Football leagues dashboard

The purpose of this project is to build a complete end-to-end data pipeline to ingest, transform, and visualize sports data from **Latin American football leagues**. The architecture follows a modern **Data Lakehouse** approach based on the **Medallion Architecture** (bronze, silver, and gold layers), leveraging the capabilities of **Azure** and **Databricks** for scalable and reliable data processing.

The data source is the public **TheSportsDB REST API** (https://www.thesportsdb.com/), from which **league standings** were extracted automatically. This extraction process was orchestrated using **Azure Data Factory**, allowing scheduled and parameterized API calls that retrieve structured data.

The data is stored in two main locations:

- **Azure Blob Storage** – used for the raw ingestion (bronze layer)
- **Azure Delta Lake** – used for the curated silver and gold layers, taking advantage of Delta format features such as ACID transactions, schema enforcement, and time travel.

All data transformation, validation, and enrichment tasks were performed in **Databricks using PySpark**, ensuring scalability, maintainability, and performance. The curated data from the gold layer is then served as the trusted source for a **Power BI dashboard**, which presents key metrics like team standings, points, goals, and other performance indicators in real time.

This pipeline provides a reusable and scalable framework to integrate more leagues or sports in the future and establishes a strong foundation for advanced sports analytics across the Latin American region.

The first phase of the project focused on establishing a structured and automated data ingestion pipeline to collect and stage football league standings from Latin American competitions.

Step-by-step Breakdown:

Storage Layer Setup
A dedicated **Azure Blob Storage container** was created and configured as a **Delta Lake**. This setup enabled the use of Delta format features such as:

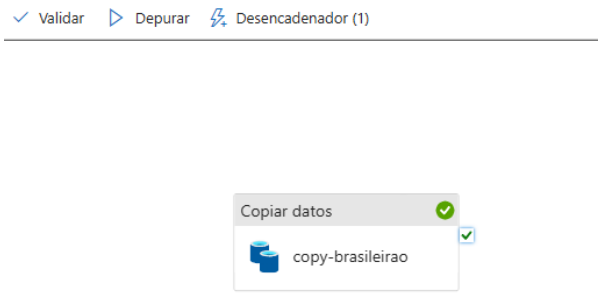
- ACID transactions
- Schema enforcement
- Time travel and versioning
- Efficient querying and updates

REST API Ingestion via Azure Data Factory
Azure Data Factory (ADF) was used to orchestrate the extraction of data from **TheSportsDB REST API**. ADF pipelines were configured to:

- Perform authenticated GET requests to the API
- Extract league standings in JSON or CSV format
- Store the raw data directly into the **Delta Lake container** on **Azure Blob Storage**

Copy Data Activity
Within ADF, a **Copy Data activity** was implemented to move and stage the extracted data. This step ensured schema mapping, format conversion (to Delta), and proper partitioning as needed. It acted as the ingestion mechanism populating the **bronze layer** of the Medallion architecture.

Scheduled Automation
The data extraction process was fully automated using **ADF pipeline triggers** configured with **crontab expressions**, enabling daily ingestion. This ensured that the system remained updated with the most recent standings, without manual intervention.



After the raw JSON data was landed in **Azure Blob Storage** by Azure Data Factory, a **streaming ingestion pipeline** was implemented in **Databricks** to continuously load and normalize this data into the **Bronze layer** using **Delta Lake format**.

Streaming Pipeline Design

- The incoming JSON files were treated as a **streaming source** using **Structured Streaming** in PySpark.
- This approach allows Databricks to monitor the landing folder continuously and ingest new or updated files incrementally.
- The stream handles schema inference (or applies a predefined schema) and writes to the Bronze table in **Delta format**.

Secure Access via External Location

To enable access between **Azure Blob Storage** and **Databricks**, the connection was established using:

- An **External Location** with an associated **Credential** backed by an **Azure IAM Role**.

- This allows fine-grained permission control and avoids hardcoding secrets or keys.

Operational Setup

- The stream was likely configured to run on a **scheduled basis** (e.g., daily) using **trigger(once=True)** or continuously if near real-time was needed.
- **Checkpointing** ensures that files are not reprocessed and provides fault tolerance in case of job failures.
- This approach guarantees **idempotent ingestion** and **incremental data loading** into the Bronze table.

After the initial ingestion of raw JSON data into the **Bronze layer**, the next step in the Medallion architecture was to **refine, clean, and normalize** the data in the **Silver layer**. This phase focused on transforming semi-structured data into well-defined, relational-style tables that enable efficient querying and modeling for downstream analytics.

