F1 TEAM PERFORMANCE TRACKER

PROJECT REPORT

Group 14

Student 1: DENNIS MATHEW JOSE

Student 2 : ARUN SOLAIAPPAN VALLIAPPAN

+1(862) 440-7362 (DENNIS MATHEW JOSE)

+1(617) 953-5343 (ARUN SOLAIAPPAN VALLIAPPAN)

jose.de@northeastern.edu

valliappan.a@northeastern.edu

Percentage of effort contributed by student 1: 50%

Percentage of effort contributed by student 2: 50%

Signature of Student 1: DENNIS JOSE

Signature of Student 2: ARUN SOLAIAPPAN VALLIAPPAN

Submission Date:12/08/24

USE CASE STUDY REPORT

Group No: Group 14

Student Names: Dennis Mathew Jose & Arun Solaiappan Valliappan

Executive Summary:

The F1 Team Performance Tracker is a comprehensive database management system (DBMS) designed to manage, track, and analyze the vast amount of performance data in Formula 1. This project aims to provide a scalable and efficient solution for storing and retrieving data related to teams, drivers, circuits, races, seasons, and standings, facilitating real-time performance monitoring and historical analysis.

The system will enable users to track driver and team results, analyze trends over multiple seasons, and access detailed race metrics such as lap times, finishing positions, and earned points. Leveraging relational database design principles, the project focuses on creating a structured and normalized data model that minimizes redundancy and supports advanced querying capabilities. This tool will be invaluable to F1 analysts, teams, and fans for conducting in-depth performance reviews and uncovering trends in race outcomes.

Key points include a relational database schema, data population and management workflows, efficient SQL queries, and trend analysis capabilities, ensuring the system remains scalable and high-performing as new seasons and races are added.

Introduction:

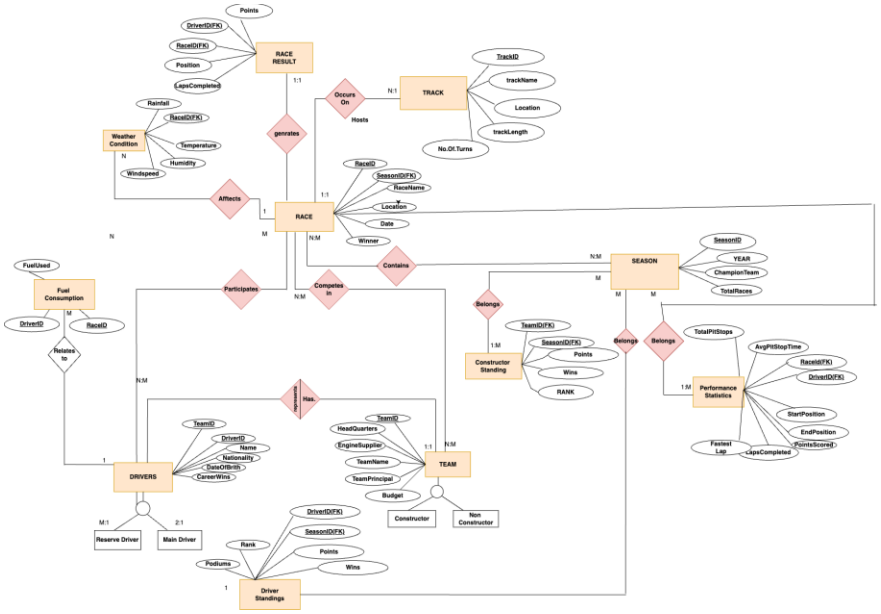
Formula 1 (F1) is one of the most popular and competitive motorsports globally, attracting millions of fans and intense scrutiny of performance metrics throughout the season. Tracking the performance of teams, drivers, and circuits requires a robust system capable of managing and analyzing complex datasets.

The F1 Team Performance Tracker project addresses this need by proposing a relational database management system (RDBMS) to organize and analyze F1-related data. The system focuses on creating a structured and efficient database for storing comprehensive information about teams, drivers, circuits, races, seasons, and standings. It supports real-time performance monitoring, historical data analysis, and trend identification, providing a valuable resource for F1 analysts, teams, and enthusiasts.

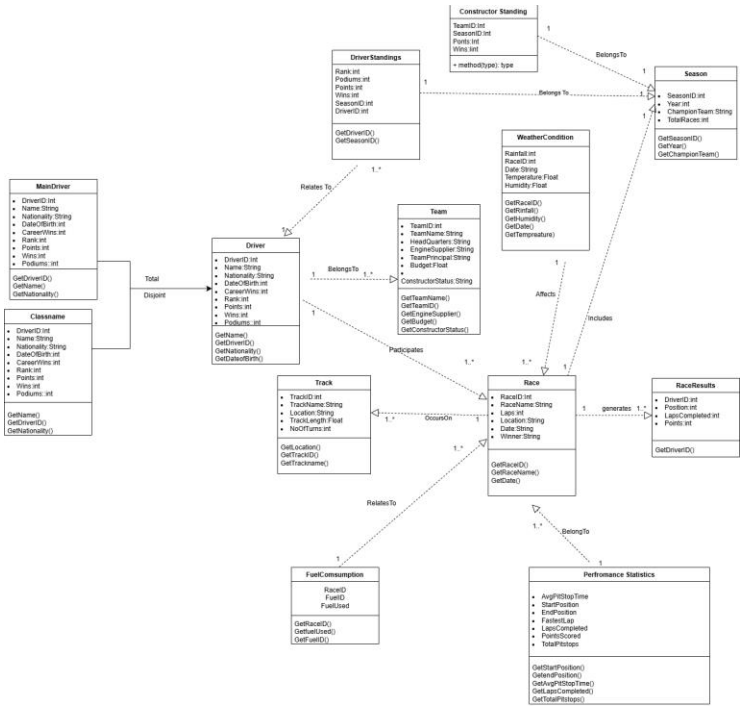
By designing a relational database with scalable querying capabilities and optional visualization integration, this project aims to deliver a powerful tool for data-driven insights into Formula 1 performance trends and outcomes. The system's design ensures efficient data storage, consistency, and scalability, making it an essential resource for understanding and analyzing the dynamic world of Formula 1.

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Relational Model – Primary keys are underlined, and foreign keys are in italics.

Team (TeamID, TeamName, HeadQuarters, TeamPrincipal, EngineSupplier, Budget)

Constructor Team(*C_TeamId*, TeamName)

C_teamId, is a foreign key which points to the TeamID in the realtion Team; NULL NOT ALLOWED.

Non Constructor Team (NC_TeamID, TeamName)

NC_TeamID is a foreign key which points to the teamID in the relation Team; NULL NOT ALLOWED

Driver (DriverID, DriverName, DateOfBirth, Nationality,

CareerWins/WorldChampionships, Podiums, Status)

TeamID is a foreign key which points to the teamID of relation team. Each driver should belong to a team; NULL NOT ALLOWED.

ReserveDriver(*R_DriverID*, DriverName)

R_DriverID is a foreign key which is pointing to the DriverID in relation Driver; NULL ALLOWED; A team might not have reserve driver

TeamID is a foreign key which is pointing to TeamID in relation Team; NULL NOT ALLOWED; A driver should always be mapped to team

MainDriver (*M_DriverID*, DriverName)

TeamID is a foreign key which is pointing to TeamID in relation Team; NULL NOT ALLOWED; A driver should always be mapped to team

M_DriverID is foreign key which is pointing to DriverID of relation Driver; NULL NOT ALLOWED because every team should atleast 1 main driver and atmost 2 main driver

Drives(DriverID, TeamID, From, To)

R_DriverID is a foreign key which is pointing to the DriverID in relation Driver;

