QUESTION 1

In [ ]:

SOLUTION

In [ ]:

Step 1

In [232]:

*#importing the necessary libararies*

**import** pandas **as** pd

**import** numpy **as** np

In [ ]:

Step 2 (*# Import the dataset from this address)*

In [233]:

url **=** 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user'

In [ ]:

Step 3 (*#Assign it to a variable called users)*

In [234]:

users **=** pd**.**read\_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user', sep **=** '|')

In [235]:

users**.**head()

Out[235]:

|  | **user\_id** | **age** | **gender** | **occupation** | **zip\_code** |
| --- | --- | --- | --- | --- | --- |
| **0** | 1 | 24 | M | technician | 85711 |
| **1** | 2 | 53 | F | other | 94043 |
| **2** | 3 | 23 | M | writer | 32067 |
| **3** | 4 | 24 | M | technician | 43537 |
| **4** | 5 | 33 | F | other | 15213 |

In [ ]:

Step 4 (*#Discover what is the mean age per occupation)*

In [236]:

users**.**groupby('occupation')['age']**.**mean()

Out[236]:

occupation

administrator 38.746835

artist 31.392857

doctor 43.571429

educator 42.010526

engineer 36.388060

entertainment 29.222222

executive 38.718750

healthcare 41.562500

homemaker 32.571429

lawyer 36.750000

librarian 40.000000

marketing 37.615385

none 26.555556

other 34.523810

programmer 33.121212

retired 63.071429

salesman 35.666667

scientist 35.548387

student 22.081633

technician 33.148148

writer 36.311111

Name: age, dtype: float64

In [ ]:

Step 5 (*#Discover the Male ratio per occupation and sort it from the most to the least)*

In [237]:

users**.**gender**.**apply(**lambda** X: **True** **if** X **==**'M' **else** **False**)

Out[237]:

0 True

1 False

2 True

3 True

4 False

...

938 False

939 True

940 True

941 False

942 True

Name: gender, Length: 943, dtype: bool

In [ ]:

(users**.**groupby('occupation')**.**is\_male**.**sum() **/** users**.**groupby('occupation')**.**gender**.**count())**.**sort\_values(ascending **=** **False**)

In [ ]:

Step 6 (*#For each occupation, calculate the minimum and maximum ages)*

In [226]:

print(users**.**groupby('occupation')**.**age**.**min())

print(users**.**groupby('occupation')**.**age**.**max())

occupation

administrator 21

artist 19

doctor 28

educator 23

engineer 22

entertainment 15

executive 22

healthcare 22

homemaker 20

lawyer 21

librarian 23

marketing 24

none 11

other 13

programmer 20

retired 51

salesman 18

scientist 23

student 7

technician 21

writer 18

Name: age, dtype: int64

occupation

administrator 70

artist 48

doctor 64

educator 63

engineer 70

entertainment 50

executive 69

healthcare 62

homemaker 50

lawyer 53

librarian 69

marketing 55

none 55

other 64

programmer 63

retired 73

salesman 66

scientist 55

student 42

technician 55

writer 60

Name: age, dtype: int64

In [227]:

users**.**groupby('occupation')**.**age**.**agg(['min', 'max'])

Out[227]:

|  | **min** | **max** |
| --- | --- | --- |
| **occupation** |  |  |
| **administrator** | 21 | 70 |
| **artist** | 19 | 48 |
| **doctor** | 28 | 64 |
| **educator** | 23 | 63 |
| **engineer** | 22 | 70 |
| **entertainment** | 15 | 50 |
| **executive** | 22 | 69 |
| **healthcare** | 22 | 62 |
| **homemaker** | 20 | 50 |
| **lawyer** | 21 | 53 |
| **librarian** | 23 | 69 |
| **marketing** | 24 | 55 |
| **none** | 11 | 55 |
| **other** | 13 | 64 |
| **programmer** | 20 | 63 |
| **retired** | 51 | 73 |
| **salesman** | 18 | 66 |
| **scientist** | 23 | 55 |
| **student** | 7 | 42 |
| **technician** | 21 | 55 |
| **writer** | 18 | 60 |

In [ ]:

Step 7 (*#For each combination of occupation and sex, calculate the mean age)*

In [239]:

users**.**groupby(['occupation' , 'gender'])**.**age**.**mean()

Out[239]:

occupation gender

administrator F 40.638889

M 37.162791

artist F 30.307692

M 32.333333

doctor M 43.571429

educator F 39.115385

M 43.101449

engineer F 29.500000

M 36.600000

entertainment F 31.000000

M 29.000000

executive F 44.000000

M 38.172414

healthcare F 39.818182

M 45.400000

homemaker F 34.166667

M 23.000000

lawyer F 39.500000

M 36.200000

librarian F 40.000000

M 40.000000

marketing F 37.200000

M 37.875000

none F 36.500000

M 18.600000

other F 35.472222

M 34.028986

programmer F 32.166667

M 33.216667

retired F 70.000000

M 62.538462

salesman F 27.000000

M 38.555556

scientist F 28.333333

M 36.321429

student F 20.750000

M 22.669118

technician F 38.000000

M 32.961538

writer F 37.631579

M 35.346154

Name: age, dtype: float64

In [ ]:

Step 8 (*#For each occupation present the percentage of women and men)*

In [241]:

*#I will first create a dataframe and apply count to gender*

gender\_occup **=** users**.**groupby(['occupation' , 'gender'])**.**agg({'gender' : 'count'})

In [242]:

*#next , i will create another dataframe and apply count for each occupation*

occup\_count **=** users**.**groupby(['occupation'])**.**count()

In [243]:

*#finally, i now divide gender\_occup by the occup\_count and multiply by 100*

occup\_gender **=** gender\_occup**.**div(occup\_count , level **=** "occupation")

occup\_gender**.**loc[:, 'gender']

Out[243]:

occupation gender

administrator F 0.455696

M 0.544304

artist F 0.464286

M 0.535714

doctor M 1.000000

educator F 0.273684

M 0.726316

engineer F 0.029851

M 0.970149

entertainment F 0.111111

M 0.888889

executive F 0.093750

M 0.906250

healthcare F 0.687500

M 0.312500

homemaker F 0.857143

M 0.142857

lawyer F 0.166667

M 0.833333

librarian F 0.568627

M 0.431373

marketing F 0.384615

M 0.615385

none F 0.444444

M 0.555556

other F 0.342857

M 0.657143

programmer F 0.090909

M 0.909091

retired F 0.071429

M 0.928571

salesman F 0.250000

M 0.750000

scientist F 0.096774

M 0.903226

student F 0.306122

M 0.693878

technician F 0.037037

M 0.962963

writer F 0.422222

M 0.577778

Name: gender, dtype: float64

In [ ]:

In [ ]:

QUESTION 2

In [ ]:

SOLUTION

In [ ]:

Step 1(*#import the necessary library)*

In [244]:

**import** pandas **as** pd

In [ ]:

Step 2 (*#Import the dataset from this address)*

In [245]:

url **=** 'https://raw.githubusercontent.com/guipsamora/pandas\_exercises/master/02\_Filtering\_%26\_Sorting/Euro12/Euro\_2012\_stats\_TEAM.csv'

In [ ]:

Step 3 (*#. Assign it to a variable called euro12)*

In [246]:

euro12 **=** pd**.**read\_csv('https://raw.githubusercontent.com/guipsamora/pandas\_exercises/master/02\_Filtering\_%26\_Sorting/Euro12/Euro\_2012\_stats\_TEAM.csv')

In [230]:

euro12**.**head()

Out[230]:

|  | **Team** | **Goals** | **Shots on target** | **Shots off target** | **Shooting Accuracy** | **% Goals-to-shots** | **Total shots (inc. Blocked)** | **Hit Woodwork** | **Penalty goals** | **Penalties not scored** | **...** | **Saves made** | **Saves-to-shots ratio** | **Fouls Won** | **Fouls Conceded** | **Offsides** | **Yellow Cards** | **Red Cards** | **Subs on** | **Subs off** | **Players Used** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Croatia | 4 | 13 | 12 | 51.9% | 16.0% | 32 | 0 | 0 | 0 | ... | 13 | 81.3% | 41 | 62 | 2 | 9 | 0 | 9 | 9 | 16 |
| **1** | Czech Republic | 4 | 13 | 18 | 41.9% | 12.9% | 39 | 0 | 0 | 0 | ... | 9 | 60.1% | 53 | 73 | 8 | 7 | 0 | 11 | 11 | 19 |
| **2** | Denmark | 4 | 10 | 10 | 50.0% | 20.0% | 27 | 1 | 0 | 0 | ... | 10 | 66.7% | 25 | 38 | 8 | 4 | 0 | 7 | 7 | 15 |
| **3** | England | 5 | 11 | 18 | 50.0% | 17.2% | 40 | 0 | 0 | 0 | ... | 22 | 88.1% | 43 | 45 | 6 | 5 | 0 | 11 | 11 | 16 |
| **4** | France | 3 | 22 | 24 | 37.9% | 6.5% | 65 | 1 | 0 | 0 | ... | 6 | 54.6% | 36 | 51 | 5 | 6 | 0 | 11 | 11 | 19 |

5 rows × 35 columns

In [ ]:

Step 4 (*#Select only the Goal column)*

In [247]:

euro12['Goals']

Out[247]:

0 4

1 4

2 4

3 5

4 3

5 10

6 5

7 6

8 2

9 2

10 6

11 1

12 5

13 12

14 5

15 2

Name: Goals, dtype: int64

In [ ]:

Step 5 (*#How many team participated in the Euro2012?)*

In [249]:

len(euro12['Team']**.**unique())

Out[249]:

16

In [ ]:

Step 6 (*#What is the number of columns in the dataset?)*

In [250]:

len(euro12**.**columns)

Out[250]:

35

In [ ]:

Step 7 (*#View only the columns Team, Yellow Cards and Red Cards and assign them to a dataframe called discipline)*

In [251]:

discipline **=** euro12[['Team','Yellow Cards','Red Cards']]

In [252]:

discipline**.**head()

Out[252]:

|  | **Team** | **Yellow Cards** | **Red Cards** |
| --- | --- | --- | --- |
| **0** | Croatia | 9 | 0 |
| **1** | Czech Republic | 7 | 0 |
| **2** | Denmark | 4 | 0 |
| **3** | England | 5 | 0 |
| **4** | France | 6 | 0 |

In [ ]:

Step 8 (*#Sort the teams by Red Cards, then to Yellow Cards)*

In [253]:

discipline**.**sort\_values(['Red Cards','Yellow Cards'], ascending**=**[**True**, **True**])

Out[253]:

|  | **Team** | **Yellow Cards** | **Red Cards** |
| --- | --- | --- | --- |
| **2** | Denmark | 4 | 0 |
| **5** | Germany | 4 | 0 |
| **3** | England | 5 | 0 |
| **8** | Netherlands | 5 | 0 |
| **15** | Ukraine | 5 | 0 |
| **4** | France | 6 | 0 |
| **12** | Russia | 6 | 0 |
| **1** | Czech Republic | 7 | 0 |
| **14** | Sweden | 7 | 0 |
| **0** | Croatia | 9 | 0 |
| **13** | Spain | 11 | 0 |
| **10** | Portugal | 12 | 0 |
| **7** | Italy | 16 | 0 |
| **11** | Republic of Ireland | 6 | 1 |
| **9** | Poland | 7 | 1 |
| **6** | Greece | 9 | 1 |

In [ ]:

Step 9 (*#Calculate the mean Yellow Cards given per Team)*

In [254]:

discipline**.**groupby('Team')['Yellow Cards']**.**mean()

Out[254]:

Team

Croatia 9.0

Czech Republic 7.0

Denmark 4.0

England 5.0

France 6.0

Germany 4.0

Greece 9.0

Italy 16.0

Netherlands 5.0

Poland 7.0

Portugal 12.0

Republic of Ireland 6.0

Russia 6.0

Spain 11.0

Sweden 7.0

Ukraine 5.0

Name: Yellow Cards, dtype: float64

In [ ]:

Step 10 (*#Filter teams that scored more than 6 goals)*

In [255]:

euro12[euro12**.**Goals **>** 6]

Out[255]:

|  | **Team** | **Goals** | **Shots on target** | **Shots off target** | **Shooting Accuracy** | **% Goals-to-shots** | **Total shots (inc. Blocked)** | **Hit Woodwork** | **Penalty goals** | **Penalties not scored** | **...** | **Saves made** | **Saves-to-shots ratio** | **Fouls Won** | **Fouls Conceded** | **Offsides** | **Yellow Cards** | **Red Cards** | **Subs on** | **Subs off** | **Players Used** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | Germany | 10 | 32 | 32 | 47.8% | 15.6% | 80 | 2 | 1 | 0 | ... | 10 | 62.6% | 63 | 49 | 12 | 4 | 0 | 15 | 15 | 17 |
| **13** | Spain | 12 | 42 | 33 | 55.9% | 16.0% | 100 | 0 | 1 | 0 | ... | 15 | 93.8% | 102 | 83 | 19 | 11 | 0 | 17 | 17 | 18 |

2 rows × 35 columns

In [ ]:

Step 11 (*#Select the teams that start with G)*

In [256]:

euro12[euro12**.**Team**.**str[0] **==** 'G']

