ÇANKAYA UNIVERSITY
FACULTY OF ENGINEERING
SOFTWARE ENGINEERING

SENG 449
Special Topics in Software Engineering

TERM PROJECT

Instructor:

Prof. Dr. Nergiz ERCİL ÇAĞILTAY

15 January 2024

Contents

Figures of Table

1. Introduction

1.1. General information about the chosen operational system

This report analyzes an operational system that contains data related to Formula 1 races. This operational system consists of a database that records and stores the results of races, qualifying rounds, tracks, drivers, teams, pit stops and lap times. This database structures and correlates the data using various tables and fields. This operational system provides detailed information about the history, statistics, and performances of Formula 1 races.

1.2. Description of the problem for the decision-making process

Using the data from this operational system, we aim to develop a decision support system to identify and predict the factors that influence success in Formula 1 races. This decision support system will be used to analyze the performances of the drivers and teams participating in the races, to predict the results of the races, to make strategic decisions such as qualifying laps, pit stops and lap times. This decision support system will be useful to gain competitive advantage and increase the success rate in Formula 1 races.

2. Requirements Analysis

2.1. What the users expect from the developed decision-making system

The potential users of this decision support system are the managers, engineers, strategists, analysts and consultants of the drivers and teams involved in Formula 1 racing. These users expect the decision support system to provide reliable, accurate, up-to-date, and easily accessible data to identify and predict the factors that influence success in Formula 1 racing. They also want the decision support system to provide a user-friendly, flexible, and interactive interface to analyze, visualize, report and share data from different perspectives.

2.2. Surveys, questionnaires, and interview forms/reports

To determine the user requirements of this decision support system, we used questionnaires and interview forms administered to Formula 1 supporters. These questionnaires and forms were designed to measure users' level of knowledge about Formula 1 racing, decision-making processes, data sources, data analysis methods, data visualization tools, data reporting and sharing habits, and their expectations and preferences for the decision support system. The results of these surveys and forms provided important inputs for the design and development of the decision support system. The surveys and forms we used in our report consist of questions asked to Formula 1 supporters in 2020, 2021, 2022. A total of 11,907 people participated in the survey. From a group of 11,907 respondents, the survey results reveal some important trends within the community. Over a two-year period, the proportion of women in the community has experienced a significant increase, from 5.7% in 2020 to 14.8% in 2022, which translates to an impressive annual growth of around 5%. The demographic landscape has changed with an overall aging trend, evidenced by a decline in the 20-29 age group and a simultaneous increase in the 30-49 age group. The importance of the institution of marriage has increased, as the percentage of married individuals in society increased from 17.1% in 2020 to 22.4% in 2022. A parallel trend is observed in the case of education, with a decrease in student representation from 33.5% in 2020. 2020 to 23.8% in 2022, potentially reflecting the maturation of the community. Notably, the survey highlights a growing interest in Formula 1, with the percentage of individuals following its first season increasing from 3.5% in 2020 to 10.2% in 2022. McLaren's share dropped from 73.3% in 2021 to 32.9% in 2022. In terms of viewing platforms, F1TV saw a significant increase from 25.5% in 2021 to 44.5% in 2022. In contrast, Formula E saw a decline. From 40% in 2020 to 22.1% in 2022. IndyCar, on the other hand, witnessed an increase in viewership from 26.7% in 2020 to 34.7% in 2022. Collectively, these insights paint a vivid picture of the dynamic shifts and evolving preferences in this diverse and engaged community. (Reddit, 2022)

2.2.1. Survey results

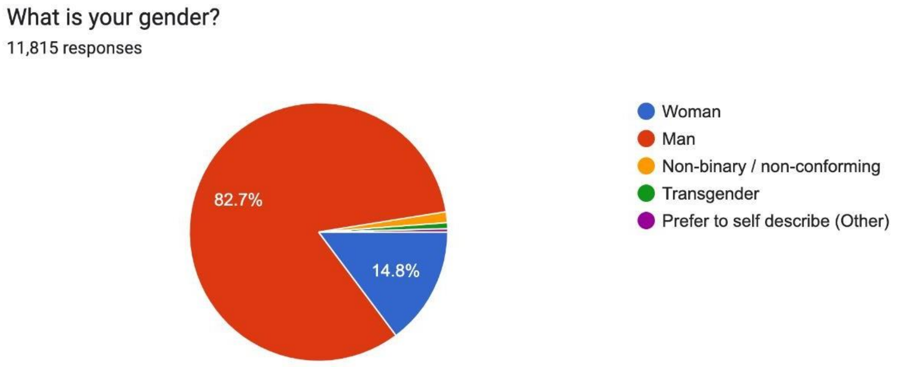


Figure 1: What is your gender?

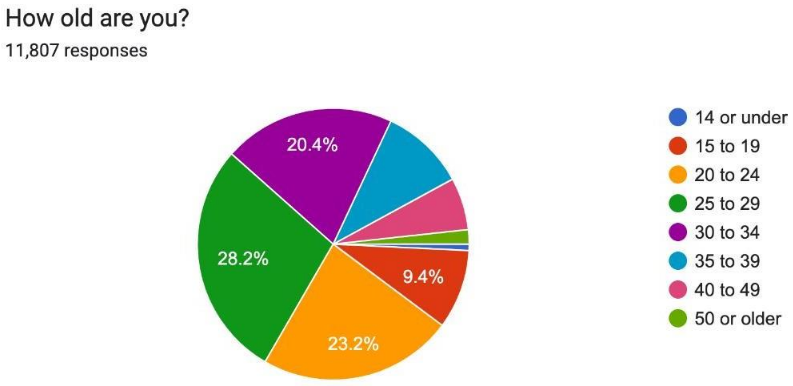


Figure 2: How old are you?

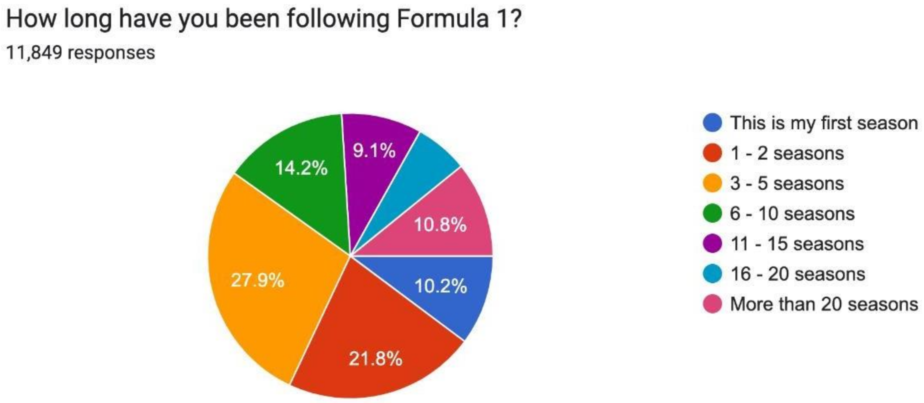


Figure 3: How long have you been following Formula 1?

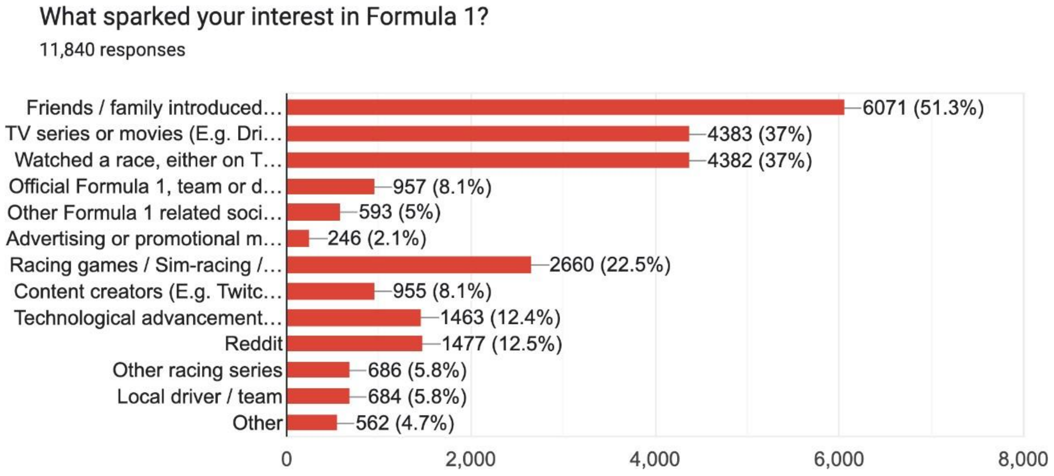


Figure 4: What sparked your interest in Formula 1?

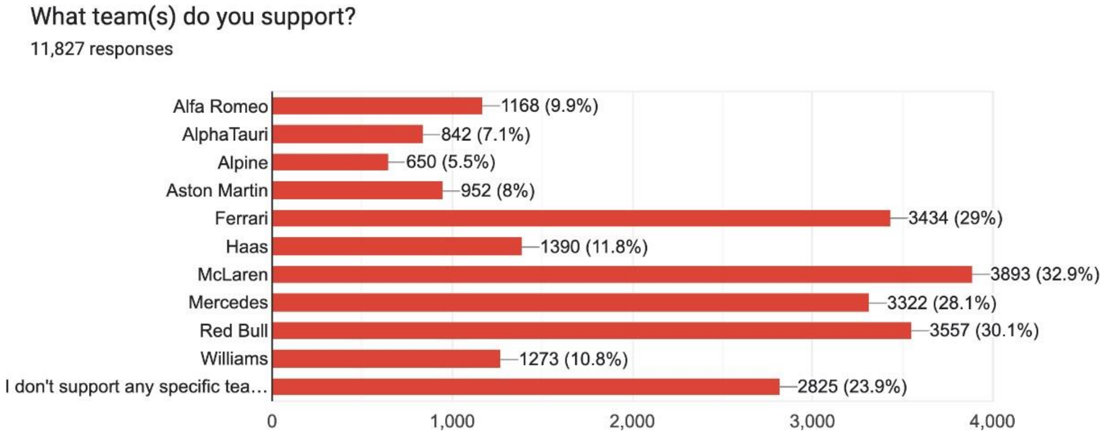


Figure 5: What team(s) do you support?

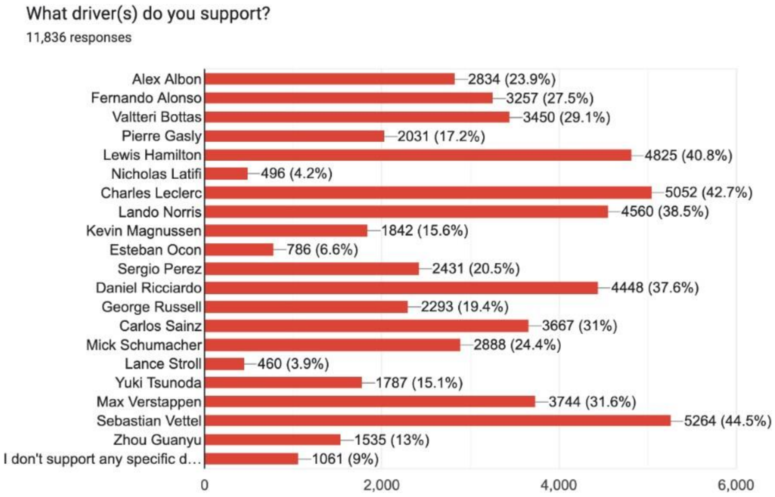


Figure 6: What driver(s) do you support?

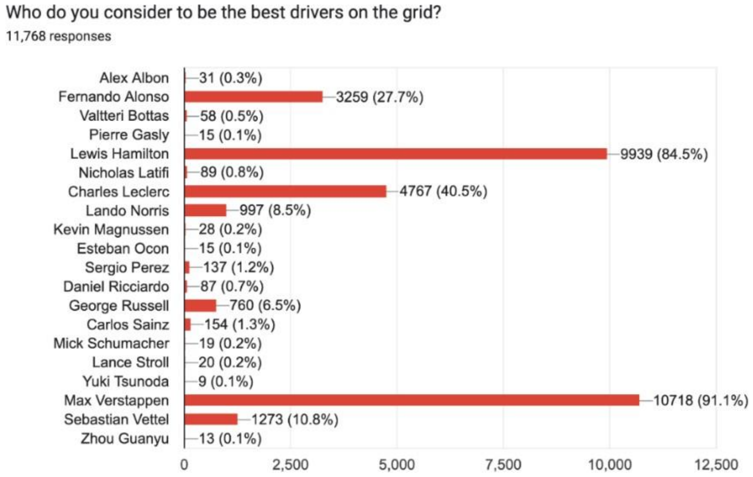


Figure 7: Who do you consider to be the best drivers on the grid?

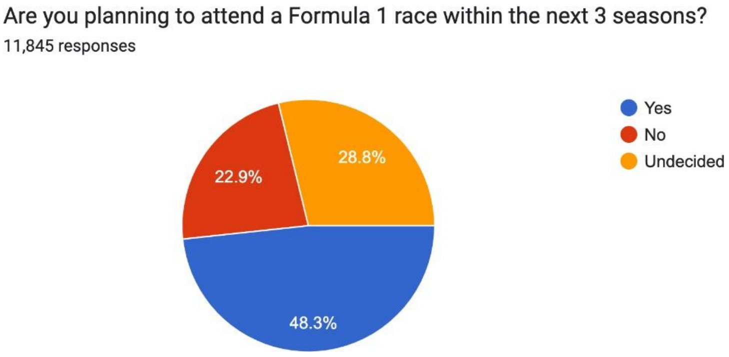


Figure 8: Are you planning to attend a Formula 1 race within the next 3 seasons?

2.3. Conceptual model of the operational systems

The conceptual model of the operational system is a schema diagram showing the structure and content of the database. This diagram shows the tables, fields, and relationships in the database. This diagram provides a reference point for understanding and analyzing the data of the operational system. This diagram is shown in the figure below:

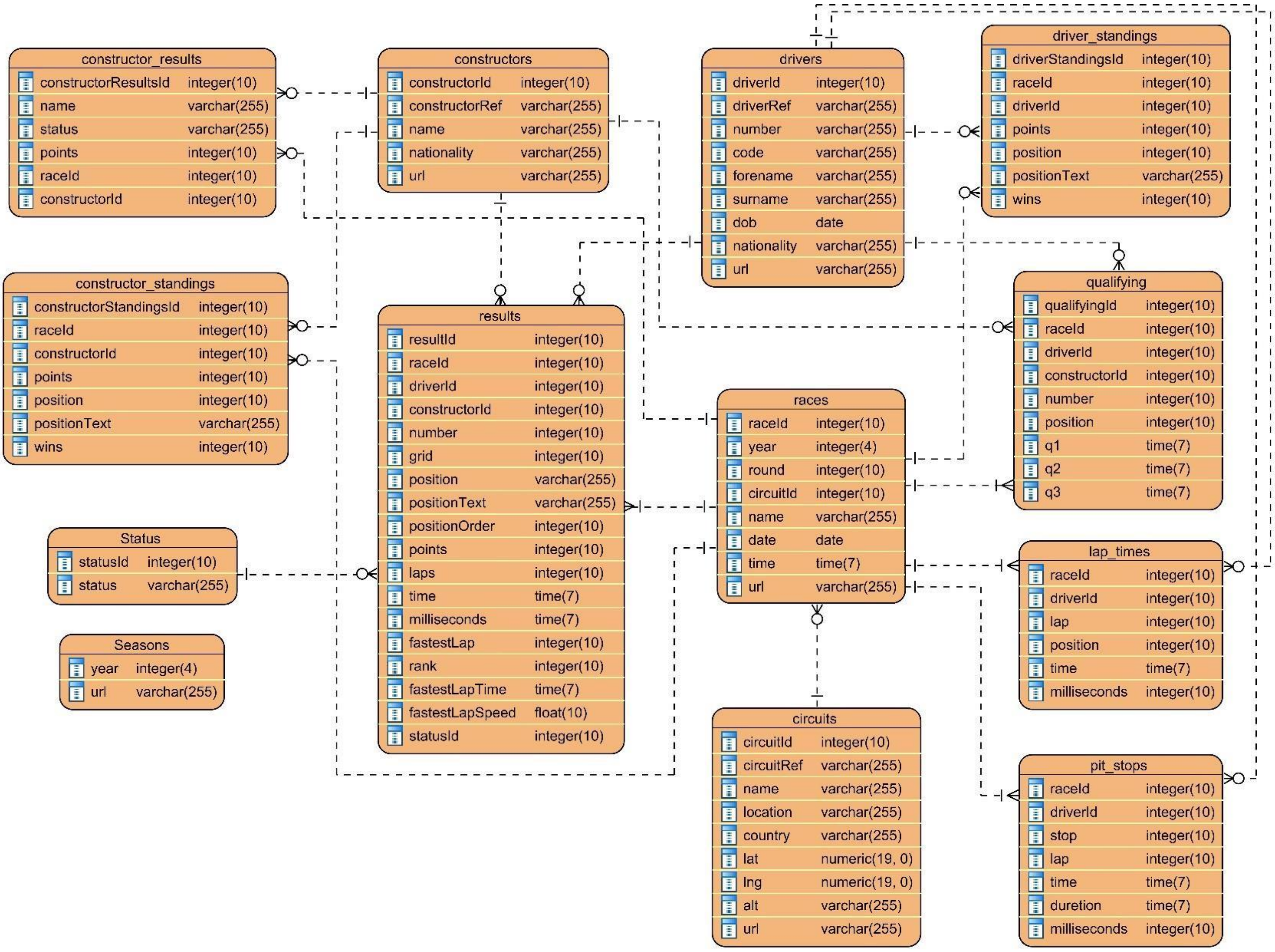


Figure 9: Relational Model

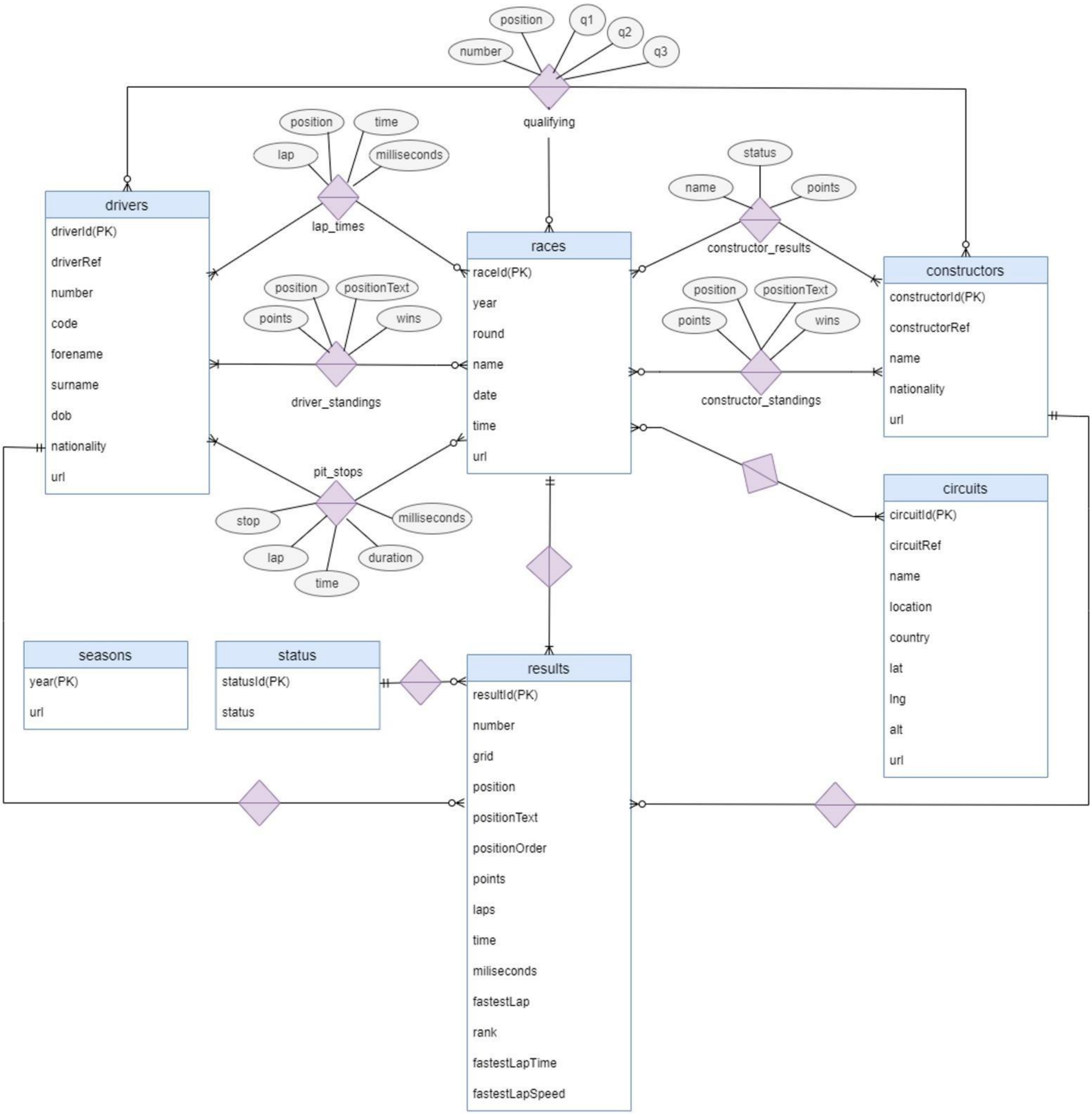


Figure 10: Conceptual Model

2.4. Table names, table column descriptions and number of records in each table

Data tables are a data model that contains various information about Formula 1 racing. The relationships between these tables, how they are used and the number of records in the table are explained in the following section.

2.4.1. **Results Table:** This table shows the results of each race. Each row represents a driver taking part in a race. This table contains fields such as race ID (raceId), driver ID (driverId), team ID (constructorId), positionOrder, points, laps, time. This table is related to the races, drivers and constructor_standings tables. number is the number the driver raced under. Grid shows the qualifying position where the driver started the race. Rank is the driver's position on the fastest lap. Status shows whether the driver completed the race and if not, why not.
(Number of Records: 23777)

2.4.2. **Races Table:** This table shows the details of each race. Each row represents one race. This table contains fields such as raceId, year, round, circuitId, name, date and time. This table is related to the results, qualifying, pit_stops and lap_times tables.
(Number of Records: 997)

2.4.3. **Qualifying Table:** This table shows the results of the qualifying rounds of each race. Each row represents a driver who took part in a qualifying round. This table contains fields such as race ID (raceId), driver ID (driverId), team ID (constructorId), qualifying position (position), best times in the first, second and third rounds (q1, q2, q3). This table is related to the races, drivers and constructor_standings tables.
(Number of Records: 7516)

Q1 (Part 1): The first session for all drivers. Drivers with the slowest lap time in this section are excluded from qualifying.

Q2 (Section 2): The second section for the drivers who passed Q1. The drivers with the slowest lap times in this section are also excluded from qualifying.

Q3 (Section 3): The final section for the best drivers who passed Q2. The driver with the fastest lap time in this section is placed first in the race and usually qualifies to start from row one.

2.4.4. **Circuits Table:** This table shows the details of each runway. Each row represents one circuit. This table contains fields such as the circuit's ID (circuitId), name, location, country, latitude (lat), longitude (long), url (url). This table is related to the races table.
(Number of Records: 73)

2.4.5. **Drivers Table:** This table shows the details of each drive. Each row represents one driver. This table contains fields like driver's ID (driverId), first name (forename), last name (surname), date of birth (dob), nationality (nationality), url (url). This table is related to the results, qualifying, driver_standings and pit_stops tables. Code is the abbreviated name of the driver. For example, Lewis Hamilton's code is HAM. URL is the address of the driver's profile page on the official Formula 1 website. For example, Lewis Hamilton's url is https://www.formula1.com/en/drivers/lewis-hamilton.html.
(Number of Records: 842)

2.4.6. **Driver Standings Table:** This table shows the status of the drivers after each race. This table contains fields such as status id (driverStandingsId), race id (raceId), driver id (driverId), points (points), position (position), position text (positionText), win status (wins). This table is related to races, results, qualifying and drivers’ tables.
(Number of Records: 31726)

2.4.7. **Constructors Table:** This table shows the details of the teams participating in Formula 1 races. Each row represents one team. This table contains fields such as the team's ID (constructorId), name, nationality, url, etc. This table is related to constructor_results, constructor_standings and results tables.
(Number of Records: 208)

2.4.8. **Constructor_Standings Table:** Bu This table shows the status of the teams after each race. Each row represents the status of one team in one race. This table contains fields such as state id (constructorStandingsId), race id (raceId), team id (constructorId), points, position, position text (positionText), win status (wins). This table is related to the races, results, qualifying and constructors' tables.
(Number of Records: 11896)

2.4.9. **Constructor_Results Table:** This table shows the points scored by the teams at the end of each race. Each row represents the points scored by a team in a race. This table contains fields such as race ID (raceId), team ID (constructorId), points, status. This table is related to the races and constructors' tables.
(Number of Records: 11142)

2.4.10. **Pit_Stops Table:** This table shows the pit stops for each race. Each row represents a pit stop made by a driver in a race. This table contains fields such as race ID (raceId), driver ID (driverId), lap, time, duration, milliseconds. This table is related to the races and drivers' tables.
(Number of Records: 6251)

2.4.11. **Lap_Times Table:** This table shows the lap times for each race. Each row represents one lap completed by a driver in a race. This table contains fields such as race ID (raceId), driver ID (driverId), lap, time, milliseconds. This table is related to races and drivers' tables.
(Number of Records: 426633)

2.4.12. **Seasons Table:** This table shows the details of the seasons in which Formula 1 races were held. Each row represents one season. This table contains fields such as the year and url of the season. This table is related to the races table.
(Number of Records: 69)

2.5. Relationships between tables

One-to-many relationships are established between tables using key fields. These relationships ensure the integrity and consistency of the data. We can explain the relationships between tables as follows:

The results table is associated with the races, drivers, constructors, and status tables. The raceId, driverId, constructorId and statusId fields in this table match the key fields of these tables. These relationships allow to combine the results of each race, the details of the race, the driver's information, the team's information and the driver's status.

The races table is related to the circuits table. The circuitId in this table matches the key fields of these tables. These relationships provide the details of each race, the track's information, and the merge.

The qualifying table is related to the races, drivers, and constructors' tables. The raceId, driverId and constructorId fields in this table match the key fields of these tables. These relationships allow to combine the results of the qualifying laps of each race, the details of the race, the driver's information, and the team's information.

The circuits table is related to the races table. The circuitId field in this table matches the key field of the races table. This relationship allows to combine the information of each circuit with the details of the races held at the circuit.

The drivers table is related to the results, qualifying, driver_standings and pit_stops tables. The driverId field in this table matches the key fields of these tables. These relationships allow to combine each driver's information, race results, qualifying laps, driver standings and pit stops.

The driver_standings table is related to the races and drivers’ tables. The raceId, driverId fields in this table match the key fields of these tables. These relationships allow to combine the drivers' statuses after each race, details of the race, race results, qualifying rounds, and driver's information.

The constructor table is related to the results, qualifying, constructor_standings and constructor_results tables. The constructorId field in this table matches the key fields of these tables. These relationships allow to combine each team's information, race results, qualifying rounds, team standings and team points.

The constructor_results table is related to the races and constructors’ tables. The raceId, constructorId fields in this table match the key fields of these tables. These relationships allow to combine the points scored by the teams at the end of each race, the details of the race and the team's information.

The constructor_standings table is related to the races, results, qualifying and constructors’ tables. The raceId, constructorId fields in this table match the key fields of these tables. These relationships allow to combine after each race the status of the teams, details of the race, race results, qualifying rounds, and team information.

The pit_stops table is related to the races and drivers’ tables. The raceId, driverId fields in this table match the key fields of these tables. These relationships allow to combine the pit stops of each race, the details of the race and the driver's information.

The lap_times table is related to the races and drivers’ tables. The raceId, driverId fields in this table match the key fields of these tables. These relationships allow to combine the lap times of each race, the details of the race and the driver's information.

2.6. Relationships between table types

Results N:1 Races
Results N:1 Drivers
Results N:1 Constructors
Results N:1 Status

Races N:1 Circuits

Qualifying N:1 Races
Qualifying N:1 Drivers
Qualifying N:1 Constructors

Constructor_Standings N:1 Races
Constructor_Standings N:1 Constructors

Driver_Standings N:1 Races
Driver_Standings N:1 Drivers

Pit_Stops N:1 Races
Pit_Stops N:1 Drivers

Lap_Times N:1 Races
Lap_Times N:1 Drivers

Constructor_Results N:1 Constructors
Constructor_Results N:1 Races

3. Dimensional Design of the system o Dimensional Model Used

3.1. Logical Model

results (resultId, raceId, driverId, constructorId, number, grid, position, positionText, positionOrder, points, laps, time, milisecond, fastestLap, rank, fastestLapTime, fastestLapSpeed, statusId) raceId is FK refers to races, driverId is FK refers to drivers, constructorId is FK refers to constructors.

races (raceId, year, round, circuitId, name, date, time, url) circuitId is FK refers to circuits.

circuits (circuitId, circuitRef, name, location, country, lat, lng, alt, url)

drivers (driverId, driverRef, number, code, forename, surname, dob, nationality, url)

driver_standings (driverStandingsId, raceId, driverId, points, position, positionText, wins) raceId is FK refers to races, driverId is FK refers to drivers.

qualifying (qualifyId, raceId, driverId, constructorId, number, position, q1, q2, q3) raceId is FK refers to races, driverId is FK refers to drivers.

constructors (constructorId, constructorRef, name, nationality, url)

constructor_standings (constructorStandingsId, raceId, constructorId, points, position, positionText, wins) raceId is FK refers to races, constructorId is FK refers to constructors.

constructor_results (constructorResultsId, raceId, constructorId, points, status) raceId is FK refers to races, constructorId is FK refers to constructors.

pit_stops (raceId, driverId, stop, lap, time, duration, milliseconds) raceId is FK refers to races, driverId is FK refers to drivers.

lap_times (raceId, driverId, lap, position, time, milliseconds) raceId is FK refers to races, driverId is FK refers to drivers.

status (statusId, status)

seasons (year, url)

3.2. Physical Model

→ **Results Table**

```
CREATE TABLE results (
   resultId INT PRIMARY KEY,
   raceId INT,
   driverId INT,
   constructorId INT,
   number INT,
   grid INT,
   position INT,
   positionText VARCHAR(10),
   positionOrder INT,
   points FLOAT,
   laps INT,
   time TIME,
   milisecond INT,
   fastestLap INT,
   rank INT,
   fastestLapTime TIME,
   fastestLapSpeed FLOAT,
   statusId INT,
   FOREIGN KEY (raceId) REFERENCES races(raceId),
   FOREIGN KEY (driverId) REFERENCES drivers(driverId),
   FOREIGN KEY (constructorId) REFERENCES constructors(constructorId)
);
```

→ **Races Table**

```
CREATE TABLE races (
   raceId INT PRIMARY KEY,
   year INT,
   round INT,
   circuitId INT,
   name VARCHAR(100),
   date DATE,
   time TIME,
   url VARCHAR(255),
   FOREIGN KEY (circuitId) REFERENCES circuits(circuitId)
);
```

→ Circuits Table

```
CREATE TABLE circuits (
circuitId INT PRIMARY KEY,
circuitRef VARCHAR(50),
name VARCHAR(100),
location VARCHAR(100),
country VARCHAR(50),
lat FLOAT,
lng FLOAT,
alt INT,
url VARCHAR(255)
);
```

→ Qualifying Table

```
CREATE TABLE qualifying (
   qualifyId INT PRIMARY KEY,
   raceId INT,
   driverId INT,
   constructorId INT,
   number INT,
   position INT,
   q1 TIME,
   q2 TIME,
   q3 TIME,
   FOREIGN KEY (raceId) REFERENCES races(raceId),
   FOREIGN KEY (driverId) REFERENCES drivers(driverId),
   FOREIGN KEY (constructorId) REFERENCES constructors(constructorId)
);
```

→ Drivers Table

```
CREATE TABLE drivers (
   driverId INT PRIMARY KEY,
   driverRef VARCHAR(50),
   number INT,
   code VARCHAR(5),
   forename VARCHAR(50),
   surname VARCHAR(50),
   dob DATE,
   nationality VARCHAR(50),
   url VARCHAR(255)
);
```

→ Driver_Standings Table

```
CREATE TABLE driver_standings (
   driverStandingsId INT PRIMARY KEY,
   raceId INT,
   driverId INT,
   points FLOAT,
   position INT,
   positionText VARCHAR(10),
   wins INT,
   FOREIGN KEY (raceId) REFERENCES races(raceId),
   FOREIGN KEY (driverId) REFERENCES drivers(driverId)
);
```

→ Constructors Table

```
CREATE TABLE constructors (
   constructorId INT PRIMARY KEY,
   constructorRef VARCHAR(50),
   name VARCHAR(100),
   nationality VARCHAR(50),
   url VARCHAR(255)
);
```

→ Constructor_Standings Table

```
CREATE TABLE constructor_standings (
   constructorStandingsId INT PRIMARY KEY,
   raceId INT,
   constructorId INT,
   points FLOAT,
   position INT,
   positionText VARCHAR(10),
   wins INT,
   FOREIGN KEY (raceId) REFERENCES races(raceId),
   FOREIGN KEY (constructorId) REFERENCES constructors(constructorId)
);
```

→ Constructor_Results Table

```
CREATE TABLE constructor_results (
   constructorResultsId INT PRIMARY KEY,
   raceId INT,
   constructorId INT,
   points FLOAT,
   status INT,
   FOREIGN KEY (raceId) REFERENCES races(raceId),
   FOREIGN KEY (constructorId) REFERENCES constructors(constructorId)
);
```

→ Pit_Stops Table

```
CREATE TABLE pit_stops (
   raceId INT,
   driverId INT,
   stop INT,
   lap INT,
   time TIME,
   duration TIME,
   milliseconds INT,
   PRIMARY KEY (raceId, driverId, stop),
   FOREIGN KEY (raceId) REFERENCES races(raceId),
   FOREIGN KEY (driverId) REFERENCES drivers(driverId)
);
```

→ **Lap_Times Table**

```
CREATE TABLE lap_times (
   raceId INT,
   driverId INT,
   lap INT,
   position INT,
   time TIME,
   milliseconds INT,
   PRIMARY KEY (raceId, driverId, lap),
   FOREIGN KEY (raceId) REFERENCES races(raceId),
   FOREIGN KEY (driverId) REFERENCES drivers(driverId)
);
```

→ **Seasons Table**

```
CREATE TABLE seasons (
   year INT PRIMARY KEY,
   url VARCHAR(255)
);
```

3.3. What are the decision-making questions?

1. Who is the team that won the most races?
2. Who is the team that won the least number of races?
3. Who is the team with the most points?
4. Who is the team with the fewest points?
5. Which team performed the best after one race?
6. Which team performed the best over the course of a season?
7. Which team performed the worst after one race?
8. Which team performed the worst over the course of a season?
9. What factors influence a team's performance?
10. Is there a relationship between a team's performance and the drivers it employs?
11. Who is the driver who has won the most races?
12. Who is the driver who has won the fewest races?
13. Who is the driver with the most points?
14. Who is the driver with the least points?
15. Which driver performed the best after one race?
16. Which driver performed the best over the course of a season?
17. Which driver had the worst performance after one race?
18. Which driver performed the worst over the course of a season?
19. What factors influence a driver's performance?
20. What are the average finishing positions and points earned by each driver?
21. How does a driver's performance vary between different circuits?
22. What is the relationship between a driver's grid position and their final race position?
23. How do the driver standings change from the beginning to the end of the season?
24. Who holds the record for the fastest lap in a season?
25. How consistent are drivers and teams in achieving similar race positions at different circuits?
26. Which circuit has the most races?

27. How do the characteristics of the track affect the outcome of the races?
28. What factors influence the outcome of races?
29. Which races have the highest average lap speeds and fastest lap times?
30. What is the average duration of pit stops and how do they affect the driver's race position?

3.4. What is the Fact?

Fact: Race Results

he basic fact in this data marketplace will revolve around race results and will collect detailed information about each race, including the performance of drivers, teams, and various race-related statistics.

This fact will include information from the 'results' table, including data such as resultId, raceId, driverId, constructorId, grid, position, points, laps, time, fastest lap and other relevant details. Race result fact will serve as the primary focus of analysis and reporting in the data marketplace, allowing users to gain insight into individual race results, driver performances, team standings and related statistics.

3.5. What are the dimensions?

In the context of the Formula 1 data mart, dimensions represent the various aspects or attributes by which the central fact (Race Results) can be analyzed. Each dimension provides additional context and details for understanding the factors that influence race results.

Time Dimension:

- timeId: Identifier for the time
- Year: The year in which the race took place
- Month: The month in which the race took place
- Day: The day in which the race took place

Location Dimension:

- CircuitId: Identifier for the circuit
- Name: Name of the circuit
- City: City of the circuit
- Country: Country in which the circuit is located

Driver Dimension:

- DriverId: Identifier for the driver
- Forename: Driver's first name
- Surname: Driver's last name
- Nationality: Driver's nationality
- DOB: Driver’s date of birth (year, month, day)

Constructor Dimension:

- ConstructorId: Identifier for the constructor
- Name: Name of the constructor
- Nationality: Nationality of the constructor

Race_Status Dimension:

- StatusId: Identifier for the race status
- Status: Description of the race status

Race_Qualifying Dimension:

- QualifyId: Identifier for the qualifying session
- Q1, Q2, Q3: Qualifying session times for each driver

These dimensions provide additional context for analyzing race results and can be used to filter, group, and aggregate data in various ways during analysis. Each dimension contains attributes that help users explore different aspects of the races and draw meaningful insights.

3.6. What are the measures?

Grid: Drive's starting position

Finished Position: Driver's order of finishing the race

Points: Points earned by each driver in a race

Laps: Number of laps completed by each driver

Duration: Total time taken by the driver to complete the race

Pit Stops: Total number of stops for car maintenance made by the driver in the race

3.7. Draw the Star Scheme?

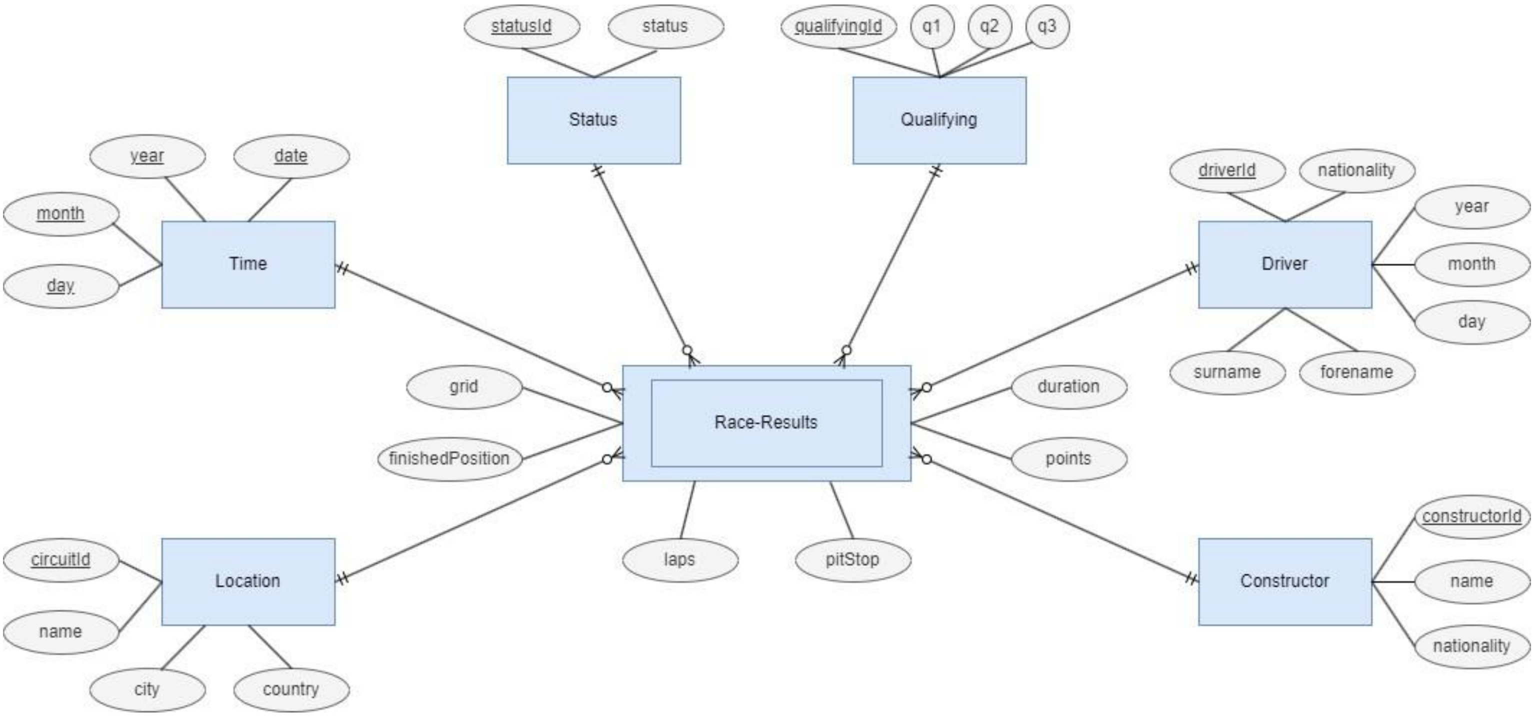


Figure 11: Star Scheme

3.7.1. Star Scheme Description

Driver Table:

driverId (Primary Key): Unique identifier of the driver (integer)

forename: Drive name (varchar, 255 characters)

surname: Last name of the driver (varchar, 255 characters)

nationality: Nationality of the driver (varchar, 255 characters)

day, month, year: Driver's date of birth (integer)

Constructor Table:

constructorId (Primary Key): Unique identifier of the team (integer)

name: Team name (varchar, 255 characters)

nationality: Nationality of the team (varchar, 255 characters)

Time Table:

day: The day of the race (integer)

month: Day of the race (integer)

year: Day of the race (integer)

timeId: Identifier for the time (integer)

Location Table:

circuitid (Primary Key): Identifier of the circuit where the race was held (integer).

name: Name of the circuit (varchar, 255 characters)

city: City of the race (varchar, 255 characters)

country: Country of the race (varchar, 255 characters)

Status Table:

statusId (Primary Key): Unique identifier of Status (integer)

status: Names of the status of the race results (varchar, 255 characters)

Qualifying Table:

qualifyingId (Primary Key): Unique identifier for sorting (integer)

q1, q2, q3: Lap times that determine who will participate in the race and in which order (time)

RaceResults Table:

driverid (Foreign Key): Reference to the driver table, identifier of the competing driver

constructorid (Foreign Key): Reference to the constructors table, identifier of the competing team statusId (Foreign Key): Reference to the status table, driver's result in the race

timeId (Foreign Key): Reference to the time table, time of the race

locationId (Foreign Key): Reference to the location table, where the race was held

qualifyingId (Foreign Key): Reference to the qualifying table, lap details of the starting race

grid: Start sequence of the drive (integer)

finishedposition: Race completion order (integer)

duration: Driver's time to finish the race (time)

points Points earned in the race (integer)

status: Driver's result in the race (varchar, 255 characters)

laps: Number of laps completed in the race (integer)

pitStop: Total number of stops for car maintenance the driver made in the race

These tables and their relationships contain basic information about Formula One races, drivers, teams, qualifying and race results.

3.7.2. Star Scheme Logical Model

driver (driverId, surname, forname, notionality, day, month, year)

constructor (constructorId, name, nationality)

time (day, month, year, timeId)

location (circuitId, name, city, country)

status (statusId, status)

qualifying (qualifyingId, q1, q2, q3)

race_results (driverid, constructorid, statusId, timeId, locationId, qualifyingId, grid, finishedposition, duration, points, status, laps, pitStop) driverid is FK refers to driver, constructorid is FK refers to constructor, statusId is FK refers to status, timeId is FK refers to time, locationId is FK refers to location, qualifyingId is FK refers to qualifying)

3.7.3. Star Scheme Physical Model

```
-- Driver Table
CREATE TABLE Driver (
   driverId INTEGER PRIMARY KEY,
   forename VARCHAR(255),
   surname VARCHAR(255),
   nationality VARCHAR(255),
   day INTEGER,
   month INTEGER,
   year INTEGER
);
```

```
-- Constructor Table
CREATE TABLE Constructor (
   constructorId INTEGER PRIMARY KEY,
   name VARCHAR(255),
   nationality VARCHAR(255)
);

-- Time Table
CREATE TABLE Time (
   day INTEGER,
   month INTEGER,
   year INTEGER,
   timeId INTEGER PRIMARY KEY
);

-- Location Table
CREATE TABLE Location (
   circuitid INTEGER PRIMARY KEY,
   name VARCHAR(255),
   city VARCHAR(255),
   country VARCHAR(255)
);

-- Status Table
CREATE TABLE Status (
   statusId INTEGER PRIMARY KEY,
   status VARCHAR(255)
);

-- Qualifying Table
CREATE TABLE Qualifying (
   qualifyingId INTEGER PRIMARY KEY,
   q1 DECIMAL(10,2),
   q2 DECIMAL(10,2),
   q3 DECIMAL(10,2)
);


-- RaceResults Table
CREATE TABLE RaceResults (
   driverid INTEGER,
   constructorid INTEGER,
   statusId INTEGER,
   timeId INTEGER,
   locationId INTEGER,
   qualifyingId INTEGER,
   grid INTEGER,
   finishedposition INTEGER,
   duration TIME,
   points INTEGER,
   status VARCHAR(255),
   laps INTEGER,
   pitStop INTEGER,
   PRIMARY KEY (driverid, timeId),
   FOREIGN KEY (driverid) REFERENCES Driver(driverId),
   FOREIGN KEY (constructorid) REFERENCES Constructor(constructorId),
   FOREIGN KEY (statusId) REFERENCES Status(statusId),
   FOREIGN KEY (timeId) REFERENCES Time(timeId),
   FOREIGN KEY (locationId) REFERENCES Location(circuitid),
   FOREIGN KEY (qualifyingId) REFERENCES Qualifying(qualifyingId)
);
```

3.8. Draw the Snowflake?

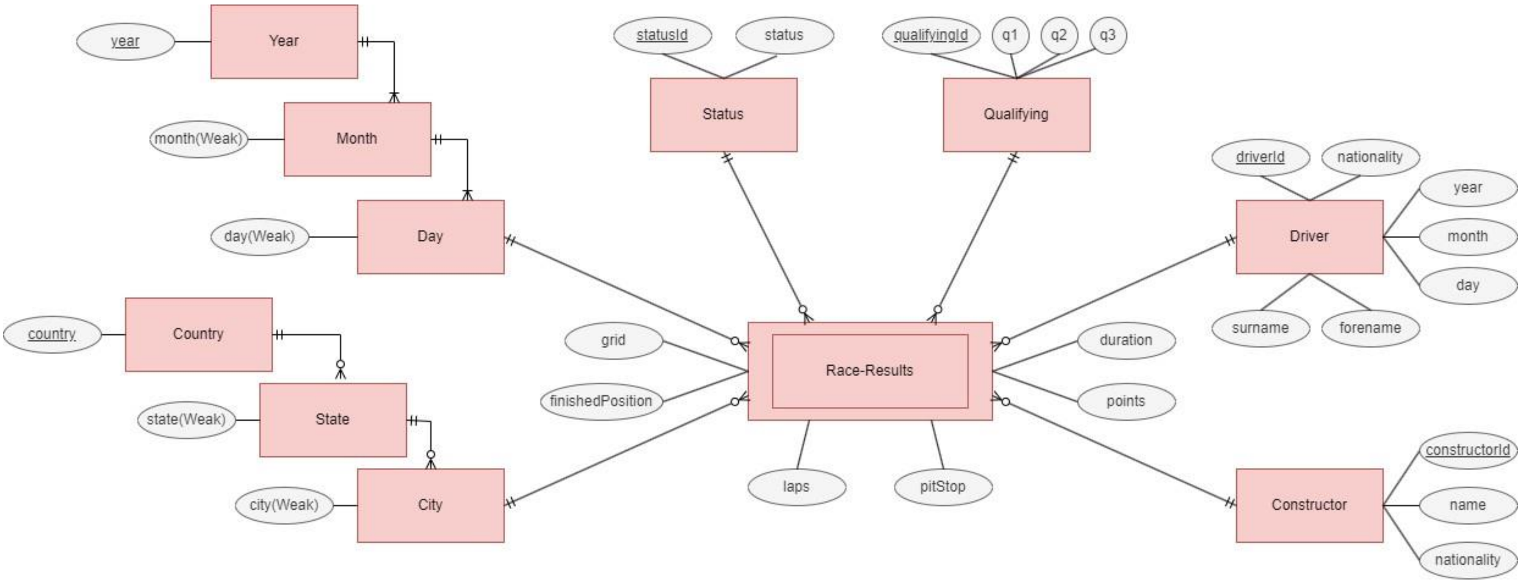


Figure 12: Snowflake Scheme

3.9. Why did we use the star scheme?

Star schema is the preferred model for a data warehouse design because it simplifies data analysis and reporting processes while improving query performance. In this model, there are dimensional tables (e.g. drivers, race locations, time, etc.) around a basic "factor" table (e.g. race results). The factor table contains the main metrics and events, while the dimensional tables provide additional information describing those metrics.

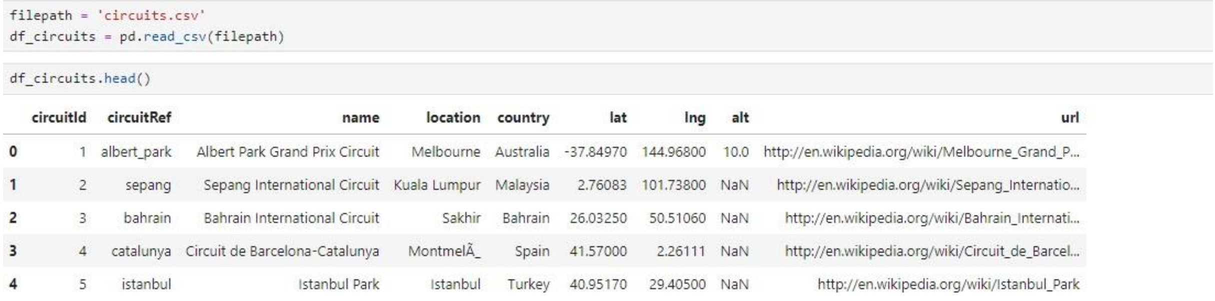
The advantages of this design include improving query performance, performing complex queries quickly and efficiently, increasing data understanding and simplifying reporting processes. Dimensional tables provide ideal filtering and grouping options to optimize queries and perform analysis. It is also easy to add new dimensional data or update existing ones, making the system scalable.

Star schema is preferred by many organizations to perform effective queries on large datasets and manage the complexity of the data warehouse. This model can be particularly effective when simplicity, flexibility and performance are at the forefront of data warehouse design.

