F1 EDA: Hadoop Cluster with Hive: Big Data 2nd Term

Enrique Ulises Báez Gómez Tagle
Mauricio Iván Ascencio Martínez
Sara Rocío Miranda Mateos

April 19, 2024

Abstract

This project focuses on constructing a Hadoop cluster integrated with a Hive database to conduct exploratory data analysis (EDA) on Formula 1 dataset. Our objective is to leverage big data technologies to uncover insights into F1 races, drivers, and performance metrics. The document outlines the technological infrastructure setup, data analysis methodology, key findings, encountered challenges, and the solutions we implemented.

Contents

1 Introduction

In the fast-paced world of Formula 1 racing, where milliseconds can determine the winner, the application of big data technologies opens up new vistas for understanding and optimizing performance. This project embarks on an exploratory journey into the vast datasets of Formula 1 races, employing Hadoop and Hive to manage and analyze this data. The goal is not only to illustrate the capabilities of these technologies in handling large-scale data, but also to explore how big data can provide unprecedented insights into race strategies, driver efficiency, and team dynamics. Our exploration goes beyond traditional analytics, delving into the nuances of data-driven decision-making in sports. Through this document, we aim to share our journey from the initial setup of our big data infrastructure to the nuanced analysis that led us to compelling insights and the myriad challenges we navigated along the way.

2 Infrastructure / Architecture

The foundation of our data analysis project is a meticulously designed Hadoop cluster, augmented with Hive to facilitate structured data storage and sophisticated querying capabilities. Our infrastructure not only ensures the efficient handling of big data but also supports complex exploratory data analysis (EDA) processes. The architecture is strategically deployed on the Google Cloud Platform (GCP), leveraging its scalable compute and storage services. Below is a comprehensive outline of our deployment strategy:

- **Hadoop Cluster Configuration:** Central to our project is a Hadoop cluster, designed for scalability and resilience. We configured it with one master node to manage the cluster and multiple worker nodes to process the data. This setup allows for efficient data processing and analysis.
- **Hive Database Integration:** For querying capabilities, we integrated Hive with our Hadoop cluster. Hive facilitates querying and managing large datasets residing in distributed storage using SQL-like syntax. It is particularly advantageous for EDA, allowing us to query the data stored in the Hadoop Distributed File System (HDFS) with ease.
- **MySQL Cloud SQL Instance for Hive Metastore:** The metadata for Hive tables, such as schema information, is managed by a MySQL instance hosted on Cloud SQL. This setup ensures that the metadata is stored reliably and is accessible across the cluster, enabling efficient query processing.
- **Cloud Storage for Hive Warehouse:** The processed and raw data are stored in a Cloud Storage bucket designated as the Hive Warehouse. This choice of storage solution offers high availability and scalability, essential for managing the large datasets involved in our project.

- **Deployment Process:**
    1. We initiated our deployment by setting up the Google Cloud SDK, enabling us to interact with Google Cloud services seamlessly.
    2. A Cloud Storage bucket was created to host our Hive data, carefully chosen to be in proximity to our compute resources to minimize latency.
    3. We proceeded to establish a Cloud SQL instance for MySQL, tailored to serve as the Hive Metastore. This instance was configured with recommended settings to balance performance with cost efficiency.
    4. The creation of the Dataproc Cluster was a pivotal step, involving the activation of the Cloud Dataproc API and the configuration of the cluster to suit our processing needs. We opted for a standard cluster with a specific mix of master and worker nodes, ensuring adequate resources for our data processing tasks.
    5. Service accounts were configured with precise permissions to manage resources across Cloud SQL and Dataproc, streamlining the interaction between different components of our architecture.

This infrastructure serves as the backbone for our project, enabling us to leverage big data technologies effectively to extract meaningful insights from the Formula 1 dataset. The deployment emphasizes scalability, efficiency, and cost-effectiveness, tailored to meet the demands of processing and analyzing large-scale data.

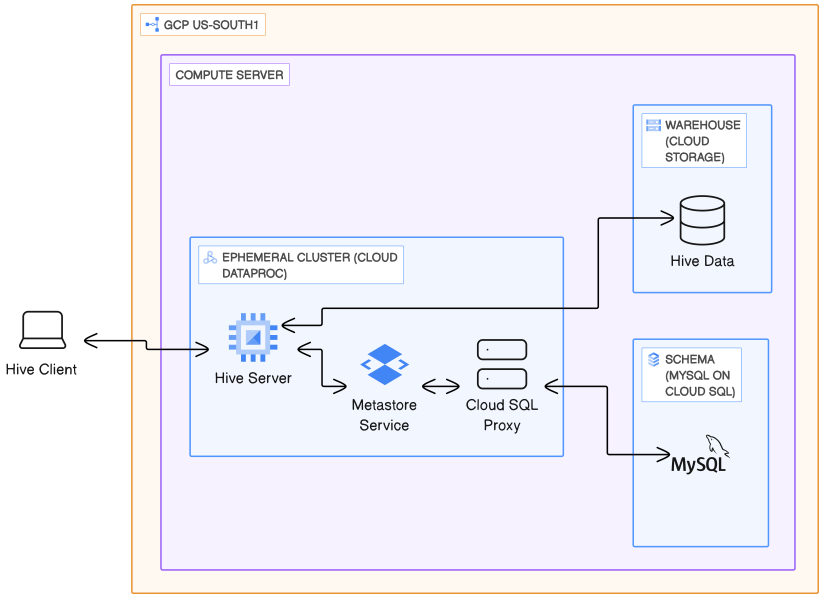


Figure 1: **Architecture Diagram**

3 Data Analysis with F1 Dataset

In our exploration of big data applications within Formula 1 racing, we obtained a dataset from Kaggle, which encompasses detailed information about the 2023 Formula 1 season. The dataset can be found at `https://www.kaggle.com/datasets/juampimgi/formula-1-season-2023?resource=download`. This dataset includes valuable data on race results, driver standings, lap times, and much more, providing a rich basis for our exploratory data analysis (EDA).

Our analysis focused on extracting actionable insights that could potentially influence race strategies and improve team and driver performances. To achieve this, we executed several Hive SQL queries on our Hadoop cluster to handle the large volumes of data efficiently. Each query was designed to test specific hypotheses about correlations between different variables such as weather conditions, circuit characteristics, and driver performance metrics. After obtaining preliminary results from these queries and formulating initial conclusions, we enhanced our analysis, by exporting these results to CSV files, which were then used for creating detailed visualizations in R. This additional step enabled us to achieve a more visual and intuitive understanding of the data, simplifying the derivation of valuable conclusions from the patterns observed in the results.

3.1 Database Structure Overview

For our exploratory data analysis (EDA), we used a dataset from the 2023 season, which is structured into several tables with specific focuses. Each table captures unique aspects of the sport, allowing for a comprehensive analysis from multiple angles. Here's a brief guide to the tables:

- **Events**: Contains information about each race event, including location, dates, and session details.
- **Fastest Laps**: Records the fastest lap times achieved during a race, along with related performance metrics.
- **Laps**: Details every lap completed during the race weekends, with timing and telemetry data.
- **Results**: Summarizes race results, positions, and points for drivers and teams.
- **Sprint Laps**: Focuses on the laps during sprint qualifying sessions, a shorter race format that determines starting positions.
- **Sprint Results**: Outlines the outcomes of sprint qualifying sessions, including grid positions and points.
- **Telemetry**: Provides technical data such as speed, gear, and throttle usage throughout the laps.
- **Weather**: Captures the weather conditions at the event location, which can significantly impact race dynamics.

3.2 Query Execution Framework

To conduct our analysis, we utilized a Spark SQL query execution framework that allowed us to maintain a consistent approach across most of our queries. The majority of queries were executed using a standard template in Spark, which we adapted for specific needs when necessary. This template ensured that all queries were run efficiently and reliably within the Spark environment with Hive support enabled.

The following code snippet outlines the general structure used to execute our SQL queries:

```
from pyspark.sql import SparkSession

# Initialize a SparkSession with Hive support enabled
spark = SparkSession.builder
    .appName("SQL Query Template")
    .enableHiveSupport()
    .getOrCreate()

# Define SQL Query
sql_query = """
"""

# Execute the SQL query
result_df = spark.sql(sql_query)

# Display the results to verify them
result_df.show()

# Stop the Spark session to free up resources when done
spark.stop()
```

3.3 Query Analysis and Insights

3.3.1 Average Number of Laps per Grand Prix

Description: This query aims to determine the average number of laps completed in each Grand Prix event. The analysis provides insights into race durations and can help identify tracks with higher or lower average laps, potentially influencing race strategy.

