

Database Systems Course Term Project

F1 RACE ANALYTICS



Ege Demir – 150220007

İsmail Yeşilyurt – 150220009

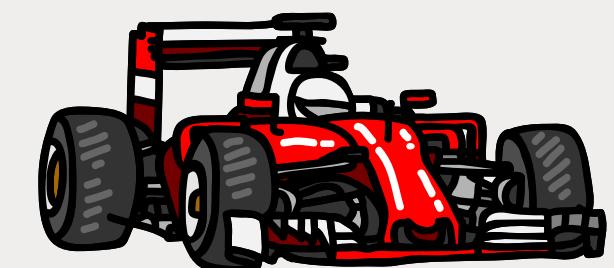
Nurettin Alper Kuzu – 150230001

Sıla Keküllüoğlu – 150210084

Turan İnceöz – 150220097

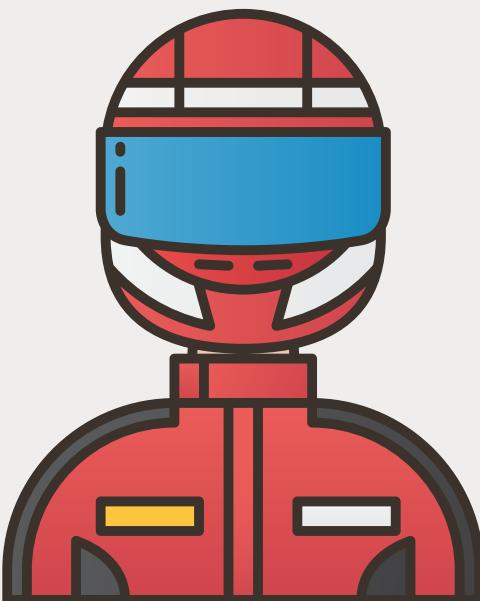
OUTLINE

1. Overview
2. Members and Responsibilities
3. Data Description
4. Tables
5. Database Diagrams
6. Website Features
7. Testing
8. Concluding Remarks



OVERVIEW

- Explore Formula 1 data since 1950, including constructors, drivers, races, circuits and detailed statistics
- Add your own race results, extending the dataset with custom data
- Comparisons between past races and performances to analyze trends and outcomes



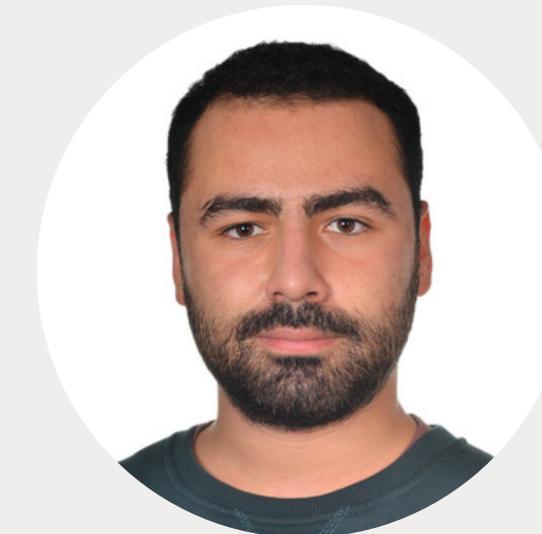
MEMBERS AND RESPONSIBILITIES



Ege Demir

Flask and Security

Tables: race,
race constructor standing



İsmail Yeşilyurt

Frontend and Authorization

Tables: user,
circuit



Alper Kuzu

Database and Backend

Tables: race data,
country



Sıla Keküllüoğlu

Backend and UI

Tables: Constructor



Turan İnceöz

Testing and UI

Tables: driver,
race driver standing

DATA DESCRIPTION

1. Source & Method

- Source: Open-source F1 database (fldb repository).
- Ingestion Method: Direct execution of a structured .sql dump file.
- Parsing: No Parsing, relied on pre-structured INSERT INTO statements for maximum data integrity.

2. Scope & Content

- Covers real-world Formula 1 history from 1950 to 2025.
- Includes: Circuits, Constructors, Drivers, Races.

3. Volume

- 75 Years of historical data.
- 1,100+ Races.
- 850+ Drivers.
- 25,000+ Race Results.

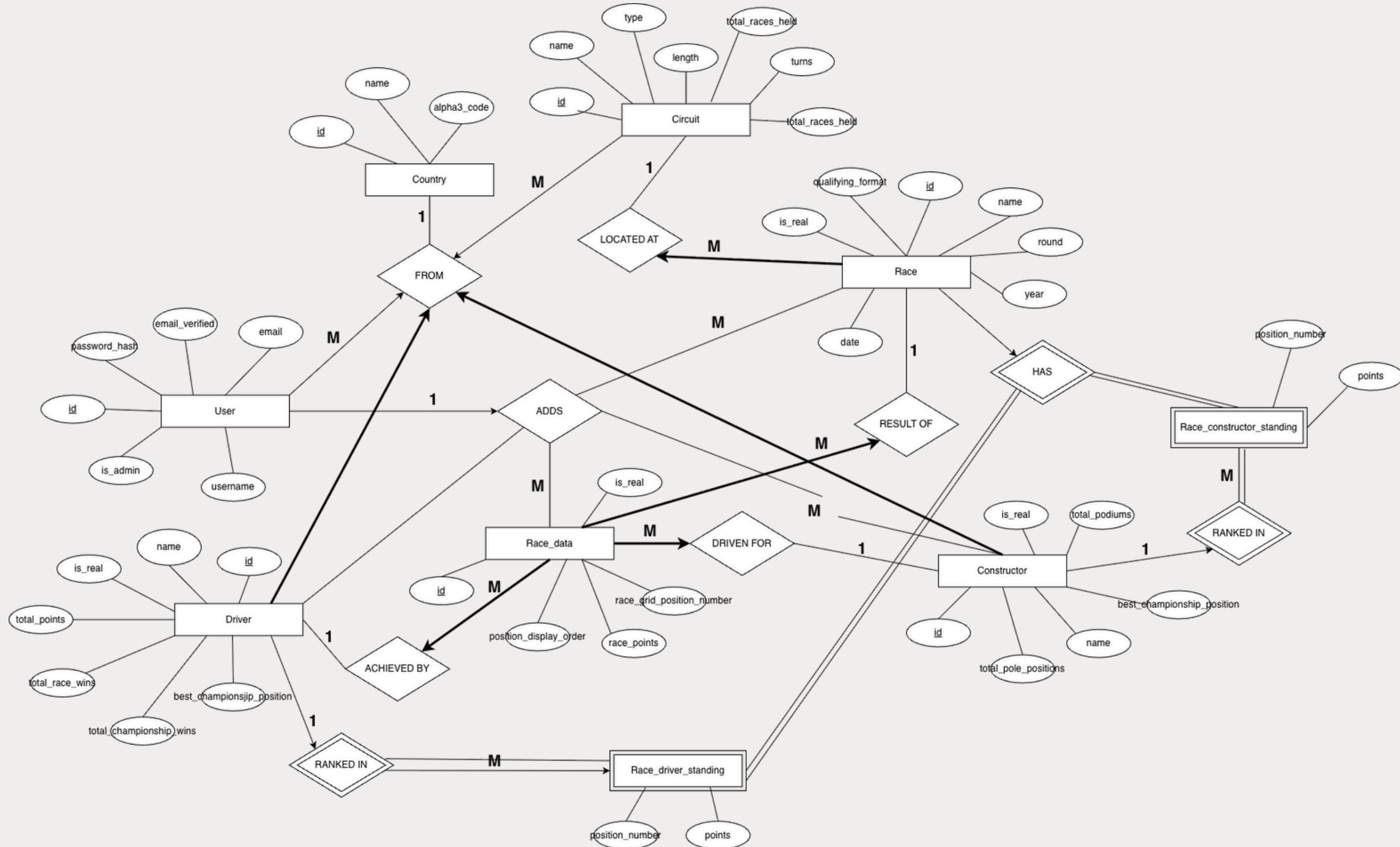
```
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (66, 3, 'maserati', 3) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (66, 4, 'brm', 2) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (67, 1, 'cooper', 19) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (67, 2, 'ferrari', 14) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (67, 3, 'vanwall', 8) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (67, 4, 'brm', 8) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (67, 5, 'maserati', 3) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (68, 1, 'cooper', 19) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (68, 2, 'ferrari', 14) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (68, 3, 'vanwall', 8) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (68, 4, 'brm', 8) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (68, 5, 'maserati', 3) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (69, 1, 'ferrari', 20) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (69, 2, 'cooper', 19) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (69, 3, 'vanwall', 16) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (69, 4, 'brm', 10) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (69, 5, 'maserati', 3) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (69, 6, 'lotus', 3) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (70, 1, 'ferrari', 28) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (70, 2, 'vanwall', 22) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (70, 4, 'brm', 10) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (70, 3, 'cooper', 19) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (70, 5, 'maserati', 6) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (70, 6, 'lotus', 3) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (71, 1, 'ferrari', 36) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (71, 2, 'vanwall', 25) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (71, 3, 'cooper', 23) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (71, 4, 'brm', 12) ON CONFLICT (race_id, constructor_id) DO NOTHING;
INSERT INTO "race_constructor_standing" ("race_id", "position_number", "constructor_id", "points") VALUES (71, 5, 'maserati', 6) ON CONFLICT (race_id, constructor_id) DO NOTHING;
```

Example part of .sql dump file.

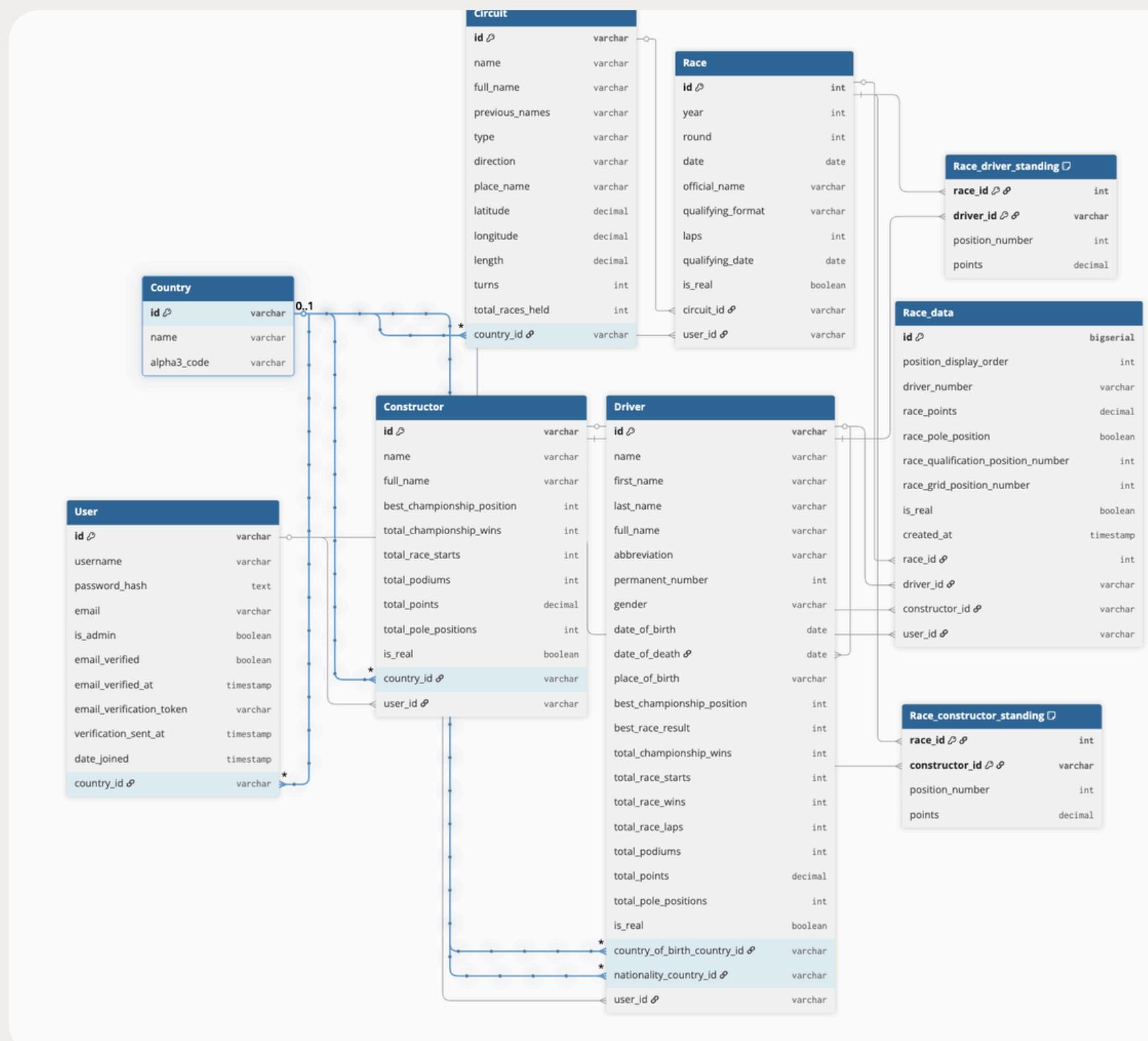
TABLES

Country PK: id FK: user_id, country_id	User PK: id FK: country_id,	Circuit PK: id FK: country_id,	Constructor PK: id FK: user_id, country_id
Driver PK: id FK: user_id, country_id	Race PK: id FK: user_id, country_id		Race_data PK: id FK: user_id, race_id, driver_id, constructor_id
Race_driver_standing PK: id FK: race_id, driver_id		Race_constructor_standing PK: id FK: race_id, constructor_id	

