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) variables. 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.

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

df = pd.read_csv('/content/drive/MyDrive/TE/Colab Notebooks/adult.csv')
df
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	na cou
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	U S
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	U S
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	U S
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	U S
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	U S
...	...	...	...	...	...	...	...	...	...	...	...	...	...	

```
df.describe()
```

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

```
df.isnull()
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relations
0	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	
48837	False	False	False	False	False	False	False	F
48838	False	False	False	False	False	False	False	F
48839	False	False	False	False	False	False	False	F
48840	False	False	False	False	False	False	False	F
48841	False	False	False	False	False	False	False	F

```
df.isnull().sum()
```

```
age                0
workclass          0
fnlwgt             0
education          0
educational-num    0
marital-status     0
occupation         0
relationship       0
race               0
gender             0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country     0
income             0
dtype: int64
```

1. Mean

```
#To find mean of all columns
df.mean()
```

```
<ipython-input-6-6e2b160ae8ee>:2: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version,
  df.mean()
age                    38.643585
fnlwgt             189664.134597
educational-num        10.078089
capital-gain         1079.067626
capital-loss           87.502314
hours-per-week         40.422382
dtype: float64
```

```
#To find mean of specific column
df.loc[:,'age'].mean()
```

```
38.64358543876172
```

```
#To find mean row wise
df.mean(axis=1)[0:3]
```

```
<ipython-input-10-2d24641b5b31>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is dep
  df.mean(axis=1)[0:3]
0    37812.333333
1    14985.166667
2    56171.833333
dtype: float64
```

```
df.groupby(['gender'])['age'].mean()
```

```
gender
Female    36.927989
Male      39.494395
Name: age, dtype: float64
```

```
df['income'].unique()
```

```
array(['<=50K', '>50K'], dtype=object)
```

2. Median

```
#To find median of all columns
df.median()
```

```
<ipython-input-11-9b6f8be3aa78>:2: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future versi
  df.median()
age                    37.0
fnlwgt             178144.5
educational-num        10.0
capital-gain            0.0
capital-loss            0.0
hours-per-week         40.0
dtype: float64
```

```
#To find median of specific column
df.loc[:,'age'].median()
```

```
37.0
```

```
#To find median row wise
df.median(axis=1)[0:4]
```

```
<ipython-input-13-6cea81e7732e>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is dep
  df.median(axis=1)[0:4]
0    16.0
1    23.5
2    20.0
3    42.0
dtype: float64
```

3. Mode

```
#To find mode of all columns
df.mode()
```

```
                                    educational-  marital-
   age  workclass  fnlwgt  education         num    status  occupation  relationship
```

```
#To find the mode of a specific column.
df.loc[:,'age'].mode()
```

```
0    36
Name: age, dtype: int64
```

4. Minimum

```
#To find minimum of all columns
df.min()
```

```
age                                17
workclass                           ?
fnlwgt                          12285
education                        10th
educational-num                     1
marital-status               Divorced
occupation                          ?
relationship                  Husband
```

```
race               Amer-Indian-Eskimo
gender                         Female
capital-gain                        0
capital-loss                        0
hours-per-week                      1
native-country                      ?
income                          <=50K
dtype: object
```

```
#To find minimum of Specific column
df.loc[:,'age'].min(skipna = False)
```

```
17
```

5. Maximum

```
#To find maximum of all columns
df.max()
```

```
age                             90
workclass              Without-pay
fnlwgt                     1490400
education             Some-college
educational-num                 16
marital-status             Widowed
occupation        Transport-moving
relationship                  Wife
race                         White
gender                        Male
capital-gain                 99999
capital-loss                  4356
hours-per-week                  99
native-country          Yugoslavia
income                        >50K
dtype: object
```

```
#To find maximum of Specific column
df.loc[:,'age'].max(skipna = False)
```

```
90
```

6. Standard Deviation

```
#To find Standard Deviation of all columns
df.std()
```

```
<ipython-input-22-1c9d8a2a52d8>:2: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future version,
  df.std()
age                   13.710510
fnlwgt            105604.025423
educational-num        2.570973
capital-gain        7452.019058
capital-loss         403.004552
hours-per-week        12.391444
dtype: float64
```

```
#To find Standard Deviation of specific column
df.loc[:,'age'].std()
```

```
13.710509934443557
```

```
#To find Standard Deviation row wise
df.std(axis=1)[0:4]
```

```
<ipython-input-24-87364a8110bc>:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is dep
  df.std(axis=1)[0:4]
0     92585.651335
1     36658.497789
2    137553.138655
3     64888.660753
dtype: float64
```

```
from sklearn import preprocessing
label_encoder = preprocessing. LabelEncoder()
df['income' ]= label_encoder.fit_transform(df['income'])
```

```
df['income'].std()
```

```
0.4266494219026857
```

kurtosis determined by the following standard deviation states that the distribution is platykurtic .(< 3)

Types of Variables:

Summary statistics of income grouped by the age groups:

*Problem Statement: *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.

```
#Categorical Variable: marital-status
#Quantitative Variable : Age

df.groupby(['marital-status'])['age'].mean()
```

```
marital-status
Divorced                 43.159204
Married-AF-spouse        31.945946
Married-civ-spouse       43.353724
Married-spouse-absent    40.613057
Never-married            28.128064
Separated                39.725490
Widowed                  59.377470
Name: age, dtype: float64
```

```
#Categorical Variable: marital-status
#Quantitative Variable : education

df.groupby(['marital-status'])['educational-num'].mean()
```

```
marital-status
Divorced                 10.052917
Married-AF-spouse        10.432432
Married-civ-spouse       10.303275
Married-spouse-absent     9.377389
Never-married             9.972141
Separated                 9.270588
Widowed                   9.088274
Name: educational-num, dtype: float64
```

```
df.groupby(['education']) ['income'].median()
```

```
education
10th            0.0
11th            0.0
12th            0.0
1st-4th         0.0
5th-6th         0.0
7th-8th         0.0
9th             0.0
Assoc-acdm      0.0
Assoc-voc       0.0
Bachelors       0.0
Doctorate       1.0
HS-grad         0.0
Masters         1.0
Preschool       0.0
Prof-school     1.0
Some-college    0.0
Name: income, dtype: float64
```

```
df.groupby(['marital-status'])['income' ].median()
```

```
marital-status
Divorced                 0.0
Married-AF-spouse        0.0
```

```
Married-civ-spouse       0.0
Married-spouse-absent    0.0
Never-married            0.0
Separated                0.0
Widowed                  0.0
Name: income, dtype: float64
```

Median can be used to separate outliers from a distribution. Mean can be used to get relative values.

```
#To create a list that contains a numeric value for each response to the categorical variable.

from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(df[['marital-status']]).toarray())
enc_df
```

	0	1	2	3	4	5	6
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...
48837	0.0	0.0	1.0	0.0	0.0	0.0	0.0
48838	0.0	0.0	1.0	0.0	0.0	0.0	0.0
48839	0.0	0.0	0.0	0.0	0.0	0.0	1.0
48840	0.0	0.0	0.0	0.0	1.0	0.0	0.0
48841	0.0	0.0	1.0	0.0	0.0	0.0	0.0

48842 rows × 7 columns

```
#To concat numerical list to dataframe
df_encode =df.join(enc_df)
df_encode
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationsh
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-ch
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husba
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husba
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husba
4	18	?	103497	Some-college	10	Never-married	?	Own-ch
...	...	...	...	...	...	...	...	

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.

Display basic statistical details on the iris dataset.

```
iris = pd.read_csv('/content/Iris (1).csv')
iris
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
#Assign Column names
col_names =['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']
iris = pd.read_csv('/content/drive/MyDrive/TE/Colab Notebooks/Iris (1).csv', names = col_names)
iris
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	Iris-virginica
147	6.3	2.5	5.0	1.9	Iris-virginica
148	6.5	3.0	5.2	2.0	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3.0	5.1	1.8	Iris-virginica

151 rows × 5 columns

```
iris.describe()
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
count	151	151	151	151	151
unique	36	24	44	23	4
top	5.0	3.0	1.5	0.2	Iris-setosa
freq	10	26	14	28	50

```
from google.colab import drive
drive.mount('/content/drive')
```

```
label_encoder = preprocessing.LabelEncoder()
iris['Species' ]= label_encoder.fit_transform(iris['Species'])
```

```
#Load all rows with Iris-setosa species in variable irisSet

irisSet = (iris['Species'] == 'Iris-setosa')
```

```
#To display basic statistical details like percentile, mean,standard deviation etc. for Iris-setosa use describe

print('Iris-setosa')
```

```
     Iris-setosa
```

```
print(iris[irisSet].describe())
```

```
            Sepal_Length Sepal_Width Petal_Length Petal_Width      Species
     count            50          50           50          50           50
     unique           15          16            9           6            1
     top             5.1         3.4          1.5         0.2  Iris-setosa
     freq              8           9           14          28           50
```

```
#Load all rows with Iris-versicolor species in variable irisVer
irisVer = (iris['Species'] == 'Iris-versicolor')
```

```
#To display basic statistical details like percentile, mean,standard deviation etc. for Iris-versicolor use describe

print('Iris-versicolor')
print(iris[irisVer].describe())
```

```
     Iris-versicolor
            Sepal_Length Sepal_Width Petal_Length Petal_Width          Species
     count            50          50           50          50               50
     unique           21          14           19           9                1
     top             5.5         3.0          4.5         1.3  Iris-versicolor
     freq              5           8            7          13               50
```

```
#Load all rows with Iris-virginica species in variable irisVir
irisVir = (iris['Species'] == 'Iris-virginica')
```

```
#To display basic statistical details like percentile, mean,standard deviation etc. for Iris-virginica use describe
print('Iris-virginica')
print(iris[irisVir].describe())
```

```
     Iris-virginica
            Sepal_Length Sepal_Width Petal_Length Petal_Width         Species
     count            50          50           50          50              50
     unique           21          13           20          12               1
     top             6.3         3.0          5.1         1.8  Iris-virginica
     freq              6          12            7          11              50
```

```
print('Iris-setosa')
print(iris[irisSet].describe())

print('Iris-versicolor')
print(iris[irisVer].describe())

print('Iris-virginica')
print(iris[irisVir].describe())
```

```
     Iris-setosa
            Sepal_Length Sepal_Width Petal_Length Petal_Width      Species
     count            50          50           50          50           50
     unique           15          16            9           6            1
     top             5.1         3.4          1.5         0.2  Iris-setosa
     freq              8           9           14          28           50
     Iris-versicolor
            Sepal_Length Sepal_Width Petal_Length Petal_Width          Species
     count            50          50           50          50               50
     unique           21          14           19           9                1
     top             5.5         3.0          4.5         1.3  Iris-versicolor
     freq              5           8            7          13               50
     Iris-virginica
            Sepal_Length Sepal_Width Petal_Length Petal_Width         Species
     count            50          50           50          50              50
     unique           21          13           20          12               1
     top             6.3         3.0          5.1         1.8  Iris-virginica
```