DATA WAREHOUSE PROJECT

A.Y. 2024/2025

Angela Di Giampaolo
Simone Giudici

AGENDA

01 | Introduction

02 | Sources of Data

03 | DFM model

04 | Data Warehouse Architecture

05 | Snowflake Schema

06 | ETL Implementation

07 | OLAP operations

Introduction

Formula 1 generates an enormous amount of data every race — from car telemetry and lap times to weather conditions, teams, and drivers - that can be difficult to analyze.

Our goal was to design and implement a **Data Warehouse** for Formula 1, aimed at collecting, integrating, and analyzing data from multiple sources.

The main advantages of this data warehouse are:

- Organize large volumes of information (telemetry, lap times, circuits, weather, teams, drivers)
- Support multidimensional analysis through OLAP operations
- Provide clear insights into performance and race-influencing factors

In this way, the Data Warehouse becomes a strategic tool to transform raw data into valuable knowledge for decision-making in Formula 1.

Data Sources

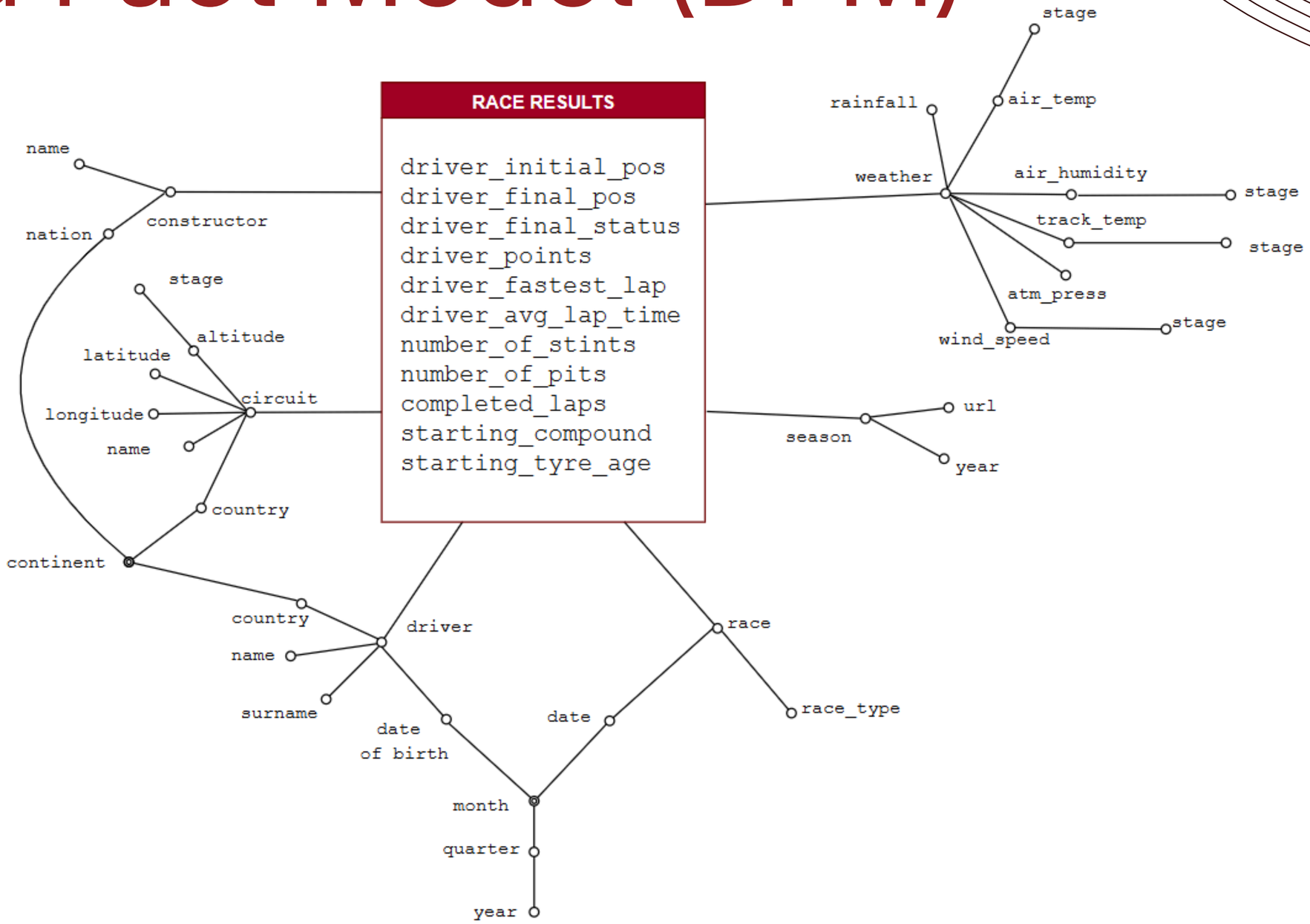
Formula 1 Wolrd Championship

The dataset consists of all information on the Formula 1 races, drivers, constructors, qualifying, circuits, lap times, pit stops, championships.
It is composed by many CSV files (14)