Out[256]:

|  | **Team** | **Goals** | **Shots on target** | **Shots off target** | **Shooting Accuracy** | **% Goals-to-shots** | **Total shots (inc. Blocked)** | **Hit Woodwork** | **Penalty goals** | **Penalties not scored** | **...** | **Saves made** | **Saves-to-shots ratio** | **Fouls Won** | **Fouls Conceded** | **Offsides** | **Yellow Cards** | **Red Cards** | **Subs on** | **Subs off** | **Players Used** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | Germany | 10 | 32 | 32 | 47.8% | 15.6% | 80 | 2 | 1 | 0 | ... | 10 | 62.6% | 63 | 49 | 12 | 4 | 0 | 15 | 15 | 17 |
| **6** | Greece | 5 | 8 | 18 | 30.7% | 19.2% | 32 | 1 | 1 | 1 | ... | 13 | 65.1% | 67 | 48 | 12 | 9 | 1 | 12 | 12 | 20 |

2 rows × 35 columns

In [ ]:

Step 12 (*# Select the first 7 columns)*

In [257]:

euro12**.**iloc[: , 0:7]

Out[257]:

|  | **Team** | **Goals** | **Shots on target** | **Shots off target** | **Shooting Accuracy** | **% Goals-to-shots** | **Total shots (inc. Blocked)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Croatia | 4 | 13 | 12 | 51.9% | 16.0% | 32 |
| **1** | Czech Republic | 4 | 13 | 18 | 41.9% | 12.9% | 39 |
| **2** | Denmark | 4 | 10 | 10 | 50.0% | 20.0% | 27 |
| **3** | England | 5 | 11 | 18 | 50.0% | 17.2% | 40 |
| **4** | France | 3 | 22 | 24 | 37.9% | 6.5% | 65 |
| **5** | Germany | 10 | 32 | 32 | 47.8% | 15.6% | 80 |
| **6** | Greece | 5 | 8 | 18 | 30.7% | 19.2% | 32 |
| **7** | Italy | 6 | 34 | 45 | 43.0% | 7.5% | 110 |
| **8** | Netherlands | 2 | 12 | 36 | 25.0% | 4.1% | 60 |
| **9** | Poland | 2 | 15 | 23 | 39.4% | 5.2% | 48 |
| **10** | Portugal | 6 | 22 | 42 | 34.3% | 9.3% | 82 |
| **11** | Republic of Ireland | 1 | 7 | 12 | 36.8% | 5.2% | 28 |
| **12** | Russia | 5 | 9 | 31 | 22.5% | 12.5% | 59 |
| **13** | Spain | 12 | 42 | 33 | 55.9% | 16.0% | 100 |
| **14** | Sweden | 5 | 17 | 19 | 47.2% | 13.8% | 39 |
| **15** | Ukraine | 2 | 7 | 26 | 21.2% | 6.0% | 38 |

In [ ]:

Step 13 (*#Select all columns except the last 3)*

In [258]:

euro12**.**iloc[: , :**-**3]

Out[258]:

|  | **Team** | **Goals** | **Shots on target** | **Shots off target** | **Shooting Accuracy** | **% Goals-to-shots** | **Total shots (inc. Blocked)** | **Hit Woodwork** | **Penalty goals** | **Penalties not scored** | **...** | **Clean Sheets** | **Blocks** | **Goals conceded** | **Saves made** | **Saves-to-shots ratio** | **Fouls Won** | **Fouls Conceded** | **Offsides** | **Yellow Cards** | **Red Cards** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Croatia | 4 | 13 | 12 | 51.9% | 16.0% | 32 | 0 | 0 | 0 | ... | 0 | 10 | 3 | 13 | 81.3% | 41 | 62 | 2 | 9 | 0 |
| **1** | Czech Republic | 4 | 13 | 18 | 41.9% | 12.9% | 39 | 0 | 0 | 0 | ... | 1 | 10 | 6 | 9 | 60.1% | 53 | 73 | 8 | 7 | 0 |
| **2** | Denmark | 4 | 10 | 10 | 50.0% | 20.0% | 27 | 1 | 0 | 0 | ... | 1 | 10 | 5 | 10 | 66.7% | 25 | 38 | 8 | 4 | 0 |
| **3** | England | 5 | 11 | 18 | 50.0% | 17.2% | 40 | 0 | 0 | 0 | ... | 2 | 29 | 3 | 22 | 88.1% | 43 | 45 | 6 | 5 | 0 |
| **4** | France | 3 | 22 | 24 | 37.9% | 6.5% | 65 | 1 | 0 | 0 | ... | 1 | 7 | 5 | 6 | 54.6% | 36 | 51 | 5 | 6 | 0 |
| **5** | Germany | 10 | 32 | 32 | 47.8% | 15.6% | 80 | 2 | 1 | 0 | ... | 1 | 11 | 6 | 10 | 62.6% | 63 | 49 | 12 | 4 | 0 |
| **6** | Greece | 5 | 8 | 18 | 30.7% | 19.2% | 32 | 1 | 1 | 1 | ... | 1 | 23 | 7 | 13 | 65.1% | 67 | 48 | 12 | 9 | 1 |
| **7** | Italy | 6 | 34 | 45 | 43.0% | 7.5% | 110 | 2 | 0 | 0 | ... | 2 | 18 | 7 | 20 | 74.1% | 101 | 89 | 16 | 16 | 0 |
| **8** | Netherlands | 2 | 12 | 36 | 25.0% | 4.1% | 60 | 2 | 0 | 0 | ... | 0 | 9 | 5 | 12 | 70.6% | 35 | 30 | 3 | 5 | 0 |
| **9** | Poland | 2 | 15 | 23 | 39.4% | 5.2% | 48 | 0 | 0 | 0 | ... | 0 | 8 | 3 | 6 | 66.7% | 48 | 56 | 3 | 7 | 1 |
| **10** | Portugal | 6 | 22 | 42 | 34.3% | 9.3% | 82 | 6 | 0 | 0 | ... | 2 | 11 | 4 | 10 | 71.5% | 73 | 90 | 10 | 12 | 0 |
| **11** | Republic of Ireland | 1 | 7 | 12 | 36.8% | 5.2% | 28 | 0 | 0 | 0 | ... | 0 | 23 | 9 | 17 | 65.4% | 43 | 51 | 11 | 6 | 1 |
| **12** | Russia | 5 | 9 | 31 | 22.5% | 12.5% | 59 | 2 | 0 | 0 | ... | 0 | 8 | 3 | 10 | 77.0% | 34 | 43 | 4 | 6 | 0 |
| **13** | Spain | 12 | 42 | 33 | 55.9% | 16.0% | 100 | 0 | 1 | 0 | ... | 5 | 8 | 1 | 15 | 93.8% | 102 | 83 | 19 | 11 | 0 |
| **14** | Sweden | 5 | 17 | 19 | 47.2% | 13.8% | 39 | 3 | 0 | 0 | ... | 1 | 12 | 5 | 8 | 61.6% | 35 | 51 | 7 | 7 | 0 |
| **15** | Ukraine | 2 | 7 | 26 | 21.2% | 6.0% | 38 | 0 | 0 | 0 | ... | 0 | 4 | 4 | 13 | 76.5% | 48 | 31 | 4 | 5 | 0 |

16 rows × 32 columns

In [ ]:

Step 14 (*#Present only the Shooting Accuracy from England, Italy and Russia)*

In [259]:

euro12**.**loc[[3,7,12] , ['Team','Shooting Accuracy']]

Out[259]:

|  | **Team** | **Shooting Accuracy** |
| --- | --- | --- |
| **3** | England | 50.0% |
| **7** | Italy | 43.0% |
| **12** | Russia | 22.5% |

In [ ]:

QUESTION 3

In [ ]:

SOLUTION

In [ ]:

Step 1 (*#Import the necessary libraries)*

In [260]:

**import** pandas **as** pd

**import** numpy **as** np

In [ ]:

Step 2(*#Create 3 differents Series, each of length 100, as follows:*

The first a random number **from** 1 to 4

The second a random number **from** 1 to 3

The third a random number **from** 10,000 to 30,000)

In [261]:

*#Series 1 -The first a random number from 1 to 4*

s1 **=** pd**.**Series(np**.**random**.**randint(1,5, size**=**(100)))

In [262]:

s1**.**head()

Out[262]:

0 4

1 3

2 1

3 4

4 3

dtype: int32

In [ ]:

*#Series 2 - The second a random number from 1 to 3*

s2 **=** pd**.**Series(np**.**random**.**randint(1,4, size**=**(100)))

In [263]:

s2**.**head()

Out[263]:

0 1

1 1

2 2

3 3

4 1

dtype: int32

In [ ]:

*#Seies 3 - The third a random number from 10,000 to 30,000)*

s3 **=** pd**.**Series(np**.**random**.**randint(10000,30001, size**=**(100)))

In [264]:

s3**.**head()

Out[264]:

0 20035

1 12185

2 23909

3 11074

4 26981

dtype: int32

In [ ]:

Step 3 (*#Create a DataFrame by joinning the Series by column)*

In [265]:

df **=** pd**.**DataFrame({'s1':s1,'s2':s2,'s3':s3})

In [266]:

df**.**head()

Out[266]:

|  | **s1** | **s2** | **s3** |
| --- | --- | --- | --- |
| **0** | 4 | 1 | 20035 |
| **1** | 3 | 1 | 12185 |
| **2** | 1 | 2 | 23909 |
| **3** | 4 | 3 | 11074 |
| **4** | 3 | 1 | 26981 |

In [ ]:

Step 4 (*#Change the name of the columns to bedrs, bathrs, price\_sqr\_meter)*

In [267]:

df**.**rename(columns **=** {'s1':'bedrs', 's2':'bathrs', 's3':'price\_sqr\_meter'}, inplace **=** **True**)

In [268]:

df**.**head()

Out[268]:

|  | **bedrs** | **bathrs** | **price\_sqr\_meter** |
| --- | --- | --- | --- |
| **0** | 4 | 1 | 20035 |
| **1** | 3 | 1 | 12185 |
| **2** | 1 | 2 | 23909 |
| **3** | 4 | 3 | 11074 |
| **4** | 3 | 1 | 26981 |

In [ ]:

Step 5 (*#Create a one column DataFrame with the values of the 3 Series and assign it to 'bigcolumn' , lets call it new dataframe nd)*

In [269]:

nd **=** pd**.**DataFrame({'bigcolumn': pd**.**concat([s1, s2, s3])})

In [270]:

nd**.**head()

Out[270]:

|  | **bigcolumn** |
| --- | --- |
| **0** | 4 |
| **1** | 3 |
| **2** | 1 |
| **3** | 4 |
| **4** | 3 |

In [ ]:

Step 6 (*#Ops it seems it is going only until index 99. Is it true?)*

In [ ]:

yes it **is** true

In [ ]:

Step 7(*#Reindex the DataFrame so it goes from 0 to 299)*

In [271]:

nd**.**index **=** pd**.**RangeIndex(start**=**0, stop**=**300)

In [272]:

pd**.**RangeIndex(start**=**0, stop**=**300)

Out[272]:

RangeIndex(start=0, stop=300, step=1)

In [ ]:

QUESTION 4

In [ ]:

SOLUTION

In [ ]:

Step 1 (*#Import the necessary libraries)*

In [321]:

**import** numpy **as** np

**import** pandas **as** pd

**import** datetime

In [ ]:

Step 2 (*#Import the dataset from the attached file wind.txt)*

In [392]:

data **=** 'https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/599c7a2702a96/12679944?X-Blackboard-Expiration=1650110400000&X-Blackboard-Signature=dgYd1m5K3ktgyNwIB7nZzNdreGD0USfJGBaawsI7XGQ%3D&X-Blackboard-Client-Id=100784&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27wind.txt&response-content-type=text%2Fplain&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEPf%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIQCN19bd1Wj9q2z8vE%2FlrwhdsYK9u0%2FOv%2BmuQ6BJ0gLJbwIgbJFsX9jkwFx%2Bd4U%2FhXXFaRsrXmZpY726qRTRXdnxgB8qgwQIoP%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw1NTY5MDM4NjEzNjEiDO6bVwzV5u6EIdFmsirXA%2BztS9VM2KyiEHEHSquYpfz5TffnBRWIgJ0SIoiR1kOvNE11%2BGzJQJEATbr6RJR15wzY%2FicQVSBxGHt0KXoeIwovl6%2FZ0TAiucqm6XrFMHChYNfRP9B6mqqhwXtR4tmpkhbKnLYXZ0fcEztKywb%2FGLSC5Y0PHXf4HNEJvA8bv1YaKMNT2z4Rn%2BdJ6vpeO6ieYO9LObFYXRrOuc%2F8qeBKjtorXU6e%2FIcj7cP5fOgzoXppaifUfzTJ73D1VAkYBMIWcWi8btPj2weI84A9yR6HxAy%2BOvFK6cqT2Pw6Fly1DE8pcqB0UFogPOQRcJxGJV3EjLKSjNHOXLRraSEzpEHSwkLUiqE2DMYXnwqupJkHxiDpN3A5%2BCbaURTxzWvwn3m4UBPzOqiyPx4QFzWR2dU4HI%2FJPkrtyF6sOfjPc9waQJrE4AMjwhUR9GHkEx8s4821LpN%2F6WGuNJKkIauJtg49dC%2Bk36yaRF7WVHoIVUyA6nfcZxiFhNkXMIKryNXsEBE%2BRG6z4obUseQdXBGFgD4WGjDFmPjETDFI0pDkczEpzpG0RHN7dygUxs%2FNxHqMXxzbUpKJwAy2CU8fjI%2B2a6D%2BH6TurGy4a6iVttAuMlYXy%2BXCjX9i5DhCnDCr0OmSBjqlAcNGaTo%2FQepvjHL9YIWW1b983aif8kr%2Bq3zK4md%2B3o%2BnMTy73j21rvwdL8oecAcBYMT%2FN%2B9JRT%2BYQbsD7sMiwlKwk2KRaFd86C4bTHqFpDzR9URqw8%2FIPJSxio6Crub%2Bj0HShdTwEspamWFMGCavMqDCirknYFE45Awcz%2B%2FQ8qRdOPsiqBVWnM360xrRIHtwQSrcUfwpMIKPPua%2FYYhVLQaCcgvXzg%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220416T060000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRY7VY5EMXA%2F20220416%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Signature=37847f84a0421c0bffbe3fc0989b42e27ce27bf46391e0ebcc02e9324a69ad2e'

