Dave Gabriel Jazmin
Junmar Perez
Aaron Bonaobra

LAB 3 DOCUMENTATION

1.1 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.

1.2 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.

2.1 Dataset Information

The dataset used in this project contains information about car prices and specifications, including brand, model, year, engine size, fuel type, transmission, mileage, doors, owner count, and price.

3.1 Loading the Dataset

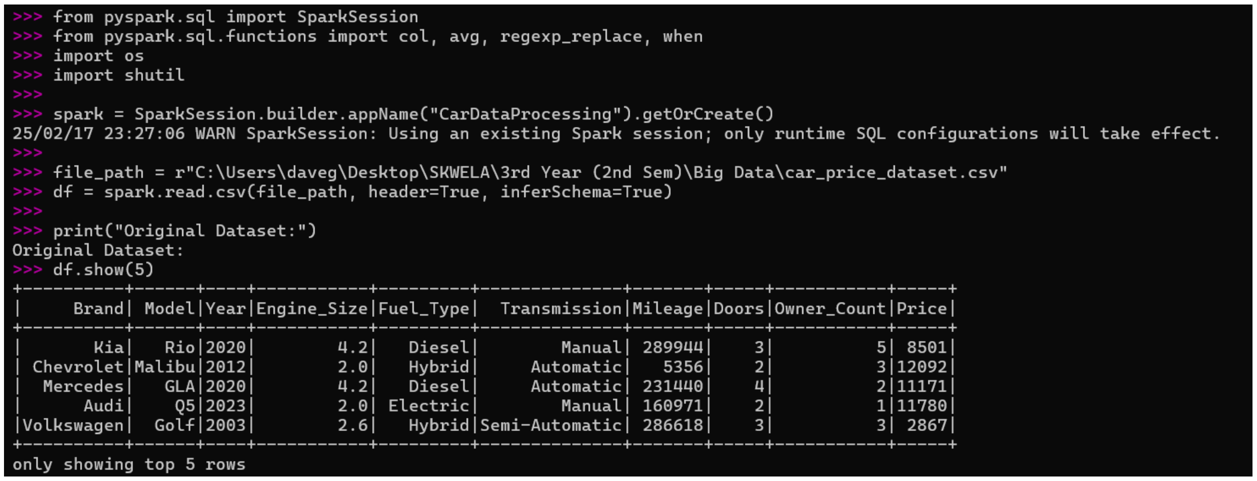
The dataset was loaded into a Spark DataFrame using the following command:

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

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

file_path = "C:\\Users\\daveg\\Desktop\\SKWELA\\3rd Year (2nd Sem)\\Big Data\\car_price_dataset.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

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



```
>>> from pyspark.sql import SparkSession
>>> from pyspark.sql.functions import col, avg, regexp_replace, when
>>> import os
>>> import shutil
>>>
>>> spark = SparkSession.builder.appName("CarDataProcessing").getOrCreate()
25/02/17 23:27:06 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>>
>>> file_path = r"C:\Users\daveg\Desktop\SKWELA\3rd Year (2nd Sem)\Big Data\car_price_dataset.csv"
>>> df = spark.read.csv(file_path, header=True, inferSchema=True)
>>>
>>> print("Original Dataset:")
Original Dataset:
>>> df.show(5)
+----------+------+----+-----------+---------+--------------+-------+-----+-----------+-----+
|     Brand| Model|Year|Engine_Size|Fuel_Type|  Transmission|Mileage|Doors|Owner_Count|Price|
+----------+------+----+-----------+---------+--------------+-------+-----+-----------+-----+
|       Kia|   Rio|2020|        4.2|   Diesel|        Manual| 289944|    3|          5| 8501|
| Chevrolet|Malibu|2012|        2.0|   Hybrid|     Automatic|   5356|    2|          3|12092|
|  Mercedes|   GLA|2020|        4.2|   Diesel|     Automatic| 231440|    4|          2|11171|
|      Audi|    Q5|2023|        2.0| Electric|        Manual| 160971|    2|          1|11780|
|Volkswagen|  Golf|2003|        2.6|   Hybrid|Semi-Automatic| 286618|    3|          3| 2867|
+----------+------+----+-----------+---------+--------------+-------+-----+-----------+-----+
only showing top 5 rows
```

