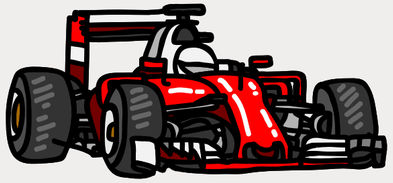
Database Sytems Course Term Project

F1 RACE ANALYTICS

Ege Demir - 150220007
İsmail Yeşilyurt - 150220009
Nurettin Alper Kuzu - 150230001
Sıla Keküllüoğlu - 150210084
Turan İnceöz - 150220097

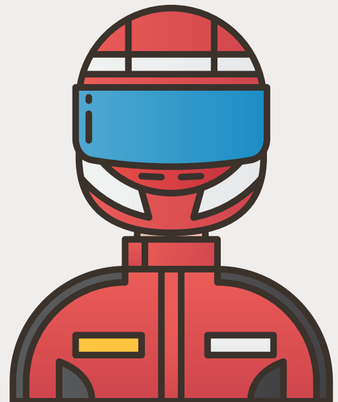
OUTLINE

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

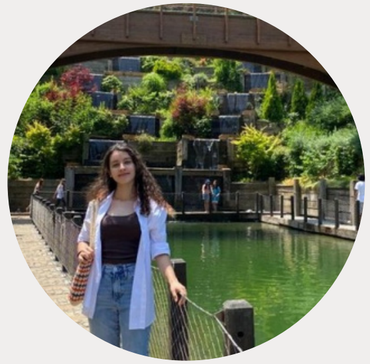
Frontend and Authorization

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 (f1db 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, 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, 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, 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

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

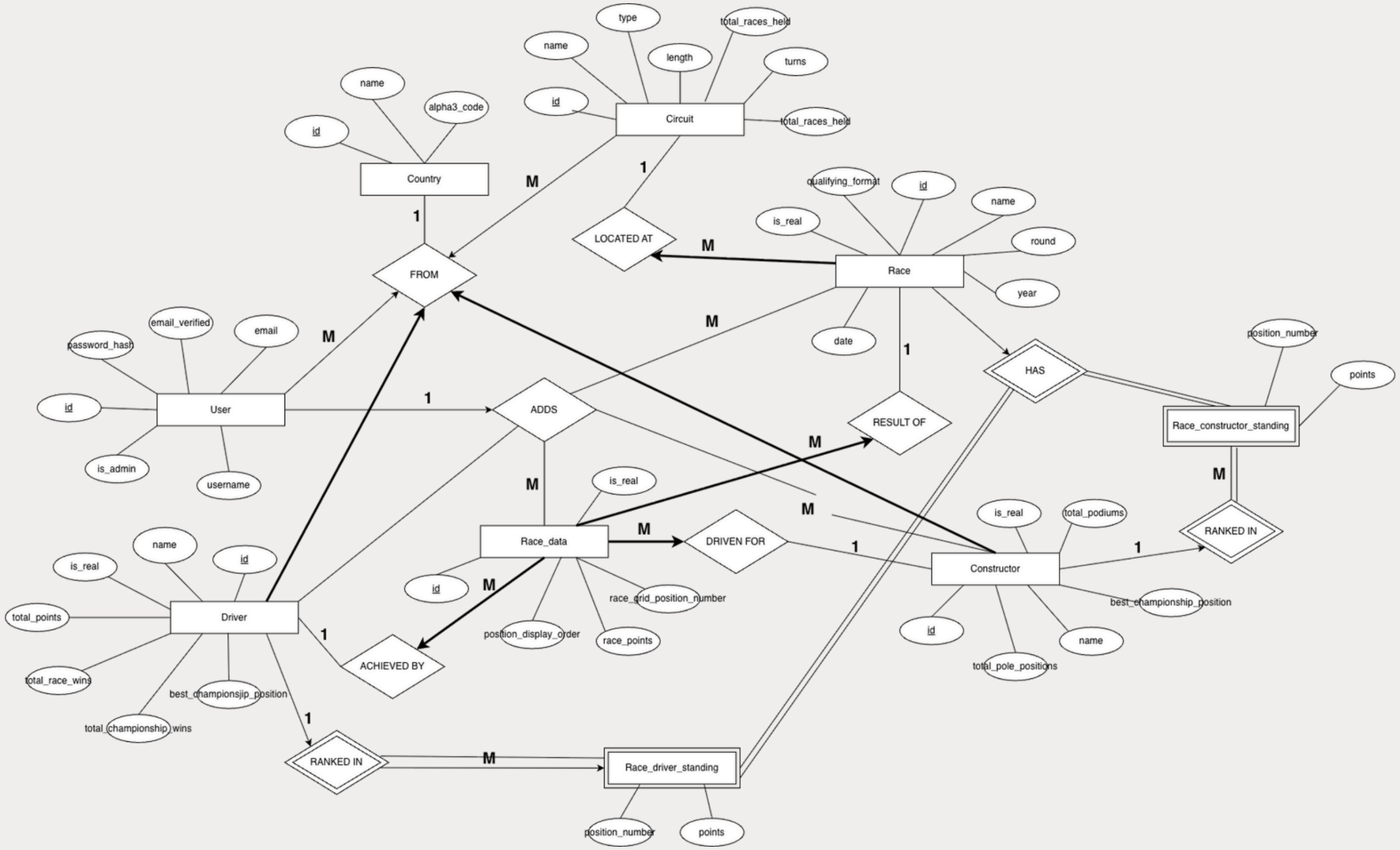
Race
PK: id
FK: user_id, country_id

Race_data
PK: id
FK: user_id, race_id,
driver_id, constructor_id

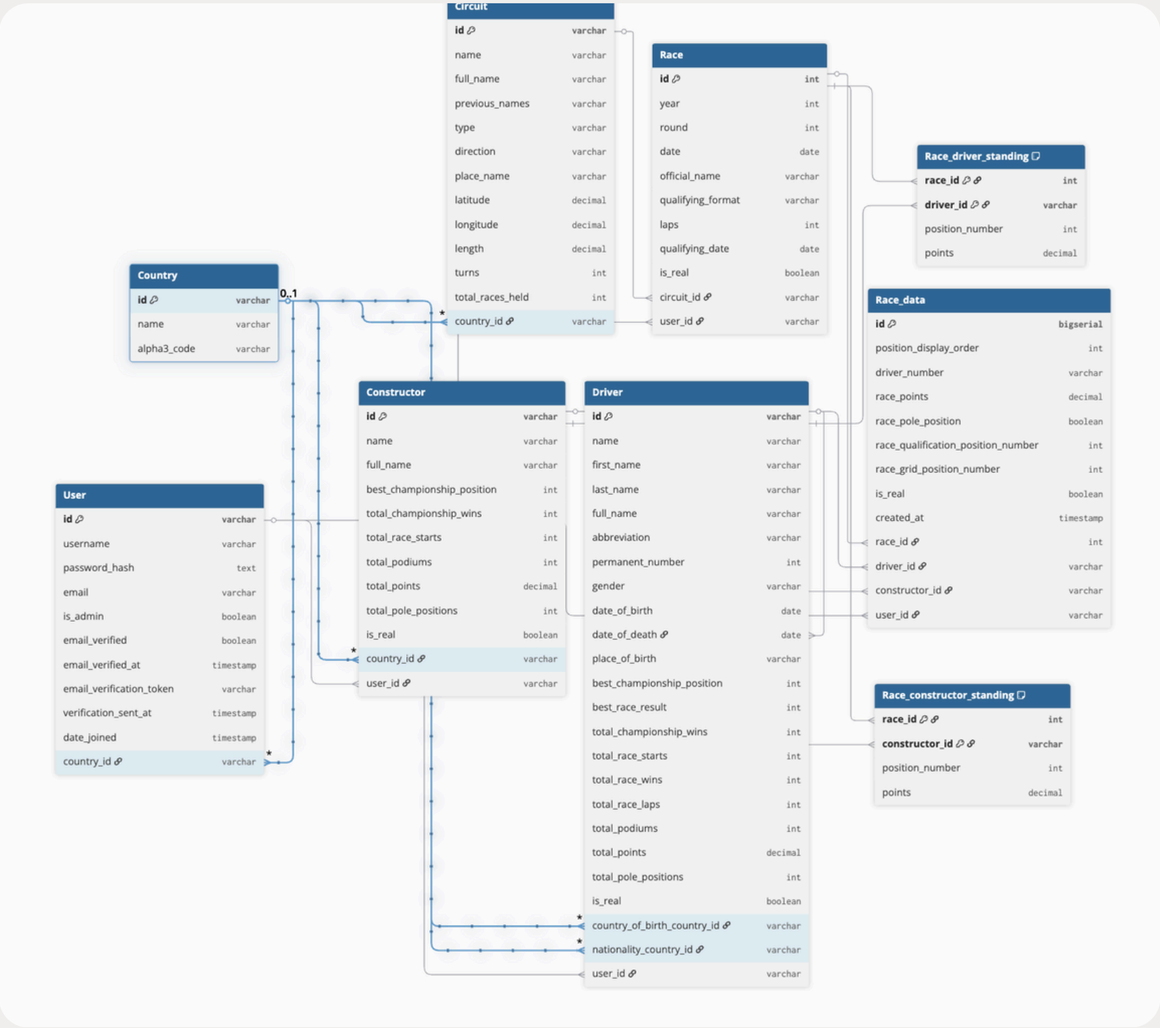
Race_driver_standing
PK: id
FK: race_id, driver_id

Race_constructor_standing
PK: id
FK: race_id, constructor_id

ER DIAGRAM


type
total_races_held
name
length
turns
id
Circuit
total_races_held
name
alpha3_code
id
Country
1
M
1
LOCATED AT
M
qualifying_format
id
name
is_real
round
Race
year
FROM
M
date
1
position_number
HAS
points
email_verified
email
password_hash
M
id
User
1
ADDS
RESULT OF
Race_constructor_standing
is_admin
username
M
M
M
M
is_real
M
is_real
total_podiums
RANKED IN
name
id
is_real
Race_data
DRIVEN FOR
1
1
Constructor
id
M
total_points
Driver
race_grid_position_number
best_championship_position
position_display_order
race_points
id
name
total_pole_positions
1
ACHIEVED BY
total_race_wins
best_championsjip_position
total_championship_wins
1
RANKED IN
M
Race_driver_standing
position_number
points

