Name : Kalyani Dhawade

Roll No. : 31416

Class : TE - 4 [K - 4]

Descriptive Statistics - Measures of Central Tendency and variability

Perform the following operations on any open-source dataset (e.g., data.csv)

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.

2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris- versicolor' of iris.csv dataset.

Provide the codes with outputs and explain everything that you do in this step.

THEORY

Measures of Central Tendency :

Measures of central tendency describe the center of the data, and are often represented by the mean, the median, and the mode.

Mean :

Mean represents the arithmetic average of the data.

$A = \frac{1}{n}\sum_{i=1}^{n} a_i$

Median :

In simple terms, median represents the 50th percentile, or the middle value of the data, that separates the distribution into two halves

$Median = [\frac{n+1}{2}]^{th} term$

Mode :

Mode represents the most frequent value of a variable in the data. This is the only central tendency measure that can be used with categorical variables, unlike the mean and the median which can be used only with quantitative data.

Mode = Observation with maximum number of occurence

Measures of Dispersion/variability :

The most popular measures of dispersion are standard deviation, variance, and the interquartile range.

Standard Deviation :

Standard deviation is a measure that is used to quantify the amount of variation of a set of data values from its mean.

$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$

Task 1

In [1]:
```
#importing the necessary library
import pandas as pd
```

In [4]:
```
#Datset used for the task : Mall_Customers.csv
ds = pd.read_csv("Mall_Customers.csv")
ds.head()
```

Out[4]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [5]:
```
ds.shape
```

Out[5]:
```
(200, 5)
```

In [6]:
```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
```

```
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [7]:

```
ds.isnull().sum() # checking for missing values
```

Out[7]:

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [8]:

```
ds.isnull().sum().sum() # total no. of missing values
```

Out[8]:

```
0
```

In [9]:

```
uni = ds["Gender"].unique()  # To find the unique values in Gender series
uni
```

Out[9]:

```
array(['Male', 'Female'], dtype=object)
```

In [10]:

```
grp= ds.groupby('Gender') # Grouping data of gender series
grp.first() # Prints the first data of the grouped data
```

Out[10]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
Gender				
Female	3	20	16	6
Male	1	19	15	39

In [11]:

```
dm = grp.get_group('Male') # using just Male group
dm.head()
```

Out[11]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
8	9	Male	64	19	3
10	11	Male	67	19	14
14	15	Male	37	20	13

In [12]:

```
df = grp.get_group('Female') # using Female group
df.head()
```

Out[12]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6

In [13]:

```
print("Mean of Annual Income By Male : ")
dm['Annual Income (k$)'].mean()
```

```
Mean of Annual Income By Male :
```

Out[13]:

```
62.22727272727273
```

In [14]:

```
print("Mean of Annual Income By Female : ")
df['Annual Income (k$)'].mean()
```

```
Mean of Annual Income By Female :
```

Out[14]:

```
59.25
```

In [15]:

```
print("Median of Annual Income By Male : ")
dm['Annual Income (k$)'].median()
```

```
Median of Annual Income By Male :
```

Out[15]:

```
62.5
```

In [16]:

```
print("Median of Annual Income By Female : ")
df['Annual Income (k$)'].median()
```

```
Median of Annual Income By Female :
```

Out[16]:

```
60.0
```

In [17]:

```
print("Mode of Annual Income by Male : ")
dm['Annual Income (k$)'].mode()
```

```
Mode of Annual Income by Male :
```

Out[17]:

```
0    54
dtype: int64
```

In [18]:

```
print("Mode of Annual Income by Female : ")
df['Annual Income (k$)'].mode()
```

```
Mode of Annual Income by Female :
```

Out[18]:

```
0    78
dtype: int64
```

In [19]:

```
## To check for max, min and standard deviation for male group
dm.describe()
```

Out[19]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	88.000000	88.000000	88.000000	88.000000
mean	104.238636	39.806818	62.227273	48.511364
std	57.483830	15.514812	26.638373	27.896770
min	1.000000	18.000000	15.000000	1.000000
25%	59.500000	27.750000	45.500000	24.500000
50%	106.500000	37.000000	62.500000	50.000000
75%	151.250000	50.500000	78.000000	70.000000
max	200.000000	70.000000	137.000000	97.000000

In [20]:

```
##To check min, max,standard deviation for female group
df.describe()
```

Out[20]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	112.000000	112.000000	112.000000	112.000000
mean	97.562500	38.098214	59.250000	51.526786
std	58.276412	12.644095	26.011952	24.114950
min	3.000000	18.000000	16.000000	5.000000
25%	46.750000	29.000000	39.750000	35.000000
50%	94.500000	35.000000	60.000000	50.000000
75%	148.250000	47.500000	77.250000	73.000000
max	197.000000	68.000000	126.000000	99.000000

Task 2

In [21]:

```
# importing libraries
import numpy as np
from statistics import stdev
```

In [22]:

```
# dataset loading
data = pd.read_csv('IRIS.csv')
data1 = data.copy()
data.head()
```

Out[22]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa

	sepal_length	sepal_width	petal_length	petal_width	species
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [23]:

```
data.shape
```

Out[23]:

```
(150, 5)
```

In [24]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [25]:

```
data.isnull().sum().sum()
```

Out[25]:

```
0
```

In [26]:

```
data.species.nunique()
```

Out[26]:

```
3
```

In [27]:

```
data.species.unique()
```

Out[27]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

Measures of Central Tendency (ungrouped) :

In [28]:

```
data.describe()
```

Out[28]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000

	sepal_length	sepal_width	petal_length	petal_width
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [29]:

```
#To calculate mode =
for i in data.columns[:4]:
    print("Mode of",i,"is:")
    print(data[i].mode()[0])
    x = data[[i]].eq(data[i].mode()[0]).sum()
    print("Count of the mode value of",i,"is:")
    print(x)
    print("\n")
```

```
Mode of sepal_length is:
5.0
Count of the mode value of sepal_length is:
sepal_length    10
dtype: int64


Mode of sepal_width is:
3.0
Count of the mode value of sepal_width is:
sepal_width    26
dtype: int64


Mode of petal_length is:
1.5
Count of the mode value of petal_length is:
petal_length    14
dtype: int64


Mode of petal_width is:
0.2
Count of the mode value of petal_width is:
petal_width    28
dtype: int64
```

In [30]:

```
#To calculate median =
for i in data.columns[:4]:
    print("median of",i,"is:")
    print(data[i].median())
    print("\n")
```

```
median of sepal_length is:
5.8


median of sepal_width is:
3.0


median of petal_length is:
```

```
4.35


median of petal_width is:
1.3
```

Measures of Central Tendency (Grouped on Categorical Variable) :

In [31]:

```
# now we'll use data1 dataframe which is copied earlier
# data will contain our original datasaet
for i in data1.columns[:4]:
        print(str(i))
        print('Mean separated by species')
        print(data1.groupby('species')[i].mean())
        print("\n")

        print('Percentile separated by species')
        print("Q1 [25%] : ")
        print(data1.groupby('species')[i].quantile(0.25))
        print("\n")

        print('Median [50%] separated by species')
        print(data1.groupby('species')[i].median())
        print("\n")

        print('Percentile separated by species')
        print("Q3 [75%] : ")
        print(data1.groupby('species')[i].quantile(0.75))
        print("\n")

        print("Standard Deviation separated by species")
        d0 = data1[i].where(data1['species'] == 'Iris-setosa')
        d0.dropna(inplace = True)
        print('Iris-setosa  '+str(stdev(d0)))

        d1= data1[i].where(data1['species'] == 'Iris-versicolor')
        d1.dropna(inplace=True)
        print('Iris-versicolor '+str(stdev(d1)))

        d2= data1[i].where(data1['species'] == 'Iris-virginica')
        d2.dropna(inplace=True)
        print('Iris-virginica '+str(stdev(d2)))
        print('\n')
        print("\n")
```

```
sepal_length
Mean separated by species
species
Iris-setosa        5.006
Iris-versicolor    5.936
Iris-virginica     6.588
Name: sepal_length, dtype: float64


Percentile separated by species
Q1 [25%] :
species
Iris-setosa        4.800
```

```
Iris-versicolor    5.600
Iris-virginica     6.225
Name: sepal_length, dtype: float64


Median [50%] separated by species
species
Iris-setosa        5.0
Iris-versicolor    5.9
Iris-virginica     6.5
Name: sepal_length, dtype: float64


Percentile separated by species
Q3 [75%] :
species
Iris-setosa        5.2
Iris-versicolor    6.3
Iris-virginica     6.9
Name: sepal_length, dtype: float64


Standard Deviation separated by species
Iris-setosa  0.3524896872134513
Iris-versicolor 0.5161711470638634
Iris-virginica 0.6358795932744321




sepal_width
Mean separated by species
species
Iris-setosa        3.418
Iris-versicolor    2.770
Iris-virginica     2.974
Name: sepal_width, dtype: float64


Percentile separated by species
Q1 [25%] :
species
Iris-setosa        3.125
Iris-versicolor    2.525
Iris-virginica     2.800
Name: sepal_width, dtype: float64


Median [50%] separated by species
species
Iris-setosa        3.4
Iris-versicolor    2.8
Iris-virginica     3.0
Name: sepal_width, dtype: float64


Percentile separated by species
Q3 [75%] :
species
Iris-setosa        3.675
Iris-versicolor    3.000
Iris-virginica     3.175
Name: sepal_width, dtype: float64
```

```
Standard Deviation separated by species
Iris-setosa  0.38102439795469095
Iris-versicolor 0.3137983233784114
Iris-virginica 0.32249663817263746




petal_length
Mean separated by species
species
Iris-setosa        1.464
Iris-versicolor    4.260
Iris-virginica     5.552
Name: petal_length, dtype: float64


Percentile separated by species
Q1 [25%] :
species
Iris-setosa        1.4
Iris-versicolor    4.0
Iris-virginica     5.1
Name: petal_length, dtype: float64


Median [50%] separated by species
species
Iris-setosa        1.50
Iris-versicolor    4.35
Iris-virginica     5.55
Name: petal_length, dtype: float64


Percentile separated by species
Q3 [75%] :
species
Iris-setosa        1.575
Iris-versicolor    4.600
Iris-virginica     5.875
Name: petal_length, dtype: float64


Standard Deviation separated by species
Iris-setosa  0.17351115943644546
Iris-versicolor 0.46991097723995795
Iris-virginica 0.5518946956639834




petal_width
Mean separated by species
species
Iris-setosa        0.244
Iris-versicolor    1.326
Iris-virginica     2.026
Name: petal_width, dtype: float64


Percentile separated by species
Q1 [25%] :
species
Iris-setosa        0.2
Iris-versicolor    1.2
```

```
Iris-virginica     1.8
Name: petal_width, dtype: float64


Median [50%] separated by species
species
Iris-setosa        0.2
Iris-versicolor    1.3
Iris-virginica     2.0
Name: petal_width, dtype: float64


Percentile separated by species
Q3 [75%] :
species
Iris-setosa        0.3
Iris-versicolor    1.5
Iris-virginica     2.3
Name: petal_width, dtype: float64


Standard Deviation separated by species
Iris-setosa  0.10720950308167838
Iris-versicolor 0.19775268000454405
Iris-virginica 0.27465005563666733
```

In [32]:

```python
# mode for groupby species
print('Mode separated by species')
print(data1.groupby('species').agg(lambda x:x.value_counts().index[0]))
print("\n")
```

```
Mode separated by species
                 sepal_length  sepal_width  petal_length  petal_width
species
Iris-setosa               5.0          3.4           1.5          0.2
Iris-versicolor           5.6          3.0           4.5          1.3
Iris-virginica            6.3          3.0           5.1          1.8
```

In [ ]:

```python
```