Formula1 Strategy Management App

COMP306 Term Project

Deniz Erdogan - Oya Suran - Ata Sayin - Sema Gure - Altay Atalay

08.06.2022

Project Description

Strategy in motor sports is one of the biggest factors in winning or losing. In modern Formula1, every team has around 5 race strategists on the circuit who are communicating with 20+ other strategists at the teams headquarters. Since they must operate under extreme pressure, divided by milliseconds with their rivals, having the right data available is key. Our web application is a tool which can be used by the strategists themselves or interested Formula1 fans.

Our database consists of tens of thousands of entries from almost every race done in the last around 50 years of racing up to 2018. Using this, one can gain crucial data about the conditions and move accordingly.

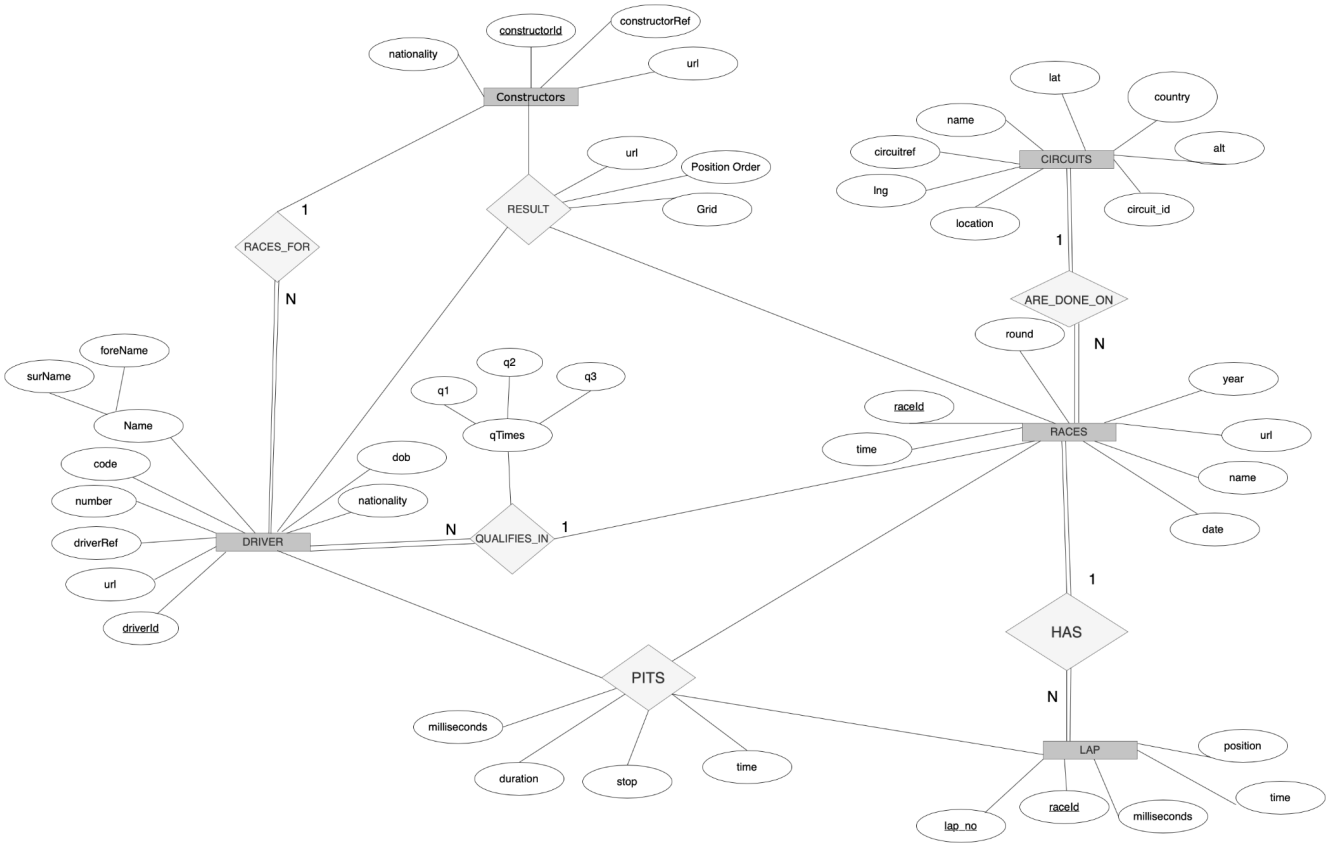
For instance, one of the biggest decisions to be made in modern motor racing is the number of pitstops to make in a race. While making a pitstop loses you crucial time, having fresher tyres gives you better lap times. This trade-off rewards the ones who decide correctly, while ruining the race for the ones who do not. However, when deciding on this, these factors must be taken into account:

- Average time lost in a pit-stop at that specific circuit (pit lanes can have different lengths or speed limits)
- Amount of tire degradation on that circuit (if degradation is higher on a circuit, having fresher tyres gives a greater advantage)
- Ease of overtaking in that specific circuit (if overtaking is easier, drivers lose less time behind slower cars, encouraging pitting more)

Our app can help in these situations by gathering the data in practical ways. For example, for the problem discussed above, one can see the average duration of a pitstop for all races done on that circuit for the last 10 years. Or, one can see the average race finish positions of drivers grouped by the number of pit stops they make.

