

About the Database System

This Formula 1 database system is designed to view and analyze data pertaining to the Formula 1 sport dating back to 1950. It provides comprehensive access to a wealth of information concerning drivers, constructors, circuits, races, and various performance metrics including constructor standings, driver results, lap times, pit stops, and qualifying sessions.

Given the project's scope, certain limitations are placed on the database functionalities. Not all tables will be accessible for modifications or advanced querying as the primary objective is to demonstrate distinct examples of database operations. Should the database system be developed further, it could become an invaluable resource for data analysts seeking deep insights into the dynamic world of Formula 1 racing, offering rich analytical capabilities to explore trends, performance, and historical data.

GitHub Repository

The repository can be accessed by clicking [here](https://github.com/juleshetzler/CMPSC431WFinalProject.git) or by copying and pasting the following URL into your browser:

<https://github.com/juleshetzler/CMPSC431WFinalProject.git>

CLI Interface

1. Add a driver, circuit, or constructor:

When selecting this option, the user is prompted to again enter a number based on the following options.

Adding a driver:

Upon choosing to add a driver, the user will be able to insert a driver into the Formula 1 database. This will not create races for that driver, add statistics for that driver, or add a constructor associated with that driver.

The CLI will prompt the user to add a driver identification number for that driver, a first name, a last name, a driver code in the form of a 3 or less letter string (the first three letters of the driver's last name), the driver's date of birth, nationality, and home location. Please note that nationality does not determine home location as it is possible a driver has moved since birth.

Example Input:

```
Enter the driver identification number (ex: 900): 900
Enter the first name of the driver (ex: 'Jules'): Jules
Enter the last name of the driver (ex: 'Hetzler'): Hetzler
Enter the three character driver code (ex: 'HET'): HET
Enter the driver's date of birth (ex: '2002-08-16'): 2002-08-16
Enter the driver's nationality (ex: 'American'): American
Enter the driver's current country of residence (ex: 'USA'): USA
```

Upon successful completion of the insertion, the user receives a message denoting the driver was successfully added to the database. If the insertion was not successful, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Before Query:

Data Output Messages Notifications							
	driverid [PK] integer	driver_forename character varying (50)	driver_surname character varying (50)	driver_code character varying (3)	driver_dob date	driver_nationality character varying (25)	driver_home character varying (25)
1	858	Logan	Sargeant	SAR	2000-12-31	American	USA

After Query:

Data Output Messages Notifications							
	driverid [PK] integer	driver_forename character varying (50)	driver_surname character varying (50)	driver_code character varying (3)	driver_dob date	driver_nationality character varying (25)	driver_home character varying (25)
1	900	Jules	Hetzler	HET	2002-08-16	American	USA

Adding a circuit:

Upon choosing to add a circuit, the user will be able to insert a circuit into the Formula 1 database. This operation will insert the circuit into `Circuit`, `CircuitLocation`, and `CircuitCountry`.

The CLI will prompt the user to add a circuit identification number for that circuit, a name for the circuit, a location for the circuit, and a country where the circuit resides.

Example Input:

```
Enter the circuit identification number (ex: 90): 90
Enter the name of the circuit (ex: 'Maple Treeway'): Maple
Treeway
Enter the name of the circuit location (ex: 'Brooklyn'):
Brooklyn
Enter the name of the circuit country (ex: 'USA'): USA
```

Upon successful completion of the insertion, the user receives a message denoting the circuit was successfully added to the database. If the insertion was not successful, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Before Query:

Data Output

Messages


Notifications


	circuitid [PK] integer	circuit_name character varying (255)
1	80	Las Vegas Strip Street Circuit


Data Output


Messages


Notifications




























	circuitid [PK] integer	circuit_location character varying (50)
1	80	Miami

Data Output		Messages	Notifications						
									
	circuitid [PK] integer	circuitcountry character varying (25)							
1	80	USA							


After Query:

Data Output


Messages

Notifications


≡+





▼




▼

















	<div><div>circuitid</div><div>[PK] integer</div></div>	<div><div>circuit_name</div><div>character varying (255)</div></div>
1	90	Maple Treeway

Data Output		Messages	Notifications						
									
	circuitid [PK] integer		circuit_location character varying (50)						
1	90		Brooklyn						

Data Output			Messages	Notifications
	circuitid [PK] integer	circuitcountry character varying (25)		
1	90	USA		

Adding a constructor:

Upon choosing to add a constructor, the user will be able to insert a constructor into the Formula 1 database. This operation will insert the circuit into all tables relevant to the circuit's information. This will not create races in which that constructor raced or add drivers associated with that constructor.