TeamID is a foreign key which is pointing to TeamID in relation Team; NULL NOT ALLOWED; A driver should always be mapped to team

RACE (RaceID, *SeasonID,* RaceName, *TrackID*, Location, Date, *Winner*)

SeasonID is a foreign key that points to SeasonID of relation Season; NULL NOT ALLOWED

TrackID is a foreign key that points to trackID of relation Track; NULL NOT ALLOWED since all race happens on a track

Winner is a foreign key which points to driverID of relation Driver; NULL NOT ALLOWED, since every race has winner

Season (SeasonId, year, championship team, totalRace)

belongs(*TeamId,seasonid*)((season-->constructor standings)

Seasonid sis a foreign key which points to seasonid of the relation season; NULL NOT ALLOWED

Teamid is a foreign key which points to teamid of the relation team; NULL NOT ALLOWED

RaceData (*raceid, seasonid, trackId*)((Race-->track)

RaceId is a foreign key which points to RaceId of the relation race; NULL NOT ALLOWED

TeamId is a foreign key which points to TeamId of the relation team; NULL NOT ALLOWED

RaceResult **(***RaceID*, *DriverID*, EndPosition, LapsCompleted, Points)

RaceID is a foreign key which points to RaceID of relation Race; NULL NOT ALLOWED

DriverID is a foreign key where the foregin key points to DriverID of relation Driver; NULL ALLOWED; A driver might not participate in a race

Track (*Trackid*, *RaceId*, *Seasonid* Trackname, location, track length, noOfTurns,)

RaceId is a foreign key which points to RaceId of the relation Race; NULL NOT ALLOWED

Seasonid is a foreign key which points to seasonid of the relation season; NULL NOT ALLOWED

DriverStandings (*DriverID*, *SeasonID*, Points, Rank, Wins, Podiums)

DriverID is a foreign key which points to DriverID of relation Driver; NULL NOT ALLOWED

SeasonID is a foreign key which points to SeasonID of relation Season; NULL NOT ALLOWED

PerformanceStatistics (*DriverID*, *TeamID*, *RaceID*, StartPositon, EndPosition, FastestLap, LapsCompleted TotalPitStops, AvgPitStopTime, PointsScored)

DriverId is a foreign key which points to driverID of the relation Driver; NULL NOT ALLOWED

TeamID is a foreign key which points to teamID of relation Team; NULL NOT ALLOWED

RaceID is a foreign key which points to RaceID of relation Race; NULL NOT ALLOWED

FuelConsumption (Fuelused, *driverId*, *RaceId*)

DriverId is a foreign key which points to driverID of the relation Driver; NULL NOT ALLOWED

RaceIdis a foreign key which points to RaceId of the relation Race; NULL NOT ALLOWED

WeatherCondition (*RaceId* , Date, Rainfall, temprature, humidity, windspeed)

RaceId is a foreign key which points to RaceId of the relation Race; NULL NOT ALLOWED

RaceId is a foreign key which points to RaceId of the relation Race; NULL NOT ALLOWED

ConstructorStandings (*TeamId*, *seasonId*, points, ranks, wins)

Seasonid is a foreign key which points to seasonid of the relation season; NULL NOT ALLOWED

Teamid is a foreign key which points to teamid of the relation team; NULL NOT ALLOWED

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL and the following queries were performed:

1) Simple Query - Find the top 5 drivers with the most podiums

```
SELECT DriverId, Name AS DriverName,Podiums
FROM Driver
ORDER BY Podiums DESC;
LIMIT 5;
```

DriverId	DriverName	Podiums
1	Lewis Hamilton	195
11	Sebastian Vettel	122
19	Fernando Alonso	101
5	Max Verstappen	95
2	Valtteri Bottas	67
NULL	NULL	NULL

2) Aggregate Query –

Find the sum of all points scored by top 10 drivers from season 2019 to 2023

```
SELECT   P.DriverId, D.Name,  sum(P.PointsScored)  as TotalPoints
FROM    PerformanceStatistics  P, Driver D
WHERE  D.DriverId = P.DriverId
GROUP BY          p.DriverID;
```