The structure of the project consists of our database that uses MySQL, our back- end which is Python and Flask, and our front-end which is written in React.js. Also, We are running our database and web application inside docker containers and orchestrate them using docker-compose.

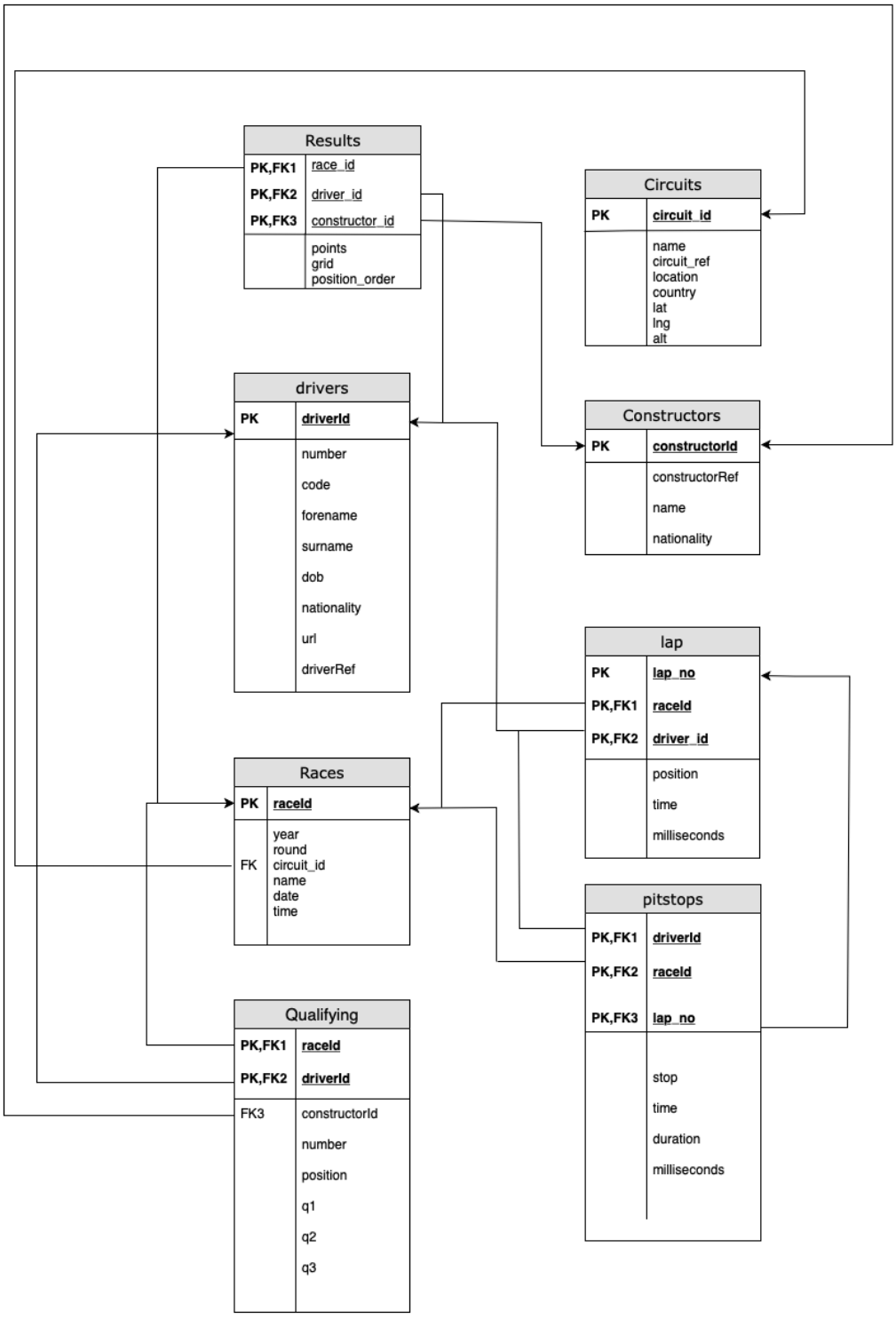
Entity-Relationship Diagram



We have 5 entities: Driver, Races, Lap, Circuits and Constructors. Drivers race for a Constructor. There are two drivers for each constructor. Thus, this relationship is 1 to N. Drivers qualify to participate in a race and determine their grid positions. There are many drivers who qualify in a race. Therefore, this relationship is also 1 to N. Additionally, drivers finish these races in different places which is both their and their constructors' results. This translates to a ternary relationship between Driver, Constructor and Race.

