This is The code for Data Preprocessing. The Questions follow after this

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
```

In [3]:

```
# reading the dataset
football_data=pd.read_csv(r'F:\MS IIITH\course structure and syllabus\Semester I\Data Analytics\Project\football_data
football_data_original_set=football_data
```

Exploring the dimentions of the data

In [4]:

```
football_data.shape
```

Out[4]:

```
(18207, 89)
```

In [5]:

```
football_data.columns
```

Out[5]:

```
Index(['Unnamed: 0', 'ID', 'Name', 'Age', 'Photo', 'Nationality', 'Flag',
       'Overall', 'Potential', 'Club', 'Club Logo', 'Value', 'Wage', 'Special',
       'Preferred Foot', 'International Reputation', 'Weak Foot',
       'Skill Moves', 'Work Rate', 'Body Type', 'Real Face', 'Position',
       'Jersey Number', 'Joined', 'Loaned From', 'Contract Valid Until',
       'Height', 'Weight', 'LS', 'ST', 'RS', 'LW', 'LF', 'CF', 'RF', 'RW',
       'LAM', 'CAM', 'RAM', 'LM', 'LCM', 'CM', 'RCM', 'RM', 'LWB', 'LDM',
       'CDM', 'RDM', 'RWB', 'LB', 'LCB', 'CB', 'RCB', 'RB', 'Crossing',
       'Finishing', 'HeadingAccuracy', 'ShortPassing', 'Volleys', 'Dribbling',
       'Curve', 'FKAccuracy', 'LongPassing', 'BallControl', 'Acceleration',
       'SprintSpeed', 'Agility', 'Reactions', 'Balance', 'ShotPower',
       'Jumping', 'Stamina', 'Strength', 'LongShots', 'Aggression',
       'Interceptions', 'Positioning', 'Vision', 'Penalties', 'Composure',
       'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving', 'GKHandling',
       'GKKicking', 'GKPositioning', 'GKReflexes', 'Release Clause'],
      dtype='object')
```

In [6]:

```
#Data preprocessing
missing_value_matrix=football_data.isna()
```

In [7]:

```
missing_value_matrix # this matix keeps a log of values that are missing in the given dataset
```

Out[7]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	Club	...	Composure	Marking	StandingTackle	SlidingTackle
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18202	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
18203	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
18204	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
18205	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
18206	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False

18207 rows × 89 columns

In [8]:

```
missing_height_index=missing_value_matrix[missing_value_matrix['Height']==True].index.tolist()
```

In [9]:

```
len(missing_height_index)
```

Out[9]: 48

The values that i feel contribute to heigt of a player can be it's nationality

In [10]:

```
missing_value_matrix[missing_value_matrix['Nationality']==True].index
```

```
# since the nationality is present for all the players so let us try to update tyhe values of the height based upon t
#height for that given nationality
```

Out[10]: Int64Index([], dtype='int64')

In [11]:

```
unique_nationalities=football_data['Nationality'].unique()
```

In [12]:

```
len(football_data)
```

Out[12]: 18207

In [13]:

```
nationality_height_averge_values=pd.DataFrame()
```

In [14]:

```
tempstr_int_cm=int(football_data['Height'][0][0])*30.48 + float(float(football_data['Height'][0][2])/12.0)*30.48
```

In [15]:

```
tempstr_int_cm
```

Out[15]: 170.18

In [19]:

```
float(float(football_data['Height'][0][2])/12.0)*30.48
```

Out[19]: 17.78

In [20]:

```
tempstr_int_cm
```

Out[20]: 170.18

In [21]:

```
#converting the height to centimeters from a string formatted feet and inches
for i in range(len(football_data['Height'])):
    if missing_value_matrix['Height'][i]==False:
```

```
        tempstr_int_cm=int(football_data['Height'][i][0])*30.48 + float(float(football_data['Height'][i][2])/12.0)*30
        football_data['Height'][i]=tempstr_int_cm
```

```
<ipython-input-21-fc7e96f0a476>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data['Height'][i]=tempstr_int_cm
```

In [22]:

```
# getting the unique nationalitites
nationality_height_averge_values['unique_nationalities']=unique_nationalities
nationality_height_averge_values['average_height']=np.zeros(len(unique_nationalities))
```

In [23]:

```
# getting the average height as per nationality
sum=0
counter=0
for i in range(len(unique_nationalities)):
    for j in range(len(football_data)):
        if football_data['Nationality'][j]==nationality_height_averge_values['unique_nationalities'][i] and missing_v
            sum=sum+football_data['Height'][j]
            counter=counter+1
    nationality_height_averge_values['average_height'][i]=(sum/counter)
    sum=0
    counter=0
```

```
<ipython-input-23-88612296aff8>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  nationality_height_averge_values['average_height'][i]=(sum/counter)
```

In [28]:

```
# updating the missing height values as per the avg height of that nationality
for i in range(len(football_data['Height'])):
```

```python
    if missing_value_matrix['Height'][i]==True:
        for j in range(len(unique_nationalities)):
            if football_data['Nationality'][i]==nationality_height_averge_values['unique_nationalities'][j] :
                football_data['Height'][i]=nationality_height_averge_values['average_height'][j]
                missing_value_matrix['Height'][i]=False
                break
```

```
<ipython-input-28-5d655eb6d4f3>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data['Height'][i]=nationality_height_averge_values['average_height'][j]
```

In [38]:

```python
nationality_height_averge_values # looking at the average height as per nationality
```

Out[38]:

	unique_nationalities	average_height
0	Argentina	172.559893
1	Portugal	174.131988
2	Brazil	174.668873
3	Spain	173.972213
4	Belgium	176.780077
...	...	...
159	Malta	167.640000
160	Belize	154.940000
161	South Sudan	200.660000
162	Indonesia	165.100000
163	Botswana	175.260000

164 rows × 2 columns

In [24]:

```
# creating categorical subdivision of the data for easy pre-processing
personal_details=football_data[['ID','Name','Age','Photo','Nationality','Height','Weight']]
position_details=[['LS', 'ST', 'RS', 'LW', 'LF', 'CF', 'RF', 'RW',
       'LAM', 'CAM', 'RAM', 'LM', 'LCM', 'CM', 'RCM', 'RM', 'LWB', 'LDM',
       'CDM', 'RDM', 'RWB', 'LB', 'LCB', 'CB', 'RCB', 'RB']]
skills=football_data[['Crossing',
       'Finishing', 'HeadingAccuracy', 'ShortPassing', 'Volleys', 'Dribbling',
       'Curve', 'FKAccuracy', 'LongPassing', 'BallControl', 'Acceleration',
       'SprintSpeed', 'Agility', 'Reactions', 'Balance', 'ShotPower',
       'Jumping', 'Stamina', 'Strength', 'LongShots', 'Aggression',
       'Interceptions', 'Positioning', 'Vision', 'Penalties', 'Composure',
       'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving', 'GKHandling',
       'GKKicking', 'GKPositioning', 'GKReflexes']]
club_details=[['Club', 'Club Logo', 'Value', 'Wage', 'Special',
       'Preferred Foot', 'International Reputation', 'Weak Foot',
       'Skill Moves', 'Work Rate', 'Body Type', 'Real Face']]
```

In [36]:

```
# converting the value in the columns of position_details from string to integer , discarding anyththing after the + s
for i in range(len(position_details[0])):
    for j in range(len(football_data)):
        if missing_value_matrix[position_details[0][i]][j]==False:
            football_data[position_details[0][i]][j]=int(football_data[position_details[0][i]][j].split("+")[0])
```

```
<ipython-input-36-e73ce568419a>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][j]=int(football_data[position_details[0][i]][j].split("+")[0])
```

In [42]:

```
# handling the missing values for columns falling under position_deatils
sum=0
counter=0
missing_positions=[]
for i in range(len(position_details[0])):
    missing_positions=missing_value_matrix[missing_value_matrix[position_details[0]]==True].index.tolist()
    for j in range(len(football_data)):
```

```
        if missing_value_matrix[position_details[0][i]][j]==False:
            sum=sum+football_data[position_details[0][i]][j]
            counter=counter+1
    avg=sum/counter
    for k in range(len(missing_positions)):
        football_data[position_details[0][i]][missing_positions[k]]=avg
        missing_value_matrix[position_details[0][i]][missing_positions[k]]=False

    sum=0
    counter=0
    avg=0
```

```
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
```

```
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
<ipython-input-42-43c68a9ea818>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[position_details[0][i]][missing_positions[k]]=avg
```

In [57]:

```
# checking for other missing values in various columns sequentially
columns_with_missing_values=[]
for i in range(len(football_data.columns)):
    tempstr=football_data.columns[i]
```

```python
    if len(missing_value_matrix[missing_value_matrix[tempstr]==True].index )>0 :
        columns_with_missing_values.append(tempstr)
```

In [58]:

```python
columns_with_missing_values # list of columns that still have missing values
```

Out[58]:

```
['Club',
 'Preferred Foot',
 'International Reputation',
 'Weak Foot',
 'Skill Moves',
 'Work Rate',
 'Body Type',
 'Real Face',
 'Position',
 'Jersey Number',
 'Joined',
 'Loaned From',
 'Contract Valid Until',
 'Weight',
 'Crossing',
 'Finishing',
 'HeadingAccuracy',
 'ShortPassing',
 'Volleys',
 'Dribbling',
 'Curve',
 'FKAccuracy',
 'LongPassing',
 'BallControl',
 'Acceleration',
 'SprintSpeed',
 'Agility',
 'Reactions',
 'Balance',
 'ShotPower',
 'Jumping',
 'Stamina',
 'Strength',
 'LongShots',
 'Aggression',
 'Interceptions',
```

```
 'Positioning',
 'Vision',
 'Penalties',
 'Composure',
 'Marking',
 'StandingTackle',
 'SlidingTackle',
 'GKDiving',
 'GKHandling',
 'GKKicking',
 'GKPositioning',
 'GKReflexes',
 'Release Clause']
```

In [85]:

```python
# handing missing values for the release clause ( adding value =0 for missing atributes )
#and converting the data to float from string
for i in range(len(football_data)):
    if missing_value_matrix['Release Clause'][i]==False:
        if football_data['Release Clause'][i].split("€")[1][-1]=='M':
            football_data['Release Clause'][i]=float(football_data['Release Clause'][i].split('M')[0].split("€")[1])
        else:
            football_data['Release Clause'][i]=(float(football_data['Release Clause'][i].split("€")[1][:-1])/1000.0)
    else:
        football_data['Release Clause'][i]=0
        missing_value_matrix['Release Clause'][i]=True
```

```
<ipython-input-85-5317e881c387>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data['Release Clause'][i]=float(football_data['Release Clause'][i].split('M')[0].split("€")[1])
<ipython-input-85-5317e881c387>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data['Release Clause'][i]=0
<ipython-input-85-5317e881c387>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data['Release Clause'][i]=(float(football_data['Release Clause'][i].split("€")[1][:-1])/1000.0)
```

In [115…

```
# converting the column value and wage to float as well( as they don't have any missing values)
listx=['Value','Wage']
for j in range(len(listx)):
    for i in range(len(football_data)):
        if missing_value_matrix[listx[j]][i]==False:
            if football_data[listx[j]][i].split("€")[1][-1]=='M':
                football_data[listx[j]][i]=float(football_data[listx[j]][i].split('M')[0].split("€")[1])
            else:
                if football_data[listx[j]][i].split("€")[1]=='0':
                    football_data[listx[j]][i]=0
                else:
                    football_data[listx[j]][i]=(float(football_data[listx[j]][i].split("€")[1][:-1])/1000.0)
        else:
            football_data[listx[j]][i]=0
            missing_value_matrix[listx[j]][i]=True
```

```
<ipython-input-115-15725b62b711>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[listx[j]][i]=float(football_data[listx[j]][i].split('M')[0].split("€")[1])
<ipython-input-115-15725b62b711>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[listx[j]][i]=0
<ipython-input-115-15725b62b711>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  football_data[listx[j]][i]=(float(football_data[listx[j]][i].split("€")[1][:-1])/1000.0)
```

In [118…

```
import os
os.chdir("F:\MS IIITH\course structure and syllabus\Semester I\Data Analytics\Project")
```

In [119…

```
# exporting the preprocessed dataset for using to solve the questions
football_data.to_csv("reformed_datset.csv")
```

Q1 Solution Data Visualization

In [2]:
```
!pip install numpy
import numpy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (1.19.5)
```

In [3]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.1.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas) (2018.9)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from pandas) (1.19.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
```

