به نام خدا

سید پوریا احمدی 9723002

گزارش بخش اول:

ابتدا داده های مورد نظر را میخوانیم و به dataframe از کتابخانه pandas تبدیل میکنیم و علاوه بر آن کتابخانه های مورد نظر را import میکنیم:

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
from mpl_toolkits.mplot3d import Axes3D
from prettytable import PrettyTable
import scipy.io
import pandas as pd
import numpy as np
from numpy import random
import matplotlib.pyplot as plt
mat = scipy.io.loadmat('data.mat')
mat = {k:v for k, v in mat.items() if k[0] != '_'}
data = pd.DataFrame({k: pd.Series(v[0]) for k, v in mat.items()}) # compatible for both python 2.x and python 3.x
data.to_csv("example.csv")
data = pd.read_csv('example.csv')
data
```

Out[1]:

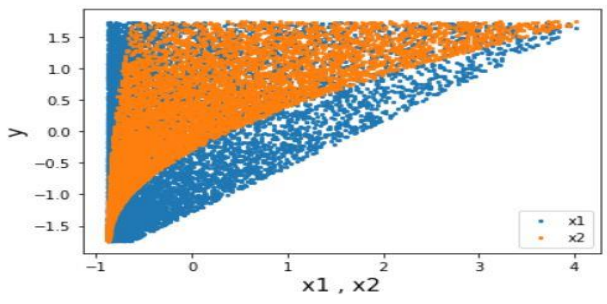
	Unnamed: 0	x2	y	x2_test	x1	y_test	x1_test
0	0	15.595565	7577.259577	19.813940	7.265000	17738.026211	10.774232
1	1	17.141869	3366.073469	16.719703	2.356963	3688.877823	2.790580
2	2	18.623714	4134.892221	16.924104	2.474309	6583.625709	5.231800
3	3	12.965940	6164.526657	4.035437	8.627121	737.624964	10.332635
4	4	12.079647	5398.716821	6.109817	8.703135	692.901679	4.052285
...	...	...	...	...	...	...	...
7995	7995	6.558751	1319.673601	NaN	7.040881	NaN	NaN
7996	7996	2.777381	329.872585	NaN	9.258345	NaN	NaN
7997	7997	0.666007	4.383710	NaN	0.522925	NaN	NaN
7998	7998	15.581607	9674.168399	NaN	9.431436	NaN	NaN
7999	7999	10.896761	4544.423692	NaN	9.009053	NaN	NaN

8000 rows × 7 columns

سپس داده ها را در scale مناسب قرار می دهیم و همچنین به عنوان نمونه داده ها را نشان میدهیم :

In [2]:
```
x1_gd = (data['x1'] - data['x1'].mean())/data['x1'].std()
x2_gd = (data['x2'] - data['x2'].mean())/data['x2'].std()
y_gd = (data['y'] - data['y'].mean())/data['y'].std()
x1_gd = x1_gd.to_numpy()
x2_gd = x2_gd.to_numpy()
y_gd = y_gd.to_numpy()
```

In [3]:
```
plt.scatter(y_gd, x1_gd, s=5, label = 'x1')
plt.scatter(y_gd, x2_gd, s=5, label = 'x2')
plt.legend(fontsize=15)
plt.xlabel('x1 , x2', fontsize=15)
plt.ylabel('y', fontsize=15)
plt.legend()
plt.show()
```



همچنین به عنوان بایاس یک ستون به x1 , x2 اضافه میکنیم :

```
In [4]: x_gd = np.c_[np.ones(x1_gd.shape[0]),x1_gd, x2_gd]
```

سپس مقادیری که لازم است مانند alpha یا همان lrate را مشخص میکنیم همچنین برای ضرایب x ها به صورت رندم یک عدد مشخص میکنیم .

```
[120]: alpha_gd = 0.0001   #learning rate
       m_gd = y_gd.size  #no. of samples
       theta_gd = np.random.rand(3)  #initializing theta with some random values
```

سپس پیاده سازی تابع gd را داریم

تابع ما داده ها یعنی x,y را میگیرد و به همرا آن learning rate یا همان alpha همچنین ضرایبی که به صورت رندوم تولید کردیم یعنی theta و m را میگیرد که تعداد رکورد های ما است

ابتدا برای اینکه بتوانیم آمار cost و theta و مقادیر predict شده را داشته باشیم یک لیست به وجود می آوریم

همچنین هزینه ابتدایی را بسیار بالا در نظر میگیریم که دلیل آن در ادامه شرح داده شده است.

حال الگوریتم را در یک while اجرا میکنیم که شرط اتمام آن این است که اختلاف دو هزینه قبل ما کمتر از 10به توان 9 باشد

ابتدا با ضرب ضرایبمان یعنی theta در داده ها یعنی x مقدار y را پیش بینی میکنیم و سپس آنرا به لیست predict اضافه میکنیم .

نوبت به محاسبه خطا میرسد که cost ما آنرا حساب میکند و سپس به لیست اضافه می شود

باید ضرایب را به روز رسانی کنیم که با استفاده از alpha ی مشخص شده آنرا انجام داده و به لیست theta مشخص می شود.

و در نهایت شرط حلقه که در بالا گفته شد چک می شود.

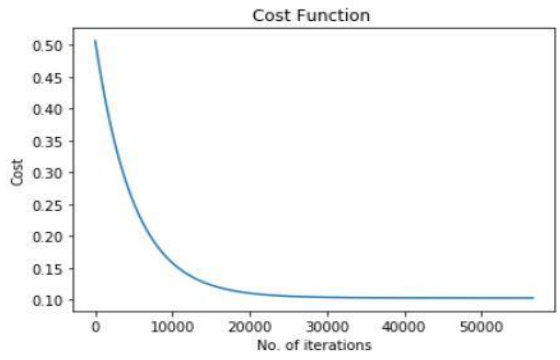
و در نهایت مقداری اولی که به cost list اضافه کردیم را حذف میکنیم که نمدار دچار اشکال نشود و لیست های theta , cost , prediction را بر می گردانیم.

