Python Pandas for Data Engineering

In [1]:
```
1 import pandas as pd
```

In [2]:
```
1 # How to create dataframe
2 mydata=pd.DataFrame({
3     'Name':['A','B','C'],
4     'Age':[1,2,3]
5 })
6 mydata
7
```

Out[2]:

	Name	Age
0	A	1
1	B	2
2	C	3

In [62]:
```
# PandasAssignment
# Q1. How do you load a CSV file into a Pandas DataFrame?
 # print row in limit
pd.options.display.max_rows==20
df=pd.read_csv("D:/Data engineering/Python Practice/customers.csv")
#dataframe=df
df.head(20)
```

Out[62]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	1	Male	19	15000	39	Healthcare	1	4
1	2	Male	21	35000	81	Engineer	3	3
2	3	Female	20	86000	6	Engineer	1	1
3	4	Female	23	59000	77	Lawyer	0	2
4	5	Female	31	38000	40	Entertainment	2	6
5	6	Female	22	58000	76	Artist	0	2
6	7	Female	35	31000	6	Healthcare	1	3
7	8	Female	23	84000	94	Healthcare	1	3
8	9	Male	64	97000	3	Engineer	0	3
9	10	Female	30	98000	72	Artist	1	4
10	11	Male	67	7000	14	Engineer	1	3
11	12	Female	35	93000	99	Healthcare	4	4
12	13	Female	58	80000	15	Executive	0	5
13	14	Female	24	91000	77	Lawyer	1	1
14	15	Male	37	19000	13	Doctor	0	1
15	16	Male	22	51000	79	Healthcare	1	2
16	17	Female	35	29000	35	Homemaker	9	5
17	18	Male	20	89000	66	Healthcare	1	6
18	19	Male	52	20000	29	Entertainment	1	4
19	20	Female	35	62000	98	Artist	0	1

In [4]:
```
# sum of null value.
df.isnull().sum()
```

Out[4]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income ($)         0
Spending Score (1-100)    0
Profession               35
Work Experience           0
Family Size               0
dtype: int64
```

In [5]:
```
df.rename(columns={"Profession":"Dandha"})
```

Out[5]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Dandha	Work Experience	Family Size
0	1	Male	19	15000	39	Healthcare	1	4
1	2	Male	21	35000	81	Engineer	3	3
2	3	Female	20	86000	6	Engineer	1	1
3	4	Female	23	59000	77	Lawyer	0	2
4	5	Female	31	38000	40	Entertainment	2	6
...	...	...	...	...	...	...	...	...
1995	1996	Female	71	184387	40	Artist	8	7
1996	1997	Female	91	73158	32	Doctor	7	7
1997	1998	Male	87	90961	14	Healthcare	9	2
1998	1999	Male	77	182109	4	Executive	7	2
1999	2000	Male	90	110610	52	Entertainment	5	2

2000 rows × 8 columns

#Q25. What is the difference between .loc and .iloc in Pandas? #iloc selects rows based on an integer index. So, if you want to select the 5th row in a DataFrame, you would use df.iloc[[4]] since the first row is at index 0, the second row is at index 1, and so on. .loc selects rows based on a labeled index. So, if you want to select the row with an index label of 5, you would directly use df.loc[[5]]

In [6]:
```
# Q2. How do you check the data type of a column in a Pandas DataFrame?
df["Age"].dtypes # check for singel column.
df.dtypes
```

Out[6]:
```
CustomerID                int64
Gender                   object
Age                       int64
Annual Income ($)         int64
Spending Score (1-100)    int64
Profession               object
Work Experience           int64
Family Size               int64
dtype: object
```

In [7]:
```
1
2
3 # loc is used to base on label  index value.
4 # and iloc used on integer index.
5 x=df.loc[(df["Age"]==20) & (df["Profession"]=="Lawyer") & (df["Annual Inc
6 x
7 # indexing
8 # df.loc[200:500]
```

Out[7]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
944	945	Female	20	184801	8	Lawyer	1	1
1801	1802	Male	20	189650	60	Lawyer	8	4

In [8]:
```
1 # Q3. How do you select rows from a Pandas DataFrame based on a condition
2 age=df.loc[(df["Age"]==25)] # row where age ==25
3 age
4 # print row where age=25 and profession=engineer
5 age=df.loc[ (df["Age"]==25) & (df["Profession"]=='Engineer')]
6 age
```

Out[8]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
144	145	Male	25	71000	12	Engineer	0	5
768	769	Female	25	106569	84	Engineer	1	3

In [9]:
```
# Q4. How do you rename columns in a Pandas DataFrame?
df=df.rename(columns={'Family Size':'Member'})
df
```

Out[9]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Member
0	1	Male	19	15000	39	Healthcare	1	4
1	2	Male	21	35000	81	Engineer	3	3
2	3	Female	20	86000	6	Engineer	1	1
3	4	Female	23	59000	77	Lawyer	0	2
4	5	Female	31	38000	40	Entertainment	2	6
...	...	...	...	...	...	...	...	...
1995	1996	Female	71	184387	40	Artist	8	7
1996	1997	Female	91	73158	32	Doctor	7	7
1997	1998	Male	87	90961	14	Healthcare	9	2
1998	1999	Male	77	182109	4	Executive	7	2
1999	2000	Male	90	110610	52	Entertainment	5	2

2000 rows × 8 columns

In [10]:
```
# Q5. How do you drop columns in a Pandas DataFrame?
df=df.drop(columns=["Spending Score (1-100)"])
df.head()
df
```

Out[10]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
0	1	Male	19	15000	Healthcare	1	4
1	2	Male	21	35000	Engineer	3	3
2	3	Female	20	86000	Engineer	1	1
3	4	Female	23	59000	Lawyer	0	2
4	5	Female	31	38000	Entertainment	2	6
...	...	...	...	...	...	...	...
1995	1996	Female	71	184387	Artist	8	7
1996	1997	Female	91	73158	Doctor	7	7
1997	1998	Male	87	90961	Healthcare	9	2
1998	1999	Male	77	182109	Executive	7	2
1999	2000	Male	90	110610	Entertainment	5	2

2000 rows × 7 columns

In [11]:
```
# Q6. How do you find the unique values in a column of a Pandas DataFrame
df_unique=df["Age"].unique()
df_unique
```

Out[11]:
```
array([19, 21, 20, 23, 31, 22, 35, 64, 30, 67, 58, 24, 37, 52, 25, 46, 54,
       29, 45, 40, 60, 53, 18, 49, 42, 36, 65, 48, 50, 27, 33, 59, 47, 51,
       69, 70, 63, 43, 68, 32, 26, 57, 38, 55, 34, 66, 39, 44, 28, 56, 41,
       16, 76, 62, 80,  1,  0, 86, 79, 83, 95, 93, 78, 15,  6, 84,  4, 91,
       14, 92, 77, 89, 12,  7, 94, 96, 74, 85, 73,  9, 10, 11, 17, 90, 61,
       13, 72,  5, 75, 99, 88, 82,  8, 87,  3, 97, 81, 98,  2, 71],
      dtype=int64)
```

In [12]:
```
# count unique number
nun=df["Age"].nunique()
nun
```

Out[12]:
```
100
```

In [13]:
```
# max salary and profession
max_salary=df["Annual Income ($)"].max()
max_salary
```

Out[13]:
```
189974
```

In [14]:
```
mean_salary=df["Annual Income ($)"].mean()
mean_salary
```

Out[14]:
```
110731.8215
```

In [15]:
```
# sum of zero value in
sum_zero=(df["Annual Income ($)"]==0).sum()
sum_zero
```

Out[15]:
```
2
```

In [16]:
```
# drop zero value.
df.drop(columns=[])
```

Out[16]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
0	1	Male	19	15000	Healthcare	1	4
1	2	Male	21	35000	Engineer	3	3
2	3	Female	20	86000	Engineer	1	1
3	4	Female	23	59000	Lawyer	0	2
4	5	Female	31	38000	Entertainment	2	6
...	...	...	...	...	...	...	...
1995	1996	Female	71	184387	Artist	8	7
1996	1997	Female	91	73158	Doctor	7	7
1997	1998	Male	87	90961	Healthcare	9	2
1998	1999	Male	77	182109	Executive	7	2
1999	2000	Male	90	110610	Entertainment	5	2

2000 rows × 7 columns

In [17]:
```
# fill zero with mean in annual income
df['Annual Income ($)']=df["Annual Income ($)"].replace(0,mean_salary)
df
```

Out[17]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
0	1	Male	19	15000.0	Healthcare	1	4
1	2	Male	21	35000.0	Engineer	3	3
2	3	Female	20	86000.0	Engineer	1	1
3	4	Female	23	59000.0	Lawyer	0	2
4	5	Female	31	38000.0	Entertainment	2	6
...	...	...	...	...	...	...	...
1995	1996	Female	71	184387.0	Artist	8	7
1996	1997	Female	91	73158.0	Doctor	7	7
1997	1998	Male	87	90961.0	Healthcare	9	2
1998	1999	Male	77	182109.0	Executive	7	2
1999	2000	Male	90	110610.0	Entertainment	5	2

2000 rows × 7 columns

In [18]:
```
# maximum salary record of employee , print all row
mx=df["Annual Income ($)"].idxmax()
max_salary_record=df.loc[mx]
print(pd.DataFrame(max_salary_record))
```

```
                        569
CustomerID              570
Gender               Female
Age                      91
Annual Income ($)  189974.0
Profession         Engineer
Work Experience           8
Member                    1
```

In [19]:
```
# min salary record of employee , print all row
min_sal=df["Annual Income ($)"].idxmin()
min_salary_record=df.loc[min_sal]
m=pd.DataFrame(min_salary_record)
m
```

Out[19]:

	272
CustomerID	273
Gender	Female
Age	96
Annual Income ($)	1000.0
Profession	Entertainment
Work Experience	0
Member	3

In [20]:
```
# second heighest salary of employee.
second_s=df["Annual Income ($)"].sort_values(ascending=False)
sc=second_s.iloc[1]
sc
```

Out[20]: 189945.0

In [21]:
```
# record with second salary
second_s=df["Annual Income ($)"].sort_values(ascending=False)
sc=second_s.iloc[1]
sc_record=df[df["Annual Income ($)"]==sc]
sc_record
```

Out[21]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
1257	1258	Male	60	189945.0	Engineer	0	5

In [22]:
```
 #profession
df['Profession'].unique()
```

Out[22]:
```
array(['Healthcare', 'Engineer', 'Lawyer', 'Entertainment', 'Artist',
       'Executive', 'Doctor', 'Homemaker', 'Marketing', nan], dtype=object)
```

In [23]:
```
# average income of group by profession
type(df)
avg=df.groupby("Profession")["Annual Income ($)"].mean()
# round is used to round value.
avgd=round(pd.DataFrame(avg))
avgd
```

Out[23]:

Profession	Annual Income ($)
Artist	108958.0
Doctor	112261.0
Engineer	111161.0
Entertainment	110650.0
Executive	113770.0
Healthcare	112574.0
Homemaker	108759.0
Lawyer	110996.0
Marketing	107994.0

In [24]:
```
df["Annual Income ($)"].mean()
```

Out[24]:
```
110842.5533215
```

In [25]:
```
# sum of null value .
df.isnull().sum()
```

Out[25]:
```
CustomerID            0
Gender                0
Age                   0
Annual Income ($)     0
Profession           35
Work Experience       0
Member                0
dtype: int64
```

In [26]:
```
# slalary agg max,min,
df.groupby("Profession").agg({'Annual Income ($)':['max','min','mean']})
```

Out[26]:

	Annual Income ($)		
	max	min	mean
Profession			
Artist	189709.0	2000.0	108957.514414
Doctor	189672.0	3000.0	112260.992680
Engineer	189974.0	7000.0	111161.240223
Entertainment	186882.0	1000.0	110650.333333
Executive	189630.0	4000.0	113770.130719
Healthcare	189689.0	3000.0	112574.041298
Homemaker	188696.0	29000.0	108758.616667
Lawyer	189650.0	3000.0	110995.838028
Marketing	186069.0	5000.0	107994.211765

In [27]:
```
# Q7. How do you find the number of missing values in each column of a Pa
print(df.isna().sum()) # isna mean missing value.
# df.isna().sum()
# Q8. How do you fill missing values in a Pandas DataFrame with a specifi
df=df.fillna("finding_job")
df
```

	CustomerID	Gender	Age	($)	Profession	Experience	Member
0	1	Male	19	15000.0	Healthcare	1	4
1	2	Male	21	35000.0	Engineer	3	3
2	3	Female	20	86000.0	Engineer	1	1
3	4	Female	23	59000.0	Lawyer	0	2
4	5	Female	31	38000.0	Entertainment	2	6
...	...	...	...	...	...	...	...
1995	1996	Female	71	184387.0	Artist	8	7
1996	1997	Female	91	73158.0	Doctor	7	7
1997	1998	Male	87	90961.0	Healthcare	9	2
1998	1999	Male	77	182109.0	Executive	7	2
1999	2000	Male	90	110610.0	Entertainment	5	2

2000 rows x 7 columns

In [28]:
```
df.isna().sum()
```

Out[28]:
```
CustomerID           0
Gender               0
Age                  0
Annual Income ($)    0
Profession           0
Work Experience      0
Member               0
dtype: int64
```

In [29]:
```
# Q9. How do you concatenate two Pandas DataFrames?
df1=pd.DataFrame({
    'Name':['Smith','Alexander','Sara'],
    'Age':[23,44,22]
})
df1
df2=pd.DataFrame({
    'City':['erlangen','Berlin','munich'],
    'Area':[2300,4400,2200]
})
df2
con_df=pd.concat([df1,df2],axis=1) # ignore index mean no seprate indexin
con_df
```

Out[29]:

	Name	Age	City	Area
0	Smith	23	erlangen	2300
1	Alexander	44	Berlin	4400
2	Sara	22	munich	2200

In [30]:
```
# Q10. How do you merge two Pandas DataFrames on a specific column?
# chose same value from both dataframe
df1=pd.DataFrame({
    'Name':['Smith','Alexander','Sara'],
    'Age':[23,44,22]
})
df1
df2=pd.DataFrame({
   'Name':['Ali','Alexander','Raza'],
    'Age':[23,44,21]
})
df2
# x=df1.merge(df2[['Name','Age']]) # in merge only matching column appera
# x
df1.merge(df2)
```

Out[30]:

	Name	Age
0	Alexander	44

In [31]:
```
right=df1.merge(df2[['Name']],on='Name',how='right') #right merge on the
print(right)
print("-----------------------")
left=df1.merge(df2[['Name']],on='Name',how='left') #left merge on the bas
print(left)
print("-----------------------")
inner=df1.merge(df2[['Name']],on='Name',how='inner') #inner merge on the
print(inner)
print("-----------------------")
outer=df1.merge(df2[['Name']],on='Name',how='outer') #outer merge on the
print(outer)

left_m=df1.merge((df2), how='outer')
left_m
```

```
        Name   Age
0        Ali   NaN
1  Alexander  44.0
2       Raza   NaN
-----------------------
        Name  Age
0      Smith   23
1  Alexander   44
2       Sara   22
-----------------------
        Name  Age
0  Alexander   44
-----------------------
        Name   Age
0      Smith  23.0
1  Alexander  44.0
2       Sara  22.0
3        Ali   NaN
4       Raza   NaN
```

Out[31]:

	Name	Age
0	Smith	23
1	Alexander	44
2	Sara	22
3	Ali	23
4	Raza	21

In [32]:
```
# Q11. How do you group data in a Pandas DataFrame by a specific column a
# Pandas comes with a whole host of sql-like aggregation functions
#you can apply when grouping on one or more columns.
group_agg=df.groupby('Profession').agg({'Work Experience':['min','max','m
group_agg
```

Out[32]:

	Work Experience		
	min	max	mean
Profession			
Artist	0	17	4.215686
Doctor	0	16	4.304348
Engineer	0	16	3.955307
Entertainment	0	15	3.500000
Executive	0	16	4.248366
Healthcare	0	16	4.002950
Homemaker	0	14	6.133333
Lawyer	0	17	3.528169
Marketing	0	15	4.305882
finding_job	0	12	4.657143

In [33]:
```
# Q12. How do you pivot a Pandas DataFrame?
# The pivot() function is used to reshaped a given DataFrame organized by
# This function does not support data aggregation, multiple values will r
try:
    x=df.pivot(index='CustomerID',values='Profession',columns='Gender')
    print(pd.DataFrame(x))
except ValueError:
    print("Index cannot be repeated")
# ValueError: Index contains duplicate entries, cannot reshape
#The Python ValueError is an exception that occurs when a function receiv

```

```
Gender             Female           Male
CustomerID
1                     NaN     Healthcare
2                     NaN       Engineer
3                Engineer            NaN
4                  Lawyer            NaN
5           Entertainment            NaN
...                   ...            ...
1996              Artist            NaN
1997              Doctor            NaN
1998                 NaN     Healthcare
1999                 NaN      Executive
2000                 NaN  Entertainment

[2000 rows x 2 columns]
```

In [34]:
```
# Q13. How do you change the data type of a column in a Pandas DataFrame?
data=pd.DataFrame({
    'Name':['Smith','Alexander','Sara'],
    'Age':[23.67,44.99,22.77]
})
print(data.dtypes)
data['Age']=data['Age'].astype(int)
data
# astype()
# to_numeric()
# convert_dtypes()

```

```
Name     object
Age     float64
dtype: object
```

Out[34]:

	Name	Age
0	Smith	23
1	Alexander	44
2	Sara	22

In [35]:
```
1 df["Annual Income ($)"].astype(int)
```

Out[35]:
```
0        15000
1        35000
2        86000
3        59000
4        38000
         ...
1995    184387
1996     73158
1997     90961
1998    182109
1999    110610
Name: Annual Income ($), Length: 2000, dtype: int32
```

In [36]:
```
1 # sum of zero age in age column
2 zero_age=(df["Age"]==0).sum()
3 zero_age
```

Out[36]:
```
24
```

In [37]:
```
1 # drop where age ==0
2 df.drop(df[df['Age']==0].index)
3 df
```

Out[37]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
0	1	Male	19	15000.0	Healthcare	1	4
1	2	Male	21	35000.0	Engineer	3	3
2	3	Female	20	86000.0	Engineer	1	1
3	4	Female	23	59000.0	Lawyer	0	2
4	5	Female	31	38000.0	Entertainment	2	6
...	...	...	...	...	...	...	...
1995	1996	Female	71	184387.0	Artist	8	7
1996	1997	Female	91	73158.0	Doctor	7	7
1997	1998	Male	87	90961.0	Healthcare	9	2
1998	1999	Male	77	182109.0	Executive	7	2
1999	2000	Male	90	110610.0	Entertainment	5	2

2000 rows × 7 columns

In [38]:
```
1 df
```

Out[38]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
0	1	Male	19	15000.0	Healthcare	1	4
1	2	Male	21	35000.0	Engineer	3	3
2	3	Female	20	86000.0	Engineer	1	1
3	4	Female	23	59000.0	Lawyer	0	2
4	5	Female	31	38000.0	Entertainment	2	6
...	...	...	...	...	...	...	...
1995	1996	Female	71	184387.0	Artist	8	7
1996	1997	Female	91	73158.0	Doctor	7	7
1997	1998	Male	87	90961.0	Healthcare	9	2
1998	1999	Male	77	182109.0	Executive	7	2
1999	2000	Male	90	110610.0	Entertainment	5	2

2000 rows × 7 columns

In [107]:
```
# Print those customer which have age greater then 18
Age=df_copy.loc[df_copy['Age']== 18]
Age
```

Out[107]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member	Income status
33	34	Male	18	62000.0	Homemaker	9	7	Bad
407	408	Male	18	103896.0	Executive	16	4	Good
1915	1916	Female	18	134752.0	Engineer	6	5	Good
1695	1696	Female	18	149915.0	Doctor	14	1	Good
1703	1704	Female	18	53812.0	Homemaker	9	5	Bad
941	942	Male	18	185861.0	Artist	3	6	Good
1624	1625	Female	18	174836.0	Healthcare	0	4	Good
65	66	Male	18	9000.0	Entertainment	0	2	Bad
1874	1875	Male	18	162983.0	Entertainment	0	6	Good
1274	1275	Male	18	157757.0	Doctor	0	4	Good
1532	1533	Male	18	107794.0	Engineer	0	5	Good
273	274	Female	18	143000.0	Artist	0	2	Good
1286	1287	Female	18	62651.0	Entertainment	0	2	Bad
1296	1297	Female	18	180956.0	Artist	2	2	Good
897	898	Male	18	62362.0	Lawyer	7	3	Bad
1803	1804	Female	18	62151.0	Artist	8	2	Bad
389	390	Female	18	86203.0	Doctor	13	1	Good
1326	1327	Male	18	77873.0	Executive	6	5	Good
91	92	Male	18	36000.0	Artist	1	4	Bad
1435	1436	Female	18	82812.0	Artist	6	2	Good
114	115	Female	18	97000.0	Executive	0	3	Good
1162	1163	Female	18	183071.0	Artist	7	4	Good
977	978	Male	18	129260.0	Doctor	1	3	Good

In [40]:
```
# Q14. How do you sort a Pandas DataFrame by a specific column?
df=df.sort_values('Age') # age is sorted
df
#sort_index
```

Out[40]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
443	444	Female	0	68761.0	Lawyer	1	4
466	467	Male	0	186002.0	Doctor	15	2
1271	1272	Female	0	61228.0	Entertainment	1	6
1248	1249	Female	0	143082.0	Entertainment	0	7
1583	1584	Female	0	120899.0	Marketing	2	6
...	...	...	...	...	...	...	...
1629	1630	Male	99	162762.0	Healthcare	1	1
1188	1189	Female	99	122548.0	finding_job	2	5
351	352	Male	99	173394.0	Engineer	13	1
361	362	Male	99	63364.0	Entertainment	1	2
347	348	Female	99	184426.0	Artist	9	1

2000 rows × 7 columns

In [41]:
```
# Q15. How do you create a copy of a Pandas DataFrame?
df_copy=df.copy()
df_copy
```

Out[41]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
443	444	Female	0	68761.0	Lawyer	1	4
466	467	Male	0	186002.0	Doctor	15	2
1271	1272	Female	0	61228.0	Entertainment	1	6
1248	1249	Female	0	143082.0	Entertainment	0	7
1583	1584	Female	0	120899.0	Marketing	2	6
...	...	...	...	...	...	...	...
1629	1630	Male	99	162762.0	Healthcare	1	1
1188	1189	Female	99	122548.0	finding_job	2	5
351	352	Male	99	173394.0	Engineer	13	1
361	362	Male	99	63364.0	Entertainment	1	2
347	348	Female	99	184426.0	Artist	9	1

2000 rows × 7 columns

In [42]:
```
1 df_copy.sort_index()
```

Out[42]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
0	1	Male	19	15000.0	Healthcare	1	4
1	2	Male	21	35000.0	Engineer	3	3
2	3	Female	20	86000.0	Engineer	1	1
3	4	Female	23	59000.0	Lawyer	0	2
4	5	Female	31	38000.0	Entertainment	2	6
...	...	...	...	...	...	...	...
1995	1996	Female	71	184387.0	Artist	8	7
1996	1997	Female	91	73158.0	Doctor	7	7
1997	1998	Male	87	90961.0	Healthcare	9	2
1998	1999	Male	77	182109.0	Executive	7	2
1999	2000	Male	90	110610.0	Entertainment	5	2

2000 rows × 7 columns

In [71]:
```
# Q16. How do you filter rows of a Pandas DataFrame by multiple condition
df=df_copy
df=df.loc[df['Age']==20]
df
```

Out[71]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member	Income status
17	18	Male	20	89000.0	Healthcare	1	6	Good
99	100	Male	20	80000.0	Engineer	3	3	Good
522	523	Male	20	119952.0	Healthcare	8	2	Good
505	506	Female	20	111896.0	Executive	1	3	Good
39	40	Female	20	69000.0	Artist	8	2	Good
1024	1025	Female	20	146221.0	Marketing	0	3	Good
473	474	Male	20	130813.0	Artist	17	5	Good
1845	1846	Female	20	113217.0	Lawyer	9	1	Good
357	358	Male	20	184324.0	Marketing	13	5	Good
1901	1902	Female	20	131810.0	Healthcare	5	7	Good
1421	1422	Female	20	124852.0	Healthcare	5	5	Good
1829	1830	Female	20	113767.0	Artist	6	2	Good
1396	1397	Female	20	92048.0	Healthcare	1	1	Good
944	945	Female	20	184801.0	Lawyer	1	1	Good
1161	1162	Female	20	155951.0	Healthcare	7	4	Good
1412	1413	Male	20	106941.0	Artist	1	1	Good
725	726	Female	20	59844.0	Doctor	0	1	Bad
386	387	Female	20	151124.0	Artist	15	3	Good
1731	1732	Male	20	103195.0	Doctor	0	7	Good
1243	1244	Female	20	101941.0	Healthcare	1	3	Good
1801	1802	Male	20	189650.0	Lawyer	8	4	Good
2	3	Female	20	86000.0	Engineer	1	1	Good
134	135	Male	20	89000.0	Engineer	1	2	Good
1277	1278	Male	20	161156.0	Engineer	0	4	Good
1476	1477	Male	20	84008.0	Artist	8	4	Good
1947	1948	Female	20	148776.0	Artist	8	5	Good
1189	1190	Female	20	141428.0	Entertainment	1	5	Good

In [44]:
```
df=df.loc[df['Gender']=='Male']
df
```

Out[44]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
17	18	Male	20	89000.0	Healthcare	1	6
99	100	Male	20	80000.0	Engineer	3	3
522	523	Male	20	119952.0	Healthcare	8	2
473	474	Male	20	130813.0	Artist	17	5
357	358	Male	20	184324.0	Marketing	13	5
1412	1413	Male	20	106941.0	Artist	1	1
1731	1732	Male	20	103195.0	Doctor	0	7
1801	1802	Male	20	189650.0	Lawyer	8	4
134	135	Male	20	89000.0	Engineer	1	2
1277	1278	Male	20	161156.0	Engineer	0	4
1476	1477	Male	20	84008.0	Artist	8	4

In [67]:
```
df=df.loc[df["Profession"]=="Engineer"]
df
```

Out[67]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
1	2	Male	21	35000	81	Engineer	3	3
2	3	Female	20	86000	6	Engineer	1	1
8	9	Male	64	97000	3	Engineer	0	3
10	11	Male	67	7000	14	Engineer	1	3
44	45	Female	49	72000	28	Engineer	8	1
...	...	...	...	...	...	...	...	...
1959	1960	Female	85	155432	21	Engineer	4	6
1961	1962	Female	52	106530	46	Engineer	7	2
1972	1973	Male	98	104249	12	Engineer	5	2
1976	1977	Male	60	127438	82	Engineer	7	2
1994	1995	Female	19	54121	89	Engineer	6	3

179 rows × 8 columns

In [73]:
```
# Q17. How do you calculate the mean of a column in a Pandas DataFrame?
# df=df_copy
# df
df=df.agg({'Annual Income ($)':['mean','max','min']})
df
```

Out[73]:

	Annual Income ($)
mean	121137.592593
max	189650.000000
min	59844.000000

In [47]:
```
# Q18. How do you calculate the standard deviation of a column in a Panda
df["Annual Income ($)"].std()

```

Out[47]: 94902.00652042686

In [48]:
```
# Q19. How do you calculate the correlation between two columns in a Pand
df=df_copy
df
df["Annual Income ($)"].corr(df['Age'])

```

Out[48]: 0.019606863282448265

In [81]:
```
# Q20. How do you select specific columns in a DataFrame using their labe
lab=df_copy[['Age',"Annual Income ($)"]]
lab
```

Out[81]:

	Age	Annual Income ($)
443	0	68761.0
466	0	186002.0
1271	0	61228.0
1248	0	143082.0
1583	0	120899.0
...	...	...
1629	99	162762.0
1188	99	122548.0
351	99	173394.0
361	99	63364.0
347	99	184426.0

2000 rows × 2 columns

In [125]:
```
# Q21. How do you select specific rows in a DataFrame using their indexes
df_copy.loc[(df_copy["Profession"]=='Engineer') & (df_copy["Work Experien
            & (df_copy["Age"]>30)& (df_copy["Age"]<40)]
```

Out[125]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member	Income status
320	321	Male	36	69841.0	Engineer	10	2	Good

In [136]:
```
# maximum work experiance of engineering profession
mx_exp=df_copy[(df_copy["Profession"]=='Engineer')].max()
mx_exp
```

Out[136]:
```
CustomerID              1995
Gender                  Male
Age                       99
Annual Income ($)    189974.0
Profession           Engineer
Work Experience            16
Member                      8
Income status            Good
dtype: object
```

In [158]:
```
max_sal=df_copy["Annual Income ($)"].idxmax()
maxx_sal=df_copy.loc[max_sal]
maxx_sal
```

Out[158]:
```
CustomerID               570
Gender                Female
Age                       91
Annual Income ($)    189974.0
Profession           Engineer
Work Experience             8
Member                      1
Income status            Good
Name: 569, dtype: object
```

In [160]:
```
# maximum salary by each profession
df_copy.groupby("Profession")["Annual Income ($)"].max()
```

Out[160]:
```
Profession
Artist           189709.0
Doctor           189672.0
Engineer         189974.0
Entertainment    186882.0
Executive        189630.0
Healthcare       189689.0
Homemaker        188696.0
Lawyer           189650.0
Marketing        186069.0
finding_job      186655.0
Name: Annual Income ($), dtype: float64
```

In [165]:
```
# Q22. How do you sort a DataFrame by a specific column?
df_copy.sort_values("Work Experience")
```

Out[165]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member	Income status
1900	1901	Female	40	54631.0	Artist	0	6	Bad
745	746	Male	89	100938.0	Entertainment	0	7	Good
288	289	Female	21	5000.0	Marketing	0	1	Bad
951	952	Male	54	51166.0	Entertainment	0	7	Bad
1415	1416	Female	54	79925.0	Healthcare	0	3	Good
...	...	...	...	...	...	...	...	...
392	393	Male	21	119116.0	Artist	17	4	Good
405	406	Female	65	119889.0	Artist	17	6	Good
603	604	Female	91	69720.0	Lawyer	17	6	Good
473	474	Male	20	130813.0	Artist	17	5	Good
566	567	Female	19	180331.0	Artist	17	5	Good

2000 rows × 8 columns

In [52]:
```
# sort value on spicific column. max to low
df.sort_values(['Work Experience'],ascending=False)
```

Out[52]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member
566	567	Female	19	180331.0	Artist	17	5
392	393	Male	21	119116.0	Artist	17	4
405	406	Female	65	119889.0	Artist	17	6
603	604	Female	91	69720.0	Lawyer	17	6
473	474	Male	20	130813.0	Artist	17	5
...	...	...	...	...	...	...	...
1114	1115	Male	56	119296.0	Engineer	0	7
623	624	Male	79	86235.0	Artist	0	2
1550	1551	Female	16	95365.0	Artist	0	5
550	551	Male	16	58127.0	Marketing	0	3
687	688	Female	52	109425.0	Engineer	0	1

2000 rows × 7 columns

In [53]:
```
# Q23. How do you create a new column in a DataFrame based on the values
import numpy as np
df['Income status']=np.where(df['Annual Income ($)']>65000,'Good','Bad')
df
```

Out[53]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member	Income status
443	444	Female	0	68761.0	Lawyer	1	4	Good
466	467	Male	0	186002.0	Doctor	15	2	Good
1271	1272	Female	0	61228.0	Entertainment	1	6	Bad
1248	1249	Female	0	143082.0	Entertainment	0	7	Good
1583	1584	Female	0	120899.0	Marketing	2	6	Good
...	...	...	...	...	...	...	...	...
1629	1630	Male	99	162762.0	Healthcare	1	1	Good
1188	1189	Female	99	122548.0	finding_job	2	5	Good
351	352	Male	99	173394.0	Engineer	13	1	Good
361	362	Male	99	63364.0	Entertainment	1	2	Bad
347	348	Female	99	184426.0	Artist	9	1	Good

2000 rows × 8 columns

In [58]:
```
# Q24. How do you remove duplicates from a DataFrame?
df = pd.DataFrame({'team': ['A', 'A', 'A', 'A'],
                   'points': [10, 10, 12, 12],
                   'assists': [5, 5, 7, 9],
                    'email':['hassanrafiq687@gmail.com','hassanrafiq687@g
#view DataFrame
df
```

Out[58]:

	team	points	assists	email
0	A	10	5	hassanrafiq687@gmail.com
1	A	10	5	hassanrafiq687@gmail.com
2	A	12	7	bigdata@gmail.com
3	A	12	9	data@gmail.com

In [59]:
```
df_duplicate=df[df.duplicated('email')]
df_duplicate
```

Out[59]:

	team	points	assists	email
1	A	10	5	hassanrafiq687@gmail.com

In [60]:
```
drop=df.drop_duplicates('email')
drop
```

Out[60]:

	team	points	assists	email
0	A	10	5	hassanrafiq687@gmail.com
2	A	12	7	bigdata@gmail.com
3	A	12	9	data@gmail.com

In [63]:
```
# cutomer data
df
```

Out[63]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	1	Male	19	15000	39	Healthcare	1	4
1	2	Male	21	35000	81	Engineer	3	3
2	3	Female	20	86000	6	Engineer	1	1
3	4	Female	23	59000	77	Lawyer	0	2
4	5	Female	31	38000	40	Entertainment	2	6
...	...	...	...	...	...	...	...	...
1995	1996	Female	71	184387	40	Artist	8	7
1996	1997	Female	91	73158	32	Doctor	7	7
1997	1998	Male	87	90961	14	Healthcare	9	2
1998	1999	Male	77	182109	4	Executive	7	2
1999	2000	Male	90	110610	52	Entertainment	5	2

2000 rows × 8 columns

In [189]:
```
df_1=pd.DataFrame({'x1':['a','b','c'],
                   'x2':[1,2,3]})

df_2=pd.DataFrame({'x1':['a','b','d'],
                   'x3':['T','F','T']})
x=pd.merge(df_1,df_2,how='left',on='x1')
x
```

Out[189]:

	x1	x2	x3
0	a	1	T
1	b	2	F
2	c	3	NaN

In [190]:
```
x=pd.merge(df_1,df_2,how='right',on='x1')
x
```

Out[190]:

	x1	x2	x3
0	a	1.0	T
1	b	2.0	F
2	d	NaN	T

In [192]:
```
x=pd.merge(df_1,df_2,how='inner',on='x1')
x
```

Out[192]:

	x1	x2	x3
0	a	1	T
1	b	2	F

In [193]:
```
x=pd.merge(df_1,df_2,how='outer',on='x1')
x
```

Out[193]:

	x1	x2	x3
0	a	1.0	T
1	b	2.0	F
2	c	3.0	NaN
3	d	NaN	T

In [196]:
```
# filter dataframe using IsIn matching row.
df_1[df_1.x1.isin(df_2.x1)]
```

Out[196]:

	x1	x2
0	a	1
1	b	2

In [198]:
```
# not matching value in two dataframes.
df_1[~df_1.x1.isin(df_2.x1)]
```

Out[198]:

	x1	x2
2	c	3

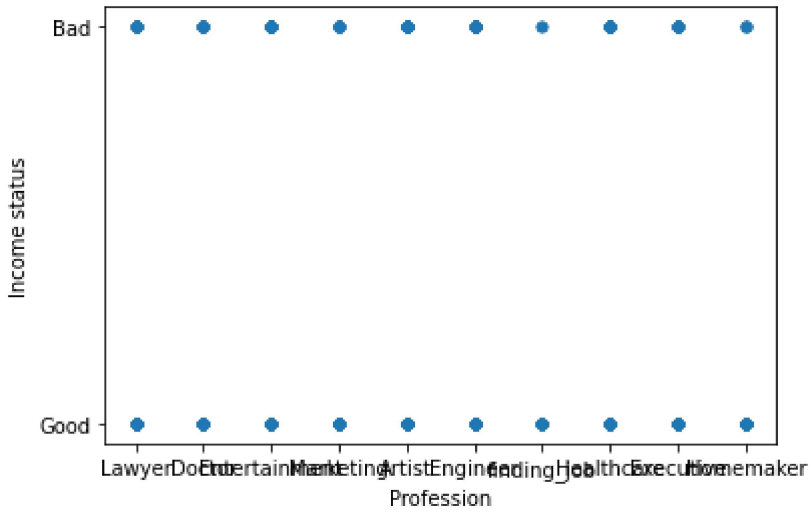
In [200]:
```
1 df_copy.head()
```

Out[200]:

	CustomerID	Gender	Age	Annual Income ($)	Profession	Work Experience	Member	Income status
443	444	Female	0	68761.0	Lawyer	1	4	Good
466	467	Male	0	186002.0	Doctor	15	2	Good
1271	1272	Female	0	61228.0	Entertainment	1	6	Bad
1248	1249	Female	0	143082.0	Entertainment	0	7	Good
1583	1584	Female	0	120899.0	Marketing	2	6	Good

In [206]:
```
1 # pandas plot graph
2 df_copy.plot.scatter(x='Profession',y='Income status')
```

Out[206]: <AxesSubplot:xlabel='Profession', ylabel='Income status'>



In [209]:
```
auto_data=pd.read_csv('D:/Data engineering/Python Practice/Automobile_dat
auto_data
```

Out[209]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	134
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	165
2	2	alfa-romero	hatchback	94.5	171.2	ohcv	six	154	19	165
3	3	audi	sedan	99.8	176.6	ohc	four	102	24	139
4	4	audi	sedan	99.4	176.6	ohc	five	115	18	174
...	...	...	...	...	...	...	...	...	...	
56	81	volkswagen	sedan	97.3	171.7	ohc	four	85	27	79
57	82	volkswagen	sedan	97.3	171.7	ohc	four	52	37	79
58	86	volkswagen	sedan	97.3	171.7	ohc	four	100	26	99
59	87	volvo	sedan	104.3	188.8	ohc	four	114	23	129
60	88	volvo	wagon	104.3	188.8	ohc	four	114	23	134

61 rows × 10 columns

In [212]:
```
# filter data
auto_data.isnull().sum()
```

Out[212]:
```
index              0
company            0
body-style         0
wheel-base         0
length             0
engine-type        0
num-of-cylinders   0
horsepower         0
average-mileage    0
price              3
dtype: int64
```

In [214]:
```
1 auto_data.dropna()
```

Out[214]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	134
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	165
2	2	alfa-romero	hatchback	94.5	171.2	ohcv	six	154	19	165
3	3	audi	sedan	99.8	176.6	ohc	four	102	24	139
4	4	audi	sedan	99.4	176.6	ohc	five	115	18	174
5	5	audi	sedan	99.8	177.3	ohc	five	110	19	152
6	6	audi	wagon	105.8	192.7	ohc	five	110	19	189
7	9	bmw	sedan	101.2	176.8	ohc	four	101	23	164
8	10	bmw	sedan	101.2	176.8	ohc	four	101	23	169
9	11	bmw	sedan	101.2	176.8	ohc	six	121	21	209
10	13	bmw	sedan	103.5	189.0	ohc	six	182	16	307
11	14	bmw	sedan	103.5	193.8	ohc	six	182	16	413
12	15	bmw	sedan	110.0	197.0	ohc	six	182	15	368
13	16	chevrolet	hatchback	88.4	141.1	l	three	48	47	51
14	17	chevrolet	hatchback	94.5	155.9	ohc	four	70	38	62
15	18	chevrolet	sedan	94.5	158.8	ohc	four	70	38	65
16	19	dodge	hatchback	93.7	157.3	ohc	four	68	31	63
17	20	dodge	hatchback	93.7	157.3	ohc	four	68	31	62
18	27	honda	wagon	96.5	157.1	ohc	four	76	30	72
19	28	honda	sedan	96.5	175.4	ohc	four	101	24	129
20	29	honda	sedan	96.5	169.1	ohc	four	100	25	103
21	30	isuzu	sedan	94.3	170.7	ohc	four	78	24	67
24	33	jaguar	sedan	113.0	199.6	dohc	six	176	15	322
25	34	jaguar	sedan	113.0	199.6	dohc	six	176	15	355
26	35	jaguar	sedan	102.0	191.7	ohcv	twelve	262	13	360
27	36	mazda	hatchback	93.1	159.1	ohc	four	68	30	51
28	37	mazda	hatchback	93.1	159.1	ohc	four	68	31	60
29	38	mazda	hatchback	93.1	159.1	ohc	four	68	31	67
30	39	mazda	hatchback	95.3	169.0	rotor	two	101	17	118
31	43	mazda	sedan	104.9	175.0	ohc	four	72	31	183
32	44	mercedes-benz	sedan	110.0	190.9	ohc	five	123	22	255
33	45	mercedes-benz	wagon	110.0	190.9	ohc	five	123	22	282
34	46	mercedes-benz	sedan	120.9	208.1	ohcv	eight	184	14	409

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	p
35	47	mercedes-benz	hardtop	112.0	199.2	ohcv	eight	184	14	454
36	49	mitsubishi	hatchback	93.7	157.3	ohc	four	68	37	53
37	50	mitsubishi	hatchback	93.7	157.3	ohc	four	68	31	61
38	51	mitsubishi	sedan	96.3	172.4	ohc	four	88	25	69
39	52	mitsubishi	sedan	96.3	172.4	ohc	four	88	25	81
40	53	nissan	sedan	94.5	165.3	ohc	four	55	45	70
41	54	nissan	sedan	94.5	165.3	ohc	four	69	31	66
42	55	nissan	sedan	94.5	165.3	ohc	four	69	31	68
43	56	nissan	wagon	94.5	170.2	ohc	four	69	31	73
44	57	nissan	sedan	100.4	184.6	ohcv	six	152	19	134
45	61	porsche	hardtop	89.5	168.9	ohcf	six	207	17	340
46	62	porsche	convertible	89.5	168.9	ohcf	six	207	17	370
48	66	toyota	hatchback	95.7	158.7	ohc	four	62	35	53
49	67	toyota	hatchback	95.7	158.7	ohc	four	62	31	63
50	68	toyota	hatchback	95.7	158.7	ohc	four	62	31	64
51	69	toyota	wagon	95.7	169.7	ohc	four	62	31	69
52	70	toyota	wagon	95.7	169.7	ohc	four	62	27	78
53	71	toyota	wagon	95.7	169.7	ohc	four	62	27	87
54	79	toyota	wagon	104.5	187.8	dohc	six	156	19	157
55	80	volkswagen	sedan	97.3	171.7	ohc	four	52	37	77
56	81	volkswagen	sedan	97.3	171.7	ohc	four	85	27	79
57	82	volkswagen	sedan	97.3	171.7	ohc	four	52	37	79
58	86	volkswagen	sedan	97.3	171.7	ohc	four	100	26	99
59	87	volvo	sedan	104.3	188.8	ohc	four	114	23	129
60	88	volvo	wagon	104.3	188.8	ohc	four	114	23	134

In [244]:
```
# most expensive car company name
exp_car=auto_data["price"].idxmax()
exp_car_brand=auto_data.loc[exp_car]
exp_car_brand
# mx_exp=df_copy[(df_copy["Profession"]=='Engineer')].max()
# mx_exp
# mx=df["Annual Income ($)"].idxmax()
# max_salary_record=df.loc[mx]
```

Out[244]:
```
index                     47
company             mercedes-benz
body-style                hardtop
wheel-base                  112.0
length                      199.2
engine-type                  ohcv
num-of-cylinders            eight
horsepower                    184
average-mileage                14
price                     45400.0
Name: 35, dtype: object
```

In [246]:
```
# print all toyta cars details
auto_data['company'].unique()
```

Out[246]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'mercedes-benz', 'mitsubishi',
       'nissan', 'porsche', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

In [249]:
```
auto_data[(auto_data['company']=='toyota')]
```

Out[249]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	pri
48	66	toyota	hatchback	95.7	158.7	ohc	four	62	35	5348
49	67	toyota	hatchback	95.7	158.7	ohc	four	62	31	6338
50	68	toyota	hatchback	95.7	158.7	ohc	four	62	31	6488
51	69	toyota	wagon	95.7	169.7	ohc	four	62	31	6918
52	70	toyota	wagon	95.7	169.7	ohc	four	62	27	7898
53	71	toyota	wagon	95.7	169.7	ohc	four	62	27	8778
54	79	toyota	wagon	104.5	187.8	dohc	six	156	19	15750

In [274]:
```
# count total cars per company
print(auto_data['company'].value_counts())
print("Total cars:",auto_data['company'].count())
```

```
toyota           7
bmw              6
mazda            5
nissan           5
audi             4
mercedes-benz    4
mitsubishi       4
volkswagen       4
alfa-romero      3
chevrolet        3
honda            3
isuzu            3
jaguar           3
porsche          3
dodge            2
volvo            2
Name: company, dtype: int64
Total cars: 61
```

In [278]:
```
# find each company heighest price car
auto_data.groupby('company')['price'].max()
```

Out[278]:
```
company
alfa-romero      16500.0
audi             18920.0
bmw              41315.0
chevrolet         6575.0
dodge             6377.0
honda            12945.0
isuzu             6785.0
jaguar           36000.0
mazda            18344.0
mercedes-benz    45400.0
mitsubishi        8189.0
nissan           13499.0
porsche          37028.0
toyota           15750.0
volkswagen        9995.0
volvo            13415.0
Name: price, dtype: float64
```

In [300]:
```
# find bmw company maximum car price
bmw_max_price =auto_data[auto_data['company'] == 'bmw']['price'].idxmax()

bmw_max_price_row=auto_data.loc[bmw_max_price]
bmw_max_price_row
```

Out[300]:
```
index                  14
company                bmw
body-style           sedan
wheel-base           103.5
length               193.8
engine-type            ohc
num-of-cylinders       six
horsepower             182
average-mileage         16
price                41315.0
Name: 11, dtype: object
```

In [305]:
```
max_price=auto_data['price'].max()
max_price
```

Out[305]:
```
45400.0
```

In [306]:
```
# maximum car price
max_price=auto_data['price'].idxmax()
max_price_detail=auto_data.loc[max_price]
max_price_detail
```

Out[306]:
```
index                      47
company            mercedes-benz
body-style               hardtop
wheel-base                 112.0
length                     199.2
engine-type                 ohcv
num-of-cylinders           eight
horsepower                   184
average-mileage               14
price                    45400.0
Name: 35, dtype: object
```

In [308]:
```
1 auto_data.head()
```

Out[308]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	pric
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	13495.
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	16500.
2	2	alfa-romero	hatchback	94.5	171.2	ohcv	six	154	19	16500.
3	3	audi	sedan	99.8	176.6	ohc	four	102	24	13950.
4	4	audi	sedan	99.4	176.6	ohc	five	115	18	17450.

In [325]:
```
1 # average milage of each car by company  Hint: by taking mean we calculat
2 avg=auto_data.groupby('company')['average-mileage'].mean()
3 avg_milage=pd.DataFrame(avg)
4 avg_milage
```

Out[325]:

company	average-mileage
alfa-romero	20.333333
audi	20.000000
bmw	19.000000
chevrolet	41.000000
dodge	31.000000
honda	26.333333
isuzu	33.333333
jaguar	14.333333
mazda	28.000000
mercedes-benz	18.000000
mitsubishi	29.500000
nissan	31.400000
porsche	17.000000
toyota	28.714286
volkswagen	31.750000
volvo	23.000000

In [346]:
```
#Sort all cars by Price column sort by ascending or descending.
auto_data.sort_values(by='price', ascending=False)
```

Out[346]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	pr
35	47	mercedes-benz	hardtop	112.0	199.2	ohcv	eight	184	14	4540
11	14	bmw	sedan	103.5	193.8	ohc	six	182	16	4131
34	46	mercedes-benz	sedan	120.9	208.1	ohcv	eight	184	14	4096
46	62	porsche	convertible	89.5	168.9	ohcf	six	207	17	3702
12	15	bmw	sedan	110.0	197.0	ohc	six	182	15	3688
...	...	...	...	...	...	...	...	...	...	
27	36	mazda	hatchback	93.1	159.1	ohc	four	68	30	519
13	16	chevrolet	hatchback	88.4	141.1	l	three	48	47	515
22	31	isuzu	sedan	94.5	155.9	ohc	four	70	38	N
23	32	isuzu	sedan	94.5	155.9	ohc	four	70	38	N
47	63	porsche	hatchback	98.4	175.7	dohcv	eight	288	17	N

61 rows × 10 columns

In [348]:
```
# sort value by descending
auto_data.sort_values(by='price',ascending=True)
```

Out[348]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	pr
13	16	chevrolet	hatchback	88.4	141.1	l	three	48	47	515
27	36	mazda	hatchback	93.1	159.1	ohc	four	68	30	519
48	66	toyota	hatchback	95.7	158.7	ohc	four	62	35	534
36	49	mitsubishi	hatchback	93.7	157.3	ohc	four	68	37	538
28	37	mazda	hatchback	93.1	159.1	ohc	four	68	31	609
...	...	...	...	...	...	...	...	...	...	
11	14	bmw	sedan	103.5	193.8	ohc	six	182	16	4131
35	47	mercedes-benz	hardtop	112.0	199.2	ohcv	eight	184	14	4540
22	31	isuzu	sedan	94.5	155.9	ohc	four	70	38	N
23	32	isuzu	sedan	94.5	155.9	ohc	four	70	38	N
47	63	porsche	hatchback	98.4	175.7	dohcv	eight	288	17	N

61 rows × 10 columns

In [359]:
```
GermanCars = pd.DataFrame({'Company': ['Ford', 'Mercedes', 'BMV', 'Audi']
japaneseCars = pd.DataFrame({'Company': ['Toyota', 'Honda', 'Nissan', 'Mi
GermanCars
```

Out[359]:

	Company	Price
0	Ford	23845
1	Mercedes	171995
2	BMV	135925
3	Audi	71400

In [360]:
```
japaneseCars
```

Out[360]:

	Company	Price
0	Toyota	29995
1	Honda	23600
2	Nissan	61500
3	Mitsubishi	58900

In [377]:
```
# conactinate two dataframe
pd.concat([GermanCars,japaneseCars],keys=['German','japan'])
```

Out[377]:

		Company	Price
German	**0**	Ford	23845
	1	Mercedes	171995
	2	BMV	135925
	3	Audi	71400
japan	**0**	Toyota	29995
	1	Honda	23600
	2	Nissan	61500
	3	Mitsubishi	58900

In [383]:
```
# Create two data frames using the following two Dicts, Merge two data fr
# second data frame as a new column to the first data frame.
Car_Price = pd.DataFrame({'Company': ['Toyota', 'Honda', 'BMV', 'Audi'],
car_Horsepower = pd.DataFrame({'Company': ['Toyota', 'Honda', 'BMV', 'Aud
car_Horsepower
```

Out[383]:

	Company	horsepower
0	Toyota	141
1	Honda	80
2	BMV	182
3	Audi	160

In [384]:
```
Car_Price
```

Out[384]:

	Company	Price
0	Toyota	23845
1	Honda	17995
2	BMV	135925
3	Audi	71400

In [385]:
```
1 pd.merge(Car_Price,car_Horsepower,how='left',on='Company')
```

Out[385]:

	Company	Price	horsepower
0	Toyota	23845	141
1	Honda	17995	80
2	BMV	135925	182
3	Audi	71400	160

In [ ]:
```
1
```