Name: Perez, Junmar A. Lab No. 3

Git Repo/Colab Link: https://github.com/junmsr/Perez_Elec2.git Date: 03/20/2025

Objective

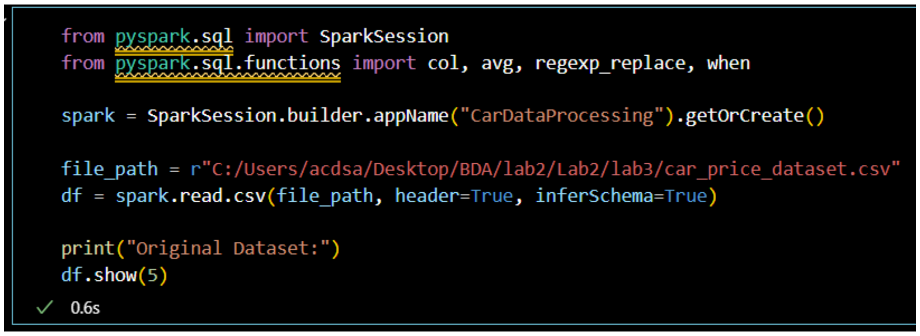
The purpose of this documentation is to outline the steps taken to preprocess a dataset using Apache Spark. This includes data loading, cleaning, transformation, and aggregation to derive meaningful insights.

Introduction

Apache Spark is a powerful distributed computing system used for big data processing. It provides efficient ways to handle large datasets using the PySpark library in Python.

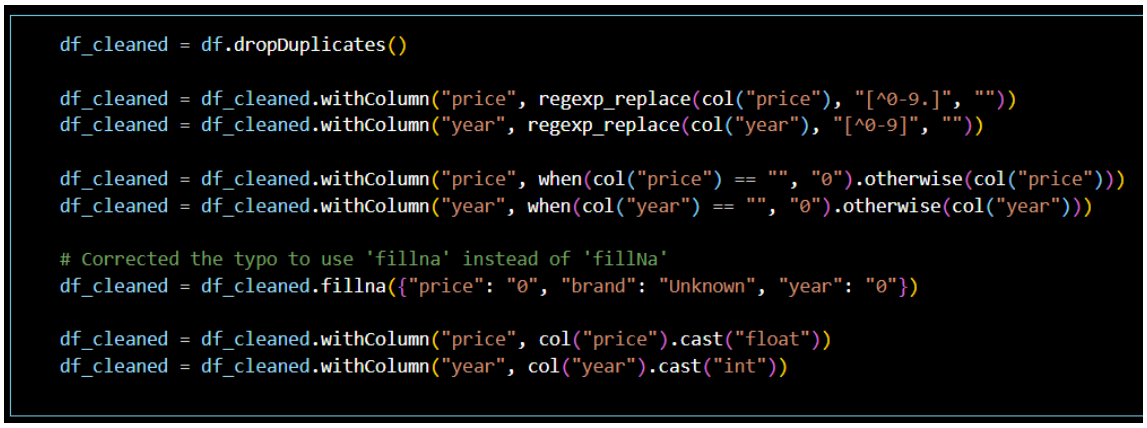
Methodology

1. Initialize Spark Session and load dataset



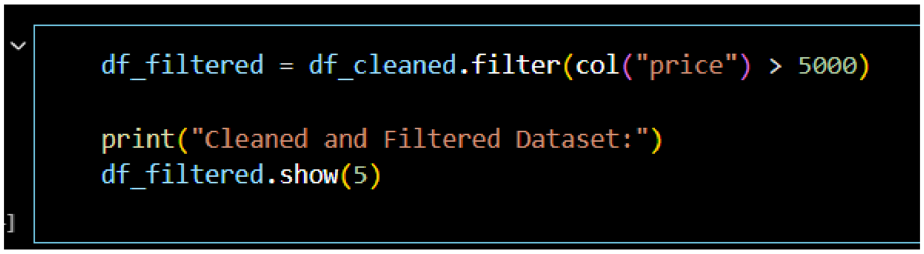
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, regexp_replace, when

