Department of Computer Engineering
College of Engineering
Polytechnic University of the Philippines Sta. Mesa

CMPE 40163: Big Data using PySpark

Data Manipulation of Formula 1 Season 2023 Results

Submitted by:

Balangao, Samantha, A.
BS Computer Engineering, Year 3 and Section 2

Submitted to:

EDCEL B. ARTIFICIO

I. Introduction

Formula 1 (F1) racing is one of the most high-profile and popular motorsport events globally, as it showcases the pinnacle of the different automotive engineering, strategic plans, as well as highlights their drivers' skills in order to determine which constructor manufactures the best car. The Season 2023 of Formula 1 consists of multiple races driven in a variety of circuits. Each race provides big data: the teams, the drivers, the races/circuits/countries, the lap times, pit stop strategies, and especially the race outcomes amongst all other data. The race outcomes provide valuable insights to the respective teams' overall performance and racing dynamics.

The motorsport industry has been using data analytics to make data-driven decisions to optimize their performances, considering their highly-competitive environment where even milliseconds can determine their position in the championship. The main objective of this activity is to observe, analyze, and transform the Formula 1 2023 results in order to identify the factors that are fundamental to the race outcomes. By manipulating their race data, teams will have ease knowing which part of their performance needs improvement; such as vehicle design, drivers' partnership dynamics and their individual performances. Thus, being able to manipulate/transform and analyze racing data is essential for them to gain a competitive edge in the constructors championship.

II. Data Dictionary

The dataset I put into use is the Formula 1 Season 2023 Results which consisted of 19 variables and 440 observations for the whole season. However, this activity of data manipulation led me to only using the necessary variables and observations in the dataset. The table below lists all the variables I have utilized in this exercise.

Variable	Data Type	Value	Definition
Broadcast Name	-	M Verstappen, S Perez, C Leclerc, C Sainz, L Hamilton, G Russel, L Norris, O Piastri, F Alonso, L Stroll, P Gasly, E Ocon, V Bottas, G Zhou, N Hulkenberg, K Magnussen, Y Tsunoda, N de Vries, D Ricciardo, L Lawson, A Albon, L Sargeant	This is used for television broadcasts of each race; widely used in the aftermaths of the race.
Team Name	-	RED BULL RACING, FERRARI, MERCEDES, MCLAREN, ASTON MARTIN, ALPINE, ALFA ROMEO, HAAS F1 TEAM, ALPHATAURI,	The constructor teams involved in the F1 Grand Prix events; the constructors.

		WILLIAMS RACING	
Position	Categorical	1-20	The ‘position’ variable displays the finishing positions of the drivers.
Status	Categorical	"DNF", "Retired", "Accident", "Engine", "Mechanical Failure", "+1 Lap", "+2 Laps", "Collision damage", "Brakes", "Oil Leak", "Undertray", "Steering", "Technical", "Gearbox", "Withdrew", "Overheating", "Disqualified"	The ‘status’ column states the race status that the drivers (and teams) were put into during the season.
Points	Continuous	25-0	These are the points that are given to every winning driver. {25 - P1, 18 -P2, 15 -P3, 12-P4, 10-P5, 8-P6, 6-P7, 4-P8, 2-P9, 1-P10}. The succeeding positions gain 0 points.
Event Name	-	{Bahrain, Saudi Arabian, Australian, Azerbaijan, Miami, Monaco, Spanish, Canadian, Austrian, British, Hungarian, Belgian, Dutch, Italian, Singapore, Japanese, Qatar, United States, Mexico, São Paulo, Las Vegas, Abu Dhabi	This column states the Grand Prix events that happened during the 2023 season.

III. Data Manipulation using an RDD

For Data Manipulation using Resilient Distributed Dataset (RDD), I mainly utilized map(), filter(), groupByKey(), and sortByKey() as methods in order to manipulate the data to sort the variables and observation by team and drivers . In this way, we can see which teams and drivers participated in the Monaco Grand Prix. First and foremost, it is necessary to import the Pyspark module and the SparkContext package to be able to use the functions. Afterwards we create the SparkContext (sc).

In order to read the dataset, we need to import the csv file (in my case, the Results.csv) and read it into an RDD file by using sc.textFile(). Then I created 19 partitions for parallel processing and made indexing easier. Then I took the first 40 values to return.

```
[ ]  !pip install pyspark py4j
```

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
F1_RDD = sc.textFile('Results.csv', minPartitions=19)
F1_RDD.take(40)
```

Image 1 and 2: Pre-RDD Manipulation (Module, Reading as textFile, partitioning, and taking the forty values)

```
['DriverNumber,BroadcastName,Abbreviation,TeamName,TeamColor,FirstName,LastName,FullName,Position,GridPosition,Q1,Q2,Q3,Time,Status,Points,RoundNumber',
'1,M VERSTAPPEN,VER,Red Bull Racing,3671C6,Max,Verstappen,Max Verstappen,1,1,,,,0 days 01:33:56.736000,Finished,25,1,Sunday Race,Bahrain Grand Prix',
'11,S PEREZ,PER,Red Bull Racing,3671C6,Sergio,Perez,Sergio Perez,2,2,,,,0 days 01:34:08.723000,Finished,18,1,Sunday Race,Bahrain Grand Prix',
'14,F ALONSO,ALO,Aston Martin,358C75,Fernando,Alonso,Fernando Alonso,3,5,,,,0 days 01:34:35.373000,Finished,15,1,Sunday Race,Bahrain Grand Prix',
'55,C SAINZ,SAI,Ferrari,F91536,Carlos,Sainz,Carlos Sainz,4,4,,,,0 days 01:34:44.788000,Finished,12,1,Sunday Race,Bahrain Grand Prix',
'44,L HAMILTON,HAM,Mercedes,6CD3BF,Lewis,Hamilton,Lewis Hamilton,5,7,,,,0 days 01:34:47.713000,Finished,10,1,Sunday Race,Bahrain Grand Prix',
'18,L STROLL,STR,Aston Martin,358C75,Lance,Stroll,Lance Stroll,6,8,,,,0 days 01:34:51.238000,Finished,8,1,Sunday Race,Bahrain Grand Prix',
'63,G RUSSELL,RUS,Mercedes,6CD3BF,George,Russell,George Russell,7,6,,,,0 days 01:34:52.609000,Finished,6,1,Sunday Race,Bahrain Grand Prix',
'77,V BOTTAS,BOT,Alfa Romeo,C92D4B,Valtteri,Bottas,Valtteri Bottas,8,12,,,,0 days 01:35:09.383000,Finished,4,1,Sunday Race,Bahrain Grand Prix',
'10,P GASLY,GAS,Alpine,2293D1,Pierre,Gasly,Pierre Gasly,9,20,,,,0 days 01:35:10.489000,Finished,2,1,Sunday Race,Bahrain Grand Prix',
'23,A ALBON,ALB,Williams,37BEDD,Alexander,Albon,Alexander Albon,10,15,,,,0 days 01:35:26.510000,Finished,1,1,Sunday Race,Bahrain Grand Prix',
'22,Y TSUNODA,TSU,AlphaTauri,5E8FAA,Yuki,Tsunoda,Yuki Tsunoda,11,14,,,,0 days 01:35:27.606000,Finished,0,1,Sunday Race,Bahrain Grand Prix',
'2,L SARGEANT,SAR,Williams,37BEDD,Logan,Sargeant,Logan Sargeant,12,16,,,,,+1 Lap,0,1,Sunday Race,Bahrain Grand Prix',
'20,K MAGNUSSEN,MAG,Haas F1 Team,B6BABD,Kevin,Magnussen,Kevin Magnussen,13,17,,,,,+1 Lap,0,1,Sunday Race,Bahrain Grand Prix',
'21,N DE VRIES,DEV,AlphaTauri,5E8FAA,Nyck,De Vries,Nyck De Vries,14,19,,,,,+1 Lap,0,1,Sunday Race,Bahrain Grand Prix',
'27,N HULKENBERG,HUL,Haas F1 Team,B6BABD,Nico,Hulkenberg,Nico Hulkenberg,15,10,,,,,+1 Lap,0,1,Sunday Race,Bahrain Grand Prix',
'24,G ZHOU,ZHO,Alfa Romeo,C92D4B,Guanyu,Zhou,Guanyu Zhou,16,13,,,,,+1 Lap,0,1,Sunday Race,Bahrain Grand Prix',
'4,L NORRIS,NOR,McLaren,F58020,Lando,Norris,Lando Norris,17,11,,,,,+2 Laps,0,1,Sunday Race,Bahrain Grand Prix',
'31,E OCON,OCO,Alpine,2293D1,Esteban,Ocon,Esteban Ocon,18,9,,,,,Mechanical,0,1,Sunday Race,Bahrain Grand Prix',