ER DIAGRAM



RELATIONAL SCHEMA



Mapping Overview

strong entity→relation with same primary key

Entity-to-Table Mapping

Country, User, Circuit, Constructor, Driver, Race
→mapped directly

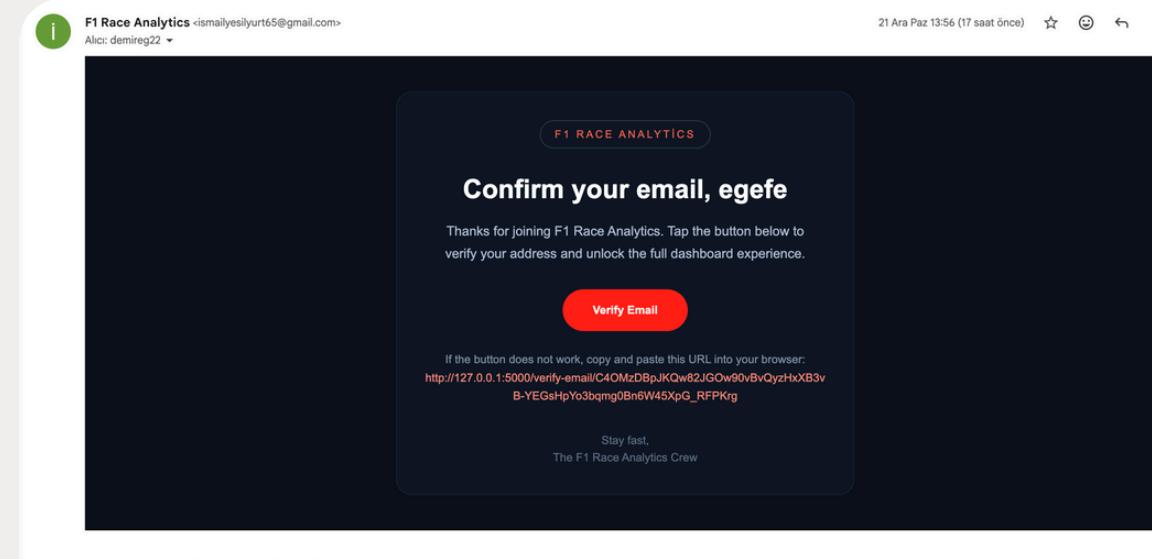
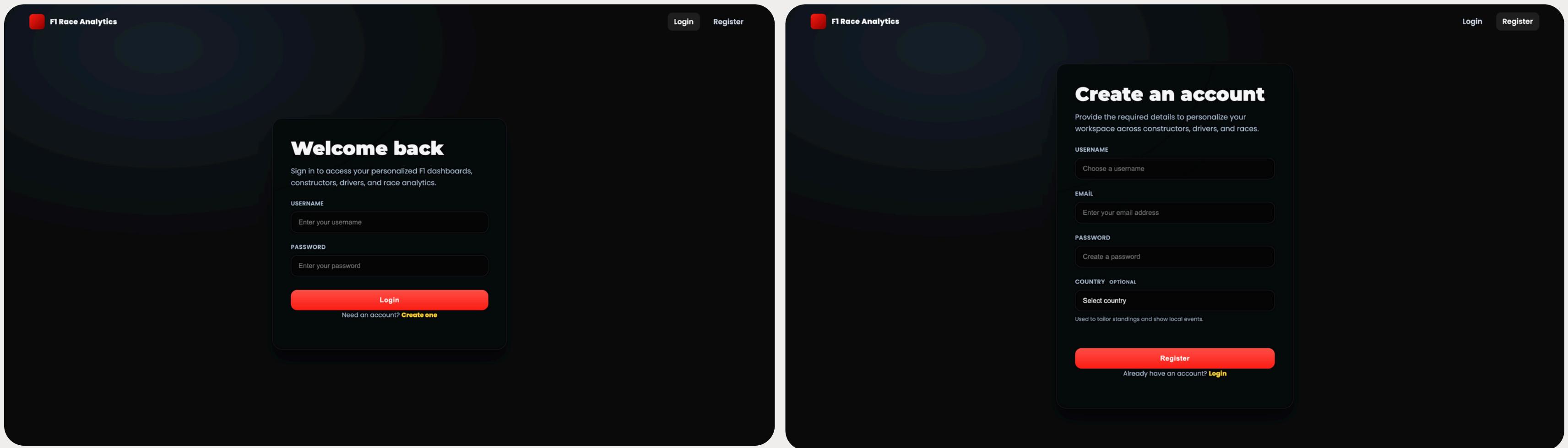
One-to-Many Relationships

implemented by using foreign keys

Many-to-Many Relationships

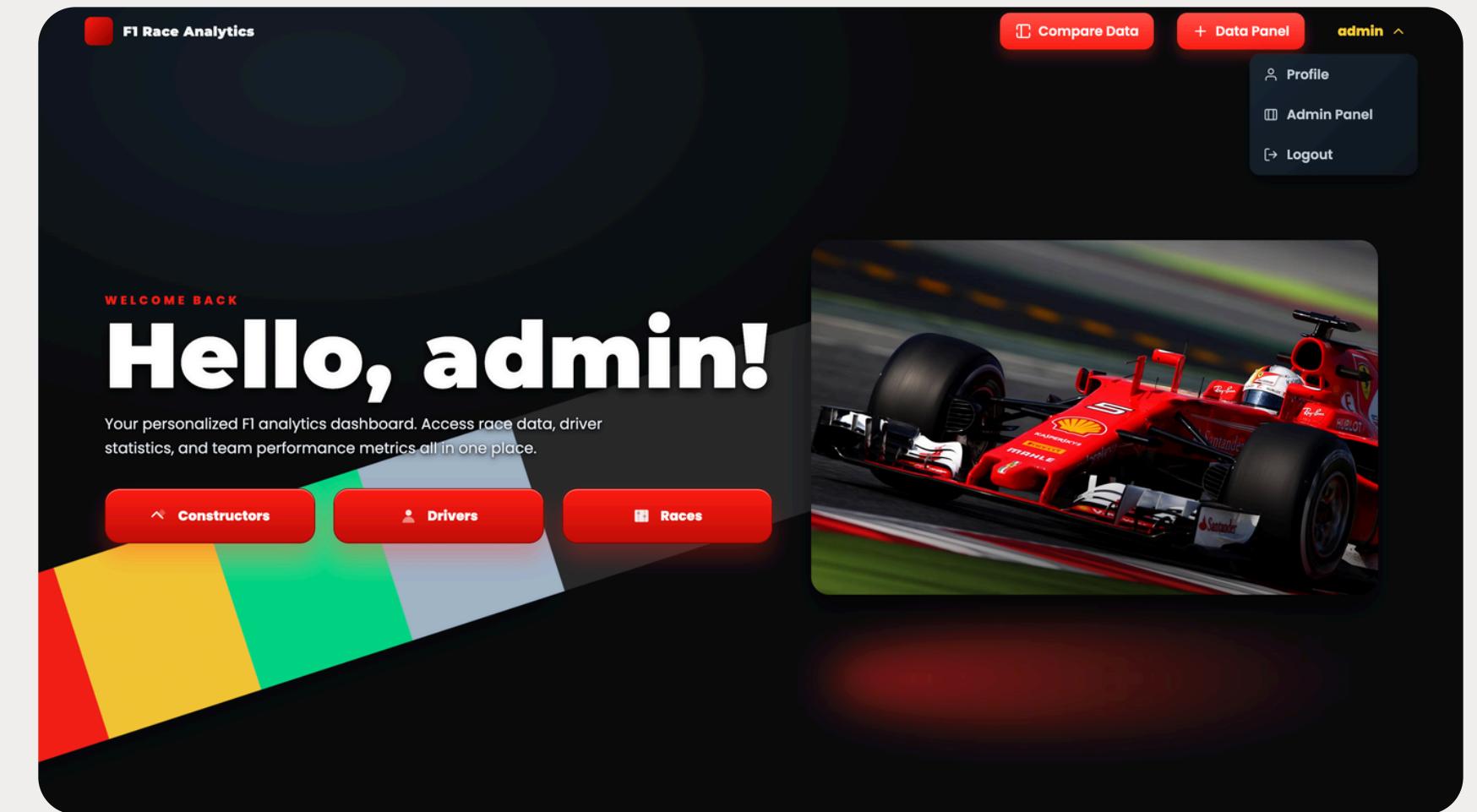
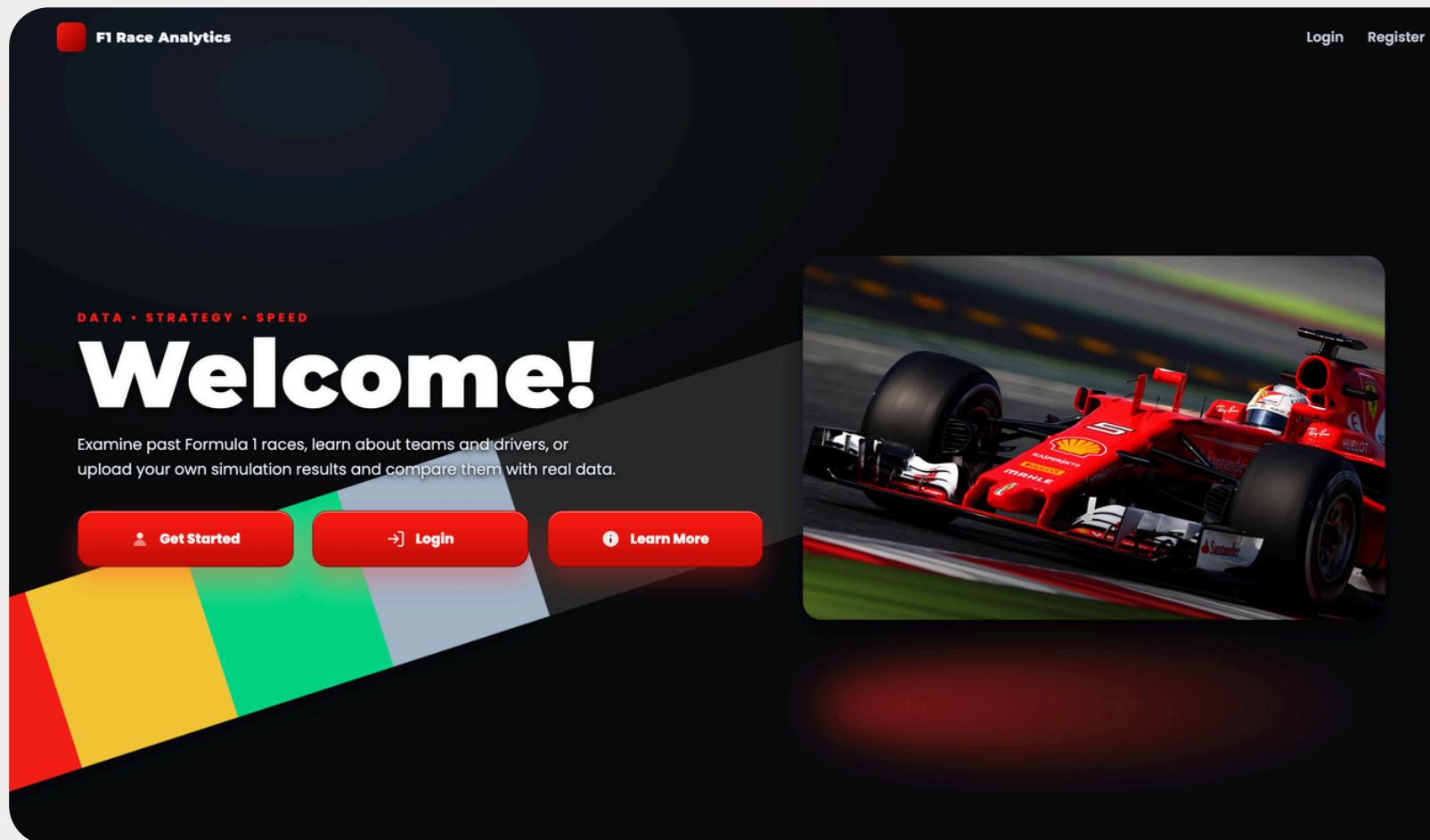
strong entity→relation with same primary key
strong entity→relation with same primary key

LOGIN AND REGISTER PAGES



← email verification

MAIN PAGE



after login

CONSTRUCTORS PAGE

Selections – READ operations

Red Bull
Austria Team

WORLD CHAMPIONSHIPS 6	TOTAL POINTS 8.076	PODIUMS 289	POLE POSITIONS 108	RACE STARTS 409
BEST FINISH 1				

V.S. NATIONAL AVERAGE (AUSTRIA) +5275.7 Points Ahead
Country Mean: 2800.3

Official F1 Team

F1 Constructors

Explore all Formula 1 teams, compare performance, and dive into the constructors behind the fastest cars in the world.

TEAM NAME: Search by name, NATIONALITY: All, CHAMPIONSHIPS: Min, TOTAL POINTS RANGE: Min - Max, Apply, Reset

Constructor Type: ALL, REAL, USER GENERATED, ABOVE COUNTRY AVERAGE

View Street Track Analysis, Participation Audit (Outer Join)

Andrea Moda Italy TITLES: 0, POINTS: 0.00 View Team	Apollon Switzerland TITLES: 0, POINTS: 0.00 View Team	Arrows United Kingdom TITLES: 0, POINTS: 142.00 View Team	Arzani-Volpini Italy TITLES: 0, POINTS: 0.00 View Team
---	---	---	--

Acton Rutterworth, Aston Martin, ATS, ATS

Street Track Specialists
Top 10 constructors ranked by their success on street circuits.

RANK	CONSTRUCTOR	NATIONALITY	RACES	STREET POINTS
#1	Frank Williams Racing Cars	United Kingdom	3	33434.00
#2	Ferrari	Italy	225	21247.00
#3	Red Bull	Austria	99	18444.00
#4	Mercedes	Germany	83	16089.00
#5	McLaren	United Kingdom	209	15783.00
#6	Williams	United Kingdom	191	6577.50
#7	Renault	France	87	2986.00
#8	Force India	India	51	2654.00
#9	Aston Martin	United Kingdom	35	2372.00
#10	Lotus	United Kingdom	108	2071.00

Participation Audit
Identifying registered constructors that have never competed in a Grand Prix.

CONSTRUCTOR	NATIONALITY	TOTAL ENTRIES	STATUS
FIRST	Italy	0	Never Raced

CONSTRUCTORS PAGE

Red Bull
Austria Team

WORLD CHAMPIONSHIPS 6	TOTAL POINTS 8.076	PODUMS 289	POLE POSITIONS 108	RACE STARTS 409
BEST FINISH 1				

VS. NATIONAL AVERAGE (AUSTRIA) **+5275.7 Points Ahead**
Country Mean: 2800.3

Official F1 Team

```
database > queries > constructors_above_avg.sql
1 -- constructors with above average performance within their country
2 SELECT
3   c.*,
4   co.name AS nationality,
5   -- calculate country average total points
6   (SELECT AVG(c2.total_points)
7    |> FROM constructor c2
8    WHERE c2.country_id = c.country_id) AS country_avg,
9   COUNT(*) OVER() AS full_count
10  FROM constructor c
11  JOIN country co ON c.country_id = co.id
12  WHERE
13    (%(name)s IS NULL OR c.full_name ILIKE '%%(name)s%' || %(name)s || '%%(name)s')
14    AND
15    (%(nationality)s IS NULL OR co.name = %(nationality)s)
16    AND
17    (%(champs_min)s IS NULL OR c.total_championship_wins >= %(champs_min)s)
18    AND
19    (%(total_points_min)s IS NULL OR c.total_points >= %(total_points_min)s)
20    AND
21    (%(total_points_max)s IS NULL OR c.total_points <= %(total_points_max)s)
22    AND
23    (%(is_real)s IS NULL OR c.is_real = %(is_real)s)
24    -- grouping by constructor and country
25  GROUP BY c.id, co.name
26  -- only include constructors with above average total points in their country
27  HAVING c.total_points > (
28    SELECT AVG(c3.total_points)
29    FROM constructor c3
30    WHERE c3.country_id = c.country_id
31  )
32  ORDER BY c.total_points DESC
33  LIMIT %(limit)s OFFSET %(offset)s;
```

Street Track Specialists
Top 10 constructors ranked by their success on street circuits.

RANK	CONSTRUCTOR	NATIONALITY	RACES	STREET POINTS
#1	Frank Williams Racing Cars	United Kingdom	3	33434.00
#2	Ferrari	Italy	225	21247.00
#3	Red Bull	Austria	99	18444.00
#4	Mercedes	Germany	83	16089.00
#5	McLaren	United Kingdom	209	15783.00
#6	Williams	United Kingdom	191	6577.50
#7	Renault	France	87	2986.00
#8	Force India	India	51	2654.00
#9	Aston Martin	United Kingdom	35	2372.00
#10	Lotus	United Kingdom	108	2071.00

database > queries > constructor_track_performance.sql

```
1 SELECT
2   -- Ranking based on total points in street track
3   RANK() OVER (ORDER BY SUM(COALESCE(rd.race_points, 0)) DESC) AS rank_position,
4   c.name AS constructor_name,
5   co.name AS nationality,
6   cir.type AS track_type,
7   COUNT(DISTINCT r.id) AS total_races,
8   -- Making sure to handle NULL race points
9   SUM(COALESCE(rd.race_points, 0)) AS total_points,
10  -- Best race position on street tracks
11  MIN(rd.position_display_order) AS best_position
12  FROM constructor c
13  JOIN country co ON c.country_id = co.id
14  JOIN race_data rd ON c.id = rd.constructor_id
15  JOIN race r ON rd.race_id = r.id
16  JOIN circuit cir ON r.circuit_id = cir.id
17  WHERE
18    UPPER(cir.type) = 'STREET'
19  GROUP BY c.name, co.name, cir.type
20  -- Only include constructors with points on street tracks
21  HAVING SUM(COALESCE(rd.race_points, 0)) > 0
22  ORDER BY total_points DESC
23  LIMIT 10;
```

CONSTRUCTORS PAGE

create, update, delete operations

FI Race Analytics

My Constructors

2 custom records found

COUNTRY	NAME	BEST CHAMPIONSHIP POSITION	TOTAL CHAMPIONSHIP WINS	TOTAL RACE STARTS	TOTAL PODIUMS	TOTAL POINTS	TOTAL POLE POSITIONS	ACTIONS
france	MyConstructor	4	1	5	1	3456.00	6	
hungary	MyConstructor2	5	0	2	0	348.00	3	

[+ Add New Constructor](#)

FI Race Analytics

Edit Constructor

Update your custom constructor details below

COUNTRY ID *	NAME *	BEST CHAMPIONSHIP POSITION	TOTAL CHAMPIONSHIP WINS *
<input type="text" value="France"/>	<input type="text" value="MyConstructor"/>	<input type="text" value="4"/>	<input type="text" value="1"/>
(Required)	(Required)	(Optional)	(Required)
TOTAL RACE STARTS *	TOTAL PODIUMS *	TOTAL POINTS *	TOTAL POLE POSITIONS *
<input type="text" value="5"/>	<input type="text" value="1"/>	<input type="text" value="3456.00"/>	<input type="text" value="6"/>
(Required)	(Required)	(Required)	(Required)

[Update Constructor](#) [Cancel](#)

Confirm Delete

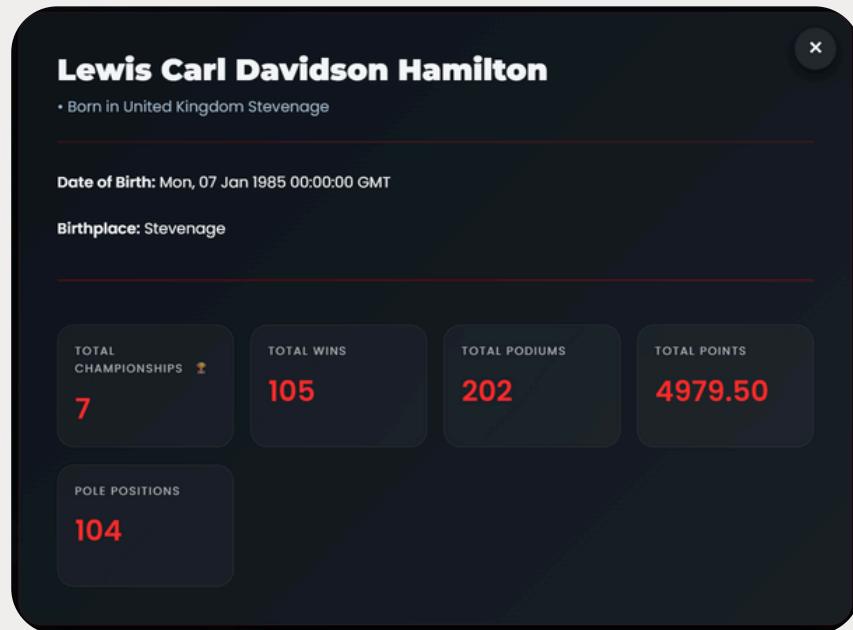
MyConstructor

This will also delete all race data entries that reference this constructor. This action cannot be undone.

[Cancel](#) [Delete](#)

DRIVERS PAGE

Selections – READ operations



F1 Race Analytics

F1 Drivers

Explore Formula 1 drivers, compare performance, and filter historical data.

