SathishKumar_Rajendiran_Quiz2_Question11

August 21, 2020

Name: Sathish Kumar Rajendiran
Task: Quiz 2: Question 11
Date: 8/20/2020

For this question, you are to write a program that reads the data in the file EUR.pop.rev.csv. For each line in this file there is a country and seven population numbers between 1989 and 1995.

Choose one of the following to accomplish:

a. Read the data using the csv reader and represent one or more columns of the data in a NumPy array as in read_gasoline_numeric.py.
b. Read the data into a pandas DataFrame using read_csv function.

Your program must also accomplish all of the following:

1. Replace missing data with 0
2. Print each country that has more than 1,000,000 people in 1995
3. Print the average population in the United Kingdom over those seven years

Submit your code along with the output of your program.

You can use code from the class as a template, but it is essential to use appropriate variable names throughout and that you write original comments for what your program does.

[1]:
```
#import libraries
import os
import csv
import numpy as np
import pandas as pd

#verify current directory
os.getcwd()
```

[1]:
```
'/Users/sathishrajendiran/ist652-python'
```

[2]:
```
# Working with file, list and sorting
try:
    filename = 'eur.pop.rev.csv'
    df = pd.read_csv(filename,skiprows = 2)
except:
    print("Is the file in correct directory?")
```

[3]:
```
#remove last 4 rows
df = df[:-4]
#reset index
df = df.reset_index()
```

[4]:
```
#copy to new dataframe for further processing
countryDF = df
# rename columns
countryDF.columns =␣
 ↪['country','y_1989','y_1990','y_1991','y_1992','y_1993','y_1994','y_1995']
#review dataframe
countryDF.head()
```

[4]:

	country	y_1989	y_1990	y_1991	y_1992	y_1993	\
0	Austria	7602431.0	7660345.0	7790957.0	7860800.0	7909575.0	
1	Belgium	9927600.0	9947800.0	9987000.0	10068319.0	10100631.0	
2	Denmark	5129800.0	5135400.0	5146500.0	5162100.0	5180614.0	
3	Finland	4954359.0	4974383.0	4998478.0	5029300.0	5054982.0	
4	France	56269800.0	NaN	56893000.0	57217500.0	57529577.0	

	y_1994	y_1995
0	7943652.0	8054800.0
1	10130574.0	10143047.0
2	5191000.0	5251027.0
3	5098754.0	5116800.0
4	57847000.0	58265400.0

[5]:
```
#review datatype
countryDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   country  17 non-null     object
 1   y_1989   17 non-null     float64
 2   y_1990   15 non-null     float64
 3   y_1991   17 non-null     float64
 4   y_1992   17 non-null     float64
 5   y_1993   17 non-null     float64
 6   y_1994   17 non-null     float64
 7   y_1995   17 non-null     float64
dtypes: float64(7), object(1)
memory usage: 1.2+ KB
```

[12]:
```
#adjust display of decimals with comma separators on thousands
pd.options.display.float_format = '{:,.2f}'.format
```

[191]:
```
#convert country population values to float
# countryDF['y_1989'] = countryDF['y_1989'].astype('float')
# countryDF['y_1990'] = countryDF['y_1990'].astype('float')
# countryDF['y_1991'] = countryDF['y_1991'].astype('float')
# countryDF['y_1992'] = countryDF['y_1992'].astype('float')
# countryDF['y_1993'] = countryDF['y_1993'].astype('float')
# countryDF['y_1994'] = countryDF['y_1994'].astype('float')
# countryDF['y_1995'] = countryDF['y_1995'].astype('float')
```

[7]:
```
# Replace Nan with 0
countryDF.fillna(0, inplace=True)
#review dataframe verify all NaN values are replaced with 0 and all aligned as␣
 ↪float
countryDF
```

[7]:

	country	y_1989	y_1990	y_1991	y_1992	\
0	Austria	7,602,431.00	7,660,345.00	7,790,957.00	7,860,800.00	
1	Belgium	9,927,600.00	9,947,800.00	9,987,000.00	10,068,319.00	
2	Denmark	5,129,800.00	5,135,400.00	5,146,500.00	5,162,100.00	
3	Finland	4,954,359.00	4,974,383.00	4,998,478.00	5,029,300.00	
4	France	56,269,800.00	0.00	56,893,000.00	57,217,500.00	
5	Germany	61,715,000.00	62,678,000.00	79,753,000.00	80,238,000.00	
6	Iceland	253,500.00	255,708.00	259,577.00	262,193.00	
7	Ireland	3,526,600.00	3,505,500.00	3,519,000.00	3,542,000.00	
8	Italy	57,504,700.00	57,576,400.00	57,746,200.00	57,788,200.00	
9	Luxemburg	374,900.00	379,300.00	384,400.00	389,800.00	
10	Netherland	14,805,240.00	14,892,574.00	15,010,445.00	15,129,200.00	
11	Norway	4,226,901.00	4,241,473.00	4,261,930.00	4,273,634.00	
12	Portugal	10,304,700.00	0.00	9,858,500.00	9,846,000.00	
13	Spain	38,851,900.00	38,924,500.00	38,993,800.00	39,055,900.00	
14	Sweden	8,458,890.00	8,527,040.00	8,590,630.00	8,644,100.00	
15	Switzerland	6,619,973.00	6,673,850.00	6,750,693.00	6,831,900.00	
16	United Kingdom	57,236,200.00	57,410,600.00	57,649,200.00	58,888,800.00	

	y_1993	y_1994	y_1995
0	7,909,575.00	7,943,652.00	8,054,800.00
1	10,100,631.00	10,130,574.00	10,143,047.00
2	5,180,614.00	5,191,000.00	5,251,027.00
3	5,054,982.00	5,098,754.00	5,116,800.00
4	57,529,577.00	57,847,000.00	58,265,400.00
5	81,338,000.00	81,353,000.00	81,845,000.00
6	264,922.00	266,783.00	267,806.00
7	3,559,985.00	3,570,700.00	3,591,200.00
8	57,114,161.00	57,201,800.00	57,268,578.00

```
9     395,200.00    400,000.00    412,800.00
10 15,354,000.00 15,341,553.00 15,492,800.00
11  4,324,577.00  4,348,410.00  4,370,000.00
12  9,987,500.00  9,776,000.00  9,920,800.00
13 39,790,955.00 39,177,400.00 39,241,900.00
14  8,700,000.00  8,749,000.00  8,837,000.00
15  6,871,500.00  7,021,200.00  7,060,400.00
16 58,191,230.00 58,380,000.00 58,684,000.00
```

[8]:
```
#decribe the dataframe for overall count,mean,standard deviation, min,max,quartiles
countryDF.describe()
```

[8]:
```
              y_1989        y_1990        y_1991        y_1992        y_1993  \
count          17.00         17.00         17.00         17.00         17.00
mean  20,456,617.29 16,634,286.65 21,623,135.88 21,778,102.71 21,862,788.76
std   23,226,752.60 22,261,316.78 25,631,952.61 25,830,236.30 25,906,543.21
min      253,500.00          0.00    259,577.00    262,193.00    264,922.00
25%    4,954,359.00  3,505,500.00  4,998,478.00  5,029,300.00  5,054,982.00
50%    8,458,890.00  6,673,850.00  8,590,630.00  8,644,100.00  8,700,000.00
75%   38,851,900.00 14,892,574.00 38,993,800.00 39,055,900.00 39,790,955.00
max   61,715,000.00 62,678,000.00 79,753,000.00 80,238,000.00 81,338,000.00

              y_1994        y_1995
count          17.00         17.00
mean  21,870,401.53 21,989,609.29
std   25,927,445.00 26,048,995.53
min      266,783.00    267,806.00
25%    5,098,754.00  5,116,800.00
50%    8,749,000.00  8,837,000.00
75%   39,177,400.00 39,241,900.00
max   81,353,000.00 81,845,000.00
```

[9]:
```
#count Number of countries with population more than 1,000,000
#new dataframe with only country and population values for simplicity
top_countryDF = countryDF[['country','y_1995']]
top_countryDF = top_countryDF[top_countryDF['y_1995']>1000000]
print('Total Number of countries with popualtion > 1,000,000 in 1995 is :','',len(top_countryDF))

print('\nCountries with more than 1,000,000 population in 1995: \n')

#sort values in descending order based on population
top_countryDF = top_countryDF.sort_values('y_1995',ascending=False)

#iterate over each rwo and print values with thousand spearator and 0 decimals
for index, row in top_countryDF.iterrows():
```

```
    print(row['country'],': ', '{:,.0f}'.format(row["y_1995"]))
```

```
Total Number of countries with popualtion > 1,000,000 in 1995 is :  15

Countries with more than 1,000,000 population in 1995:

Germany :  81,845,000
United Kingdom :  58,684,000
France :  58,265,400
Italy :  57,268,578
Spain :  39,241,900
Netherland :  15,492,800
Belgium :  10,143,047
Portugal :  9,920,800
Sweden :  8,837,000
Austria :  8,054,800
Switzerland :  7,060,400
Denmark :  5,251,027
Finland :  5,116,800
Norway :  4,370,000
Ireland :  3,591,200
```

[13]:
```
#Average Population of United Kingdom across 7 years

print('\nAverage Population of United Kingdom across 7 years: \n')
countryDF[countryDF['country']== 'United Kingdom'].mean(axis=1)
```

```
Average Population of United Kingdom across 7 years:
```

[13]:
```
16   58,062,861.43
dtype: float64
```

[14]:
```
print("End of Quiz 12")

#reset the display option on the decimal values to default
# pd.reset_option('display.float_format')
```

```
End of Quiz 12
```