AMERICAN UNIVERSITY OF SCIENCE & TECHNOLOGY

FACULTY OF ARTS AND SCIENCES

DEPARTMENT OF COMPUTER SCIENCE



CSI 316L – Database Systems Lab

**Lebanese Football Association Soccer Database**

Prepared by:

Peter Bou Ghannam : 22230066

Robert Abou Nader: 22130054

Mohamad Ali El Bakawi : 22230060

Marc Elio Awad : 22130091

Submitted to

Mr. Samir Saad

Zahle, Lebanon

Tuesday, May 28, 2024

Abstract

This project talks about a Database System Built in Access focused on The Lebanese Football Association, its design, and use.

1. **Title**: Lebanese Football Association Soccer Database
2. **General Topic:** This access Database System includes many forms, queries, tables and reports and real data based upon the Lebanese Premiere League generated from a specifically built Python Scraper. And the System include forms as a GUI for the user to login, navigate between features, fill, read, delete, update data and view statistics and useful information.
3. **Problem:** In Lebanon, there is no local, non-foreign Entity that documents data about the Lebanese football association in a public and relational database manner to be used for enhancing the performance, management, and enjoyment of the sport at various levels.
4. **Method**: This database was created using Microsoft Access, utilizing most of its potential, to design tables, queries, forms, navigation menus, reports, charts and graphs with deep focus on functionality and ease of access for the user with an even deeper focus on validation on entering and storing data, in addition to using real Lebanese data using various methods like scrapping public data from a website called :

<https://www.transfermarkt.com/>

Or gathering data and information through research using Wikipedia and Trusted Lebanese News / Sports Websites and utilizing excel and filling the data manually.

1. **Results:** The Lebanese Football Association Soccer Database system turned out to be a user-friendly on the Graphical user Interface side for its ability to allow the user to interact with the Database with no problems, and on the Tables and data side, it can be a reliable source of information for use in other projects because of the data integrity, reliability, the relationships, and the validation implementation.
2. **Conclusion:** As a conclusion, this System proved effective to hold the data and use it in a useful manner to generate statistics and other information and display them in an informative way, and this database can be considered a founding stone for a public Lebanese Database, that is bigger, more detailed and can provide actionable insights that help clubs, coaches, players, fans, and administrators make better decisions.

# Table of Contents

Abstract…………………………………………………………………………………………….2

[Table of Contents 4](#_Toc167962314)

[I. Introduction 5](#_Toc167962315)

[II. Objectives 5](#_Toc167962316)

[III. Background 6](#_Toc167962317)

[IV. Design and Procedures: 7](#_Toc167962318)

[A. Relational Model: 7](#_Toc167962319)

[B. E/R diagram 10](#_Toc167962320)

[C. Normalization 11](#_Toc167962321)

[D. Forms Used 13](#_Toc167962322)

[E. Reports Used 22](#_Toc167962323)

[F. Queries 27](#_Toc167962324)

[G. Python Scrapper 32](#_Toc167962325)

[H. Validation rules: 40](#_Toc167962326)

[V. POSTER: 43](#_Toc167962327)

[VI. Problems faced and solutions 45](#_Toc167962328)

[VII. Project Location 45](#_Toc167962329)

**References**…………………………………………………………………………………

**Appendix A**

# Introduction

This Project includes a Microsoft Access Database System, about the Lebanese football association, its players, clubs, coaches, referees, games, stadiums the competitions in it, all organized and made into a relational database with an appropriate Graphical user interface built on Forms that involves a login screen, a dashboard, and dozens of submenus, each written and connected using VBA and Access built in Macro Builder

It also includes a python scrapper written in python and executed in Jupyter Notebook kernel, that scraps some of the public data that are used in the project from <https://www.transfermarkt.com/> which goes with agreement to the website’s policy.

# Objectives

The objective behind the Lebanese Football Association Soccer Database project is to have:

* + A Relational Database model of the Lebanese football association with a minor focus on the Lebanese premiere league.
  + One to one, one to many relationships with data integrity.
  + Validation, both in tables and in forms.
  + Use of most of the features provided by Microsoft Access, including Forms, Reports, tables, Query Wizard to set up queries, Lookup wizard to set the relationships, custom display controls for Datasheets (combo boxes, yes/no options...), sub forms inside reports and forms.
  + A Login Menu to enter the Database.
  + A Built in Navigation Menu to interact with the various parts of system.
  + The ability to switch between menus very easily.
  + Master Form to enter Data into the Tables with full validation in all forms
  + A python scrapper that can get information from a sports website to assist in the data entry for the Players.
  + Interesting queries that can give useful information and insights to be used in real life.

# Background

The Lebanese football league system, also known as the Lebanese football pyramid, is a series of [interconnected leagues](https://en.wikipedia.org/wiki/League_system) for men's [association football](https://en.wikipedia.org/wiki/Association_football) clubs in [Lebanon](https://en.wikipedia.org/wiki/Lebanon).

The Lebanese football pyramid is composed of five leagues, all governed by the [Lebanese Football Association](https://en.wikipedia.org/wiki/Lebanese_Football_Association), which operate on a system of [promotion and relegation](https://en.wikipedia.org/wiki/Promotion_and_relegation). The first tier of Lebanese football is the [Lebanese Premier League](https://en.wikipedia.org/wiki/Lebanese_Premier_League), which is made up of 12 teams.[[1]](https://en.wikipedia.org/wiki/Lebanese_football_league_system#cite_note-1) Next is the [Lebanese Second Division](https://en.wikipedia.org/wiki/Lebanese_Second_Division), which also has 12 teams.[[2]](https://en.wikipedia.org/wiki/Lebanese_football_league_system#cite_note-2) Both of these leagues cover the whole of Lebanon. The first two leagues participate in the [Lebanese FA Cup](https://en.wikipedia.org/wiki/Lebanese_FA_Cup), the national domestic cup competition.[[3]](https://en.wikipedia.org/wiki/Lebanese_football_league_system#cite_note-3)

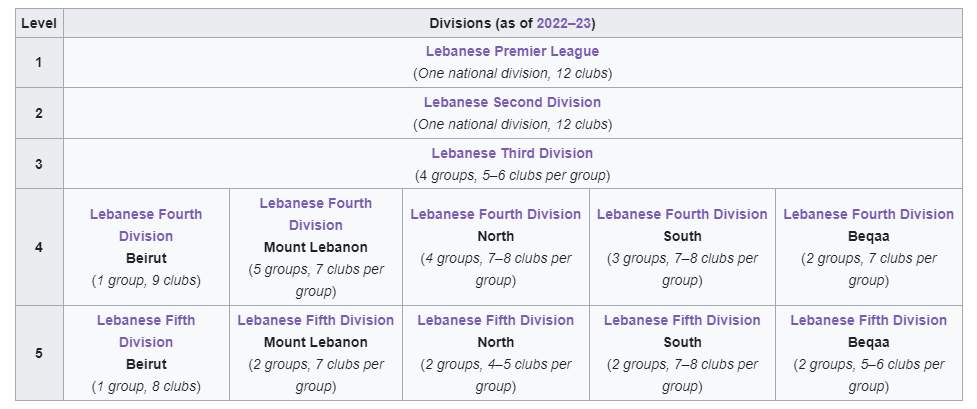


Figure 1 - Lebanese Divisions

The Lebanese First Division ([Arabic](https://en.wikipedia.org/wiki/Arabic_language): الدوري اللبناني الدرجة الأولى), commonly known as the Lebanese Premier League is the top division of the [Lebanese football league system](https://en.wikipedia.org/wiki/Lebanese_football_league_system). There are 12 teams competing in the league, which operates on a system of [promotion and relegation](https://en.wikipedia.org/wiki/Promotion_and_relegation) with the [Lebanese Second Division](https://en.wikipedia.org/wiki/Lebanese_Second_Division).

The league's first season began in May 1934, with [Nahda](https://en.wikipedia.org/wiki/Al_Nahda_SC) winning the maiden title. [Ansar](https://en.wikipedia.org/wiki/Al_Ansar_FC) is the most successful club in the league having won 14 titles; they also set a [*Guinness World Record*](https://en.wikipedia.org/wiki/Guinness_World_Records) by winning 11 consecutive league titles between 1988 and 1999.[[b]](https://en.wikipedia.org/wiki/Lebanese_Premier_League#cite_note-2) The league has featured a "split" system since 2020, in which the season is divided in two phases.

# Design and Procedures:

## Relational Model:

* The Database was made in Microsoft access, it involves the following tables with the following relational model:

1)  **tbl\_login**

UserID (AutoNumber PK)

FirstName (Short Text)

LastName (Short Text)

UserName (Short Text)

Password (Short Text)

2) Players

Player\_ID (AutoNumber PK)

First\_Name (Short Text)

Last\_Name (Short Text)

Current\_club\_ID (Number FK)

Country\_Birth (Short Text)

Date\_of\_Birth (Date/Time Extended)

Position (Short Text)

foot (Short Text)

height\_in\_cms (Number)

contract\_begins\_date (Date/Time)

current\_market\_value (Currency)

3) clubs

club\_ID (Number PK)

club\_code (Short Text)

club\_name (Short Text)

Competition\_ID (Number FK)

Coach\_ID (Number FK )

4) Coaches

Coach\_ID (Number PK)

Coach\_Name (Short Text)

Nation (Short Text)

5) Stadiums

Stadium\_ID (AutoNumber PK)

Stadium\_Name (Short Text)

Stadium\_City (Short Text)

Stadium\_Capacity (Number)

6) Matches

Match\_Num (AutoNumber PK)

Competition\_Name (Number FK)

season (Short Text)

Round (Number)

home\_club (Number)

home\_club\_goals (Number)

away\_club (Number)

away\_club\_goals (Number)

Stadium (Number FK)

Referee\_ID (Number FK)

7) Referee

Referee\_ID (AutoNumber PK)

Referee\_Name (Short Text)

Country (Short Text)

Debut\_Date (Date/Time)

Seasons\_Played (Number)

Appearances (Number)

Yellow\_Cards\_Given (Number)

Second\_Yellow\_Cards\_Given (Number)

Red\_Cards\_Given (Number)

Penalty\_Kicks\_Awarded (Number)

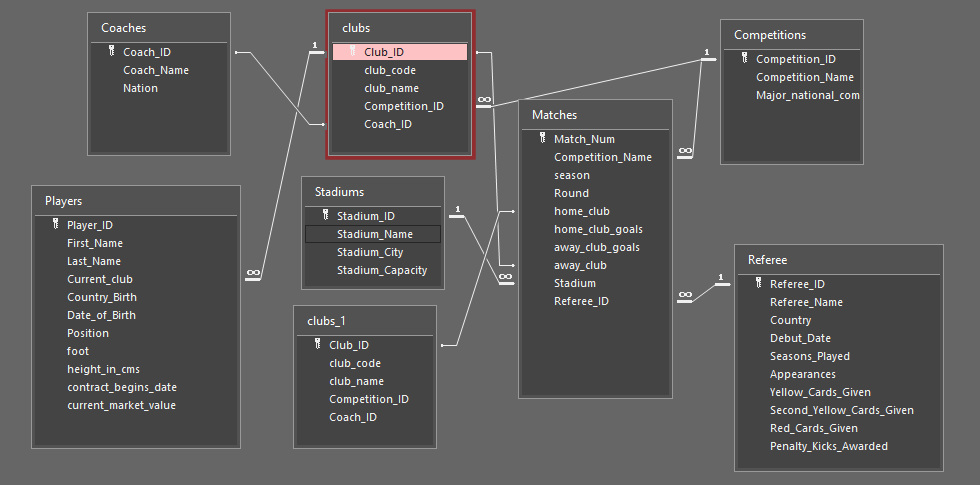
8) Competitions

Competition\_ID (AutoNumber PK)

Competition\_Name (Short Text)

Major\_national\_competition (Yes/No)

## E/R diagram



Each club has many Players, each player can only be in one club

Each club has one optional coach (in case of coach change) , each coach can be In one club

Each club can be part of one competition (like division), (on relegation it changes), each competition can have many clubs in it

Each unique match played can be part of one competition, a competition can have many matches.

