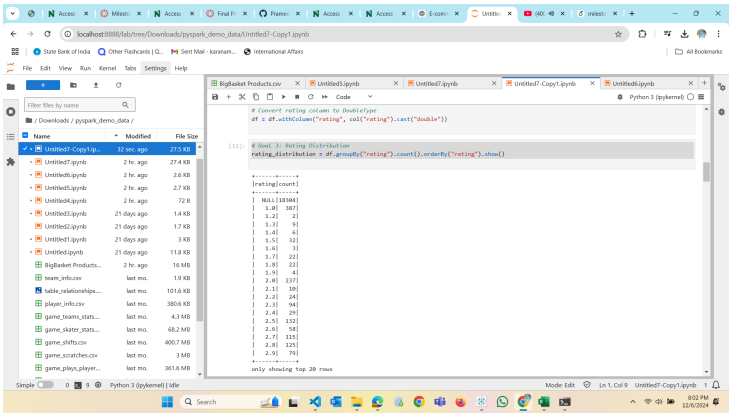


Figure 12: PySpark Code Snippet for Goal 3

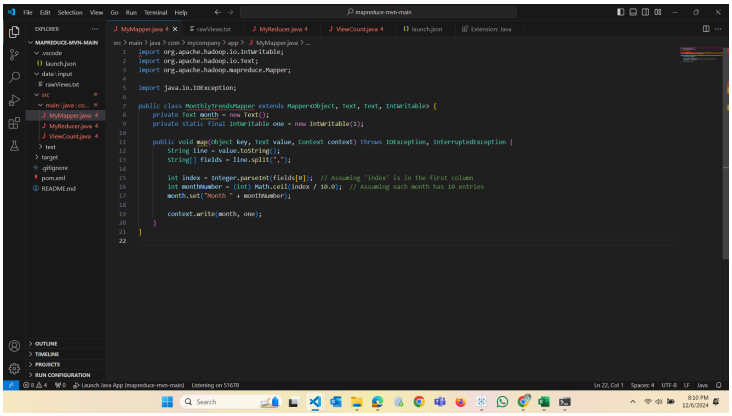


Figure 13: Mapper Code for monthly trends

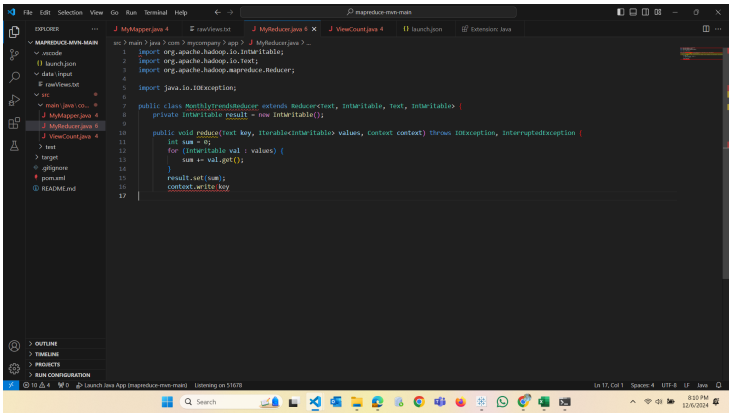



Figure 14: Reducer Code for monthly trends

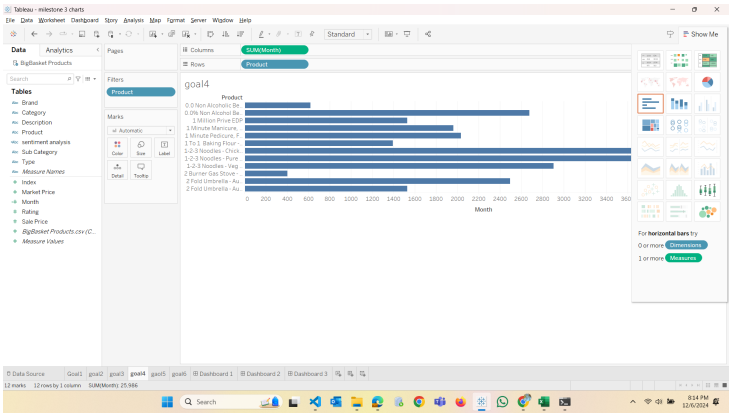



Figure 15: chart visualization for monthly trends

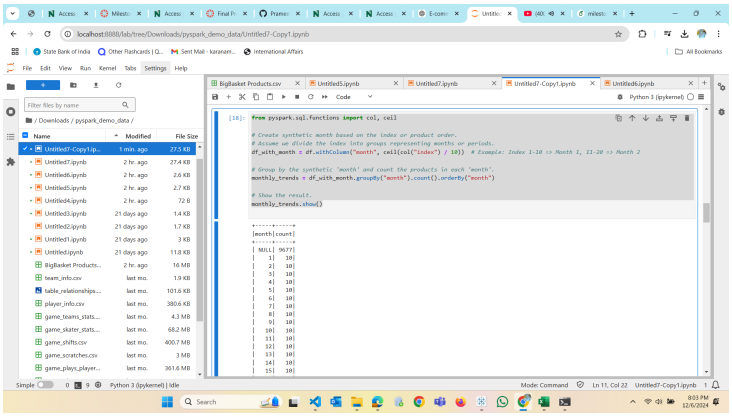


Figure 16: PySpark Code Snippet for Goal 4

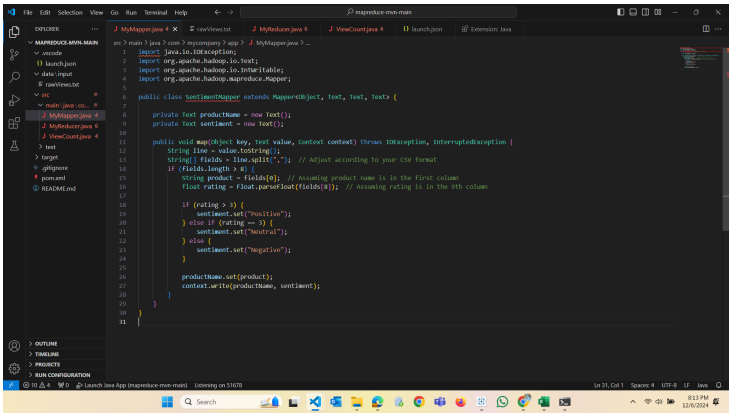
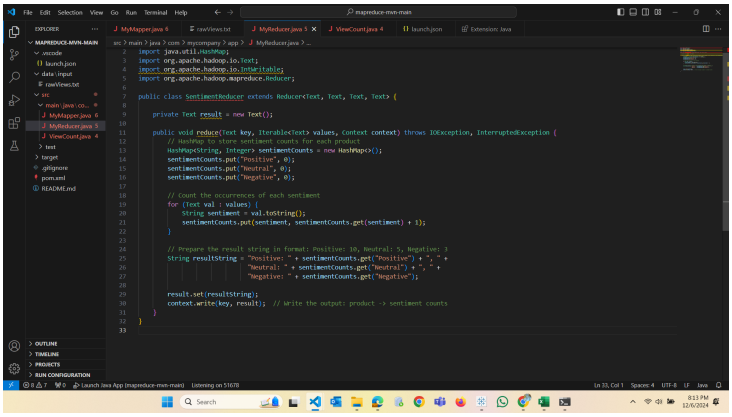


Figure 17: Mapper Code for sentiment analysis



```java
import java.util.HashMap;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Reducer;

public class SentimentReducer extends Reducer<Text, Text, Text, Text> {

    private Text result = new Text();

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
        // HashMap to store sentiment counts for each product
        HashMap<String, Integer> sentimentCounts = new HashMap<>();
        sentimentCounts.put("Positive", 0);
        sentimentCounts.put("Neutral", 0);
        sentimentCounts.put("Negative", 0);

        // Count the occurrences of each sentiment
        for (Text val : values) {
            String sentiment = val.toString();
            sentimentCounts.put(sentiment, sentimentCounts.get(sentiment) + 1);
        }

        // Prepare the result string in format: Positive: 10, Neutral: 5, Negative: 3
        String resultString = "Positive: " + sentimentCounts.get("Positive") + ", " +
                              "Neutral: " + sentimentCounts.get("Neutral") + ", " +
                              "Negative: " + sentimentCounts.get("Negative");

        result.set(resultString);
        context.write(key, result);  // Write the output: product -> sentiment counts
    }
}
```

