In [1]:

```
import pandas as pd
import numpy as np
```

In [5]:

```
df = pd.read_csv("Desktop/Iris.csv")
df.head()
```

Out[5]:

	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

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [9]:

```
grouped_stats = df.groupby("Species")["SepalLengthCm"].agg(
    Mean="mean",
    Median="median",
    Minimum="min",
    Maximum="max",
    Standard_Deviation="std"
)
grouped_stats
```

Out[9]:

	Mean	Median	Minimum	Maximum	Standard_Deviation
Species					
Iris-setosa	5.006	5.0	4.3	5.8	0.352490
Iris-versicolor	5.936	5.9	4.9	7.0	0.516171
Iris-virginica	6.588	6.5	4.9	7.9	0.635880

In [11]:

```
species_lists = {
    species: df[df["Species"] == species]["SepalLengthCm"].tolist()
    for species in df["Species"].unique()
}

species_lists
```

```
Out[11]: {'Iris-setosa': [5.1,
           4.9,
           4.7,
           4.6,
           5.0,
           5.4,
           4.6,
           5.0,
           4.4,
           4.9,
           5.4,
           4.8,
           4.8,
           4.3,
           5.8,
           5.7,
           5.4,
           5.1,
           5.7,
           5.1,
           5.4,
           5.1,
           4.6,
           5.1,
           4.8,
           5.0,
           5.0,
           5.2,
           5.2,
           4.7,
           4.8,
           5.4,
           5.2,
           5.5,
           4.9,
           5.0,
           5.5,
           4.9,
           4.4,
           5.1,
           5.0,
           4.5,
           4.4,
           5.0,
           5.1,
           4.8,
           5.1,
           4.6,
           5.3,
           5.0],
          'Iris-versicolor': [7.0,
           6.4,
           6.9,
           5.5,
           6.5,
           5.7,
           6.3,
           4.9,
           6.6,
           5.2,
 5.0,
 5.9,
 6.0,
 6.1,
 5.6,
 6.7,
 5.6,
 5.8,
 6.2,
 5.6,
 5.9,
 6.1,
 6.3,
 6.1,
 6.4,
 6.6,
 6.8,
 6.7,
 6.0,
 5.7,
 5.5,
 5.5,
 5.8,
 6.0,
 5.4,
 6.0,
 6.7,
 6.3,
 5.6,
 5.5,
 5.5,
 6.1,
 5.8,
 5.0,
 5.6,
 5.7,
 5.7,
 6.2,
 5.1,
 5.7],
'Iris-virginica': [6.3,
 5.8,
 7.1,
 6.3,
 6.5,
 7.6,
 4.9,
 7.3,
 6.7,
 7.2,
 6.5,
 6.4,
 6.8,
 5.7,
 5.8,
 6.4,
 6.5,
 7.7,
 7.7,
 6.0,
  6.9,
  5.6,
  7.7,
  6.3,
  6.7,
  7.2,
  6.2,
  6.1,
  6.4,
  7.2,
  7.4,
  7.9,
  6.4,
  6.3,
  6.1,
  7.7,
  6.3,
  6.4,
  6.0,
  6.9,
  6.7,
  6.9,
  5.8,
  6.8,
  6.7,
  6.7,
  6.3,
  6.5,
  6.2,
  5.9]}
```

In [13]:

```python
species_names = df["Species"].unique()

for species in species_names:
    print(f"\nStatistics for {species}")
    print("-" * 30)

    sp_data = df[df["Species"] == species].iloc[:, 1:-1]  # numerical col

    print("\nMean:")
    print(sp_data.mean())

    print("\nStandard Deviation:")
    print(sp_data.std())

    print("\nPercentiles (25%, 50%, 75%):")
    print(sp_data.quantile([0.25, 0.5, 0.75]))
```

```
Statistics for Iris-setosa
------------------------------

Mean:
SepalLengthCm    5.006
SepalWidthCm     3.418
PetalLengthCm    1.464
PetalWidthCm     0.244
dtype: float64

Standard Deviation:
SepalLengthCm    0.352490
SepalWidthCm     0.381024
PetalLengthCm    0.173511
PetalWidthCm     0.107210
dtype: float64

Percentiles (25%, 50%, 75%):
      SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0.25            4.8         3.125          1.400           0.2
0.50            5.0         3.400          1.500           0.2
0.75            5.2         3.675          1.575           0.3

Statistics for Iris-versicolor
------------------------------

Mean:
SepalLengthCm    5.936
SepalWidthCm     2.770
PetalLengthCm    4.260
PetalWidthCm     1.326
dtype: float64

Standard Deviation:
SepalLengthCm    0.516171
SepalWidthCm     0.313798
PetalLengthCm    0.469911
PetalWidthCm     0.197753
dtype: float64

Percentiles (25%, 50%, 75%):
      SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0.25            5.6         2.525           4.00           1.2
0.50            5.9         2.800           4.35           1.3
0.75            6.3         3.000           4.60           1.5

Statistics for Iris-virginica
------------------------------

Mean:
SepalLengthCm    6.588
SepalWidthCm     2.974
PetalLengthCm    5.552
PetalWidthCm     2.026
dtype: float64

Standard Deviation:
SepalLengthCm    0.635880
SepalWidthCm     0.322497
PetalLengthCm    0.551895
```

```
PetalWidthCm     0.274650
dtype: float64

Percentiles (25%, 50%, 75%):
      SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0.25          6.225         2.800          5.100           1.8
0.50          6.500         3.000          5.550           2.0
0.75          6.900         3.175          5.875           2.3
```

In [15]:

```
df.describe()
```

Out[15]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [17]:

```
sp_data.mean()
sp_data.std()
sp_data.quantile([0.25, 0.5, 0.75])
```

Out[17]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0.25	6.225	2.800	5.100	1.8
0.50	6.500	3.000	5.550	2.0
0.75	6.900	3.175	5.875	2.3

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
```

In [3]:

```
####part 1
```

In [5]:

```
import pandas as pd
```

In [11]:

```
df1 = pd.read_csv('Desktop/adult.csv')
```

In [13]:

```
df1.head()
```

Out[13]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	rela
0	90	?	77053	HS-grad	9	Widowed	?	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	
2	66	?	186061	Some-college	10	Widowed	?	U
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	U
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	C

In [87]:

```
df1.columns = df1.columns.str.strip()
```

In [107…

```
numeric_vars = ['age', 'hours.per.week']
cat_var = 'workclass'
```

In [57]:

```
print(df1.columns.tolist())
df1.columns = df1.columns.str.strip()
```

```
['age', 'workclass', 'fnlwgt', 'education', 'education.num', 'marital.stat
us', 'occupation', 'relationship', 'race', 'sex', 'capital.gain', 'capita
l.loss', 'hours.per.week', 'native.country', 'income']
```

In [71]:

```
df1_complete = df1.dropna(subset=numeric_vars + [cat_var])
df1_clean = df1_complete[df1_complete[cat_var] != '?']
```

In [73]:

```
grouped = df1_clean.groupby(cat_var)
```

In [83]:

```
summary = grouped[numeric_vars].agg(['mean', 'median', 'min', 'max', 'std
print("##### CLEAN GROUPED SUMMARY STATISTICS #####")
print(summary)
```

```
##### CLEAN GROUPED SUMMARY STATISTICS #####
                        age                           hours.per.week      
\
                       mean median min max        std           mean media
n
workclass
Federal-gov       42.590625   43.0  17  90  11.509171      41.379167   40.
0
Local-gov         41.751075   41.0  17  90  12.272856      40.982800   40.
0
Never-worked      20.571429   18.0  17  30   4.613644      28.428571   35.
0
Private           36.797585   35.0  17  90  12.827721      40.267096   40.
0
Self-emp-inc      46.017025   45.0  17  84  12.553194      48.818100   50.
0
Self-emp-not-inc  44.969697   44.0  17  90  13.338162      44.421881   40.
0
State-gov         39.436055   39.0  17  81  12.431065      39.031587   40.
0
Without-pay       47.785714   57.0  19  72  21.075610      32.714286   27.
5


                 min max        std
workclass
Federal-gov        4  99   8.838605
Local-gov          2  99  10.771559
Never-worked       4  40  15.186147
Private            1  99  11.256298
Self-emp-inc       1  99  13.900417
Self-emp-not-inc   1  99  16.674958
State-gov          1  99  11.697014
Without-pay       10  65  17.357900
```

In [77]:

```python
cat_lists = {grp: list(data[numeric_vars].values) for grp, data in groupe
```

In [79]:

```python
print("\n====== NUMERIC LISTS PER CATEGORY ======")
for k, v in cat_lists.items():
    print(f"{k}: {len(v)} rows")
```

```
====== NUMERIC LISTS PER CATEGORY ======
Federal-gov: 960 rows
Local-gov: 2093 rows
Never-worked: 7 rows
Private: 22696 rows
Self-emp-inc: 1116 rows
Self-emp-not-inc: 2541 rows
State-gov: 1298 rows
Without-pay: 14 rows
```

In [89]:

```python
#new
```

In [91]:

```python
numeric_cols = ['age', 'hours.per.week']
categorical_cols = ['workclass', 'education', 'marital.status', 'occupati
```

In [95]:

```python
# Remove rows where numeric columns are missing
df2_clean = df1.dropna(subset=numeric_cols + categorical_cols)

# Remove rows with '?' in any categorical column
```

```
for col in categorical_cols:
    df2_clean = df2_clean[df2_clean[col] != '?']
```

In [97]:

```
print("====== CATEGORICAL ANALYSIS ======")
for col in categorical_cols:
    counts = df2_clean[col].value_counts()
    percents = df2_clean[col].value_counts(normalize=True) * 100
    print(f"\nColumn: {col}")
    print("Counts:\n", counts)
    print("Percentages:\n", percents.round(2))
```

```
====== CATEGORICAL ANALYSIS ======

Column: workclass
Counts:
 workclass
Private             22696
Self-emp-not-inc     2541
Local-gov            2093
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Name: count, dtype: int64
Percentages:
 workclass
Private             73.89
Self-emp-not-inc     8.27
Local-gov            6.81
State-gov            4.23
Self-emp-inc         3.63
Federal-gov          3.13
Without-pay          0.05
Name: proportion, dtype: float64

Column: education
Counts:
 education
HS-grad         9968
Some-college    6775
Bachelors       5182
Masters         1675
Assoc-voc       1321
11th            1056
Assoc-acdm      1020
10th             831
7th-8th          573
Prof-school      558
9th              463
Doctorate        398
12th             393
5th-6th          303
1st-4th          156
Preschool         46
Name: count, dtype: int64
Percentages:
 education
HS-grad         32.45
Some-college    22.06
Bachelors       16.87
Masters          5.45
Assoc-voc        4.30
11th             3.44
Assoc-acdm       3.32
10th             2.71
7th-8th          1.87
Prof-school      1.82
9th              1.51
Doctorate        1.30
12th             1.28
5th-6th          0.99
```

```
1st-4th          0.51
Preschool        0.15
Name: proportion, dtype: float64

Column: marital.status
Counts:
 marital.status
Married-civ-spouse       14339
Never-married             9912
Divorced                  4258
Separated                  959
Widowed                    840
Married-spouse-absent      389
Married-AF-spouse           21
Name: count, dtype: int64
Percentages:
 marital.status
Married-civ-spouse       46.68
Never-married            32.27
Divorced                 13.86
Separated                 3.12
Widowed                   2.73
Married-spouse-absent     1.27
Married-AF-spouse         0.07
Name: proportion, dtype: float64

Column: occupation
Counts:
 occupation
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspct    2002
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: count, dtype: int64
Percentages:
 occupation
Prof-specialty       13.48
Craft-repair         13.34
Exec-managerial      13.24
Adm-clerical         12.27
Sales                11.88
Other-service        10.73
Machine-op-inspct     6.52
Transport-moving      5.20
Handlers-cleaners     4.46
Farming-fishing       3.24
Tech-support          3.02
Protective-serv       2.11
Priv-house-serv       0.49
Armed-Forces          0.03
```

```
Name: proportion, dtype: float64

Column: sex
Counts:
 sex
Male      20788
Female     9930
Name: count, dtype: int64
Percentages:
 sex
Male      67.67
Female    32.33
Name: proportion, dtype: float64

Column: race
Counts:
 race
White                 26301
Black                  2909
Asian-Pac-Islander      974
Amer-Indian-Eskimo      286
Other                   248
Name: count, dtype: int64
Percentages:
 race
White                 85.62
Black                  9.47
Asian-Pac-Islander     3.17
Amer-Indian-Eskimo     0.93
Other                  0.81
Name: proportion, dtype: float64
```

In [101…

```
cat_var = 'workclass'
grouped = df2_clean.groupby(cat_var)
```

In [103…

```
summary = grouped[numeric_cols].agg(['mean', 'median', 'min', 'max', 'std
print("\n====== NUMERIC SUMMARY STATISTICS GROUPED BY WORKCLASS ======")
print(summary)
```

```
====== NUMERIC SUMMARY STATISTICS GROUPED BY WORKCLASS ======
                        age                              hours.per.week         
\
                       mean median min max        std           mean media
n
workclass                                                                   
Federal-gov       42.590625   43.0  17  90  11.509171      41.379167   40.
0
Local-gov         41.751075   41.0  17  90  12.272856      40.982800   40.
0
Private           36.797585   35.0  17  90  12.827721      40.267096   40.
0
Self-emp-inc      46.017025   45.0  17  84  12.553194      48.818100   50.
0
Self-emp-not-inc  44.969697   44.0  17  90  13.338162      44.421881   40.
0
State-gov         39.436055   39.0  17  81  12.431065      39.031587   40.
0
Without-pay       47.785714   57.0  19  72  21.075610      32.714286   27.
5


                 min max        std  
workclass                            
Federal-gov        4  99   8.838605  
Local-gov          2  99  10.771559  
Private            1  99  11.256298  
Self-emp-inc       1  99  13.900417  
Self-emp-not-inc   1  99  16.674958  
State-gov          1  99  11.697014  
Without-pay       10  65  17.357900  
```

In [105…

```python
cat_lists = {grp: list(data[numeric_cols].values) for grp, data in groupe

print("\n====== NUMBER OF ROWS PER CATEGORY ======")
for k, v in cat_lists.items():
    print(f"{k}: {len(v)} rows")
```

```
====== NUMBER OF ROWS PER CATEGORY ======
Federal-gov: 960 rows
Local-gov: 2093 rows
Private: 22696 rows
Self-emp-inc: 1116 rows
Self-emp-not-inc: 2541 rows
State-gov: 1298 rows
Without-pay: 14 rows
```

In [ ]: