Tracy Michaels - Assignment 1

In [1]:

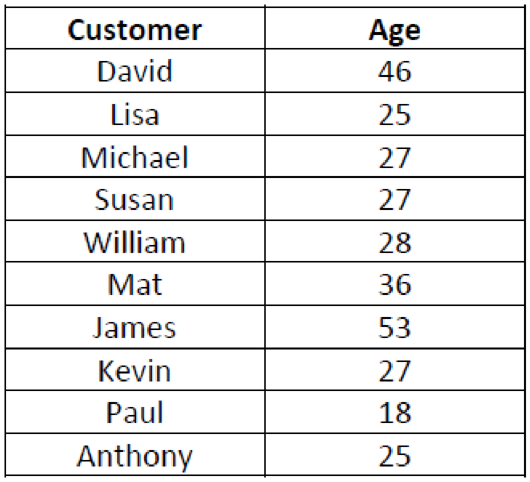
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display
from scipy import stats
from scipy.spatial.distance import euclidean
from sklearn.preprocessing import minmax_scale
```

Question 1:

Calculating the mean, median, and mode of a given data set

> **Mean** - the average of a data set; calculated by summing the all the values together dividing by the total number of values in the data set
> **Median** - The middle value of a data set
> **Mode** - Most common value represented in a data set

Given Data Set: BestBuy customer data table



Customer	Age
David	46
Lisa	25
Michael	27
Susan	27
William	28
Mat	36
James	53
Kevin	27
Paul	18
Anthony	25

In [2]:
```python
# create dataframe from table data
cust_data = {'Customer': ['David', 'Lisa', 'Michael', 'Susan', 'William',\
                          'Mat', 'James', 'Kevin', 'Paul', 'Anthony'],\
             'Age': [46, 25, 27, 27, 28, 36, 53, 27, 18, 25]}
cust_df = pd.DataFrame(data=cust_data)
#display data frame
display(cust_df)

# feature to calculate values on
feat = cust_df['Age'].to_numpy()

# calculating the mean
# (sum of all ages)/number of ages
print('mean: ' + str(np.mean(feat)))

# calculating the median
# value that is in the middle of the data set
print('median: ' + str(np.median(feat)))

# calculating the mode
# value that appears most frequently
print('mode: ' + str(stats.mode(cust_df['Age'])[0][0]))
```

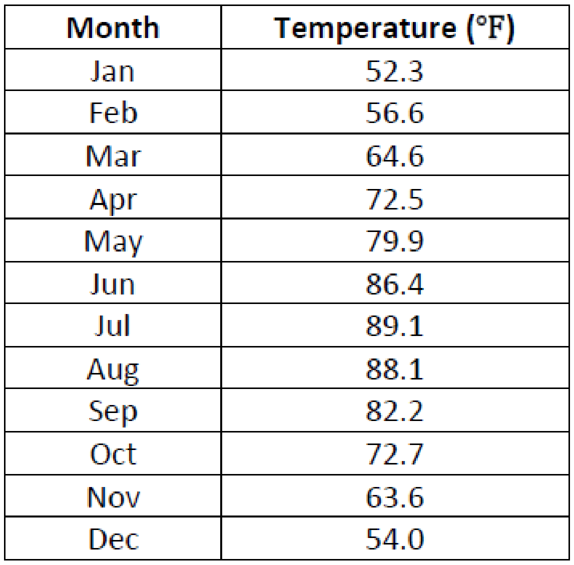
	Customer	Age
0	David	46
1	Lisa	25
2	Michael	27
3	Susan	27
4	William	28
5	Mat	36
6	James	53
7	Kevin	27
8	Paul	18
9	Anthony	25