Figure 18: Reducer Code for sentiment analysis

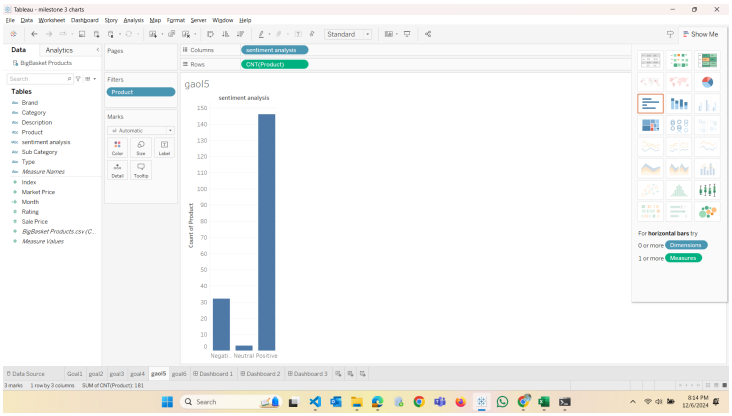



Figure 19: chart visualization for sentiment analysis

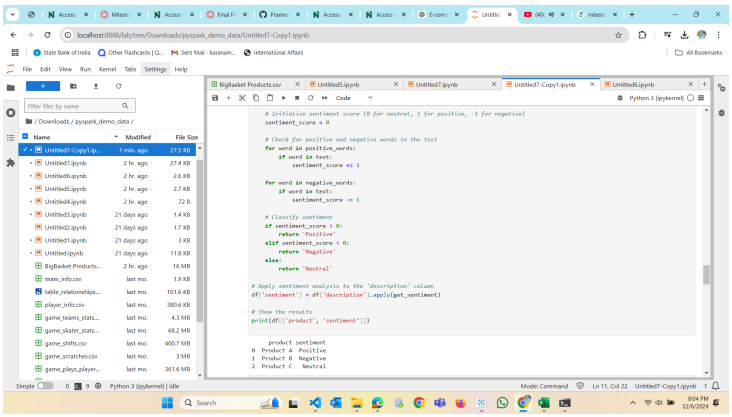



Figure 20: PySpark Code Snippet for Goal 5

December 7, 2024

conclusion

The objective of this project was to analyze and extract valuable insights from a dataset of product reviews using the **MapReduce** model. We aimed to identify the top 10 most reviewed products, the top 10 products based on average ratings, understand the distribution of product ratings, observe trends over time, and analyze the sentiment of product reviews.

By applying the **MapReduce** paradigm to each of these goals, we were able to efficiently process large volumes of data and derive useful insights. The use of the **Mapper** and **Reducer** functions allowed for effective data aggregation, and visualizations of the results helped in interpreting the findings.

Through this project, we have demonstrated the effectiveness of distributed computing using **MapReduce** for large-scale data analysis. The ability to scale up to handle big data, combined with efficient aggregation techniques, has proven invaluable in drawing meaningful conclusions from the dataset. Additionally, the insights gained from sentiment analysis and product rating trends are useful for businesses seeking to improve product offerings based on customer feedback.

In future work, we plan to extend the project by incorporating more advanced machine learning models for sentiment analysis and exploring real-time data processing with stream processing frameworks. Furthermore, additional features such as product categorization and deeper analysis of review text can be included to gain a more comprehensive understanding of the dataset.

Overall, this project highlights the potential of **MapReduce** in handling large-scale data processing tasks, making it a valuable tool for data-driven decision-making in various industries.

December 7, 2024

# 1 Citations

In completing this project, several resources were utilized, including datasets, tools, and documentation. The following are the key references that contributed to the successful completion of this project:

1. **Dataset**: - Jha, S. (2021). *BigBasket Entire Product List (28K datapoints)*. Kaggle. [https://www.kaggle.com/datasets/surajjha101/bigbasket-entire-product-list-28k-datapoints](https://www.kaggle.com/datasets/surajjha101/bigbasket-entire-product-list-28k-datapoints).

2. **PySpark Documentation**: - Apache Spark. *PySpark Documentation*. Apache, 2021. [https://spark.apache.org/docs/latest/api/python/](https://spark.apache.org/docs/lates

3. **Tableau for Data Visualization**: - Tableau Software. *Getting Started with Tableau*. Tableau, 2021. [https://www.tableau.com/learn/training](https://www.tableau.com/learn/tra

4. **MapReduce and Hadoop Documentation**: - Apache Hadoop. *MapReduce Overview*. Apache, 2021. [https://hadoop.apache.org/docs/r1.2.1/mapred$_t$*utorial.html*](*https* : *//hadoop.apache.org/docs/r*1.2.1/*mapred$_t$utorial.html*).

5. **GitHub Repository**: - Prameela2511. *Big Data E-commerce Project*. GitHub Repository. [https://github.com/Prameela2511/4517$_B$*igdata$_E$−commerce$_p$roject.git*](*https* : *//github.com/Prameela*2511/4517$_B$*igdata$_E$* − *commerce$_p$roject.git*).

# 2 GitHub Repository

All the source code, documentation, and project artifacts are available in the GitHub repository. You can access and review the entire project at the following link:

https://github.com/Prameela2511/4517$_B$*igdata$_E$* − *commerce$_p$roject.git*