DBMS Report

F1 StatsHub: A Database System for Managing Formula 1 Data

By- Hrudhay R(CS226), G Praneeth(CS202)

Abstract-

The F1 StatsHub is a Streamlit-based web application designed to efficiently manage and analyze Formula 1 racing data using MySQL as the backend database. The system maintains information about drivers, constructors, cars, races, and results, providing users with a dynamic interface for viewing and manipulating data. It implements key database concepts such as stored procedure, functions, and triggers to automate operations like assigning points, recalculating race ranks, and maintaining data consistency. The application supports full CRUD operations, including adding, updating, and deleting race results, and visualizes analytical insights such as World Driver's and Constructor's Championship standings. By integrating SQL logic with a Python-driven interface, the project demonstrates the practical application of database normalization, query optimization, and backend automation for real-world data management scenarios.

User specification requirement-

1. Purpose

The purpose of the Formula 1 Database Management and Analytics System is to design and implement a database-driven application that enables efficient management, retrieval, and analysis of Formula 1 racing data. The system allows users to view, insert, update, and delete race results while automatically calculating driver standings, constructor points, and race rankings through stored procedures and triggers.

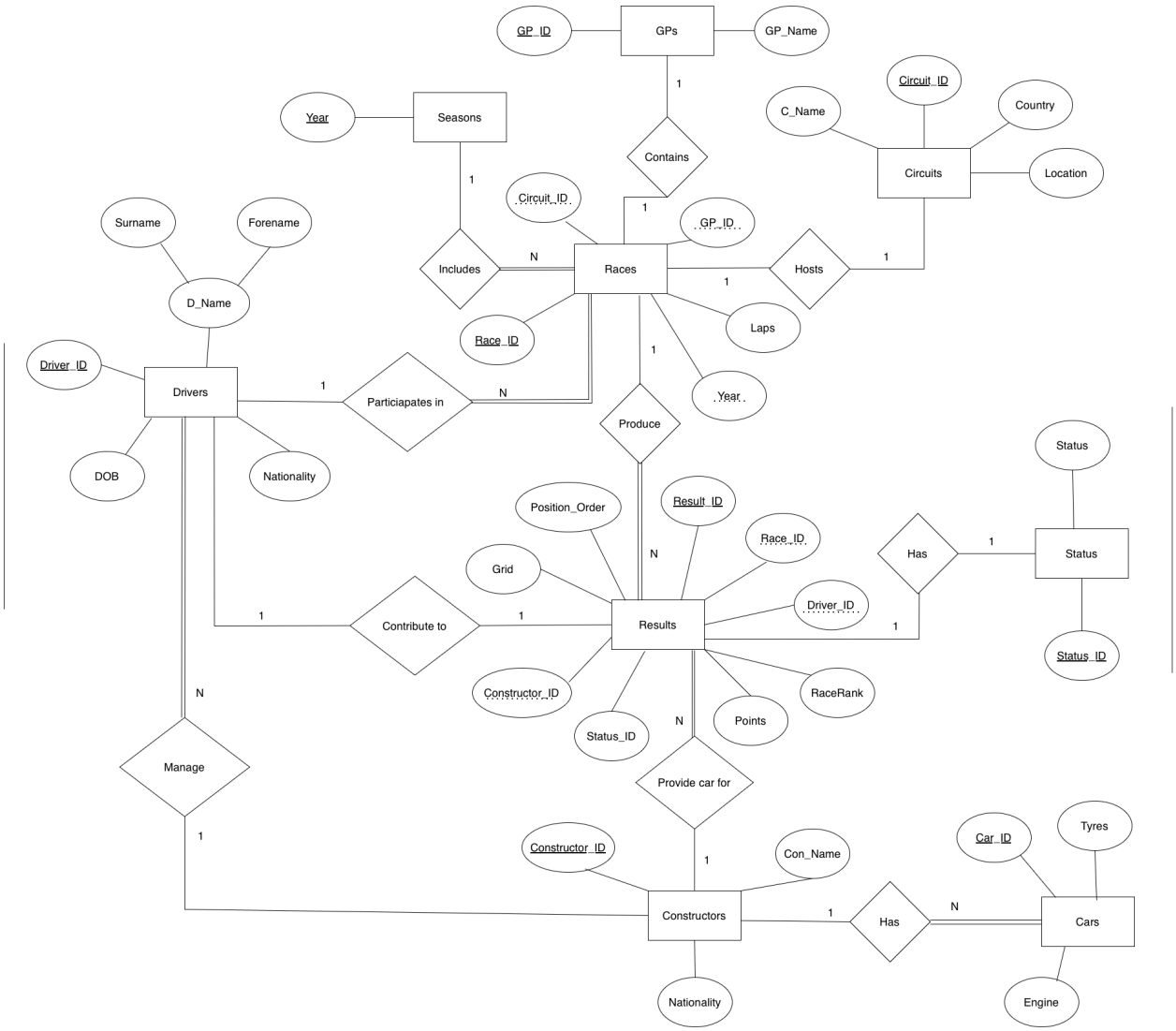
2. Intended Users

- Database Administrator (DBA): Responsible for managing database structure, user privileges, and maintenance of stored procedures and triggers.
- Application User (Analyst / Staff): Interacts with the Streamlit interface to view race results, add or modify entries, and analyze championship standings.

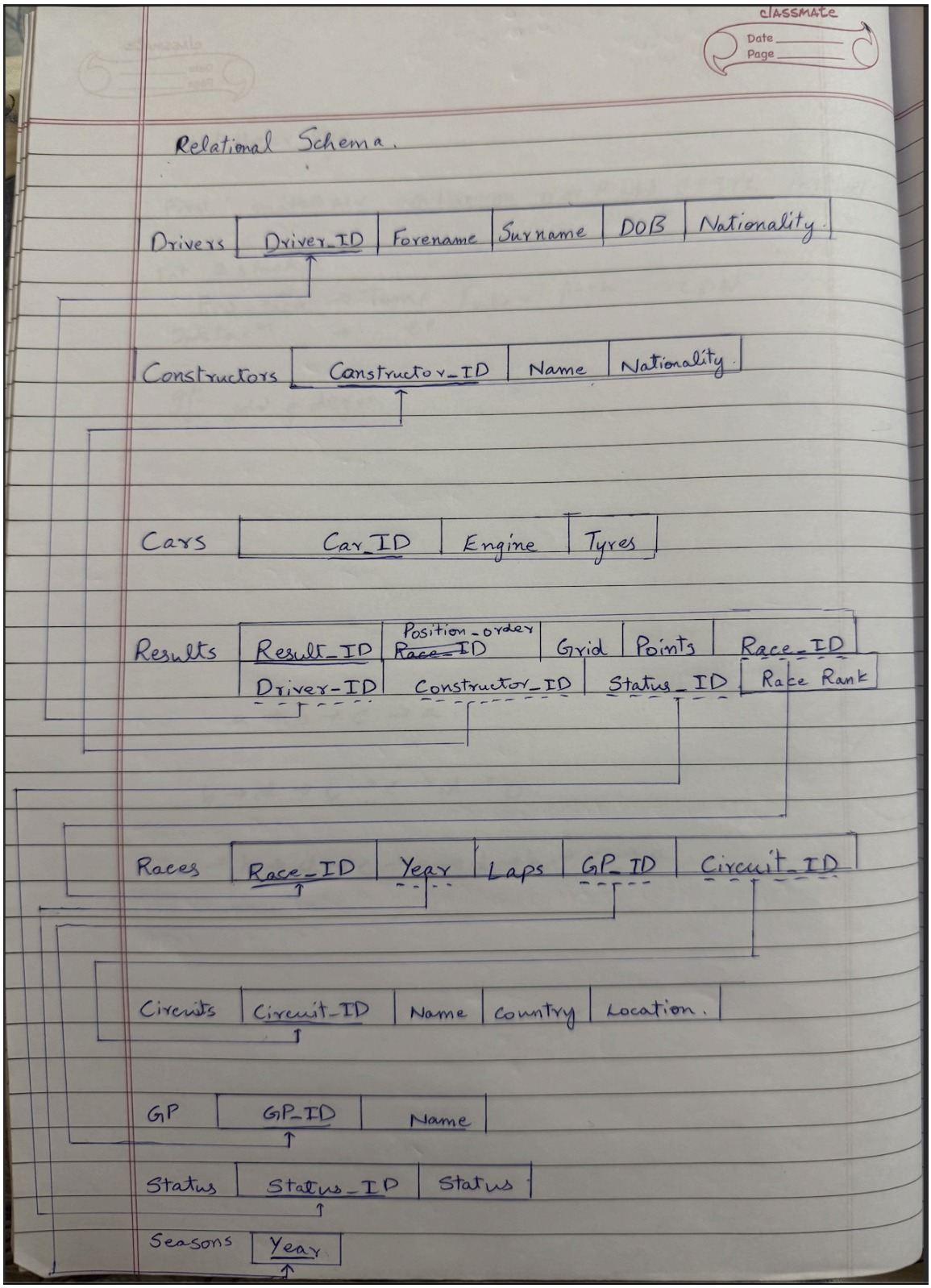
List of softwares used-

Category	Name / Tool	Purpose / Description
Frontend / GUI	Streamlit	To create the interactive web-based dashboard for managing and displaying Formula 1 data.
Backend Database	MySQL	To store, retrieve, and manage all Formula 1 data including drivers, constructors, cars, races, and results.
Programming Language	Python	Used to connect the Streamlit interface with the MySQL database and execute queries, procedures, and triggers.
Database Connectivity Library	mysql-connector-python	Provides communication between Python (Streamlit) and MySQL.
Data Handling Library	Pandas	Used for handling and displaying tabular data in dataframes within the Streamlit app.
Database Management Tool	MySQL Workbench / phpMyAdmin	For creating tables, executing SQL scripts, and managing stored procedures and triggers.
Code Editor / IDE	Visual Studio Code / PyCharm	For writing and debugging Python and SQL scripts.
Version Control (Optional)	Git & GitHub	For maintaining version control and collaborative development.