'16,C LECLERC,LEC,Ferrari,F91536,Charles,Leclerc,Charles Leclerc,19,3,,,,,Engine,0,1,Sunday Race,Bahrain Grand Prix',
'81,O PIASTRI,PIA,McLaren,F58020,Oscar,Piastri,Oscar Piastri,20,18,,,,,Electrical,0,1,Sunday Race,Bahrain Grand Prix',
'11,S PEREZ,PER,Red Bull Racing,3671C6,Sergio,Perez,Sergio Perez,1,1,,,,0 days 01:21:14.894000,Finished,25,2,Sunday Race,Saudi Arabian Grand Prix',
'1,M VERSTAPPEN,VER,Red Bull Racing,3671C6,Max,Verstappen,Max Verstappen,2,15,,,,0 days 01:21:20.249000,Finished,19,2,Sunday Race,Saudi Arabian Grand Prix',
'14,F ALONSO,ALO,Aston Martin,358C75,Fernando,Alonso,Fernando Alonso,3,2,,,,0 days 01:21:35.622000,Finished,15,2,Sunday Race,Saudi Arabian Grand Prix',
'63,G RUSSELL,RUS,Mercedes,6CD3BF,George,Russell,George Russell,4,3,,,,0 days 01:21:40.760000,Finished,12,2,Sunday Race,Saudi Arabian Grand Prix',
'44,L HAMILTON,HAM,Mercedes,6CD3BF,Lewis,Hamilton,Lewis Hamilton,5,7,,,,0 days 01:21:45.959000,Finished,10,2,Sunday Race,Saudi Arabian Grand Prix',
'55,C SAINZ,SAI,Ferrari,F91536,Carlos,Sainz,Carlos Sainz,6,4,,,,0 days 01:21:50.770000,Finished,8,2,Sunday Race,Saudi Arabian Grand Prix',
'16,C LECLERC,LEC,Ferrari,F91536,Charles,Leclerc,Charles Leclerc,7,12,,,,0 days 01:21:58.056000,Finished,6,2,Sunday Race,Saudi Arabian Grand Prix',
'31,E OCON,OCO,Alpine,2293D1,Esteban,Ocon,Esteban Ocon,8,6,,,,0 days 01:22:07.726000,Finished,4,2,Sunday Race,Saudi Arabian Grand Prix',
'10,P GASLY,GAS,Alpine,2293D1,Pierre,Gasly,Pierre Gasly,9,9,,,,0 days 01:22:09.641000,Finished,2,2,Sunday Race,Saudi Arabian Grand Prix',
'20,K MAGNUSSEN,MAG,Haas F1 Team,B6BABD,Kevin,Magnussen,Kevin Magnussen,10,13,,,,0 days 01:22:19.720000,Finished,1,2,Sunday Race,Saudi Arabian Grand Prix',
```

Image 3: Pre-RDD transformation code output using take(40).

Note: This image was cropped so that the output can still be visible; it did return 40 values.

Then we started using **map()** which allows us to create a new RDD by transforming each element of the former RDD using the lambda function, and thus, we can create tuples. In my activity, I created a tuple of three elements:

- x.split(‘ , ’)[3]: represents the constructor (from TeamName)

- x.split(‘ , ’)[1]: Represents the driver’s name (from BroadcastName)
- x.split(‘ , ‘)[18]: Represents the Grand Prix event name (from EventName)

The **filter()** method was also used to retain only the observations necessary for Monaco Grand Prix, this ensures that we only see the teams and drivers who participated in the specific event (there were instances that the teams/drivers were disqualified or changed due to poor performance). Then again using the **map()** function to create a new RDD, F1_Monaco that contains tuples of the driver and team names. Then, **groupByKey()** was utilized to group the new RDD by the key (team names) and each value is a list of drivers in the team. We then sort the teams alphabetically for ease of reading by **sortByKey().** Ultimately, using **collect(),** we retrieve all the sorted results from the latest RDD and print them using a for-loop, which iterates over the collected results, unpacking each tuple into team and drivers then printing the output.