3.2 Data Cleaning

Several preprocessing steps were performed:

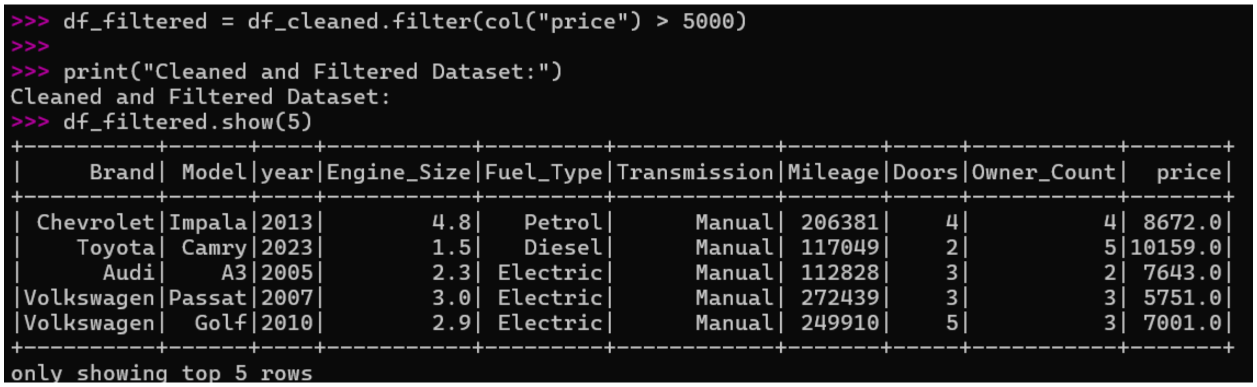
- Removing duplicate records
- Handling missing and incorrect values
- Converting data types for consistency

13. df_cleaned = df.dropDuplicates()
14. df_cleaned = df_cleaned.withColumn("price", regexp_replace(col("price"), "[^0-9.]", ""))
15. df_cleaned = df_cleaned.withColumn("year", regexp_replace(col("year"), "[^0-9]", ""))
16. df_cleaned = df_cleaned.withColumn("price", when(col("price") == "", "0").otherwise(col("price")))
17. df_cleaned = df_cleaned.withColumn("year", when(col("year") == "", "0").otherwise(col("year")))
18.
19. df_cleaned = df_cleaned.fillna({"price": "0", "brand": "Unknown", "year": "0"})
20. df_cleaned = df_cleaned.withColumn("price", col("price").cast("float"))
21. df_cleaned = df_cleaned.withColumn("year", col("year").cast("int"))

3.3 Filtering Data

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

22. df_filtered = df_cleaned.filter(col("price") > 5000)
23. print("Cleaned and Filtered Dataset:")
24. df_filtered.show(5)



```
>>> df_filtered = df_cleaned.filter(col("price") > 5000)
>>>
>>> print("Cleaned and Filtered Dataset:")
Cleaned and Filtered Dataset:
>>> df_filtered.show(5)
+----------+------+----+-----------+---------+------------+-------+-----+-----------+-------+
|     Brand| Model|year|Engine_Size|Fuel_Type|Transmission|Mileage|Doors|Owner_Count|  price|
+----------+------+----+-----------+---------+------------+-------+-----+-----------+-------+
| Chevrolet|Impala|2013|        4.8|   Petrol|      Manual| 206381|    4|          4| 8672.0|
|    Toyota| Camry|2023|        1.5|   Diesel|      Manual| 117049|    2|          5|10159.0|
|      Audi|    A3|2005|        2.3| Electric|      Manual| 112828|    3|          2| 7643.0|
|Volkswagen|Passat|2007|        3.0| Electric|      Manual| 272439|    3|          3| 5751.0|
|Volkswagen|  Golf|2010|        2.9| Electric|      Manual| 249910|    5|          3| 7001.0|
+----------+------+----+-----------+---------+------------+-------+-----+-----------+-------+
only showing top 5 rows
```

