جامعة أم القرى
UMM AL-QURA UNIVERSITY

# Data Engineering Project

Renad Alhasani - 444007199 - G1
Athari Almuqati - 444002521 - G1
Manar Alqurashi - 444005895 - G2
Lamyaa Almutairi - 444005917 - G2

# Design overview

This project demonstrates an end-to-end data engineering pipeline using a Formula 1 race results dataset. The goal is to ingest, store, query, process, and visualize racing data from multiple angles using different storage and processing technologies.

## Dataset Description:

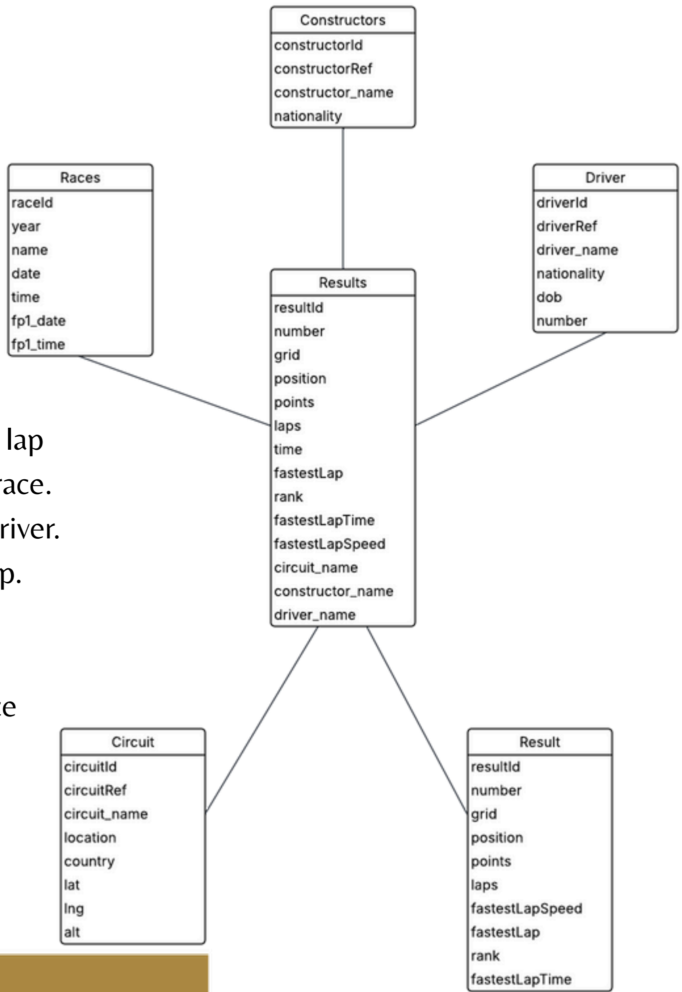
The dataset used in this project is Formula 1 racing data, originally spread across five separate files containing details about drivers, constructors, circuits, and race results. To simplify processing and analysis, we merged these files into one unified dataset by combining important columns like driver_name, constructor_name, and circuit_name. This combined dataset contains 26,760 rows and 15 columns, with each row representing a driver's performance in a specific race.

## Denormalization:

We decided to use denormalization because it was more useful for our project than normalization. Instead of keeping many separate tables, we combined all the important data like driver names, constructors, and circuit details into one big table called Results. This made it easier and faster to get the information we needed without doing many complicated joins between tables, which made working with the data more quickly and simply.

## Column Descriptions:

resultId: A unique identifier for each race result record
number: The racing number of the driver
grid: The position the driver started the race from
position: The final position the driver finished in the race
points: Points awarded to the driver for the race
laps: Number of laps completed by the driver in the race.
time: Total race time recorded for the driver
fastestLap: The lap number where driver recorded their fastest lap
rank: Rank of the fastest lap compared to other drivers in the race.
fastestLapTime: The actual fastest lap time recorded for that driver.
fastestLapSpeed: Speed (in km/h or mph) during the fastest lap.
driver_name: The full name of the driver
constructor_name: The name of the team the driver raced for
circuit_name: The name of the circuit where the race took place



UMM AL-QURA UNIVERSITY
جامعة أم القرى

# Design overview

## Phase One:

First we designed a denormalized schema by using SQLite for implementation. We created the Results table by joining relevant attributes from drivers, constructors, and circuits, perform CRUD operations, and applied indexing and basic optimization (INDEX ON driver_name) to improve query speed.