In [ ]:

Step 3 (*#Assign it to a variable called data and replace the first 3 columns by a proper datetime index.)*

In [393]:

data **=** pd**.**read\_csv('https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/599c7a2702a96/12679944?X-Blackboard-Expiration=1650110400000&X-Blackboard-Signature=dgYd1m5K3ktgyNwIB7nZzNdreGD0USfJGBaawsI7XGQ%3D&X-Blackboard-Client-Id=100784&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27wind.txt&response-content-type=text%2Fplain&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEPf%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIQCN19bd1Wj9q2z8vE%2FlrwhdsYK9u0%2FOv%2BmuQ6BJ0gLJbwIgbJFsX9jkwFx%2Bd4U%2FhXXFaRsrXmZpY726qRTRXdnxgB8qgwQIoP%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw1NTY5MDM4NjEzNjEiDO6bVwzV5u6EIdFmsirXA%2BztS9VM2KyiEHEHSquYpfz5TffnBRWIgJ0SIoiR1kOvNE11%2BGzJQJEATbr6RJR15wzY%2FicQVSBxGHt0KXoeIwovl6%2FZ0TAiucqm6XrFMHChYNfRP9B6mqqhwXtR4tmpkhbKnLYXZ0fcEztKywb%2FGLSC5Y0PHXf4HNEJvA8bv1YaKMNT2z4Rn%2BdJ6vpeO6ieYO9LObFYXRrOuc%2F8qeBKjtorXU6e%2FIcj7cP5fOgzoXppaifUfzTJ73D1VAkYBMIWcWi8btPj2weI84A9yR6HxAy%2BOvFK6cqT2Pw6Fly1DE8pcqB0UFogPOQRcJxGJV3EjLKSjNHOXLRraSEzpEHSwkLUiqE2DMYXnwqupJkHxiDpN3A5%2BCbaURTxzWvwn3m4UBPzOqiyPx4QFzWR2dU4HI%2FJPkrtyF6sOfjPc9waQJrE4AMjwhUR9GHkEx8s4821LpN%2F6WGuNJKkIauJtg49dC%2Bk36yaRF7WVHoIVUyA6nfcZxiFhNkXMIKryNXsEBE%2BRG6z4obUseQdXBGFgD4WGjDFmPjETDFI0pDkczEpzpG0RHN7dygUxs%2FNxHqMXxzbUpKJwAy2CU8fjI%2B2a6D%2BH6TurGy4a6iVttAuMlYXy%2BXCjX9i5DhCnDCr0OmSBjqlAcNGaTo%2FQepvjHL9YIWW1b983aif8kr%2Bq3zK4md%2B3o%2BnMTy73j21rvwdL8oecAcBYMT%2FN%2B9JRT%2BYQbsD7sMiwlKwk2KRaFd86C4bTHqFpDzR9URqw8%2FIPJSxio6Crub%2Bj0HShdTwEspamWFMGCavMqDCirknYFE45Awcz%2B%2FQ8qRdOPsiqBVWnM360xrRIHtwQSrcUfwpMIKPPua%2FYYhVLQaCcgvXzg%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220416T060000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRY7VY5EMXA%2F20220416%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Signature=37847f84a0421c0bffbe3fc0989b42e27ce27bf46391e0ebcc02e9324a69ad2e',sep**=**"\s+",parse\_dates**=**[[0,1,2]])

In [394]:

data**.**head()

Out[394]:

|  | **Yr\_Mo\_Dy** | **RPT** | **VAL** | **ROS** | **KIL** | **SHA** | **BIR** | **DUB** | **CLA** | **MUL** | **CLO** | **BEL** | **MAL** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2061-01-01 | 15.04 | 14.96 | 13.17 | 9.29 | NaN | 9.87 | 13.67 | 10.25 | 10.83 | 12.58 | 18.50 | 15.04 |
| **1** | 2061-01-02 | 14.71 | NaN | 10.83 | 6.50 | 12.62 | 7.67 | 11.50 | 10.04 | 9.79 | 9.67 | 17.54 | 13.83 |
| **2** | 2061-01-03 | 18.50 | 16.88 | 12.33 | 10.13 | 11.17 | 6.17 | 11.25 | NaN | 8.50 | 7.67 | 12.75 | 12.71 |
| **3** | 2061-01-04 | 10.58 | 6.63 | 11.75 | 4.58 | 4.54 | 2.88 | 8.63 | 1.79 | 5.83 | 5.88 | 5.46 | 10.88 |
| **4** | 2061-01-05 | 13.33 | 13.25 | 11.42 | 6.17 | 10.71 | 8.21 | 11.92 | 6.54 | 10.92 | 10.34 | 12.92 | 11.83 |

In [ ]:

Step 4 (*#Year 2061? Do we really have data from this year? Create a function to fix it and apply it.)*

In [395]:

**def** fix\_century(x):

year**=**x**.**year**-**100 **if** x**.**year **>** 1979 **else** x**.**year

**return** datetime**.**date(year,x**.**month,x**.**day)

data['Yr\_Mo\_Dy']**=**data['Yr\_Mo\_Dy']**.**apply(fix\_century)

data**.**head()

Out[395]:

|  | **Yr\_Mo\_Dy** | **RPT** | **VAL** | **ROS** | **KIL** | **SHA** | **BIR** | **DUB** | **CLA** | **MUL** | **CLO** | **BEL** | **MAL** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1961-01-01 | 15.04 | 14.96 | 13.17 | 9.29 | NaN | 9.87 | 13.67 | 10.25 | 10.83 | 12.58 | 18.50 | 15.04 |
| **1** | 1961-01-02 | 14.71 | NaN | 10.83 | 6.50 | 12.62 | 7.67 | 11.50 | 10.04 | 9.79 | 9.67 | 17.54 | 13.83 |
| **2** | 1961-01-03 | 18.50 | 16.88 | 12.33 | 10.13 | 11.17 | 6.17 | 11.25 | NaN | 8.50 | 7.67 | 12.75 | 12.71 |
| **3** | 1961-01-04 | 10.58 | 6.63 | 11.75 | 4.58 | 4.54 | 2.88 | 8.63 | 1.79 | 5.83 | 5.88 | 5.46 | 10.88 |
| **4** | 1961-01-05 | 13.33 | 13.25 | 11.42 | 6.17 | 10.71 | 8.21 | 11.92 | 6.54 | 10.92 | 10.34 | 12.92 | 11.83 |

In [ ]:

Step 5 (*#Set the right dates as the index. Pay attention at the data type, it should be datetime64[ns].)*

In [396]:

data['Yr\_Mo\_Dy']**=**pd**.**to\_datetime(data['Yr\_Mo\_Dy'])

data**=**data**.**set\_index('Yr\_Mo\_Dy')

data**.**head()

Out[396]:

|  | **RPT** | **VAL** | **ROS** | **KIL** | **SHA** | **BIR** | **DUB** | **CLA** | **MUL** | **CLO** | **BEL** | **MAL** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Yr\_Mo\_Dy** |  |  |  |  |  |  |  |  |  |  |  |  |
| **1961-01-01** | 15.04 | 14.96 | 13.17 | 9.29 | NaN | 9.87 | 13.67 | 10.25 | 10.83 | 12.58 | 18.50 | 15.04 |
| **1961-01-02** | 14.71 | NaN | 10.83 | 6.50 | 12.62 | 7.67 | 11.50 | 10.04 | 9.79 | 9.67 | 17.54 | 13.83 |
| **1961-01-03** | 18.50 | 16.88 | 12.33 | 10.13 | 11.17 | 6.17 | 11.25 | NaN | 8.50 | 7.67 | 12.75 | 12.71 |
| **1961-01-04** | 10.58 | 6.63 | 11.75 | 4.58 | 4.54 | 2.88 | 8.63 | 1.79 | 5.83 | 5.88 | 5.46 | 10.88 |
| **1961-01-05** | 13.33 | 13.25 | 11.42 | 6.17 | 10.71 | 8.21 | 11.92 | 6.54 | 10.92 | 10.34 | 12.92 | 11.83 |

In [ ]:

Step 6 (*#Compute how many values are missing for each location over the entire record.They should be ignored in all calculations below.)*

In [397]:

data**.**isnull()**.**sum()

Out[397]:

RPT 6

VAL 3

ROS 2

KIL 5

SHA 2

BIR 0

DUB 3

CLA 2

MUL 3

CLO 1

BEL 0

MAL 4

dtype: int64

In [ ]:

Step 7 (*#Compute how many non-missing values there are in total.)*

In [398]:

data**.**shape[0]**-**data**.**isnull()**.**sum()

Out[398]:

RPT 6568

VAL 6571

ROS 6572

KIL 6569

SHA 6572

BIR 6574

DUB 6571

CLA 6572

MUL 6571

CLO 6573

BEL 6574

MAL 6570

dtype: int64

In [ ]:

Step 8 (*#Calculate the mean windspeeds of the windspeeds over all the locations and all the times.)*

In [399]:

data**.**mean()**.**mean()

Out[399]:

10.227982360836924

In [ ]:

Step 9 (*#Create a DataFrame called loc\_stats and calculate the min, max and mean windspeeds and standard deviations of the windspeeds at each location over all the days)*

In [400]:

loc\_stats**=**pd**.**DataFrame()

loc\_stats['min']**=**data**.**min()

loc\_stats['max']**=**data**.**max()

loc\_stats['mean']**=**data**.**mean()

loc\_stats['std']**=**data**.**std()

loc\_stats

Out[400]:

|  | **min** | **max** | **mean** | **std** |
| --- | --- | --- | --- | --- |
| **RPT** | 0.67 | 35.80 | 12.362987 | 5.618413 |
| **VAL** | 0.21 | 33.37 | 10.644314 | 5.267356 |
| **ROS** | 1.50 | 33.84 | 11.660526 | 5.008450 |
| **KIL** | 0.00 | 28.46 | 6.306468 | 3.605811 |
| **SHA** | 0.13 | 37.54 | 10.455834 | 4.936125 |
| **BIR** | 0.00 | 26.16 | 7.092254 | 3.968683 |
| **DUB** | 0.00 | 30.37 | 9.797343 | 4.977555 |
| **CLA** | 0.00 | 31.08 | 8.495053 | 4.499449 |
| **MUL** | 0.00 | 25.88 | 8.493590 | 4.166872 |
| **CLO** | 0.04 | 28.21 | 8.707332 | 4.503954 |
| **BEL** | 0.13 | 42.38 | 13.121007 | 5.835037 |
| **MAL** | 0.67 | 42.54 | 15.599079 | 6.699794 |

In [ ]:

Step 10 (*#Create a DataFrame called day\_stats and calculate the min, max and mean windspeed and standard deviations of the windspeeds across all the locations at each day.)*

In [401]:

day\_stats**=**pd**.**DataFrame()

day\_stats['min']**=**data**.**min(axis**=**1)

day\_stats['max']**=**data**.**max(axis**=**1)

day\_stats['mean']**=**data**.**mean(axis**=**1)

day\_stats['std']**=**data**.**std(axis**=**1)

day\_stats**.**head()

Out[401]:

|  | **min** | **max** | **mean** | **std** |
| --- | --- | --- | --- | --- |
| **Yr\_Mo\_Dy** |  |  |  |  |
| **1961-01-01** | 9.29 | 18.50 | 13.018182 | 2.808875 |
| **1961-01-02** | 6.50 | 17.54 | 11.336364 | 3.188994 |
| **1961-01-03** | 6.17 | 18.50 | 11.641818 | 3.681912 |
| **1961-01-04** | 1.79 | 11.75 | 6.619167 | 3.198126 |
| **1961-01-05** | 6.17 | 13.33 | 10.630000 | 2.445356 |

In [ ]:

Step 11 (*#Find the average windspeed in January for each location.*

*#Treat January 1961 and January 1962 both as January)*

In [402]:

data['date']**=**data**.**index

data['month']**=**data['date']**.**apply(**lambda** date:date**.**month)

data['year']**=**data['date']**.**apply(**lambda** date:date**.**year)

data['day']**=**data['date']**.**apply(**lambda** date:date**.**day)

january\_winds**=**data**.**query('month==1')

january\_winds

january\_winds**.**loc[:,'RPT':'MAL']**.**mean()