In [6]:
```
import pandas as pd
football = pd.read_csv(r'/reformed_datset.csv', encoding='iso-8859-1')
football.head()
```

Out[6]:

	Unnamed: 0	Unnamed: 0.1	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	Club	Club Log
0	0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	FC Barcelona	https://cdn.sofifa.org/teams/2/light/241.pr
1	1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	Juventus	https://cdn.sofifa.org/teams/2/light/45.pr
2	2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	Paris Saint-Germain	https://cdn.sofifa.org/teams/2/light/73.pr
3	3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	Manchester United	https://cdn.sofifa.org/teams/2/light/11.pr
4	4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	Manchester City	https://cdn.sofifa.org/teams/2/light/10.pr

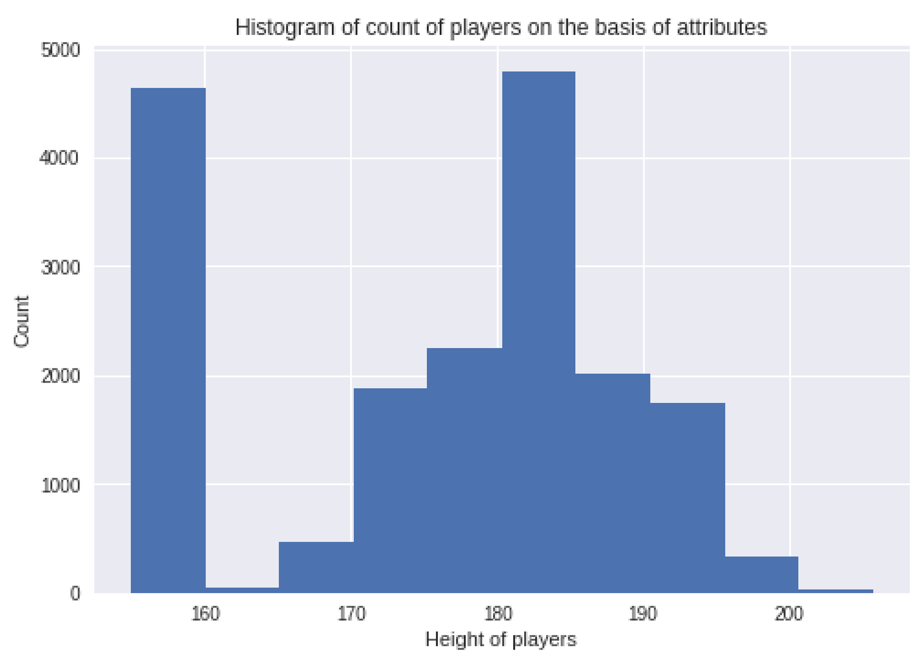
5 rows × 90 columns

In [8]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [9]:
```
plt.style.use('seaborn')
```

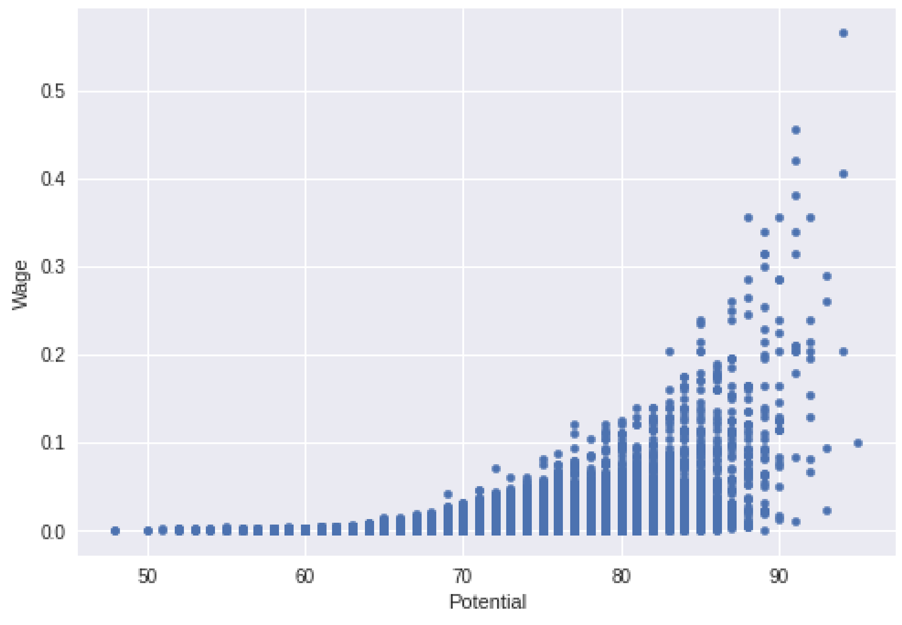
Histogram showing count of players based on height

In [10]:
```
plt.hist(football['Height'])
plt.xlabel('Height of players')
plt.ylabel('Count')
plt.title('Histogram of count of players on the basis of attributes')
plt.show()
```



Scatter plot showing Potential v/s Wages

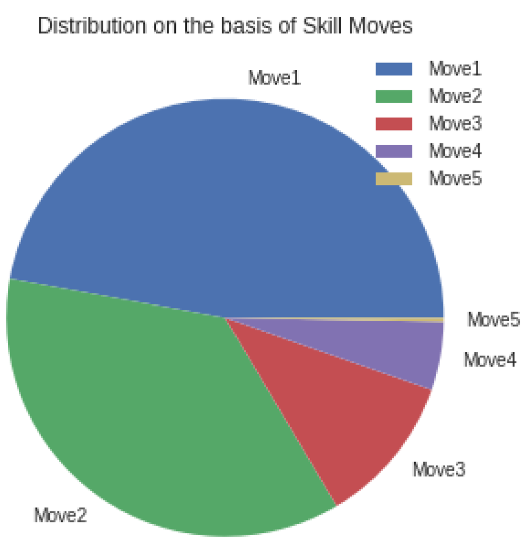
In [11]:
```
football.plot(kind='scatter',x='Potential',y='Wage')
plt.show()
```



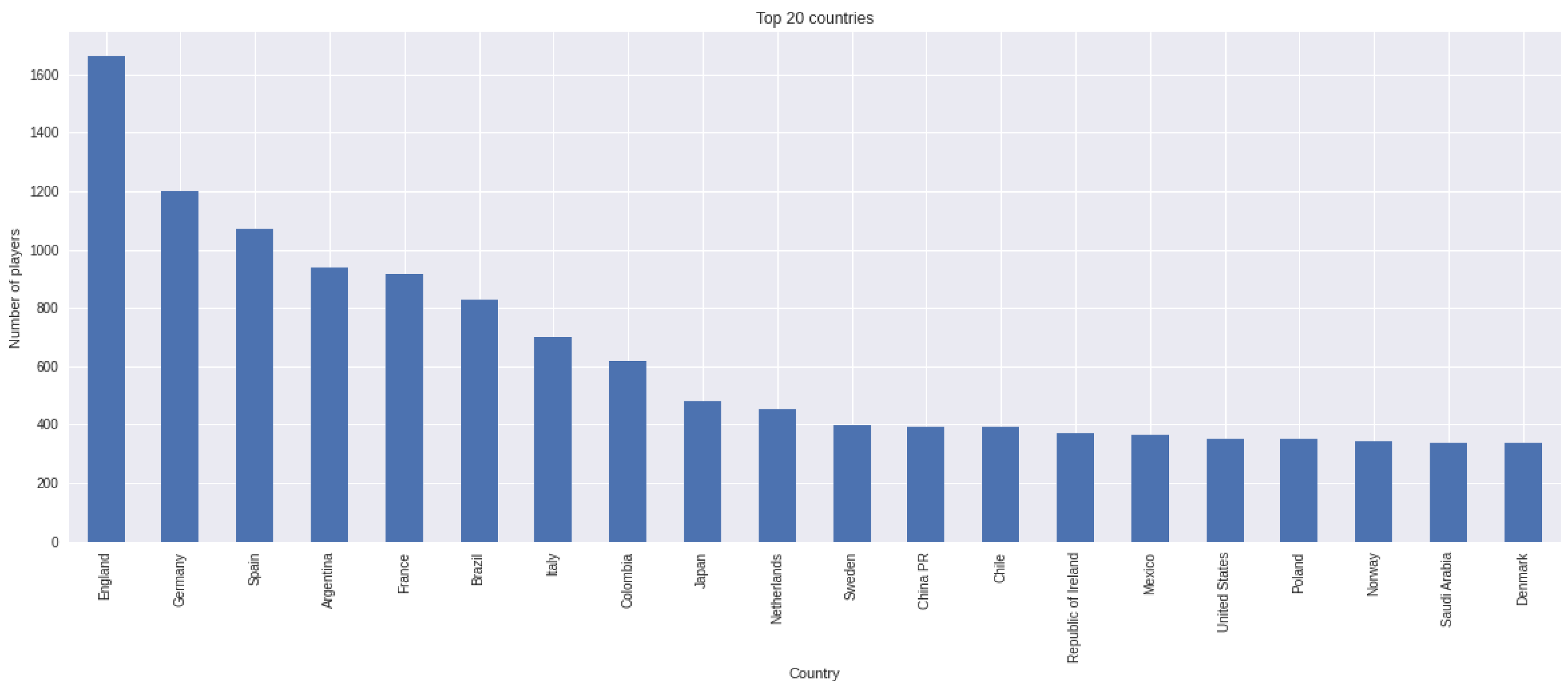
An interesting observation made in above plot, player with maximum potential has low Wage (0.1)

In [13]:
```
import numpy as np
```

In [14]:
```
pos=['Move1','Move2','Move3','Move4','Move5']
counts=football['Skill Moves'].value_counts()
pos1=np.array(pos)
plt.title("Distribution on the basis of Skill Moves")
plt.pie(counts,labels=pos)
plt.legend();
plt.show()
```

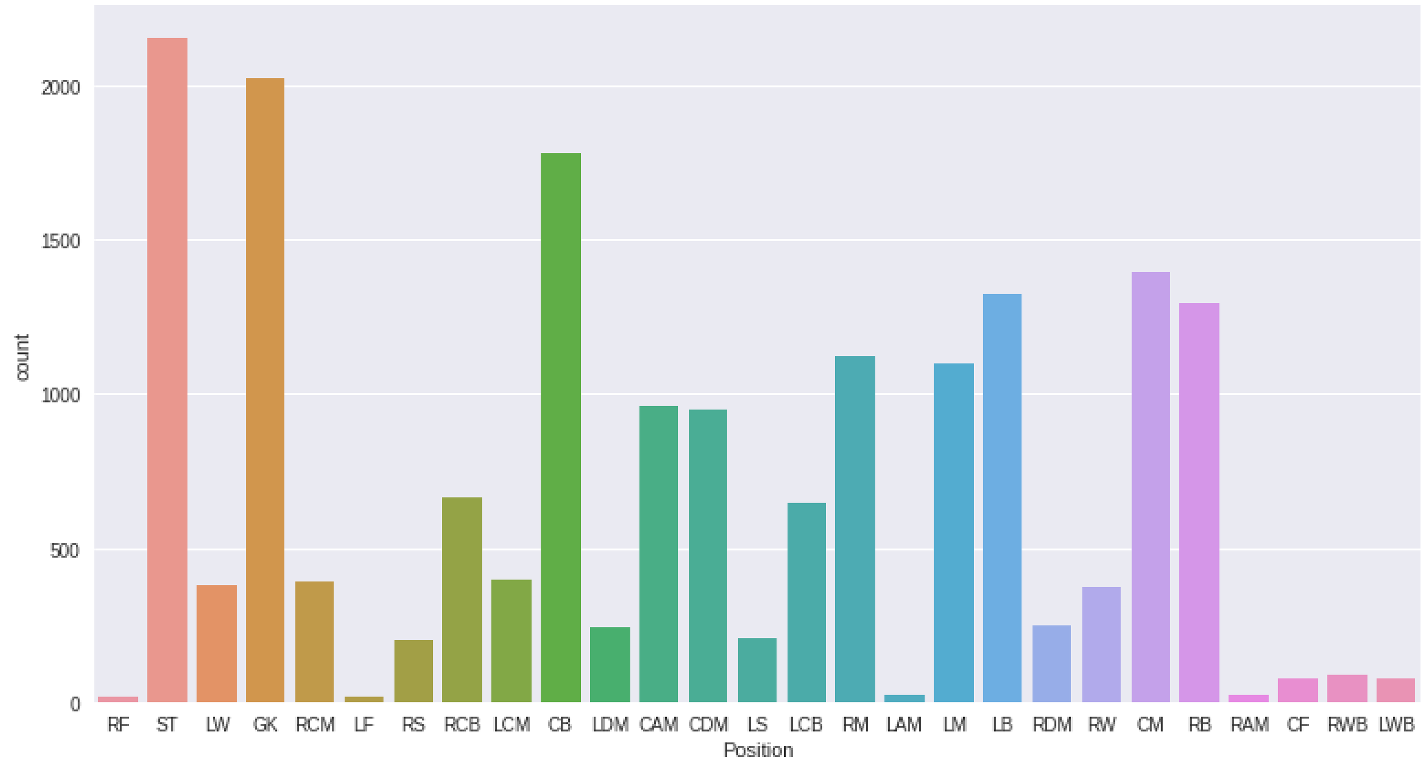


In [15]:
```
football['Nationality'].value_counts().head(20).plot.bar(figsize = (20, 7))
plt.title('Top 20 countries')
plt.xlabel('Country')
plt.ylabel('Number of players')
plt.show()
```