In Lebanon, each unique match is hosted by a referee, and one referee can host many unique matches

Each match is played in one stadium, a stadium can have many matches played in it.

Each match home club can be one unique club, and each match away club can be one unique club

## Normalization

**Levels of normalization included:**

**First Normal Form (1NF)**

* **1NF** requires that all table columns contain only atomic (indivisible) values, and each column contains values of a single type.

**Second Normal Form (2NF)**

* **2NF** requires the database to be in 1NF and that all non-key attributes are fully functionally dependent on the primary key. This means there should be no partial dependency of any column on the primary key.

**Third Normal Form (3NF)**

* **3NF** requires the database to be in 2NF and that all non-key attributes are not transitively dependent on the primary key. This means that no non-key column should depend on another non-key column.

1. **tbl\_login**:
   * Complies with 1NF as all attributes are atomic.
   * Complies with 2NF and 3NF as there are no partial or transitive dependencies.
2. **Players**:
   * Complies with 1NF.
   * Complies with 2NF as **Player\_ID** is the primary key and all other attributes depend on it.
   * Complies with 3NF as there are no transitive dependencies among the non-key attributes.
3. **clubs**:
   * Complies with 1NF.
   * Complies with 2NF and 3NF for the same reasons as above.
4. **Coaches**:
   * Complies with 1NF.
   * Complies with 2NF and 3NF.
5. **Stadiums**:
   * Complies with 1NF.
   * Complies with 2NF and 3NF.
6. **Matches**:
   * Complies with 1NF.
   * Complies with 2NF as **Match\_Num** is the primary key and all attributes depend on it.
   * Complies with 3NF as there are no transitive dependencies among the non-key attributes.
7. **Referee**:
   * Complies with 1NF.
   * Complies with 2NF and 3NF.
8. **Competitions**:
   * Complies with 1NF.
   * Complies with 2NF and 3NF.

## Forms Used

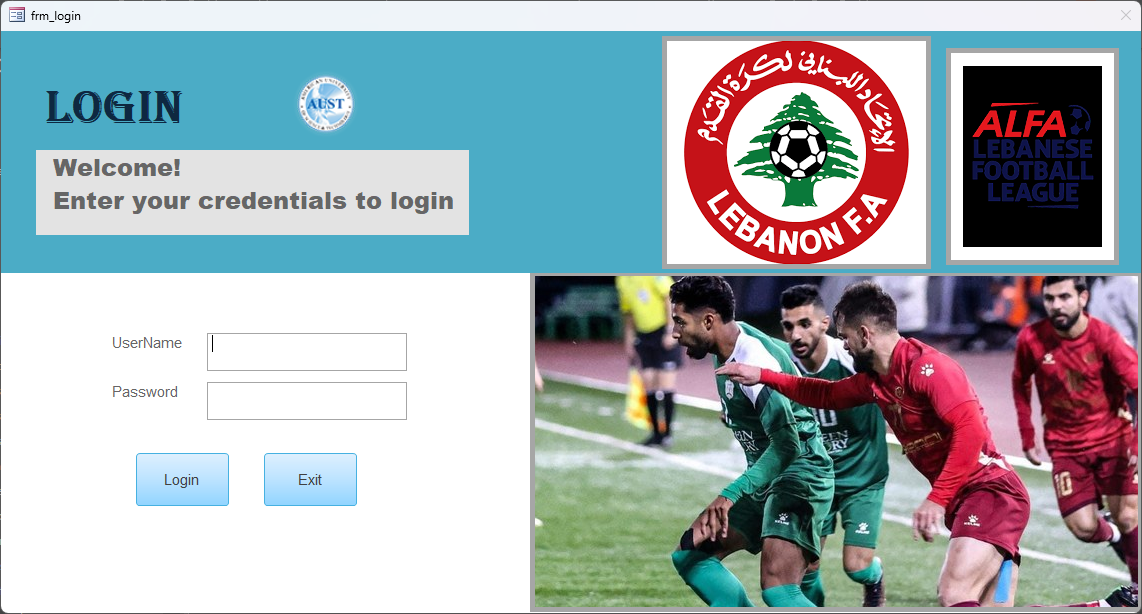


Figure 2 frm\_login Form

When you enter the database, this form opens and cannot be closed. When login is clicked it validates the data entered with the tbl\_login using this VBA code

Option Compare Database

Private Sub cmd\_cancel\_Click()

DoCmd.Quit acQuitSaveAll

End Sub

Private Sub cmd\_login\_Click()

Dim db As DAO.Database

Dim rst As DAO.Recordset

Dim strSQL As String

If Trim(Me.txt\_username.Value & vbNullString) = vbNullString Then

MsgBox prompt:="Username should not be left blank.", buttons:=vbInformation, title:="Username Required"

Me.txt\_username.SetFocus

Exit Sub

End If

If Trim(Me.txt\_password.Value & vbNullString) = vbNullString Then

MsgBox prompt:="Password should not be left blank.", buttons:=vbInformation, title:="Password Required"

Me.txt\_password.SetFocus

Exit Sub

End If

'query to check if login details are correct

strSQL = "SELECT FirstName FROM tbl\_login WHERE Username = """ & Me.txt\_username.Value & """ AND Password = """ & Me.txt\_password.Value & """"

Set db = CurrentDb

Set rst = db.OpenRecordset(strSQL)

If rst.EOF Then

MsgBox prompt:="Incorrect username/password. Try again.", buttons:=vbCritical, title:="Login Error"

Me.txt\_username.SetFocus

Else

loggedInFirstName = rst.Fields(0)

MsgBox prompt:="Hello, " & rst.Fields(0).Value & ".", buttons:=vbOKOnly, title:="Login Successful"

DoCmd.Close acForm, "frm\_login", acSaveYes

DoCmd.OpenForm "dashboard"

End If

Set db = Nothing

Set rst = Nothing

End Sub

Private Sub Form\_Click()

Public loggedInFirstName As String

End Sub

Private Sub Form\_Load()

If Not Me.NewRecord Then

DoCmd.RunCommand acCmdRecordsGoToNew

End If

End Sub

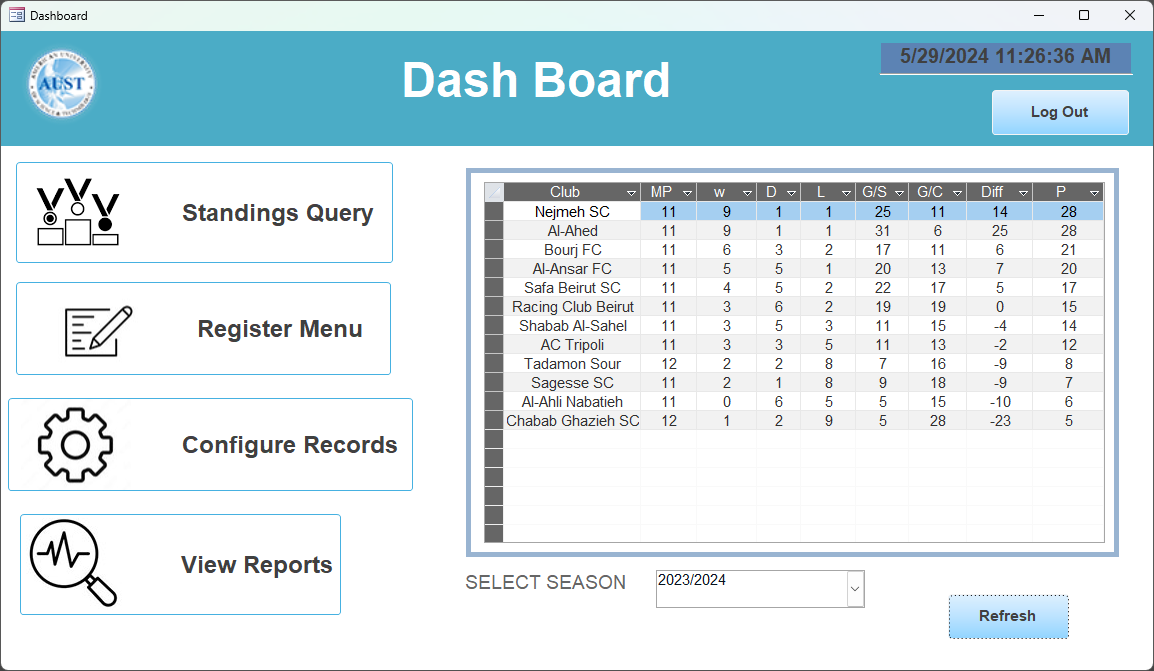
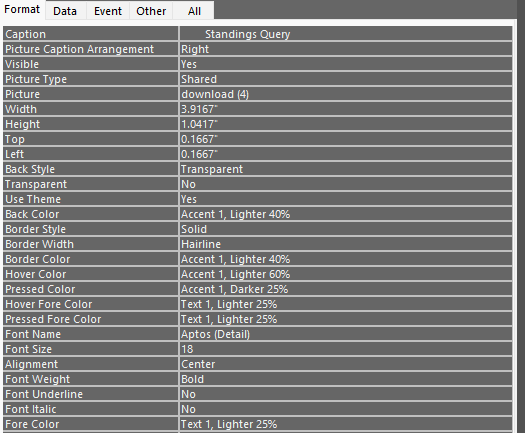


Figure 3 Dashboard Form

Here the left part of the forms contains Buttons with images using this format  


On the Right side of the form, it contains a subform which contains another subform that has the Standings query as a Source Object and takes parameter from the combo box.

The reason of nested Forms is in the Problems faced Section.

When the refresh button is clicked it requeries the subform using this VBA code.

Option Compare Database

Private Sub Command201\_Click()

Dim season As String

season = Me.Combo203.Value

If Not IsNull(season) And season <> "" Then

Dim sql As String

sql = \_

"SELECT clubs.club\_name AS Club\_Name, Sum(Subquery.Matches\_Played) AS Total\_Matches\_Played, " & \_

"Sum(Subquery.Wins) AS Total\_Wins, Sum(Subquery.Draws) AS Total\_Draws, Sum(Subquery.Losses) AS Total\_Losses, " & \_

"Sum(Subquery.Goals\_Scored) AS Total\_Goals\_Scored, Sum(Subquery.Goals\_Conceded) AS Total\_Goals\_Conceded, " & \_

"Sum(Subquery.Goal\_Difference) AS Total\_Goal\_Difference, Sum(Subquery.Points) AS Total\_Points " & \_

"FROM (SELECT Matches.home\_club AS Club\_ID, Count(\*) AS Matches\_Played, " & \_

"Sum(IIf([home\_club\_goals]>[away\_club\_goals],1,0)) AS Wins, Sum(IIf([home\_club\_goals]=[away\_club\_goals],1,0)) AS Draws, " & \_

"Sum(IIf([home\_club\_goals]<[away\_club\_goals],1,0)) AS Losses, Sum(home\_club\_goals) AS Goals\_Scored, Sum(away\_club\_goals) AS Goals\_Conceded, " & \_

"Sum(home\_club\_goals)-Sum(away\_club\_goals) AS Goal\_Difference, Sum(IIf([home\_club\_goals]>[away\_club\_goals],3,IIf([home\_club\_goals]=[away\_club\_goals],1,0))) AS Points " & \_

"FROM Matches WHERE Matches.season = '" & season & "' GROUP BY Matches.home\_club UNION ALL " & \_

"SELECT Matches.away\_club AS Club\_ID, Count(\*) AS Matches\_Played, Sum(IIf([away\_club\_goals]>[home\_club\_goals],1,0)) AS Wins, " & \_

"Sum(IIf([away\_club\_goals]=[home\_club\_goals],1,0)) AS Draws, Sum(IIf([away\_club\_goals]<[home\_club\_goals],1,0)) AS Losses, Sum(away\_club\_goals) AS Goals\_Scored, " & \_

