CS 5200 Final Report

Group: BaoYYanK

README

This project is built on Java as the language choice for front-end code. The database-related code is written with MySQL commands.
The IDE I used is IntelliJ IDEA CE, and the newest version (make sure yours is updated) can be downloaded here: https://www.jetbrains.com/idea/download/#section=mac Intellij comes with Gradle and my project uses Gradle to implement dependencies. If you still want to use another IDE, you should make sure Gradle is installed. If not, you can use "pip install gradle" in Terminal to install gradle. But I highly suggest you use Intellij IDEA (preferably on Mac) to run the project because it's the tested way.
Steps you need to run our project on your computer:

1. Unzip BaoYYanK_project.zip file. The Java project code is inside the folder "formula1".
2. Open Intellij IDEA. On the menu bar, click "File > Open" and open the folder "formula1".
3. Change settings in your Intellij IDEA:
    a. Go to Intellij's settings (on Mac, hit "command + ," keys)
    b. Go to "Editor > GUI Designer > General GUI Into: " and select "Java source code"
    c. Go to "Build, Execution, Deployment > Build Tools > Gradle" and change both "Build and run using:" and "Run tests using:" to "Intellij IDEA"
    d. Click "Apply" on the bottom row of the settings panel then click "OK"
4. Open file "formula1_data_dump_2.2.sql" in your MySQL Workbench using your local connection. This is the dump file that contains all the data and procedures for our project. (If you want to check out the source data, they are stored inside the "source data" folder; procedures were originally written inside the "formula1_procedures_2.1.sql" file with comments for each procedure explaining what it does; data cleaning work can be seen inside "formula1_data cleaning_1.0.sql" file; the presentation slides used in our video is "Project Presentation.pdf")
5. Inside the "formula1_data_dump_2.2.sql", click the lightning button without the "I" letter inside to run everything inside this file. It should establish the database in your MySQL Workbench's local connection.
6. Now we are ready to run the Java code. Head to Intellij IDEA. Click on the "src > main > java" folder tree on the left (i.e. the "Project" panel in Intellij). Select "Formula1DataChecker".
7. At the upper right corner, you should already see "Formula1DataChecker" inside the dropdown box to the left of the green triangle "run" button and the green "debug" button. You can also click on the dropdown and "edit configuration". I am using Java 11 SDK.
8. Click the green triangle button to run.
9. Follow the instructions on the window that popped up. In the "Connect to MySQL Database" window, type in your MySQL username and password and hit "Connect".

10. In the “Login/Signup to our formula 1 database” Window, if you are a first-time user, you should input the email address and set a password and hit “Sign up”. Old users can hit “Log In” to log in. If signup or login is not successful, follow the error message displayed right above the buttons. A test account you can use is “test1@gmail.com” and its password is “test1”.
11. In the main window, there are 4 selectors at the top. All thre except “Constructor” selector can be used for “Qualifying”, “Lap Time”, and “Pit Stop” buttons. Try out different combinations (you don’t have to select something for all three). If the given selection has no data, you will see the red message above “results” after clicking on one of these four buttons.
12. For the “race” button, you can also choose to select something for either the “Constructor” selector or the “Driver” selector, but not both.
13. Click on “Favorite Constructors” Button at the bottom to open up the “Favorite Constructors” window.
14. You can select a constructor that’s not yet you favorite in the “Select Constructor to Add” selector and hit the “Add to Favorite Constructors” button. You can see that it is added to your Favorite Constructors.
15. You can also select a favorite constructor and replace it with another constructor not yet your favorite by clicking the “Update” button.
16. You can also select a constructor that’s already your favorite and click “Delete From Favorite Constructors” to remove it.
17. Notice that all selector items in this window are automatically updated as you change your favorite constructors.
18. The “Favorite Drivers” window operates in a similar way.
19. If you are confused about Formula 1 and want to better understand the real-world relationship between different entities in Formula 1, Click on the bottom right button on the Main window.