```
mean: 31.2
median: 27.0
mode: 27
```

Question 2:

Computing the five-number summary, identifying outliers, and visualising a given data set

Given data set: Climate Data for Atlanta



Month	Temperature (°F)
Jan	52.3
Feb	56.6
Mar	64.6
Apr	72.5
May	79.9
Jun	86.4
Jul	89.1
Aug	88.1
Sep	82.2
Oct	72.7
Nov	63.6
Dec	54.0

Five-Number Summary:

> **Minimum** - this is the minimum value of a data set
> **Q1 - First Quartile** - this is the value that is at the 25% mark of a data set
> **Median** - The middle value of a data set
> **Q3 - Third Quartile** - this is the value that is at the 75% mark of a data set
> **Maximum** - this is the largest value of a data set

Finding Outliers: Q1 - (1.5 * IQR), Q3 + (1.5 * IQR)

> To find any existing outliers, find the interquartile range (IQR) by subtracting Q3 from Q1. This quantity is then multiplied by 1.5, which is then subtracted from Q1 and added to Q3. If there are values that fall outside of this new range then they are to be considered an outlier.

2.1) Computing the five-number summary:

In [3]:

```python
# create DataFrame of given values
temp_data = {'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',\
                       'Jul', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec'],\
             'Temp': [52.3, 56.6, 64.6, 72.5, 79.9, 86.4,\
                      89.1, 88.1, 82.2, 72.7, 63.6, 54.0]}
temp_df = pd.DataFrame(data=temp_data)
#display data frame
display(temp_df)

# calculate quartiles
quarts = np.percentile(temp_df['Temp'].to_numpy(), [25, 50, 75], interpolation='midpoint')

# calculate minimum and maximum
temp_min = temp_df['Temp'].min()
temp_max = temp_df['Temp'].max()

# display five-number summary
print('Five-number summary:')
print('Minimum: ' + str(temp_min))
print('Q1: ' + str(quarts[0]))
print('Median: ' + str(quarts[1]))
print('Q3: ' + str(quarts[2]))
print('Maximum: ' + str(temp_max))
print()

# calculating IQR
iqr = quarts[2] - quarts[0]
print('IQR: ' + str(iqr))

# determining outliers
iqr_e = 1.5 * iqr
out_min = quarts[0] - iqr_e
out_max = quarts[2] + iqr_e
print('Outliers: ')
for i in temp_df['Temp']:
    if(out_min > i or i > out_max):
        display(temp_df.loc[temp_df['Temp'] == i])

```

	Month	Temp
0	Jan	52.3
1	Feb	56.6
2	Mar	64.6
3	Apr	72.5
4	May	79.9
5	Jun	86.4
6	Jul	89.1
7	Aug	88.1
8	Sept	82.2
9	Oct	72.7
10	Nov	63.6
11	Dec	54.0

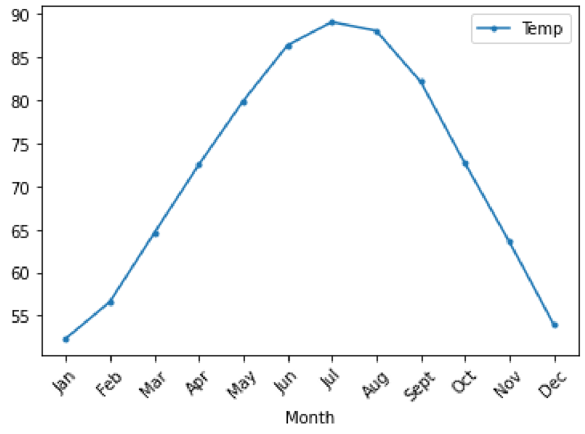
```
Five-number summary:
Minimum: 52.3
Q1: 60.1
Median: 72.6
Q3: 84.30000000000001
Maximum: 89.1

IQR: 24.20000000000001
Outliers:
```

2.2) Determining if there are outliers:

> As there were no outputs from the above determining outliers function there are no outliers, this can be confirmed as there are no values in the dataset that fall out side the range [(Q1 - 1.5 * IQR), (Q3 + 1.5 * IQR)]

2.3) Visualizing Data:

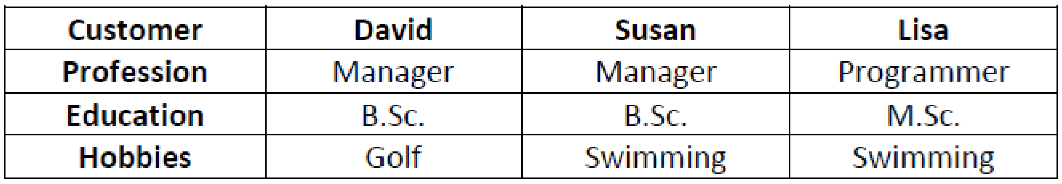
In [4]:
```python
# displaying plot of data set
ax = plt.gca()
temp_df.plot(kind='line', x='Month', y='Temp', ax=ax, marker='.')
plt.xticks([x for x in range(len(temp_df['Month']))], temp_df['Month'].tolist(), rotation=45)
plt.show()
```



> Based on this visualization, it is clear to see that the temperature is higher in the middle months

Question 3:

Given the following table of customer information



Customer	David	Susan	Lisa
Profession	Manager	Manager	Programmer
Education	B.Sc.	B.Sc.	M.Sc.
Hobbies	Golf	Swimming	Swimming

3.1) Types of attibutes in chart:

> - Profession - Categorical - Nominal
> - Education - Categorical - Ordinal
> - Hobbies - Categorical - Nominal

3.2) Computing the similarity values between "David" and "Susan":

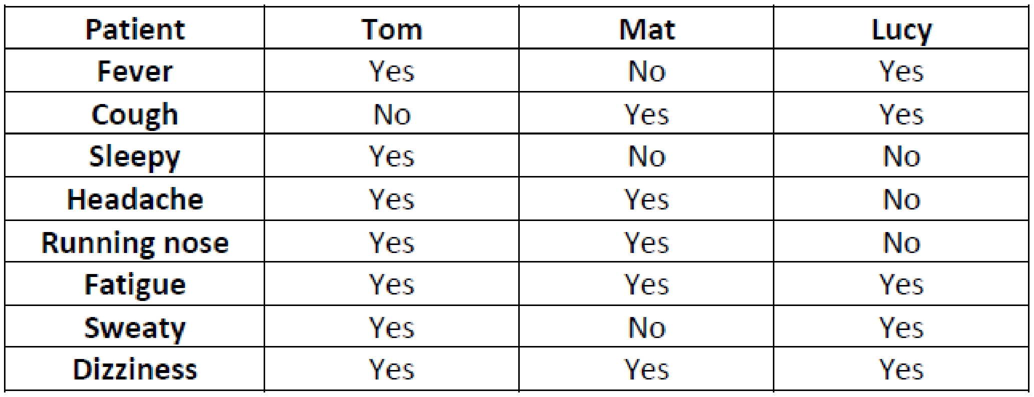
- using Simple matching:
- To calculate the similarity the number of attibutes that both customers have in common will be divided by the total number of attibutes.
- in this case David and Susan both have the same profession and education, while their hobbies differ
- therefore their similarity is said to be 2/3

3.3) Computing the similarity values between "Susan" and "Lisa":

- using the same method as in 3.2, Susan and Lisa have the same Hobbies, but their Profession and Education differ
- therefore their similarity is said to be 1/3

Question 4:

Given the following table of patient information:



Patient	Tom	Mat	Lucy
Fever	Yes	No	Yes
Cough	No	Yes	Yes
Sleepy	Yes	No	No
Headache	Yes	Yes	No
Running nose	Yes	Yes	No
Fatigue	Yes	Yes	Yes
Sweaty	Yes	No	Yes
Dizziness	Yes	Yes	Yes

4.1) Types of attributes in table:

- All attributes are binary

4.2) Computing the similarity values between "Tom" and "Mat"

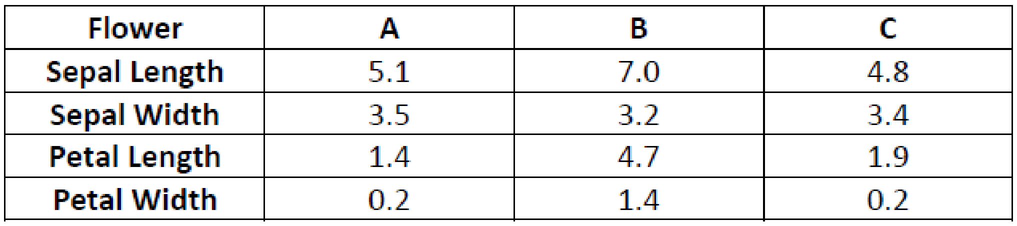
- Using Jaccard: to compute the similarity, the attributes which are mutually absent are discarded and all others are given equal weight for matches and non-matches
- in the case of Tom and Mat they have 0 mutual absences, 4 matches, and 4 non-matches
- using the formula (matches)/(non-matches + matches) = 4/8 = 1/2
- therefore Tom and Mat have a similarity of 1/2

4.3) Computing the similarity values between "Mat" and "Lucy"

- using the above method:
- 1 mutual absences, 3 matches, 4 non-matches
- discarding the mutual absence, (3 matches)/(4 non-matches + 3 matches) = 3/7
- Therefore Mat and Lucy have a similarity of 3/7

Question 5:

Given the following table of Fisher's iris data:



Flower	A	B	C
Sepal Length	5.1	7.0	4.8
Sepal Width	3.5	3.2	3.4
Petal Length	1.4	4.7	1.9
Petal Width	0.2	1.4	0.2

In [5]:
```python
# turn table into dataframe
iris_data = {'Flower': ['A', 'B', 'C',],\
             'Sepal Length': [5.1, 7.0, 4.8],\
             'Sepal Width': [3.5, 3.2, 3.4],\
             'Petal Length': [3.5, 3.2, 3.4],\
             'Petal Width': [3.5, 3.2, 3.4]}

iris_df = pd.DataFrame(data=iris_data)
iris_df.set_index('Flower', inplace=True)
display(iris_df)
```

	Sepal Length	Sepal Width	Petal Length	Petal Width
Flower				
A	5.1	3.5	3.5	3.5
B	7.0	3.2	3.2	3.2
C	4.8	3.4	3.4	3.4

5.1) Types of attributes in table:

- Attributes are numeric type

5.2) Type of similarity measure chosen:

- Euclidean distance = $\sqrt{(x_{i,1} - x_{j,1})^2 + (x_{i,2} - x_{j,2})^2 + \ldots + (x_{i,p} - x_{j,p})^2}$

- to convert distance into similarity use formula: $\frac{1}{1+(Euclidean Distance)}$

5.3) Computing the similarity values between "A" and "B":

In [6]:
```python
A = iris_df.loc['A'].to_numpy()
B = iris_df.loc['B'].to_numpy()

euc_dis = euclidean(A, B)
print(f'Euclidean distance = {euc_dis}')
print(f'Similarity: {1/(1+euc_dis)}')
```

```
Euclidean distance = 1.9697715603592212
Similarity: 0.3367262362358406
```

5.4) Computing the similarity values between "B" and "C":

In [7]:
```python
B = iris_df.loc['B'].to_numpy()
C = iris_df.loc['C'].to_numpy()

euc_dis = euclidean(B, C)
print(f'Euclidean distance = {euc_dis}')
print(f'Similarity: {1/(1+euc_dis)}')
```

```
Euclidean distance = 2.227105745132009
Similarity: 0.30987518816464865
```

Question 6:

Given the following table of Customer information data and ranking options:

Customer	Kevin	John	Daniel
Credit Score Range	Excellent	Very good	Good
Salary Range	High	Very High	Medium
Age	Senior	Middle Age	Young

The ranking options within each attribute are provided in the following tables.

Credit Score Range
Excellent
Very good
Good
Fair
Poor

Salary Range
Very High
High
Medium
Low

Age
Senior
Middle Age
Young

6.1) Tpyes of attributes in table:

- Ordinal

6.2) Compute the similarity values between "Kevin" and "John": (note: problem number changed to allign with estabolished convention of rest of assignment)

- First finding the numeric ranks of each attribute using the following formula:

$$z_f = \frac{r_f - 1}{r_{max} - 1}$$

- where r is a ranking starting from 1 for the lowest rank and increasing by 1 for each incremented rank up to r_{max}

Credit Score = [1, 0.75, 0.50, 0.25, 0]
Salary Range = [1, 0.67, 0.33, 0]
Age = [1, 0.50, 0]

- mapping new values to customer table (will use a DataFrame for this):

In [8]:
```python
cust_data2 = {'Customer': ['Kevin', 'John', 'Daniel'],\
              'Credit Score Range': [1, 0.75, 0.50],\
              'Salary Range': [0.67, 1, 0.33],\
              'Age': [1, 0.50, 0]}
cust_df2 = pd.DataFrame(data=cust_data2)
cust_df2.set_index('Customer', inplace=True)
display(cust_df2)

```

	Credit Score Range	Salary Range	Age
Customer			
Kevin	1.00	0.67	1.0
John	0.75	1.00	0.5
Daniel	0.50	0.33	0.0

- computing the Euclidean distance and similarity between Kevin and John:

In [9]:
```python
K = cust_df2.loc['Kevin'].to_numpy()
J = cust_df2.loc['John'].to_numpy()

euc_dis = euclidean(K, J)
print(f'Euclidean distance = {euc_dis}')
print(f'Similarity: {1/(1+euc_dis)}')
```

```
Euclidean distance = 0.6491532946846993
Similarity: 0.6063717686057738
```

6.3) Computing the similarity values between "John" and "Daniel": (note: problem number changed)

- using the same method as in 6.2:

In [10]:
```python
J = cust_df2.loc['John'].to_numpy()
D = cust_df2.loc['Daniel'].to_numpy()

euc_dis = euclidean(J, D)
print(f'Euclidean distance = {euc_dis}')
print(f'Similarity: {1/(1+euc_dis)}')
```

```
Euclidean distance = 0.8725823743349391
Similarity: 0.5340219013623672
```

Question 7:

Normalizing the following dataset using the min-max normalization method to range [0, 1]:

Patient	Tom	Mat	Lucy	Brian
Height (feet)	5.7	6.2	5.1	6.4

Formula for min-max normalization:

$$v_i' = \frac{v_i - min_A}{max_A - min_A} \cdot \left(new_{max A} - new_{minA}\right) + new_{min A}$$

converting table to DataFrame:

In [11]:
```python
pat_data = {'Patient': ['Tom', 'Mat', 'Lucy', 'Brian'],\
            'Height (feet)': [5.7, 6.2, 5.1, 6.4]}
pat_df = pd.DataFrame(data=pat_data)
display(pat_df)
```

	Patient	Height (feet)
0	Tom	5.7
1	Mat	6.2
2	Lucy	5.1
3	Brian	6.4

Normalizing Data:

In [12]:
```python
# get feature to normalize
h_arr = pat_df['Height (feet)'].to_numpy()

# normalize feature
h_arr_norm = minmax_scale(h_arr)

# insert into dataframe
pat_df['Height (norm)'] = h_arr_norm
display(pat_df)

```

	Patient	Height (feet)	Height (norm)
0	Tom	5.7	0.461538
1	Mat	6.2	0.846154
2	Lucy	5.1	0.000000
3	Brian	6.4	1.000000