```
[6]: def gradient_descent(x, y, m, theta, alpha):
         cost_list = []   #to record all cost values to this list
         theta_list = []  #to record all theta_0 and theta_1 values to this list
         prediction_list = []
         run = True
         cost_list.append(1e10)    #we append some large value to the cost list
         i=0
         while run:
             prediction = np.dot(x, theta)   #predicted y values theta_0*x0+theta_1*x1
             prediction_list.append(prediction)
             error = prediction - y
             cost = 1/(2*m) * np.dot(error.T, error)   #  (1/2m)*sum[(error)^2]
             cost_list.append(cost)
             theta = theta - (alpha * (1/m) * np.dot(x.T, error))   # alpha * (1/m) * sum[error*x]
             theta_list.append(theta)
             if cost_list[i]-cost_list[i+1] < 1e-9:   #checking if the change in cost function is less than 10^(-9)
                 run = False

             i+=1
         cost_list.pop(0)   # Remove the large number we added in the begining
         return prediction_list, cost_list, theta_list

[7]: prediction_list_gd, cost_list_gd, theta_list_gd = gradient_descent(x_gd, y_gd, m_gd, theta_gd, alpha_gd)
     theta_gd = theta_list_gd[-1]
```

و تابع را به صورت بالا فراخوانی میکنیم و آخرین element لیست theta که ضرایب نهایی هستند را نیز مشخص میکنیم.

سپس cost را plot میکنیم و در نهایت yp که y پیش بینی شده است را حساب کرده و خطای آنرا محاسبه میکنیم و چاپ میکنیم.

```
[50]: plt.title('Cost Function ')
      plt.xlabel('No. of iterations')
      plt.ylabel('Cost')
      plt.plot(cost_list_gd)
      plt.show()
```




```
[48]: yp_gd = theta_gd[0] +theta_gd[1]*x_gd[:,1] + theta_gd[2]*x_gd[:,2]

[49]: MSE_equ = ((yp_gd-y_gd)**2).mean()  #Using yp from equation of hyperplane

      print('Mean Square Error using equation of hyperplane : {}'.format(round(MSE_equ,3)))

      Mean Square Error using equation of hyperplane : 0.205
```

برای داده های تست مراحل بالا را تکرار میکنیم :

test

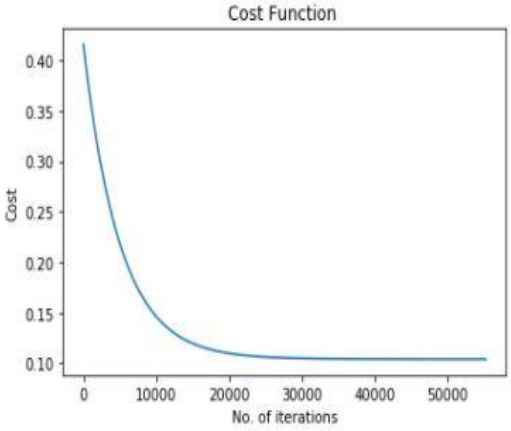
[126]:
```
x1_test_gd = (new_data_gd['x1_test'] - new_data_gd['x1_test'].mean())/new_data_gd['x1_test'].std()
x2_test_gd = (new_data_gd['x2_test'] - new_data_gd['x2_test'].mean())/new_data_gd['x2_test'].std()
y_test_gd = (new_data_gd['y_test'] - new_data_gd['y_test'].mean())/new_data_gd['y_test'].std()
x1_test_gd = x1_test_gd.to_numpy()
x2_test_gd = x2_test_gd.to_numpy()
y_test_gd = y_test_gd.to_numpy()
```

[127]:
```
x_test_gd = np.c_[np.ones(x1_test_gd.shape[0]),x1_test_gd, x2_test_gd]
```

[128]:
```
alpha_test_gd = 0.0001   #learning rate
m_test_gd = y_test_gd.size  #no. of samples

theta_test_gd = np.random.rand(3)  #initializing theta with some random values
```

[129]:
```
prediction_list_test_gd, cost_list_test_gd, theta_list_test_gd = gradient_descent(x_test_gd, y_test_gd, m_test_gd, theta_test_
theta_test_gd = theta_list_test_gd[-1]
```

[130]:
```
plt.title('Cost Function ')
plt.xlabel('No. of iterations')
plt.ylabel('Cost')
plt.plot(cost_list_test_gd)
plt.show()
```



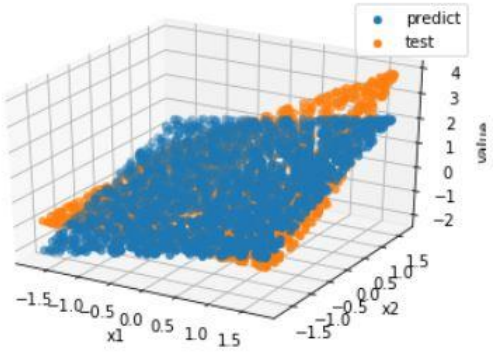
[131]:
```
yp_test_gd = theta_test_gd[0] +theta_test_gd[1]*x_test_gd[:,1] + theta_test_gd[2]*x_test_gd[:,2]
```

[132]:
```
MSE_equ_test_gd = ((yp_test_gd-y_test_gd)**2).mean()  #Using yp from equation of hyperplane
print('Mean Square Error on test data : {}'.format(round(MSE_equ_test_gd,3)))
```

```
Mean Square Error on test data : 0.208
```

و در نهایت برای داده های تست y_test و خروجی خودمان را نشان می دهیم:

[27]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

xs = x1_test_gd
ys = x2_test_gd
zs = yp_test_gd
zzs = y_test_gd

ax.scatter(xs, ys, zs ,  label = "predict")
ax.scatter(xs, ys, zzs,label = "test")
ax.legend()
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('value')

plt.show()
```



حال به قسمت sgd می رسیم :

ابتدا داده های train و تست را آماده می کنیم:

```
In [28]: train = (data[['x1', 'x2']] - data[['x1', 'x2']].mean())/data[['x1', 'x2']].std()
         train = train.to_numpy()

         y = (data['y'] - data['y'].mean())/data['y'].std()
         y = y.to_numpy()
         train_data=pd.DataFrame(train)
         train_data['y']=y
```

```
In [29]: new_data = data.iloc [:2000 , :]

         x_test = (new_data[['x1_test', 'x2_test']] - new_data[['x1_test', 'x2_test']].mean())/new_data[['x1_test', 'x2_test']].std()
         x_test = x_test.to_numpy()

         y_test = (new_data['y_test'] - new_data['y_test'].mean())/new_data['y_test'].std()
         y_test = y_test.to_numpy()

         x_test=np.array(x_test)
         y_test=np.array(y_test)
         print(x_test[: , 1])
```

```
[ 1.34458712  0.86286158  0.89468365 ... -0.88427743  0.26772321
  1.36319991]
```

همچنین داده ها را به scale مناسب می بریم زمان و هزینه الگوریتم کاهش یابد.