ER-diagram-


GP_ID
GPs
GP_Name
1
Year
Seasons
Contains
Circuit_ID
C_Name
Country
Circuits
Location
1
Circuit_ID
1
GP_ID
Surname
Forename
Includes
N
Races
Hosts
1
1
D_Name
Laps
Race_ID
1
Driver_ID
Drivers
1
Particiapates in
N
Year
Produce
Status
DOB
Nationality
Position_Order
Result_ID
Race_ID
Has
1
Status
N
Grid
Driver_ID
1
Contribute to
1
Results
1
Status_ID
Constructor_ID
RaceRank
N
Status_ID
Points
N
Manage
Provide car for
1
Car_ID
Tyres
Constructor_ID
1
Con_Name
Constructors
1
Has
N
Cars
Nationality
Engine

Relational schema-



DDL commands-

```
-- Circuits
CREATE DATABASE F1;
USE F1;

CREATE TABLE Circuits (
    Circuit_ID INT PRIMARY KEY,
    C_Name VARCHAR(100),
    Country VARCHAR(50),
    Location VARCHAR(100)
);

-- GPs (Grand Prix)
CREATE TABLE GPs (
    GP_ID INT PRIMARY KEY,
    GP_Name VARCHAR(100)
);

-- Seasons
CREATE TABLE Seasons (
    Year INT PRIMARY KEY
);

-- Races
CREATE TABLE Races (
    Race_ID INT PRIMARY KEY,
    Year INT,
    GP_ID INT,
    Circuit_ID INT,
    Laps INT,
    FOREIGN KEY (Year) REFERENCES Seasons(Year),
    FOREIGN KEY (GP_ID) REFERENCES GPs(GP_ID),
    FOREIGN KEY (Circuit_ID) REFERENCES Circuits(Circuit_ID)
);

-- Drivers
CREATE TABLE Drivers (
    Driver_ID INT PRIMARY KEY,
    Forename VARCHAR(50),
    Surname VARCHAR(50),
    DOB DATE,
```

```
    Nationality VARCHAR(50)
);

-- Constructors
CREATE TABLE Constructors (
    Constructor_ID INT PRIMARY KEY,
    Con_Name VARCHAR(100),
    Nationality VARCHAR(50)
);

-- Cars
CREATE TABLE Cars (
    Car_ID INT PRIMARY KEY,
    Constructor_ID INT,
    Engine VARCHAR(50),
    Tyres VARCHAR(50),
    FOREIGN KEY (Constructor_ID) REFERENCES Constructors(Constructor_ID)
);

-- Status
CREATE TABLE Status (
    Status_ID INT PRIMARY KEY,
    Status VARCHAR(50)
);

-- Results
CREATE TABLE Results (
    Result_ID INT PRIMARY KEY,
    Race_ID INT,
    Driver_ID INT,
    Constructor_ID INT,
    Car_ID INT,
    Position_Order INT,
    Grid INT,
    Points INT,
    Status_ID INT,
    FOREIGN KEY (Race_ID) REFERENCES Races(Race_ID),
    FOREIGN KEY (Driver_ID) REFERENCES Drivers(Driver_ID),
    FOREIGN KEY (Constructor_ID) REFERENCES Constructors(Constructor_ID),
    FOREIGN KEY (Car_ID) REFERENCES Cars(Car_ID),
    FOREIGN KEY (Status_ID) REFERENCES Status(Status_ID)
);

-- Seasons (10 rows: 2016..2025)
```

```
INSERT INTO Seasons (Year) VALUES (2016);
INSERT INTO Seasons (Year) VALUES (2017);
INSERT INTO Seasons (Year) VALUES (2018);
INSERT INTO Seasons (Year) VALUES (2019);
INSERT INTO Seasons (Year) VALUES (2020);
INSERT INTO Seasons (Year) VALUES (2021);
INSERT INTO Seasons (Year) VALUES (2022);
INSERT INTO Seasons (Year) VALUES (2023);
INSERT INTO Seasons (Year) VALUES (2024);
INSERT INTO Seasons (Year) VALUES (2025);

-- Circuits (10)
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (1, 'Albert Park Circuit', 'Australia', 'Melbourne');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (2, 'Shanghai International Circuit', 'China', 'Shanghai');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (3, 'Suzuka Circuit', 'Japan', 'Suzuka');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (4, 'Bahrain International Circuit', 'Bahrain', 'Sakhir');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (5, 'Jeddah Corniche Circuit', 'Saudi Arabia', 'Jeddah');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (6, 'Circuit de Monaco', 'Monaco', 'Monte Carlo');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (7, 'Silverstone Circuit', 'United Kingdom', 'Silverstone');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (8, 'Circuit de Spa-Francorchamps', 'Belgium', 'Stavelot');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (9, 'Autodromo Nazionale Monza', 'Italy', 'Monza');
INSERT INTO Circuits (Circuit_ID, C_Name, Country, Location) VALUES (10, 'Yas Marina Circuit', 'United Arab Emirates', 'Abu Dhabi');

-- GPs (10)
INSERT INTO GPs (GP_ID, GP_Name) VALUES (1, 'Australian Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (2, 'Chinese Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (3, 'Japanese Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (4, 'Bahrain Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (5, 'Saudi Arabian Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (6, 'Monaco Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (7, 'British Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (8, 'Belgian Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (9, 'Italian Grand Prix');
INSERT INTO GPs (GP_ID, GP_Name) VALUES (10, 'Abu Dhabi Grand Prix');
```

```
-- Races (10)
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (1, 2025, 1, 1, 58);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (2, 2025, 2, 2, 56);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (3, 2025, 3, 3, 53);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (4, 2025, 4, 4, 57);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (5, 2025, 5, 5, 50);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (6, 2025, 6, 6, 78);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (7, 2025, 7, 7, 52);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (8, 2025, 8, 8, 44);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (9, 2025, 9, 9, 53);
INSERT INTO Races (Race_ID, Year, GP_ID, Circuit_ID, Laps) VALUES (10, 2025, 10, 10, 58);

-- Drivers (10)
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (1, 'Max',
'Verstappen', '1997-09-30', 'Dutch');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (2, 'Lando',
'Norris', '1999-11-13', 'British');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (3, 'Oscar',
'Piastri', '2001-04-06', 'Australian');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (4, 'Charles',
'Leclerc', '1997-10-16', 'Monegasque');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (5, 'Lewis',
'Hamilton', '1985-01-07', 'British');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (6, 'George',
'Russell', '1998-02-15', 'British');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (7, 'Andrea
Kimi', 'Antonelli', '2006-08-25', 'Italian');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (8, 'Yuki',
'Tsunoda', '2000-05-11', 'Japanese');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (9,
'Fernando', 'Alonso', '1981-07-29', 'Spanish');
INSERT INTO Drivers (Driver_ID, Forename, Surname, DOB, Nationality) VALUES (10, 'Lance',
'Stroll', '1998-10-29', 'Canadian');

-- Constructors (10)
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (1, 'McLaren',
'British');
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (2, 'Ferrari',
'Italian');
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (3, 'Mercedes',
'German');
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (4, 'Red Bull
Racing', 'Austrian');
```

```
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (5, 'Williams', 'British');
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (6, 'Aston Martin', 'British');
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (7, 'Racing Bulls', 'New Zealand');
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (8, 'Kick Sauber', 'Swiss');
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (9, 'Haas', 'American');
INSERT INTO Constructors (Constructor_ID, Con_Name, Nationality) VALUES (10, 'Alpine', 'French');

-- Cars (10)
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (1, 1, 'Mercedes-AMG PU', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (2, 2, 'Ferrari PU', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (3, 3, 'Mercedes-AMG PU', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (4, 4, 'Honda RBPT', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (5, 5, 'Mercedes-AMG PU', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (6, 6, 'Mercedes-AMG PU', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (7, 7, 'Red Bull Powertrains', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (8, 8, 'Ferrari PU', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (9, 9, 'Ferrari PU', 'Pirelli');
INSERT INTO Cars (Car_ID, Constructor_ID, Engine, Tyres) VALUES (10, 10, 'Renault/Alpine PU', 'Pirelli');

-- Status (10)
INSERT INTO Status (Status_ID, Status) VALUES (1, 'Finished');
INSERT INTO Status (Status_ID, Status) VALUES (2, '+1 Lap');
INSERT INTO Status (Status_ID, Status) VALUES (3, '+2 Laps');
INSERT INTO Status (Status_ID, Status) VALUES (4, 'Disqualified');
INSERT INTO Status (Status_ID, Status) VALUES (5, 'Retired');
INSERT INTO Status (Status_ID, Status) VALUES (6, 'Accident');
INSERT INTO Status (Status_ID, Status) VALUES (7, 'Engine Failure');
INSERT INTO Status (Status_ID, Status) VALUES (8, 'DNS');
```

```
INSERT INTO Status (Status_ID, Status) VALUES (9, 'DNF');
INSERT INTO Status (Status_ID, Status) VALUES (10, 'Lap Down');

-- Results (10)
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (1, 1, 2, 1, 1, 1, 1, 25, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (2, 2, 1, 4, 4, 1, 2, 25, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (3, 3, 1, 4, 4, 1, 1, 25, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (4, 6, 2, 1, 1, 1, 1, 25, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (5, 7, 2, 1, 1, 1, 3, 25, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (6, 8, 3, 1, 1, 1, 2, 25, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (7, 9, 1, 4, 4, 1, 1, 25, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (8, 4, 4, 2, 2, 2, 4, 18, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (9, 5, 9, 6, 6, 6, 6, 8, 1);
INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID,
Position_Order, Grid, Points, Status_ID) VALUES (10, 10, 5, 2, 2, 4, 5, 12, 1);
```

Triggers, functions, procedure-

```
-- ================================================
-- F1 DB: Procedures, Triggers, Functions
-- Works with your existing data
-- ================================================

USE F1;
-------------------------------------------------------------------------------
-- 1) FUNCTION: getDriverFullName(driver_id)
-- Returns "Forename Surname" for a given Driver_ID
-------------------------------------------------------------------------------
DELIMITER $$
CREATE FUNCTION getDriverFullName(dID INT) RETURNS VARCHAR(200)
DETERMINISTIC
BEGIN
    DECLARE fname VARCHAR(50);
```

```
    DECLARE sname VARCHAR(50);
    SELECT Forename, Surname INTO fname, sname
    FROM Drivers
    WHERE Driver_ID = dID;
    RETURN CONCAT_WS(' ', fname, sname);
END$$
DELIMITER ;

-------------------------------------------------------------------------------
-- 2) TRIGGER: Before insert on Results
-- Automatically assigns Points if NULL or 0 based on Position_Order
-------------------------------------------------------------------------------
DELIMITER $$
CREATE TRIGGER trg_results_before_insert
BEFORE INSERT ON Results
FOR EACH ROW
BEGIN
    IF NEW.Points IS NULL OR NEW.Points = 0 THEN
        SET NEW.Points =
            CASE
                WHEN NEW.Position_Order = 1 THEN 25
                WHEN NEW.Position_Order = 2 THEN 18
                WHEN NEW.Position_Order = 3 THEN 15
                WHEN NEW.Position_Order = 4 THEN 12
                WHEN NEW.Position_Order = 5 THEN 10
                WHEN NEW.Position_Order = 6 THEN 8
                WHEN NEW.Position_Order = 7 THEN 6
                WHEN NEW.Position_Order = 8 THEN 4
                WHEN NEW.Position_Order = 9 THEN 2
                WHEN NEW.Position_Order = 10 THEN 1
                ELSE 0
            END;
    END IF;
END$$
DELIMITER ;
-------------------------------------------------------------------------------
-- 3) TRIGGER: Before insert on Results
-- Throws an error if duplicate
-------------------------------------------------------------------------------
DELIMITER $$

CREATE TRIGGER PreventDuplicateResults
BEFORE INSERT ON Results
FOR EACH ROW
```

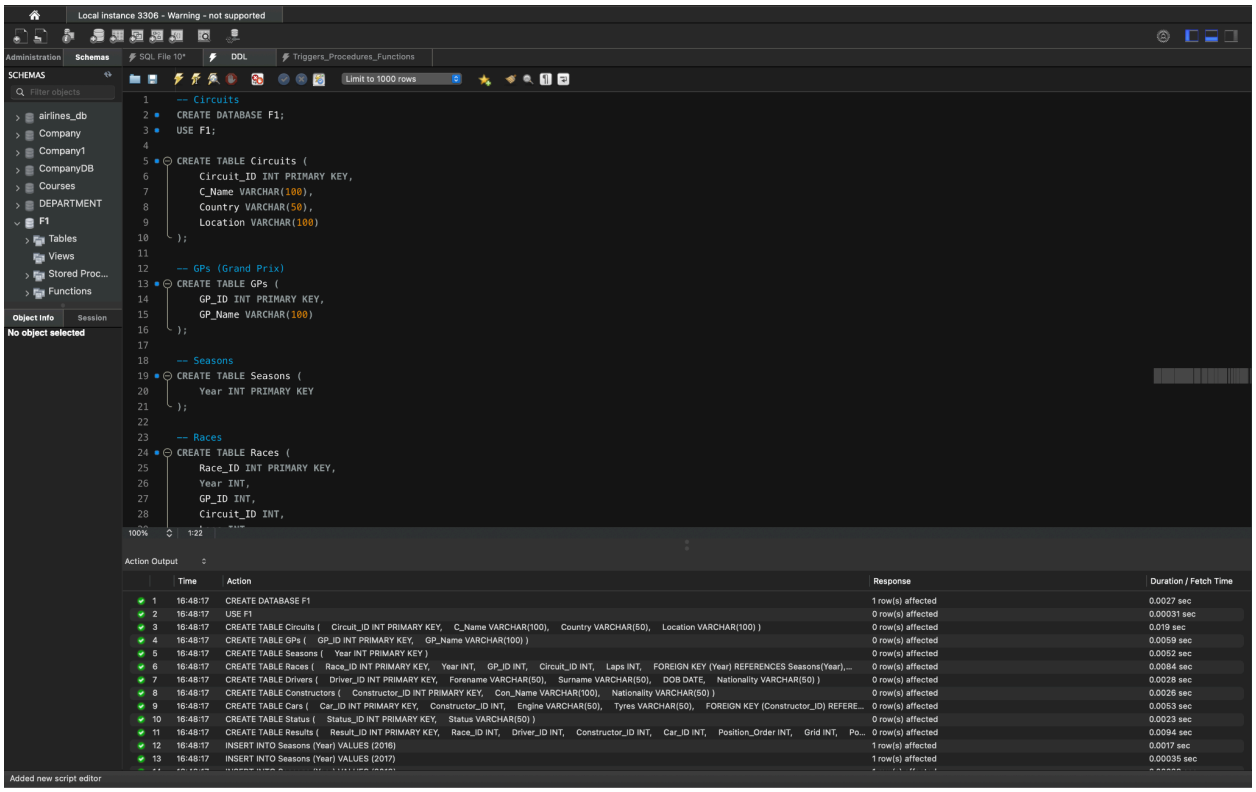
```
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Results
        WHERE Driver_ID = NEW.Driver_ID
          AND Constructor_ID = NEW.Constructor_ID
          AND Position_Order = NEW.Position_Order
          AND Race_ID = NEW.Race_ID
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Duplicate entry: This driver-constructor-position combination
already exists for the race.';
    END IF;
END$$

DELIMITER ;

-------------------------------------------------------------------------------
-- 4) PROCEDURE: RecalculateAllRaceRanks()
-- Recalculates RaceRank for every race in Results.
-- Returns all rows grouped by Race_ID and sorted by RaceRank.
-------------------------------------------------------------------------------
DELIMITER $$
CREATE PROCEDURE RecalculateAllRaceRanks()
BEGIN
    -- Initialize variables
    SET @current_race := 0;
    SET @rank := 0;

    -- Update all rows with proper ranking
    UPDATE Results r
    JOIN (
        SELECT
            Result_ID,
            Race_ID,
            @rank := IF(@current_race = Race_ID, @rank + 1, 1) AS new_rank,
            @current_race := Race_ID
        FROM Results
        CROSS JOIN (SELECT @rank := 0, @current_race := 0) AS vars
        ORDER BY Race_ID ASC, Position_Order ASC
    ) AS sub
    ON r.Result_ID = sub.Result_ID
    SET r.RaceRank = sub.new_rank;
```

