Filtering data

In [40]:
```
import pandas as pd
```

In [41]:
```
data_csv=pd.read_csv('diamonds.csv')
data_csv
```

Out[41]:

	carat	cut	color	clarity	depth	table	price
0	0.23	Ideal	E	SI2	61.5	55.0	326
1	0.21	Premium	E	SI1	59.8	61.0	326
2	0.23	Good	E	VS1	56.9	65.0	327
3	0.29	Premium	I	VS2	62.4	58.0	334
4	0.31	Good	J	SI2	63.3	58.0	335
...	...	...	...	...	...	...	...
53935	0.72	Ideal	D	SI1	60.8	57.0	2757
53936	0.72	Good	D	SI1	63.1	55.0	2757
53937	0.70	Very Good	D	SI1	62.8	60.0	2757
53938	0.86	Premium	H	SI2	61.0	58.0	2757
53939	0.75	Ideal	D	SI2	62.2	55.0	2757

53940 rows × 7 columns

Find all the rows with cut=Ideal

In [42]:
```
flt=data_csv['cut']=='Ideal'
data_csv[flt]
```

Out[42]:

	carat	cut	color	clarity	depth	table	price
0	0.23	Ideal	E	SI2	61.5	55.0	326
11	0.23	Ideal	J	VS1	62.8	56.0	340
13	0.31	Ideal	J	SI2	62.2	54.0	344
16	0.30	Ideal	I	SI2	62.0	54.0	348
39	0.33	Ideal	I	SI2	61.8	55.0	403
...	...	...	...	...	...	...	...
53925	0.79	Ideal	I	SI1	61.6	56.0	2756
53926	0.71	Ideal	E	SI1	61.9	56.0	2756
53929	0.71	Ideal	G	VS1	61.4	56.0	2756
53935	0.72	Ideal	D	SI1	60.8	57.0	2757
53939	0.75	Ideal	D	SI2	62.2	55.0	2757

21551 rows × 7 columns

Find all the rows with cut=Ideal and color=E

In [43]:
```
flt= (data_csv['cut']=='Ideal')& (data_csv['color']=='E')
data_csv[flt]
```

Out[43]:

	carat	cut	color	clarity	depth	table	price
0	0.23	Ideal	E	SI2	61.5	55.0	326
82	0.26	Ideal	E	VVS2	62.9	58.0	554
90	0.70	Ideal	E	SI1	62.5	57.0	2757
109	0.59	Ideal	E	VVS2	62.0	55.0	2761
111	0.74	Ideal	E	SI2	62.2	56.0	2761
...	...	...	...	...	...	...	...
53876	0.70	Ideal	E	SI1	61.7	55.0	2745
53878	0.51	Ideal	E	VVS1	61.9	54.0	2745
53891	0.56	Ideal	E	VVS1	62.1	56.0	2750
53915	0.77	Ideal	E	SI2	62.1	56.0	2753
53926	0.71	Ideal	E	SI1	61.9	56.0	2756

3903 rows × 7 columns

Filtering using loc

In [44]:
```
data_csv.loc[flt,'carat']
```

Out[44]:
```
0        0.23
82       0.26
90       0.70
109      0.59
111      0.74
         ...
53876    0.70
53878    0.51
53891    0.56
53915    0.77
53926    0.71
Name: carat, Length: 3903, dtype: float64
```

Filtering using *query* method

In [45]:
```
data_csv.query("cut=='Ideal'& color=='E'")
```

Out[45]:

	carat	cut	color	clarity	depth	table	price
0	0.23	Ideal	E	SI2	61.5	55.0	326
82	0.26	Ideal	E	VVS2	62.9	58.0	554
90	0.70	Ideal	E	SI1	62.5	57.0	2757
109	0.59	Ideal	E	VVS2	62.0	55.0	2761
111	0.74	Ideal	E	SI2	62.2	56.0	2761
...	...	...	...	...	...	...	...
53876	0.70	Ideal	E	SI1	61.7	55.0	2745
53878	0.51	Ideal	E	VVS1	61.9	54.0	2745
53891	0.56	Ideal	E	VVS1	62.1	56.0	2750
53915	0.77	Ideal	E	SI2	62.1	56.0	2753
53926	0.71	Ideal	E	SI1	61.9	56.0	2756

3903 rows × 7 columns

Filtering using *isin* method

In [46]:
```
cuts=['Ideal','Premium']
mask=data_csv['cut'].isin(cuts)
data_csv[mask]
```

Out[46]:

	carat	cut	color	clarity	depth	table	price
0	0.23	Ideal	E	SI2	61.5	55.0	326
1	0.21	Premium	E	SI1	59.8	61.0	326
3	0.29	Premium	I	VS2	62.4	58.0	334
11	0.23	Ideal	J	VS1	62.8	56.0	340
12	0.22	Premium	F	SI1	60.4	61.0	342
...	...	...	...	...	...	...	...
53931	0.71	Premium	F	SI1	59.8	62.0	2756
53934	0.72	Premium	D	SI1	62.7	59.0	2757
53935	0.72	Ideal	D	SI1	60.8	57.0	2757
53938	0.86	Premium	H	SI2	61.0	58.0	2757
53939	0.75	Ideal	D	SI2	62.2	55.0	2757

35342 rows × 7 columns

Filtering using *str* method

In [47]:
```
#str methods

mask=data_csv['cut'].str.contains('V')

data_csv[mask]
```

Out[47]:

	carat	cut	color	clarity	depth	table	price
5	0.24	Very Good	J	VVS2	62.8	57.0	336
6	0.24	Very Good	I	VVS1	62.3	57.0	336
7	0.26	Very Good	H	SI1	61.9	55.0	337
9	0.23	Very Good	H	VS1	59.4	61.0	338
19	0.30	Very Good	J	SI1	62.7	59.0	351
...	...	...	...	...	...	...	...
53921	0.70	Very Good	E	VS2	62.8	60.0	2755
53922	0.70	Very Good	D	VS1	63.1	59.0	2755
53932	0.70	Very Good	E	VS2	60.5	59.0	2757
53933	0.70	Very Good	E	VS2	61.2	59.0	2757
53937	0.70	Very Good	D	SI1	62.8	60.0	2757

Manipulating data

In [48]:
```
sales_data = pd.DataFrame({
"name":["William","Emma","Sofia","Markus","Edward","Thomas","Ethan","Olivia","Arun","Anika","P
,"region":["East","North","East","South","West","West","South","West","West","East","South"]
,"sales":[50000,52000,90000,34000,42000,72000,49000,55000,67000,65000,67000]
,"expenses":[42000,43000,50000,44000,38000,39000,42000,60000,39000,44000,45000]})
print(sales_data)
```

```
       name region  sales  expenses
0   William   East  50000     42000
1      Emma  North  52000     43000
2     Sofia   East  90000     50000
3    Markus  South  34000     44000
4    Edward   West  42000     38000
5    Thomas   West  72000     39000
6     Ethan  South  49000     42000
7    Olivia   West  55000     60000
8      Arun   West  67000     39000
9     Anika   East  65000     44000
10    Paulo  South  67000     45000
```

In [49]:
```
print(sales_data.columns)
```

```
Index(['name', 'region', 'sales', 'expenses'], dtype='object')
```

In [50]:
```
sales_data.rename(columns={'name': 'firstname'}, inplace=True)
print(sales_data.columns)
```

```
Index(['firstname', 'region', 'sales', 'expenses'], dtype='object')
```

In [51]:
```
sales_data.columns=[x.capitalize() for x in sales_data.columns]
print(sales_data.columns)
```

```
Index(['Firstname', 'Region', 'Sales', 'Expenses'], dtype='object')
```

Adding a new column

In [52]:
```
# using a list
discount=[10,20,12,32,10,15,25,15,10,20,5]
sales_data['Discount']=discount
print(sales_data)
```

```
   Firstname Region  Sales  Expenses  Discount
0    William   East  50000     42000        10
1       Emma  North  52000     43000        20
2      Sofia   East  90000     50000        12
3     Markus  South  34000     44000        32
4     Edward   West  42000     38000        10
5     Thomas   West  72000     39000        15
6      Ethan  South  49000     42000        25
7     Olivia   West  55000     60000        15
8       Arun   West  67000     39000        10
9      Anika   East  65000     44000        20
10     Paulo  South  67000     45000         5
```

In [53]:
```
# using another column (or columns) of the dataframe
sales_data['Benefit']=sales_data['Sales']-sales_data['Expenses']
print(sales_data)
```

```
   Firstname Region  Sales  Expenses  Discount  Benefit
0    William   East  50000     42000        10     8000
1       Emma  North  52000     43000        20     9000
2      Sofia   East  90000     50000        12    40000
3     Markus  South  34000     44000        32   -10000
4     Edward   West  42000     38000        10     4000
5     Thomas   West  72000     39000        15    33000
6      Ethan  South  49000     42000        25     7000
7     Olivia   West  55000     60000        15    -5000
8       Arun   West  67000     39000        10    28000
9      Anika   East  65000     44000        20    21000
10     Paulo  South  67000     45000         5    22000
```