Out[402]:

RPT 14.847325

VAL 12.914560

ROS 13.299624

KIL 7.199498

SHA 11.667734

BIR 8.054839

DUB 11.819355

CLA 9.512047

MUL 9.543208

CLO 10.053566

BEL 14.550520

MAL 18.028763

dtype: float64

In [ ]:

Step 12 (*#Downsample the record to a yearly frequency for each location.)*

In [403]:

data**.**query('month == 1 and day == 1')

Out[403]:

|  | **RPT** | **VAL** | **ROS** | **KIL** | **SHA** | **BIR** | **DUB** | **CLA** | **MUL** | **CLO** | **BEL** | **MAL** | **date** | **month** | **year** | **day** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Yr\_Mo\_Dy** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1961-01-01** | 15.04 | 14.96 | 13.17 | 9.29 | NaN | 9.87 | 13.67 | 10.25 | 10.83 | 12.58 | 18.50 | 15.04 | 1961-01-01 | 1 | 1961 | 1 |
| **1962-01-01** | 9.29 | 3.42 | 11.54 | 3.50 | 2.21 | 1.96 | 10.41 | 2.79 | 3.54 | 5.17 | 4.38 | 7.92 | 1962-01-01 | 1 | 1962 | 1 |
| **1963-01-01** | 15.59 | 13.62 | 19.79 | 8.38 | 12.25 | 10.00 | 23.45 | 15.71 | 13.59 | 14.37 | 17.58 | 34.13 | 1963-01-01 | 1 | 1963 | 1 |
| **1964-01-01** | 25.80 | 22.13 | 18.21 | 13.25 | 21.29 | 14.79 | 14.12 | 19.58 | 13.25 | 16.75 | 28.96 | 21.00 | 1964-01-01 | 1 | 1964 | 1 |
| **1965-01-01** | 9.54 | 11.92 | 9.00 | 4.38 | 6.08 | 5.21 | 10.25 | 6.08 | 5.71 | 8.63 | 12.04 | 17.41 | 1965-01-01 | 1 | 1965 | 1 |
| **1966-01-01** | 22.04 | 21.50 | 17.08 | 12.75 | 22.17 | 15.59 | 21.79 | 18.12 | 16.66 | 17.83 | 28.33 | 23.79 | 1966-01-01 | 1 | 1966 | 1 |
| **1967-01-01** | 6.46 | 4.46 | 6.50 | 3.21 | 6.67 | 3.79 | 11.38 | 3.83 | 7.71 | 9.08 | 10.67 | 20.91 | 1967-01-01 | 1 | 1967 | 1 |
| **1968-01-01** | 30.04 | 17.88 | 16.25 | 16.25 | 21.79 | 12.54 | 18.16 | 16.62 | 18.75 | 17.62 | 22.25 | 27.29 | 1968-01-01 | 1 | 1968 | 1 |
| **1969-01-01** | 6.13 | 1.63 | 5.41 | 1.08 | 2.54 | 1.00 | 8.50 | 2.42 | 4.58 | 6.34 | 9.17 | 16.71 | 1969-01-01 | 1 | 1969 | 1 |
| **1970-01-01** | 9.59 | 2.96 | 11.79 | 3.42 | 6.13 | 4.08 | 9.00 | 4.46 | 7.29 | 3.50 | 7.33 | 13.00 | 1970-01-01 | 1 | 1970 | 1 |
| **1971-01-01** | 3.71 | 0.79 | 4.71 | 0.17 | 1.42 | 1.04 | 4.63 | 0.75 | 1.54 | 1.08 | 4.21 | 9.54 | 1971-01-01 | 1 | 1971 | 1 |
| **1972-01-01** | 9.29 | 3.63 | 14.54 | 4.25 | 6.75 | 4.42 | 13.00 | 5.33 | 10.04 | 8.54 | 8.71 | 19.17 | 1972-01-01 | 1 | 1972 | 1 |
| **1973-01-01** | 16.50 | 15.92 | 14.62 | 7.41 | 8.29 | 11.21 | 13.54 | 7.79 | 10.46 | 10.79 | 13.37 | 9.71 | 1973-01-01 | 1 | 1973 | 1 |
| **1974-01-01** | 23.21 | 16.54 | 16.08 | 9.75 | 15.83 | 11.46 | 9.54 | 13.54 | 13.83 | 16.66 | 17.21 | 25.29 | 1974-01-01 | 1 | 1974 | 1 |
| **1975-01-01** | 14.04 | 13.54 | 11.29 | 5.46 | 12.58 | 5.58 | 8.12 | 8.96 | 9.29 | 5.17 | 7.71 | 11.63 | 1975-01-01 | 1 | 1975 | 1 |
| **1976-01-01** | 18.34 | 17.67 | 14.83 | 8.00 | 16.62 | 10.13 | 13.17 | 9.04 | 13.13 | 5.75 | 11.38 | 14.96 | 1976-01-01 | 1 | 1976 | 1 |
| **1977-01-01** | 20.04 | 11.92 | 20.25 | 9.13 | 9.29 | 8.04 | 10.75 | 5.88 | 9.00 | 9.00 | 14.88 | 25.70 | 1977-01-01 | 1 | 1977 | 1 |
| **1978-01-01** | 8.33 | 7.12 | 7.71 | 3.54 | 8.50 | 7.50 | 14.71 | 10.00 | 11.83 | 10.00 | 15.09 | 20.46 | 1978-01-01 | 1 | 1978 | 1 |

In [ ]:

Step 13 (*#Downsample the record to a monthly frequency for each location.)*

In [404]:

data**.**query('day == 1')

Out[404]:

|  | **RPT** | **VAL** | **ROS** | **KIL** | **SHA** | **BIR** | **DUB** | **CLA** | **MUL** | **CLO** | **BEL** | **MAL** | **date** | **month** | **year** | **day** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Yr\_Mo\_Dy** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1961-01-01** | 15.04 | 14.96 | 13.17 | 9.29 | NaN | 9.87 | 13.67 | 10.25 | 10.83 | 12.58 | 18.50 | 15.04 | 1961-01-01 | 1 | 1961 | 1 |
| **1961-02-01** | 14.25 | 15.12 | 9.04 | 5.88 | 12.08 | 7.17 | 10.17 | 3.63 | 6.50 | 5.50 | 9.17 | 8.00 | 1961-02-01 | 2 | 1961 | 1 |
| **1961-03-01** | 12.67 | 13.13 | 11.79 | 6.42 | 9.79 | 8.54 | 10.25 | 13.29 | NaN | 12.21 | 20.62 | NaN | 1961-03-01 | 3 | 1961 | 1 |
| **1961-04-01** | 8.38 | 6.34 | 8.33 | 6.75 | 9.33 | 9.54 | 11.67 | 8.21 | 11.21 | 6.46 | 11.96 | 7.17 | 1961-04-01 | 4 | 1961 | 1 |
| **1961-05-01** | 15.87 | 13.88 | 15.37 | 9.79 | 13.46 | 10.17 | 9.96 | 14.04 | 9.75 | 9.92 | 18.63 | 11.12 | 1961-05-01 | 5 | 1961 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1978-08-01** | 19.33 | 15.09 | 20.17 | 8.83 | 12.62 | 10.41 | 9.33 | 12.33 | 9.50 | 9.92 | 15.75 | 18.00 | 1978-08-01 | 8 | 1978 | 1 |
| **1978-09-01** | 8.42 | 6.13 | 9.87 | 5.25 | 3.21 | 5.71 | 7.25 | 3.50 | 7.33 | 6.50 | 7.62 | 15.96 | 1978-09-01 | 9 | 1978 | 1 |
| **1978-10-01** | 9.50 | 6.83 | 10.50 | 3.88 | 6.13 | 4.58 | 4.21 | 6.50 | 6.38 | 6.54 | 10.63 | 14.09 | 1978-10-01 | 10 | 1978 | 1 |
| **1978-11-01** | 13.59 | 16.75 | 11.25 | 7.08 | 11.04 | 8.33 | 8.17 | 11.29 | 10.75 | 11.25 | 23.13 | 25.00 | 1978-11-01 | 11 | 1978 | 1 |
| **1978-12-01** | 21.29 | 16.29 | 24.04 | 12.79 | 18.21 | 19.29 | 21.54 | 17.21 | 16.71 | 17.83 | 17.75 | 25.70 | 1978-12-01 | 12 | 1978 | 1 |

216 rows × 16 columns

In [ ]:

Step 14 (*#Downsample the record to a weekly frequency for each location.)*

In [405]:

weekly\_resampled\_data **=** data**.**resample('W')**.**mean()

weekly\_resampled\_data

Out[405]:

|  | **RPT** | **VAL** | **ROS** | **KIL** | **SHA** | **BIR** | **DUB** | **CLA** | **MUL** | **CLO** | **BEL** | **MAL** | **month** | **year** | **day** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Yr\_Mo\_Dy** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1961-01-01** | 15.040000 | 14.960000 | 13.170000 | 9.290000 | NaN | 9.870000 | 13.670000 | 10.250000 | 10.830000 | 12.580000 | 18.500000 | 15.040000 | 1.000000 | 1961.0 | 1.000000 |
| **1961-01-08** | 13.541429 | 11.486667 | 10.487143 | 6.417143 | 9.474286 | 6.435714 | 11.061429 | 6.616667 | 8.434286 | 8.497143 | 12.481429 | 13.238571 | 1.000000 | 1961.0 | 5.000000 |
| **1961-01-15** | 12.468571 | 8.967143 | 11.958571 | 4.630000 | 7.351429 | 5.072857 | 7.535714 | 6.820000 | 5.712857 | 7.571429 | 11.125714 | 11.024286 | 1.000000 | 1961.0 | 12.000000 |
| **1961-01-22** | 13.204286 | 9.862857 | 12.982857 | 6.328571 | 8.966667 | 7.417143 | 9.257143 | 7.875714 | 7.145714 | 8.124286 | 9.821429 | 11.434286 | 1.000000 | 1961.0 | 19.000000 |
| **1961-01-29** | 19.880000 | 16.141429 | 18.225714 | 12.720000 | 17.432857 | 14.828571 | 15.528571 | 15.160000 | 14.480000 | 15.640000 | 20.930000 | 22.530000 | 1.000000 | 1961.0 | 26.000000 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1978-12-03** | 14.934286 | 11.232857 | 13.941429 | 5.565714 | 10.215714 | 8.618571 | 9.642857 | 7.685714 | 9.011429 | 9.547143 | 11.835714 | 18.728571 | 11.428571 | 1978.0 | 17.142857 |
| **1978-12-10** | 20.740000 | 19.190000 | 17.034286 | 9.777143 | 15.287143 | 12.774286 | 14.437143 | 12.488571 | 13.870000 | 14.082857 | 18.517143 | 23.061429 | 12.000000 | 1978.0 | 7.000000 |
| **1978-12-17** | 16.758571 | 14.692857 | 14.987143 | 6.917143 | 11.397143 | 7.272857 | 10.208571 | 7.967143 | 9.168571 | 8.565714 | 11.102857 | 15.562857 | 12.000000 | 1978.0 | 14.000000 |
| **1978-12-24** | 11.155714 | 8.008571 | 13.172857 | 4.004286 | 7.825714 | 6.290000 | 7.798571 | 8.667143 | 7.151429 | 8.072857 | 11.845714 | 18.977143 | 12.000000 | 1978.0 | 21.000000 |
| **1978-12-31** | 14.951429 | 11.801429 | 16.035714 | 6.507143 | 9.660000 | 8.620000 | 13.708571 | 10.477143 | 10.868571 | 11.471429 | 12.947143 | 26.844286 | 12.000000 | 1978.0 | 28.000000 |

940 rows × 15 columns

In [ ]:

Step 15 (*#Calculate the min, max and mean windspeeds and standard deviations of the windspeeds across all locations for each week (assume that the first week starts on January 2 1961) for the first 52 weeks.)*

In [406]:

df\_1961 **=** data[data**.**index **<** pd**.**to\_datetime('1962-01-01')]

df\_1961**.**resample('W')**.**mean()

df\_1961**.**resample('W')**.**min()

df\_1961**.**resample('W')**.**max()

df\_1961**.**resample('W')**.**std()