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

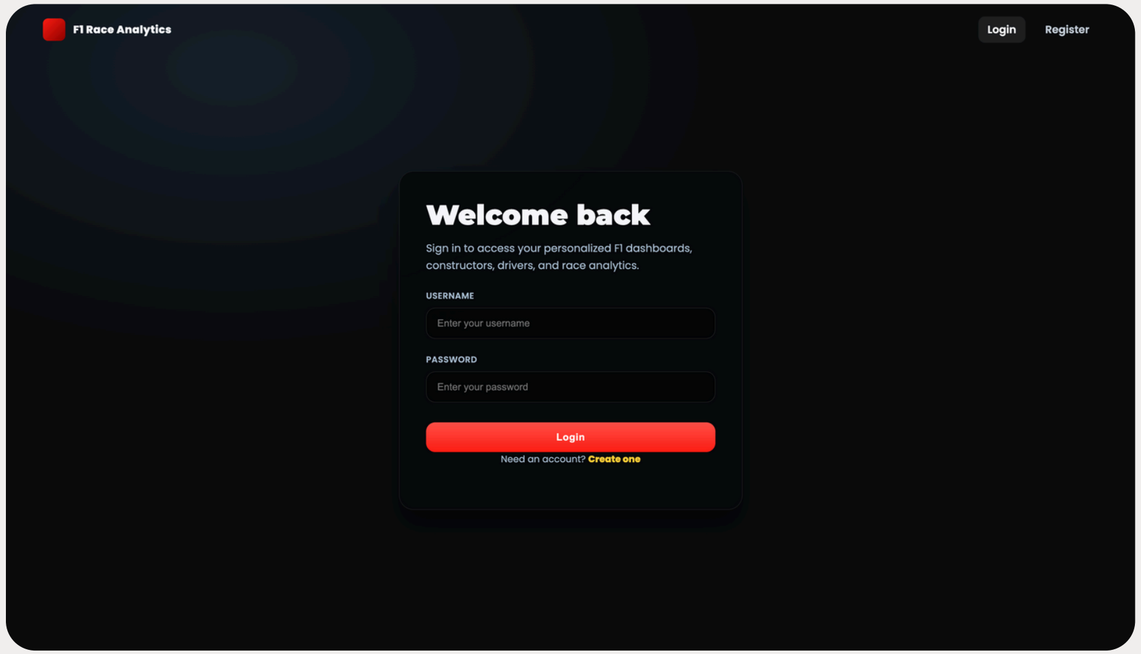
implemented by using foreign keys

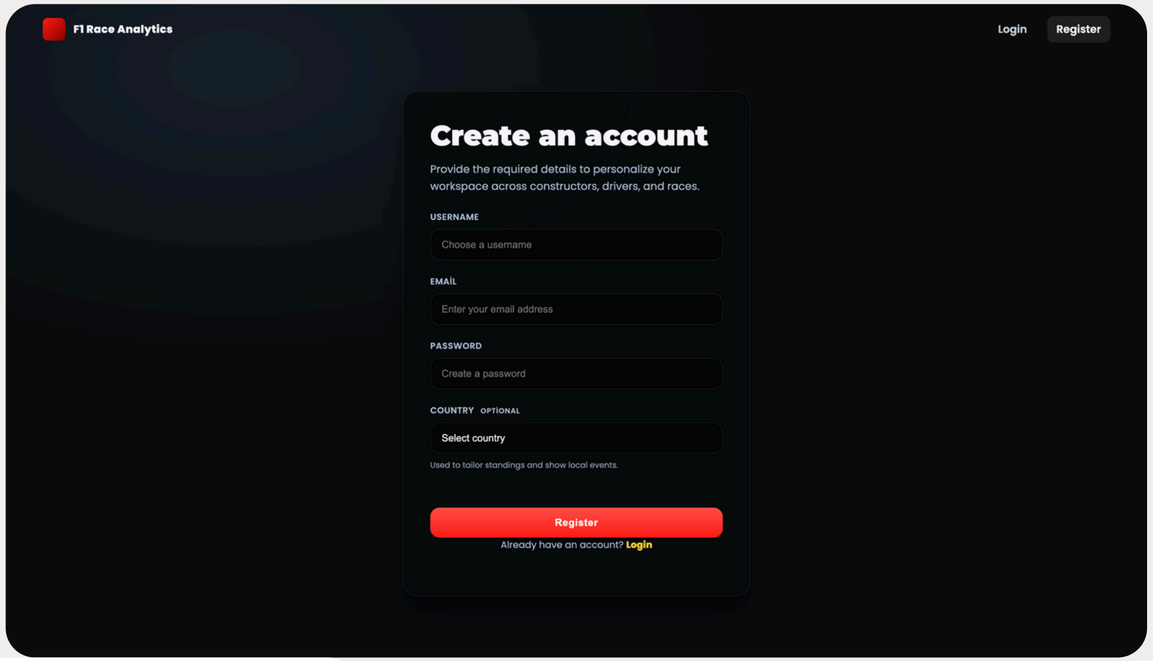
Many-to-Many Relationships

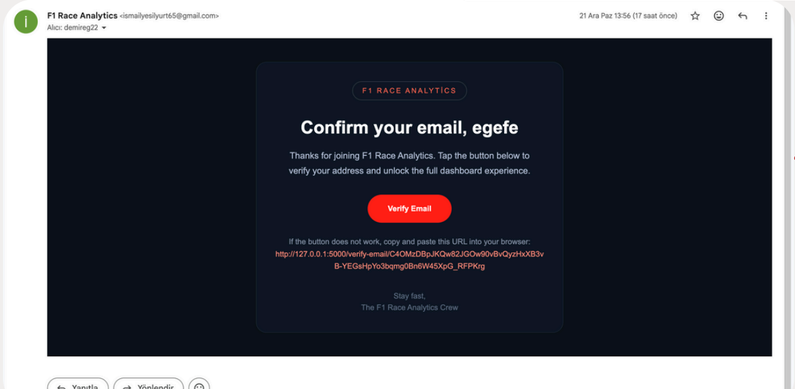
strong entity->relation with same primary key

strong entity->relation with same primary key

LOGIN AND REGISTER PAGES

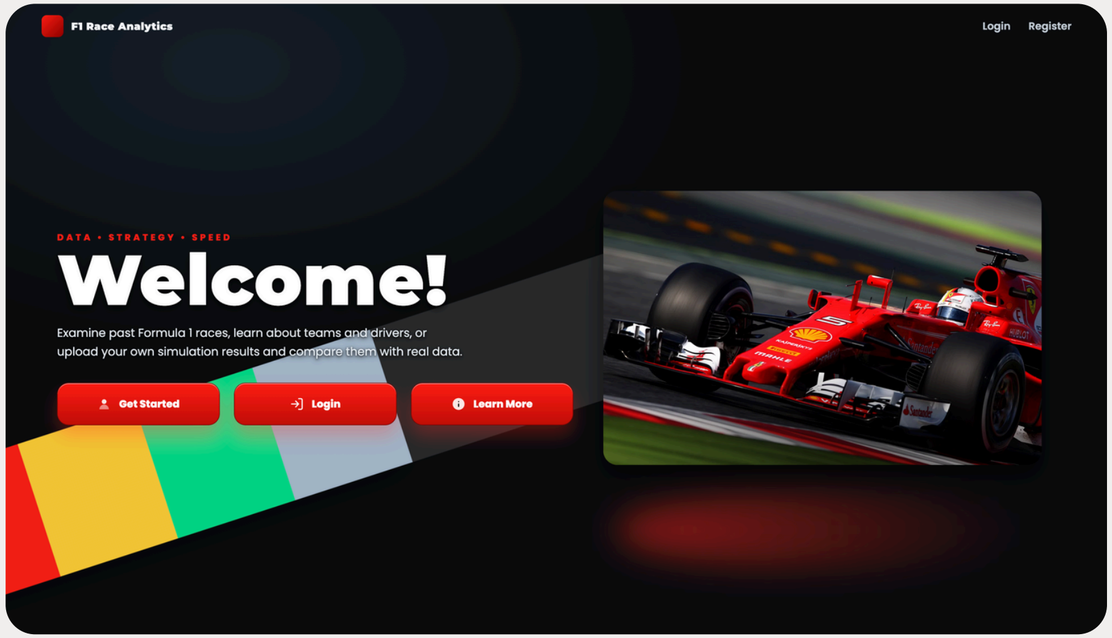


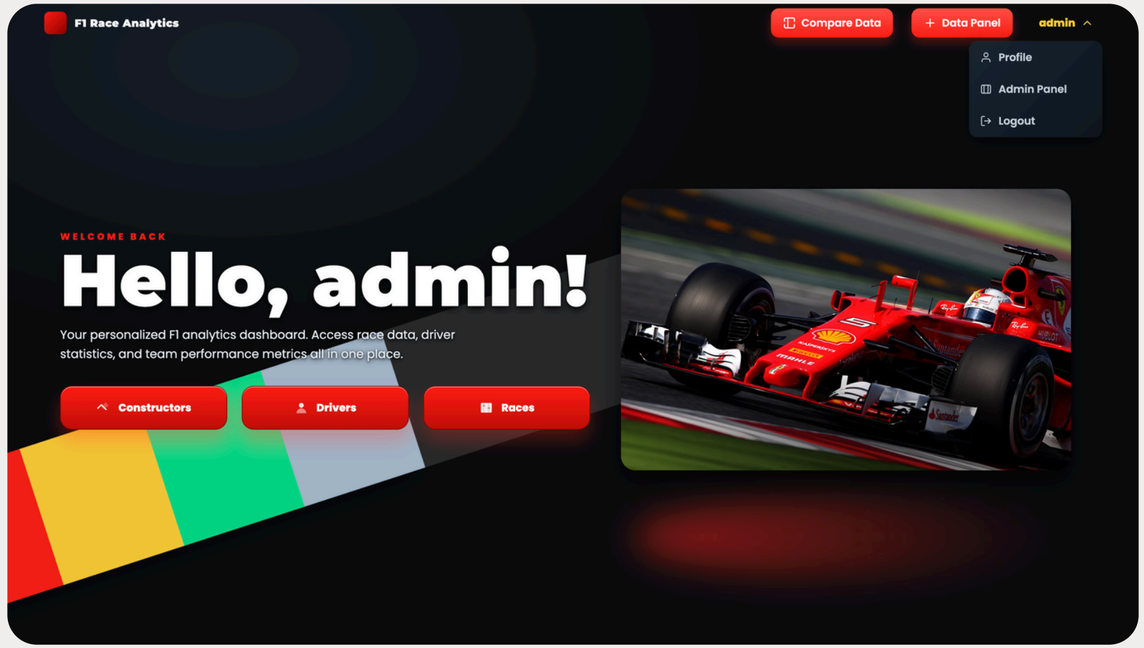




email verification

MAIN PAGE

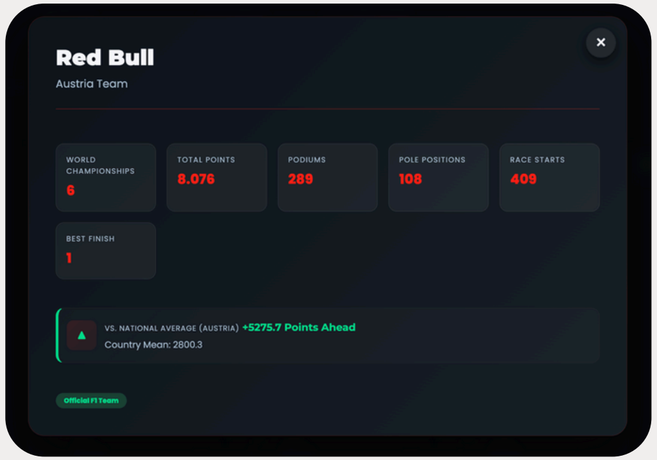


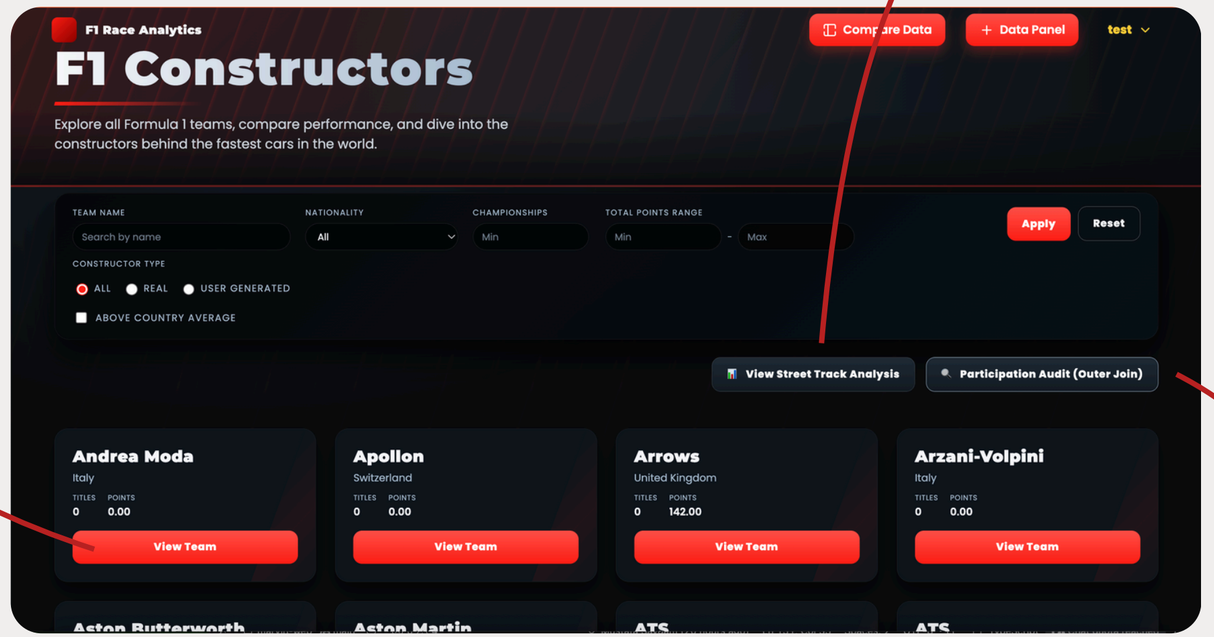


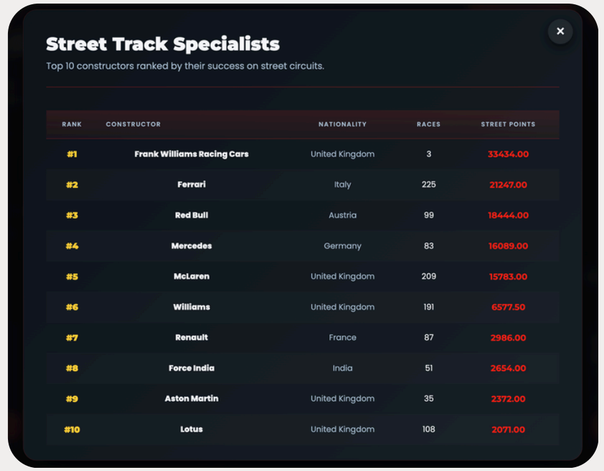
after login

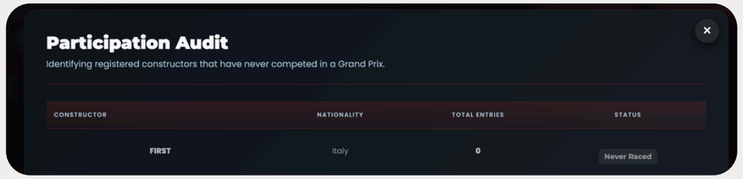
CONSTRUCTORS PAGE

Selections - READ operations

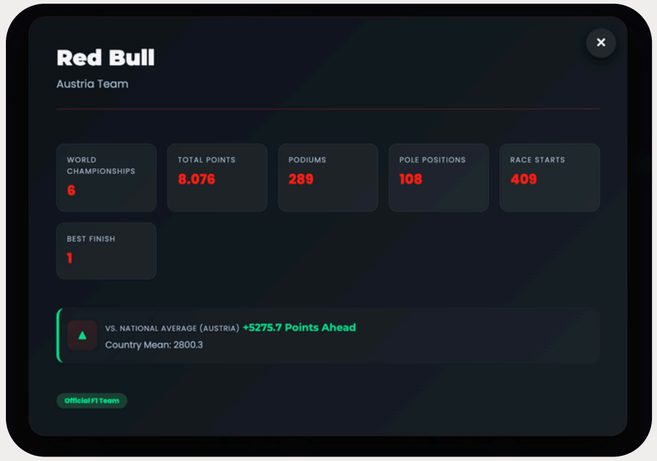


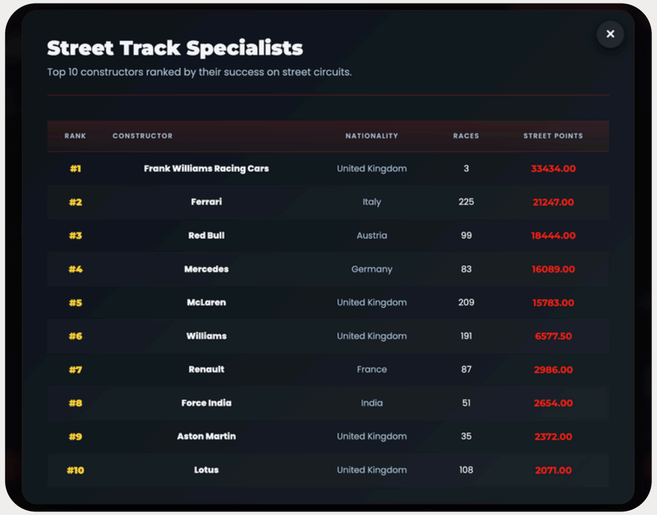






CONSTRUCTORS PAGE





database > queries > constructors_above_avg.sql

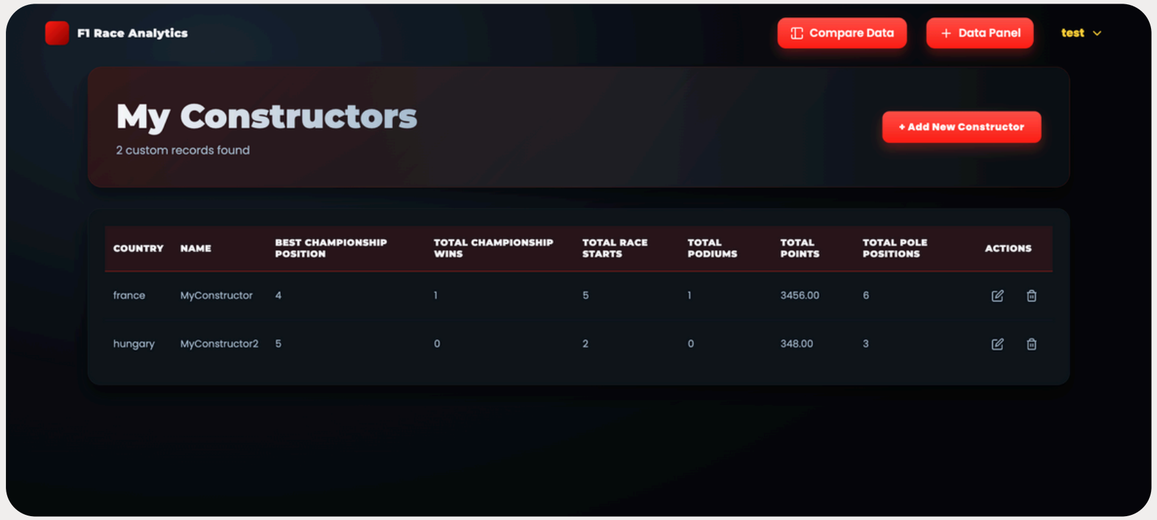
```sql
-- constructors with above average performance within their country
SELECT
    c.*,
    co.name AS nationality,
    -- calculate country average total points
    (SELECT AVG(c2.total_points)
     FROM constructor c2
     WHERE c2.country_id = c.country_id) AS country_avg,
    COUNT(*) OVER() as full_count
FROM constructor c
JOIN country co ON c.country_id = co.id
WHERE
    (%(name)s IS NULL OR c.full_name ILIKE '%%' || %(name)s || '%%')
    AND
    (%(nationality)s IS NULL OR co.name = %(nationality)s)
    AND
    (%(champs_min)s IS NULL OR c.total_championship_wins >= %(champs_min)s)
    AND
    (%(total_points_min)s IS NULL OR c.total_points >= %(total_points_min)s)
    AND
    (%(total_points_max)s IS NULL OR c.total_points <= %(total_points_max)s)
    AND
    (%(is_real)s IS NULL OR c.is_real = %(is_real)s)
-- grouping by constructor and country
GROUP BY c.id, co.name
-- only include constructors with above average total points in their country
HAVING c.total_points > (
    SELECT AVG(c3.total_points)
    FROM constructor c3
    WHERE c3.country_id = c.country_id
)
ORDER BY c.total_points DESC
LIMIT %(limit)s OFFSET %(offset)s;
```

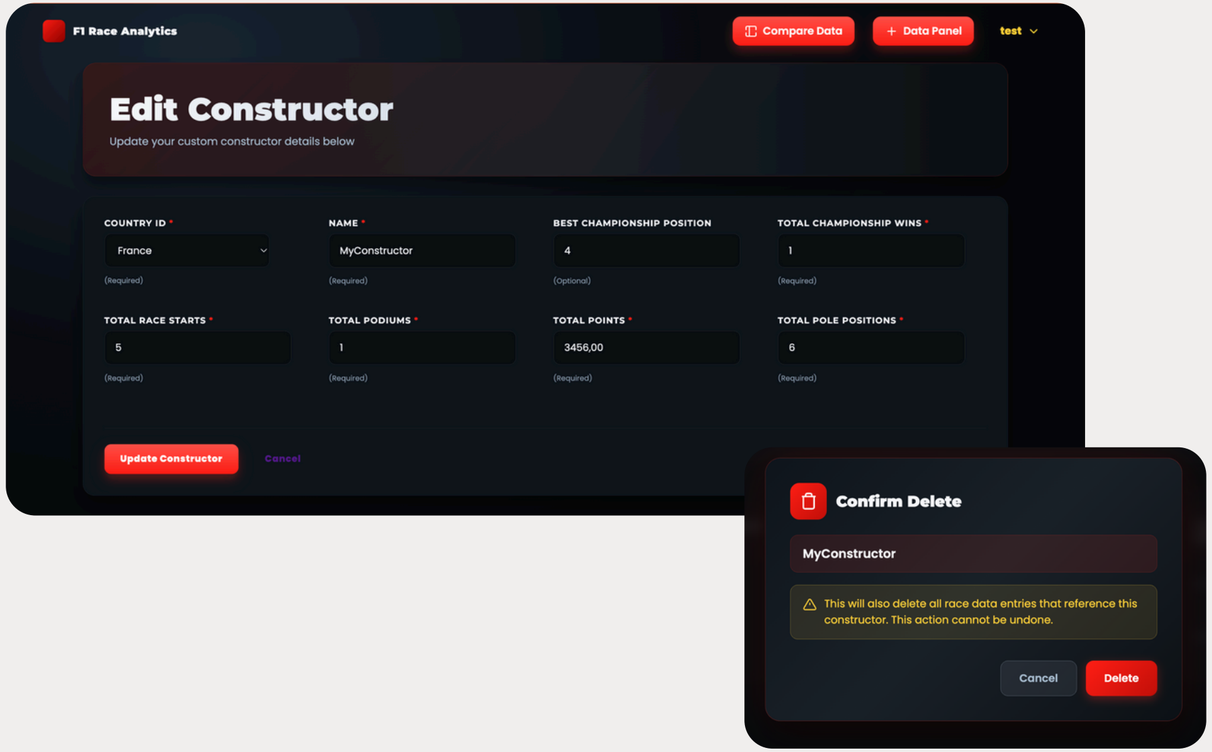
database > queries > constructor_track_performance.sql

```sql
SELECT
    -- Ranking based on total points in street track
    RANK() OVER (ORDER BY SUM(COALESCE(rd.race_points, 0)) DESC) as rank_position,
    c.name AS constructor_name,
    co.name AS nationality,
    cir.type AS track_type,
    COUNT(DISTINCT r.id) AS total_races,
    -- Making sure to handle NULL race points
    SUM(COALESCE(rd.race_points, 0)) AS total_points,
    -- Best race position on street tracks
    MIN(rd.position_display_order) AS best_position
FROM constructor c
JOIN country co ON c.country_id = co.id
JOIN race_data rd ON c.id = rd.constructor_id
JOIN race r ON rd.race_id = r.id
JOIN circuit cir ON r.circuit_id = cir.id
WHERE
    UPPER(cir.type) = 'STREET'
GROUP BY c.name, co.name, cir.type
-- Only include constructors with points on street tracks
HAVING SUM(COALESCE(rd.race_points, 0)) > 0
ORDER BY total_points DESC
LIMIT 10;
```

CONSTRUCTORS PAGE

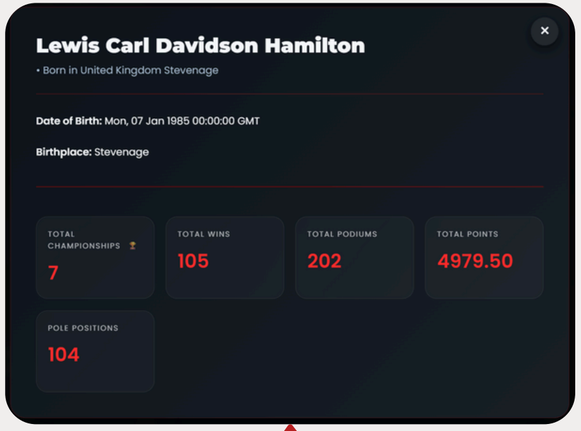
create, update, delete operations

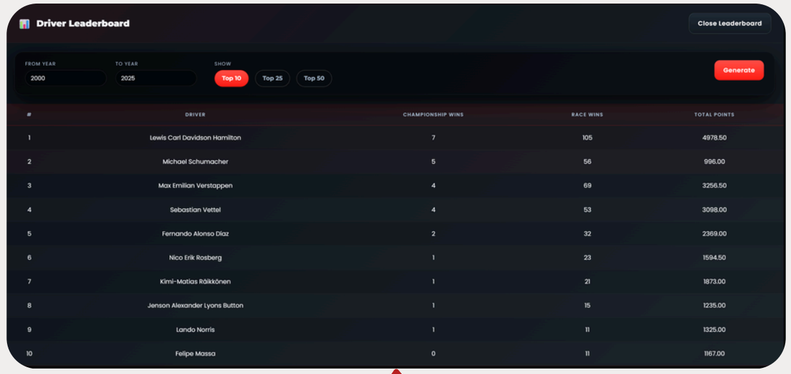


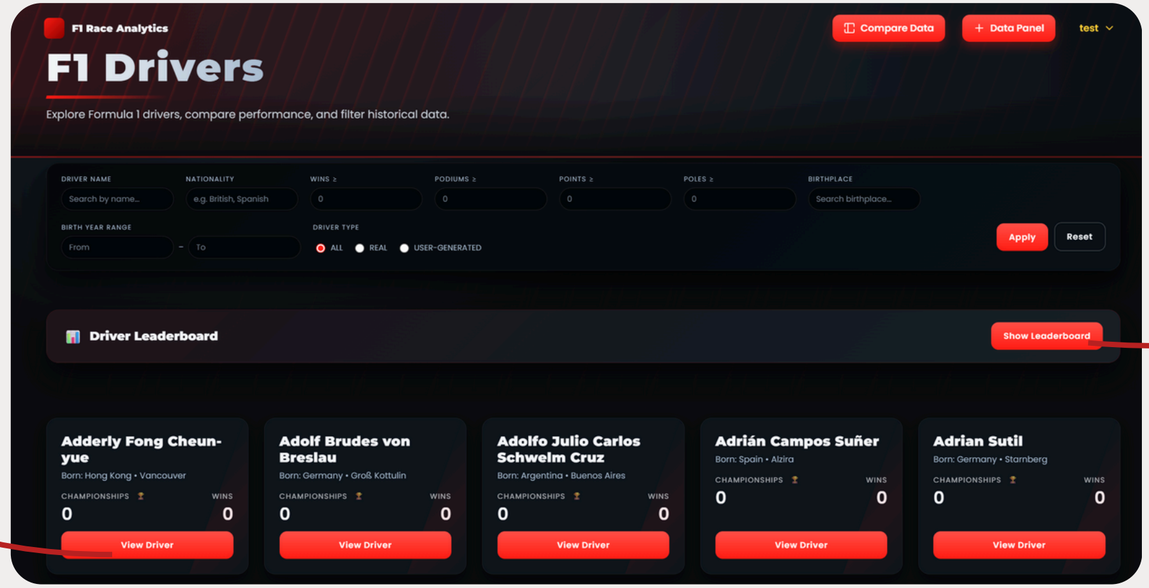


DRIVERS PAGE

Selections - READ operations

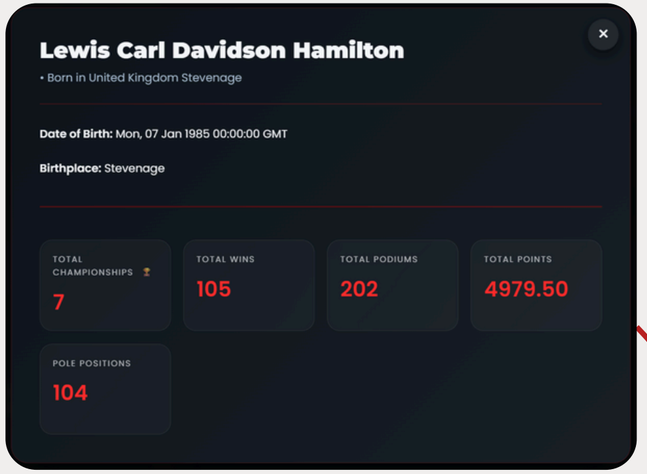


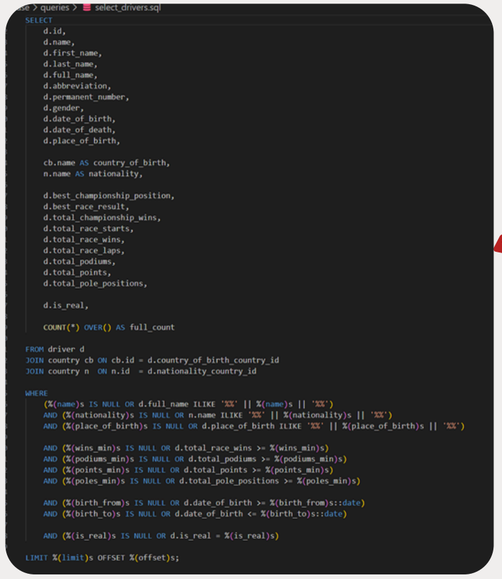


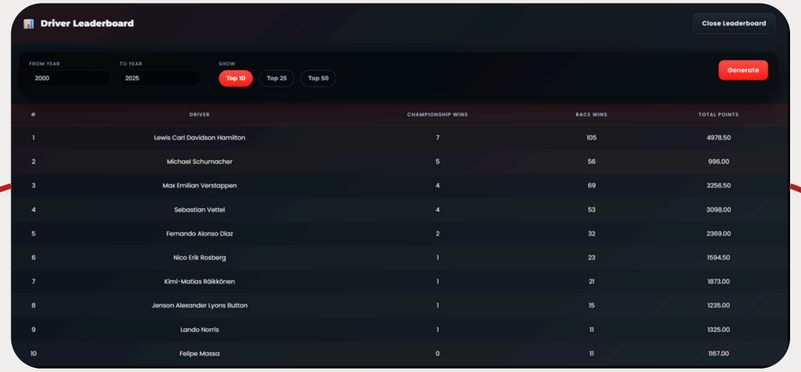


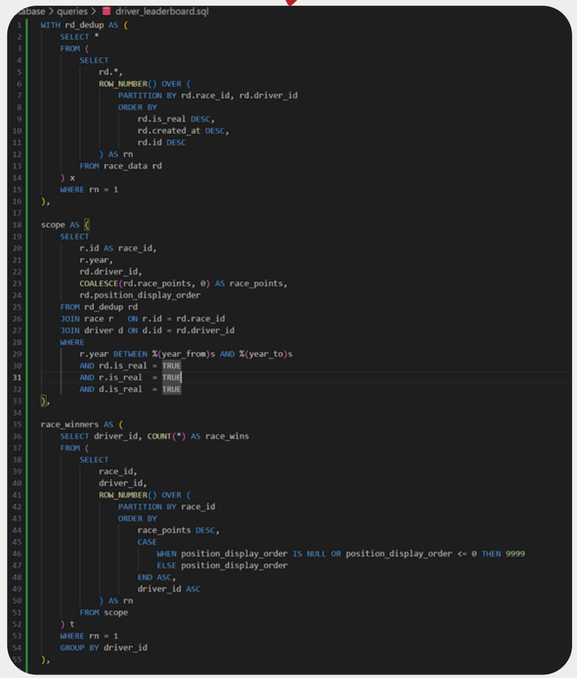
DRIVERS PAGE

Complex Queries



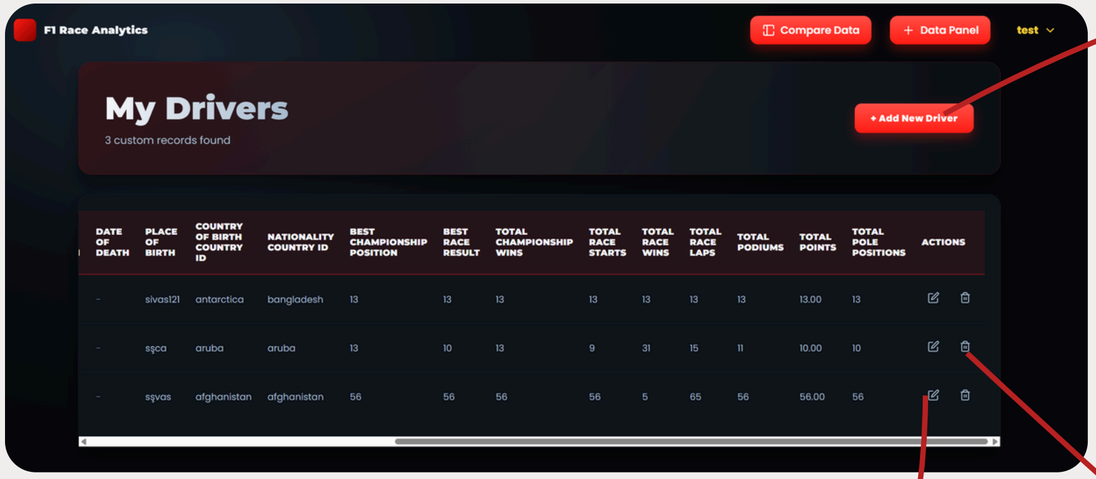


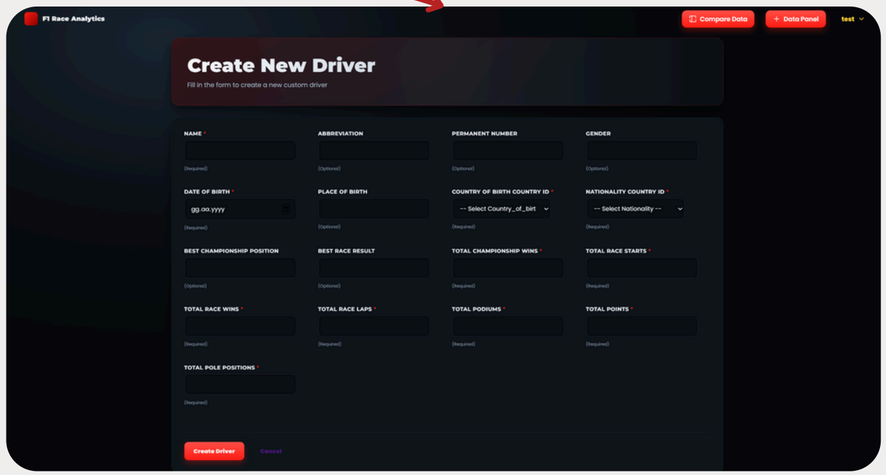


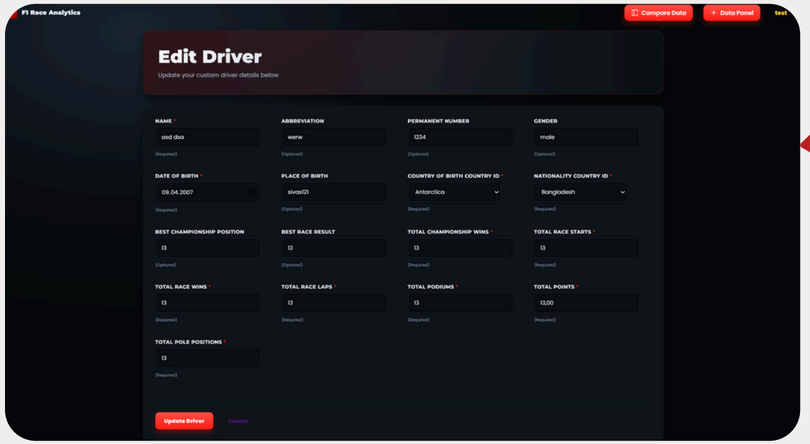


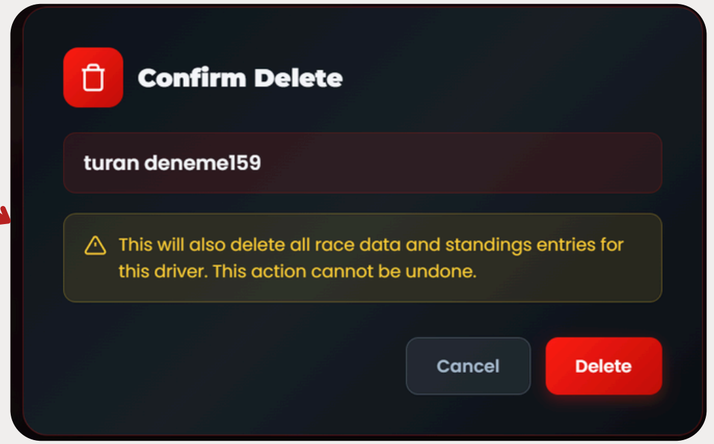
DRIVERS PAGE

create, update, delete operations



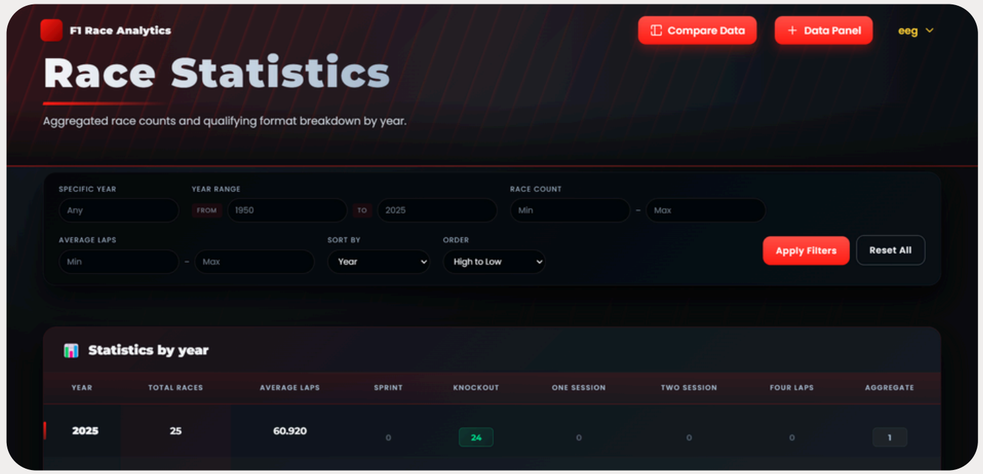


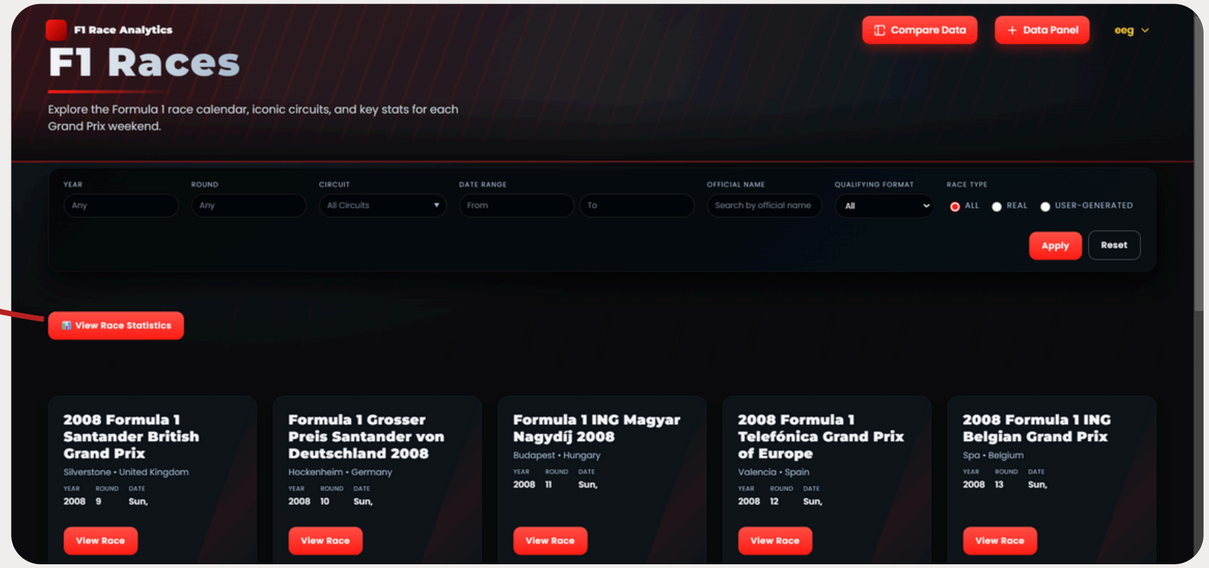




RACE PAGE

Selections - READ operations





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;
```

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

```
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;
```

For Control Mechanism

Formula 1 ING Magyar Nagydíj 2008

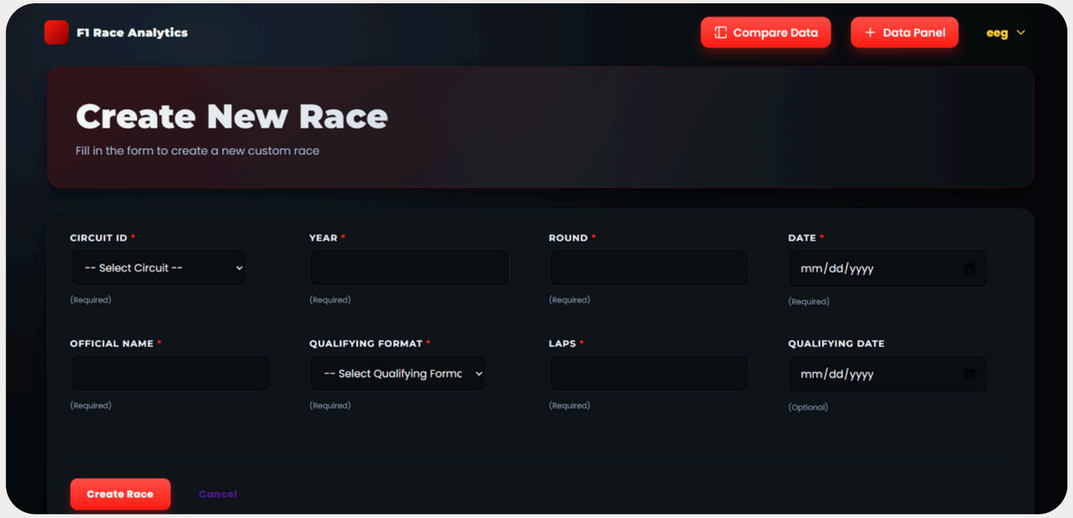
Year: **2008** Round: **11** Date: **Sun,** Format: **Knockout** Laps: **70**

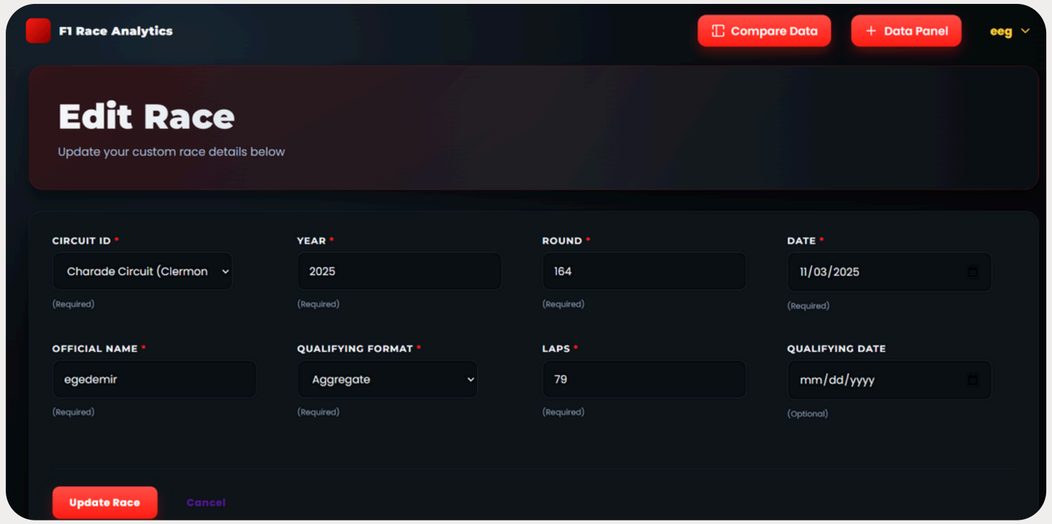
Race Results

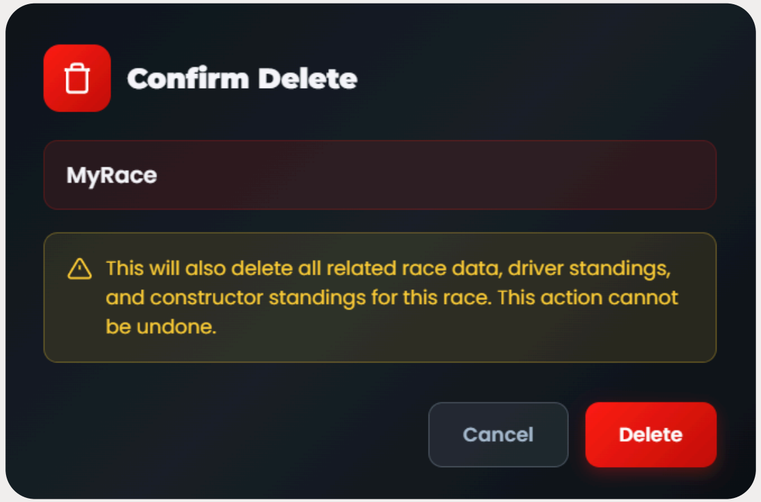
P	NO	DRIVER	CONSTRUCTOR	GRID	QUAL	POINTS
1	23	Heikki Johannes Kovalainen	McLaren Racing	2	2	10.00
2	12	Timo Glock	Toyota Racing	5	5	8.00
3	1	Kimi-Matias Räikkönen	Scuderia Ferrari	6	6	6.00
4	5	Fernando Alonso Díaz	Renault Sport Formula One Team	7	7	5.00
5	22	Lewis Carl Davidson Hamilton	McLaren Racing	1	1	4.00
6	6	Nelson Ângelo Tamsma 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	-

RACE PAGE

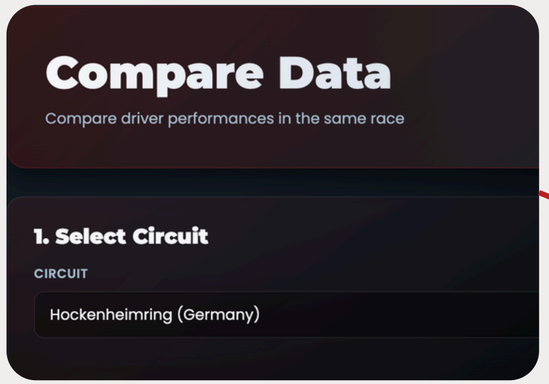
create, update, delete operations





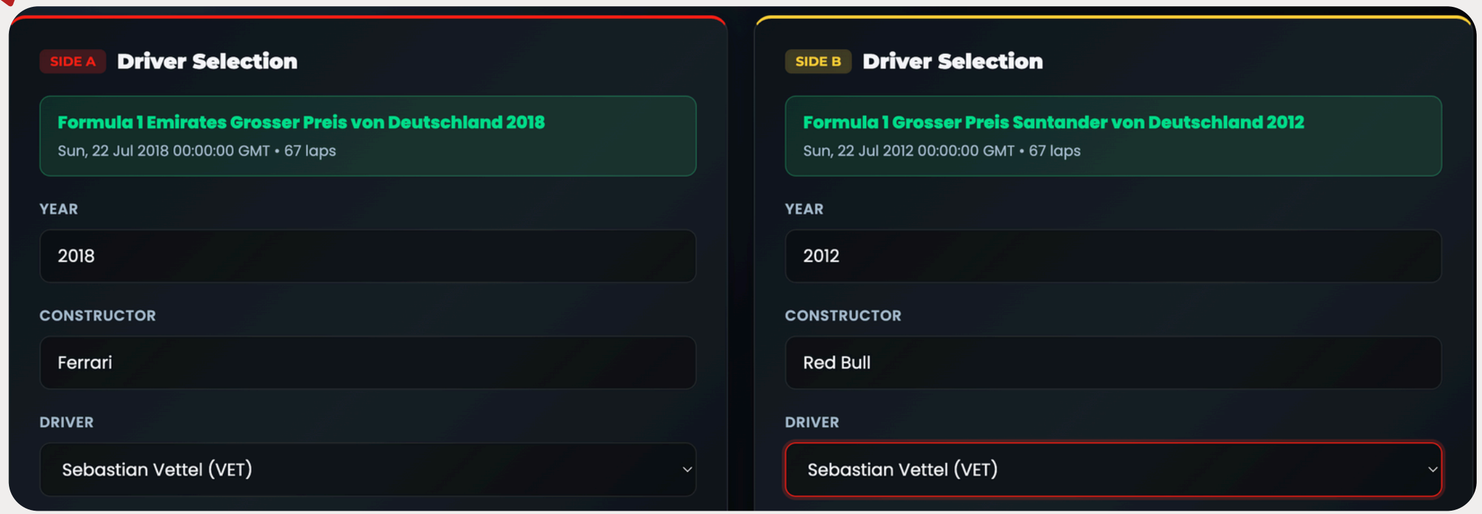


DATA COMPARISON



Only relevant data shown

Searchable dropdowns



DATA COMPARISON

Selections - READ operations/Complex Queries

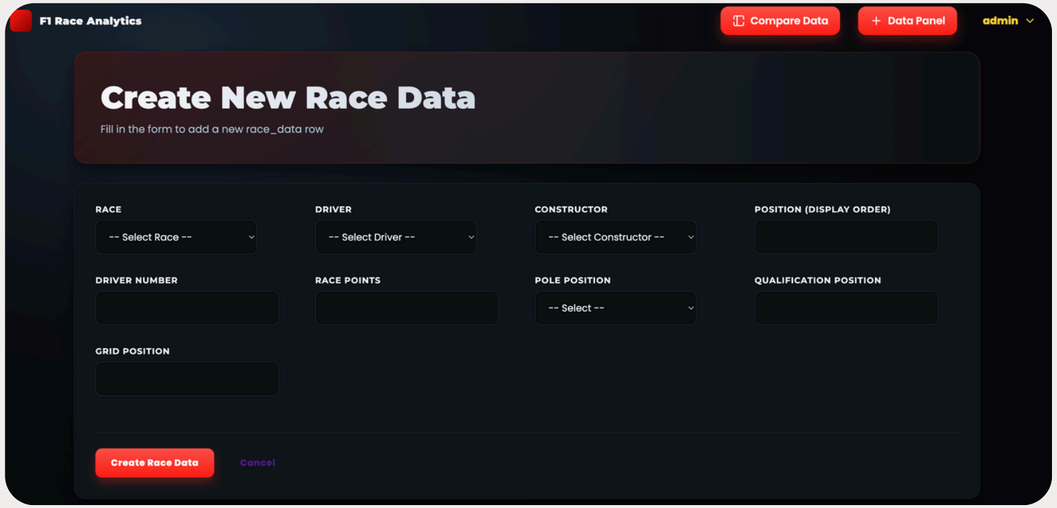
comparison info derived from

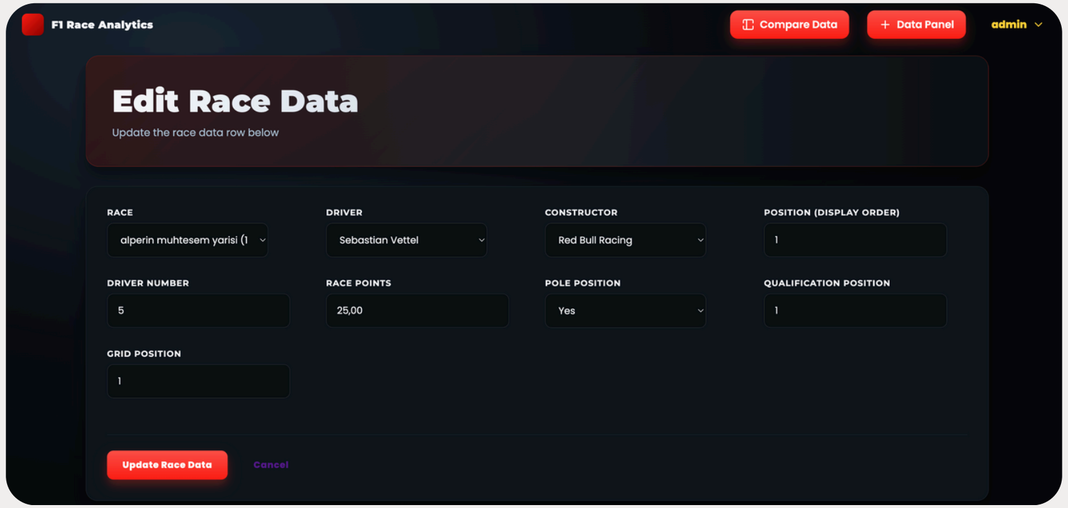
```sql
race_1_positions AS (
    SELECT rd.driver_id, rd.constructor_id, rd.race_points, rd.race_grid_position_number AS grid,
           r.year, c.name AS constructor, RANK() OVER (ORDER BY rd.position_display_order) AS finish
    FROM race_data rd
    INNER JOIN race r ON rd.race_id = r.id
    INNER JOIN constructor c ON rd.constructor_id = c.id
    WHERE rd.race_id = %s
),
driver_1_race AS (SELECT * FROM race_1_positions WHERE driver_id = %s),
driver_2_race AS (SELECT * FROM race_1_positions WHERE driver_id = %s),

driver_1_history AS (
    SELECT rd.driver_id, COUNT(DISTINCT rd.race_id) AS races,
           COUNT(CASE WHEN rd.position_display_order = 1 THEN 1 END) AS wins,
           COUNT(CASE WHEN rd.position_display_order <= 3 THEN 1 END) AS podiums,
           ROUND(AVG(rd.position_display_order)::numeric, 1) AS avg_finish,
           SUM(COALESCE(rd.race_points, 0)) AS total_points
    FROM race_data rd INNER JOIN race r ON rd.race_id = r.id
    WHERE rd.driver_id = %s AND r.circuit_id = %s
    GROUP BY rd.driver_id
),
driver_2_history AS (
    SELECT rd.driver_id, COUNT(DISTINCT rd.race_id) AS races,
           COUNT(CASE WHEN rd.position_display_order = 1 THEN 1 END) AS wins,
           COUNT(CASE WHEN rd.position_display_order <= 3 THEN 1 END) AS podiums,
           ROUND(AVG(rd.position_display_order)::numeric, 1) AS avg_finish,
           SUM(COALESCE(rd.race_points, 0)) AS total_points
    FROM race_data rd INNER JOIN race r ON rd.race_id = r.id
    WHERE rd.driver_id = %s AND r.circuit_id = %s
    GROUP BY rd.driver_id
),
```

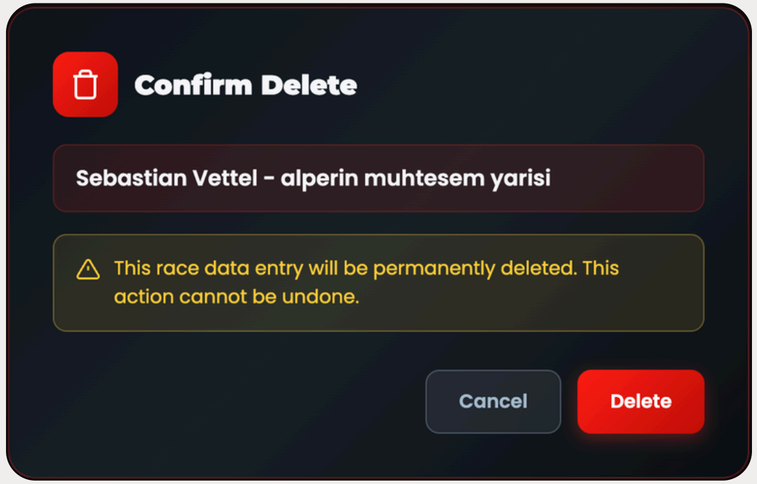
```sql
driver_1_season AS (
    SELECT rds.driver_id, rds.position_number AS champ_pos, rds.points,
           (SELECT COUNT(*) FROM race_data rd2 INNER JOIN race r2 ON rd2.race_id = r2.id
            WHERE rd2.driver_id = %s AND rd2.position_display_order = 1
            AND r2.year = (SELECT year FROM race WHERE id = %s)) AS wins
    FROM race_driver_standing rds WHERE rds.driver_id = %s AND rds.race_id = %s
),
driver_2_season AS (
    SELECT rds.driver_id, rds.position_number AS champ_pos, rds.points,
           (SELECT COUNT(*) FROM race_data rd2 INNER JOIN race r2 ON rd2.race_id = r2.id
            WHERE rd2.driver_id = %s AND rd2.position_display_order = 1
            AND r2.year = (SELECT year FROM race WHERE id = %s)) AS wins
    FROM race_driver_standing rds WHERE rds.driver_id = %s AND rds.race_id = %s
),

circuit_info AS (
    SELECT ci.full_name, co.name AS country, ci.length, ci.turns
    FROM circuit ci INNER JOIN country co ON ci.country_id = co.id WHERE ci.id = %s
)

SELECT ci.*, d1.full_name AS driver_1, d1r.finish AS d1_finish, d1h.wins AS d1_wins,
       d2.full_name AS driver_2, d2r.finish AS d2_finish, d2h.wins AS d2_wins
FROM circuit_info ci
CROSS JOIN driver_1_race d1r LEFT JOIN driver d1 ON d1r.driver_id = d1.id
CROSS JOIN driver_2_race d2r LEFT JOIN driver d2 ON d2r.driver_id = d2.id
LEFT JOIN driver_1_history d1h ON d1.id = d1h.driver_id
LEFT JOIN driver_2_history d2h ON d2.id = d2h.driver_id
LEFT JOIN driver_1_season d1s ON d1.id = d1s.driver_id
LEFT JOIN driver_2_season d2s ON d2.id = d2s.driver_id;
```

RACE DATA

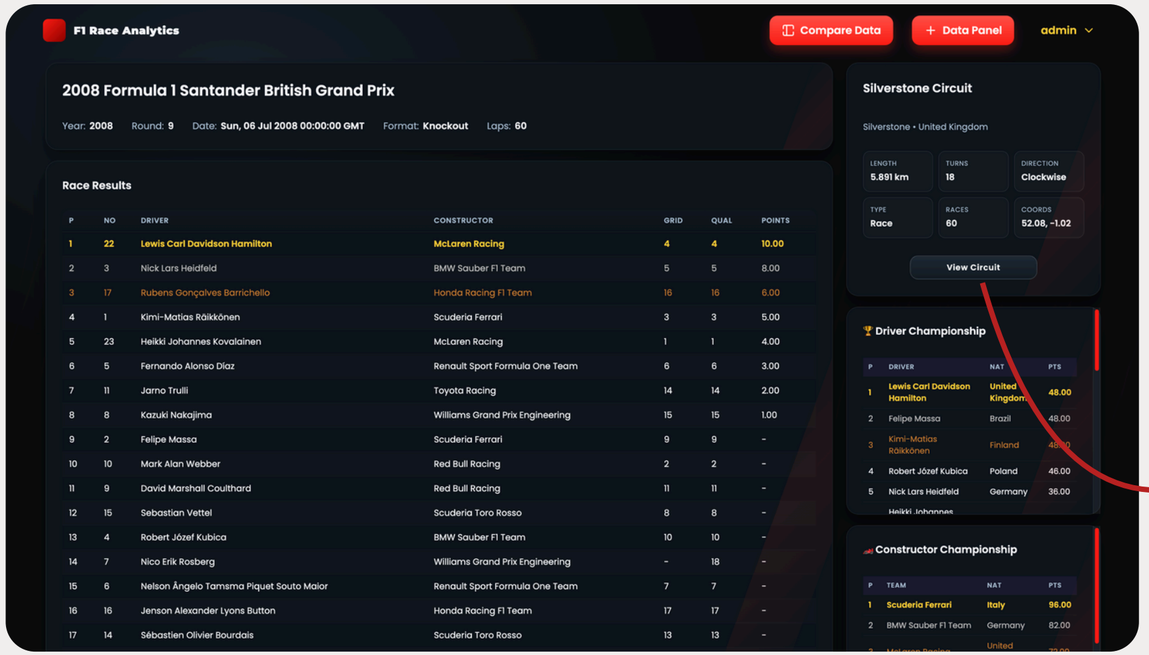
create, update, delete operations

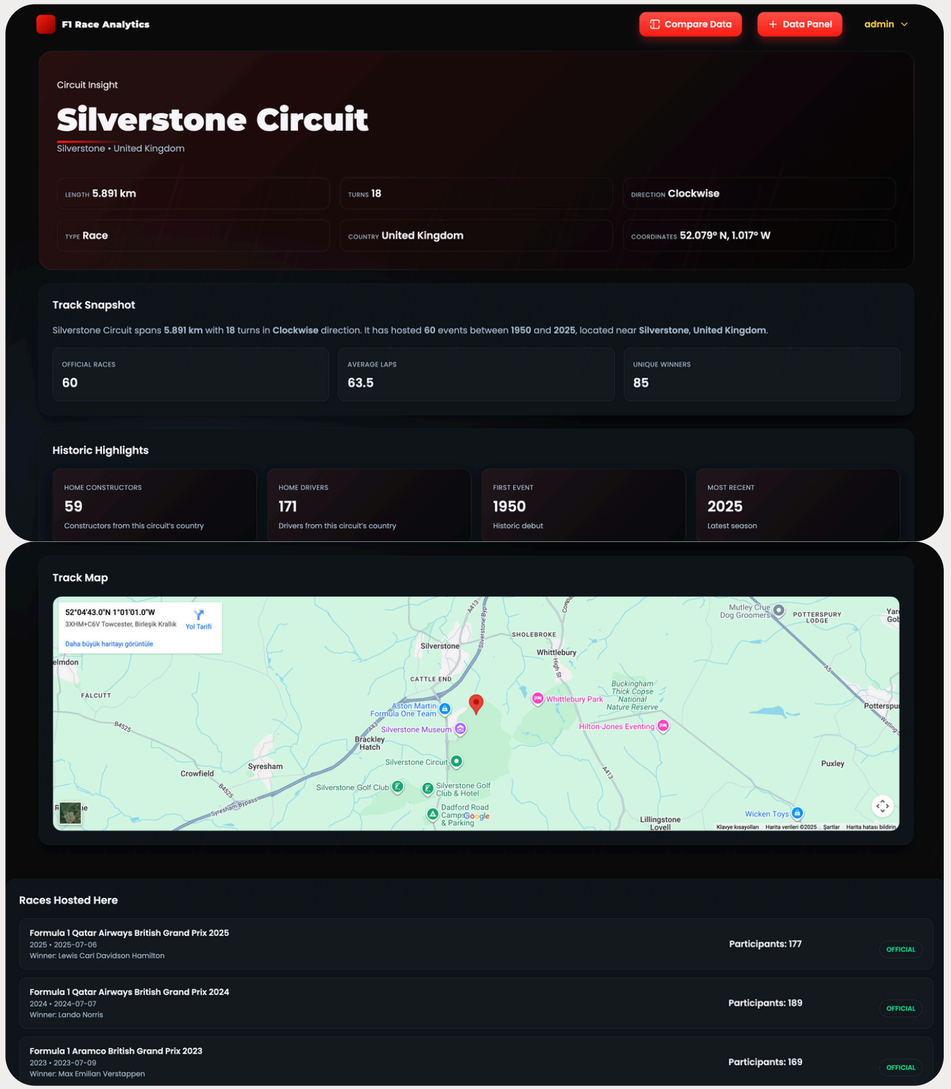






CIRCUIT


F1 Race Analytics
Compare Data
Data Panel
admin
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 Díaz 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
RACES 60
COORDS 52.08, -1.02
View Circuit
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
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

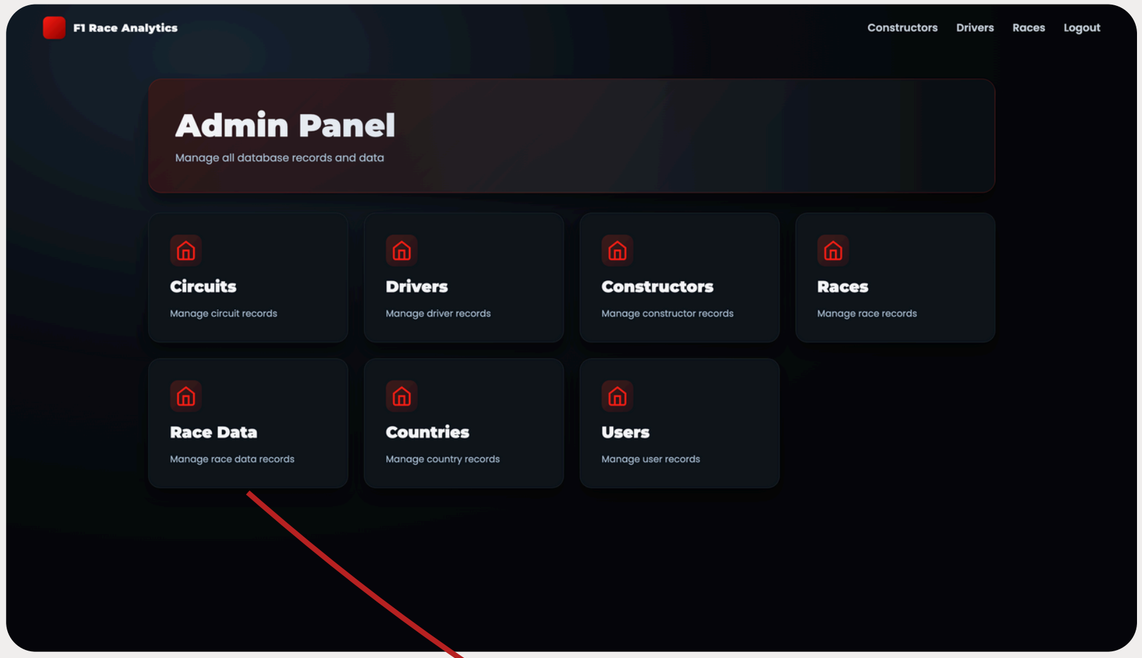

F1 Race Analytics
Compare Data
Data Panel
admin
Circuit Insight
Silverstone Circuit
Silverstone • United Kingdom
LENGTH 5.891 km
TURNS 18
DIRECTION Clockwise
TYPE Race
COUNTRY United Kingdom
COORDINATES 52.079° N, 1.017° W
Track Snapshot
Silverstone Circuit spans 5.891 km with 18 turns in Clockwise direction. 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
Constructors from this circuit's country
HOME DRIVERS 171
Drivers from this circuit's country
FIRST EVENT 1950
Historic debut
MOST RECENT 2025
Latest season
Track Map
52°04'43.0"N 1°01'01.0"W
Yol Tarifi
Daha büyük haritayı görüntüle
Silverstone
Whittlebury
SHOLEBROKE
CATTLE END
FALCUTT
Aston Martin Formula One Team
Silverstone Museum
Brackley Hatch
Silverstone Circuit
Silverstone Golf Club
Silverstone Golf Club & Hotel
Dadford Road Campground & Parking
Crowfield
Syresham
Puxley
Lillingstone Lovell
Wicken Toys
Whittlebury Park
Hilton Jones Eventing
Buckingham Thick Copse National Nature Reserve
Mutley Crue Dog Groomers
POTTERSPURY LODGE
Klavye kısayolları Harita verileri ©2025 Şartlar Harita hatası bildirin
Races Hosted Here
Formula 1 Qatar Airways British Grand Prix 2025
2025 • 2025-07-06
Winner: Lewis Carl Davidson Hamilton
Participants: 177
OFFICIAL
Formula 1 Qatar Airways British Grand Prix 2024
2024 • 2024-07-07
Winner: Lando Norris
Participants: 189
OFFICIAL
Formula 1 Aramco British Grand Prix 2023
2023 • 2023-07-09
Winner: Max Emilian Verstappen
Participants: 169
OFFICIAL

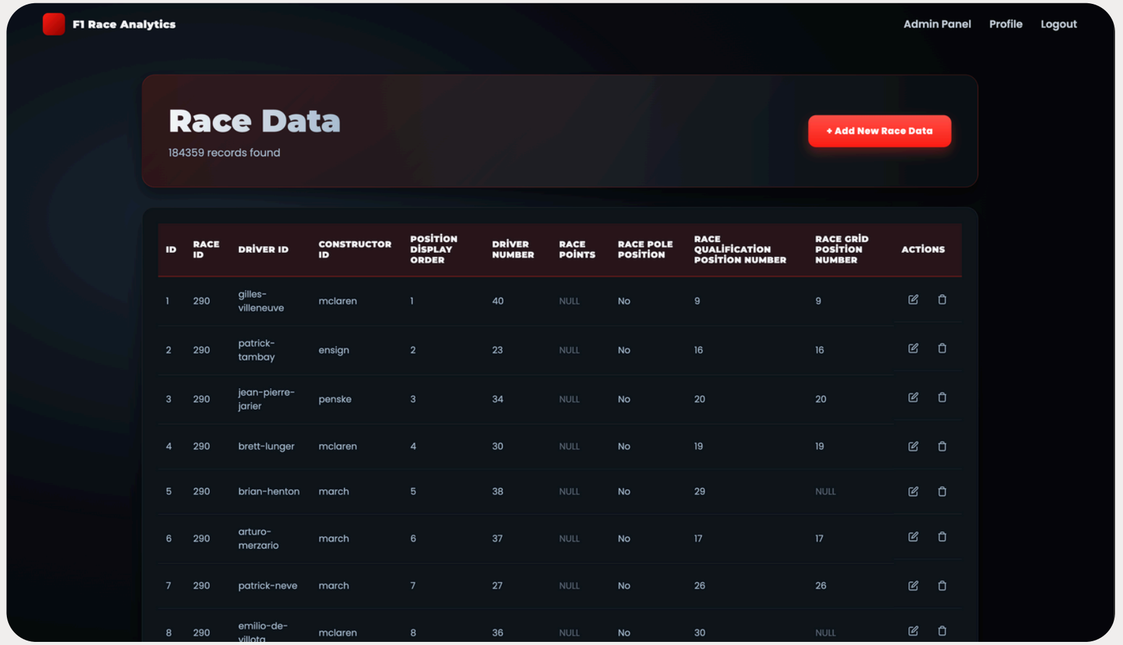
CIRCUIT

