:[229] In

```
import requests
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas import json_normalize
from flatten_json import flatten
import missingno as msno
pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 100)
```

:[ ] In

```
player_data ="https://cdnapi.bamboo-video.com/api/football/stats?format=json&iid=5738
player_data_chapionshipround ="https://cdnapi.bamboo-video.com/api/football/stats?for
player_data_relegationround = "https://cdnapi.bamboo-video.com/api/football/stats?for
player_data_regularseason = "https://cdnapi.bamboo-video.com/api/football/stats?forma
```

:[230] In

```
# Find the relevant links in the Premier League website
player_id ="https://cdnapi.bamboo-video.com/api/football/player?format=json&iid=57388
club_id =  "https://cdnapi.bamboo-video.com/api/football/team?format=json&iid=573881b
player_data_roundstage ="https://cdnapi.bamboo-video.com/api/football/stats?format=js
stage_ls =["RegularSeason","ChampionshipRound","RelegationRound","totalStage"]
# A function that convert the json files that were scraped from the website into data
def js_to_df(URL):
    page = requests.get(URL)
    js =page.json()
    js = js['data']
    ls_keys = list(js.keys())
    df = pd.DataFrame(data=[js[ls_keys[0]].values()])
    for i in range(1,len(ls_keys)):
        df= df.append([js[ls_keys[i]].values()])

    df.columns = js[ls_keys[0]].keys()
    return(df)

def js_to_df_rounds(URL,r,s):
    URL1 = URL.format(R=r,S=s)
    return js_to_df(URL1)

```

:[231] In

```
df_data_RegularSeason = pd.DataFrame()
df_data_ChampionshipRound = pd.DataFrame()
df_data_RelegationRound = pd.DataFrame()
for i in range(1,27):
    df_data_RegularSeason= pd.concat([df_data_RegularSeason,js_to_df_rounds(player_da

for i in range(1,8):
    df_data_ChampionshipRound= pd.concat([df_data_ChampionshipRound,js_to_df_rounds(p

for i in range(1,6):
    df_data_RelegationRound= pd.concat([df_data_RelegationRound,js_to_df_rounds(playe
```

:[232] In

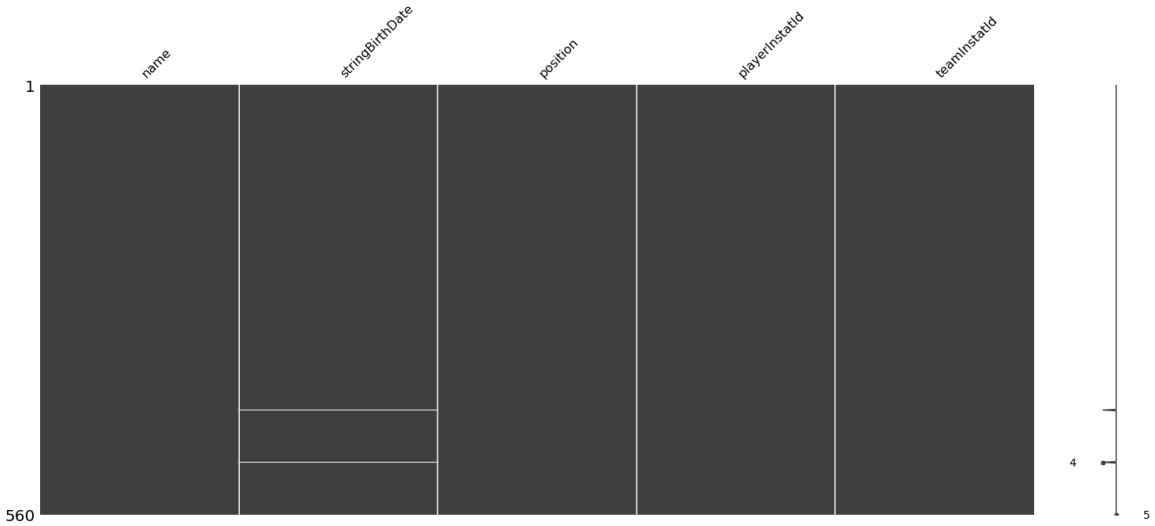
```
df_data_RegularSeason_team=df_data_RegularSeason[df_data_RegularSeason['playerId']==
df_data_RegularSeason_player=df_data_RegularSeason[df_data_RegularSeason['playerId']!
df_data_ChampionshipRound_team=df_data_ChampionshipRound[df_data_ChampionshipRound['p
df_data_ChampionshipRound_player=df_data_ChampionshipRound[df_data_ChampionshipRound|
df_data_RelegationRound_team=df_data_RelegationRound[df_data_RelegationRound['playerI
df_data_RelegationRound_player=df_data_RelegationRound[df_data_RelegationRound['playe
```

:[233] In

```
df_team_season_data = pd.concat([df_data_RegularSeason_team,df_data_ChampionshipRound
df_player_season_data = pd.concat([df_data_RegularSeason_player,df_data_ChampionshipR
df_player_id = js_to_df(player_id)
df_club_id = js_to_df(club_id)
```

:[234] In

```
# select relevat data and find na
df_player_filter = df_player_id.loc[:,['name','stringBirthDate','position','instatId'
msno.matrix(df_player_filter)
plt.show()
df_player_filter[df_player_filter.isnull().any(axis=1)]
```



Out[234]:

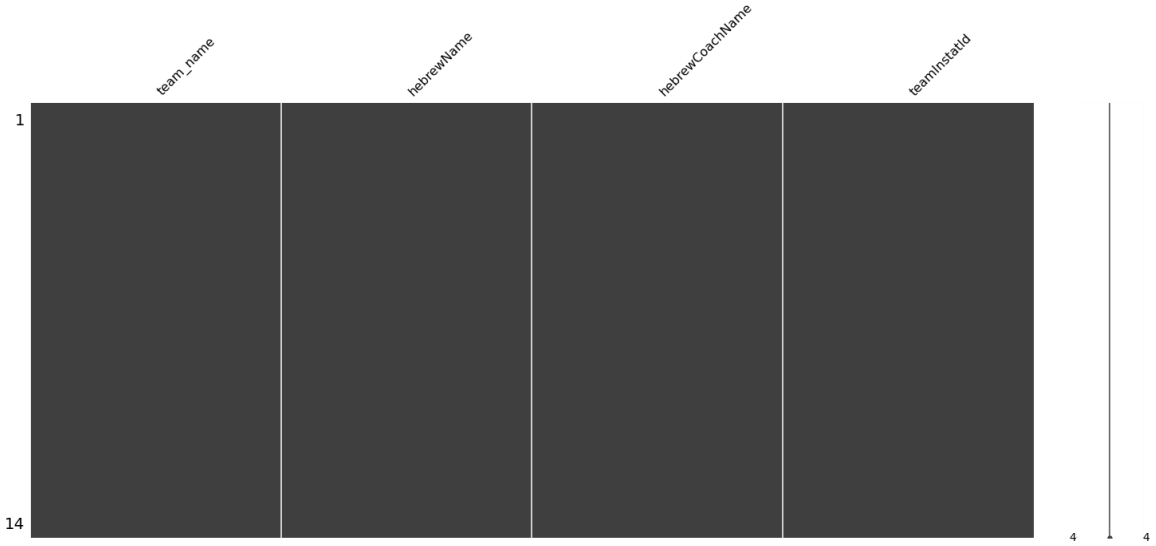
	teamInstatId	playerInstatId	position	stringBirthDate	name	
	2425	1724099	mid-fielder	None	Dolev Zilberberg	**423**
	130	1452318	mid-fielder	None	Leo Benbenisti	**491**

:[235] In

```
df_player_filter.loc[423,"stringBirthDate"]= "2002-01-01"
df_player_filter.loc[491,"stringBirthDate"]= "2004-01-01"
df_player_filter.stringBirthDate = pd.to_datetime(df_player_filter.stringBirthDate)
df_player_filter['player_age']= (pd.to_datetime("2023") - df_player_filter.stringBirt
```