```
#Balangao, Samantha A.
#Data Manipulation using an RDD
#post-RDD transformation
from pyspark import SparkContext

sc = SparkContext.getOrCreate()

F1_RDD = sc.textFile('Results.csv', minPartitions=19)

#Using map(), filter()
F1COL = F1_RDD.map(lambda x: (x.split(',')[3],x.split(',')[1], x.split(',')[18]))
F1TEAMS = F1COL.filter(lambda x:x[2] == 'Monaco Grand Prix')
F1_Monaco = F1TEAMS.map(lambda x: (x[0], x[1]))

#Using groupbyKey() and sortByKey()
F1Grouped = F1_Monaco.groupByKey()
F1Sort = F1Grouped.sortByKey(ascending = True)

for team, drivers in F1Sort.collect():
  print(f"Team: {team}, Drivers: {list(drivers)}")
```

```
Team: Alfa Romeo, Drivers: ['V BOTTAS', 'G ZHOU']
Team: AlphaTauri, Drivers: ['N DE VRIES', 'Y TSUNODA']
Team: Alpine, Drivers: ['E OCON', 'P GASLY']
Team: Aston Martin, Drivers: ['F ALONSO', 'L STROLL']
Team: Ferrari, Drivers: ['C LECLERC', 'C SAINZ']
Team: Haas F1 Team, Drivers: ['N HULKENBERG', 'K MAGNUSSEN']
Team: McLaren, Drivers: ['L NORRIS', 'O PIASTRI']
Team: Mercedes, Drivers: ['L HAMILTON', 'G RUSSELL']
Team: Red Bull Racing, Drivers: ['M VERSTAPPEN', 'S PEREZ']
Team: Williams, Drivers: ['A ALBON', 'L SARGEANT']
```

Image 4: Post-RDD transformation code using map(), filter(), groupByKey() and sortByKey()
Note: This includes the output showing the ten teams involved in F1; sorted in their teams and drivers.

This activity, Data Manipulation using Resilient Distributed Dataset (RDD) provided one to have an experience and learn how to handle datasets in a distributed environment. Through the use of RDD transformations like map(), filter(), groupByKey(), and sortByKey(), I have learned that data can be efficiently processed through a variety of ways, this being in a direct manner. In my view, RDDs can be less efficient and need to be more ‘exact’ when it comes to data transformation, it is optimal for low-latency operations in a distributed computing environment. They can be more complex and difficult compared to dataframes, especially when it comes to having to visualize the data. By manipulating through an RDD, we can still picture the data we want but not as modeled or ‘structured’ unlike if we manipulated it as a dataframe.

Having Formula 1 2023 Results as my dataset is a challenge as you have to consider many factors such as: teams, drivers, wins, lap times, car performance, overall performance, and other more internal and external components within the events. But in this part of the laboratory exercise, I simply narrowed

down the teams and drivers who participated in the Monaco Grand Prix to view the output for data manipulation as an RDD.

IV. Data Manipulation using a Dataframe

In data manipulation using dataframe, we can visualize the data that we want better due to its tabular form. In the activity, I imported the required library, SparkSession from pyspark.sql, which serves as the entry point for us to use DataFrames in Pyspark; then, creating a spark session using the **getOrCreate()** method, with the application name as "Formula 1". By initializing the spark session, we are now able to read the data and perform the operations.

We may now read our file into a dataframe using spark.read.csv; in my case, I created a new dataframe (F1_df) to read the file. We can see that in F1_Df, we had "header = True" which indicates that the first row contains the name of the columns (DriverNumber, BroadcastName, TeamName, etc.), and "InferSchema = True" deduces/concludes the data types of the columns.

Then we start using the method **select()** in a new dataframe (df_F1results) so that in our table, we include only the specific columns relevant to our analysis. Then by using **withColumnRenamed()**, we rename the column for better titles (replacing BroadcastName to Driver; TeamName to Constructor; EventName to Grand Prix); we store it in another dataframe called df_F1results1, and then we display the output.

```
#Balangao, Samantha A.
#Data Manipulation using a DataFrame
#pre-dataframe manipulation
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Formula1").getOrCreate()

F1_df = spark.read.csv("Results.csv", header=True, inferSchema=True)
F1_df.show()
```

Image 5: Pre-DataFrame Manipulation

