Aaron Stefano F. Bonaobra

Dave Jazmin

Junmar Perez

BSIT – 3A, Big Data Analysis

LAB 2 ACTIVITY DOCUMENTATION:

Title: Spark RDD Partitioning and Transformation Pipeline

1. Introduction This lab activity demonstrates how to utilize Apache Spark to process a dataset using partitioning strategies and a transformation pipeline. The dataset used is "supermarket_sales.csv," which contains sales data from different branches of a supermarket. The objective is to apply at least two partitioning strategies and perform transformations such as filtering, sorting, and aggregation.

2. Dataset Description The dataset includes the following attributes:

- Invoice ID
- Branch
- City
- Customer Type
- Gender
- Product Line
- Unit Price
- Quantity
- Tax 5%
- Total
- Date
- Time
- Payment Method
- COGS
- Gross Margin Percentage
- Gross Income
- Rating

3. Partition Strategies Used

1. **Manual Partitioning by City**
    - The dataset was manually repartitioned into 4 partitions based on the "City" column using the repartition() function.
    - This ensures that data is grouped by city, improving query performance when filtering by city.
2. **Implicit Partitioning When Writing to File**

- When writing the transformed data to CSV format, Spark automatically partitions the data based on its internal optimization.

-This allows for efficient distributed processing and storage management.

4. Transformation Pipeline Applied The following transformations were applied to the dataset:

1. **Repartition Data by City**
    - df = df.repartition(4, "City")
    - This divides the dataset into four partitions based on the city column.
2. **Group Data by Product Line and Summarize Quantity**
    - summary_df = df.groupBy("Product line").sum("Quantity")
    - This groups data by product line and calculates the total quantity sold per product category.
3. **Sort the Summarized Data in Descending Order**
    - sorted_df = summary_df.orderBy("sum(Quantity)", ascending=False)
    - The summarized data is sorted based on the total quantity sold to identify the most popular product lines.
4. **Write the Transformed Data to CSV**
    - sorted_df.write.mode("overwrite").csv("C:/Users/Aaron/Documents/sales_output", header=True)
    - The processed data is written to an output directory as a CSV file.

5. Execution and Results

- The dataset was successfully loaded and partitioned by city.

- The transformation pipeline grouped and sorted the data by product line and quantity.

- The final dataset was saved as partitioned CSV files in the specified directory.

6. Conclusion This activity demonstrated the use of partitioning strategies and transformation pipelines in Spark. By partitioning data efficiently and applying transformations such as grouping, sorting, and aggregation, we can process large datasets in a distributed computing environment effectively. Apache Spark provides powerful tools for handling big data, improving both performance and scalability.

7. References

- Apache Spark Documentation: https://spark.apache.org/docs/latest/

- Dataset Source: Provided by instructor or downloaded from Kaggle