4. ETL Operations and Create Data Warehouse

4.1. ETL Operations

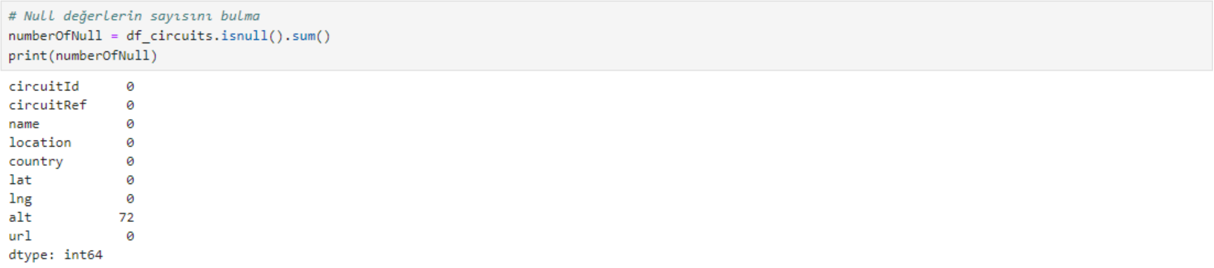
4.1.1. Circuits Table



```
filepath = 'circuits.csv'
df_circuits = pd.read_csv(filepath)
```

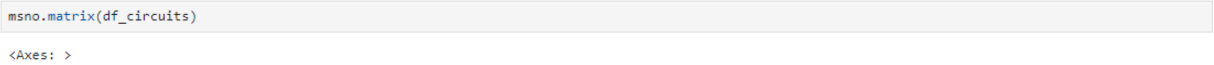
```
df_circuits.head()
```

	circuitId	circuitRef	name	location	country	lat	lng	alt	url
0	1	albert_park	Albert Park Grand Prix Circuit	Melbourne	Australia	-37.84970	144.96800	10.0	http://en.wikipedia.org/wiki/Melbourne_Grand_P...
1	2	sepang	Sepang International Circuit	Kuala Lumpur	Malaysia	2.76083	101.73800	NaN	http://en.wikipedia.org/wiki/Sepang_Internatio...
2	3	bahrain	Bahrain International Circuit	Sakhir	Bahrain	26.03250	50.51060	NaN	http://en.wikipedia.org/wiki/Bahrain_Internati...
3	4	catalunya	Circuit de Barcelona-Catalunya	MontmelÃ_	Spain	41.57000	2.26111	NaN	http://en.wikipedia.org/wiki/Circuit_de_Barcel...
4	5	istanbul	Istanbul Park	Istanbul	Turkey	40.95170	29.40500	NaN	http://en.wikipedia.org/wiki/Istanbul_Park



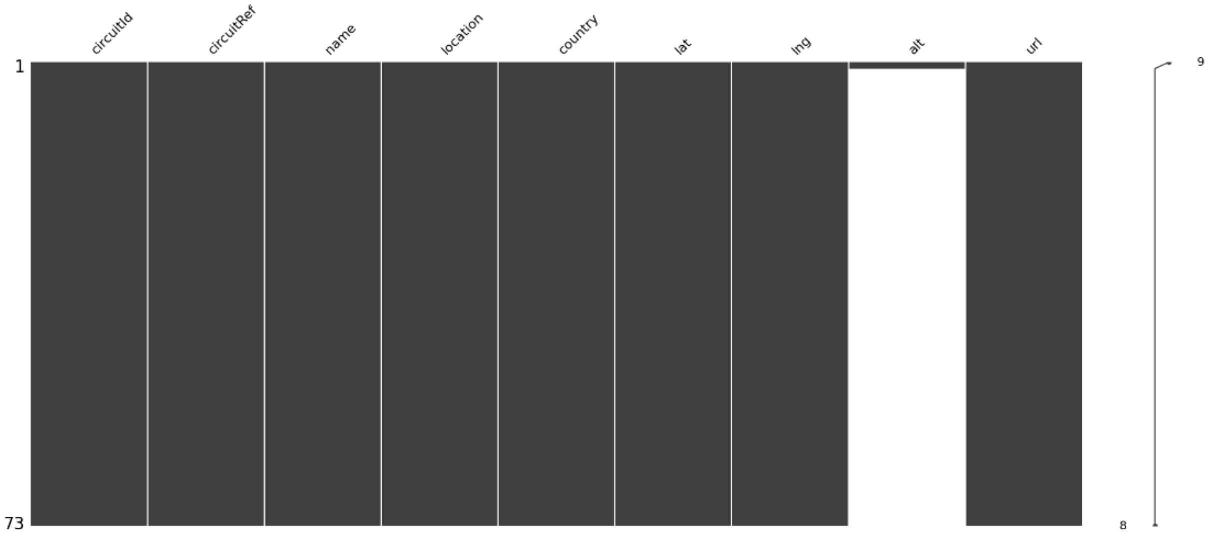
```
# Null değerlerin sayısını bulma
numberOfNull = df_circuits.isnull().sum()
print(numberOfNull)
```

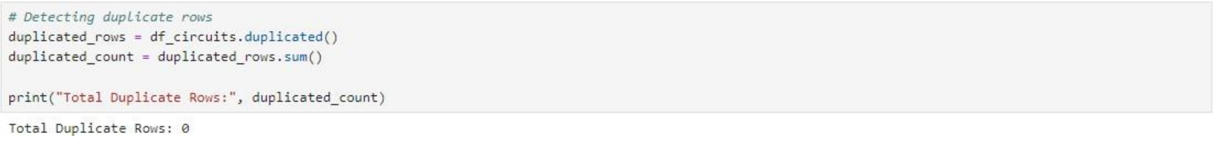
```
circuitId      0
circuitRef     0
name           0
location       0
country        0
lat            0
lng            0
alt           72
url            0
dtype: int64
```



```
msno.matrix(df_circuits)
```

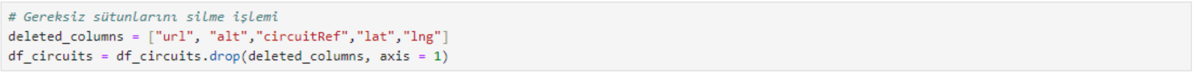
```
<Axes: >
```



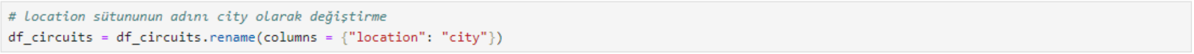


```
# Detecting duplicate rows
duplicated_rows = df_circuits.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

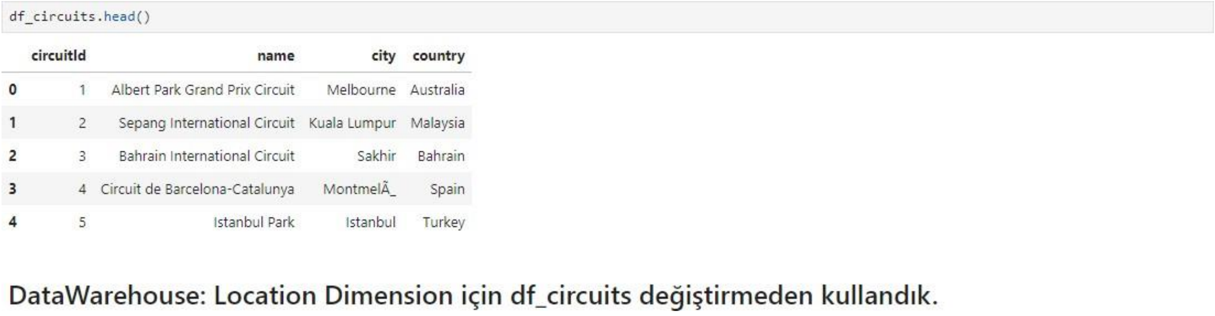
```
Total Duplicate Rows: 0
```



```
# Gereksiz sütunlarını silme işlemi
deleted_columns = ["url", "alt","circuitRef","lat","lng"]
df_circuits = df_circuits.drop(deleted_columns, axis = 1)
```



```
# location sütununun adını city olarak değiştirme
df_circuits = df_circuits.rename(columns = {"location": "city"})
```

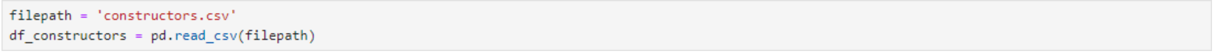


```
df_circuits.head()
```

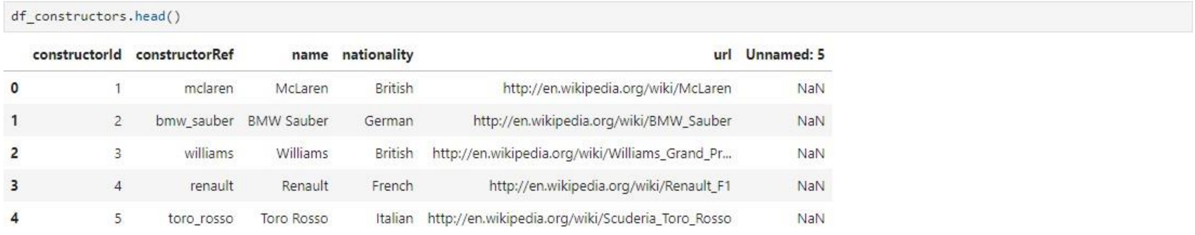
	circuitId	name	city	country
0	1	Albert Park Grand Prix Circuit	Melbourne	Australia
1	2	Sepang International Circuit	Kuala Lumpur	Malaysia
2	3	Bahrain International Circuit	Sakhir	Bahrain
3	4	Circuit de Barcelona-Catalunya	MontmelÃ_	Spain
4	5	Istanbul Park	Istanbul	Turkey

DataWarehouse: Location Dimension için df_circuits değiştirmeden kullandık.

4.1.2. Constructors Table

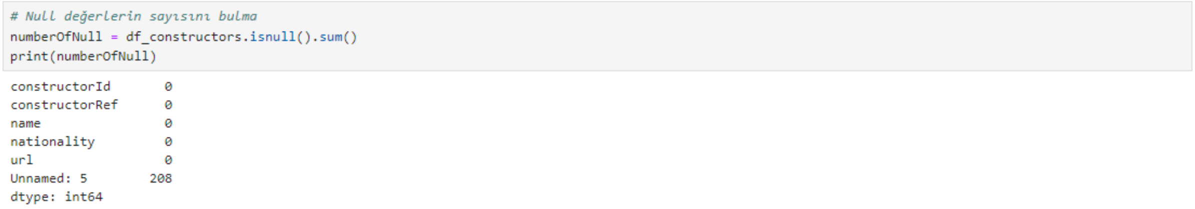


```
filepath = 'constructors.csv'
df_constructors = pd.read_csv(filepath)
```



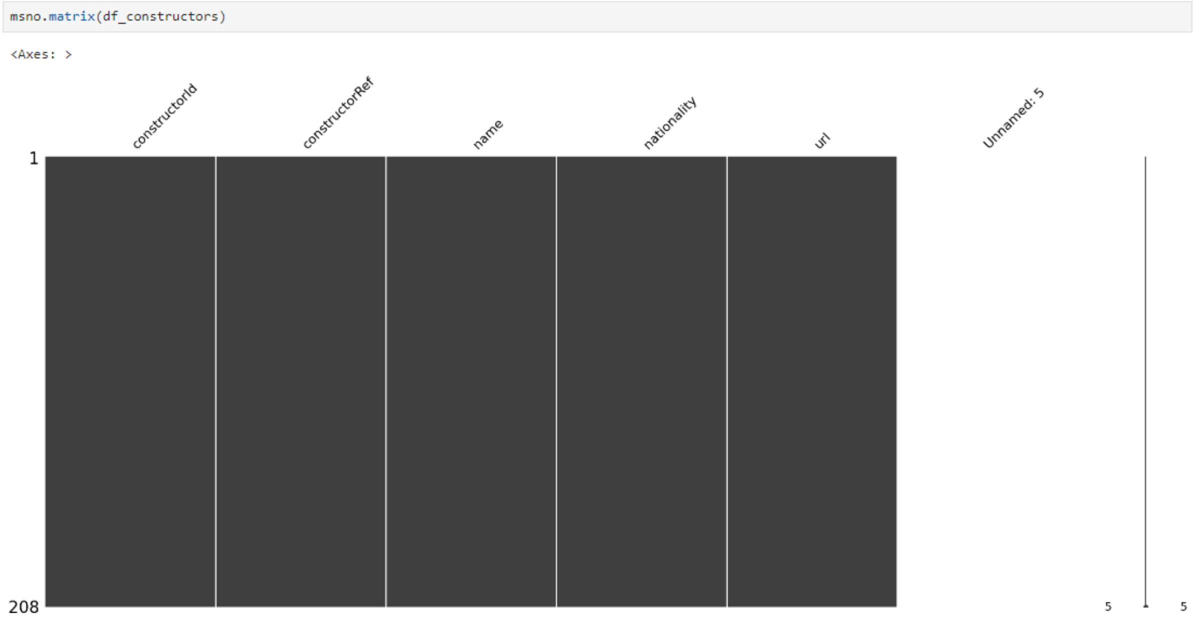
```
df_constructors.head()
```

	constructorId	constructorRef	name	nationality	url	Unnamed: 5
0	1	mclaren	McLaren	British	http://en.wikipedia.org/wiki/McLaren	NaN
1	2	bmw_sauber	BMW Sauber	German	http://en.wikipedia.org/wiki/BMW_Sauber	NaN
2	3	williams	Williams	British	http://en.wikipedia.org/wiki/Williams_Grand_Pr...	NaN
3	4	renault	Renault	French	http://en.wikipedia.org/wiki/Renault_F1	NaN
4	5	toro_rosso	Toro Rosso	Italian	http://en.wikipedia.org/wiki/Scuderia_Toro_Rosso	NaN



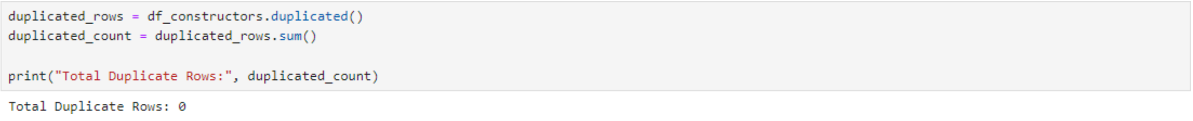
```
# Null değerlerin sayısını bulma
numberOfNull = df_constructors.isnull().sum()
print(numberOfNull)
```

```
constructorId       0
constructorRef      0
name                0
nationality         0
url                 0
Unnamed: 5        208
dtype: int64
```



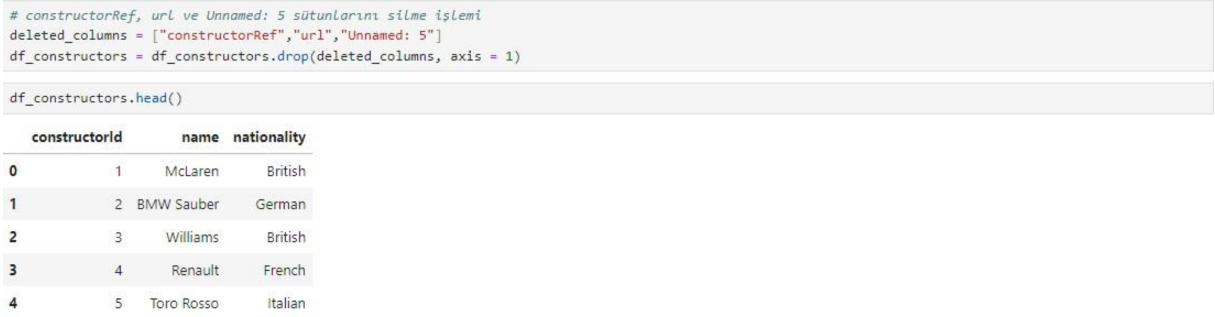
```
msno.matrix(df_constructors)
```

```
<Axes: >
```



```
duplicated_rows = df_constructors.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```



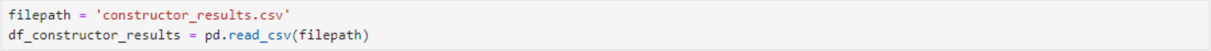
```
# constructorRef, url ve Unnamed: 5 sütunlarını silme işlemi
deleted_columns = ["constructorRef","url","Unnamed: 5"]
df_constructors = df_constructors.drop(deleted_columns, axis = 1)
```

```
df_constructors.head()
```

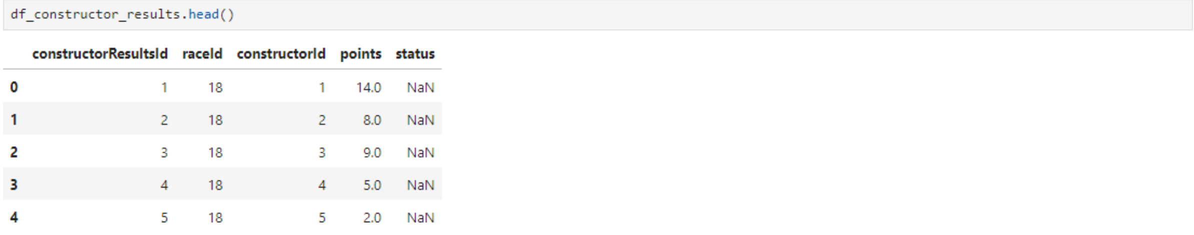
	constructorId	name	nationality
0	1	McLaren	British
1	2	BMW Sauber	German
2	3	Williams	British
3	4	Renault	French
4	5	Toro Rosso	Italian

DataWarehouse: Constructor Dimension için df_constructors değiştirmeden kullandık.

4.1.3. Constructor_Results Table

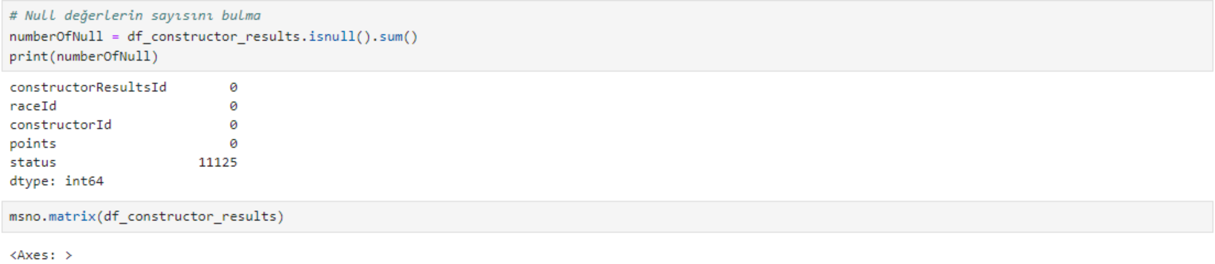


```
filepath = 'constructor_results.csv'
df_constructor_results = pd.read_csv(filepath)
```



```
df_constructor_results.head()
```

	constructorResultsId	raceId	constructorId	points	status
0	1	18	1	14.0	NaN
1	2	18	2	8.0	NaN
2	3	18	3	9.0	NaN
3	4	18	4	5.0	NaN
4	5	18	5	2.0	NaN

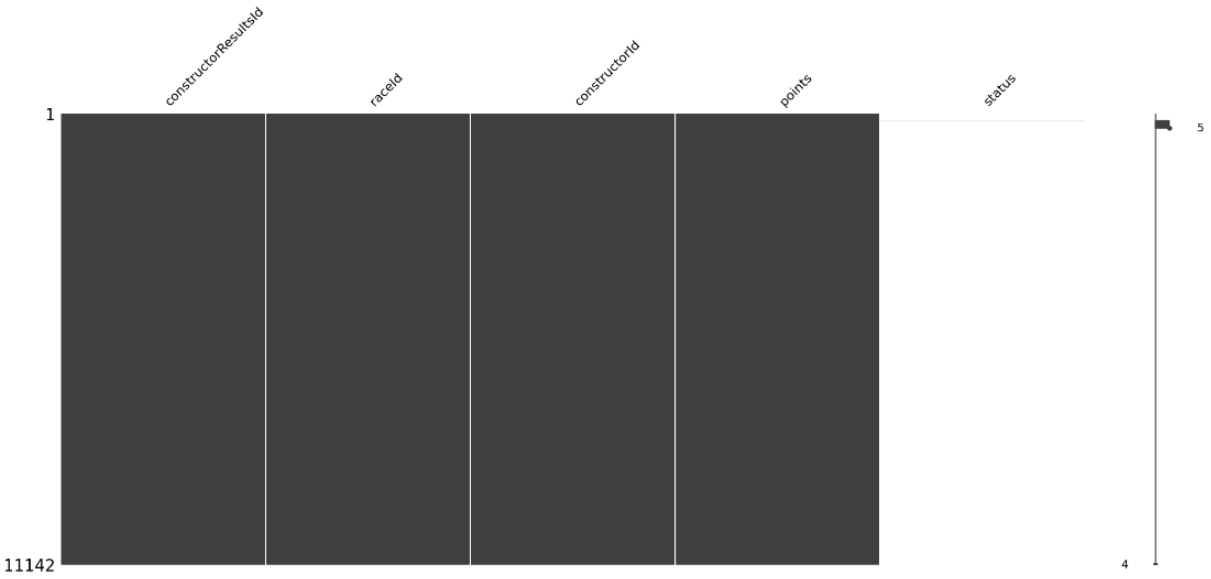


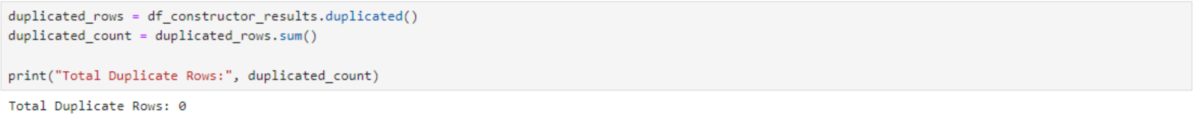
```
# Null değerlerin sayısını bulma
numberOfNull = df_constructor_results.isnull().sum()
print(numberOfNull)
```

```
constructorResultsId        0
raceId                      0
constructorId               0
points                      0
status                  11125
dtype: int64
```

```
msno.matrix(df_constructor_results)
```

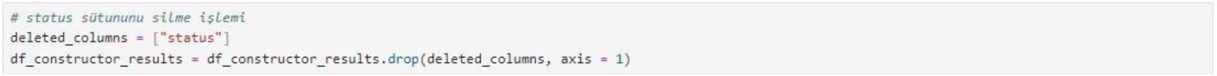
```
<Axes: >
```



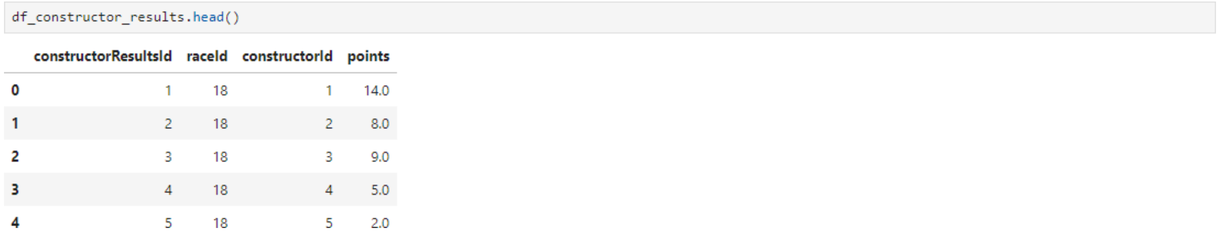


```
duplicated_rows = df_constructor_results.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```



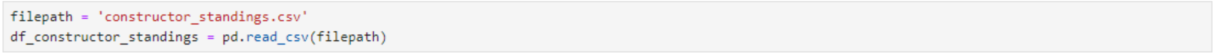
```
# status sütununu silme işlemi
deleted_columns = ["status"]
df_constructor_results = df_constructor_results.drop(deleted_columns, axis = 1)
```



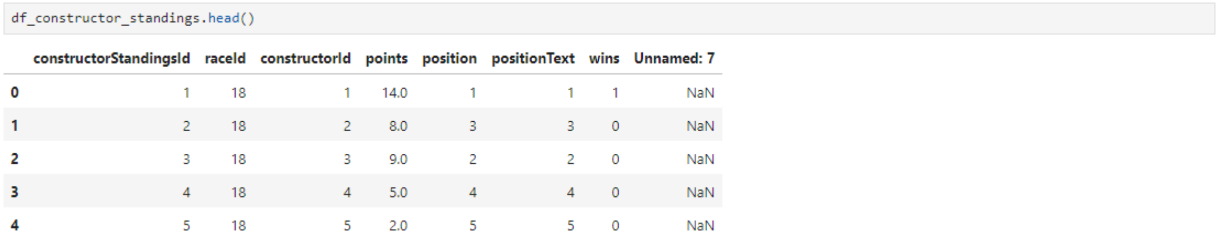
```
df_constructor_results.head()
```

	constructorResultsId	raceId	constructorId	points
0	1	18	1	14.0
1	2	18	2	8.0
2	3	18	3	9.0
3	4	18	4	5.0
4	5	18	5	2.0

4.1.4. Constructor_Standings Table

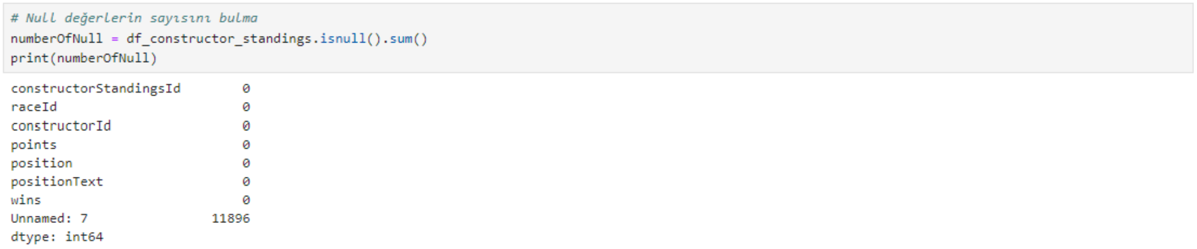


```
filepath = 'constructor_standings.csv'
df_constructor_standings = pd.read_csv(filepath)
```



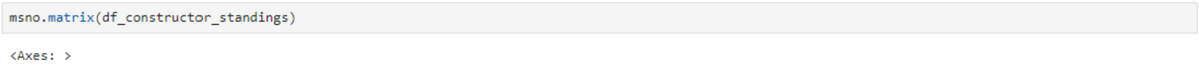
```
df_constructor_standings.head()
```

	constructorStandingsId	raceId	constructorId	points	position	positionText	wins	Unnamed: 7
0	1	18	1	14.0	1	1	1	NaN
1	2	18	2	8.0	3	3	0	NaN
2	3	18	3	9.0	2	2	0	NaN
3	4	18	4	5.0	4	4	0	NaN
4	5	18	5	2.0	5	5	0	NaN



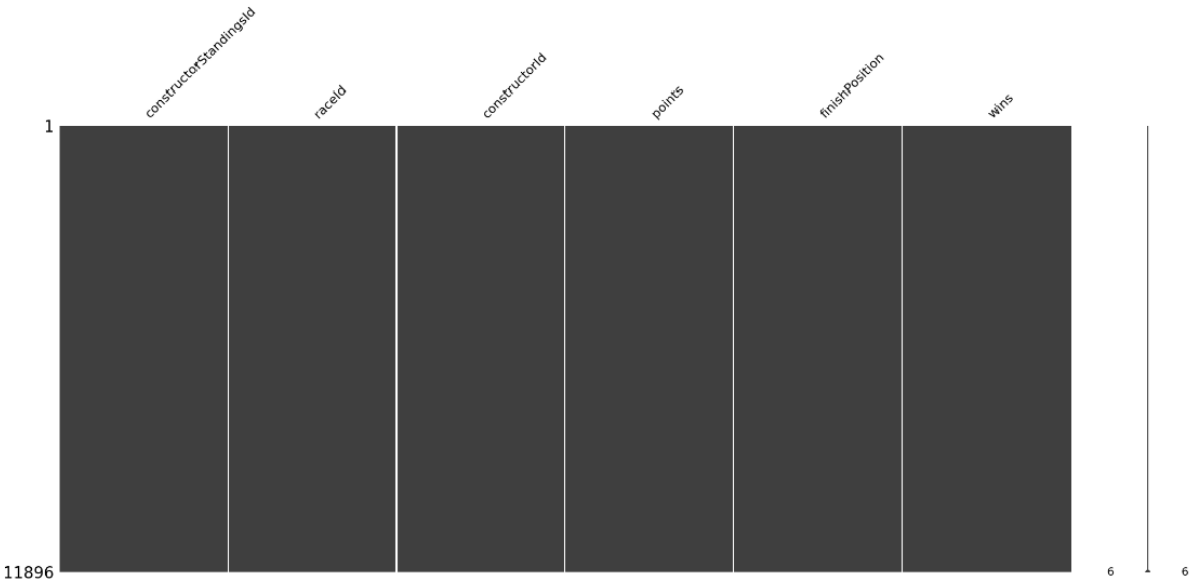
```
# Null değerlerin sayısını bulma
numberOfNull = df_constructor_standings.isnull().sum()
print(numberOfNull)
```

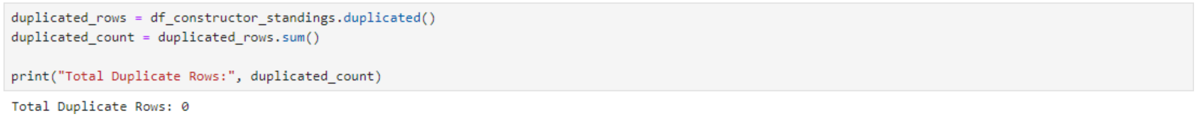
```
constructorStandingsId        0
raceId                        0
constructorId                 0
points                        0
position                      0
positionText                  0
wins                          0
Unnamed: 7                11896
dtype: int64
```