```sql
WITH circuit_base AS (
    SELECT c.id, c.name, c.full_name, c.place_name, c.type, c.direction, c.length, c.turns,
           c.total_races_held, c.latitude, c.longitude, co.name AS country_name
    FROM circuit c
    LEFT JOIN country co ON c.country_id = co.id
    WHERE c.id = %(circuit_id)s
    LIMIT 1
),
race_rollup AS (
    SELECT
        r.circuit_id,
        COUNT(*) AS total_races,
        COUNT(*) FILTER (WHERE r.is_real) AS official_races,
        COUNT(*) FILTER (WHERE NOT r.is_real) AS simulated_races,
        MIN(r.year) AS first_year,
        MAX(r.year) AS last_year,
        AVG(r.laps)::NUMERIC(10,2) AS avg_laps
    FROM race r
    WHERE r.circuit_id = %(circuit_id)s
    GROUP BY r.circuit_id
),
winner_stats AS (
    SELECT r.circuit_id,
           COUNT(DISTINCT CASE WHEN rd.position_display_order = 1 THEN rd.driver_id END) AS unique_winners
    FROM race r
    LEFT JOIN race_data rd ON rd.race_id = r.id
    WHERE r.circuit_id = %(circuit_id)s
    GROUP BY r.circuit_id
),
home_driver_counts AS (
    SELECT c.id AS circuit_id, COUNT(DISTINCT d.id) AS home_drivers
    FROM circuit c
    LEFT JOIN country co ON c.country_id = co.id
    LEFT JOIN driver d ON d.nationality_country_id = co.id
    WHERE c.id = %(circuit_id)s
    GROUP BY c.id
),
home_constructor_counts AS (
    SELECT
        c.id AS circuit_id,
        COUNT(DISTINCT cons.id) AS home_constructors
    FROM circuit c
    LEFT JOIN country co ON c.country_id = co.id
    LEFT JOIN constructor cons ON cons.country_id = co.id
    WHERE c.id = %(circuit_id)s
    GROUP BY c.id
),
```

```sql
race_payload AS (
    SELECT
        r.circuit_id,
        json_agg(
            json_build_object(
                'id', r.id,'year', r.year,'round', r.round,'official_name', r.official_name,'date', r.date,
                'qualifying_format', r.qualifying_format,'laps', r.laps,'is_real', r.is_real,
                'participant_count', (
                    SELECT COUNT(*)
                    FROM race_data rd
                    WHERE rd.race_id = r.id
                ),
                'winner', (
                    SELECT json_build_object(
                        'driver_id', d.id,
                        'driver_name', d.full_name,
                        'constructor_name', cons.full_name
                    )
                    FROM race_data rd2
                    JOIN driver d ON rd2.driver_id = d.id
                    LEFT JOIN constructor cons ON cons.id = rd2.constructor_id
                    WHERE rd2.race_id = r.id
                    ORDER BY rd2.position_display_order ASC
                    LIMIT 1
                )
            )
            ORDER BY r.year DESC, r.round DESC
        ) AS races
    FROM race r
    WHERE r.circuit_id = %(circuit_id)s
    GROUP BY r.circuit_id
)
SELECT
    cb.*,
    rr.total_races,rr.official_races,rr.simulated_races,rr.first_year,rr.last_year,
    rr.avg_laps,ws.unique_winners,hd.home_drivers,hc.home_constructors,rp.races
FROM circuit_base cb
LEFT JOIN race_rollup rr ON rr.circuit_id = cb.id
LEFT JOIN winner_stats ws ON ws.circuit_id = cb.id
LEFT JOIN home_driver_counts hd ON hd.circuit_id = cb.id
LEFT JOIN home_constructor_counts hc ON hc.circuit_id = cb.id
LEFT JOIN race_payload rp ON rp.circuit_id = cb.id;
```

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.

```sql
CREATE TABLE "user" (
    id              VARCHAR(100) PRIMARY KEY,
    country_id      VARCHAR(100) REFERENCES country(id) ON DELETE CASCADE,
    username        VARCHAR(100) NOT NULL UNIQUE,
    password_hash   TEXT         NOT NULL,
    email           VARCHAR(255) NOT NULL UNIQUE,
    email_verified  BOOLEAN      NOT NULL DEFAULT FALSE,
    email_verified_at TIMESTAMP,
    email_verification_token VARCHAR(255),
    verification_sent_at TIMESTAMP,
    date_joined     TIMESTAMP    NOT NULL DEFAULT NOW(),
    is_admin        BOOLEAN      NOT NULL DEFAULT FALSE
);

CREATE INDEX user_username_idx ON "user"(username);
CREATE INDEX user_country_id_idx ON "user"(country_id);
```

NORMALIZATION

Circuit Table

✅ **1NF:** Atomic columns
✅ **2NF:** Single-attribute primary key (***id***) ⇒ 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).

```sql
CREATE TABLE circuit (
  id               VARCHAR(100)  NOT NULL,
  name             VARCHAR(100)  NOT NULL,
  full_name        VARCHAR(100)  NOT NULL,
  previous_names   VARCHAR(255),
  type             VARCHAR(6)    NOT NULL,
  direction        VARCHAR(14)   NOT NULL,
  place_name       VARCHAR(100)  NOT NULL,
  country_id       VARCHAR(100)  NOT NULL,
  latitude         DECIMAL(10,6) NOT NULL,
  longitude        DECIMAL(10,6) NOT NULL,
  length           DECIMAL(6,3)  NOT NULL,
  turns            INT           NOT NULL,
  total_races_held INT           NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (country_id) REFERENCES country (id) ON DELETE CASCADE
);

CREATE INDEX circuit_name_idx ON circuit(name);
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.

```sql
CREATE TABLE race (
    id                  INT PRIMARY KEY,
    circuit_id          VARCHAR(100) NOT NULL REFERENCES circuit(id),
    year                INT          NOT NULL,
    round               INT          NOT NULL,
    date                DATE         NOT NULL,
    official_name       VARCHAR(100) NOT NULL,
    qualifying_format   VARCHAR(20)  NOT NULL,
    laps                INT          NOT NULL,
    qualifying_date     DATE,
    is_real             BOOLEAN      DEFAULT TRUE,
    user_id             VARCHAR(100)  DEFAULT NULL,

    FOREIGN KEY (user_id)    REFERENCES "user"(id) ON DELETE CASCADE,
    UNIQUE (year, round)
);

CREATE INDEX race_year_idx       ON race(year);
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).

