Data Processing Pipeline

1. Data Layers and How Data Changes (Bronze → Silver → Gold)

The pipeline follows a **Bronze → Silver → Gold** pattern.
Below we describe **what changes** at each step for the main entities.

3.1 Bronze Layer – Raw Structured Data

Goal:
Store the raw CSVs in Delta format with **schema enforcement** and **minimal transformation**.

Tables (examples):

- `f1_bronze_races`
- `f1_bronze_drivers`
- `f1_bronze_constructors`
- `f1_bronze_results`
- `f1_bronze_sprint_results`
- `f1_bronze_lap_times`
- `f1_bronze_pit_stops`
- `f1_bronze_qualifying`
- `f1_bronze_driver_standings`
- `f1_bronze_constructor_standings`
- `f1_bronze_circuits`
- `f1_bronze_seasons`
- `f1_bronze_status`

Quarantine tables for bad records:

- `f1_invalid_results`
- `f1_invalid_sprint_results`
- `f1_invalid_lap_times`

Updates vs raw CSV:

- Enforced schemas (data types explicitly specified).
- Basic validation:
  - **Results & sprint_results**:
    - require non-null `raceId`, `driverId`, `constructorId`
    - only keep rows where `year` ∈ [1950, 2030] (via join with `races`)

    - invalid rows → moved to `f1_invalid_results` / `f1_invalid_sprint_results`
  - **Lap times**:
    - require non-null `raceId`, `driverId`, `lap`
    - allow `milliseconds` to be null or ≥ 0
    - invalid rows → moved to `f1_invalid_lap_times`
- Partitioning:
  - `f1_bronze_races` is partitioned by `year`.
  - Other bronze tables are typically **unpartitioned** to keep Bronze simple.

The idea: **Bronze is close to raw**, but with safe schemas and obvious garbage removed.

3.2 Silver Layer – Cleaned, Joined, Analytics-Ready

Goal:
Create **clean, denormalized tables** that are ready for analytics and modeling.

Core Silver tables:

- `f1_silver_race_results`
- `f1_silver_sprint_results`
- `f1_silver_lap_times`
- `f1_silver_pit_stops`
- `f1_silver_qualifying`
- `f1_silver_driver_standings`
- `f1_silver_constructor_standings`
- `f1_silver_seasons`

3.2.1 Race Results (`f1_silver_race_results`)

Input:

- `f1_bronze_results`
- `f1_bronze_races`
- `f1_bronze_drivers`
- `f1_bronze_constructors`
- `f1_bronze_circuits`
- `f1_bronze_status`

Transformations / Updates:

- Deduplicate:
  - `races`, `drivers`, `constructors`, `circuits`, `status` → `dropDuplicates` on their IDs.
- Create readable `driverName`:

    - `driverName = trim(forename + ' ' + surname)`
- Create readable `constructorName` (team):
    - `constructorName = trim(name)` from constructors.
- Join all dimensions:
    - Join results with races to bring `year` (renamed to `season`), `round`, `raceName`.
    - Join circuits to add `circuitName`, `location`, `country`.
    - Join drivers to add `driverName`, `driverNationality`.
    - Join constructors to add `teamName`, `teamNationality`.
    - Join status to convert `statusId` → human-readable `statusDescription`.
- New columns:
    - `season` (from races.year)
    - `finishTimeStr`, `finishTimeMs` (renamed from raw)
- Partitioning:
    - Table is **partitioned by** `season` for faster year-based analysis.

Result: one **denormalized row per driver–race** with all context (driver, team, track, status).

3.2.2 Sprint Results (`f1_silver_sprint_results`)
Same pattern as race results:

- Reads `f1_bronze_sprint_results` + dimensions.
- Adds `season`, `raceName`, `circuitName`, `driverName`, `teamName`, `statusDescription`.
- Partitioned by `season`.

3.2.3 Lap Times (`f1_silver_lap_times`)
Input:

- `f1_bronze_lap_times`
- `f1_bronze_races`
- `f1_bronze_drivers`

Transformations / Updates:

- Add `season` by joining with races on `raceId`.
- Add `driverName` by joining with drivers on `driverId`.
- Rename columns for clarity:
    - `position` → `lapPosition`
    - `time` → `lapTimeStr`
    - `milliseconds` → `lapTimeMs`
- Partitioning:

    Partitioned by `season`, `raceId` (lap data can be large → good partitioning key).

3.2.4 Pit Stops (`f1_silver_pit_stops`)

Input:

- `f1_bronze_pit_stops`
- `f1_bronze_races`
- `f1_bronze_drivers`

Transformations / Updates:

- Add `season` via join with races.
- Add `driverName` via join with drivers.
- Rename:
  - `time` → `pitTimeStr`
  - `milliseconds` → `pitTimeMs`
- Partitioned by `season` , `raceId` .

3.2.5 Qualifying (`f1_silver_qualifying`)

Input:

- `f1_bronze_qualifying`
- `f1_bronze_races`
- `f1_bronze_drivers`
- `f1_bronze_constructors`

Transformations / Updates:

- Add `season` , `raceName` .
- Add `driverName` , `teamName` .
- Keep `q1` , `q2` , `q3` times.
- Partitioned by `season` .

3.2.6 Driver / Constructor Standings (`f1_silver_driver_standings` , `f1_silver_constructor_standings`)

Input:

- corresponding bronze tables + races + drivers/constructors.

Transformations / Updates:

- Add `season` , `raceName` , `round` .
- Add `driverName` or `teamName` .
- Partition by `season` .

3.2.7 Seasons (`f1_silver_seasons`)

Just cleaned dim from `f1_bronze_seasons` (deduped).

3.3 Gold Layer – Feature-Engineered Tables

Goal:

Provide **high-level aggregates and ML-ready features**.

Gold tables:

- `f1_gold_driver_season_stats`
- `f1_gold_constructor_season_stats`
- `f1_gold_race_driver_features`

3.3.1 Driver Season Stats (`f1_gold_driver_season_stats`)
Input: `f1_silver_race_results` , `f1_silver_driver_standings`
Transformations / Features:
From `race_results_silver` :

- Per (season, driverId, driverName, teamName):
  - `races_count` – total starts
  - `finished_races` – count where `statusDescription == 'Finished'`
  - `total_points`
  - `avg_grid_position` – average grid position (optionally excluding grid=0)
  - `avg_finish_position` – average finishing position (null DNFs ignored)
  - `wins` – count of `position == 1`
  - `podiums` – count of `position <= 3`
  - `dnf_count` – count where `statusDescription != 'Finished'`
  - `points_per_race = total_points / races_count`
  - `avg_grid_vs_finish = avg_grid_position - avg_finish_position`
  - (negative → tends to gain positions)

From `driver_standings_silver` :

- For the **final race** of a season:
  - `final_champ_points`
  - `final_champ_position`
  - `season_wins_recorded` (from standings table)
- Joined to driver stats based on `(season, driverId)` .

Missing championship fields are filled with 0 to simplify downstream analysis.

3.3.2 Constructor Season Stats (`f1_gold_constructor_season_stats`)
Input: `f1_silver_race_results` , `f1_silver_constructor_standings`
Transformations / Features:
From `race_results_silver` :

- Per (season, constructorId, teamName):
  - `entries_count` – total car-race entries
  - `team_total_points`
  - `wins` – count of `position == 1`

- `podiums` – count of `position <= 3`
- `dnf_count` – count of non-finished entries

From `constructor_standings_silver` (final race):

- `final_cons_points`
- `final_cons_position`
- `season_cons_wins_recorded`

Null championship fields are filled with defaults (0 / 0.0).

3.3.3 Race-Driver Features (`f1_gold_race_driver_features`)

Input:

- `f1_silver_race_results`
- `f1_silver_lap_times`
- `f1_silver_pit_stops`
- `f1_silver_qualifying`
- `f1_silver_sprint_results`
- `f1_silver_driver_standings`

Transformations / Features:

From **lap times** (per season, raceId, driverId):

- `lap_count_recorded`
- `best_lap_ms`
- `avg_lap_ms`

From **pit stops**:

- `pit_stop_count`
- `avg_pit_stop_ms`
- `total_pit_stop_ms`

From **qualifying**:

- `quali_best_position`

From **sprint results**:

- `sprint_grid`
- `sprint_finish_position`
- `sprint_points`

From **driver standings** (using a window over (season, driverId) ordered by round):

- `champ_points_after_race`
- `champ_position_after_race`
- `prev_champ_points` – previous race championship points

- `prev_champ_position` – previous race championship position

From **race results**:

- context columns: `season`, `raceId`, `round`, `raceName`, `circuitName`, `country`
- driver/team info: `driverName`, `teamName`, nationalities
- race outcome:
  - `grid`
  - `race_finish_position`
  - `positionText`
  - `positionOrder`
  - `race_points`
  - `race_laps`
  - `statusDescription`

Handling missing values:
To simplify downstream EDA and modeling, numeric nulls are filled with 0, e.g.:

- no pit stops → `pit_stop_count = 0`, `avg_pit_stop_ms = 0`, etc.
- no sprint race → `sprint_points = 0`, etc.
- no standings row (e.g., early seasons) → championship fields = 0.

Partitioning:

- `f1_gold_driver_season_stats` → partitioned by `season`
- `f1_gold_constructor_season_stats` → partitioned by `season`
- `f1_gold_race_driver_features` → partitioned by `season`

---

2. Storage Architecture & Justification

Storage:
All data is stored in **Databricks File System (DBFS)** as **Delta tables** (no S3 needed).
Why Delta + Bronze/Silver/Gold?

- **Delta Lake**
  - ACID transactions → safe concurrent reads/writes.
  - Time travel → easier debugging and reproducibility.
  - Schema enforcement & evolution → protects against bad data.
- **Medallion architecture (Bronze / Silver / Gold)**
  - Bronze: raw/landing, keeps you close to source data.
  - Silver: cleaned, integrated layer → business logic is centralized.
  - Gold: optimized for specific analytics/ML tasks.

Partitioning strategy:

- Many queries filter by **season** (year) → tables partitioned by `season`.
- Lap and pit-stop data can be large and is naturally accessed by race → partitioned by `season`, `raceId`.
- This reduces the amount of data scanned and improves performance for typical analytics workloads.