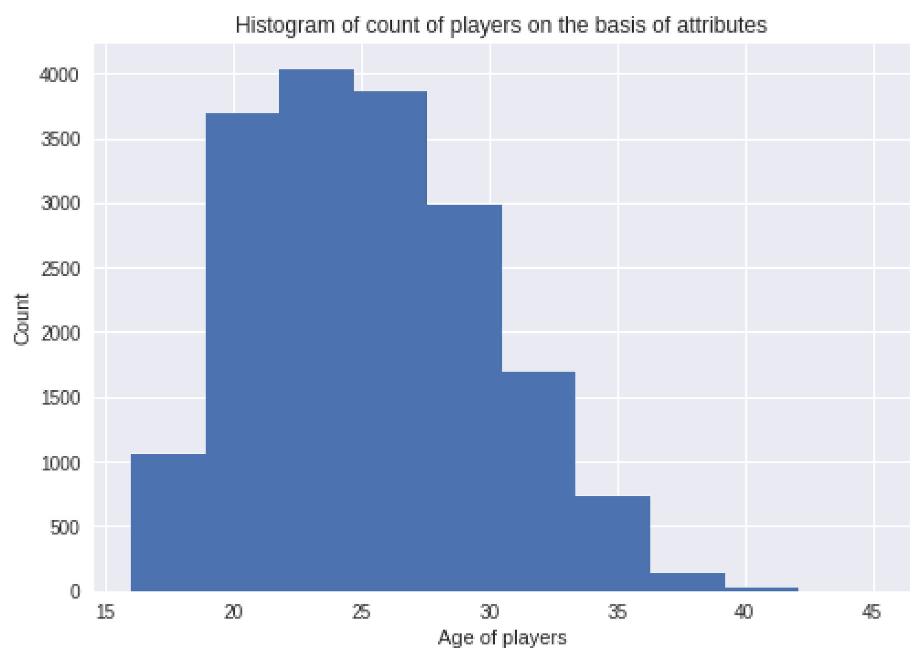


In [16]:
```
import seaborn as sb
plt.figure(figsize = (13, 7))
sb.countplot('Position', data = football)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



In [19]:
```
plt.hist(football['Age'])
plt.xlabel('Age of players')
plt.ylabel('Count')
plt.title('Histogram of count of players on the basis of attributes')
plt.show()
```



In [ ]:

Q2(1) K Means code for K=3

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import random as rd
```

In [2]:
```
dataset=pd.read_csv(r'F:\MS IIITH\course structure and syllabus\Semester I\Data Analytics\Project\reformed_datset.csv
```

In [3]:
```
# let us begin with implementing k means for value and wage
dataset_considered=pd.DataFrame()
dataset_considered['Value']=dataset['Value']
dataset_considered['Wage']=dataset['Wage']
```

In [4]:
```
K=3# value of specified
```

In [5]:
```
n_iter=30# It defines the total no. of itterations for convergence
```

In [6]:
```
m=dataset_considered.shape[0] #number of training examples
n=dataset_considered.shape[1] #number of features. Here n=2
```

In [7]:
```
#initialize centroids randomly from data points
Centroids = pd.DataFrame(index=range(n),columns=range(10))
```

In [8]:
```
for i in range(K):
    rand=rd.randint(0,m-1)
    Centroids[i][0]=dataset_considered['Value'][rand]
    Centroids[i][1]=dataset_considered['Wage'][rand]
```

In [9]:
```
output= pd.DataFrame(index=range(len(dataset_considered)),columns=range(K))
```

In [10]:

```
def euclidean_distance():
    #print("within elcid dist calculation ")
    # finding Euclidean distance between each point to all the centroids
    p=[0,0]
    q=[0,0]
    for i  in range(len(dataset_considered)):
        for j in range(K):
            #print(i)
            #print(j)
            #print("----------")
            p[0]=Centroids[j][0]
            p[1]=Centroids[j][1]
            q[0]=dataset_considered['Value'][i]
            q[1]=dataset_considered['Wage'][i]
            output[j][i]=math.dist(p,q)
```

In [11]:

```
def cluster_labels():
    #print("within cluster label generation")
    # updating the cluster labels for points
    for i in range(1,len(cluster_seg)+1):
        valset=output.iloc[i-1].to_list()
        cluster_seg[i-1]=valset.index(np.min(valset))
```

In [12]:

```
val_checker=pd.DataFrame(index=range(n+1),columns=range(K))
```

In [13]:

```
counter_value=np.zeros(K)
```

In [14]:

```
def centroid_updation(K):
    #print("within centroid updation")
    sum_value=np.zeros(K)
    wage_value=np.zeros(K)
    counter_value=np.zeros(K)
```

```
    #updating the centroid value as per the points in the cluster
    for i in range(len(cluster_seg)):
        for j in range(K):
            if cluster_seg[i]==j :
                sum_value[j]=sum_value[j]+dataset_considered['Value'][i]
                wage_value[j]=wage_value[j]+dataset_considered['Wage'][i]
                counter_value[j]=counter_value[j]+1


    for i in range(K):
        for j in range(len(Centroids)):
                if j==0:
                    Centroids[i][j]=(sum_value[i]/counter_value[i]) #value for cluster in centroid
                else:
                    Centroids[i][j]=(wage_value[i]/counter_value[i])#wage for cluster 1 in centroid
```

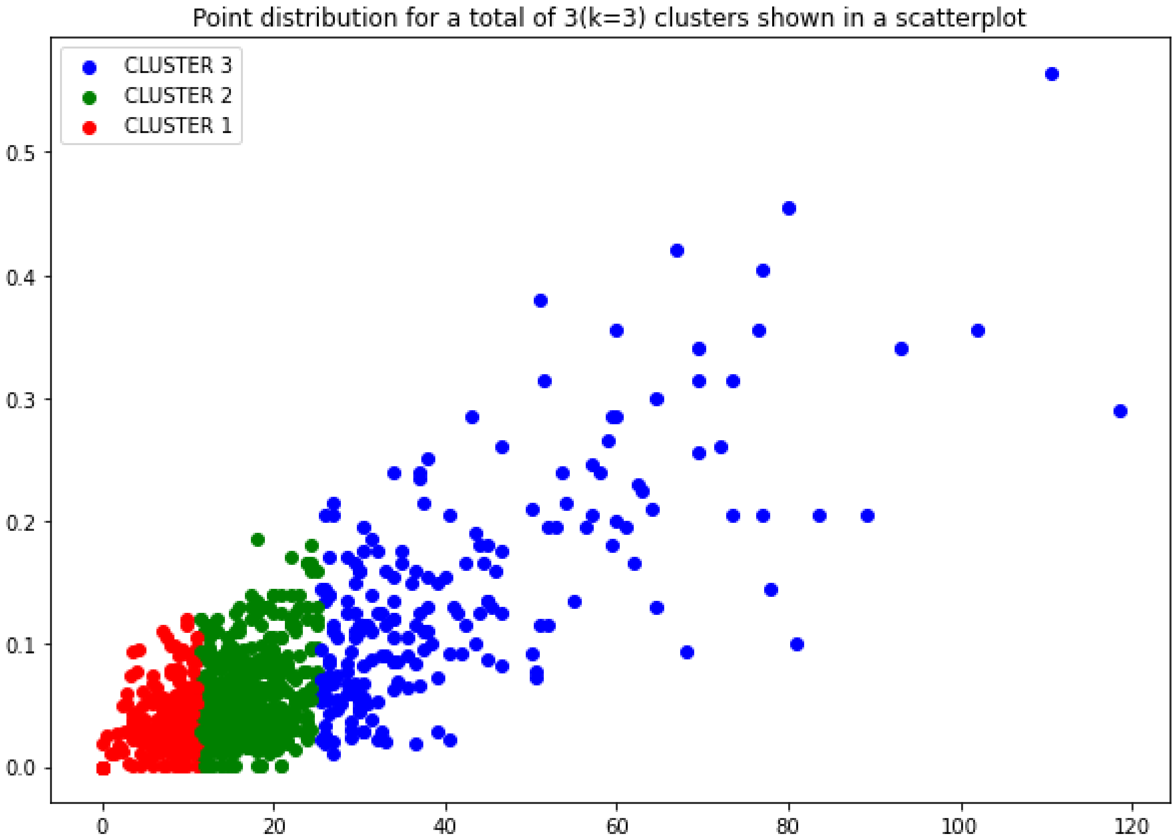
In [15]:

```
#def k_means(K):
#print("within kmeans calculation")
for i in range(n_iter):
    #print("itteration in k means")
    #print(i)
    #print("calling euclidean_distance")
    euclidean_distance()
    #creating an empty list to store the clusters where the points need to be fit
    cluster_seg=[None]*len(output)
    cluster_labels()
    centroid_updation(K)
```

In [16]:

```
plt.figure(figsize=(10, 7))
c1=0
c2=0
c3=0
c4=0
c5=0
c6=0
c7=0
for j in range(len(cluster_seg)):
  if cluster_seg[j]==0:
    if c1==0:
```

```
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Red', label="CLUSTER 1")
      c1=c1+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Red')

  if cluster_seg[j]==1:
    if c2==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Green' , label="CLUSTER 2")
      c2=c2+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Green')

  if cluster_seg[j]==2:
    if c3==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Blue' , label="CLUSTER 3")
      c3=c3+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Blue')




plt.title("Point distribution for a total of 3(k=3) clusters shown in a scatterplot")
plt.legend()
plt.show()
```


Point distribution for a total of 3(k=3) clusters shown in a scatterplot
CLUSTER 3
CLUSTER 2
CLUSTER 1
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
120

Q2(1) K Means Code for K=5

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import random as rd
```

In [2]:
```
dataset=pd.read_csv(r'F:\MS IIITH\course structure and syllabus\Semester I\Data Analytics\Project\reformed_datset.csv
```

In [3]:
```
# let us begin with implementing k means for value and wage
dataset_considered=pd.DataFrame()
dataset_considered['Value']=dataset['Value']
dataset_considered['Wage']=dataset['Wage']
```

In [4]:
```
K=5# value of specified
```

In [5]:
```
n_iter=30# It defines the total no. of itterations for convergence
```

In [6]:
```
m=dataset_considered.shape[0] #number of training examples
n=dataset_considered.shape[1] #number of features. Here n=2
```

In [7]:
```
#initialize centroids randomly from data points
Centroids = pd.DataFrame(index=range(n),columns=range(10))
```

In [8]:
```
for i in range(K):
    rand=rd.randint(0,m-1)
    Centroids[i][0]=dataset_considered['Value'][rand]
    Centroids[i][1]=dataset_considered['Wage'][rand]
```

In [9]:
```
output= pd.DataFrame(index=range(len(dataset_considered)),columns=range(K))
```

In [10]:

```
def euclidean_distance():
    #print("within elcid dist calculation ")
    # finding Euclidean distance between each point to all the centroids
    p=[0,0]
    q=[0,0]
    for i  in range(len(dataset_considered)):
        for j in range(K):
            #print(i)
            #print(j)
            #print("----------")
            p[0]=Centroids[j][0]
            p[1]=Centroids[j][1]
            q[0]=dataset_considered['Value'][i]
            q[1]=dataset_considered['Wage'][i]
            output[j][i]=math.dist(p,q)
```

In [11]:

```
def cluster_labels():
    #print("within cluster label generation")
    # updating the cluster labels for points
    for i in range(1,len(cluster_seg)+1):
        valset=output.iloc[i-1].to_list()
        cluster_seg[i-1]=valset.index(np.min(valset))
```

In [12]:

```
val_checker=pd.DataFrame(index=range(n+1),columns=range(K))
```

In [13]:

```
counter_value=np.zeros(K)
```

In [14]:

```
def centroid_updation(K):
    #print("within centroid updation")
    sum_value=np.zeros(K)
    wage_value=np.zeros(K)
    counter_value=np.zeros(K)
```

```
    #updating the centroid value as per the points in the cluster
    for i in range(len(cluster_seg)):
        for j in range(K):
            if cluster_seg[i]==j :
                sum_value[j]=sum_value[j]+dataset_considered['Value'][i]
                wage_value[j]=wage_value[j]+dataset_considered['Wage'][i]
                counter_value[j]=counter_value[j]+1


    for i in range(K):
        for j in range(len(Centroids)):
                if j==0:
                    Centroids[i][j]=(sum_value[i]/counter_value[i]) #value for cluster in centroid
                else:
                    Centroids[i][j]=(wage_value[i]/counter_value[i])#wage for cluster 1 in centroid
```

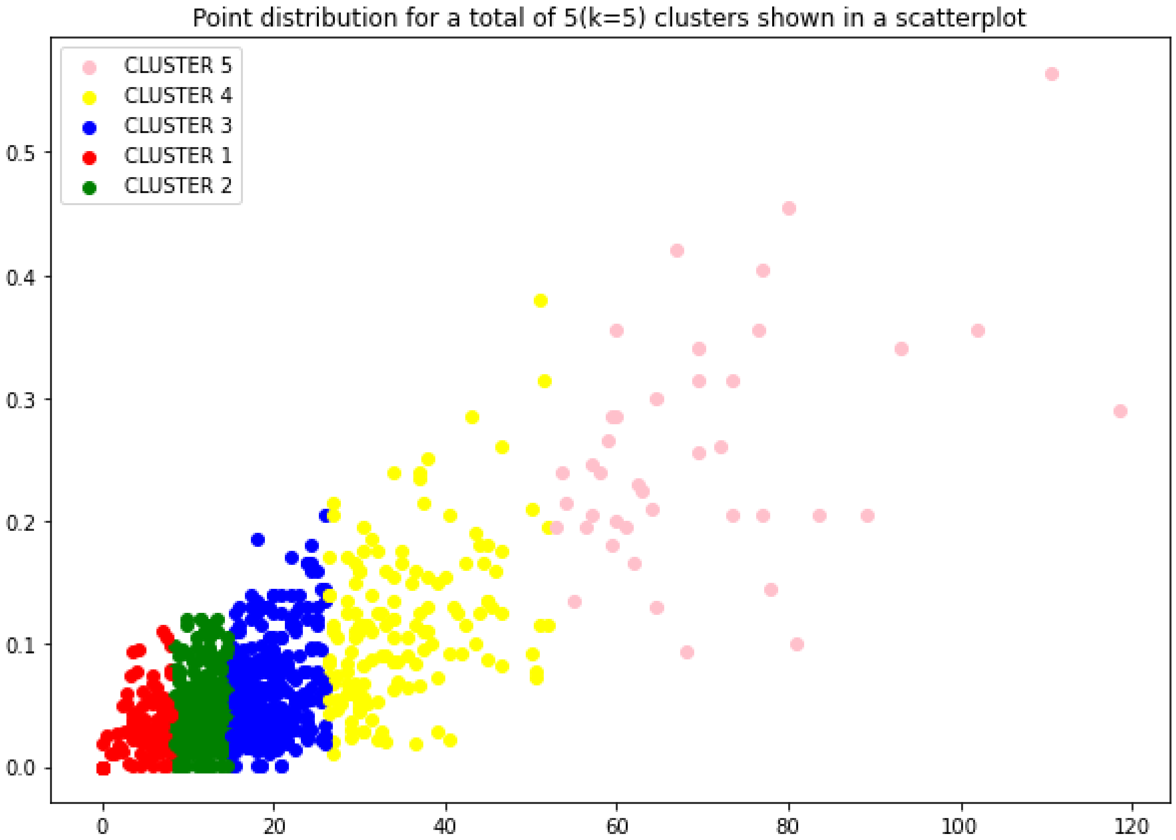
In [15]:

```
#def k_means(K):
#print("within kmeans calculation")
for i in range(n_iter):
    #print("itteration in k means")
    #print(i)
    #print("calling euclidean_distance")
    euclidean_distance()
    #creating an empty list to store the clusters where the points need to be fit
    cluster_seg=[None]*len(output)
    cluster_labels()
    centroid_updation(K)
```

In [16]:

```
plt.figure(figsize=(10, 7))
c1=0
c2=0
c3=0
c4=0
c5=0
c6=0
c7=0
for j in range(len(cluster_seg)):
  if cluster_seg[j]==0:
    if c1==0:
```

```
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Red', label="CLUSTER 1")
            c1=c1+1
        else:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Red')

    if cluster_seg[j]==1:
        if c2==0:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Green' , label="CLUSTER 2")
            c2=c2+1
        else:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Green')

    if cluster_seg[j]==2:
        if c3==0:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Blue' , label="CLUSTER 3")
            c3=c3+1
        else:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Blue')

    if cluster_seg[j]==3:
        if c4==0:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Yellow' , label="CLUSTER 4")
            c4=c4+1
        else:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Yellow')

    if cluster_seg[j]==4:
        if c5==0:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Pink' , label="CLUSTER 5")
            c5=c5+1
        else:
            plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Pink' )




plt.title("Point distribution for a total of 5(k=5) clusters shown in a scatterplot")
plt.legend()
plt.show()
```



Point distribution for a total of 5(k=5) clusters shown in a scatterplot
CLUSTER 5
CLUSTER 4
CLUSTER 3
CLUSTER 1
CLUSTER 2
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
120


```
In [ ]:
```

Q2(1) K means code for K=7

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import random as rd
```

In [2]:
```
dataset=pd.read_csv(r'F:\MS IIITH\course structure and syllabus\Semester I\Data Analytics\Project\reformed_datset.csv
```

In [3]:
```
# let us begin with implementing k means for value and wage
dataset_considered=pd.DataFrame()
dataset_considered['Value']=dataset['Value']
dataset_considered['Wage']=dataset['Wage']
```

In [4]:
```
K=7# value of specified
```

In [5]:
```
n_iter=30# It defines the total no. of itterations for convergence
```

In [6]:
```
m=dataset_considered.shape[0] #number of training examples
n=dataset_considered.shape[1] #number of features. Here n=2
```

In [7]:
```
#initialize centroids randomly from data points
Centroids = pd.DataFrame(index=range(n),columns=range(10))
```

In [8]:
```
for i in range(K):
    rand=rd.randint(0,m-1)
    Centroids[i][0]=dataset_considered['Value'][rand]
    Centroids[i][1]=dataset_considered['Wage'][rand]
```

In [9]:
```
output= pd.DataFrame(index=range(len(dataset_considered)),columns=range(K))
```

In [10]:

```
def euclidean_distance():
    #print("within elcid dist calculation ")
    # finding Euclidean distance between each point to all the centroids
    p=[0,0]
    q=[0,0]
    for i  in range(len(dataset_considered)):
        for j in range(K):
            #print(i)
            #print(j)
            #print("----------")
            p[0]=Centroids[j][0]
            p[1]=Centroids[j][1]
            q[0]=dataset_considered['Value'][i]
            q[1]=dataset_considered['Wage'][i]
            output[j][i]=math.dist(p,q)
```

In [11]:

```
# creating an empty list to store the clusters where the points need to be fit
#cluster_seg=[None]*len(output)
```

In [12]:

```
def cluster_labels():
    #print("within cluster label generation")
    # updating the cluster labels for points
    for i in range(1,len(cluster_seg)+1):
        valset=output.iloc[i-1].to_list()
        cluster_seg[i-1]=valset.index(np.min(valset))
```

In [13]:

```
val_checker=pd.DataFrame(index=range(n+1),columns=range(K))
```

In [14]:

```
counter_value=np.zeros(K)
```

In [15]:

```
def centroid_updation(K):
    #print("within centroid updation")
    sum_value=np.zeros(K)
```

```
    wage_value=np.zeros(K)
    counter_value=np.zeros(K)


    #updating the centroid value as per the points in the cluster
    for i in range(len(cluster_seg)):
        for j in range(K):
            if cluster_seg[i]==j :
                sum_value[j]=sum_value[j]+dataset_considered['Value'][i]
                wage_value[j]=wage_value[j]+dataset_considered['Wage'][i]
                counter_value[j]=counter_value[j]+1


    for i in range(K):
        for j in range(len(Centroids)):
                if j==0:
                    Centroids[i][j]=(sum_value[i]/counter_value[i]) #value for cluster in centroid
                else:
                    Centroids[i][j]=(wage_value[i]/counter_value[i])#wage for cluster 1 in centroid
```

In [16]:

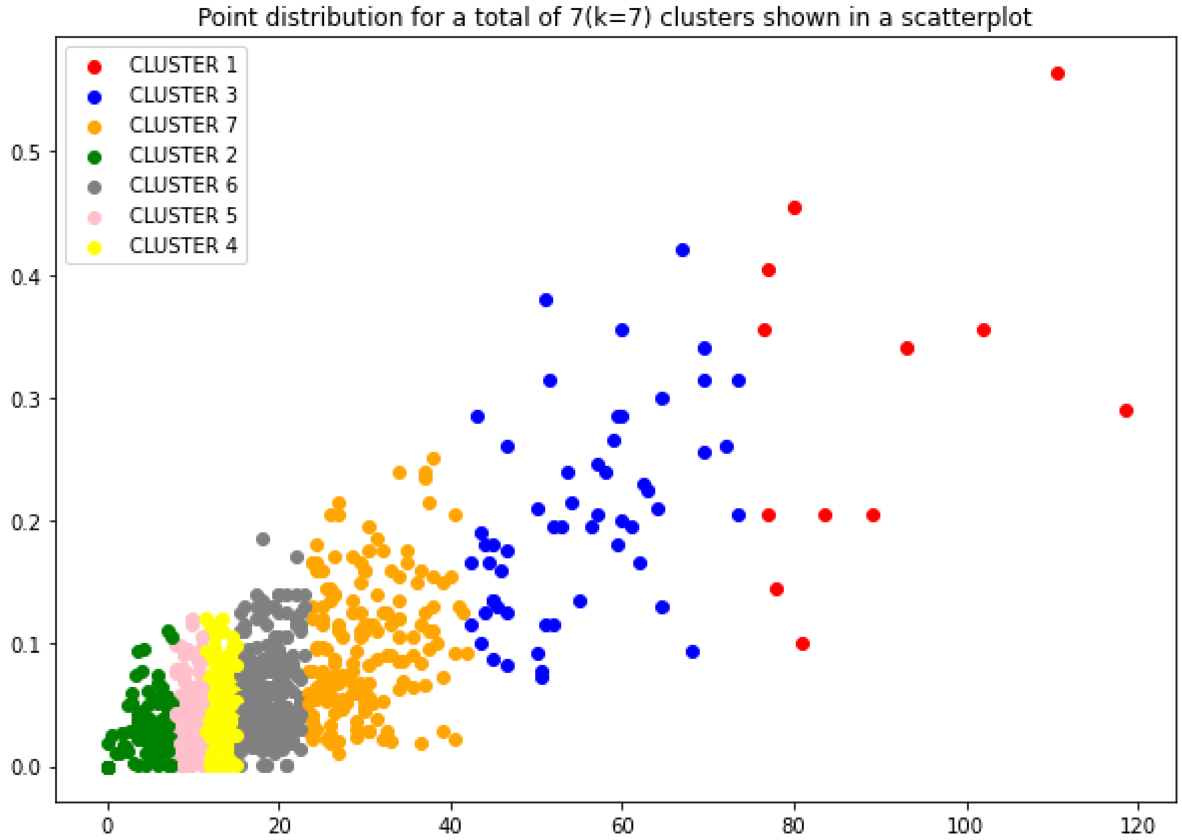
```
#def k_means(K):
#print("within kmeans calculation")
for i in range(n_iter):
    #print("itteration in k means")
    #print(i)
    #print("calling euclidean_distance")
    euclidean_distance()
    #creating an empty list to store the clusters where the points need to be fit
    cluster_seg=[None]*len(output)
    cluster_labels()
    centroid_updation(K)
```

In [17]:

```
plt.figure(figsize=(10, 7))
c1=0
c2=0
c3=0
c4=0
c5=0
c6=0
```

```
c7=0
for j in range(len(cluster_seg)):
  if cluster_seg[j]==0:
    if c1==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Red', label="CLUSTER 1")
      c1=c1+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Red')

  if cluster_seg[j]==1:
    if c2==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Green' , label="CLUSTER 2")
      c2=c2+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Green')

  if cluster_seg[j]==2:
    if c3==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Blue' , label="CLUSTER 3")
      c3=c3+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Blue')

  if cluster_seg[j]==3:
    if c4==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Yellow' , label="CLUSTER 4")
      c4=c4+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Yellow')

  if cluster_seg[j]==4:
    if c5==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Pink' , label="CLUSTER 5")
      c5=c5+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Pink' )

  if cluster_seg[j]==5:
    if c6==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j],c='Grey',  label="CLUSTER 6")
      c6=c6+1
    else:
```

```
        plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j],c='Grey')

    if cluster_seg[j]==6:
      if c7==0:
        plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j],c='Orange',  label="CLUSTER 7")
        c7=c7+1
      else:
        plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j],c='Orange')



plt.title("Point distribution for a total of 7(k=7) clusters shown in a scatterplot")
plt.legend()
plt.show()
```



Q2 Code for Elbow method and Silhouette Score

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import random as rd
from sklearn.metrics import silhouette_score
```

In [2]:

```
dataset=pd.read_csv(r'F:\MS IIITH\course structure and syllabus\Semester I\Data Analytics\Project\reformed_datset.csv
```

In [3]:

```
dataset.describe()
```

Out[3]:

	Unnamed: 0	Unnamed: 0.1	ID	Age	Overall	Potential	Value	Wage	Special	Internatio Reputati
count	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18159.0000
mean	9103.000000	9103.000000	214298.338606	25.122206	66.238699	71.307299	2.410696	0.009731	1597.809908	1.1132
std	5256.052511	5256.052511	29965.244204	4.669943	6.908930	6.136496	5.594933	0.021999	272.586016	0.3940
min	0.000000	0.000000	16.000000	16.000000	46.000000	48.000000	0.000000	0.000000	731.000000	1.0000
25%	4551.500000	4551.500000	200315.500000	21.000000	62.000000	67.000000	0.300000	0.001000	1457.000000	1.0000
50%	9103.000000	9103.000000	221759.000000	25.000000	66.000000	71.000000	0.675000	0.003000	1635.000000	1.0000
75%	13654.500000	13654.500000	236529.500000	28.000000	71.000000	75.000000	2.000000	0.009000	1787.000000	1.0000
max	18206.000000	18206.000000	246620.000000	45.000000	94.000000	95.000000	118.500000	0.565000	2346.000000	5.0000

8 rows × 75 columns

In [4]:

```
# let us begin with implementing k means for value and wage
dataset_considered=pd.DataFrame()
dataset_considered['Value']=dataset['Value']
dataset_considered['Wage']=dataset['Wage']
```

In [5]:
```
m=dataset_considered.shape[0] #number of training examples
n=dataset_considered.shape[1] #number of features. Here n=2
```

In [6]:
```
m
```

Out[6]: 1000

In [7]:
```
n
```

Out[7]: 2

In [8]:
```
n_iter=30 # total no of itterations for k to converge
```

In [9]:
```
K=10 # number of clusters( initial value )
```

In [10]:
```
#initialize centroids randomly from data points
Centroids = pd.DataFrame(index=range(n),columns=range(10))
```

In [11]:
```
Centroids.shape
```

Out[11]: (2, 10)

Centroids is a n x K dimentional matrix, where each column will be a centroid for one cluster.(2 x 5)

In [12]:
```
for i in range(K):
    rand=rd.randint(0,m-1)
    Centroids[i][0]=dataset_considered['Value'][rand]
    Centroids[i][1]=dataset_considered['Wage'][rand]
```

In [13]:
```
Centroids
```

```
# zeroth row is value
# first row is wage
```

Out[13]:

	0	1	2	3	4	5	6	7	8	9
0	13.0	34.0	21.5	45.0	10.5	8.5	50.0	10.5	8.5	17.5
1	0.026	0.085	0.021	0.18	0.046	0.037	0.092	0.023	0.024	0.037

In [14]:

```
output= pd.DataFrame(index=range(len(dataset_considered)),columns=range(K))
```

In [17]:

```
def euclidean_distance():
    print("within elcid dist calculation ")
    # finding Euclidean distance between each point to all the centroids
    p=[0,0]
    q=[0,0]
    for i  in range(len(dataset_considered)):
        for j in range(K):
            #print(i)
            #print(j)
            #print("----------")
            p[0]=Centroids[j][0]
            p[1]=Centroids[j][1]
            q[0]=dataset_considered['Value'][i]
            q[1]=dataset_considered['Wage'][i]
            output[j][i]=math.dist(p,q)
```

In [20]:

```
# creating an empty list to store the clusters where the points need to be fit
cluster_seg=[None]*len(output)
```

In [21]:

```
def cluster_labels():
    print("within cluster label generation")
    # updating the cluster labels for points
    for i in range(1,len(cluster_seg)+1):
        valset=output.iloc[i-1].to_list()
```

```
        cluster_seg[i-1]=valset.index(np.min(valset))
```

In [23]:

```
val_checker=pd.DataFrame(index=range(n+1),columns=range(K))
```

In [24]:

```
counter_value=np.zeros(K)
```

In [25]:

```
def centroid_updation(K):
    print("within centroid updation")
    sum_value=np.zeros(K)
    wage_value=np.zeros(K)
    counter_value=np.zeros(K)


    #updating the centroid value as per the points in the cluster
    for i in range(len(cluster_seg)):
        for j in range(K):
            if cluster_seg[i]==j :
                sum_value[j]=sum_value[j]+dataset_considered['Value'][i]
                wage_value[j]=wage_value[j]+dataset_considered['Wage'][i]
                counter_value[j]=counter_value[j]+1


    for i in range(K):
        for j in range(len(Centroids)):
                if j==0:
                    Centroids[i][j]=(sum_value[i]/counter_value[i]) #value for cluster in centroid
                else:
                    Centroids[i][j]=(wage_value[i]/counter_value[i])#wage for cluster 1 in centroid
```

In [28]:

```
def k_means(K):
    print("within kmeans calculation")
    for i in range(n_iter):
        print("itteration in k means")
        print(i)
        print("calling euclidean_distance")
        euclidean_distance()
```

```
        # creating an empty list to store the clusters where the points need to be fit
        cluster_seg=[None]*len(output)
        cluster_labels()
        centroid_updation(K)
```

In [30]:

```
#wcss calculation (elbow method) Within Cluster Sum of Squares (WCSS)
def wcss_calculation():
    print("within wcss_calculation ")
    wcss=0
    wcss_final=0
    distance=0
    counter=0
    v1=[0,0]
    v2=[0,0]


    for j in range(1):
        for i in range(len(cluster_seg)):
            if cluster_seg[i]==2:
                v1[0]=dataset_considered['Value'][i]
                v1[1]=dataset_considered['Wage'][i]
                v2[0]=Centroids[cluster_seg[i]][0]
                v2[1]=Centroids[cluster_seg[i]][1]
                distance=math.dist(v1,v2)
                distance=math.pow(distance,2)
                wcss=wcss+distance
                counter=counter+1
        if counter >0:
            wcss=wcss/counter
        else:
            wcss=0
        wcss_final=wcss_final+wcss
        counter=0
        v1=[0,0]
        v2=[0,0]
        wcss=0
    wcss_final=wcss_final
    return  wcss_final
```

In [34]:

```
wcss_final_values=[]
silhouettescore_values=[]
for i in range(1,11):# loop is just to provide with the flexibility of doing calculations for multiple k values
    #print("calling k means for time")
    print(i)
    #print("kmeans called ")
    k_means(i)
    print("calling wcss_calculation")
    wcss_result=wcss_calculation()
    #print("wcss_result obtained for times:")
    #print(i)
    silhouettescore_values.append(silhouette_score(dataset_considered, cluster_seg))
    wcss_final_values.append(wcss_result)
```

```
1
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
```

```
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
2
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
```

```
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
```

```
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
3
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
```

```
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
4
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
```

```
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
```

```
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
5
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
```

```
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
6
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
```

```
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
7
```

```
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
```

```
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
8
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
```

```
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
9
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
```

```
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
```

```
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
10
within kmeans calculation
itteration in k means
0
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
1
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
2
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
3
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
4
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
5
```

```
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
6
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
7
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
8
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
itteration in k means
9
calling euclidean_distance
within elcid dist calculation
within cluster label generation
within centroid updation
calling wcss_calculation
within wcss_calculation
```

In [35]:

```
# the silhouettescore values for K=1 ,2,3,4,5,6,7,8,9,10 are as under
silhouettescore_values
```

Out[35]:

```
[0.28151386005829737,
 0.28174167139517303,
 0.2868111714876412,
 0.2870625364201909,
 0.3970160591390798,
 0.5009924565312796,
 0.505824589332642,
 0.5202671113083313,
 0.529359709365208,
 0.5292515923417874]
```

In [36]:

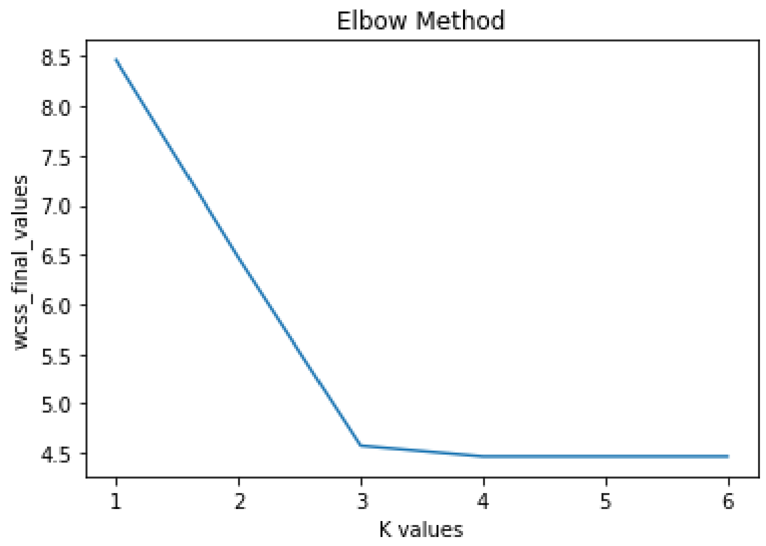
```
wcss_final_values
```

Out[36]:

```
[8.461596433566438,
 6.4708793333333325,
 4.572737801426452,
 4.463905137963228,
 4.463905137963228,
 4.463905137963228,
 6.12809866884214,
 8.100667120752984,
 8.391650772130177,
 5.721483756768014]
```

In [49]:

```
kmn=np.arange(1,7)

plt.plot(kmn,wcss_final_values[0:6] )
plt.title("Elbow Method")
plt.xlabel("K values")
plt.ylabel("wcss_final_values")
plt.show()
```



Q3 Hierarchical clustering(Agglomerative method)

In [12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import math
```

In [2]:
```
data=pd.read_csv('/content/drive/MyDrive/reformed_datset.csv')
data.head()
```

Out[2]:

	Unnamed: 0	Unnamed: 0.1	ID	Name	Age	Photo	Nationality	Flag	Overall	Potent
0	0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	
1	1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	
2	2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	
3	3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	
4	4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	

5 rows × 90 columns

In [ ]:

In [3]:
```
dataset_considered=pd.DataFrame()
dataset_considered['Value']=data['Value']
dataset_considered['Wage']=data['Wage']
```

```
#dataset_considered['Height']=data['Height']
#dataset_considered['Weight']=data['Weight']
```

In [4]:

```
import scipy.cluster.hierarchy as shc
```

In [5]:

```
#Agglomerative Clustering done for 6 clusters
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=6, affinity='euclidean', linkage='ward')
cluster.fit_predict(dataset_considered)
```