:[236] In

```
# select relevat data and find na
df_club_filter=df_club_id[df_club_id['leagueId']==902].loc[:,['name','hebrewName','he
msno.matrix(df_club_filter)
plt.show()
df_club_filter.loc[11,"hebrewCoachName"]="זיו אריה"
df_club_filter.loc[13,"hebrewCoachName"]="שרון מימר"
```



:[237] In

```
# create a dataframe of palyer name,id,team name and team id
df_player_filter.teamInstatId = df_player_filter.teamInstatId.astype("int")
df_club_filter.teamInstatId = df_club_filter.teamInstatId.astype("int")
df_player_team=df_player_filter.merge(df_club_filter, on='teamInstatId')
```

:[238] In

```
# Create a stat data frame by player and team
df_player_season_data=df_player_season_data.merge(df_player_team,on="playerInstatId")
df_team_season_data= df_team_season_data.merge(df_player_team.loc[:,["hebrewName","he
```

:[241] In

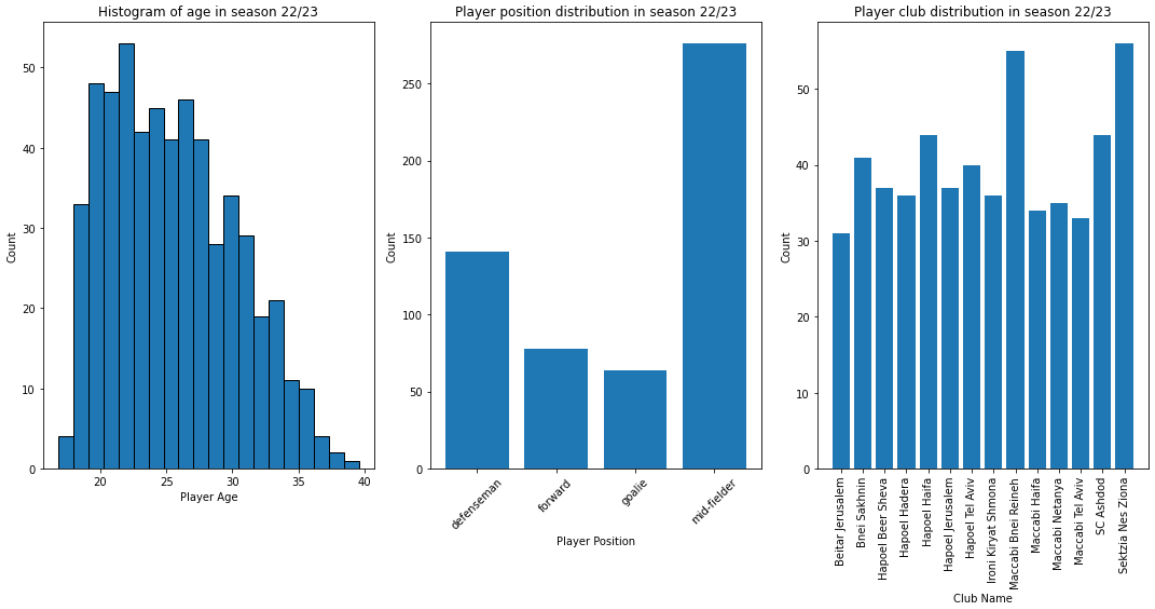
```
# Results
display(df_team_season_data.head())
display(df_player_season_data.tail())
```

curateAttackingPasses	chanceTotal	opponentGoal	opponentCross	opponentPenaltyShot_Goal
268	3	0	0	0
378	2	0	0	0
337	2	0	0	0
543	3	0	0	0
239	5	0	0	0

atId_y	position	stringBirthDate	name	accurateCrosses	accurateAttackingPasses	chanceTotal
1037	mid-fielder	2005-05-13	Noam Ben Harush	0	0	0
1037	mid-fielder	2005-05-13	Noam Ben Harush	0	1	0
130	forward	2004-04-22	Liad Ramot	0	2	2
130	forward	2004-04-22	Liad Ramot	0	3	0
13240	goalie	2003-01-07	Shahar Golan	0	0	0

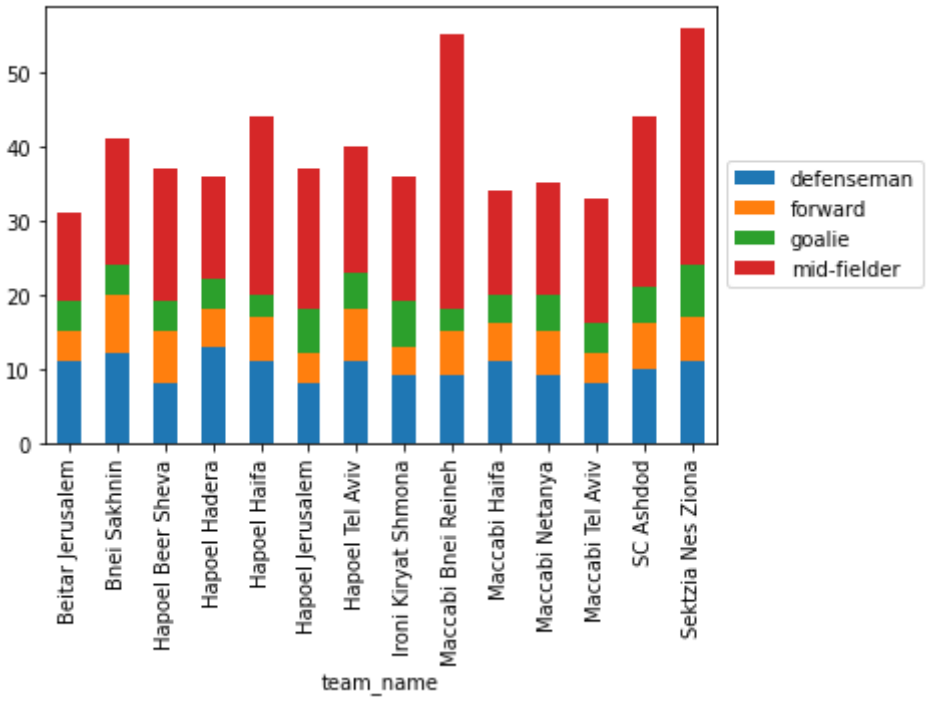
:[321] In

```
# Example, Expolratory Data analysis
fig,ax = plt.subplots(1,3,figsize=(15,8))
ax[0].hist(df_player_team['player_age'],edgecolor="black",bins=20)
ax[0].set_title("Histogram of age in season 22/23")
ax[0].set_ylabel("Count")
ax[0].set_xlabel("Player Age")

ax[1].bar(height=df_player_team.groupby("position").size(),x=list(df_player_team.grou
ax[1].set_title("Player position distribution in season 22/23")
ax[1].set_ylabel("Count")
ax[1].set_xlabel("Player Position")
ax[1].set_xticklabels(labels=list(df_player_filter.groupby("position").size().index),

ax[2].bar(height=df_player_team.groupby("team_name").size(),x=list(df_player_team.gro
ax[2].set_title("Player club distribution in season 22/23")
ax[2].set_ylabel("Count")
ax[2].set_xlabel("Club Name")
ax[2].set_xticklabels(labels=list(df_player_team.groupby("team_name").size().index),r

plt.tight_layout()
plt.show()
```



:[323] In

```
df_player_team.groupby(["team_name","position"]).size().unstack().plot.bar(stacked=Tr
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```



:[259] In

```
df_player_season_data.groupby("name").sum().GoalRegular.nlargest(10).plot.bar()
plt.ylabel("Count")
plt.xlabel("Player Name")
plt.title("Top 10 field goal scorers")
plt.show()
```