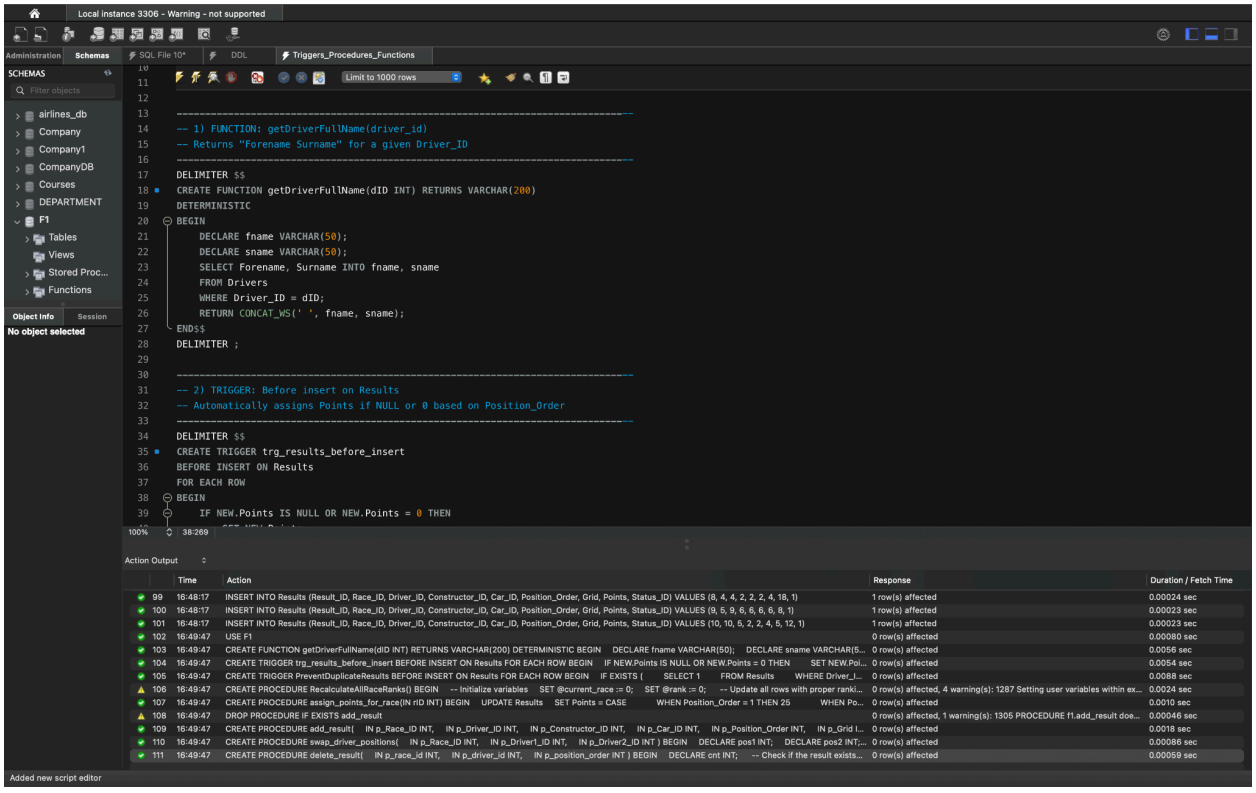
```
    -- Return all results sorted
    SELECT *
    FROM Results
    ORDER BY Race_ID ASC, RaceRank ASC;
END$$

DELIMITER ;

-- ------------------------------------------------------------------------
-- 5) PROCEDURE: assign_points_for_race(race_id)
-- Updates Points for all results of a race based on Position_Order
-- ------------------------------------------------------------------------
DELIMITER $$
CREATE PROCEDURE assign_points_for_race(IN rID INT)
BEGIN
    UPDATE Results
    SET Points = CASE
            WHEN Position_Order = 1 THEN 25
            WHEN Position_Order = 2 THEN 18
            WHEN Position_Order = 3 THEN 15
            WHEN Position_Order = 4 THEN 12
            WHEN Position_Order = 5 THEN 10
            WHEN Position_Order = 6 THEN 8
            WHEN Position_Order = 7 THEN 6
            WHEN Position_Order = 8 THEN 4
            WHEN Position_Order = 9 THEN 2
            WHEN Position_Order = 10 THEN 1
            ELSE 0
        END
    WHERE Race_ID = rID;
END$$
DELIMITER ;

-- ------------------------------------------------------------------------
-- 6) PROCEDURE: add_result(...)
-- Inserts a result row with validation
-- Points will be auto-assigned by trigger
-- ------------------------------------------------------------------------
DELIMITER $$

DROP PROCEDURE IF EXISTS add_result $$
CREATE PROCEDURE add_result(
    IN p_Race_ID INT,
    IN p_Driver_ID INT,
```

```
    IN p_Constructor_ID INT,
    IN p_Car_ID INT,
    IN p_Position_Order INT,
    IN p_Grid INT,
    IN p_Status_ID INT
)
BEGIN
    -- Validate foreign keys
    IF NOT EXISTS (SELECT 1 FROM Races WHERE Race_ID = p_Race_ID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Race does not exist';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM Drivers WHERE Driver_ID = p_Driver_ID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Driver does not exist';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM Constructors WHERE Constructor_ID =
p_Constructor_ID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Constructor does not exist';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM Cars WHERE Car_ID = p_Car_ID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Car does not exist';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM Status WHERE Status_ID = p_Status_ID) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Status does not exist';
    END IF;

    -- Insert new result row
    INSERT INTO Results (Race_ID, Driver_ID, Constructor_ID, Car_ID, Position_Order, Grid,
Status_ID)
    VALUES (p_Race_ID, p_Driver_ID, p_Constructor_ID, p_Car_ID, p_Position_Order, p_Grid,
p_Status_ID);

    -- Recalculate RaceRank using ROW_NUMBER()
    WITH ranked AS (
        SELECT Result_ID,
               ROW_NUMBER() OVER (PARTITION BY Race_ID ORDER BY Position_Order ASC)
AS new_rank
        FROM Results
        WHERE Race_ID = p_Race_ID
    )
    UPDATE Results r
```

```
    JOIN ranked rk ON r.Result_ID = rk.Result_ID
    SET r.RaceRank = rk.new_rank;

END $$

DELIMITER ;

-- ------------------------------------------------------------------------
-- 7) PROCEDURE: swap
-- Swaps any two driver's finishing posi in result table
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE PROCEDURE swap_driver_positions(
    IN p_Race_ID INT,
    IN p_Driver1_ID INT,
    IN p_Driver2_ID INT
)
BEGIN
    DECLARE pos1 INT;
    DECLARE pos2 INT;

    -- Get both drivers' positions in the race
    SELECT Position_Order INTO pos1
    FROM Results
    WHERE Race_ID = p_Race_ID AND Driver_ID = p_Driver1_ID;

    SELECT Position_Order INTO pos2
    FROM Results
    WHERE Race_ID = p_Race_ID AND Driver_ID = p_Driver2_ID;

    -- Validate existence
    IF pos1 IS NULL OR pos2 IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'One or both drivers not found in the
race.';
    END IF;

    -- Swap positions
    UPDATE Results
    SET Position_Order = CASE
        WHEN Driver_ID = p_Driver1_ID THEN pos2
        WHEN Driver_ID = p_Driver2_ID THEN pos1
    END
    WHERE Race_ID = p_Race_ID AND Driver_ID IN (p_Driver1_ID, p_Driver2_ID);
```

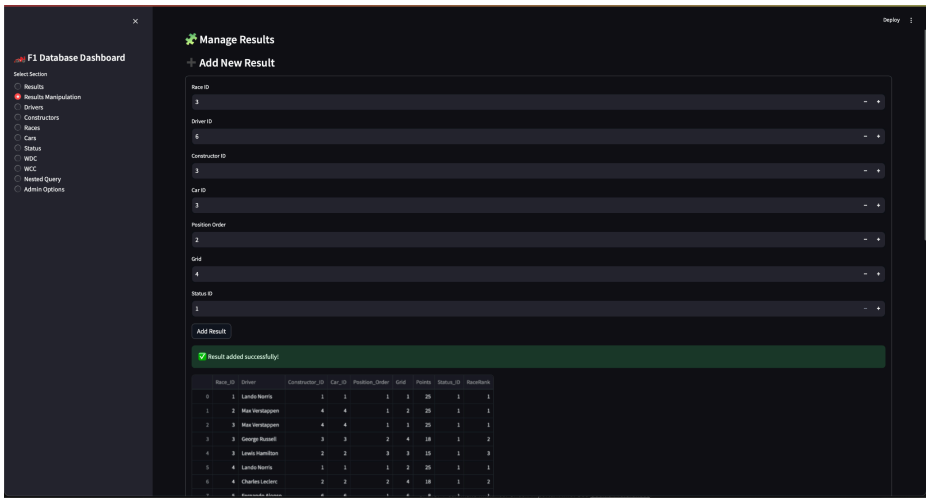
```
    -- Recalculate race ranks after swap
    CALL RecalculateAllRaceRanks();
END$$

DELIMITER ;

-- ------------------------------------------------------------------------
-- 8) PROCEDURE: delete a result tuple
-- Deletes a tuple from result table
-- ------------------------------------------------------------------------
DELIMITER $$
CREATE PROCEDURE delete_result(
    IN p_race_id INT,
    IN p_driver_id INT,
    IN p_position_order INT
)
BEGIN
    DECLARE cnt INT;

    -- Check if the result exists
    SELECT COUNT(*) INTO cnt
    FROM Results
    WHERE Race_ID = p_race_id
      AND Driver_ID = p_driver_id
      AND Position_Order = p_position_order;

    IF cnt = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '❌No such result found for this race, driver, and position.';
    ELSE
        DELETE FROM Results
        WHERE Race_ID = p_race_id
          AND Driver_ID = p_driver_id
          AND Position_Order = p_position_order
        LIMIT 1;
    END IF;
END$$
DELIMITER ;
```


Local instance 3306 - Warning - not supported
Administration
Schemas
SQL File 10*
DDL
Triggers_Procedures_Functions
SCHEMAS
Filter objects
airlines_db
Company
Company1
CompanyDB
Courses
DEPARTMENT
F1
Tables
Views
Stored Proc...
Functions
Object Info
Session
No object selected
Limit to 1000 rows
-- Circuits
CREATE DATABASE F1;
USE F1;
CREATE TABLE Circuits (
Circuit_ID INT PRIMARY KEY,
C_Name VARCHAR(100),
Country VARCHAR(50),
Location VARCHAR(100)
);
-- GPs (Grand Prix)
CREATE TABLE GPs (
GP_ID INT PRIMARY KEY,
GP_Name VARCHAR(100)
);
-- Seasons
CREATE TABLE Seasons (
Year INT PRIMARY KEY
);
-- Races
CREATE TABLE Races (
Race_ID INT PRIMARY KEY,
Year INT,
GP_ID INT,
Circuit_ID INT,
100%
1:22
Action Output
Time
Action
Response
Duration / Fetch Time
1 16:48:17 CREATE DATABASE F1 1 row(s) affected 0.0027 sec
2 16:48:17 USE F1 0 row(s) affected 0.00031 sec
3 16:48:17 CREATE TABLE Circuits (Circuit_ID INT PRIMARY KEY, C_Name VARCHAR(100), Country VARCHAR(50), Location VARCHAR(100)) 0 row(s) affected 0.019 sec
4 16:48:17 CREATE TABLE GPs (GP_ID INT PRIMARY KEY, GP_Name VARCHAR(100)) 0 row(s) affected 0.0059 sec
5 16:48:17 CREATE TABLE Seasons (Year INT PRIMARY KEY) 0 row(s) affected 0.0052 sec
6 16:48:17 CREATE TABLE Races (Race_ID INT PRIMARY KEY, Year INT, GP_ID INT, Circuit_ID INT, Laps INT, FOREIGN KEY (Year) REFERENCES Seasons(Year),... 0 row(s) affected 0.0084 sec
7 16:48:17 CREATE TABLE Drivers (Driver_ID INT PRIMARY KEY, Forename VARCHAR(50), Surname VARCHAR(50), DOB DATE, Nationality VARCHAR(50)) 0 row(s) affected 0.0028 sec
8 16:48:17 CREATE TABLE Constructors (Constructor_ID INT PRIMARY KEY, Con_Name VARCHAR(100), Nationality VARCHAR(50)) 0 row(s) affected 0.0026 sec
9 16:48:17 CREATE TABLE Cars (Car_ID INT PRIMARY KEY, Constructor_ID INT, Engine VARCHAR(50), Tyres VARCHAR(50), FOREIGN KEY (Constructor_ID) REFERE... 0 row(s) affected 0.0053 sec
10 16:48:17 CREATE TABLE Status (Status_ID INT PRIMARY KEY, Status VARCHAR(50)) 0 row(s) affected 0.0023 sec
11 16:48:17 CREATE TABLE Results (Result_ID INT PRIMARY KEY, Race_ID INT, Driver_ID INT, Constructor_ID INT, Car_ID INT, Position_Order INT, Grid INT, Po... 0 row(s) affected 0.0094 sec
12 16:48:17 INSERT INTO Seasons (Year) VALUES (2016) 1 row(s) affected 0.0017 sec
13 16:48:17 INSERT INTO Seasons (Year) VALUES (2017) 1 row(s) affected 0.00035 sec
Added new script editor