"Sum(home\_club\_goals) AS Goals\_Conceded, Sum(away\_club\_goals)-Sum(home\_club\_goals) AS Goal\_Difference, Sum(IIf([away\_club\_goals]>[home\_club\_goals],3,IIf([away\_club\_goals]=[home\_club\_goals],1,0))) AS Points " & \_

"FROM Matches WHERE Matches.season = '" & season & "' GROUP BY Matches.away\_club) AS Subquery INNER JOIN clubs ON Subquery.Club\_ID = clubs.club\_ID " & \_

"GROUP BY clubs.club\_name ORDER BY Sum(Subquery.Points) DESC;"

' Set the RecordSource of the subform

Me.Child199.Form.RecordSource = sql

Me.Child199.Requery

Else

MsgBox "Please select a season.", vbExclamation

End If

End Sub

Private Sub LogOutBtn\_Click()

' Close the current form

DoCmd.Close acForm, Me.Name, acSaveYes

' Open the login form

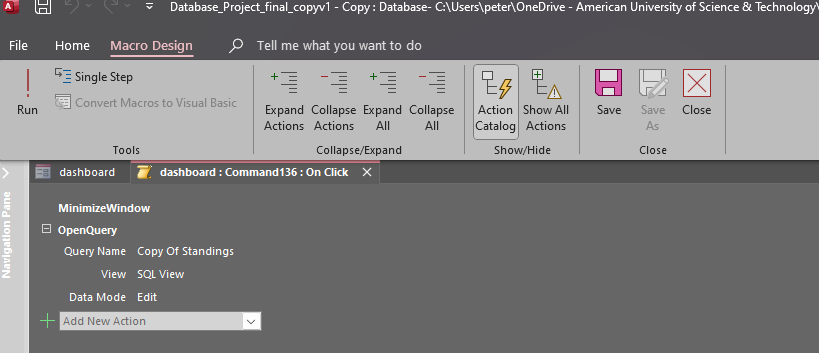
DoCmd.OpenForm "frm\_login"

End Sub

Private Sub NavigationButton114\_Click()

End Sub

The first button of the dashboard minimizes the current form then opens a copy of the Standings query as SQL view.

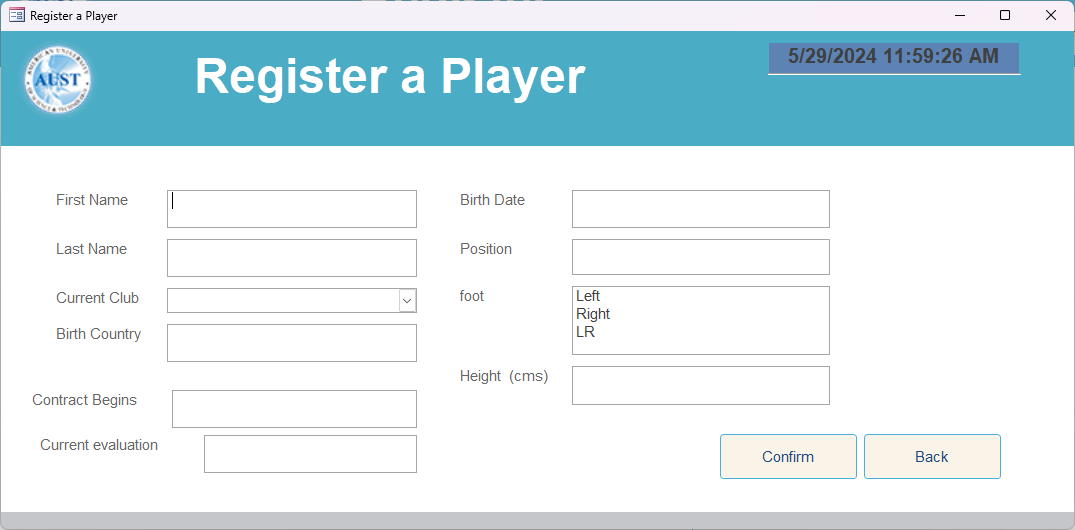


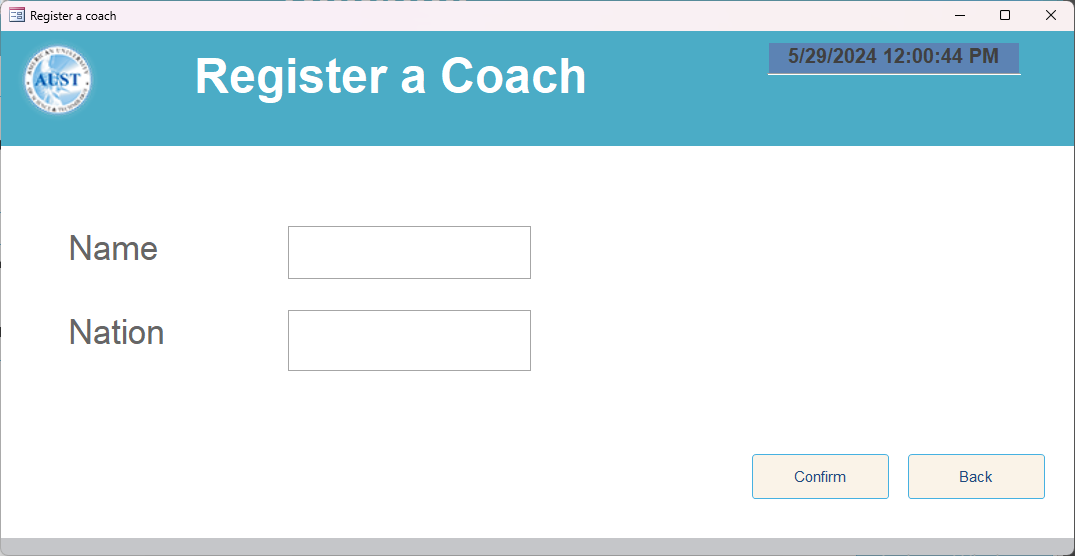
The Rest of the Buttons open these Forms with similar methods.

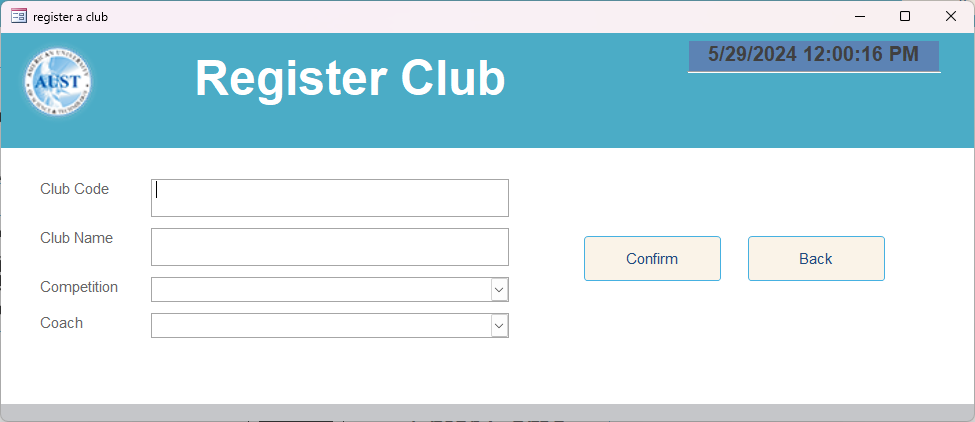
**Register Menu Button**

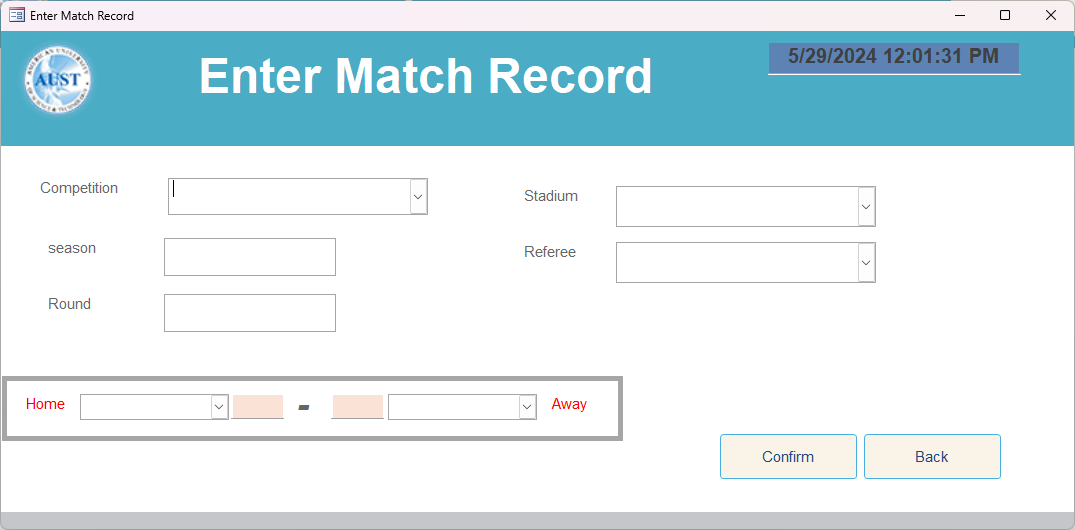


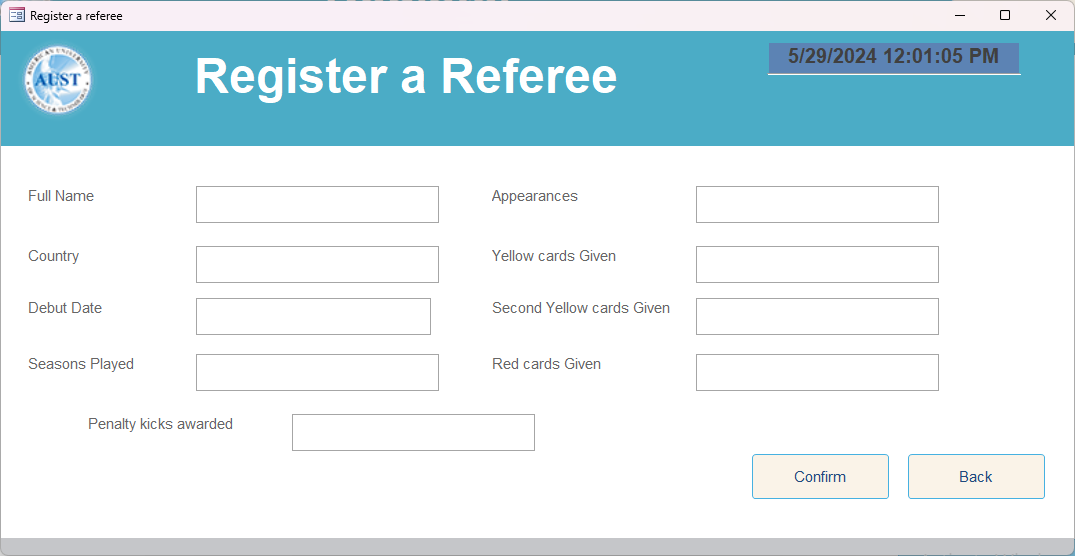
Which also opens the following forms to enter data:











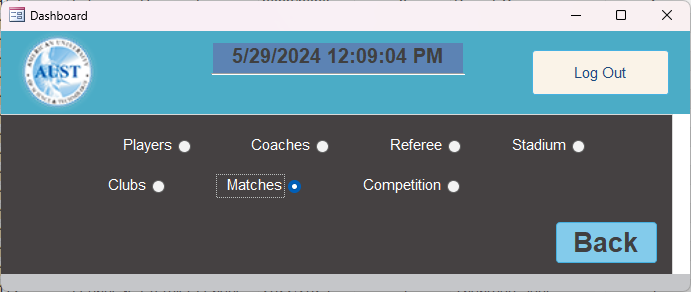
Each of these Forms include a back and confirm Buttons, the back button clears the current record and Close it without saving. The confirm buttons saves the record if the text matches the validation rules or input mask or the combo boxes.