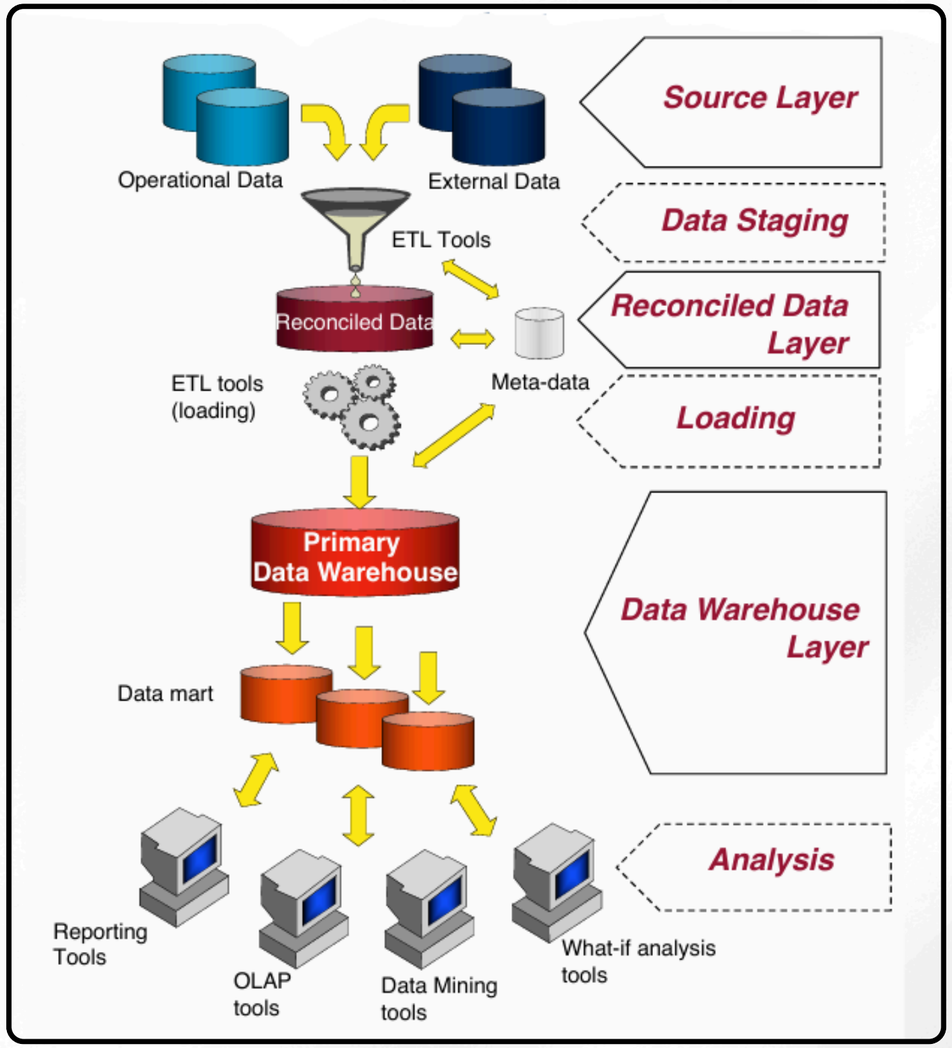
Open F1

OpenF1 is an open-source API that provides real-time and historical Formula 1 data.
The API offers a wealth of information, including lap timings, car telemetry, radio communications, and more.
Data can be accessed in either JSON or CSV formats, and we choose the second one in order to be conform to data coming from the first dataset.
We accessed 7 CSV files concerning drivers, meetings, sessions, speed, stints and weather.

Data Fact Model (DFM)



Data Warehouse Architecture



Source Layer

It contanis the data coming from different datasets

Reconciled Data Layer

Central, integrated, and cleansed repository containing data from heterogeneous sources

Data Warehouse Layer

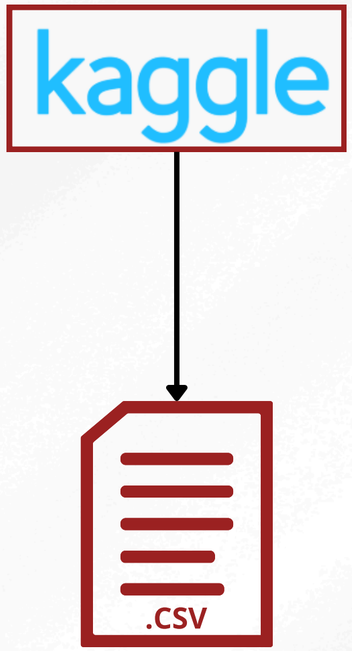
It contains integrated, consistent, and historical data

ETL tools

Tools for Extracting, Transforming and Loading data

Source Layer

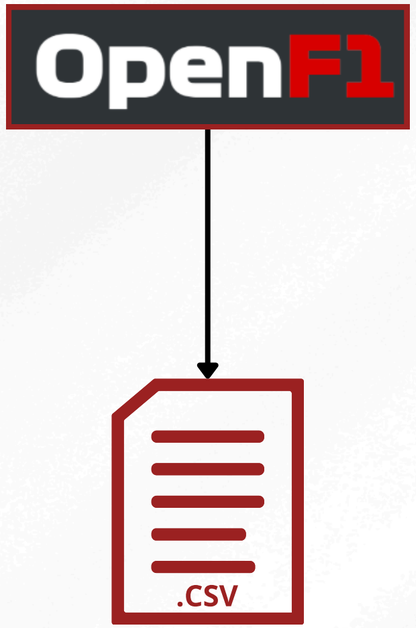
Formula 1 Wolrd Championship



Name	Format	Size
circuits	CSV	10 kB
constructor_results	CSV	214 kB
constructor_standings	CSV	310 kB
constructors	CSV	17 kB
driver_standings	CSV	863 kB
drivers	CSV	92 kB
lap_times	CSV	16.8 MB
pit_stops	CSV	433 kB
qualifying	CSV	454 kB
races	CSV	160 kB
results	CSV	1.6 MB
seasons	CSV	4 kB
sprint_results	CSV	24 kB
status	CSV	2 kB