Driver Name: Search by name... Nationality: e.g. British, Spanish Wins: 0 Podiums: 0 Points: 0 Poles: 0 Birthplace: Search birthplace...
From: To Driver Type: ALL REAL USER-GENERATED Apply Reset

Show Leaderboard

Adderly Fong Cheun-yue
Born: Hong Kong • Vancouver
CHAMPIONSHIPS: 0 WINS: 0 View Driver
Adolf Brudes von Breslau
Born: Germany • Groß Kottulin
CHAMPIONSHIPS: 0 WINS: 0 View Driver
Adolfo Julio Carlos Schweim Cruz
Born: Argentina • Buenos Aires
CHAMPIONSHIPS: 0 WINS: 0 View Driver
Adrián Campos Suñer
Born: Spain • Alzira
CHAMPIONSHIPS: 0 WINS: 0 View Driver
Adrian Sutil
Born: Germany • Starnberg
CHAMPIONSHIPS: 0 WINS: 0 View Driver

Driver Leaderboard

#	DRIVER	CHAMPIONSHIP WINS	RACE WINS	TOTAL POINTS
1	Lewis Carl Davidson Hamilton	7	105	4979.50
2	Michael Schumacher	5	56	696.00
3	Max Emilian Verstappen	4	69	3256.50
4	Sebastian Vettel	4	53	3098.00
5	Fernando Alonso Diaz	2	32	2369.00
6	Nico Erik Rosberg	1	23	1694.50
7	Kimi-Matias Räikkönen	1	21	1673.00
8	Jenson Alexander Lyons Button	1	15	1235.00
9	Lando Norris	1	11	1025.00
10	Felipe Massa	0	11	1067.00

DRIVERS PAGE

Complex Queries



```
database > queries > select_drivers.sql
SELECT
    d.id,
    d.name,
    d.first_name,
    d.last_name,
    d.full_name,
    d.abbreviation,
    d.permanent_number,
    d.gender,
    d.date_of_birth,
    d.date_of_death,
    d.place_of_birth,
    cb.name AS country_of_birth,
    n.name AS nationality,
    d.best_championship_position,
    d.best_race_result,
    d.total_championship_wins,
    d.total_race_starts,
    d.total_race_wins,
    d.total_race_laps,
    d.total_podiums,
    d.total_points,
    d.total_pole_positions,
    d.is_real,
    COUNT(*) OVER() AS full_count
FROM driver d
JOIN country cb ON cb.id = d.country_of_birth_country_id
JOIN country n ON n.id = d.nationality_country_id
WHERE
    (%(name)s IS NULL OR d.full_name ILIKE '%||%(name)s||%' OR
    AND (%(nationality)s IS NULL OR n.name ILIKE '%||%(nationality)s||%' OR
    AND (%(place_of_birth)s IS NULL OR d.place_of_birth ILIKE '%||%(place_of_birth)s||%' OR
    AND (%(wins_min)s IS NULL OR d.total_race_wins >=%(wins_min)s)
    AND (%(podiums_min)s IS NULL OR d.total_podiums >=%(podiums_min)s)
    AND (%(points_min)s IS NULL OR d.total_points >=%(points_min)s)
    AND (%(poles_min)s IS NULL OR d.total_pole_positions >=%(poles_min)s)
    AND (%(birth_from)s IS NULL OR d.date_of_birth >=%(birth_from)s::date)
    AND (%(birth_to)s IS NULL OR d.date_of_birth <=%(birth_to)s::date)
    AND (%(is_real)s IS NULL OR d.is_real = %(is_real)s)
LIMIT %(limit)s OFFSET %(offset)s;
```

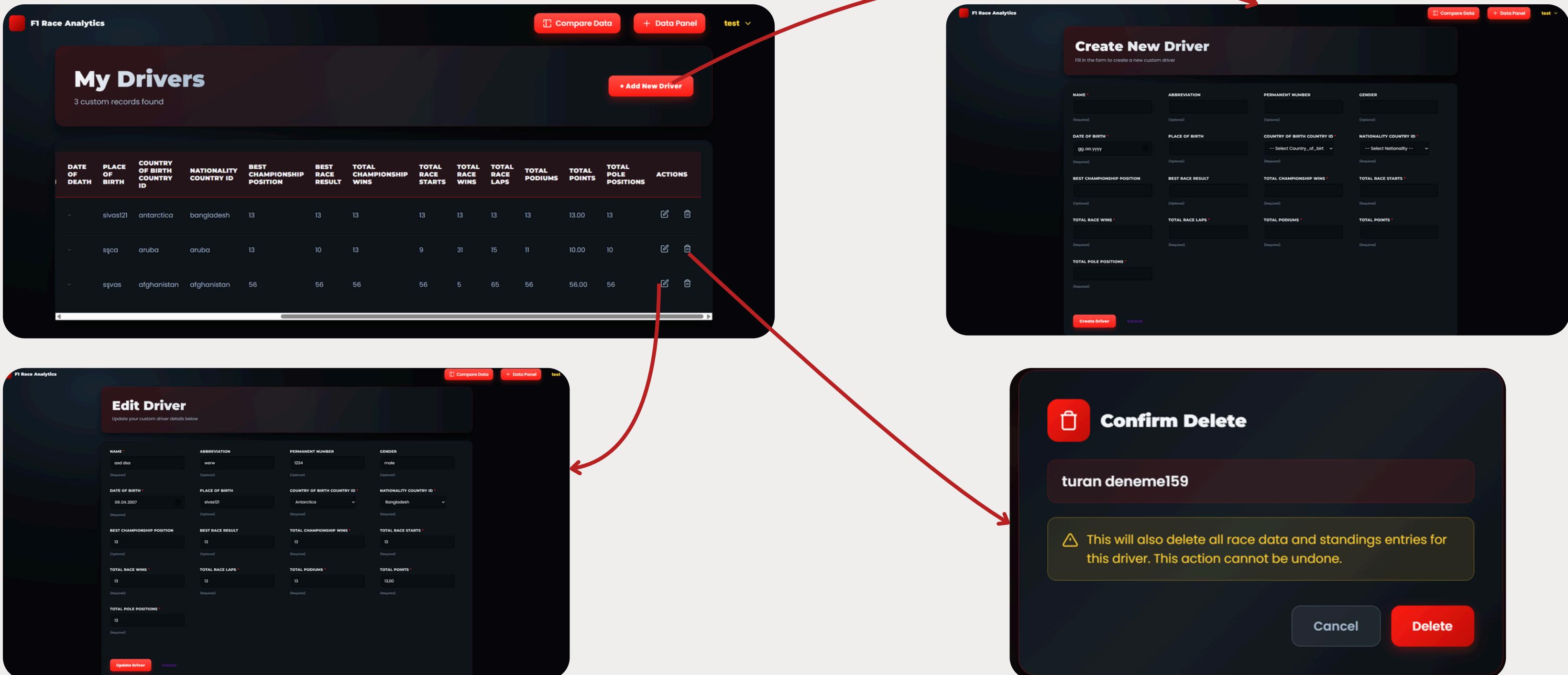
```
database > queries > driver_leaderboard.sql
WITH rd_dedup AS (
    SELECT *
    FROM (
        SELECT
            rd.*,
            ROW_NUMBER() OVER (
                PARTITION BY rd.race_id, rd.driver_id
                ORDER BY
                    rd.is_real DESC,
                    rd.created_at DESC,
                    rd.id DESC
            ) AS rn
        FROM race_data rd
    ) x
    WHERE rn = 1
),
scope AS (
    SELECT
        r.id AS race_id,
        r.year,
        rd.driver_id,
        COALESCE(rd.race_points, 0) AS race_points,
        rd.position_display_order
    FROM rd_dedup rd
    JOIN race r ON r.id = rd.race_id
    JOIN driver d ON d.id = rd.driver_id
    WHERE
        r.year BETWEEN %(year_from)s AND %(year_to)s
        AND rd.is_real = TRUE
        AND r.is_real = TRUE
        AND d.is_real = TRUE
),
race_winners AS (
    SELECT driver_id, COUNT(*) AS race_wins
    FROM (
        SELECT
            race_id,
            driver_id,
            ROW_NUMBER() OVER (
                PARTITION BY race_id
                ORDER BY
                    race_points DESC,
                    CASE
                        WHEN position_display_order IS NULL OR position_display_order <= 0 THEN 9999
                        ELSE position_display_order
                    END ASC,
                    driver_id ASC
        ) AS rn
        FROM scope
    ) t
    WHERE rn = 1
    GROUP BY driver_id
),
championship_counts AS (
    SELECT
        driver_id,
        COUNT(*) AS championship_wins
    FROM scope
    WHERE rn = 1
    GROUP BY driver_id
),
driver_points AS (
    SELECT
        driver_id,
        SUM(race_points) AS total_points
    FROM scope
    GROUP BY driver_id
)
SELECT
    d.id,
    d.full_name,
    COALESCE(cc.championship_wins, 0) AS championship_wins,
    COALESCE(rw.race_wins, 0) AS race_wins,
    COALESCE(dp.total_points, 0) AS total_points
FROM driver d
JOIN driver_points dp ON dp.driver_id = d.id
LEFT JOIN race_winners rw ON rw.driver_id = d.id
LEFT JOIN championship_counts cc ON cc.driver_id = d.id
ORDER BY
    championship.wins DESC,
    race_wins DESC,
    total_points DESC,
    d.full_name ASC
LIMIT %(limit)s;
```

#	DRIVER	CHAMPIONSHIP WINS	RACE WINS	TOTAL POINTS
1	Lewis Carl Davidson Hamilton	7	105	4979.50
2	Michael Schumacher	5	56	996.00
3	Max Emilian Verstappen	4	69	3256.50
4	Sebastian Vettel	4	53	3098.00
5	Fernando Alonso Diaz	2	32	2369.00
6	Nico Erik Rosberg	1	23	1594.50
7	Kimi-Matias Räikkönen	1	21	1873.00
8	Jenson Alexander Lyons Button	1	15	1235.00
9	Lando Norris	1	11	1325.00
10	Felipe Massa	0	11	1167.00

```
database > queries > driver_leaderboard.sql
season_points AS (
    SELECT
        year,
        driver_id,
        SUM(race_points) AS season_points
    FROM scope
    GROUP BY year, driver_id
),
season_ranked AS (
    SELECT
        year,
        driver_id,
        season_points,
        ROW_NUMBER() OVER (PARTITION BY year ORDER BY season_points DESC) AS rn
    FROM season_points
),
championship_counts AS (
    SELECT
        driver_id,
        COUNT(*) AS championship_wins
    FROM season_ranked
    WHERE rn = 1
    GROUP BY driver_id
),
driver_points AS (
    SELECT
        driver_id,
        SUM(race_points) AS total_points
    FROM scope
    GROUP BY driver_id
)
SELECT
    d.id,
    d.full_name,
    COALESCE(cc.championship_wins, 0) AS championship_wins,
    COALESCE(rw.race_wins, 0) AS race_wins,
    COALESCE(dp.total_points, 0) AS total_points
FROM driver d
JOIN driver_points dp ON dp.driver_id = d.id
LEFT JOIN race_winners rw ON rw.driver_id = d.id
LEFT JOIN championship_counts cc ON cc.driver_id = d.id
ORDER BY
    championship.wins DESC,
    race_wins DESC,
    total_points DESC,
    d.full_name ASC
LIMIT %(limit)s;
```

DRIVERS PAGE

create, update, delete operations



RACE PAGE

Selections – READ operations

F1 Race Analytics

Race Statistics

Aggregated race counts and qualifying format breakdown by year.

SPECIFIC YEAR: Any | YEAR RANGE: FROM 1950 TO 2025 | RACE COUNT: Min - Max

AVERAGE LAPS: Min - Max | SORT BY: Year | ORDER: High to Low

Statistics by year

YEAR	TOTAL RACES	AVERAGE LAPS	SPRINT	KNOCKOUT	ONE SESSION	TWO SESSION	FOUR LAPS	AGGREGATE
2025	25	60.920	0	24	0	0	0	1

Compare Data **+ Data Panel** eeg

F1 Race Analytics

F1 Races

Explore the Formula 1 race calendar, iconic circuits, and key stats for each Grand Prix weekend.

YEAR: Any | ROUND: Any | CIRCUIT: All Circuits | DATE RANGE: From _____ To _____ | OFFICIAL NAME: Search by official name | QUALIFYING FORMAT: All | RACE TYPE: ALL | REAL | USER-GENERATED

View Race Statistics

2008 Formula 1 Santander British Grand Prix
Silverstone • United Kingdom
YEAR ROUND DATE
2008 9 Sun,

Formula 1 Grosser Preis Santander von Deutschland 2008
Hockenheim • Germany
YEAR ROUND DATE
2008 10 Sun,

Formula 1 ING Magyar Nagydíj 2008
Budapest • Hungary
YEAR ROUND DATE
2008 11 Sun,

2008 Formula 1 Telefónica Grand Prix of Europe
Valencia • Spain
YEAR ROUND DATE
2008 12 Sun,

2008 Formula 1 ING Belgian Grand Prix
Spa • Belgium
YEAR ROUND DATE
2008 13 Sun,

Compare Data **+ Data Panel** eeg

RACE PAGE

Complex Queries

```

SELECT
    stats.decade, stats.year,
    stats.race_count, stats.avg_laps,
    stats.sprint_races, stats.knockout_races,
    stats.one_session_races, stats.two_session_races,
    stats.four_laps_races, stats.aggregate_races,
    SUM(stats.race_count) OVER (PARTITION BY stats.decade) AS decade_race_count,
    ROUND(AVG(stats.avg_laps) OVER (PARTITION BY stats.decade), 3) AS decade_avg_laps,
    COUNT(*) OVER() AS full_count
FROM (
    SELECT
        (r.year / 10) * 10 AS decade,
        r.year,
        COUNT(*) AS race_count,
        ROUND(AVG(r.laps), 3) AS avg_laps,
        SUM(CASE WHEN r.qualifying_format = 'SPRINT_RACE' THEN 1 ELSE 0 END) AS sprint_races,
        SUM(CASE WHEN r.qualifying_format = 'KNOCKOUT' THEN 1 ELSE 0 END) AS knockout_races,
        SUM(CASE WHEN r.qualifying_format = 'ONE_SESSION' THEN 1 ELSE 0 END) AS one_session_races,
        SUM(CASE WHEN r.qualifying_format = 'TWO_SESSION' THEN 1 ELSE 0 END) AS two_session_races,
        SUM(CASE WHEN r.qualifying_format = 'FOUR_LAPS' THEN 1 ELSE 0 END) AS four_laps_races,
        SUM(CASE WHEN r.qualifying_format = 'AGGREGATE' THEN 1 ELSE 0 END) AS aggregate_races
    FROM race r
    WHERE (%year)s IS NULL OR r.year = %year)s
        AND (%year_from)s IS NULL OR r.year >= %year_from)s
        AND (%year_to)s IS NULL OR r.year <= %year_to)s
    GROUP BY (r.year / 10) * 10, r.year
) AS stats
WHERE (%race_count_min)s IS NULL OR stats.race_count >= %(race_count_min)s
    AND (%race_count_max)s IS NULL OR stats.race_count <= %(race_count_max)s
    AND (%avg_laps_min)s IS NULL OR stats.avg_laps >= %(avg_laps_min)s
    AND (%avg_laps_max)s IS NULL OR stats.avg_laps <= %(avg_laps_max)s
ORDER BY
    CASE WHEN %(sort_by)s = 'year' AND %(sort_dir)s = 'ASC' THEN stats.year END ASC,
    CASE WHEN %(sort_by)s = 'year' AND %(sort_dir)s = 'DESC' THEN stats.year END DESC,
    CASE WHEN %(sort_by)s = 'race_count' AND %(sort_dir)s = 'ASC' THEN stats.race_count END ASC,
    CASE WHEN %(sort_by)s = 'race_count' AND %(sort_dir)s = 'DESC' THEN stats.race_count END DESC,
    CASE WHEN %(sort_by)s = 'avg_laps' AND %(sort_dir)s = 'ASC' THEN stats.avg_laps END ASC,
    CASE WHEN %(sort_by)s = 'avg_laps' AND %(sort_dir)s = 'DESC' THEN stats.avg_laps END DESC,
    stats.decade ASC, stats.year DESC
LIMIT %(limit)s OFFSET %(offset)s;

```

```

SELECT
    r.id AS race_id, r.year AS race_year,
    r.round AS race_round, r.official_name AS race_name,
    r.date AS race_date, r.laps AS total_laps,
    r.qualifying_format, cir.id AS circuit_id,
    cir.full_name AS circuit_name, cir.place_name AS circuit_place,
    cir.type AS circuit_type, cir.length AS circuit_length_km,
    cir.turns AS circuit_turns, cir.direction AS circuit_direction,
    co.id AS country_id, co.name AS country_name,
    co.alpha3_code AS country_code, rd.id AS result_id,
    rd.position_display_order AS finish_position, rd.driver_number,
    rd.race_points, rd.race_pole_position,
    rd.race_qualification_position_number AS quali_position, rd.race_grid_position_number AS grid_position,
    d.id AS driver_id, d.full_name AS driver_name,
    d.abbreviation AS driver_abbr, d.nationality_country_id AS driver_nationality_id,
    con.id AS constructor_id, con.full_name AS constructor_name,
    con.name AS constructor_short_name, dco.name AS driver_nationality,
    COUNT(*) OVER() AS full_count
FROM race r
INNER JOIN circuit cir ON r.circuit_id = cir.id
INNER JOIN country co ON cir.country_id = co.id
INNER JOIN race_data rd ON rd.race_id = r.id
INNER JOIN driver d ON rd.driver_id = d.id
INNER JOIN constructor con ON rd.constructor_id = con.id
LEFT JOIN country dco ON d.nationality_country_id = dco.id
WHERE r.id = %(race_id)s
ORDER BY rd.position_display_order ASC NULLS LAST, rd.race_points DESC NULLS LAST
LIMIT %(limit)s OFFSET %(offset)s;

```

Statistics by year					
YEAR	TOTAL RACES	AVERAGE LAPS	SPRINT	KNOCKOUT	ONE SESSION
2025	25	60.920	0	24	0
2024	24	60.167	0	24	0
2023	22	60.227	0	22	0
2022	23	56.435	3	19	0
2021	22	58.955	3	19	0
2020	17	61.000	0	17	0
2019	21	60.095	0	21	0

Formula 1 ING Magyar Nagydíj 2008								
Year: 2008 Round: 11 Date: Sun, Format: Knockout Laps: 70		Race Results						
P	NO	DRIVER	CONSTRUCTOR	GRID	QUAL	POINTS		
1	23	Heikki Johannes Kovalainen	McLaren Racing	2	2	10.00		
2	12	Timo Glock	Toyota Racing	5	5	8.00		
3	1	Kimi-Matias Räikkönen	Scuderia Ferrari	6	6	6.00		
4	5	Fernando Alonso Diaz	Renault Sport Formula One Team	7	7	5.00		
5	22	Lewis Carl Davidson Hamilton	McLaren Racing	1	1	4.00		
6	6	Nelson Ângelo Tâmas Piquet Souto Maior	Renault Sport Formula One Team	10	10	3.00		
7	11	Jarno Trulli	Toyota Racing	9	9	2.00		
8	4	Robert Józef Kubica	BMW Sauber F1 Team	4	4	1.00		
9	2	Felipe Massa	Scuderia Ferrari	3	3	-		
10	10	Mark Alan Webber	Red Bull Racing	8	8	-		

For Control Mechanism

RACE PAGE

create, update, delete operations

F1 Race Analytics

Create New Race

Fill in the form to create a new custom race

CIRCUIT ID *

YEAR *

ROUND *

DATE *

OFFICIAL NAME *

QUALIFYING FORMAT *

LAPS *

QUALIFYING DATE

Create Race **Cancel**

F1 Race Analytics

Edit Race

Update your custom race details below

CIRCUIT ID *

YEAR *

ROUND *

DATE *

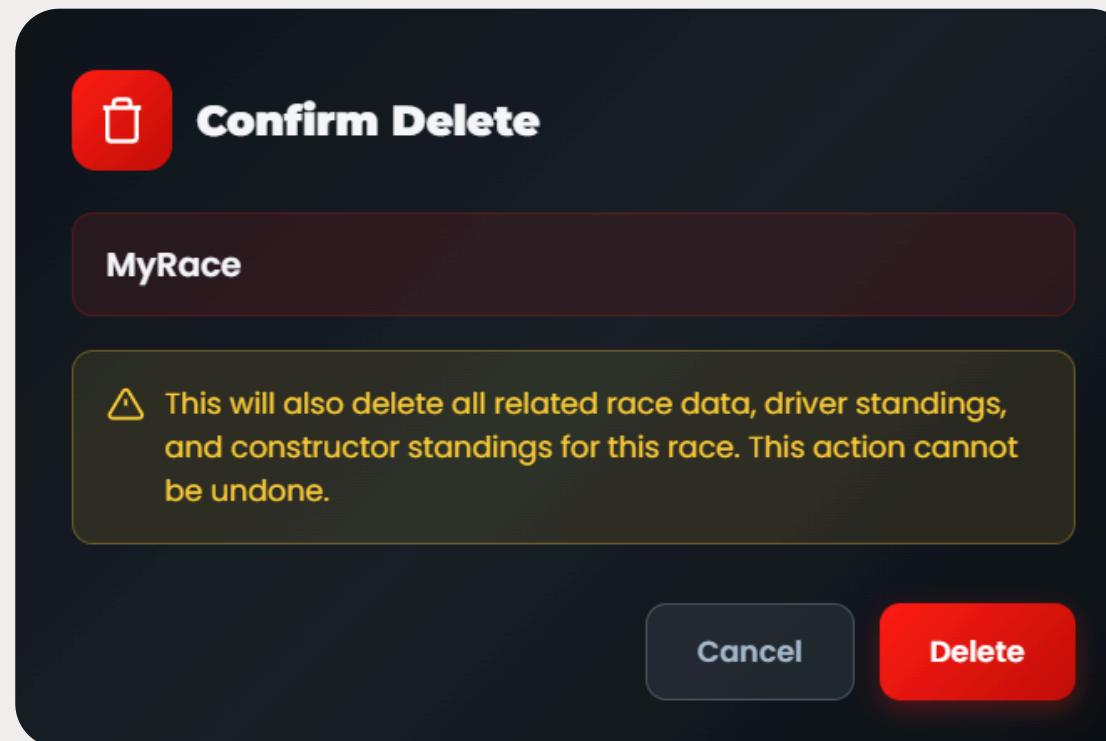
OFFICIAL NAME *

QUALIFYING FORMAT *

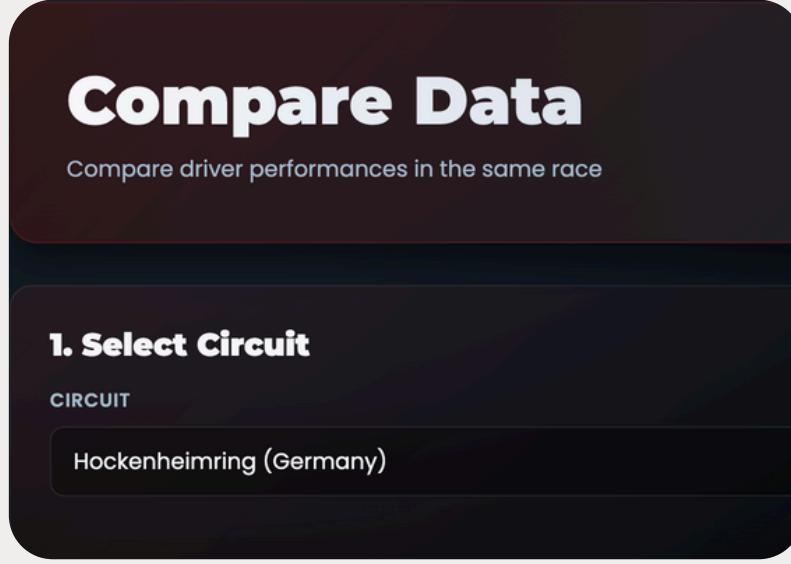
LAPS *

QUALIFYING DATE

Update Race **Cancel**



DATA COMPARISON



Only relevant data shown

Searchable dropdowns

SIDE A Driver Selection

Formula 1 Emirates Grosser Preis von Deutschland 2018
Sun, 22 Jul 2018 00:00:00 GMT • 67 laps

YEAR
2018

CONSTRUCTOR
Ferrari

DRIVER
Sebastian Vettel (VET)

SIDE B Driver Selection

Formula 1 Grosser Preis Santander von Deutschland 2012
Sun, 22 Jul 2012 00:00:00 GMT • 67 laps

