F1 DATA WEBSITE PROJECT

EFE ERTEM, 71739
DORUK ÖZER, 70192
SARP GÖL, 72368
YİĞİT YAKAR, 72269
DUHA EMIR GANİOĞLU, 71753

Project Description:

We created an F1 database website for the Formula 1 fans who would like to learn some specific information regarding the status of constructors, drivers and circuits. This website can also be used in F1 predictions and general analysis of constructors and pilots.

Relational Database Design:

```
CREATE TABLE CIRCUITS (circuit_id VARCHAR(50) NOT NULL,
                    circuit_ref VARCHAR(50),
                    name VARCHAR(50),
                    location VARCHAR(50),
                    country VARCHAR(50),
                    lat float,
                    lng float,
                    alt VARCHAR(50),
                    PRIMARY KEY (circuit_id)
                    )

CREATE TABLE LAPS (race_id VARCHAR(50) NOT NULL,
                driver_id VARCHAR(50) NOT NULL,
                lap VARCHAR(50) NOT NULL,
                position VARCHAR(50),
                time VARCHAR(50),
                milliseconds VARCHAR(50),
                PRIMARY KEY (race_id,driver_id,lap),
                FOREIGN KEY (race_id)
                REFERENCES RACES(race_id),
                FOREIGN KEY (driver_id)
                REFERENCES DRIVERS(driver_id)
            )

CREATE TABLE PIT_STOPS (race_id VARCHAR(50) NOT NULL,
            driver_id VARCHAR(50),
            stop VARCHAR(50),
            lap VARCHAR(50),
            time VARCHAR(50),
            duration VARCHAR(50),
            milliseconds VARCHAR(50),
            PRIMARY KEY (race_id,driver_id,stop),
            FOREIGN KEY (race_id)
            REFERENCES RACES(race_id),
```

```
            FOREIGN KEY (driver_id)
            REFERENCES DRIVERS(driver_id)
            )

CREATE TABLE CONSTRUCTOR_RESULTS (constructor_results_id VARCHAR(50)
NOT NULL,
                race_id VARCHAR(50),
                constructor_id VARCHAR(50),
                points VARCHAR(50),
                status VARCHAR(50),
                PRIMARY KEY (race_id,constructor_id),
                FOREIGN KEY (race_id)
                REFERENCES RACES(race_id),
                FOREIGN KEY (constructor_id)
                REFERENCES CONSTRUCTORS(constructor_id)
                )

CREATE TABLE CONSTRUCTOR_STANDINGS (constructor_standings_id
VARCHAR(50) NOT NULL,
                    race_id VARCHAR(50),
                    constructor_id VARCHAR(50),
                    points VARCHAR(50),
                    position VARCHAR(50),
                    positionText VARCHAR(50),
                    wins VARCHAR(50),
                    PRIMARY KEY (race_id,constructor_id),
                    FOREIGN KEY (race_id)
                    REFERENCES RACES(race_id),
                    FOREIGN KEY (constructor_id)
                     REFERENCES CONSTRUCTORS(constructor_id)
                    )

CREATE TABLE DRIVER_STANDINGS (driver_standings_id VARCHAR(50) NOT
NULL,
                race_id VARCHAR(50),
                driver_id VARCHAR(50),
                points VARCHAR(50),
                position VARCHAR(50),
                positionText VARCHAR(50),
                wins VARCHAR(50),
                PRIMARY KEY (race_id,driver_id),
                FOREIGN KEY (race_id) REFERENCES RACES(race_id),
                FOREIGN KEY (driver_id) REFERENCES DRIVERS(driver_id)
                )

CREATE TABLE DRIVERS (driver_id VARCHAR(50) NOT NULL,
                driver_ref VARCHAR(50),
                number VARCHAR(50),
                code VARCHAR(50),
                firstname VARCHAR(50),
                surname VARCHAR(50),
```

```
                dob VARCHAR(50),
                nationality VARCHAR(50),
                PRIMARY KEY (driver_id)
            )

CREATE TABLE CONSTRUCTORS (constructor_id VARCHAR(50) NOT NULL,
                    constructor_ref VARCHAR(50),
                    name VARCHAR(50),
                    nationality VARCHAR(50),
                    PRIMARY KEY (constructor_id)
        )

CREATE TABLE QUALIFYING (qualify_id VARCHAR(50) NOT NULL,
                    race_id VARCHAR(50),
                    driver_id VARCHAR(50),
                    constructor_id VARCHAR(50),
                    number VARCHAR(50),
                    position VARCHAR(50),
                    q1 VARCHAR(50),
                    q2 VARCHAR(50),
                    q3 VARCHAR(50),
                    PRIMARY KEY (race_id,driver_id,constructor_id),
                    FOREIGN KEY (race_id)
                    REFERENCES RACES(race_id),
                    FOREIGN KEY (driver_id)
                    REFERENCES DRIVERS(driver_id),
                    FOREIGN KEY (constructor_id)
                    REFERENCES CONSTRUCTORS(constructor_id)
                )

CREATE TABLE RACES (race_id VARCHAR(50) NOT NULL,
                year VARCHAR(50),
                round VARCHAR(50),
                circuit_id VARCHAR(50),
                name VARCHAR(50),
                date VARCHAR(50),
                time VARCHAR(50),
                PRIMARY KEY (race_id)
                )

CREATE TABLE STATUS (status_id VARCHAR(50) NOT NULL,
                    status VARCHAR(50))

CREATE TABLE RESULTS (result_id VARCHAR(50) NOT NULL,
                    race_id VARCHAR(50),
                    driver_id VARCHAR(50),
                    constructor_id VARCHAR(50),
                    number VARCHAR(50),
                    grid VARCHAR(50),
                    position VARCHAR(50),
                    positionText VARCHAR(50),
```

```
                    positionOrder VARCHAR(50),
                    points VARCHAR(50),
                    laps VARCHAR(50),
                    time VARCHAR(50),
                    milliseconds VARCHAR(50),
                    fastest_lap VARCHAR(50),
                    rank_ VARCHAR(50),
                    fastest_lap_time VARCHAR(50),
                    fastest_lap_speed VARCHAR(50),
                    status_id VARCHAR(50),
                    PRIMARY KEY
(result_id,race_id,driver_id,constructor_id),
                    FOREIGN KEY (race_id)
                    REFERENCES RACES(race_id),
                    FOREIGN KEY (driver_id)
                    REFERENCES DRIVERS(driver_id),
                    FOREIGN KEY (constructor_id)
                    REFERENCES CONSTRUCTORS(constructor_id)
                    )

CREATE TABLE
S AS(
        select  *
        from STATUS NATURAL JOIN T1
)
```

Data Sources:

We got the data as csv from Keggle.com. The link was https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020
We created functions to populate tables in python as we learned in the homeworks. The functions open the file, read the file, take each row as inputs and split them by the comma, and finally insert the data into the database.

You can see the functions to populate tables below;

```
def populate_circuits_table(db_connection, db_cursor, insert_query,
file_path):
   with open(file_path, mode='r') as csv_data:

       reader = csv.reader(csv_data, delimiter=';')
       csv_data_list = list(reader)
       for row in csv_data_list[1:]:
           row = tuple(map(lambda x: None if x == "" else x,
row[0].split(',')))

           db_cursor.execute(insert_query, row[0:8])

   db_connection.commit()
```

```
def populate_constructors_table(db_connection, db_cursor, insert_query,
file_path):
   with open(file_path, mode='r') as csv_data:

       reader = csv.reader(csv_data, delimiter=';')
       csv_data_list = list(reader)
       for row in csv_data_list[1:]:
           row = tuple(map(lambda x: None if x == "" else x,
row[0].split(',')))

           db_cursor.execute(insert_query, row[0:4])

   db_connection.commit()
def populate_table(db_connection, db_cursor, insert_query, file_path):
   count=0
   with open(file_path, mode='r') as csv_data:
       reader = csv.reader(csv_data, delimiter=';')
       csv_data_list = list(reader)
       for row in csv_data_list[1:]:
           row = tuple(map(lambda x: None if x == "" else x,
row[0].split(',')))
           db_cursor.execute(insert_query, row)

   db_connection.commit()
def populate_races_table(db_connection, db_cursor, insert_query,
file_path):
   with open(file_path, mode='r') as csv_data:

       reader = csv.reader(csv_data, delimiter=';')
       csv_data_list = list(reader)
       for row in csv_data_list[1:]:
           row = tuple(map(lambda x: None if x == "" else x,
row[0].split(',')))

           db_cursor.execute(insert_query, row[0:7])
```

Complex SQL Queries:

- **First Advanced Query;**

```
select sum(f.points) as 'total points'
from final_results f
where f.driver_id=
               (select drivers.driver_id from drivers where
drivers.firstname like '%$firstname1%' and drivers.surname like
'%$surname1%')
       group by f.driver_id";
```

This query is implemented under the “Driver Comparator” fieldset in the comparator.html file. This query takes the firstname and surname of the driver as inputs, and calculates the total points the given driver has won till now. The Driver Comparator calculates this result for two of the Formula 1 pilots and lists them.

- **Second Advanced Query;**

```
select drivers.firstname as 'name' , drivers.surname as 'surname',
drivers.code as 'code'
from final_results,drivers
where final_results.driver_id = drivers.driver_id
group by final_results.driver_id
having count(*) > $val;
```

This query is implemented under the “Result Count” fieldset in the comparator.html file. The user should enter a number as the input for this query to work. The given number indicates the minimum number of a drivers’ result count to be in the final result. So essentially this query shows the drivers who have attended more races than the entry from the user.

- **Third Advanced Query;**

```
select count(*) as 'Podium' ,firstname,surname
from final_results,drivers
where drivers.driver_id = final_results.driver_id and position < 4 and
position > 0 and grid > $grid
group by drivers.driver_id
order by count(*) desc
```

This query is implemented under the “Bottom Podium” fieldset in the comparator.html file. This query finds the podiums won without starting in the top three grid

positions. The input from the user signifies the minimum grid position to have to be in the podium. The short definition of this query can be called “Podiums without a top X start”.

- **Fourth Advanced Query;**

```
select count(*) as 'Count',Firstname,Surname
   from races,circuits,final_results,drivers
   where drivers.driver_id = final_results.driver_id and
races.circuit_id = circuits.circuit_id and final_results.race_id =
races.race_id and lat $lat 0 and lng $lng 0 and position = 1
   group by drivers.driver_id
   order by count(*) desc;
```

This query is implemented under the “” fieldset in the comparator.html file. The inputs taken by the user are either ‘>’ and ‘>’ or ‘<’and ‘<’. If the inputs are ‘>’ that means the latitude and longitude are greater than 0 which basically means the North Hemisphere and it is the opposite for the South Hemisphere. So this query returns the number of times a driver has won in the given Hemisphere.

- **Fifth Advanced Query:**

```
     select count(*) as 'Wins', drivers.Nationality
        from drivers,final_results
        where drivers.driver_id = final_results.driver_id and position
< 2 and position > 0
        group by nationality
        order by count(*) desc;
```

This query is implemented under the “Nations” fieldset in the comparator.html file. This query does not take inputs, and lists how many wins a nation has in the history of Formula 1.

- **Sixth Advanced Query;**

```
select driv.firstname as fname, driv.surname as sname, sum(f.points) as
'total points'
from final_results f, drivers driv
where f.driver_id in (select driver.driver_id
        from final_results as res
        Join (   select distinct race.race_id
           from  drivers mydriver, races race, final_results res
            where mydriver.firstname like '%$fname%' and
mydriver.surname like '%$sname%' and
   mydriver.driver_id= res.driver_id and
   res.race_id in (select race2.race_id
                    from final_results race2
                    where race2.driver_id = mydriver.driver_id)) raced
on raced.race_id = res.race_id , drivers driver, drivers minedriver
   where driver.driver_id = res.driver_id and minedriver.firstname like
'%$fname%' and minedriver.surname like '%$sname%' and driver.driver_id
!= minedriver.driver_id and driv.driver_id = res.driver_id
   )
   group by f.driver_id
   order by sum(f.points) desc;
```

This query is implemented under the “Raced Against” fieldset in the comparator.html file. This query takes one driver's fullname and lists all of the drivers that he has raced with in his career.

- **Seventh Advanced Query;**

```
select  distinct cons2.name as consname,result2.position,race2.name,
race2.year
from  races as race2
Join (select race.year , race.race_id
    from constructor_standings result, races race, constructors cons,
circuits cir
    where cons.constructor_id = result.constructor_id and
    race.race_id= result.race_id and
    race.circuit_id = cir.circuit_id and
     cir.name= \"\"\"".$name."\"\"\"
    group by race.year
    order by race.year desc
```

```
    limit 3) YY  on YY.race_id= race2.race_id,constructor_standings
result2, constructors cons2, circuits cir2
    where
    cons2.constructor_id = result2.constructor_id and
    race2.race_id= result2.race_id and
    race2.circuit_id = cir2.circuit_id and
     result2.position in (1,2,3)
    order by race2.year,result2.position asc
              ;
```

These queries are integrated at the “Current Constructors” drop-box at index.html. The “Current Constructors” drop-box includes the most relevant circuits. When the user clicks any of the circuits in that drop-box, the query finds the last 3 races that are held in that circuit and gives the standing of the top 3 constructors in those 3 races. Output table includes the constructor's name, constructor's standing, name of the circuit and the year. This query is included in the project, because it gives the top 3 standing of the latest 3 races held on that circuit, so the user can infer which drivers are comparably better than the racers in that specific circuit or which driver can most likely to race better based on the last 3 races held in that circuit. In addition, many F1 fans are interested in the top 3 standings in any circuit. This query flexiblelly calculates and shows the results.

- **Other SQL queries:**

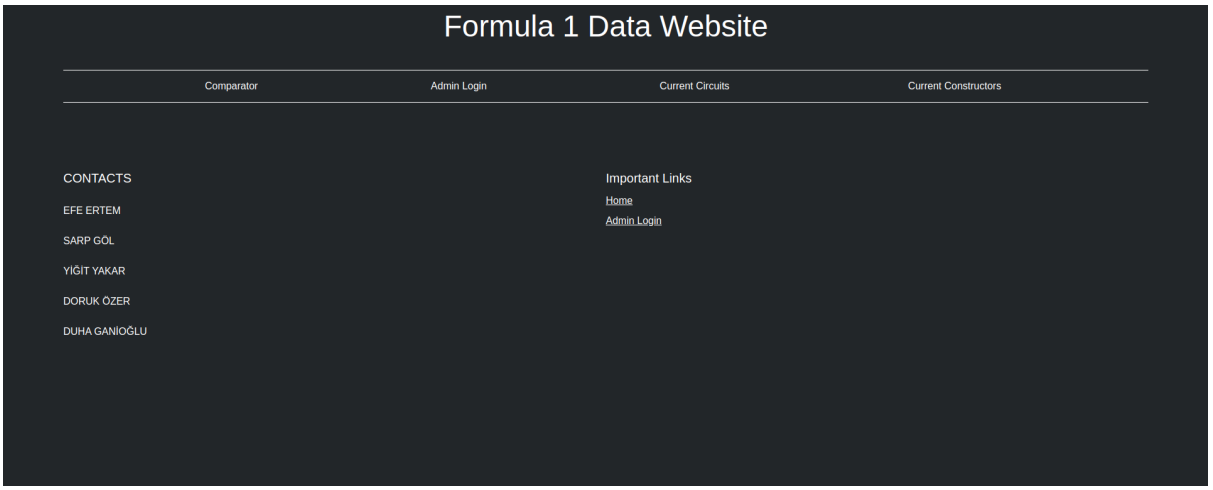
```
                                Query 1
"select  distinct  drive.firstname, drive.surname
    from constructors cons, drivers drive, results res
    where res.driver_id = drive.driver_id AND res.constructor_id =
cons.constructor_id and cons.name= \"\"\".$name."\"\"\";



                                Query 2
 "select  distinct  drive.firstname, drive.surname
    from constructors cons, drivers drive, results res, races race
    where res.driver_id = drive.driver_id AND  race.race_id=
res.race_id and race.year=2022 and res.constructor_id =
cons.constructor_id and cons.name= \"\"\".$name."\"\"\";

                Query to insert a new “constructor”
      "INSERT INTO constructors(constructor_id, constructor_ref, name,
nationality) VALUES('$i_id', '$ref', '$name', '$nat');";
```

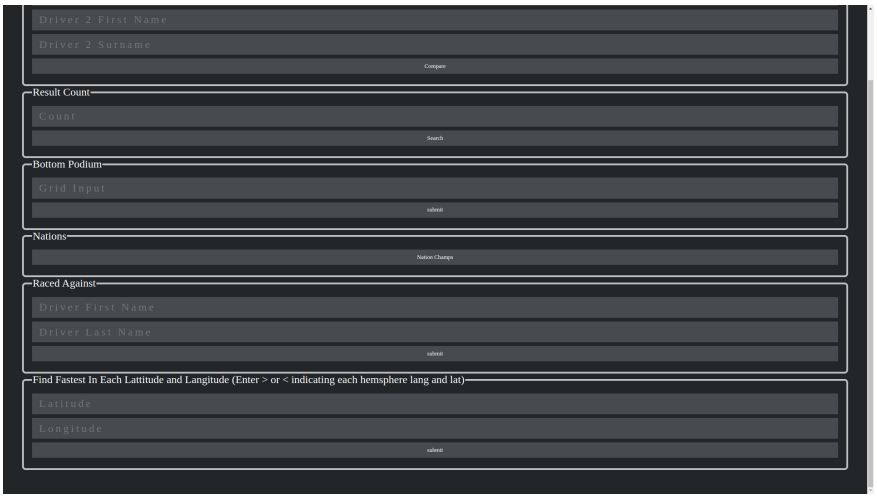
```
Query to delete a constructor
"DELETE FROM constructors c WHERE c.constructor_id='$d_id'";
```

These queries are integrated at the “Current Constructors” drop-box at index.html. The constructors are listed In the “Current Constructors” drop box, and the user selects any of the constructors. Then the first query finds all of the drivers that competed for that constructor. Second constructor finds the drivers who are the current drivers of that constructor. Then the function print_table_const compares the results of those two queries and finds the currently active drivers and marks them as “Active”. In the output table name and surname of all of the drivers that have raced for that constructor are included with the extra column that states if the driver is still competing for that constructor or not. So that the user of the website can monitor the current drivers that are racing for a specific constructor with all of the drivers who have raced for that constructor.

Screenshots:

The first screenshots of the site is down below. And we will have a short video demonstration in the submitted file.





Entity Relationship Diagram