Double-click (or enter) to edit

Download the sampledata_orders.csv dataset. As the name says it is a .csv file. The file contains information regarding sales order details of United Shipping Agency. Open it in a text editor or excel and inspect it first. Write a python program to load the data stored in a csv data file sampledata_orders.csv in a pandas DataFrame and perform the following operations.

i. Print the top 5 row and top 10 row

ii. Print the all column names

iii. Print the all row index ranges

iv. Print the Order Quantity column (top 5 values)

v. Print The Sales column (top 10 values)

vi. Print the row with ID:50

vii. Print the third row

viii. Print the Order Quantity, Sales, Discount and Profit of the 2nd,4th, 6th and 8th row:

ix. Print the Order Quantity, Sales, Discount and Profit of orders with discount > 10%

x. Print the production cost.(production cost= Sales-Profit)

```
import pandas as pd
orders = pd.read_csv("/content/sampledata_orders.csv")
#print("First 5 entries:")
#print(orders.head())
#print(orders.head(10))
#print(orders.columns)  #Print the all column names
#Print the all row index ranges
#print(orders.index)
#print(orders["Order Quantity"].head()) #Print the Order Quantity column (top 5 values)
#print(orders["Sales"].head(10))#Print The Sales column (top 10 values)
#print("The row with ID:50")
#r50 = orders.loc[50]   #Print the row with ID:50
#print(r50)
#r3=orders.iloc[3]
#print(r3)  #Print the third row
#print(orders[2:9:2][["Order Quantity", "Sales","Discount", "Profit"]])
#print(orders[orders["Discount"] > 0.1][["Order Quantity","Sales","Discount", "Profit"]])
pcost=orders["Sales"]-orders["Profit"]
print(pcost)
```

```
0        474.7900
1       9665.2100
2        197.8600
3       3766.7895
4        363.3300
```

```
           ...
8394    1617.2200
8395     370.3200
8396     480.7300
8397     629.0775
8398    1445.5800
Length: 8399, dtype: float64
```

Download the diamonds.csv dataset. As the name says it is a .csv file. The file contains information regarding diamonds details of Laisha diamond shop. Open it in a text editor or excel and inspect it first. Write a python program to load the data stored in a csv file diamonds.csv in a pandas DataFrame and perform the following operations.

I. find carat weight at least 0.3 and write this detail into cart.csv file

II. find total price of cart weight at maximum 0.3

III. Sort the ‘carat’ column by ascending and descending order.

IV. Print the premium diamond details

V. Find the most expensive diamond whose color is D

```
import pandas as pd
df = pd.read_csv('/content/diamonds.csv')
sample=df[df["carat"]<.3]
sample.to_csv('cart.csv')
tot=df[df["carat"]>.3]
tot["price"].sum()
result = df.sort_values('carat')
print(result)
result = df.sort_values('carat', ascending=False)
print(result)
pre=df.groupby("cut")
pre.get_group("Premium")
```

```
       carat      cut color clarity  depth  table  price      x      y     z
31593   0.20  Premium     E     VS2   61.1   59.0    367   3.81   3.78  2.32
31597   0.20    Ideal     D     VS2   61.5   57.0    367   3.81   3.77  2.33
31596   0.20  Premium     F     VS2   62.6   59.0    367   3.73   3.71  2.33
31595   0.20    Ideal     E     VS2   59.7   55.0    367   3.86   3.84  2.30
31594   0.20  Premium     E     VS2   59.7   62.0    367   3.84   3.80  2.28
...      ...      ...   ...     ...    ...    ...    ...    ...    ...   ...
25999   4.01  Premium     J      I1   62.5   62.0  15223  10.02   9.94  6.24
25998   4.01  Premium     I      I1   61.0   61.0  15223  10.14  10.10  6.17
27130   4.13     Fair     H      I1   64.8   61.0  17329  10.00   9.85  6.43
27630   4.50     Fair     J      I1   65.8   58.0  18531  10.23  10.16  6.72
27415   5.01     Fair     J      I1   65.5   59.0  18018  10.74  10.54  6.98

[53940 rows x 10 columns]
       carat      cut color clarity  depth  table  price      x      y     z
27415   5.01     Fair     J      I1   65.5   59.0  18018  10.74  10.54  6.98
27630   4.50     Fair     J      I1   65.8   58.0  18531  10.23  10.16  6.72
27130   4.13     Fair     H      I1   64.8   61.0  17329  10.00   9.85  6.43
25999   4.01  Premium     J      I1   62.5   62.0  15223  10.02   9.94  6.24
25998   4.01  Premium     I      I1   61.0   61.0  15223  10.14  10.10  6.17
...      ...      ...   ...     ...    ...    ...    ...    ...    ...   ...
31592   0.20  Premium     E     VS2   59.0   60.0    367   3.81   3.78  2.24
31591   0.20  Premium     E     VS2   59.8   62.0    367   3.79   3.77  2.26
31601   0.20  Premium     D     VS2   61.7   60.0    367   3.77   3.72  2.31
14      0.20  Premium     E     SI2   60.2   62.0    345   3.79   3.75  2.27
31596   0.20  Premium     F     VS2   62.6   59.0    367   3.73   3.71  2.33

[53940 rows x 10 columns]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
12	0.22	Premium	F	SI1	60.4	61.0	342	3.88	3.84	2.33

1. WAP program to read a sampledata_orders.csv file

and perform descriptive statics and print the result.