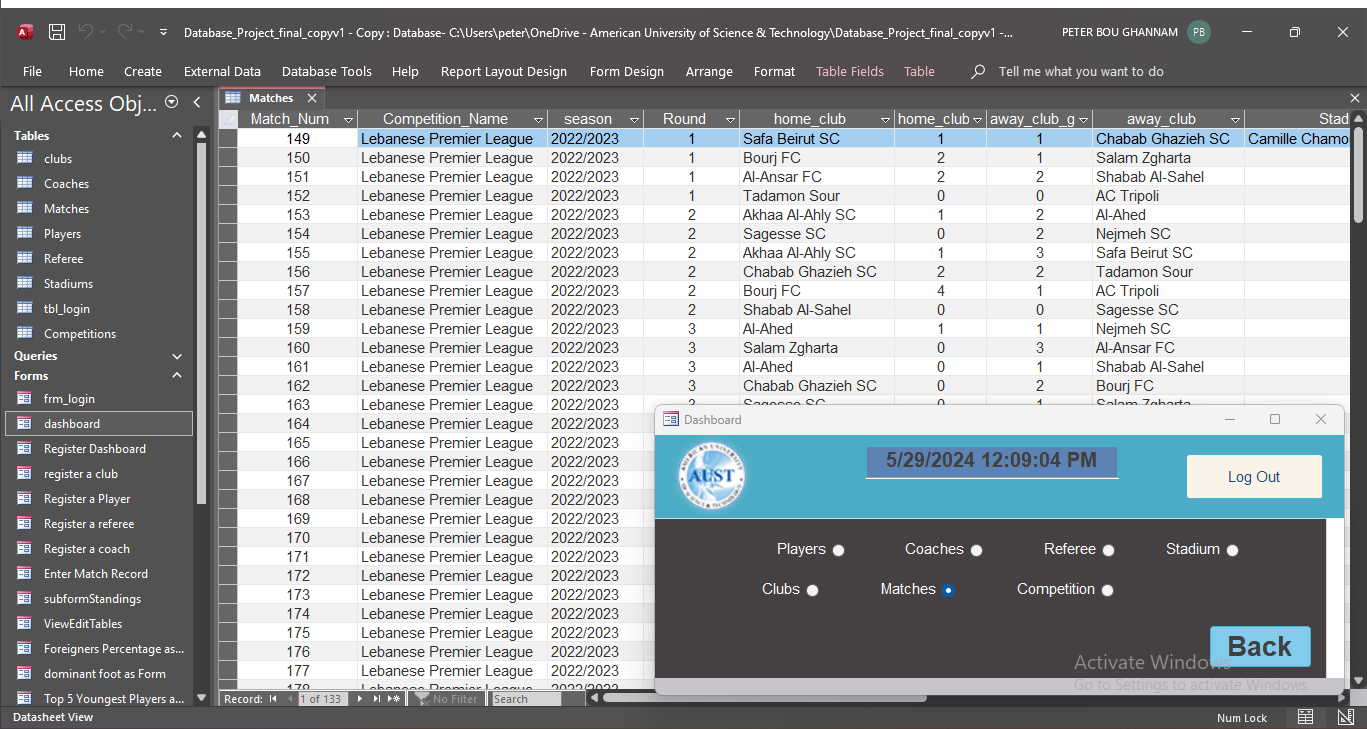
**Configure Records Button**

This form acts as navigation menu between the tables, so the user can take a quick look at the tables and records available, this form importance is to handle errors which are caused by opening multiple tables at once. And since validation exists on both the Forms, and tables, the user can also edit them.

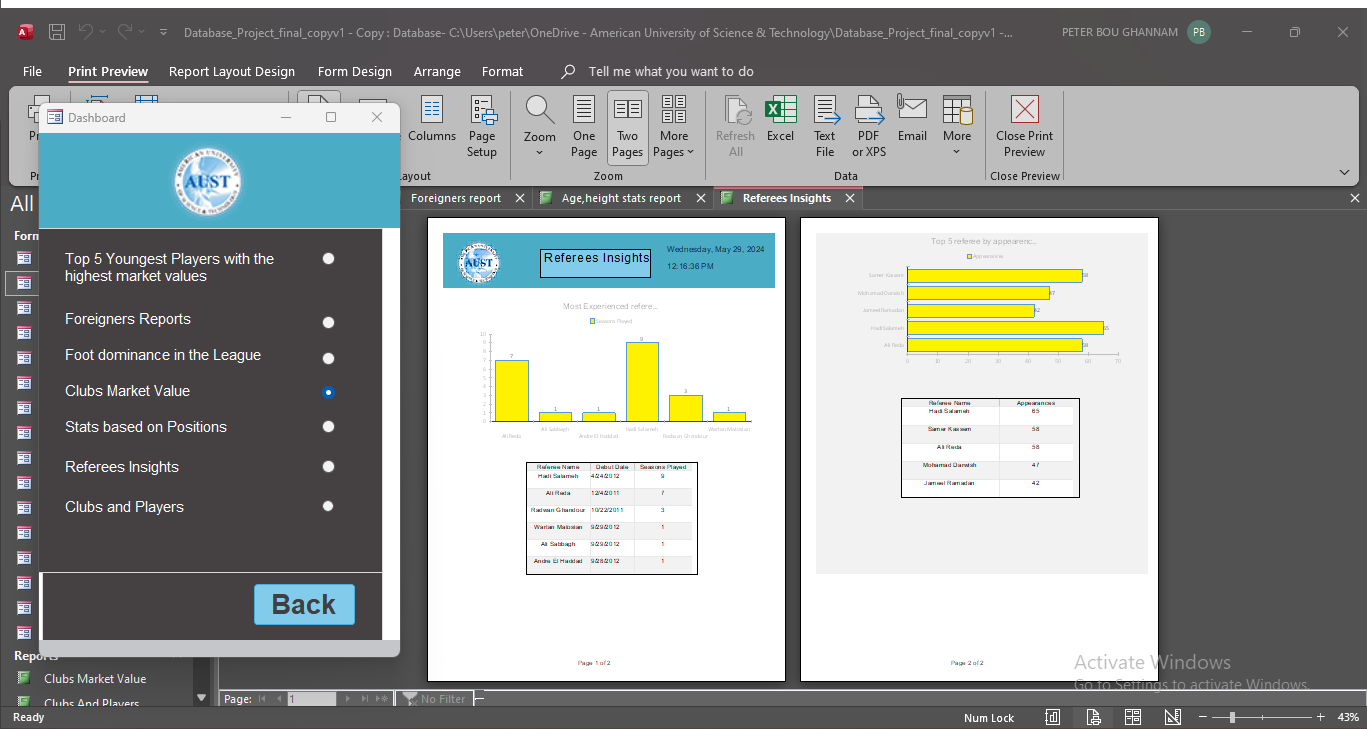
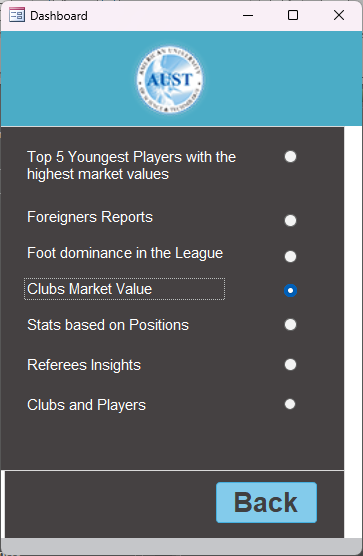
Each radio Button when clicked opens a Table in datasheet View, and since all the radio Buttons are in a group, only one can be opened at a time.

When unclicked automatically after another radio button is clicked, the table is closed and saved.

****

****

**View Reports Button**



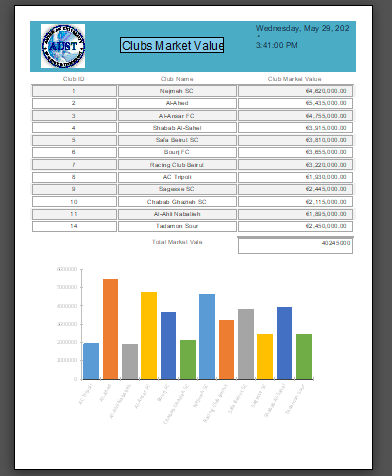
This form acts as navigation menu between the tables, so the user can take a quick look at the tables and records available, this form importance is to handle errors which are caused by opening multiple tables at once. And since validation exists on both the Forms, and tables, the user can also edit them.

Each radio Button when clicked opens a Table in datasheet View, and since all the radio Buttons are in a group, only one can be opened at a time.

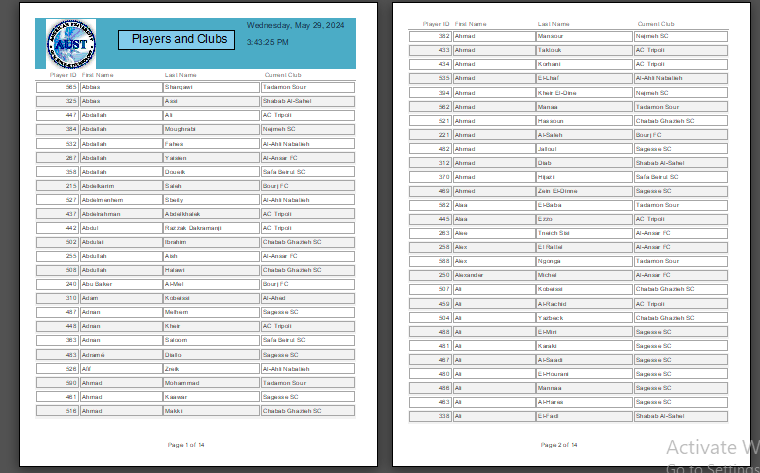
## Reports Used

Each report is connected to a query, and some reports include subforms viewed in datasheet view and some include multiple queries at once, and many types of charts were used

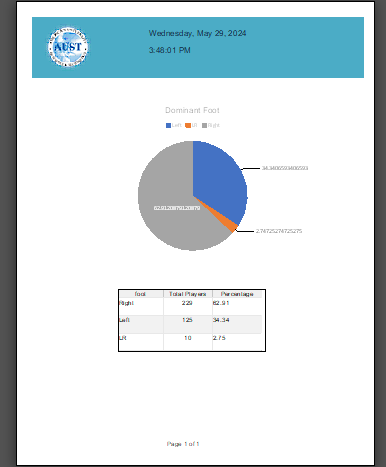
Clubs Market Value



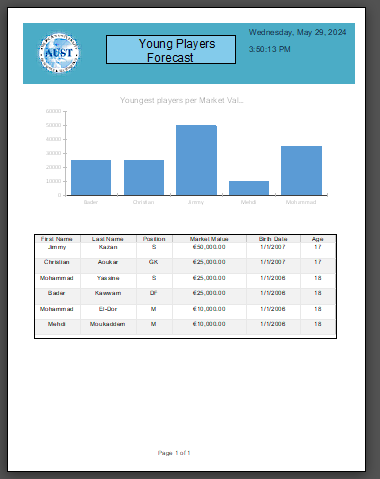
Clubs and Players



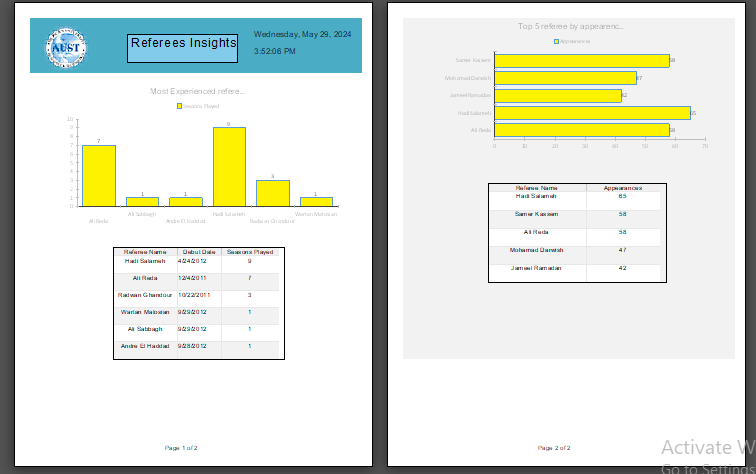
Dominant Foot report

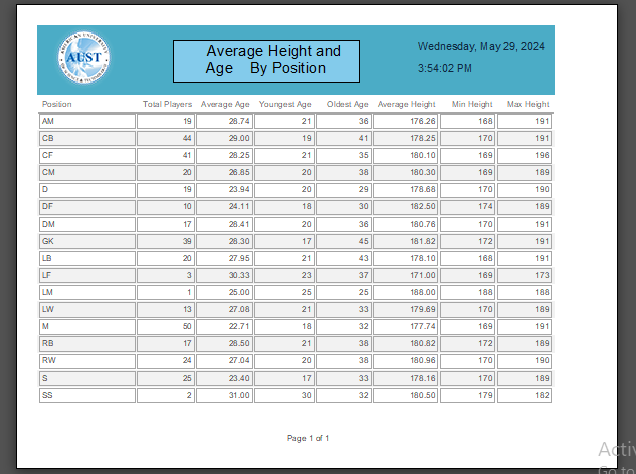


Young players forecast



Referee Insights





Average Height and age by Position Report

## Queries

Here are some important queries used:

1. The Standings query:

PARAMETERS chosen\_season Text ( 255 ) = NULL;

SELECT clubs.club\_name AS Club\_Name, Sum(Subquery.Matches\_Played) AS Total\_Matches\_Played, Sum(Subquery.Wins) AS Total\_Wins, Sum(Subquery.Draws) AS Total\_Draws, Sum(Subquery.Losses) AS Total\_Losses, Sum(Subquery.Goals\_Scored) AS Total\_Goals\_Scored, Sum(Subquery.Goals\_Conceded) AS Total\_Goals\_Conceded, Sum(Subquery.Goal\_Difference) AS Total\_Goal\_Difference, Sum(Subquery.Points) AS Total\_Points

FROM (SELECT

Matches.home\_club AS Club\_ID,

Count(\*) AS Matches\_Played,

Sum(IIf([home\_club\_goals]>[away\_club\_goals],1,0)) AS Wins,

Sum(IIf([home\_club\_goals]=[away\_club\_goals],1,0)) AS Draws,

Sum(IIf([home\_club\_goals]<[away\_club\_goals],1,0)) AS Losses,

Sum(home\_club\_goals) AS Goals\_Scored,

Sum(away\_club\_goals) AS Goals\_Conceded,

Sum(home\_club\_goals)-Sum(away\_club\_goals) AS Goal\_Difference,

Sum(IIf([home\_club\_goals]>[away\_club\_goals],3,IIf([home\_club\_goals]=[away\_club\_goals],1,0))) AS Points

FROM

Matches

WHERE

Matches.season = chosen\_season

GROUP BY

Matches.home\_club

UNION ALL

SELECT

Matches.away\_club AS Club\_ID,

Count(\*) AS Matches\_Played,

Sum(IIf([away\_club\_goals]>[home\_club\_goals],1,0)) AS Wins,

Sum(IIf([away\_club\_goals]=[home\_club\_goals],1,0)) AS Draws,

Sum(IIf([away\_club\_goals]<[home\_club\_goals],1,0)) AS Losses,

Sum(away\_club\_goals) AS Goals\_Scored,

Sum(home\_club\_goals) AS Goals\_Conceded,

Sum(away\_club\_goals)-Sum(home\_club\_goals) AS Goal\_Difference,

Sum(IIf([away\_club\_goals]>[home\_club\_goals],3,IIf([away\_club\_goals]=[home\_club\_goals],1,0))) AS Points

FROM

Matches

WHERE

Matches.season = chosen\_season

GROUP BY

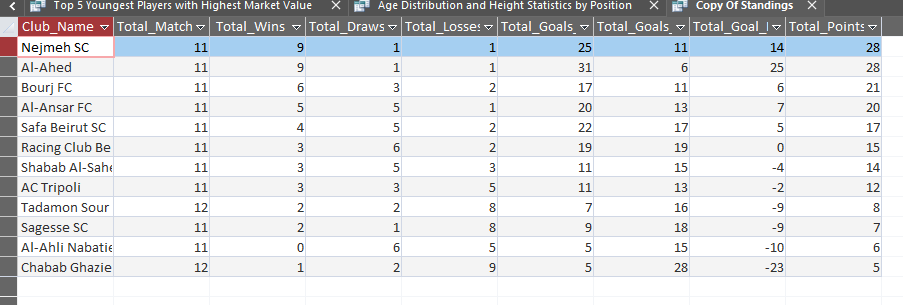
Matches.away\_club

) AS Subquery INNER JOIN clubs ON Subquery.Club\_ID = clubs.club\_ID

GROUP BY clubs.club\_name

ORDER BY Sum(Subquery.Points) DESC;

Results:



**Overview**

This query generates the standings of football clubs for a specific season based on match results. It includes the total number of matches played, wins, draws, losses, goals scored, goals conceded, goal difference, and points for each club.

**Steps Involved**

1. **Parameters**
   * **chosen\_season**: The season for which we want to see the standings.
2. **Subquery 1: Home Matches**
   * Selects **home\_club** as **Club\_ID**.
   * Counts total matches played at home.
   * Sums wins, draws, and losses based on home match results.
   * Sums goals scored and conceded by the home club.
   * Calculates goal difference and points for home matches.
3. **Subquery 2: Away Matches**
   * Similar to Subquery 1 but for **away\_club**.
   * Counts total matches played away.
   * Sums wins, draws, and losses based on away match results.
   * Sums goals scored and conceded by the away club.
   * Calculates goal difference and points for away matches.
4. **Union of Subqueries**
   * Combines results from both home and away subqueries.
   * Groups by **Club\_ID**.
5. **Main Query**
   * Joins the unioned subquery results with the **clubs** table on **Club\_ID**.
   * Groups by **club\_name**.
   * Sums total matches played, wins, draws, losses, goals scored, goals conceded, goal difference, and points.
   * Orders the results by total points in descending order.

**Output**

The final output lists each club’s name along with their:

* Total matches played
* Total wins, draws, and losses
* Total goals scored and conceded
* Total goal difference
* Total points

This allows us to see the standings of the clubs for the selected season, ranked by their performance.

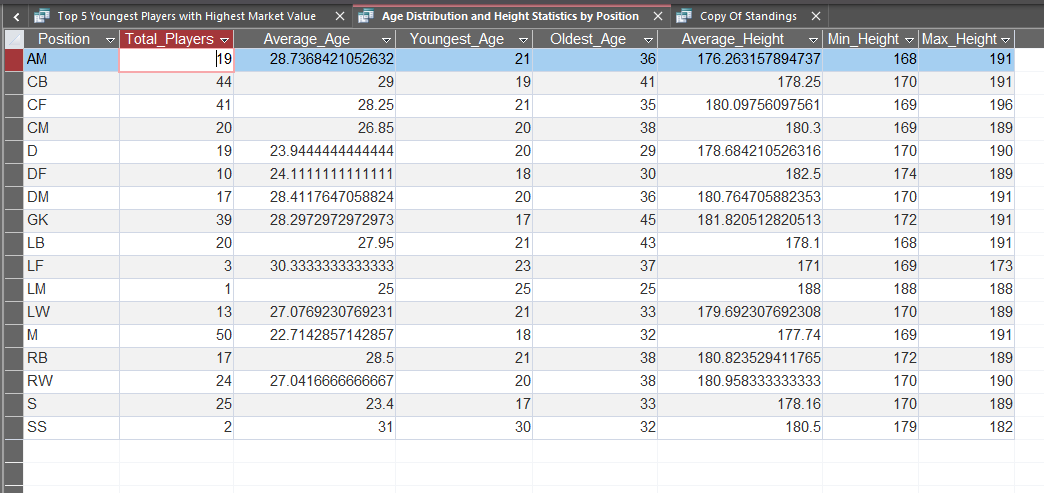
1. Positions Study

SELECT p.Position, Count(p.Player\_ID) AS Total\_Players, Avg(DateDiff("yyyy",p.Date\_of\_Birth,Date())) AS Average\_Age, Min(DateDiff("yyyy",p.Date\_of\_Birth,Date())) AS Youngest\_Age, Max(DateDiff("yyyy",p.Date\_of\_Birth,Date())) AS Oldest\_Age, Avg(p.height\_in\_cms) AS Average\_Height, Min(p.height\_in\_cms) AS Min\_Height, Max(p.height\_in\_cms) AS Max\_Height

FROM Players AS p

GROUP BY p.Position;

Results:



**Overview**

This query calculates and summarizes various statistics for football players, grouped by their playing positions. It includes the total number of players, average age, youngest and oldest player ages, and height statistics (average, minimum, and maximum) for each position.

**Steps Involved**

1. **Parameters and Functions**
   * **DateDiff("yyyy", p.Date\_of\_Birth, Date())**: This function calculates the difference in years between a player's date of birth and the current date, effectively computing the player's age.
2. **Select Clause**
   * **p.Position**: Selects the position of the player.
   * **Count(p.Player\_ID) AS Total\_Players**: Counts the total number of players for each position.
   * **Avg(DateDiff("yyyy", p.Date\_of\_Birth, Date())) AS Average\_Age**: Calculates the average age of players for each position.
   * **Min(DateDiff("yyyy", p.Date\_of\_Birth, Date())) AS Youngest\_Age**: Finds the age of the youngest player for each position.
   * **Max(DateDiff("yyyy", p.Date\_of\_Birth, Date())) AS Oldest\_Age**: Finds the age of the oldest player for each position.
   * **Avg(p.height\_in\_cms) AS Average\_Height**: Calculates the average height of players for each position.
   * **Min(p.height\_in\_cms) AS Min\_Height**: Finds the minimum height of players for each position.
   * **Max(p.height\_in\_cms) AS Max\_Height**: Finds the maximum height of players for each position.
3. **From Clause**
   * **Players AS p**: Specifies the table (Players) and its alias (p) from which to retrieve data.
4. **Group By Clause**
   * **p.Position**: Groups the results by the position of the players to calculate the aggregated statistics for each position.

**Output**

The final output lists each player's position along with their:

* Total number of players
* Average age
* Youngest age
* Oldest age
* Average height
* Minimum height
* Maximum height

This query provides a detailed summary of player demographics and physical attributes based on their playing positions.

## Python Scrapper

We wrote the scraper to help with getting data during the development of the database before we set up the validation rules.

The scrapper was written using python, on [**Jupyter Notebook**](https://jupyter.org/)which is an interactive web application for creating and sharing computational documents, combining code, narrative text, equations, and visualizations.

A **notebook** is a shareable document that integrates computer code, plain language descriptions, data, and rich visualizations like 3D models, charts, and graphs.

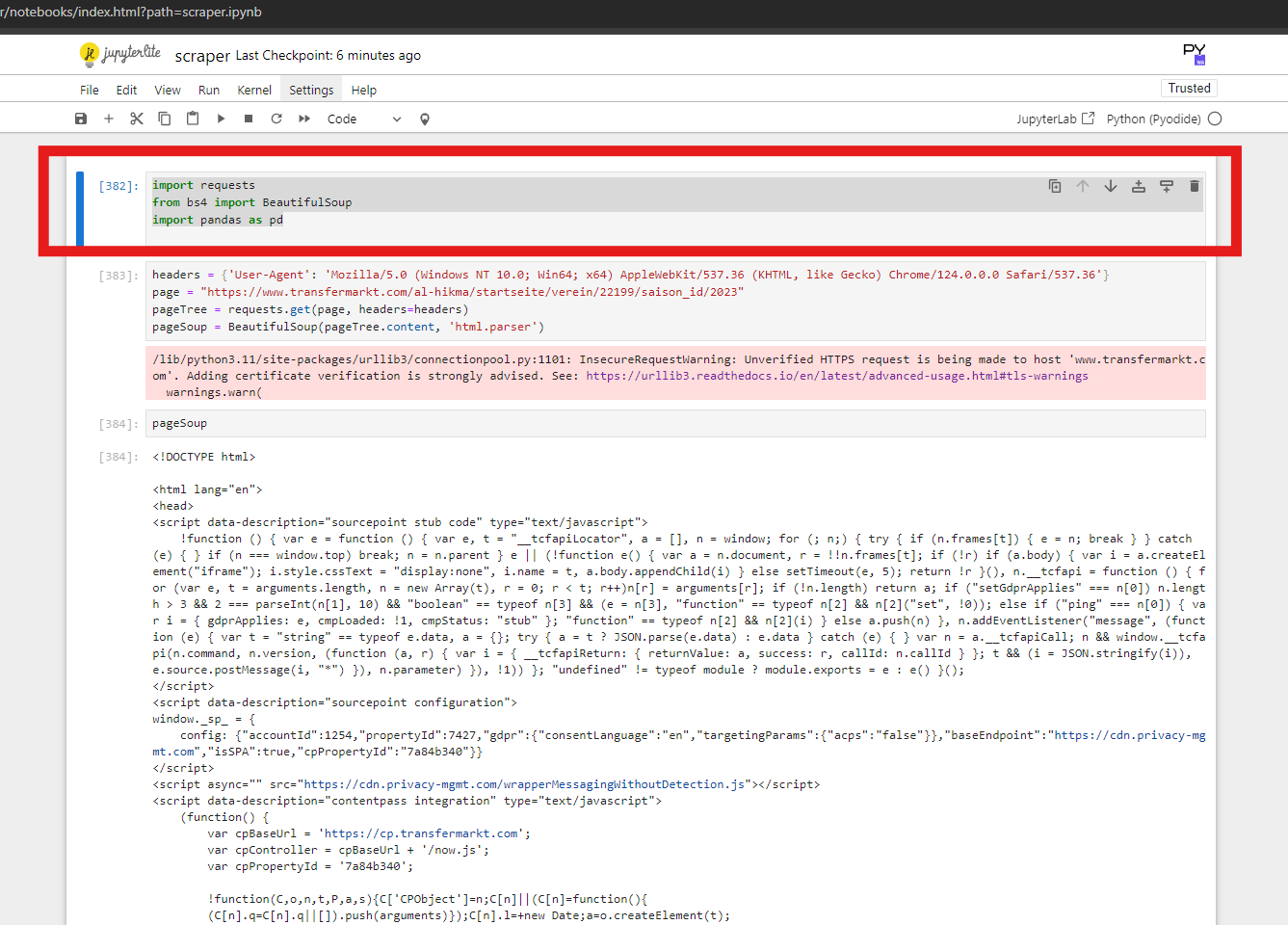
It's widely used in data science for interactive development and presentation of projects.

[How do Jupyter Notebooks work?](https://www.databricks.com/glossary/jupyter-notebook#:~:text=How%20do%20Jupyter%20Notebooks%20work,code%20and%20returns%20the%20results.) A Jupyter notebook has two components: a front-end web page and a back-end kernel. The front-end web page allows data scientists to enter programming code or text in rectangular "cells." The browser then passes the code to the back-end kernel which runs the code and returns the results.

The scrapper was only built to scrap data from some Html pages in <https://www.transfermarkt.com/>

using find and regular expressions, and the scrapping must be done manually.

***CODE***



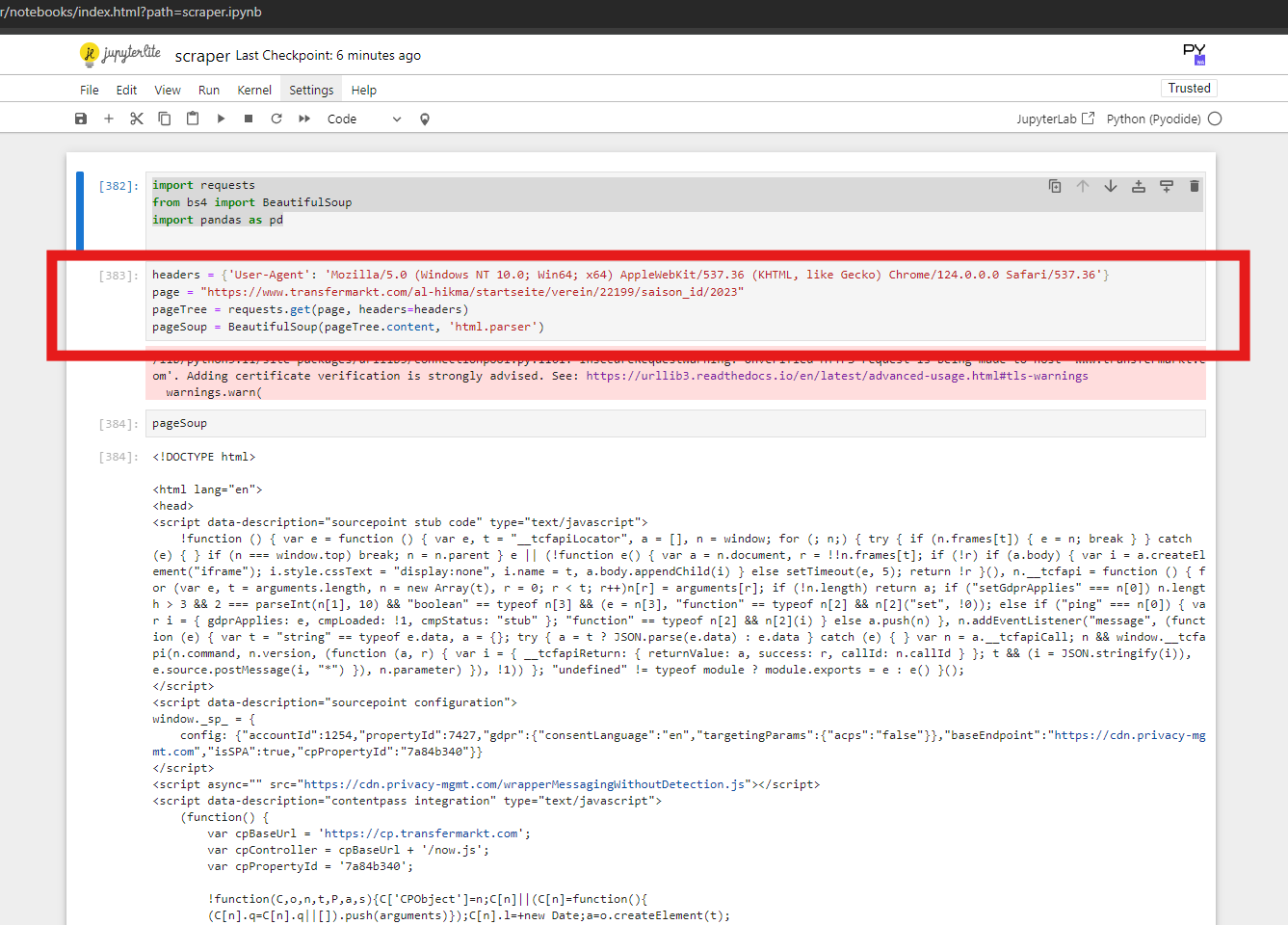
1. First cell:

import requests

from bs4 import BeautifulSoup

import pandas as pd

1. import requests:
   * This imports the **requests** library, which is used to send HTTP requests in Python. It allows you to fetch the content of web pages.
2. from bs4 import BeautifulSoup:
   * This imports the **BeautifulSoup** class from the **bs4** (Beautiful Soup) library, which is used for parsing HTML and XML documents. It helps in extracting data from web pages in a hierarchical and readable manner.
3. import pandas as pd:
   1. This imports the **pandas** library with the alias **pd**. Pandas is a powerful data manipulation and analysis library in Python. It provides data structures like DataFrames that are useful for handling and analyzing structured data.
4. Second cell



headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36'}

page = "https://www.transfermarkt.com/al-hikma/startseite/verein/22199/saison\_id/2023"

pageTree = requests.get(page, headers=headers)

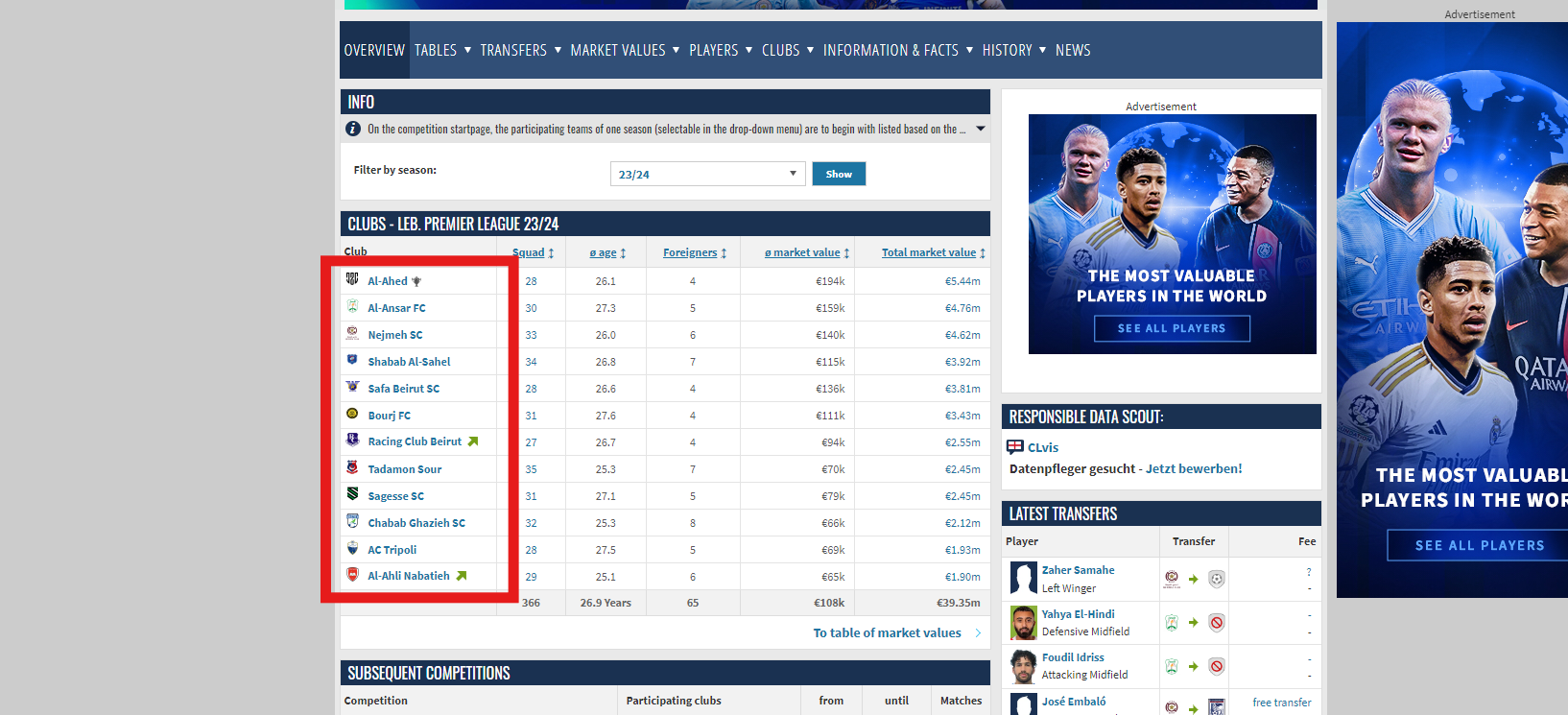
pageSoup = BeautifulSoup(pageTree.content, 'html.parser')

1. Set Headers:
   * Sets a custom User-Agent header to mimic a real web browser.

(here I used my real user agent, Peter Bou Ghannam)

1. Define the Page URL:
   * Specifies the URL of the web page to scrape.

! VERY IMPORTANT !

For the scrapper to work with no errors a link of only these clubs must be entered, no testing was done for other pages under the same site  


The page links to be be put in the second cell are:  
<https://www.transfermarkt.com/al-ahed/startseite/verein/15720/saison_id/2023>

<https://www.transfermarkt.com/al-ansar/startseite/verein/2468/saison_id/2023>

<https://www.transfermarkt.com/al-nejmeh/startseite/verein/7142/saison_id/2023>

<https://www.transfermarkt.com/shabab-sahel/startseite/verein/22845/saison_id/2023>

<https://www.transfermarkt.com/safa-beirut-sc/startseite/verein/15724/saison_id/2023>

<https://www.transfermarkt.com/bourj-fc/startseite/verein/76184/saison_id/2023>

<https://www.transfermarkt.com/racing-club-beirut/startseite/verein/18761/saison_id/2023>

<https://www.transfermarkt.com/tadamon-sour/startseite/verein/388/saison_id/2023>

<https://www.transfermarkt.com/al-hikma/startseite/verein/22199/saison_id/2023>

<https://www.transfermarkt.com/shabab-gazieh/startseite/verein/22844/saison_id/2023>

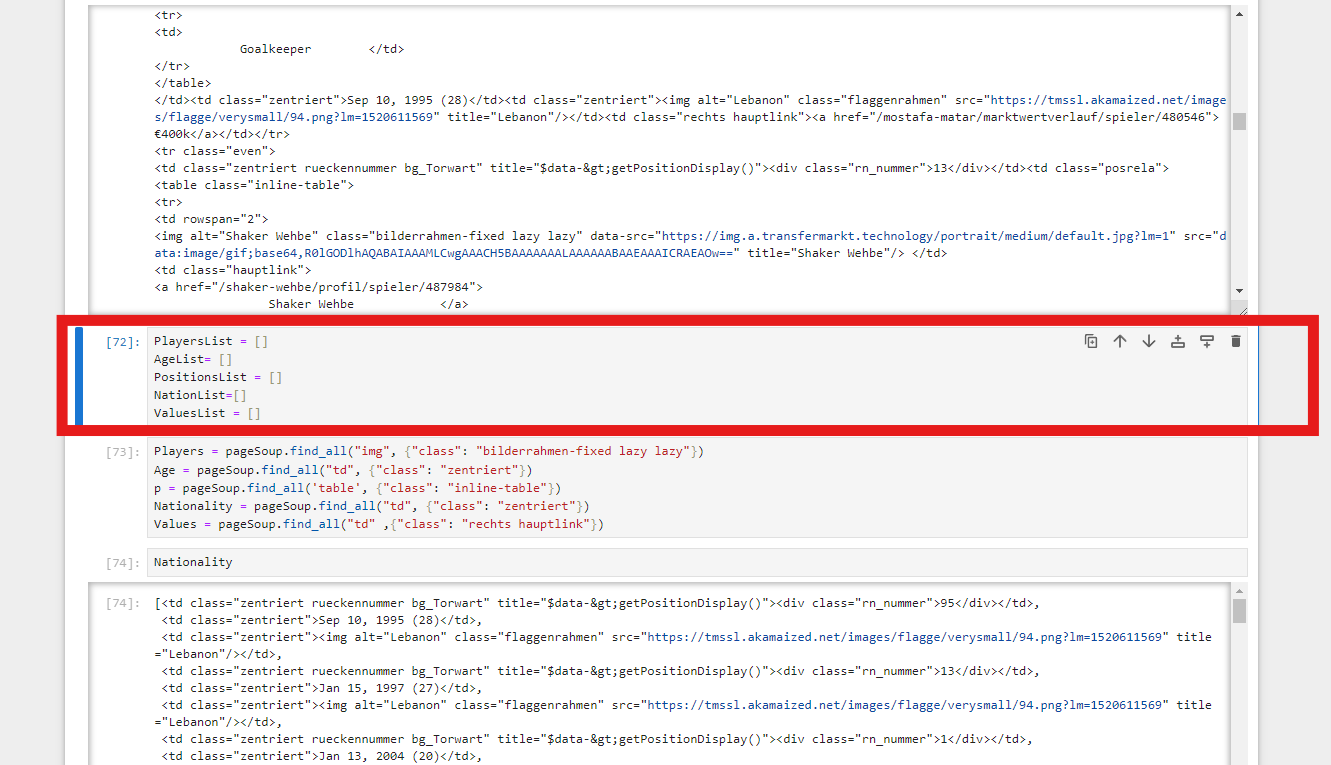
<https://www.transfermarkt.com/ac-tripoli/startseite/verein/5565/saison_id/2023>

<https://www.transfermarkt.com/al-ahli-club-nabatieh/startseite/verein/74492/saison_id/2023>

1. pageTree = requests.get(page, headers=headers)
   * Sends an HTTP GET request to the URL with the custom headers.
2. pageSoup = BeautifulSoup(pageTree.content, 'html.parser')
   * Parses the HTML content of the fetched page using BeautifulSoup.
3. Third cell

Just displays the fetched html

1. Fourth cell



PlayersList = []

AgeList= []

PositionsList = []

NationList=[]

ValuesList = []

**Here we just create arrays to use them to fill a data frame later on, and to fill them with the data scraped from the html.**

1. Fifth cell



Players = pageSoup.find\_all("img", {"class": "bilderrahmen-fixed lazy lazy"})

Age = pageSoup.find\_all("td", {"class": "zentriert"})

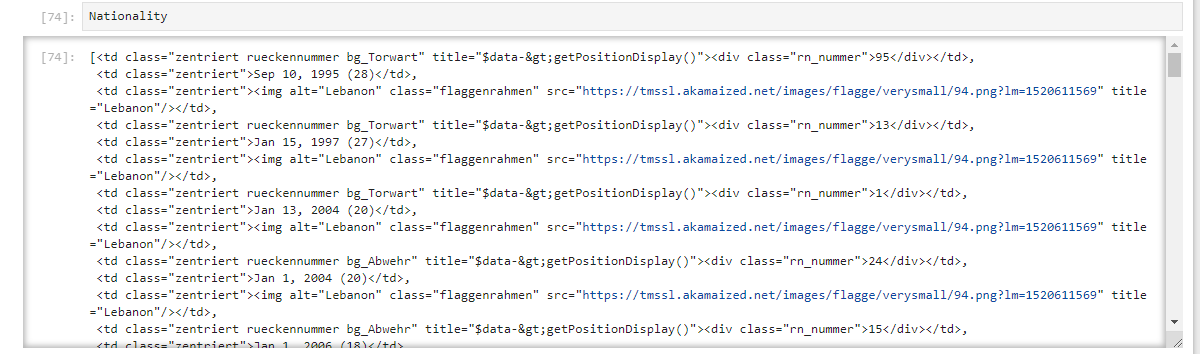
p = pageSoup.find\_all('table', {"class": "inline-table"})

Nationality = pageSoup.find\_all("td", {"class": "zentriert"})

Values = pageSoup.find\_all("td" ,{"class": "rechts hauptlink"})

This code uses BeautifulSoup to scrape specific data from an HTML page, targeting player images, ages, tables with inline-table class, nationalities, and values. It finds and stores these elements based on their respective HTML tags and classes for further processing.

Sample when running the nationality



1. Sixth cell



for player in Players:

PlayersList.append(player['alt'])

for td in Age:

# Check if the text contains a birthdate and age in parentheses

if '(' in td.text and ')' in td.text:

# Extract the age using regular expression

age\_text = td.text.split('(')[1].split(')')[0]

AgeList.append(age\_text)

for player in p:

c = player.find\_all('tr')[1]

PositionsList.append(c.find('td').text.strip())

for i in range(2, min(len(Nationality), len(PlayersList) \* 3), 3): # Ensure it stops at the length of PlayersList

nationality\_img = Nationality[i].find('img')

if nationality\_img:

NationList.append(nationality\_img['title'])

else:

NationList.append(None)

# Function to convert value strings to integers

def convert\_value(value):

if value == '-':

return None

value = value.replace('€', '').replace('k', '000').replace('m', '000000')

return int(value)

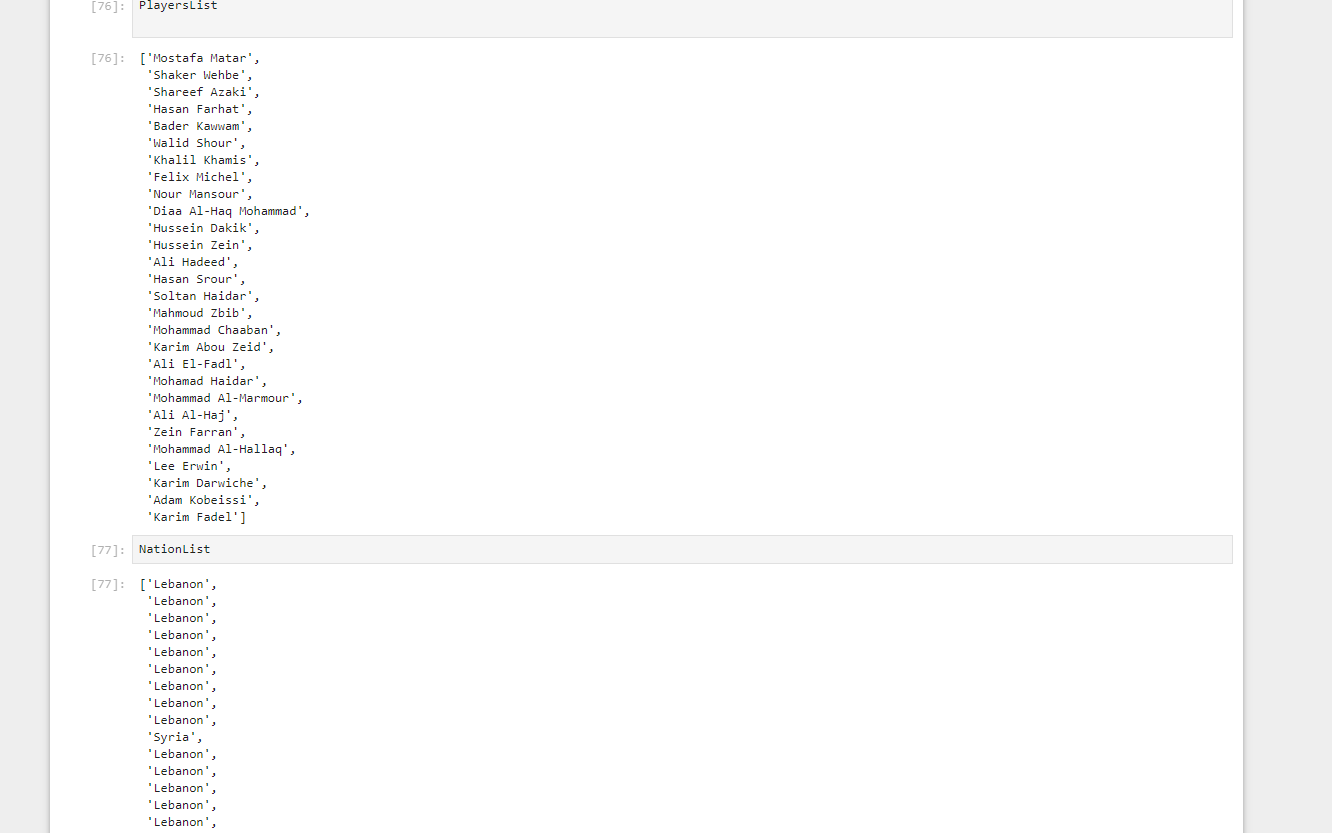
# Extract and convert values

for value in Values:

ValuesList.append(convert\_value(value.text.strip()))

* + Extract player names: PlayersList.append(player['alt'])
  + Extract ages: Use regex to parse and add to AgeList
  + Extract positions: Parse HTML and add to PositionsList
  + Extract nationalities: Parse images in nationality cells, ensure length matches PlayersList
  + Convert value strings to integers: Define convert\_value function
  + Extract and convert values: Use convert\_value to add to ValuesList

1. The 7, 8 , 9 cells



1. The 11th cell

max\_length = len(PlayersList)

PlayersList += [None] \* (max\_length - len(PlayersList))

AgeList += [None] \* (max\_length - len(AgeList))

PositionsList += [None] \* (max\_length - len(PositionsList))

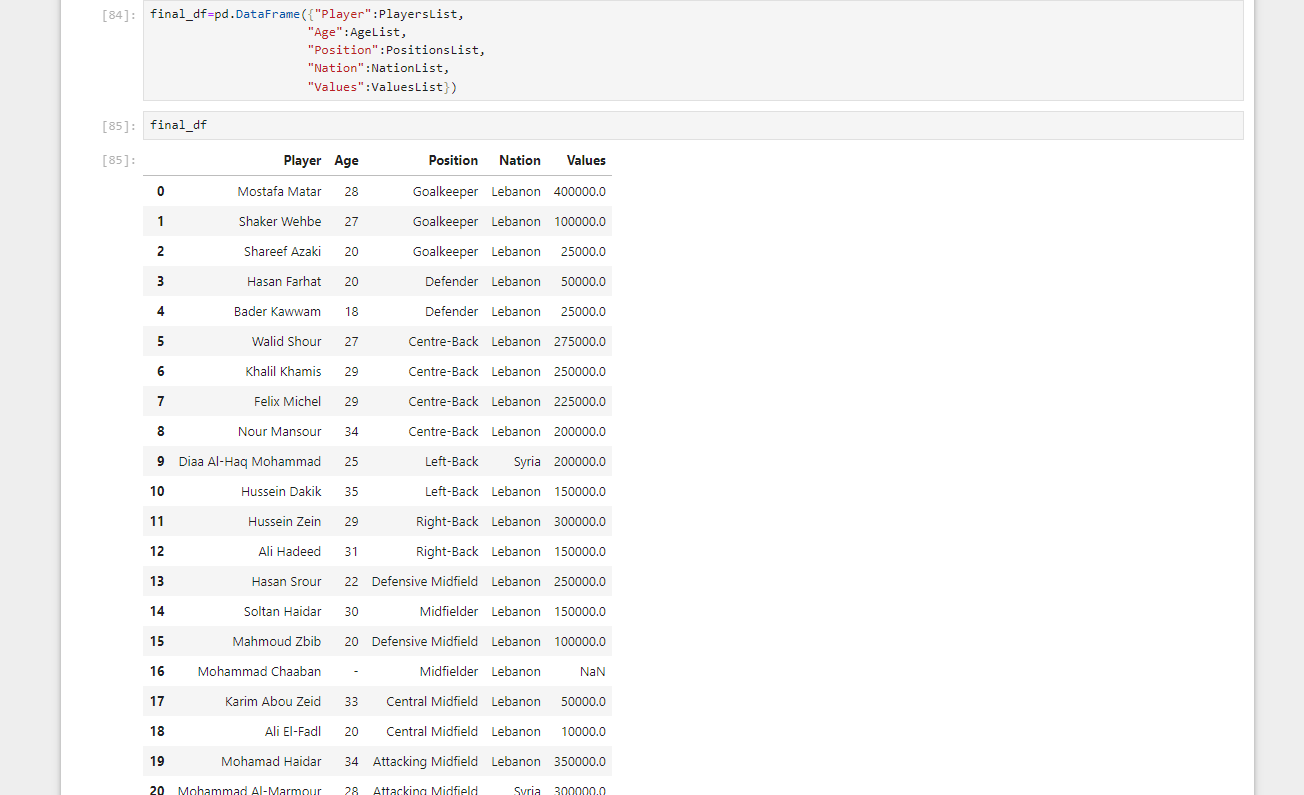
NationList += [None] \* (max\_length - len(NationList))

ValuesList += [None] \* (max\_length - len(ValuesList))



This makes sure all arrays are equal so they can be put in a data frame

1. The 13th and 14th cell



final\_df=pd.DataFrame({"Player":PlayersList,

"Age":AgeList,

"Position":PositionsList,

"Nation":NationList,

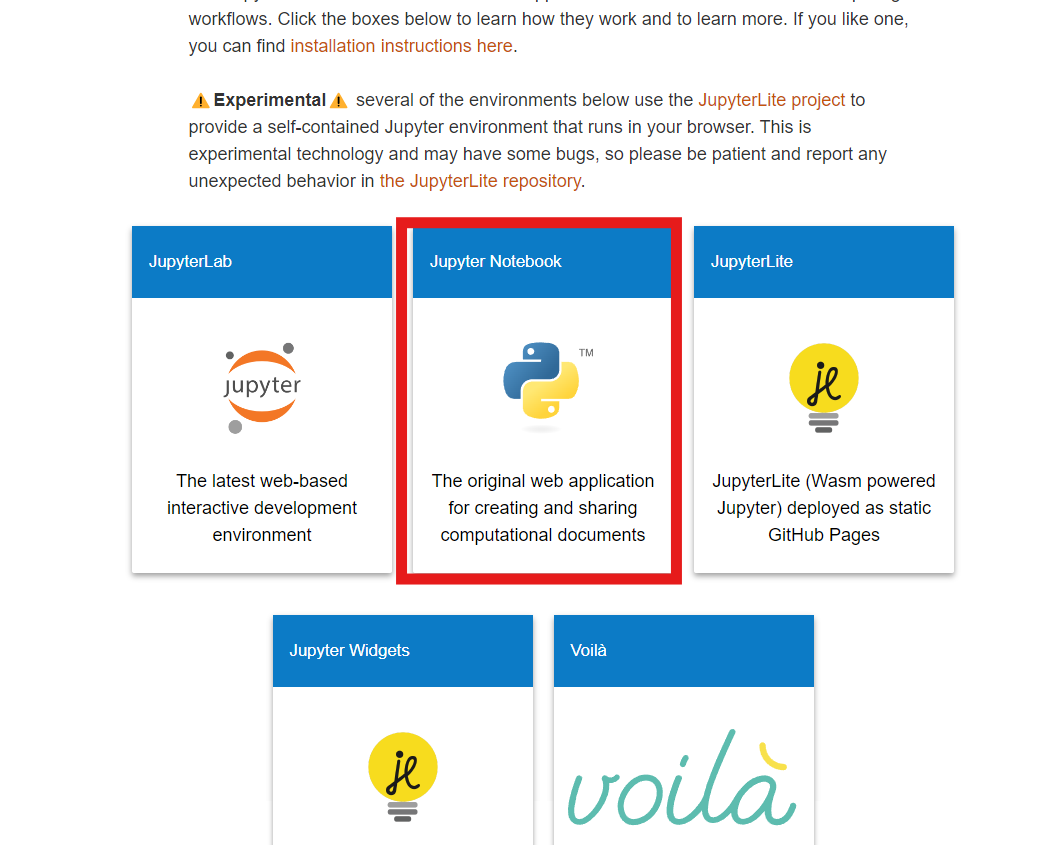
"Values":ValuesList})

final\_df

The 13th cell makes the arrays into a data frame and the 14th runs it

**How to run:**

* + 1. **Download the scraper.ipynb file from the github at the end**
    2. **Open this website** [**https://jupyter.org/try**](https://jupyter.org/try)
    3. **Click on the Jupyter notebook**

****

**Upload the file and click run all cells**

1. Validation rules:

We used similar validation on both the forms and tables.

|  |  |
| --- | --- |
| **Clubs** |  |
| club\_ID (AutoNumber PK) | AutoNumber |
| club\_code (Short Text) | Is Not Null And Not Like "\*[!a-zA-Z0-9]\*" And Len([club\_code])<=30 |
| club\_name (Short Text) | Is Not Null And Len([club\_name])<=50 |
| Competition\_ID (Number FK) | Lookup wizard |
| Coach\_ID (Number FK ) | Lookup wizard |
|  |  |
| **Players** |  |
| Player\_ID (AutoNumber PK) | Autonumber |
| First\_Name (Short Text) | Is Not Null And Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| Last\_Name (Short Text) | Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| Current\_club\_ID (Number FK) | Lookup wizard |
| Country\_Birth (Short Text) | Is Not Null And Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| Date\_of\_Birth (Date/Time Extended) | Is Not Null And [Date\_of\_Birth]<Date() And [Date\_of\_Birth]>DateAdd("yyyy",-100,Date()) |
| Position (Short Text) | Is Not Null And Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| foot (Short Text) | Combo box value list, not editable |
| height\_in\_cms (Number) | Is Not Null And [height\_in\_cms]>=100 And [height\_in\_cms]<=250 |
| contract\_begins\_date (Date/Time) | Input mask 99/99/0000 |
| current\_market\_value (Currency) | [current\_market\_value]>=0 |
|  |  |
|  |  |
| **Coaches** |  |
| Coach\_ID (Number PK) | AutoNumber |
| Coach\_Name (Short Text) | Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| Nation (Short Text) | Is Not Null And Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
|  |  |
| **Stadiums** |  |
| Stadium\_ID (AutoNumber PK) | AutoNumber |
| Stadium\_Name (Short Text) | Is Not Null And Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| Stadium\_City (Short Text) | Is Not Null And Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| Stadium\_Capacity (Number) | >0 |
|  |  |
| **Matches** |  |
| Match\_Num (AutoNumber PK) | Autonumber |
| Competition\_Name (Number FK) | Lookup Wizard |
| season (Short Text) | Input mask: 0000/0000;0; |
| Round (Number) | Is Not Null And >0 |
| home\_club (Number) | Lookup wizard |
| home\_club\_goals (Number) | Not Like "\*[@#\$%^&\*()\_+{}|<>?/~]\*" |
| away\_club (Number) | Lookup wizard |
| away\_club\_goals (Number) | Not Like "\*[@#\$%^&\*()\_+{}|<>?/~]\*" |
| Stadium (Number FK) | Lookup wizard |
| Referee\_ID (Number FK) | Lookup wizard |
|  |  |
| **Competitions** |  |
| Competition\_ID (AutoNumber PK) | Autonumber |
| Competition\_Name (Short Text) | Text |
| Major\_national\_competition (Yes/No) | Yes/No check box |
|  |  |
| **Referee** |  |
| Referee\_ID (AutoNumber PK) | AutoNumber |
| Referee\_Name (Short Text) | Is Not Null And Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| Country (Short Text) | Is Not Null And Not Like "\*[0-9!@#\$%^&\*()\_+{}|<>?/~]\*" |
| Debut\_Date (Date/Time) | Input mask: 99/99/0000;0;\_ |
| Seasons\_Played (Number) | Is Not Null And Not Like "\*[@#\$%^&\*()\_+{}|<>?/~]\*" And >=0 |
| Appearances (Number) | Is Not Null And Not Like "\*[@#\$%^&\*()\_+{}|<>?/~]\*" And >=0 |
| Yellow\_Cards\_Given (Number) | Is Not Null And Not Like "\*[@#\$%^&\*()\_+{}|<>?/~]\*" And >=0 |
| Second\_Yellow\_Cards\_Given (Number) | Is Not Null And Not Like "\*[@#\$%^&\*()\_+{}|<>?/~]\*" And >=0 |
| Red\_Cards\_Given (Number) | Is Not Null And Not Like "\*[@#\$%^&\*()\_+{}|<>?/~]\*" And >=0 |
| Penalty\_Kicks\_Awarded (Number) | Is Not Null And Not Like "\*[@#\$%^&\*()\_+{}|<>?/~]\*" And >=0 |

# POSTER:

Website used canva.com

Our project location:  
<https://www.canva.com/design/DAFkl6r_VUU/SLtSW33Mrf7L-BVOUSioSw/edit?utm_content=DAFkl6r_VUU&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>

This is the link for our poster, u can click it and click share button to download it



# Problems faced and solutions

Problems:

1. Having to enter a lot of correct data manually
2. Making advanced button logics in VBA, since we didn’t learn it
3. Reports not displaying correctly in print view

Solutions:

1. Creating a scrapper to assist and using Microsoft Excel
2. Using Macro Builder
3. Fixing the Reports in Layout View

# Project Location

<https://github.com/Sleek-0/Database-Scrapper-Project-Footaball/>