```
+------------+-------------+------------+--------------+---------+---------+----------+---------------+--------+------------+----+----+----+--------------------+
|DriverNumber|BroadcastName|Abbreviation|      TeamName|TeamColor|FirstName|  LastName|       FullName|Position|GridPosition|  Q1|  Q2|  Q3|                Time|
+------------+-------------+------------+--------------+---------+---------+----------+---------------+--------+------------+----+----+----+--------------------+
|           1| M VERSTAPPEN|         VER|Red Bull Racing|   3671C6|      Max|Verstappen| Max Verstappen|       1|           1|NULL|NULL|NULL|0 days 01:33:56.7...|
|          11|      S PEREZ|         PER|Red Bull Racing|   3671C6|   Sergio|     Perez|    Sergio Perez|       2|           2|NULL|NULL|NULL|0 days 01:34:08.7...|
|          14|     F ALONSO|         ALO|  Aston Martin|   358C75| Fernando|    Alonso|Fernando Alonso|       3|           5|NULL|NULL|NULL|0 days 01:34:35.3...|
|          55|      C SAINZ|         SAI|       Ferrari|   F91536|   Carlos|     Sainz|    Carlos Sainz|       4|           4|NULL|NULL|NULL|0 days 01:34:44.7...|
|          44|   L HAMILTON|         HAM|      Mercedes|   6CD3BF|    Lewis|  Hamilton| Lewis Hamilton|       5|           7|NULL|NULL|NULL|0 days 01:34:47.7...|
|          18|     L STROLL|         STR|  Aston Martin|   358C75|    Lance|    Stroll|   Lance Stroll|       6|           8|NULL|NULL|NULL|0 days 01:34:51.2...|
|          63|   G RUSSELL|         RUS|      Mercedes|   6CD3BF|   George|   Russell| George Russell|       7|           6|NULL|NULL|NULL|0 days 01:34:52.6...|
|          77|     V BOTTAS|         BOT|    Alfa Romeo|   C92D48| Valtteri|    Bottas|Valtteri Bottas|       8|          12|NULL|NULL|NULL|0 days 01:35:09.3...|
|          10|      P GASLY|         GAS|        Alpine|   2293D1|   Pierre|     Gasly|    Pierre Gasly|       9|          20|NULL|NULL|NULL|0 days 01:35:10.4...|
|          23|      A ALBON|         ALB|      Williams|   37BEDD| Alexander|     Albon|Alexander Albon|      10|          15|NULL|NULL|NULL|0 days 01:35:26.5...|
|          22|    Y TSUNODA|         TSU|     AlphaTauri|   5E8FAA|     Yuki|   Tsunoda|   Yuki Tsunoda|      11|          14|NULL|NULL|NULL|0 days 01:35:27.6...|
|           2|   L SARGEANT|         SAR|      Williams|   37BEDD|    Logan|  Sargeant| Logan Sargeant|      12|          16|NULL|NULL|NULL|                NULL|
|          20|  K MAGNUSSEN|         MAG|   Haas F1 Team|   B6BABD|    Kevin| Magnussen|Kevin Magnussen|      13|          17|NULL|NULL|NULL|                NULL|
|          21|   N DE VRIES|         DEV|     AlphaTauri|   5E8FAA|     Nyck|  De Vries|  Nyck De Vries|      14|          19|NULL|NULL|NULL|                NULL|
|          27| N HULKENBERG|         HUL|   Haas F1 Team|   B6BABD|     Nico|Hulkenberg|Nico Hulkenberg|      15|          10|NULL|NULL|NULL|                NULL|
|          24|       G ZHOU|         ZHO|    Alfa Romeo|   C92D48|   Guanyu|      Zhou|    Guanyu Zhou|      16|          13|NULL|NULL|NULL|                NULL|
|           4|     L NORRIS|         NOR|       McLaren|   F58020|    Lando|    Norris|   Lando Norris|      17|          11|NULL|NULL|NULL|                NULL|
|          31|       E OCON|         OCO|        Alpine|   2293D1|  Esteban|      Ocon|    Esteban Ocon|      18|           9|NULL|NULL|NULL|                NULL|
|          16|    C LECLERC|         LEC|       Ferrari|   F91536|  Charles|   Leclerc|Charles Leclerc|      19|           3|NULL|NULL|NULL|                NULL|
|          81|    O PIASTRI|         PIA|       McLaren|   F58020|    Oscar|   Piastri|  Oscar Piastri|      20|          18|NULL|NULL|NULL|                NULL|
+------------+-------------+------------+--------------+---------+---------+----------+---------------+--------+------------+----+----+----+--------------------+
only showing top 20 rows
```

Status	Points	RoundNumber	Race Type	EventName
Finished	25	1	Sunday Race	Bahrain Grand Prix
Finished	18	1	Sunday Race	Bahrain Grand Prix
Finished	15	1	Sunday Race	Bahrain Grand Prix
Finished	12	1	Sunday Race	Bahrain Grand Prix
Finished	10	1	Sunday Race	Bahrain Grand Prix
Finished	8	1	Sunday Race	Bahrain Grand Prix
Finished	6	1	Sunday Race	Bahrain Grand Prix
Finished	4	1	Sunday Race	Bahrain Grand Prix
Finished	2	1	Sunday Race	Bahrain Grand Prix
Finished	1	1	Sunday Race	Bahrain Grand Prix
Finished	0	1	Sunday Race	Bahrain Grand Prix
+1 Lap	0	1	Sunday Race	Bahrain Grand Prix
+1 Lap	0	1	Sunday Race	Bahrain Grand Prix
+1 Lap	0	1	Sunday Race	Bahrain Grand Prix
+1 Lap	0	1	Sunday Race	Bahrain Grand Prix
+1 Lap	0	1	Sunday Race	Bahrain Grand Prix
+2 Laps	0	1	Sunday Race	Bahrain Grand Prix
Mechanical	0	1	Sunday Race	Bahrain Grand Prix
Engine	0	1	Sunday Race	Bahrain Grand Prix
Electrical	0	1	Sunday Race	Bahrain Grand Prix

Images 6&7: Pre-DataFrame Manipulation output

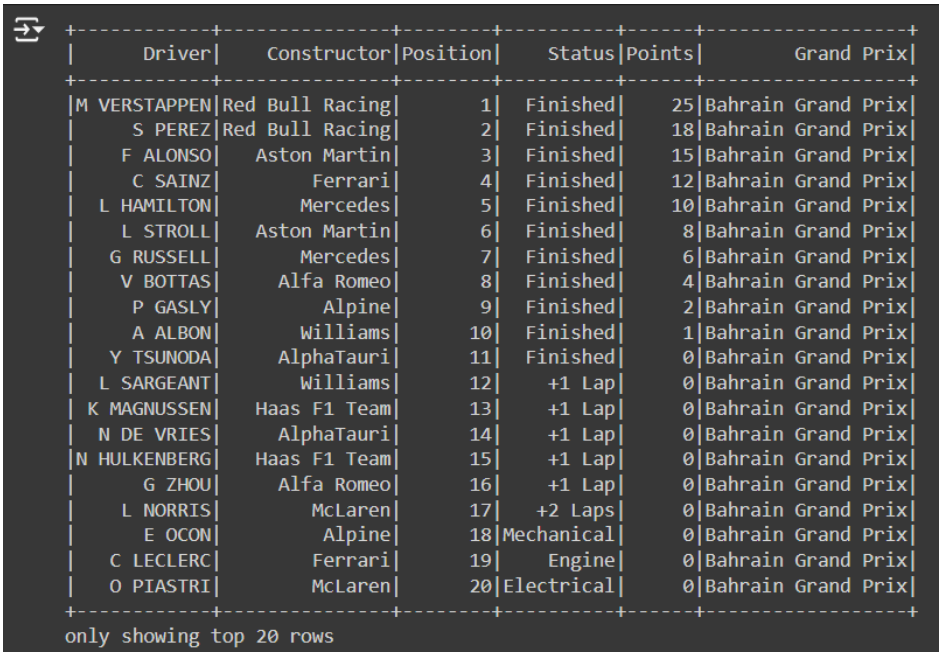
```python
# Balangao, Samantha A.
# Data Manipulation using a DataFrame
#post-dataframe manipulation