حال به الگوریتم می رسیم ، در این قسمت الگوریتم SGD را پیاده سازی کرده ایم اما به دلیل فضایی که اشغال می کرد عین قبل داده ها را در لیست ها ذخیره نکردیم و تنها وزن ها و بایاس مناسب را return میکنیم و بعد به تابع predict می دهیم تا با گرفتن داده های تست y_test را پیش بینی کند.

در هر iteration ، k نمونه از داده train را انتخاب میکنیم.

```
[34]: def SGD(train_data,learning_rate,n_iter,k,divideby):
          w=np.zeros(shape=(1,train_data.shape[1]-1))
          b=0
          cur_iter=1
          while(cur_iter<=n_iter):
              temp=train_data.sample(k)
              y=np.array(temp['y'])
              x=np.array(temp.drop('y',axis=1))
              w_gradient=np.zeros(shape=(1,train_data.shape[1]-1))
              b_gradient=0
              for i in range(k):
                  prediction=np.dot(w,x[i])+b
                  w_gradient=w_gradient+(-2)*x[i]*(y[i]-(prediction))
                  b_gradient=b_gradient+(-2)*(y[i]-(prediction))
              w=w-learning_rate*(w_gradient/k)
              b=b-learning_rate*(b_gradient/k)
              cur_iter=cur_iter+1
              learning_rate=learning_rate/divideby
          return w,b
```

```
[35]: def predict(x,w,b):
          y_pred=[]
          for i in range(len(x)):
              y=np.asscalar(np.dot(w,x[i])+b)
              y_pred.append(y)
          return np.array(y_pred)
```

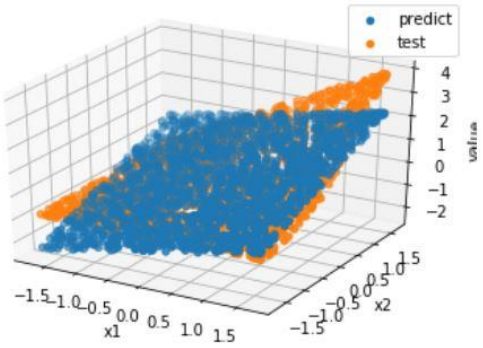
در نهایت تابع را فراخوانی کرده و با داده های train ، w,b را آموزش داده و پیدا میکنیم و با x_test با تابع predict ارسال کرده و y_test_predict را محاسبه میکنم و MSE را نیز نمایش می دهیم

```
w,b=SGD(train_data,learning_rate=1,n_iter=100,divideby=2,k=10)
y_pred_sgd=predict(x_test,w,b)


MSE_equ_test = ((y_pred_sgd-y_test)**2).mean()  #Using yp from equation of hyperplane
print('Mean Square on test data : {}'.format(round(MSE_equ_test,3)))
```

```
Mean Square on test data : 0.223
```

همچنین در نهایت y_test و y_test پیش بینی شده را در یک نمودار سه بعدی نمایش می دهیم:

```
[37]: fig = plt.figure()
      ax = fig.add_subplot(111, projection='3d')


      xs = x_test[: , 0]
      ys = x_test[: , 1]
      zs = y_pred_sgd
      zzs = y_test
      ax.scatter(xs, ys, zs ,  label = "predict")
      ax.scatter(xs, ys, zzs,label = "test")
      ax.legend()
      ax.set_xlabel('x1')
      ax.set_ylabel('x2')
      ax.set_zlabel('value')

      plt.show()
```



طبق خواسته سوال مقدار تابع خطا را گزارش میکنیم:

gd

```
In [54]: print('Mean Square Error  on train data : {}'.format(round(MSE_equ_gd,3)))
         print('Mean Square Error  on test data : {}'.format(round(MSE_equ_test_gd,3)))
```

```
Mean Square Error  on train data : 0.205
Mean Square Error  on test data : 0.208
```

sgd

```
In [55]: y_pred_tarin_sgd=predict(train,w,b)
         MSE_equ_train = ((y_pred_tarin_sgd-y)**2).mean()
         print('Mean Square Error on train data : {}'.format(round(MSE_equ_train,3)))
         print('Mean Square Error on test data : {}'.format(round(MSE_equ_test,3)))
```

```
Mean Square Error on train data : 0.219
Mean Square Error on test data : 0.221
```

مقایسه دو الگوریتم:

به طور کلی میدانیم که sgd بسیار سریع تر از gd همگرا می شود اما مقدار تابع خطای آن معمولا بالا تر از gd می باشد

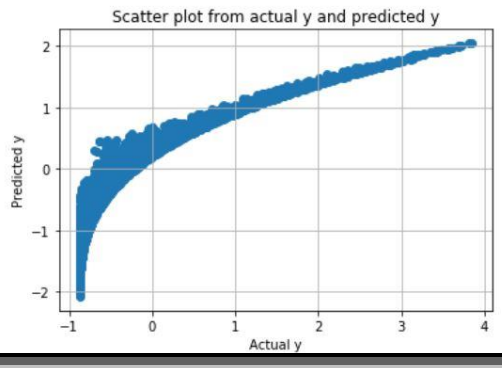
همانطور که ر کد هم مشخص است ما با 100 iteration به جواب خوبی با الگوریتم sgd رسیدیم که این تعداد iteration در مقایسه با الگوریتم gd اصلا قابل مقایسه نیست و به طرز قابل توجهی کمتر می باشد

مقایسه جواب های بدست آمده:

مقایسه مقادیر پیش بینی شده و تست در gd

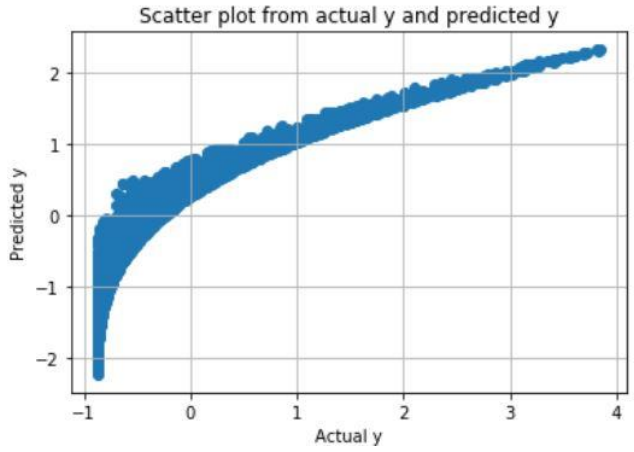
diffrence between actual y_test and y_test prediction in GD ¶

```
52]: plt.scatter(y_test_gd,yp_test_gd)
     plt.grid()
     plt.xlabel('Actual y')
     plt.ylabel('Predicted y')
     plt.title('Scatter plot from actual y and predicted y')
     plt.show()
```