```sql
CREATE TABLE race_constructor_standing (
    race_id            INT          NOT NULL REFERENCES race(id) ON DELETE CASCADE,
    constructor_id     VARCHAR(100) NOT NULL REFERENCES constructor(id) ON DELETE CASCADE,
    position_number    INT,
    points             DECIMAL(8,2) NOT NULL,

    PRIMARY KEY (race_id, constructor_id)
);

CREATE INDEX rcst_race_id_idx ON race_constructor_standing(race_id);

```

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).

```sql
CREATE TABLE constructor (
    id                          VARCHAR(100)  NOT NULL,
    user_id                     VARCHAR(100)  DEFAULT NULL,
    country_id                  VARCHAR(100)  NOT NULL,
    name                        VARCHAR(100)  NOT NULL,
    full_name                   VARCHAR(100)  NOT NULL,
    best_championship_position  INTEGER,
    total_championship_wins     INTEGER       NOT NULL,
    total_race_starts           INTEGER       NOT NULL,
    total_podiums               INTEGER       NOT NULL,
    total_points                DECIMAL(8,2)  NOT NULL,
    total_pole_positions        INTEGER       NOT NULL,
    is_real                     BOOLEAN       DEFAULT TRUE,
    PRIMARY KEY (id),
    FOREIGN KEY (country_id) REFERENCES country(id) ON DELETE CASCADE,
    FOREIGN KEY (user_id)    REFERENCES "user"(id) ON DELETE CASCADE
);

CREATE INDEX constructor_name_idx ON constructor(name);
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).

```sql
CREATE TABLE race_data (
    id                                  BIGSERIAL    PRIMARY KEY,
    race_id                             INT          NOT NULL,          -- FK race(id)
    driver_id                           VARCHAR(100) NOT NULL,          -- FK driver(id)
    constructor_id                      VARCHAR(100) NOT NULL,          -- FK constructor(id)
    user_id                             VARCHAR(100) DEFAULT NULL,      -- FK user(id)
    position_display_order              INT          NOT NULL,
    driver_number                       VARCHAR(3)   NOT NULL,
    race_points                         DECIMAL(8,2),
    race_pole_position                  BOOLEAN,
    race_qualification_position_number  INT,
    race_grid_position_number           INT,
    is_real                             BOOLEAN      DEFAULT FALSE,
    created_at                          TIMESTAMP    NOT NULL DEFAULT NOW(),

    FOREIGN KEY (user_id)        REFERENCES "user"(id) ON DELETE CASCADE,
    FOREIGN KEY (constructor_id) REFERENCES constructor (id) ON DELETE CASCADE,
    FOREIGN KEY (driver_id)      REFERENCES driver      (id) ON DELETE CASCADE,
    FOREIGN KEY (race_id)        REFERENCES race        (id) ON DELETE CASCADE
);


CREATE INDEX rcda_race_id_idx               ON race_data(race_id);
CREATE INDEX rcda_position_display_order_idx ON race_data(position_display_order);
CREATE INDEX rcda_driver_id_idx             ON race_data(driver_id);
CREATE INDEX rcda_constructor_id_idx        ON race_data(constructor_id);
```

NORMALIZATION

Country Table

✅**1NF:** All attributes are atomic (no lists / repeating groups).
✅ **2NF:** Single-attribute primary key (***id***) ⇒ 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***).

```sql
CREATE TABLE country (
    id            VARCHAR(100)  PRIMARY KEY,
    alpha3_code   VARCHAR(3)    NOT NULL UNIQUE,
    name          VARCHAR(100)  NOT NULL
);

```

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.

```sql
CREATE TABLE driver (
    id                          VARCHAR(100) PRIMARY KEY,
    name                        VARCHAR(100) NOT NULL,
    first_name                  VARCHAR(100),
    last_name                   VARCHAR(100),
    full_name                   VARCHAR(100) NOT NULL,
    abbreviation                VARCHAR(10),
    permanent_number            INT,
    gender                      VARCHAR(20),
    date_of_birth               DATE,
    date_of_death               DATE,
    place_of_birth              VARCHAR(100),
    country_of_birth_country_id VARCHAR(100) NOT NULL,
    nationality_country_id      VARCHAR(100) NOT NULL,
    best_championship_position  INT,
    best_race_result            INT,
    total_championship_wins     INT          NOT NULL,
    total_race_starts           INT          NOT NULL,
    total_race_wins             INT          NOT NULL,
    total_race_laps             INT          NOT NULL,
    total_podiums               INT          NOT NULL,
    total_points                DECIMAL(8,2) NOT NULL,
    total_pole_positions        INT          NOT NULL,
    is_real                     BOOLEAN      DEFAULT TRUE,
    user_id                     VARCHAR(100)  DEFAULT NULL,

    FOREIGN KEY (country_of_birth_country_id) REFERENCES country(id) ON DELETE CASCADE,
    FOREIGN KEY (nationality_country_id)      REFERENCES country(id) ON DELETE CASCADE,
    FOREIGN KEY (user_id)    REFERENCES "user"(id) ON DELETE CASCADE
);

CREATE INDEX driver_name_idx ON driver(full_name);
CREATE INDEX driver_abbreviation_idx ON driver(abbreviation);
CREATE INDEX drv_nationality_idx          ON driver(nationality_country_id);
CREATE INDEX drv_country_of_birth_idx     ON driver(country_of_birth_country_id);
```

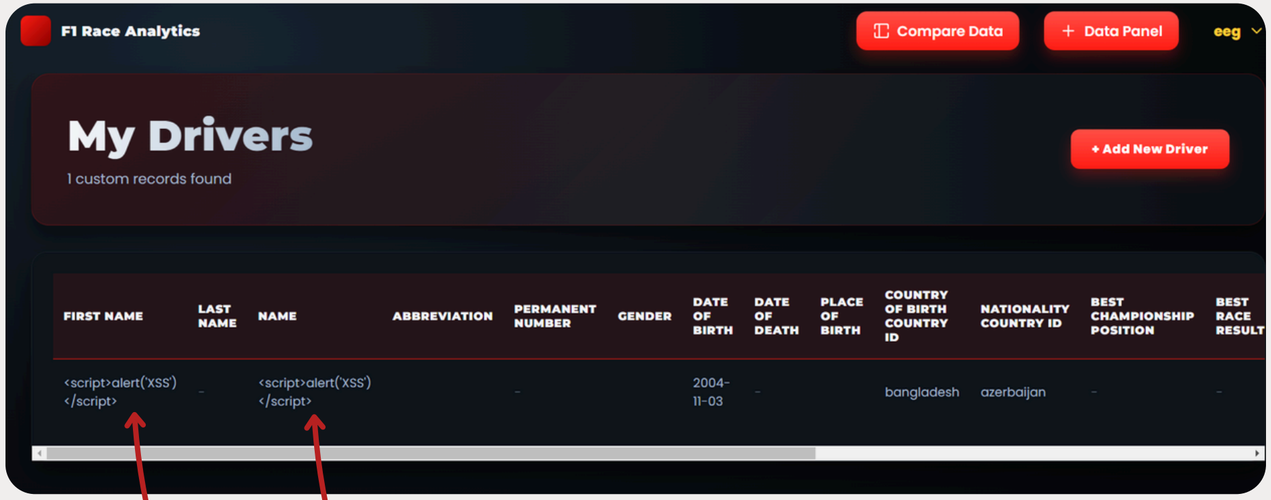
NORMALIZATION

Race Driver Standing Table

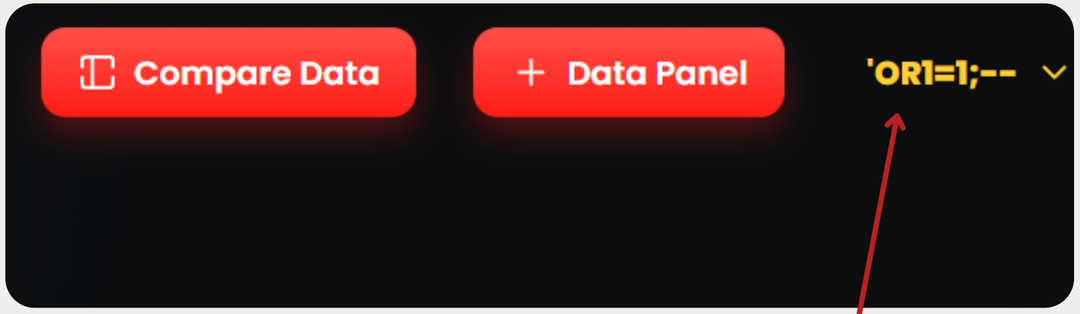
✅ **1NF:** Atomic values; no repeating groups.
✅ **2NF:** Composite PK (***race_id, driver_id***) ⇒ non-key columns depend on the full key.
✅ **3NF:** No non-key-to-non-key dependencies (position/points depend only on the key).
✅ **BCNF:** The only determinants are the candidate key itself (***race_id,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

```sql
CREATE TABLE race_driver_standing (
    race_id             INT          NOT NULL,
    driver_id           VARCHAR(100) NOT NULL,

    position_number     INT          NOT NULL,
    points              DECIMAL(8,2) NOT NULL,

    PRIMARY KEY (race_id, driver_id),

    FOREIGN KEY (race_id)   REFERENCES race(id) ON DELETE CASCADE,
    FOREIGN KEY (driver_id) REFERENCES driver(id) ON DELETE CASCADE
);

CREATE INDEX rds_race_id_idx ON race_driver_standing(race_id);
CREATE INDEX rds_driver_id_idx ON race_driver_standing(driver_id);

```

SECURITY TESTING

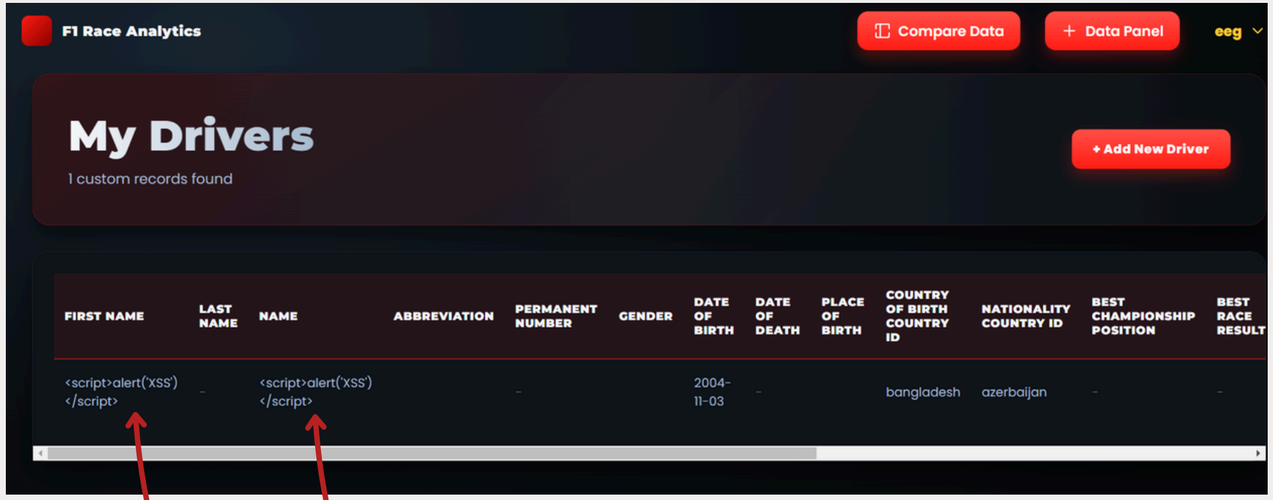


Eq: XSS(Cross-Site Scripting) bypassed.

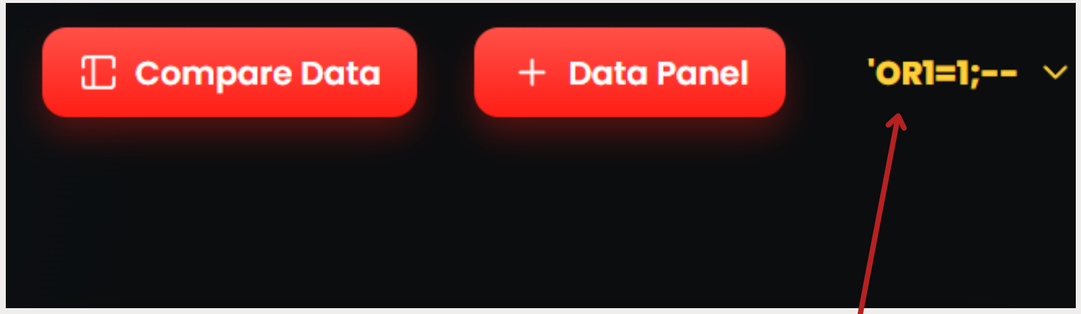


Eq: SQL Injection bypassed.

SECURITY TESTING



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.