

Name: Perez, Junmar A.

Lab No. 4

Git Repo/ Colab Link: https://github.com/junmsr/Perez_Elec2.git

Date: 03/20/2025

Objective

The goal is to analyze car price data using Apache Spark. Tasks include loading, cleaning, filtering, querying, and saving data.

Introduction

Apache Spark is a tool for processing large data sets. This lab uses Spark to clean, filter, and analyze car price data. The results are saved in CSV, JSON, and TXT formats.

Methodology

1. A sample dataset was loaded into a Spark DataFrame from a CSV file.

```
df = spark.read.csv(input_path, header=True, inferSchema=True)
```

✓ 0.2s

2. The dataset was cleaned by selecting relevant columns, removing missing values, and converting the year column to an integer.

```
# preprocessing
df = df.select(
    col("Brand").alias("brand"),
    col("Model").alias("model"),
    col("Year").alias("year"),
    col("Price").alias("price"),
    col("Mileage").alias("mileage"),
    col("Fuel_Type").alias("fuel_type"),
    col("Transmission").alias("transmission")
).dropna()

df = df.withColumn("year", col("year").cast("int"))
```

✓ 0.0s

- Basic operations were performed, including filtering data to keep only car listings with a price above 5000.

```
# filtering
filtered_df = df.filter(col("price") > 5000)
✓ 0.0s
```

- SQL queries were executed on the dataset using PySpark SQL. The top five most frequent car brands were identified, and the average price per brand was computed using grouping and filtering.

```
# SQL Queries
df.createOrReplaceTempView("car_data")

# extract top 5 most Frequent Car Brands
top_brands = spark.sql("""
    SELECT brand, COUNT(*) as count
    FROM car_data
    GROUP BY brand
    ORDER BY count DESC
    LIMIT 5
""")
[11] ✓ 0.0s

# average price per brand
avg_price_per_brand = spark.sql("""
    SELECT brand, AVG(price) as avg_price
    FROM car_data
    GROUP BY brand
    ORDER BY avg_price DESC
""")
[12] ✓ 0.0s
```

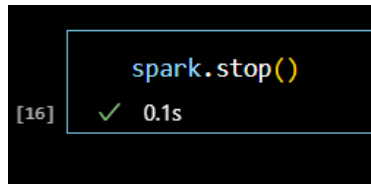
- The processed data was saved in various formats. The filtered dataset was stored as a CSV file, SQL query results were saved in JSON format, and a summary report was written to a TXT file and old output files were deleted before saving new ones to avoid duplication

```
# SQL Queries
df.createOrReplaceTempView("car_data")

# extract top 5 most Frequent Car Brands
top_brands = spark.sql("""
    SELECT brand, COUNT(*) as count
    FROM car_data
    GROUP BY brand
    ORDER BY count DESC
    LIMIT 5
""")
[11] ✓ 0.0s

# average price per brand
avg_price_per_brand = spark.sql("""
    SELECT brand, AVG(price) as avg_price
    FROM car_data
    GROUP BY brand
    ORDER BY avg_price DESC
""")
[12] ✓ 0.0s
```

6. The Spark session was stopped after processing to free up resources.



```
[16]: spark.stop()
✓ 0.1s
```

Results and Analysis

The filtered dataset was saved as a CSV file. The top five most frequent car brands were identified and stored in a JSON file. The summary report included the most frequent brands and their average prices.

Challenges and Solutions

A PySpark Java error occurred due to incorrect Java settings. This was fixed by setting JAVA_HOME to a JDK instead of a JRE. A file deletion error happened because some files were open in another program, which was resolved by closing them before deletion. Processing was initially slow, but filtering the data early improved performance.

Conclusion

This lab demonstrated how to clean and analyze data using Apache Spark. Key takeaways included cleaning data, executing SQL queries, exporting results, and troubleshooting common errors. These skills are essential for handling large datasets efficiently.