DriverId	Name	TotalPoints
1	Lewis Hamilton	159
2	Valtteri Bottas	117
3	George Russell	137
5	Max Verstappen	91
6	Sergio Perez	67
7	Alexander Albon	87
9	Charles Leclerc	64
10	Carlos Sainz	70
13	Lando Norris	73
14	Daniel Ricciardo	83

3) INNER/OUTER JOIN - Find all drivers and the teams they drove for, including drivers who do not currently belong to any team.

```
SELECT d.DriverId, d.Name  AS DriverName,  t.TeamName,  dr.FromYear,   dr.ToYear
FROM Driver d
LEFT JOIN Drives dr ON d.DriverId = dr.DriverId
LEFT JOIN Team  t ON dr.TeamID  = t.TeamId
ORDER BY d.Name;
```

DriverId	Name	TotalPoints
1	Lewis Hamilton	159
2	Valtteri Bottas	117
3	George Russell	137
5	Max Verstappen	91
6	Sergio Perez	67
7	Alexander Albon	87
9	Charles Leclerc	64
10	Carlos Sainz	70
13	Lando Norris	73
14	Daniel Ricciardo	83

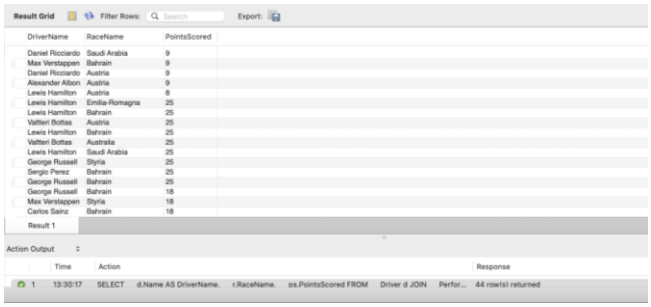
4) Nested Query - Find the name of the driver who has the highest number of wins in a season.

```
SELECT d.Name  AS DriverName
 FROM Driver d WHERE d.DriverId = (SELECT ds.Driver FROM DriverStandings d WHERE
        ds.Wins = (SELECT MAX(Wins) FROM DriverStandings)
     LIMIT 1);
```

DriverName
Max Verstappen

5) Correlated Query - Find all drivers who scored more points in a race than the average points scored in that race by all drivers.

```
SELECT d.Name AS DriverName, r.RaceName, ps.PointsScored FROM Driver d
JOIN PerformanceStatistics ps ON d.DriverId = ps.DriverId JOIN Race r ON ps.RaceId = r.RaceId
WHERE ps.PointsScored > (SELECT AVG(ps2.PointsScored) FROM PerformanceStatistics ps2 WHERE ps2.RaceId = ps.RaceId) ORDER BY ps.PointsScored DESC;
```



DriverName	RaceName	PointsScored
Daniel Ricciardo	Saudi Arabia	9
Max Verstappen	Bahrain	9
Daniel Ricciardo	Austria	9
Alexander Albon	Austria	9
Lewis Hamilton	Austria	8
Lewis Hamilton	Emilia-Romagna	25
Lewis Hamilton	Bahrain	25
Valtteri Bottas	Austria	25
Lewis Hamilton	Bahrain	25
Valtteri Bottas	Australia	25
Lewis Hamilton	Saudi Arabia	25
George Russell	Styria	25
Sergio Perez	Bahrain	25
George Russell	Bahrain	25
George Russell	Bahrain	18
Max Verstappen	Styria	18
Carlos Sainz	Bahrain	18