GP Name	Average Laps
Monaco Grand Prix	38.544914134742406
Austrian Grand Prix	35.53141167775314
Dutch Grand Prix	35.497393894266565
Hungarian Grand Prix	35.11741214057508
Canadian Grand Prix	34.9209726443769
Mexico City Grand...	34.917316692667704
São Paulo Grand Prix	34.30568079350766
Spanish Grand Prix	33.301829268292686
Singapore Grand Prix	30.797235023041473
Abu Dhabi Grand Prix	29.42610198789974
Miami Grand Prix	28.95079086115993
Australian Grand ...	28.687936191425724
Qatar Grand Prix	28.521868787276343
Bahrain Grand Prix	27.900473933649288
United States Gra...	27.87475345167653
British Grand Prix	25.822680412371135
Italian Grand Prix	25.488935721812435
Azerbaijan Grand ...	25.47034339229969
Las Vegas Grand Prix	25.2322375397667
Japanese Grand Prix	25.18409090909091

only showing top 20 rows

Figure 2: Number of Laps GCP Querie Result

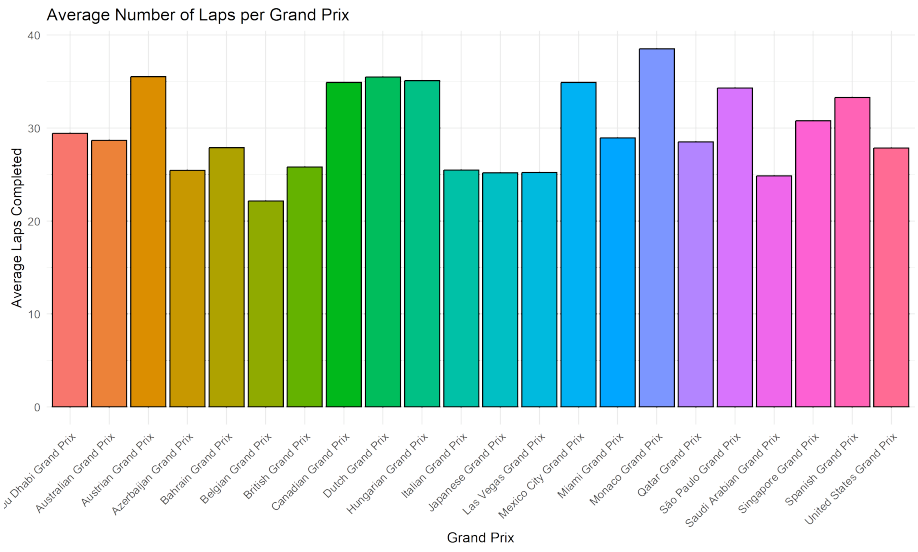


Figure 3: Bar Chart Average Number of Laps per Grand Prix

Interpretation: The bar graph visually represents the average number of laps for each Grand Prix, sorted in descending order. Longer races may indicate a need for different tire strategies or fuel management plans. Conversely, shorter races might lead to more aggressive racing tactics. We can say that Monaco GP is the longest and Belgian GP is the shortest with a difference of around 16 laps.

3.3.2 Average Speed per Sector

Description: This query calculates the average speed for each sector across all laps where speed data is available. It helps in understanding which sectors of the track might demand higher technical skills or allow for greater speeds.

AvgSpeedSector1	AvgSpeedSector2	AvgSpeedFinalSector
254.82625151392813	243.8354864755753	265.1631005248284

Figure 4: Speed Per Sector GCP Querie Result

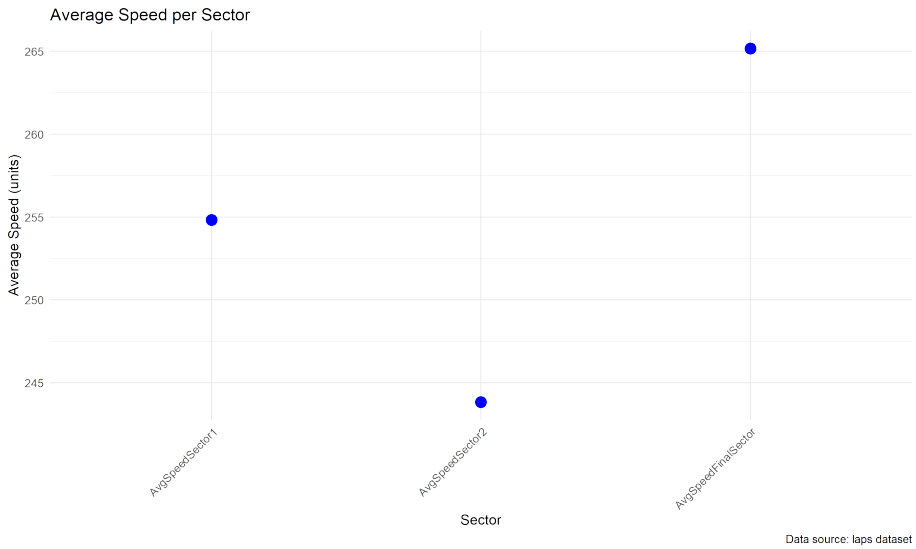


Figure 5: Scatter Plot: Average Speed per Sector

Interpretation: The scatter plot displays the average speeds for the first, second, and final sectors. From this, we can interpret that the final sector allows for higher speeds, which could be crucial for overtaking or qualifying laps.

Conversely, the first and second sectors show lower speeds, possibly indicating tighter corners or technical parts of the track where drivers must balance speed with precision.

3.3.3 Best Lap Times per Grand Prix

Description: This query identifies the best lap time achieved for each Grand Prix. It gives us a benchmark to aim for when seeking to achieve a fast lap time. Notably, a modification was made to this query to use a later-described data integrity check, ensuring accurate differentiation between null values and empty strings. The full process of this data verification will be outlined later.

GP Name	Best Lap Time
Austrian Grand Prix	1.1168667
São Paulo Grand Prix	1.2081
Dutch Grand Prix	1.2306167
Canadian Grand Prix	1.24135
Monaco Grand Prix	1.2608334
Spanish Grand Prix	1.2721666
Australian Grand ...	1.33725
Hungarian Grand Prix	1.3417333
Mexico City Grand...	1.3555666
Qatar Grand Prix	1.4053167
Italian Grand Prix	1.4178667
Abu Dhabi Grand Prix	1.4498833
Miami Grand Prix	1.4951333
British Grand Prix	1.5045834
Saudi Arabian Gra...	1.5317667
Bahrain Grand Prix	1.5666
Japanese Grand Prix	1.5697167
Las Vegas Grand Prix	1.5915
Singapore Grand Prix	1.5977833
United States Gra...	1.63565

Figure 6: Best Lap Times per Grand Prix GCP Querie Result

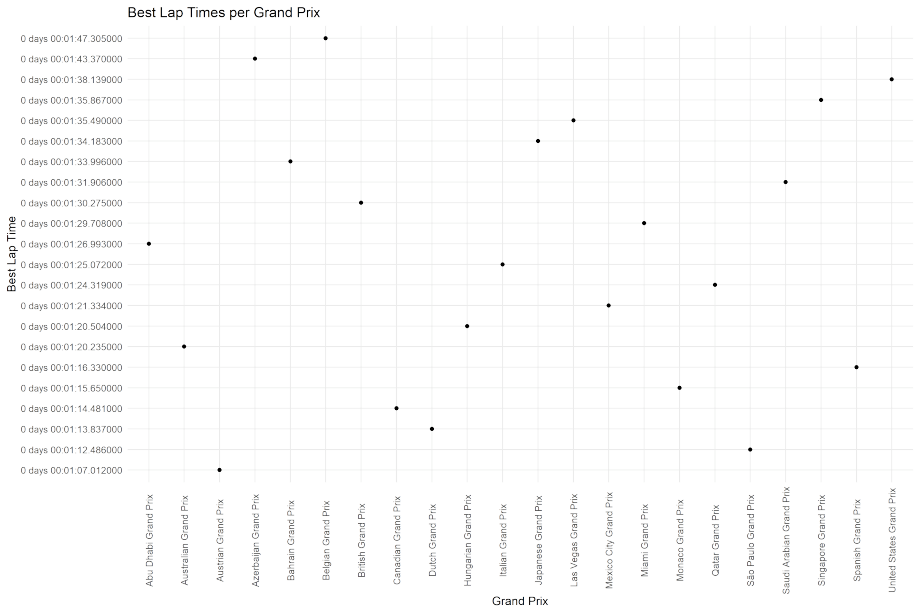


Figure 7: Dot plot of the Best Lap Times per Grand Prix visualized using R

Interpretation: The dot plot illustrates the best lap times for each Grand Prix, highlighting the disparity in circuit performance. It's interesting to see how, even in small increments, there are significant variations in lap times across different Grand Prix events. This underscores the importance of analyzing circuit performance, as even minor differences can impact race outcomes.