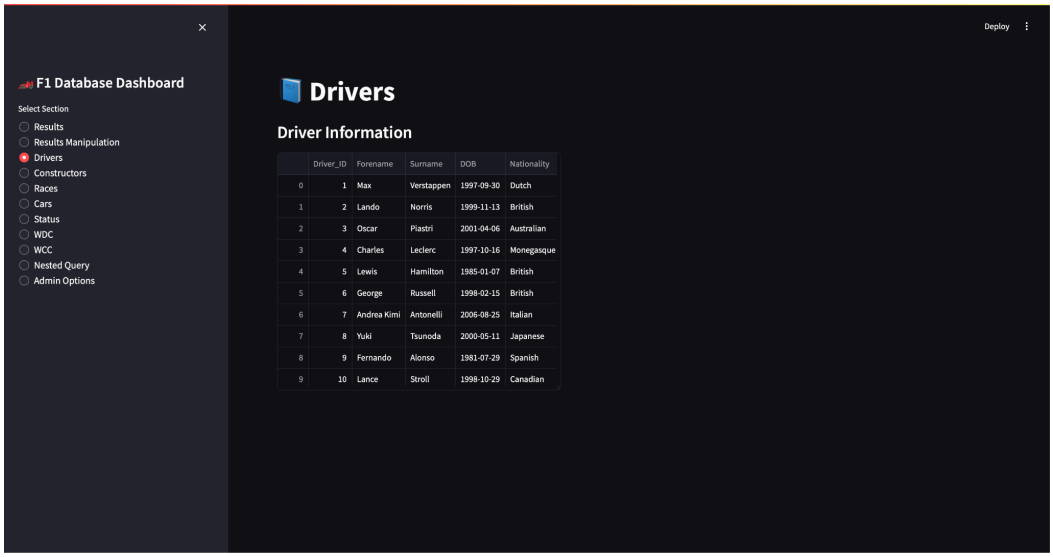

Local instance 3306 - Warning - not supported
Administration
Schemas
SQL File 10*
DDL
Triggers_Procedures_Functions
SCHEMAS
Filter objects
airlines_db
Company
Company1
CompanyDB
Courses
DEPARTMENT
F1
Tables
Views
Stored Proc...
Functions
Object Info
Session
No object selected
Limit to 1000 rows
-- 1) FUNCTION: getDriverFullName(driver_id)
-- Returns "Forename Surname" for a given Driver_ID
DELIMITER $$
CREATE FUNCTION getDriverFullName(dID INT) RETURNS VARCHAR(200)
DETERMINISTIC
BEGIN
DECLARE fname VARCHAR(50);
DECLARE sname VARCHAR(50);
SELECT Forename, Surname INTO fname, sname
FROM Drivers
WHERE Driver_ID = dID;
RETURN CONCAT_WS(' ', fname, sname);
END$$
DELIMITER ;
-- 2) TRIGGER: Before insert on Results
-- Automatically assigns Points if NULL or 0 based on Position_Order
DELIMITER $$
CREATE TRIGGER trg_results_before_insert
BEFORE INSERT ON Results
FOR EACH ROW
BEGIN
IF NEW.Points IS NULL OR NEW.Points = 0 THEN
100%
38:269
Action Output
Time
Action
Response
Duration / Fetch Time
99 16:48:17 INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID, Position_Order, Grid, Points, Status_ID) VALUES (8, 4, 4, 2, 2, 2, 4, 18, 1) 1 row(s) affected 0.00024 sec
100 16:48:17 INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID, Position_Order, Grid, Points, Status_ID) VALUES (9, 5, 9, 6, 6, 6, 6, 8, 1) 1 row(s) affected 0.00023 sec
101 16:48:17 INSERT INTO Results (Result_ID, Race_ID, Driver_ID, Constructor_ID, Car_ID, Position_Order, Grid, Points, Status_ID) VALUES (10, 10, 5, 2, 2, 4, 5, 12, 1) 1 row(s) affected 0.00023 sec
102 16:49:47 USE F1 0 row(s) affected 0.00080 sec
103 16:49:47 CREATE FUNCTION getDriverFullName(dID INT) RETURNS VARCHAR(200) DETERMINISTIC BEGIN DECLARE fname VARCHAR(50); DECLARE sname VARCHAR(5... 0 row(s) affected 0.0056 sec
104 16:49:47 CREATE TRIGGER trg_results_before_insert BEFORE INSERT ON Results FOR EACH ROW BEGIN IF NEW.Points IS NULL OR NEW.Points = 0 THEN SET NEW.Poi... 0 row(s) affected 0.0054 sec
105 16:49:47 CREATE TRIGGER PreventDuplicateResults BEFORE INSERT ON Results FOR EACH ROW BEGIN IF EXISTS (SELECT 1 FROM Results WHERE Driver_I... 0 row(s) affected 0.0088 sec
106 16:49:47 CREATE PROCEDURE RecalculateAllRaceRanks() BEGIN -- Initialize variables SET @current_race := 0; SET @rank := 0; -- Update all rows with proper ranki... 0 row(s) affected, 4 warning(s): 1287 Setting user variables within ex... 0.0024 sec
107 16:49:47 CREATE PROCEDURE assign_points_for_race(IN rID INT) BEGIN UPDATE Results SET Points = CASE WHEN Position_Order = 1 THEN 25 WHEN Po... 0 row(s) affected 0.0010 sec
108 16:49:47 DROP PROCEDURE IF EXISTS add_result 0 row(s) affected, 1 warning(s): 1305 PROCEDURE f1.add_result doe... 0.00046 sec
109 16:49:47 CREATE PROCEDURE add_result(IN p_Race_ID INT, IN p_Driver_ID INT, IN p_Constructor_ID INT, IN p_Car_ID INT, IN p_Position_Order INT, IN p_Grid I... 0 row(s) affected 0.0018 sec
110 16:49:47 CREATE PROCEDURE swap_driver_positions(IN p_Race_ID INT, IN p_Driver1_ID INT, IN p_Driver2_ID INT) BEGIN DECLARE pos1 INT; DECLARE pos2 INT;... 0 row(s) affected 0.00086 sec
111 16:49:47 CREATE PROCEDURE delete_result(IN p_race_id INT, IN p_driver_id INT, IN p_position_order INT) BEGIN DECLARE cnt INT; -- Check if the result exists... 0 row(s) affected 0.00059 sec
Added new script editor