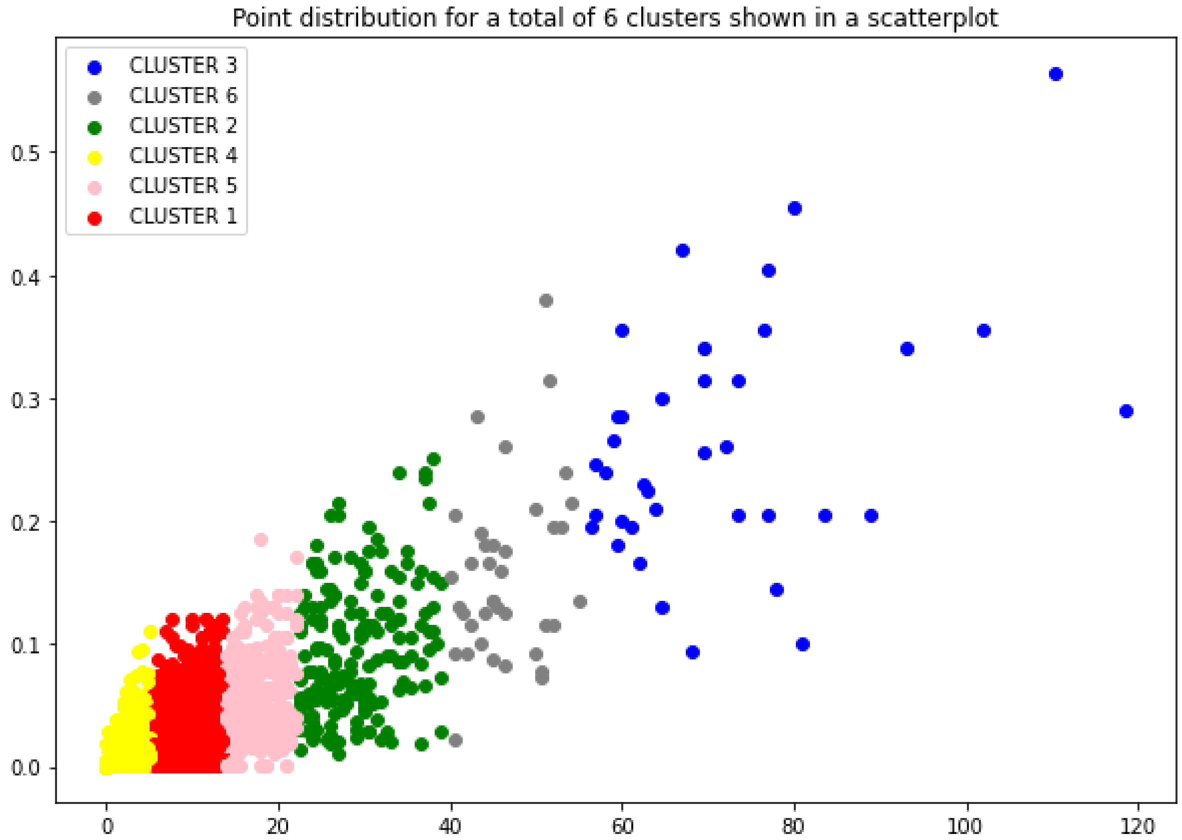
Out[5]:

```
array([2, 2, 2, ..., 3, 3, 3])
```

In [8]:

```
plt.figure(figsize=(10, 7))
c1=0
c2=0
c3=0
c4=0
c5=0
c6=0
for j in range(len(cluster.labels_)):
  if cluster.labels_[j]==0:
    if c1==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Red', label="CLUSTER 1")
      c1=c1+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Red')

  if cluster.labels_[j]==1:
    if c2==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Green' , label="CLUSTER 2")
      c2=c2+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Green')

  if cluster.labels_[j]==2:
    if c3==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Blue' , label="CLUSTER 3")
      c3=c3+1
```

```
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Blue')

  if cluster.labels_[j]==3:
    if c4==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Yellow' , label="CLUSTER 4")
      c4=c4+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Yellow')

  if cluster.labels_[j]==4:
    if c5==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Pink' , label="CLUSTER 5")
      c5=c5+1
    else:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j] ,c='Pink' )

  if cluster.labels_[j]==5:
    if c6==0:
      plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j],c='Grey',  label="CLUSTER 6")
      c6=c6+1
    else:
       plt.scatter(dataset_considered['Value'][j], dataset_considered['Wage'][j],c='Grey')

plt.title("Point distribution for a total of 6 clusters shown in a scatterplot")
plt.legend()
plt.show()
```

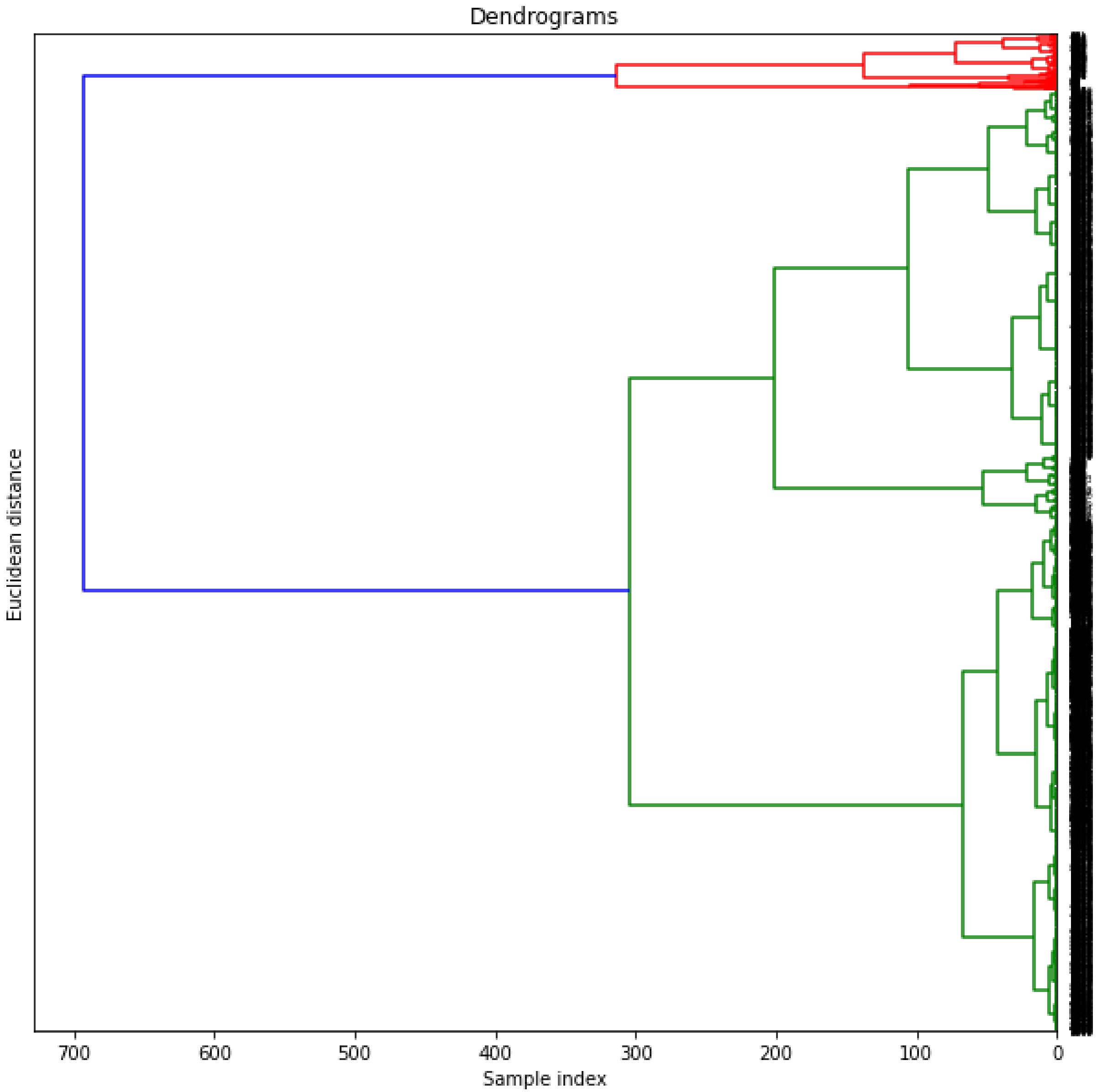


In [7]:

```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 10))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(dataset_considered, method='ward'), orientation='left',p=30)#limiting the levels at
plt.xlabel('Sample index')
plt.ylabel('Euclidean distance')
```

Out[7]:

```
Text(0, 0.5, 'Euclidean distance')
```


Dendrograms
Euclidean distance
700
600
500
400
300
200
100
0
Sample index

Q3 Hierarchical clustering (Divisive Method)

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import math
```

In [9]:
```
data=pd.read_csv('/content/drive/MyDrive/reformed_datset.csv')
data.head()
```

Out[9]:

	Unnamed: 0	Unnamed: 0.1	ID	Name	Age	Photo	Nationality	Flag	Overall	Potent
0	0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	
1	1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	
2	2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	
3	3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	
4	4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	

5 rows × 90 columns

In [11]:
```
# slicing the dataset
dataset_considered=pd.DataFrame()
dataset_considered['Value']=data['Value']
dataset_considered['Wage']=data['Wage']
```

In [14]:
```
# Couputation of distance matrix
```

```
Distance_matrix=pd.DataFrame(index=range(len(dataset_considered['Value'])),columns=range(len(dataset_considered['Wage
```

In [15]:

```
p=[0,0]
q=[0,0]
for i in range(len(Distance_matrix)):
  for j in range(len(Distance_matrix)):
    p[0]=dataset_considered['Value'][j]
    p[1]=dataset_considered['Wage'][j]
    q[0]=dataset_considered['Value'][i]
    q[1]=dataset_considered['Wage'][i]
    #Distance_matrix[i][j]=math.dist(dataset_considered['Value'][i],dataset_considered['Wage'][j])
    dist=math.sqrt(( (q[0]-p[0])**2)+((q[1]-p[1])**2))
    Distance_matrix[i][j]=dist
```

In [16]:

```
num_clusters = 0
all_elements=Distance_matrix.columns.tolist()
dissimilarity_matrix=Distance_matrix
```

In [17]:

```
# Average dissimilarity calculation within a cluster
def avg_dissimilarity_within_group_element(ele, element_list):
    max_diameter = -np.inf
    sum_dissimilarity = 0
    for i in element_list:
        sum_dissimilarity += dissimilarity_matrix[ele][i]
        if( dissimilarity_matrix[ele][i]  > max_diameter):
            max_diameter = dissimilarity_matrix[ele][i]
    if(len(element_list)>1):
        avg = sum_dissimilarity/(len(element_list)-1)
    else:
        avg = 0
    return avg
```

In [18]:

```
# calculation of average dissimalrity across clusters
def avg_dissimilarity_across_group_element(ele, main_list, spl_list):
    if len(spl_list) == 0:
        return 0
```

```
    sum_dissimilarity = 0
    for j in spl_list:
        sum_dissimilarity = sum_dissimilarity + dissimilarity_matrix[ele][j]
    avg = sum_dissimilarity/(len(spl_list))
    return avg
```

In [19]:

```
def dissimilarity_calc(main_list, dissimilarity_calc_group): #calculation of disimilarity for the matrix elements
    most_dissimilar = -np.inf
    most_dissimilar = None
    for ele in main_list:
        x = avg_dissimilarity_within_group_element(ele, main_list)
        y = avg_dissimilarity_across_group_element(ele, main_list, dissimilarity_calc_group)
        diff= x -y
        if diff > most_dissimilar:
            most_dissimilar = diff
            most_dissm_object_index = ele
    if(most_dissimilar>0):
        return  (most_dissm_object_index, 1)
    else:
        return (-1, -1)
```

In [20]:

```
def split(element_list):# splitting a cluster
    main_list = element_list
    dissimilarity_calc_group = []
    (most_dissm_object_index,flag) = dissimilarity_calc(main_list, dissimilarity_calc_group)
    while(flag > 0):
        main_list.remove(most_dissm_object_index)
        dissimilarity_calc_group.append(most_dissm_object_index)
        (most_dissm_object_index,flag) = dissimilarity_calc(element_list, dissimilarity_calc_group)

    return (main_list, dissimilarity_calc_group)
```

In [21]:

```
def max_diameter(cluster_list): # calculation of maximum diameter of a given cluster
    max_diameter_cluster_index = None
    max_diameter_cluster_value = -np.inf
    index = 0
    for element_list in cluster_list:
        for i in element_list:
```

```
            for j in element_list:
                if dissimilarity_matrix[i][j]  > max_diameter_cluster_value:
                    max_diameter_cluster_value = dissimilarity_matrix[i][j]
                    max_diameter_cluster_index = index

        index +=1

    if(max_diameter_cluster_value <= 0):
        return -1
    #print("The max diameter cluster index is{0}".format(max_diameter_cluster_index))
    return max_diameter_cluster_index
```

In [22]:

```
current_clusters = ([all_elements])
level = 1
index = 0
counterz=0# to count the total number of clusters formed
while(index!=-1):
    if counterz==5 :# Total clusters set to (5+1)=6 clusters
      break# Ending while loop when the desired cluster is achieved
    print(level, current_clusters)
    print(type(current_clusters[0]))
    counterz=counterz+1# my mod
    (sub_cluster_1, sub_cluster_2) = split(current_clusters[index])
    del current_clusters[index]
    current_clusters.append(sub_cluster_1)
    current_clusters.append(sub_cluster_2)
    index = max_diameter(current_clusters)
    level +=1


print(level, current_clusters) # printing the final cluster segregation
```

```
1 [[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 6
0, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 8
9, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114,
115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 13
8, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161,
162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 18
5, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 23
```

```
2, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255,
256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 27
9, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302,
303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 32
6, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349,
350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 37
3, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396,
397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 42
0, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443,
444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 46
7, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490,
491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 51
4, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537,
538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 56
1, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584,
585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 60
8, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631,
632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 65
5, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678,
679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 70
2, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725,
726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 74
9, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772,
773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 79
6, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819,
820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 84
3, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866,
867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 89
0, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913,
914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 93
7, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960,
961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 98
4, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000]]
<class 'list'>
2 [[24, 39, 40, 41, 46, 70, 73, 75, 76, 89, 92, 95, 96, 98, 99, 102, 103, 104, 106, 107, 108, 109, 112, 120, 123, 12
6, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149,
150, 151, 152, 153, 154, 157, 158, 159, 160, 165, 168, 170, 171, 172, 173, 174, 175, 177, 178, 180, 181, 182, 183, 18
4, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 23
1, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254,
255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 27
8, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301,
302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 32
5, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348,
349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 37
```

```
2, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395,
396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 41
9, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442,
443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 46
6, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489,
490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 51
3, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536,
537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 56
0, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583,
584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 60
7, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630,
631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 65
4, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677,
678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 70
1, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724,
725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 74
8, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771,
772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 79
5, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818,
819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 84
2, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865,
866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 88
9, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912,
913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 93
6, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959,
960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 98
3, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000], [2, 0, 4, 5, 15, 16, 25, 7,
17, 10, 1, 11, 30, 31, 3, 28, 32, 26, 9, 6, 23, 43, 45, 14, 47, 29, 55, 13, 36, 21, 27, 56, 33, 18, 38, 42, 60, 63, 7
9, 19, 61, 58, 48, 20, 8, 44, 83, 78, 62, 80, 53, 65, 52, 77, 34, 67, 87, 50, 66, 86, 59, 72, 12, 74, 68, 35, 116, 11
7, 124, 57, 156, 84, 82, 114, 155, 71, 122, 94, 100, 121, 22, 85, 49, 93, 97, 88, 91, 105, 125, 90, 64, 81, 115, 166,
37, 101, 119, 163, 176, 69, 179, 167, 54, 51, 113, 118, 161, 110, 162, 164, 169, 111]]
<class 'list'>
3 [[24, 39, 40, 41, 46, 70, 73, 75, 76, 89, 92, 95, 96, 98, 99, 102, 103, 104, 106, 107, 108, 109, 112, 120, 123, 12
6, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149,
150, 151, 152, 153, 154, 157, 158, 159, 160, 165, 168, 170, 171, 172, 173, 174, 175, 177, 178, 180, 181, 182, 183, 18
4, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 23
1, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254,
255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 27
8, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301,
302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 32
5, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348,
349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 37
2, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395,
396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 41
```

```
9, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442,
443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 46
6, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489,
490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 51
3, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536,
537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 56
0, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583,
584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 60
7, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630,
631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 65
4, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677,
678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 70
1, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724,
725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 74
8, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771,
772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 79
5, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818,
819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 84
2, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865,
866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 88
9, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912,
913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 93
6, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959,
960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 98
3, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000], [38, 42, 60, 63, 79, 19, 6
1, 58, 48, 20, 8, 44, 83, 78, 62, 80, 53, 65, 52, 77, 34, 67, 87, 50, 66, 86, 59, 72, 12, 74, 68, 35, 116, 117, 124,
57, 156, 84, 82, 114, 155, 71, 122, 94, 100, 121, 22, 85, 49, 93, 97, 88, 91, 105, 125, 90, 64, 81, 115, 166, 37, 10
1, 119, 163, 176, 69, 179, 167, 54, 51, 113, 118, 161, 110, 162, 164, 169, 111], [2, 0, 4, 5, 15, 16, 25, 7, 17, 10,
1, 11, 30, 31, 3, 26, 28, 32, 9, 6, 23, 43, 45, 14, 47, 29, 55, 36, 13, 21, 27, 56, 33, 18]]
<class 'list'>
4 [[24, 39, 40, 41, 46, 70, 73, 75, 76, 89, 92, 95, 96, 98, 99, 102, 103, 104, 106, 107, 108, 109, 112, 120, 123, 12
6, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149,
150, 151, 152, 153, 154, 157, 158, 159, 160, 165, 168, 170, 171, 172, 173, 174, 175, 177, 178, 180, 181, 182, 183, 18
4, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 23
1, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254,
255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 27
8, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301,
302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 32
5, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348,
349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 37
2, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395,
396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 41
9, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442,
443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 46
```

```
6, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489,
490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 51
3, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536,
537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 56
0, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583,
584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 60
7, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630,
631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 65
4, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677,
678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 70
1, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724,
725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 74
8, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771,
772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 79
5, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818,
819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 84
2, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865,
866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 88
9, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912,
913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 93
6, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959,
960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 98
3, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000], [38, 42, 60, 63, 79, 19, 6
1, 58, 48, 20, 8, 44, 83, 78, 62, 80, 53, 65, 52, 77, 34, 67, 87, 50, 66, 86, 59, 72, 12, 74, 68, 35, 116, 117, 124,
57, 156, 84, 82, 114, 155, 71, 122, 94, 100, 121, 22, 85, 49, 93, 97, 88, 91, 105, 125, 90, 64, 81, 115, 166, 37, 10
1, 119, 163, 176, 69, 179, 167, 54, 51, 113, 118, 161, 110, 162, 164, 169, 111], [16, 25, 7, 17, 10, 1, 11, 30, 31,
3, 26, 28, 32, 9, 6, 23, 43, 45, 14, 47, 29, 55, 36, 13, 21, 27, 56, 33, 18], [2, 0, 4, 5, 15]]
<class 'list'>
5 [[38, 42, 60, 63, 79, 19, 61, 58, 48, 20, 8, 44, 83, 78, 62, 80, 53, 65, 52, 77, 34, 67, 87, 50, 66, 86, 59, 72, 1
2, 74, 68, 35, 116, 117, 124, 57, 156, 84, 82, 114, 155, 71, 122, 94, 100, 121, 22, 85, 49, 93, 97, 88, 91, 105, 125,
90, 64, 81, 115, 166, 37, 101, 119, 163, 176, 69, 179, 167, 54, 51, 113, 118, 161, 110, 162, 164, 169, 111], [16, 25,
7, 17, 10, 1, 11, 30, 31, 3, 26, 28, 32, 9, 6, 23, 43, 45, 14, 47, 29, 55, 36, 13, 21, 27, 56, 33, 18], [2, 0, 4, 5,
15], [41, 102, 104, 108, 109, 145, 152, 153, 154, 201, 212, 213, 218, 221, 222, 223, 224, 230, 231, 232, 287, 291, 29
8, 299, 300, 301, 303, 305, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 320, 330, 338, 346, 353, 358,
359, 360, 366, 368, 369, 372, 377, 378, 383, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 396, 398, 399, 400, 40
1, 402, 403, 404, 405, 407, 408, 409, 410, 411, 419, 420, 422, 423, 425, 426, 428, 431, 433, 434, 438, 439, 443, 445,
447, 450, 451, 452, 453, 454, 455, 458, 461, 462, 463, 464, 465, 466, 467, 468, 470, 471, 472, 473, 474, 475, 477, 47
8, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 497, 498, 499, 500, 502, 503, 504, 505, 506,
507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 53
0, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553,
554, 556, 563, 564, 565, 566, 567, 568, 571, 572, 573, 574, 576, 577, 578, 580, 581, 582, 583, 584, 585, 587, 588, 59
0, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 614, 615,
616, 617, 618, 619, 620, 621, 622, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 64
0, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663,
664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 68
```

```
7, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710,
711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 73
4, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757,
758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 78
1, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804,
805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 82
8, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851,
852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 87
5, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898,
899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 92
2, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945,
946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 96
9, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992,
993, 994, 995, 996, 997, 998, 999, 1000], [99, 183, 168, 173, 184, 228, 171, 174, 112, 92, 136, 127, 185, 236, 120, 1
23, 186, 134, 98, 160, 244, 193, 205, 170, 140, 46, 40, 95, 252, 151, 178, 130, 70, 73, 128, 243, 138, 129, 263, 165,
233, 246, 227, 234, 142, 229, 177, 270, 271, 159, 182, 265, 89, 157, 137, 144, 206, 262, 319, 131, 280, 187, 226, 17
2, 24, 190, 248, 195, 321, 275, 250, 254, 192, 239, 103, 181, 253, 208, 326, 240, 268, 204, 269, 323, 96, 247, 350, 2
49, 238, 194, 199, 126, 180, 235, 148, 328, 289, 139, 241, 245, 340, 335, 334, 211, 135, 188, 200, 220, 39, 132, 143,
343, 141, 325, 189, 329, 259, 362, 415, 267, 380, 302, 146, 292, 281, 257, 337, 332, 351, 414, 158, 256, 272, 345, 21
9, 294, 284, 364, 107, 361, 133, 347, 413, 336, 282, 106, 278, 297, 355, 255, 76, 237, 371, 283, 198, 327, 395, 260,
376, 203, 322, 357, 264, 197, 196, 273, 416, 421, 295, 274, 286, 288, 258, 331, 429, 412, 342, 304, 430, 324, 352, 25
1, 215, 216, 427, 437, 440, 339, 435, 441, 210, 214, 242, 175, 293, 444, 460, 481, 448, 449, 367, 344, 446, 279, 277,
442, 225, 562, 418, 370, 202, 569, 149, 191, 341, 207, 147, 456, 306, 417, 290, 348, 333, 469, 285, 261, 349, 501, 37
5, 436, 276, 381, 75, 266, 296, 150, 560, 356, 363, 354, 561, 382, 589, 559, 570, 480, 557, 575, 558, 555, 374, 209,
613, 457, 384, 432, 379, 217, 496, 623, 373, 476, 459, 424, 579, 586, 365, 479, 406, 397, 602]]
<class 'list'>
6 [[38, 42, 60, 63, 79, 19, 61, 58, 48, 20, 8, 44, 83, 78, 62, 80, 53, 65, 52, 77, 34, 67, 87, 50, 66, 86, 59, 72, 1
2, 74, 68, 35, 116, 117, 124, 57, 156, 84, 82, 114, 155, 71, 122, 94, 100, 121, 22, 85, 49, 93, 97, 88, 91, 105, 125,
90, 64, 81, 115, 166, 37, 101, 119, 163, 176, 69, 179, 167, 54, 51, 113, 118, 161, 110, 162, 164, 169, 111], [16, 25,
7, 17, 10, 1, 11, 30, 31, 3, 26, 28, 32, 9, 6, 23, 43, 45, 14, 47, 29, 55, 36, 13, 21, 27, 56, 33, 18], [41, 102, 10
4, 108, 109, 145, 152, 153, 154, 201, 212, 213, 218, 221, 222, 223, 224, 230, 231, 232, 287, 291, 298, 299, 300, 301,
303, 305, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 320, 330, 338, 346, 353, 358, 359, 360, 366, 36
8, 369, 372, 377, 378, 383, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 396, 398, 399, 400, 401, 402, 403, 404,
405, 407, 408, 409, 410, 411, 419, 420, 422, 423, 425, 426, 428, 431, 433, 434, 438, 439, 443, 445, 447, 450, 451, 45
2, 453, 454, 455, 458, 461, 462, 463, 464, 465, 466, 467, 468, 470, 471, 472, 473, 474, 475, 477, 478, 482, 483, 484,
485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 497, 498, 499, 500, 502, 503, 504, 505, 506, 507, 508, 509, 51
0, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533,
534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 556, 563, 56
4, 565, 566, 567, 568, 571, 572, 573, 574, 576, 577, 578, 580, 581, 582, 583, 584, 585, 587, 588, 590, 591, 592, 593,
594, 595, 596, 597, 598, 599, 600, 601, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 614, 615, 616, 617, 618, 61
9, 620, 621, 622, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643,
644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 66
7, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690,
691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 71
4, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737,
```

