practical3-firsthalf-piyusha

April 3, 2025

0.1 23CO315 Piyusha Supe - DSBDA Practical 3

[4]:
```
import pandas as pd
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving income.csv to income.csv
```

[4]:
```
{'income.csv': b'id,age,income,occupation,years of experience\r\n1,64,35397,deve
loper,10\r\n2,54,32943,teacher,3\r\n3,43,21947,professor,9\r\n4,21,37092,enginee
r,5\r\n5,64,35935,accountant,10\r\n6,52,38933,mechanic,1\r\n7,37,22040,electrici
an,8\r\n8,37,36656,software engineer,5\r\n9,43,32220,tester,13\r\n10,26,36691,de
signer,3\r\n11,32,32283,developer,0\r\n12,40,27773,teacher,1\r\n13,43,30195,prof
essor,5\r\n14,36,23926,engineer,0\r\n15,59,39369,accountant,10\r\n16,53,28486,me
chanic,10\r\n17,65,37904,electrician,20\r\n18,20,32601,software engineer,19\r\n1
9,62,20494,tester,1\r\n20,58,35202,designer,9\r\n21,50,23974,developer,11\r\n22,
39,21239,teacher,20\r\n23,34,23712,professor,18\r\n24,62,27435,engineer,13\r\n25
,26,29457,accountant,19\r\n26,27,33222,mechanic,8\r\n27,33,29918,electrician,7\r
\n28,43,37929,software engineer,17\r\n29,46,21063,tester,12\r\n30,37,28859,desig
ner,6\r\n31,38,28751,developer,14\r\n32,47,25256,teacher,14\r\n33,51,30667,profe
ssor,2\r\n34,24,37021,engineer,7\r\n35,58,26798,accountant,9\r\n36,64,20465,mech
anic,14\r\n37,53,37197,electrician,11\r\n38,26,35320,software engineer,7\r\n39,4
7,21012,tester,15\r\n40,45,26067,designer,20\r\n41,36,38519,developer,15\r\n42,4
0,22165,teacher,1\r\n43,45,36675,professor,2\r\n44,30,26865,engineer,11\r\n45,44
,34992,accountant,12\r\n46,20,21214,mechanic,11\r\n47,54,26332,electrician,3\r\n
48,57,20762,software engineer,1\r\n49,46,25015,tester,4\r\n50,26,23520,designer,
15\r\n51,37,31716,developer,0\r\n52,57,31347,teacher,1\r\n53,40,25089,professor,
10\r\n54,20,39122,engineer,13\r\n55,34,39371,accountant,16\r\n56,25,24785,mechan
ic,2\r\n57,32,26542,electrician,9\r\n58,41,32278,software engineer,5\r\n59,52,34
374,tester,1\r\n60,44,27391,designer,19\r\n61,55,32322,developer,20\r\n62,60,368
66,teacher,10\r\n63,23,30692,professor,7\r\n64,43,31789,engineer,15\r\n65,30,391
48,accountant,17\r\n66,36,21191,mechanic,19\r\n67,30,31950,electrician,16\r\n68,
61,31861,software engineer,10\r\n69,46,22332,tester,16\r\n70,37,39543,designer,1
5\r\n71,42,25617,developer,2\r\n72,28,22100,teacher,17\r\n73,63,38487,professor,
1\r\n74,54,34357,engineer,14\r\n75,29,29091,accountant,7\r\n76,36,37788,mechanic
,18\r\n77,45,26830,electrician,3\r\n78,41,35312,software engineer,20\r\n79,30,38
864,tester,10\r\n80,32,32462,designer,8\r\n81,59,30955,developer,3\r\n82,33,3219
```

```
5,teacher,7\r\n83,35,33452,professor,11\r\n84,40,23181,engineer,17\r\n85,58,3220
4,accountant,17\r\n86,65,27389,mechanic,3\r\n87,37,34208,electrician,16\r\n88,35
,38912,software engineer,0\r\n89,57,30736,tester,20\r\n90,55,20472,designer,2\r\
n91,39,28007,developer,10\r\n92,29,30789,teacher,11\r\n93,23,28714,professor,8\r
\n94,29,39385,engineer,14\r\n95,24,30714,accountant,6\r\n96,25,20576,mechanic,10
\r\n97,26,21664,electrician,16\r\n98,55,20861,software engineer,8\r\n99,46,34936
,tester,18\r\n100,31,33452,designer,2\r\n101,54,27633,developer,2\r\n'}
```

