⚽ Technical Assessment: Soccer Scouting Data Pipeline

Project Overview

The goal is to develop an end-to-end pipeline that ingests match and player data, transforms it into an analytics-ready schema, and generates actionable scouting insights. We are looking for clean, modular Python code and a sensible approach to data modeling.

Technical Stack

- **Language:** Python 3.9+
- **Storage:** SQLite or DuckDB
- **Processing:** Pandas or Polars
- **Validation:** you can use things like dbt-core, Great Expectations, or Pydantic
- **Testing:** pytest

Data Sources

You are provided with five core datasets. You will need to manage varied formats and specific data quality hurdles.

File	Format	Description
teams.csv	CSV	Reference data for 10 teams
players.csv	CSV	Player profiles (Market value, DOB, position)
matches.json	JSON	Match metadata (Scores, venues, dates)
player_match_stats.csv	CSV	Aggregate match stats (Goals, xG, etc.)
match_events.json	JSON	Event-stream data (Individual passes/shots)

Known Quality Issues to Resolve:

- **Date Formats:** Mixed YYYY-MM-DD and DD/MM/YYYY strings.

- **Categorical Noise:** Inconsistent position labels (e.g., "GK" vs "Goalkeeper").
- **Logic Errors:** Negative xG values or xG exceeding the total shots taken.
- **Integrity Issues:** Orphaned records and duplicate event_ids.
- **Nulls:** Roughly 15% missing data in market_value and contract_until.

Suggested pipeline structure

This is a real structure case scenario. Here you can find a sample data pipeline structure that is suitable to this problem

OBS: You can take a different approach, this is just a simple suggestion

STEP 1	**STEP 2**	**STEP 3**	**STEP 4**	**STEP 5**
EXTRACT	**VALIDATE**	**TRANSFORM**	**LOAD**	**REPORT**
📥 **Input:** Raw Data	🔍 **Check:** Quality	🛠️ **Process:** Logic	💾 **Store:** DB	📈 **Output:** Insights
CSV & JSON	*Schema & Rules*	*Cleaning & Per-90*	*SQLite / DuckDB*	*Scouting Metrics*

Tasks

Part 1: Ingestion & Data Integrity

Objective: Build a Python-based ingestion module to move raw files into your local database.

Requirements:

- **Normalization:** Standardize all dates and map positions to four categories: *Goalkeeper, Defender, Midfielder, Forward*.
- **Cleaning:** Filter out orphaned records and resolve duplicates.
- **Logging:** Provide a summary of data quality issues (counts and examples).
- **Idempotency:** Ensure the script can be run multiple times without corrupting or duplicating data.

Part 2: Analytics & Scouting Logic

Objective: Enable the recruitment team to query the processed data.

2.1 Player Performance View

Create an aggregated view of player stats for the season, including:

- **Cumulative:** Total matches, minutes, goals, assists, xG, xA.
- **Efficiency (Per 90):** Goals, assists, xG, xA, key passes, and defensive actions (tackles + interceptions).
- **Success Rates:** Pass completion %, shot accuracy %, and duel win rate %.

2.2 Scouting Search Engine

Develop a function to filter players based on specific recruitment profiles (you can decide custom cases)

- **Example Case:** Identify Midfielders under 25 with 80%+ pass completion and >1.5 key passes per 90.
- **Parameters:** Position, age, minimum minutes, performance thresholds, and market value.

2.3 Team Comparison

A summary query showing team-level output, average pass accuracy, and the top 3 players by goal contributions (Goals + Assists).

Part 3: Production Deployment Strategy

Objective: Demonstrate your understanding of how this solution would move from local development to a production environment.

Requirements:

Provide a written explanation (no code required) covering the following concepts:

- **Local Development to Production Timeline:** Describe the steps and considerations for moving your code from your local machine to a production environment.
- **CI/CD Pipeline:** Explain how you would implement continuous integration and continuous deployment for this data pipeline.

- **Orchestration:** Describe how you would schedule and orchestrate the pipeline in production (e.g., tools like Airflow, Prefect, or cloud-native solutions).
- **Monitoring & Reporting:** Explain your approach to monitoring pipeline health, data quality, and generating reports for stakeholders.
- **Infrastructure Considerations:** Discuss database hosting, compute resources, scalability, and any cloud services you would leverage.

⚠️ Important Note on Documentation

We understand that time constraints and lack of infrastructure may prevent you from implementing certain features or best practices. Please document anything you would have liked to implement but could not due to these limitations.

Specifically, include:

- Features or improvements you would add with more time
- Infrastructure or tooling you would use in a real production environment
- Additional validation, testing, or monitoring you would implement
- Any technical debt or shortcuts taken, with explanations of how you would properly address them

This documentation should be detailed and demonstrate your understanding of production-grade data engineering practices. Include this in your README.md or a separate FUTURE_IMPROVEMENTS.md file.

Submission Guidelines

Project Structure

```
your-solution/
├── README.md
├── requirements.txt
├── data/
│   └── raw/          # Provided source files
├── src/
│   ├── ingest.py
│   └── analytics.py
│   └── ....
├── tests/
│   └── test_*.py     # Part 3: Unit tests
└── output/
    └── sample_outputs.txt  # Query results
```

Submission Process

Push your finalized code to a GitHub repository and share it with the evaluators

Evaluation Criteria

Your submission will be evaluated on:

- **Code Quality:** Clean, modular, well-documented Python code
- **Data Modeling:** Sensible schema design and normalization
- **Data Quality:** Effective handling of quality issues and validation
- **Analytics:** Accurate and useful scouting insights
- **Testing:** Appropriate test coverage
- **Production Readiness:** Understanding of deployment, CI/CD, and production best practices
- **Documentation:** Clear README and thorough explanation of design decisions and future improvements