```
msno.matrix(df_constructor_standings)
```

```
<Axes: >
```





```
duplicated_rows = df_constructor_standings.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```

```
# Unnamed: 7 ve position sütunlarını silme işlemi
deleted_columns = ["Unnamed: 7","position"]
df_constructor_standings = df_constructor_standings.drop(deleted_columns, axis = 1)
```

```
# positionText sütununun adını finishPosition olarak değiştirme
df_constructor_standings = df_constructor_standings.rename(columns = {"positionText": "finishPosition"})
```

```
df_constructor_standings.head()
```

	constructorStandingsId	raceId	constructorId	points	finishPosition	wins
0	1	18	1	14.0	1	1
1	2	18	2	8.0	3	0
2	3	18	3	9.0	2	0
3	4	18	4	5.0	4	0
4	5	18	5	2.0	5	0

4.1.5. Drivers Table

```
filepath = 'drivers.csv'
df_drivers = pd.read_csv(filepath)
```

```
df_drivers.head()
```

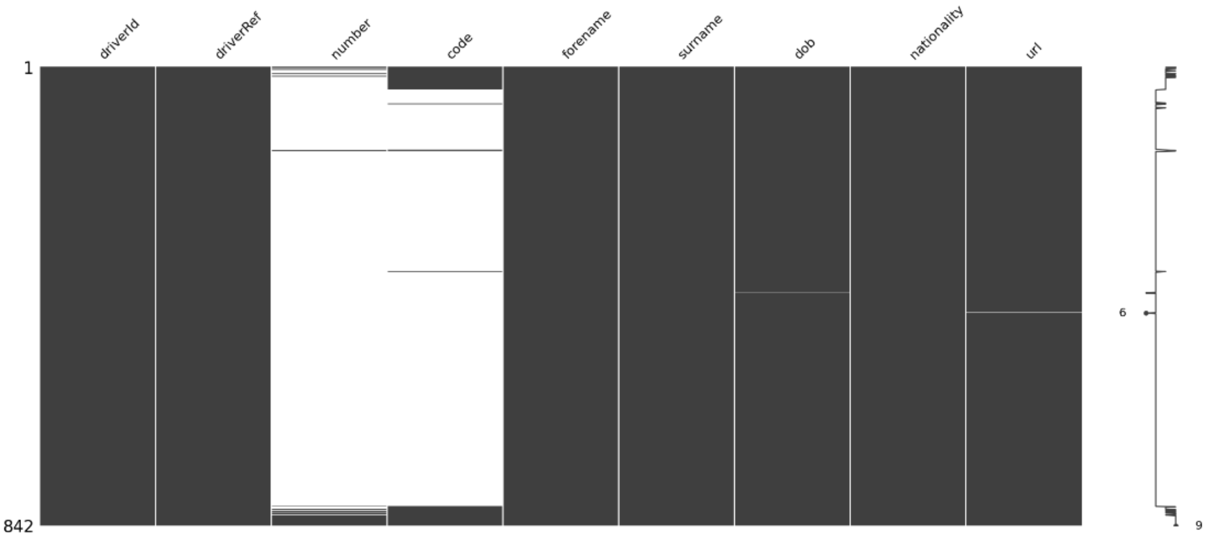
	driverId	driverRef	number	code	forename	surname	dob	nationality	url
0	1	hamilton	44.0	HAM	Lewis	Hamilton	07/01/1985	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
1	2	heidfeld	NaN	HEI	Nick	Heidfeld	10/05/1977	German	http://en.wikipedia.org/wiki/Nick_Heidfeld
2	3	rosberg	6.0	ROS	Nico	Rosberg	27/06/1985	German	http://en.wikipedia.org/wiki/Nico_Rosberg
3	4	alonso	14.0	ALO	Fernando	Alonso	29/07/1981	Spanish	http://en.wikipedia.org/wiki/Fernando_Alonso
4	5	kovalainen	NaN	KOV	Heikki	Kovalainen	19/10/1981	Finnish	http://en.wikipedia.org/wiki/Heikki_Kovalainen

```
# Null değerlerin sayısını bulma
numberOfNull = df_drivers.isnull().sum()
print(numberOfNull)
```

```
driverId         0
driverRef        0
number         804
code           757
forename         0
surname          0
dob              1
nationality      0
url              1
dtype: int64
```

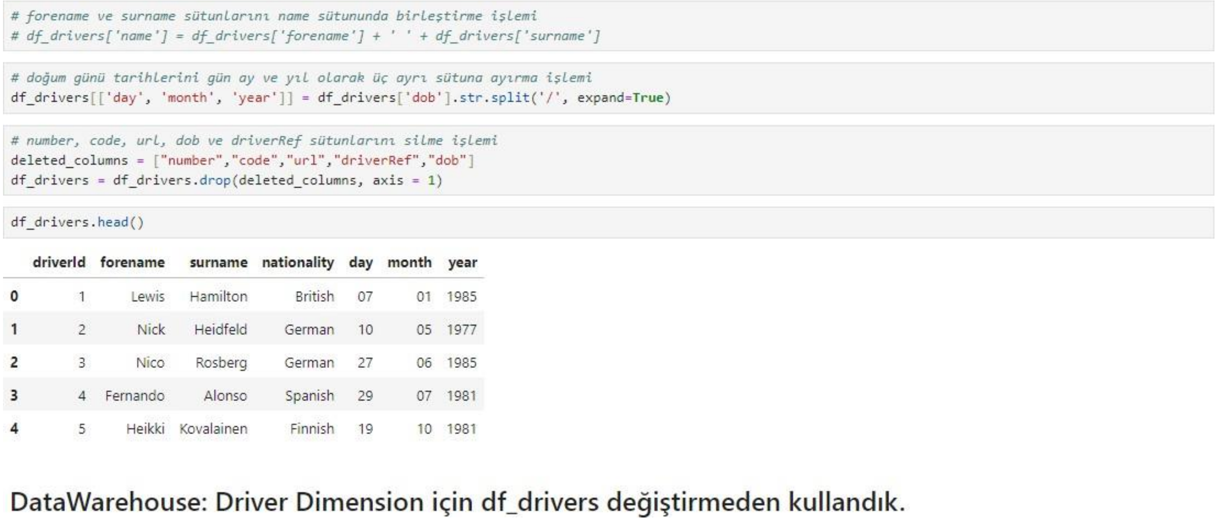
```
msno.matrix(df_drivers)
```

```
<Axes: >
```



```
duplicated_rows = df_drivers.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```



```
# forename ve surname sütunlarını name sütununda birleştirme işlemi
# df_drivers['name'] = df_drivers['forename'] + ' ' + df_drivers['surname']
```

```
# doğum günü tarihlerini gün ay ve yıl olarak üç ayrı sütuna ayırma işlemi
df_drivers[['day', 'month', 'year']] = df_drivers['dob'].str.split('/', expand=True)
```

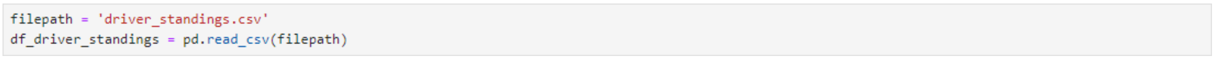
```
# number, code, url, dob ve driverRef sütunlarını silme işlemi
deleted_columns = ["number","code","url","driverRef","dob"]
df_drivers = df_drivers.drop(deleted_columns, axis = 1)
```

```
df_drivers.head()
```

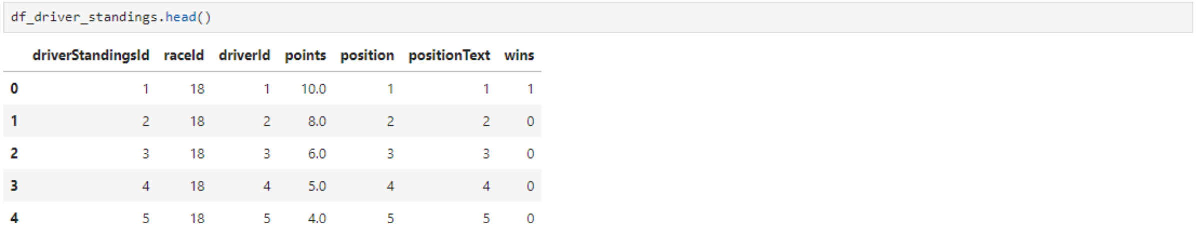
	driverId	forename	surname	nationality	day	month	year
0	1	Lewis	Hamilton	British	07	01	1985
1	2	Nick	Heidfeld	German	10	05	1977
2	3	Nico	Rosberg	German	27	06	1985
3	4	Fernando	Alonso	Spanish	29	07	1981
4	5	Heikki	Kovalainen	Finnish	19	10	1981

DataWarehouse: Driver Dimension için df_drivers değiştirmeden kullandık.

4.1.6. Driver_Standings Table

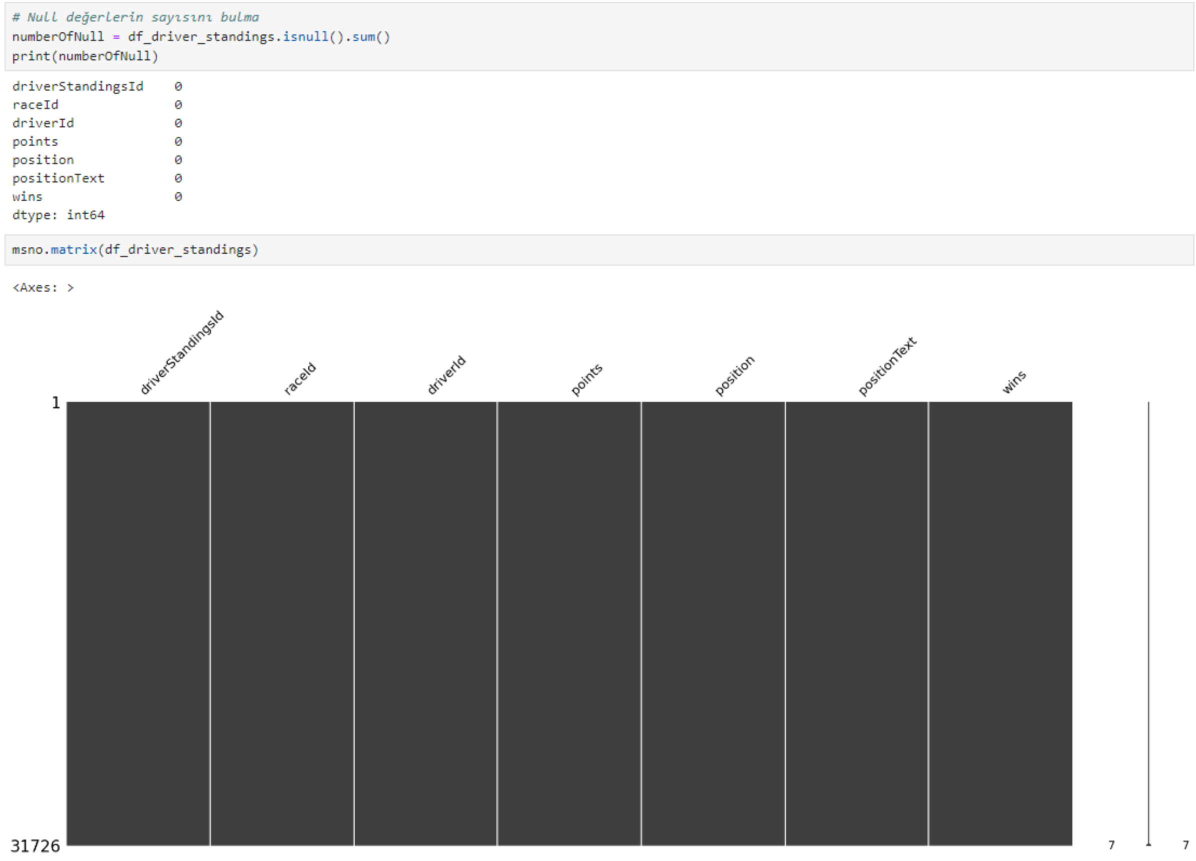


```
filepath = 'driver_standings.csv'
df_driver_standings = pd.read_csv(filepath)
```



```
df_driver_standings.head()
```

	driverStandingsId	raceId	driverId	points	position	positionText	wins
0	1	18	1	10.0	1	1	1
1	2	18	2	8.0	2	2	0
2	3	18	3	6.0	3	3	0
3	4	18	4	5.0	4	4	0
4	5	18	5	4.0	5	5	0

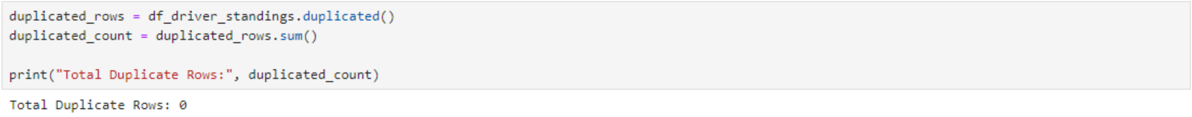


```
# Null değerlerin sayısını bulma
numberOfNull = df_driver_standings.isnull().sum()
print(numberOfNull)
```

```
driverStandingsId    0
raceId               0
driverId             0
points               0
position             0
positionText         0
wins                 0
dtype: int64
```

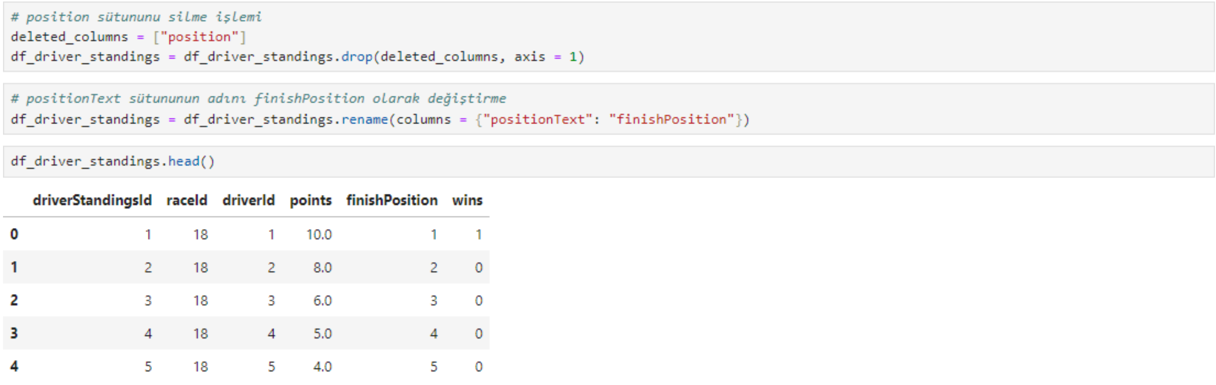
```
msno.matrix(df_driver_standings)
```

```
<Axes: >
```



```
duplicated_rows = df_driver_standings.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```



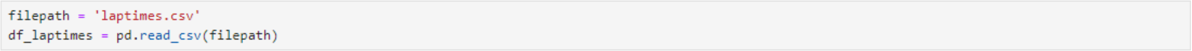
```
# position sütununu silme işlemi
deleted_columns = ["position"]
df_driver_standings = df_driver_standings.drop(deleted_columns, axis = 1)
```

```
# positionText sütununun adını finishPosition olarak değiştirme
df_driver_standings = df_driver_standings.rename(columns = {"positionText": "finishPosition"})
```

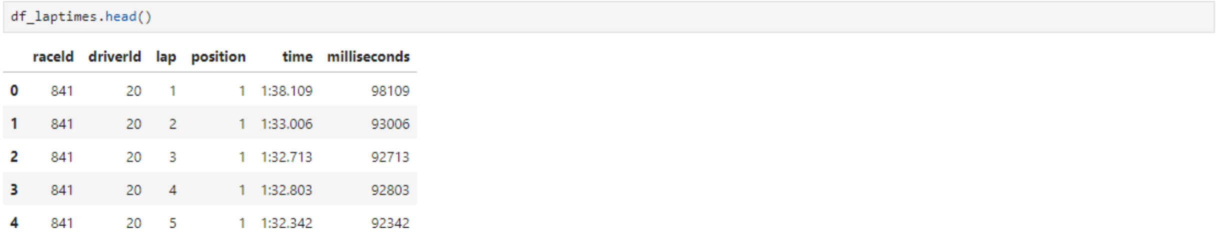
```
df_driver_standings.head()
```

	driverStandingsId	raceId	driverId	points	finishPosition	wins
0	1	18	1	10.0	1	1
1	2	18	2	8.0	2	0
2	3	18	3	6.0	3	0
3	4	18	4	5.0	4	0
4	5	18	5	4.0	5	0

4.1.7. Laptimes Table

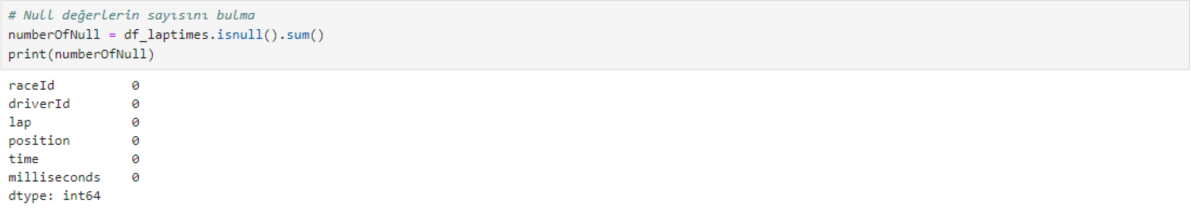


```
filepath = 'laptimes.csv'
df_laptimes = pd.read_csv(filepath)
```



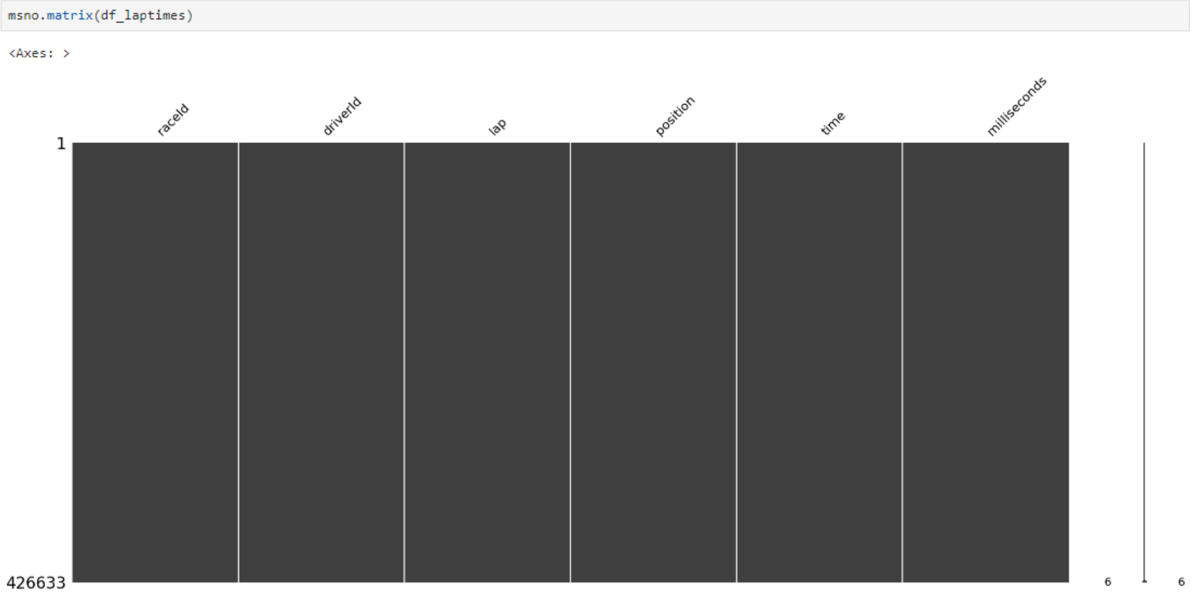
```
df_laptimes.head()
```

	raceId	driverId	lap	position	time	milliseconds
0	841	20	1	1	1:38.109	98109
1	841	20	2	1	1:33.006	93006
2	841	20	3	1	1:32.713	92713
3	841	20	4	1	1:32.803	92803
4	841	20	5	1	1:32.342	92342



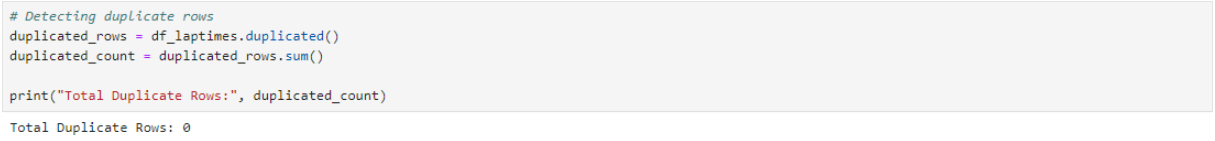
```
# Null değerlerin sayısını bulma
numberOfNull = df_laptimes.isnull().sum()
print(numberOfNull)
```

```
raceId          0
driverId        0
lap             0
position        0
time            0
milliseconds    0
dtype: int64
```



```
msno.matrix(df_laptimes)
```

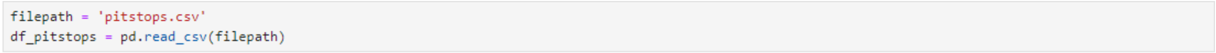
```
<Axes: >
```



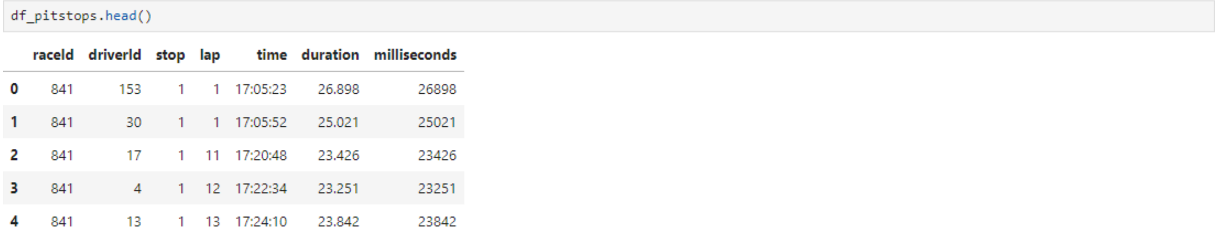
```
# Detecting duplicate rows
duplicated_rows = df_laptimes.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```

4.1.8. Pitstops Table

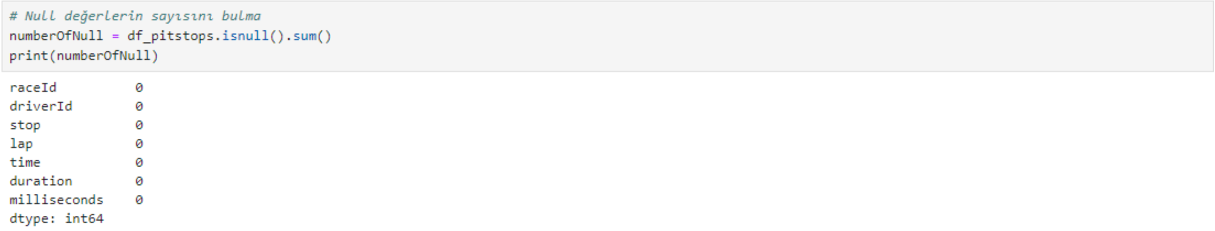


```
filepath = 'pitstops.csv'
df_pitstops = pd.read_csv(filepath)
```



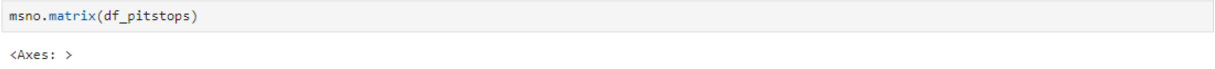
```
df_pitstops.head()
```

	raceId	driverId	stop	lap	time	duration	milliseconds
0	841	153	1	1	17:05:23	26.898	26898
1	841	30	1	1	17:05:52	25.021	25021
2	841	17	1	11	17:20:48	23.426	23426
3	841	4	1	12	17:22:34	23.251	23251
4	841	13	1	13	17:24:10	23.842	23842



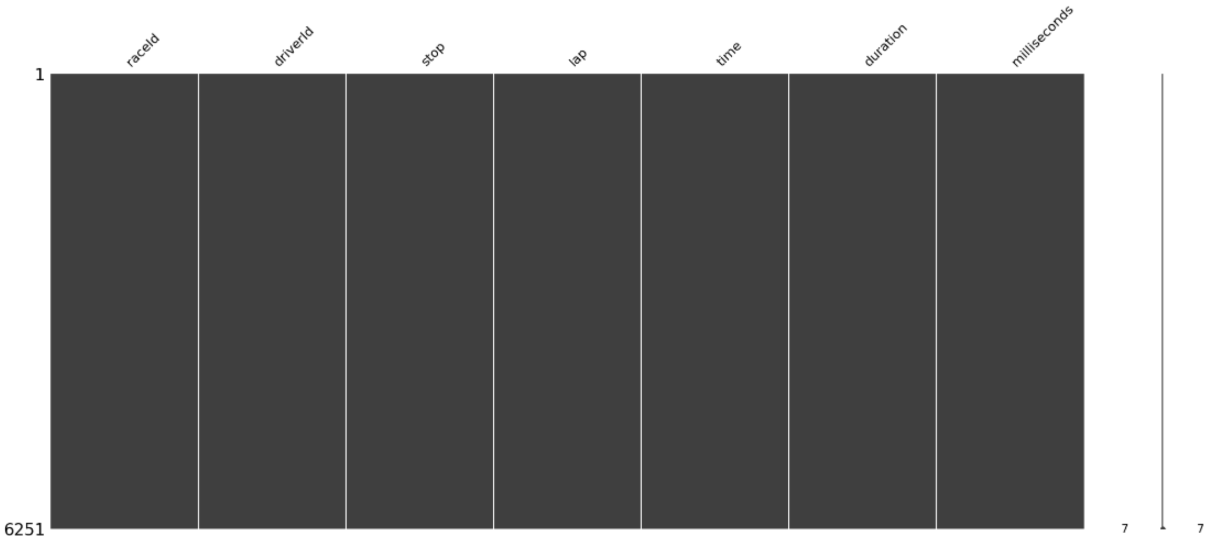
```
# Null değerlerin sayısını bulma
numberOfNull = df_pitstops.isnull().sum()
print(numberOfNull)
```

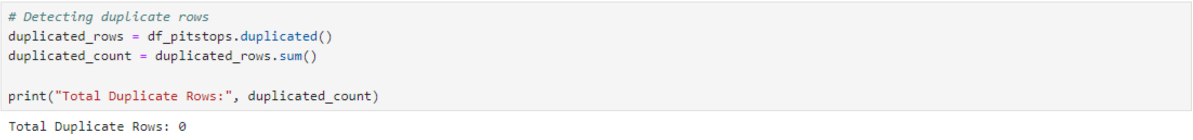
```
raceId          0
driverId        0
stop            0
lap             0
time            0
duration        0
milliseconds    0
dtype: int64
```



