Sales Prediction Using Python

****importing the libraries

In [2]:

```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
```

****loading the dataset

In [3]:

```
df=pd.read_csv('E:\Advertising.csv',encoding='latin1')
df
```

Out[3]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

200 rows × 5 columns

In [4]:

```
df.head()
```

Out[4]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

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

Out[5]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

In [6]:
```
df.describe()
```

Out[6]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

In [7]:
```
df.describe
```

Out[7]:
```
<bound method NDFrame.describe of      Unnamed: 0     TV  Radio  Newspaper
Sales
0             1  230.1   37.8       69.2   22.1
1             2   44.5   39.3       45.1   10.4
2             3   17.2   45.9       69.3    9.3
3             4  151.5   41.3       58.5   18.5
4             5  180.8   10.8       58.4   12.9
..          ...    ...    ...        ...    ...
195         196   38.2    3.7       13.8    7.6
196         197   94.2    4.9        8.1    9.7
197         198  177.0    9.3        6.4   12.8
198         199  283.6   42.0       66.2   25.5
199         200  232.1    8.6        8.7   13.4

[200 rows x 5 columns]>
```

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  200 non-null    int64  
 1   TV          200 non-null    float64
 2   Radio       200 non-null    float64
 3   Newspaper   200 non-null    float64
 4   Sales       200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

In [9]:

```
df.columns
```

Out[9]:

```
Index(['Unnamed: 0', 'TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [10]:

```
df.isnull().sum()
```

Out[10]:

```
Unnamed: 0    0
TV            0
Radio         0
Newspaper     0
Sales         0
dtype: int64
```

In [11]:

```
df.drop_duplicates()
```

Out[11]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

200 rows × 5 columns

In [12]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [13]:
```
df
```

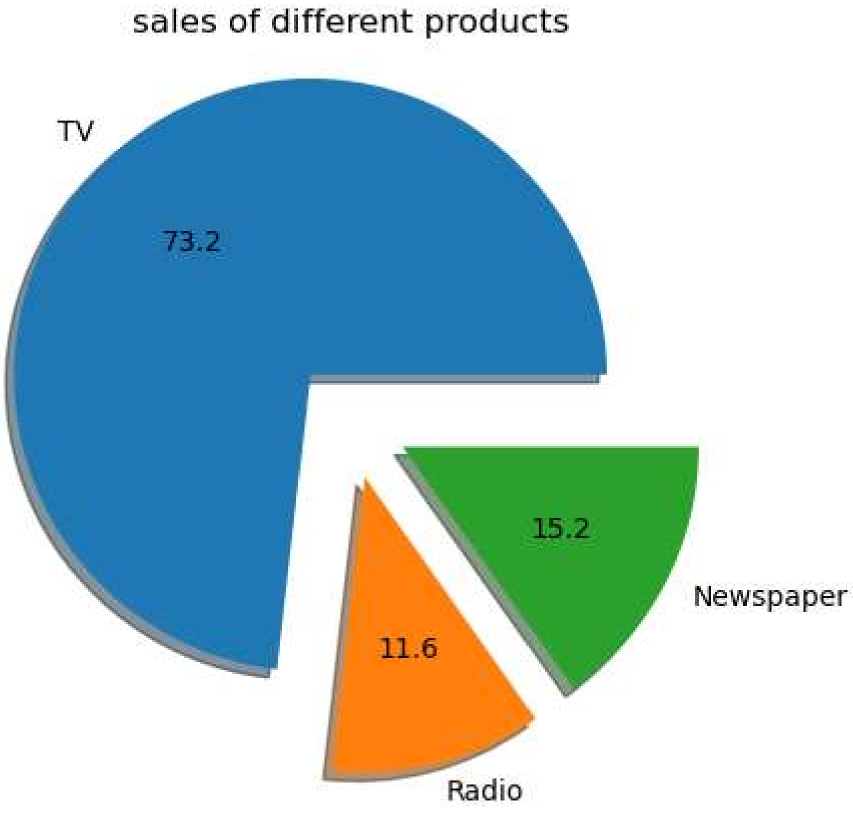
Out[13]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows × 4 columns

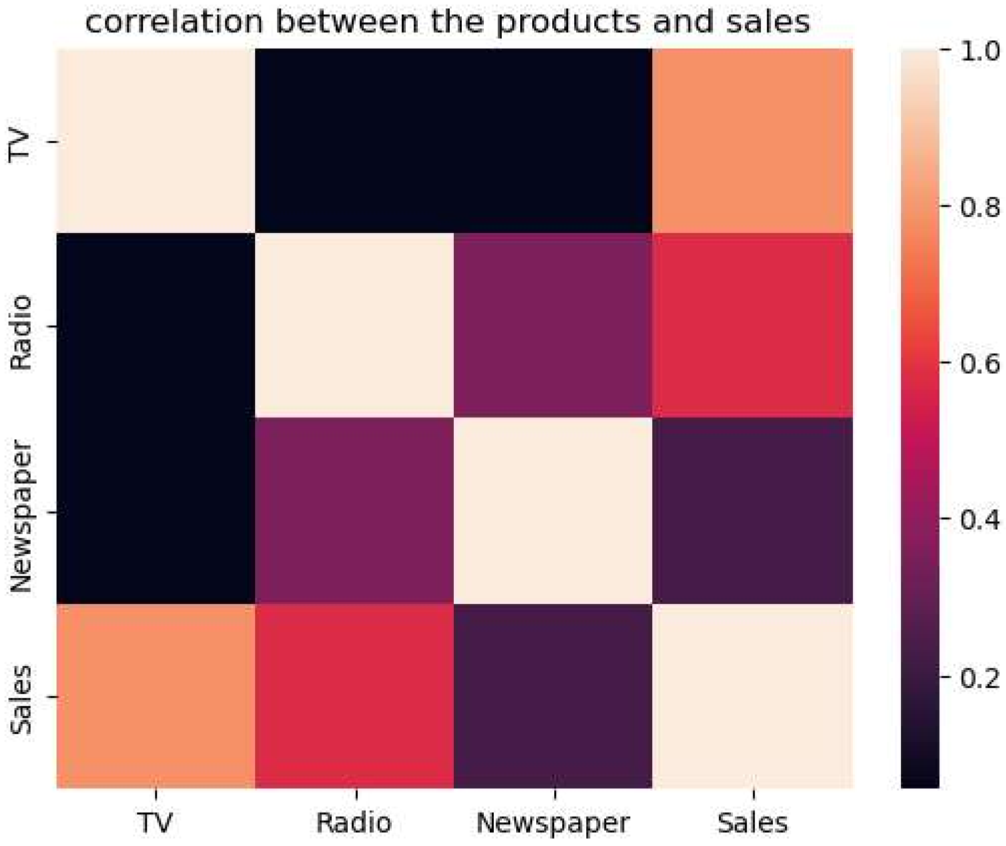
In [14]:
```
l=['TV','Radio','Newspaper']
i1=df['TV'].mean()
i2=df['Radio'].mean()
i3=df['Newspaper'].mean()
d=[i1,i2,i3]
e=[0.2,0.2,0.2]


