Name: Kashish Shah
Student ID: 200545460
Course: Data Programming
Course Code: BDAT 1004
Problem Set: 3
Prof: Ethan Davis

Question 1

OCCUPATION

In [1]:
```
import pandas as pd

# Read data from URL into a DataFrame
url = "https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user"
data = pd.read_csv(url, sep='|')

# Write DataFrame to Excel file
excel_file = "user_data.xlsx"
data.to_excel(excel_file, index=False)

print("Data has been successfully written to", excel_file)
```

```
Data has been successfully written to user_data.xlsx
```

In [2]:
```
import shutil

# Source file path
source_file = "user_data.xlsx"

# Destination directory where you want to save the file
destination_directory = "C:\\Users\\kashi\\OneDrive\\Desktop\\Resume"

# Copy the file to the destination directory
shutil.copy(source_file, destination_directory)

print("File has been successfully copied to", destination_directory)
```

```
File has been successfully copied to C:\Users\kashi\OneDrive\Desktop\Resume
```

In [3]:

```python
import pandas as pd

# Step 3: Import the dataset
users = pd.read_excel("Dataset_Q1.xlsx")

# Step 4: Discover the mean age per occupation
print("Step 4: Mean age per occupation")
mean_age_per_occupation = users.groupby('occupation')['age'].mean()
print(mean_age_per_occupation)
print("\n")

# Step 5: Discover the Male ratio per occupation and sort it from the most to th
print("Step 5: Male ratio per occupation (from most to least)")
male_ratio_per_occupation = users[users['gender'] == 'M'].groupby('occupation')
male_ratio_per_occupation = male_ratio_per_occupation.sort_values(ascending=Fals
print(male_ratio_per_occupation)
print("\n")

# Step 6: For each occupation, calculate the minimum and maximum ages
print("Step 6: Minimum and maximum ages per occupation")
min_max_age_per_occupation = users.groupby('occupation')['age'].agg(['min', 'max
print(min_max_age_per_occupation)
print("\n")

# Step 7: For each combination of occupation and sex, calculate the mean age
print("Step 7: Mean age per occupation and sex")
mean_age_per_occupation_sex = users.groupby(['occupation', 'gender'])['age'].mea
print(mean_age_per_occupation_sex)
print("\n")

# Step 8: For each occupation present the percentage of women and men
print("Step 8: Percentage of women and men per occupation")
total_gender_per_occupation = users.groupby('occupation')['gender'].value_counts
percentage_gender_per_occupation = total_gender_per_occupation.groupby(level=0)
print(percentage_gender_per_occupation)
```

```
Step 4: Mean age per occupation
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


Step 5: Male ratio per occupation (from most to least)
occupation
doctor           1.000000
engineer         0.970149
technician       0.962963
retired          0.928571
programmer       0.909091
executive        0.906250
scientist        0.903226
entertainment    0.888889
lawyer           0.833333
salesman         0.750000
educator         0.726316
student          0.693878
other            0.657143
marketing        0.615385
writer           0.577778
none             0.555556
administrator    0.544304
artist           0.535714
librarian        0.431373
healthcare       0.312500
homemaker        0.142857
dtype: float64


Step 6: Minimum and maximum ages per occupation
               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


Step 7: Mean age per occupation and sex
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


Step 8: Percentage of women and men per occupation
occupation     gender
administrator  M          54.430380
               F          45.569620
artist         M          53.571429
               F          46.428571
doctor         M         100.000000
educator       M          72.631579
               F          27.368421
engineer       M          97.014925
               F           2.985075
entertainment  M          88.888889
               F          11.111111
executive      M          90.625000
               F           9.375000
healthcare     F          68.750000
               M          31.250000
homemaker      F          85.714286
               M          14.285714
lawyer         M          83.333333
               F          16.666667
librarian      F          56.862745
               M          43.137255
marketing      M          61.538462
               F          38.461538
none           M          55.555556
               F          44.444444
other          M          65.714286
               F          34.285714
programmer     M          90.909091
               F           9.090909
retired        M          92.857143
               F           7.142857
salesman       M          75.000000
               F          25.000000
scientist      M          90.322581
               F           9.677419
student        M          69.387755
               F          30.612245
technician     M          96.296296
               F           3.703704
writer         M          57.777778
               F          42.222222
Name: gender, dtype: float64
```

```
C:\Users\kashi\AppData\Local\Temp\ipykernel_24024\1698577168.py:34: FutureWarn
ing: Not prepending group keys to the result index of transform-like apply. In
the future, the group keys will be included in the index, regardless of whethe
r the applied function returns a like-indexed object.
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  percentage_gender_per_occupation = total_gender_per_occupation.groupby(level
=0).apply(lambda x: 100 * x / float(x.sum()))
```

Question 2

EURO TEAMS

In [4]:

```python
import pandas as pd

# Read data from URL into a DataFrame
url = "https://raw.githubusercontent.com/guipsamora/pandas_exercises/master/02_
data = pd.read_csv(url, sep='|')

# Write DataFrame to Excel file
excel_file = "user_data1.xlsx"
data.to_excel(excel_file, index=False)

print("Data has been successfully written to", excel_file)
```

```
Data has been successfully written to user_data1.xlsx
```

In [5]:

```python
import shutil

# Source file path
source_file = "user_data1.xlsx"

# Destination directory where you want to save the file
destination_directory = "C:\\Users\\kashi\\OneDrive\\Desktop\\Resume"

# Copy the file to the destination directory
shutil.copy(source_file, destination_directory)

print("File has been successfully copied to", destination_directory)
```

```
File has been successfully copied to C:\Users\kashi\OneDrive\Desktop\Resume
```

In [6]:

```python
import pandas as pd

# Step 3: Import the dataset and assign it to a variable called euro12
euro12 = pd.read_csv("Dataset_Q2.csv")

# Step 4: Select only the Goal column
goals = euro12['Goals']
print("\nStep 4: Goals column")
print(goals)

# Step 5: How many teams participated in the Euro2012?
num_teams = euro12['Team'].nunique()
print("\nStep 5: Number of teams participated in Euro2012")
print(num_teams)

# Step 6: What is the number of columns in the dataset?
num_columns = euro12.shape[1]
print("\nStep 6: Number of columns in the dataset")
print(num_columns)

# Step 7: View only the columns Team, Yellow Cards and Red Cards and assign ther
discipline = euro12[['Team', 'Yellow Cards', 'Red Cards']]
print("\nStep 7: Discipline dataframe")
print(discipline)

# Step 8: Sort the teams by Red Cards, then by Yellow Cards
discipline_sorted = discipline.sort_values(by=['Red Cards', 'Yellow Cards'], as
print("\nStep 8: Discipline sorted by Red Cards and Yellow Cards")
print(discipline_sorted)

# Step 9: Calculate the mean Yellow Cards given per Team
mean_yellow_cards = discipline['Yellow Cards'].mean()
print("\nStep 9: Mean Yellow Cards per Team")
print(mean_yellow_cards)

# Step 10: Filter teams that scored more than 6 goals
high_scorers = euro12[euro12['Goals'] > 6]
print("\nStep 10: Teams with more than 6 goals")
print(high_scorers)

# Step 11: Select the teams that start with G
teams_starting_with_G = euro12[euro12['Team'].str.startswith('G')]
print("\nStep 11: Teams starting with G")
print(teams_starting_with_G)

# Step 12: Select the first 7 columns
first_seven_columns = euro12.iloc[:, :7]
print("\nStep 12: First 7 columns")
print(first_seven_columns)

# Step 13: Select all columns except the last 3
all_except_last_three = euro12.iloc[:, :-3]
print("\nStep 13: All columns except the last 3")
print(all_except_last_three)

# Step 14: Present only the Shooting Accuracy from England, Italy and Russia
shooting_accuracy = euro12.loc[euro12['Team'].isin(['England', 'Italy', 'Russia
print("\nStep 14: Shooting Accuracy of England, Italy, and Russia")
```

```
print(shooting_accuracy)
```

```
Step 4: Goals column
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

Step 5: Number of teams participated in Euro2012
16

Step 6: Number of columns in the dataset
35

Step 7: Discipline dataframe
                   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
5               Germany             4          0
6                Greece             9          1
7                 Italy            16          0
8           Netherlands             5          0
9                Poland             7          1
10             Portugal            12          0
11  Republic of Ireland             6          1
12               Russia             6          0
13                Spain            11          0
14               Sweden             7          0
15              Ukraine             5          0

Step 8: Discipline sorted by Red Cards and Yellow Cards
                   Team  Yellow Cards  Red Cards
6                Greece             9          1
9                Poland             7          1
11  Republic of Ireland             6          1
7                 Italy            16          0
10             Portugal            12          0
13                Spain            11          0
0               Croatia             9          0
1        Czech Republic             7          0
14               Sweden             7          0
4                France             6          0
12               Russia             6          0
```

```
3               England             5          0
8           Netherlands             5          0
15              Ukraine             5          0
2               Denmark             4          0
5               Germany             4          0

Step 9: Mean Yellow Cards per Team
7.4375

Step 10: Teams with more than 6 goals
       Team  Goals  Shots on target  Shots off target Shooting Accuracy  \
5   Germany     10               32                32             47.8%
13    Spain     12               42                33             55.9%

   % Goals-to-shots  Total shots (inc. Blocked)  Hit Woodwork  Penalty goals
\
5             15.6%                          80             2              1
13            16.0%                         100             0              1

    Penalties not scored  ...  Saves made  Saves-to-shots ratio  Fouls Won  \
5                      0  ...          10                62.6%         63
13                     0  ...          15                93.8%        102

   Fouls Conceded  Offsides  Yellow Cards  Red Cards  Subs on  Subs off  \
5              49        12             4          0       15        15
13             83        19            11          0       17        17

    Players Used
5             17
13            18

[2 rows x 35 columns]

Step 11: Teams starting with G
      Team  Goals  Shots on target  Shots off target Shooting Accuracy  \
5  Germany     10               32                32             47.8%
6   Greece      5                8                18             30.7%

  % Goals-to-shots  Total shots (inc. Blocked)  Hit Woodwork  Penalty goals  \
5            15.6%                          80             2              1
6            19.2%                          32             1              1

   Penalties not scored  ...  Saves made  Saves-to-shots ratio  Fouls Won  \
5                     0  ...          10                62.6%         63
6                     1  ...          13                65.1%         67

  Fouls Conceded  Offsides  Yellow Cards  Red Cards  Subs on  Subs off  \
5             49        12             4          0       15        15
6             48        12             9          1       12        12

   Players Used
5            17
6            20

[2 rows x 35 columns]

Step 12: First 7 columns
                  Team  Goals  Shots on target  Shots off target  \
0              Croatia      4               13                12
```