[5]:
```
# Load the CSV file
file_path = "income.csv"  # Change this to your actual file path
df = pd.read_csv(file_path)

# Display first few rows of the dataset
print("Sample Dataset:\n", df.head())
```

```
Sample Dataset:
    id  age  income  occupation  years of experience
0   1   64   35397   developer                   10
1   2   54   32943     teacher                    3
2   3   43   21947   professor                    9
3   4   21   37092    engineer                    5
4   5   64   35935  accountant                   10
```

[6]:
```
df.tail()
```

[6]:

	id	age	income	occupation	years of experience
96	97	26	21664	electrician	16
97	98	55	20861	software engineer	8
98	99	46	34936	tester	18
99	100	31	33452	designer	2
100	101	54	27633	developer	2

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   101 non-null    int64
 1   age                  101 non-null    int64
 2   income               101 non-null    int64
 3   occupation           101 non-null    object
 4   years of experience  101 non-null    int64
dtypes: int64(4), object(1)
memory usage: 4.1+ KB
```

[8]:
```
df.describe(include="all")
```

[8]:

	id	age	income	occupation	years of experience
count	101.000000	101.000000	101.000000	101	101.000000
unique	NaN	NaN	NaN	10	NaN
top	NaN	NaN	NaN	developer	NaN
freq	NaN	NaN	NaN	11	NaN
mean	51.000000	41.732673	30203.564356	NaN	9.732673
std	29.300171	12.670352	5925.322711	NaN	6.177202
min	1.000000	20.000000	20465.000000	NaN	0.000000
25%	26.000000	32.000000	25256.000000	NaN	4.000000
50%	51.000000	40.000000	30736.000000	NaN	10.000000
75%	76.000000	53.000000	35202.000000	NaN	15.000000
max	101.000000	65.000000	39543.000000	NaN	20.000000

[10]:
```
print(df.shape)
print(df.size)
print(df.ndim)
print(df.columns)
```

```
(101, 5)
505
2
Index(['id', 'age', 'income', 'occupation', 'years of experience'],
dtype='object')
```

[11]:
```
print(df.isnull())
print(df.isna())
print(df.isna().sum())
```

```
        id    age  income  occupation  years of experience
0    False  False   False       False                False
1    False  False   False       False                False
2    False  False   False       False                False
3    False  False   False       False                False
4    False  False   False       False                False
..     …    …     …          …                    …
96   False  False   False       False                False
97   False  False   False       False                False
98   False  False   False       False                False
99   False  False   False       False                False
100  False  False   False       False                False

[101 rows x 5 columns]
        id    age  income  occupation  years of experience
0    False  False   False       False                False
1    False  False   False       False                False
2    False  False   False       False                False
```

```
3    False  False   False       False                False
4    False  False   False       False                False
..     …      …      …          …                    …
96   False  False   False       False                False
97   False  False   False       False                False
98   False  False   False       False                False
99   False  False   False       False                False
100  False  False   False       False                False

[101 rows x 5 columns]
id                     0
age                    0
income                 0
occupation             0
years of experience    0
dtype: int64
```

[13]:
```python
categorical_col = "age"  # Change this to your categorical column
numerical_col = "income"  # Change this to your numerical column

# Check for missing values and fill them if necessary
df[numerical_col] = df[numerical_col].fillna(df[numerical_col].median())

# Compute summary statistics grouped by the categorical column
summary_stats = df.groupby(categorical_col)[numerical_col].agg(
    mean="mean",
    median="median",
    min="min",
    max="max",
    std="std"
)

# Create a list of grouped values
grouped_values = df.groupby(categorical_col)[numerical_col].apply(list)

# Display results
print("\nSummary Statistics:")
print(summary_stats)

print("\nGrouped Values as List:")
print(grouped_values)

# Save the results to CSV files
summary_stats.to_csv("summary_statistics.csv")
grouped_values.to_csv("grouped_values.csv")

print("\nSummary statistics saved to 'summary_statistics.csv'")
```

```
print("Grouped values saved to 'grouped_values.csv'")
```

```
Summary Statistics:
              mean   median    min    max           std
age
20   30979.000000  32601.0  21214  39122   9063.513612
21   37092.000000  37092.0  37092  37092           NaN
23   29703.000000  29703.0  28714  30692   1398.657213
24   33867.500000  33867.5  30714  37021   4459.722469
25   22680.500000  22680.5  20576  24785   2976.212442
26   29330.400000  29457.0  21664  36691   6756.556172
27   33222.000000  33222.0  33222  33222           NaN
28   22100.000000  22100.0  22100  22100           NaN
29   33088.333333  30789.0  29091  39385   5518.768824
30   34206.750000  35407.0  26865  39148   5918.900313
31   33452.000000  33452.0  33452  33452           NaN
32   30429.000000  32283.0  26542  32462   3367.430326
33   31056.500000  31056.5  29918  32195   1610.082141
34   31541.500000  31541.5  23712  39371  11072.585087
35   36182.000000  36182.0  33452  38912   3860.803025
36   30356.000000  30857.0  21191  38519   9077.652376
37   32170.333333  32962.0  22040  39543   6203.619997
38   28751.000000  28751.0  28751  28751           NaN
39   24623.000000  24623.0  21239  28007   4785.698695
40   24552.000000  24135.0  22165  27773   2465.807508
41   33795.000000  33795.0  32278  35312   2145.361974
42   25617.000000  25617.0  25617  25617           NaN
43   30816.000000  31789.0  21947  37929   5756.681249
44   31191.500000  31191.5  27391  34992   5374.718644
45   29857.333333  26830.0  26067  36675   5916.584854
46   25836.500000  23673.5  21063  34936   6286.058251
47   23134.000000  23134.0  21012  25256   3000.961179
50   23974.000000  23974.0  23974  23974           NaN
51   30667.000000  30667.0  30667  30667           NaN
52   36653.500000  36653.5  34374  38933   3223.699815
53   32841.500000  32841.5  28486  37197   6159.607171
54   30316.250000  30288.0  26332  34357   3928.594098
55   24551.666667  20861.0  20472  32322   6732.116334
57   27615.000000  30736.0  20762  31347   5942.729760
58   31401.333333  32204.0  26798  35202   4259.108984
59   35162.000000  35162.0  30955  39369   5949.596457
60   36866.000000  36866.0  36866  36866           NaN
61   31861.000000  31861.0  31861  31861           NaN
62   23964.500000  23964.5  20494  27435   4908.028168
63   38487.000000  38487.0  38487  38487           NaN
64   30599.000000  35397.0  20465  35935   8780.422997
```

```
65   32646.500000  32646.5  27389  37904   7435.227804

Grouped Values as List:
age
20                          [32601, 21214, 39122]
21                                       [37092]
23                                [30692, 28714]
24                                [37021, 30714]
25                                [24785, 20576]
26           [36691, 29457, 35320, 23520, 21664]
27                                       [33222]
28                                       [22100]
29                          [29091, 30789, 39385]
30                  [26865, 39148, 31950, 38864]
31                                       [33452]
32                          [32283, 26542, 32462]
33                                [29918, 32195]
34                                [23712, 39371]
35                                [33452, 38912]
36                  [23926, 38519, 21191, 37788]
37    [22040, 36656, 28859, 31716, 39543, 34208]
38                                       [28751]
39                                [21239, 28007]
40                  [27773, 22165, 25089, 23181]
41                                [32278, 35312]
42                                       [25617]
43           [21947, 32220, 30195, 37929, 31789]
44                                [34992, 27391]
45                          [26067, 36675, 26830]
46                  [21063, 25015, 22332, 34936]
47                                [25256, 21012]
50                                       [23974]
51                                       [30667]
52                                [38933, 34374]
53                                [28486, 37197]
54                  [32943, 26332, 34357, 27633]
55                          [32322, 20472, 20861]
57                          [20762, 31347, 30736]
58                          [35202, 26798, 32204]
59                                [39369, 30955]
60                                       [36866]
61                                       [31861]
62                                [20494, 27435]
63                                       [38487]
64                          [35397, 35935, 20465]
65                                [37904, 27389]
Name: income, dtype: object
```

```
Summary statistics saved to 'summary_statistics.csv'
Grouped values saved to 'grouped_values.csv'
```

[ ]:

piyusha-pract-3

March 5, 2025

Piyusha Supe 23CO315

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

[2]:
```
import pandas as pd
import statistics as st
import numpy as np
```

[3]:
```
df = pd.read_csv("IRIS.csv")
```

DATA PREPROCESSING

Exploratory data analysis

[4]:
```
df.head()
```

[4]:

	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
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

[5]:
```
df.tail()
```

[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

[6]:
```
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  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
```

[7]:
```
df.describe()
```

[7]:

	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
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

[8]:
```
df.describe(include="all")
```

[8]:

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

[9]:
```
df.shape
```

[9]: (150, 5)

[10]:
```
df.size
```

[10]:
```
750
```

[11]:
```
df.columns
```

[11]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

[13]:
```
df.dtypes
```

[13]:
```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

[14]:
```
df.ndim
```

[14]:
```
2
```

[15]:
```
summary = df.groupby('species')['sepal_length'].describe()
```

[16]:
```
print(summary)
```

```
                 count   mean       std  min    25%  50%  75%  max
species
Iris-setosa       50.0  5.006  0.352490  4.3  4.800  5.0  5.2  5.8
Iris-versicolor   50.0  5.936  0.516171  4.9  5.600  5.9  6.3  7.0
Iris-virginica    50.0  6.588  0.635880  4.9  6.225  6.5  6.9  7.9
```

Custom summary of data stats: Median, mode

[18]:
```
def custom_describe(x):
    mode_value = st.mode(x)
    median_value = np.median(x)
    return pd.Series([x.count(),x.mean(),x.std(),mode_value,median_value,x.
 ↪min(),x.max()])
    index=['count','mean','std','mode','median','min','max']
```

[20]:
```
summary_with_mode_median = df.groupby('species')['sepal_length'].
 ↪apply(custom_describe)
print(summary_with_mode_median)
```

```
species
Iris-setosa      0    50.000000
                 1     5.006000
                 2     0.352490
```

```
                  3     5.100000
                  4     5.000000
                  5     4.300000
                  6     5.800000
Iris-versicolor   0    50.000000
                  1     5.936000
                  2     0.516171
                  3     5.500000
                  4     5.900000
                  5     4.900000
                  6     7.000000
Iris-virginica    0    50.000000
                  1     6.588000
                  2     0.635880
                  3     6.300000
                  4     6.500000
                  5     4.900000
                  6     7.900000
Name: sepal_length, dtype: float64
```

MODE SUMMARY

[21]:
```
mode_summary = df.groupby('species')['sepal_length'].apply(lambda x: st.mode(x))
print("Mode:")
print(mode_summary)
```

```
species
Iris-setosa        5.1
Iris-versicolor    5.5
Iris-virginica     6.3
Name: sepal_length, dtype: float64
```

MEDIAN SUMMARY

[22]:
```
median_summary = df.groupby('species')['sepal_length'].median()
print("Median: ")
print(median_summary)
```

```
Median:
species
Iris-setosa        5.0
Iris-versicolor    5.9
Iris-virginica     6.5
Name: sepal_length, dtype: float64
```

MEAN SUMMARY

[23]:
```
mean_summary = df.groupby('species')['sepal_length'].mean()
print("Mean: ")
print(mean_summary)
```

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

GROUPING ACCORDING TO THE SPECIES

[24]:
```
species_numeric_value = df.groupby('species').size().tolist()
print(species_numeric_value)
```

```
[50, 50, 50]
```

[ ]:
```
species_stats = df.groupby('species').describe()
print(species_stats)
```

PRINTING THE SUMMARY FOR EACH OF THE SPECIES

[27]:
```
setosa_stats = df[df['species']=='Iris-setosa'].describe()
versicolor_stats = df[df['species']=='Iris-versicolor'].describe()
virginica_stats = df[df['species']=='Iris-virginica'].describe()
#print the statistics

print("Setosa statistics:")
print(setosa_stats)
print("\nVersicolor statistics: ")
print(versicolor_stats)
print("\nVirginica statistics:")
print(virginica_stats)
```

```
Setosa statistics:
       sepal_length  sepal_width  petal_length  petal_width
count      50.00000    50.000000     50.000000     50.00000
mean        5.00600     3.418000      1.464000      0.24400
std         0.35249     0.381024      0.173511      0.10721
min         4.30000     2.300000      1.000000      0.10000
25%         4.80000     3.125000      1.400000      0.20000
50%         5.00000     3.400000      1.500000      0.20000
75%         5.20000     3.675000      1.575000      0.30000
max         5.80000     4.400000      1.900000      0.60000

Versicolor statistics:
       sepal_length  sepal_width  petal_length  petal_width
count     50.000000    50.000000     50.000000    50.000000
mean       5.936000     2.770000      4.260000     1.326000
std        0.516171     0.313798      0.469911     0.197753
min        4.900000     2.000000      3.000000     1.000000
25%        5.600000     2.525000      4.000000     1.200000
```

```
50%        5.900000     2.800000      4.350000     1.300000
75%        6.300000     3.000000      4.600000     1.500000
max        7.000000     3.400000      5.100000     1.800000

Virginica statistics:
       sepal_length  sepal_width  petal_length  petal_width
count      50.00000    50.000000     50.000000     50.00000
mean        6.58800     2.974000      5.552000      2.02600
std         0.63588     0.322497      0.551895      0.27465
min         4.90000     2.200000      4.500000      1.40000
25%         6.22500     2.800000      5.100000      1.80000
50%         6.50000     3.000000      5.550000      2.00000
75%         6.90000     3.175000      5.875000      2.30000
max         7.90000     3.800000      6.900000      2.50000
```

CONCLUSION: Thus we have successfully done the descriptive data analysis on the given dataset and summarized it according to species