Bronze to Silver Transformation

Using **Databricks notebooks with PySpark**, the following steps were carried out:

- Read the **Delta table from the Bronze layer**, which contained raw match standings data for Latin American football leagues.
- **Data cleansing** operations were applied:
    - Removing nulls or irrelevant records
    - Casting data types
    - Filtering out incomplete entries
- Columns were selected and **renamed for clarity** and standardization.

Dimensional Normalization: Teams and Leagues

To organize the data following best practices in data warehousing, two core **dimension tables** were created in the Silver layer:

dim_teams

- Extracted unique team identifiers (idTeam), names, logos (strBadge), and metadata.
- Ensured deduplication using .dropDuplicates() or .distinct().
- This allows joining standings to team details without duplication of team attributes.

dim_leagues

- Extracted league-level data such as league ID, name, and associated country.
- This enables filtering or aggregating standings across different competitions or regions.

Both dimensions were saved as **Delta tables** in the Silver layer, forming the backbone for dimensional modeling.

Fact Table: Standings by League

In addition to the dimension tables, a **fact-like Silver table** was created, containing cleansed league standings per team and season. This table:

- Links to dim_teams and dim_leagues via foreign keys (idTeam, idLeague)
- Contains performance metrics like:
    - Matches played
    - Points earned
    - Goal difference
    - Wins, losses, draws

This structured design enables flexible aggregations, filtering by country or league, and ranking operations for dashboarding.

The Gold Layer

Represents the most refined stage of the Medallion architecture, designed to serve curated, analysis-ready data directly to the dashboard layer—in this case, **Power BI**. This phase focuses on merging and modeling normalized data into a single, optimized structure for easy and performant consumption.

Consolidating Data Through Joins

Using **Databricks with PySpark**, joins were performed between previously created Silver Layer tables:

- fact_standings: contains performance metrics per team and league
- dim_teams: includes descriptive team data such as names and logos
- dim_leagues: includes league metadata such as league name and country

These joins were based on foreign keys:

- idTeam → links to dim_teams
- idLeague → links to dim_leagues

This merging process allowed the creation of a **flattened, denormalized dataset** that combines performance and descriptive attributes into a single table.

Selecting Relevant Columns for the Dashboard

After joining, only the necessary fields were selected for the final output—those intended to be visualized in **Power BI**:

- Team Info: strTeam, strBadge
- League Info: strLeague, strCountry
- Match Metrics: intPts, intPlayed, intGoalDifference, intWins, intLosses

The data was also **sorted** by league and total points in descending order to support leaderboard visualizations.

Integration with Power BI and Automation

After building and modeling the curated Gold Layer data, the final phase of the project focused on enabling seamless access for analytics and automating the full data pipeline.

Native Power BI Integration with Databricks

A **native connector** between **Power BI** and **Azure Databricks** was configured using:

- A **personal access token (PAT)** or
- A combination of **Databricks Key and Secret Key**

This connection allowed Power BI to directly query the **Gold Delta table** (gold.dashboard_standings), ensuring:

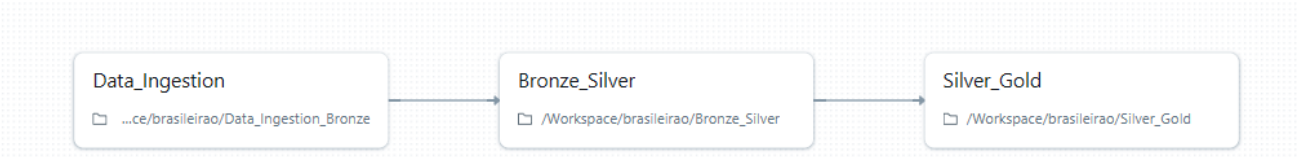
- Real-time or scheduled access to clean, production-ready data
- Elimination of redundant data loading or transformation in Power BI
- Fully cloud-native, scalable architecture

Automated Execution Using Databricks Jobs

The Databricks notebooks (covering ingestion, normalization, and modeling) were orchestrated using **Databricks Jobs**, with a **weekly schedule** configured via:

- Job UI or REST API
- Cron expression for weekly execution
- Defined task dependencies to maintain proper flow (Landing → Bronze → Silver → Gold)

This ensured that the latest data was ingested and processed **once a week**, automatically.



Data Testing and Alerting via Email

To ensure **data consistency and reliability**, automated **data quality tests** were implemented as part of the pipeline. These tests checked for:

- Schema mismatches

- Null values in critical columns
- Unexpected metric drops (e.g., points or matches played)

When inconsistencies were detected, a **notification email** was automatically sent to the data engineering team. This was accomplished using:

- Python smtplib or integration with Azure Logic Apps
- A conditional trigger at the end of the testing notebook