```
msno.matrix(df_pitstops)
```

```
<Axes: >
```

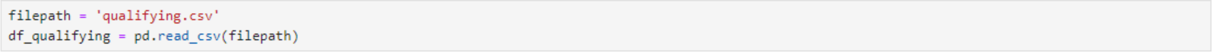




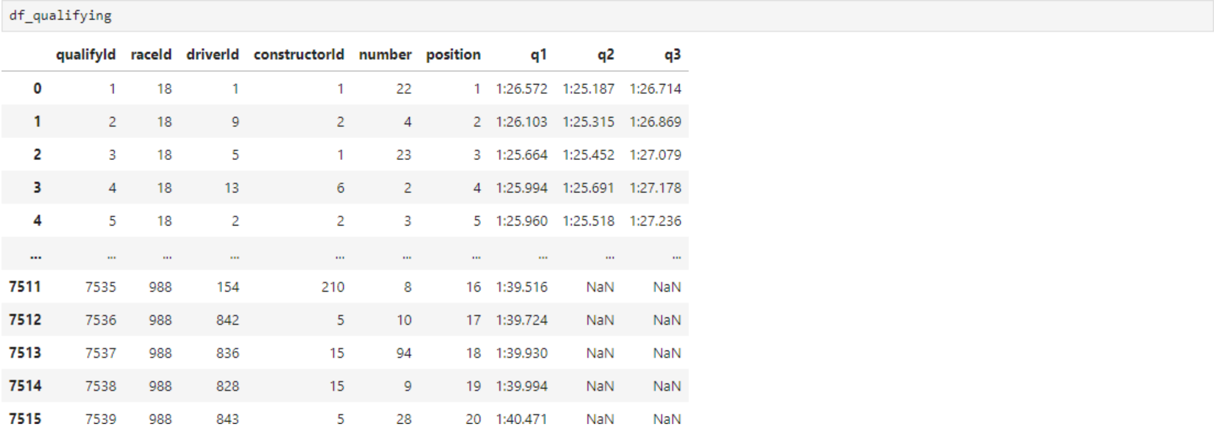
```
# Detecting duplicate rows
duplicated_rows = df_pitstops.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```

4.1.9. Qualifying Table

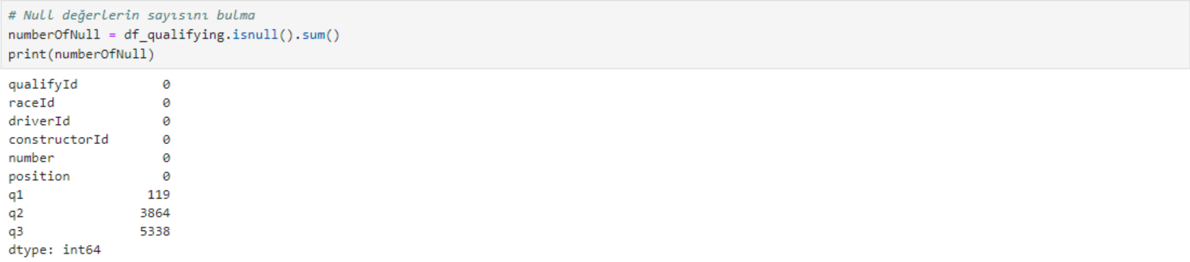


```
filepath = 'qualifying.csv'
df_qualifying = pd.read_csv(filepath)
```



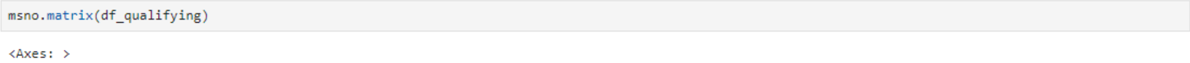
```
df_qualifying
```

	qualifyId	raceId	driverId	constructorId	number	position	q1	q2	q3
0	1	18	1	1	22	1	1:26.572	1:25.187	1:26.714
1	2	18	9	2	4	2	1:26.103	1:25.315	1:26.869
2	3	18	5	1	23	3	1:25.664	1:25.452	1:27.079
3	4	18	13	6	2	4	1:25.994	1:25.691	1:27.178
4	5	18	2	2	3	5	1:25.960	1:25.518	1:27.236
...	...	...	...	...	...	...	...	...	...
7511	7535	988	154	210	8	16	1:39.516	NaN	NaN
7512	7536	988	842	5	10	17	1:39.724	NaN	NaN
7513	7537	988	836	15	94	18	1:39.930	NaN	NaN
7514	7538	988	828	15	9	19	1:39.994	NaN	NaN
7515	7539	988	843	5	28	20	1:40.471	NaN	NaN



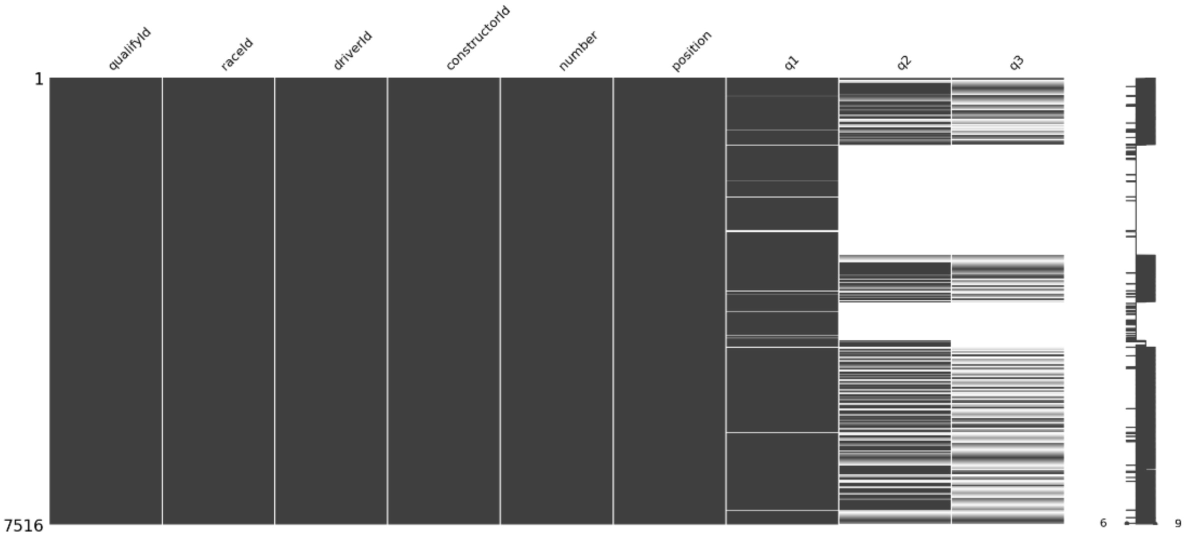
```
# Null değerlerin sayısını bulma
numberOfNull = df_qualifying.isnull().sum()
print(numberOfNull)
```

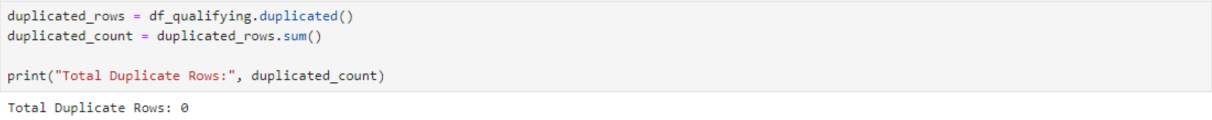
```
qualifyId           0
raceId              0
driverId            0
constructorId       0
number              0
position            0
q1                119
q2               3864
q3               5338
dtype: int64
```



```
msno.matrix(df_qualifying)
```

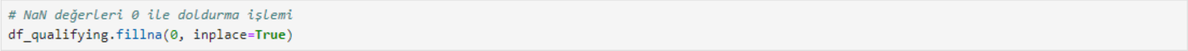
```
<Axes: >
```



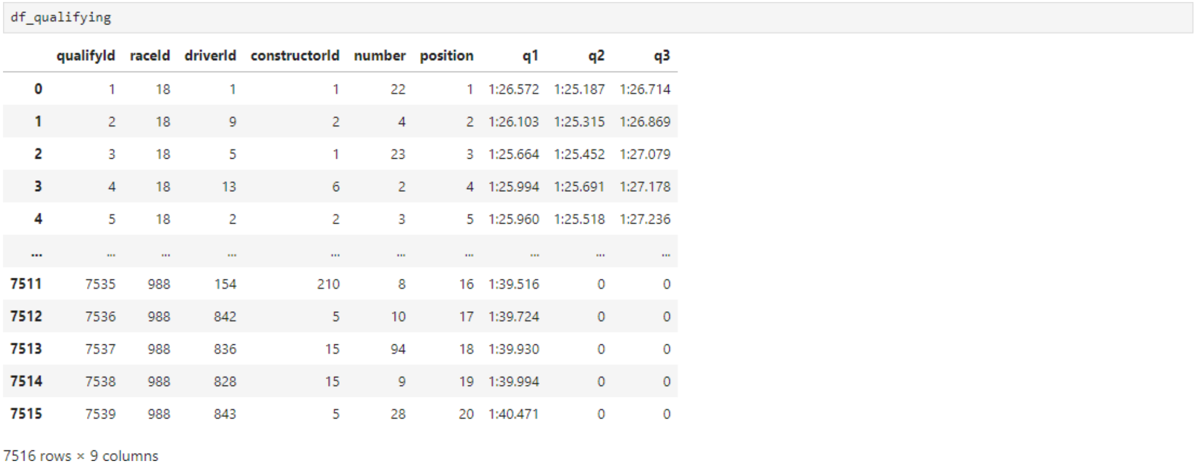


```
duplicated_rows = df_qualifying.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```



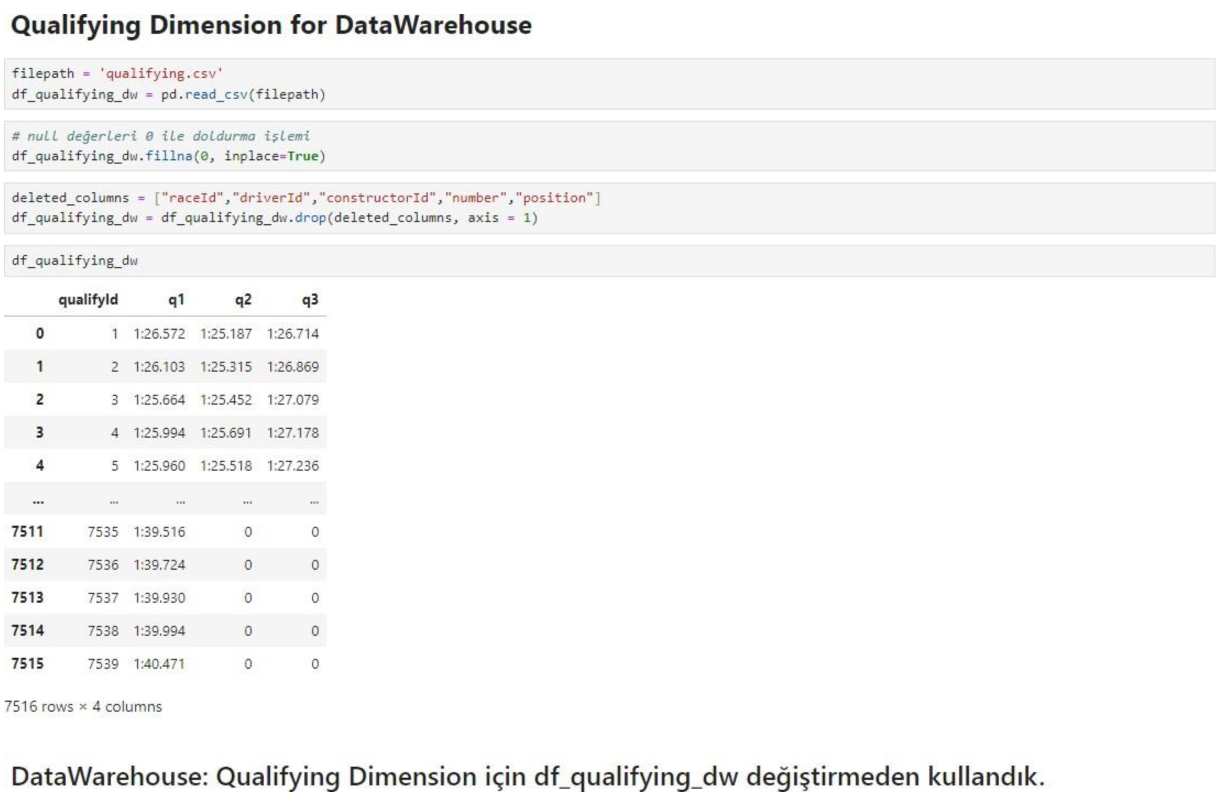
```
# NaN değerleri 0 ile doldurma işlemi
df_qualifying.fillna(0, inplace=True)
```



```
df_qualifying
```

	qualifyId	raceId	driverId	constructorId	number	position	q1	q2	q3
0	1	18	1	1	22	1	1:26.572	1:25.187	1:26.714
1	2	18	9	2	4	2	1:26.103	1:25.315	1:26.869
2	3	18	5	1	23	3	1:25.664	1:25.452	1:27.079
3	4	18	13	6	2	4	1:25.994	1:25.691	1:27.178
4	5	18	2	2	3	5	1:25.960	1:25.518	1:27.236
...	...	...	...	...	...	...	...	...	...
7511	7535	988	154	210	8	16	1:39.516	0	0
7512	7536	988	842	5	10	17	1:39.724	0	0
7513	7537	988	836	15	94	18	1:39.930	0	0
7514	7538	988	828	15	9	19	1:39.994	0	0
7515	7539	988	843	5	28	20	1:40.471	0	0

7516 rows × 9 columns



Qualifying Dimension for DataWarehouse

```
filepath = 'qualifying.csv'
df_qualifying_dw = pd.read_csv(filepath)
```

```
# null değerleri 0 ile doldurma işlemi
df_qualifying_dw.fillna(0, inplace=True)
```

```
deleted_columns = ["raceId","driverId","constructorId","number","position"]
df_qualifying_dw = df_qualifying_dw.drop(deleted_columns, axis = 1)
```

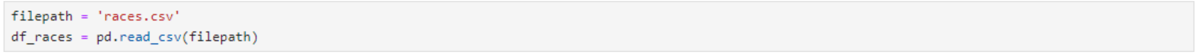
```
df_qualifying_dw
```

	qualifyId	q1	q2	q3
0	1	1:26.572	1:25.187	1:26.714
1	2	1:26.103	1:25.315	1:26.869
2	3	1:25.664	1:25.452	1:27.079
3	4	1:25.994	1:25.691	1:27.178
4	5	1:25.960	1:25.518	1:27.236
...	...	...	...	...
7511	7535	1:39.516	0	0
7512	7536	1:39.724	0	0
7513	7537	1:39.930	0	0
7514	7538	1:39.994	0	0
7515	7539	1:40.471	0	0

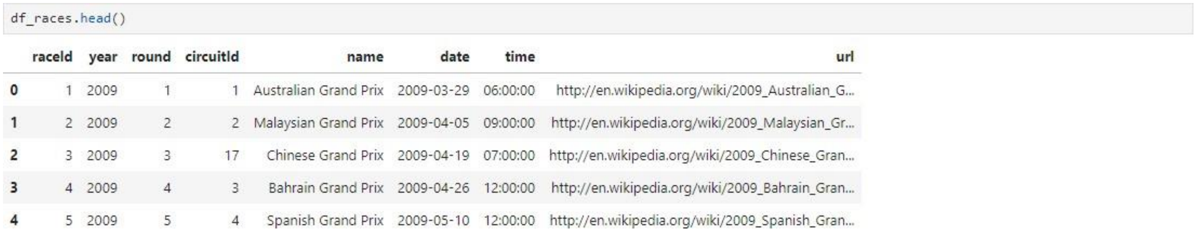
7516 rows × 4 columns

DataWarehouse: Qualifying Dimension için df_qualifying_dw değiştirmeden kullandık.

4.1.10. Races Table

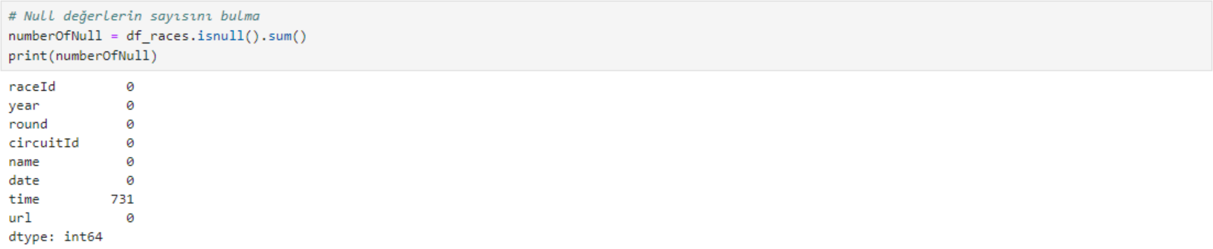


```
filepath = 'races.csv'
df_races = pd.read_csv(filepath)
```



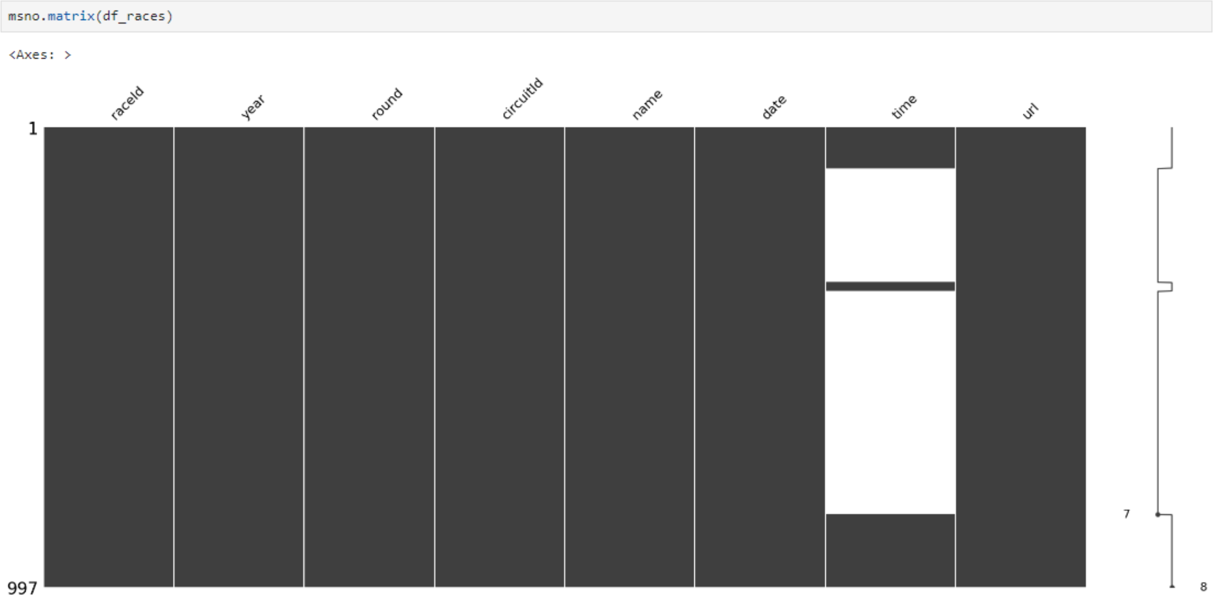
```
df_races.head()
```

	raceId	year	round	circuitId	name	date	time	url
0	1	2009	1	1	Australian Grand Prix	2009-03-29	06:00:00	http://en.wikipedia.org/wiki/2009_Australian_G...
1	2	2009	2	2	Malaysian Grand Prix	2009-04-05	09:00:00	http://en.wikipedia.org/wiki/2009_Malaysian_Gr...
2	3	2009	3	17	Chinese Grand Prix	2009-04-19	07:00:00	http://en.wikipedia.org/wiki/2009_Chinese_Gran...
3	4	2009	4	3	Bahrain Grand Prix	2009-04-26	12:00:00	http://en.wikipedia.org/wiki/2009_Bahrain_Gran...
4	5	2009	5	4	Spanish Grand Prix	2009-05-10	12:00:00	http://en.wikipedia.org/wiki/2009_Spanish_Gran...



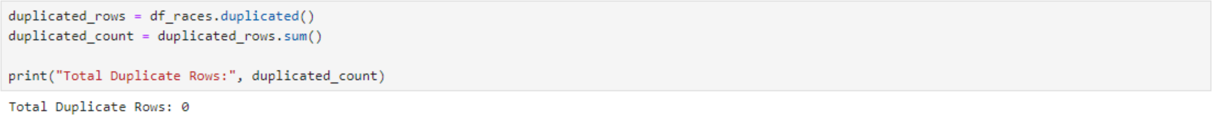
```
# Null değerlerin sayısını bulma
numberOfNull = df_races.isnull().sum()
print(numberOfNull)
```

```
raceId         0
year           0
round          0
circuitId      0
name           0
date           0
time         731
url            0
dtype: int64
```



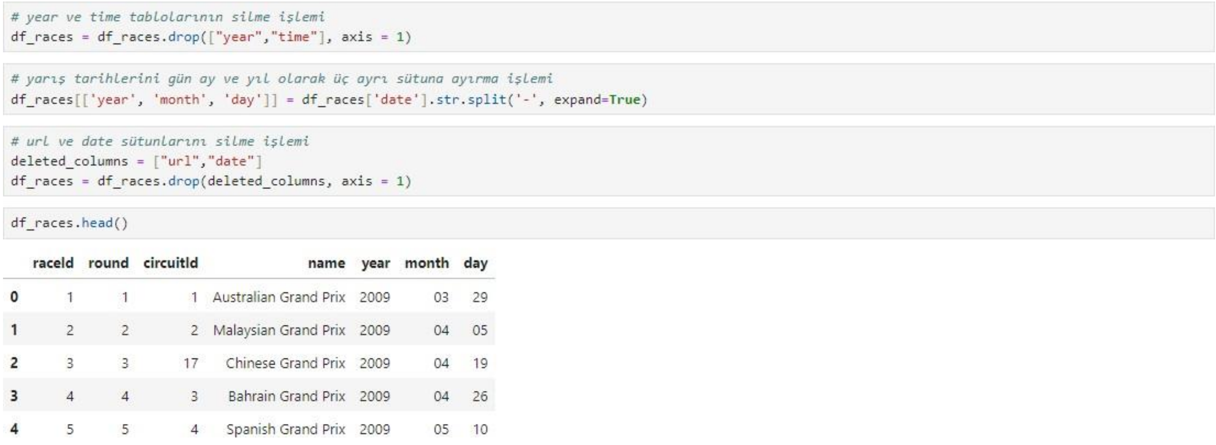
```
msno.matrix(df_races)
```

```
<Axes: >
```



```
duplicated_rows = df_races.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```



```
# year ve time tablolarının silme işlemi
df_races = df_races.drop(["year","time"], axis = 1)
```

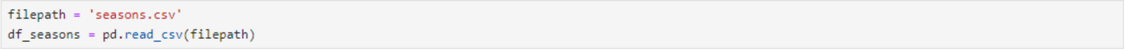
```
# yarış tarihlerini gün ay ve yıl olarak üç ayrı sütuna ayırma işlemi
df_races[['year', 'month', 'day']] = df_races['date'].str.split('-', expand=True)
```

```
# url ve date sütunlarını silme işlemi
deleted_columns = ["url","date"]
df_races = df_races.drop(deleted_columns, axis = 1)
```

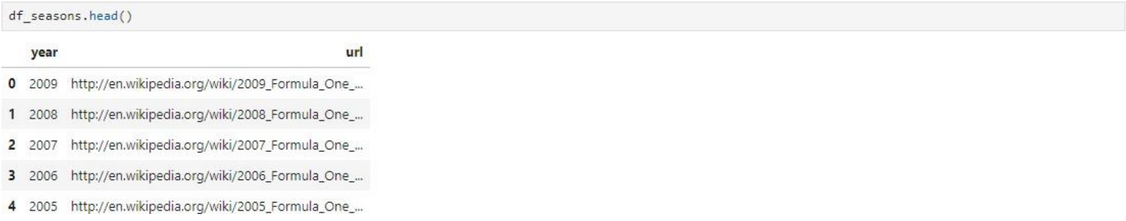
```
df_races.head()
```

	raceId	round	circuitId	name	year	month	day
0	1	1	1	Australian Grand Prix	2009	03	29
1	2	2	2	Malaysian Grand Prix	2009	04	05
2	3	3	17	Chinese Grand Prix	2009	04	19
3	4	4	3	Bahrain Grand Prix	2009	04	26
4	5	5	4	Spanish Grand Prix	2009	05	10

4.1.11. Seasons Table



```
filepath = 'seasons.csv'
df_seasons = pd.read_csv(filepath)
```



```
df_seasons.head()
```

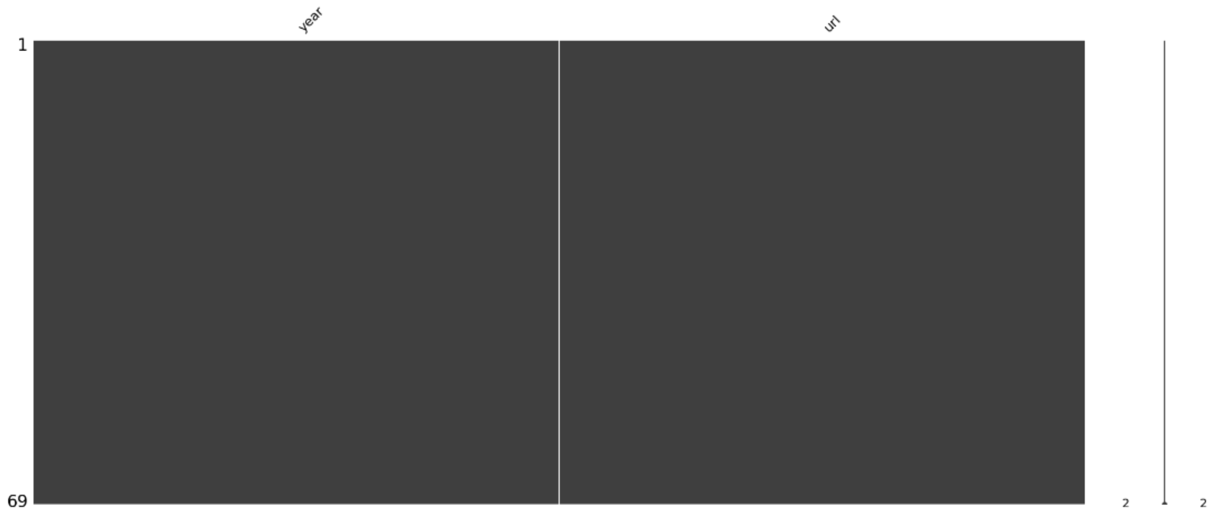
	year	url
0	2009	http://en.wikipedia.org/wiki/2009_Formula_One_...
1	2008	http://en.wikipedia.org/wiki/2008_Formula_One_...
2	2007	http://en.wikipedia.org/wiki/2007_Formula_One_...
3	2006	http://en.wikipedia.org/wiki/2006_Formula_One_...
4	2005	http://en.wikipedia.org/wiki/2005_Formula_One_...

```
# Null değerlerin sayısını bulma
numberOfNull = df_seasons.isnull().sum()
print(numberOfNull)
```

```
year    0
url     0
dtype: int64
```

```
msno.matrix(df_seasons)
```

```
<Axes: >
```



```
# Detecting duplicate rows
duplicated_rows = df_seasons.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```

```
# url sütunlarını silme işlemi
deleted_columns = ["url"]
df_seasons = df_seasons.drop(deleted_columns, axis = 1)
```