```
import pandas as pd
orders = pd.read_csv("/content/sampledata_orders.csv")
print("A description of the numerical values:")
print(orders.describe())
orders.min()
orders.max()
orders.std()
orders.var()
orders.mean()
orders.median()
orders.mode()
orders.sum()
orders.count()
orders.rank()
orders.skew()
orders.kurt()
```

```
A description of the numerical values:
            Row ID      Order ID  Order Quantity         Sales     Discount  \
count  8399.000000   8399.000000     8399.000000   8399.000000  8399.000000
mean   4200.000000  29965.179783       25.571735   1775.878179     0.049671
std    2424.726789  17260.883447       14.481071   3585.050525     0.031823
min       1.000000      3.000000        1.000000      2.240000     0.000000
25%    2100.500000  15011.500000       13.000000    143.195000     0.020000
50%    4200.000000  29857.000000       26.000000    449.420000     0.050000
75%    6299.500000  44596.000000       38.000000   1709.320000     0.080000
max    8399.000000  59973.000000       50.000000  89061.050000     0.250000

             Profit   Unit Price  Shipping Cost  Product Base Margin
count   8399.000000  8399.000000    8399.000000          8336.000000
mean     181.184423    89.346259      12.838557             0.512513
std     1196.653372   290.354383      17.264052             0.135589
min   -14140.700000     0.990000       0.490000             0.350000
25%      -83.315000     6.480000       3.300000             0.380000
50%       -1.500000    20.990000       6.070000             0.520000
75%      162.750000    85.990000      13.990000             0.590000
max    27220.690000  6783.020000     164.730000             0.850000
<ipython-input-7-ec6ac894d235>:7: FutureWarning: Dropping of nuisance columns i
  orders.std()
<ipython-input-7-ec6ac894d235>:8: FutureWarning: Dropping of nuisance columns i
  orders.var()
<ipython-input-7-ec6ac894d235>:9: FutureWarning: Dropping of nuisance columns i
  orders.mean()
<ipython-input-7-ec6ac894d235>:10: FutureWarning: Dropping of nuisance columns
  orders.median()
<ipython-input-7-ec6ac894d235>:15: FutureWarning: Dropping of nuisance columns
  orders.skew()
<ipython-input-7-ec6ac894d235>:16: FutureWarning: Dropping of nuisance columns
  orders.kurt()
Row ID                  -1.200000
Order ID                -1.178317
Order Quantity          -1.208020
Sales                   60.928376
Discount                -0.959411
Profit                  67.349705
Unit Price             271.168733
Shipping Cost            7.751587
Product Base Margin     -0.660870
dtype: float64
```

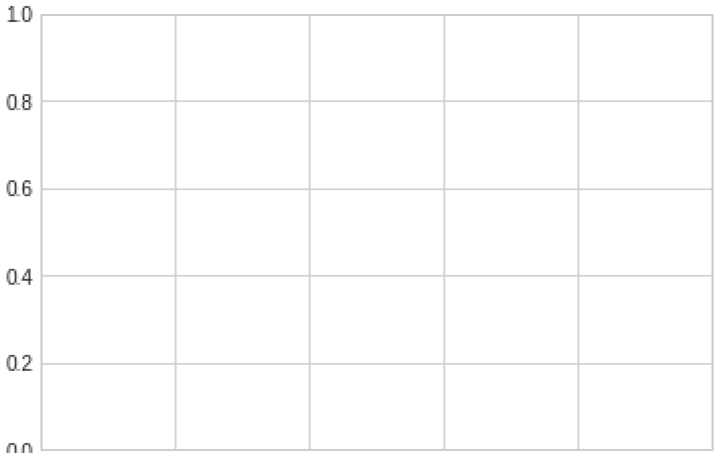
2. Write a python program to read a sampledata_orders.csv file perform descriptive statics for the first 25 rows of the sales column and print the result.

```
import pandas as pd
orders = pd.read_csv("sampledata_orders.csv")
```

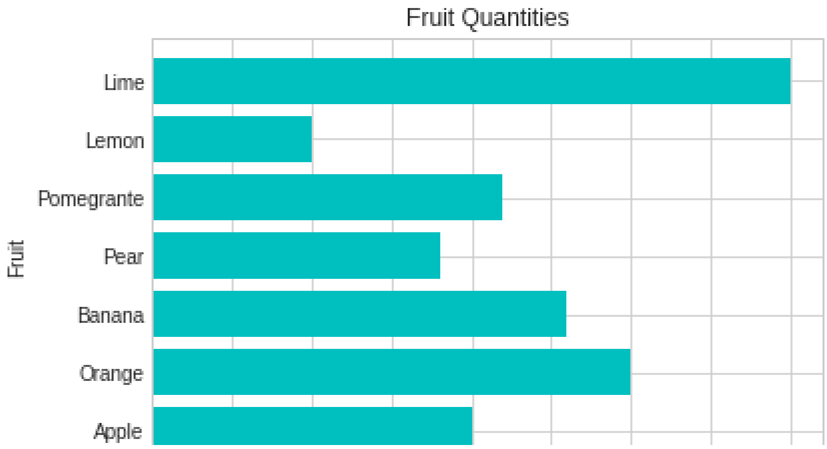
```
orders[0:25:1][["Sales"]].min()
orders[0:25:1][["Sales"]].max()
orders[0:25:1][["Sales"]].std()
orders[0:25:1][["Sales"]].var()
orders[0:25:1][["Sales"]].mean()
orders[0:25:1][["Sales"]].median()
orders[0:25:1][["Sales"]].mode()
orders[0:25:1][["Sales"]].sum()
orders[0:25:1][["Sales"]].count()
orders[0:25:1][["Sales"]].rank()
orders[0:25:1][["Sales"]].skew()
orders[0:25:1][["Sales"]].kurt()
```

Horizontal bar chart

```
import numpy as np
import pandas as pd
#!pip install matplotlib==3.4
from matplotlib import pyplot as plt
plt.style.use('seaborn-whitegrid')
import pandas as pd
import numpy as np
fig = plt.figure()
ax = plt.axes()
```

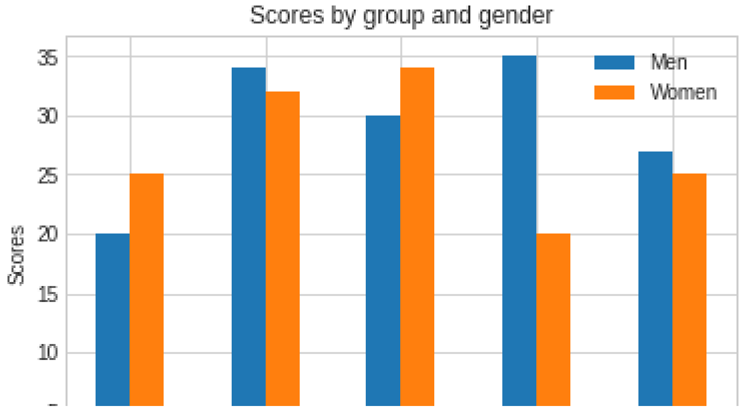


```
from matplotlib import pyplot as plt
plt.style.use('seaborn-whitegrid')
fig = plt.figure()
ax = plt.axes()
data = {'Apple': 10,
        'Orange': 15,
        'Banana': 13,
        'Pear': 9,
        'Pomegrante': 11,
        'Lemon': 5,
        'Lime': 20}
names = list(data.keys())
values = list(data.values())
ax = plt.axes()
ax.barh(names, values, color='c', height=0.80)            # Horizontal bar-chart
ax.set_xlabel('Quantity')
ax.set_ylabel('Fruit')
ax.set_title('Fruit Quantities');
```

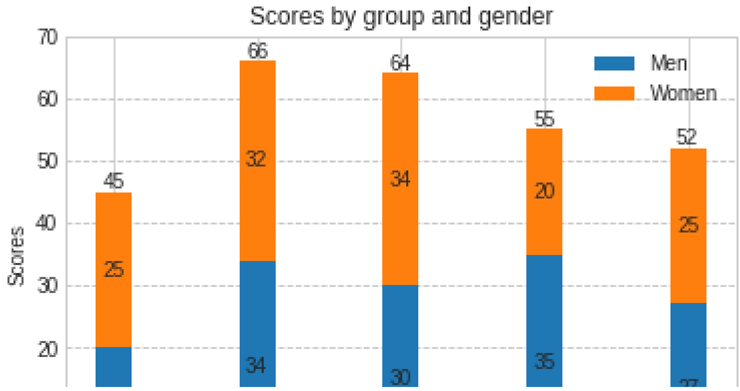


```
labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 34, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]
x = np.arange(len(labels))  # the label locations
width = 0.25  # the width of the bars
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xlabel('Groups')
ax.set_xticklabels(labels)
ax.legend();

# plt.show()
```



```
fig, ax = plt.subplots()
ax.grid(linestyle='--', color='0.75', axis = 'y')
ax.set_axisbelow(True)    # Grid behind bars
p1 = ax.bar(labels, men_means, width, label='Men')
p2 = ax.bar(labels, women_means, width, bottom=men_means,
       label='Women')
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.legend()
# Label with label_type 'center' instead of the default 'edge'
ax.bar_label(p1, label_type='center')
ax.bar_label(p2, label_type='center')
ax.bar_label(p2)
ax.set_ylim(0,70)
```



flights Data set

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
flights = pd.read_csv("/content/flights.csv")
flights.head()
```

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum
0	2013	1	1	517.0	2.0	830.0	11.0	UA	N14228
1	2013	1	1	533.0	4.0	850.0	20.0	UA	N24211
2	2013	1	1	542.0	2.0	923.0	33.0	AA	N619AA
3	2013	1	1	554.0	-6.0	812.0	-25.0	DL	N668DN
4	2013	1	1	554.0	-4.0	740.0	12.0	UA	N39463

```
flights.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 160754 entries, 0 to 160753
Data columns (total 16 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   year       160754 non-null  int64  
 1   month      160754 non-null  int64  
 2   day        160754 non-null  int64  
 3   dep_time   158418 non-null  float64
 4   dep_delay  158418 non-null  float64
 5   arr_time   158275 non-null  float64
 6   arr_delay  157927 non-null  float64
 7   carrier    160754 non-null  object 
 8   tailnum    159321 non-null  object 
 9   flight     160754 non-null  int64  
 10  origin     160754 non-null  object
 11  dest       160754 non-null  object
 12  air_time   157927 non-null  float64
 13  distance   160754 non-null  int64
 14  hour       158418 non-null  float64
 15  minute     158418 non-null  float64
dtypes: float64(7), int64(5), object(4)
memory usage: 19.6+ MB
```

```
# Select the rows that have at least one missing value
flights[flights.isnull().any(axis=1)]
```

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailn
330	2013	1	1	1807.0	29.0	2251.0	NaN	UA	N314
403	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EH
404	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EV
855	2013	1	2	2145.0	16.0	NaN	NaN	UA	N122
858	2013	1	2	NaN	NaN	NaN	NaN	AA	N
...	...	...	...	...	...	...	...	...	
159681	2013	9	28	1214.0	-11.0	1801.0	NaN	AA	N488
159854	2013	9	28	NaN	NaN	NaN	NaN	AA	N320
159855	2013	9	28	NaN	NaN	NaN	NaN	US	N
160185	2013	9	29	1734.0	23.0	2159.0	NaN	UA	N463
160286	2013	9	29	NaN	NaN	NaN	NaN	UA	N

2827 rows × 16 columns

2. Filter all the rows where arr_delay value is missing:

```
flights1 = flights[ flights['arr_delay'].notnull( )]
flights1.head(10)
```

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	f
0	2013	1	1	517.0	2.0	830.0	11.0	UA	N14228	
1	2013	1	1	533.0	4.0	850.0	20.0	UA	N24211	
2	2013	1	1	542.0	2.0	923.0	33.0	AA	N619AA	
3	2013	1	1	554.0	-6.0	812.0	-25.0	DL	N668DN	
4	2013	1	1	554.0	-4.0	740.0	12.0	UA	N39463	

3. Remove all the observations with missing values

```
flights2 = flights.dropna()
flights2
```

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailn
0	2013	1	1	517.0	2.0	830.0	11.0	UA	N142
1	2013	1	1	533.0	4.0	850.0	20.0	UA	N242
2	2013	1	1	542.0	2.0	923.0	33.0	AA	N619
3	2013	1	1	554.0	-6.0	812.0	-25.0	DL	N668
4	2013	1	1	554.0	-4.0	740.0	12.0	UA	N394
...	...	...	...	...	...	...	...	...	
160749	2013	9	30	2105.0	-1.0	2329.0	-25.0	UA	N477
160750	2013	9	30	2121.0	21.0	2349.0	-25.0	DL	N193
160751	2013	9	30	2140.0	0.0	10.0	-30.0	AA	N335
160752	2013	9	30	2149.0	-7.0	2245.0	-23.0	UA	N813
160753	2013	9	30	2233.0	80.0	112.0	42.0	UA	N578

157927 rows × 16 columns

4. Fill missing values with zeros for dep_delay

```
nomiss =flights['dep_delay'].fillna(0)
nomiss
```

```
nomiss.isnull().any()
```

```
False
```

5. Let's compute summary statistic per a group':

6. Using agg() methods for aggregation fetch min,max and mean for columns dep_delay and arr_delay:

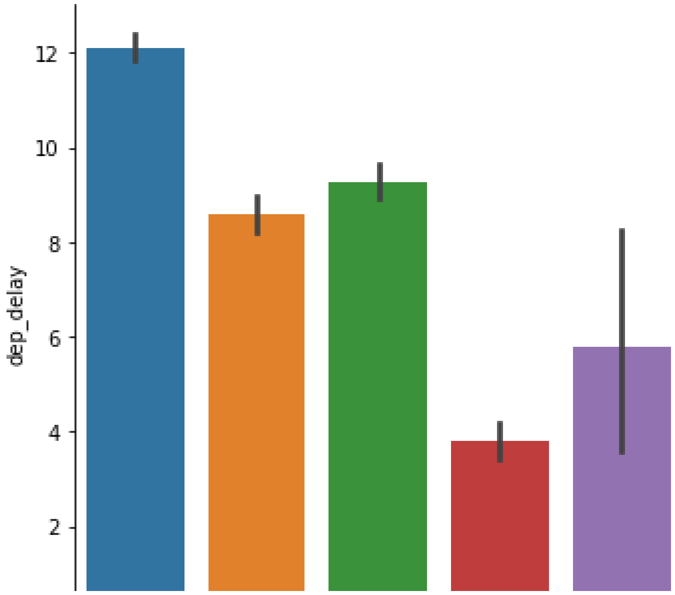
```
flights[['dep_delay','arr_delay']].agg(['min','mean','max'])
```

	dep_delay	arr_delay
min	-33.000000	-75.000000
mean	9.463773	2.094537
max	1014.000000	1007.000000

7. Using seaborn plot factorplot

```
import seaborn as sns
sns.catplot(x='carrier',y='dep_delay', data=flights, kind='bar')
```

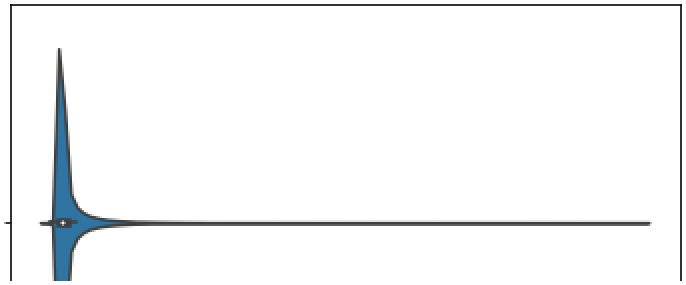
```
<seaborn.axisgrid.FacetGrid at 0x7fbdeb0ed760>
```



Violin plot

```
sns.violinplot(x = "dep_delay", data=flights)
```

```
<Axes: xlabel='dep_delay'>
```



```
sns.scatterplot(x='carrier', y='dep_delay', data=flights)
```

```
sns.jointplot(x='carrier', y='dep_delay', data=flights)
```

```
sns.boxplot(x='carrier',y='dep_delay', data=flights)
```

```
sns.swarmplot(x='carrier',y='dep_delay', data=flights)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-18-85e526d2b4d0> in <module>
----> 1 sns.swarmplot(x='carrier',y='dep_delay', data=flights)
```

14 frames

```
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py in
first_non_overlapping_candidate(self, candidates, neighbors)
   3502             dx = neighbors_x - x_i
   3503             dy = neighbors_y - y_i
-> 3504             sq_distances = np.square(dx) + np.square(dy)
   3505 
   3506             sep_needed = np.square(neighbors_r + r_i)

KeyboardInterrupt: 
```

SEARCH STACK OVERFLOW

```
Error in callback <function flush_figures at 0x7fbe1ea9f0d0> (for post_execute)
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/usr/local/lib/python3.9/dist-packages/ipykernel/pylab/backend_inline.py in
flush_figures()
    119         # ignore the tracking, just draw and close all figures
    120         try:
--> 121             return show(True)
    122         except Exception as e:
    123             # safely show traceback if in IPython, else raise
```

18 frames

```
<decorator-gen-2> in __call__(self, obj)

/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py in
first_non_overlapping_candidate(self, candidates, neighbors)
   3509             # squared distance between candidate and any of the
neighbors has
   3510             # to be at least square of the summed radii
-> 3511             good_candidate = np.all(sq_distances >= sep_needed)
   3512 
   3513             if good_candidate:

KeyboardInterrupt: 
```

SEARCH STACK OVERFLOW

```
sns.pairplot(flights)
```

Salaries data set

```
import pandas as pd
df = pd.read_csv("/content/Salaries.csv")
```

```
df.head(10)
```

Rename the columns

```
df_new =df.rename(columns={
        'discipline': 'subject',
        'sex': 'gender'
    })
df_new
```

Create a new column

```
df = df.assign( salary_k = lambda x: x.salary/1000.0)
df.head(10)
```

	rank	discipline	phd	service	sex	salary	salary_k
0	Prof	B	56	49	Male	186960	186.960
1	Prof	A	12	6	Male	93000	93.000
2	Prof	A	23	20	Male	110515	110.515
3	Prof	A	40	31	Male	131205	131.205
4	Prof	B	20	18	Male	104800	104.800
5	Prof	A	20	20	Male	122400	122.400
6	AssocProf	A	20	17	Male	81285	81.285
7	Prof	A	18	18	Male	126300	126.300
8	Prof	A	29	19	Male	94350	94.350
9	Prof	A	51	51	Male	57800	57.800

Check how many unique values in a column

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

```
array(['Prof', 'AssocProf', 'AsstProf'], dtype=object)
```

Get frequency table for a categorical or binary column

```
df['rank'].value_counts()
```

Calculate the mean salary for men and women.

```
df.groupby('sex')['salary'].mean()
```

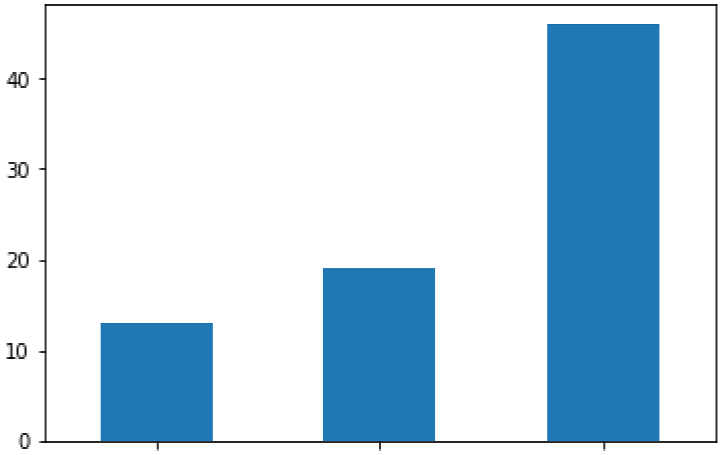
Group using 2 variables - sex and rank:

```
df.groupby(['rank','sex'], sort=True)[['salary']].mean()
```

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.hist(df['salary'],bins=8, density=True)
```

```
df.groupby(['rank'])['salary'].count().plot(kind='bar')
```

```
<Axes: xlabel='rank'>
```



Data Cleaning

```
import pandas as pd
data =pd.read_csv('/content/data.csv')
print(data)
```

```
data.describe()
```

```
data.rank()
```

```
data.isnull()
```

```
data.dropna()
```

```
from numpy import NaN
data.replace({NaN:0.00})
data.fillna(3)
```

```
data.fillna(method='pad')
```

```
data.fillna(method='backfill') #Example of replacing missing values by filling backward
```

```
data.drop_duplicates()
```

```
del data['YOB']
print(data)
```

```
print(data.rename(columns={'Name':'FirstName','Surname':'LastName'}))
```

Simple bar chart

```
data = {'Apple': 10, 'Orange': 15, 'Lemon': 5, 'Lime': 20}
names = list(data.keys())
values = list(data.values())

fig = plt.figure()
ax = plt.axes()
# ax.grid(linestyle='--', color='0.85')  # Color: grayscale between 0 and 1

ax.bar(names, values);
```

```
data = {'Apple': 10, 'Orange': 15, 'Lemon': 5, 'Lime': 20, 'Bannan': 30, 'WM': 18, 'mango'
names = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize=(10,7))
ax = plt.axes()

ax.bar(names, values, color='r', width = 0.40);
ax.set_xlabel('Fruit')
ax.set_ylabel('Quantity')
ax.set_title('Fruit Quantities');
```

```
data = {'Apple': 10,
        'Orange': 15,
        'Banana': 13,
        'Pear': 9,
        'Pomegrante': 11,
        'Lemon': 5,
        'Lime': 20}
names = list(data.keys())
values = list(data.values())

ax = plt.axes()

ax.barh(names, values, color='c', height=0.80)            # Horizontal bar-chart
ax.set_xlabel('Quantity')
ax.set_ylabel('Fruit')
ax.set_title('Fruit Quantities');
```

```
labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 34, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]

x = np.arange(len(labels))  # the label locations
width = 0.40  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')

# # Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_xlabel('Groups')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend();
plt.show()
```

Scatter Plot

```
x = np.random.randn(20)
y = np.random.randn(20)
fig, ax = plt.subplots()
ax.scatter(x, y);
fig, ax = plt.subplots()        # a figure with a single Axes
ax.scatter(x, y, marker = "v");
fig, axs = plt.subplots(2, 3, sharex=True, sharey=True, figsize=(16,12));

# marker symbol
axs[0, 0].scatter(x, y, s=80, marker=">")
axs[0, 0].set_title("Right Triangle ")
# marker from TeX
axs[0, 1].scatter(x, y, s=80, marker=r'$\alpha$')
axs[0, 1].set_title(r"marker=r'\$\alpha\$'")

# marker from path
verts = [[-1, -1], [1, -1], [1, 1], [-1, -1]]
axs[0, 2].scatter(x, y, s=80, marker=verts)
axs[0, 2].set_title("marker=verts")

# regular polygon marker
axs[1, 0].scatter(x, y, s=80, marker=(5, 0))
axs[1, 0].set_title("polygon")

# regular star marker
axs[1, 1].scatter(x, y, s=80, marker=(5, 1))
axs[1, 1].set_title("star")

# regular asterisk marker
axs[1, 2].scatter(x, y, s=80, marker=(5, 2))
axs[1, 2].set_title("asterisk");
```