3.3.4 Density of Number of Drivers in Grand Prix Events

Description: This query assesses the frequency at which the starting grid is at full capacity by counting the distinct number of drivers in each Grand Prix event. This metric highlights the Grand Prix events where the grid was complete, providing insights into participation consistency which may impact the overall competitiveness and race dynamics.

GP Name	Number Of Drivers
Singapore Grand Prix	19
São Paulo Grand Prix	20
Spanish Grand Prix	20
Miami Grand Prix	20
Italian Grand Prix	20
Las Vegas Grand Prix	20
Dutch Grand Prix	20
Canadian Grand Prix	20
Hungarian Grand Prix	20
Azerbaijan Grand ...	20
Qatar Grand Prix	20
British Grand Prix	20
Monaco Grand Prix	20
Austrian Grand Prix	20
Saudi Arabian Gra...	20
Australian Grand ...	20
Bahrain Grand Prix	20
Mexico City Grand...	20
Abu Dhabi Grand Prix	20
Belgian Grand Prix	20

Figure 8: Density Number of Drivers GCP Querie Result

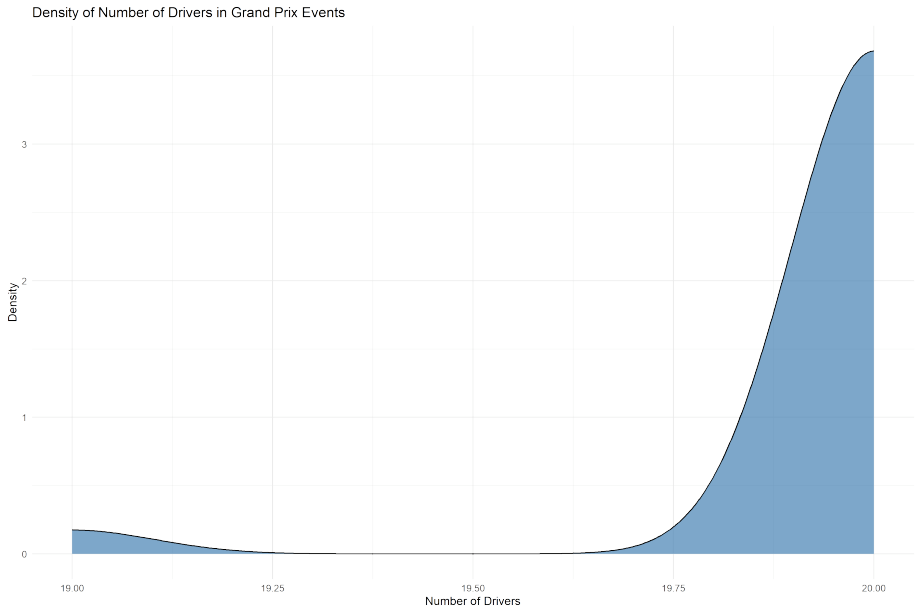


Figure 9: Density Plot of the Number of Drivers per Grand Prix

Interpretation: The density plot provides a visual distribution of the number of distinct drivers participating in each Grand Prix event. It helps to identify the general trend in driver participation across the season. This analysis can be

particularly useful for teams and organizers in understanding the draw of each event and planning strategies.

3.3.5 Grand Prix Schedule 2023

Description: This query lists all the events in the 2023 Formula 1 season, sorted by date. It helps to visualize the distribution of Grand Prix events throughout the year, providing insight into the periods with the most intense schedules for the teams and drivers.

GP Name	Date
Australian Grand ...	02/04/23 17:00
Austrian Grand Prix	02/07/23 17:00
Italian Grand Prix	03/09/23 17:00
Spanish Grand Prix	04/06/23 17:00
Bahrain Grand Prix	05/03/23 20:00
São Paulo Grand Prix	05/11/23 16:00
Miami Grand Prix	07/05/23 17:30
Qatar Grand Prix	08/10/23 22:00
British Grand Prix	09/07/23 17:00
Singapore Grand Prix	17/09/23 22:00
Canadian Grand Prix	18/06/23 16:00
Las Vegas Grand Prix	18/11/23 23:59
Saudi Arabian Gra...	19/03/23 22:00
United States Gra...	22/10/23 16:00
Hungarian Grand Prix	23/07/23 17:00
Japanese Grand Prix	24/09/23 16:00
Abu Dhabi Grand Prix	26/11/23 19:00
Dutch Grand Prix	27/08/23 17:00
Monaco Grand Prix	28/05/23 17:00
Mexico City Grand...	29/10/23 16:00

Figure 10: Event Dates Distribution GCP Querie Result

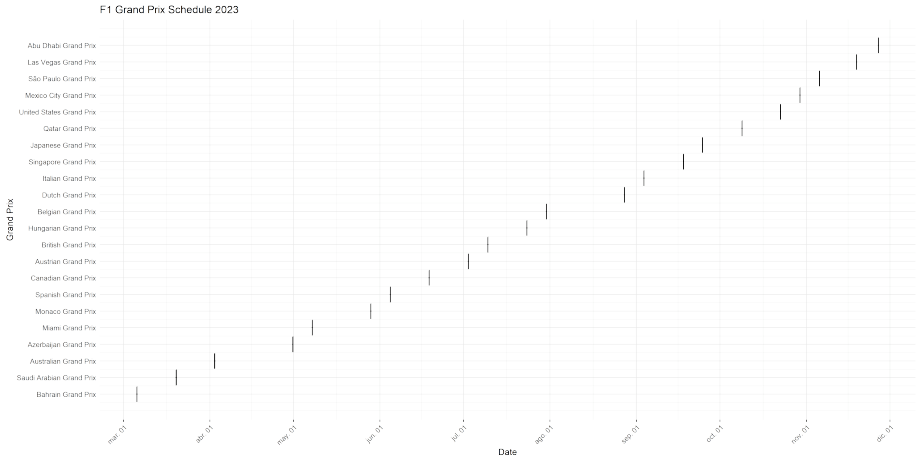


Figure 11: Gantt Chart of the F1 Grand Prix Schedule 2023

Interpretation: The Gantt chart illustrates the Formula 1 event schedule for the 2023 season. The timing of the races shows the workload and travel demands on the teams and drivers, highlighting periods with back-to-back races as well as breaks within the season. This information is crucial for teams to plan their logistics, training, and development activities efficiently.

3.3.6 Total Laps by Tire Compound

Description: This query categorizes the total number of laps completed on each tire compound, offering a clear depiction of tire utilization throughout the season. By examining the percentages of laps completed with each compound, teams can gain valuable insights into wear rates and preferences, which can significantly influence strategic decisions regarding tire management during races.

Compound	Total Laps
HARD	11853
MEDIUM	8300
SOFT	3476
INTERMEDIATE	723
WET	48

Figure 12: Total Laps per Compound GCP Querie Result

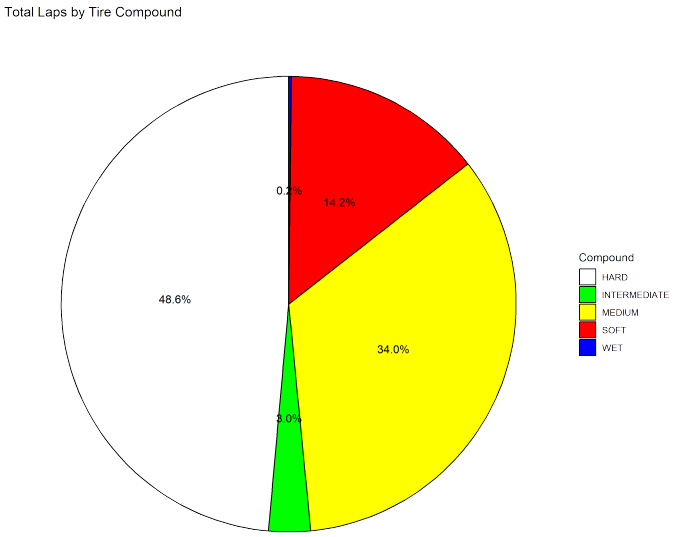


Figure 13: Pie Chart of Total Laps by Tire Compound

Interpretation: The pie chart visually represents the proportion of total laps completed on different tire compounds throughout the season. The distribution can reflect not only a preference for certain compounds but also race conditions and regulations affecting tire choice. For example, a larger proportion of one compound could indicate its effectiveness or preference under specific race conditions.