Source Layer

Open F1



Name	Format	Size
drivers_openf1	CSV	1.2 MB
meetings	CSV	7 kB
sessions	CSV	31 kB
speed	CSV	1.3 MB
stints	CSV	102 kB
weather	CSV	1.8 MB

Reconciled Data Layer

The data sources we adopted were different among them:

- differences in unit of measures (i.e., time measure)
- different primary key
- different driver identification
- different race identification

In order to solve this problem we chose to implement a **Reconciled Data Layer**, that allowed us to obtain a more homogeneous source of data

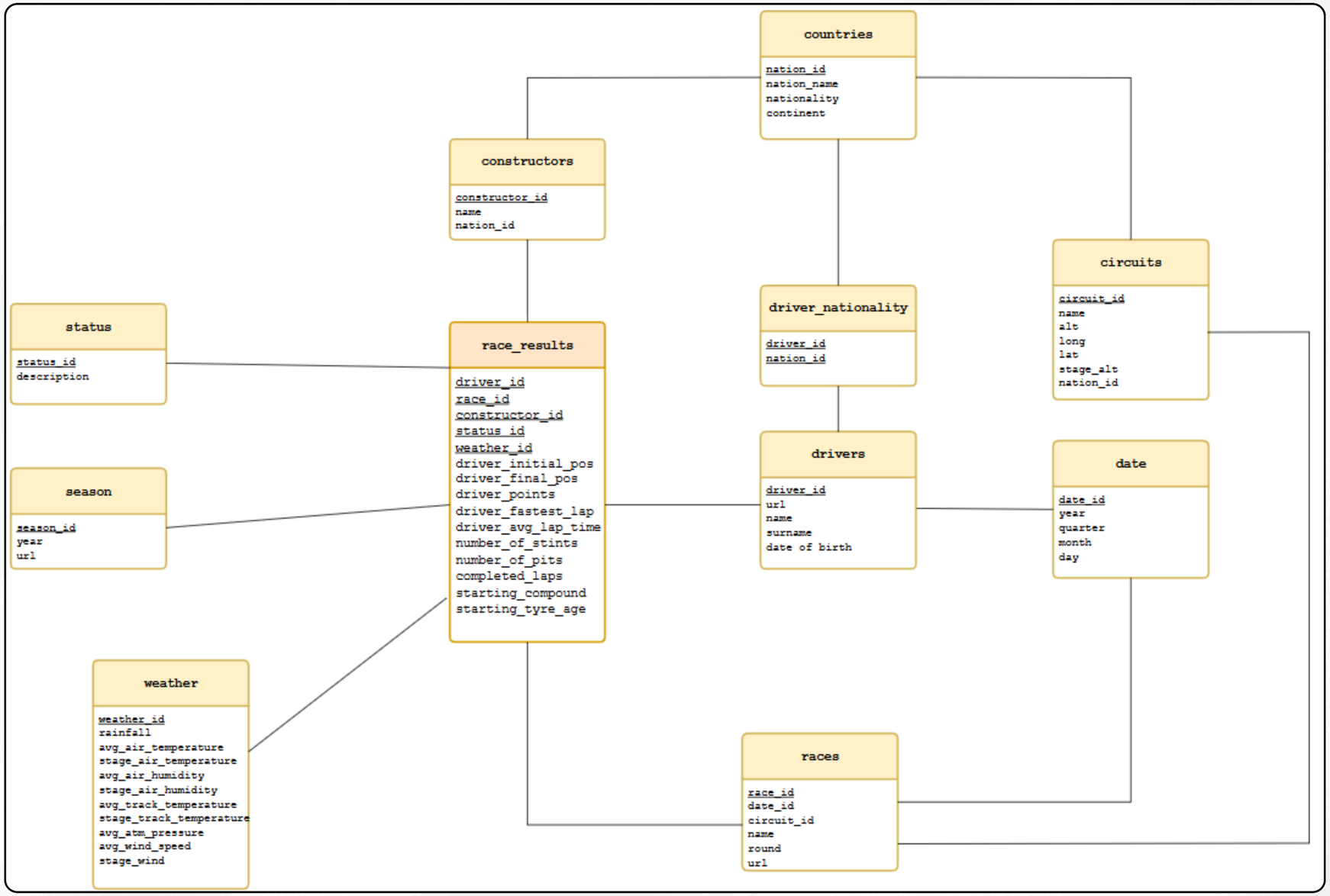
Thanks to RDL we were able to:

- obtain homogeneous unit of measures
- get a uniform primary key
- get a homogeneous identification of drivers, races and other elements

PostgreSQL RDL schema has been implemented by using SQL scripts

Operations concerning data extraction, validation, transformation and loading have been implemented through ETL

Snowflake Schema



Data Warehouse Layer

In this project phase we had to choose between Star Schema and Snowflake Schema

Star Schema	Snowflake Schema
Denormalized	Normalized
Simpler and more intuitive, because each dimension is represented as a table	More complex because it requires multiple join operations in order to construct dimensions
Faster queries	Slower queries
Requires more space	Requires less space

We chose the snowflake schema because:

- the Dimensional Fact Model contains several **shared hierarchies** → normalization allows more efficient space management by reducing redundancy.
- most **expensive operations** are expected to be **infrequent** → even if normalization increases the number of join, this type of workload is primarily analytical and these operation will not be performed too often

ETL

Key Operations

Extraction

Data have been extracted from the two data sources:

- Kaggle Dataset → simple download
- OpenF1 → http requests

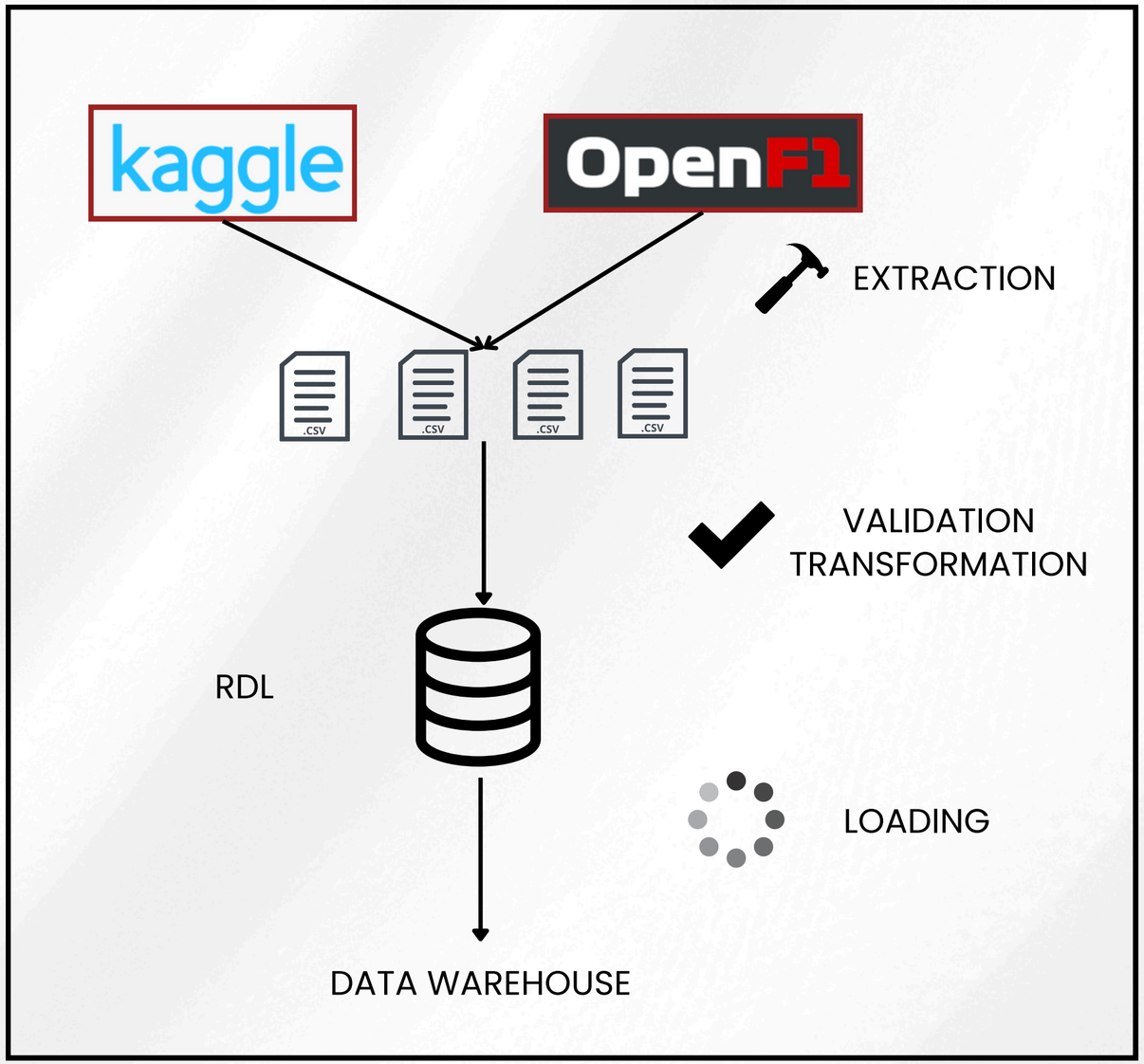
Transformation

Data have been validated and transformed:

- homogeneous identifier for main elements
- homogeneous unit of measure
- management of missing information

Loading

Transformed and validated data populated the RDL.
The information needed for the DW are inserted in the aforementioned schema



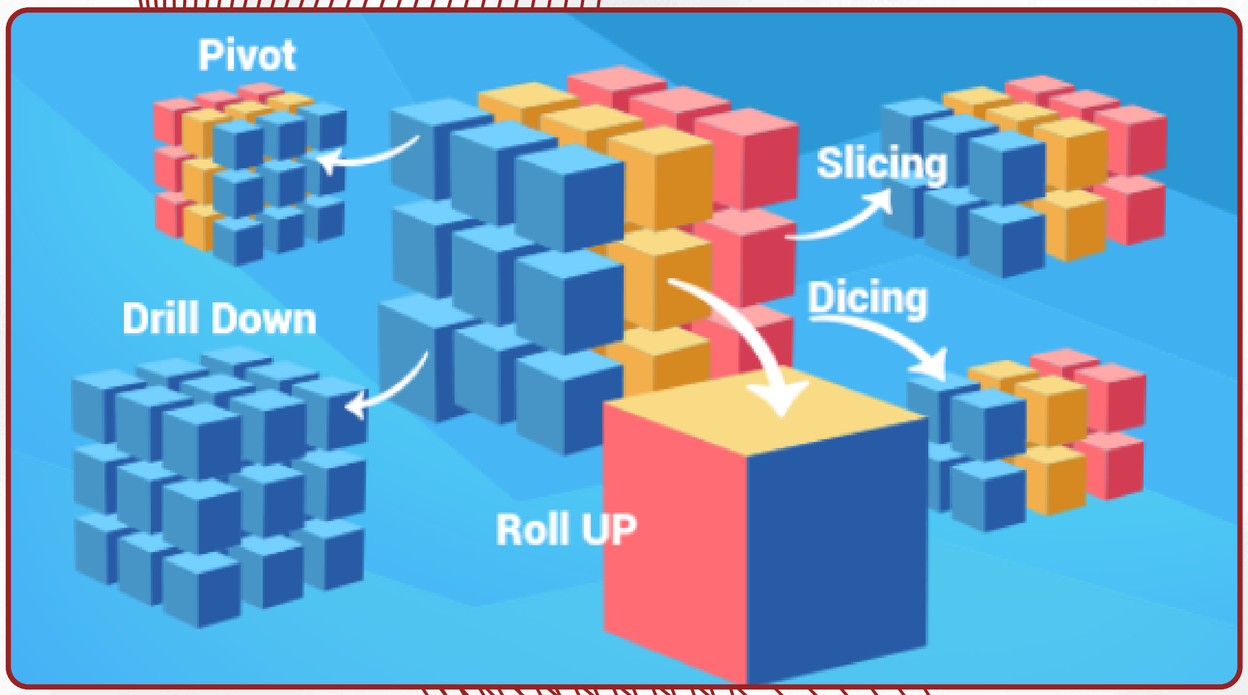
ETL functionalities have been implemented by using Python scripts