1	Czech Republic	4	13	18
2	Denmark	4	10	10
3	England	5	11	18
4	France	3	22	24
5	Germany	10	32	32
6	Greece	5	8	18
7	Italy	6	34	45
8	Netherlands	2	12	36
9	Poland	2	15	23
10	Portugal	6	22	42
11	Republic of Ireland	1	7	12
12	Russia	5	9	31
13	Spain	12	42	33
14	Sweden	5	17	19
15	Ukraine	2	7	26

	Shooting Accuracy %	Goals-to-shots	Total shots (inc. Blocked)
0	51.9%	16.0%	32
1	41.9%	12.9%	39
2	50.0%	20.0%	27
3	50.0%	17.2%	40
4	37.9%	6.5%	65
5	47.8%	15.6%	80
6	30.7%	19.2%	32
7	43.0%	7.5%	110
8	25.0%	4.1%	60
9	39.4%	5.2%	48
10	34.3%	9.3%	82
11	36.8%	5.2%	28
12	22.5%	12.5%	59
13	55.9%	16.0%	100
14	47.2%	13.8%	39
15	21.2%	6.0%	38

Step 13: All columns except the last 3

	Team	Goals	Shots on target	Shots off target	\
0	Croatia	4	13	12	
1	Czech Republic	4	13	18	
2	Denmark	4	10	10	
3	England	5	11	18	
4	France	3	22	24	
5	Germany	10	32	32	
6	Greece	5	8	18	
7	Italy	6	34	45	
8	Netherlands	2	12	36	
9	Poland	2	15	23	
10	Portugal	6	22	42	
11	Republic of Ireland	1	7	12	
12	Russia	5	9	31	
13	Spain	12	42	33	
14	Sweden	5	17	19	
15	Ukraine	2	7	26	

	Shooting Accuracy %	Goals-to-shots	Total shots (inc. Blocked)	\
0	51.9%	16.0%	32	
1	41.9%	12.9%	39	
2	50.0%	20.0%	27	
3	50.0%	17.2%	40	
4	37.9%	6.5%	65	

5	47.8%	15.6%	80
6	30.7%	19.2%	32
7	43.0%	7.5%	110
8	25.0%	4.1%	60
9	39.4%	5.2%	48
10	34.3%	9.3%	82
11	36.8%	5.2%	28
12	22.5%	12.5%	59
13	55.9%	16.0%	100
14	47.2%	13.8%	39
15	21.2%	6.0%	38

	Hit Woodwork	Penalty goals	Penalties not scored	...	Clean Sheets	\
0	0	0	0	...	0	
1	0	0	0	...	1	
2	1	0	0	...	1	
3	0	0	0	...	2	
4	1	0	0	...	1	
5	2	1	0	...	1	
6	1	1	1	...	1	
7	2	0	0	...	2	
8	2	0	0	...	0	
9	0	0	0	...	0	
10	6	0	0	...	2	
11	0	0	0	...	0	
12	2	0	0	...	0	
13	0	1	0	...	5	
14	3	0	0	...	1	
15	0	0	0	...	0	

	Blocks	Goals conceded	Saves made	Saves-to-shots ratio	Fouls Won	\
0	10	3	13	81.3%	41	
1	10	6	9	60.1%	53	
2	10	5	10	66.7%	25	
3	29	3	22	88.1%	43	
4	7	5	6	54.6%	36	
5	11	6	10	62.6%	63	
6	23	7	13	65.1%	67	
7	18	7	20	74.1%	101	
8	9	5	12	70.6%	35	
9	8	3	6	66.7%	48	
10	11	4	10	71.5%	73	
11	23	9	17	65.4%	43	
12	8	3	10	77.0%	34	
13	8	1	15	93.8%	102	
14	12	5	8	61.6%	35	
15	4	4	13	76.5%	48	

	Fouls Conceded	Offsides	Yellow Cards	Red Cards
0	62	2	9	0
1	73	8	7	0
2	38	8	4	0
3	45	6	5	0
4	51	5	6	0
5	49	12	4	0
6	48	12	9	1
7	89	16	16	0
8	30	3	5	0
9	56	3	7	1

```
10              90        10            12          0
11              51        11             6          1
12              43         4             6          0
13              83        19            11          0
14              51         7             7          0
15              31         4             5          0

[16 rows x 32 columns]

Step 14: Shooting Accuracy of England, Italy, and Russia
       Team Shooting Accuracy
3   England             50.0%
7     Italy             43.0%
12   Russia             22.5%
```