Out[406]:

|  | **RPT** | **VAL** | **ROS** | **KIL** | **SHA** | **BIR** | **DUB** | **CLA** | **MUL** | **CLO** | **BEL** | **MAL** | **month** | **year** | **day** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Yr\_Mo\_Dy** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1961-01-01** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1961-01-08** | 2.631321 | 3.949525 | 1.604761 | 1.810743 | 3.251660 | 2.059546 | 1.872222 | 3.098404 | 1.722255 | 1.704941 | 4.349139 | 1.773062 | 0.000000 | 0.0 | 2.160247 |
| **1961-01-15** | 3.555392 | 3.148945 | 5.034959 | 3.549559 | 3.471726 | 3.251039 | 4.709309 | 3.936894 | 3.500975 | 4.084293 | 5.552215 | 4.692355 | 0.000000 | 0.0 | 2.160247 |
| **1961-01-22** | 5.337402 | 3.837785 | 5.086229 | 6.245541 | 3.612875 | 3.453432 | 5.166300 | 3.164990 | 4.169112 | 4.783952 | 3.626584 | 4.237239 | 0.000000 | 0.0 | 2.160247 |
| **1961-01-29** | 4.619061 | 5.170224 | 4.665843 | 4.301325 | 4.858116 | 3.749415 | 4.508449 | 4.436222 | 4.902057 | 3.713368 | 5.210726 | 3.874721 | 0.000000 | 0.0 | 2.160247 |
| **1961-02-05** | 5.251408 | 5.187395 | 3.975166 | 2.709106 | 2.334619 | 2.397066 | 2.423454 | 4.081158 | 2.802490 | 2.839501 | 4.210858 | 4.336104 | 0.487950 | 0.0 | 13.483676 |
| **1961-02-12** | 3.587677 | 3.608373 | 3.290303 | 2.262056 | 5.571108 | 3.048976 | 2.974059 | 3.022753 | 2.914760 | 1.746749 | 4.063753 | 1.828705 | 0.000000 | 0.0 | 2.160247 |
| **1961-02-19** | 5.064609 | 3.575012 | 4.196621 | 4.311569 | 2.321716 | 3.024078 | 4.958631 | 2.283444 | 2.560591 | 2.531361 | 5.910938 | 4.685377 | 0.000000 | 0.0 | 2.160247 |
| **1961-02-26** | 7.020716 | 5.147348 | 5.578470 | 4.482075 | 6.480712 | 5.029874 | 6.037916 | 4.869668 | 4.705163 | 4.920064 | 5.091162 | 6.182283 | 0.000000 | 0.0 | 2.160247 |
| **1961-03-05** | 0.997721 | 2.851955 | 1.796871 | 1.652572 | 2.957129 | 2.022247 | 3.338177 | 2.877395 | 2.610124 | 1.593685 | 4.332331 | 3.021387 | 0.487950 | 0.0 | 12.027746 |
| **1961-03-12** | 3.732263 | 3.230167 | 3.592909 | 2.609928 | 3.110857 | 3.440419 | 5.537269 | 3.760557 | 3.657690 | 3.655113 | 4.358759 | 5.769890 | 0.000000 | 0.0 | 2.160247 |
| **1961-03-19** | 3.860036 | 2.352867 | 2.939244 | 2.416746 | 2.654289 | 2.045448 | 4.046593 | 3.373421 | 3.301880 | 3.099472 | 3.779727 | 4.331958 | 0.000000 | 0.0 | 2.160247 |
| **1961-03-26** | 3.613298 | 3.657265 | 4.041121 | 2.410127 | 2.927188 | 2.285729 | 2.892270 | 2.789682 | 2.469432 | 2.538224 | 4.318069 | 3.701846 | 0.000000 | 0.0 | 2.160247 |
| **1961-04-02** | 5.046922 | 4.687315 | 3.678996 | 3.625488 | 5.657679 | 4.054639 | 4.047068 | 4.689504 | 4.025317 | 3.191115 | 4.179854 | 3.924555 | 0.487950 | 0.0 | 13.483676 |
| **1961-04-09** | 4.604392 | 2.845399 | 2.902991 | 2.854052 | 2.743895 | 2.525046 | 2.109769 | 1.996609 | 1.935058 | 2.336182 | 3.147781 | 2.598271 | 0.000000 | 0.0 | 2.160247 |
| **1961-04-16** | 3.937727 | 2.607118 | 4.585752 | 2.664416 | 1.941836 | 2.388107 | 1.970667 | 2.379446 | 2.123279 | 2.161137 | 3.641464 | 2.747842 | 0.000000 | 0.0 | 2.160247 |
| **1961-04-23** | 5.676655 | 4.631736 | 5.456290 | 3.835403 | 4.122437 | 3.835998 | 3.254663 | 3.935731 | 3.656682 | 3.347972 | 4.735096 | 5.908542 | 0.000000 | 0.0 | 2.160247 |
| **1961-04-30** | 4.349662 | 2.871425 | 3.732776 | 1.897616 | 3.259907 | 2.528436 | 3.565777 | 3.426255 | 2.718926 | 2.840568 | 2.948237 | 5.108365 | 0.000000 | 0.0 | 2.160247 |
| **1961-05-07** | 5.025507 | 3.750835 | 4.301778 | 3.141495 | 5.781916 | 3.324038 | 4.419008 | 4.461950 | 3.558856 | 3.620819 | 8.003490 | 7.728504 | 0.000000 | 0.0 | 2.160247 |
| **1961-05-14** | 3.371022 | 3.782947 | 3.002225 | 3.509695 | 4.813159 | 4.660173 | 5.744586 | 4.690361 | 5.126826 | 5.460237 | 3.968272 | 7.858246 | 0.000000 | 0.0 | 2.160247 |
| **1961-05-21** | 3.631730 | 2.468906 | 3.767505 | 0.668791 | 2.541271 | 1.377071 | 1.650089 | 2.221636 | 1.613885 | 2.216889 | 1.975853 | 3.310819 | 0.000000 | 0.0 | 2.160247 |
| **1961-05-28** | 2.739433 | 3.378537 | 6.355768 | 1.378803 | 2.945134 | 1.873381 | 2.350104 | 3.207147 | 1.990891 | 2.575661 | 3.024524 | 3.811818 | 0.000000 | 0.0 | 2.160247 |
| **1961-06-04** | 3.099701 | 1.868125 | 3.474607 | 2.323826 | 2.305695 | 1.332009 | 2.300028 | 0.915184 | 1.954609 | 2.096989 | 2.611139 | 2.593586 | 0.534522 | 0.0 | 14.738999 |
| **1961-06-11** | 2.248597 | 1.524836 | 1.887475 | 1.603775 | 2.238228 | 2.204087 | 2.514689 | 2.857527 | 1.695090 | 2.158323 | 3.993062 | 4.925055 | 0.000000 | 0.0 | 2.160247 |
| **1961-06-18** | 3.009482 | 3.509444 | 3.429057 | 2.130352 | 4.175947 | 3.292713 | 3.417989 | 4.858023 | 2.900464 | 3.792400 | 6.477887 | 6.242673 | 0.000000 | 0.0 | 2.160247 |
| **1961-06-25** | 1.982035 | 2.212460 | 1.916454 | 1.332234 | 1.422401 | 1.448318 | 2.455385 | 2.109358 | 2.042257 | 2.286218 | 2.498386 | 3.063011 | 0.000000 | 0.0 | 2.160247 |
| **1961-07-02** | 2.557856 | 2.902411 | 1.161629 | 1.911018 | 2.172125 | 1.626363 | 1.927673 | 3.065467 | 1.745125 | 1.564144 | 6.303747 | 3.652313 | 0.487950 | 0.0 | 12.998168 |
| **1961-07-09** | 3.664855 | 2.686658 | 2.995919 | 2.898890 | 3.661383 | 2.459639 | 3.493425 | 3.160424 | 3.453210 | 3.657179 | 4.537988 | 3.665705 | 0.000000 | 0.0 | 2.160247 |
| **1961-07-16** | 5.168710 | 3.849630 | 2.166206 | 2.665625 | 4.596567 | 3.164715 | 4.489523 | 4.083629 | 3.978635 | 3.271899 | 4.971060 | 4.974273 | 0.000000 | 0.0 | 2.160247 |
| **1961-07-23** | 1.047978 | 1.400010 | 2.783208 | 1.158908 | 0.764956 | 0.617815 | 1.757905 | 1.479152 | 0.873207 | 1.439785 | 2.050218 | 2.133994 | 0.000000 | 0.0 | 2.160247 |
| **1961-07-30** | 4.157641 | 3.203206 | 3.276465 | 2.843207 | 4.514505 | 3.786633 | 5.110195 | 4.060478 | 4.504844 | 4.350268 | 5.580903 | 6.664574 | 0.000000 | 0.0 | 2.160247 |
| **1961-08-06** | 2.950887 | 3.985226 | 3.483894 | 1.647275 | 4.189524 | 2.642246 | 3.709319 | 3.921508 | 3.217501 | 2.903018 | 4.901377 | 4.448251 | 0.377964 | 0.0 | 10.533394 |
| **1961-08-13** | 4.422268 | 2.053326 | 2.174407 | 1.921080 | 3.071872 | 2.045195 | 3.073187 | 2.088368 | 2.718228 | 2.073777 | 2.931302 | 3.356585 | 0.000000 | 0.0 | 2.160247 |
| **1961-08-20** | 2.283635 | 2.523416 | 1.766039 | 1.938109 | 2.568539 | 2.227127 | 3.197566 | 2.972809 | 2.339544 | 2.730237 | 4.086725 | 3.934238 | 0.000000 | 0.0 | 2.160247 |
| **1961-08-27** | 3.395857 | 3.174702 | 3.071403 | 2.455711 | 4.380624 | 3.416186 | 4.170083 | 4.900671 | 4.120185 | 3.855302 | 6.711322 | 4.947608 | 0.000000 | 0.0 | 2.160247 |
| **1961-09-03** | 4.398615 | 7.474025 | 4.395383 | 3.790030 | 6.128522 | 4.455552 | 3.920110 | 5.012759 | 4.155931 | 3.993736 | 7.678051 | 6.308087 | 0.534522 | 0.0 | 14.738999 |
| **1961-09-10** | 5.207278 | 4.003996 | 3.922643 | 3.151502 | 3.178791 | 2.965546 | 4.258696 | 3.074692 | 3.767708 | 3.649278 | 4.220584 | 6.049619 | 0.000000 | 0.0 | 2.160247 |
| **1961-09-17** | 7.679190 | 5.360585 | 6.638947 | 5.560509 | 6.279010 | 4.706530 | 4.235755 | 4.565163 | 4.847073 | 5.128338 | 4.464252 | 6.332885 | 0.000000 | 0.0 | 2.160247 |
| **1961-09-24** | 1.267399 | 3.445262 | 2.322229 | 1.609614 | 3.125219 | 2.293031 | 1.444916 | 3.342247 | 1.523142 | 2.354092 | 5.235868 | 3.113507 | 0.000000 | 0.0 | 2.160247 |
| **1961-10-01** | 4.559572 | 2.812482 | 4.201062 | 2.050553 | 3.353967 | 1.901786 | 2.098203 | 3.425963 | 1.790306 | 3.908397 | 5.091268 | 4.696504 | 0.377964 | 0.0 | 10.160615 |
| **1961-10-08** | 5.596710 | 5.060803 | 4.394234 | 3.249050 | 3.830935 | 3.686641 | 3.420894 | 3.313201 | 3.717239 | 4.296870 | 4.800403 | 5.462002 | 0.000000 | 0.0 | 2.160247 |
| **1961-10-15** | 4.780675 | 2.707483 | 5.979099 | 2.367850 | 2.246657 | 1.969085 | 3.409743 | 2.506401 | 2.947025 | 3.569308 | 4.113200 | 4.098130 | 0.000000 | 0.0 | 2.160247 |
| **1961-10-22** | 7.888314 | 5.998199 | 5.463782 | 4.989763 | 6.095112 | 3.801637 | 5.334910 | 5.199929 | 3.785875 | 5.890511 | 5.645871 | 7.468377 | 0.000000 | 0.0 | 2.160247 |
| **1961-10-29** | 7.957637 | 6.879973 | 7.428776 | 5.576503 | 7.344744 | 5.101188 | 4.542684 | 5.506544 | 4.448500 | 6.277629 | 7.056150 | 8.340881 | 0.000000 | 0.0 | 2.160247 |
| **1961-11-05** | 3.369201 | 3.900278 | 2.461109 | 2.448363 | 3.664806 | 2.513051 | 2.727816 | 3.599111 | 3.003274 | 2.784450 | 4.038493 | 3.870800 | 0.487950 | 0.0 | 13.483676 |
| **1961-11-12** | 3.939811 | 2.141191 | 6.779554 | 2.930324 | 2.090641 | 1.414308 | 2.388943 | 2.181840 | 2.153102 | 2.843518 | 2.532196 | 3.690752 | 0.000000 | 0.0 | 2.160247 |
| **1961-11-19** | 2.784358 | 3.208548 | 9.415716 | 4.176374 | 2.092809 | 3.440953 | 7.027159 | 3.452202 | 3.488521 | 4.402588 | 3.643285 | 3.787654 | 0.000000 | 0.0 | 2.160247 |
| **1961-11-26** | 3.214368 | 3.277904 | 3.622254 | 1.824938 | 2.132751 | 1.822767 | 3.551685 | 2.276663 | 2.392934 | 2.747452 | 5.407223 | 6.475867 | 0.000000 | 0.0 | 2.160247 |
| **1961-12-03** | 5.704669 | 5.107089 | 4.163650 | 4.097123 | 5.536334 | 3.534980 | 4.186505 | 3.744471 | 3.658796 | 3.680477 | 5.552648 | 5.233192 | 0.534522 | 0.0 | 14.205968 |
| **1961-12-10** | 4.890152 | 4.115506 | 4.682044 | 3.782631 | 3.345941 | 3.481252 | 4.357862 | 4.503161 | 3.452864 | 4.156207 | 4.667933 | 7.345893 | 0.000000 | 0.0 | 2.160247 |
| **1961-12-17** | 4.095106 | 3.587886 | 3.843380 | 3.501489 | 3.943048 | 3.312448 | 6.474478 | 3.482027 | 4.333020 | 4.633398 | 6.531043 | 5.665006 | 0.000000 | 0.0 | 2.160247 |
| **1961-12-24** | 4.959717 | 2.220866 | 5.653229 | 5.341288 | 4.898600 | 4.019469 | 5.724775 | 4.134367 | 4.243508 | 4.637096 | 5.065308 | 5.048035 | 0.000000 | 0.0 | 2.160247 |
| **1961-12-31** | 5.787783 | 4.566479 | 9.739918 | 4.167851 | 3.018856 | 2.674370 | 5.142213 | 4.289240 | 3.325214 | 3.526625 | 3.262217 | 3.012729 | 0.000000 | 0.0 | 2.160247 |

