Analysis of Formula 1 World Championship

Team member 1:

Chaitanya Krishna Kommineni
ckommine@buffalo.edu
50539270

Team member 2:

Sai Koushik Vimmadisetty
svimmadi@buffalo.edu
50470993

Team member 3:

Deeshma Lavanya Gorrela
deeshmal@buffalo.edu
50470947

Problem Statement:

Formula 1 is a sport in which the technologies, strategies, and even the split-second decisions of the runners often hold the key to success or failure. Besides, tire selection and pit stop timing, every bit of race-day decision-making is carefully crafted so as to ensure maximum performance and competitive advantage. The dataset at hand contains detailed information on every aspect of the sport—drivers, constructors, races, lap times, pit stops, qualifying sessions, and more—spanning from the first Formula 1 season in 1950 through to the 2024 season.

The model is to forecast F1 results which will be a major part of the research. It will try to link the key features of the racing day with the winning probability. The data collected about the previous races, which will include driver performance, track conditions, lap times, pit stops, and other race-related factors, as well as the methods the drivers use in the race will be the foundation of our research in finding the connection between them and the correct strategies to be implemented. This model will apply the scientific approach, for example, testing of pit stop times and tire options be one of the scenarios, and furthermore, positively impact the race. Our core objectives are actionable insights for Formula 1 teams and race strategists to come up with alternatives that way around to push issues that arise towards the pits or not.

Through this project, our objective is to develop a system for all F1 teams that will be able to predict the end of the race more precisely. Thus, they will be able to change strategies while a race is going on, which will, in turn, result in improved performance on race day. In this regard, the predictive model will be a pride achievement for the teams who are interested in elaborating their competitive spirit. Likewise, such teams will be able to come up with more successful strategies for the Formula 1 setting which, as always, will be forever changing.

Target Users:

Formula 1 teams and engineers are the faces of technological innovation and strategic planning in the world of motor sports. These F1 teams include: McLaren, Red Bull racing, Ferrari, Mercedes, Aston Martin, RB, Haas, etc.

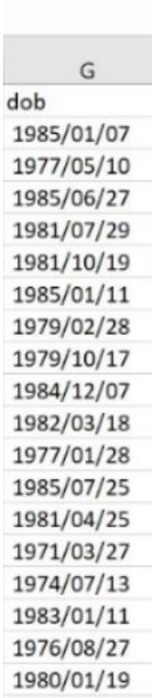
Each of these teams consists of a diverse group of professionals, including race engineers, aerodynamics, data analysts, and mechanics, all working together in order to maximize the performance of their particular cars on race day.

Using this predictive model, F1 strategists and engineers can analyze how parameters such as pit stop timing, tire type, and track-specific conditions really do to a race outcome and take the necessary steps. This approach could also help them to predict the results of possible race winning strategies by enabling them to adapt and make real-time decisions in order to optimize their race performance.

In particular, this model can help:

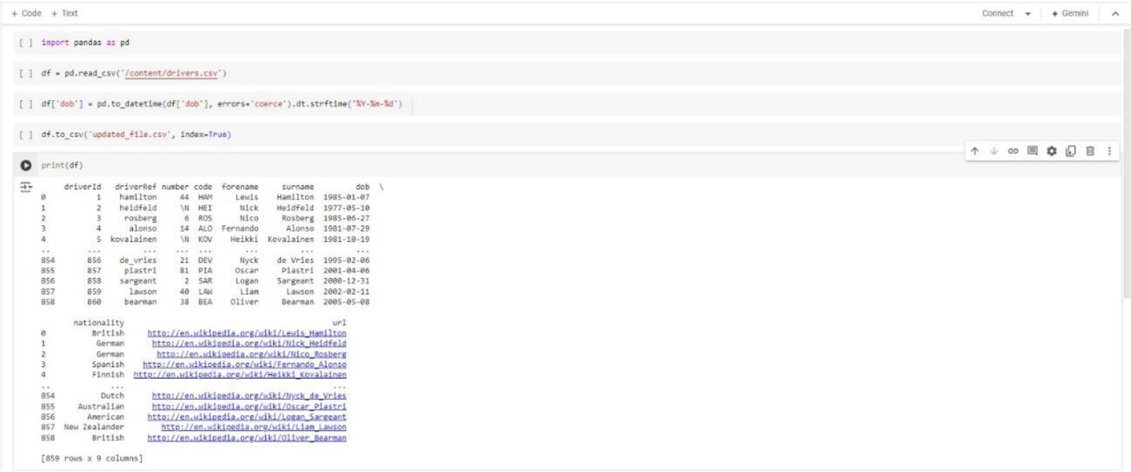
- Engineers of each team can identify patterns in car performance under different track conditions and can make adjustments that align with the model's recommendations.
- Race strategists can use this model to simulate potential scenarios and select the optimal strategies for race day, including timing of pit stops, tire selections, etc.
- Data analysts and Sports analysts can use these insights from the data to explore trends which can consist of individual driver performances and the factors that are responsible for winning the races.

