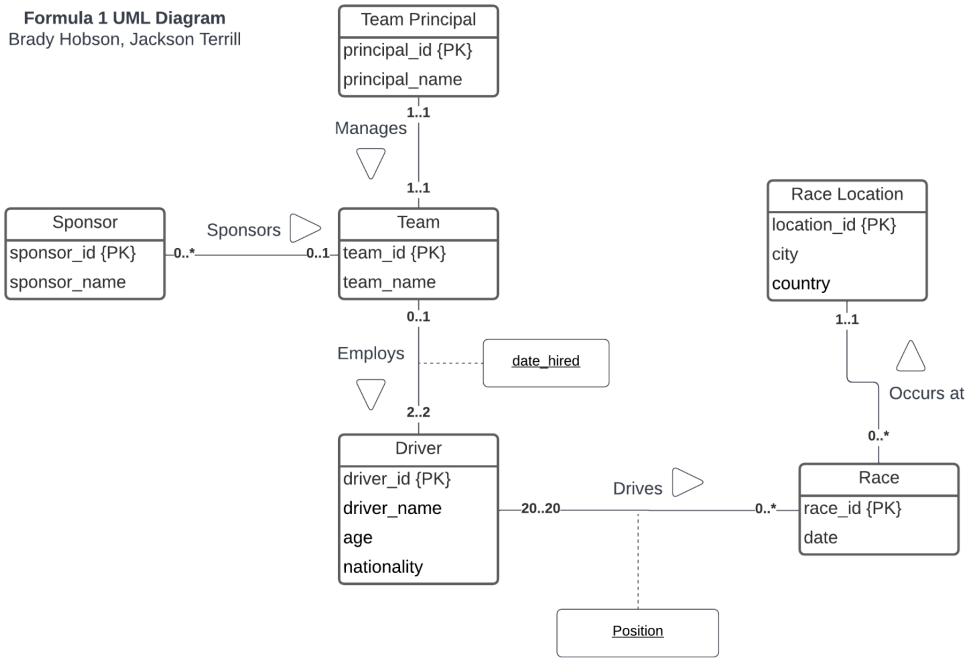
README:

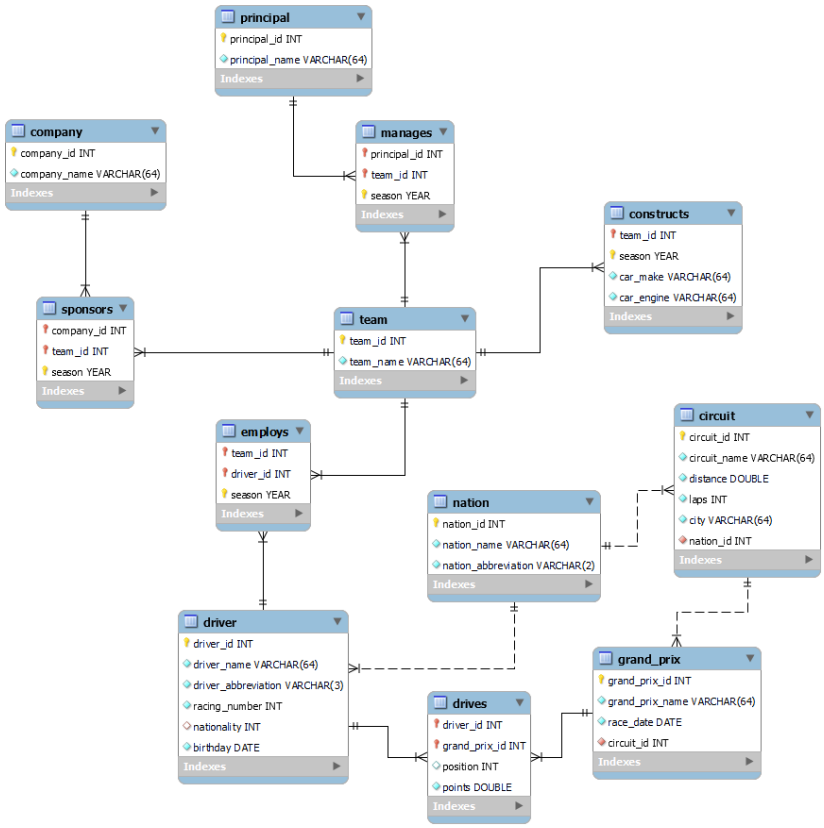
- Use Python 3.9.x (tested on both versions 3.9.5 and 3.8.8), download and install from https://www.python.org/downloads/ if not already installed
- Use MySQL Workbench 8.0.x (tested on version 8.0.28), download and install from https://dev.mysql.com/downloads/workbench/ if not already installed
- Necessary libraries:
    - Install "pymysql" library for Python, using the following command in terminal:
        - *pip install pymysql*
        - Installation Directory (default): C:\Users*UserName*\AppData\Local\Programs\Python\Python39\Lib\site-packages
    - Install "tabulate" library for Python, using the following command in terminal:
        - *pip install tabulate*
        - Installation Directory (default): C:\Users*UserName*\AppData\Local\Programs\Python\Python39\Lib\site-packages
- Import schema from "f1db-dump.sql" using "Server -> Data Import -> Import from Self-Contained File" in MySQL Workbench
- Run "application.py" Python script in terminal/command line or chosen application capable of running Python scripts (IDEs such as VS Code, Spyder, etc.)
    - If using terminal/command line, use the command *py application.py* to run the program after navigating to the file's directory

The technical specifications for this project are any computer that can run version 3.9.x of Python and version 8.0.x of MySQL Workbench. This script was only tested on Windows, but should function correctly regardless of operating system. The f1db database application requires the *pymysql* Python library to connect the front-end Python code to the back-end MySQL database that stores and organizes the data. It also requires the *tabulate* Python library to handle visual output of retrieved data in the command line.