Crud operations-

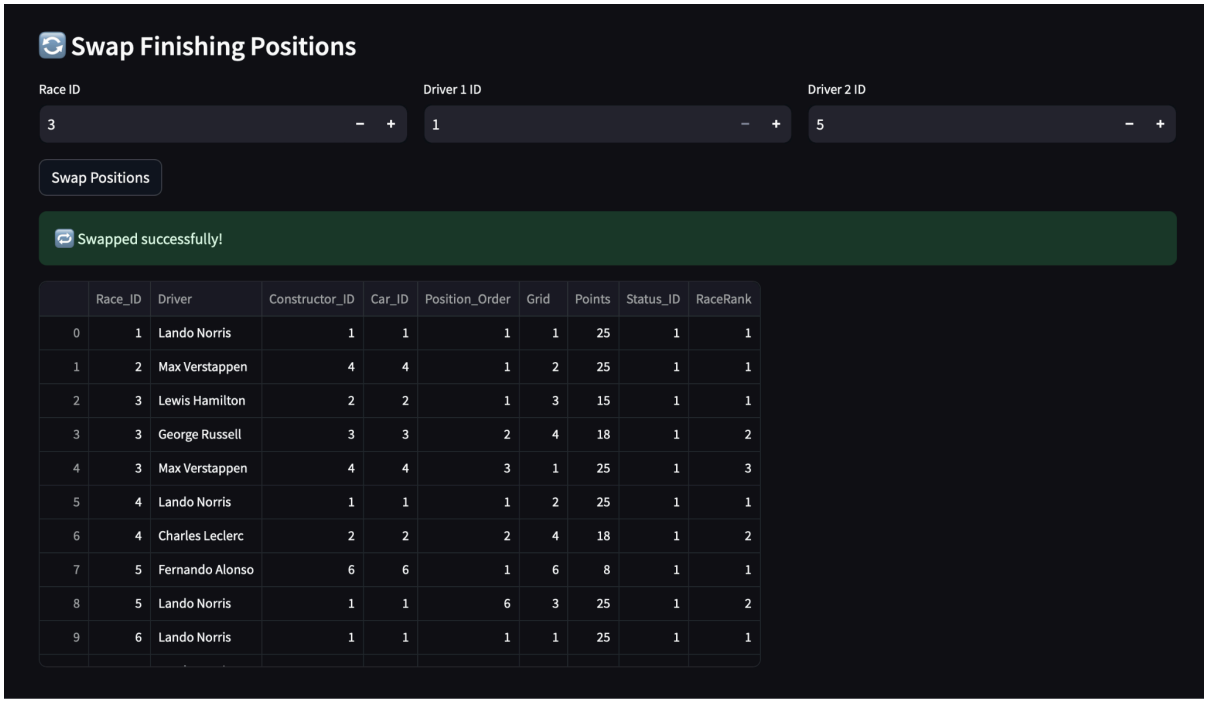
Create-



Read-

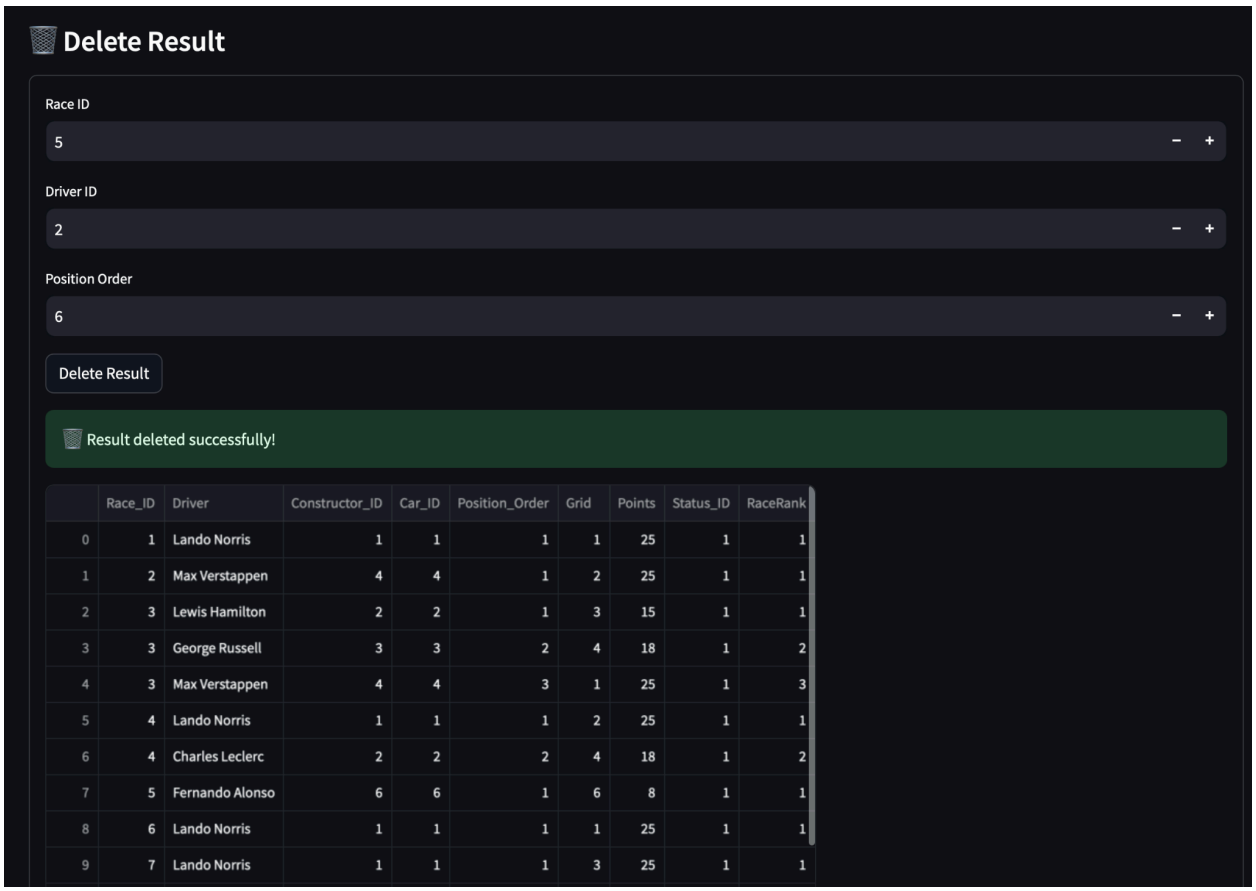


Update-



	Race_ID	Driver	Constructor_ID	Car_ID	Position_Order	Grid	Points	Status_ID	RaceRank
0	1	Lando Norris	1	1	1	1	25	1	1
1	2	Max Verstappen	4	4	1	2	25	1	1
2	3	Lewis Hamilton	2	2	1	3	15	1	1
3	3	George Russell	3	3	2	4	18	1	2
4	3	Max Verstappen	4	4	3	1	25	1	3
5	4	Lando Norris	1	1	1	2	25	1	1
6	4	Charles Leclerc	2	2	2	4	18	1	2
7	5	Fernando Alonso	6	6	1	6	8	1	1
8	5	Lando Norris	1	1	6	3	25	1	2
9	6	Lando Norris	1	1	1	1	25	1	1

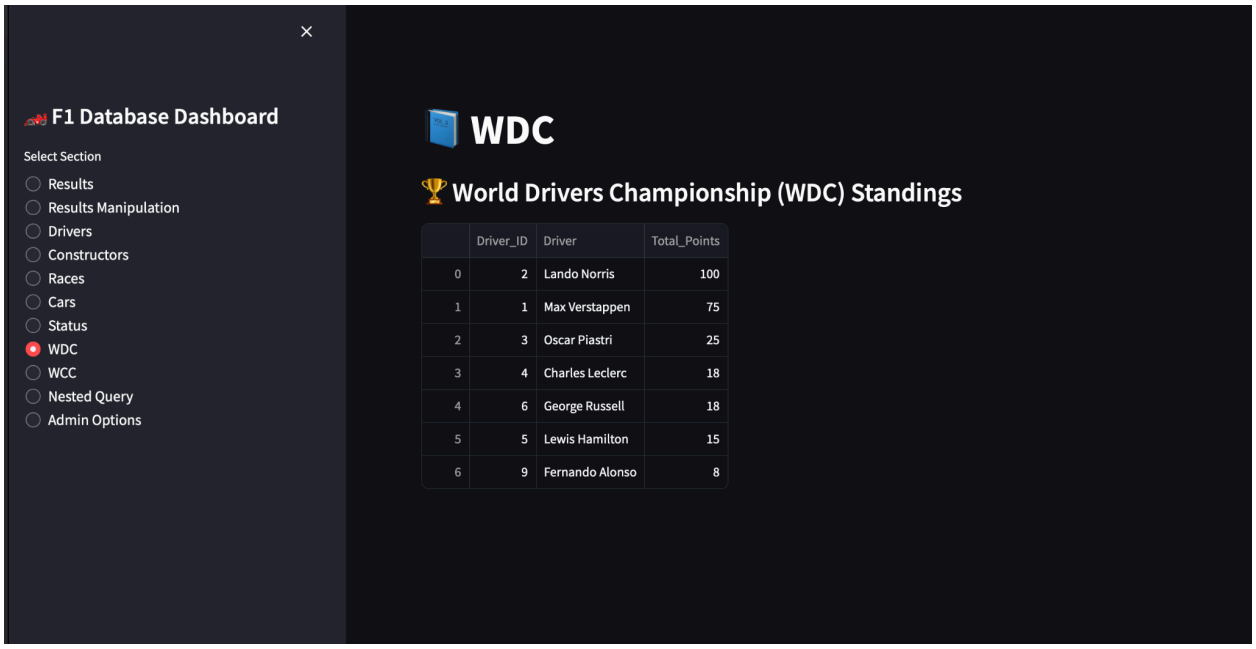
Delete-

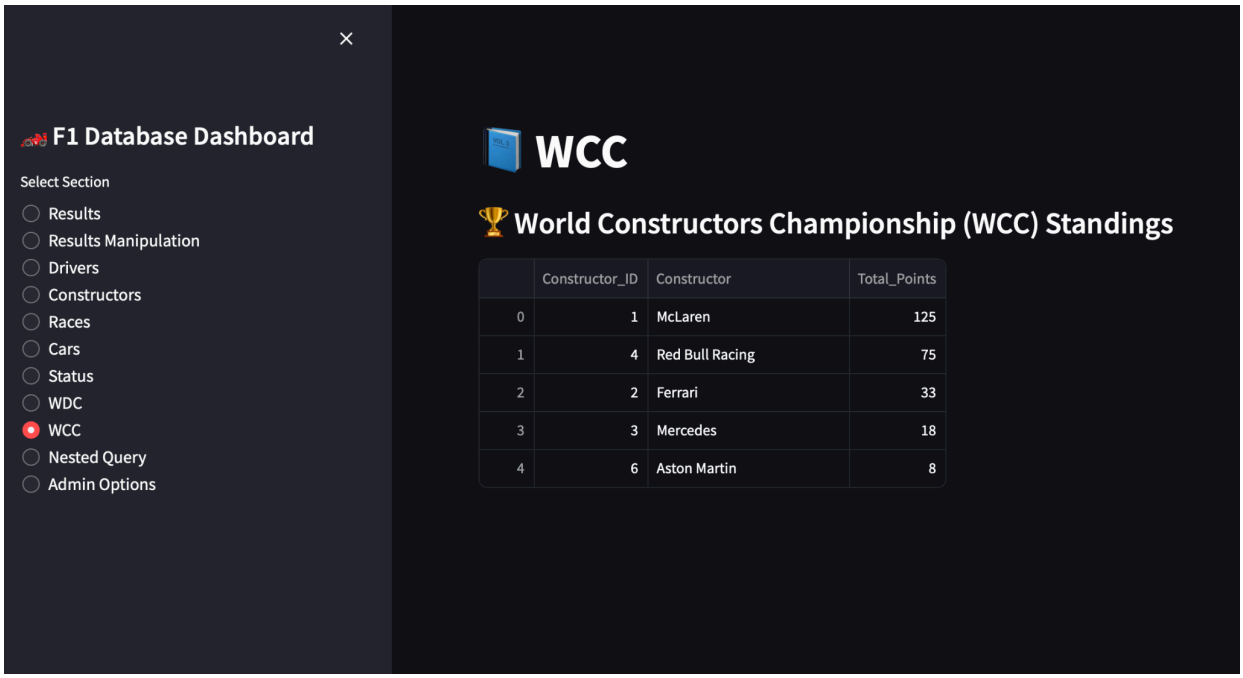


	Race_ID	Driver	Constructor_ID	Car_ID	Position_Order	Grid	Points	Status_ID	RaceRank
0	1	Lando Norris	1	1	1	1	25	1	1
1	2	Max Verstappen	4	4	1	2	25	1	1
2	3	Lewis Hamilton	2	2	1	3	15	1	1
3	3	George Russell	3	3	2	4	18	1	2
4	3	Max Verstappen	4	4	3	1	25	1	3
5	4	Lando Norris	1	1	1	2	25	1	1
6	4	Charles Leclerc	2	2	2	4	18	1	2
7	5	Fernando Alonso	6	6	1	6	8	1	1
8	6	Lando Norris	1	1	1	1	25	1	1
9	7	Lando Norris	1	1	1	3	25	1	1