```
df_seasons.head()
```

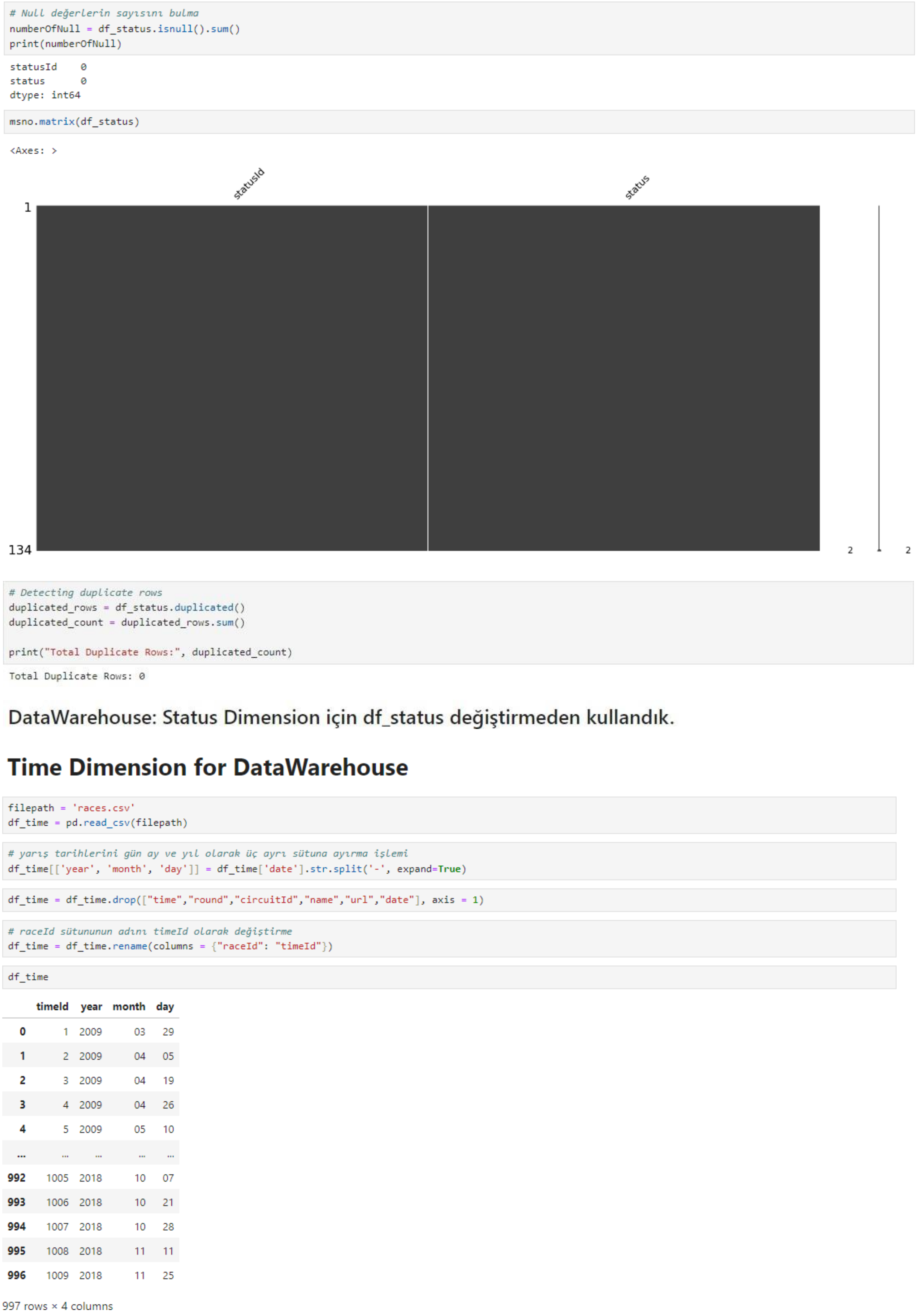
	year
0	2009
1	2008
2	2007
3	2006
4	2005

4.1.12. Status Table

```
filepath = 'status.csv'
df_status = pd.read_csv(filepath)
```

```
df_status.head()
```

	statusId	status
0	1	Finished
1	2	Disqualified
2	3	Accident
3	4	Collision
4	5	Engine



```
# Null değerlerin sayısını bulma
numberOfNull = df_status.isnull().sum()
print(numberOfNull)
```

```
statusId    0
status      0
dtype: int64
```

```
msno.matrix(df_status)
```

```
<Axes: >
```

```
# Detecting duplicate rows
duplicated_rows = df_status.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

```
Total Duplicate Rows: 0
```

DataWarehouse: Status Dimension için df_status değiştirmeden kullandık.

Time Dimension for DataWarehouse

```
filepath = 'races.csv'
df_time = pd.read_csv(filepath)
```

```
# yarış tarihlerini gün ay ve yıl olarak üç ayrı sütuna ayırma işlemi
df_time[['year', 'month', 'day']] = df_time['date'].str.split('-', expand=True)
```

```
df_time = df_time.drop(["time","round","circuitId","name","url","date"], axis = 1)
```

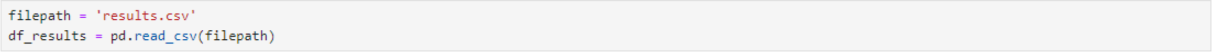
```
# raceId sütununun adını timeId olarak değiştirme
df_time = df_time.rename(columns = {"raceId": "timeId"})
```

```
df_time
```

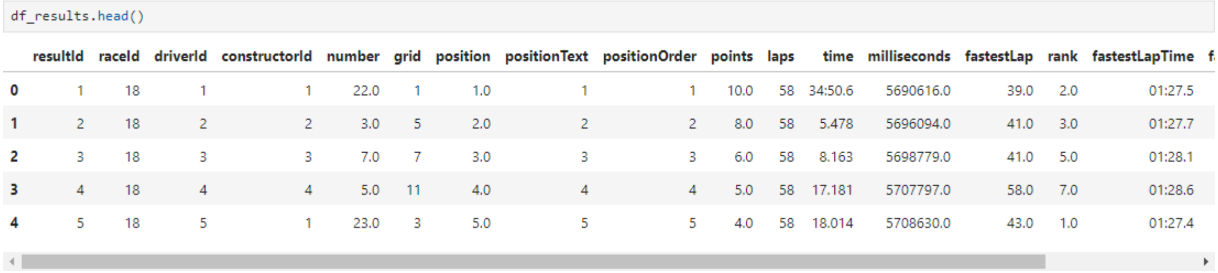
	timeId	year	month	day
0	1	2009	03	29
1	2	2009	04	05
2	3	2009	04	19
3	4	2009	04	26
4	5	2009	05	10
...	...	...	...	...
992	1005	2018	10	07
993	1006	2018	10	21
994	1007	2018	10	28
995	1008	2018	11	11
996	1009	2018	11	25

997 rows × 4 columns

4.1.13. Results Table

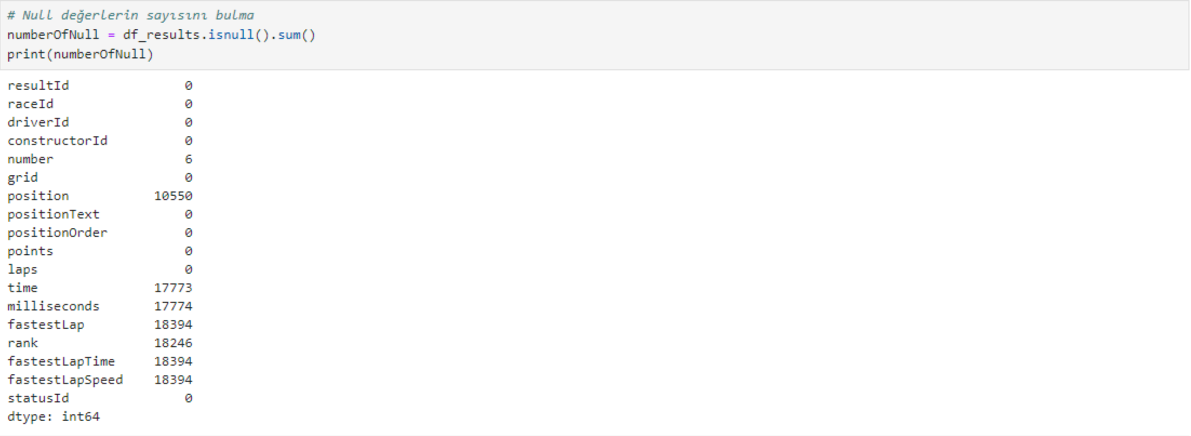


```
filepath = 'results.csv'
df_results = pd.read_csv(filepath)
```



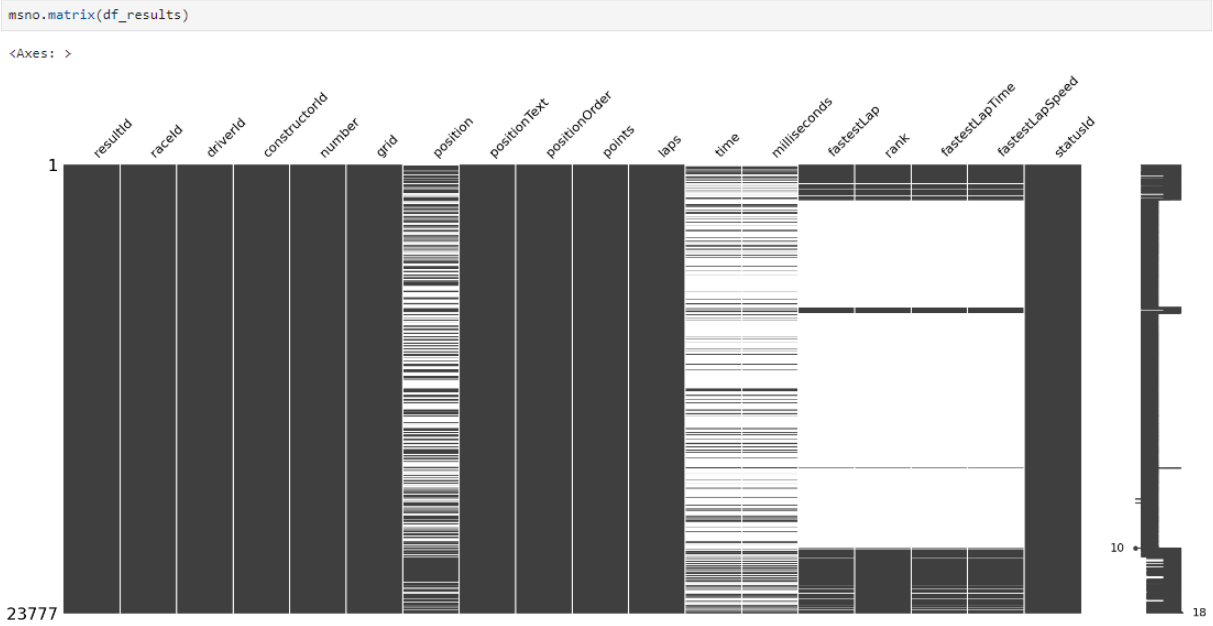
```
df_results.head()
```

	resultId	raceId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds	fastestLap	rank	fastestLapTime	f
0	1	18	1	1	22.0	1	1.0	1	1	10.0	58	34:50.6	5690616.0	39.0	2.0	01:27.5	
1	2	18	2	2	3.0	5	2.0	2	2	8.0	58	5.478	5696094.0	41.0	3.0	01:27.7	
2	3	18	3	3	7.0	7	3.0	3	3	6.0	58	8.163	5698779.0	41.0	5.0	01:28.1	
3	4	18	4	4	5.0	11	4.0	4	4	5.0	58	17.181	5707797.0	58.0	7.0	01:28.6	
4	5	18	5	1	23.0	3	5.0	5	5	4.0	58	18.014	5708630.0	43.0	1.0	01:27.4	



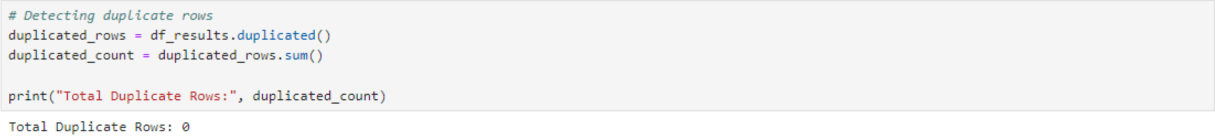
```
# Null değerlerin sayısını bulma
numberOfNull = df_results.isnull().sum()
print(numberOfNull)
```

```
resultId               0
raceId                 0
driverId               0
constructorId          0
number                 6
grid                   0
position           10550
positionText           0
positionOrder          0
points                 0
laps                   0
time               17773
milliseconds       17774
fastestLap         18394
rank               18246
fastestLapTime     18394
fastestLapSpeed    18394
statusId               0
dtype: int64
```



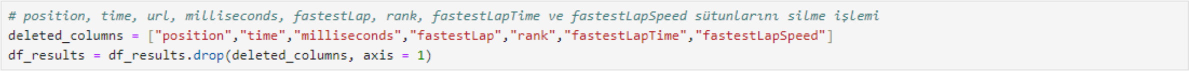
```
msno.matrix(df_results)
```

```
<Axes: >
```

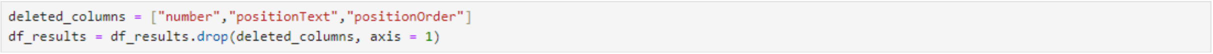


```
# Detecting duplicate rows
duplicated_rows = df_results.duplicated()
duplicated_count = duplicated_rows.sum()

print("Total Duplicate Rows:", duplicated_count)
```

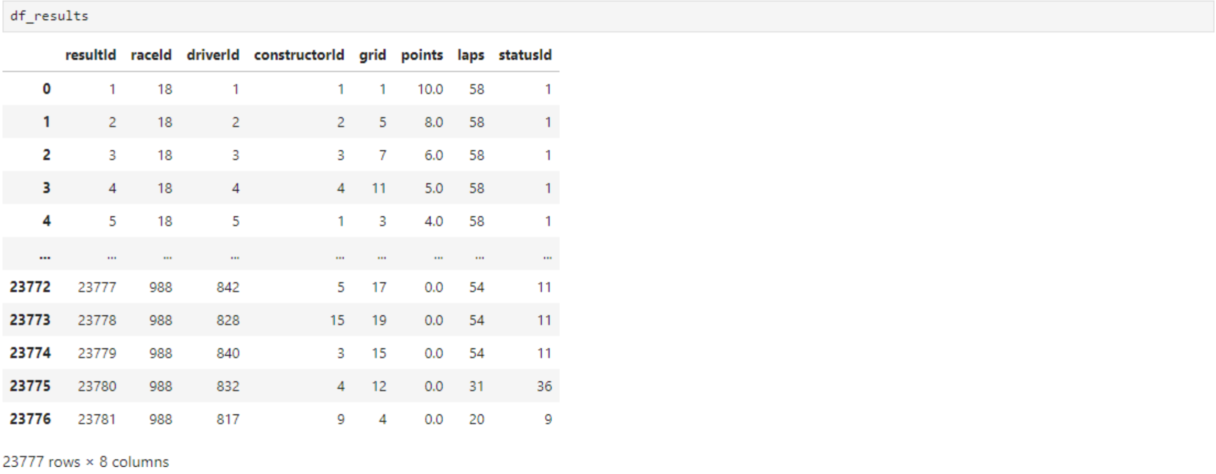
```
Total Duplicate Rows: 0
```



```
# position, time, url, milliseconds, fastestLap, rank, fastestLapTime ve fastestLapSpeed sütunlarını silme işlemi
deleted_columns = ["position","time","milliseconds","fastestLap","rank","fastestLapTime","fastestLapSpeed"]
df_results = df_results.drop(deleted_columns, axis = 1)
```



```
deleted_columns = ["number","positionText","positionOrder"]
df_results = df_results.drop(deleted_columns, axis = 1)
```



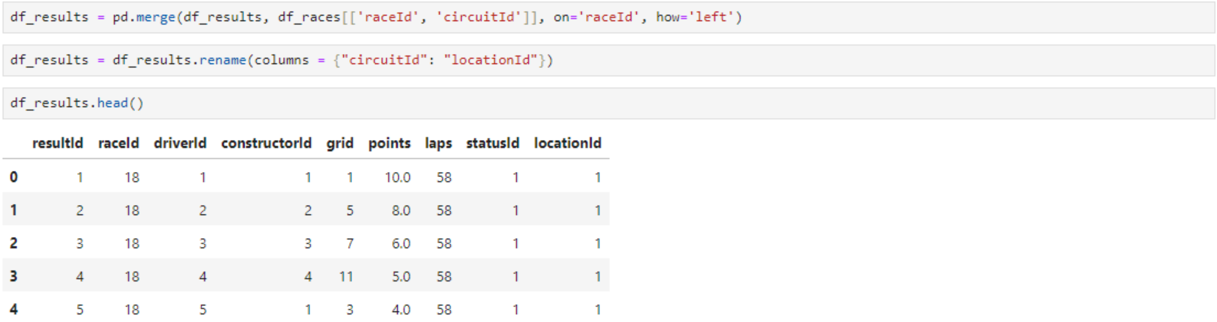
```
df_results
```

	resultId	raceId	driverId	constructorId	grid	points	laps	statusId
0	1	18	1	1	1	10.0	58	1
1	2	18	2	2	5	8.0	58	1
2	3	18	3	3	7	6.0	58	1
3	4	18	4	4	11	5.0	58	1
4	5	18	5	1	3	4.0	58	1
...	...	...	...	...	...	...	...	...
23772	23777	988	842	5	17	0.0	54	11
23773	23778	988	828	15	19	0.0	54	11
23774	23779	988	840	3	15	0.0	54	11
23775	23780	988	832	4	12	0.0	31	36
23776	23781	988	817	9	4	0.0	20	9

23777 rows × 8 columns

4.1.14. Creating the Fact Table

df_results tablosuna Location Dimension'ın id sini ekleme. Bunu yapmak içinde races tablosundaki circuit_id merge ettik.



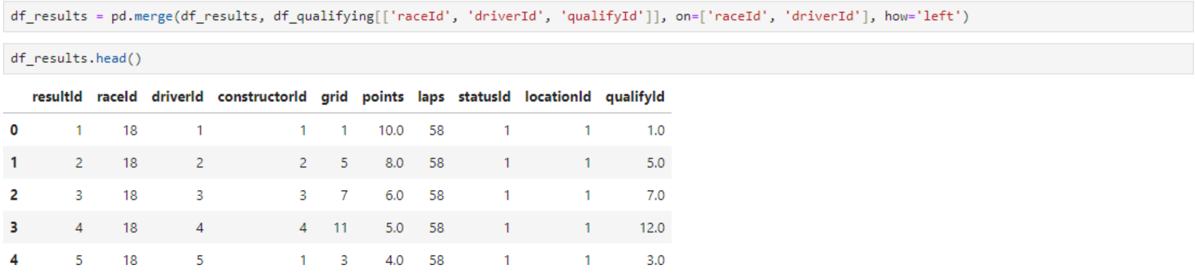
```
df_results = pd.merge(df_results, df_races[['raceId', 'circuitId']], on='raceId', how='left')
```

```
df_results = df_results.rename(columns = {"circuitId": "locationId"})
```

```
df_results.head()
```

	resultId	raceId	driverId	constructorId	grid	points	laps	statusId	locationId
0	1	18	1	1	1	10.0	58	1	1
1	2	18	2	2	5	8.0	58	1	1
2	3	18	3	3	7	6.0	58	1	1
3	4	18	4	4	11	5.0	58	1	1
4	5	18	5	1	3	4.0	58	1	1

df_results tablosuna Qualifying Dimension'ın id sini ekleme. Bunu yapmak içinde qualifying tablosundaki qualifyId merge ettik.

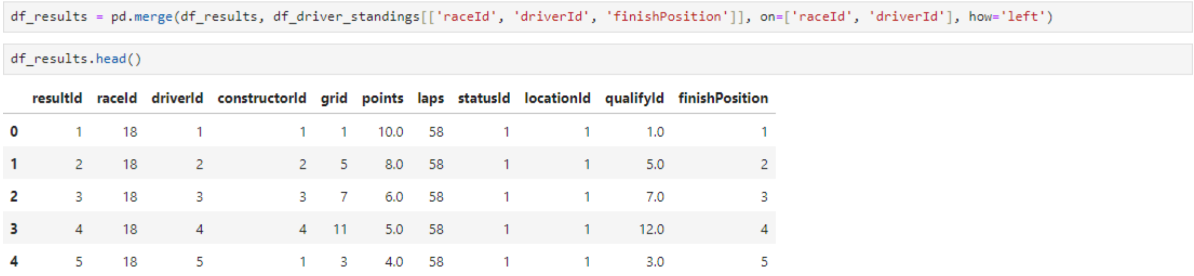


```
df_results = pd.merge(df_results, df_qualifying[['raceId', 'driverId', 'qualifyId']], on=['raceId', 'driverId'], how='left')
```

```
df_results.head()
```

	resultId	raceId	driverId	constructorId	grid	points	laps	statusId	locationId	qualifyId
0	1	18	1	1	1	10.0	58	1	1	1.0
1	2	18	2	2	5	8.0	58	1	1	5.0
2	3	18	3	3	7	6.0	58	1	1	7.0
3	4	18	4	4	11	5.0	58	1	1	12.0
4	5	18	5	1	3	4.0	58	1	1	3.0

df_results tablosuna finishPosition ekleme. Bunu yapmak içinde df_driver_standings tablosundaki finishPosition merge ettik.

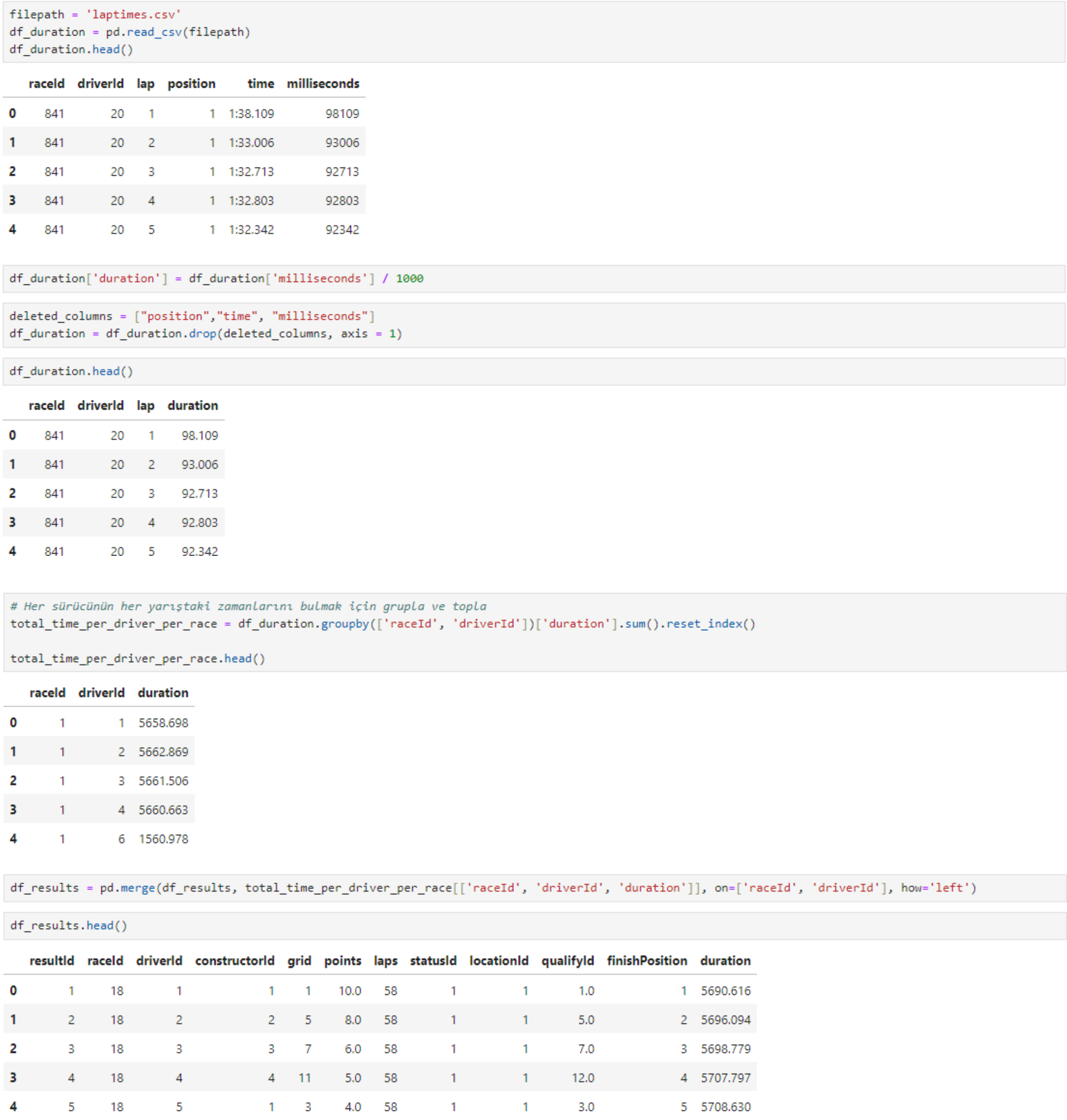


```
df_results = pd.merge(df_results, df_driver_standings[['raceId', 'driverId', 'finishPosition']], on=['raceId', 'driverId'], how='left')
```

```
df_results.head()
```

	resultId	raceId	driverId	constructorId	grid	points	laps	statusId	locationId	qualifyId	finishPosition
0	1	18	1	1	1	10.0	58	1	1	1.0	1
1	2	18	2	2	5	8.0	58	1	1	5.0	2
2	3	18	3	3	7	6.0	58	1	1	7.0	3
3	4	18	4	4	11	5.0	58	1	1	12.0	4
4	5	18	5	1	3	4.0	58	1	1	3.0	5

Her yarışçının her yarışta geçirdiği toplam süreyi laptimes tablosundan elde ettik.



```
filepath = 'laptimes.csv'
df_duration = pd.read_csv(filepath)
df_duration.head()
```

	raceId	driverId	lap	position	time	milliseconds
0	841	20	1	1	1:38.109	98109
1	841	20	2	1	1:33.006	93006
2	841	20	3	1	1:32.713	92713
3	841	20	4	1	1:32.803	92803
4	841	20	5	1	1:32.342	92342

```
df_duration['duration'] = df_duration['milliseconds'] / 1000
```

```
deleted_columns = ["position","time", "milliseconds"]
df_duration = df_duration.drop(deleted_columns, axis = 1)
```

```
df_duration.head()
```

	raceId	driverId	lap	duration
0	841	20	1	98.109
1	841	20	2	93.006
2	841	20	3	92.713
3	841	20	4	92.803
4	841	20	5	92.342

```
# Her sürücünün her yarıştaki zamanlarını bulmak için grupla ve topla
total_time_per_driver_per_race = df_duration.groupby(['raceId', 'driverId'])['duration'].sum().reset_index()

total_time_per_driver_per_race.head()
```

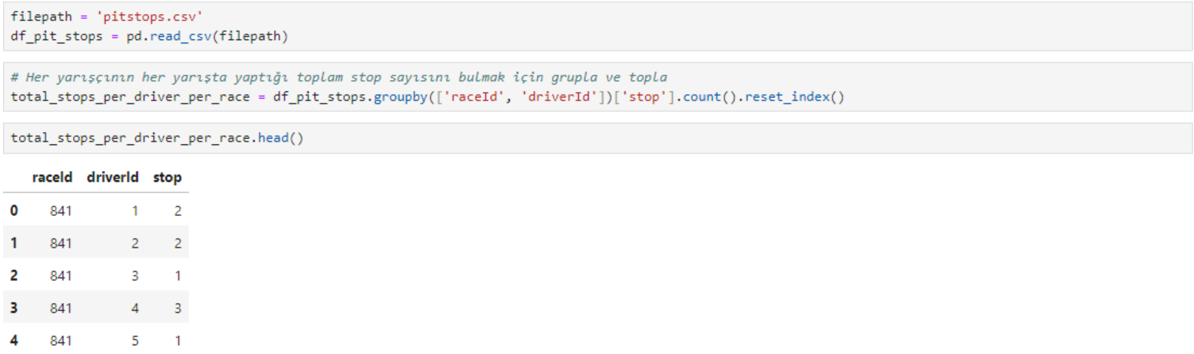
	raceId	driverId	duration
0	1	1	5658.698
1	1	2	5662.869
2	1	3	5661.506
3	1	4	5660.663
4	1	6	1560.978

```
df_results = pd.merge(df_results, total_time_per_driver_per_race[['raceId', 'driverId', 'duration']], on=['raceId', 'driverId'], how='left')
```

```
df_results.head()
```

	resultId	raceId	driverId	constructorId	grid	points	laps	statusId	locationId	qualifyId	finishPosition	duration
0	1	18	1	1	1	10.0	58	1	1	1.0	1	5690.616
1	2	18	2	2	5	8.0	58	1	1	5.0	2	5696.094
2	3	18	3	3	7	6.0	58	1	1	7.0	3	5698.779
3	4	18	4	4	11	5.0	58	1	1	12.0	4	5707.797
4	5	18	5	1	3	4.0	58	1	1	3.0	5	5708.630

Her yarışçının her yarışta yaptığı toplam pitStop sayısı pitstops tablosundan elde ettik.