YEAR
2012

CONSTRUCTOR
Red Bull

DRIVER
Sebastian Vettel (VET)

DATA COMPARISON

Selections – READ operations/Complex Queries



comparison info derived from

```

1 race_1_positions AS (
2     SELECT rd.driver_id, rd.constructor_id, rd.race_points, rd.race_grid_position_number AS grid,
3         r.year, c.name AS constructor, RANK() OVER (ORDER BY rd.position_display_order) AS finish
4     FROM race_data rd
5     INNER JOIN race r ON rd.race_id = r.id
6     INNER JOIN constructor c ON rd.constructor_id = c.id
7     WHERE rd.race_id = %s
8 ),
9 driver_1_race AS (SELECT * FROM race_1_positions WHERE driver_id = %s),
10 driver_2_race AS (SELECT * FROM race_1_positions WHERE driver_id = %s),
11 driver_1_history AS (
12     SELECT rd.driver_id, COUNT(DISTINCT rd.race_id) AS races,
13         COUNT(CASE WHEN rd.position_display_order = 1 THEN 1 END) AS wins,
14         COUNT(CASE WHEN rd.position_display_order <= 3 THEN 1 END) AS podiums,
15         ROUND(AVG(rd.position_display_order)::numeric, 1) AS avg_finish,
16         SUM(COALESCE(rd.race_points, 0)) AS total_points
17     FROM race_data rd INNER JOIN race r ON rd.race_id = r.id
18     WHERE rd.driver_id = %s AND r.circuit_id = %s
19     GROUP BY rd.driver_id
20 ),
21 driver_2_history AS (
22     SELECT rd.driver_id, COUNT(DISTINCT rd.race_id) AS races,
23         COUNT(CASE WHEN rd.position_display_order = 1 THEN 1 END) AS wins,
24         COUNT(CASE WHEN rd.position_display_order <= 3 THEN 1 END) AS podiums,
25         ROUND(AVG(rd.position_display_order)::numeric, 1) AS avg_finish,
26         SUM(COALESCE(rd.race_points, 0)) AS total_points
27     FROM race_data rd INNER JOIN race r ON rd.race_id = r.id
28     WHERE rd.driver_id = %s AND r.circuit_id = %s
29     GROUP BY rd.driver_id
30 ),
31

```

```

1 driver_1_season AS (
2     SELECT rds.driver_id, rds.position_number AS champ_pos, rds.points,
3         (SELECT COUNT(*) FROM race_data rd2 INNER JOIN race r2 ON rd2.race_id = r2.id
4          WHERE rd2.driver_id = %s AND rd2.position_display_order = 1
5          AND r2.year = (SELECT year FROM race WHERE id = %s)) AS wins
6     FROM race_driver_standing rds WHERE rds.driver_id = %s AND rds.race_id = %s
7 ),
8 driver_2_season AS (
9     SELECT rds.driver_id, rds.position_number AS champ_pos, rds.points,
10        (SELECT COUNT(*) FROM race_data rd2 INNER JOIN race r2 ON rd2.race_id = r2.id
11          WHERE rd2.driver_id = %s AND rd2.position_display_order = 1
12          AND r2.year = (SELECT year FROM race WHERE id = %s)) AS wins
13     FROM race_driver_standing rds WHERE rds.driver_id = %s AND rds.race_id = %s
14 ),
15
16 circuit_info AS (
17     SELECT ci.full_name, co.name AS country, ci.length, ci.turns
18     FROM circuit ci INNER JOIN country co ON ci.country_id = co.id WHERE ci.id = %s
19 )
20
21 SELECT ci.*, d1.full_name AS driver_1, d1r.finish AS d1_finish, d1h.wins AS d1_wins,
22       d2.full_name AS driver_2, d2r.finish AS d2_finish, d2h.wins AS d2_wins
23 FROM circuit_info ci
24 CROSS JOIN driver_1_race d1r LEFT JOIN driver d1 ON d1r.driver_id = d1.id
25 CROSS JOIN driver_2_race d2r LEFT JOIN driver d2 ON d2r.driver_id = d2.id
26 LEFT JOIN driver_1_history d1h ON d1.id = d1h.driver_id
27 LEFT JOIN driver_2_history d2h ON d2.id = d2h.driver_id
28 LEFT JOIN driver_1_season d1s ON d1.id = d1s.driver_id
29 LEFT JOIN driver_2_season d2s ON d2.id = d2s.driver_id;

```

RACE DATA

create, update, delete operations

FI Race Analytics

Create New Race Data

Fill in the form to add a new race_data row

RACE	DRIVER	CONSTRUCTOR	POSITION (DISPLAY ORDER)
-- Select Race --	-- Select Driver --	-- Select Constructor --	
DRIVER NUMBER	RACE POINTS	POLE POSITION	QUALIFICATION POSITION
GRID POSITION			

Create Race Data **Cancel**

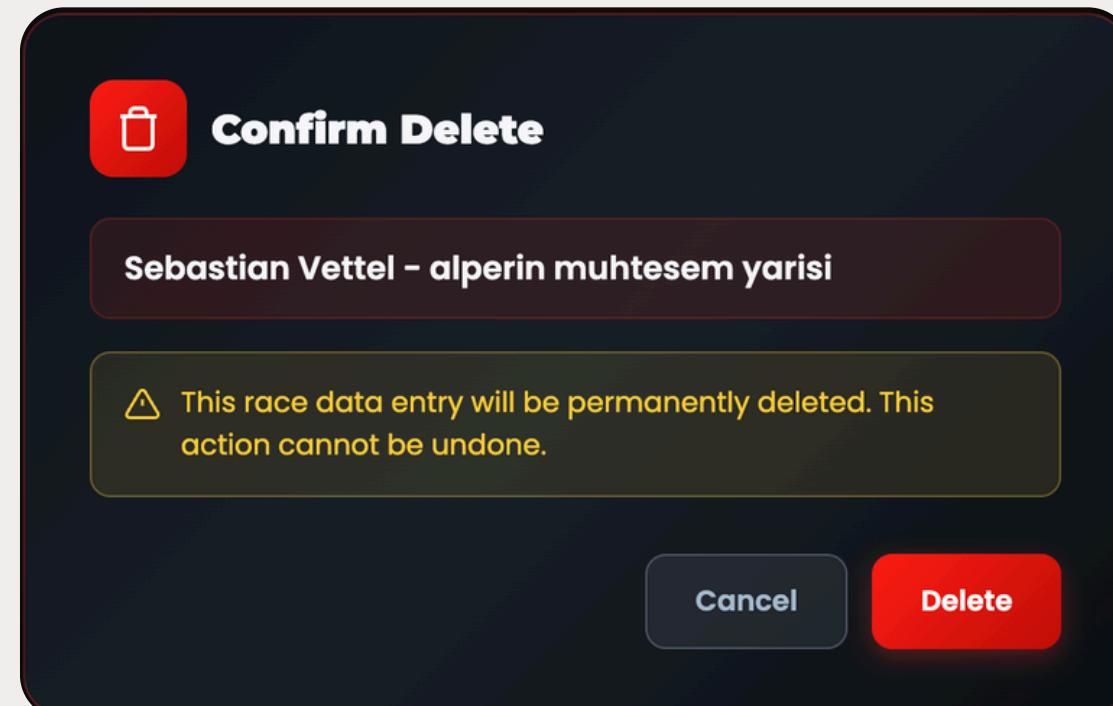
FI Race Analytics

Edit Race Data

Update the race data row below

RACE	DRIVER	CONSTRUCTOR	POSITION (DISPLAY ORDER)
alperin muhtesem yarisi (1)	Sebastian Vettel	Red Bull Racing	1
DRIVER NUMBER	RACE POINTS	POLE POSITION	QUALIFICATION POSITION
5	25,00	Yes	1
GRID POSITION			

Update Race Data **Cancel**



CIRCUIT

F1 Race Analytics

2008 Formula 1 Santander British Grand Prix

Year: 2008 Round: 9 Date: Sun, 06 Jul 2008 00:00:00 GMT Format: Knockout Laps: 60

Race Results

P	NO	DRIVER	CONSTRUCTOR	GRID	QUAL	POINTS
1	22	Lewis Carl Davidson Hamilton	McLaren Racing	4	4	10.00
2	3	Nick Lars Heidfeld	BMW Sauber F1 Team	5	5	8.00
3	17	Rubens Gonçalves Barrichello	Honda Racing F1 Team	16	16	6.00
4	1	Kimi-Matias Räikkönen	Scuderia Ferrari	3	3	5.00
5	23	Heikki Johannes Kovalainen	McLaren Racing	1	1	4.00
6	5	Fernando Alonso Diaz	Renault Sport Formula One Team	6	6	3.00
7	11	Jarno Trulli	Toyota Racing	14	14	2.00
8	8	Kazuki Nakajima	Williams Grand Prix Engineering	15	15	1.00
9	2	Felipe Massa	Scuderia Ferrari	9	9	-
10	10	Mark Alan Webber	Red Bull Racing	2	2	-
11	9	David Marshall Coulthard	Red Bull Racing	11	11	-
12	15	Sebastian Vettel	Scuderia Toro Rosso	8	8	-
13	4	Robert Józef Kubica	BMW Sauber F1 Team	10	10	-
14	7	Nico Erik Rosberg	Williams Grand Prix Engineering	-	18	-
15	6	Nelson Ângelo Tamsma Piquet Souto Maior	Renault Sport Formula One Team	7	7	-
16	16	Jenson Alexander Lyons Button	Honda Racing F1 Team	17	17	-
17	14	Sébastien Olivier Bourdais	Scuderia Toro Rosso	13	13	-

Silverstone Circuit

Silverstone • United Kingdom

LENGTH 5.891 km | TURNS 18 | DIRECTION Clockwise

TYPE Race | COUNTRY United Kingdom | COORDINATES 52.079° N, 1.017° W

Driver Championship

P	DRIVER	NAT	PTS
1	Lewis Carl Davidson Hamilton	United Kingdom	48.00
2	Felipe Massa	Brazil	48.00
3	Kimi-Matias Räikkönen	Finland	48.00
4	Robert Józef Kubica	Poland	46.00
5	Nick Lars Heidfeld	Germany	36.00

Constructor Championship

P	TEAM	NAT	PTS
1	Scuderia Ferrari	Italy	96.00
2	BMW Sauber F1 Team	Germany	82.00
3	McLaren Racing	United	72.00

F1 Race Analytics

Silverstone Circuit

Silverstone • United Kingdom

LENGTH 5.891 km | TURNS 18 | DIRECTION Clockwise

TYPE Race | COUNTRY United Kingdom | COORDINATES 52.079° N, 1.017° W

Track Snapshot

Silverstone Circuit spans 5.891 km with 18 turns in Clockwise direction. It has hosted 60 events between 1950 and 2025, located near Silverstone, United Kingdom.

