Practical 3

Title: Descriptive Analysis

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 groupedby the age groups. Create a list that contains a numeric value for each response to the categorical variable.

[ ]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

[ ]:
```
#load data
df = pd.read_csv("loan_data_set.csv")
```

[ ]:
```
df.head()
```

[ ]:
```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
```

```
3             1.0         Urban           Y
4             1.0         Urban           Y
```

Basic statistical

[ ]:
```
df.shape
```

[ ]:
```
(614, 13)
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

[ ]:
```
df.isna().sum()
```

[ ]:
```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

Filling null values with mean

```
[ ]: df["ApplicantIncome"].fillna(df["ApplicantIncome"].mean(), inplace=True)
```

```
[ ]: df["CoapplicantIncome"].fillna(df["CoapplicantIncome"].mean(), inplace=True)
```

```
[ ]: df["LoanAmount"].fillna(df["LoanAmount"].mean(), inplace=True)
```

```
[ ]: df["Credit_History"].fillna(np.random.randint(0,2), inplace=True)
```

Let us group the quantitative variables 'ApplicantIncome', 'Coapplicant Income', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History' by 'Loan_Status' categorical variable

```
[ ]: grouped_df = df[["ApplicantIncome", "CoapplicantIncome", "LoanAmount",␣
      ↪"Credit_History"]].groupby(df["Loan_Status"])
```

Measures of Central Tendency

```
[ ]: #mean
     mean = grouped_df.mean()
     mean
```

```
[ ]:              ApplicantIncome  CoapplicantIncome  LoanAmount  Credit_History
     Loan_Status
     N                5446.078125        1877.807292  150.945488        0.572917
     Y                5384.068720        1504.516398  144.349606        0.983412
```

```
[ ]: #median
     median = grouped_df.median()
     median
```

```
[ ]:              ApplicantIncome  CoapplicantIncome  LoanAmount  Credit_History
     Loan_Status
     N                      3833.5              268.0       133.5             1.0
     Y                      3812.5             1239.5       128.0             1.0
```

```
[ ]: #mode
     mode = df['LoanAmount'].mode()
     mode
```

```
[ ]: 0    146.412162
     Name: LoanAmount, dtype: float64
```

```
[ ]: #Minimum
     min = grouped_df.min()
     min
```

```
[ ]:              ApplicantIncome  CoapplicantIncome  LoanAmount  Credit_History
     Loan_Status
```

N	150	0.0	9.0	0.0
Y	210	0.0	17.0	0.0

[ ]:
```
#Maximum
max = grouped_df.max()
max
```

[ ]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	81000	41667.0	570.0	1.0
Y	63337	20000.0	700.0	1.0

Measures of Dispersion

[ ]:
```
#standard deviation
std = grouped_df.std()
std
```

[ ]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	6819.558528	4384.060103	83.361163	0.495948
Y	5765.441615	1924.754855	84.361109	0.127872

[ ]:
```
#variance
var = grouped_df.var()
var
```

[ ]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	4.650638e+07	1.921998e+07	6949.083562	0.245964
Y	3.324032e+07	3.704681e+06	7116.796734	0.016351

Interquartile Range

[ ]:
```
from scipy.stats import iqr
iqr(df['LoanAmount'])
```

[ ]: 64.5

Skewness

[ ]:
```
grouped_df.skew()
```

[ ]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	7.822895	6.487784	2.170045	-0.297145
Y	5.500757	3.041562	2.987803	-7.596877

Putting everything together

[ ]:
```
df.describe()
```

[ ]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	\
count	614.000000	614.000000	614.000000	600.00000	
mean	5403.459283	1621.245798	146.412162	342.00000	
std	6109.041673	2926.248369	84.037468	65.12041	
min	150.000000	0.000000	9.000000	12.00000	
25%	2877.500000	0.000000	100.250000	360.00000	
50%	3812.500000	1188.500000	129.000000	360.00000	
75%	5795.000000	2297.250000	164.750000	360.00000	
max	81000.000000	41667.000000	700.000000	480.00000	

	Credit_History
count	614.000000
mean	0.855049
std	0.352339
min	0.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	1.000000

[ ]:
```
df.describe(include='all')
```

[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	\
count	614	601	611	599	614	582	
unique	614	2	2	4	2	2	
top	LP001002	Male	Yes	0	Graduate	No	
freq	1	489	398	345	480	500	
mean	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	NaN	NaN	

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	\
count	614.000000	614.000000	614.000000	600.00000	
unique	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	
mean	5403.459283	1621.245798	146.412162	342.00000	
std	6109.041673	2926.248369	84.037468	65.12041	
min	150.000000	0.000000	9.000000	12.00000	
25%	2877.500000	0.000000	100.250000	360.00000	
50%	3812.500000	1188.500000	129.000000	360.00000	
75%	5795.000000	2297.250000	164.750000	360.00000	
max	81000.000000	41667.000000	700.000000	480.00000	

```
        Credit_History Property_Area Loan_Status
count       614.000000           614         614
unique             NaN             3           2
top                NaN     Semiurban           Y
freq               NaN           233         422
mean          0.855049           NaN         NaN
std           0.352339           NaN         NaN
min           0.000000           NaN         NaN
25%           1.000000           NaN         NaN
50%           1.000000           NaN         NaN
75%           1.000000           NaN         NaN
max           1.000000           NaN         NaN
```

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.

[ ]:
```
# import and load iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
iris.keys()
```

[ ]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

[ ]:
```
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df["label"] = iris.target
```

[ ]:
```
iris.target_names
```

[ ]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

0 -> setosa 1 -> versicolor 2 -> virginica

[ ]:
```
#dataframe shape
iris_df.shape
```

[ ]:
```
(150, 5)
```

[ ]:
```
iris_df.head()
```

[ ]:
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
```

```
   label
0      0
1      0
2      0
3      0
4      0
```

Basic statistical

[ ]:
```
#Datframe info
iris_df.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 (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   label              150 non-null    int32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

[ ]:
```
#dataframe desciption
iris_df.describe()
```

[ ]:
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)       label
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
```

Species statistical

Setosa stats

```
[ ]: setosa = iris_df[iris_df["label"] == 0].drop("label", axis=1)
```

```
[ ]: setosa.describe()
```

```
[ ]:        sepal length (cm)  sepal width (cm)  petal length (cm)  \
count           50.00000         50.000000          50.000000
mean             5.00600          3.428000           1.462000
std              0.35249          0.379064           0.173664
min              4.30000          2.300000           1.000000
25%              4.80000          3.200000           1.400000
50%              5.00000          3.400000           1.500000
75%              5.20000          3.675000           1.575000
max              5.80000          4.400000           1.900000

       petal width (cm)
count         50.000000
mean           0.246000
std            0.105386
min            0.100000
25%            0.200000
50%            0.200000
75%            0.300000
max            0.600000
```

Versicolor stats

```
[ ]: versicolor = iris_df[iris_df["label"] == 1].drop("label", axis=1)
```

```
[ ]: versicolor.describe()
```

```
[ ]:        sepal length (cm)  sepal width (cm)  petal length (cm)  \
count          50.000000         50.000000          50.000000
mean            5.936000          2.770000           4.260000
std             0.516171          0.313798           0.469911
min             4.900000          2.000000           3.000000
25%             5.600000          2.525000           4.000000
50%             5.900000          2.800000           4.350000
75%             6.300000          3.000000           4.600000
max             7.000000          3.400000           5.100000

       petal width (cm)
count         50.000000
mean           1.326000
std            0.197753
min            1.000000
```

```
25%            1.200000
50%            1.300000
75%            1.500000
max            1.800000
```

Virginica stats

[ ]:
```
virginica = iris_df[iris_df["label"] == 2].drop("label", axis=1)
```

[ ]:
```
virginica.describe()
```

[ ]:
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count           50.00000         50.000000          50.000000
mean             6.58800          2.974000           5.552000
std              0.63588          0.322497           0.551895
min              4.90000          2.200000           4.500000
25%              6.22500          2.800000           5.100000
50%              6.50000          3.000000           5.550000
75%              6.90000          3.175000           5.875000
max              7.90000          3.800000           6.900000

       petal width (cm)
count          50.00000
mean            2.02600
std             0.27465
min             1.40000
25%             1.80000
50%             2.00000
75%             2.30000
max             2.50000
```