The CLI will prompt the user to add a constructor identification number for that constructor, a name for the constructor, and a home country for the constructor.

Example Input:

```
Enter the constructor identification number (ex: 230): 230
Enter the constructor name (ex: 'MINI'): MINI
Enter the constructor home (ex: 'UK'): UK
```

Upon successful completion of the insertion, the user receives a message denoting the constructor was successfully added to the database. If the insertion was not successful, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Before Query:

Data Output			Messages	Notifications
	constructorid [PK] integer	constructor_name character varying (25)	constructor_home character varying (25)	
1	214	Alpine F1 Team	France	

After Query:

Data Output Messages Notifications			
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>			
	constructorid [PK] integer	constructor_name character varying (25)	constructor_home character varying (25)
1	230	MINI	UK

2. Delete a driver, circuit, or constructor

Deleting a driver:

Upon choosing to delete a driver, the user will be able to delete a driver from the Formula 1 database. This will not delete races for that driver, delete statistics for that driver, or delete constructors associated with that driver, it will only remove the driver from consideration for further data.

The CLI will prompt the user to enter the driver identification number for the driver the user wishes to remove from the database.

Example Input:

```
Enter the driver identification number (ex: 900): 900
```

(This removal is valid only after the insertion example from "Adding a driver" option above is completed.)

Upon successful completion of the deletion, the user receives a message denoting the driver was successfully deleted from the database. If the deletion was not successful, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Before Query:

Data Output Messages Notifications							
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>							
	driverid [PK] integer	driver_forename character varying (50)	driver_surname character varying (50)	driver_code character varying (3)	driver_dob date	driver_nationality character varying (25)	driver_home character varying (25)
1	900	Jules	Hetzler	HET	2002-08-16	American	USA

After Query:

Data Output Messages Notifications							
	driverid [PK] integer	driver_forename character varying (50)	driver_surname character varying (50)	driver_code character varying (3)	driver_dob date	driver_nationality character varying (25)	driver_home character varying (25)
1	858	Logan	Sargeant	SAR	2000-12-31	American	USA

Deleting a circuit:

Upon choosing to delete a circuit, the user will be able to delete a circuit from the Formula 1 database. This will delete the circuit from `Circuit`, `CircuitLocation`, and `CircuitCountry`.

The CLI will prompt the user to enter the circuit identification number for the circuit the user wishes to remove from the database.

Example Input:

Enter the circuit identification number (ex: 90): **90**












(This removal is valid only after the insertion example from "Adding a circuit" option above is completed.)

Upon successful completion of the deletion, the user receives a message denoting the circuit was successfully deleted from the database. If the deletion was not successful, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Before Query:

Data Output Messages Notifications		
	circuitid [PK] integer	circuit_name character varying (255)
1	90	Maple Treeway

Data Output Messages Notifications		
	circuitid [PK] integer	circuit_location character varying (50)
1	90	Brooklyn

Data Output		Messages	Notifications					
								
	circuitid [PK] integer 	circuitcountry character varying (25) 						
1	90	USA						

After Query:

Data Output

Messages

Notifications

circuitid

[PK] integer

circuit_name

character varying (255)

1

80

Las Vegas Strip Street Circuit

Data Output

Messages

Notifications

circuitid

[PK] integer










circuit_location

character varying (50)

1

80

Miami

Data Output		Messages	Notifications					
								
	circuitid [PK] integer		circuitcountry character varying (25)					
1	80		USA					

Deleting a constructor:

Upon choosing to delete a constructor, the user will be able to delete a constructor from the Formula 1 database. This will not delete races in which that constructor raced or delete drivers associated with that constructor.

The CLI will prompt the user to enter the constructor identification number for the constructor the user wishes to remove from the database.








Example Input:

Enter the constructor identification number (ex: 230): **230**










(This removal is valid only after the insertion example from "Adding a constructor" option above is completed.)

Upon successful completion of the deletion, the user receives a message denoting the circuit was successfully deleted from the database. If the deletion was not successful, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Before Query:

Data Output Messages Notifications			
        			
	constructorid [PK] integer	constructor_name character varying (25)	constructor_home character varying (25)
1	230	MINI	UK

After Query:

Data Output Messages Notifications			
        			
	constructorid [PK] integer	constructor_name character varying (25)	constructor_home character varying (25)
1	214	Alpine F1 Team	France

3. Update a driver's home country

Upon choosing to update a driver's home country, the user will be able to update the driver's residence in the Formula 1 database.

