Premier League Data Pipeline Documentation

1. Objectives

The purpose of this pipeline is to:

1. **Automate Data Collection** – Fetch real-time Premier League data (standings, fixtures, results, stats) from the RapidAPI Premier League endpoint.

2. **Data Storage & Persistence** – Store the collected data into a structured **MySQL database** for easy querying, historical analysis, and integration with BI tools.

3. **Ensure Data Quality** – Standardize and clean data before storage, ensuring consistent formats for dates, team names, and statistics.

4. **Enable Analytics & Insights** – Provide a foundation for building dashboards, reports, and machine learning models for match predictions, performance tracking, and fan engagement.

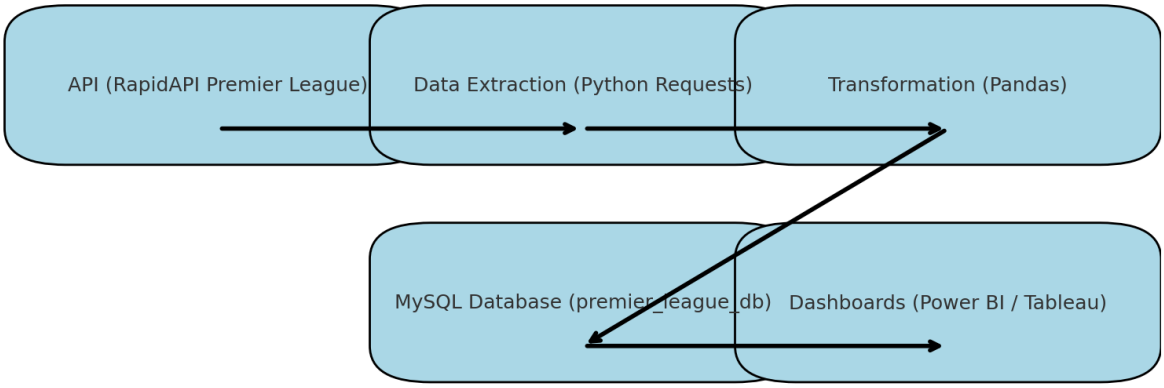
5. **Scalability** – Design the pipeline to handle future data sources (e.g., player statistics, injuries, transfer data) and support scheduled runs.

---

2. Pipeline Workflow

Dataflow Diagram

Premier League Data Pipeline Flow



Step 1 – Environment Setup

- Configuration is stored in a `.env` file to keep sensitive information secure:
    - **API credentials** for RapidAPI (key, host).
    - **MySQL credentials** (host, user, password, database, port).

This ensures **separation of concerns**, allowing easy updates without modifying code.

Step 2 – Data Extraction

- The pipeline connects to the **RapidAPI Premier League Standings API**:
    - Retrieves JSON responses containing **league standings, match details, and statistics**.
- API requests are authenticated using the **API_KEY** and **API_HOST** from `.env`.

Step 3 – Data Transformation

- API response is parsed and cleaned:
    - Convert JSON into **Pandas DataFrames**.
    - Normalize nested structures (teams, fixtures, results).
    - Standardize date formats (UTC → local if required).
    - Ensure numeric fields (goals, wins, losses) are integers.
    - Handle missing or inconsistent data.

Step 4 – Data Loading (ETL to MySQL)

- The processed DataFrame is written into a **MySQL database**:
    - Database: `premier_league_db`
    - Typical tables:
        - `teams` (team_id, name, stadium, coach, etc.)
        - `standings` (season, team_id, wins, losses, draws, points, rank, etc.)
        - `matches` (match_id, home_team, away_team, score, date, venue, status).
- SQLAlchemy or `mysql-connector-python` ensures safe inserts/updates.

Step 5 – Scheduling & Automation

- The notebook can be:

- Converted into a **Python script**.
- Scheduled with **cron jobs** (Linux) or **Task Scheduler** (Windows).
- Orchestrated with **Airflow**/**Prefect** for advanced scheduling, retries, and monitoring.

Step 6 – Downstream Applications

Once the pipeline runs successfully, the MySQL database can serve as a **single source of truth** for:

- **Power BI Dashboards** – visualize standings, match results, trends.
- **Data Science Models** – predict match outcomes, player performance.
- **APIs / Web Apps** – expose endpoints for fan engagement platforms.

---

3. ⚙️ Technologies Used

- **Python** – ETL scripting and data transformation.
- **RapidAPI** – Data source for Premier League stats.
- **MySQL** – Relational database for structured storage.
- **SQLAlchemy / mysql-connector-python** – Database connection and persistence.
- **Pandas** – Data parsing, cleaning, and transformation.
- **dotenv** – Secure credential management.