Races take place in Circuits such as ‘Istanbul Park’. There might be multiple races taking place at the same circuit. Therefore this is a 1 to N relation. Races have multiple laps. Finally we have another ternary relation between Driver, Lap and Race, where a driver can make a pitstop multiple times in a single race.

Relational Database



```
CREATE TABLE Races
    (race_id: INTEGER,
    year: INTEGER,
    round: INTEGER,
    circuit_id: INTEGER,
    name: CHAR(50),
    date: DATE,
    time: TIME,
    PRIMARY KEY (race_id),
    FOREIGN KEY circuit_id
        REFERENCES Circuits)

CREATE TABLE Drivers
    (driver_id: INTEGER,
    number: INTEGER,
    code: CHAR(5),
    forename: CHAR(20),
    surname: CHAR(20),
    dob: DATE,
    nationality: CHAR(20),
    url: CHAR(60),
    driver_ref: CHAR(20),
    PRIMARY KEY (driver_id))
```

```
CREATE TABLE Lap
    (race_id: INTEGER,
    lap_no: INTEGER,
    driver_id: INTEGER,
    position: INTEGER,
    time: TIME,
    milliseconds: INTEGER,
    FOREIGN KEY race_id
        REFERENCES Races,
    FOREIGN KEY driver_id
        REFERENCES Drivers,
    PRIMARY KEY (race_id, driver_id, lap_no))

CREATE TABLE Pitstops
    (driver_id: INTEGER,
    race_id: INTEGER,
    lap_no: INTEGER,
    stop: INTEGER,
    time: TIME,
    duration: TIME,
    milliseconds: INTEGER,
    FOREIGN KEY race_id
        REFERENCES Races,
    FOREIGN KEY driver_id
        REFERENCES Drivers,
    PRIMARY KEY (driver_id, race_id, lap_no))
```

```
CREATE TABLE Circuits
    (circuit_id: INTEGER,
    name: CHAR(50),
    circuit_ref: CHAR(20),
    location: CHAR(20),
    country: CHAR(20),
    lat: INTEGER,
    lng: INTEGER,
    PRIMARY KEY (circuit_id))


CREATE TABLE Constructors
    (constructor_id: INTEGER,
    constructor_ref: CHAR(20),
    name: CHAR(20),
    nationality: CHAR(20),
    PRIMARY KEY (constructor_id))
```

```
CREATE TABLE Qualifying
    (race_id: INTEGER,
    driver_id: INTEGER,
    constructor_id: INTEGER,
    number: INTEGER,
    position: INTEGER,
    q1: INTEGER,
    q2: INTEGER,
    q3: INTEGER,
    FOREIGN KEY race_id
        REFERENCES Races,
    FOREIGN KEY driver_id
        REFERENCES Drivers,
    FOREIGN KEY constructor_id
        REFERENCES Constructors,
    PRIMARY KEY (race_id, driver_id, constructor_id))


CREATE TABLE Results
    (race_id: INTEGER,
    driver_id: INTEGER,
    constructor_id: INTEGER,
    points: INTEGER,
    grid: INTEGER,
    position_order: INTEGER,
    FOREIGN KEY race_id
        REFERENCES Races,
    FOREIGN KEY driver_id
        REFERENCES Drivers,
    FOREIGN KEY constructor_id
        REFERENCES Constructors,
    PRIMARY KEY (race_id, driver_id, constructor_id))
```

Result has a composite key with the attributes: race_id, driver_id, constructor_id. This is the case because there might be a race with the second driver of the specified constructor or the driver can change their team. Pitstops also has a composite key with attributes: driver_id, race_id, laps. Since a driver can have pitstops at different laps. Qualifying also has a composite key with attributes: race_id and driver_id. Since a driver can qualify for a race but may not be qualified for another race or might end up in a different grid position.

Data Sources

Our data was found on Kaggle from the following link:
https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020

This dataset contains close to 1 Million entries from many fields like race results to each driver's lap times on each lap. However, due to some inconsistencies, we had to delete some of the recent years.

