Databases Final Project

601.315

Christine Liu cliu168@jhu.edu
Stephen Zhang szhan141@jh.edu

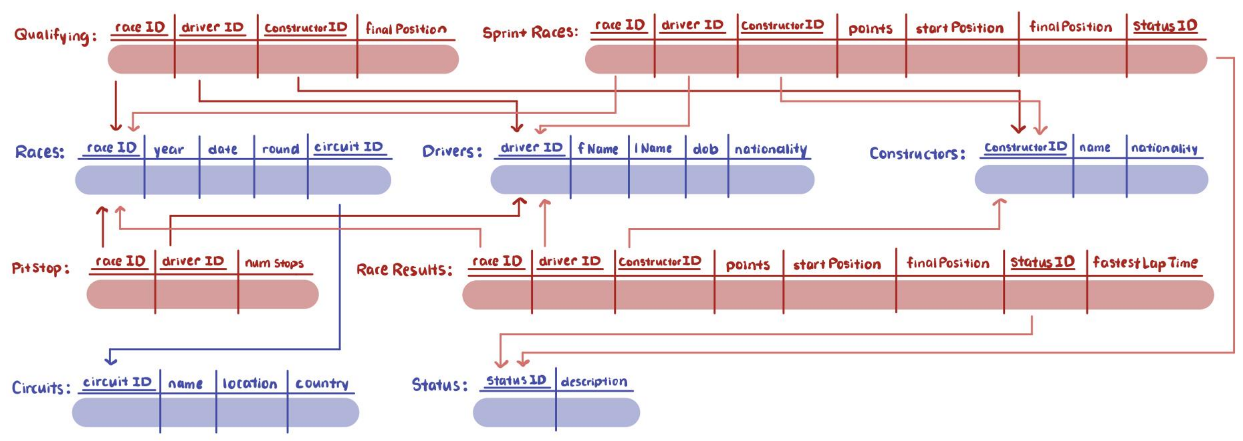
A Formula One database containing drivers, constructors (teams), circuits, races, and various race results. We generate results along the lines of showing the averages for different attributes and the total amount of an attribute.

Website https://www.ugrad.cs.jhu.edu/~szhan141/
Code cliu168/Formula-One-Database: Christine Liu, Stephen Zhang (github.com)

Sample Queries

1. Return nationalities and driver count. Order by descending number of drivers.
2. Return the first name and last name of all drivers of a specific nationality.
3. Return countries and circuit count. Order by descending number of circuits.
4. Return the name and location of all circuits in a specific country.
5. Return the first name and last name of all drivers who have won a race.
6. Return the first name and last name of all drivers who have gotten pole from qualifying.
7. Return the first name and last name of all drivers who have won a sprint race.
8. Return the <u>average number points for a specific circuit</u> of each <u>driver</u> and the respective driver's first name and last name. Order by descending number of points.
9. Return the <u>average number of points per season</u> of each <u>driver</u> and the respective driver's first name and last name. Order by descending number of points.
10. Return the <u>average number points for a specific circuit</u> of each <u>constructor</u> and the respective constructor's name. Order by descending number of points.
11. Return the <u>average number of points per season</u> of each <u>constructor</u> and the respective constructor's name. Order by descending number of points.
12. Return driver first name and last names, and their respective number of wins in a specific circuit. Order by descending number of wins.
13. Return driver first name and last names, and their respective fastest lap time in a specific circuit. Order by ascending all time fastest lap time.
14. Return the name and location of all circuits and it's all time <u>total race accidents/collisions</u>. Order by descending accidents.
15. Return the name and location of all circuits and its <u>average accidents/collisions per race</u>. Order by descending average accidents.
16. Return first name and last name of all drivers who have ever had an accident/collision in a specific circuit.
17. Return the first name and last name of all drivers, and their all time number of races with issues (status not equal to one). Order by descending number of races with issues.
18. Return average number of pit stops of all drivers that have won at a specific circuit. Order by ascending pit stops.
19. Return nationalities and all its drivers' average number of points per season. Order by descending points.
20. Return year of birth and all its drivers' average number of points per season. Order by descending points.

Relational Model


Qualifying: raceID | driverID | ConstructorID | final Position
Sprint Races: raceID | driverID | ConstructorID | points | start Position | final Position | Status ID
Races: raceID | year | date | round | circuit ID
Drivers: driver ID | fName | lName | dob | nationality
Constructors: ConstructorID | name | nationality
PitStop: race ID | driver ID | num Stops
Race Results: race ID | driver ID | ConstructorID | points | start Position | final Position | Status ID | fastest Lap Time
Circuits: circuit ID | name | location | country
Status: Status ID | description