Removing a column

In [54]:
```
sales_data.drop('Discount', axis=1,inplace=True)    # axis=1 for ropping a column
print(sales_data)
```

```
   Firstname Region  Sales  Expenses  Benefit
0    William   East  50000     42000     8000
1       Emma  North  52000     43000     9000
2      Sofia   East  90000     50000    40000
3     Markus  South  34000     44000   -10000
4     Edward   West  42000     38000     4000
5     Thomas   West  72000     39000    33000
6      Ethan  South  49000     42000     7000
7     Olivia   West  55000     60000    -5000
8       Arun   West  67000     39000    28000
9      Anika   East  65000     44000    21000
10     Paulo  South  67000     45000    22000
```

Updating rows values

In [55]:
```
sales_data.loc[2,['Sales','Expenses']]= [55000,10000]
print(sales_data)
```

```
   Firstname Region  Sales  Expenses  Benefit
0    William   East  50000     42000     8000
1       Emma  North  52000     43000     9000
2      Sofia   East  55000     10000    40000
3     Markus  South  34000     44000   -10000
4     Edward   West  42000     38000     4000
5     Thomas   West  72000     39000    33000
6      Ethan  South  49000     42000     7000
7     Olivia   West  55000     60000    -5000
8       Arun   West  67000     39000    28000
9      Anika   East  65000     44000    21000
10     Paulo  South  67000     45000    22000
```

In [56]:
```
sales_data.loc[2]=['Anna','East',15000,10000,50000]   #updating the third row
print(sales_data)
```

```
   Firstname Region  Sales  Expenses  Benefit
0    William   East  50000     42000     8000
1       Emma  North  52000     43000     9000
2       Anna   East  15000     10000    50000
3     Markus  South  34000     44000   -10000
4     Edward   West  42000     38000     4000
5     Thomas   West  72000     39000    33000
6      Ethan  South  49000     42000     7000
7     Olivia   West  55000     60000    -5000
8       Arun   West  67000     39000    28000
9      Anika   East  65000     44000    21000
10     Paulo  South  67000     45000    22000
```

Updating Rows and Columns Based On Condition

In [57]:
```
filt = (sales_data['Sales'] > 65000)
sales_data.loc[filt,'Benefit'] = sales_data['Benefit']*1.5
# sales_data[filt]['Benefit'] = sales_data['Benefit']*1.5     -----   is not the correct syntax
print(sales_data)
```

```
   Firstname Region  Sales  Expenses  Benefit
0    William   East  50000     42000   8000.0
1       Emma  North  52000     43000   9000.0
2       Anna   East  15000     10000  50000.0
3     Markus  South  34000     44000 -10000.0
4     Edward   West  42000     38000   4000.0
5     Thomas   West  72000     39000  49500.0
6      Ethan  South  49000     42000   7000.0
7     Olivia   West  55000     60000  -5000.0
8       Arun   West  67000     39000  42000.0
9      Anika   East  65000     44000  21000.0
10     Paulo  South  67000     45000  33000.0
```

Updating using *apply* method

Example 1 - applying len() function

In [59]:
```
sales_data['Namelen']=sales_data['Firstname'].apply(len)
print(sales_data)
```

```
   Firstname Region  Sales  Expenses  Benefit  Namelen
0    William   East  50000     42000   8000.0        7
1       Emma  North  52000     43000   9000.0        4
2       Anna   East  15000     10000  50000.0        4
3     Markus  South  34000     44000 -10000.0        6
4     Edward   West  42000     38000   4000.0        6
5     Thomas   West  72000     39000  49500.0        6
6      Ethan  South  49000     42000   7000.0        5
7     Olivia   West  55000     60000  -5000.0        6
8       Arun   West  67000     39000  42000.0        4
9      Anika   East  65000     44000  21000.0        5
10     Paulo  South  67000     45000  33000.0        5
```

Example 1 - applying custom function

In [61]:
```
def rate(x):
    if x> 55000:
        return 'good'
    else:
        return "bad"

sales_data['Rate']=sales_data['Sales'].apply(rate)
print(sales_data)
```

```
   Firstname Region  Sales  Expenses  Benefit  Namelen  Rate
0    William   East  50000     42000   8000.0        7   bad
1       Emma  North  52000     43000   9000.0        4   bad
2       Anna   East  15000     10000  50000.0        4   bad
3     Markus  South  34000     44000 -10000.0        6   bad
4     Edward   West  42000     38000   4000.0        6   bad
5     Thomas   West  72000     39000  49500.0        6  good
6      Ethan  South  49000     42000   7000.0        5   bad
7     Olivia   West  55000     60000  -5000.0        6   bad
8       Arun   West  67000     39000  42000.0        4  good
9      Anika   East  65000     44000  21000.0        5  good
10     Paulo  South  67000     45000  33000.0        5  good
```