مقایسه مقادیر پیش بینی شده و تست در sgd

```
plt.scatter(y_test,y_pred_sgd)
plt.grid()
plt.xlabel('Actual y')
plt.ylabel('Predicted y')
plt.title('Scatter plot from actual y and predicted y')
plt.show()
```



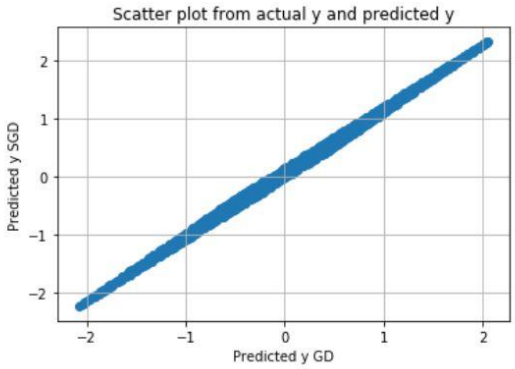
مقایسه مقادیر پیش بینی شده ggd و sgd

البته قابل ذکر چون داده ها scale شده اند تفاوت خیلی در نمودار مشخص نمی باشد همچنین تفاوت آنها در یک dataframe نیز در کد آورده شده.

به تمام سوالات بخش اول جواب داده شد.

diffrence between y_test prediction in GD and y_test prediction in SGD

```
plt.scatter(yp_test_gd,y_pred_sgd)
plt.grid()
plt.xlabel('Predicted y GD')
plt.ylabel('Predicted y SGD')
plt.title('Scatter plot from actual y and predicted y')
plt.show()
```



گزارش سوال دوم :

ابتدا کتابخانه pandas را برای خوانده فایل داده import میکنیم و با استفاده از read_csv داده ها را میخوانیم.

در سوال اول خواسته شده که بازیکنان ابتدایی و انتهایی را نمایش دهیم که به دو شکل این کار انجام شده است.

```
3]: import pandas as pd

4]: players =pd.read_csv('players.csv')
```

حال بازیکنان ابتدایی و انتهایی را بدون استفاده از متد های کتابخانه نمایش می دهیم ، این کار را با استفاده از

[ ]Iloc انجام می دهیم باید توجه داشت که [: , 10:]iloc ده داده ابتدایی با تمام ستون های آنرا نمایش می دهد :

without library

```
]: ## first 10 players
   players.iloc[:10 , :]
```

]:

	ID	Name	FullName	Age	Height	Weight	PhotoUrl	Nationality	Overall	Potential	...	LMRating	CMRating
0	158023	L. Messi	Lionel Messi	33	170	72	https://cdn.sofifa.com/players/158/023/21_60.png	Argentina	93	93	...	93	90
1	20801	Cristiano Ronaldo	C. Ronaldo dos Santos Aveiro	35	187	83	https://cdn.sofifa.com/players/020/801/21_60.png	Portugal	92	92	...	91	84
2	200389	J. Oblak	Jan Oblak	27	188	87	https://cdn.sofifa.com/players/200/389/21_60.png	Slovenia	91	93	...	38	41
3	192985	K. De Bruyne	Kevin De Bruyne	29	181	70	https://cdn.sofifa.com/players/192/985/21_60.png	Belgium	91	91	...	91	91
4	190871	Neymar Jr	Neymar da Silva Santos Jr.	28	175	68	https://cdn.sofifa.com/players/190/871/21_60.png	Brazil	91	91	...	91	86
5	188545	R. Lewandowski	Robert Lewandowski	31	184	80	https://cdn.sofifa.com/players/188/545/21_60.png	Poland	91	91	...	86	83
6	208722	S. Mané	Sadio Mané	28	175	69	https://cdn.sofifa.com/players/208/722/21_60.png	Senegal	90	90	...	90	84
7	192448	M. ter Stegen	Marc-André ter Stegen	28	187	85	https://cdn.sofifa.com/players/192/448/21_60.png	Germany	90	93	...	42	48
8	203376	V. van Dijk	Virgil van Dijk	28	193	92	https://cdn.sofifa.com/players/203/376/21_60.png	Netherlands	90	91	...	73	79
9	167495	M. Neuer	Manuel Neuer	34	193	92	https://cdn.sofifa.com/players/167/495/21_60.png	Germany	90	90	...	47	53

10 rows × 90 columns

```
In [9]: ## last ten players
        players.iloc[-10: , :]
```

Out[9]:

	ID	Name	FullName	Age	Height	Weight	PhotoUrl	Nationality	Overall	Potential	...	LMRating	CMRating
19010	259381	W. Delvin	Willads Delvin	19	165	55	https://cdn.sofifa.com/players/259/381/21_60.png	Denmark	49	62	...	50	44
19011	248379	S. Doherty	Stephen Doherty	20	176	77	https://cdn.sofifa.com/players/248/379/21_60.png	Republic of Ireland	49	59	...	51	48
19012	255547	Cha Oh Yeon	Oh Yeon Cha	22	186	77	https://cdn.sofifa.com/players/255/547/21_60.png	Korea Republic	49	60	...	49	51
19013	255549	Kang Sang Hee	Sang Hee Kang	22	180	73	https://cdn.sofifa.com/players/255/549/21_60.png	Korea Republic	49	61	...	36	35
19014	257116	C. Diabate	Cheick Diabate	19	192	78	https://cdn.sofifa.com/players/257/116/21_60.png	England	49	60	...	42	41
19015	257371	M. Nzongong	Mike Nzongong	19	179	74	https://cdn.sofifa.com/players/257/371/21_60.png	Republic of Ireland	49	60	...	51	47
19016	259160	L. Bell	Lewis Bell	17	181	70	https://cdn.sofifa.com/players/259/160/21_60.png	England	49	67	...	51	44
19017	259157	Y. Arai	Yasin Arai	16	176	70	https://cdn.sofifa.com/players/259/157/21_60.png	England	49	74	...	53	47
19018	253763	R. Dinanga	Ricardo Dinanga	18	174	73	https://cdn.sofifa.com/players/253/763/21_60.png	Republic of Ireland	49	59	...	49	43
19019	241493	S. Cartwright	Samuel Cartwright	19	185	75	https://cdn.sofifa.com/players/241/493/21_60.png	England	49	65	...	38	36