In [ ]:

In [ ]:

QUESTION 5

In [ ]:

SOLUTION

In [ ]:

Step 1 (*#Import the necessary libraries)*

In [418]:

**import** pandas **as** pd

In [ ]:

Step 2 (*#Import the dataset from this address.)*

In [419]:

url **=** 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv'

In [ ]:

Step 3 (*#Assign it to a variable called chipo.)*

In [420]:

chipo **=** pd**.**read\_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv', sep**=**'\t')

In [421]:

chipo**.**head()

Out[421]:

|  | **order\_id** | **quantity** | **item\_name** | **choice\_description** | **item\_price** |
| --- | --- | --- | --- | --- | --- |
| **0** | 1 | 1 | Chips and Fresh Tomato Salsa | NaN | $2.39 |
| **1** | 1 | 1 | Izze | [Clementine] | $3.39 |
| **2** | 1 | 1 | Nantucket Nectar | [Apple] | $3.39 |
| **3** | 1 | 1 | Chips and Tomatillo-Green Chili Salsa | NaN | $2.39 |
| **4** | 2 | 2 | Chicken Bowl | [Tomatillo-Red Chili Salsa (Hot), [Black Beans... | $16.98 |

In [ ]:

Step 4 (*#See the first 10 entries)*

In [422]:

print(chipo**.**head(10))

order\_id quantity item\_name \

0 1 1 Chips and Fresh Tomato Salsa

1 1 1 Izze

2 1 1 Nantucket Nectar

3 1 1 Chips and Tomatillo-Green Chili Salsa

4 2 2 Chicken Bowl

5 3 1 Chicken Bowl

6 3 1 Side of Chips

7 4 1 Steak Burrito

8 4 1 Steak Soft Tacos

9 5 1 Steak Burrito

choice\_description item\_price

0 NaN $2.39

1 [Clementine] $3.39

2 [Apple] $3.39

3 NaN $2.39

4 [Tomatillo-Red Chili Salsa (Hot), [Black Beans... $16.98

5 [Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou... $10.98

6 NaN $1.69

7 [Tomatillo Red Chili Salsa, [Fajita Vegetables... $11.75

8 [Tomatillo Green Chili Salsa, [Pinto Beans, Ch... $9.25

9 [Fresh Tomato Salsa, [Rice, Black Beans, Pinto... $9.25

In [ ]:

Step 5 (*#What is the number of observations in the dataset?)*

In [423]:

print(chipo**.**shape[0])

4622

In [ ]:

Step 6 (*#What is the number of columns in the dataset?)*

In [424]:

print(chipo**.**shape[1])

5

In [ ]:

Step 7 (*#Print the name of all the columns.)*

In [425]:

print(chipo**.**columns)

Index(['order\_id', 'quantity', 'item\_name', 'choice\_description',

'item\_price'],

dtype='object')

In [ ]:

Step 8 (*#How is the dataset indexed?)*

In [426]:

print(chipo**.**index)

RangeIndex(start=0, stop=4622, step=1)

In [ ]:

Step 9 (*#Which was the most-ordered item?)*

In [427]:

print(chipo[chipo**.**quantity**==**chipo**.**quantity**.**max()]**.**item\_name)

3598 Chips and Fresh Tomato Salsa

Name: item\_name, dtype: object

In [ ]:

Step 10 (*#For the most-ordered item, how many items were ordered?)*

In [428]:

print(chipo[chipo**.**quantity**==**chipo**.**quantity**.**max()]**.**quantity)

3598 15

Name: quantity, dtype: int64

In [ ]:

Step 11 (*#What was the most ordered item in the choice\_description column?)*

In [429]:

print(chipo**.**groupby('choice\_description')**.**agg({'quantity':'sum'})**.**sort\_values(by**=**'quantity', ascending**=False**)**.**head(1)**.**index[0])

[Diet Coke]

In [ ]:

Step 12 (*#How many items were orderd in total?)*

In [430]:

print(chipo**.**quantity**.**sum())

4972

In [ ]:

Step 13 (*#Turn the item price into a float*

*#Check the item price type*

*#Create a lambda function and change the type of item price*

*#Check the item price type)*

In [ ]:

"""chipo['item\_price'] = chipo['item\_price'].str.replace('$', '').astype(float)"""

In [300]:

print(chipo['item\_price']**.**dtype)

object

In [ ]:

Step 14 (*#How much was the revenue for the period in the dataset?)*

In [440]:

dollarizer **=** **lambda** x: float(x[1:**-**1])

chipo**.**item\_price **=** chipo**.**item\_price**.**apply(dollarizer)

In [442]:

revenue **=** (chipo['quantity'] **\*** chipo['item\_price'])**.**sum()

revenue

Out[442]:

39237.02

In [ ]:

Step 15 (*#How many orders were made in the period?)*

In [433]:

print(len(chipo**.**groupby('order\_id')**.**agg({'order\_id':'count'})))

1834

In [ ]:

Step 16 (*#What is the average revenue amount per order?)*

In [445]:

print(chipo**.**groupby('order\_id')**.**agg({'item\_price':'mean'})**.**mean())

item\_price 7.841911

dtype: float64

In [ ]:

Step 17 (*#How many different items are sold?)*

In [446]:

print(len(chipo**.**groupby('item\_name')**.**agg({'item\_name':'count'})))

50

In [ ]:

In [ ]:

QUESTION 6 (*#Create a line plot showing the number of marriages and divorces per capita in the U.S. between 1867 and 2014. Label both lines and show the legend.*

Don't forget to label your axes!)

In [ ]:

SOLUTION

In [447]:

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

In [377]:

df **=** pd**.**read\_csv('https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/599c7a2702a96/4479949?X-Blackboard-Expiration=1650099600000&X-Blackboard-Signature=XXvVqe7bsGGodQONGdnCCFQXyLY4j%2FALenf0rlSUeKc%3D&X-Blackboard-Client-Id=100784&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27us-marriages-divorces-1867-2014.csv&response-content-type=text%2Fcsv&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEPP%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIQC7HYJAq4cCX1t477xaYvUIL1p6c3ic%2BI9Ka39z%2BkfVyQIgCtnPj1OlpUU33NtdQNs%2BLzwh17qsaPrhidTNZgEt8OEqgwQInP%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw1NTY5MDM4NjEzNjEiDEPUoDpQhx%2F4T7pAoSrXA%2B9q%2B6L3qbGiwuqyDRB4Aa5vj9f7eYB1X7eRyjMVhKXQgo78nMzqfRDnk3Sp0a7mqaN2kcdZaxitvezu2tljOcmH48IUWdBjBmmtcrhu1nbHoJNUWBjJV36jRxkafvKV%2BNL%2Bued9qqyi4SO0CMG4O9EB4wm8c%2FGaDvKqwQ0cwUfLFeyyXMMMLgrbedhEQKbgUN0irT8%2BazU%2F9E70cbPjmRTnejccJ9136GLHj0M00D%2Bd8he%2BptalpyFO8pgcKFLiWuGKdG9mK0Lq%2BK7AN0mVcf%2Bbh8S%2FsgCdzhhLtykloI5KUaqnrl%2BehxlNCCWAARFo8DWd%2FnDa84nMVmdHzCQ%2FkBxdoT5v9e5SzGIn4rBtkBFejwJ%2FuaiosxZLAZb5AG6ePeuxLYubismb0IUOBNk2NF9MSVwLT6T1YCF7xUQo7dHDNq287KRycZT7PcaBjqMW3%2Bq7359Ny7YcjsE13JyMjt6nEwioaYJj3Ed74BLP%2B0PxcKB9FeB1H5Ghd2DCMaJT6zpLI%2FsYoeKDIEH%2Fj%2BIMp%2BIF%2F4i1gsVzLJAOOPfcCmENIVSN2qoxUjTQ87OG10tFB9mXfxD6jbsUkrecFeQHXROHdDvB8PSgJIDOG1AHpyEkgie7Zyb1MjDi1eiSBjqlAYzisI1lP4VhB5%2F%2BneLZptEbJA3TQhKOMWej6QgpSM3IT3gdR4HDU9b7PxiZHITKZ22VXUMtYoQIw%2FkD7rzw0xgWx77mranvof1%2FWHxwLmXAVPKmMgFJxiNVIHWbc52Sq5Yj0FxzMMRewpYIdW7Koy%2Fxap2qlF4pYjjbzmneybNnBwSiA94sL8h%2FIBB%2Br1LayoMZ7Iu%2Ft4qweC3SPy0faKOGvVtuqg%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220416T030000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRYY4ZTBZHQ%2F20220416%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Signature=6ea4790148f646b326a590a3e91276afbf39d6534a30274741ed8ae36cd6bef9')

In [378]:

df**.**head()

Out[378]:

|  | **Year** | **Marriages** | **Divorces** | **Population** | **Marriages\_per\_1000** | **Divorces\_per\_1000** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 1867 | 357000.0 | 10000.0 | 36970000 | 9.7 | 0.3 |
| **1** | 1868 | 345000.0 | 10000.0 | 37885000 | 9.1 | 0.3 |
| **2** | 1869 | 348000.0 | 11000.0 | 38870000 | 9.0 | 0.3 |
| **3** | 1870 | 352000.0 | 11000.0 | 39905000 | 8.8 | 0.3 |
| **4** | 1871 | 359000.0 | 12000.0 | 41010000 | 8.8 | 0.3 |

In [379]:

df**.**plot(x**=**'Year', y**=**['Marriages\_per\_1000', 'Divorces\_per\_1000'])

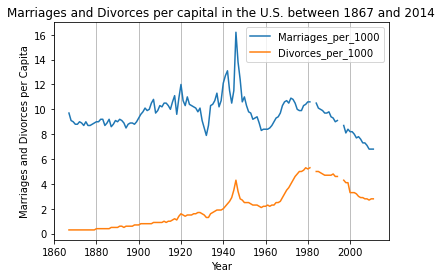
plt**.**title('Marriages and Divorces per capital in the U.S. between 1867 and 2014')

plt**.**ylabel('Marriages and Divorces per Capita')

plt**.**xlabel('Year')

plt**.**grid(axis**=**'x')

plt**.**show()



In [ ]:

In [ ]:

QUESTION 7

In [ ]:

SOLUTION

In [ ]:

*# Importing the matplotlib library*

In [380]:

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

In [382]:

plt**.**figure(figsize**=**[15, 10])

marriage **=** [709000, 1667000, 2315000]

divorce **=** [56000, 385000, 944000]

X **=** np**.**arange(len(marriage))

plt**.**bar(X, marriage, color **=** 'blue', width **=** 0.25)

plt**.**bar(X **+** 0.25, divorce, color **=** 'yellow', width **=** 0.25)

plt**.**legend(['Marriage', 'Divorce'])

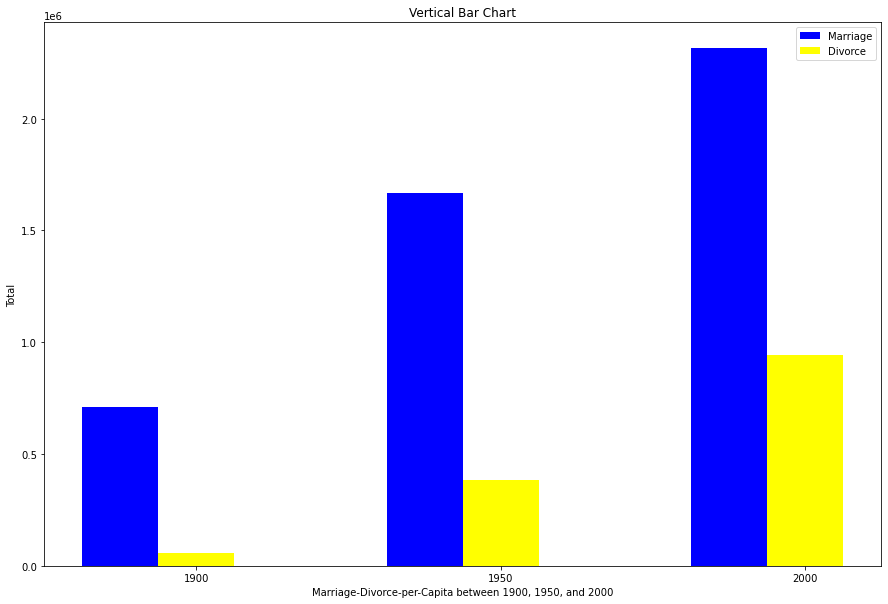
plt**.**xticks([i **+** 0.25 **for** i **in** range(3)], ['1900', '1950', '2000'])

plt**.**title("Vertical Bar Chart")

plt**.**xlabel('Marriage-Divorce-per-Capita between 1900, 1950, and 2000')

plt**.**ylabel('Total')

plt**.**show()



In [ ]:

In [ ]:

QUESTION 8 (*#Create a horizontal bar chart that compares the deadliest actors in Hollywood. Sort the actors by their kill count and label each bar with the corresponding actor's name.*

Don't forget to label your axes!)