OLAP Operations

OLAP allow us to analyze and obtain precious information from data.

We focused on:

- ROLL-UP → aggregate data
- DRILL-DOWN → detailed view
- SLICING → filter by one dimension
- DICING → select sub-cube
- PIVOT → change perspective



Roll-up

? Compute the total points earned by each team per continent.

```sql
WITH circuit_races AS (
                        SELECT race_id, cir.circuit_id, nation_id
                        FROM dw.circuits cir INNER JOIN dw.races ra ON cir.circuit_id = ra.circuit_id
                        ),
     race_continent AS (
                        SELECT race_id, continent
                        FROM circuit_races cr INNER JOIN dw.countries c ON cr.nation_id = c.nation_id
                        )
SELECT rr.constructor_id, c.name, rc.continent, SUM(rr.driver_points) AS total_point
FROM dw.race_results rr INNER JOIN race_continent rc ON rr.race_id = rc.race_id
     INNER JOIN dw.constructors c ON rr.constructor_id = c.constructor_id
GROUP BY rr.constructor_id, rc.continent, c.name
ORDER BY rr.constructor_id, rc.continent
```

	constructor_id bigint	name character varying (100)	continent character varying (100)	total_point bigint
1	1	McLaren	Africa	77
2	1	McLaren	Asia	1606
3	1	McLaren	Europe	3737
4	1	McLaren	North America	802
5	1	McLaren	Oceania	363
6	1	McLaren	South America	437
7	2	BMW Sauber	Asia	91
8	2	BMW Sauber	Europe	152

Total rows: 607 Query complete 00:00:00.209

circuit → country → continent

This query performs a ROLL-UP to the continent level, since it aggregates constructor points across races and circuits, summarizing them by continent

Slicing

? Compute the average points per driver in wet races

```sql
SELECT dr.driver_id, dr.name, dr.surname, aux.average_points
FROM (
        SELECT rr.driver_id, AVG(rr.driver_points) AS average_points
        FROM dw.race_results rr LEFT OUTER JOIN dw.weather w ON rr.weather_id = w.weather_id
        WHERE w.rainfall IS TRUE
        GROUP BY rr.driver_id
     ) AS aux
     INNER JOIN dw.drivers dr ON aux.driver_id = dr.driver_id
ORDER BY aux.average_points DESC
```

	driver_id [PK] bigint	name character varying (100)	surname character varying (100)	average_points numeric
1	830	Max	Verstappen	20.7777777777777778
2	846	Lando	Norris	11.8148148148148148
3	1	Lewis	Hamilton	10.3703703703703704
4	844	Charles	Leclerc	9.2962962962962963
5	832	Carlos	Sainz	8.3333333333333333
6	815	Sergio	Pérez	8.2592592592592593
7	857	Oscar	Piastri	8.0000000000000000
8	847	George	Russell	7.4074074074074074

Total rows: 24 Query complete 00:00:00.413

This query performs a SLICING to race levels, because we are selecting only the "wet races" subset along one dimension and aggregating (average).

Slice-and-Dice

Compute the top 5 teams performing best in high/medium-altitude circuits

```sql
WITH circuit_races AS (
                        SELECT r.race_id, c.circuit_id, c.alt_stage
                        FROM dw.races r
                        INNER JOIN dw.circuits c ON r.circuit_id = c.circuit_id
                        WHERE c.alt_stage IN ('high', 'medium')
                      ),
    scores AS (
        SELECT rr.constructor_id, SUM(rr.driver_points) AS score
        FROM dw.race_results rr
        INNER JOIN circuit_races cr ON rr.race_id = cr.race_id
        GROUP BY rr.constructor_id
                  ),
    score_rank AS (
        SELECT c.constructor_id, c.name, cs.score, RANK() OVER (ORDER BY cs.score DESC) AS rank
        FROM scores cs INNER JOIN dw.constructors c ON cs.constructor_id = c.constructor_id
    )
SELECT name, score
FROM score_rank
WHERE rank <= 5
ORDER BY rank;
```

	name character varying (100)	score bigint
1	Ferrari	3225
2	McLaren	2143
3	Mercedes	2036
4	Red Bull	1957
5	Williams	1170

Total rows: 5 | Query complete 00:00:00.162

This query performs a slice-and-dice because it produces “sub-cubes” starting from cubes.
This means that some conditions have been applied to data (in particular we have conditions on circuits altitude)

Pivoting

? Compute the comparison of constructors by continent

```sql
WITH race_continent AS (
                        SELECT aux.race_id, aux.race_name, aux.circuit_id, aux.name, aux.nation_id, c.continent
                        FROM (
                                SELECT r.race_id, r.name AS race_name, c.circuit_id, c.name, c.nation_id
                                FROM dw.races r INNER JOIN dw.circuits c ON r.circuit_id = c.circuit_id
                                ) AS aux
                                INNER JOIN dw.countries c ON aux.nation_id = c.nation_id
                        ORDER BY c.continent
                        )
SELECT c.name, SUM(CASE WHEN rc.continent = 'Europe'   THEN rr.driver_points ELSE 0 END) AS points_europe,
       SUM(CASE WHEN rc.continent = 'North America' OR rc.continent = 'South America'
                THEN rr.driver_points ELSE 0 END) AS points_america,
       SUM(CASE WHEN rc.continent = 'Asia'     THEN rr.driver_points ELSE 0 END) AS points_asia,
       SUM(CASE WHEN rc.continent = 'Oceania'  THEN rr.driver_points ELSE 0 END) AS points_oceania,
       SUM(CASE WHEN rc.continent = 'Africa'   THEN rr.driver_points ELSE 0 END) AS points_africa
FROM   dw.race_results rr INNER JOIN race_continent rc ON rr.race_id = rc.race_id
       INNER JOIN dw.constructors c ON rr.constructor_id = c.constructor_id
GROUP BY rr.constructor_id, c.name
ORDER BY rr.constructor_id, c.name
```

	name character varying (100)	points_europe bigint	points_america bigint	points_asia bigint	points_oceania bigint	points_africa bigint
1	McLaren	3737	1239	1606	363	77
2	BMW Sauber	152	44	91	21	0
3	Williams	2175	699	522	189	56
4	Renault	949	249	449	103	27
5	Toro Rosso	213	102	156	29	0
6	Ferrari	6105	2004	2445	456	79

Total rows: 211 | Query complete 00:00:00.129

Ranking

Compute the best race for each constructor

```sql
WITH races_constructors AS (
                                SELECT rr.constructor_id, rr.race_id, r.name AS race_name,
                                       c.name AS constructor_name, SUM(rr.driver_points) AS score
                                FROM dw.race_results rr INNER JOIN dw.races r ON rr.race_id = r.race_id
                                     INNER JOIN dw.constructors c ON rr.constructor_id = c.constructor_id
                                GROUP BY rr.constructor_id, rr.race_id, r.name, c.name
                                ORDER BY rr.constructor_id, score DESC
                            ),
             score_rank AS (
                                SELECT rc.constructor_id, rc.race_id, rc.race_name,
                                       rc.constructor_name, rc.score,
                                       DENSE_RANK() OVER (PARTITION BY rc.constructor_id ORDER BY rc.score DESC ) AS rank
                                FROM races_constructors rc
                            )
SELECT sc.constructor_name, sc.race_name, sc.score
FROM score_rank sc
WHERE sc.rank = 1 AND NOT(score = 0)
```

	constructor_name character varying (100)	race_name character varying (100)	score bigint
1	McLaren	Italian Grand Prix	44
2	BMW Sauber	Canadian Grand Prix	18
3	Williams	Abu Dhabi Grand Prix	66
4	Renault	Sakhir Grand Prix	28
5	Toro Rosso	German Grand Prix	23
6	Ferrari	Bahrain Grand Prix	44

Total rows: 156 | Query complete 00:00:00.112

Pivoting

? Compute the comparison between drivers' nationalities and points scored across continents

```sql
WITH driver_nationality AS (
                                SELECT d.driver_id, d.name, d.surname, c.nationality
                                FROM dw.drivers d INNER JOIN dw.driver_nationality dn ON d.driver_id = dn.driver_id
                                     INNER JOIN dw.countries c ON dn.nation_id = c.nation_id
                            ),
         race_continent AS (
                                SELECT r.race_id, r.name AS race_name, c.circuit_id, c.name AS circuit_name, co.continent
                                FROM dw.races r INNER JOIN dw.circuits c ON r.circuit_id = c.circuit_id
                                     INNER JOIN dw.countries co ON c.nation_id = co.nation_id
                            )
SELECT dn.name, dn.surname, dn.nationality,
       SUM(CASE WHEN rc.continent = 'Europe'  THEN rr.driver_points ELSE 0 END) AS europe_points,
       SUM(CASE WHEN rc.continent = 'North America' OR rc.continent = 'South America'
                THEN rr.driver_points ELSE 0 END) AS america_points,
       SUM(CASE WHEN rc.continent = 'Asia'    THEN rr.driver_points ELSE 0 END) AS asia_points,
       SUM(CASE WHEN rc.continent = 'Africa'    THEN rr.driver_points ELSE 0 END) AS africa_points,
       SUM(CASE WHEN rc.continent = 'Oceania'    THEN rr.driver_points ELSE 0 END) AS oceania_points
FROM dw.race_results rr INNER JOIN driver_nationality dn ON rr.driver_id = dn.driver_id
     INNER JOIN race_continent rc ON rr.race_id = rc.race_id
GROUP BY dn.driver_id, dn.name, dn.surname, dn.nationality
ORDER BY dn.surname
```

	name character varying (100)	surname character varying (100)	nationality character varying (100)	europe_points bigint	america_points bigint	asia_points bigint	africa_points bigint	oceania_points bigint
1	Carlo	Abate	Italian	0	0	0	0	0
2	George	Abecassis	British	0	0	0	0	0
3	Kenny	Acheson	British	0	0	0	0	0
4	Philippe	Adams	Belgian	0	0	0	0	0
5	Walt	Ader	American	0	0	0	0	0
6	Kurt	Adolff	German	0	0	0	0	0
7	Fred	Agabashian	American	0	2	0	0	0
8	Kurt	Ahrens	German	0	0	0	0	0

Total rows: 863 | Query complete 00:00:00.167 | CRLI

Pivoting

? For each team, compute points and podiums for decade

```sql
SELECT
    c.name AS team,
    SUM(rr.driver_points) FILTER (WHERE d.year BETWEEN 1950 AND 1959) AS points_1950s,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3 AND d.year BETWEEN 1950 AND 1959) AS podiums_1950s,
    SUM(rr.driver_points) FILTER (WHERE d.year BETWEEN 1960 AND 1969) AS points_1960s,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3 AND d.year BETWEEN 1960 AND 1969) AS podiums_1960s,
    SUM(rr.driver_points) FILTER (WHERE d.year BETWEEN 1970 AND 1979) AS points_1970s,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3 AND d.year BETWEEN 1970 AND 1979) AS podiums_1970s,
    SUM(rr.driver_points) FILTER (WHERE d.year BETWEEN 1980 AND 1989) AS points_1980s,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3 AND d.year BETWEEN 1980 AND 1989) AS podiums_1980s,
    SUM(rr.driver_points) FILTER (WHERE d.year BETWEEN 1990 AND 1999) AS points_1990s,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3 AND d.year BETWEEN 1990 AND 1999) AS podiums_1990s,
    SUM(rr.driver_points) FILTER (WHERE d.year BETWEEN 2000 AND 2009) AS points_2000s,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3 AND d.year BETWEEN 2000 AND 2009) AS podiums_2000s,
    SUM(rr.driver_points) FILTER (WHERE d.year BETWEEN 2010 AND 2019) AS points_2010s,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3 AND d.year BETWEEN 2010 AND 2019) AS podiums_2010s,
    SUM(rr.driver_points) FILTER (WHERE d.year BETWEEN 2020 AND 2029) AS points_2020s,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3 AND d.year BETWEEN 2020 AND 2029) AS podiums_2020s,
    -- Totals across all decades
    SUM(rr.driver_points) AS total_points,
    COUNT(*) FILTER (WHERE rr.driver_final_pos BETWEEN 1 AND 3) AS total_podiums
FROM dw.race_results rr, dw.races r , dw.date d, dw.constructors c
WHERE rr.race_id = r.race_id AND r.date_id = d.date_id AND rr.constructor_id = c.constructor_id
GROUP BY c.name
ORDER BY total_points DESC,total_podiums DESC;
```

Pivoting

? For each team, compute points and podiums for decade

	team character varying (100)	points_1950s bigint	podiums_1950s bigint	points_1960s bigint	podiums_1960s bigint	points_1970s bigint	podiums_1970s bigint	points_1980s bigint	podiums_1980s bigint
1	Ferrari	754	117	404	56	751	96	558	72
2	Mercedes	139	17	[null]	0	[null]	0	[null]	0
3	Red Bull	[null]	0	[null]	0	[null]	0	[null]	0
4	McLaren	[null]	0	0	0	483	59	887	114
5	Williams	[null]	0	[null]	0	92	12	783	100
6	Renault	[null]	0	[null]	0	29	4	283	37
7	Force India	[null]	0	[null]	0	[null]	0	[null]	0
8	Team Lotus	8	0	52	7	548	68	356	39
9	Benetton	[null]	0	[null]	0	[null]	0	125	15
10	Tyrrell	[null]	0	[null]	0	541	69	109	4

Total rows: 211 | Query complete 00:00:00.170

points_1990s bigint	podiums_1990s bigint	points_2000s bigint	podiums_2000s bigint	points_2010s bigint	podiums_2010s bigint	points_2020s bigint	podiums_2020s bigint	total_points bigint	total_podiums bigint
791	107	1737	192	4164	142	1930	59	11089	841
[null]	0	[null]	0	5112	194	2480	87	7731	298
[null]	0	257	19	4468	151	2948	112	7673	282
907	116	1272	131	1972	50	1501	38	7022	508
1121	136	610	48	961	16	74	1	3641	313
[null]	0	770	54	514	5	181	3	1777	103
[null]	0	13	1	1085	5	[null]	0	1098	6
31	0	[null]	0	[null]	0	[null]	0	995	114
707	83	30	4	[null]	0	[null]	0	862	102
61	4	[null]	0	[null]	0	[null]	0	711	77

Dicing/Roll-up

Rank drivers by computing achieved points in hot and humid races for seasons 2023 and 2024

```sql
SELECT rr.driver_id,drivers.name, drivers.surname, races.name, date.year,
        SUM(rr.driver_points) as tot_points
FROM dw.weather as weather, dw.race_results as rr,
     dw.races as races, dw.date as date, dw.drivers as drivers
WHERE date.date_id = races.date_id
        AND races.race_id = rr.race_id AND weather.weather_id = rr.race_id
        AND drivers.driver_id = rr.driver_id
        AND (date.year = 2023 OR date.year=2024)
        AND (weather.stage_air_temperature = 'high' OR weather.stage_air_temperature = 'medium')
        AND weather.stage_track_temperature = 'high'
        AND weather.stage_air_humidity = 'high'
GROUP BY rr.driver_id,drivers.name,drivers.surname,races.name,date.year
ORDER BY  date.year,tot_points DESC
```

	driver_id bigint	name character varying (100)	surname character varying (100)	name character varying (100)	year integer	tot_points bigint
1	815	Sergio	Pérez	Azerbaijan Grand Prix	2023	25
2	832	Carlos	Sainz	Singapore Grand Prix	2023	25
3	846	Lando	Norris	Singapore Grand Prix	2023	18
4	830	Max	Verstappen	Azerbaijan Grand Prix	2023	18
5	1	Lewis	Hamilton	Singapore Grand Prix	2023	16
6	844	Charles	Leclerc	Azerbaijan Grand Prix	2023	15
7	4	Fernando	Alonso	Azerbaijan Grand Prix	2023	12