### example of CRUD operations

```
conn = sqlite3.connect("F1.db")
cursor = conn.cursor()


cursor.execute("""
SELECT driver_name, SUM(points) AS total_points
FROM Results
GROUP BY driver_name
ORDER BY total_points DESC
LIMIT 5
""")

top_drivers = cursor.fetchall()

print("Top 5 Drivers by Total Points:")
for driver in top_drivers:
    print(f"Driver Name: {driver[0]}, Total Points: {driver[1]}"

Top 5 Drivers by Total Points:
Driver Name: Lewis Hamilton, Total Points: 4845.5
Driver Name: Sebastian Vettel, Total Points: 3098.0
Driver Name: Max Verstappen, Total Points: 2912.5
Driver Name: Fernando Alonso, Total Points: 2329.0
Driver Name: Kimi Räikkönen, Total Points: 1873.0
```

### indexing and query optimization

```
query = '''
SELECT driver_name, COUNT(*) AS races_participated
FROM Results
GROUP BY driver_name
ORDER BY races_participated DESC
LIMIT 10
'''

df = pd.read_sql_query(query, conn)
df
```

| | driver_name | races_participated |
|---|---|---|
| 0 | Fernando Alonso | 404 |
| 1 | Lewis Hamilton | 357 |
| 2 | Kimi Räikkönen | 352 |
| 3 | Rubens Barrichello | 326 |
| 4 | Jenson Button | 309 |
| 5 | Michael Schumacher | 308 |
| 6 | Sebastian Vettel | 300 |
| 7 | Sergio Pérez | 283 |
| 8 | Felipe Massa | 271 |
| 9 | Riccardo Patrese | 257 |

```
[ ] start = time.time()
    df = pd.read_sql_query(query, conn)
    print("Time without index:", time.time() - start)

    cursor.execute("CREATE INDEX IF NOT EXISTS idx_driver_name ON Results(driver_name)")

    start = time.time()
    df = pd.read_sql_query(query, conn)
    print("Time with index:", time.time() - start)

Time without index: 0.028913497924804688
Time with index: 0.003632783889770508
```

## Phase Two:

In this phase, we explored NoSQL data handling by storing and interacting with a subset of the Formula 1 dataset in a document-based format using TinyDB. The dataset was transformed into a JSON-like structure and inserted into the database.

We carried out all the fundamental NoSQL tasks: inserting new records, query optimization, aggregating data to highlight top performers, updating selected fields, and deleting individual records

```
from tinydb import TinyDB, Query

from google.colab import files
uploaded = files.upload()

F1_results.csv
• F1_results.csv(text/csv) - 2299826 bytes, last modified: 12100%
Saving F1_results.csv to F1_results.csv

[4] import pandas as pd

    df = pd.read_csv('F1_results.csv')
    subset_df = df.head(1000)

[5] data = subset_df.to_dict(orient='records')

from tinydb import TinyDB
db = TinyDB('f1_results.json')


db.insert_multiple(data)

print("Subset inserted into TinyDB")

Subset inserted into TinyDB
```

```
Result = Query()

drivers = db.search(Result.constructor_name == 'McLaren'

unique_driver_names = set()

for r in drivers:
    unique_driver_names.add(r['driver_name'])
print("Drivers from McLaren:\n")
for name in sorted(unique_driver_names):
    print(name)

Drivers from McLaren:

Fernando Alonso
Heikki Kovalainen
Juan Pablo Montoya
Kimi Räikkönen
Lewis Hamilton
Pedro de la Rosa
```

```
from collections import Counter

winners = [r['driver_name'] for r in db.search(Result.
position == '1')]
top_winners = Counter(winners).most_common(5)

print("Top 5 Drivers with Most Wins:\n")
for name, wins in top_winners:
    print(f"{name}: {wins} wins")

Top 5 Drivers with Most Wins:

Fernando Alonso: 36 wins
Lewis Hamilton: 27 wins
Felipe Massa: 27 wins
Kimi Räikkönen: 24 wins
Michael Schumacher: 15 wins
```

# Design overview

## Phase Three:

We simulated a live data stream by reading the f1_subset_results.csv file, which was a result of phase2 and contained preprocessed and merged race data from various sources, including driver, constructor, and circuit information.