3.4 Aggregation

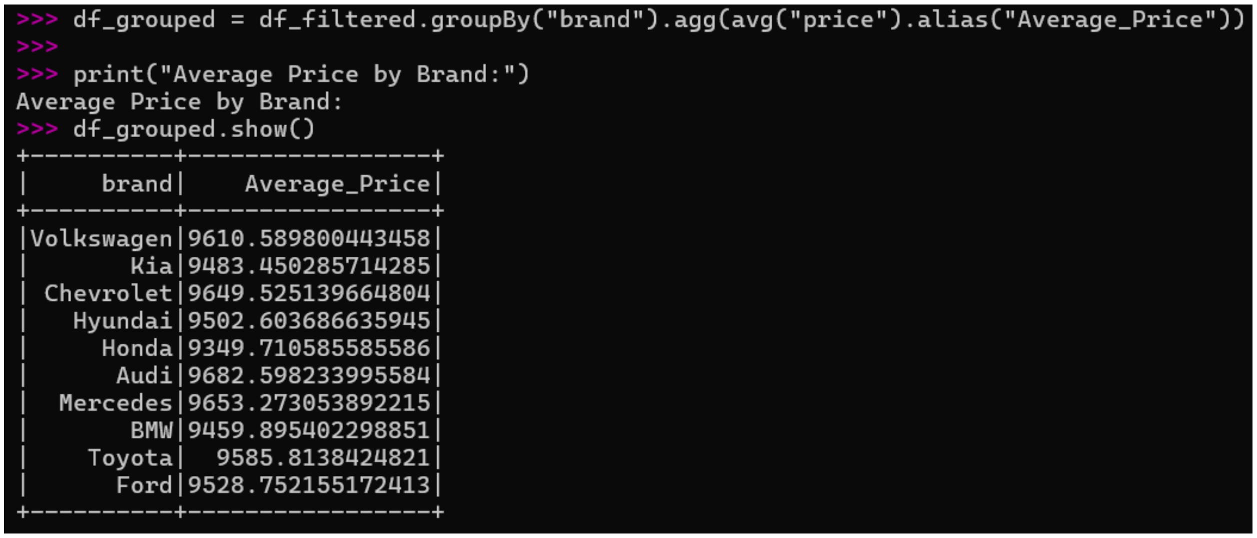
The average price of cars was calculated based on the brand.

25. df_grouped = df_filtered.groupBy("brand").agg(avg("price").alias("Average_Price"))
26. print("Average Price by Brand:")
27. df_grouped.show()

3.5 Saving the Processed Data

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

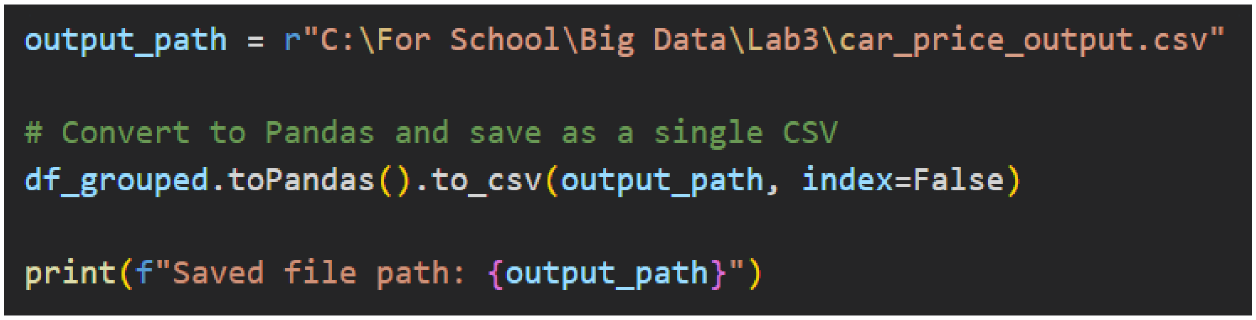
28. output_path = "C:\\Users\\daveg\\Desktop\\SKWELA\\3rd Year (2nd Sem)\\Big Data\\car_price_dataset"
29. df_grouped.write.csv(output_path, header=True)
30. print(f"Saved file path: {output_path}")



```
>>> df_grouped = df_filtered.groupBy("brand").agg(avg("price").alias("Average_Price"))
>>>
>>> print("Average Price by Brand:")
Average Price by Brand:
>>> df_grouped.show()
+----------+-----------------+
|     brand|    Average_Price|
+----------+-----------------+
|Volkswagen|9610.589800443458|
|       Kia|9483.450285714285|
| Chevrolet|9649.525139664804|
|   Hyundai|9502.603686635945|
|     Honda|9349.710585585586|
|      Audi|9682.598233995584|
|  Mercedes|9653.273053892215|
|       BMW|9459.895402298851|
|    Toyota|  9585.8138424821|
|      Ford|9528.752155172413|
+----------+-----------------+
```