Technical Specifications

Database language: MySQL
Database application: MySQL Workbench
Front-end language: Java
Front-end application: Java Swing, written in Intellij IDEA Community Edition

Top-Level Description

This project aims at creating a relational database for information related to Formula One racing. Formula One, or more commonly known as F1, is the highest class of international racing for open-wheel single-seater formula racing cars. Since its inaugural season in 1950, it has attracted millions of fans around the world, especially in Europe.

Each Formula One season stretches around the entire calendar year, with a summer break around July and a winter break in January for the teams to make adjustments mid-season and finish up their preparation for the next season, respectively. Each season has around 20 races held at different circuits (i.e., in different cities), with the number changing over the years as new circuits entered and old ones left.

Each race, also named a “Grand Prix”, starts on Friday with Free Practice sessions, giving drivers and teams an opportunity to get familiar with the circuit and test out their car settings. The Qualifying Session happens on Saturday, which determines the order on the starting grid for each driver in Sunday’s actual race session. In the first round, named Q3, all drivers try to record a fastest lap and the bottom five will be dropped out. They will be the last five to start the race on Sunday. Then in the second round, named Q2, all remaining drivers try to set their fastest lap again, with the bottom five being dropped out. They will start from the 11th to the 15th in the race, respectively. At last, the top ten drivers in the Q2 attend the last round, Q1, in which they again try to become the fastest driver by recording a per-lap pace that outperforms others. Their ranking in Q1 is their starting position for Sunday’s race, from pole position to the 10th.

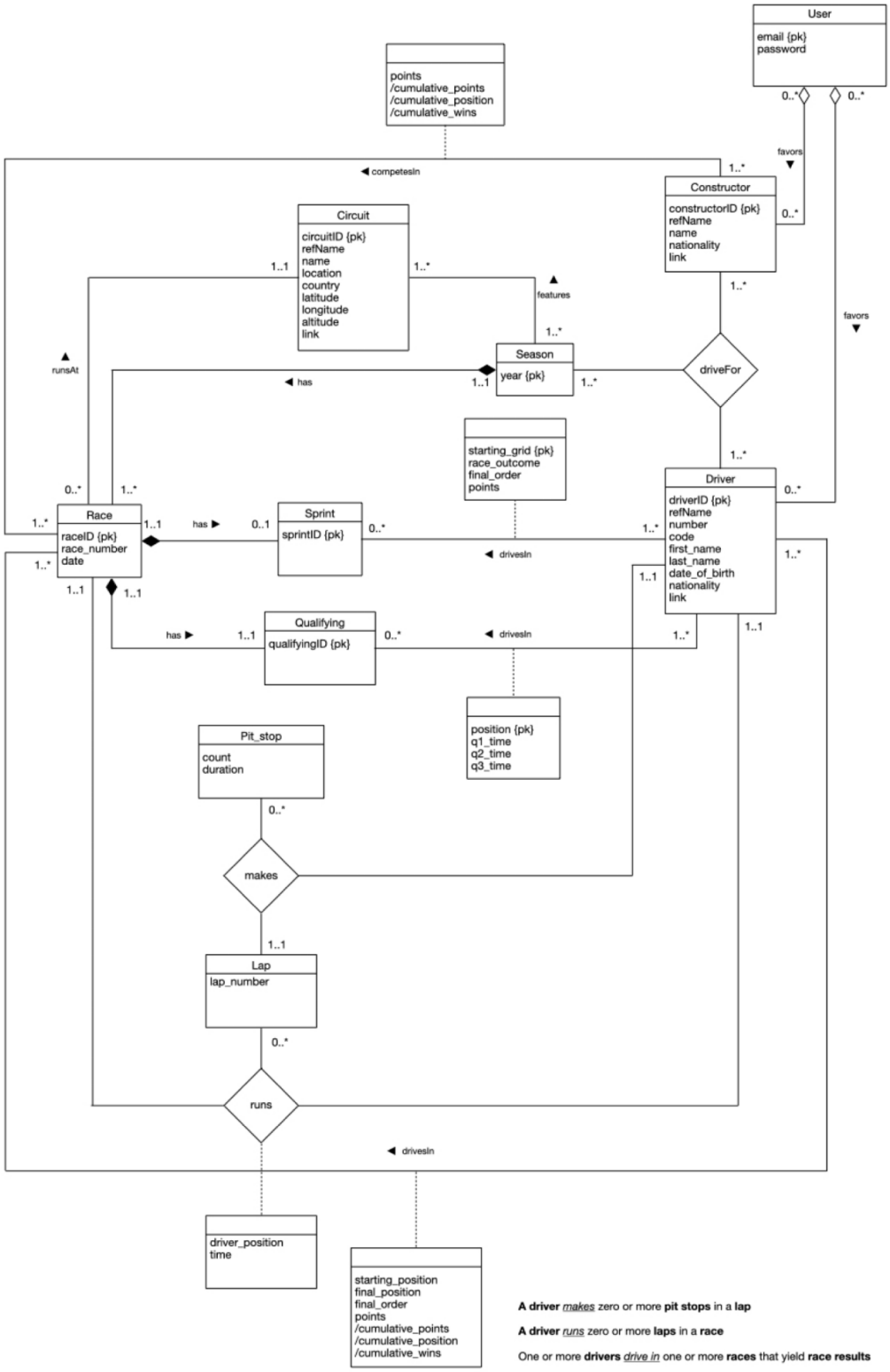
Each race contains a different number of laps depending on the circuit. Each driver in the race must complete these designated number of laps. In these races, each driver’s time for each lap is recorded. Drivers and teams also come up with plans to take pit stops, during which drivers drive into the pits (think “gas station” for cars) to change tires. In the past, teams can also fuel the car during pit stops, but now each car must finish the race without refueling. The pit stops for each driver are also recorded in the database.