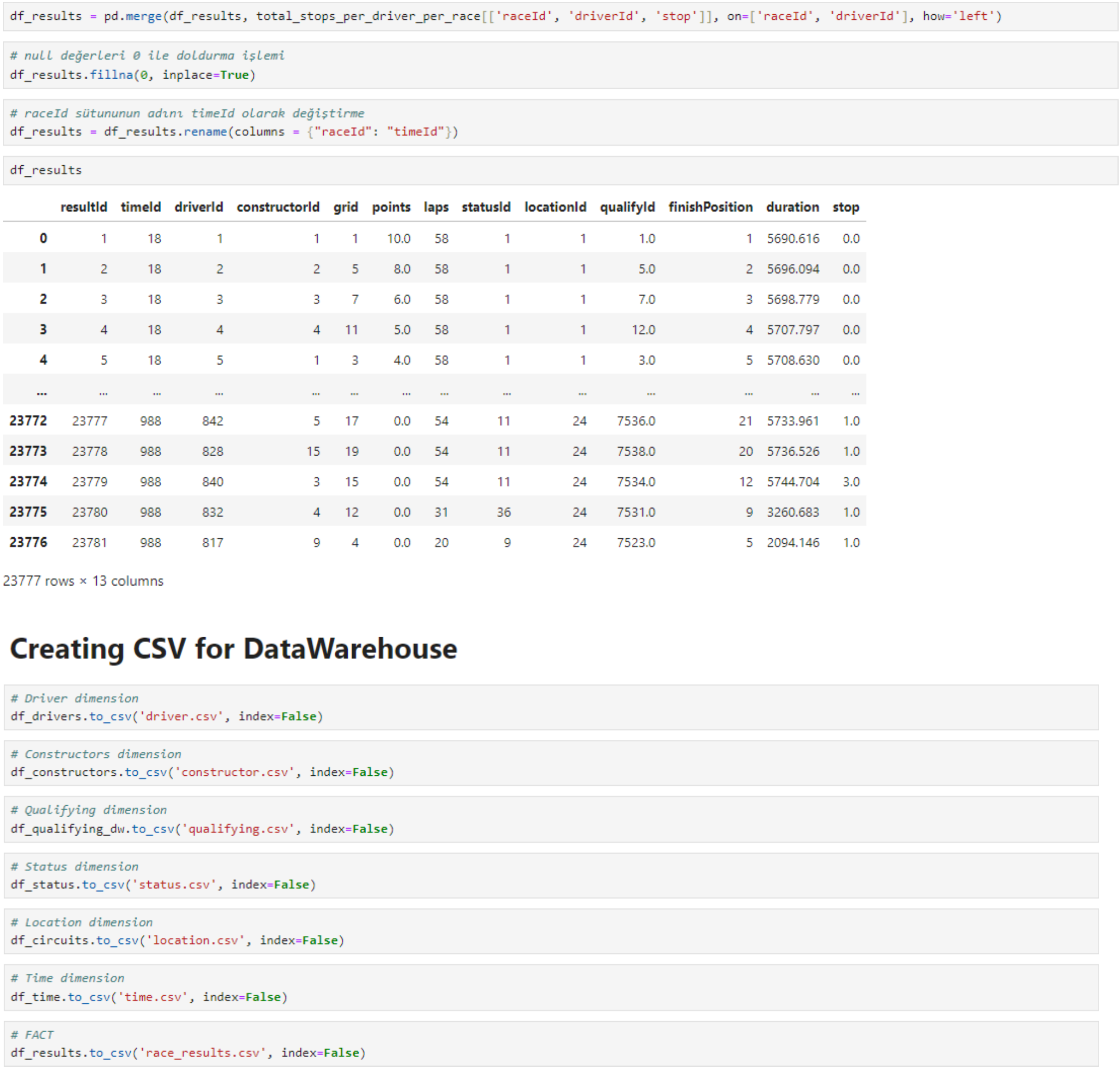


```
filepath = 'pitstops.csv'
df_pit_stops = pd.read_csv(filepath)
```

```
# Her yarışçının her yarışta yaptığı toplam stop sayısını bulmak için grupla ve topla
total_stops_per_driver_per_race = df_pit_stops.groupby(['raceId', 'driverId'])['stop'].count().reset_index()
```

```
total_stops_per_driver_per_race.head()
```

	raceId	driverId	stop
0	841	1	2
1	841	2	2
2	841	3	1
3	841	4	3
4	841	5	1



```
df_results = pd.merge(df_results, total_stops_per_driver_per_race[['raceId', 'driverId', 'stop']], on=['raceId', 'driverId'], how='left')
```

```
# null değerleri 0 ile doldurma işlemi
df_results.fillna(0, inplace=True)
```

```
# raceId sütununun adını timeId olarak değiştirme
df_results = df_results.rename(columns = {"raceId": "timeId"})
```

```
df_results
```

	resultId	timeId	driverId	constructorId	grid	points	laps	statusId	locationId	qualifyId	finishPosition	duration	stop
0	1	18	1	1	1	10.0	58	1	1	1.0	1	5690.616	0.0
1	2	18	2	2	5	8.0	58	1	1	5.0	2	5696.094	0.0
2	3	18	3	3	7	6.0	58	1	1	7.0	3	5698.779	0.0
3	4	18	4	4	11	5.0	58	1	1	12.0	4	5707.797	0.0
4	5	18	5	1	3	4.0	58	1	1	3.0	5	5708.630	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
23772	23777	988	842	5	17	0.0	54	11	24	7536.0	21	5733.961	1.0
23773	23778	988	828	15	19	0.0	54	11	24	7538.0	20	5736.526	1.0
23774	23779	988	840	3	15	0.0	54	11	24	7534.0	12	5744.704	3.0
23775	23780	988	832	4	12	0.0	31	36	24	7531.0	9	3260.683	1.0
23776	23781	988	817	9	4	0.0	20	9	24	7523.0	5	2094.146	1.0

23777 rows × 13 columns

Creating CSV for DataWarehouse

```
# Driver dimension
df_drivers.to_csv('driver.csv', index=False)
```

```
# Constructors dimension
df_constructors.to_csv('constructor.csv', index=False)
```

```
# Qualifying dimension
df_qualifying_dw.to_csv('qualifying.csv', index=False)
```

```
# Status dimension
df_status.to_csv('status.csv', index=False)
```

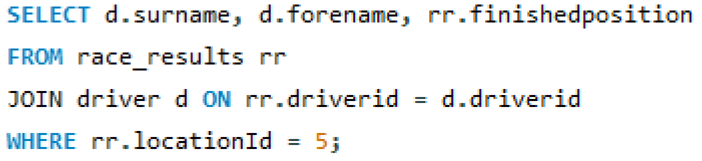
```
# Location dimension
df_circuits.to_csv('location.csv', index=False)
```

```
# Time dimension
df_time.to_csv('time.csv', index=False)
```

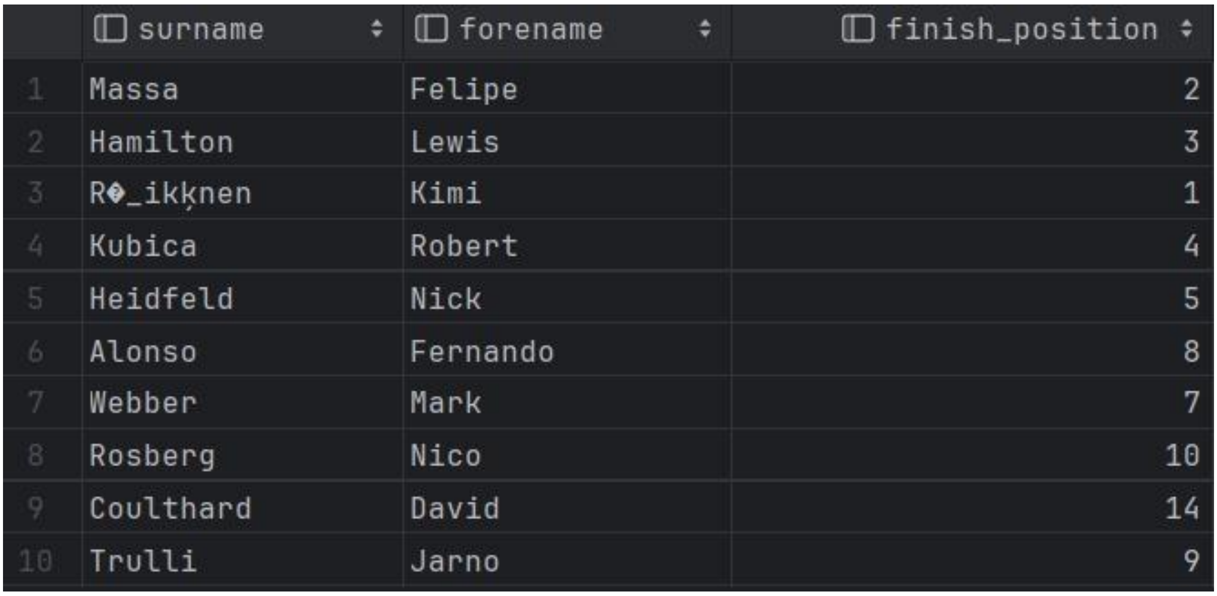
```
# FACT
df_results.to_csv('race_results.csv', index=False)
```

5. SQL Queries for Data Warehouse

→Names and finishing positions of drivers in a given race: (e.g. for 5)

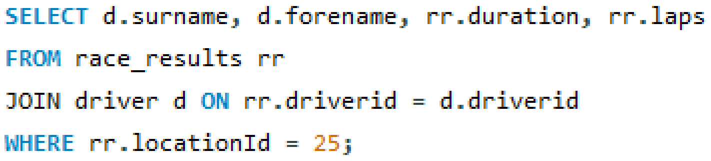


```
SELECT d.surname, d.forename, rr.finishedposition
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
WHERE rr.locationId = 5;
```

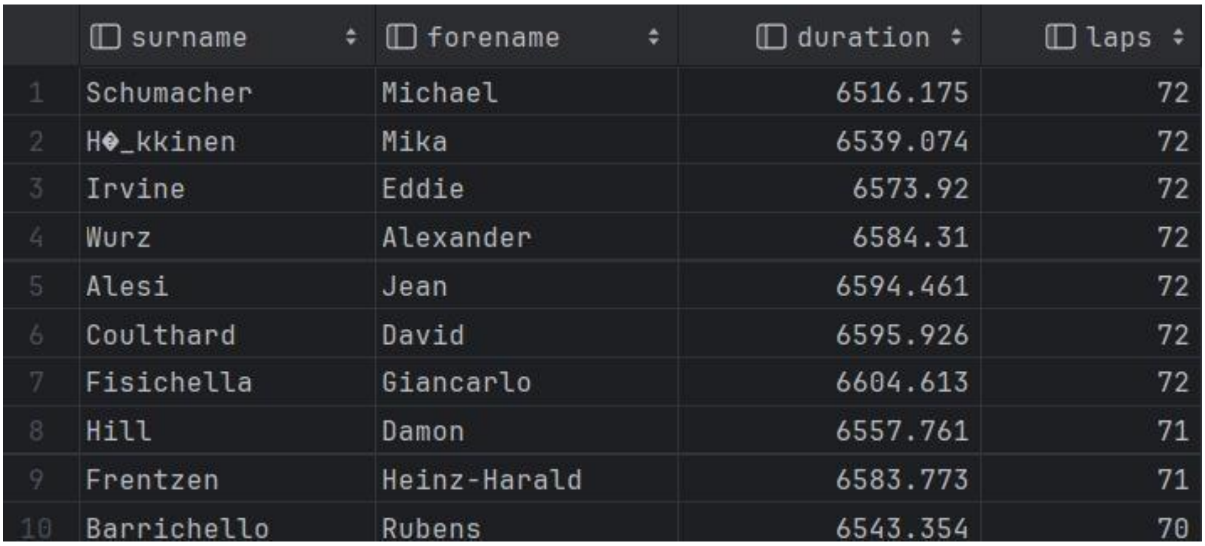


	surname	forename	finish_position
1	Massa	Felipe	2
2	Hamilton	Lewis	3
3	R�_ikknen	Kimi	1
4	Kubica	Robert	4
5	Heidfeld	Nick	5
6	Alonso	Fernando	8
7	Webber	Mark	7
8	Rosberg	Nico	10
9	Coulthard	David	14
10	Trulli	Jarno	9

→Drivers' finish times and number of laps in a given race: (e.g. for 25)

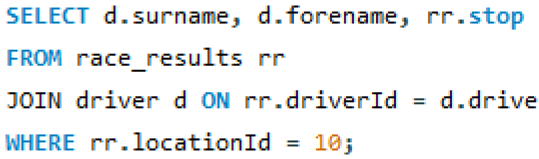


```
SELECT d.surname, d.forename, rr.duration, rr.laps
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
WHERE rr.locationId = 25;
```

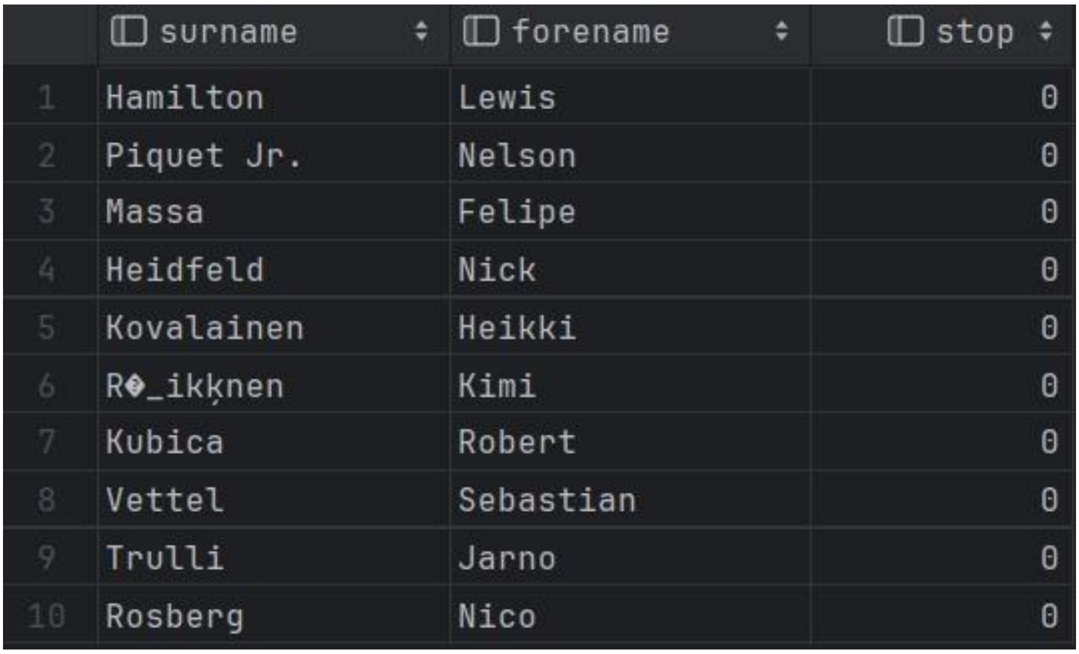


	surname	forename	duration	laps
1	Schumacher	Michael	6516.175	72
2	H�_kkinen	Mika	6539.074	72
3	Irvine	Eddie	6573.92	72
4	Wurz	Alexander	6584.31	72
5	Alesi	Jean	6594.461	72
6	Coulthard	David	6595.926	72
7	Fisichella	Giancarlo	6604.613	72
8	Hill	Damon	6557.761	71
9	Frentzen	Heinz-Harald	6583.773	71
10	Barrichello	Rubens	6543.354	70

→The number of stops for drivers in a given race: (e.g. for 10)

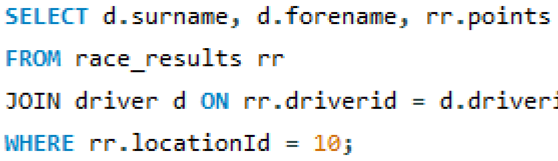


```
SELECT d.surname, d.forename, rr.stop
FROM race_results rr
JOIN driver d ON rr.driverId = d.driverId
WHERE rr.locationId = 10;
```

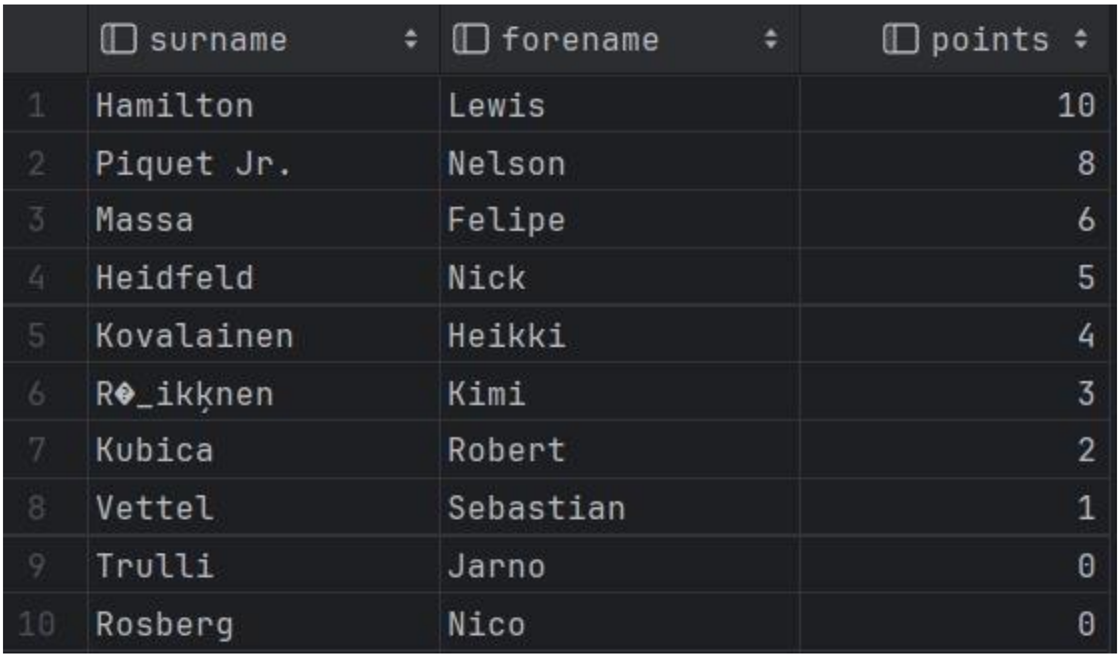


	surname	forename	stop
1	Hamilton	Lewis	0
2	Piquet Jr.	Nelson	0
3	Massa	Felipe	0
4	Heidfeld	Nick	0
5	Kovalainen	Heikki	0
6	R�_ikķnen	Kimi	0
7	Kubica	Robert	0
8	Vettel	Sebastian	0
9	Trulli	Jarno	0
10	Rosberg	Nico	0

→The points scored by drivers in a particular race: (e.g. for 10)

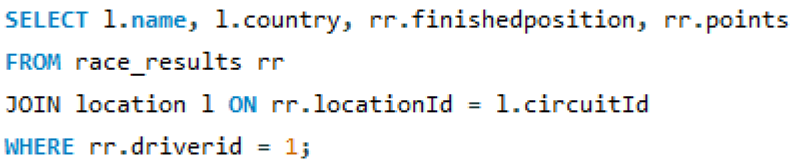


```
SELECT d.surname, d.forename, rr.points
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
WHERE rr.locationId = 10;
```

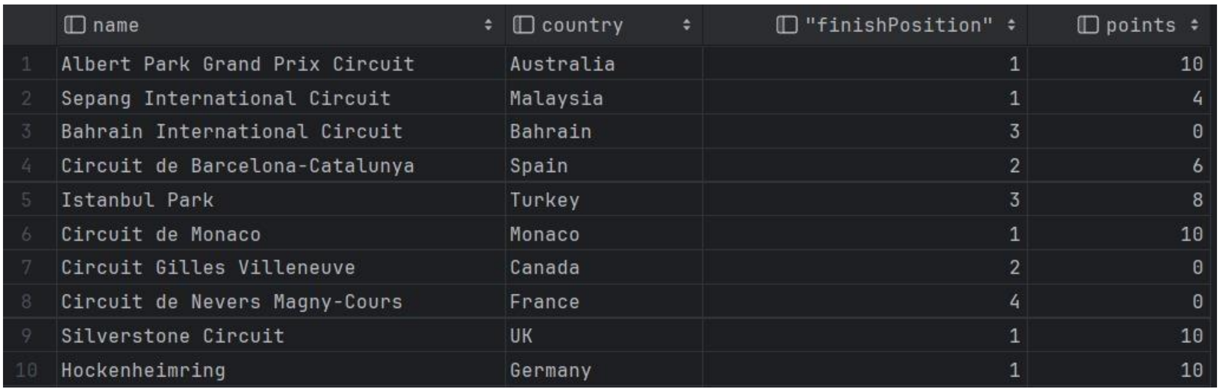


	surname	forename	points
1	Hamilton	Lewis	10
2	Piquet Jr.	Nelson	8
3	Massa	Felipe	6
4	Heidfeld	Nick	5
5	Kovalainen	Heikki	4
6	R�_ikķnen	Kimi	3
7	Kubica	Robert	2
8	Vettel	Sebastian	1
9	Trulli	Jarno	0
10	Rosberg	Nico	0

→The racing performance of a particular driver: (e.g. for 1)

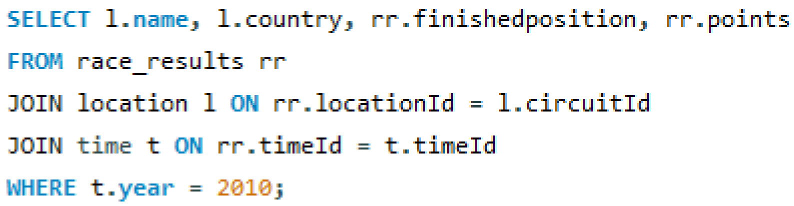


```
SELECT l.name, l.country, rr.finishedposition, rr.points
FROM race_results rr
JOIN location l ON rr.locationId = l.circuitId
WHERE rr.driverid = 1;
```

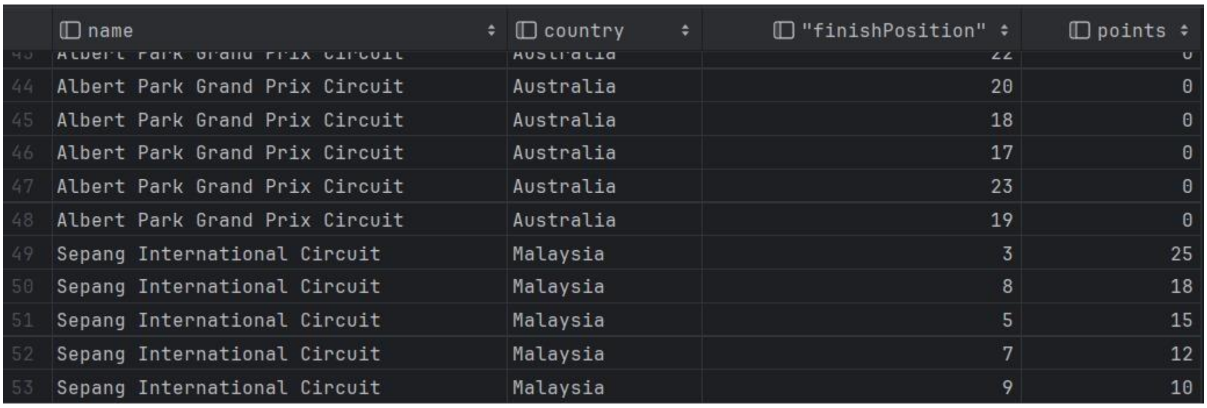


	name	country	"finishPosition"	points
1	Albert Park Grand Prix Circuit	Australia	1	10
2	Sepang International Circuit	Malaysia	1	4
3	Bahrain International Circuit	Bahrain	3	0
4	Circuit de Barcelona-Catalunya	Spain	2	6
5	Istanbul Park	Turkey	3	8
6	Circuit de Monaco	Monaco	1	10
7	Circuit Gilles Villeneuve	Canada	2	0
8	Circuit de Nevers Magny-Cours	France	4	0
9	Silverstone Circuit	UK	1	10
10	Hockenheimring	Germany	1	10

→All races and results in a given year: (e.g. for 2010)