6) Query with >=ALL/>ANY/EXISTS/NOT EXISTS - Find drivers who have completed all races in the 2023 season.

```
SELECT d.DriverId, d.Name AS DriverName FROM Driver d
WHERE NOT EXISTS (SELECT 1 FROM Race rWHERE r.SeasonId = ( SELECT SeasonId FROM Season WHERE Year = '2023-01-01') AND r.RaceId NOT IN ( SELECT ps.RaceId FROM PerformanceStatistics ps WHERE ps.DriverId = d.DriverId )
);
```

DriverId	DriverName
1	Lewis Hamilton
2	Valtteri Bottas
3	George Russell
5	Max Verstappen
6	Sergio Perez
7	Alexander Albon
9	Charles Leclerc
10	Carlos Sainz
13	Lando Norris
14	Daniel Ricciardo
NULL	NULL

7) SET operations (UNION) - Get all unique driver names, including both main and reserve drivers

```
SELECT DriverName FROM MainDriver UNION SELECT DriverName FROM ReserveDriver;
```

DriverName
Lewis Hamilton
Valtteri Bottas
George Russell
Max Verstappen
Sergio Perez
Alexander Albon
Liam Lawson
Charles Leclerc
Carlos Sainz
Lando Norris
Daniel Ricciardo
Oscar Piastri
Esteban Ocon
Pierre Gasly
Fernando Alonso
Jack Doohan

Result 27

8) Sub queries in SELECT and FROM - Find the total number of laps completed by each driver in the 2022 season.

SELECT d.DriverId, d.Name AS DriverName, (SELECT SUM(ps.LapsCompleted) FROM PerformanceStatistics ps JOIN Race r ON ps.RaceId = r.RaceId WHERE ps.DriverId = d.Driver AND r.SeasonId = (SELECT SeasonId FROM Season WHERE Year = '2022-01-01')) AS TotalLapsCompleted FROM Driver ORDER BY TotalLapsCompleted DESC;

DriverId	DriverName
1	Lewis Hamilton
2	Valtteri Bottas
3	George Russell
5	Max Verstappen
6	Sergio Perez
7	Alexander Albon
9	Charles Leclerc
10	Carlos Sainz
13	Lando Norris
14	Daniel Ricciardo
NULL	NULL

NoSQL Implementation;

Query 1: Find a driver by name

db.Driver.find({"Name":"Lewis Hamilton"});

```
> db.Driver.find({"Name":"Lewis Hamilton"});
< {
    _id: ObjectId('674d0837e8daaaea9de381d9'),
    DriverId: 1,
    Name: 'Lewis Hamilton',
    DateOfBirth: '1985-01-07 00:00:00',
    Nationality: 'British',
    Championships: 7,
    Podiums: 195,
    Status: 'Active'
  }
```

Query 2: Find active drivers with more than 5 championships

db.Driver.find({ "Championships": { $gt: 5 },"Status": "Active"})

```
< {
    _id: ObjectId('674d0837e8daaaea9de381d9'),
    DriverId: 1,
    Name: 'Lewis Hamilton',
    DateOfBirth: '1985-01-07 00:00:00',
    Nationality: 'British',
    Championships: 7,
    Podiums: 195,
    Status: 'Active'
  }
```

Query 3: Aggregate total championships by nationality

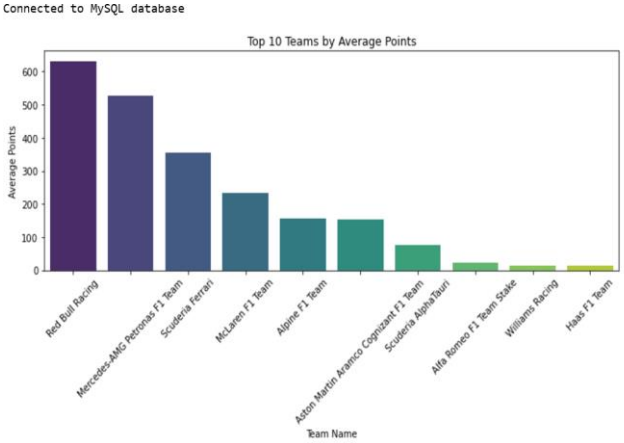
db.Driver.aggregate([

{$group: {_id: "$Nationality", totalChampionships: { $sum: "$Championships" }}},{ $sort: { totalChampionships: -1 }}])