The CLI will prompt the user to enter the driver identification number for the driver the user wishes to update and the new home location.

Example Input:

```
Enter the driver identification number (ex: 858): 858
Enter the driver's new home (ex: 'Israel'): Israel
```

Upon successful completion of the update, the user receives a message denoting the operation was successful. If the update was not successful, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Before Query:

Data Output Messages Notifications							
	driverid [PK] integer	driver_forename character varying (50)	driver_surname character varying (50)	driver_code character varying (3)	driver_dob date	driver_nationality character varying (25)	driver_home character varying (25)
1	858	Logan	Sargeant	SAR	2000-12-31	American	USA

After Query:

Data Output Messages Notifications							
	driverid [PK] integer	driver_forename character varying (50)	driver_surname character varying (50)	driver_code character varying (3)	driver_dob date	driver_nationality character varying (25)	driver_home character varying (25)
1	858	Logan	Sargeant	SAR	2000-12-31	American	Israel

4. Search all races for a driver

Upon choosing to search races for a particular driver, the user will be able to find all races associated with a driver of their choosing.

The CLI will prompt the user to enter the last name for the driver for which the user would like to find races (please note that entering the last name is case-sensitive). This prompts for the last name for ease of use as users may not know the driver identification number associated with the driver as it is unique to this database system.

Example Input:

```
Enter the driver's last name (ex: 'Hamilton', case-sensitive):  
Hamilton
```

Upon successful completion of the search, if results are found, a list will be generated including all relevant race information and the circuit identification number and circuit name. The circuit identification number and circuit name is included for usability. If no results are found, the program will notify the user. If an error occurred during the search, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Output:

Race ID	Year	Round	Circuit ID	Circuit Name

18	2008	1	1	Albert Park Grand Prix Circuit
19	2008	2	2	Sepang International Circuit
20	2008	3	3	Bahrain International Circuit

(There are more entries in the list not shown here for brevity but shown in demo video.)

5. Find total points for a team

Upon choosing to find all points for a given team, the user will be able to find the total points associated with a team.

The CLI will prompt the user to enter the constructor name for which the user would like to find points (please note that entering the team name is case-sensitive). This prompts for the team name for ease of use as users may not know the constructor identification number associated with the constructor as it is unique to this database system.

Example Input:

```
Enter the team name (ex: 'Ferrari', case-sensitive): Ferrari
```

Upon successful completion of the operation, if results are found, the total points for that team will be displayed. If no points were found, the program will notify the user. If an error occurred, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Output:



```
Total points for Ferrari: 91389.0
```

(There are more entries in the list not shown here for brevity but shown in demo video.)

6. List drivers or constructors by points

Upon choosing to sort by points, the user is prompted to again enter a number based on the following options.

Sort drivers by points:

Upon choosing to sort drivers by points, the user will be able to see all drivers by points in ascending or descending order.

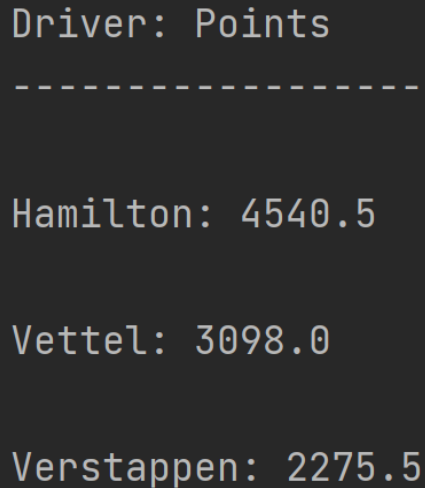
The CLI will prompt the user to enter the order in which the user would like to sort (either ascending or descending).

Example Input:

```
Enter the sort order ('ASC' for ascending, 'DESC' for  
descending): DESC
```

Upon successful completion of the operation, if results are found, a list will be generated sorting all drivers by points. If an error occurred, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Output:

A screenshot of a terminal window with a dark background and light gray text. The text displays a list of drivers and their points, separated by a dashed line. The drivers listed are Hamilton, Vettel, and Verstappen.

```
Driver: Points
-----

Hamilton: 4540.5

Vettel: 3098.0

Verstappen: 2275.5
```

(There are more entries in the list not shown here for brevity but shown in demo video.)