Complex SQL Queries

1.

```
SELECT Pitstopcount, AVG(position_order)
FROM (Select COUNT(*) as Pitstopcount, surname, position_order FROM PITSTOP,
DRIVERS, RESULTS
WHERE PITSTOP.driver_id = DRIVERS.driver_id AND RESULTS.driver_id =
DRIVERS.driver_id AND
      PITSTOP.race_id = 901 AND RESULTS.race_id = PITSTOP.race_id
GROUP BY DRIVERS.driver_id) as aggregates
GROUP BY Pitstopcount
HAVING Pitstopcount <= 4;
```

This SQL query returns the average race finishing positions of drivers grouped by the number of pitstops they made in a specific race. The input is the id of the race of

interest.

2.

```
SELECT drivers.forename, drivers.surname, drivers.nationality,
SUM(results.points)
        FROM drivers
        JOIN results ON drivers.driver_id = results.driver_id
        GROUP BY drivers.driver_id
        ORDER BY SUM(results.points) DESC
        LIMIT 10;
```

This query return all time highest cumulative point scoring 10 drivers in Formula1.

3.

```
SELECT DRIVERS.forename, DRIVERS.surname, DRIVERS.nationality
FROM DRIVERS
WHERE DRIVERS.driver_id IN
    (SELECT DRIVERS.driver_id
    FROM DRIVERS,RESULTS,RACES
    WHERE DRIVERS.driver_id = RESULTS.driver_id AND RESULTS.position_order <10
AND RESULTS.race_id=RACES.race_id  AND  RACES.race_id IN
        (SELECT RACES.race_id
        FROM RACES
        WHERE RACES.year>2010));
```

This query returns the name, surname and nationality of the drivers who finished in top ten in any race since 2010.

4.

```sql
SELECT DRIVERS.nationality, COUNT(*) as TotalDriverswhoNeverWon
FROM DRIVERS
WHERE DRIVERS.driver_id NOT IN
    (SELECT DRIVERS.driver_id
    FROM DRIVERS,RESULTS
    WHERE DRIVERS.driver_id = RESULTS.driver_id AND RESULTS.position_order =1)
    GROUP BY DRIVERS.nationality;
```

This query returns the country and numbers of drivers from that country who never won a race in Formula1 for each country.

5.

```sql
SELECT DISTINCT(DRIVERS.driver_id), DRIVERS.forename, DRIVERS.surname, RACES.year, DRIVERS.nationality,
Constructors.nationality
FROM DRIVERS, Constructors, RESULTS,RACES
WHERE DRIVERS.nationality = Constructors.nationality AND RESULTS.race_id = RACES.race_id AND
RESULTS.constructor_id = Constructors.constructor_id AND DRIVERS.driver_id = RESULTS.driver_id AND
DRIVERS.driver_id IN
(SELECT DISTINCT(DRIVERS.driver_id)
FROM DRIVERS, RESULTS, RACES
WHERE RESULTS.position_order = 1 AND RESULTS.race_id = RACES.race_id AND RESULTS.driver_id =
DRIVERS.driver_id)
```

This query returns the ids, the names, surnames, year and nationality of all drivers who have a race for a constructor whom they share the same nationality with for each year.

6.

```
SELECT D.forename, D.surname, sum(Re.points) as TotalPoints
FROM Constructors as Co, Results as Re, Drivers as D
WHERE Co.constructor_id = Re.constructor_id
and D.driver_id = Re.driver_id
and Co.constructor_id in (
            SELECT Co.constructor_id
            FROM Constructors as Co, Results as Re
            WHERE Co.constructor_id = Re.constructor_id and Re.position_order =1
            group by Co.constructor_id
            HAVING count(*) > 100
            )
GROUP BY D.driver_id
ORDER BY sum(Re.points) dESC;
```

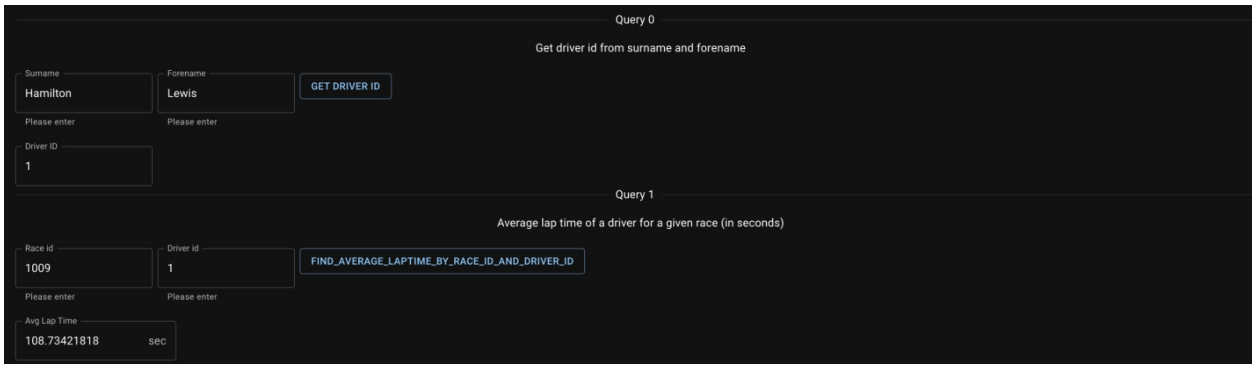
This query returns the total number of cumulative points of each driver who raced for a constructor which has more than 100 wins to its name.