from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Formula1").getOrCreate()
F1_df = spark.read.csv("Results.csv", header=True, inferSchema=True)

#Using select()
df_F1results = F1_df.select("BroadcastName", "TeamName", "Position", "Status", "Points", "EventName")

#Using withColumnRenamed()
df_F1results1 = df_F1results.withColumnRenamed("BroadcastName", "Driver") \
                            .withColumnRenamed("TeamName", "Constructor") \
                            .withColumnRenamed("EventName", "Grand Prix")
df_F1results1.show()
```

Image 8: Post-DataFrame with .select() and .withColumnRenamed() methods



```
+------------+---------------+--------+----------+------+-----------------+
|      Driver|    Constructor|Position|    Status|Points|        Grand Prix|
+------------+---------------+--------+----------+------+-----------------+
|M VERSTAPPEN|Red Bull Racing|       1|  Finished|    25|Bahrain Grand Prix|
|     S PEREZ|Red Bull Racing|       2|  Finished|    18|Bahrain Grand Prix|
|    F ALONSO|   Aston Martin|       3|  Finished|    15|Bahrain Grand Prix|
|     C SAINZ|        Ferrari|       4|  Finished|    12|Bahrain Grand Prix|
|  L HAMILTON|       Mercedes|       5|  Finished|    10|Bahrain Grand Prix|
|    L STROLL|   Aston Martin|       6|  Finished|     8|Bahrain Grand Prix|
|   G RUSSELL|       Mercedes|       7|  Finished|     6|Bahrain Grand Prix|
|    V BOTTAS|     Alfa Romeo|       8|  Finished|     4|Bahrain Grand Prix|
|     P GASLY|         Alpine|       9|  Finished|     2|Bahrain Grand Prix|
|     A ALBON|       Williams|      10|  Finished|     1|Bahrain Grand Prix|
|   Y TSUNODA|     AlphaTauri|      11|  Finished|     0|Bahrain Grand Prix|
|  L SARGEANT|       Williams|      12|    +1 Lap|     0|Bahrain Grand Prix|
| K MAGNUSSEN|   Haas F1 Team|      13|    +1 Lap|     0|Bahrain Grand Prix|
|  N DE VRIES|     AlphaTauri|      14|    +1 Lap|     0|Bahrain Grand Prix|
|N HULKENBERG|   Haas F1 Team|      15|    +1 Lap|     0|Bahrain Grand Prix|
|      G ZHOU|     Alfa Romeo|      16|    +1 Lap|     0|Bahrain Grand Prix|
|    L NORRIS|        McLaren|      17|   +2 Laps|     0|Bahrain Grand Prix|
|      E OCON|         Alpine|      18|Mechanical|     0|Bahrain Grand Prix|
|   C LECLERC|        Ferrari|      19|    Engine|     0|Bahrain Grand Prix|
|   O PIASTRI|        McLaren|      20|Electrical|     0|Bahrain Grand Prix|
+------------+---------------+--------+----------+------+-----------------+
only showing top 20 rows
```

Image 9: Post-DataFrame with .select() and .withColumnRenamed() output

Afterwards, I created three new dataframes (df_Position, df_Drvers,df_Struggles), and in order to filter the winners (positions that are equal to 1), and drivers with 25 or more points (typically in Formula 1, a maximum of 25 points is awarded for the winner, but in my dataset, there is a 26 point marked which could have been an error) I utilized the **filter()** method. The filtering method is an advantage for SQL querying. Moreso, I added another variable that lists the statuses, mainly more about the complications they encountered, then filtered it in the third dataframe mentioned above. In here there is an additional method, isin() which checks if the status column contains one of the predefined race status; this method helps in filtering specific categories in the data.

```
#Using filter
df_Position = df_F1results1.filter(df_F1results1["Position"] == 1)
df_Drivers = df_F1results1.filter(df_F1results1["Points"] >= 25)

race_statuses = ["DNF", "Retired", "Accident", "Engine", "Mechanical Failure", "+1 Lap", "+2 Laps",
                 "Collision damage", "Brakes", "Oil Leak", "Undertray", "Steering", "Technical",
                 "Gearbox", "Withdrew", "Overheating", "Disqualified"]
df_Struggles = df_F1results1.filter(df_F1results1["Status"].isin(race_statuses))
```

Image 10: Post-DataFrame Manipulation with .filter()

Then, I used the **groupBy()** to group the most recent dataframe in a new dataframe and then utilized **count()** and **withColumnRenamed()** for counting the total struggles and renamed the column. Afterwards, I registered the dataframes as temporary views for SQL querying with **createOrReplaceTempView()**;

```
#Group by 'Constructor' and count struggles
df_team_struggles = df_Struggles.groupBy("Constructor").count().withColumnRenamed("count", "Number of Struggles")

#Register DataFrames as temporary views for SQL querying
df_Position.createOrReplaceTempView("df_Position")
df_Drivers.createOrReplaceTempView("df_Drivers")
df_team_struggles.createOrReplaceTempView("df_team_struggles")
```

Image 11: Post-DataFrame Manipulation with groupBy(), count(), withColumnRenamed(); and registered dataframes

Then, as I analyzed my dataset, I wanted to see which teams and drivers performed the best and which teams underperformed for the whole season. In the image below, it shows the three questions I have reflected upon:

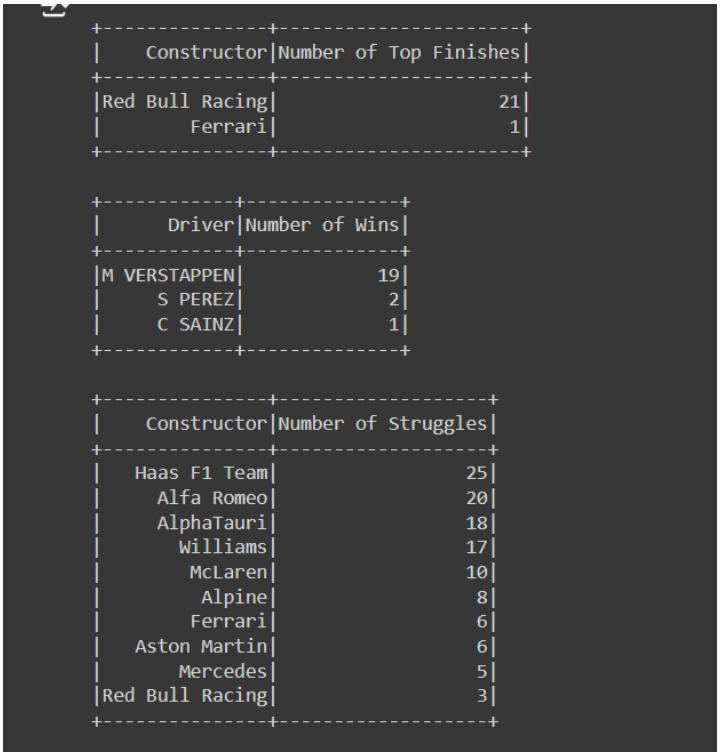
1. Which team performed the best in most races?
2. Which driver performed the best in most races?
3. What teams had the most struggles based on their status?

```
#SQL Query 1: Which team performed the best in most races? (COUNT top finishes per team)
spark.sql("""
    SELECT Constructor, COUNT(*) AS `Number of Top Finishes`
    FROM df_Position
    GROUP BY Constructor
    ORDER BY `Number of Top Finishes` DESC
""").show()

#SQL Query 2: Which driver performed the best in most races? (COUNT wins per driver)
spark.sql("""
    SELECT Driver, COUNT(*) AS `Number of Wins`
    FROM df_Drivers
    GROUP BY Driver
    ORDER BY `Number of Wins` DESC
""").show()

#SQL Query 3: What teams had the most struggles based on their 'status'?
spark.sql("""
    SELECT Constructor, `Number of Struggles`
    FROM df_team_struggles
    ORDER BY `Number of Struggles` DESC
""").show()
```

Image 12: Post-DataFrame Manipulation SQL Querying



```
+--------------+---------------------+
|   Constructor|Number of Top Finishes|
+--------------+---------------------+
|Red Bull Racing|                   21|
|       Ferrari|                    1|
+--------------+---------------------+

+-----------+-------------+
|     Driver|Number of Wins|
+-----------+-------------+
|M VERSTAPPEN|           19|
|    S PEREZ|            2|
|    C SAINZ|            1|
+-----------+-------------+

+--------------+-------------------+
|   Constructor|Number of Struggles|
+--------------+-------------------+
|  Haas F1 Team|                 25|
|    Alfa Romeo|                 20|
|    AlphaTauri|                 18|
|      Williams|                 17|
|       McLaren|                 10|
|        Alpine|                  8|
|       Ferrari|                  6|
|  Aston Martin|                  6|
|      Mercedes|                  5|
|Red Bull Racing|                  3|
+--------------+-------------------+
```

Image 13: Post-DataFrame Manipulation output.

In this part of the laboratory exercise, Data Manipulation using DataFrames easily portrays the dataset I imported. Like in manipulation as an RDD, it still efficiently handles large-scale datasets as it benefits from Spark module. However, in this case, data manipulation was more flexible, and user-friendly. Due to its tabular structure, DataFrames allowed me to have a clearer view of the transformation I was doing: renaming, filtering, grouping data can easily be displayed, hence I can track the changes I made.

Moreover, initializing DataFrames as temporary SQL views allowed me to perform complex queries in a more concise and understandable way compared to using RDD transformations. Because of this nature, it allowed me to perform analytical queries on my dataset, such as which teams/drivers performed the best or underperformed. By using SQL to manipulate the data frame, we can connect data analysis to automotive engineering, and possibly to any other fields one may wish to do data analytics on.

Especially in datasets that Formula 1 has, it is more beneficial for an analyst to visualize their data in a more optimal manner, so that they can make use of it for the upcoming races and seasons. This helps them provide the team with more structured data so that they know which decision should be made. Like I have mentioned above, "*Having Formula 1 2023 Results as my dataset is a challenge as you have to consider many factors such as: teams, drivers, wins, lap times, car performance, overall performance, and other more internal and external components within the events.*"; it is better to perform data manipulation for this type of dataset with data frames compared to RDDs.

V. Synthesis and Moving Forward

This activity has taught me that big data can be transformed and analyzed based on what the user wants. It can be manipulated in a variety of ways, such as a Resilient Distributed Dataset (RDD) or even as a DataFrame, and it also fundamental to learn how to structure it in order to provide insights or solutions to problems that may need data-driven or data-referencing approaches. I have also learned that it is not an easy task analyzing large chunks of data, especially when it concerns results that may affect the people around us. Just like in data manipulation of the 2023 results of Formula 1, if the data was not analyzed, manipulated, and presented properly, it may change the trajectory of a team's overall performance for the next event.

The plans I have for the knowledge I have gained in this activity involves improving my skills, and extending my knowledge of data manipulation. I think that I still have a lot to learn in the area of RDDs, since there are still some concepts that are confusing to manipulate my data into something that I wanted to see, just like in the concept of mapping. However, I think that the learnings I have acquired in this exercise can be a foundation and a practice for now. Understanding the concept of RDDs and dataframes while coding has been challenging yet when I am able to see the visualization I had (particularly dataframes), it gives the feeling of fulfillment. Hopefully in the near future, the foundation I gained in this exercise can help in

In conclusion, data manipulation can be quite challenging but yet a good experience and foundation in the future, especially in the industry. Knowing how to manipulate data may offer various opportunities in different fields one may take; whether in business, automotive, information technology, healthcare, and other more industries. It is versatile and fundamental, especially that data is more abundant nowadays due to technology.