3.3.7 Number of Fastest Laps per Driver

Description: This query calculates the total number of the fastest laps each driver achieved throughout the season, highlighting which drivers consistently performed at the top of their game during the critical moments of a race.

Driver	NumberOfFastestLaps
VER	9
HAM	4
PIA	2
PER	2
ZHO	1
RUS	1
NOR	1
ALO	1
TSU	1

Figure 14: Fastest Laps per Driver GCP Query Result

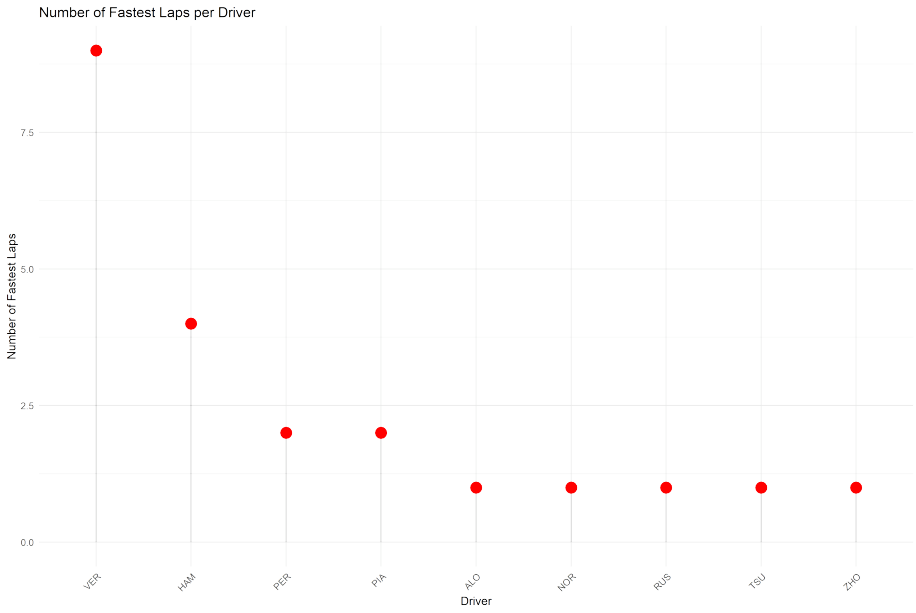


Figure 15: Lollipop Chart of the Number of Fastest Laps per Driver

Interpretation: The lollipop chart shows the number of fastest laps achieved by each driver, providing a clear visual of those who frequently led the pack. Drivers with a higher number of fastest laps may have a combination of skill, speed, and a well-performing car suited to the tracks, indicating their potential for success in races and qualifications.

3.3.8 Driver Lap Completion Analysis

Description: This query analyzes each driver's completed versus incomplete laps, expressed both in absolute terms and as percentages. It highlights drivers'

reliability and consistency during races. For this analysis, we applied a rigorous data validation approach similar to one previously mentioned. This step was crucial to confirm the precision of our dataset, particularly in distinguishing between non-existent (null) and explicitly recorded empty lap times. A thorough explanation of this validation technique will be provided in the upcoming segments.

Driver	CompletedLaps	IncompleteLaps	PercentageOfCompletedLaps	PercentageOfIncompleteLaps
NOR	1235	38	97.01	2.99
RIC	403	14	96.64	3.36
GAS	1206	42	96.63	3.37
ZHO	1202	42	96.62	3.38
DEV	543	19	96.62	3.38
ALB	1112	39	96.61	3.39
TSU	1163	41	96.59	3.41
HUL	1207	43	96.56	3.44
ALO	1249	45	96.52	3.48
BOT	1185	43	96.5	3.5
MAG	1130	43	96.33	3.67
HAM	1222	47	96.3	3.7
PIA	1141	44	96.29	3.71
LEC	1106	43	96.26	3.74
RUS	1207	47	96.25	3.75
VER	1275	50	96.23	3.77
LAW	281	11	96.23	3.77
STR	1120	44	96.22	3.78
PER	1169	46	96.21	3.79
SAR	1099	45	96.07	3.93

Figure 16: Driver Lap Completion Rate GCP Querie Result

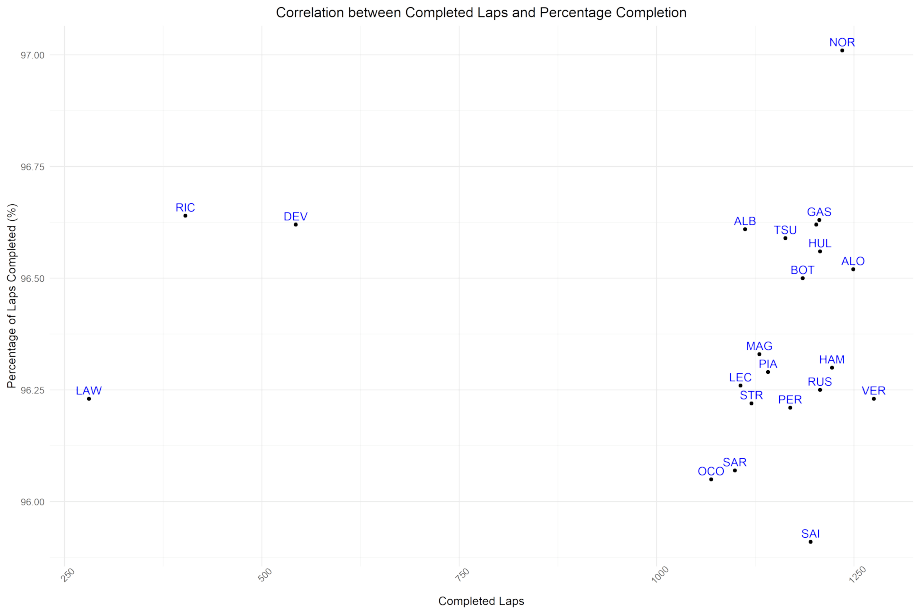


Figure 17: Scatter Plot of Driver Lap Completion Rates

Interpretation: The scatter plot showcases the completion rate of laps for each driver across the season, providing insights into which drivers are more likely to finish laps, and hence, potentially races. This measure of consistency is essential for teams when evaluating driver performance and endurance over the season.

3.3.9 Telemetry Analysis by Driver and Event

Description: This query delves into telemetry data to extract average throttle usage, braking frequency, speed statistics, RPM levels, and DRS activation percentages for each driver across events. Such analysis paints a comprehensive picture of driving styles and car performance under different race conditions.

EventName	DriverNumber	AvgThrottleUsage	AvgBrakeUsage	AvgSpeed	MaxSpeed	MinSpeed	StdDevSpeed	AvgRPM	MaxRPM	MinRPM	StdDevRPM	PercentageDRSActivated
Abu Dhabi Grand Prix	81	81.1918381892381	0.12849	199.1416576867816	336.0	1.0	113.66586014228912	6444.354447747542	12932.0	0.0	4839.80631606502	33.463869197347 36
Abu Dhabi Grand Prix	55	85.51043135043699	0.24363	199.8839468446915	335.0	1.0	113.29369542775582	6338.995529384862	12619.0	0.0	4819.508746672043	37.843813949233 94
Abu Dhabi Grand Prix	4	76.98982472963614	0.13015	198.83630499059225	333.0	1.0	113.71907595234467	6586.312079807912	12687.0	0.0	4896.185767766256	33.489595245353990
Abu Dhabi Grand Prix	10	84.68154082877912	0.13235	198.41604282927273	333.0	3.0	112.61548073985809	6466.460124628402	12443.0	0.0	4920.212296322841	18.219757603475 87
Abu Dhabi Grand Prix	27	81.4575727288447	0.13870	198.26210018046625	334.0	1.0	112.31474680542365	6519.810163045964	12514.0	0.0	4884.522495316021	33.435301760084 94
Abu Dhabi Grand Prix	20	81.34992840621214	0.14224	197.97779787474565	340.0	1.0	112.62045973712398	6384.656843128288	12425.0	0.0	4834.546629597155 5	33.552298193459 87
Abu Dhabi Grand Prix	14	83.48556533437973	0.12371	197.81951502184185	331.0	1.0	113.1518165022427 5	6461.432369083012	12963.0	0.0	4845.750921786812	9.735879259898 87
Abu Dhabi Grand Prix	31	84.47030983913899	0.17379	197.71431802759557	331.0	3.0	112.86476688882033	6451.371684189491	12121.0	0.0	4766.541664276657	18.785730619711 87
Abu Dhabi Grand Prix	18	88.2220054623488	0.13840	197.58685413185413	333.0	1.0	112.68277654876033	6470.011548136291	12827.0	0.0	4815.026267504889	12.525726046192 55
Abu Dhabi Grand Prix	16	79.36572612935136	0.19709	197.38508627955463	332.0	1.0	112.78170372973008	6655.319403155729	12442.0	0.0	4718.356920949527	36.393780013720 56
Abu Dhabi Grand Prix	1	80.75447663795353	0.15333	197.37417914368268	333.0	3.0	112.13960138209451	6470.013634804482	12635.0	0.0	4772.013394567441	14.718156871712 78
Abu Dhabi Grand Prix	22	84.23569627346596	0.11874	197.25226465364122	329.0	1.0	112.47521012707233	6548.570868965013	12726.0	0.0	4991.682337813339 5	11.388063114566 66
Abu Dhabi Grand Prix	63	80.97819470086323	0.16636	197.11729684947795	332.0	1.0	112.64721547996638	6595.271524125314 5	12944.0	0.0	4782.527345269344	14.109307111822 55
Abu Dhabi Grand Prix	3	84.88802234764266	0.11728	197.06448686426414	338.0	1.0	112.4955285193864 4	6500.833037960421	12411.0	0.0	4918.266816582623 5	11.622455978876 51
Abu Dhabi Grand Prix	44	77.79161050304631	0.12449	197.0252390725296	336.0	2.0	112.21655145049095	6449.894866224566	13190.0	0.0	4744.730344582316	8.632517772238 738
Abu Dhabi Grand Prix	24	85.51223445924151	0.24783	196.75190299167994	338.0	2.0	111.91819725643127	6418.329693574205 5	12504.0	0.0	4768.4565966869	33.372398810884 98
Abu Dhabi Grand Prix	11	80.28776447898349	0.11879	196.67978525157757	343.0	1.0	113.52430677814577	6290.917276469243	12486.0	0.0	4846.097956869517	13.071689915338 985
Abu Dhabi Grand Prix	23	74.84663223943583	0.13137	196.5670939981369	339.0	1.0	112.58762239569943	6514.492653784588	12805.0	0.0	4844.290663004521	10.581980333866 91
Abu Dhabi Grand Prix	2	80.00765102203951	0.13650	196.36638774516315	343.0	1.0	112.49230597381672	6430.768179739311	13003.0	0.0	4784.565836392221	11.465241253146 29
Abu Dhabi Grand Prix	77	86.65628232389613	0.27776	196.49582408656268	332.0	1.0	111.73777466422183	6287.8898639378	12596.0	0.0	4824.863525475 64	32.729247656 87135

Figure 18: Event Telemetry GCP Querie Result

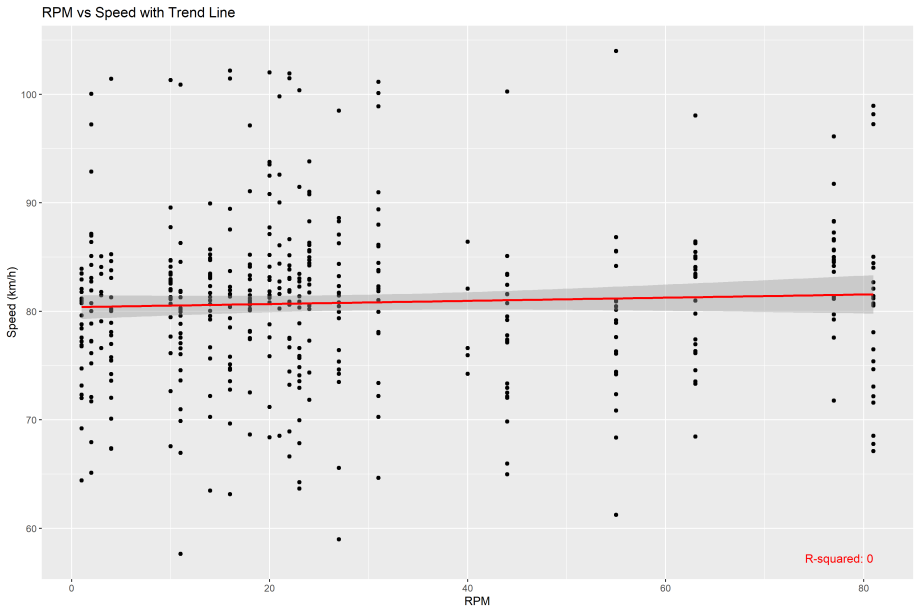


Figure 19: Scatter Plot with Trend Line: RPM vs Speed Analysis

Interpretation: The scatter plot with the overlaying trend line represents the relationship between RPM (revolutions per minute) and the speed of the vehicles. Despite the variability, the trend line suggests a weak positive correlation; however, with an R-squared value of 0, this indicates no predictive power. This could suggest that factors other than RPM may have a more significant influence on speed, or it could reflect a variety of driving styles and strategies employed by the drivers. Additionally, the spread of data points might indicate differing car performances or the impact of track conditions.

3.3.10 Correlation between Lap Times by Tire Compound and Weather Conditions

Description: This analysis investigates the interaction between tire compound choices and weather conditions, looking at average lap times. Through the use of window functions in Spark SQL, we've been able to categorize weather conditions and directly correlate them with performance metrics such as average lap times for different tire compounds.

EventName	Compound	AvgLapTime	TemperatureCondition	HumidityCondition
Abu Dhabi Grand Prix	HARD	1.5122646982506376	Hot	Low/Medium Humidity
Abu Dhabi Grand Prix	MEDIUM	1.5209387151731386	Hot	Low/Medium Humidity
Abu Dhabi Grand Prix	SOFT	1.8474500179290771	Hot	Low/Medium Humidity
Australian Grand ...	HARD	1.4127811685958738	Mild	Low/Medium Humidity
Australian Grand ...	MEDIUM	1.5869237087391042	Mild	Low/Medium Humidity
Australian Grand ...	SOFT	1.5925022204717	Mild	Low/Medium Humidity
Austrian Grand Prix	HARD	1.2054956539081094	Mild	Low/Medium Humidity
Austrian Grand Prix	MEDIUM	1.230662143250541	Mild	Low/Medium Humidity
Austrian Grand Prix	SOFT	1.2903500199317932	Mild	Low/Medium Humidity
Azerbaijan Grand ...	HARD	1.8087791345163744	Cold	Low/Medium Humidity
Azerbaijan Grand ...	MEDIUM	1.822383662587718	Cold	Low/Medium Humidity
Azerbaijan Grand ...	SOFT	1.9831583201885223	Cold	Low/Medium Humidity
Bahrain Grand Prix	SOFT	1.6717516039274378	Hot	Low/Medium Humidity
Bahrain Grand Prix	HARD	1.6731211771703747	Hot	Low/Medium Humidity
Bahrain Grand Prix	MEDIUM	1.6864450216293334	Hot	Low/Medium Humidity
Belgian Grand Prix	SOFT	1.8901269266041376	Mild	Low/Medium Humidity
Belgian Grand Prix	MEDIUM	1.9117902502851578	Mild	Low/Medium Humidity
Belgian Grand Prix	HARD	1.9305416544278462	Mild	Low/Medium Humidity
British Grand Prix	MEDIUM	1.5841517329469101	Mild	Low/Medium Humidity
British Grand Prix	HARD	1.6585703455168626	Mild	Low/Medium Humidity

Figure 20: Weather Insights per Event GCP Querie Result

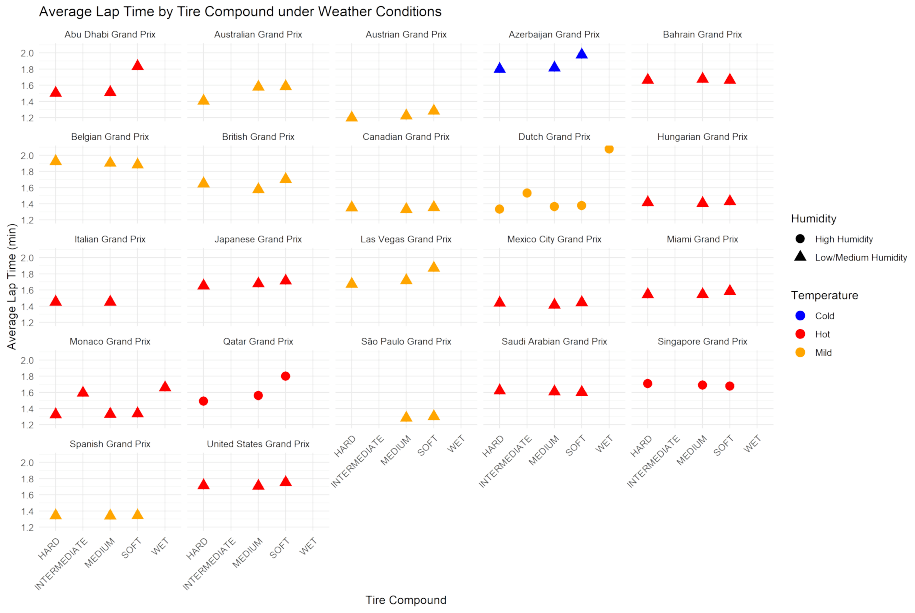


Figure 21: Interaction Plot of Performance vs. Weather Conditions

Interpretation: The interaction plot demonstrates how tire compounds perform under various temperature and humidity conditions during different Grand Prix events. Circular markers for 'High Humidity' and triangular for varying temperature conditions give a visual representation of their impact on lap times. This kind of analysis is essential for teams to adapt their strategies, such as tire

choices and car setups, to the predicted weather on a race day.

3.3.11 Dot Chart Overview by Driver

Description: This analysis offers a visualization of average lap times juxtaposed with the completion rate and the most used tire compounds for each driver. It serves as a way to observe correlations between performance and tire choices.

Driver	Compound	AvgLapTime	CompletionRate
NOR	HARD	2266.4761125772993	97.01492537313433
RIC	MEDIUM	2310.7345497094293	96.64268585131894
GAS	HARD	2286.4971731356004	96.63461538461539
ZHO	HARD	2299.8155260130898	96.62379421221866
DEV	HARD	2212.583917343831	96.61921708185054
ALB	HARD	2304.693481218646	96.61164205039097
TSU	HARD	2278.907522872827	96.59468438538205
HUL	HARD	2316.922136730159	96.56
ALO	HARD	2270.015386081986	96.52241112828439
BOT	HARD	2297.7400995647267	96.49837133550488
MAG	HARD	2315.7818197940705	96.33418584825235
HAM	MEDIUM	2262.5382499057005	96.29629629629629
PIA	HARD	2238.780951517528	96.28691983122363
LEC	HARD	2288.9775550924946	96.25761531766753
RUS	HARD	2278.837804129689	96.25199362041468
LAW	HARD	2348.8217946018985	96.23287671232876
VER	HARD	2245.0851447963937	96.22641509433963
STR	HARD	2250.8564061167554	96.21993127147766
PER	HARD	2263.125911348334	96.21399176954732
SAR	HARD	2317.7484995750983	96.06643356643356

Figure 22: Driver Overview GCP Querie Result

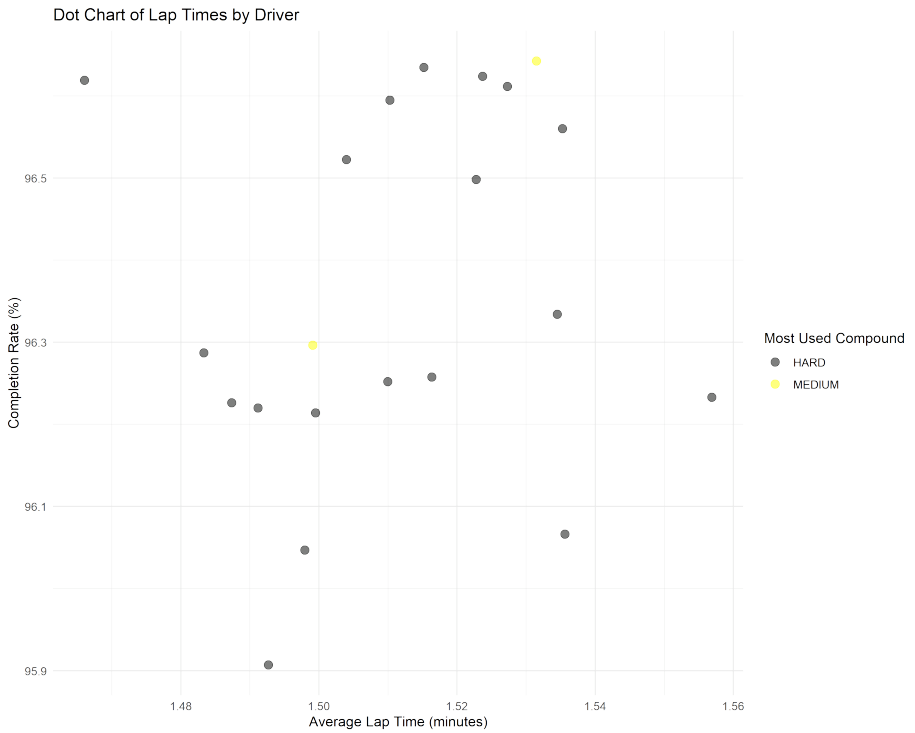


Figure 23: Dot Chart Overview by Driver

Interpretation: The dot chart presents a comparison of drivers' average lap times with a color-coded indication of their most utilized tire compound. This visualization can highlight patterns and preferences in tire usage, as well as their impact on lap performance. The completion rate on the y-axis further adds context, indicating reliability or potential issues during races. Together, these metrics allow teams to fine-tune their strategies, particularly around tire selection and management during various phases of a race.

4 Challenges Encountered

During the development and analysis phases of our project, we encountered several challenges that fell into two main categories: infrastructure-related and query/code-related. Below, we detail these issues and how we addressed them.

4.1 Infrastructure-Related Challenges

Google Cloud Storage (GCS) Bucket Naming Error: We encountered an error related to the naming of the Google Cloud Storage (GCS) bucket.

The error message indicated that the bucket name contained invalid characters, which are restricted in GCS naming conventions.

1. **Bucket Name:** GCS bucket names can only include lowercase letters (a-z), numbers (0-9), underscores (_), hyphens (-), and dots (.). Spaces and uppercase letters are not permitted. This error stemmed from an oversight in the initial naming phase.

Figure 24: Bucket Name Error

To solve this issue, we renamed the bucket to comply with the GCS naming conventions by removing any special characters.

4.2 Query and Code-Related Challenges

SQL Syntax Errors: While executing our SQL queries on the Google Cloud Platform, we faced some errors. These issues were largely due to our initial lack of familiarity with the specific limitations and unique characteristics of SQL functions and data types supported by Google Cloud services. This lack of understanding led to incorrect usage in our queries, impacting our ability to execute them efficiently.

Spark SQL AnalysisException: In the data processing phase using Spark SQL, we encountered some 'AnalysisException' which indicated a misuse of outer query columns in subqueries. Specifically, we faced issues with queries that incorrectly referenced outer columns in subqueries due to the scoping and execution constraints of Spark SQL.

```
SELECT
    l.Driver,
    (SELECT Compound
     FROM laps
     WHERE Driver = l.Driver AND LapTime IS NOT NULL
     GROUP BY Compound
```

```
      ORDER BY COUNT(*) DESC LIMIT 1) AS MostUsedCompound,
    AVG(
        SUBSTRING_INDEX(SUBSTRING_INDEX(LapTime, ' days ',
            1), ' days ', -1) * 1440 +
        SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING_INDEX(
            LapTime, ' days ', -1), ':', 1), ' ', -1) * 60 +
        SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING_INDEX(
            LapTime, ':', 2), ':', -1), ' ', 1) +
        SUBSTRING_INDEX(SUBSTRING_INDEX(LapTime, ':', -1), '
            .', 1) / 60
    ) AS AvgLapTime,
    (COUNT(l.LapTime) / COUNT(*)) * 100 AS CompletionRate
FROM
    laps l
GROUP BY
    l.Driver
ORDER BY
    CompletionRate DESC, AvgLapTime;
```

Figure 25: Spark SQL AnalysisException encountered during data processing

To address this problem, the SQL query needed to be restructured to avoid direct references to outer query columns within subqueries. Instead of segregating computations, we implemented window functions that allowed us to perform necessary calculations in a single pass of the data. This method retains the context of each row within its partition, enabling us to carry out more complex analyses like ranking and aggregation without resorting to subqueries.

Window Functions Implementation: The updated code segment utilizes window functions to identify the most used compound by each driver. Following

this, we aggregate average lap times and completion rates, all within the same query framework. This approach effectively sidesteps the restrictions imposed by subquery referencing and leverages the powerful analytical capabilities provided by Spark SQL's window functions:

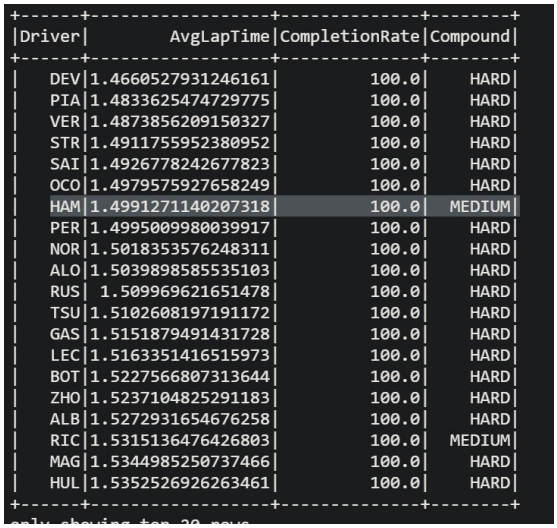
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, expr, count, when, avg
from pyspark.sql.window import Window

spark = SparkSession.builder \
    .appName("Driver Analysis Without Subqueries") \
    .enableHiveSupport() \
    .getOrCreate()

compounds_usage = spark.sql("""
SELECT Driver, Compound, COUNT(*) AS UsageCount
FROM laps
WHERE LapTime IS NOT NULL
GROUP BY Driver, Compound
""")

windowSpec = Window.partitionBy("Driver").orderBy(col("UsageCount").desc())
most_used_compound = compounds_usage.withColumn("rank", row_number().over(windowSpec)) \
    .filter(col("rank") == 1) \
    .drop("UsageCount", "rank")

driver_stats = spark.table("laps_temp").groupBy("Driver").agg(
    avg(
        when(col("LapTime").isNotNull(),
             expr("""
                SUBSTRING_INDEX(SUBSTRING_INDEX(LapTime, ' days ', 1), ' days ', -1) * 1440 +
                SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING_INDEX(LapTime, ' days ', -1), ':', 1), ' ', -1) * 60 +
                SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING_INDEX(LapTime, ':', 2), ':', -1), ' ', 1) +
                SUBSTRING_INDEX(SUBSTRING_INDEX(LapTime, ':', -1), '.', 1) / 60
             """)
        )
    ).alias("AvgLapTime"),
    (count(when(col("LapTime").isNotNull(), 1)).cast("double") / count("*").cast("double") * 100).alias("CompletionRate")
```

```
)

final_result = driver_stats.join(most_used_compound, ["Driver"], "left") \
    .select("Driver", "Compound", "AvgLapTime", "CompletionRate") \
    .orderBy(col("CompletionRate").desc(), col("AvgLapTime"))

final_result.show()
```

By applying this method, we successfully circumvented the limitation and obtained the required analysis results.

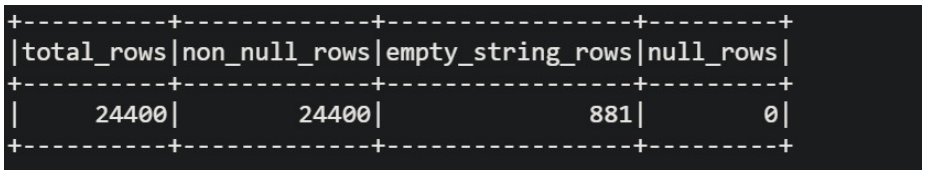
4.3 Data Integrity Challenges

Handling Nulls as Empty Strings: We experienced challenges in calculating accurate percentages in our data analysis due to an unexpected behavior of Google Cloud, which treated null values as empty strings. This discrepancy led to incorrect computation of metrics, affecting the integrity of our results.



Driver	AvgLapTime	CompletionRate	Compound
DEV	1.4660527931246161	100.0	HARD
PIA	1.4833625474729775	100.0	HARD
VER	1.4873856209150327	100.0	HARD
STR	1.4911755952380952	100.0	HARD
SAI	1.4926778242677823	100.0	HARD
OCO	1.4979575927658249	100.0	HARD
HAM	1.4991271140207318	100.0	MEDIUM
PER	1.4995009980039917	100.0	HARD
NOR	1.5018353576248311	100.0	HARD
ALO	1.5039898585535103	100.0	HARD
RUS	1.509969621651478	100.0	HARD
TSU	1.5102608197191172	100.0	HARD
GAS	1.5151879491431728	100.0	HARD
LEC	1.5163351416515973	100.0	HARD
BOT	1.5227566807313644	100.0	HARD
ZHO	1.5237104825291183	100.0	HARD
ALB	1.5272931654676258	100.0	HARD
RIC	1.5315136476426803	100.0	MEDIUM
MAG	1.5344985250737466	100.0	HARD
HUL	1.5352526926263461	100.0	HARD

Figure 26: Error when calculating Average Lap Time and Completion rate



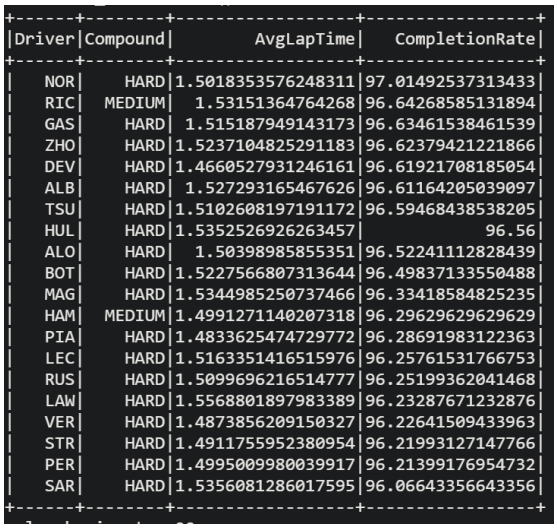
```
+----------+-------------+-----------------+---------+
|total_rows|non_null_rows|empty_string_rows|null_rows|
+----------+-------------+-----------------+---------+
|     24400|        24400|              881|        0|
+----------+-------------+-----------------+---------+
```

Figure 27: Null and Empty Rows

To solve this, we modified our data processing steps using Spark SQL to explicitly replace empty strings with null values, ensuring the accuracy of subsequent computations. The following code snippet shows the adjustments made to the 'LapTime' data:

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

spark = SparkSession.builder
    .appName("Update LapTimes")
    .enableHiveSupport()
    .getOrCreate()

laps_df = spark.table("laps")

laps_df_updated = laps_df.withColumn(
    "LapTime",
    when(col("LapTime") != "", col("LapTime")).otherwise(
        None)
)

laps_df_updated.createOrReplaceTempView("laps_temp")

result = spark.sql("""
    SELECT Driver, COUNT(*) AS TotalLaps, SUM(CASE WHEN
        LapTime IS NULL THEN 1 ELSE 0 END) AS NullLapTimes
    FROM laps_temp
    GROUP BY Driver
""")

result.show()
```

The corrected calculations revealed the true distribution of null values, which is essential for accurate data analysis. This adjustment was a critical step in maintaining the quality and reliability of our data-driven insights.



Driver	Compound	AvgLapTime	CompletionRate
NOR	HARD	1.5018353576248311	97.01492537313433
RIC	MEDIUM	1.53151364764268	96.64268585131894
GAS	HARD	1.515187949143173	96.63461538461539
ZHO	HARD	1.5237104825291183	96.62379421221866
DEV	HARD	1.4660527931246161	96.61921708185054
ALB	HARD	1.527293165467626	96.61164205039097
TSU	HARD	1.5102608197191172	96.59468438538205
HUL	HARD	1.5352526926263457	96.56
ALO	HARD	1.50398985855351	96.52241112828439
BOT	HARD	1.5227566807313644	96.49837133550488
MAG	HARD	1.5344985250737466	96.33418584825235
HAM	MEDIUM	1.4991271140207318	96.29629629629629
PIA	HARD	1.4833625474729772	96.28691983122363
LEC	HARD	1.5163351416515976	96.25761531766753
RUS	HARD	1.5099696216514777	96.25199362041468
LAW	HARD	1.5568801897983389	96.23287671232876
VER	HARD	1.4873856209150327	96.22641509433963
STR	HARD	1.4911755952380954	96.21993127147766
PER	HARD	1.4995009980039917	96.21399176954732
SAR	HARD	1.5356081286017595	96.06643356643356

Figure 28: Final Result With Accurate Null Lap Times

Each challenge was carefully addressed to ensure continuation of our project, allowing us to leverage full capabilities of our big data infrastructure and analytical tools.

5 Data Processing Optimizations

5.1 Conversion of Lap Times for Simplified Access

During our analysis, we recognized the need for a standardized format for lap times to facilitate easier access and more efficient queries. To achieve this, we created a user-defined function (UDF) that converts the lap times into a consistent unit of minutes, thus normalizing the data and making subsequent analyses more straightforward.

Lap Time Conversion: The code snippet below details the UDF convert_laptime_to_minutes that we applied to the entire dataset. This function parses the original lap time strings and calculates the total time in minutes, which is then used to update the dataset. The resulting table, laps_temp_updated, provides a clean, uniform set of lap times across all entries, significantly optimizing our data processing pipeline:

```
1 from pyspark.sql import SparkSession
2
3 spark = SparkSession.builder \
4     .appName("Weather Insights") \
5     .enableHiveSupport() \
6     .getOrCreate()
7
```

```
weather_insights_query = """
SELECT
    EventName,
    Compound,
    AVG(LapTime) AS AvgLapTime,
    TemperatureCondition,
    HumidityCondition
FROM (
    SELECT
        l.EventName,
        l.Compound,
        l.LapTime,
        CASE
            WHEN AVG(w.AirTemp) OVER (PARTITION BY l.
                EventName) >= 25 THEN 'Hot'
            WHEN AVG(w.AirTemp) OVER (PARTITION BY l.
                EventName) BETWEEN 15 AND 24 THEN 'Mild'
            ELSE 'Cold'
        END AS TemperatureCondition,
        CASE
            WHEN AVG(w.Humidity) OVER (PARTITION BY l.
                EventName) > 70 THEN 'High Humidity'
            ELSE 'Low/Medium Humidity'
        END AS HumidityCondition
    FROM
        laps_temp_updated l
    JOIN
        weather w ON l.EventName = w.EventName
    WHERE
        l.LapTime IS NOT NULL
) sub
GROUP BY
    EventName, Compound, TemperatureCondition,
       HumidityCondition
ORDER BY
    EventName, AvgLapTime;
"""

weather_insights = spark.sql(weather_insights_query)
weather_insights.show()
```

This transformation has streamlined our data handling, making it much more efficient for subsequent SQL queries to perform analyses, such as calculating weather insights or identifying best times in each Grand Prix. By normalizing the lap times, we have reduced the complexity and potential for errors in our data-driven insights.

6 Hive Metastore

6.0.1 SQL Connection to the Hive Metastore with the SDK

```
cloud sql connect formula1-bigdata-term2-up-sql --user=root
```

6.0.2 Selecting Database

Once the connection is established, we select the *hive_metastore* database to perform operations on the metastore tables.

```
USE hive_metastore;
```

6.0.3 Querying the Table Location

The following query is designed to retrieve metadata from the Hive Metastore regarding the storage details of a specific table. In this case, the table in question is ‘laps‘.

```
SELECT s.* FROM ‘hive_metastore‘.‘TBLS‘ t
JOIN ‘hive_metastore‘.‘DBS‘ d ON t.‘DB_ID‘ = d.‘DB_ID‘
JOIN ‘hive_metastore‘.‘SDS‘ s ON t.‘SD_ID‘ = s.‘SD_ID‘
WHERE t.‘TBL_NAME‘ = ’laps’ AND d.‘name‘ = ’default’;
```

This query joins three critical tables of the Hive Metastore:

- `TBLS` - Contains metadata about the tables including the table name and associated database and storage descriptors.
- `DBS` - Stores details about databases such as names and descriptions.
- `SDS` - Holds storage descriptors which include information about the physical storage of the data (such as location, input/output format, etc.).

```
mysql> SELECT s.* FROM `hive_metastore`.`TBLS` t
    -> JOIN `hive_metastore`.`DBS` d ON t.`DB_ID` = d.`DB_ID`
    -> JOIN `hive_metastore`.`SDS` s ON t.`SD_ID` = s.`SD_ID`
    -> WHERE t.`TBL_NAME` = 'laps' AND d.`name` = 'default';
+-------+-------+------------------------------------------------------------+---------------+-----------------------+-----------------------------------------------------------+-------------+------------------------------------------------------------+----------+
| SD_ID | CD_ID | INPUT_FORMAT                                               | IS_COMPRESSED | IS_STOREDASSUBDIRECTORIES | LOCATION                                                  | NUM_BUCKETS | OUTPUT_FORMAT                                              | SERDE_ID |
+-------+-------+------------------------------------------------------------+---------------+-----------------------+-----------------------------------------------------------+-------------+------------------------------------------------------------+----------+
|    19 |    19 | org.apache.hadoop.mapred.TextInputFormat                   | 0x00          | 0x00                  | gs://formula1-bigdata-term2-up/datasets/Laps              |          -1 | org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |       19 |
+-------+-------+------------------------------------------------------------+---------------+-----------------------+-----------------------------------------------------------+-------------+------------------------------------------------------------+----------+
1 row in set (0.04 sec)
```

Figure 29: Searching Data Location

6.0.4 Querying Data Format and Location

The SQL command below is utilized to fetch the data input format and the storage location of the ‘Laps‘ table. This query is crucial for understanding how the data is read and where it is physically stored.

```sql
SELECT INPUT_FORMAT, LOCATION
FROM SDS s, TBLS t
WHERE s.SD_ID = t.SD_ID AND t.TBL_NAME = 'Laps';
```

The query performs a simple join between the ‘SDS‘ (Storage Descriptors) table and the ‘TBLS‘ (Tables) table on the ‘SD_ID‘ field, which links the table's storage information to its physical metadata. The ‘INPUT_FORMAT‘ field provides insight into the data serialization and deserialization format, and the ‘LOCATION‘ field specifies the exact path in the storage system where the table's data files reside.

```
mysql> SELECT s.INPUT_FORMAT, s.LOCATION FROM SDS s JOIN TBLS t ON s.SD_ID = t.SD_ID WHERE t.TBL_NAME = 'laps';
+-----------------------------------------+-----------------------------------------------+
| INPUT_FORMAT                            | LOCATION                                      |
+-----------------------------------------+-----------------------------------------------+
| org.apache.hadoop.mapred.TextInputFormat | gs://formula1-bigdata-term2-up/datasets/Laps |
+-----------------------------------------+-----------------------------------------------+
1 row in set (0.04 sec)
```

Figure 30: Searching Data Format

7 Conclusions

Through the utilization of big data technologies such as Hadoop and Hive on the Google Cloud Platform, this project has provided significant insights into the 2023 Formula 1 season. Our comprehensive analysis spanned multiple aspects of the sport, from individual driver performance and tire strategies to the influence of weather conditions on race dynamics.

Insights and Impact: The queries conducted revealed:

- The average number of laps per Grand Prix can inform race strategies and pit stop planning.
- Tire compound usage rates can help teams optimize tire strategies for different tracks and conditions.
- The density of drivers per event can influence the competitiveness and excitement of a race.
- Driver lap completion rates are crucial for assessing reliability and potential technical issues.
- Telemetry data provides a deep dive into driving styles and can influence car setup and driver training.

Furthermore, the challenges faced during the project, such as data integrity issues and the necessity for query optimization, were successfully overcome. These solutions not only enhanced the accuracy of our analysis but also enriched our understanding of the capabilities and limitations of the tools used.

Future Directions: Our project serves as a testament to the potential of big data in sports analytics. The methodologies and processes established here can be extended to other sports disciplines. There's an opportunity to integrate real-time data analysis for dynamic strategic decision-making during races. Additionally, machine learning could be employed to predict outcomes based on historical and live data streams, providing an even richer set of insights for teams and fans alike.

Final Thoughts: The intersection of big data and sports represents a fertile ground for innovation. This project has not only contributed to the field of sports analytics but also demonstrated the practical applications of big data technologies in real-world scenarios. The findings can aid teams in making data-driven decisions that refine performance and strategies, ultimately elevating the sporting spectacle of Formula 1 racing.

As the technology landscape evolves, so too will the capabilities for data analysis within the sport. We look forward to seeing how future developments will continue to transform the way teams and fans engage with Formula 1.

8 Project Repository

`https://github.com/enriquegomeztagle/BigData/tree/main/SecondTerm/F1-EDA-Project`

9 References

1. Hadoop: `https://hadoop.apache.org/`
2. Hive: `https://hive.apache.org/`
3. Cloud SQL: `https://cloud.google.com/sql`