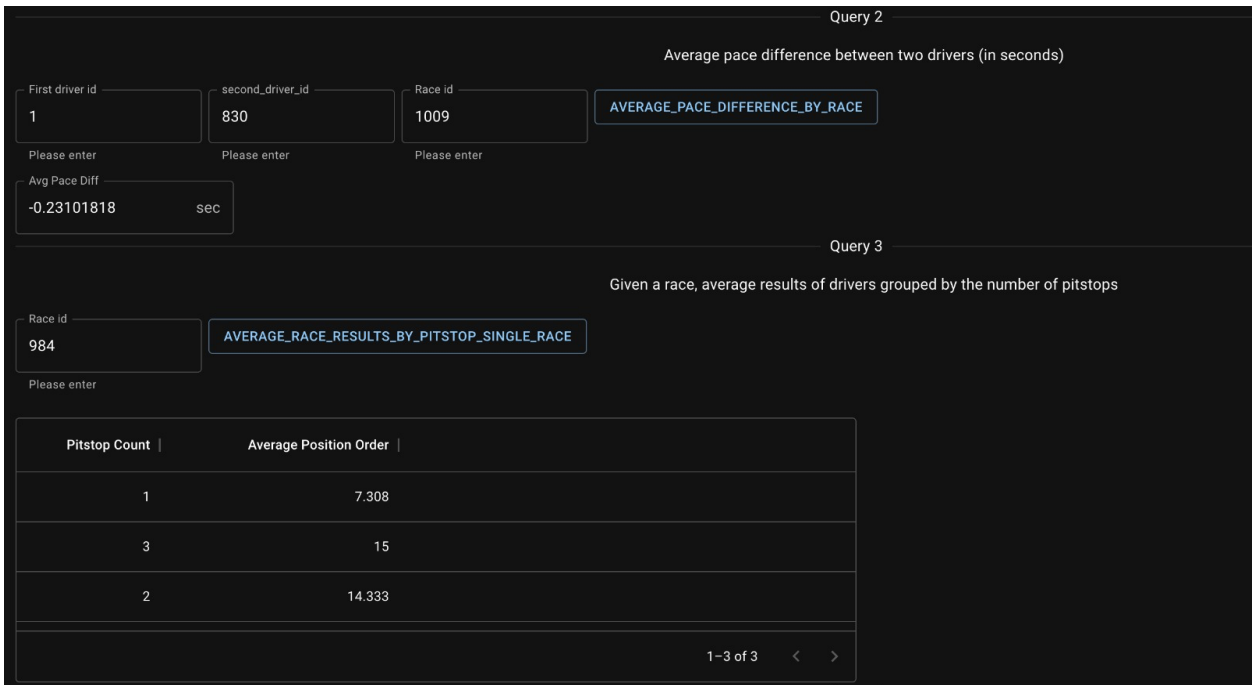
7.