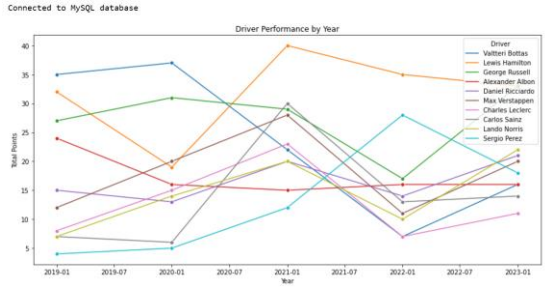
V. Database Access via Python

The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.excecute to run and fetchall from query, followed by converting the list into a dataframe using pandas library and using matplotlib to plot the graphs for the analytics.

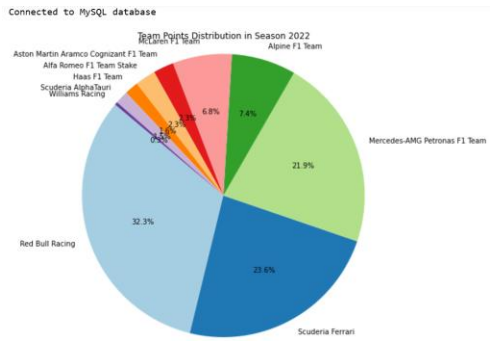
Graph 1: Top 10 Teams by Average Points



Graph2: Driver Performance by Year



Graph 3: Team Points Distribution in Season



VII. Summary and Recommendation:

The Formula 1 (F1) Team Performance Tracker project aims to build a comprehensive and scalable database management system (DBMS) to efficiently store, track, and analyze Formula 1 data, including teams, drivers, circuits, races, seasons, and standings. This project uses a combination of relational databases (SQL), NoSQL databases (MongoDB), and Python-based analytics to achieve its objectives. It targets users such as analysts, fans, and teams who seek insights into driver performance, race trends, and team standings.

The Entity-Relationship (ER) Diagram represents the logical structure of the database, showcasing entities like Driver, Team, Race, Track, and Season, along with their relationships. It ensures normalization by reducing redundancy while enabling complex queries, such as tracking driver performance across seasons or analyzing team standings. The UML Diagram captures the system's detailed design with attributes and methods for each entity. It highlights interactions, such as how races impact standings and weather conditions influence race outcomes and scalability of the system.

The project also features SQL queries that handle structured data effectively. Like finding top-performing drivers, aggregating total points or laps, and executing advanced joins and subqueries to fetch insights like drivers outperforming race averages or competing without team affiliations. Meanwhile, MongoDB provides queries aggregating championships by nationality or identifying active drivers with specific achievements.

Python plays a crucial role in the project, enabling data visualization, analytics, and integration with SQL and NoSQL databases. Python generates insights such as trend analyses, performance comparisons, and real-time statistics while using libraries like pandas for data manipulation, matplotlib for visualization, and Mysql connector for database connectivity.

To further enhance the project, Optimizing database design by indexing frequently queried fields and normalization will improve performance and scalability. Using Python, machine learning models can be introduced to predict race outcomes or trends based on historical and real-time data.

The system can be further enhanced by integrating APIs like Ergast Developer API to populate the database with real-time and historical F1 data. Building a REST API layer will enable external applications to interact with the system, adding flexibility. Deploying the system on cloud platforms like AWS, Azure ensures availability, scalability, and secure storage. Managed database services such as Amazon RDS for SQL and MongoDB Atlas for NoSQL can simplify deployment and management. Additionally, implementing robust security measures, including authentication, access control, and automated backups, will ensure data safety and recovery.