Question 3

HOUSING

In [7]:

```python
import pandas as pd
import numpy as np

# Step 1: Import the necessary libraries
print("Step 1: Libraries imported successfully.")

# Step 2: Create 3 different Series
series1 = pd.Series(np.random.randint(1, 5, size=100))
series2 = pd.Series(np.random.randint(1, 4, size=100))
series3 = pd.Series(np.random.randint(10000, 30001, size=100))
print("\nStep 2: 3 Series created successfully.")
print("Series 1:")
print(series1)
print("\nSeries 2:")
print(series2)
print("\nSeries 3:")
print(series3)

# Step 3: Create a DataFrame by joining the Series by column
df = pd.DataFrame({'bedrs': series1, 'bathrs': series2, 'price_sqr_meter': seri
print("\nStep 3: DataFrame created successfully.")
print("\nDataFrame:")
print(df)

# Step 4: Change the name of the columns
df.columns = ['bedrs', 'bathrs', 'price_sqr_meter']
print("\nStep 4: Columns renamed successfully.")
print("\nDataFrame with renamed columns:")
print(df)

# Step 5: Create a one-column DataFrame with the values of the 3 Series and ass
bigcolumn = pd.concat([series1, series2, series3], axis=0, ignore_index=True)
print("\nStep 5: One-column DataFrame created successfully.")
print("\nOne-column DataFrame:")
print(bigcolumn)

# Step 6: Check if it is going only until index 99
print("\nStep 6: Is it true that it is going only until index 99?")
print(bigcolumn.index.max() == 99)

# Step 7: Reindex the DataFrame so it goes from 0 to 299
df = df.reindex(range(300))
print("\nStep 7: DataFrame reindexed successfully.")
print("\nDataFrame after reindexing:")
print(df)
```

```
Step 1: Libraries imported successfully.

Step 2: 3 Series created successfully.
Series 1:
0     3
1     2
2     4
3     4
4     4
     ..
95    4
96    4
97    3
98    3
99    4
Length: 100, dtype: int32

Series 2:
0     3
1     2
2     1
3     1
4     1
     ..
95    3
96    1
97    3
98    1
99    1
Length: 100, dtype: int32

Series 3:
0     12520
1     20575
2     13338
3     14083
4     16817
      ...
95    10302
96    26801
97    29235
98    21339
99    12025
Length: 100, dtype: int32

Step 3: DataFrame created successfully.

DataFrame:
    bedrs  bathrs  price_sqr_meter
0       3       3            12520
1       2       2            20575
2       4       1            13338
3       4       1            14083
4       4       1            16817
..    ...     ...              ...
95      4       3            10302
96      4       1            26801
97      3       3            29235
```

```
98      3       1            21339
99      4       1            12025

[100 rows x 3 columns]

Step 4: Columns renamed successfully.

DataFrame with renamed columns:
    bedrs  bathrs  price_sqr_meter
0       3       3            12520
1       2       2            20575
2       4       1            13338
3       4       1            14083
4       4       1            16817
..    ...     ...              ...
95      4       3            10302
96      4       1            26801
97      3       3            29235
98      3       1            21339
99      4       1            12025

[100 rows x 3 columns]

Step 5: One-column DataFrame created successfully.

One-column DataFrame:
0          3
1          2
2          4
3          4
4          4
       ...
295    10302
296    26801
297    29235
298    21339
299    12025
Length: 300, dtype: int32

Step 6: Is it true that it is going only until index 99?
False

Step 7: DataFrame reindexed successfully.

DataFrame after reindexing:
     bedrs  bathrs  price_sqr_meter
0      3.0     3.0          12520.0
1      2.0     2.0          20575.0
2      4.0     1.0          13338.0
3      4.0     1.0          14083.0
4      4.0     1.0          16817.0
..     ...     ...              ...
295    NaN     NaN              NaN
296    NaN     NaN              NaN
297    NaN     NaN              NaN
298    NaN     NaN              NaN
299    NaN     NaN              NaN

[300 rows x 3 columns]
```

Question 4

WIND STATISTICS

In [8]:

```python
import pandas as pd

# Step 2: Import the dataset
data = pd.read_csv("wind.txt", sep="\s+", parse_dates=[[0, 1, 2]])

# Step 3: Replace the first 3 columns by a proper datetime index
data.set_index('Yr_Mo_Dy', inplace=True)

# Step 4: Create a function to fix year 2061 and apply it
def fix_year(x):
    year = x.year - 100 if x.year > 1989 else x.year
    return pd.to_datetime(str(year) + '-' + str(x.month) + '-' + str(x.day))

data.index = data.index.map(fix_year)

# Step 5: Set the right dates as the index
data.index = pd.to_datetime(data.index)

# Step 6: Compute how many values are missing for each location over the entire
missing_values = data.isnull().sum()

# Step 7: Compute how many non-missing values there are in total
total_non_missing = data.notnull().sum().sum()

# Step 8: Calculate the mean windspeeds of the windspeeds over all the locations
mean_windspeed = data.stack().mean()

# Step 9: Create loc_stats DataFrame and calculate min, max, mean, and standard
loc_stats = data.describe(percentiles=[])

# Step 10: Create day_stats DataFrame and calculate min, max, mean, and standard
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)

# Step 11: Find the average windspeed in January for each location
january_avg = data[data.index.month == 1].mean()

# Step 12: Downsample the record to a yearly frequency for each location
yearly_mean = data.resample('Y').mean()

# Step 13: Downsample the record to a monthly frequency for each location
monthly_mean = data.resample('M').mean()

# Step 14: Downsample the record to a weekly frequency for each location
weekly_mean = data.resample('W').mean()

# Step 15: Calculate min, max, mean, and std of windspeeds across all locations
weekly_stats = data.resample('W').agg(['min', 'max', 'mean', 'std']).iloc[:52]

# Displaying results
print("Step 6: Missing values for each location:")
print(missing_values)
print("\nStep 7: Total non-missing values:", total_non_missing)
print("\nStep 8: Mean windspeed over all locations and times:", mean_windspeed)
print("\nStep 9: Summary statistics for each location:")
```

```
print(loc_stats)
print("\nStep 10: Summary statistics for each day:")
print(day_stats)
print("\nStep 11: Average windspeed in January for each location:")
print(january_avg)
print("\nStep 12: Yearly mean windspeed for each location:")
print(yearly_mean)
print("\nStep 13: Monthly mean windspeed for each location:")
print(monthly_mean)
print("\nStep 14: Weekly mean windspeed for each location:")
print(weekly_mean)
print("\nStep 15: Weekly statistics (min, max, mean, std) for the first 52 weeks
print(weekly_stats)
```

```
Step 6: Missing values for each location:
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

Step 7: Total non-missing values: 78857

Step 8: Mean windspeed over all locations and times: 10.22788376428218

Step 9: Summary statistics for each location:
                RPT          VAL          ROS          KIL          SHA  \
count  6568.000000  6571.000000  6572.000000  6569.000000  6572.000000
mean     12.362987    10.644314    11.660526     6.306468    10.455834
std       5.618413     5.267356     5.008450     3.605811     4.936125
min       0.670000     0.210000     1.500000     0.000000     0.130000
50%      11.710000    10.170000    10.920000     5.750000     9.960000
max      35.800000    33.370000    33.840000    28.460000    37.540000

                BIR          DUB          CLA          MUL          CLO  \
count  6574.000000  6571.000000  6572.000000  6571.000000  6573.000000
mean      7.092254     9.797343     8.495053     8.493590     8.707332
std       3.968683     4.977555     4.499449     4.166872     4.503954
min       0.000000     0.000000     0.000000     0.000000     0.040000
50%       6.830000     9.210000     8.080000     8.170000     8.290000
max      26.160000    30.370000    31.080000    25.880000    28.210000

                BEL          MAL
count  6574.000000  6570.000000
mean     13.121007    15.599079
std       5.835037     6.699794
min       0.130000     0.670000
50%      12.500000    15.000000
max      42.380000    42.540000

Step 10: Summary statistics for each day:
              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
...          ...    ...        ...       ...
1978-12-27  8.08  40.08  16.708333  7.868076
1978-12-28  5.00  41.46  15.150000  9.687857
1978-12-29  8.71  29.58  14.890000  5.756836
1978-12-30  9.13  28.79  15.367500  5.540437
1978-12-31  9.59  27.29  15.402500  5.702483
```

[6574 rows x 4 columns]

Step 11: Average windspeed in January for each location:
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

Step 12: Yearly mean windspeed for each location:

Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR
1961-12-31	12.299583	10.351796	11.362369	6.958227	10.881763	7.729726
1962-12-31	12.246923	10.110438	11.732712	6.960440	10.657918	7.393068
1963-12-31	12.813452	10.836986	12.541151	7.330055	11.724110	8.434712
1964-12-31	12.363661	10.920164	12.104372	6.787787	11.454481	7.570874
1965-12-31	12.451370	11.075534	11.848767	6.858466	11.024795	7.478110
1966-12-31	13.461973	11.557205	12.020630	7.345726	11.805041	7.793671
1967-12-31	12.737151	10.990986	11.739397	7.143425	11.630740	7.368164
1968-12-31	11.835628	10.468197	11.409754	6.477678	10.760765	6.067322
1969-12-31	11.166356	9.723699	10.902000	5.767973	9.873918	6.189973
1970-12-31	12.600329	10.726932	11.730247	6.217178	10.567370	7.609452
1971-12-31	11.273123	9.095178	11.088329	5.241507	9.440329	6.097151
1972-12-31	12.463962	10.561311	12.058333	5.929699	9.430410	6.358825
1973-12-31	11.828466	10.680493	10.680493	5.547863	9.640877	6.548740
1974-12-31	13.643096	11.811781	12.336356	6.427041	11.110986	6.809781
1975-12-31	12.008575	10.293836	11.564712	5.269096	9.190082	5.668521
1976-12-31	11.737842	10.203115	10.761230	5.109426	8.846339	6.311038
1977-12-31	13.099616	11.144493	12.627836	6.073945	10.003836	8.586438
1978-12-31	12.504356	11.044274	11.380000	6.082356	10.167233	7.650658

Yr_Mo_Dy	DUB	CLA	MUL	CLO	BEL	MAL
1961-12-31	9.733923	8.858788	8.647652	9.835577	13.502795	13.680773
1962-12-31	11.020712	8.793753	8.316822	9.676247	12.930685	14.323956
1963-12-31	11.075699	10.336548	8.903589	10.224438	13.638877	14.999014
1964-12-31	10.259153	9.467350	7.789016	10.207951	13.740546	14.910301
1965-12-31	10.618712	8.879918	7.907425	9.918082	12.964247	15.591644
1966-12-31	10.579808	8.835096	8.514438	9.768959	14.265836	16.307260
1967-12-31	10.652027	9.325616	8.645014	9.547425	14.774548	17.135945
1968-12-31	8.859180	8.255519	7.224945	7.832978	12.808634	15.017486
1969-12-31	8.564493	7.711397	7.924521	7.754384	12.621233	15.762904
1970-12-31	9.609890	8.334630	9.297616	8.289808	13.183644	16.456027
1971-12-31	8.385890	6.757315	7.915370	7.229753	12.208932	15.025233
1972-12-31	9.704508	7.680792	8.357295	7.515273	12.727377	15.028716
1973-12-31	8.482110	7.614274	8.245534	7.812411	12.169699	15.441096
1974-12-31	10.084603	9.896986	9.331753	8.736356	13.252959	16.947671
1975-12-31	8.562603	7.843836	8.797945	7.382822	12.631671	15.307863
1976-12-31	9.149126	7.146202	8.883716	7.883087	12.332377	15.471448
1977-12-31	11.523205	8.378384	9.098192	8.821616	13.459068	16.590849

```
1978-12-31   9.489342   8.800466  9.089753   8.301699  12.967397  16.771370

Step 13: Monthly mean windspeed for each location:
                  RPT        VAL        ROS       KIL        SHA        BIR  \
Yr_Mo_Dy
1961-01-31  14.841333  11.988333  13.431613  7.736774  11.072759   8.588065
1961-02-28  16.269286  14.975357  14.441481  9.230741  13.852143  10.937500
1961-03-31  10.890000  11.296452  10.752903  7.284000  10.509355   8.866774
1961-04-30  10.722667   9.427667   9.998000  5.830667   8.435000   6.495000
1961-05-31   9.860968   8.850000  10.818065  5.905333   9.490323   6.574839
...               ...        ...        ...       ...        ...        ...
1978-08-31   9.645161   8.259355   9.032258  4.502903   7.368065   5.935161
1978-09-30  10.913667  10.895000  10.635000  5.725000  10.372000   9.278333
1978-10-31   9.897742   8.670968   9.295806  4.721290   8.525161   6.774194
1978-11-30  16.151667  14.802667  13.508000  7.317333  11.475000   8.743000
1978-12-31  16.175484  13.748065  15.635161  7.094839  11.398710   9.241613

                  DUB        CLA        MUL        CLO        BEL        MAL
Yr_Mo_Dy
1961-01-31  11.184839   9.245333   9.085806  10.107419  13.880968  14.703226
1961-02-28  11.890714  11.846071  11.821429  12.714286  18.583214  15.411786
1961-03-31   9.644194   9.829677  10.294138  11.251935  16.410968  15.720000
1961-04-30   6.925333   7.094667   7.342333   7.237000  11.147333  10.278333
1961-05-31   7.604000   8.177097   8.039355   8.499355  11.900323  12.011613
...               ...        ...        ...        ...        ...        ...
1978-08-31   5.650323   5.417742   7.241290   5.536774  10.466774  12.054194
1978-09-30  10.790333   9.583000  10.069333   8.939000  15.680333  19.391333
1978-10-31   8.115484   7.337742   8.297742   8.243871  13.776774  17.150000
1978-11-30  11.492333   9.657333  10.701333  10.676000  17.404667  20.723000
1978-12-31  12.077419  10.194839  10.616774  11.028710  13.859677  21.371613

[216 rows x 12 columns]

Step 14: Weekly mean windspeed for each location:
                  RPT        VAL        ROS        KIL        SHA        BIR
\
Yr_Mo_Dy
1961-01-01  15.040000  14.960000  13.170000   9.290000        NaN   9.870000
1961-01-08  13.541429  11.486667  10.487143   6.417143   9.474286   6.435714
1961-01-15  12.468571   8.967143  11.958571   4.630000   7.351429   5.072857
1961-01-22  13.204286   9.862857  12.982857   6.328571   8.966667   7.417143
1961-01-29  19.880000  16.141429  18.225714  12.720000  17.432857  14.828571
...               ...        ...        ...        ...        ...        ...
1978-12-03  14.934286  11.232857  13.941429   5.565714  10.215714   8.618571
1978-12-10  20.740000  19.190000  17.034286   9.777143  15.287143  12.774286
1978-12-17  16.758571  14.692857  14.987143   6.917143  11.397143   7.272857
1978-12-24  11.155714   8.008571  13.172857   4.004286   7.825714   6.290000
1978-12-31  14.951429  11.801429  16.035714   6.507143   9.660000   8.620000

                  DUB        CLA        MUL        CLO        BEL        MAL
Yr_Mo_Dy
1961-01-01  13.670000  10.250000  10.830000  12.580000  18.500000  15.040000
1961-01-08  11.061429   6.616667   8.434286   8.497143  12.481429  13.238571
1961-01-15   7.535714   6.820000   5.712857   7.571429  11.125714  11.024286
1961-01-22   9.257143   7.875714   7.145714   8.124286   9.821429  11.434286
1961-01-29  15.528571  15.160000  14.480000  15.640000  20.930000  22.530000
...               ...        ...        ...        ...        ...        ...
1978-12-03   9.642857   7.685714   9.011429   9.547143  11.835714  18.728571
1978-12-10  14.437143  12.488571  13.870000  14.082857  18.517143  23.061429
```

```
1978-12-17  10.208571   7.967143   9.168571   8.565714  11.102857  15.562857
1978-12-24   7.798571   8.667143   7.151429   8.072857  11.845714  18.977143
1978-12-31  13.708571  10.477143  10.868571  11.471429  12.947143  26.844286

[940 rows x 12 columns]

Step 15: Weekly statistics (min, max, mean, std) for the first 52 weeks:
              RPT                                   VAL                    \
              min    max       mean       std    min    max       mean
Yr_Mo_Dy
1961-01-01  15.04  15.04  15.040000       NaN  14.96  14.96  14.960000
1961-01-08  10.58  18.50  13.541429  2.631321   6.63  16.88  11.486667
1961-01-15   9.04  19.75  12.468571  3.555392   3.54  12.08   8.967143
1961-01-22   4.92  19.83  13.204286  5.337402   3.42  14.37   9.862857
1961-01-29  13.62  25.04  19.880000  4.619061   9.96  23.91  16.141429
1961-02-05  10.58  24.21  16.827143  5.251408   9.46  24.21  15.460000
1961-02-12  16.00  24.54  19.684286  3.587677  11.54  21.42  16.417143
1961-02-19   6.04  22.50  15.130000  5.064609  11.63  20.17  15.091429
1961-02-26   7.79  25.80  15.221429  7.020716   7.08  21.50  13.625714
1961-03-05  10.96  13.33  12.101429  0.997721   8.83  17.00  12.951429
1961-03-12   4.88  14.79   9.376667  3.732263   8.08  16.96  11.578571
1961-03-19   4.92  16.88  11.911429  3.860036   9.46  15.54  13.501429
1961-03-26   6.29  15.00   9.567143  3.613298   2.58  11.63   8.387143
1961-04-02   5.88  18.25  10.757143  5.046922   3.50  16.29   8.852857
1961-04-09   4.50  18.12  11.964286  4.604392   7.04  14.62  10.654286
1961-04-16   4.71  15.50   8.965714  3.937727   4.83  12.25   8.000000
1961-04-23   4.00  21.09  12.621429  5.676655   3.71  15.41  10.438571
1961-04-30   4.08  16.29  10.117143  4.349662   6.50  14.46   9.798571
1961-05-07   9.87  23.00  15.367143  5.025507  10.29  19.79  13.970000
1961-05-14   3.54  12.79   7.772857  3.371022   3.96  15.12   8.712857
1961-05-21   4.88  15.04   8.225714  3.631730   3.58  10.17   5.631667
1961-05-28   4.96  11.79   8.155714  2.739433   3.67  12.50   7.388571
1961-06-04   7.00  15.92  10.321429  3.099701   4.75   9.79   7.407143
1961-06-11   8.29  14.42  10.917143  2.248597   6.83  11.54   8.992857
1961-06-18   6.13  14.33  10.571429  3.009482   4.12  14.54   9.565714
1961-06-25   4.00   9.00   7.345714  1.982035   3.25   9.50   6.108571
1961-07-02   7.21  13.13  10.236667  2.557856   6.34  14.37   9.482857
1961-07-09   7.29  17.50  11.715714  3.664855   3.75  10.75   7.220000
1961-07-16   8.63  22.50  16.680000  5.168710   7.87  19.29  13.518571
1961-07-23   3.04   5.88   4.202857  1.047978   2.92   6.79   4.255714
1961-07-30   6.13  16.08  10.561429  4.157641   4.63  13.79   8.445714
1961-08-06   7.67  16.08  10.870000  2.950887   5.09  15.79   8.792857
1961-08-13   2.88  14.21  10.058333  4.422268   4.42  10.00   7.941429
1961-08-20  13.13  18.91  15.607143  2.283635   7.25  13.62  10.565714
1961-08-27   7.67  18.16  12.391429  3.395857   6.87  14.58  11.430000
1961-09-03   3.63  14.46   8.678571  4.398615   1.13  22.00   8.821429
1961-09-10   5.00  17.62  10.541429  5.207278   3.04  13.59   8.798571
1961-09-17   5.33  28.75  17.160000  7.679190   7.12  22.08  14.440000
1961-09-24   6.92  10.25   8.500000  1.267399   2.92  13.62   7.154286
1961-10-01  10.34  23.21  16.044286  4.559572   6.96  13.62  11.361667
1961-10-08   3.13  16.08  11.250000  5.596710   3.63  16.96   8.757143
1961-10-15   3.71  17.12  10.881667  4.780675   6.13  13.25   9.734286
1961-10-22  10.46  28.62  19.260000  7.888314   3.75  19.46  13.364286
1961-10-29   4.33  26.42  16.077143  7.957637   2.75  21.25  11.867143
1961-11-05   5.88  15.79  11.571429  3.369201   3.96  13.46   9.590000
1961-11-12   5.46  16.08  10.428571  3.939811   5.50  10.75   7.690000
1961-11-19   7.50  15.00  10.798571  2.784358   4.21  13.00   7.951429
1961-11-26   5.75  14.29   9.154286  3.214368   3.29  11.79   7.337143
1961-12-03   7.92  23.75  12.608571  5.704669   4.67  18.71  10.442857
```

```
1961-12-10  10.83  23.71  17.362857  4.890152   9.21  21.37  14.362857
1961-12-17   9.29  21.34  14.985714  4.095106   8.63  17.62  13.472857
1961-12-24  11.46  24.41  15.757143  4.959717   9.79  16.13  11.971429

                     ROS         ...        CLO              BEL         \
                 std    min    max  ...       mean       std    min    max
Yr_Mo_Dy                             ...
1961-01-01       NaN  13.17  13.17  ...  12.580000       NaN  18.50  18.50
1961-01-08  3.949525   7.62  12.33  ...   8.497143  1.704941   5.46  17.54
1961-01-15  3.148945   7.08  19.50  ...   7.571429  4.084293   5.25  20.71
1961-01-22  3.837785   7.29  20.79  ...   8.124286  4.783952   6.50  15.92
1961-01-29  5.170224  12.67  25.84  ...  15.640000  3.713368  14.04  27.71
1961-02-05  5.187395   9.04  19.70  ...   9.460000  2.839501   9.17  19.33
1961-02-12  3.608373  13.67  21.34  ...  14.440000  1.746749  15.21  26.38
1961-02-19  3.575012   6.13  19.41  ...  13.542857  2.531361  14.09  29.63
1961-02-26  5.147348   6.08  22.42  ...  12.730000  4.920064   9.59  23.21
1961-03-05  2.851955   8.17  13.67  ...  12.370000  1.593685  11.58  23.45
1961-03-12  3.230167   7.54  16.38  ...  10.458571  3.655113  10.21  22.71
1961-03-19  2.352867   5.25  13.96  ...  11.627143  3.099472  11.29  22.79
1961-03-26  3.657265   4.79  15.63  ...  11.481429  2.538224   8.25  21.34
1961-04-02  4.687315   5.09  14.96  ...   9.631429  3.191115   7.21  18.63
1961-04-09  2.845399   9.29  18.29  ...   7.238571  2.336182   7.62  17.16
1961-04-16  2.607118   3.92  15.79  ...   6.178571  2.161137   5.75  16.17
1961-04-23  4.631736   3.33  17.00  ...   9.551429  3.347972   6.75  19.21
1961-04-30  2.871425   2.54  14.96  ...   6.124286  2.840568   5.13  13.04
1961-05-07  3.750835   8.42  21.21  ...  11.585714  3.620819   4.79  28.08
1961-05-14  3.782947   4.63  12.33  ...   7.822857  5.460237   6.54  18.66
1961-05-21  2.468906   5.91  15.96  ...   7.114286  2.216889   6.63  12.00
1961-05-28  3.378537   3.58  20.96  ...   7.535714  2.575661   6.13  14.33
1961-06-04  1.868125   6.83  15.96  ...   9.035714  2.096989   8.71  17.00
1961-06-11  1.524836   6.04  11.58  ...   8.397143  2.158323   5.37  16.17
1961-06-18  3.509444   6.50  15.50  ...   9.594286  3.792400   7.29  25.25
1961-06-25  2.212460   5.13  10.37  ...  11.257143  2.286218  11.00  19.08
1961-07-02  2.902411   7.62  10.79  ...  10.268571  1.564144   6.96  21.87
1961-07-09  2.686658   7.41  14.92  ...  10.547143  3.657179   7.08  20.41
1961-07-16  3.849630   8.00  14.29  ...  10.157143  3.271899   5.04  20.25
1961-07-23  1.400010   4.08  12.67  ...   6.041429  1.439785   4.21  10.13
1961-07-30  3.203206   6.42  15.67  ...   9.470000  4.350268   7.87  22.17
1961-08-06  3.985226   5.54  15.59  ...   8.951429  2.903018   6.17  18.54
1961-08-13  2.053326   6.00  11.63  ...   7.235000  2.073777   5.88  14.29
1961-08-20  2.523416   9.17  14.04  ...  12.244286  2.730237   9.59  21.92
1961-08-27  3.174702   9.50  17.83  ...  12.507143  3.855302   5.75  23.38
1961-09-03  7.474025   2.42  12.75  ...   8.924286  3.993736   4.79  24.71
1961-09-10  4.003996   3.63  14.88  ...   7.565714  3.649278   5.41  15.83
1961-09-17  5.360585   7.29  26.50  ...  14.268571  5.128338  13.92  23.91
1961-09-24  3.445262   5.33  12.17  ...   6.798571  2.354092   4.54  20.96
1961-10-01  2.812482   7.87  17.58  ...  11.840000  3.908397   7.79  21.37
1961-10-08  5.060803   4.96  17.83  ...   7.662857  4.296870   4.79  17.83
1961-10-15  2.707483   4.21  20.96  ...   9.494286  3.569308   8.54  20.46
1961-10-22  5.998199   9.92  22.63  ...  16.461429  5.890511  10.88  24.71
1961-10-29  6.879973   4.79  23.09  ...  12.952857  6.277629   8.50  27.29
1961-11-05  3.900278   6.54  14.33  ...  11.864286  2.784450  10.29  20.17
1961-11-12  2.141191   3.83  20.41  ...   6.415714  2.843518   6.13  12.58
1961-11-19  3.208548   6.46  32.71  ...   6.031429  4.402588   0.71   9.59
1961-11-26  3.277904   4.42  14.46  ...   9.582857  2.747452   6.50  20.46
1961-12-03  5.107089   8.00  19.92  ...   7.911429  3.680477   5.50  19.04
1961-12-10  4.115506   9.71  20.54  ...  12.022857  4.156207  10.71  21.79
1961-12-17  3.587886   9.04  19.00  ...  11.058571  4.633398   5.50  22.83
1961-12-24  2.220866   8.08  22.13  ...   7.697143  4.637096   5.29  17.67
```

			MAL			
	mean	std	min	max	mean	std
Yr_Mo_Dy						
1961-01-01	18.500000	NaN	15.04	15.04	15.040000	NaN
1961-01-08	12.481429	4.349139	10.88	16.46	13.238571	1.773062
1961-01-15	11.125714	5.552215	5.17	16.92	11.024286	4.692355
1961-01-22	9.821429	3.626584	6.79	17.96	11.434286	4.237239
1961-01-29	20.930000	5.210726	17.50	27.63	22.530000	3.874721
1961-02-05	14.012857	4.210858	7.17	19.25	11.935714	4.336104
1961-02-12	21.832857	4.063753	17.04	21.84	19.155714	1.828705
1961-02-19	21.167143	5.910938	10.96	22.58	16.584286	4.685377
1961-02-26	16.304286	5.091162	6.67	23.87	14.322857	6.182283
1961-03-05	17.842857	4.332331	8.83	17.54	13.951667	3.021387
1961-03-12	16.701429	4.358759	5.54	22.54	14.420000	5.769890
1961-03-19	19.350000	3.779727	11.34	22.95	16.227143	4.331958
1961-03-26	14.037143	4.318069	13.13	22.50	18.134286	3.701846
1961-04-02	13.471429	4.179854	7.17	19.58	13.900000	3.924555
1961-04-09	11.712857	3.147781	7.21	15.34	11.371429	2.598271
1961-04-16	9.482857	3.641464	5.66	12.87	8.690000	2.747842
1961-04-23	13.620000	4.735096	4.96	20.46	12.470000	5.908542
1961-04-30	9.720000	2.948237	2.67	17.50	8.637143	5.108365
1961-05-07	17.548571	8.003490	3.83	26.58	14.571429	7.728504
1961-05-14	10.421429	3.968272	3.33	26.30	10.382857	7.858246
1961-05-21	9.624286	1.975853	5.91	14.96	10.612857	3.310819
1961-05-28	10.518571	3.024524	8.00	17.04	11.697143	3.811818
1961-06-04	12.298571	2.611139	10.63	17.96	13.597143	2.593586
1961-06-11	10.148571	3.993062	5.96	19.83	12.250000	4.925055
1961-06-18	15.351429	6.477887	6.13	24.71	15.025714	6.242673
1961-06-25	14.370000	2.498386	13.75	21.50	17.410000	3.063011
1961-07-02	14.535714	6.303747	8.50	16.79	12.133333	3.652313
1961-07-09	12.220000	4.537988	12.08	21.29	15.987143	3.665705
1961-07-16	13.520000	4.971060	5.96	21.96	12.524286	4.974273
1961-07-23	7.524286	2.050218	5.41	10.92	8.415714	2.133994
1961-07-30	12.841429	5.580903	6.13	25.37	13.761429	6.664574
1961-08-06	11.595714	4.901377	9.08	20.25	13.760000	4.448251
1961-08-13	10.934286	2.931302	5.88	15.16	10.125714	3.356585
1961-08-20	14.922857	4.086725	13.04	24.30	16.626667	3.934238
1961-08-27	16.251429	6.711322	8.29	22.29	16.485714	4.947608
1961-09-03	13.664286	7.678051	5.41	22.54	11.022857	6.308087
1961-09-10	10.700000	4.220584	3.37	20.25	11.034286	6.049619
1961-09-17	19.878571	4.464252	14.67	33.09	18.984286	6.332885
1961-09-24	11.018571	5.235868	5.25	14.62	9.814286	3.113507
1961-10-01	16.208571	5.091268	4.04	17.16	13.338571	4.696504
1961-10-08	8.810000	4.800403	4.83	19.62	11.410000	5.462002
1961-10-15	14.451429	4.113200	10.75	21.04	15.260000	4.098130
1961-10-22	17.477143	5.645871	13.46	33.45	23.641429	7.468377
1961-10-29	15.592857	7.056150	9.83	30.88	18.404286	8.340881
1961-11-05	16.322857	4.038493	13.37	23.58	19.195714	3.870800
1961-11-12	9.208571	2.532196	5.71	15.54	10.858571	3.690752
1961-11-19	5.875714	3.643285	2.00	13.25	5.737143	3.787654
1961-11-26	11.772857	5.407223	4.25	22.58	12.732857	6.475867
1961-12-03	11.464286	5.552648	5.88	21.29	14.725714	5.233192
1961-12-10	15.975714	4.667933	7.58	29.33	16.241429	7.345893
1961-12-17	15.112857	6.531043	6.50	21.12	14.644286	5.665006
1961-12-24	9.958571	5.065308	2.62	16.62	8.164286	5.048035

[52 rows x 48 columns]

Question 5

In [9]:
```
import pandas as pd

# Read data from URL into a DataFrame
url = "https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.t
data = pd.read_csv(url, sep='|')

# Write DataFrame to Excel file
excel_file = "user_data2.xlsx"
data.to_excel(excel_file, index=False)

print("Data has been successfully written to", excel_file)
```

```
Data has been successfully written to user_data2.xlsx
```

In [10]:
```
import shutil

# Source file path
source_file = "user_data2.xlsx"

# Destination directory where you want to save the file
destination_directory = "C:\\Users\\kashi\\OneDrive\\Desktop\\Resume"

# Copy the file to the destination directory
shutil.copy(source_file, destination_directory)

print("File has been successfully copied to", destination_directory)
```

```
File has been successfully copied to C:\Users\kashi\OneDrive\Desktop\Resume
```

In [11]:

```python
import pandas as pd

# Step 2: Import the dataset
chipo = pd.read_excel("Dataset_Q5.xlsx")

# Step 3: Assign it to a variable called chipo
chipo = pd.read_excel("Dataset_Q5.xlsx")

# Step 4: See the first 10 entries
print("Step 4: First 10 entries:")
print(chipo.head(10))

# Step 7: Print the name of all the columns
print("\nStep 7: Name of all columns:")
print(chipo.columns)

# Step 5: What is the number of observations in the dataset?
num_observations = len(chipo)

# Step 6: What is the number of columns in the dataset?
num_columns = len(chipo.columns)

# Step 8: How is the dataset indexed?
index_type = type(chipo.index)

# Step 9: Which was the most-ordered item?
most_ordered_item = chipo['item_name'].value_counts().idxmax()

# Step 10: For the most-ordered item, how many items were ordered?
most_ordered_item_count = chipo['item_name'].value_counts().max()

# Step 11: What was the most ordered item in the choice_description column?
most_ordered_choice = chipo['choice_description'].value_counts().idxmax()

# Step 12: How many items were ordered in total?
total_items_ordered = chipo['quantity'].sum()

# Step 13: Turn the item price into a float
chipo['item_price'] = chipo['item_price'].apply(lambda x: float(x.replace('$',

# Check the item price type after conversion
item_price_type_after = chipo['item_price'].dtype

# Step 14: How much was the revenue for the period in the dataset?
revenue = (chipo['quantity'] * chipo['item_price']).sum()

# Step 15: How many orders were made in the period?
num_orders = chipo['order_id'].nunique()

# Step 16: What is the average revenue amount per order?
average_revenue_per_order = revenue / num_orders

# Step 17: How many different items are sold?
num_unique_items_sold = chipo['item_name'].nunique()
```

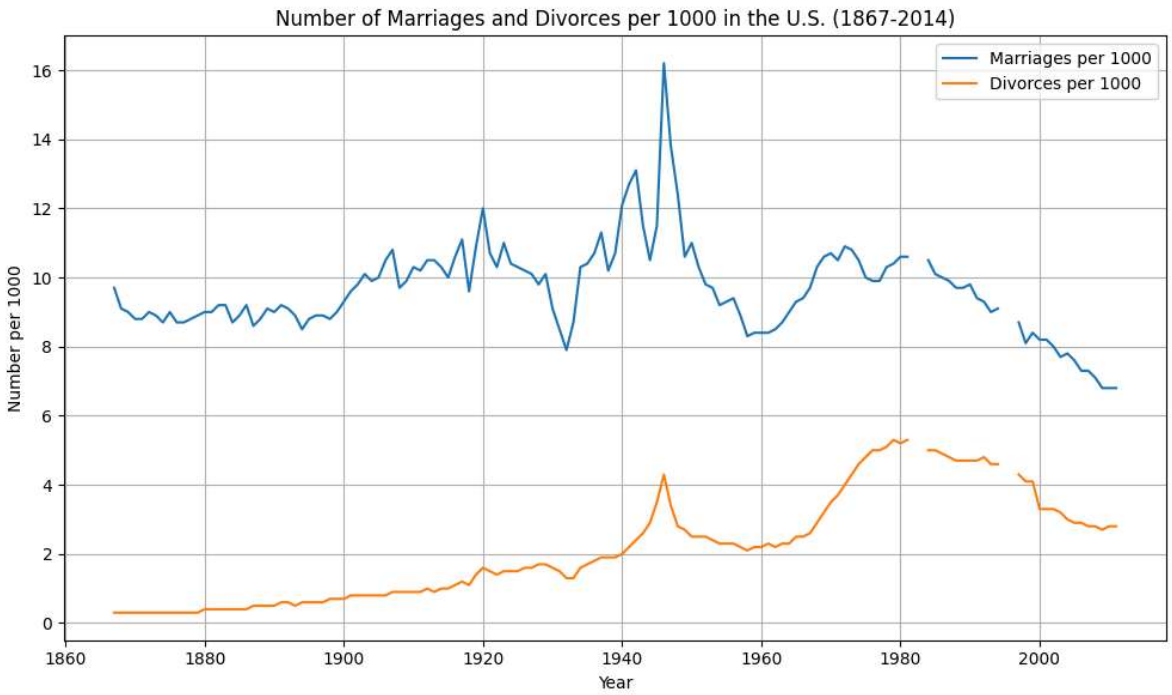
```
# Displaying results
print("\nStep 5: Number of observations in the dataset:", num_observations)
print("\nStep 6: Number of columns in the dataset:", num_columns)
print("\nStep 8: Type of index:", index_type)
print("\nStep 9: Most ordered item:", most_ordered_item)
print("\nStep 10: Number of orders for the most ordered item:", most_ordered_ite
print("\nStep 11: Most ordered item in the choice_description column:", most_ord
print("\nStep 12: Total items ordered:", total_items_ordered)
print("\nStep 13: Item price type after conversion:", item_price_type_after)
print("\nStep 14: Revenue for the period in the dataset:", revenue)
print("\nStep 15: Number of orders made in the period:", num_orders)
print("\nStep 16: Average revenue amount per order:", average_revenue_per_order
print("\nStep 17: Number of different items sold:", num_unique_items_sold)
```

```
Step 4: First 10 entries:
   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

Step 7: Name of all columns:
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')

Step 5: Number of observations in the dataset: 4622

Step 6: Number of columns in the dataset: 5

Step 8: Type of index: <class 'pandas.core.indexes.range.RangeIndex'>

Step 9: Most ordered item: Chicken Bowl

Step 10: Number of orders for the most ordered item: 726

Step 11: Most ordered item in the choice_description column: [Diet Coke]

Step 12: Total items ordered: 4972

Step 13: Item price type after conversion: float64

Step 14: Revenue for the period in the dataset: 39237.02

Step 15: Number of orders made in the period: 1834

Step 16: Average revenue amount per order: 21.39423118865867

Step 17: Number of different items sold: 50
```

Question 6

LINE PLOT

In [12]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Import the dataset
marriage_divorce_data = pd.read_csv("Dataset_Q6.csv")

# Step 2: Plotting
plt.figure(figsize=(10, 6))

# Plot marriages per capita
plt.plot(marriage_divorce_data['Year'], marriage_divorce_data['Marriages_per_100

# Plot divorces per capita
plt.plot(marriage_divorce_data['Year'], marriage_divorce_data['Divorces_per_1000

# Add labels and legend
plt.xlabel('Year')
plt.ylabel('Number per 1000')
plt.title('Number of Marriages and Divorces per 1000 in the U.S. (1867-2014)')
plt.legend()

# Show plot
plt.grid(True)
plt.tight_layout()
plt.show()
```

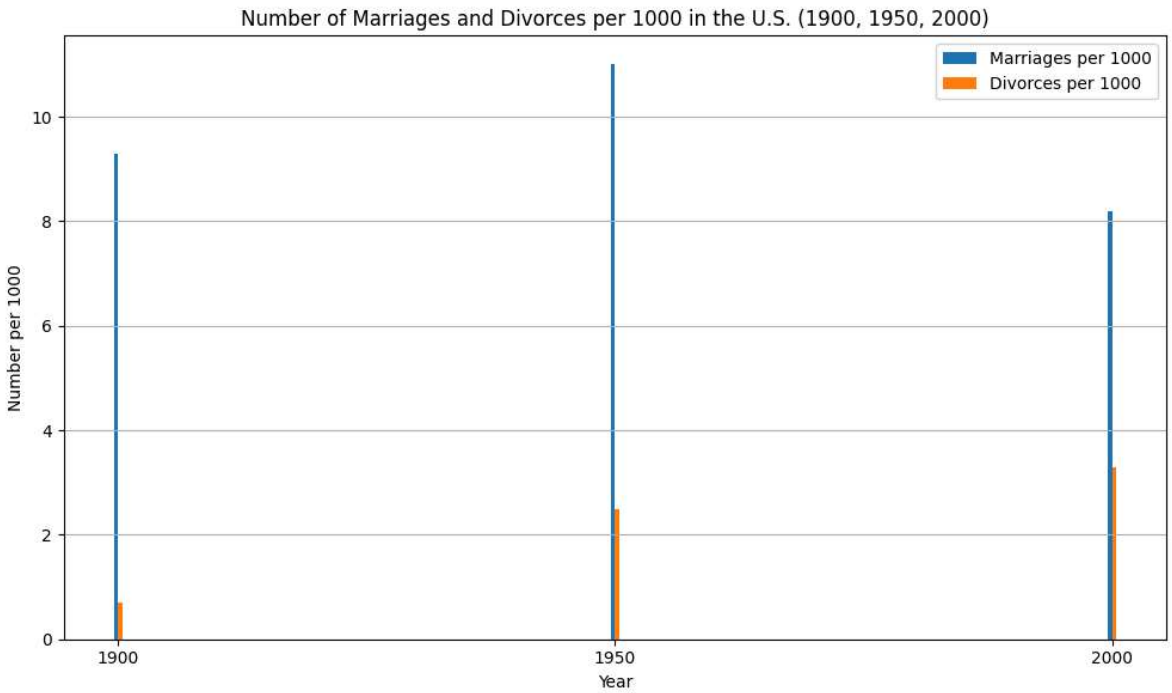


Question 7

VERTICAL BAR CHART

In [13]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Import the dataset
marriage_divorce_data = pd.read_csv("Dataset_Q6.csv")

# Step 2: Filter the data for years 1900, 1950, and 2000
years_of_interest = [1900, 1950, 2000]
filtered_data = marriage_divorce_data[marriage_divorce_data['Year'].isin(years_

# Step 3: Plotting
plt.figure(figsize=(10, 6))

# Plot marriages per capita
plt.bar(filtered_data['Year'] - 0.2, filtered_data['Marriages_per_1000'], width=

# Plot divorces per capita
plt.bar(filtered_data['Year'] + 0.2, filtered_data['Divorces_per_1000'], width=0

# Add labels and legend
plt.xlabel('Year')
plt.ylabel('Number per 1000')
plt.title('Number of Marriages and Divorces per 1000 in the U.S. (1900, 1950, 20
plt.xticks(filtered_data['Year'], filtered_data['Year'])  # Set x-axis ticks to
plt.legend()

# Show plot
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

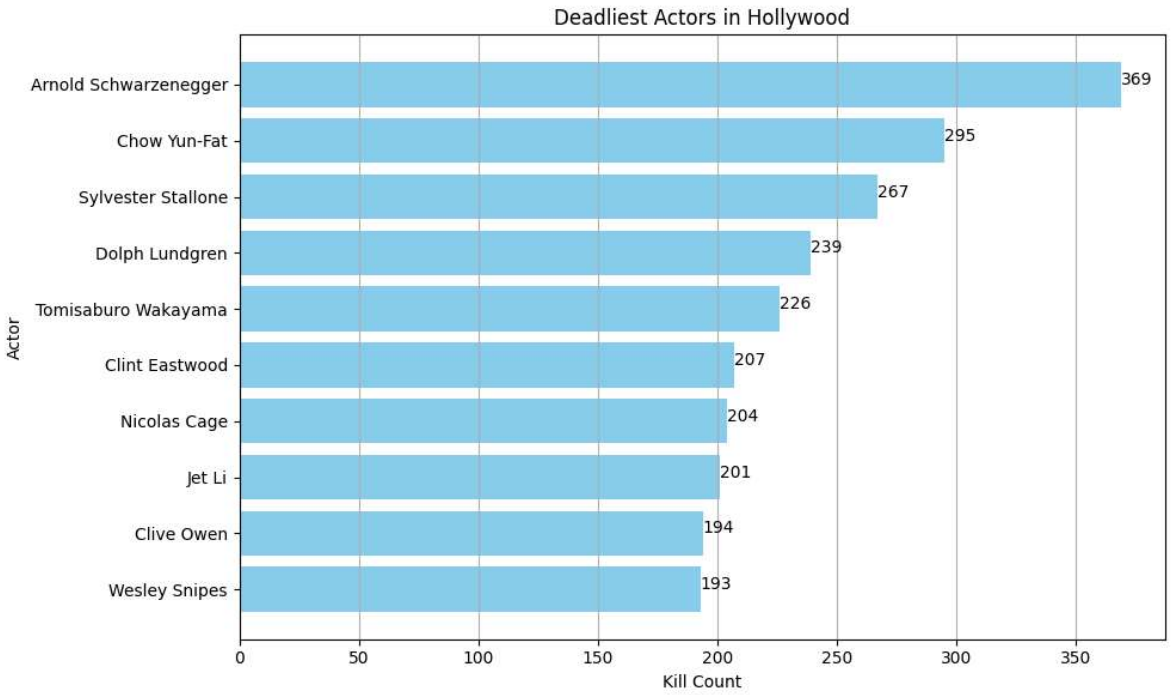


Question 8

HORIZONTAL BAR CHART

In [14]:

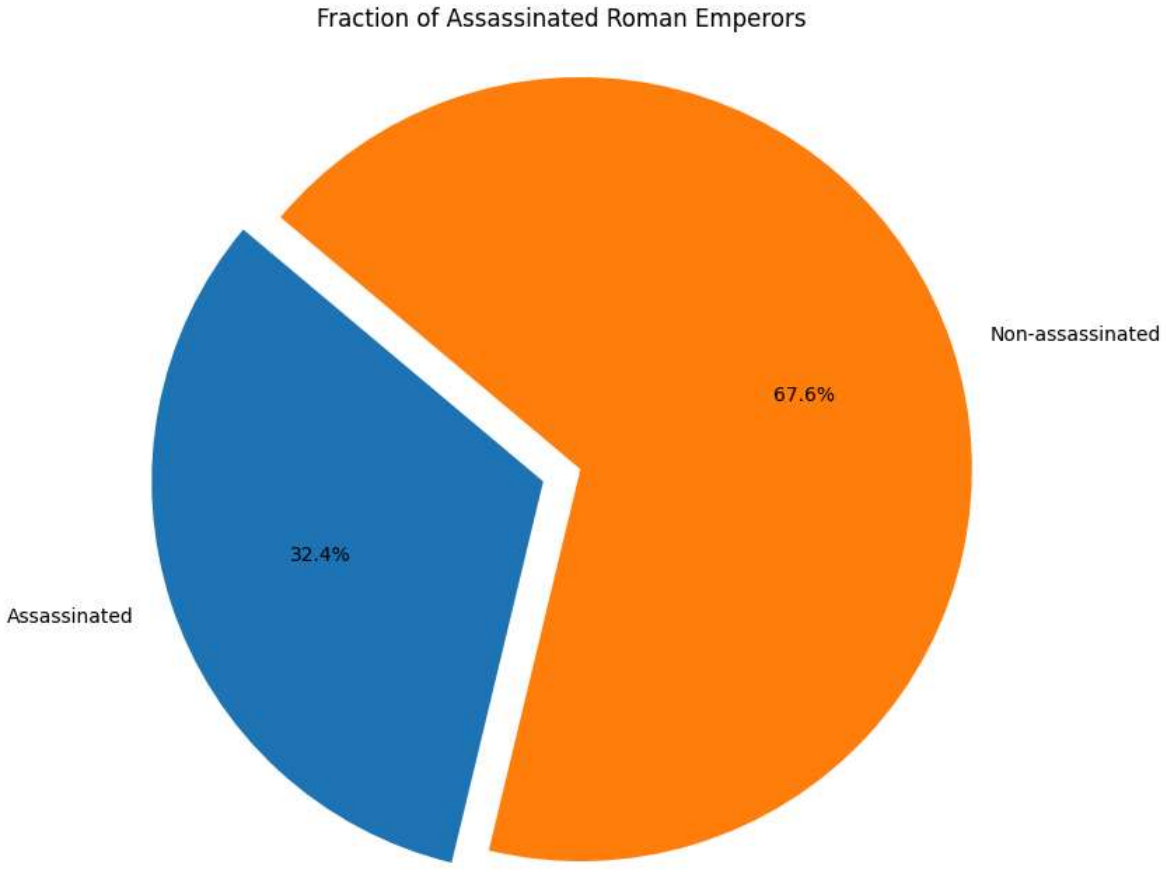
```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Import the dataset
actors_data = pd.read_csv("Dataset_Q8.csv")

# Step 2: Sort the actors by their kill count
sorted_actors = actors_data.sort_values(by='Count', ascending=True)

# Step 3: Plotting
plt.figure(figsize=(10, 6))

# Horizontal bar chart
plt.barh(sorted_actors['Actor'], sorted_actors['Count'], color='skyblue')

# Add labels and title
plt.xlabel('Kill Count')
plt.ylabel('Actor')
plt.title('Deadliest Actors in Hollywood')

# Add labels for each bar
for index, value in enumerate(sorted_actors['Count']):
    plt.text(value, index, str(value))

# Show plot
plt.grid(axis='x')
plt.tight_layout()
plt.show()
```



Question 9

In [15]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Import the dataset
emperors_data = pd.read_csv("Dataset_Q9.csv")

# Step 2: Calculate the fraction of assassinated emperors
total_emperors = len(emperors_data)
assassinated_emperors = emperors_data[emperors_data['Cause_of_Death'] == 'Assas
fraction_assassinated = len(assassinated_emperors) / total_emperors

# Step 3: Create the pie chart
labels = ['Assassinated', 'Non-assassinated']
sizes = [fraction_assassinated, 1 - fraction_assassinated]
explode = (0.1, 0)  # explode the first slice (Assassinated)

plt.figure(figsize=(8, 8))
plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', startangle=14
plt.title('Fraction of Assassinated Roman Emperors')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle

# Show plot
plt.show()
```



Question 10

SCATTER PLOT

In [16]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Import the dataset
data = pd.read_csv("Dataset_Q10.csv")

# Step 2: Extract data for plotting
years = data['Year']
revenue = data['Total Arcade Revenue (billions)']
phd_awarded = data['Computer Science Doctorates Awarded (US)']

# Step 3: Create the scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(revenue, phd_awarded, c=years, cmap='viridis', alpha=0.8)
plt.colorbar(label='Year')
plt.title('Relationship Between Arcade Revenue and Computer Science PhDs')
plt.xlabel('Total Arcade Revenue (billions)')
plt.ylabel('Computer Science Doctorates Awarded (US)')
plt.grid(True)

# Show plot
plt.show()
```