10 rows × 90 columns

همچنین با استفاده از کتابخانه می توان به سادگی داده های ابتدایی و انتهایی را نمایش داد:

```
In [5]: players.head()
```

Out[5]:

	ID	Name	FullName	Age	Height	Weight	PhotoUrl	Nationality	Overall	Potential	...	LMRating	CMRating	RMRa
0	158023	L. Messi	Lionel Messi	33	170	72	https://cdn.sofifa.com/players/158/023/21_60.png	Argentina	93	93	...	93	90	
1	20801	Cristiano Ronaldo	C. Ronaldo dos Santos Aveiro	35	187	83	https://cdn.sofifa.com/players/020/801/21_60.png	Portugal	92	92	...	91	84	
2	200389	J. Oblak	Jan Oblak	27	188	87	https://cdn.sofifa.com/players/200/389/21_60.png	Slovenia	91	93	...	38	41	
3	192985	K. De Bruyne	Kevin De Bruyne	29	181	70	https://cdn.sofifa.com/players/192/985/21_60.png	Belgium	91	91	...	91	91	
4	190871	Neymar Jr	Neymar da Silva Santos Jr.	28	175	68	https://cdn.sofifa.com/players/190/871/21_60.png	Brazil	91	91	...	91	86	

5 rows × 90 columns

```
In [6]: players.tail()
```

Out[6]:

	ID	Name	FullName	Age	Height	Weight	PhotoUrl	Nationality	Overall	Potential	...	LMRating	CMRating
19015	257371	M. Nzongong	Mike Nzongong	19	179	74	https://cdn.sofifa.com/players/257/371/21_60.png	Republic of Ireland	49	60	...	51	47
19016	259160	L. Bell	Lewis Bell	17	181	70	https://cdn.sofifa.com/players/259/160/21_60.png	England	49	67	...	51	44
19017	259157	Y. Arai	Yasin Arai	16	176	70	https://cdn.sofifa.com/players/259/157/21_60.png	England	49	74	...	53	47
19018	253763	R. Dinanga	Ricardo Dinanga	18	174	73	https://cdn.sofifa.com/players/253/763/21_60.png	Republic of Ireland	49	59	...	49	43
19019	241493	S. Cartwright	Samuel Cartwright	19	185	75	https://cdn.sofifa.com/players/241/493/21_60.png	England	49	65	...	38	36

5 rows × 90 columns

حال به جواب سوال **دوم** میپردازیم:

برای missing value ها باید cell هایی را null هستند را باید پیدا کنیمو با استفاده از ()isnull می توان اینکار را انجام داد

روی کل dataframe که داریم این کار را انجام داده و جمع تعداد خانه هایی که در هر ستون خالی هستند را ذخیر میکنیم و سپس نمایش می دهیم:

missing values

```
In [14]: count_missing_value = players.isnull().sum()

In [15]: for i in range(players.shape[1]):
             if count_missing_value[i]>0:
                 print ("{} column has {} missing valus".format(list(players.columns)[i],count_missing_value[i]))

ClubPosition column has 234 missing valus
ContractUntil column has 234 missing valus
ClubNumber column has 234 missing valus
NationalPosition column has 17895 missing valus
NationalNumber column has 17895 missing valus
```

سوال سه:

باید میانگین و حداقل و حداکثر وزن بازیکنان را بدست آوریم ، برای اینکار از دو روش استفاده شده، اول با استفاده از متد های کتابخانه و دیگری پیاده سازی شخصی:

weight min max avg

```
In [9]: players['Weight'].describe()
```

```
Out[9]: count    19020.000000
        mean        75.052419
        std          7.058038
        min         50.000000
        25%         70.000000
        50%         75.000000
        75%         80.000000
        max        110.000000
        Name: Weight, dtype: float64
```

و پیاده سازی شخصی :

در این روش ابتدا در یک حلقه کل رکورد ها را خوانده و مقادیر W_min و W_max را update میکنیم و جمع وزن ها را نیز حساب میکنیم و در نهایت مقادیر خواسته شده را پرینت میکنیم

```
In [11]: W_min = 0
         W_max = 0
         W_sum = 0
         for i  in range (players.shape[0]):
             if i ==0:
                 W_min = players.iloc[i]['Weight']
                 W_max = W_min
                 W_sum += W_min
             else:
                 weightt = players.iloc[i]['Weight']
                 if W_min > weightt:
                     W_min = weightt
                 elif W_max < weightt :
                     W_max = weightt
                 W_sum += weightt

         print("MInimum weight is {}".format(W_min))
         print("Maximum weight is {}".format(W_max))
         print("average weight is {}".format(W_sum/players.shape[0]))
```

```
MInimum weight is 50
Maximum weight is 110
average weight is 75.05241850683491
```

سوال چهار:

دوباره از دو روش این سوال حل شده ابتدا جواب را با استفاده از توابع کتابخانه نشان می دهیم:

most and least players

```
In [12]: players['Nationality'].value_counts().head(1)

Out[12]: England    1706
         Name: Nationality, dtype: int64

In [13]: players['Nationality'].value_counts().tail(23)

Out[13]: Tanzania               1
         Rwanda                 1
         Papua New Guinea       1
         Saint Lucia            1
         New Caledonia          1
         Singapore              1
         Guam                   1
         Barbados               1
         Suriname               1
         Indonesia              1
         Malaysia               1
         Guatemala              1
         Puerto Rico            1
         Hong Kong              1
         Bermuda                1
         Malta                  1
         Korea DPR              1
         Andorra                1
         Aruba                  1
         Malawi                 1
         São Tomé & Príncipe    1
         Chad                   1
         South Sudan            1
         Name: Nationality, dtype: int64
```

حال روش دوم را توضیح می دهیم:

ابتدا تمام کشور ها را در یک لیست ذخیره میکنیم و تمام کشور های جمع شده را در یک دیکشنری به عنوان key ذخیره می کنیم که value آنها تعداد بازیکنان این کشور ها است سپس با تعداد بازیکنان را با خواندن ستون Nationality آپدیت میکنیم و مقدار های minو max ، value های این دیکشنری را بدست می آوریم و در نهایت در یک حلقه کشور هایی که دارای بیشترین و کمتری تعداد بازیکن هستند را در لیست مربوطه append میکنیم و نشان می دهیم.

```
[14]: ## second approach

[15]: nationality_dict = {}
      zz= players['Nationality'].unique()
      for i in range (len(zz)):
            nationality_dict[zz[i]] = 0
      for i  in range (players.shape[0]):
           nationality_dict[players.iloc[i]['Nationality']] +=1

[16]:
      all_values = nationality_dict.values()
      max_value = max(all_values)
      min_value = min(all_values)
      min_list = []
      max_list = []
      for i in range(len(nationality_dict)):
          if nationality_dict[zz[i]] == min_value:
              min_list.append(zz[i])
          elif nationality_dict[zz[i]] == max_value:
              max_list.append(zz[i])

      print("Maximum numbers of players is for {} with {} players".format(max_list,max_value))

      print("Minimum numbers of players is for {} with {} players".format(min_list,min_value))

      Maximum numbers of players is for ['England'] with 1706 players
      Minimum numbers of players is for ['Tanzania', 'Chad', 'Bermuda', 'Puerto Rico', 'New Caledonia', 'São Tomé & Príncipe', 'Malaw
      i', 'Aruba', 'Suriname', 'Guam', 'Rwanda', 'Andorra', 'Papua New Guinea', 'Saint Lucia', 'Korea DPR', 'Malta', 'Guatemala', 'Ba
      rbados', 'South Sudan', 'Singapore', 'Hong Kong', 'Indonesia', 'Malaysia'] with 1 players
```

سوال پنج :

در این سوال به سادگی شروط سوال را روی dataframe اعمال میکنیم و player_new ، dataframe جدید ما با شرایط سوال است و بعد روی آن یک حلقه زده و نام بازیکنان را نمایش می دهیم:

Growth < 3 and potential < 84

```
38]: players_list = []
player_new1=players[players['Growth']<3]
player_new=player_new1[player_new1['Potential']<84]
for i  in range (player_new.shape[0]):
    players_list.append(player_new.iloc[i]['Name'])

players_list
```

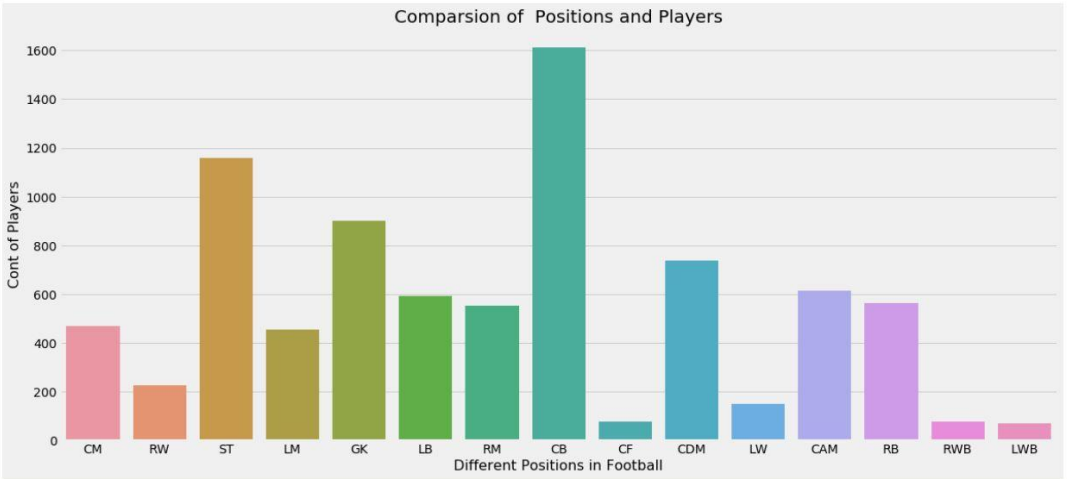
```
38]: ['Renato Augusto',
 'C. Vela',
 'D. Zapata',
 'Rafa',
 'M. Dúbravka',
 'M. Acuña',
 'Ronaldo Cabrais',
 'João Moutinho',
 'J. Vertonghen',
 'D. Tadić',
 'C. Aránguiz',
 'W. Zaha',
 'I. Gueye',
 'Aitor',
 'K. Manolas',
 'L. Hrádecký',
 'Douglas Costa',
 'Muniain',
 'Coutinho',
 'K. Trapp'
```

سوال 6 :

برای این سوال بر روی dataframe، player_new بازیکنان را بر اساس best position آنها رو چارت میبریم.

chart

```
1 [41]: plt.figure(figsize = (18,8))
ax = sns.countplot('BestPosition', data =player_new)
ax.set_xlabel(xlabel = 'Different Positions in Football', fontsize = 16)
ax.set_ylabel(ylabel = 'Cont of Players', fontsize = 16)
ax.set_title(label = 'Comparsion of  Positions and Players', fontsize = 20)
plt.show()
```



سوال 7 :

برای این سوال بنده بازیکنان آینده دار را به این صورت تعریف کردم که پتانسیل بالا 85 و سن کوچکتر از 23 داشته باشند

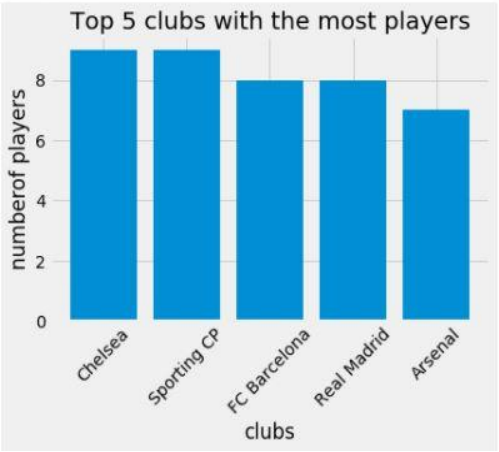
برای این منظور ابتدا نام باشگاه ها در یک دیکشنری به عنوان key قرار داده شد و value های آن تعداد بازیکنان آینده دار آن خواهند بود و سپس روی کل dataframe دو شرط گفته شده اعمال شد و اگر بازیکن شرایط را داشت یک عدد به value باشگاهش اضافه می شود سپس بر اساس value دیکشنری را sort میکنیم در این حالت باشگاه با بیشترین تعداد بازیکن آینده دار در اول دیکشنری قرار دارد ، سپس روی یک چارت آن ها را نمایش داده ایم

players with future!!!