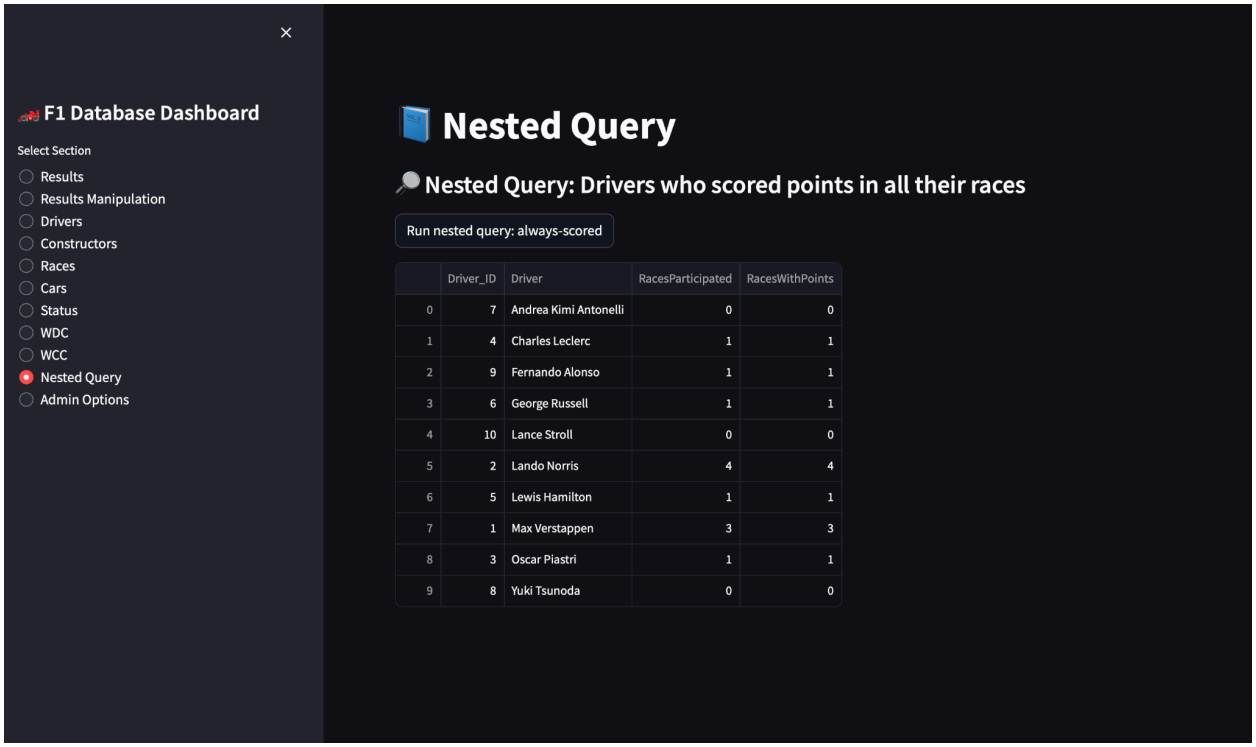
List of functionalities/features of the application and its associated screenshots using front end

The previous screen shots are also included

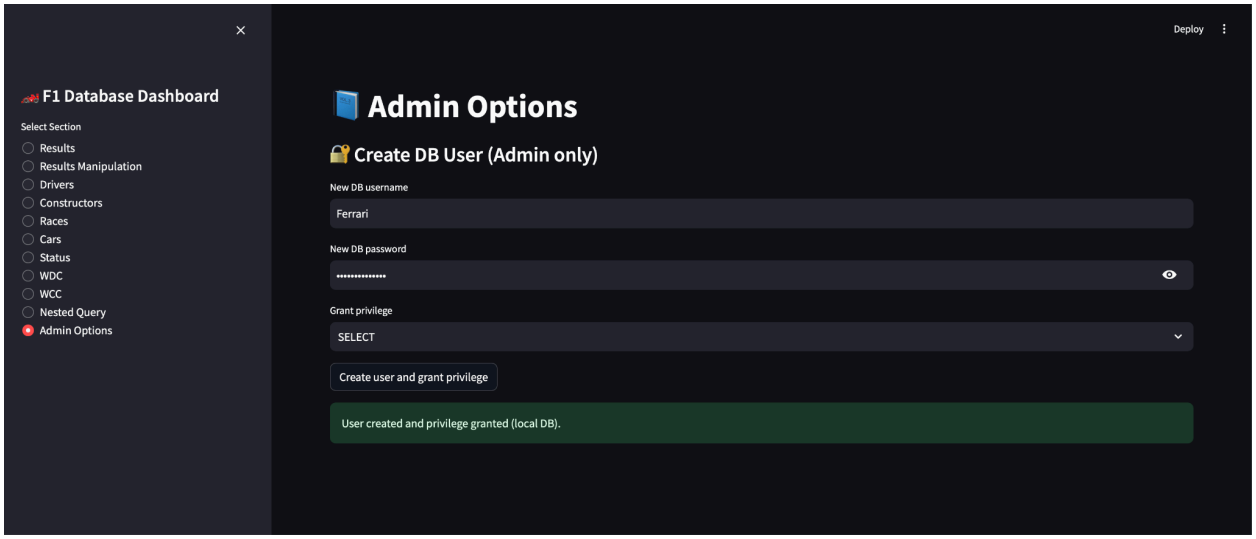




Nested query to get drivers who scored points-



User creation-



Triggers, Procedures/Functions, Nested query, Join, Aggregate queries

```
USE F1;

-- ========================
-- TRIGGERS (how to test)
-- ========================

-- 1) trg_results_before_insert
-- Purpose: If Points is NULL or 0, automatically compute Points from Position_Order.
-- Test: insert without Points or call add_result with Car_ID NULL (if allowed).
CALL add_result(1, 1, 1, NULL, 1, 1, 1); -- should create a row with Points = 25 (pos 1)

-- 2) PreventDuplicateResults
-- Purpose: Prevent inserting same (Race, Driver, Constructor, Position) twice.
-- Test: try inserting same result twice and see the SIGNAL error.
-- First insert (should succeed):
CALL add_result(1, 6, 5, 5, 12, 6, 1);
-- Second insert (should fail with the SIGNAL message):
-- CALL add_result(1, 6, 5, 5, 12, 6, 1);

-- ========================
-- PROCEDURES & FUNCTIONS
-- ========================

-- Function: getDriverFullName(dID)
SELECT getDriverFullName(1) AS DriverName; -- returns "Forename Surname"

-- Procedure: assign_points_for_race(rID)
-- Use when you changed Position_Order manually and want Points recalculated for race 1
CALL assign_points_for_race(1);

-- Procedure: RecalculateAllRaceRanks()
-- Recomputes RaceRank column for all races and returns Results
CALL RecalculateAllRaceRanks();

-- Procedure: add_result(...)
-- Example: add a result (Car_ID NULL allowed if your add_result allows it)
CALL add_result(1, 7, 6, NULL, 5, 7, 1); -- position 5 -> Points = 10 (trigger)
```

```
-- Procedure: swap_driver_positions(race, driverA, driverB)
-- Example: swap positions of driver 1 and driver 2 in race 1
CALL swap_driver_positions(1, 1, 2);

-- Procedure: delete_result(race, driver, position)
-- Example: delete a result (safely signals if not found)
CALL delete_result(1, 6, 12);

-- ========================
-- NESTED / SUBQUERIES
-- ========================

-- A) Non-correlated subquery (driver totals greater than average total)
SELECT Driver_ID, SUM(Points) AS total_points
FROM Results
GROUP BY Driver_ID
HAVING SUM(Points) > (
   SELECT AVG(driver_total) FROM (
      SELECT SUM(Points) AS driver_total
      FROM Results
      GROUP BY Driver_ID
   ) AS driver_totals_sub
);

-- B) Correlated subquery (races where winning constructor nationality equals circuit country)
SELECT r.Race_ID, r.Year, g.GP_Name, c.C_Name, c.Country
FROM Races r
JOIN GPs g ON r.GP_ID = g.GP_ID
JOIN Circuits c ON r.Circuit_ID = c.Circuit_ID
WHERE EXISTS (
   SELECT 1
   FROM Results res
   JOIN Constructors cons ON res.Constructor_ID = cons.Constructor_ID
   WHERE res.Race_ID = r.Race_ID
     AND res.Position_Order = 1
     AND cons.Nationality = c.Country
);

-- C) Using WITH (CTE) + nested aggregation: drivers with their season totals
WITH driver_season AS (
   SELECT d.Driver_ID, CONCAT(d.Forename,' ', d.Surname) AS DriverName, ra.Year,
SUM(res.Points) AS SeasonPoints
   FROM Drivers d
   JOIN Results res ON d.Driver_ID = res.Driver_ID
```

```
    JOIN Races ra ON res.Race_ID = ra.Race_ID
    GROUP BY d.Driver_ID, ra.Year
)
SELECT * FROM driver_season WHERE SeasonPoints >= (
    SELECT AVG(SeasonPoints) FROM driver_season
);

-- ========================
-- JOINS (useful reports)
-- ========================
-- 1) Driver standings for a particular season (example: 2023)
SELECT d.Driver_ID, CONCAT(d.Forename,' ',d.Surname) AS DriverName,
COALESCE(SUM(res.Points),0) AS SeasonPoints
FROM Drivers d
LEFT JOIN Results res ON d.Driver_ID = res.Driver_ID
LEFT JOIN Races ra ON res.Race_ID = ra.Race_ID
WHERE ra.Year = 2023
GROUP BY d.Driver_ID
ORDER BY SeasonPoints DESC;



-- ========================
-- AGGREGATE QUERIES (GROUP BY, HAVING, WINDOW)
-- ========================

-- A) Total points per constructor (all-time)
SELECT cons.Constructor_ID, cons.Con_Name, COALESCE(SUM(res.Points),0) AS
ConstructorPoints
FROM Constructors cons
LEFT JOIN Results res ON cons.Constructor_ID = res.Constructor_ID
GROUP BY cons.Constructor_ID
ORDER BY ConstructorPoints DESC;
```

Triggers, Procedures/Functions, Nested query, Join, Aggregate queries

Triggers-(Auto-assign Points Trigger)

```
Database changed
mysql> INSERT INTO Results (Race_ID, Driver_ID, Constructor_ID, Car_ID, Position_Order, Grid, Status_ID)
    -> VALUES (1, 3, 1, 1, 1, 5, 1);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM Results WHERE Race_ID = 1 AND Driver_ID = 3;
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
| Result_ID | Race_ID | Driver_ID | Constructor_ID | Car_ID | Position_Order | Grid | Points | Status_ID | RaceRank |
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
|        15 |       1 |         3 |              1 |      1 |              1 |    5 |     25 |         1 |     NULL |
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
1 row in set (0.00 sec)
```

Procedure-

```
mysql> SELECT getDriverFullName(1) AS DriverName;
+-----------------+
| DriverName      |
+-----------------+
| Max Verstappen  |
+-----------------+
1 row in set (0.01 sec)
```

```
mysql> CALL add_result(1, 4, 2, 2, 3, 3, 1);  -- New valid entry
Query OK, 2 rows affected (0.01 sec)

mysql> SELECT * FROM Results WHERE Race_ID = 1 ORDER BY Position_Order;
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
| Result_ID | Race_ID | Driver_ID | Constructor_ID | Car_ID | Position_Order | Grid | Points | Status_ID | RaceRank |
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
|         1 |       1 |         2 |              1 |      1 |              1 |    1 |     25 |         1 |        1 |
|        15 |       1 |         3 |              1 |      1 |              1 |    5 |     25 |         1 |        2 |
|        16 |       1 |         4 |              2 |      2 |              3 |    3 |     15 |         1 |        3 |
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
3 rows in set (0.00 sec)
```

```
mysql> UPDATE Results SET Points = NULL WHERE Race_ID = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> CALL assign_points_for_race(2);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT Race_ID, Driver_ID, Position_Order, Points FROM Results WHERE Race_ID = 2;
+---------+-----------+----------------+--------+
| Race_ID | Driver_ID | Position_Order | Points |
+---------+-----------+----------------+--------+
|       2 |         1 |              1 |     25 |
+---------+-----------+----------------+--------+
1 row in set (0.00 sec)
```

```
mysql> CALL RecalculateAllRaceRanks();
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
| Result_ID | Race_ID | Driver_ID | Constructor_ID | Car_ID | Position_Order | Grid | Points | Status_ID | RaceRank |
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
|         1 |       1 |         2 |              1 |      1 |              1 |    1 |     25 |         1 |        1 |
|        15 |       1 |         3 |              1 |      1 |              1 |    5 |     25 |         1 |        2 |
|        16 |       1 |         4 |              2 |      2 |              3 |    3 |     15 |         1 |        3 |
|         2 |       2 |         1 |              4 |      4 |              1 |    2 |     25 |         1 |        1 |
|        13 |       3 |         5 |              2 |      2 |              1 |    3 |     15 |         1 |        1 |
|        14 |       3 |         6 |              3 |      3 |              2 |    4 |     18 |         1 |        2 |
|         3 |       3 |         1 |              4 |      4 |              3 |    1 |     25 |         1 |        3 |
|        11 |       4 |         2 |              1 |      1 |              1 |    2 |     25 |         1 |        1 |
|         8 |       4 |         4 |              2 |      2 |              2 |    4 |     18 |         1 |        2 |
|         9 |       5 |         9 |              6 |      6 |              1 |    6 |      8 |         1 |        1 |
|         4 |       6 |         2 |              1 |      1 |              1 |    1 |     25 |         1 |        1 |
|         5 |       7 |         2 |              1 |      1 |              1 |    3 |     25 |         1 |        1 |
|         6 |       8 |         3 |              1 |      1 |              1 |    2 |     25 |         1 |        1 |
|         7 |       9 |         1 |              4 |      4 |              1 |    1 |     25 |         1 |        1 |
+-----------+---------+-----------+----------------+--------+----------------+------+--------+-----------+----------+
14 rows in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> SELECT Race_ID, Driver_ID, RaceRank FROM Results ORDER BY Race_ID, RaceRank;
+---------+-----------+----------+
| Race_ID | Driver_ID | RaceRank |
+---------+-----------+----------+
|       1 |         2 |        1 |
|       1 |         3 |        2 |
|       1 |         4 |        3 |
|       2 |         1 |        1 |
|       3 |         5 |        1 |
|       3 |         6 |        2 |
|       3 |         1 |        3 |
|       4 |         2 |        1 |
|       4 |         4 |        2 |
|       5 |         9 |        1 |
|       6 |         2 |        1 |
|       7 |         2 |        1 |
|       8 |         3 |        1 |
|       9 |         1 |        1 |
+---------+-----------+----------+
14 rows in set (0.00 sec)
```

Aggregate queries-(Total points per driver)

```
mysql> SELECT d.Forename, d.Surname, SUM(r.Points) AS TotalPoints
    -> FROM Results r
    -> JOIN Drivers d ON r.Driver_ID = d.Driver_ID
    -> GROUP BY d.Driver_ID
    -> ORDER BY TotalPoints DESC;
+----------+-----------+-------------+
| Forename | Surname   | TotalPoints |
+----------+-----------+-------------+
| Lando    | Norris    |         100 |
| Max      | Verstappen|          75 |
| Oscar    | Piastri   |          50 |
| Charles  | Leclerc   |          33 |
| George   | Russell   |          18 |
| Lewis    | Hamilton  |          15 |
| Fernando | Alonso    |           8 |
+----------+-----------+-------------+
7 rows in set (0.00 sec)

mysql> SELECT c.Con_Name, AVG(r.Grid) AS AvgGrid
    -> FROM Results r
    -> JOIN Constructors c ON r.Constructor_ID = c.Constructor_ID
    -> GROUP BY c.Constructor_ID
    -> ORDER BY AvgGrid;
+----------------+---------+
| Con_Name       | AvgGrid |
+----------------+---------+
| Red Bull Racing |  1.3333 |
| McLaren        |  2.3333 |
| Ferrari        |  3.3333 |
| Mercedes       |  4.0000 |
| Aston Martin   |  6.0000 |
+----------------+---------+
5 rows in set (0.00 sec)
```

Join queries-

```
mysql> SELECT d.Forename, d.Surname, g.GP_Name, c.C_Name, r.Points
    -> FROM Results r
    -> JOIN Drivers d ON r.Driver_ID = d.Driver_ID
    -> JOIN Races ra ON r.Race_ID = ra.Race_ID
    -> JOIN GPs g ON ra.GP_ID = g.GP_ID
    -> JOIN Circuits c ON ra.Circuit_ID = c.Circuit_ID;
+----------+------------+-------------------------+-------------------------------+--------+
| Forename | Surname    | GP_Name                 | C_Name                        | Points |
+----------+------------+-------------------------+-------------------------------+--------+
| Lando    | Norris     | Australian Grand Prix   | Albert Park Circuit           |     25 |
| Oscar    | Piastri    | Australian Grand Prix   | Albert Park Circuit           |     25 |
| Charles  | Leclerc    | Australian Grand Prix   | Albert Park Circuit           |     15 |
| Max      | Verstappen | Chinese Grand Prix      | Shanghai International Circuit |     25 |
| Max      | Verstappen | Japanese Grand Prix     | Suzuka Circuit                |     25 |
| Lewis    | Hamilton   | Japanese Grand Prix     | Suzuka Circuit                |     15 |
| George   | Russell    | Japanese Grand Prix     | Suzuka Circuit                |     18 |
| Charles  | Leclerc    | Bahrain Grand Prix      | Bahrain International Circuit |     18 |
| Lando    | Norris     | Bahrain Grand Prix      | Bahrain International Circuit |     25 |
| Fernando | Alonso     | Saudi Arabian Grand Prix | Jeddah Corniche Circuit       |      8 |
| Lando    | Norris     | Monaco Grand Prix       | Circuit de Monaco             |     25 |
| Lando    | Norris     | British Grand Prix      | Silverstone Circuit           |     25 |
| Oscar    | Piastri    | Belgian Grand Prix      | Circuit de Spa-Francorchamps  |     25 |
| Max      | Verstappen | Italian Grand Prix      | Autodromo Nazionale Monza     |     25 |
+----------+------------+-------------------------+-------------------------------+--------+
14 rows in set (0.00 sec)
```

Nested/subqueries-

```
mysql> SELECT c.Con_Name, AVG(r.Points) AS AvgPoints
    -> FROM Constructors c
    -> JOIN Results r ON c.Constructor_ID = r.Constructor_ID
    -> GROUP BY c.Constructor_ID
    -> HAVING AvgPoints = (
    ->     SELECT MAX(avg_pts)
    ->     FROM (
    ->         SELECT AVG(Points) AS avg_pts
    ->         FROM Results
    ->         GROUP BY Constructor_ID
    ->     ) sub
    -> );
+-----------------+-----------+
| Con_Name        | AvgPoints |
+-----------------+-----------+
| McLaren         |   25.0000 |
| Red Bull Racing |   25.0000 |
+-----------------+-----------+
2 rows in set (0.01 sec)
```