SQL Implementation

```
create table Circuits (
        circuitID               INTEGER NOT NULL, -- 1
        name                    VARCHAR(100), -- Yarowsky Circuit
        location                VARCHAR(100), -- Maryland
        country                 VARCHAR(100), -- USA
        PRIMARY KEY (circuitID)
);


create table Constructors (
        constructorID           INTEGER NOT NULL, -- 1
        name                    VARCHAR(100), -- Yarowsky Team
        nationality             VARCHAR(100), -- American
        PRIMARY KEY (constructorID)
);

create table Drivers (
        driverID                INTEGER NOT NULL, -- 1
        fName                   VARCHAR(100), -- David
        lName                   VARCHAR(100), -- Yarowsky
        dob                     DATE, -- 1982-10-01
        nationality             VARCHAR(100), -- American
        PRIMARY KEY (driverID)
);

create table Status (
        statusID                INTEGER NOT NULL, -- 1
        description             VARCHAR(100), -- Finished
        PRIMARY KEY (statusID)
);

create table Races (
        raceID                  INTEGER NOT NULL, -- 1
        year                    INTEGER, -- 2022
        round                   INTEGER, -- 1
        circuitID               INTEGER, -- 1
        date                    DATE, -- 2022-12-17
        PRIMARY KEY (raceID)
);
```

```
create table RaceResults (
        raceID                  INTEGER NOT NULL, -- 1
        driverID                INTEGER NOT NULL, -- 1
        constructorID           INTEGER NOT NULL, -- 1
        startPosition           INTEGER, -- 1
        finalPosition           INTEGER, -- 1
        points                  INTEGER, -- 22
        fastestLapTime          TIME, -- 01:34.2
        statusID                INTEGER NOT NULL, -- 1
        FOREIGN KEY (raceID) REFERENCES Races(raceID),
        FOREIGN KEY (driverID) REFERENCES Drivers(driverID),
        FOREIGN KEY (constructorID) REFERENCES Constructors(constructorID),
        FOREIGN KEY (statusID) REFERENCES Status(statusID)
);

create table Qualifying (
        raceID                  INTEGER NOT NULL, -- 1
        driverID                INTEGER NOT NULL, -- 1
        constructorID           INTEGER NOT NULL, -- 1
        finalPosition           INTEGER, -- 1
        FOREIGN KEY (raceID) REFERENCES Races(raceID),
        FOREIGN KEY (driverID) REFERENCES Drivers(driverID),
        FOREIGN KEY (constructorID) REFERENCES Constructors(constructorID)
);

create table SprintRaces (
        raceID                  INTEGER NOT NULL, -- 1
        driverID                INTEGER NOT NULL, -- 1
        constructorID           INTEGER NOT NULL, -- 1
        startPosition           INTEGER, -- 1
        finalPosition           INTEGER, -- 1
        points                  INTEGER, -- 3
        statusID                INTEGER, -- 1
        FOREIGN KEY (raceID) REFERENCES Races(raceID),
        FOREIGN KEY (driverID) REFERENCES Drivers(driverID),
        FOREIGN KEY (constructorID) REFERENCES Constructors(constructorID)

);

create table PitStops (
        raceID                  INTEGER NOT NULL, -- 1
        driverID                INTEGER NOT NULL, -- 1
        numPitStops  INTEGER, -- 0
        FOREIGN KEY (raceID) REFERENCES Races(raceID),
        FOREIGN KEY (driverID) REFERENCES Drivers(driverID)
);
```

Load Database
Data is extracted from:
https://www.kaggle.com/datasets/thedevastator/formula-one-racing-a-comprehensive-data-analysis.
We modified the .csv files to follow our database implementation.
Then the website https://sqlizer.io/ was used to convert our edited .csv files into sql format to be loaded into the database.

Software
mysql on dbase.cs.jhu.edu

Views
One view we create is that for every pair of races and drivers there will be a row that consists of the driver, circuit, constructor, year and points to help with calculating point averages.
Another view we create is that for every circuit there will be a count of accidents and total races to help with calculating accident averages.
The last view we create is a collection of races with the circuit and driver that won, to help with some queries that want to look at winners at a specific circuit.

User's guide
The user can run our code by visiting: https://www.ugrad.cs.jhu.edu/~szhan141/.
Each query is contained within a box, choose an option in the dropdown menu if it is present, and click the submit button to run. Then wait for results to load.

Specialized Topics
particularly advanced GUI form interface and/or report generation
See the strengths listed below.

Strengths

- The website is neatly organized, indicating what the results of each query will look like.
- Ranked queries include column graphs to visualize the top 10 items to the user.
- To limit user error, a dropdown menu exists for queries that require input. It allows the user to know what options they can search for and removes the possibility of typos.

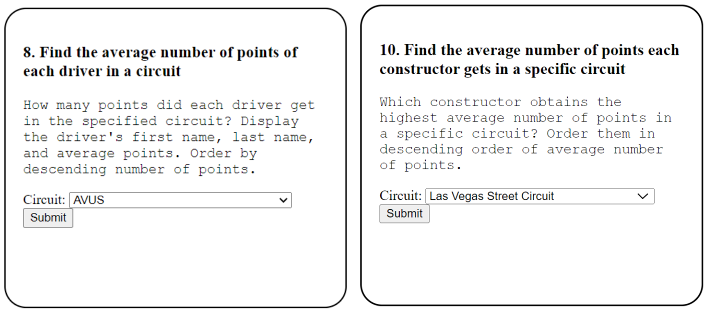
Limitations & Possible Improvements

- When servers are slow, some queries (8, 10, 16, 18) may take a few seconds to load for circuits with many races due to large amount of data to join on. In the unlikely case it takes over a minute, it will be stopped with an internal server error.
- The PitStops table and RaceResults table for the fastestLapTime attribute are missing data for many races, resulting in empty results (13, 18) when it should not be the case. To help fix this issue more data can be found to fill in.
- For our graphs we could not display characters with accent marks. To get around this issue, we replaced them with the same character but without the accent mark. This could be improved by finding a way to display the graphs while maintaining the original characters that were present in the database.