Starting in 2021, in some of the races, Formula 1 also introduces a Sprint race on Saturday, which is a quick race of 100 km with no limitation on the number of pit stops. During the race weekends with a Sprint race, the Qualifying session is moved to Friday, and the driver’s rankings in Qualifying determines the starting order of the Sprint race. The driver’s rankings of the Sprint race determines who starts in the front for Sunday’s race.

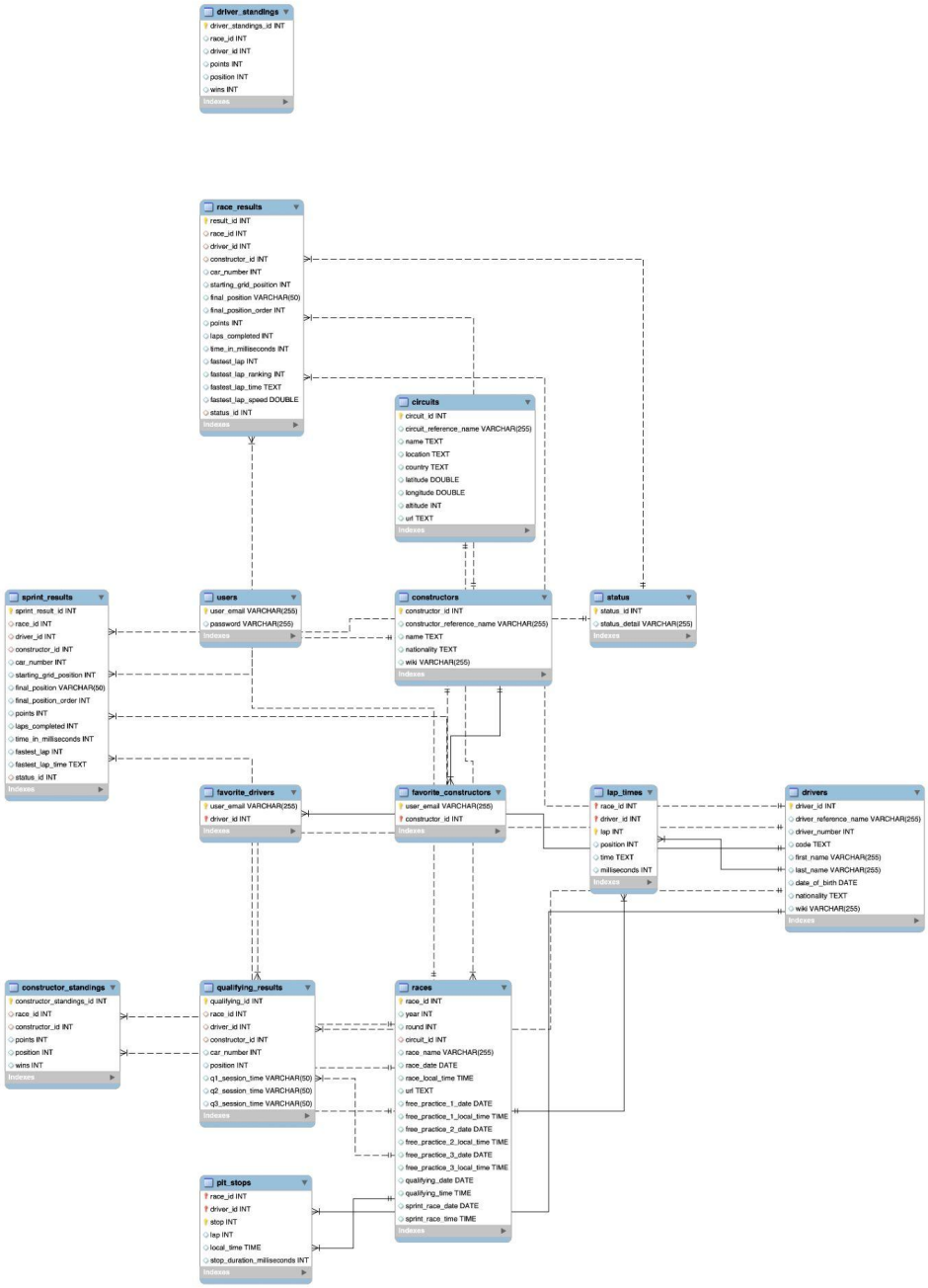
Currently, there are 10 teams (or “constructors” in racing terms) and 20 drivers competing in 22 races per season. In each race, the top drivers receive points, and the constructors’ points are calculated as the sum of their drivers’ points. By the end of each season, the driver with most points earned in the season is crowned the driver’s championship, and the constructor with most points earned by its two drivers becomes the constructor’s championship.

Our dataset contains all the data related to races from the 1950 season to the 2022 season. There is data on all the constructors and drivers that have competed in Formula One. The dataset also includes data on each race, including the qualifying session, each of the laps in the race, and the final result. We also have access to information