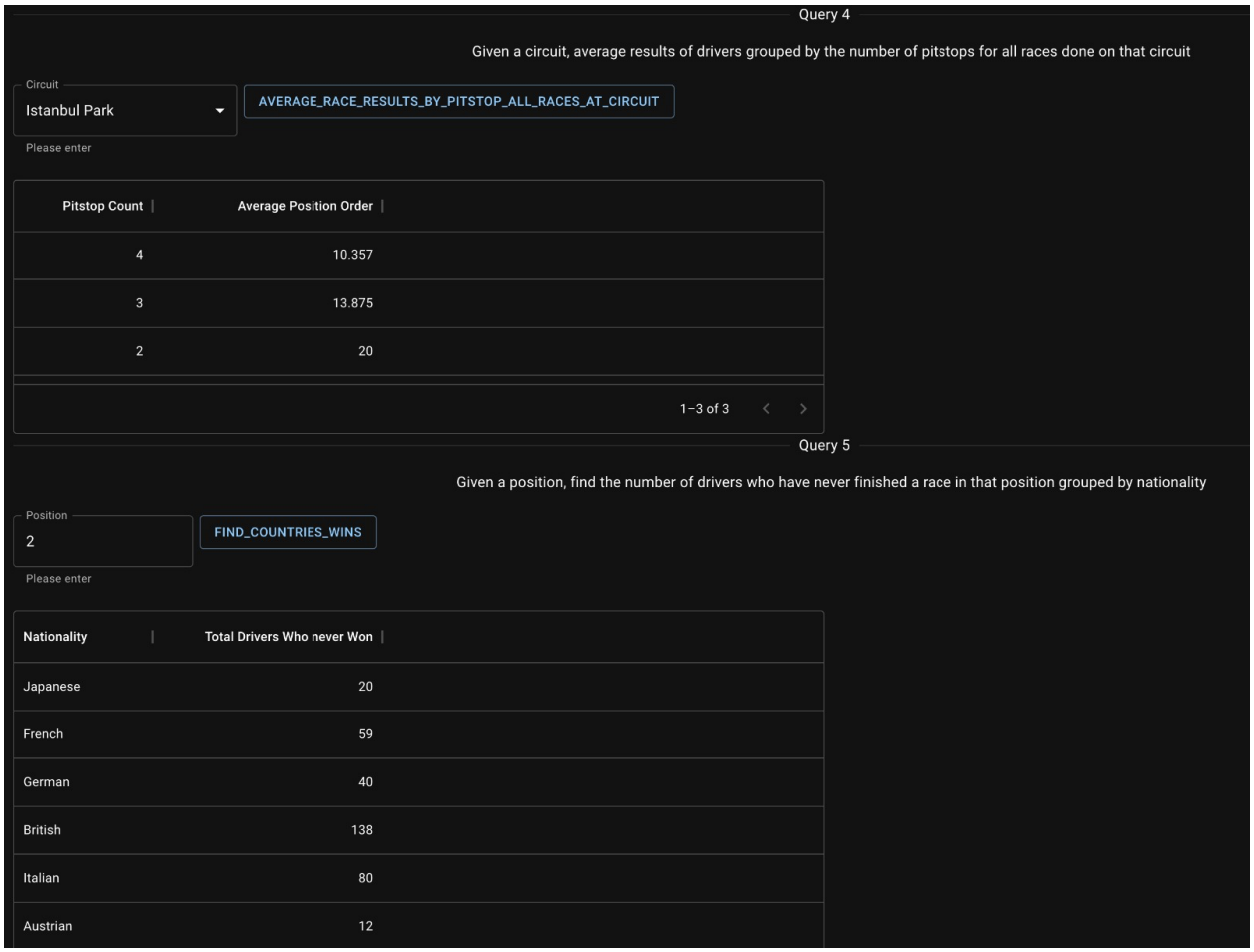
```
SELECT Pitstopcount, AVG(position_order)
FROM (Select COUNT(*) as Pitstopcount, surname,
position_order FROM PITSTOP, DRIVERS, RESULTS,
CIRCUITS, RACES
WHERE PITSTOP.driver_id = DRIVERS.driver_id AND
RESULTS.driver_id = DRIVERS.driver_id AND
     RESULTS.race_id = PITSTOP.race_id AND
RACES.circuit_id = CIRCUITS.circuit_id
    AND RACES.race_id = RESULTS.race_id AND
CIRCUITS.circuit_ref = "Istanbul Park"
GROUP BY DRIVERS.driver_id) as aggregates
GROUP BY Pitstopcount
HAVING Pitstopcount <= 4;
```