Conceptual UML Diagram for f1db:



Logical design of final f1db schema:



User Interaction / Flow:

- **First, the user runs the application on python**
- **The user enters the correct username**
- **The user enters the correct password**
- **The user is brought to the main menu and is prompted to choose one of the following operations: CALCULATE, CREATE, READ, UPDATE, or DELETE**
- **To calculate rankings:**
    - **When prompted, user types in the terminal “CALCULATE”**

- **When prompted, user types in the name of the operation they wish to calculate, selecting from DRIVER_STANDINGS, TEAM_STANDINGS, or DRIVER_PROFILES**
    - **If DRIVER_STANDINGS, or TEAM_STANDINGS is selected, user types a supported year (database currently only supports 2021)**
        - **If the year does not exist, an error statement is shown explaining the problem**
        - **If the year is written correctly, then the calculations for that year will be shown**
    - **If DRIVER_PROFILES is selected, user is shown the calculated driver profiles**
    - **If the selected operation is not supported, an error message is shown explaining the problem**
- **The user is returned to the main menu**

- **To create a record:**
    - **When prompted, user types in the terminal "CREATE"**
    - **When prompted, user types the name of one of the listed tables**
        - **If the table name is not written correctly, an error message is shown and the user is returned to the main menu**
    - **When prompted user types in the desired values for each field in the table**
        - **If the data does not meet constraints (NOT NULL, foreign key constraints, incorrect data types, etc.), an error statement is shown explaining the problem**

        - **If the data does meet constraints, the record is created**
    - **The user is returned to the main menu**
- **To read a record:**
    - **When prompted, user types in the terminal “READ”**
    - **When prompted, user types the name of one of the listed tables**
        - **If the table name is not written correctly, an error statement is shown explaining the problem**
        - **If the table name is written correctly, all of the data under that table is displayed in the terminal**
    - **The user is returned to the main menu**
- **To update a record:**
    - **When prompted, user types in the terminal “UPDATE”**
    - **User types the name of one of the listed tables**
        - **If the table name is not written correctly, an error statement is shown explaining the problem**
    - **When prompted, user chooses the tuple to update by typing the primary key of the tuple**
        - **If the primary key does not exist, an error statement is shown explaining the problem and the user is returned to the main menu**
    - **User chooses the field of the tuple to update**
        - **If the field does not exist, an error statement is shown explaining the problem and the user is returned to the main menu**
    - **User chooses the the value of the field to update**

        - **If the value is invalid, an error statement is shown explaining the problem**
        - **If the value is written correctly, then that tuple will be updated**
    - **The user is returned to the main menu**
- **To delete a record:**
    - **When prompted, user types in the terminal "DELETE"**
    - **User types the name of one of the listed tables**
        - **If the table name is not written correctly, an error statement is shown explaining the problem and the user is returned to the main menu**
    - **User chooses the tuple to delete by typing the primary key of the tuple**
        - **If the primary key does not exist or is written incorrectly, an error statement is shown explaining the problem**
        - **If the value is written correctly, then that tuple will be deleted**
    - **The user is returned to the main menu**

Lessons Learned:

While working on the project, we learned a significant amount about how to develop database applications as we built our MySQL schema and Python front-end. One of the biggest technical issues to overcome was to make the application outputs readable. To do this, we researched available Python libraries and discovered **tabulate**, a Python library that displays data and prompts in a digestible, logical manner, drastically improving f1db's user experience. Another technical issue we overcame was how to efficiently validate inputs to make text entry flexible for users and determine what should be given an error statement and what shouldn't. These struggles slowed our development, but they ultimately made our application significantly

more accessible and easy to use. To manage our time and development process, we met frequently in-person and updated each other through text. This way we always knew what the other person was working on and what our next step should be to finish the project before the deadline. We both learned how to use GitHub to share our work by downloading the git library on Python and learning the commands to commit, push, and pull each other's changes to ensure an efficient development process. While working on the project we decided to add more tables to our database, as we realized that our current schema did not adequately handle the discrepancies caused by Formula One's ever-changing regulations and format. Doing so made our project more representative of Formula One and the wealth of data it generates. We also decided to add another function to the CRUD list by adding a CALCULATE feature. All of our code works and has no technical issues.

Future Work:

Looking forward towards future uses for the f1db database, we hope to expand the number of supported seasons, adding data for not only the current 2022 season and beyond, but also the historic seasons dating back to the first championship series in 1950. These older seasons had very different rules and regulations, making the records severely different from modern records, but the schema was specifically designed to support the older generation of Formula One's different format. For example, the point values earned for positions have never been consistent, but the current database allows for point values and positions earned during each race to vary independently, which allows for these discrepancies over time to exist and allows the database to operate properly as the sport evolves. Similar considerations were taken with the fact that teams used to have more than two drivers, and originally were not required to construct their

own cars, and as Formula One continues to change, f1db's intentional modularity will ensure future functionality.

In addition to improving the breadth of data covered by f1db, we hope to develop a more accessible front-end experience through a web-application. Although our team lacks web development experience, a simplistic browser-based program with basic graphics would still be far more appealing to an average user than a text-based terminal application. This iteration of the application could also implement several desired features, such as live data-scraping of current races, graph displays for interesting statistics, and additional calculations for users to view, including fastest laps and unique world records held by the various drivers. Overall, we hope to continue building upon the success of f1db after the completion of CS 3200, creating a truly useful tool for users looking to learn more about the data behind this incredible motorsport.