Sort constructors by points:

Upon choosing to sort constructors by points, the user will be able to see all drivers by points in ascending or descending order.

The CLI will prompt the user to enter the order in which the user would like to sort (either ascending or descending).

Example Input:

```
Enter the sort order ('ASC' for ascending, 'DESC' for
descending): DESC
```

Upon successful completion of the operation, if results are found, a list will be generated sorting all constructors by points. If an error occurred, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Output:

```
Constructor: Points
```

```
-----
```

```
Ferrari: 91389.0
```

```
Mercedes: 73379.5
```

```
Red Bull: 68243.5
```

(There are more entries in the list not shown here for brevity but shown in demo video.)

7. Find drivers and their teams per race

Upon choosing to find drivers and their constructors per race, the user will get a list of all drivers and their constructors for a given race.

The CLI will prompt the user to enter the race identification number for which the user would like to find drivers and constructors.

Example Input:

```
Enter the race identification number (ex: 1100): 1100
```

Upon successful completion of the operation, if results are found, a list will be generated with all the names of the drivers and their teams. If no points were found, the program will notify the user. If an error occurred, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Output:

```
Name, Constructor
-----

Max Verstappen, Red Bull

George Russell, Mercedes

Lewis Hamilton, Mercedes
```

(There are more entries in the list not shown here for brevity but shown in demo video.)

8. Group drivers by nationality

Upon choosing this option, the user will get a list of nationalities and the number of drivers associated with them.

Upon successful completion of the operation, a list will be generated with all the different nationalities and the number of drivers associated with them. If an error occurred, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Output:

```
Nationality: Number of Drivers
-----

Belgian: 23

Swiss: 23

Dutch: 18
```

(There are more entries in the list not shown here for brevity but shown in demo video.)

9. Find races in which a driver finished in a higher position than their grid position

Upon choosing this option, the user will get a list of races in which a given driver finished in a higher position than their grid position.

The CLI will prompt the user to enter the driver identification number for which the user would like to find races.

Example Input:

```
Enter the driver identification number (ex: 1): 1
```

Upon successful completion of the operation, if results are found, a list will be generated with the driver code and the race identification number. If no races were found, the program will notify the user. If an error occurred, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Output:

Driver Code	RaceID

HAM	23
HAM	26
HAM	10

(There are more entries in the list not shown here for brevity but shown in demo video.)

10. Update points for a constructor and driver

Upon choosing this option, the user will be able to update driver results and constructor points after a race has concluded.

The CLI will prompt the user to enter the driver identification number for which the user would like to update, the race identification number, constructor identification number, and the number of points.

Example Input:

```
Enter the driver identification number (ex: 857): 857
Enter the race identification number (ex: 1110): 1110
Enter the constructor identification number (ex: 1): 1
Enter the points earned (ex: 10): 10
```

Upon successful completion of the operation, the program will notify the user. If an error occurred, the program will notify the user and display the error message. The program will then return to the home menu for the next operation.

Before Query:

Data Output Messages Notifications						
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>						
	constructorstandingid [PK] integer	raceid integer	constructorid integer	points double precision	position integer	wins integer
1	28572	1110	1	103	5	0

Data Output Messages Notifications										
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>										
	resultid [PK] integer	raceid integer	driverid integer	constructorid integer	grid_position integer	position integer	points double precision	laps integer	fastestlap integer	fastestlaprank integer
1	26085	1110	857	1	5	20	0	0	0	0

After Query:

Data Output Messages Notifications						
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>📦</div> <div>⬇️</div> <div>📈</div> </div>						
	constructorstandingid [PK] integer	raceid integer	constructorid integer	points double precision	position integer	wins integer
1	28572	1110	1	113	5	0

Data Output Messages Notifications										
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>📦</div> <div>⬇️</div> <div>📈</div> </div>										
	resultid [PK] integer	raceid integer	driverid integer	constructorid integer	grid_position integer	position integer	points double precision	laps integer	fastestlap integer	fastestlaprank integer
1	26085	1110	857	1	5	20	10	0	0	0

Demo Video

The demo video showcasing all functionalities can be accessed by clicking [here](#) or by copying and pasting the following URL into your browser:

<https://youtu.be/XHr-oebJ6e8>