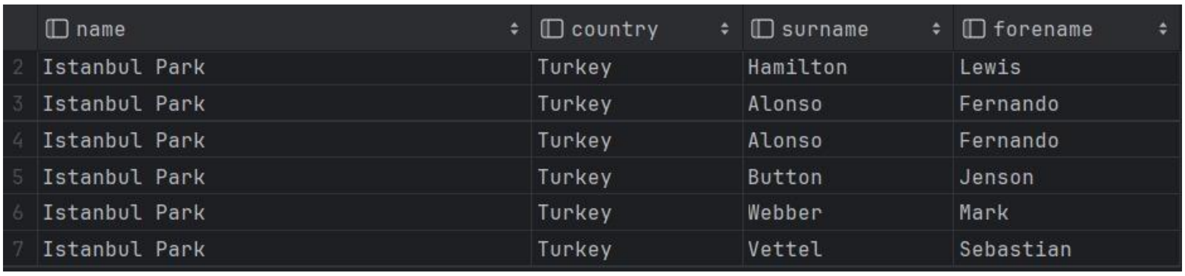
```
SELECT l.name, l.country, rr.finishedposition, rr.points
FROM race_results rr
JOIN location l ON rr.locationId = l.circuitId
JOIN time t ON rr.timeId = t.timeId
WHERE t.year = 2010;
```



	name	country	"finishPosition"	points
43	Albert Park Grand Prix Circuit	Australia	22	0
44	Albert Park Grand Prix Circuit	Australia	20	0
45	Albert Park Grand Prix Circuit	Australia	18	0
46	Albert Park Grand Prix Circuit	Australia	17	0
47	Albert Park Grand Prix Circuit	Australia	23	0
48	Albert Park Grand Prix Circuit	Australia	19	0
49	Sepang International Circuit	Malaysia	3	25
50	Sepang International Circuit	Malaysia	8	18
51	Sepang International Circuit	Malaysia	5	15
52	Sepang International Circuit	Malaysia	7	12
53	Sepang International Circuit	Malaysia	9	10

→Winners of races in a particular country: (e.g. for Turkey)

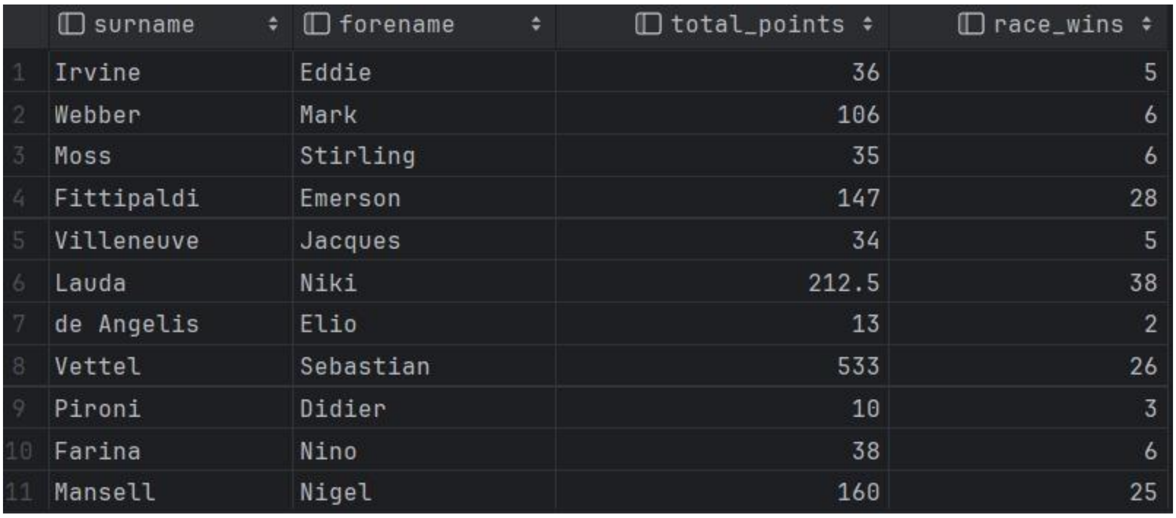
```
SELECT l.name, l.country, d.surname, d.forename
FROM race_results rr
JOIN location l ON rr.locationId = l.circuitId
JOIN driver d ON rr.driverid = d.driverid
WHERE rr.finishedposition = 1 AND l.country = 'Turkey';
```



	name	country	surname	forename
2	Istanbul Park	Turkey	Hamilton	Lewis
3	Istanbul Park	Turkey	Alonso	Fernando
4	Istanbul Park	Turkey	Alonso	Fernando
5	Istanbul Park	Turkey	Button	Jenson
6	Istanbul Park	Turkey	Webber	Mark
7	Istanbul Park	Turkey	Vettel	Sebastian

→The total number of points and races won in a driver's career:

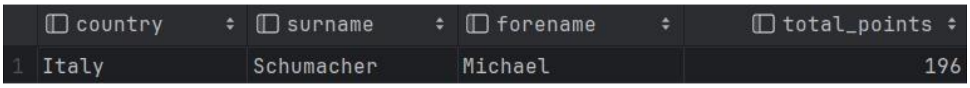
```
SELECT d.surname, d.forename, SUM(rr.points) AS total_points, COUNT(rr.finishedposition) AS race_wins
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
WHERE rr.finishedposition = 1
GROUP BY d.surname, d.forename;
```



	surname	forename	total_points	race_wins
1	Irvine	Eddie	36	5
2	Webber	Mark	106	6
3	Moss	Stirling	35	6
4	Fittipaldi	Emerson	147	28
5	Villeneuve	Jacques	34	5
6	Lauda	Niki	212.5	38
7	de Angelis	Elio	13	2
8	Vettel	Sebastian	533	26
9	Pironi	Didier	10	3
10	Farina	Nino	38	6
11	Mansell	Nigel	160	25

→The driver with the most points in a country's races:

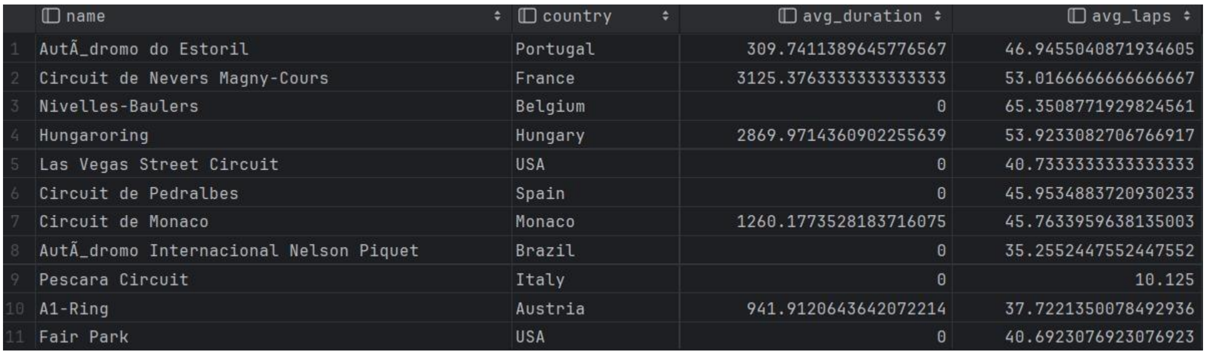
```
SELECT l.country, d.surname, d.forename, SUM(rr.points) AS total_points
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
JOIN location l ON rr.locationId = l.circuitId
GROUP BY l.country, d.surname, d.forename
ORDER BY total_points DESC
LIMIT 1;
```



	country	surname	forename	total_points
1	Italy	Schumacher	Michael	196

→Average finish time and number of laps of each race:

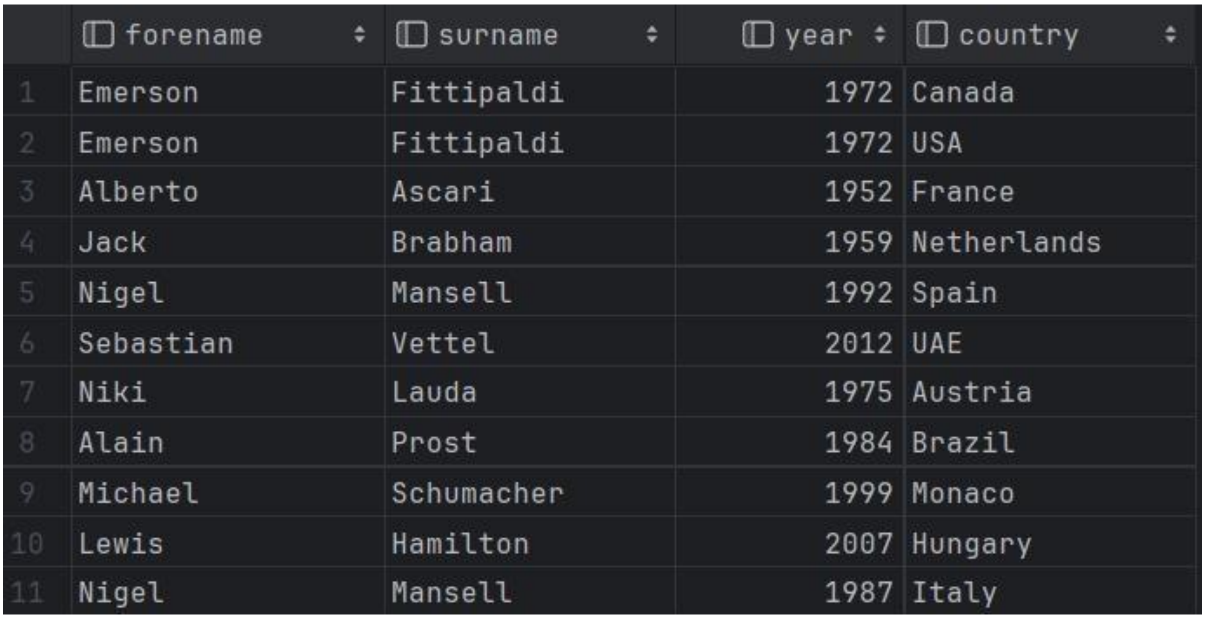
```sql
SELECT l.name, l.country, AVG(rr.duration) AS avg_duration, AVG(rr.laps) AS avg_laps
FROM race_results rr
JOIN location l ON rr.locationId = l.circuitId
GROUP BY l.name, l.country;
```



	name	country	avg_duration	avg_laps
1	AutÃ_dromo do Estoril	Portugal	309.7411389645776567	46.9455040871934605
2	Circuit de Nevers Magny-Cours	France	3125.3763333333333333	53.0166666666666667
3	Nivelles-Baulers	Belgium	0	65.3508771929824561
4	Hungaroring	Hungary	2869.9714360902255639	53.9233082706766917
5	Las Vegas Street Circuit	USA	0	40.7333333333333333
6	Circuit de Pedralbes	Spain	0	45.9534883720930233
7	Circuit de Monaco	Monaco	1260.1773528183716075	45.7633959638135003
8	AutÃ_dromo Internacional Nelson Piquet	Brazil	0	35.2552447552447552
9	Pescara Circuit	Italy	0	10.125
10	A1-Ring	Austria	941.9120643642072214	37.7221350078492936
11	Fair Park	USA	0	40.6923076923076923

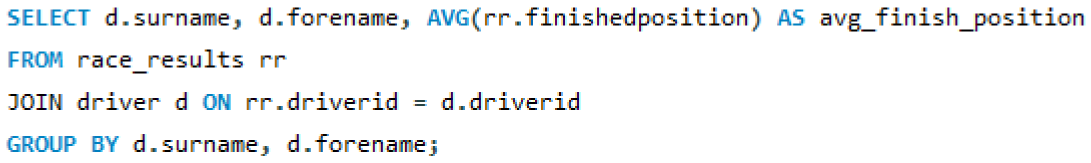
→The winning driver of every country in a year:

```sql
SELECT d.forename,d.surname, t.year, l.country
FROM race_results rr
JOIN location l ON rr.locationId = l.circuitId
JOIN driver d ON rr.driverId = d.driverId
JOIN time t ON rr.timeId = t.timeId
WHERE rr.finishPosition = 1
GROUP BY t.year, l.country, d.forename, d.surname;
```

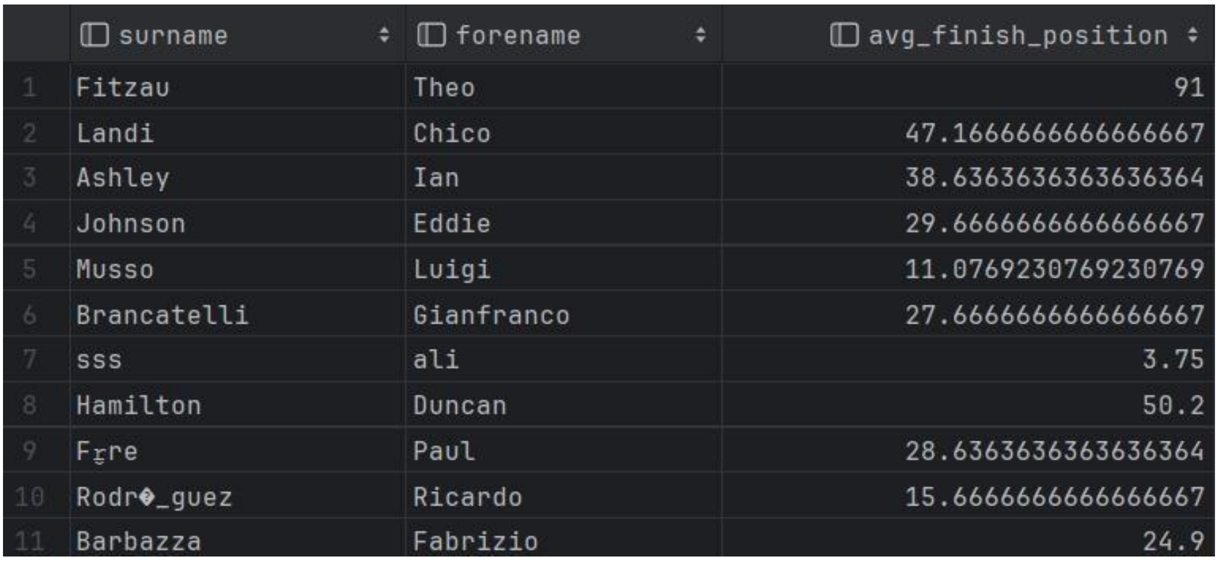


	forename	surname	year	country
1	Emerson	Fittipaldi	1972	Canada
2	Emerson	Fittipaldi	1972	USA
3	Alberto	Ascari	1952	France
4	Jack	Brabham	1959	Netherlands
5	Nigel	Mansell	1992	Spain
6	Sebastian	Vettel	2012	UAE
7	Niki	Lauda	1975	Austria
8	Alain	Prost	1984	Brazil
9	Michael	Schumacher	1999	Monaco
10	Lewis	Hamilton	2007	Hungary
11	Nigel	Mansell	1987	Italy

→The average finishing position of a given driver in the races in which he/she participated:

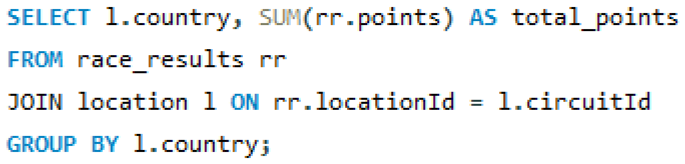


```
SELECT d.surname, d.forename, AVG(rr.finishedposition) AS avg_finish_position
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
GROUP BY d.surname, d.forename;
```

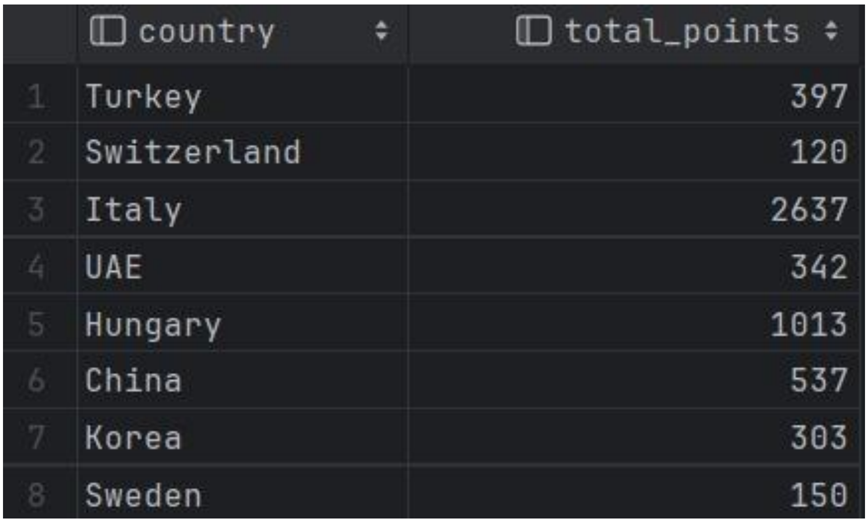


	surname	forename	avg_finish_position
1	Fitzau	Theo	91
2	Landi	Chico	47.1666666666666667
3	Ashley	Ian	38.6363636363636364
4	Johnson	Eddie	29.6666666666666667
5	Musso	Luigi	11.0769230769230769
6	Brancatelli	Gianfranco	27.6666666666666667
7	sss	ali	3.75
8	Hamilton	Duncan	50.2
9	Fr̯re	Paul	28.6363636363636364
10	Rodr�_guez	Ricardo	15.6666666666666667
11	Barbazza	Fabrizio	24.9

→The total points scored in all races in a given country:



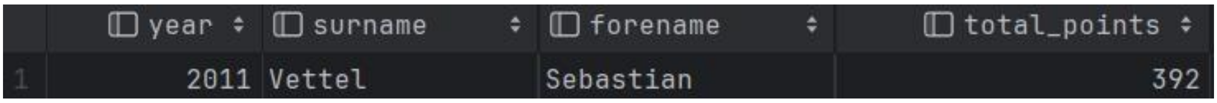
```
SELECT l.country, SUM(rr.points) AS total_points
FROM race_results rr
JOIN location l ON rr.locationId = l.circuitId
GROUP BY l.country;
```



	country	total_points
1	Turkey	397
2	Switzerland	120
3	Italy	2637
4	UAE	342
5	Hungary	1013
6	China	537
7	Korea	303
8	Sweden	150

→The driver with the most points in a given year:

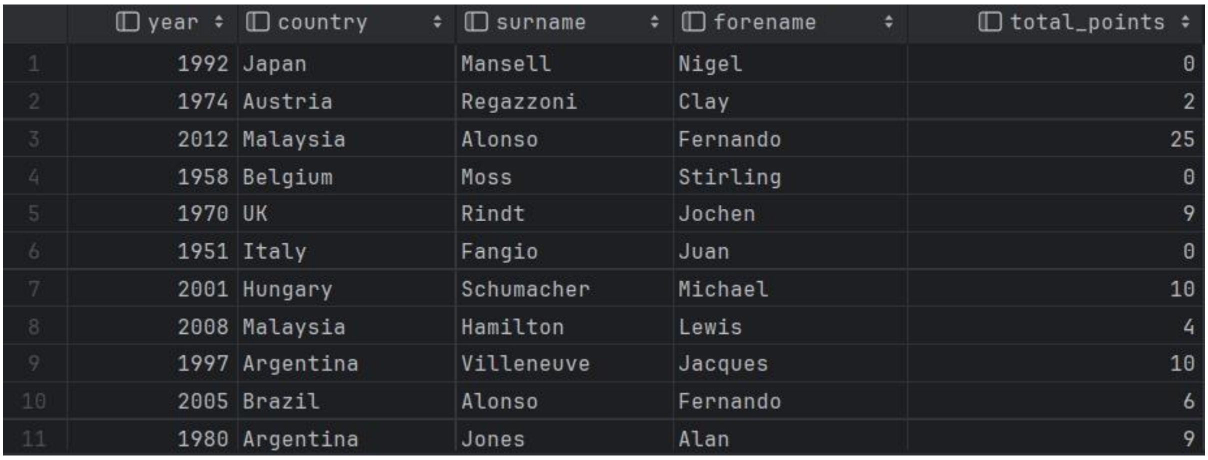
```
SELECT t.year, d.surname, d.forename, SUM(rr.points) AS total_points
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
JOIN time t ON rr.timeId = t.timeId
GROUP BY t.year, d.surname, d.forename
ORDER BY total_points DESC
LIMIT 1;
```



	year	surname	forename	total_points
1	2011	Vettel	Sebastian	392

→The winner of the races in each country in a given year and the points awarded:

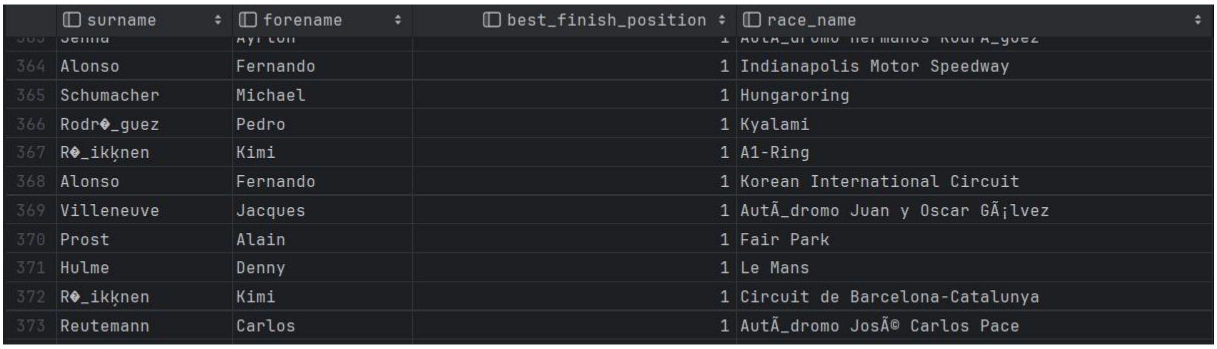
```
SELECT t.year, l.country, d.surname, d.forename, SUM(rr.points) AS total_points
FROM race_results rr
JOIN location l ON rr.locationId = l.circuitId
JOIN driver d ON rr.driverid = d.driverid
JOIN time t ON rr.timeId = t.timeId
WHERE rr.finishedposition = 1
GROUP BY t.year, l.country, d.surname, d.forename;
```



	year	country	surname	forename	total_points
1	1992	Japan	Mansell	Nigel	0
2	1974	Austria	Regazzoni	Clay	2
3	2012	Malaysia	Alonso	Fernando	25
4	1958	Belgium	Moss	Stirling	0
5	1970	UK	Rindt	Jochen	9
6	1951	Italy	Fangio	Juan	0
7	2001	Hungary	Schumacher	Michael	10
8	2008	Malaysia	Hamilton	Lewis	4
9	1997	Argentina	Villeneuve	Jacques	10
10	2005	Brazil	Alonso	Fernando	6
11	1980	Argentina	Jones	Alan	9

→The best career finishing position of a particular driver and the race in which it was achieved:

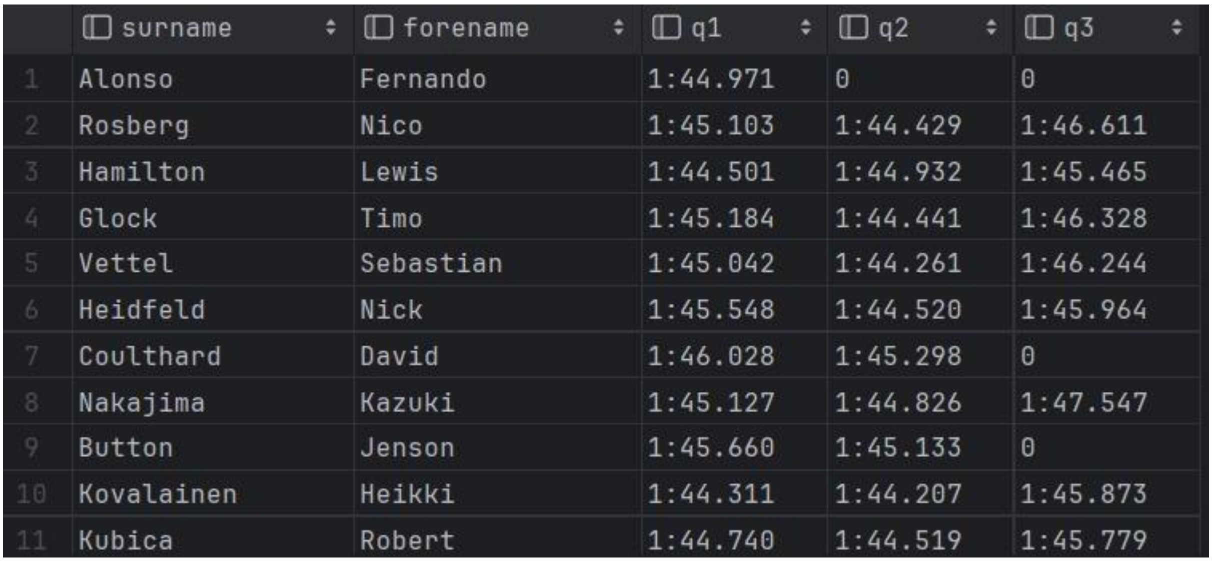
```sql
SELECT d.surname, d.forename, MIN(rr.finishedposition) AS best_finish_position, l.name AS race_name
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
JOIN location l ON rr.locationId = l.circuitId
GROUP BY d.surname, d.forename, l.name
ORDER BY best_finish_position;
```



	surname	forename	best_finish_position	race_name
363	Senna	Ayrton	1	AutÃ_dromo Hermanos RodrÃ_guez
364	Alonso	Fernando	1	Indianapolis Motor Speedway
365	Schumacher	Michael	1	Hungaroring
366	Rodr◆_guez	Pedro	1	Kyalami
367	R◆_ikķnen	Kimi	1	A1-Ring
368	Alonso	Fernando	1	Korean International Circuit
369	Villeneuve	Jacques	1	AutÃ_dromo Juan y Oscar GÃ¡lvez
370	Prost	Alain	1	Fair Park
371	Hulme	Denny	1	Le Mans
372	R◆_ikķnen	Kimi	1	Circuit de Barcelona-Catalunya
373	Reutemann	Carlos	1	AutÃ_dromo JosÃ© Carlos Pace

→The best times of drivers in qualifying for a particular race: (e.g. for 15)

```sql
SELECT d.surname, d.forename, q.q1, q.q2, q.q3
FROM race_results rr
JOIN driver d ON rr.driverid = d.driverid
JOIN qualifying q ON rr.qualifyingId = q.qualifyingId
WHERE rr.locationId = 15;
```



	surname	forename	q1	q2	q3
1	Alonso	Fernando	1:44.971	0	0
2	Rosberg	Nico	1:45.103	1:44.429	1:46.611
3	Hamilton	Lewis	1:44.501	1:44.932	1:45.465
4	Glock	Timo	1:45.184	1:44.441	1:46.328
5	Vettel	Sebastian	1:45.042	1:44.261	1:46.244
6	Heidfeld	Nick	1:45.548	1:44.520	1:45.964
7	Coulthard	David	1:46.028	1:45.298	0
8	Nakajima	Kazuki	1:45.127	1:44.826	1:47.547
9	Button	Jenson	1:45.660	1:45.133	0
10	Kovalainen	Heikki	1:44.311	1:44.207	1:45.873
11	Kubica	Robert	1:44.740	1:44.519	1:45.779

This data model can be used to analyze various aspects of Formula 1 racing. For example, the answers to questions such as which driver or team have won the most races, which track is the most challenging, which driver has the best qualifying lap time, which driver has the fastest pit stops, which driver has the most consistent lap times, etc. can be obtained from this data model. This data model can also be used to create a detailed article about the history, statistics, and performances of Formula 1 races.

6. References

[1] Reddit. (2022). r/formula1 2022 Census Results