References

We received help from the teaching assistant Jessie Luo on how to display the graphs. From looking at Jessie's code, we were able to explore and learn more on the documentation of the CanvasJS library that was used to create the graphs.

Output



8. Find the average number of points of each driver in a circuit

```
How many points did each driver get
in the specified circuit? Display
the driver's first name, last name,
and average points. Order by
descending number of points.
```

Circuit: AVUS

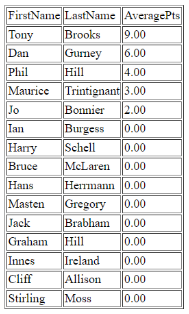
Submit

10. Find the average number of points each constructor gets in a specific circuit

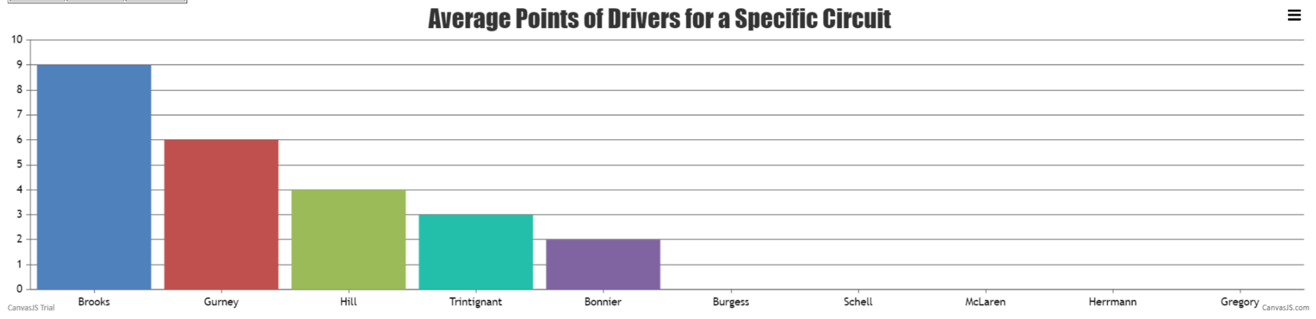
```
Which constructor obtains the
highest average number of points in
a specific circuit? Order them in
descending order of average number
of points.
```

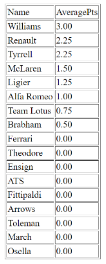
Circuit: Las Vegas Street Circuit

Submit

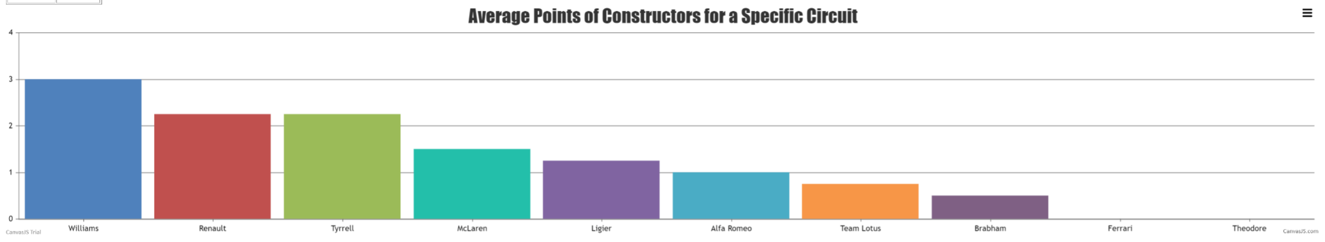


FirstName	LastName	AveragePts
Tony	Brooks	9.00
Dan	Gurney	6.00
Phil	Hill	4.00
Maurice	Trintignant	3.00
Jo	Bonnier	2.00
Ian	Burgess	0.00
Harry	Schell	0.00
Bruce	McLaren	0.00
Hans	Herrmann	0.00
Masten	Gregory	0.00
Jack	Brabham	0.00
Graham	Hill	0.00
Innes	Ireland	0.00
Cliff	Allison	0.00
Stirling	Moss	0.00





Name	AveragePts
Williams	3.00
Renault	2.25
Tyrrell	2.25
McLaren	1.50
Ligier	1.25
Alfa Romeo	1.00
Team Lotus	0.75
Brabham	0.50
Ferrari	0.00
Theodore	0.00
Ensign	0.00
ATS	0.00
Fittipaldi	0.00
Arrows	0.00
Toleman	0.00
March	0.00
Osella	0.00


Average Points of Constructors for a Specific Circuit
Williams
Renault
Tyrrell
McLaren
Ligier
Alfa Romeo
Team Lotus
Brabham
Ferrari
Theodore
CanvasJS Trial
CanvasJS.com