Data Acquisition and Pre-processing



G
dob
1985/01/07
1977/05/10
1985/06/27
1981/07/29
1981/10/19
1985/01/11
1979/02/28
1979/10/17
1984/12/07
1982/03/18
1977/01/28
1985/07/25
1981/04/25
1971/03/27
1974/07/13
1983/01/11
1976/08/27
1980/01/19

In the raw dataset, the dob (date of birth) column was formatted as dd/mm/yyyy. Since PostgreSQL accepts dates in the yyyy-mm-dd format, the date format was converted to this standard during preprocessing. The following Python code was used to read the CSV file, transform the dob column to the desired format using pandas, and save the updated dataset:



```
[ ] import pandas as pd
```

```
[ ] df = pd.read_csv('/content/drivers.csv')
```

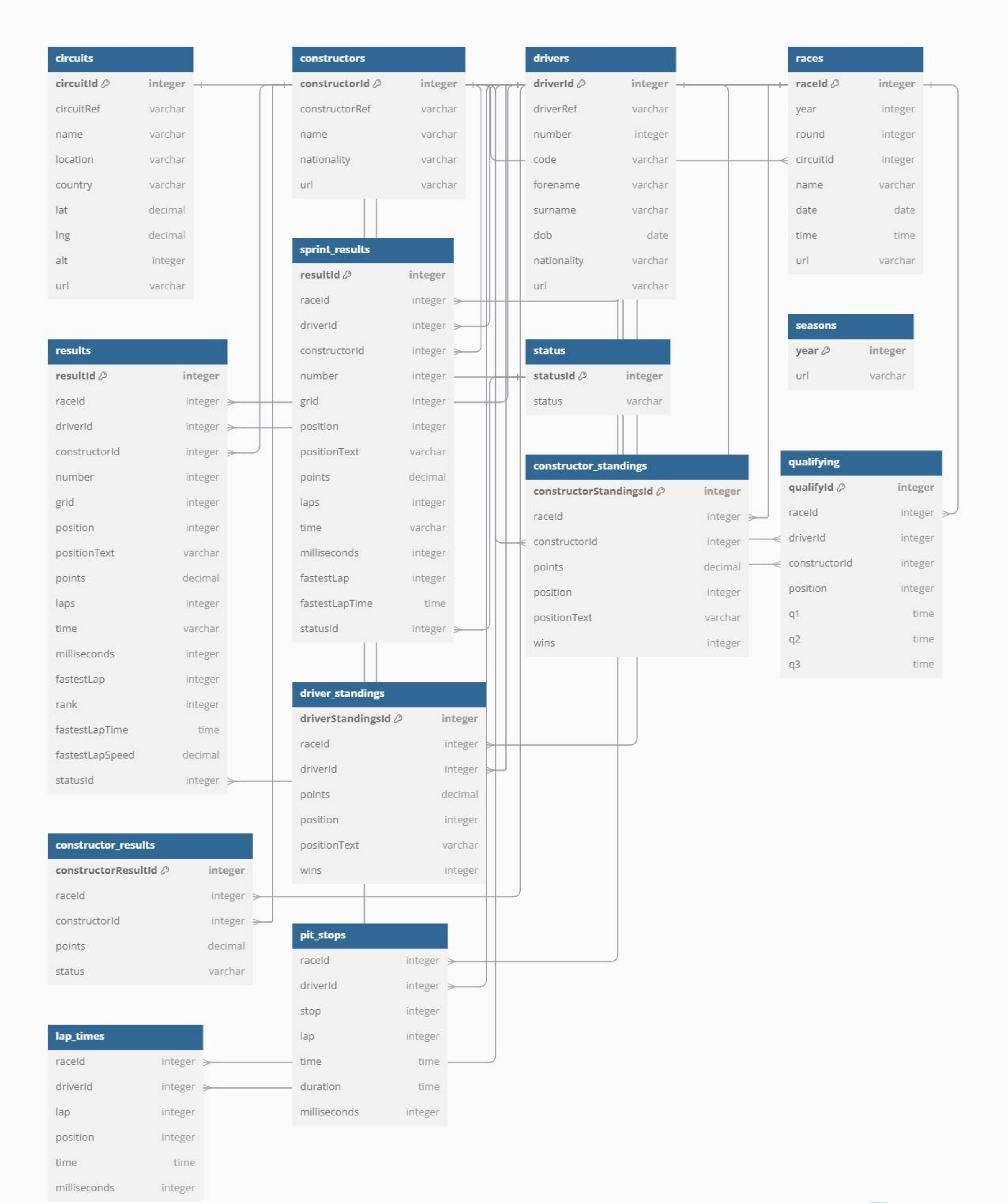
```
[ ] df['dob'] = pd.to_datetime(df['dob'], errors='coerce').dt.strftime('%Y-%m-%d')
```

```
[ ] df.to_csv('updated_file.csv', index=True)
```

```
print(df)
```

```
     driverId   driverRef number code  forename     surname         dob  \
0           1    hamilton     44  HAM     Lewis    Hamilton  1985-01-07
1           2    heidfeld     \N  HEI      Nick    Heidfeld  1977-05-10
2           3     rosberg      6  ROS      Nico     Rosberg  1985-06-27
3           4      alonso     14  ALO  Fernando      Alonso  1981-07-29
4           5  kovalainen     \N  KOV    Heikki  Kovalainen  1981-10-19
..        ...         ...    ...  ...       ...         ...         ...
854       856     de_vries     21  DEV      Nyck    de Vries  1995-02-06
855       857     piastri     81  PIA     Oscar     Piastri  2001-04-06
856       858    sargeant      2  SAR     Logan    Sargeant  2000-12-31
857       859      lawson     40  LAW      Liam      Lawson  2002-02-11
858       860     bearman     38  BEA    Oliver     Bearman  2005-05-08

       nationality                                          url
0          British     http://en.wikipedia.org/wiki/Lewis_Hamilton
1           German      http://en.wikipedia.org/wiki/Nick_Heidfeld
2           German       http://en.wikipedia.org/wiki/Nico_Rosberg
3          Spanish    http://en.wikipedia.org/wiki/Fernando_Alonso
4          Finnish  http://en.wikipedia.org/wiki/Heikki_Kovalainen
..             ...                                             ...
854          Dutch      http://en.wikipedia.org/wiki/Nyck_de_Vries
855     Australian      http://en.wikipedia.org/wiki/Oscar_Piastri
856       American     http://en.wikipedia.org/wiki/Logan_Sargeant
857  New Zealander        http://en.wikipedia.org/wiki/Liam_Lawson
858        British     http://en.wikipedia.org/wiki/Oliver_Bearman

[859 rows x 9 columns]
```

ER Diagram with relations between the tables:



circuits	
circuitId	integer
circuitRef	varchar
name	varchar
location	varchar
country	varchar
lat	decimal
lng	decimal
alt	integer
url	varchar

constructors	
constructorId	integer
constructorRef	varchar
name	varchar
nationality	varchar
url	varchar

drivers	
driverId	integer
driverRef	varchar
number	integer
code	varchar
forename	varchar
surname	varchar
dob	date
nationality	varchar
url	varchar

races	
raceId	integer
year	integer
round	integer
circuitId	integer
name	varchar
date	date
time	time
url	varchar

sprint_results	
resultId	integer
raceId	integer
driverId	integer
constructorId	integer
number	integer
grid	integer
position	integer
positionText	varchar
points	decimal
laps	integer
time	varchar
milliseconds	integer
fastestLap	integer
fastestLapTime	time
statusId	integer

results	
resultId	integer
raceId	integer
driverId	integer
constructorId	integer
number	integer
grid	integer
position	integer
positionText	varchar
points	decimal
laps	integer
time	varchar
milliseconds	integer
fastestLap	integer
rank	integer
fastestLapTime	time
fastestLapSpeed	decimal
statusId	integer

status	
statusId	integer
status	varchar

seasons	
year	integer
url	varchar

constructor_standings	
constructorStandingsId	integer
raceId	integer
constructorId	integer
points	decimal
position	integer
positionText	varchar
wins	integer

qualifying	
qualifyId	integer
raceId	integer
driverId	integer
constructorId	integer
position	integer
q1	time
q2	time
q3	time

driver_standings	
driverStandingsId	integer
raceId	integer
driverId	integer
points	decimal
position	integer
positionText	varchar
wins	integer

constructor_results	
constructorResultId	integer
raceId	integer
constructorId	integer
points	decimal
status	varchar

pit_stops	
raceId	integer
driverId	integer
stop	integer
lap	integer
time	time
duration	time
milliseconds	integer

lap_times	
raceId	integer
driverId	integer
lap	integer
position	integer
time	time
milliseconds	integer

Relationships between the tables:

1. **Circuits – Races**: The relationship is one-to-many. A circuit can host many races, but each race is held at only one circuit.
2. **Constructors – Sprint_Results**: The relationship is one-to-many. A constructor can appear in many sprint results, but a sprint result belongs to only one constructor.
3. **Constructors – Results**: The relationship is one-to-many. A constructor can have multiple race results, but each result corresponds to a single constructor.
4. **Constructors – Constructor_Standings**: The relationship is one-to-many. A constructor can appear in multiple standings across different races, but each constructor standing refers to a single constructor.
5. **Constructors – Constructor_Results**: The relationship is one-to-many. A constructor can have multiple results in races, but each constructor result refers to only one constructor.
6. **Drivers – Sprint_Results**: The relationship is one-to-many. A driver can appear in many sprint results, but a sprint result belongs to only one driver.
7. **Drivers – Results**: The relationship is one-to-many. A driver can participate in many race results, but each result corresponds to a single driver.
8. **Drivers – Driver_Standings**: The relationship is one-to-many. A driver can have multiple standings across different races, but each driver standing refers to a single driver.
9. **Drivers – Qualifying**: The relationship is one-to-many. A driver can participate in many qualifying sessions, but each qualifying session is for one driver.
10. **Drivers – Pit_Stops**: The relationship is one-to-many. A driver can have multiple pit stops during races, but each pit stop corresponds to a single driver.
11. **Races – Sprint_Results**: The relationship is one-to-many. A race can have multiple sprint results, but each sprint result is associated with one race.
12. **Races – Results**: The relationship is one-to-many. A race can have many results, but each result corresponds to one race.
13. **Races – Constructor_Standings**: The relationship is one-to-many. A race can have many constructor standings, but each constructor standing corresponds to one race.
14. **Races – Driver_Standings**: The relationship is one-to-many. A race can have many driver standings, but each driver standing corresponds to one race.
15. **Races – Pit_Stops**: The relationship is one-to-many. A race can have multiple pit stops, but each pit stop is associated with one race.
16. **Races – Lap_Times**: The relationship is one-to-many. A race can have multiple lap times recorded, but each lap time is associated with a single race.
17. **Races – Qualifying**: The relationship is one-to-many. A race can have multiple qualifying sessions, but each qualifying session is for a single race.
18. **Seasons – Races**: The relationship is one-to-many. A season can have multiple races, but each race belongs to only one season.
19. **Status – Results**: The relationship is one-to-many. A result can have one status, but multiple results may share the same status.
20. **Status – Sprint_Results**: The relationship is one-to-many. A sprint result can have one status, but multiple sprint results may share the same status.
21. **Constructor_Results – Status**: The relationship is one-to-many. A constructor result can have one status, but multiple constructor results may share the same status.
22. **Lap_Times – Drivers**: The relationship is one-to-many. A driver can have multiple lap times recorded during a race, but each lap time is for a single driver.

<u>Database constraints:</u>

List of primary and foreign keys for each relation:

1. **Circuits Table**:
   - **Primary Key**: circuitId
   - **Justification**: circuitId uniquely identifies each circuit in the table.

2. **Constructors Table**:
   - **Primary Key**: constructorId
   - **Justification**: constructorId uniquely identifies each constructor in the table.

3. **Drivers Table**:
   - **Primary Key**: driverId
   - **Justification**: driverId uniquely identifies each driver in the table.

4. **Races Table**:
   - **Primary Key**: raceId
   - **Justification**: raceId uniquely identifies each race in the table.

- **Foreign Keys**:

- circuitId references circuitId in the **Circuits** table.
- **Justification**: The circuit where the race takes place is linked to the circuits table.

5. **Seasons Table**:
   - **Primary Key**: year
   - **Justification**: year uniquely identifies each season in the table.

6. **Sprint_Results Table**:
   - **Primary Key**: resultId
   - **Justification**: resultId uniquely identifies each sprint result in the table.

- **Foreign Keys**:
  - raceId references raceId in the **Races** table.
  - driverId references driverId in the **Drivers** table.
  - constructorId references constructorId in the **Constructors** table.
  - statusId references statusId in the **Status** table.
- **Justification**: The sprint results are linked to specific races, drivers, constructors, and statuses.

7. **Results Table**:
   - **Primary Key**: resultId

  - **Justification**: resultId uniquely identifies each race result in the table.

- **Foreign Keys**:

  - raceId references raceId in the **Races** table.
  - driverId references driverId in the **Drivers** table.
  - constructorId references constructorId in the **Constructors** table.
  - statusId references statusId in the **Status** table.
  - **Justification**: The results are associated with races, drivers, constructors, and statuses.

8. **Status Table**:
   - **Primary Key**: statusId
   - **Justification**: statusId uniquely identifies each status in the table.

9. **Constructor_Standings Table**:
   - **Primary Key**: constructorStandingsId
   - **Justification**: constructorStandingsId uniquely identifies each constructor standing in the table.

- **Foreign Keys**:

  - raceId references raceId in the **Races** table.
  - constructorId references constructorId in the **Constructors** table.

  - **Justification**: The constructor standings are linked to specific races and constructors.

10. **Driver_Standings Table**:
    - **Primary Key**: driverStandingsId
    - **Justification**: driverStandingsId uniquely identifies each driver standing in the table.

- **Foreign Keys**:

  - raceId references raceId in the **Races** table.
  - driverId references driverId in the **Drivers** table.
  - **Justification**: The driver standings are linked to specific races and drivers.

11. **Constructor_Results Table**:
    - **Primary Key**: constructorResultId
    - **Justification**: constructorResultId uniquely identifies each constructor result in the table.

- **Foreign Keys**:

- constructorId references constructorId in the **Constructors** table.
- statusId references statusId in the **Status** table.
- **Justification**: The constructor results are associated with constructors and statuses.

12. **Lap_Times Table**:
    - **Composite Primary Key**: (raceId, driverId, lap, position)
    - **Justification**: The combination of raceId, driverId, lap, and position uniquely identifies each lap time.

- **Foreign Keys**:

    - raceId references raceId in the **Races** table.
    - driverId references driverId in the **Drivers** table.
    - **Justification**: The lap times are associated with specific races and drivers.

13. **Pit_Stops Table**:
    - **Composite Primary Key**: (raceId, driverId, stop)
    - **Justification**: The combination of raceId, driverId, and stop uniquely identifies each pit stop.

1. **Foreign Keys**:

    - raceId references raceId in the **Races** table.
    - driverId references driverId in the **Drivers** table.
    - **Justification**: Pit stops are linked to specific races and drivers.

14. **Qualifying Table**:

1. Primary Key: qualifyId
2. Justification: qualifyId uniquely identifies each qualifying result in the table.

- Foreign Keys:

    - raceId references raceId in the **Races** table.
    - driverId references driverId in the **Drivers** table.
    - constructorId references constructorId in the **Constructors** table.
    - **Justification**: The qualifying results are linked to races, drivers, and constructors.

<u>Description of Attributes:</u>

Table Name	Attribute Name	Purpose	Datatype
Circuits	circuitId	Unique identifier for each circuit.	Integer
	circuitRef	Reference code for the circuit.	Varchar
	Name	Name of the circuit.	Varchar
	Location	Location of the circuit.	Varchar
	Country	Country where the circuit is located.	Varchar
	Lat	Latitude of the circuit's location.	Decimal
	Lng	Longitude of the circuit's location.	Decimal
	Alt	Altitude of the circuit.	Integer
	url	Web link for more details about the circuit.	varchar
Constructors	constructorId	Unique identifier for each constructor (team).	Integer
	constructorRef	Reference code for the constructor.	Varchar
	Name	Name of the constructor.	Varchar
	Nationality	Nationality of the constructor.	Varchar
	url	Web link for more details about the constructor.	varchar
Drivers	driverId	Unique identifier for each driver.	Integer
	driverRef	Reference code for the driver.	Varchar
	Number	Driver's racing number.	Integer
	Code	Short code for the driver.	Varchar
	Forename	Driver's first name.	Varchar
	Surname	Driver's last name.	Varchar
	Dob	Date of birth of the driver.	Date
	Nationality	Nationality of the driver.	Varchar
	url	Web link for more details about the driver.	Varchar

Races	raceId	Unique identifier for each race.	Integer
	Year	Year when the race was held.	Integer
	Round	The round number of the race in the season.	Integer
	circuitId	ID of the circuit where the race was held.	Integer
	Name	Name of the race.	Varchar
	Date	Date when the race took place.	Date
	Time	Time when the race started.	Time
	url	Web link for more details about the race.	Varchar
Seasons	Year	Year representing the season.	Integer
	url	Web link for more details about the season.	varchar
Sprint_Results	resultsId	Unique identifier for each sprint result.	Integer
	raceId	ID of the race associated with the sprint.	Integer
	driverId	ID of the driver in the sprint.	Integer
	constructorId	ID of the constructor associated with the sprint.	Integer
	Number	Racing number of the driver.	Integer
	Grid	Starting position of the driver in the sprint.	Integer
	Position	Finishing position of the driver.	Integer
	positionText	Text description of the finishing position.	Varchar
	Points	Points earned by the driver.	Decimal
	Laps	Number of laps completed.	Integer

	Time	Total time taken by the driver.	Varchar
	Milliseconds	total time taken in milliseconds.	Integer
	fastestLap	Fastest lap achieved by the driver.	Integer
	fastestLapTime	Time of the fastest lap.	Time
	Statusid	ID representing the driver's status in the sprint.	Integer
Results	resultId	Unique identifier for each race result.	Integer
	raceId	ID of the race associated with the result.	Integer
	driverId	ID of the driver in the race.	Integer
	constructorId	ID of the constructor associated with the result.	Integer
	Number	Racing number of the driver.	Integer
	Grid	Starting position of the driver.	Integer
	Position	Finishing position of the driver.	Integer
	positionText	Text description of the finishing position.	Varchar
	Points	Points earned by the driver.	Decimal
	Laps	Number of laps completed.	Integer
	Time	Total race time.	Varchar
	Milliseconds	Total race time in milliseconds.	Integer
	fastestLap	Fastest lap achieved by the driver.	Integer
	rank	Rank of the fastest lap.	
	fastestLapTime	Time of the fastest lap.	Time

	fastestLapSpeed	Speed achieved during the fastest lap.	Decimal
	statusId	ID representing the driver’s status in the race.	Integer
Status	statusId	Unique identifier for each status.	Integer
	Status	Description of the driver’s status (e.g., finished, retired).	varchar
Constructor_Standings	constructorStandingsId	Unique identifier for each constructor standing.	Integer
	raceId	ID of the race associated with the constructor standing.	Integer
	constructorId	ID of the constructor.	Integer
	Points	Points earned by the constructor.	Decimal
	Position	Constructor’s position in the standings.	Integer
	positionText	Text description of the position.	Varchar
	Wins	Number of wins by the constructor.	Integer
Driver_Standings	driverStandingsId	Unique identifier for each driver standing.	Integer
	raceId	ID of the race associated with the driver standing.	Integer
	driverId	ID of the driver.	Integer
	Points	Points earned by the driver.	Decimal
	Position	Driver’s position in the standings.	Integer
	positionText	Text description of the position.	Varchar
	Wins	Number of wins by the driver.	Integer
	constructorResultId	Unique identifier for each constructor result.	Integer
	raceId	ID of the constructor.	Integer

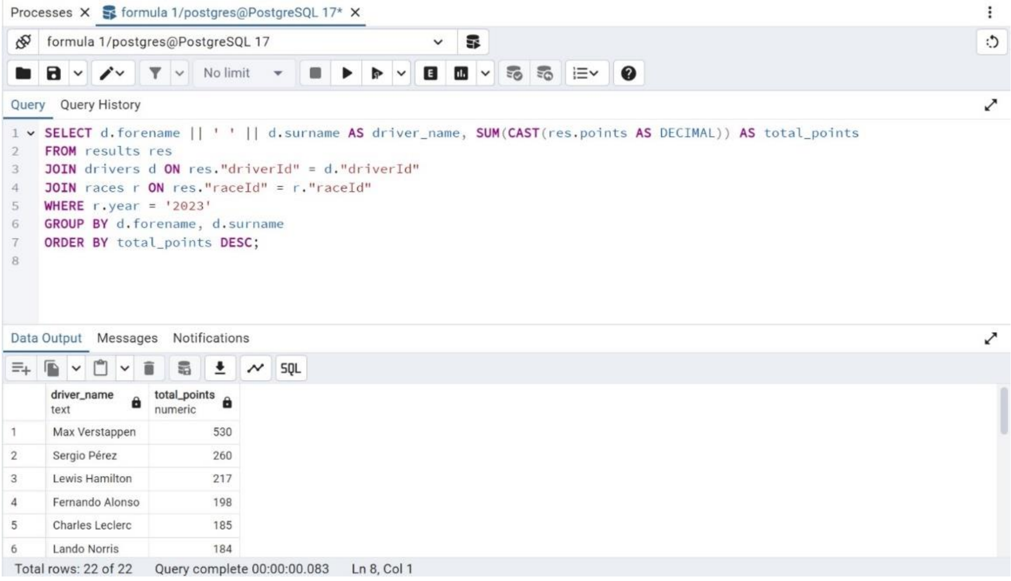
Constructor_results	constructorId	ID of the race associated with the constructor result.	Integer
	Points	Points earned by the constructor.	Decimal
	Status	Status description of the constructor in the race.	Varchar
Lap_Times	raceId	ID of the race associated with the lap time.	Integer
	driverId	ID of the driver recording the lap time.	Integer
	Lap	The lap number.	Integer
	Positon	Driver's position during the lap.	Integer
	Time	Total time taken to complete the lap.	Time
	Milliseconds	Total time taken in milliseconds.	Integer
Pit_Stops	raceId	ID of the race associated with the pit stop.	Integer
	driverId	ID of the driver making the pit stop.	Integer
	Stop	The stop number (i.e., the nth stop during the race).	Integer
	Lap	The lap during which the pit stop occurred.	Integer
	Time	Time when the pit stop was made.	Time
	Duration	Total duration of the pit stop.	Time
	Milliseconds	Duration of the pit stop in milliseconds.	Integer
	qualifyId	Unique identifier for each qualifying session.	Integer
	raceId	ID of the race associated with the qualifying session.	Integer
	driverId	ID of the driver in the qualifying session.	Integer

Qualifying	constructorId	ID of the constructor associated with the driver.	Integer
	Position	The position achieved by the driver.	Integer
	Q1	Time of the driver’s first qualifying session.	Time
	Q2	Time of the driver’s second qualifying session.	Time
	Q3	Time of the driver’s third qualifying session.	time

SQL Queries executed:

1.Retrieve the race results along with the driver’s name and constructor name for all races in a given year.

Table used: results, drivers, races.

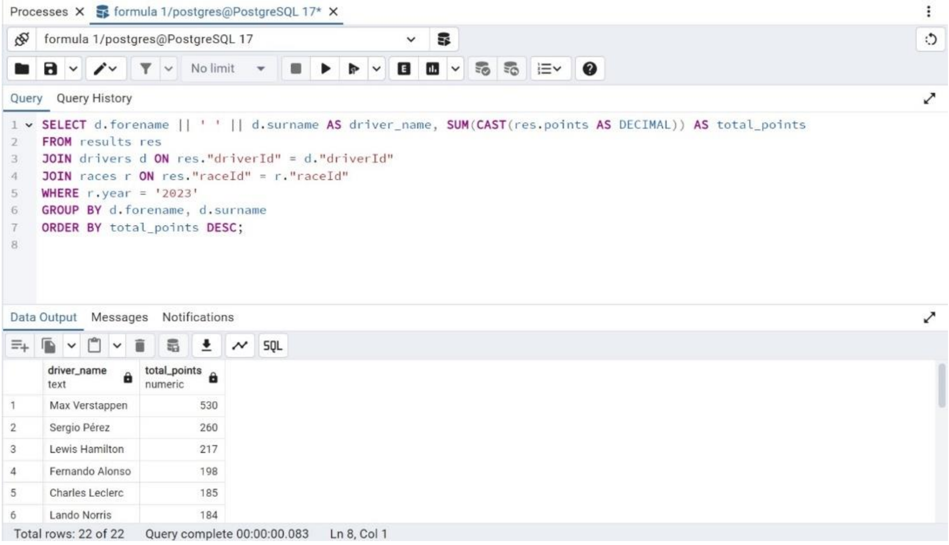


```sql
SELECT d.forename || ' ' || d.surname AS driver_name, SUM(CAST(res.points AS DECIMAL)) AS total_points
FROM results res
JOIN drivers d ON res."driverId" = d."driverId"
JOIN races r ON res."raceId" = r."raceId"
WHERE r.year = '2023'
GROUP BY d.forename, d.surname
ORDER BY total_points DESC;
```

	driver_name text	total_points numeric
1	Max Verstappen	530
2	Sergio Pérez	260
3	Lewis Hamilton	217
4	Fernando Alonso	198
5	Charles Leclerc	185
6	Lando Norris	184

Total rows: 22 of 22   Query complete 00:00:00.083   Ln 8, Col 1

2. Finding the total points scored by each driver in the current season, ordered by the highest points first.
 Tables used: results, drivers, races.

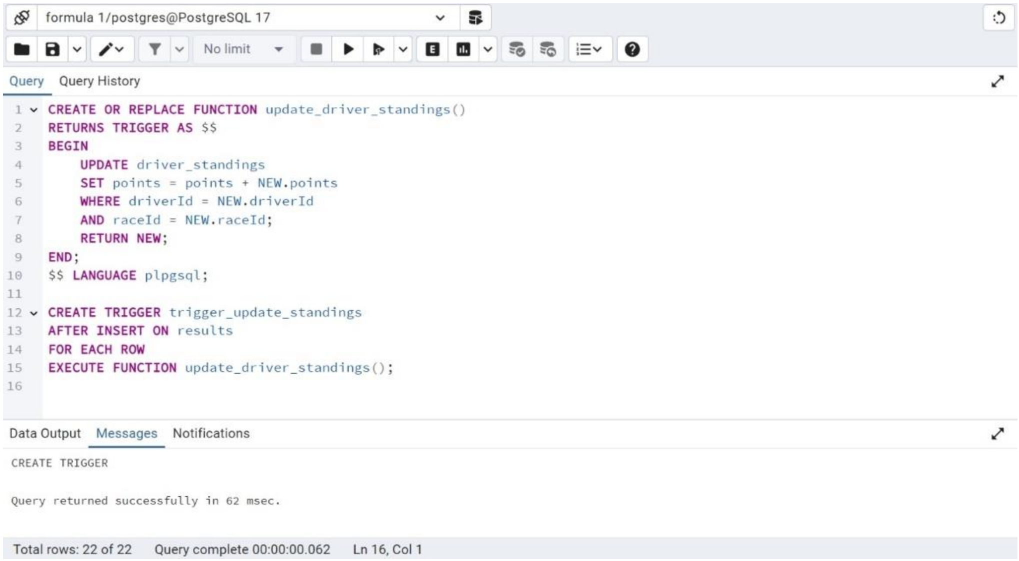


```sql
SELECT d.forename || ' ' || d.surname AS driver_name, SUM(CAST(res.points AS DECIMAL)) AS total_points
FROM results res
JOIN drivers d ON res."driverId" = d."driverId"
JOIN races r ON res."raceId" = r."raceId"
WHERE r.year = '2023'
GROUP BY d.forename, d.surname
ORDER BY total_points DESC;
```

	driver_name text	total_points numeric
1	Max Verstappen	530
2	Sergio Pérez	260
3	Lewis Hamilton	217
4	Fernando Alonso	198
5	Charles Leclerc	185
6	Lando Norris	184

```
Total rows: 22 of 22    Query complete 00:00:00.083    Ln 8, Col 1
```

3. Creating a trigger that automatically updates the driver standings after inserting a new result into the results table.

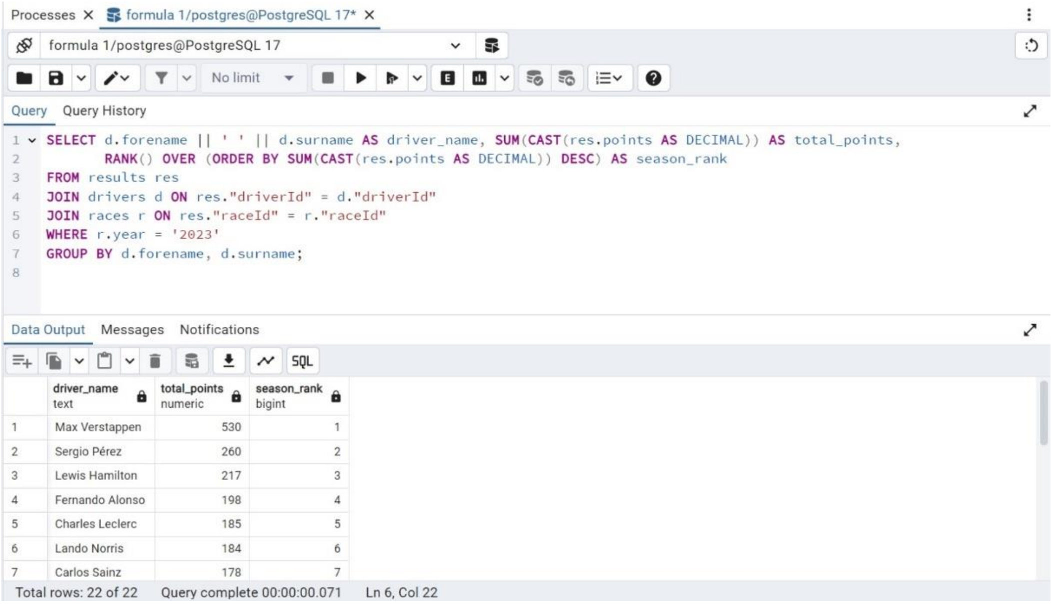


```sql
CREATE OR REPLACE FUNCTION update_driver_standings()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE driver_standings
    SET points = points + NEW.points
    WHERE driverId = NEW.driverId
    AND raceId = NEW.raceId;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_standings
AFTER INSERT ON results
FOR EACH ROW
EXECUTE FUNCTION update_driver_standings();
```

```
CREATE TRIGGER

Query returned successfully in 62 msec.
```

```
Total rows: 22 of 22    Query complete 00:00:00.062    Ln 16, Col 1
```

4. Retrieve each driver's total points and their rank (position) within the season, using a window function.
Tables used: results, drivers, races.

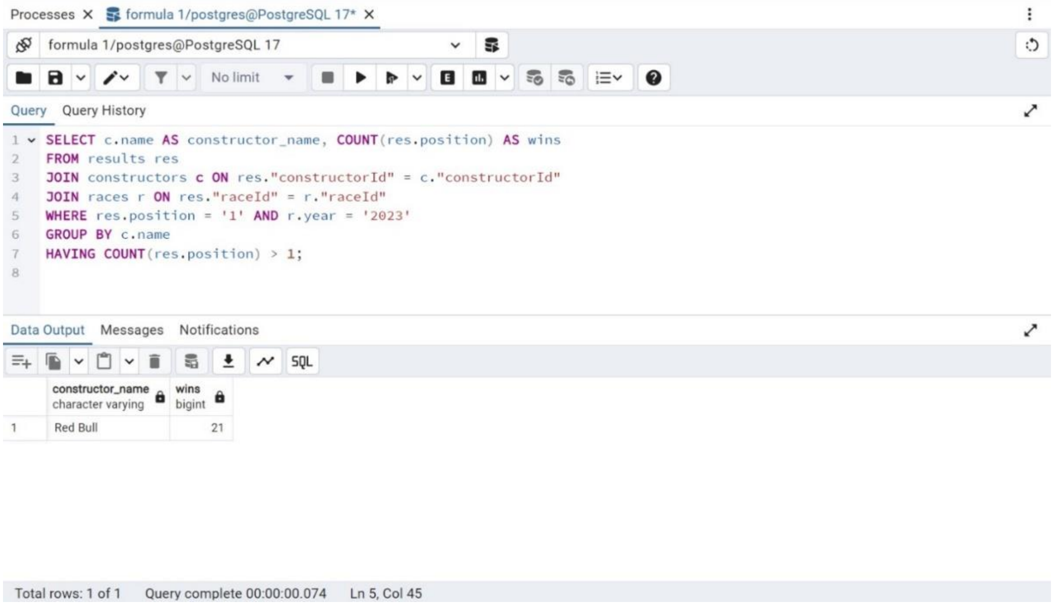


```sql
SELECT d.forename || ' ' || d.surname AS driver_name, SUM(CAST(res.points AS DECIMAL)) AS total_points,
       RANK() OVER (ORDER BY SUM(CAST(res.points AS DECIMAL)) DESC) AS season_rank
FROM results res
JOIN drivers d ON res."driverId" = d."driverId"
JOIN races r ON res."raceId" = r."raceId"
WHERE r.year = '2023'
GROUP BY d.forename, d.surname;
```

	driver_name text	total_points numeric	season_rank bigint
1	Max Verstappen	530	1
2	Sergio Pérez	260	2
3	Lewis Hamilton	217	3
4	Fernando Alonso	198	4
5	Charles Leclerc	185	5
6	Lando Norris	184	6
7	Carlos Sainz	178	7

Total rows: 22 of 22   Query complete 00:00:00.071   Ln 6, Col 22

5. Finding the constructors who won more than one race in a given year.



```sql
SELECT c.name AS constructor_name, COUNT(res.position) AS wins
FROM results res
JOIN constructors c ON res."constructorId" = c."constructorId"
JOIN races r ON res."raceId" = r."raceId"
WHERE res.position = '1' AND r.year = '2023'
GROUP BY c.name
HAVING COUNT(res.position) > 1;
```

	constructor_name character varying	wins bigint
1	Red Bull	21

Total rows: 1 of 1   Query complete 00:00:00.074   Ln 5, Col 45

REFERENCES:

Dataset: https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020