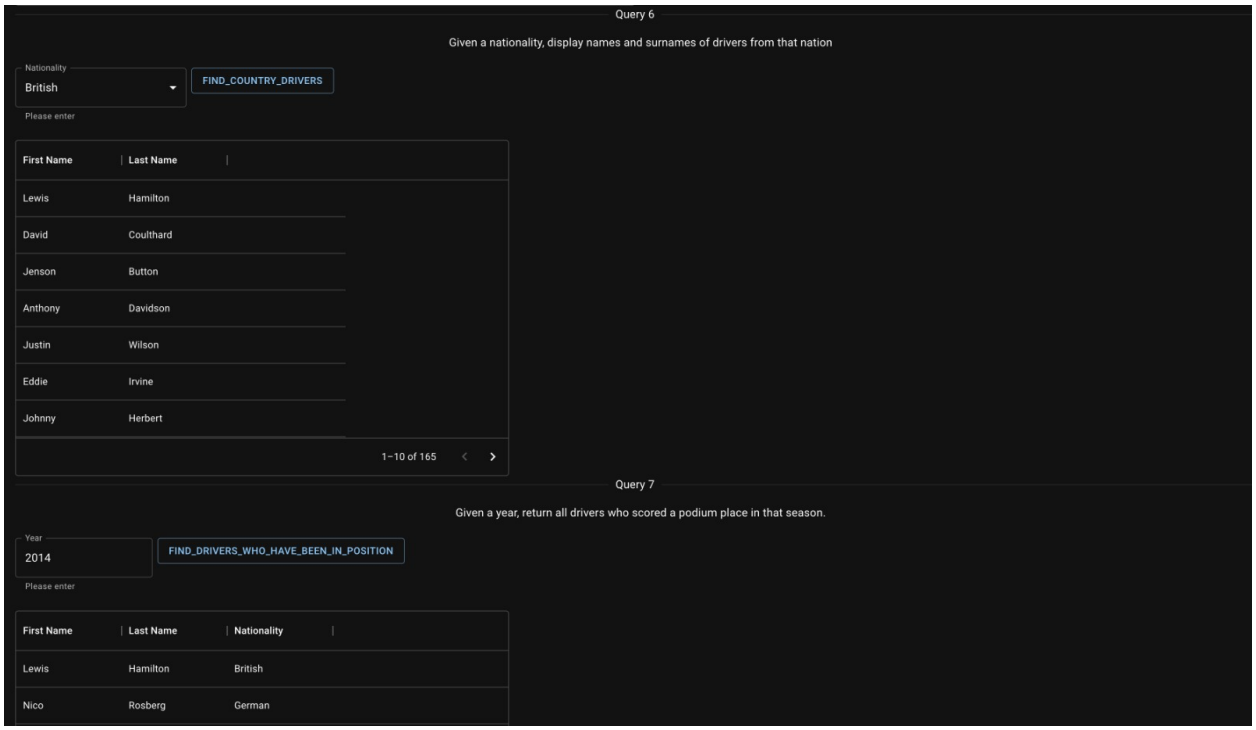
This query returns the average race results grouped by number of pit stops made in all races done on that circuit. The circuit is referenced by its reference name.