```
738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 76
1, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784,
785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 80
8, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831,
832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 85
5, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878,
879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 90
2, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925,
926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 94
9, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972,
973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 99
6, 997, 998, 999, 1000], [99, 183, 168, 173, 184, 228, 171, 174, 112, 92, 136, 127, 185, 236, 120, 123, 186, 134, 98,
160, 244, 193, 205, 170, 140, 46, 40, 95, 252, 151, 178, 130, 70, 73, 128, 243, 138, 129, 263, 165, 233, 246, 227, 23
4, 142, 229, 177, 270, 271, 159, 182, 265, 89, 157, 137, 144, 206, 262, 319, 131, 280, 187, 226, 172, 24, 190, 248, 1
95, 321, 275, 250, 254, 192, 239, 103, 181, 253, 208, 326, 240, 268, 204, 269, 323, 96, 247, 350, 249, 238, 194, 199,
126, 180, 235, 148, 328, 289, 139, 241, 245, 340, 335, 334, 211, 135, 188, 200, 220, 39, 132, 143, 343, 141, 325, 18
9, 329, 259, 362, 415, 267, 380, 302, 146, 292, 281, 257, 337, 332, 351, 414, 158, 256, 272, 345, 219, 294, 284, 364,
107, 361, 133, 347, 413, 336, 282, 106, 278, 297, 355, 255, 76, 237, 371, 283, 198, 327, 395, 260, 376, 203, 322, 35
7, 264, 197, 196, 273, 416, 421, 295, 274, 286, 288, 258, 331, 429, 412, 342, 304, 430, 324, 352, 251, 215, 216, 427,
437, 440, 339, 435, 441, 210, 214, 242, 175, 293, 444, 460, 481, 448, 449, 367, 344, 446, 279, 277, 442, 225, 562, 41
8, 370, 202, 569, 149, 191, 341, 207, 147, 456, 306, 417, 290, 348, 333, 469, 285, 261, 349, 501, 375, 436, 276, 381,
75, 266, 296, 150, 560, 356, 363, 354, 561, 382, 589, 559, 570, 480, 557, 575, 558, 555, 374, 209, 613, 457, 384, 43
2, 379, 217, 496, 623, 373, 476, 459, 424, 579, 586, 365, 479, 406, 397, 602], [4, 5, 15], [2, 0]]
```

In [23]:

```
value_cluster_1_points=[]
value_cluster_2_points=[]
value_cluster_3_points=[]
value_cluster_4_points=[]
value_cluster_5_points=[]
value_cluster_6_points=[]

wage_cluster_1_points=[]
wage_cluster_2_points=[]
wage_cluster_3_points=[]
wage_cluster_4_points=[]
wage_cluster_5_points=[]
wage_cluster_6_points=[]
```

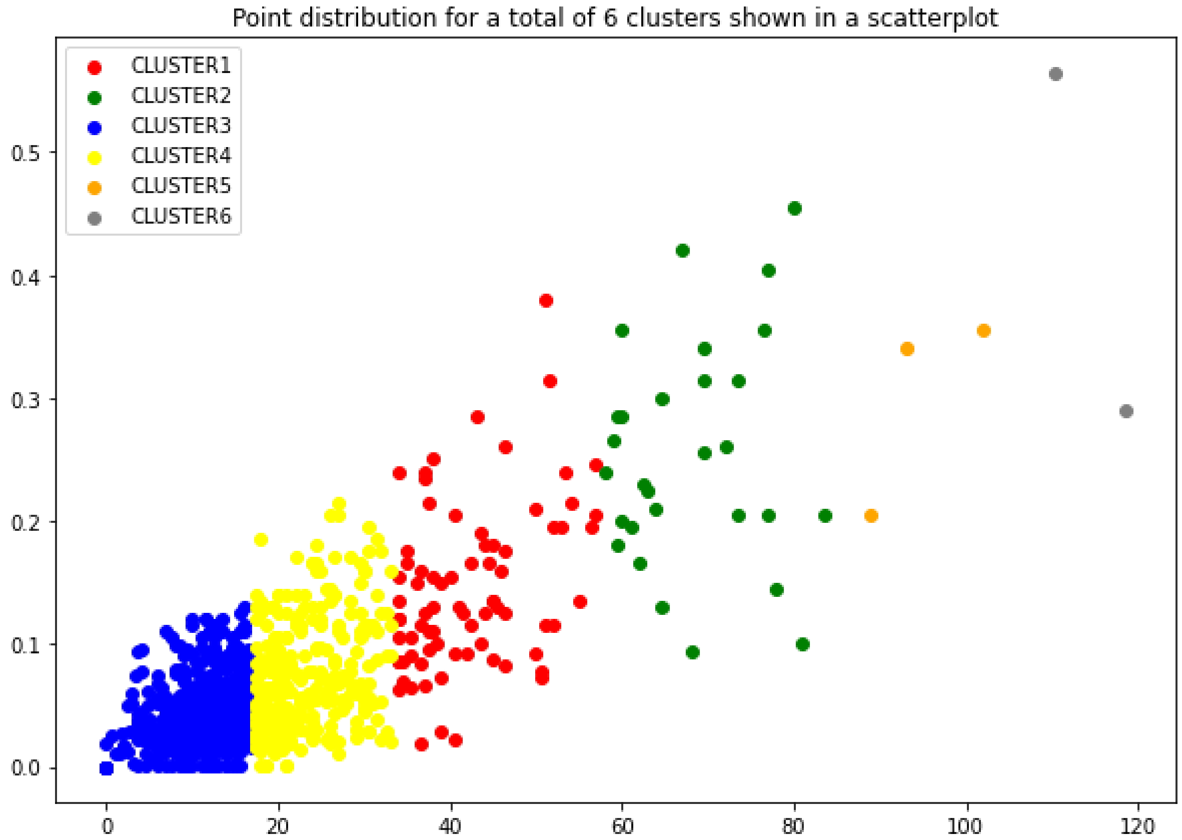
In [24]:

```
for i in range(len(current_clusters[0])):
    value_cluster_1_points.append(dataset_considered['Value'][current_clusters[0][i]])
    wage_cluster_1_points.append(dataset_considered['Wage'][current_clusters[0][i]])
```

```
for i in range(len(current_clusters[1])):
    value_cluster_2_points.append(dataset_considered['Value'][current_clusters[1][i]])
    wage_cluster_2_points.append(dataset_considered['Wage'][current_clusters[1][i]])

for i in range(len(current_clusters[2])):
    value_cluster_3_points.append(dataset_considered['Value'][current_clusters[2][i]])
    wage_cluster_3_points.append(dataset_considered['Wage'][current_clusters[2][i]])

for i in range(len(current_clusters[3])):
    value_cluster_4_points.append(dataset_considered['Value'][current_clusters[3][i]])
    wage_cluster_4_points.append(dataset_considered['Wage'][current_clusters[3][i]])


for i in range(len(current_clusters[4])):
    value_cluster_5_points.append(dataset_considered['Value'][current_clusters[4][i]])
    wage_cluster_5_points.append(dataset_considered['Wage'][current_clusters[4][i]])


for i in range(len(current_clusters[5])):
    value_cluster_6_points.append(dataset_considered['Value'][current_clusters[5][i]])
    wage_cluster_6_points.append(dataset_considered['Wage'][current_clusters[5][i]])
```

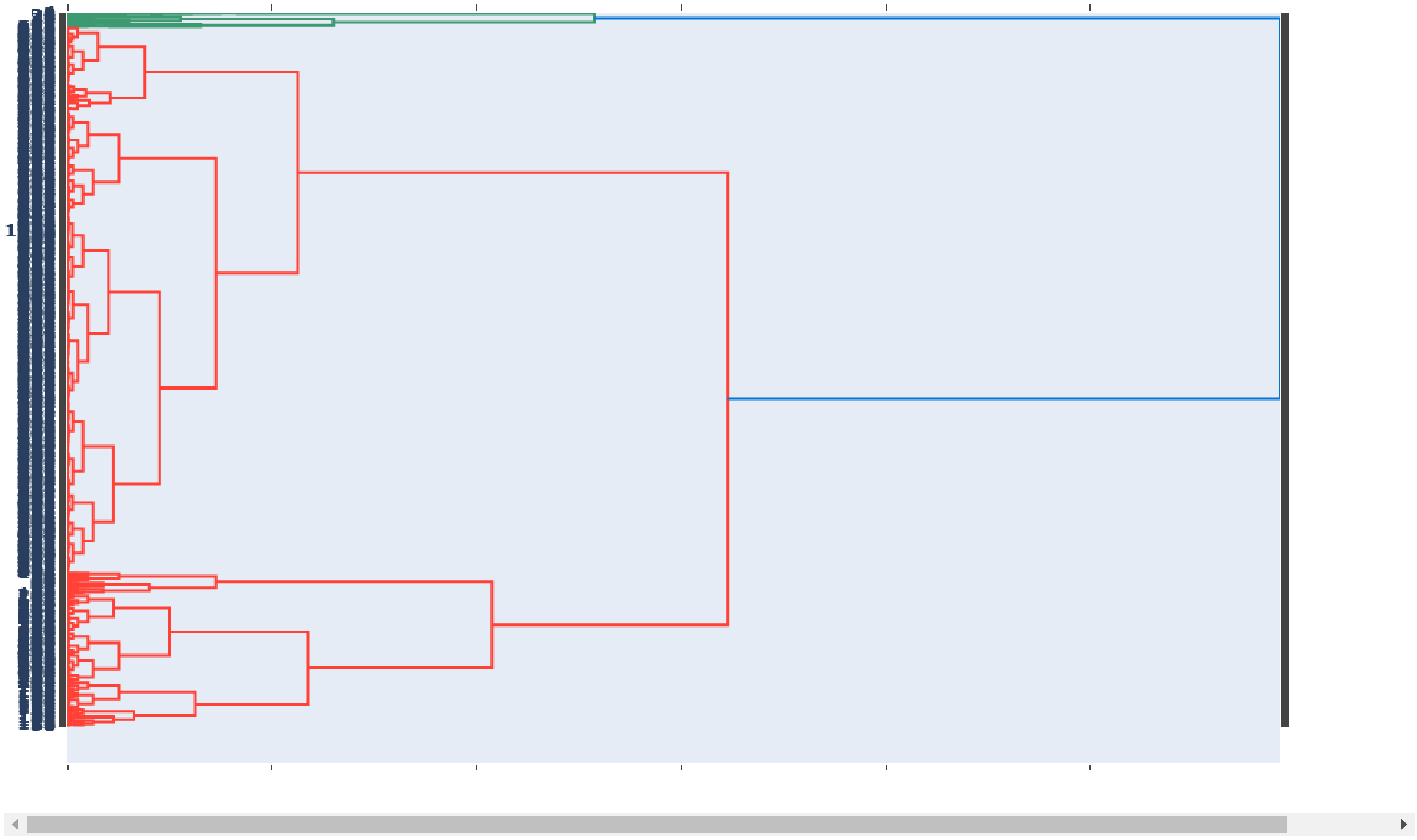
In [26]:

```
plt.figure(figsize=(10, 7))

plt.scatter(value_cluster_1_points, wage_cluster_1_points ,c='Red')
plt.scatter(value_cluster_2_points, wage_cluster_2_points ,c='Green')
plt.scatter(value_cluster_3_points, wage_cluster_3_points ,c='Blue')
plt.scatter(value_cluster_4_points, wage_cluster_4_points ,c='Yellow')
plt.scatter(value_cluster_5_points, wage_cluster_5_points ,c='Orange')
plt.scatter(value_cluster_6_points, wage_cluster_6_points ,c='Grey')


plt.title("Point distribution for a total of 6 clusters shown in a scatterplot")
plt.legend(["CLUSTER1","CLUSTER2","CLUSTER3","CLUSTER4","CLUSTER5","CLUSTER6"])
plt.show()
```



In [37]:

```
import plotly.figure_factory as ff
fig = ff.create_dendrogram(dataset_considered, orientation='left')
fig.update_layout(width=1000, height=700)
fig.show()
```


1