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

file_path = r"C:/Users/acdsa/Desktop/BDA/lab2/Lab2/lab3/car_price_dataset.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

print("Original Dataset:")
df.show(5)
✓ 0.6s
```

2. Several preprocessing steps were performed, including removing duplicate records, handling missing and incorrect values, and converting data types for consistency.



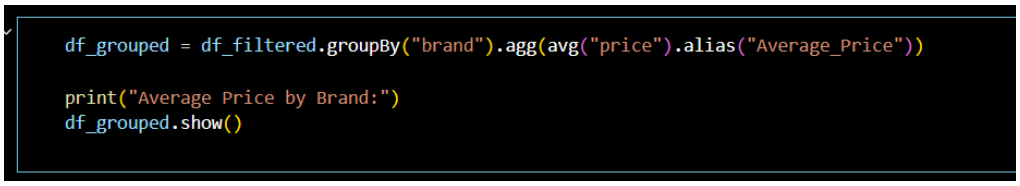
```
df_cleaned = df.dropDuplicates()

df_cleaned = df_cleaned.withColumn("price", regexp_replace(col("price"), "[^0-9.]", ""))
df_cleaned = df_cleaned.withColumn("year", regexp_replace(col("year"), "[^0-9]", ""))

df_cleaned = df_cleaned.withColumn("price", when(col("price") == "", "0").otherwise(col("price")))
df_cleaned = df_cleaned.withColumn("year", when(col("year") == "", "0").otherwise(col("year")))

# Corrected the typo to use 'fillna' instead of 'fillNa'
df_cleaned = df_cleaned.fillna({"price": "0", "brand": "Unknown", "year": "0"})

df_cleaned = df_cleaned.withColumn("price", col("price").cast("float"))
df_cleaned = df_cleaned.withColumn("year", col("year").cast("int"))
```

3. A filter was applied to remove records where the price was below 5000.



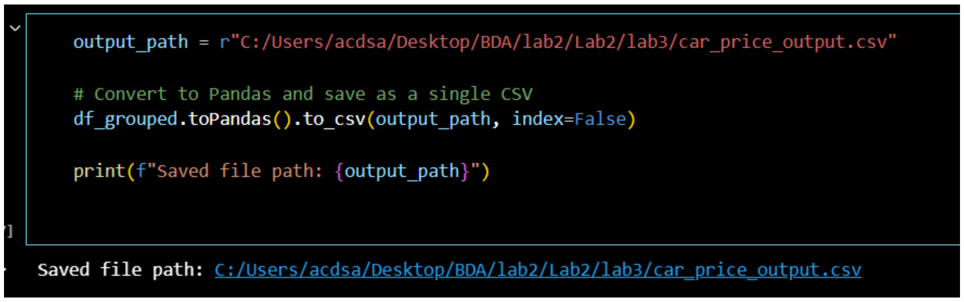
```
df_filtered = df_cleaned.filter(col("price") > 5000)

print("Cleaned and Filtered Dataset:")
df_filtered.show(5)
```

4. The average price of cars was calculated based on the brand.



```
df_grouped = df_filtered.groupBy("brand").agg(avg("price").alias("Average_Price"))

print("Average Price by Brand:")
df_grouped.show()
```

5. The cleaned and aggregated dataset was saved as a CSV file.



```
output_path = r"C:/Users/acdsa/Desktop/BDA/lab2/Lab2/lab3/car_price_output.csv"

# Convert to Pandas and save as a single CSV
df_grouped.toPandas().to_csv(output_path, index=False)

print(f"Saved file path: {output_path}")
```

```
Saved file path: C:/Users/acdsa/Desktop/BDA/lab2/Lab2/lab3/car_price_output.csv
```

Results and Analysis

The final output consists of a cleaned dataset and an aggregated report showing the average price of cars by brand.

Challenges and Solutions

A Unicode error occurred due to incorrect handling of backslashes in the file path. Additionally, data type mismatches caused some numeric columns to be read as strings, leading to type conversion errors.

Conclusion

In this project, we successfully loaded, cleaned, transformed, and analyzed a car dataset using Apache Spark. The data was processed efficiently, and the final aggregated results provided insights into average car prices by brand. This demonstrates the power of PySpark in handling large-scale data processing tasks.