on the circuits where the races are held, including the name, location, altitude, etc. Finally, there is data on both the driver and the constructor standings in the season after each race.

UML


User
email {pk}
password
points
/cumulative_points
/cumulative_position
/cumulative_wins
0..*
0..*
favors
competesIn
1..*
Constructor
constructorID {pk}
refName
name
nationality
link
0..*
Circuit
circuitID {pk}
refName
name
location
country
latitude
longitude
altitude
link
1..1
1..*
features
1..*
favors
runsAt
Season
year {pk}
has
1..1
1..*
driveFor
1..*
starting_grid {pk}
race_outcome
final_order
points
Driver
driverID {pk}
refName
number
code
first_name
last_name
date_of_birth
nationality
link
0..*
0..*
1..*
Race
raceID {pk}
race_number
date
1..1
has
0..1
Sprint
sprintID {pk}
0..*
drivesIn
1..*
1..*
1..1
1..1
has
1..1
Qualifying
qualifyingID {pk}
0..*
drivesIn
1..*
1..1
position {pk}
q1_time
q2_time
q3_time
Pit_stop
count
duration
0..*
makes
1..1
Lap
lap_number
0..*
runs
drivesIn
driver_position
time
starting_position
final_position
final_order
points
/cumulative_points
/cumulative_position
/cumulative_wins
A driver makes zero or more pit stops in a lap
A driver runs zero or more laps in a race
One or more drivers drive in one or more races that yield race results

Logical Design



Final User Flow

- Login/Signup to our data checker
- Once logged in/signed up, enter the main interface.
- Select any combination of season, race, constructor, and driver (can select nothing; can select one for each option)
    - Hit one of the four buttons: Qualifying, Race, Lap Time, Pit Stop to check the selected data (check specific qualifying session results, race results, lap time results, or pit stop results given the selection of season, race, constructor, and driver; constructor selector only for race button; cannot have both constructor and driver selected for race button)
- Click on the Favorite Driver button to open up the Favorite Drivers window.
    - Select a driver currently not in the favorite driver for this user
        - Click “Add to Favorite Drivers” to add to favorite drivers.
    - Select a driver on the left and a driver on the right
        - Click “Replace Favorite Driver” to replace the one on the left with the one on the right in the favorite drivers table.
    - Select a driver currently in the favorite driver table for this user
        - Click “Delete From Favorite Drivers” to delete from favorite drivers.
- Click on the Favorite Constructor button to open up the Favorite Constructors window.
    - Select a constructor currently not in the favorite constructors for this user
        - Click “Add to Favorite Constructors” to add to favorite constructors.
    - Select a constructor on the left and a constructor on the right
        - Click “Replace Favorite Constructor” to replace the one on the left with the one on the right in the favorite constructors table.
    - Select a constructor currently in the favorite driver table for this user
        - Click “Delete From Favorite Constructors” to delete from favorite constructors.
- Click on the lower right button to know more about Formula 1 in case you are not familiar with the sport.

Lesson Learned

1. In real-world usage of databases, the data is usually not as clean as what we use for homework assignments. A lot of data cleaning must be done before we can further utilize it. Sometimes it will require some expertise on the specific subject of the data. For example, I won't notice the fact that some constructor's should have the same Wikipedia link if I do not watch Formula 1 and recognize that some teams are essentially the same team but with different names and ownerships.
2. It's better to have a plan and keep the work organized if the work quickly grows. For example, because of the flexibility of our user interaction design, I had to write more than 20 procedures. It's much easier to keep track of what has been created since I followed a specific naming convention. Never a mistake to stay organized and not be confused by my own work.
3. Simple user interactions can have complex logic behind it. For example, on the user's side, all he/she does is select a season, a race and hit the "Qualifying" button, but I have to write layers of if statements to determine which MySQL procedure to run. Also when handling user errors in signup and login, it took some work to cover all possible user mistakes and provide relevant error messages. I am also thankful for the idea to use dropdown selectors, so that on the user's end, it's less likely to make a mistake in typing and faster (you can actually type after clicking on the selector to quickly locate an item) to operate. On the developer's end, it's less work when it comes to error handling because all input is controlled ultimately by the developer.
4. Java swing is not the most beautiful GUI in the world, but it works.
5. If given more time, I could possibly explore the possibility of using Python and Flask to implement this project so that it can be opened from a browser.
6. In terms of time management, it's better to start early. It was a bit tight in the end because you never know what bugs will appear in the coding process and how long each takes to be removed.

Future Work

1. This database can be further updated with the newest race results in the 2023 season.
2. If I can migrate the GUI on Python and Flask, I could do data visualization with Python packages like MatPlotLib to show trends such as a single driver's fastest lap time at the same Grand Prix over the years, or a constructor's points by the end of the season over different seasons.
3. GUI is functional at the moment but not beautiful. For favorite drivers and constructors, I can make the row clickable and show a new window that has some summary information. For example, for drivers, show their age, active years, number of wins, number of championships, etc.