Investigate Soccer Database

1.Preface:

I will be using the soccer database which comes from Kaggle,this database contains data for more than 25,000 soccer matches,more than 10,000 players, and teams from several European countries from 2008 to 2016.

2. Preparing tools and the dataset:

I will be using the Anaconda environment to analyze the dataset, I have used python scripts , I have created an environment for this investigation and I have exported it to soccer_investigate.yaml, you can import it to the local computer by using this command:

```
conda env create -f soccer_investigate.yaml
```

you can run each script in the Anaconda Prompt using python word before each file name.

you can download the dataset from this site: https://www.kaggle.com/hugomathien/soccer,

and Extract the zip file called "soccer.zip".

3.Posing questions:

This analysis is answering on these important questions related to the soccer database:

1.Which league has the most number of matches in all of the time period?

2.Is there a relashionship between the type of the team (home, away) and the result of the match?

3.Is there a relationship between the type of the team and the average betting number from diffrent companies?

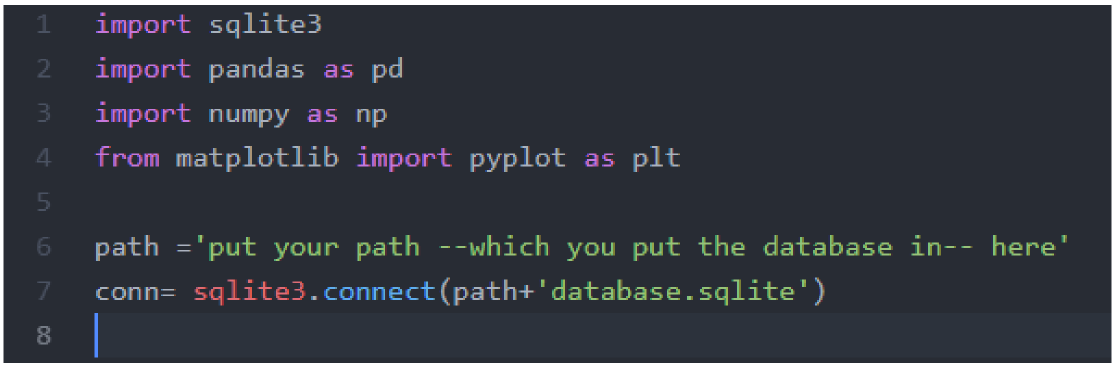
4.What teams improved the most over the time period?

5.What team attributes lead to the most vectories?

4.import libraries and Data collection:

we will import these python libraries:sqlite3 for create a connection with the soccer database, Pandas and Numpy for data ingestion and manipulation, Matplotlib for Data visualization .

To create the connection with the database ,we will use sqlite3.connect(path) function.



```python
import sqlite3
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

path ='put your path --which you put the database in-- here'
conn= sqlite3.connect(path+'database.sqlite')

```

1.we will perform the sql query by the pd.read_sql_query(query,conn) function in every investigation and store the result in a dataframe.

2.in every plot i have written plt.show() function to show the visualization, when the code is executed.

3. in every use for the year, i have used substr(date,1,4) function to take the first four characters which are the year number, that sqlite doesn't have a convert function to date and time.

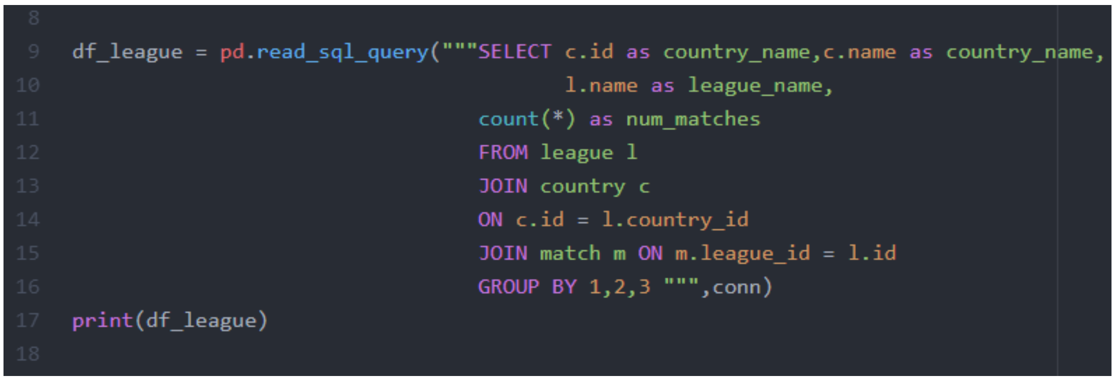
5.data analysis step by step for each question

Now let's investigate our question from exploring data, to wrangling and cleaning then plotting to make the conclusions :

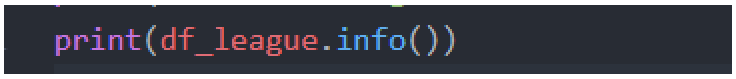
Q1: Which league has the most number of matches in all of the time period?

from looking at the database and reading about it , i have had a good experience in how the tables relate to each other.the analysis code of this question is in (df_leagues.py)

In the gathering data step, I have read the below sql query To answer this question we want to count number of matches grouped by the leagues id ,I have joined the league table with country table and with the match table to know the names of leagues and related countries, and I have stored it in df_league dataframe.

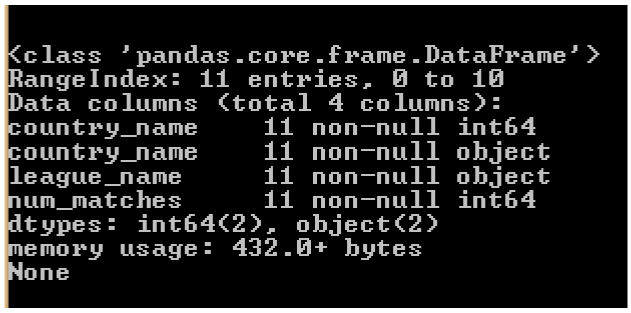


```
df_league = pd.read_sql_query("""SELECT c.id as country_name,c.name as country_name,
                                           l.name as league_name,
                                    count(*) as num_matches
                                    FROM league l
                                    JOIN country c
                                    ON c.id = l.country_id
                                    JOIN match m ON m.league_id = l.id
                                    GROUP BY 1,2,3 """,conn)
print(df_league)
```

Then I have assessed df_league by checking if there are null values with this function: 

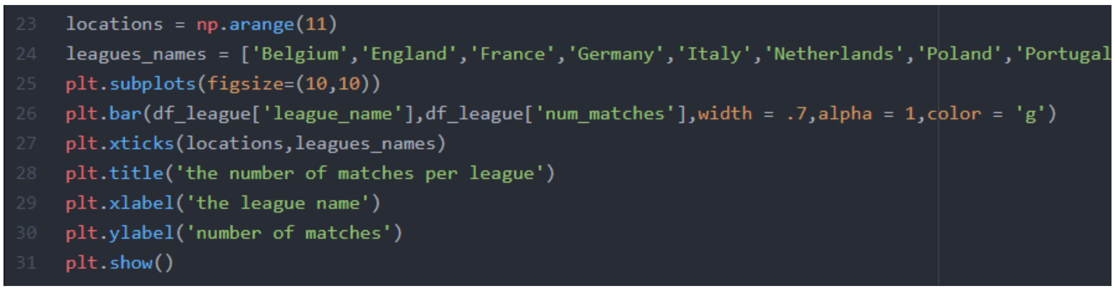
```
print(df_league.info())
```

and from the output ,theres is no null values:

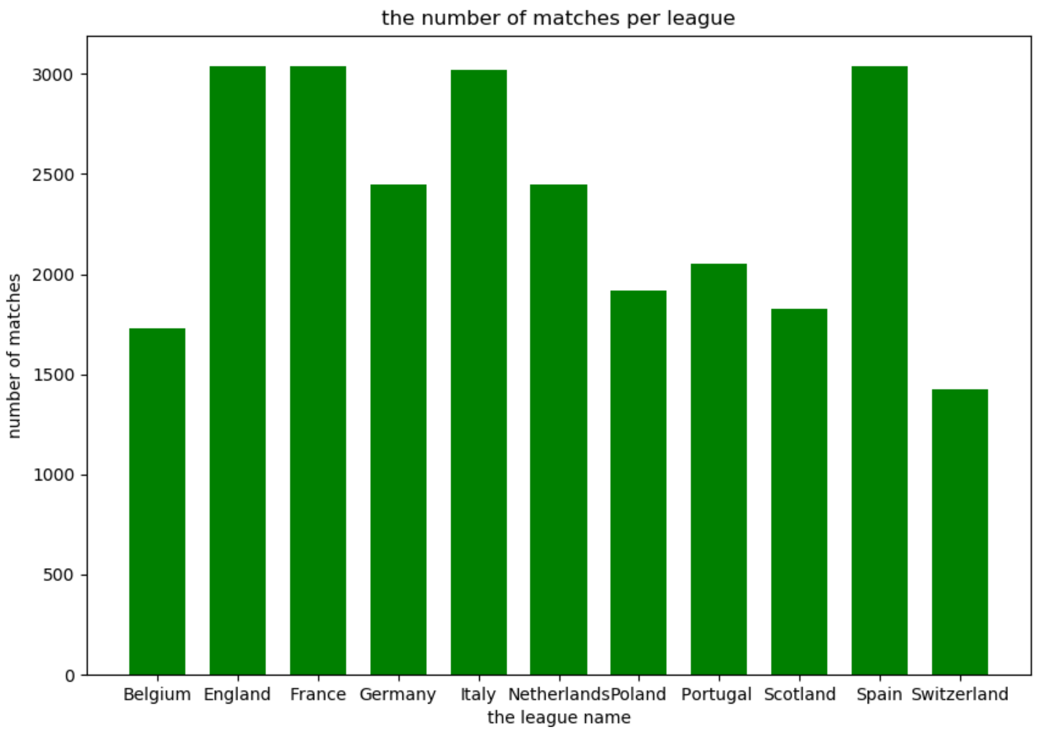


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 4 columns):
country_name     11 non-null int64
country_name     11 non-null object
league_name      11 non-null object
num_matches      11 non-null int64
dtypes: int64(2), object(2)
memory usage: 432.0+ bytes
None
```

Then I visualize it using matplotlib library with bar function :



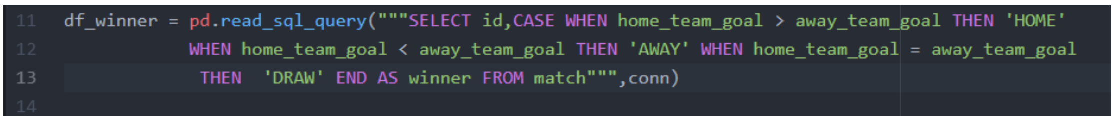
```
locations = np.arange(11)
leagues_names = ['Belgium','England','France','Germany','Italy','Netherlands','Poland','Portugal
plt.subplots(figsize=(10,10))
plt.bar(df_league['league_name'],df_league['num_matches'],width = .7,alpha = 1,color = 'g')
plt.xticks(locations,leagues_names)
plt.title('the number of matches per league')
plt.xlabel('the league name')
plt.ylabel('number of matches')
plt.show()
```



From the visual we can see the leagues have the most number of matches are England,France,Italy,and Spain ,which have more than 3000 matches in all of the time period,and the lowest number of matches is in Switzerland.

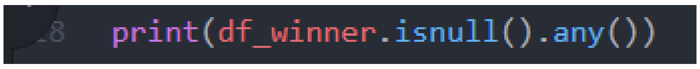
Q2.Is there a relashionship between the type of the team (home, away) and the result of the match?

In this simple analysis which is in the python script (winners.py), i have performed a new sql query in the gathering data step to catch the id for each match with a new column created with case statement that has the type of the winner team (H for home,D for draw,A for away) , and have stored to df_winner data frame.



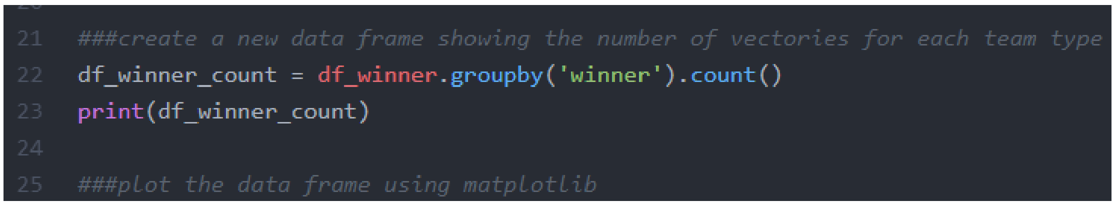
```
df_winner = pd.read_sql_query("""SELECT id,CASE WHEN home_team_goal > away_team_goal THEN 'HOME'
            WHEN home_team_goal < away_team_goal THEN 'AWAY' WHEN home_team_goal = away_team_goal
             THEN  'DRAW' END AS winner FROM match""",conn)
```

Then I have checked df_winner for missing values and there are no null values:

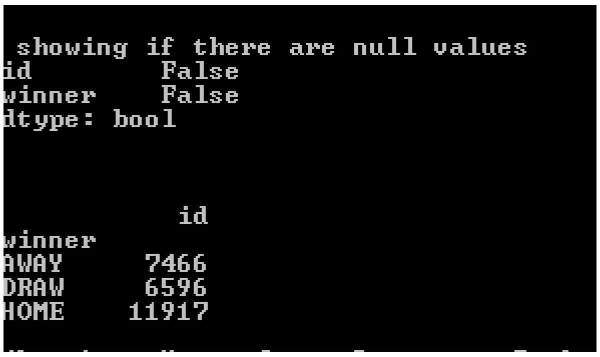


```
print(df_winner.isnull().any())
```

To make conclusions I have grouped it by the winner team type to count the number of victories for each team type.



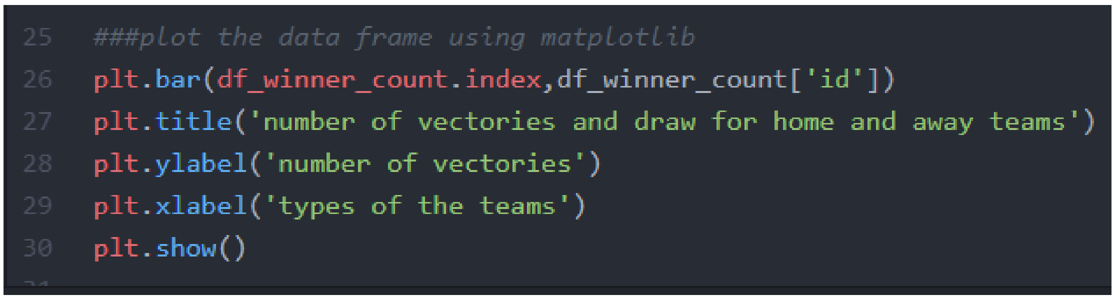
```
###create a new data frame showing the number of vectories for each team type
df_winner_count = df_winner.groupby('winner').count()
print(df_winner_count)

###plot the data frame using matplotlib
```



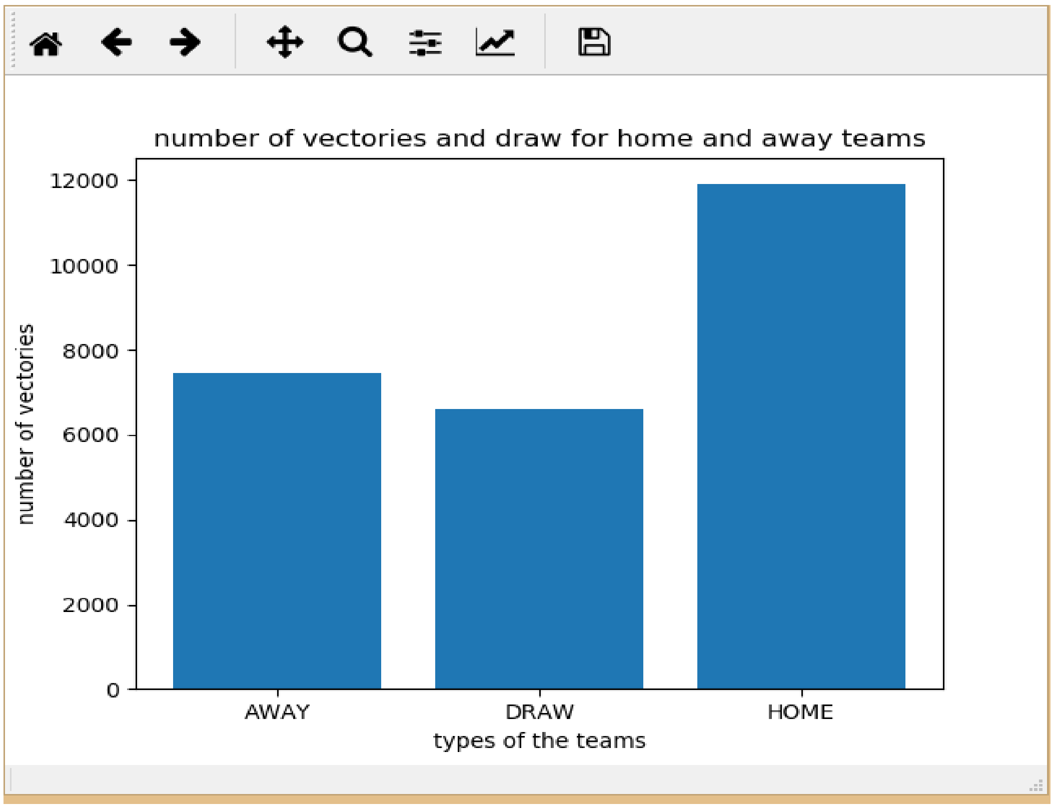
```
 showing if there are null values
id          False
winner      False
dtype: bool


               id
winner
AWAY        7466
DRAW        6596
HOME       11917
```

this is the output:

then i have plotted it using matplotlib library:



```
###plot the data frame using matplotlib
plt.bar(df_winner_count.index,df_winner_count['id'])
plt.title('number of vectories and draw for home and away teams')
plt.ylabel('number of vectories')
plt.xlabel('types of the teams')
plt.show()
```



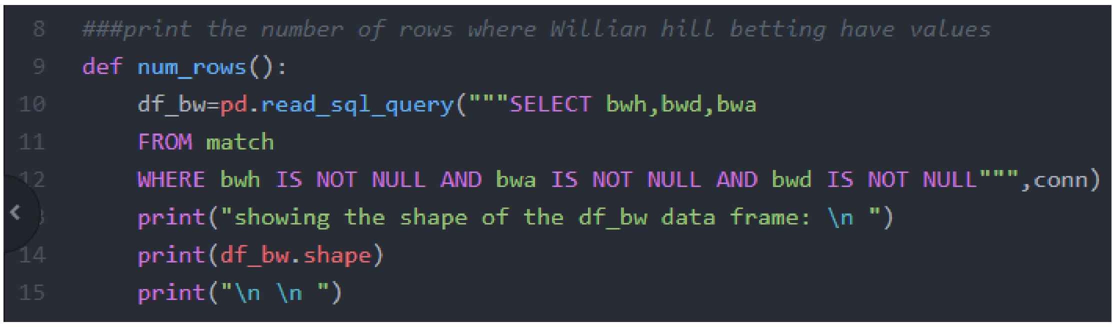
From this visualization we can see clearly that the home team in general have more chance to win in the match.

Q3.Is there a relationship between the type of the team and the average betting number from diffrent companies?

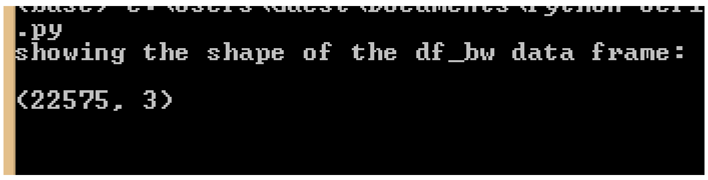
In this database ,in the match table, 10 betting companies are listed in 30 columns ,showing the betting number for home,draw, and away ,but these columns have a lot of null values, these values for each match is close to each other ,so i have determined the average for all of them for each match .that 30 columns will be 3 columns.

In this python script(betting.py)I have divided the code into functions to be clear and general as possible.

In the beginning ,I have seen the number of matches with no null values for william hill betting (BWH,BWA,BWD) ,using the function that i have defined num_rows(), which is excuted in the main function(as i will remind that).



```
###print the number of rows where Willian hill betting have values
def num_rows():
    df_bw=pd.read_sql_query("""SELECT bwh,bwd,bwa
    FROM match
    WHERE bwh IS NOT NULL AND bwa IS NOT NULL AND bwd IS NOT NULL""",conn)
    print("showing the shape of the df_bw data frame: \n ")
    print(df_bw.shape)
    print("\n \n ")
```

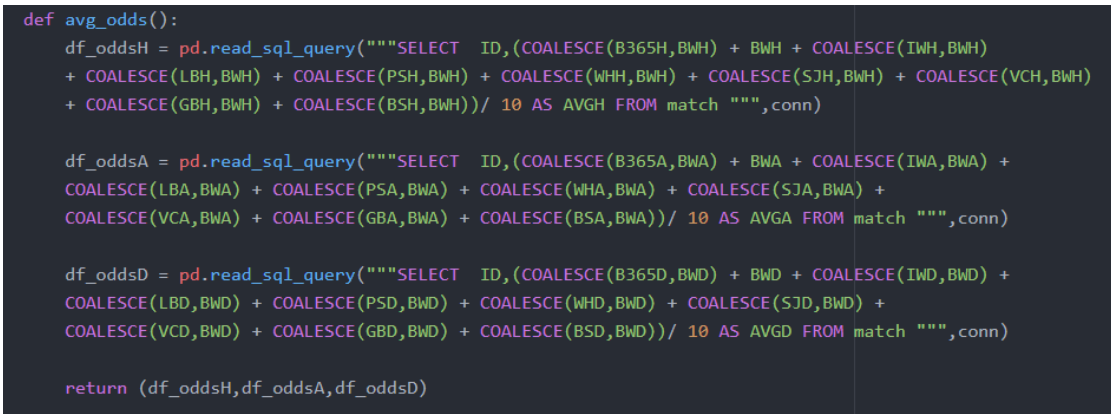


```
.py
showing the shape of the df_bw data frame:

(22575, 3)
```

The number of rows is more than 22000 ,which is good amount of data to investigate the diffrence in the betting number for each mach result(H,A,D).

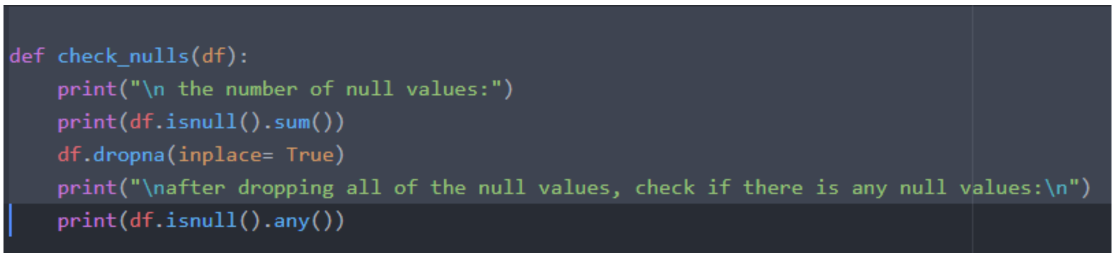
So I have determined the average with COALESCE function to fill any null value with the (BWH,BWA,BWD) , using function avg_odds(),this function will return a tuple with three data frames

Now there are three data frames for the average betting number for the expected result of each match(H,A,D)



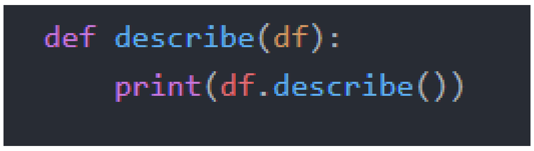
```
def avg_odds():
    df_oddsH = pd.read_sql_query("""SELECT  ID,(COALESCE(B365H,BWH) + BWH + COALESCE(IWH,BWH)
    + COALESCE(LBH,BWH) + COALESCE(PSH,BWH) + COALESCE(WHH,BWH) + COALESCE(SJH,BWH) + COALESCE(VCH,BWH)
    + COALESCE(GBH,BWH) + COALESCE(BSH,BWH))/ 10 AS AVGH FROM match """,conn)

    df_oddsA = pd.read_sql_query("""SELECT  ID,(COALESCE(B365A,BWA) + BWA + COALESCE(IWA,BWA) +
    COALESCE(LBA,BWA) + COALESCE(PSA,BWA) + COALESCE(WHA,BWA) + COALESCE(SJA,BWA) +
    COALESCE(VCA,BWA) + COALESCE(GBA,BWA) + COALESCE(BSA,BWA))/ 10 AS AVGA FROM match """,conn)

    df_oddsD = pd.read_sql_query("""SELECT  ID,(COALESCE(B365D,BWD) + BWD + COALESCE(IWD,BWD) +
    COALESCE(LBD,BWD) + COALESCE(PSD,BWD) + COALESCE(WHD,BWD) + COALESCE(SJD,BWD) +
    COALESCE(VCD,BWD) + COALESCE(GBD,BWD) + COALESCE(BSD,BWD))/ 10 AS AVGD FROM match """,conn)

    return (df_oddsH,df_oddsA,df_oddsD)
```

Then in the assessing step, I have checked for the number of null values,as wrangle the data, I have dropped the rows with missing values , then I have checked again to be sure ,all these steps have put in the defined function check_nulls(df).



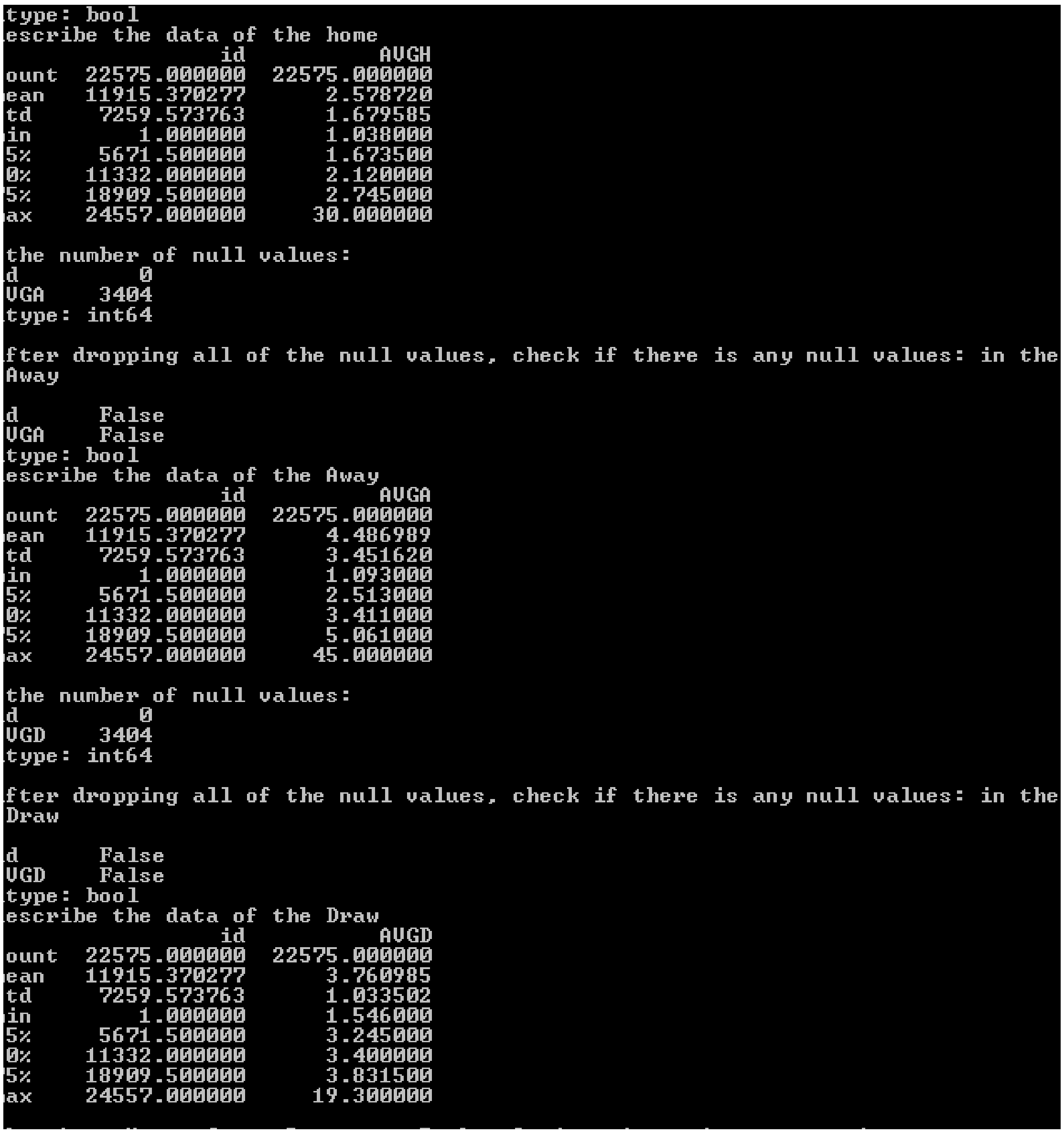
```
def check_nulls(df):
    print("\n the number of null values:")
    print(df.isnull().sum())
    df.dropna(inplace= True)
    print("\nafter dropping all of the null values, check if there is any null values:\n")
    print(df.isnull().any())
```

After that i have used the describe method for each data frame,I put it in a function called describe.



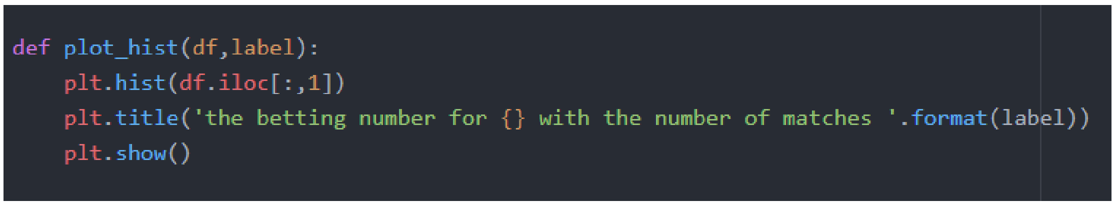
```
def describe(df):
    print(df.describe())
```

The output of the describe function and the check_nulls function, when i execute it in the main:



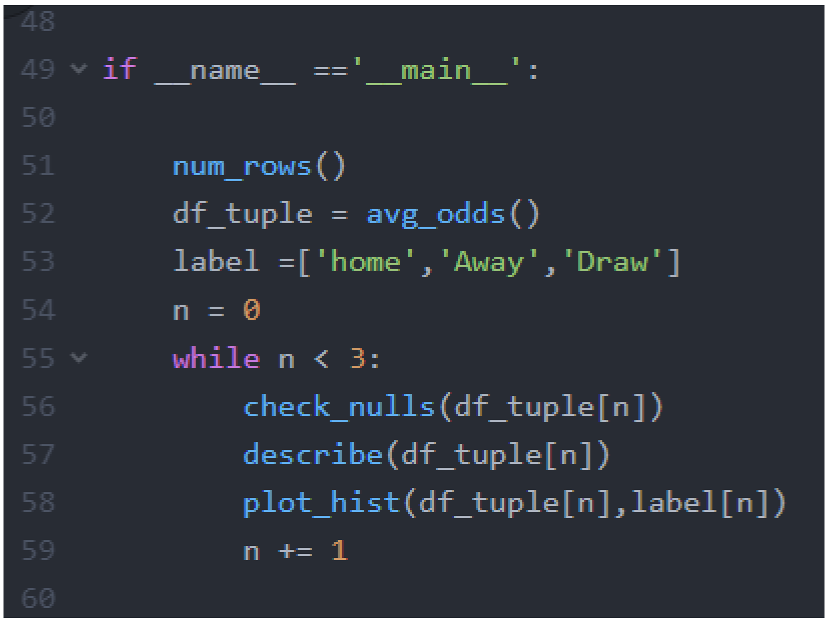
```
type: bool
escribe the data of the home
                 id          AVGH
ount  22575.000000  22575.000000
ean   11915.370277      2.578720
td     7259.573763      1.679585
in        1.000000      1.038000
5%     5671.500000      1.673500
0%    11332.000000      2.120000
5%    18909.500000      2.745000
ax    24557.000000     30.000000

the number of null values:
d         0
VGA    3404
type: int64

fter dropping all of the null values, check if there is any null values: in the
Away

d      False
VGA    False
type: bool
escribe the data of the Away
                 id          AVGA
ount  22575.000000  22575.000000
ean   11915.370277      4.486989
td     7259.573763      3.451620
in        1.000000      1.093000
5%     5671.500000      2.513000
0%    11332.000000      3.411000
5%    18909.500000      5.061000
ax    24557.000000     45.000000

the number of null values:
d         0
VGD    3404
type: int64

fter dropping all of the null values, check if there is any null values: in the
Draw

d      False
VGD    False
type: bool
escribe the data of the Draw
                 id          AVGD
ount  22575.000000  22575.000000
ean   11915.370277      3.760985
td     7259.573763      1.033502
in        1.000000      1.546000
5%     5671.500000      3.245000
0%    11332.000000      3.400000
5%    18909.500000      3.831500
ax    24557.000000     19.300000
```

then I visualize three hisograms for these information using the hist function in matplotlib library in the function that i have defined, plot_hist(df,label).



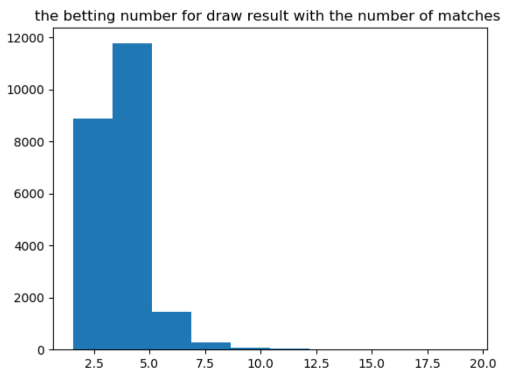
```python
def plot_hist(df,label):
    plt.hist(df.iloc[:,1])
    plt.title('the betting number for {} with the number of matches '.format(label))
    plt.show()
```

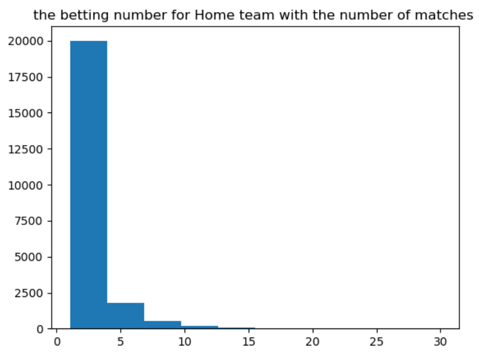
finally, I have executed these functions in the main :

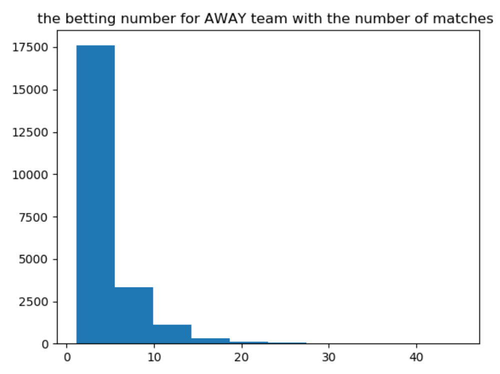


```python
if __name__ =='__main__':

    num_rows()
    df_tuple = avg_odds()
    label =['home','Away','Draw']
    n = 0
    while n < 3:
        check_nulls(df_tuple[n])
        describe(df_tuple[n])
        plot_hist(df_tuple[n],label[n])
        n += 1

```

the output of visualizations:





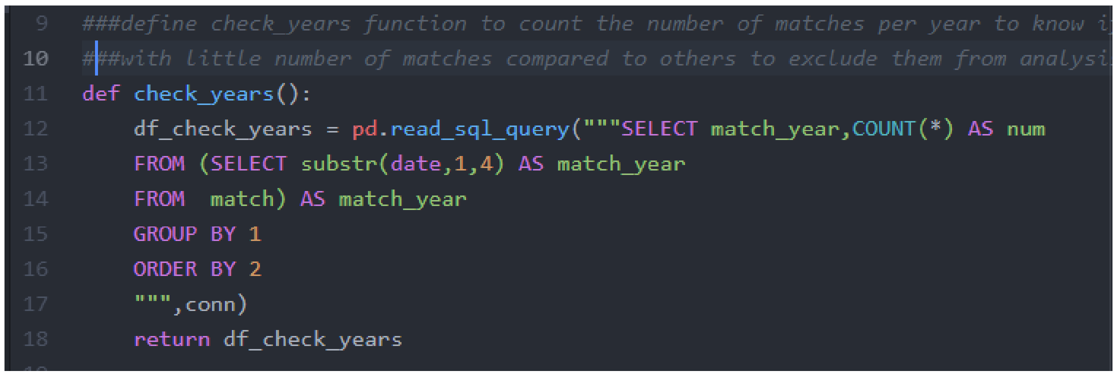


From the output of visualization and the description of the data, the away teams have the biggest betting number, and the mean of the away betting numbers is the highest .

Q4.What teams improved the most over the time period?

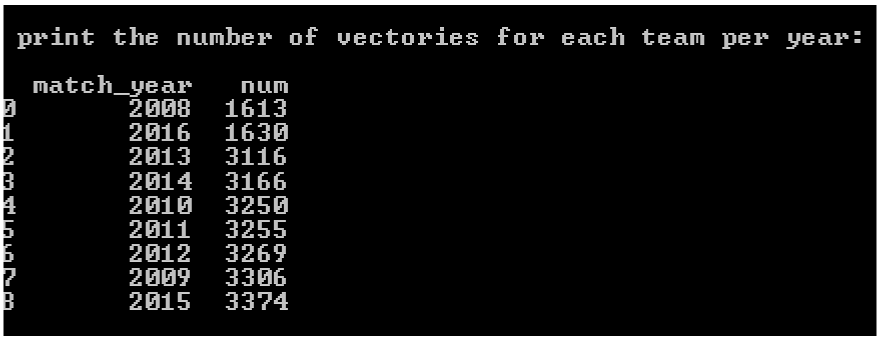
I have divided the python script(improve_teams.py) in many functions:

1.check_years () function: to count the number of matches per year ,to know if there are years with little number of matches compared to others to exclude them from analysis. for this i have used a simple query for that and the function will return a data frame fetches the wanted data.



```
###define check_years function to count the number of matches per year to know i
###with little number of matches compared to others to exclude them from analysi
def check_years():
    df_check_years = pd.read_sql_query("""SELECT match_year,COUNT(*) AS num
    FROM (SELECT substr(date,1,4) AS match_year
    FROM  match) AS match_year
    GROUP BY 1
    ORDER BY 2
    """,conn)
    return df_check_years
```

after I have executed this function in the main, the output is like this:

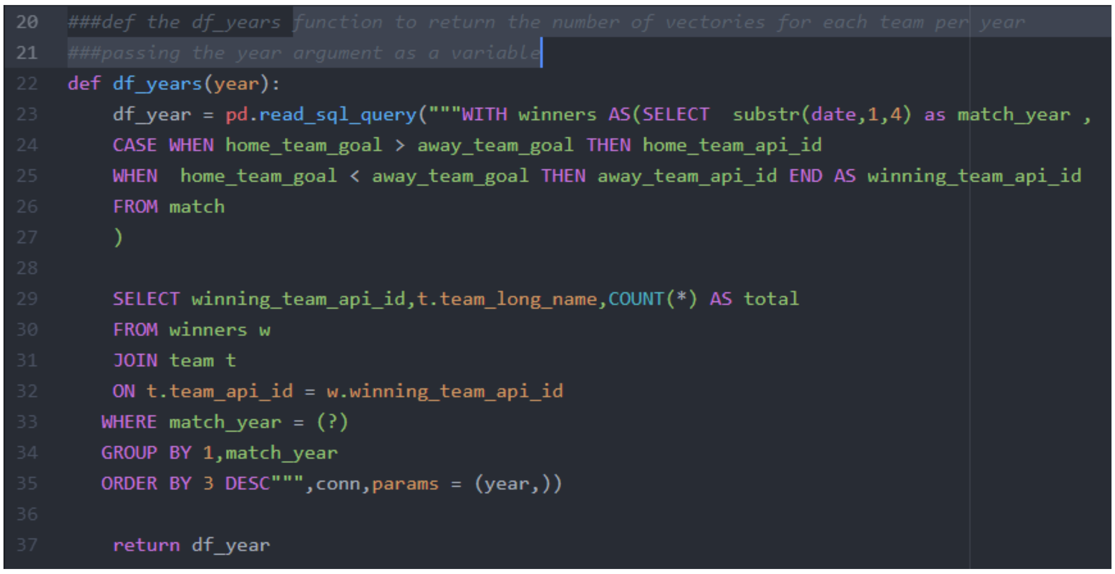


```
print the number of vectories for each team per year:

  match_year   num
0       2008  1613
1       2016  1630
2       2013  3116
3       2014  3166
4       2010  3250
5       2011  3255
6       2012  3269
7       2009  3306
8       2015  3374
```

from this data, the years 2008 and 2016 have the lowest number of matches which means to ignore these years in our analysis.

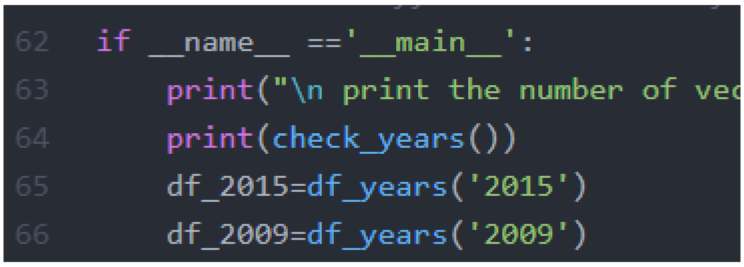
2.df_years() function: this function for return the number of victories for each team per year ,passing the year argument as a variable, to be general for any year we want to know about it .

In the sql query, i have used a CTE to fetch a table with the year of each match and the team_api_id for the winning team, then i used it to know the number of victories for each team in the specified year(which is an argument in the function) , then store the result data in the df_year data frame and return it .



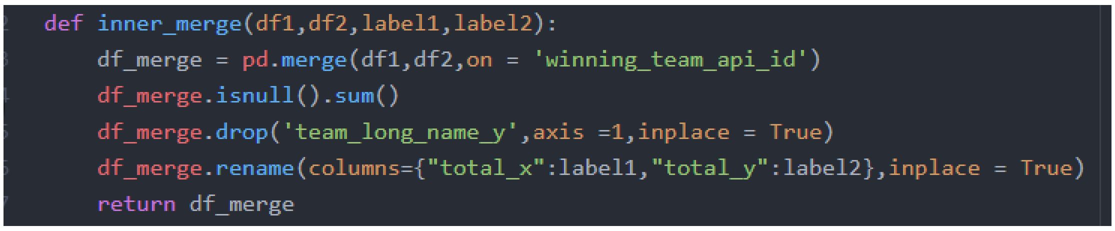
```
###def the df_years function to return the number of vectories for each team per year
###passing the year argument as a variable
def df_years(year):
    df_year = pd.read_sql_query("""WITH winners AS(SELECT  substr(date,1,4) as match_year ,
    CASE WHEN home_team_goal > away_team_goal THEN home_team_api_id
    WHEN  home_team_goal < away_team_goal THEN away_team_api_id END AS winning_team_api_id
    FROM match
    )

    SELECT winning_team_api_id,t.team_long_name,COUNT(*) AS total
    FROM winners w
    JOIN team t
    ON t.team_api_id = w.winning_team_api_id
   WHERE match_year = (?)
   GROUP BY 1,match_year
   ORDER BY 3 DESC""",conn,params = (year,))

    return df_year
```

From this function we want to know the number of victories for each team in 2009 and 2015 ,to know which teams improved the most by taking the diffrence between the two values ,so i have passed a 2015 and 2009 as arguments in the main and store the two data frames in df_2015 and df_2009.



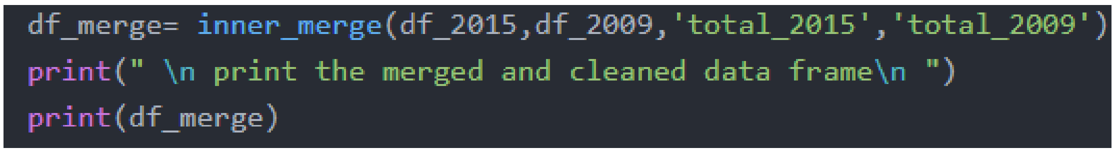
```python
if __name__ =='__main__':
    print("\n print the number of vec
    print(check_years())
    df_2015=df_years('2015')
    df_2009=df_years('2009')
```

3.inner_merge()function: this function for making an inner merge between two data frames and test it for missing values, there are no missing values .the merge has produced a repeated column(team_long_name),drop the repeated column ,and renaming the two columns that have the total number of victories for each team in the specific year .



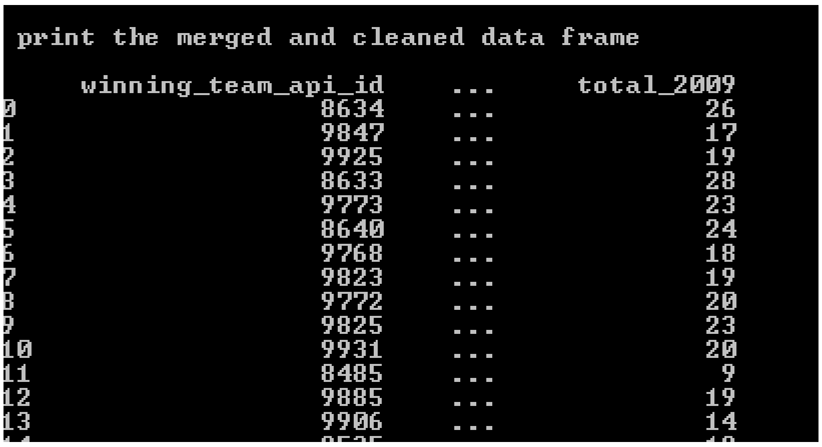
```python
def inner_merge(df1,df2,label1,label2):
    df_merge = pd.merge(df1,df2,on = 'winning_team_api_id')
    df_merge.isnull().sum()
    df_merge.drop('team_long_name_y',axis =1,inplace = True)
    df_merge.rename(columns={"total_x":label1,"total_y":label2},inplace = True)
    return df_merge
```

I have passed the df_2015 and df_2009 to this function and new names for the total columns.



```python
df_merge= inner_merge(df_2015,df_2009,'total_2015','total_2009')
print(" \n print the merged and cleaned data frame\n ")
print(df_merge)
```

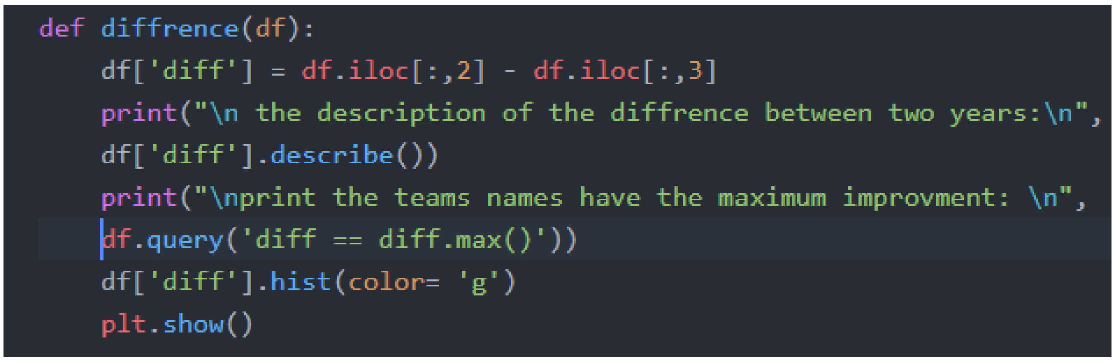
the output of this code:



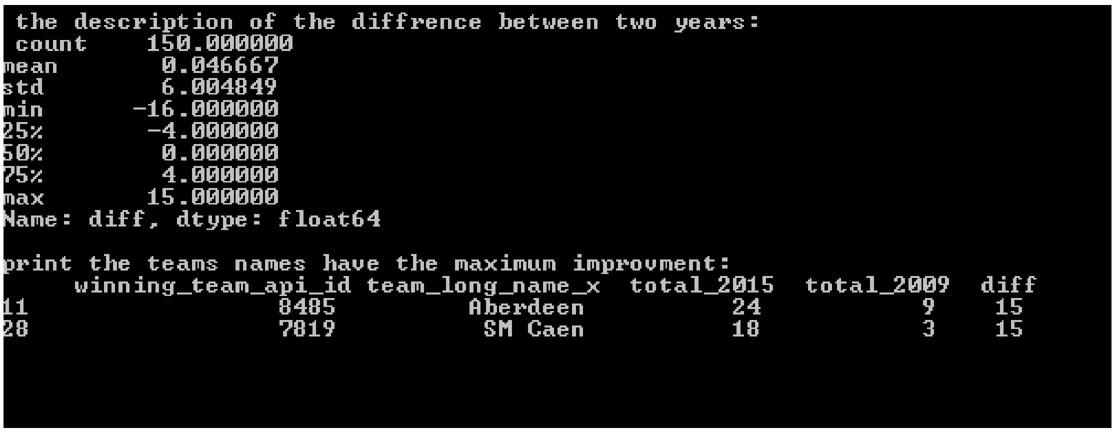
```
print the merged and cleaned data frame

    winning_team_api_id    ...     total_2009
0                  8634    ...             26
1                  9847    ...             17
2                  9925    ...             19
3                  8633    ...             28
4                  9773    ...             23
5                  8640    ...             24
6                  9768    ...             18
7                  9823    ...             19
8                  9772    ...             20
9                  9825    ...             23
10                 9931    ...             20
11                 8485    ...              9
12                 9885    ...             19
13                 9906    ...             14
```

4. the difference() function: this function has taken the cleaned df_merge to calculate the difference between the number of

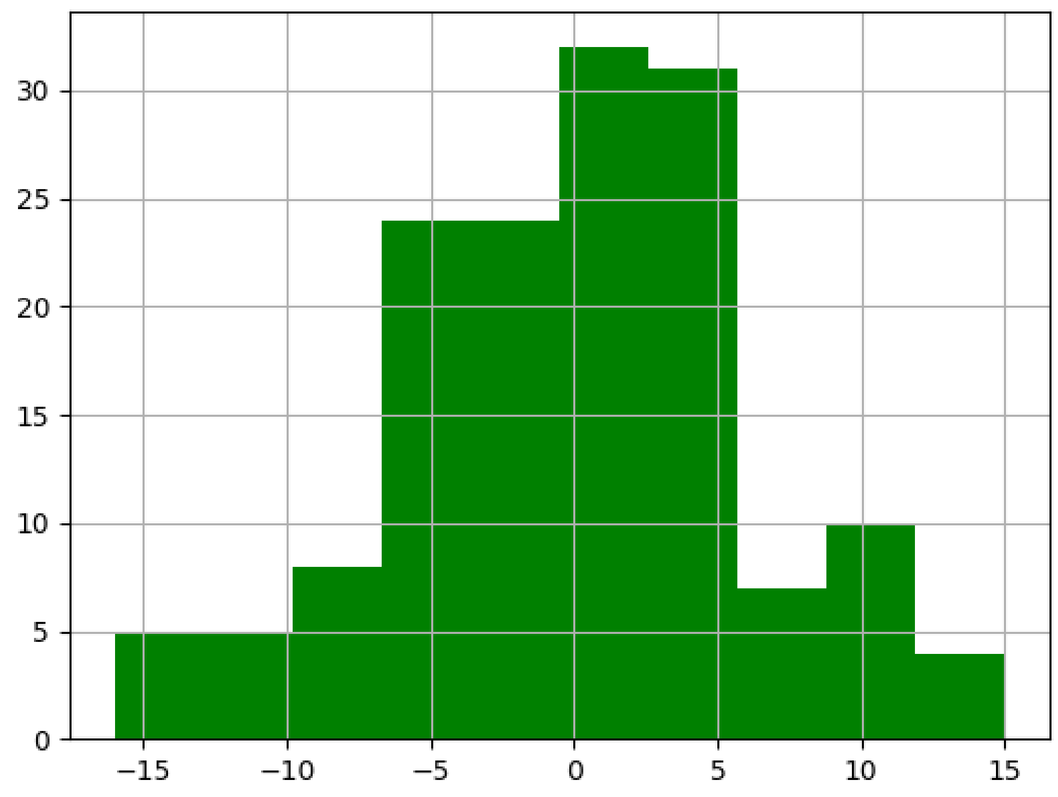
victories for each team in 2015 and 2009 , the new column is called diff , i have shown the description of this column and the max values for it , then i query the rows have the maximum difference , and i plot a histogram for this column.



```
def diffrence(df):
    df['diff'] = df.iloc[:,2] - df.iloc[:,3]
    print("\n the description of the diffrence between two years:\n",
    df['diff'].describe())
    print("\nprint the teams names have the maximum improvment: \n",
    df.query('diff == diff.max()'))
    df['diff'].hist(color= 'g')
    plt.show()
```



```
 the description of the diffrence between two years:
 count    150.000000
mean        0.046667
std         6.004849
min       -16.000000
25%        -4.000000
50%         0.000000
75%         4.000000
max        15.000000
Name: diff, dtype: float64

print the teams names have the maximum improvment:
     winning_team_api_id team_long_name_x  total_2015  total_2009  diff
11                  8485         Aberdeen          24           9    15
28                  7819          SM Caen          18           3    15
```

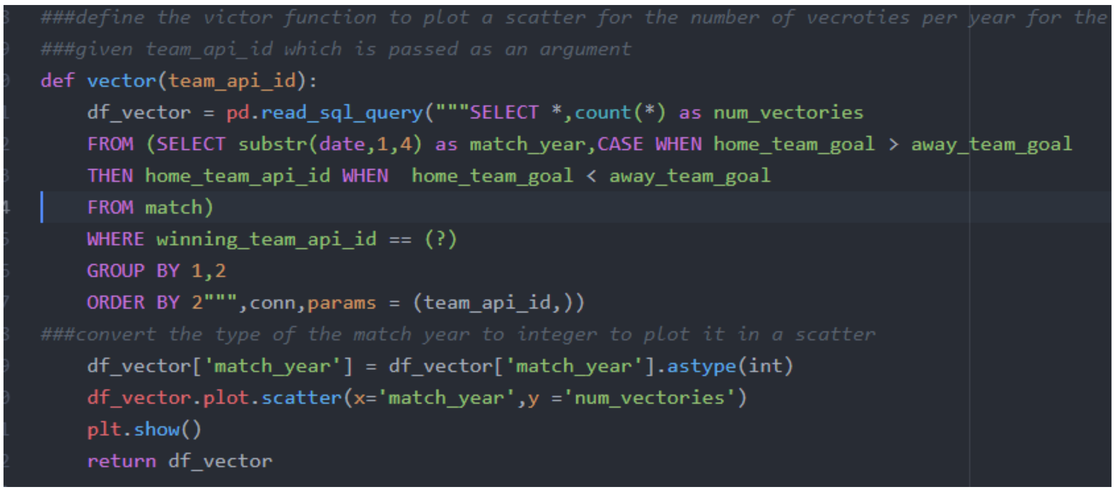


From the output of this script, the maximum difference in the number of victories between 2015 and 2009 is 15 for the (Aberdeen,SM Cean) teams .

5.What team attributes lead to the most victories?

In this analysis, I have divided the python scripts(team_attri.py) into many functions, then i have called them in the main with passing the suitable argument.

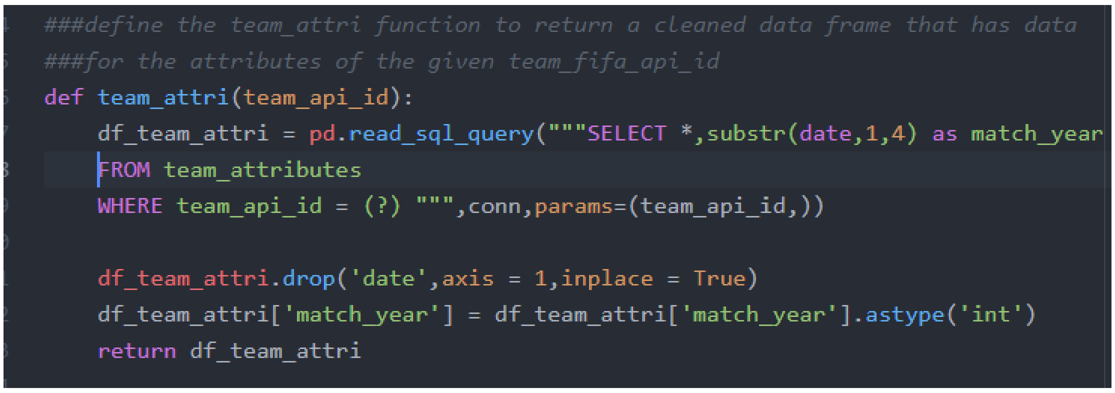
1.vector () function: this function have been defined to create a data frame from the database which has for the number of victories per year for the given team_api_id which is passed as argument ,it is variable ,i have put the variable in an sql query with params parameter in the read_sql_query () function and i have assigned it to a tuple (team_api_id,), then i have plotted the resulting data frame as a scatter to see the improvement of each team .



```
###define the victor function to plot a scatter for the number of vecroties per year for the
###given team_api_id which is passed as an argument
def vector(team_api_id):
    df_vector = pd.read_sql_query("""SELECT *,count(*) as num_vectories
    FROM (SELECT substr(date,1,4) as match_year,CASE WHEN home_team_goal > away_team_goal
    THEN home_team_api_id WHEN  home_team_goal < away_team_goal
    FROM match)
    WHERE winning_team_api_id == (?)
    GROUP BY 1,2
    ORDER BY 2""",conn,params = (team_api_id,))
###convert the type of the match year to integer to plot it in a scatter
    df_vector['match_year'] = df_vector['match_year'].astype(int)
    df_vector.plot.scatter(x='match_year',y ='num_vectories')
    plt.show()
    return df_vector
```

I have read the sql query ,then i have converted the match_year column type to be an integer to plot it in a scatter

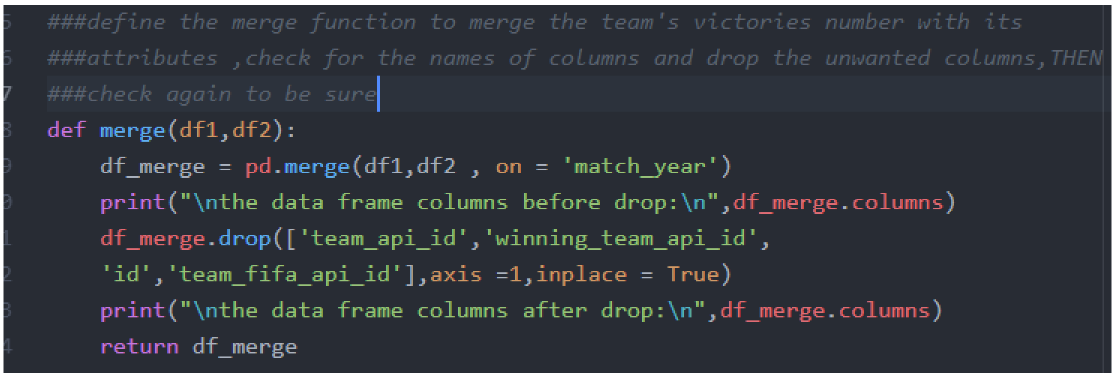
2.team_attri() function: this function return a cleaned data frame that has all of the team attributes for each year .



```
###define the team_attri function to return a cleaned data frame that has data
###for the attributes of the given team_fifa_api_id
def team_attri(team_api_id):
    df_team_attri = pd.read_sql_query("""SELECT *,substr(date,1,4) as match_year
    FROM team_attributes
    WHERE team_api_id = (?) """,conn,params=(team_api_id,))

    df_team_attri.drop('date',axis = 1,inplace = True)
    df_team_attri['match_year'] = df_team_attri['match_year'].astype('int')
    return df_team_attri
```

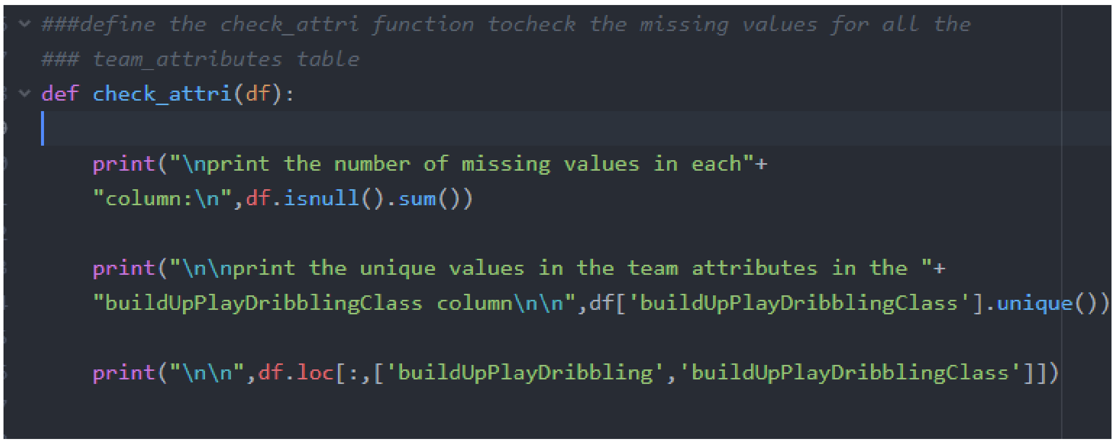
Then i have dropped the date column, because there is the match_year instead, and i have converted the type of the match_year to be integer.

3.merge() function: this function for merging the last two data frames to be the number of victories per year and the other attributes in the same data frame to visualize the relationship between them. then i have dropped the unwanted columns.



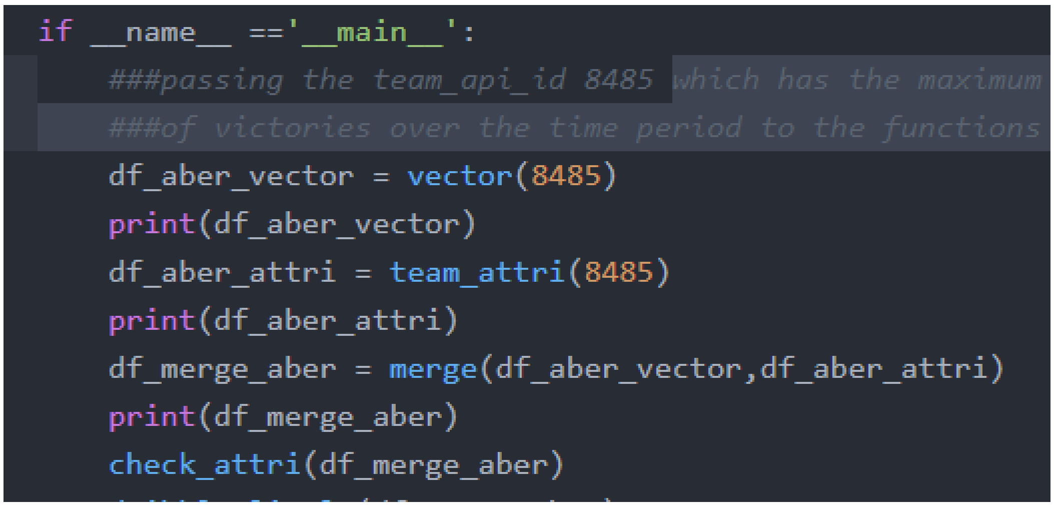
```python
###define the merge function to merge the team's victories number with its
###attributes ,check for the names of columns and drop the unwanted columns,THEN
###check again to be sure
def merge(df1,df2):
    df_merge = pd.merge(df1,df2 , on = 'match_year')
    print("\nthe data frame columns before drop:\n",df_merge.columns)
    df_merge.drop(['team_api_id','winning_team_api_id',
    'id','team_fifa_api_id'],axis =1,inplace = True)
    print("\nthe data frame columns after drop:\n",df_merge.columns)
    return df_merge
```

4.check_attri() function : this function is for assessing the data and checking for null values , then printing the unique values for the buildupplaydribblingclass column,and printing the buildupplaydribbling and buildupplaydribblingclass columns values only, like this:



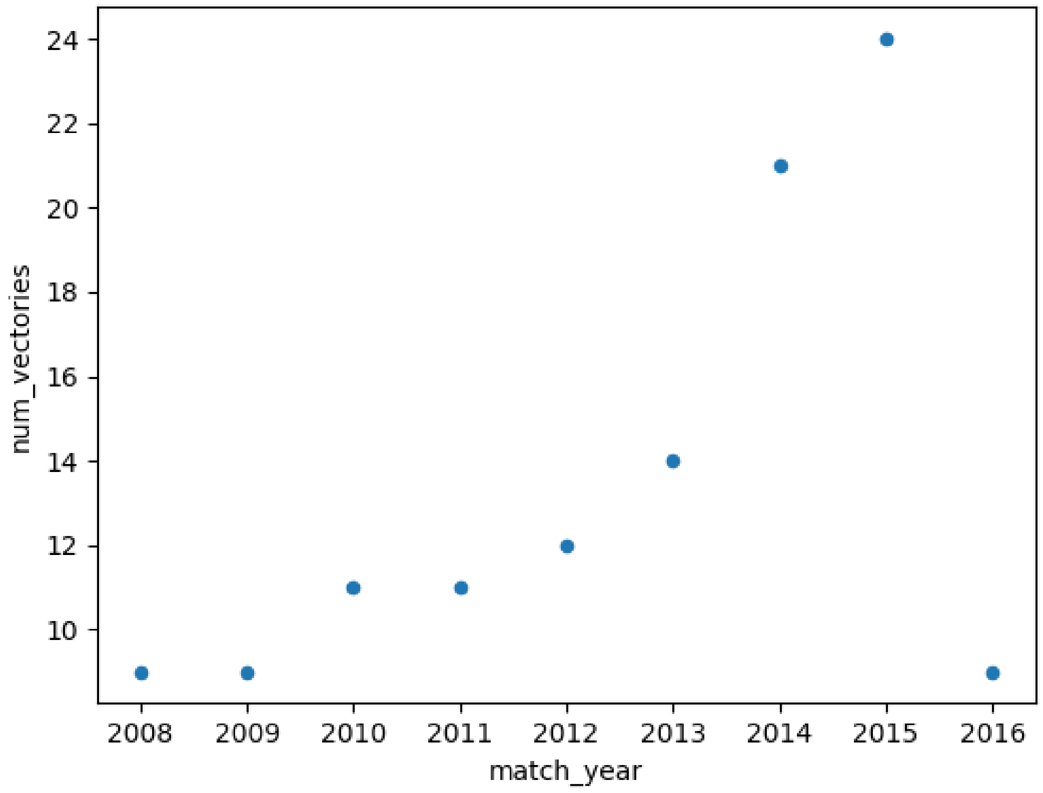
```python
###define the check_attri function tocheck the missing values for all the
### team_attributes table
def check_attri(df):

    print("\nprint the number of missing values in each"+
    "column:\n",df.isnull().sum())

    print("\n\nprint the unique values in the team attributes in the "+
    "buildUpPlayDribblingClass column\n\n",df['buildUpPlayDribblingClass'].unique())

    print("\n\n",df.loc[:,['buildUpPlayDribbling','buildUpPlayDribblingClass']])
```

in the main function, i have passed the team_api_id=8485 which has the maximum diffrence in the number of victories over the time period to the functions above like this:



```
if __name__ =='__main__':
    ###passing the team_api_id 8485 which has the maximum
    ###of victories over the time period to the functions
    df_aber_vector = vector(8485)
    print(df_aber_vector)
    df_aber_attri = team_attri(8485)
    print(df_aber_attri)
    df_merge_aber = merge(df_aber_vector,df_aber_attri)
    print(df_merge_aber)
    check_attri(df_merge_aber)
```

The result of this code is a scatter that shows clearly the linear relashionship between the years and the number of victories for this team:



The merged result has null values as the result of the check_attri()function ,when passing the merged df_merge_aber to it , to fix the nulls in that column,i have defined a new functions for that.

buildupplaydribbling column has many missing vlaues ,but also the buildupplaydribblingclass column hasn't nulls, so the function is defined to fill the missing values with the mean of the numbers that have the same class in the buildupplaydribblingclass column .

When i have performed this query :

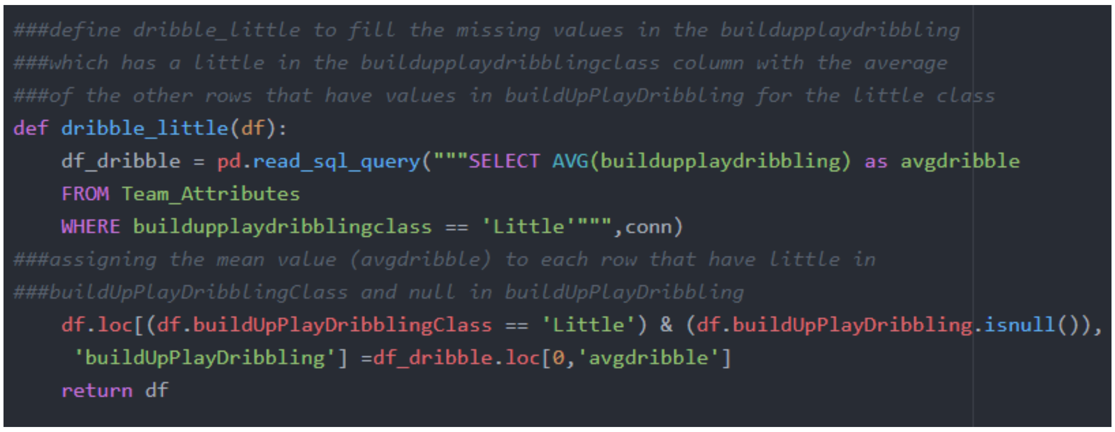
df=pd.read_sql_query("""SELECT DISTINCT buildUpPlayDribblingClass

FROM team_attributes """,conn)

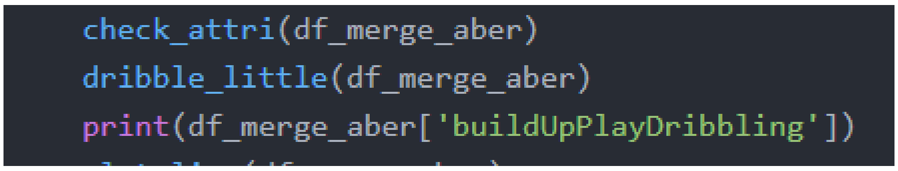
the result is three unique values(little,alots,normal).

I have defined three functions to fix the values of buildUpPlayDribbling column for each class, by calculating the average of this coumn foe each class ,then assigning the value to the row that have null in buildupplaydribbling and the specified class(little,normal,lots) in buildupplaydribblingclass for buildupplaydribbling column :

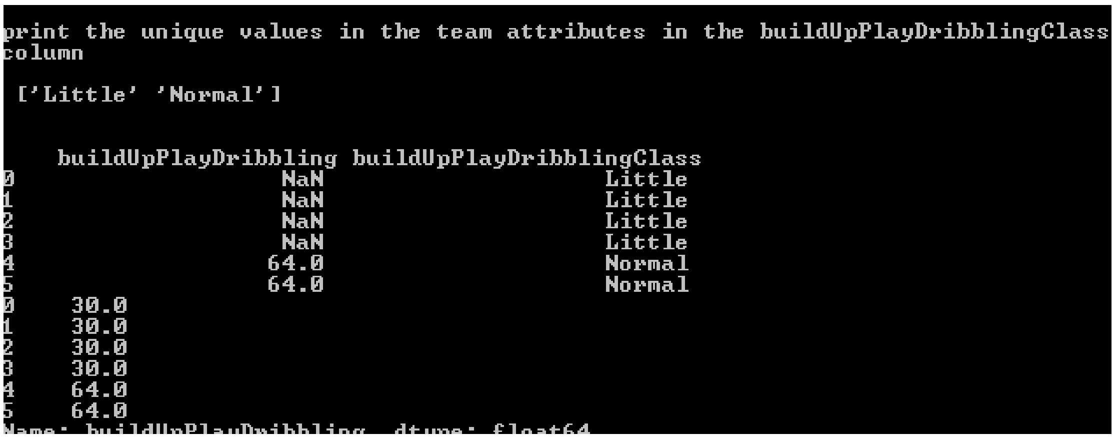
5.dribble_little(), dribble_lots(), dribble_normal()



```
###define dribble_little to fill the missing values in the buildupplaydribbling
###which has a little in the buildupplaydribblingclass column with the average
###of the other rows that have values in buildUpPlayDribbling for the little class
def dribble_little(df):
    df_dribble = pd.read_sql_query("""SELECT AVG(buildupplaydribbling) as avgdribble
    FROM Team_Attributes
    WHERE buildupplaydribblingclass == 'Little'""",conn)
###assigning the mean value (avgdribble) to each row that have little in
###buildUpPlayDribblingClass and null in buildUpPlayDribbling
    df.loc[(df.buildUpPlayDribblingClass == 'Little') & (df.buildUpPlayDribbling.isnull()),
     'buildUpPlayDribbling'] =df_dribble.loc[0,'avgdribble']
    return df
```



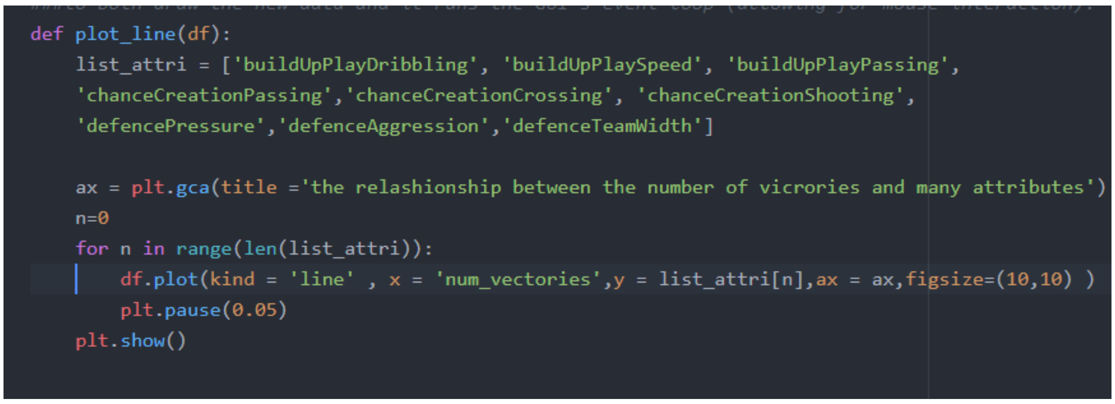
```
check_attri(df_merge_aber)
dribble_little(df_merge_aber)
print(df_merge_aber['buildUpPlayDribbling'])
```



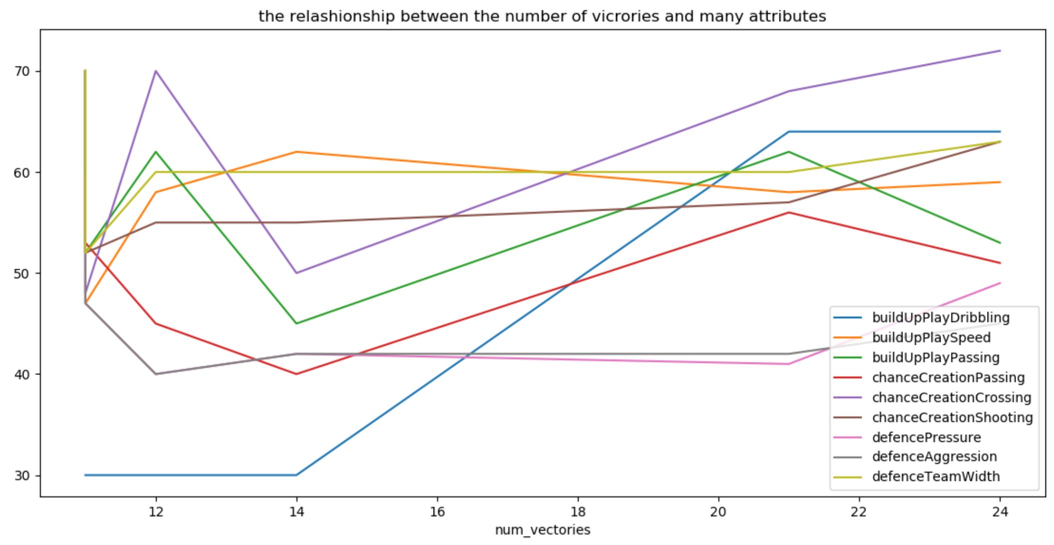
```
print the unique values in the team attributes in the buildUpPlayDribblingClass
column

 ['Little' 'Normal']


   buildUpPlayDribbling buildUpPlayDribblingClass
0                   NaN                    Little
1                   NaN                    Little
2                   NaN                    Little
3                   NaN                    Little
4                  64.0                    Normal
5                  64.0                    Normal
0    30.0
1    30.0
2    30.0
3    30.0
4    64.0
5    64.0
Name: buildUpPlayDribbling, dtype: float64
```

From the above output in the prompt, we notice that when passing the cleaned merged data frame(df_merge_aber) to the check_attri() ,there are nulls only in the buildupplaydribbling column, and from the output the nulls for the little class as shown, so to fix it , I have passed the data frame to the dribble_little function to fill the missing values with the average number of the little class.

6.the last function is plot_line():which take a data frame as an argument, plot the relationship between number victories and the other attributies which is stored in a list to iterate the plotting over its element in a for loop and Calling plt.pause(0.05)to both draw the new data and it runs the GUI's event loop (allowing for mouse interaction).



```
def plot_line(df):
    list_attri = ['buildUpPlayDribbling', 'buildUpPlaySpeed', 'buildUpPlayPassing',
    'chanceCreationPassing','chanceCreationCrossing', 'chanceCreationShooting',
    'defencePressure','defenceAggression','defenceTeamWidth']

    ax = plt.gca(title ='the relashionship between the number of vicrories and many attributes')
    n=0
    for n in range(len(list_attri)):
        df.plot(kind = 'line' , x = 'num_vectories',y = list_attri[n],ax = ax,figsize=(10,10) )
        plt.pause(0.05)
    plt.show()
```



From the visualization ,we can see the relations with the number of victories ,the buildupplaydribbling has a linear relationship,and it is fixed or increasing.

Resources:

1. https://stackoverflow.com/questions/11874767/real-time-plotting-in-while-loop-with-matplotlib

2. https://stackoverflow.com/questions/41324503/pandas-sqlite-query-using-variable

3. https://healthfully.com/read-soccer-odds-8522032.html

4. http://www.sqlitetutorial.net/sqlite-functions/

5. https://stackoverflow.com/questions/4428795/sqlite-convert-string-to-date

6. https://matplotlib.org/users/pyplot_tutorial.html

7. https://pandas.pydata.org/pandas-docs/version/0.22.0/generated/pandas.read_sql_query.html

8. http://chris.friedline.net/2015-12-15-rutgers/lessons/python2/08-working-with-sql.html

9. https://stackoverflow.com/questions/43352023/pandas-fillna-with-subset-of-rows