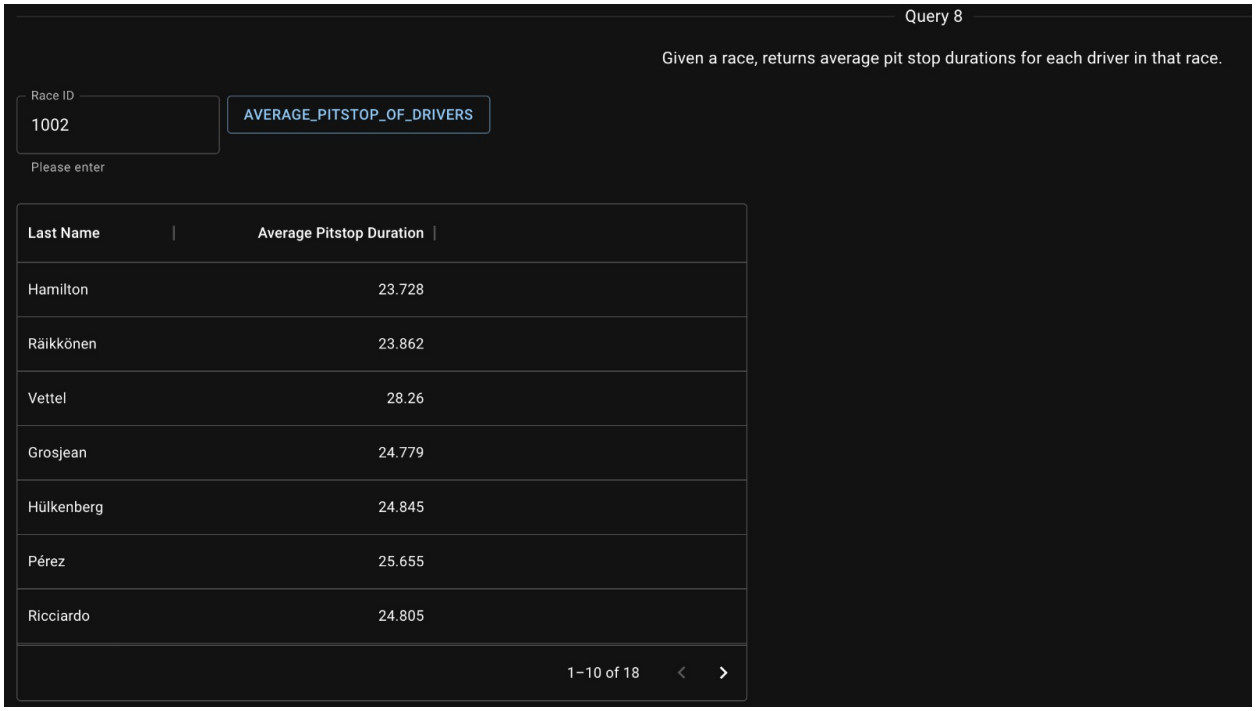
Screenshots

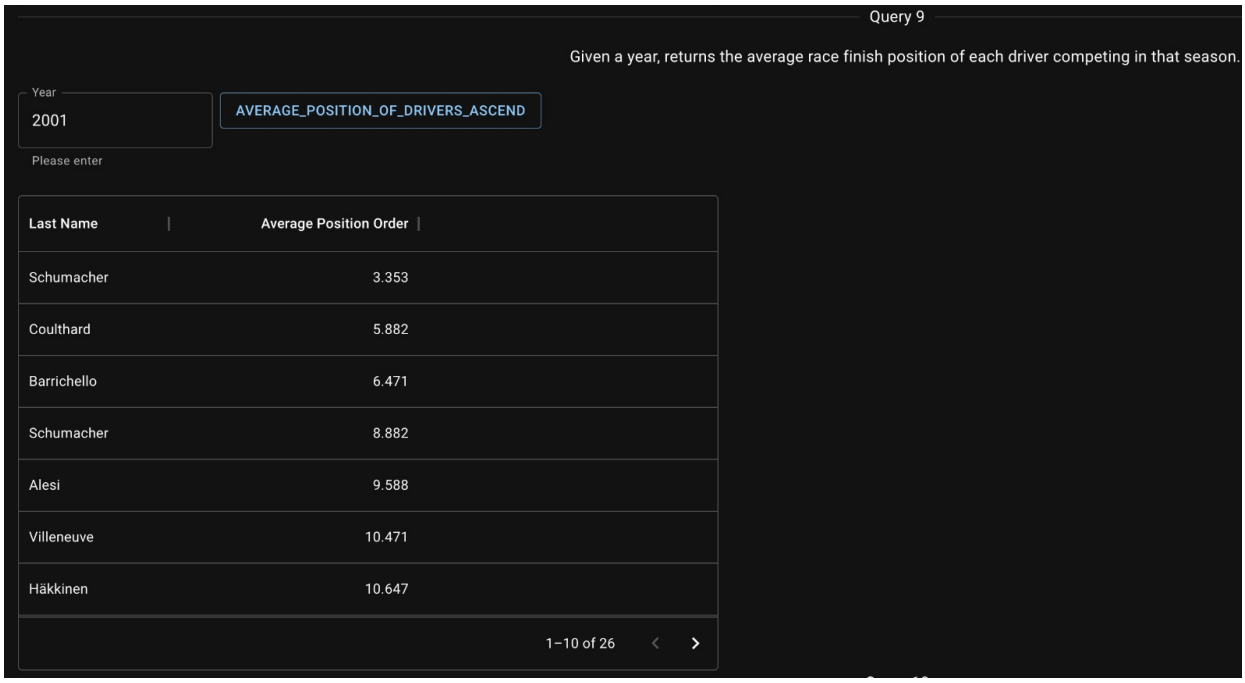





Query 4
Given a circuit, average results of drivers grouped by the number of pitstops for all races done on that circuit
Circuit
Istanbul Park
Please enter
AVERAGE_RACE_RESULTS_BY_PITSTOP_ALL_RACES_AT_CIRCUIT
Pitstop Count
Average Position Order
4
10.357
3
13.875
2
20
1–3 of 3
Query 5
Given a position, find the number of drivers who have never finished a race in that position grouped by nationality
Position
2
Please enter
FIND_COUNTRIES_WINS
Nationality
Total Drivers Who never Won
Japanese
20
French
59
German
40
British
138
Italian
80
Austrian
12


Query 6
Given a nationality, display names and surnames of drivers from that nation
Nationality
British
FIND_COUNTRY_DRIVERS
Please enter
First Name
Last Name
Lewis
Hamilton
David
Coulthard
Jenson
Button
Anthony
Davidson
Justin
Wilson
Eddie
Irvine
Johnny
Herbert
1–10 of 165
Query 7
Given a year, return all drivers who scored a podium place in that season.
Year
2014
FIND_DRIVERS_WHO_HAVE_BEEN_IN_POSITION
Please enter
First Name
Last Name
Nationality
Lewis
Hamilton
British
Nico
Rosberg
German


Query 8
Given a race, returns average pit stop durations for each driver in that race.
Race ID
1002
AVERAGE_PITSTOP_OF_DRIVERS
Please enter
Last Name
Average Pitstop Duration
Hamilton
23.728
Räikkönen
23.862
Vettel
28.26
Grosjean
24.779
Hülkenberg
24.845
Pérez
25.655
Ricciardo
24.805
1–10 of 18


Query 9
Given a year, returns the average race finish position of each driver competing in that season.
Year
2001
Please enter
AVERAGE_POSITION_OF_DRIVERS_ASCEND
Last Name
Average Position Order
Schumacher 3.353
Coulthard 5.882
Barrichello 6.471
Schumacher 8.882
Alesi 9.588
Villeneuve 10.471
Häkkinen 10.647
1–10 of 26