Using PySpark, we applied filtering operations to retain only those records where drivers scored more than 10 or 18 points, allowing us to identify high performing participants across different races.

```
+--------+------+----+--------+------+----+-----------+-----------+----+---------------+---------------+----------------+----------------+-------------------+
|resultId|number|grid|position|points|laps|       time|fastestLap|rank|fastestLapTime|fastestLapSpeed|      driver_name|constructor_name|        circuit_name|
+--------+------+----+--------+------+----+-----------+-----------+----+---------------+---------------+----------------+----------------+-------------------+
|       1|    22|   1|       1|  10.0|  58|1:34:50.616|         39|   2|       1:27.452|        218.300|   Lewis Hamilton|         McLaren|Albert Park Grand...|
|       2|     3|   5|       2|   8.0|  58|     +5.478|         41|   3|       1:27.739|        217.586|    Nick Heidfeld|      BMW Sauber|Albert Park Grand...|
|       3|     7|   7|       3|   6.0|  58|     +8.163|         41|   5|       1:28.090|        216.719|     Nico Rosberg|        Williams|Albert Park Grand...|
|       5|    23|   3|       5|   4.0|  58|    +18.014|         43|   1|       1:27.418|        218.385|Heikki Kovalainen|         McLaren|Albert Park Grand...|
|       6|     8|  13|       6|   3.0|  57|         \N|         50|  14|       1:29.639|        212.974|  Kazuki Nakajima|        Williams|Albert Park Grand...|
+--------+------+----+--------+------+----+-----------+-----------+----+---------------+---------------+----------------+----------------+-------------------+
only showing top 5 rows

+--------+------+----+--------+------+----+-----------+-----------+----+---------------+---------------+--------------+----------------+-----------------+
|resultId|number|grid|position|points|laps|       time|fastestLap|rank|fastestLapTime|fastestLapSpeed|   driver_name|constructor_name|     circuit_name|
+--------+------+----+--------+------+----+-----------+-----------+----+---------------+---------------+--------------+----------------+-----------------+
|    1001|    22|   3|       1|  25.0|  58|1:34:56.789|         42|   1|       1:31.456|          228.3|Lewis Hamilton|         McLaren|Yas Marina Circuit|
+--------+------+----+--------+------+----+-----------+-----------+----+---------------+---------------+--------------+----------------+-----------------+

Files saved to: ['_SUCCESS', '.part-00000-bac770c1-c0ed-4518-85a7-baee5e340aa6-c000.csv.crc', '._SUCCESS.crc', 'part-00000-bac770c1-c0ed-4518-85a7-baee5e340aa6-c000.csv']
```

## Phase Four:

We built the dashboard in a structured project folder and imported race results data from the F1_results.csv file using a dedicated Python script. The data was loaded into a SQLite database and displayed interactively through the dashboard interface. The dashboard provided dynamic updates, allowing any changes in the underlying data to be reflected in real time upon refresh

# 🏎️ F1 Results Dashboard

## 📊 F1 Results (SQLite Database)

| | resultId | number | grid | position | points | laps | time | fastestLap | rank | fastestLapTime |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 22 | 1 | 1 | 10 | 58 | 1:34:50.616 | 39 | 2 | 1:27.452 |
| 1 | 2 | 3 | 5 | 2 | 8 | 58 | +5.478 | 41 | 3 | 1:27.739 |
| 2 | 3 | 7 | 7 | 3 | 6 | 58 | +8.163 | 41 | 5 | 1:28.090 |
| 3 | 4 | 5 | 11 | 4 | 5 | 58 | +17.181 | 58 | 7 | 1:28.603 |
| 4 | 5 | 23 | 3 | 5 | 4 | 58 | +18.014 | 43 | 1 | 1:27.418 |
| 5 | 6 | 8 | 13 | 6 | 3 | 57 | \N | 50 | 14 | 1:29.639 |
| 6 | 7 | 14 | 17 | 7 | 2 | 55 | \N | 54 | 8 | 1:29.534 |
| 7 | 8 | 1 | 15 | 8 | 1 | 53 | \N | 20 | 4 | 1:27.903 |

## 🏁 Top 5 Drivers by Points

| | Driver Name | Total Points |
|---|---|---|
| 0 | Lewis Hamilton | 4845.5 |
| 1 | Sebastian Vettel | 3098 |
| 2 | Max Verstappen | 2912.5 |
| 3 | Fernando Alonso | 2329 |
| 4 | Kimi Räikkönen | 1873 |

## 📝 McLaren Drivers

[
0 : "Alain Prost"
1 : "Alexander Wurz"
2 : "Andrea de Cesaris"
3 : "Ayrton Senna"
4 : "Brett Lunger"
5 : "Brian Redman"
6 : "Bruce McLaren"
7 : "Bruno Giacomelli"
8 : "Carlos Sainz"
9 : "Daniel Ricciardo"
10 : "Dave Charlton"
11 : "David Coulthard"
12 : "David Hobbs"
13 : "Denny Hulme"

## 🏆 Top 5 Winners

Lewis Hamilton: 260 wins

Michael Schumacher: 207 wins

Max Verstappen: 126 wins

Fernando Alonso: 124 wins

Sebastian Vettel: 111 wins

جامعة أم القرى
UMM AL-QURA UNIVERSITY

# Lessons Learned

- Combining datasets early improves consistency and reduces redundancy
- While denormalization and indexing can speed up queries in SQLite, they require thoughtful planning to manage data duplication and maintain consistency
- NoSQL databases like TinyDB are ideal for flexible, nested data structures
- PySpark requires clean and schema-consistent JSON files for efficient processing
- Each technology has strengths; integration teaches adaptability across storage paradigms
- Visualization tools make insights more accessible and highlight the importance of data quality
- Organizing code and files clearly makes the project easier to manage
- Working with a team improved our communication, time management, and problem-solving skills
- Data engineering is not just about storing data, it's making it ready for analysis, automation, and insights.
- Realizing how data engineers play a key role in turning data into value for companies and users