OFFICIAL RACES: 60 | AVERAGE LAPS: 63.5 | UNIQUE WINNERS: 85

Historic Highlights

HOME CONSTRUCTORS: 59 | HOME DRIVERS: 171 | FIRST EVENT: 1950 | MOST RECENT: 2025

Track Map

52°0'44.3"N 1°0'1.0"W
3XH+M69 Towcester, Birleşik Krallik Yol Tarifi
Daha büyük harita görüntüle

Races Hosted Here

- Formula 1 Qatar Airways British Grand Prix 2025 (2025-07-06) Participants: 177 OFFICIAL
- Formula 1 Qatar Airways British Grand Prix 2024 (2024-07-07) Participants: 189 OFFICIAL
- Formula 1 Aramco British Grand Prix 2023 (2023-07-09) Participants: 169 OFFICIAL

CIRCUIT

```

1  WITH circuit_base AS (
2      SELECT c.id, c.name, c.full_name, c.place_name, c.type, c.direction, c.length, c.turns,
3          c.total_races_held, c.latitude, c.longitude, co.name AS country_name
4      FROM circuit c
5      LEFT JOIN country co ON c.country_id = co.id
6      WHERE c.id = %(circuit_id)s
7      LIMIT 1
8  ),
9  race_rollup AS (
10     SELECT
11         r.circuit_id,
12         COUNT(*) AS total_races,
13         COUNT(*) FILTER (WHERE r.is_real) AS official_races,
14         COUNT(*) FILTER (WHERE NOT r.is_real) AS simulated_races,
15         MIN(r.year) AS first_year,
16         MAX(r.year) AS last_year,
17         AVG(r.laps)::NUMERIC(10,2) AS avg_laps
18     FROM race r
19     WHERE r.circuit_id = %(circuit_id)s
20     GROUP BY r.circuit_id
21  ),
22  winner_stats AS (
23      SELECT r.circuit_id,
24          COUNT(DISTINCT CASE WHEN rd.position_display_order = 1 THEN rd.driver_id END) AS unique_winners
25      FROM race r
26      LEFT JOIN race_data rd ON rd.race_id = r.id
27      WHERE r.circuit_id = %(circuit_id)s
28      GROUP BY r.circuit_id
29  ),
30  home_driver_counts AS (
31      SELECT c.id AS circuit_id, COUNT(DISTINCT d.id) AS home_drivers
32      FROM circuit c
33      LEFT JOIN country co ON c.country_id = co.id
34      LEFT JOIN driver d ON d.nationality_country_id = co.id
35      WHERE c.id = %(circuit_id)s
36      GROUP BY c.id
37  ),
38  home_constructor_counts AS (
39      SELECT
40          c.id AS circuit_id,
41          COUNT(DISTINCT cons.id) AS home_constructors
42      FROM circuit c
43      LEFT JOIN country co ON c.country_id = co.id
44      LEFT JOIN constructor cons ON cons.country_id = co.id
45      WHERE c.id = %(circuit_id)s
46      GROUP BY c.id
47  ),

```

```

1  race_payload AS (
2      SELECT
3          r.circuit_id,
4          json_agg(
5              json_build_object(
6                  'id', r.id,'year', r.year,'round', r.round,'official_name', r.official_name,'date', r.date,
7                  'qualifying_format', r.qualifying_format,'laps', r.laps,'is_real', r.is_real,
8                  'participant_count', (
9                      SELECT COUNT(*)
10                     FROM race_data rd
11                     WHERE rd.race_id = r.id
12                 ),
13                  'winner', (
14                      SELECT json_build_object(
15                          'driver_id', d.id,
16                          'driver_name', d.full_name,
17                          'constructor_name', cons.full_name
18                      )
19                      FROM race_data rd2
20                      JOIN driver d ON rd2.driver_id = d.id
21                      LEFT JOIN constructor cons ON cons.id = rd2.constructor_id
22                      WHERE rd2.race_id = r.id
23                      ORDER BY rd2.position_display_order ASC
24                      LIMIT 1
25                  )
26                  )
27                  ORDER BY r.year DESC, r.round DESC
28              ) AS races
29      FROM race r
30      WHERE r.circuit_id = %(circuit_id)s
31      GROUP BY r.circuit_id
32  )
33  SELECT
34      cb.*,
35      rr.total_races,rr.official_races,rr.simulated_races,rr.first_year,rr.last_year,
36      rr.avg_laps,ws.unique_winners,hd.home_drivers,hc.home_constructors,rp.races
37  FROM circuit_base cb
38  LEFT JOIN race_rollup rr ON rr.circuit_id = cb.id
39  LEFT JOIN winner_stats ws ON ws.circuit_id = cb.id
40  LEFT JOIN home_driver_counts hd ON hd.circuit_id = cb.id
41  LEFT JOIN home_constructor_counts hc ON hc.circuit_id = cb.id
42  LEFT JOIN race_payload rp ON rp.circuit_id = cb.id;
43

```

ADMIN PANEL

F1 Race Analytics

Constructors Drivers Races Logout

Admin Panel

Manage all database records and data

Circuits Manage circuit records

Drivers Manage driver records

Constructors Manage constructor records

Races Manage race records

Race Data Manage race data records

Countries Manage country records

Users Manage user records

F1 Race Analytics

Admin Panel Profile Logout

Race Data

184359 records found

+ Add New Race Data

ID	RACE_ID	DRIVER_ID	CONSTRUCTOR_ID	POSITION_DISPLAY_ORDER	DRIVER_NUMBER	RACE_POINTS	RACE_POLE_POSITION	RACE_QUALIFICATION_POSITION_NUMBER	RACE_GRID_POSITION_NUMBER	ACTIONS
1	290	gilles-villeneuve	mclaren	1	40	NULL	No	9	9	 
2	290	patrick-tambay	ensign	2	23	NULL	No	16	16	 
3	290	jean-pierre-jarier	penske	3	34	NULL	No	20	20	 
4	290	brett-lunger	mclaren	4	30	NULL	No	19	19	 
5	290	brian-henton	march	5	38	NULL	No	29	NULL	 
6	290	arturo-merzario	march	6	37	NULL	No	17	17	 
7	290	patrick-neve	march	7	27	NULL	No	26	26	 
8	290	emilio-de-villota	mclaren	8	36	NULL	No	30	NULL	 

NORMALIZATION

User Table

- ✓ **1NF:** Atomic columns; each row uniquely identified by ***id*** (plus UNIQUE ***email*, *username***).
- ✓ **2NF:** Single-attribute primary key
- ✓ **3NF:** Country details are not duplicated; only ***country_id*** is stored (avoids transitive dependencies).
- ✓ **BCNF:** Determinants like email and username are candidate keys (UNIQUE), so FDs satisfy BCNF.

```
1 CREATE TABLE "user" (
2     id          VARCHAR(100) PRIMARY KEY,
3     country_id  VARCHAR(100) REFERENCES country(id) ON DELETE CASCADE,
4     username    VARCHAR(100) NOT NULL UNIQUE,
5     password_hash TEXT        NOT NULL,
6     email       VARCHAR(255) NOT NULL UNIQUE,
7     email_verified BOOLEAN     NOT NULL DEFAULT FALSE,
8     email_verified_at TIMESTAMP,
9     email_verification_token VARCHAR(255),
10    verification_sent_at TIMESTAMP,
11    date_joined  TIMESTAMP   NOT NULL DEFAULT NOW(),
12    is_admin     BOOLEAN     NOT NULL DEFAULT FALSE
13 );
14
15 CREATE INDEX user_username_idx ON "user"(username);
16 CREATE INDEX user_country_id_idx ON "user"(country_id);
```

NORMALIZATION

Circuit Table

- 1NF:** Atomic columns
- 2NF:** Single-attribute primary key (***id***) \Rightarrow no partial dependencies.
- 3NF:** No country name redundancy; ***country_id*** references country (no transitive dependency inside the table).
- BCNF:** Core dependencies are key-based (***id*** determines circuit attributes).

```
1 CREATE TABLE circuit (
2     id          VARCHAR(100) NOT NULL,
3     name        VARCHAR(100) NOT NULL,
4     full_name   VARCHAR(100) NOT NULL,
5     previous_names VARCHAR(255),
6     type        VARCHAR(6)  NOT NULL,
7     direction   VARCHAR(14) NOT NULL,
8     place_name  VARCHAR(100) NOT NULL,
9     country_id  VARCHAR(100) NOT NULL,
10    latitude    DECIMAL(10,6) NOT NULL,
11    longitude   DECIMAL(10,6) NOT NULL,
12    length      DECIMAL(6,3) NOT NULL,
13    turns       INT        NOT NULL,
14    total_races_held INT        NOT NULL,
15    PRIMARY KEY (id),
16    FOREIGN KEY (country_id) REFERENCES country (id) ON DELETE CASCADE
17 );
18
19 CREATE INDEX circuit_name_idx ON circuit(name);
20 CREATE INDEX circuit_country_id_idx ON circuit(country_id);
```

NORMALIZATION

Race Table

- ✓ **1NF:** Atomic race attributes; each race is one row.
- ✓ **2NF:** Single-attribute primary key (***id***) ⇒ no partial dependencies.
- ✓ **3NF:** Circuit details are not repeated in race; only ***circuit_id*** is stored.
- ✓ **BCNF:** ***id*** is a key, and (year, round) is also enforced as UNIQUE (candidate key), so determinants are keys.

```
1 CREATE TABLE race (
2     id          INT PRIMARY KEY,
3     circuit_id VARCHAR(100) NOT NULL REFERENCES circuit(id),
4     year        INT        NOT NULL,
5     round       INT        NOT NULL,
6     date        DATE      NOT NULL,
7     official_name VARCHAR(100) NOT NULL,
8     qualifying_format VARCHAR(20) NOT NULL,
9     laps         INT        NOT NULL,
10    qualifying_date DATE,
11    is_real      BOOLEAN   DEFAULT TRUE,
12    user_id     VARCHAR(100) DEFAULT NULL,
13
14    FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE,
15    UNIQUE (year, round)
16 );
17
18 CREATE INDEX race_year_idx      ON race(year);
19 CREATE INDEX race_circuit_id_idx ON race(circuit_id);
```

NORMALIZATION

Race Constructor Standing Table

- ✓ **1NF:** Atomic columns; each row is one constructor's standing in one race.
- ✓ **2NF:** Composite PK (***race_id, constructor_id***)
⇒ no partial dependencies.
- ✓ **3NF:** Non-key attributes are fully dependent on the key; no transitive dependencies.
- ✓ **BCNF:** Determinants are candidate keys (the composite primary key).

```
● ● ●
1 CREATE TABLE race_constructor_standing (
2     race_id          INT      NOT NULL REFERENCES race(id) ON DELETE CASCADE,
3     constructor_id   VARCHAR(100) NOT NULL REFERENCES constructor(id) ON DELETE CASCADE,
4     position_number  INT,
5     points           DECIMAL(8,2) NOT NULL,
6
7     PRIMARY KEY (race_id, constructor_id)
8 );
9
10 CREATE INDEX rcst_race_id_idx ON race_constructor_standing(race_id);
11
```

NORMALIZATION

Constructor Table

- ✓ **1NF:** Attributes are stored as atomic values.
- ✓ **2NF:** Single-attribute primary key (***id***) ⇒ no partial dependencies.
- ✓ **3NF:** Descriptive constructor data depends only on ***id***; **country** is referenced via FK (no duplicated country attributes).

```
1 CREATE TABLE constructor (
2     id          VARCHAR(100) NOT NULL,
3     user_id    VARCHAR(100) DEFAULT NULL,
4     country_id VARCHAR(100) NOT NULL,
5     name        VARCHAR(100) NOT NULL,
6     full_name   VARCHAR(100) NOT NULL,
7     best_championship_position INTEGER,
8     total_championship_wins    INTEGER NOT NULL,
9     total_race_starts          INTEGER NOT NULL,
10    total_podiums             INTEGER NOT NULL,
11    total_points              DECIMAL(8,2) NOT NULL,
12    total_pole_positions      INTEGER NOT NULL,
13    is_real                  BOOLEAN DEFAULT TRUE,
14    PRIMARY KEY (id),
15    FOREIGN KEY (country_id) REFERENCES country(id) ON DELETE CASCADE,
16    FOREIGN KEY (user_id)    REFERENCES "user"(id) ON DELETE CASCADE
17 );
18
19 CREATE INDEX constructor_name_idx ON constructor(name);
20 CREATE INDEX constructor_country_id_idx ON constructor(country_id);
```

NORMALIZATION

Race Data Table

- ✓ **1NF:** One row represents one driver's performance in one race; attributes are atomic.
- ✓ **2NF:** Result attributes depend on the whole business concept (race, driver) rather than just one side.
- ✓ **3NF:** Race metadata is in race, driver metadata in driver, constructor metadata in constructor (no transitive dependencies inside race_data).

```
1 CREATE TABLE race_data (
2     id          BIGSERIAL   PRIMARY KEY,
3     race_id    INT        NOT NULL,           -- FK race(id)
4     driver_id  VARCHAR(100) NOT NULL,          -- FK driver(id)
5     constructor_id  VARCHAR(100) NOT NULL,      -- FK constructor(id)
6     user_id    VARCHAR(100) DEFAULT NULL,       -- FK user(id)
7     position_display_order  INT        NOT NULL,
8     driver_number  VARCHAR(3)  NOT NULL,
9     race_points   DECIMAL(8,2),
10    race_pole_position BOOLEAN,
11    race_qualification_position_number INT,
12    race_grid_position_number INT,
13    is_real      BOOLEAN   DEFAULT FALSE,
14    created_at   TIMESTAMP  NOT NULL DEFAULT NOW(),
15
16    FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE,
17    FOREIGN KEY (constructor_id) REFERENCES constructor (id) ON DELETE CASCADE,
18    FOREIGN KEY (driver_id) REFERENCES driver      (id) ON DELETE CASCADE,
19    FOREIGN KEY (race_id) REFERENCES race        (id) ON DELETE CASCADE
20 );
21
22
23 CREATE INDEX rcda_race_id_idx      ON race_data(race_id);
24 CREATE INDEX rcda_position_display_order_idx ON race_data(position_display_order);
25 CREATE INDEX rcda_driver_id_idx      ON race_data(driver_id);
26 CREATE INDEX rcda_constructor_id_idx ON race_data(constructor_id);
27
```

NORMALIZATION

Country Table

- ✓ **1NF:** All attributes are atomic (no lists / repeating groups).
- ✓ **2NF:** Single-attribute primary key (***id***) \Rightarrow no partial dependencies.
- ✓ **3NF:** No non-key attribute determines another non-key attribute (country facts stored only here).
- ✓ **BCNF:** Every functional dependency is determined by a candidate key (***id*** or ***alpha3_code***).

```
● ○ ●  
1 CREATE TABLE country (  
2   id          VARCHAR(100) PRIMARY KEY,  
3   alpha3_code  VARCHAR(3)  NOT NULL UNIQUE,  
4   name        VARCHAR(100) NOT NULL  
5 );  
6
```

NORMALIZATION

Driver Table

- ✓ **1NF:** Atomic attributes per driver row.
- ✓ **2NF:** Single-attribute primary key (***id***) ⇒ no partial dependencies.
- ✓ **3NF:** Country-related facts are not stored as text; they are referenced via **country_id** FKS.

```
1 CREATE TABLE driver (
2     id          VARCHAR(100) PRIMARY KEY,
3     name        VARCHAR(100) NOT NULL,
4     first_name  VARCHAR(100),
5     last_name   VARCHAR(100),
6     full_name   VARCHAR(100) NOT NULL,
7     abbreviation VARCHAR(10),
8     permanent_number INT,
9     gender      VARCHAR(20),
10    date_of_birth DATE,
11    date_of_death DATE,
12    place_of_birth VARCHAR(100),
13    country_of_birth_country_id VARCHAR(100) NOT NULL,
14    nationality_country_id VARCHAR(100) NOT NULL,
15    best_championship_position INT,
16    best_race_result INT,
17    total_championship_wins INT      NOT NULL,
18    total_race_starts INT      NOT NULL,
19    total_race_wins INT      NOT NULL,
20    total_race_laps INT      NOT NULL,
21    total_podiums INT      NOT NULL,
22    total_points DECIMAL(8,2) NOT NULL,
23    total_pole_positions INT      NOT NULL,
24    is_real      BOOLEAN      DEFAULT TRUE,
25    user_id      VARCHAR(100) DEFAULT NULL,
26
27    FOREIGN KEY (country_of_birth_country_id) REFERENCES country(id) ON DELETE CASCADE,
28    FOREIGN KEY (nationality_country_id)      REFERENCES country(id) ON DELETE CASCADE,
29    FOREIGN KEY (user_id)      REFERENCES "user"(id) ON DELETE CASCADE
30 );
31
32 CREATE INDEX driver_name_idx ON driver(full_name);
33 CREATE INDEX driver_abbreviation_idx ON driver(abbreviation);
34 CREATE INDEX drv_nationality_idx      ON driver(nationality_country_id);
35 CREATE INDEX drv_country_of_birth_idx  ON driver(country_of_birth_country_id);
36
37
```

NORMALIZATION

Race Driver Standing Table

- ✓ **1NF:** Atomic values; no repeating groups.
- ✓ **2NF:** Composite PK (***race_id, driver_id***) ⇒ non-key columns depend on the full key.
- ✓ **3NF:** No non-key-to-non-key dependencies (position/points depend only on the key).
- ✓ **BCNF:** The only determinants are the candidate key itself (***race_id, driver_id***).

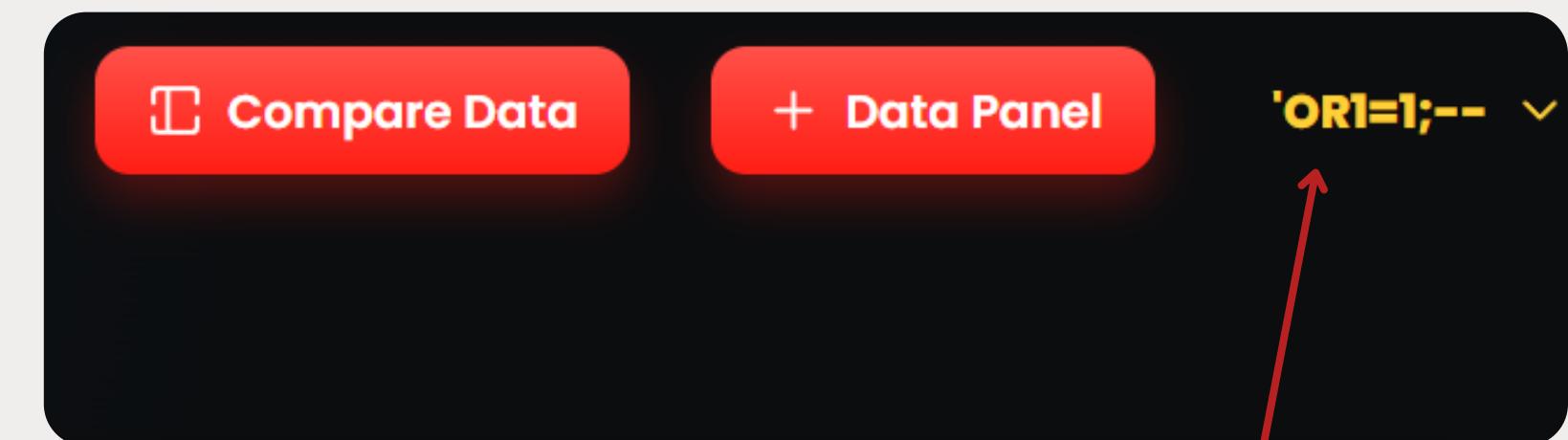
```
1 CREATE TABLE race_driver_standing (
2     race_id          INT      NOT NULL,
3     driver_id        VARCHAR(100) NOT NULL,
4
5     position_number  INT      NOT NULL,
6     points           DECIMAL(8,2) NOT NULL,
7
8     PRIMARY KEY (race_id, driver_id),
9
10    FOREIGN KEY (race_id) REFERENCES race(id) ON DELETE CASCADE,
11    FOREIGN KEY (driver_id) REFERENCES driver(id) ON DELETE CASCADE
12 );
13
14 CREATE INDEX rds_race_id_idx ON race_driver_standing(race_id);
15 CREATE INDEX rds_driver_id_idx ON race_driver_standing(driver_id);
16
```

SECURITY TESTING

The screenshot shows a dark-themed web application for 'F1 Race Analytics'. In the top right corner, there are buttons for 'Compare Data' and '+ Data Panel', and a dropdown menu set to 'eeg'. Below this, a section titled 'My Drivers' displays one custom record found. A red '+' button labeled '+ Add New Driver' is visible. The main table has columns: FIRST NAME, LAST NAME, NAME, ABBREVIATION, PERMANENT NUMBER, GENDER, DATE OF BIRTH, DATE OF DEATH, PLACE OF BIRTH, COUNTRY OF BIRTH COUNTRY ID, NATIONALITY COUNTRY ID, BEST CHAMPIONSHIP POSITION, and BEST RACE RESULT. The first row of data contains the following values:

FIRST NAME	LAST NAME	NAME	ABBREVIATION	PERMANENT NUMBER	GENDER	DATE OF BIRTH	DATE OF DEATH	PLACE OF BIRTH	COUNTRY OF BIRTH COUNTRY ID	NATIONALITY COUNTRY ID	BEST CHAMPIONSHIP POSITION	BEST RACE RESULT
<script>alert('XSS')</script>	-	<script>alert('XSS')</script>	-	-	-	2004-11-03	-	-	bangladesh	azerbaijan	-	-

Eq: XSS(Cross-Site Scripting) bypassed.

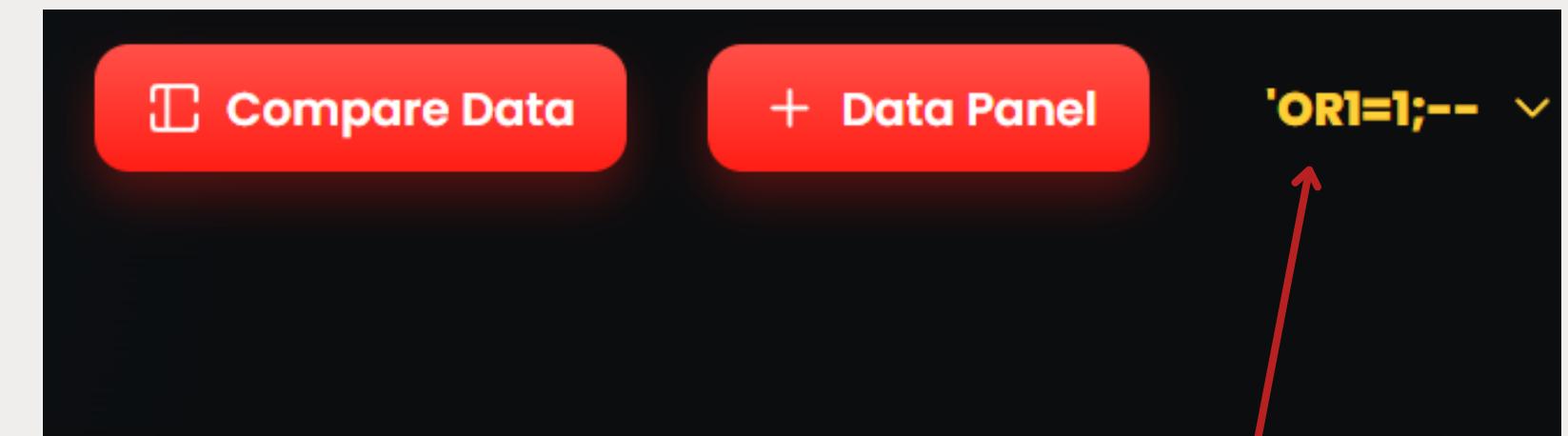


Eq: SQL Injection bypassed.

SECURITY TESTING

FIRST NAME	LAST NAME	NAME	ABBREVIATION	PERMANENT NUMBER	GENDER	DATE OF BIRTH	DATE OF DEATH	PLACE OF BIRTH	COUNTRY OF BIRTH COUNTRY ID	NATIONALITY COUNTRY ID	BEST CHAMPIONSHIP POSITION	BEST RACE RESULT
<script>alert('XSS')</script>	-	<script>alert('XSS')</script>	-	-	-	2004-11-03	-	-	bangladesh	azerbaijan	-	-
-	-	<script>alert('XSS')</script>	-	-	-	-	-	-	-	-	-	-

Ex: XSS(Cross-Site Scripting) bypassed.



Ex: SQL Injection bypassed.

CONCLUSION

- This project presents a comprehensive Formula 1 analytics platform that combines historical race data with user-defined entries, enabling a richer and more flexible exploration of motorsport information.
- By employing a well-structured relational database design, the system ensures data consistency, efficient querying, and scalable management of complex entities such as drivers, constructors, races, and circuits.
- Overall, the developed system provides a solid foundation for future enhancements, including advanced analytics, visualization techniques, and expanded data sources.

Thank you for your attention.