In [ ]:

SOLUTION

In [ ]:

*#import the neccessary library*

In [383]:

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

In [384]:

df **=** pd**.**read\_csv('https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/599c7a2702a96/10029930?X-Blackboard-Expiration=1650099600000&X-Blackboard-Signature=RNgPpA4nvjTXVDJnpl5EmZrDwikbe%2FTRk9CMrI04ITM%3D&X-Blackboard-Client-Id=100784&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27actor\_kill\_counts.csv&response-content-type=text%2Fcsv&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEPT%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIDfwVB4HTtHhl%2BNMtub8SNNouEb15EfHJz5PqOsh%2FyI8AiEAuL4%2B2HpTshGN05dhcZucYPy%2FmU9Lgs90GIv80VL9S6sqgwQInP%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw1NTY5MDM4NjEzNjEiDIK3FdEoSuEbrxmk2irXA9FYyJO062DioWTMaPr5ZRVsxrzoiBEUIAvKk9m5TxsC8vGVZZf9QgQeva5SwD7S%2B3sgDU9kjAgbS3t7h8RRWpGGEJUWbXGD6SjnRLle0hPRpzY8edLh%2Bm6B2Ynd1y5dvbXxdJiqfYpYDKTfN6X3EKMbdPQDRsH40tB8%2B4HfkGUSSrhktMCJAMq8HUnx77M0Tj0FgKD4IUnLg72pYOY2mDrUmelh9mdlKhx8acIJe0KDoNMBGRKPZo1UNoyENcow13ghtAdojyXGJZs5zh8fKPZQsrABLDl810Xs%2BDLok2NPsMEOVgeJohMJhX0ZHmZ55DH63HL2U5FcOITuSNFhuKVRVqju9JhOKCLhs8zclCe2KexU5P0hMIYpUMMFNHruOpH6aQ%2BZGK5OmIqrc3DVOWI75TnZzv3rb9E9qpL%2FZxFboK7rxLjOqZ8UcnYhOcDvpvETKQM0Ja%2BLgZpXXFOUgnLgOHJFvaPApLljS3O4%2FcUXmNMzLFZfBlEEvEF3NPjPrH9CuQhL8SVgWEt7pqmRIX0eCrwbyLBlwtDLxU5bXyMpoaaWkrTY4XMpeUEIf3HWaZVbQMQdbWbgjggtDLWOH0fgxPEg%2BmZZo8iGDWw%2BANBTpvsx5aF%2F3TDK6eiSBjqlAUBFTv98MRgmF1ypvuAHqmkQAPFKoldYIYxtuoefeHzym8p5XU90brKlI0kFx%2B6XnbER3ch3K7ZwvfjSIsJahobDNf8eXX1vo6sPtXsUAwWHS0NFijDiEU%2BnRjfiiGW4EIaxtba2zDWwrRa0GzKkia1IYFEufNSpiG864LA%2FQSik0%2FP8nE6P94t3yHRbCc%2FH5bduj4a13B2E5kkSZL%2FPu2xxUcf6mg%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220416T030000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRY3SD63MKY%2F20220416%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Signature=0e286031350adee6ebdd30cbda6c465a09d6602a4f40470ceee1b128e5690297')

In [385]:

df**.**head()

Out[385]:

|  | **Actor** | **Count** |
| --- | --- | --- |
| **0** | Arnold Schwarzenegger | 369 |
| **1** | Chow Yun-Fat | 295 |
| **2** | Clint Eastwood | 207 |
| **3** | Clive Owen | 194 |
| **4** | Dolph Lundgren | 239 |

In [311]:

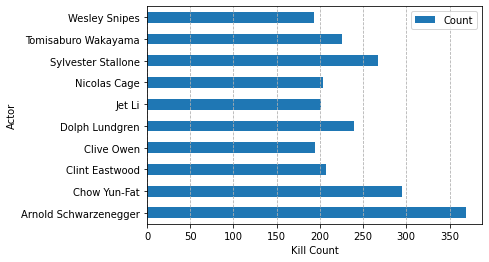
df**.**plot**.**barh(x**=**'Actor', y**=**'Count')

plt**.**ylabel('Actor')

plt**.**xlabel('Kill Count')

plt**.**grid(axis**=**'x', linestyle **=** '--')

plt**.**show()



In [ ]:

In [ ]:

QUESTION 9 (*#Create a pie chart showing the fraction of all Roman Emperors that were assassinated.*

Make sure that the pie chart **is** an even circle, labels the categories, **and** shows the percentage breakdown of the categories**.**)

In [ ]:

SOLUTION

In [ ]:

*#import the necessary library*

In [386]:

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

In [306]:

roman\_emperors **=** pd**.**read\_csv('https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/599c7a2702a96/10029932?X-Blackboard-Expiration=1650099600000&X-Blackboard-Signature=Yn0ZHx3uVQLJL83etb0f6uu4n8eAkW%2B5CXlAjbTDupQ%3D&X-Blackboard-Client-Id=100784&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27roman-emperor-reigns.csv&response-content-type=text%2Fcsv&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEPX%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJIMEYCIQDu9R92BI4bcjeEA1p1qoBqn1KY3pubU5OtsALaca4X4QIhAKbPWFZuFxaKnWzKEYDgYrH1USxyGR7C8RLtAIWk6kH5KoMECJ7%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEQABoMNTU2OTAzODYxMzYxIgydaaAFXpzoLTRnHacq1wPJhXiADl7g0%2BoqQQxNswFRkOXqsuvLr4tK0jE%2FlpqIrkO3pSNd%2BSERiMSrbkm889DJaTSvxYP%2BHFQ5%2FRVNDtOaDyvViXJ5kyV0jiUSSZ83S2t%2FjkQiU47DaE0jtrvmEglasb8Y19rZB48YWazrqDmFtZBMOvj1pNebfZhtfzWoVnP34MQ0XM3mDE6VtxAaOxuRkwPyWDY184u9216%2FepNGIxnzRN%2FD%2B3ebjhQK1EJEPXi3HDtlhLhyT6MMeDVKB2pg1w278BMY4OgzA3qoN0giJ%2BUxye43fVHbeSeZ3E5snsX4tAStLW45CSAl1%2BXSMJleL47Cs9yZlqPOkmFXCaoArgAfrxwieyAxPzy7Uy1ZoS8oTyCbdjkL0h1W3Juj79yfxwIkOxz%2F4w%2FrNJ51aTrT%2Fw3JGmIknzE1o9k%2Bc4vRiPuCQaRzqSztRAxi5kWZZSoJSGMYw6ynq43L9rQ7iYWsgcFYUIhIiGRo3ZtxSFF5yeeluEnO%2FyZn958TGCCPTMCRD9PBOqY3OtQGe3%2FWUdWVeBpg1UXNF%2FcmctCPz5SXWsYJlZLd7TLdHO3nRY0ELaEchWgvN4O8Ul36JCSb94iXQvklyLkCqFNKY0CByouh30htgN6AcigwxpfpkgY6pAEUn80ZoXFqsChUR%2FYz3UTQKti7HcGBhT2mjclFK0Ybw02oUYmrLeM%2FjEtyBALS5iMtU6yHV%2BdGeTjVjnJflbsHvIN0EaXFLV71rXAbtRU%2B%2F%2BUDLyyUHii6GPYIb8hEafd38CuGS%2FCY1N6BX3zUtAFJz2UOhBt5MAlKsMgteA7wQhIpbwdYq1XILnlz6UCM2ysd8jMY7NOztmPypQXW8Lw7TURWdQ%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220416T030000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRY32ITDAOL%2F20220416%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Signature=aeaa31ae8358bee791e0f208a082c396604975f6c3746de6ef45f0e35cc8419c')

assassinated\_emperors **=** roman\_emperors[

roman\_emperors['Cause\_of\_Death']**.**apply(**lambda** x: 'assassinated' **in** x**.**lower())]

print(assassinated\_emperors)

number\_assassinated **=** len(assassinated\_emperors)

print(number\_assassinated)

other\_deaths **=** len(roman\_emperors) **-** number\_assassinated

print(other\_deaths)

emperor **=** assassinated\_emperors["Emperor"]

cause\_of\_death **=** assassinated\_emperors["Cause\_of\_Death"]

plt**.**pie(range(len(cause\_of\_death)), labels**=**emperor,autopct**=**'%1.1f%%', startangle**=**90, radius**=**0.04 **\*** 100)

Emperor Length\_of\_Reign Cause\_of\_Death

0 Augustus 40.58 Possibly assassinated

1 Tiberius 22.50 Possibly assassinated

2 Caligula 4.83 Assassinated

3 Claudius 13.75 Possibly assassinated

5 Galba 0.58 Assassinated

7 Vitellius 0.67 Assassinated

10 Domitian 15.00 Assassinated

17 Commodus 15.00 Assassinated

18 Pertinax 0.25 Assassinated

21 Caracalla 19.00 Assassinated

22 Geta 3.00 Assassinated

24 Elagabalus 3.75 Assassinated

25 Severus Alexander 13.00 Assassinated

26 Maximinus I 3.25 Assassinated

29 Pupienus 0.25 Assassinated

30 Balbinus 0.25 Assassinated

31 Gordian III 6.00 Possibly assassinated

35 Trebonianus Gallus 2.00 Assassinated

36 Aemilian 0.16 Assassinated

38 Gallienus 15.00 Assassinated

40 Quintillus 0.83 Possibly assassinated

41 Aurelian 5.00 Assassinated

42 Tacitus 0.75 Possibly assassinated

43 Florian 0.25 Assassinated

44 Probus 6.00 Assassinated

46 Numerian 1.00 Possibly assassinated

52 Severus II 1.00 Assassinated

59 Constans I 13.00 Assassinated

65 Gratian 16.00 Assassinated

66 Valentinian II 17.00 Possibly assassinated

30

38

Out[306]:

([<matplotlib.patches.Wedge at 0x281dcd99b80>,

<matplotlib.patches.Wedge at 0x281dcda6340>,

<matplotlib.patches.Wedge at 0x281dcda6a60>,

<matplotlib.patches.Wedge at 0x281dcdb21c0>,

<matplotlib.patches.Wedge at 0x281dcdb28e0>,

<matplotlib.patches.Wedge at 0x281dcdb2fd0>,

<matplotlib.patches.Wedge at 0x281dcdbe760>,

<matplotlib.patches.Wedge at 0x281dcdbee80>,

<matplotlib.patches.Wedge at 0x281dcdc95e0>,

<matplotlib.patches.Wedge at 0x281dcdc9d00>,

<matplotlib.patches.Wedge at 0x281dc96c760>,

<matplotlib.patches.Wedge at 0x281dcdd8b50>,

<matplotlib.patches.Wedge at 0x281dcde52b0>,

<matplotlib.patches.Wedge at 0x281dcde59d0>,

<matplotlib.patches.Wedge at 0x281dcdef130>,

<matplotlib.patches.Wedge at 0x281dcdef850>,

<matplotlib.patches.Wedge at 0x281dcdeff70>,

<matplotlib.patches.Wedge at 0x281dcdfd6d0>,

<matplotlib.patches.Wedge at 0x281dcdfddf0>,

<matplotlib.patches.Wedge at 0x281dce0a550>,

<matplotlib.patches.Wedge at 0x281dce0ac70>,

<matplotlib.patches.Wedge at 0x281dce173d0>,

<matplotlib.patches.Wedge at 0x281dce17af0>,

<matplotlib.patches.Wedge at 0x281dce25250>,

<matplotlib.patches.Wedge at 0x281dce25970>,

<matplotlib.patches.Wedge at 0x281dce330d0>,

<matplotlib.patches.Wedge at 0x281dce337f0>,

<matplotlib.patches.Wedge at 0x281dce33f10>,

<matplotlib.patches.Wedge at 0x281dce3f670>,

<matplotlib.patches.Wedge at 0x281dce3fd90>],

[Text(2.6942229581241775e-16, 4.4, 'Augustus'),

Text(-0.03177675278419733, 4.399885252819952, 'Tiberius'),

Text(-0.12709043759740613, 4.398164164815964, 'Caligula'),

Text(-0.28579192629584915, 4.390708709862693, 'Claudius'),

Text(-0.507301749046913, 4.370657265837021, 'Galba'),

Text(-0.7901165380116754, 4.328477313832249, 'Vitellius'),

Text(-1.1311283790398228, 4.252122833377553, 'Domitian'),

Text(-1.5247781988342142, 4.127354048825941, 'Commodus'),

Text(-1.962085350560337, 3.938301801172242, 'Pertinax'),

Text(-2.4296267179942834, 3.668366668042922, 'Caracalla'),

Text(-2.908580193791541, 3.3015392253134848, 'Geta'),

Text(-3.373997411424452, 2.8242063429751556, 'Elagabalus'),

Text(-3.794524017596911, 2.2274621163737445, 'Severus Alexander'),

Text(-4.132833132477549, 1.5098643313542484, 'Maximinus I'),

Text(-4.347064366561511, 0.6804641011627081, 'Pupienus'),

Text(-4.3935470223869455, -0.23821117117970206, 'Balbinus'),

Text(-4.231008228849504, -1.2077124522864648, 'Gordian III'),

Text(-3.82630079602044, -2.172423121395404, 'Trebonianus Gallus'),

Text(-3.161413826326028, -3.0603043343292877, 'Aemilian'),

Text(-2.2411497112429983, -3.786455858952462, 'Gallienus'),

Text(-1.1003901448354518, -4.260180926809214, 'Quintillus'),

Text(0.1906024608964493, -4.395869732134953, 'Aurelian'),

Text(1.5247781505303664, -4.1273540666709465, 'Tacitus'),

Text(2.7628313144554104, -3.424436176637636, 'Florian'),

Text(3.7453766419133196, -2.3091456883899952, 'Probus'),

Text(4.313506920674576, -0.8681348082484307, 'Numerian'),

Text(4.33678257060909, 0.7431802844944269, 'Severus II'),

Text(3.745376722987502, 2.309145556889691, 'Constans I'),

Text(2.5604798057379368, 3.578259795544228, 'Gratian'),

Text(0.9148114866096122, 4.303849433236148, 'Valentinian II')],

[Text(1.4695761589768238e-16, 2.4, '0.0%'),

Text(-0.017332774245925817, 2.3999374106290645, '0.2%'),

Text(-0.06932205687131243, 2.398998635354162, '0.5%'),

Text(-0.15588650525228134, 2.3949320235614686, '0.7%'),

Text(-0.2767100449346798, 2.3839948722747386, '0.9%'),

Text(-0.43097265709727745, 2.3609876257266813, '1.1%'),

Text(-0.6169791158399033, 2.3193397272968466, '1.4%'),

Text(-0.8316971993641167, 2.2512840266323315, '1.6%'),

Text(-1.070228373032911, 2.1481646188212227, '1.8%'),

Text(-1.325250937087791, 2.000927273477957, '2.1%'),

Text(-1.5864982875226585, 1.8008395774437187, '2.3%'),

Text(-1.8403622244133373, 1.5404761870773573, '2.5%'),

Text(-2.0697403732346786, 1.2149793362038603, '2.8%'),

Text(-2.2542726177150265, 0.8235623625568627, '3.0%'),

Text(-2.3711260181244604, 0.37116223699784073, '3.2%'),

Text(-2.396480194029243, -0.12993336609801928, '3.4%'),

Text(-2.3078226702815474, -0.658752246701708, '3.7%'),

Text(-2.0870731614656943, -1.184958066215675, '3.9%'),

Text(-1.724407541632379, -1.669256909634157, '4.1%'),

Text(-1.2224452970416353, -2.0653395594286152, '4.4%'),

Text(-0.6002128062738827, -2.3237350509868437, '4.6%'),

Text(0.10396497867079052, -2.397747126619065, '4.8%'),

Text(0.8316971730165633, -2.2512840363659707, '5.1%'),

Text(1.50699889879386, -1.8678742781659832, '5.3%'),

Text(2.0429327137709015, -1.2595340118490883, '5.5%'),

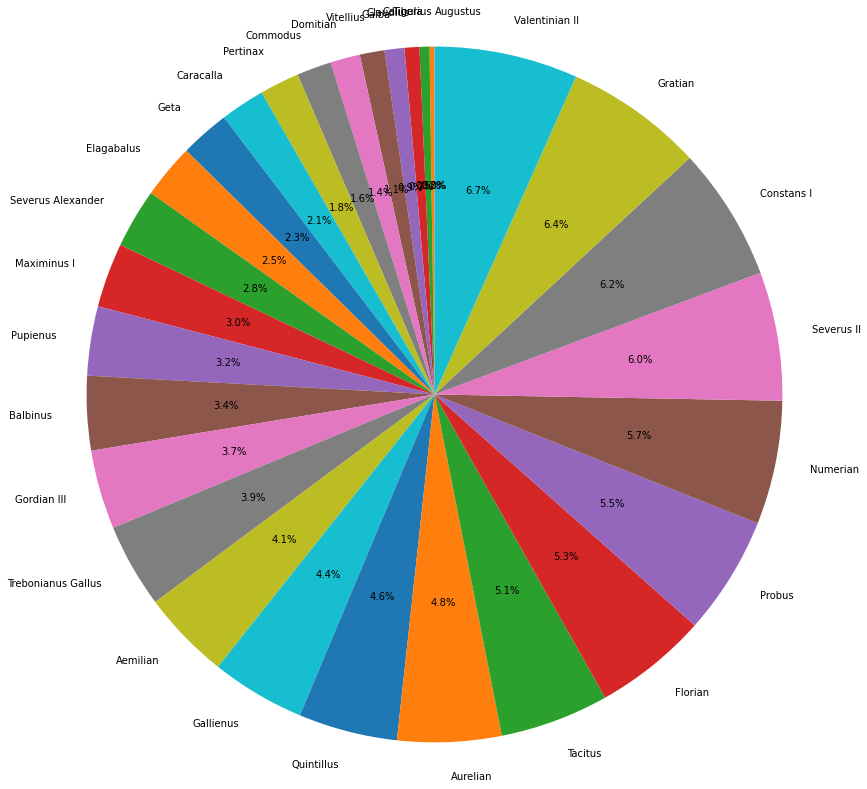
Text(2.352821956731587, -0.4735280772264167, '5.7%'),

Text(2.365517765786776, 0.40537106426968733, '6.0%'),

Text(2.042932757993183, 1.2595339401216494, '6.2%'),

Text(1.396625348584329, 1.9517780702968515, '6.4%'),

Text(0.49898808360524294, 2.347554236310626, '6.7%')])



In [ ]:

In [ ]:

QUESTION 10 (*#Create a scatter plot showing the relationship between the total revenue earned by arcades and the number of Computer Science PhDs awarded in the U.S. between 2000 and 2009.*

Don't forget to label your axes!

Color each dot according to its year)

In [ ]:

SOLUTION

In [ ]:

*#import the necessary library*

In [387]:

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

In [388]:

url **=** 'https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/599c7a2702a96/4479952?X-Blackboard-Expiration=1650099600000&X-Blackboard-Signature=LcB0xSrs94cFdOACeoSTFolhBOhzwvGpZQdMjyyynKw%3D&X-Blackboard-Client-Id=100784&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27arcade-revenue-vs-cs-doctorates.csv&response-content-type=text%2Fcsv&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEPb%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJIMEYCIQCQBHXy5XvDApNT5O0cIZ3syuJtcCbQuogbvcdZFiAzwAIhAMUqBc2J8Rkit4PXLISn10V%2BMsHmlNP9Bu3aq%2FG90h7DKoMECJ%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEQABoMNTU2OTAzODYxMzYxIgxyCZ2u0Ro37bwV39cq1wOs%2FDraLlzPWGsSvnl3hyn1o4g2scOttVHR69HT3FDnueTWIN00c7vpFNNZtwZw5fLZc9A90K11WlXXUlj1t71HLTyGiFm7VzxlfGM%2FGhhl7Fj7kkF%2Fl5c7a4e9ig73SYWcoZBMDXSQnhFTrUvzsI9u9VXSuPkB9EgIfqCPhaKZRjERpgORt92mqP8c66E5y%2BGv4HENk4D%2BYGxitF%2BlDtRFk%2FTGdJejoXXijtbVOYaGyEn3%2FyU4MZJF7drI7SgrD3NxML3V7MTSI2jV%2Fp3q%2BMUyRmxb1iOQAhp1THBwWxXBvxrpuxuD%2BWex6HqO4L0AimW7wH%2Fo47xYWJkIuBrC8rFdh425hLaWqv7eWn3axL8JErBHT0MD1QJTkYCBEwFhAJa%2BIs5vtKiIpXS%2F22z2Otgh5M6O4PySqlR3WFvOX03zPn0VszeKzIYHW%2BHYC%2BmHkssbPgZ1vAbQ0UGcl4CNAImDxXLbNbvgin3xcJzTfhB%2BRdvYzDRu3%2Fg85rrO2c9uMkE%2Fj%2FAwCVNTTk2ximR70NzkmJM%2FxZYQzWL%2Bq7uLvFL1oOuoZWgYdnXJ1yC%2FOfA2Bl%2FD6QDJDjnpLSvNLolZS%2BzO%2FydWkvjjjxxP3XaM0YtvN%2FgvIHiyi98w7LTpkgY6pAEdHJoM4oViN5fOINoudfSCDaZVlIeJOJRkns%2Bi8hzn%2BxBSUKIeNU8pH17AN7PS18Hi6furtm%2Ba11%2BEebb1u1HdDaE28lEzteDPuf6h7bMl65urzIEscjZNn%2FlYapOtSmP7lGHyMFjEDSeGpeqqr3pDvxPPRHYy5kvGV87CVMsyz2YkQTr1SOCkh88HytGtQUKVdZoeBOz4iVJADj0y6Kcru1K45Q%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220416T030000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRYZ7RPYTZI%2F20220416%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Signature=a4d1136a0d9fa0e90daefadad2540cecc6356adbd341c41dd60acc0243b55af6'

In [389]:

data **=** pd**.**read\_csv('https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/599c7a2702a96/4479952?X-Blackboard-Expiration=1650110400000&X-Blackboard-Signature=nLYpHJxyz7IjNWvHc9NGwUUXP7cbXYId77f%2F5ORnOlU%3D&X-Blackboard-Client-Id=100784&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27arcade-revenue-vs-cs-doctorates.csv&response-content-type=text%2Fcsv&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEPb%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJIMEYCIQCQBHXy5XvDApNT5O0cIZ3syuJtcCbQuogbvcdZFiAzwAIhAMUqBc2J8Rkit4PXLISn10V%2BMsHmlNP9Bu3aq%2FG90h7DKoMECJ%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEQABoMNTU2OTAzODYxMzYxIgxyCZ2u0Ro37bwV39cq1wOs%2FDraLlzPWGsSvnl3hyn1o4g2scOttVHR69HT3FDnueTWIN00c7vpFNNZtwZw5fLZc9A90K11WlXXUlj1t71HLTyGiFm7VzxlfGM%2FGhhl7Fj7kkF%2Fl5c7a4e9ig73SYWcoZBMDXSQnhFTrUvzsI9u9VXSuPkB9EgIfqCPhaKZRjERpgORt92mqP8c66E5y%2BGv4HENk4D%2BYGxitF%2BlDtRFk%2FTGdJejoXXijtbVOYaGyEn3%2FyU4MZJF7drI7SgrD3NxML3V7MTSI2jV%2Fp3q%2BMUyRmxb1iOQAhp1THBwWxXBvxrpuxuD%2BWex6HqO4L0AimW7wH%2Fo47xYWJkIuBrC8rFdh425hLaWqv7eWn3axL8JErBHT0MD1QJTkYCBEwFhAJa%2BIs5vtKiIpXS%2F22z2Otgh5M6O4PySqlR3WFvOX03zPn0VszeKzIYHW%2BHYC%2BmHkssbPgZ1vAbQ0UGcl4CNAImDxXLbNbvgin3xcJzTfhB%2BRdvYzDRu3%2Fg85rrO2c9uMkE%2Fj%2FAwCVNTTk2ximR70NzkmJM%2FxZYQzWL%2Bq7uLvFL1oOuoZWgYdnXJ1yC%2FOfA2Bl%2FD6QDJDjnpLSvNLolZS%2BzO%2FydWkvjjjxxP3XaM0YtvN%2FgvIHiyi98w7LTpkgY6pAEdHJoM4oViN5fOINoudfSCDaZVlIeJOJRkns%2Bi8hzn%2BxBSUKIeNU8pH17AN7PS18Hi6furtm%2Ba11%2BEebb1u1HdDaE28lEzteDPuf6h7bMl65urzIEscjZNn%2FlYapOtSmP7lGHyMFjEDSeGpeqqr3pDvxPPRHYy5kvGV87CVMsyz2YkQTr1SOCkh88HytGtQUKVdZoeBOz4iVJADj0y6Kcru1K45Q%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220416T060000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRYZ7RPYTZI%2F20220416%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Signature=0e119161f6517e22a989832e879c58108a2ff41c6b1396c9b3f9f965cf65aaf0')

In [390]:

data**.**head()

Out[390]:

|  | **Year** | **Total Arcade Revenue (billions)** | **Computer Science Doctorates Awarded (US)** |
| --- | --- | --- | --- |
| **0** | 2000 | 1.196 | 861 |
| **1** | 2001 | 1.176 | 830 |
| **2** | 2002 | 1.269 | 809 |
| **3** | 2003 | 1.240 | 867 |
| **4** | 2004 | 1.307 | 948 |

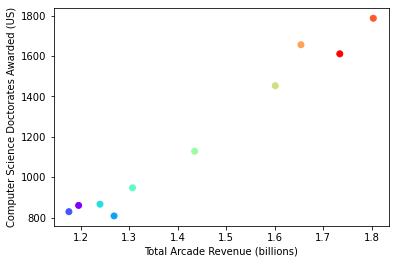
In [391]:

plt**.**scatter(x**=**"Total Arcade Revenue (billions)",y**=**"Computer Science Doctorates Awarded (US)",c**=**data**.**Year,cmap**=**"rainbow",data**=**data)

plt**.**xlabel('Total Arcade Revenue (billions)')

plt**.**ylabel('Computer Science Doctorates Awarded (US)')

plt**.**show()



In [ ]:

In [ ]: