Formula 1 Database System
Database Implementation Report

[Database design overview and schemas] (1-2 short paragraphs) [20]
[Feature descriptions] (bullet points with description) [20]

Names	Roll numbers
Muhammad Bazaf Shakeel	26100146
Muhammad Abdullah	26100192
Sulaiman Ahmad	26100350
Zaeem Mohtashin khan	26100107

Database Design Overview and Schemas

The F1 Database System is a comprehensive relational database designed to capture and analyze all aspects of Formula 1 racing data, from championship seasons down to individual lap telemetry. The database integrates real-time data from the OpenF1 API and maintains over 1 million records across 15+ normalized tables following Third Normal Form (3NF) principles. The schema is organized into four logical layers: Core Entities (Season, Event, Session, Driver, Team, Circuit), Competition Data(SessionResult, Contract), Performance Metrics (Lap, Stint, PitStop), and Telemetry & Analytics (PositionTick, IntervalTick, RaceControlEvent, TeamRadio). All tables enforce referential integrity through foreign key constraints, with cascading rules for data consistency.

The database architecture supports complex analytical queries through a sophisticated hierarchy: Seasons contain Events (Grand Prix weekends), Events contain Sessions (Practice, Qualifying, Race), and Sessions contain detailed performance data. The design handles temporal relationships through Contract tables that manage driver-team associations across seasons. Scalability is achieved through strategic indexing on high-cardinality columns (session_id, driver_id, event_id) and composite indexes on frequently joined columns. The schema accommodates both historical analysis and real-time race monitoring, with time-series tables (PositionTick, IntervalTick) capturing minute-by-minute race progression. Data validation is enforced at multiple levels: database constraints prevent invalid relationships, application-level filters handle incomplete API data, and custom JSON converters gracefully manage inconsistent API responses.

Stored Procedures

- sp_GetDriverPerformance – returns a driver's season performance including wins, podiums, points, DNFs, and averages.
- sp_GetHeadToHead – compares two drivers race-by-race and summarizes who performed better.
- sp_GetCircuitStats – shows top winners and fastest laps for a selected circuit.

Functions

- fn_GetPoints – scalar function that returns FIA points for a finishing position.
- fn_GetDriverYearResults – table-valued function that returns all race results for a driver in a specific year.

Triggers

- trg_AfterRaceControlInsert – logs new Race Control events into a log table after insertion.
- trg_AfterDriverUpdate – automatically updates full_name when driver's first or last name changes.

Views

- vw_EventSchedule – displays season schedule with GP info and session count.
- vw_DriverStandings – calculates driver points, wins, podiums, and championship rank.
- vw_TeamStandings – calculates constructor points and rankings.
- vw_QualifyingResults – shows qualifying positions, lap times, and gaps.
- vw_RaceResults – displays race results with status, gaps, and awarded points.
- vw_FastestLaps – extracts the fastest lap per session using ranking.
- vw_Last5Races – shows last five races for each driver.
- vw_AvgTeamPitStops – calculates average pit stop duration per team.

Common Table Expressions (CTEs)

- Used in vw_DriverStandings to compute points and rankings.
- Used in vw_TeamStandings for constructor aggregation.
- Used in vw_FastestLaps to rank and select fastest laps.
- Used in sp_GetHeadToHead to compute per-driver race data before comparison.

Indexes

- Covering index on Lap for faster lap-time queries.
- Filtered index on RaceControlEvent for flag-only entries.
- Indexes on meeting_key and session_key for fast API lookups.
- Indexes on PitStop, Stint, and SessionResult to optimize joins.
- Index on driver_number for quick driver identification.

Table Partitioning

- Partitioned table PositionTick_Partitioned created to store time-series data split by year using the partition_year computed column.

Steps to Populate Database:

After running the primary SQL script, run the following commands in a terminal:

```
cd F1DataLoader
dotnet restore
dotnet build
```

Then edit `Program.cs` and update the connection string.

Finally, run these two one by one:

```
dotnet run load-season 2023 --time-series
dotnet run load-season 2024 --time-series
```

(running either of these should take approx 30 minutes since there is a lot of data to load)