4. Results

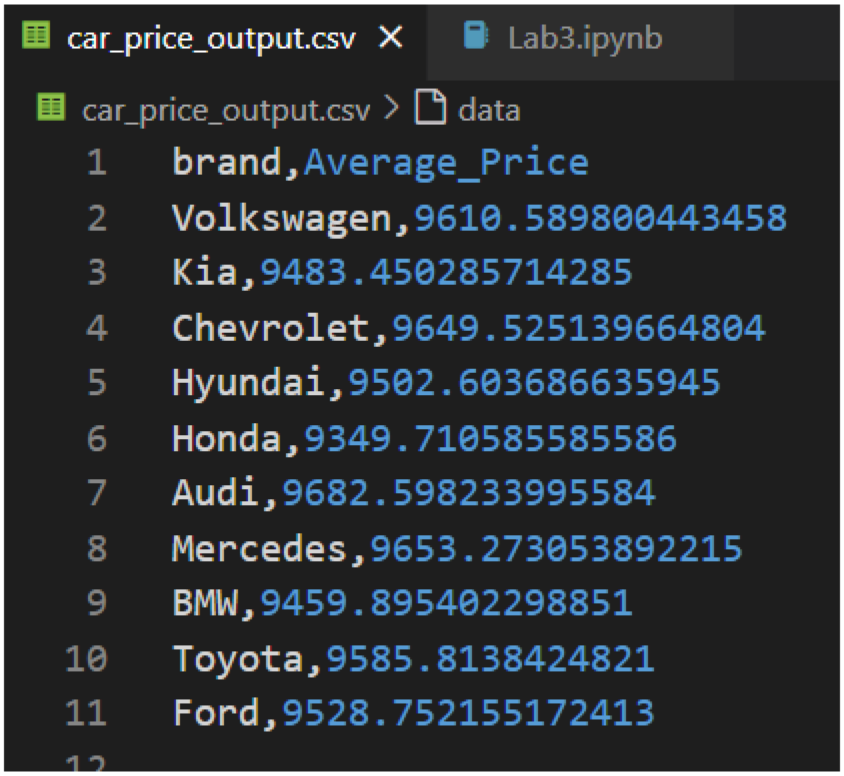
The final output consists of a cleaned dataset and an aggregated report showing the average price of cars by brand.



```
output_path = r"C:\For School\Big Data\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}")
```



```
brand,Average_Price
Volkswagen,9610.589800443458
Kia,9483.450285714285
Chevrolet,9649.525139664804
Hyundai,9502.603686635945
Honda,9349.710585585586
Audi,9682.598233995584
Mercedes,9653.273053892215
BMW,9459.895402298851
Toyota,9585.8138424821
Ford,9528.752155172413
```

5. Challenges & Solutions

Challenges:

1. **Unicode Error in File Path** - The error `unicodeescape` was encountered due to incorrect handling of backslashes in the file path.
2. **Data Type Mismatches** - Some numeric columns were read as strings, leading to type conversion errors.

6. 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.