I define it with potential more than 85 and age under 23

In [48]:
```
import numpy as np
import matplotlib.pyplot as plt
#clubs
clubs = players['Club'].unique()
good_future_dict = {}
for club in clubs:
    good_future_dict[club] = 0
for i in range (players.shape[0]):
    if players.iloc[i]['Potential']>=85 and players.iloc[i]['Age']<=23:
        good_future_dict[players.iloc[i]['Club']] += 1
good_future_dict = sorted(good_future_dict.items(), key=lambda x: x[1], reverse=True)
height = [good_future_dict[0][1], good_future_dict[1][1], good_future_dict[2][1], good_future_dict[3][1], good_future_dict[4][1]]
bars = (good_future_dict[0][0], good_future_dict[1][0], good_future_dict[2][0], good_future_dict[3][0], good_future_dict[4][0])
# Create bars
plt.bar(bars, height)
plt.xticks(rotation=45)
# Add title and axis names
plt.title('Top 5 clubs with the most players')
plt.xlabel('clubs')
plt.ylabel('numberof players')
```

Out[48]: Text(0, 0.5, 'numberof players')



سوال 8 :

برای این سوال دو روش استفاده شده اولین روش مانند سوال growth و potential عمل شده که در شکل زیر قابل مشاهده است

In [56]:
```
## second approach
players_until2021= []
player_2021=players[players['ContractUntil'] == 2021]
player_noTeam2021=player_2021[player_2021['NationalTeam'] == 'Not in team']

for i  in range (player_noTeam2021.shape[0]):
    players_until2021.append(player_noTeam2021.iloc[i]['Name'])

print(players_until2021  )
```

```
['L. Modrić', 'Fernandinho', 'Thiago Silva', 'J. Boateng', 'E. Cavani', 'Z. Ibrahimović', 'Jesús Navas', 'Renato Augusto', 'D
ouglas Costa', 'Juan Bernat', 'E. Višća', 'F. Thauvin', 'G. Higuaín', 'M. Perin', 'A. Areola', 'F. Muslera', 'B. Matuidi',
'G. Buffon', 'M. Politano', 'Angeliño', 'Jonathan Viera', 'Taison', 'Marlos', 'A. Kolarov', 'S. Bender', 'L. Bender', 'M. Özi
l', 'M. Hamšík', 'Felipe Anderson', 'S. Arias', 'L. Torreira', 'A. Gignac', 'F. Ribéry', 'Iniesta', 'M. Dmitrović', 'N. Nkoul
ou', 'L. Stindl', 'David Luiz', 'S. Khedira', 'Diego Costa', 'Nacho Monreal', 'Javi Martínez', 'A. Robben', 'Lucas Vázquez',
'Otávio', 'D. Rugani', 'João Mário', 'Raúl García', 'Joaquín', 'F. Quagliarella', 'M. Dembélé', 'S. Ruffier', 'Miranda', 'Gua
ita', 'Escudero', 'N. Lodeiro', 'Nacho Fernández', 'C. Bakambu', 'Deulofeu', 'S. Romero', 'N. Guzmán', 'Juan Mata', 'Éder',
'Ángel', 'M. Valbuena', 'N. Amrabat', 'V. Ćorluka', 'Nani', 'A. Mirante', 'D. Valeri', 'L. Piszczek', 'Pedro León', 'M. Día
z', 'Morales', 'T. Bakayoko', 'Júnior Moraes', 'Vinícius', 'M. Parolo', 'Mata', 'A. Mena', 'Dani Ceballos', 'R. Saïss', 'M. M
arega', 'M. Antonio', 'B. Mee', 'Jaume Costa', 'A. Mandi', 'E. Hysaj', 'S. Lulić', 'M. Debuchy', 'D. Mbokani', 'K. Boateng',
'C. Ansaldi', 'A. Candreva', 'V. Moses', 'Sokratis', 'Fábio Coentrão', 'Gervinho', 'R. Bertrand', 'M. Škrtel', 'Ş. Radu', 'K.
Gameiro', 'De Marcos', 'P. Diop', 'Moyá', 'L. López', 'C. Tévez', 'M. Musacchio', 'L. Montes', 'P. Piatti', 'B. Ivanović',
'M. Hitz', 'Emerson', 'A. Lafont', 'N. De la Cruz', 'R. Barkley', 'J. Svensson', 'Sérgio Oliveira', 'A. Townsend', 'Y. El Ara
bi', 'Roberto Torres', 'M. De Sciglio', 'Leo Baptistao', 'K. Lala', 'L. Karius', 'R. Borré', 'Jony', 'J. Amavi', 'Maicon',
'S. Mustafi', 'Herrerín', 'I. Marcone', 'D. Zappacosta', 'Marcelo Goiano', 'M. Silvestri', 'O. Al Soma', 'Fábio Martins', 'L.
Olaza', 'J. Lingard', 'M. Pereyra', 'J. Cork', 'Pedro Mendes', 'E. Rigoni', 'S. Benrahma', 'M. Guendouzi', 'Oier', 'C. Basha
m', 'Rony Lopes', 'Víctor Ruiz', 'T. Heaton', 'Beñat', 'Rochina', 'Y. Belhanda', 'M. Guendouzi', 'L. Biglia', 'W. Cyprien',
'Marcelo', 'F. Vázquez', 'G. Cahill', 'Miguel Veloso', 'Pablo', 'A. Young', 'G. Kakuta', 'P. van Aanholt', 'D. Da Silva', 'So
ldado', 'D. Rose', 'Pedro Henrique', 'Hilton', 'N. Maksimović', 'Kike García', 'Borja Valero', 'R. Loftus-Cheek', 'L. Unnerst
all', 'Y. Chará', 'Dyego Sousa', 'C. Kabasele', 'O. Igbalo', 'Bruno Peres', 'M. Livaja', 'W. Rooney', 'Jorge', 'N. Gaitán',
```

در روش بعدی یک dataframe جدید نساختی و با اعمال شرط ها صرفا اسم ها را به لیستی که داشتیم اضافه کردیم:

In [55]:
```
players_until2021_noteam= []

for i  in range (players.shape[0]):
    if players.iloc[i]['ContractUntil'] == 2021 and  players.iloc[i]['NationalTeam'] == 'Not in team' :
            players_until2021_noteam.append(players.iloc[i]['Name'])
print("players are{}".format(players_until2021_noteam))
```

```
players are['L. Modrić', 'Fernandinho', 'Thiago Silva', 'J. Boateng', 'E. Cavani', 'Z. Ibrahimović', 'Jesús Navas', 'Renato A
ugusto', 'Douglas Costa', 'Juan Bernat', 'E. Višća', 'F. Thauvin', 'G. Higuaín', 'M. Perin', 'A. Areola', 'F. Muslera', 'B. M
atuidi', 'G. Buffon', 'M. Politano', 'Angeliño', 'Jonathan Viera', 'Taison', 'Marlos', 'A. Kolarov', 'S. Bender', 'L. Bende
r', 'M. Özil', 'M. Hamšík', 'Felipe Anderson', 'S. Arias', 'L. Torreira', 'A. Gignac', 'F. Ribéry', 'Iniesta', 'M. Dmitrovi
ć', 'N. Nkoulou', 'L. Stindl', 'David Luiz', 'S. Khedira', 'Diego Costa', 'Nacho Monreal', 'Javi Martínez', 'A. Robben', 'Luc
as Vázquez', 'Otávio', 'D. Rugani', 'João Mário', 'Raúl García', 'Joaquín', 'F. Quagliarella', 'M. Dembélé', 'S. Ruffier', 'M
iranda', 'Guaita', 'Escudero', 'N. Lodeiro', 'Nacho Fernández', 'C. Bakambu', 'Deulofeu', 'S. Romero', 'N. Guzmán', 'Juan Mat
a', 'Éder', 'Ángel', 'M. Valbuena', 'N. Amrabat', 'V. Ćorluka', 'Nani', 'A. Mirante', 'D. Valeri', 'L. Piszczek', 'Pedro Leó
n', 'M. Díaz', 'Morales', 'T. Bakayoko', 'Júnior Moraes', 'Vinícius', 'M. Parolo', 'Mata', 'A. Mena', 'Dani Ceballos', 'R. Sa
ïss', 'M. Marega', 'M. Antonio', 'B. Mee', 'Jaume Costa', 'A. Mandi', 'E. Hysaj', 'S. Lulić', 'M. Debuchy', 'D. Mbokani', 'K.
Boateng', 'C. Ansaldi', 'A. Candreva', 'V. Moses', 'Sokratis', 'Fábio Coentrão', 'Gervinho', 'R. Bertrand', 'M. Škrtel', 'Ş.
Radu', 'K. Gameiro', 'De Marcos', 'P. Diop', 'Moyá', 'L. López', 'C. Tévez', 'M. Musacchio', 'L. Montes', 'P. Piatti', 'B. Iv
anović', 'M. Hitz', 'Emerson', 'A. Lafont', 'N. De la Cruz', 'R. Barkley', 'J. Svensson', 'Sérgio Oliveira', 'A. Townsend',
'Y. El Arabi', 'Roberto Torres', 'M. De Sciglio', 'Leo Baptistao', 'K. Lala', 'L. Karius', 'R. Borré', 'Jony', 'J. Amavi', 'M
aicon', 'S. Mustafi', 'Herrerín', 'I. Marcone', 'D. Zappacosta', 'Marcelo Goiano', 'M. Silvestri', 'O. Al Soma', 'Fábio Marti
ns', 'L. Olaza', 'J. Lingard', 'M. Pereyra', 'J. Cork', 'Pedro Mendes', 'E. Rigoni', 'S. Benrahma', 'M. Guendouzi', 'Oier',
'C. Basham', 'Rony Lopes', 'Víctor Ruiz', 'T. Heaton', 'Beñat', 'Rochina', 'Y. Belhanda', 'M. Guendouzi', 'L. Biglia', 'W. Cy
prien', 'Marcelo', 'F. Vázquez', 'G. Cahill', 'Miguel Veloso', 'Pablo', 'A. Young', 'G. Kakuta', 'P. van Aanholt', 'D. Da Sil
va', 'Soldado', 'D. Rose', 'Pedro Henrique', 'Hilton', 'N. Maksimović', 'Kike García', 'Borja Valero', 'R. Loftus-Cheek', 'L.
Unnerstall', 'Y. Chará', 'Dyego Sousa', 'C. Kabasele', 'O. Igbalo', 'Bruno Peres', 'M. Livaja', 'W. Rooney', 'Jorge', 'N. Gai
```

سوال 9 :

در این سوال ابتدا یک دیکشنری ساختیم که key آن نام بازیکن و value آن ارزش آن باشد

ابتدا در یک حلقه شرط اینکه بازیکن زیر 24 و از باشگاه چلسی باشد را چک میکنیم و در آن صورت مقادیر لازم را وارد دیکشنری می کنیم و در نهایت آن را نشان می دهیم:

value of chelsea players who are under 24

```
n [57]: chelsea_under24 = {}
        for i  in range (players.shape[0]):
            if  players.iloc[i]['Club'] == 'Chelsea' and  players.iloc[i]['Age'] < 24 :
                chelsea_under24[players.iloc[i]['Name']] = players.iloc[i]['ValueEUR']
        print("chelsea players under 24 with their value {}".format(chelsea_under24))
```

```
chelsea players under 24 with their value {'K. Havertz': 121000000, 'B. Chilwell': 34500000, 'C. Pulisic': 41500000, 'M. Moun
t': 43000000, 'R. James': 29500000, 'T. Abraham': 29000000, 'F. Tomori': 15500000, 'C. Hudson-Odoi': 10000000, 'C. Musonda': 48
00000, 'B. Gilmour': 4400000, 'D. Sterling': 2300000, 'J. Wakely': 500000}
```

سوال 10 :

در این سوال ابتدا بازیکن را پیدا کرده و مقادیر خواسته شده را پیدا میکنیم و نمایش می دهیم:

Eden Hazard club salary position

```
32]: for i in range(players.shape[0]):
         if players.iloc[i]['Name'] == 'E. Hazard':
             club = players.iloc[i]['Club']
             pos = players.iloc[i]['Positions']
             wage = players.iloc[i]['WageEUR']
     print("Eden Hazard is playing in {} club with {} wage and {} positions".format(club,wage,pos))
```

```
Eden Hazard is playing in Real Madrid club with 350000 wage and LW positions
```