Slicing

Compute the podiums with constructors, drivers and circuits of the same nation/ nationality

```
SELECT dw.drivers.name,dw.drivers.surname,dw.constructors.name,dw.circuits.location,dw.races.name,dw.date.year
FROM dw.races,dw.race_results,dw.circuits,dw.drivers,dw.date,dw.constructors
WHERE dw.race_results.driver_final_pos = 1 AND
      dw.constructors.constructor_id = dw.race_results.constructor_id AND
      dw.date.date_id = dw.races.date_id AND
      dw.race_results.race_id = dw.races.race_id AND
      dw.races.circuit_id = dw.circuits.circuit_id AND
      dw.drivers.driver_id = dw.race_results.driver_id AND
      (dw.circuits.circuit_id,dw.race_results.constructor_id,dw.race_results.driver_id) in
            ( SELECT DISTINCT circuits.circuit_id,constructors.constructor_id,nn.driver_id
              FROM dw.circuits as circuits, dw.constructors as constructors, dw.driver_nationality as nn
              WHERE circuits.nation_id = constructors.nation_id AND nn.nation_id = circuits.nation_id AND nn.nation_id = constructors.nation_id
            )
ORDER BY dw.drivers.name,dw.drivers.surname
```

	name character varying (100)	surname character varying (100)	name character varying (100)	location character varying (100)	name character varying (100)	year integer
1	Alain	Prost	Renault	Le Castellet	French Grand Prix	1983
2	Alain	Prost	Renault	Dijon	French Grand Prix	1981
3	Alberto	Ascari	Ferrari	Monza	Italian Grand Prix	1951
4	Alberto	Ascari	Ferrari	Monza	Italian Grand Prix	1952
5	Bill	Vukovich	Kurtis Kraft	Indianapolis	Indianapolis 500	1954
6	Bill	Vukovich	Kurtis Kraft	Indianapolis	Indianapolis 500	1953
7	Bob	Sweikert	Kurtis Kraft	Indianapolis	Indianapolis 500	1955

Ranking

List drivers ranked by the total position delta (final position - starting position), considering only finished races, including also the average position delta, and only drivers with at least 50 completed races are considered.

```sql
SELECT DISTINCT d.driver_id, d.name, d.surname ,
       SUM(rr.driver_initial_pos-rr.driver_final_pos) as delta_pos,
       AVG(rr.driver_initial_pos-rr.driver_final_pos) as mean_delta_pos
FROM dw.race_results as rr, dw.drivers as d
WHERE rr.driver_id = d.driver_id AND rr.status_id = 1
GROUP BY d.driver_id,d.name,d.surname
HAVING COUNT(DISTINCT rr.race_id) >= 50
ORDER BY delta_pos DESC
```

	driver_id [PK] bigint	name character varying (100)	surname character varying (100)	delta_pos bigint	mean_delta_pos numeric
1	4	Fernando	Alonso	464	1.7777777777777778
2	8	Kimi	Räikkönen	376	1.7013574660633484
3	815	Sergio	Pérez	369	2.1705882352941176
4	18	Jenson	Button	346	2.1490683229813665
5	30	Michael	Schumacher	312	1.4246575342465753
6	20	Sebastian	Vettel	297	1.3141592920353982
7	1	Lewis	Hamilton	290	0.92948717948717948718

Total rows: 47 | Query complete 00:00:00.129

Ranking

List circuits ranked by the average number of overtakes, considering only circuits that have hosted at least 10 races.

```sql
SELECT
    c.circuit_id,
    c.name AS circuit_name,
    COUNT(DISTINCT rr.race_id) AS races_held,
    SUM(ABS(rr.driver_initial_pos - rr.driver_final_pos)) / 2.0 AS total_abs_delta,
    SUM(ABS(rr.driver_initial_pos - rr.driver_final_pos)) / (2.0 * COUNT(DISTINCT rr.race_id)) AS mean_abs_delta_per_race
FROM dw.race_results AS rr
JOIN dw.races AS r
    ON rr.race_id = r.race_id
JOIN dw.circuits AS c
    ON r.circuit_id = c.circuit_id
WHERE rr.status_id = 1
GROUP BY c.circuit_id, c.name
HAVING  COUNT(DISTINCT rr.race_id) > 10
ORDER BY mean_abs_delta_per_race DESC;
```

	circuit_id [PK] bigint	circuit_name character varying (100)	races_held bigint	total_abs_delta numeric	mean_abs_delta_per_race numeric
1	19	Indianapolis Motor Speedway	19	647.5000000000000000	34.0789473684210526
2	15	Marina Bay Street Circuit	15	305.0000000000000000	20.3333333333333333
3	17	Shanghai International Circuit	17	291.0000000000000000	17.1176470588235294
4	20	Nürburgring	41	676.0000000000000000	16.4878048780487805
5	69	Circuit of the Americas	12	196.0000000000000000	16.3333333333333333
6	2	Sepang International Circuit	19	289.5000000000000000	15.2368421052631579
7	3	Bahrain International Circuit	21	315.0000000000000000	15.0000000000000000

Total rows: 32 | Query complete 00:00:00.135

THANK YOU FOR THE ATTENTION!

+123-456-7890

123 Anywhere St., Any City