plt.pie(d,labels=l,autopct='%1.1f',explode=e,shadow=True)
plt.title('sales of different products')
plt.show()
```



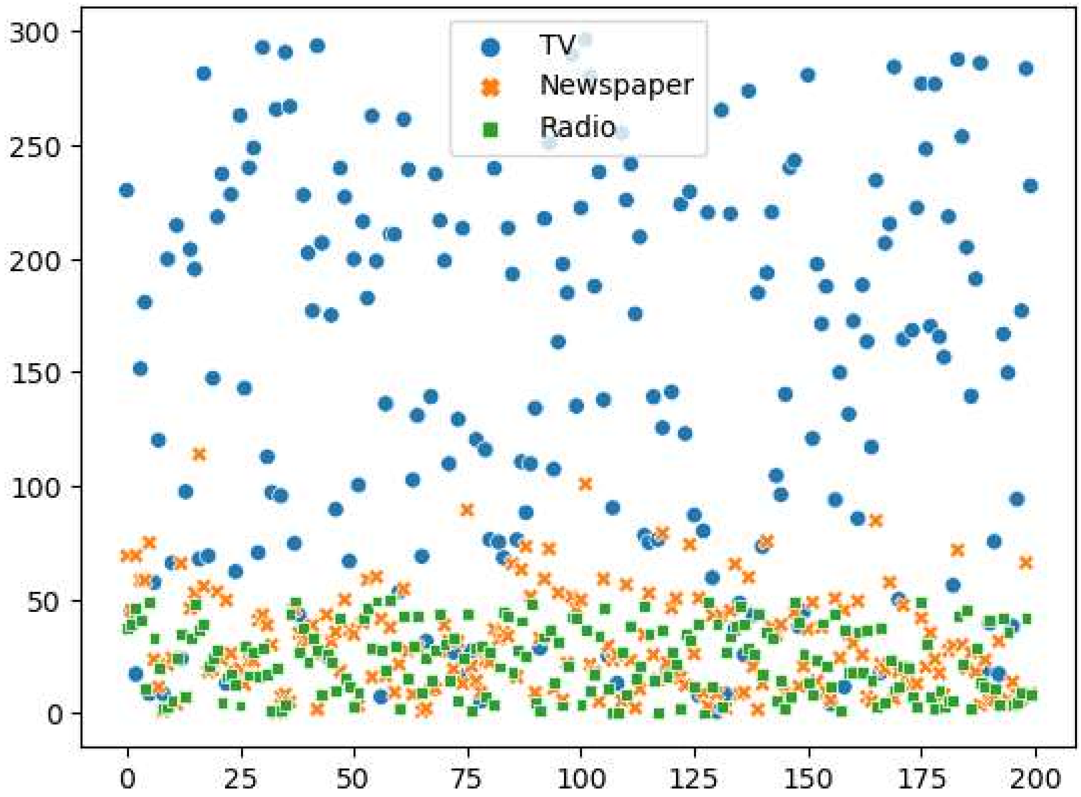
In [15]:
```
sn.heatmap(df.corr())
plt.title("correlation between the products and sales ")
```

Out[15]:
```
Text(0.5, 1.0, 'correlation between the products and sales ')
```



In [16]:
```
sn.scatterplot(data=[df['TV'],df['Newspaper'],df['Radio']])
#sn.scatterplot(data=df['Radio'])
#n.scatterplot(data=df['Newspaper'])
#snscatterplot(data=df['Sales'])
```

Out[16]:
```
<Axes: >
```



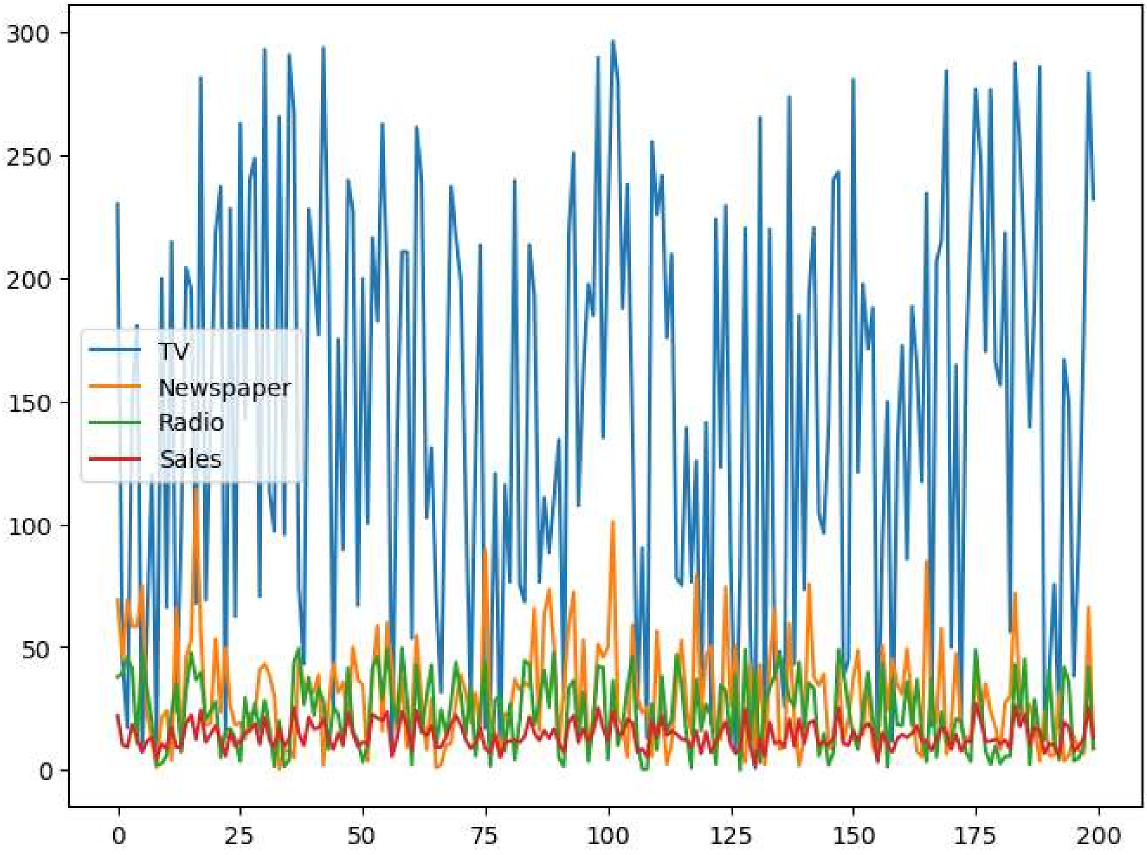
In [43]:

```
plt.figure(figsize=(8,6))
plt.plot(df['TV'],label='TV')
plt.plot(df['Newspaper'],label='Newspaper')
plt.plot(df['Radio'],label='Radio')
plt.plot(df['Sales'],label='Sales')

plt.legend()
```

Out[43]:

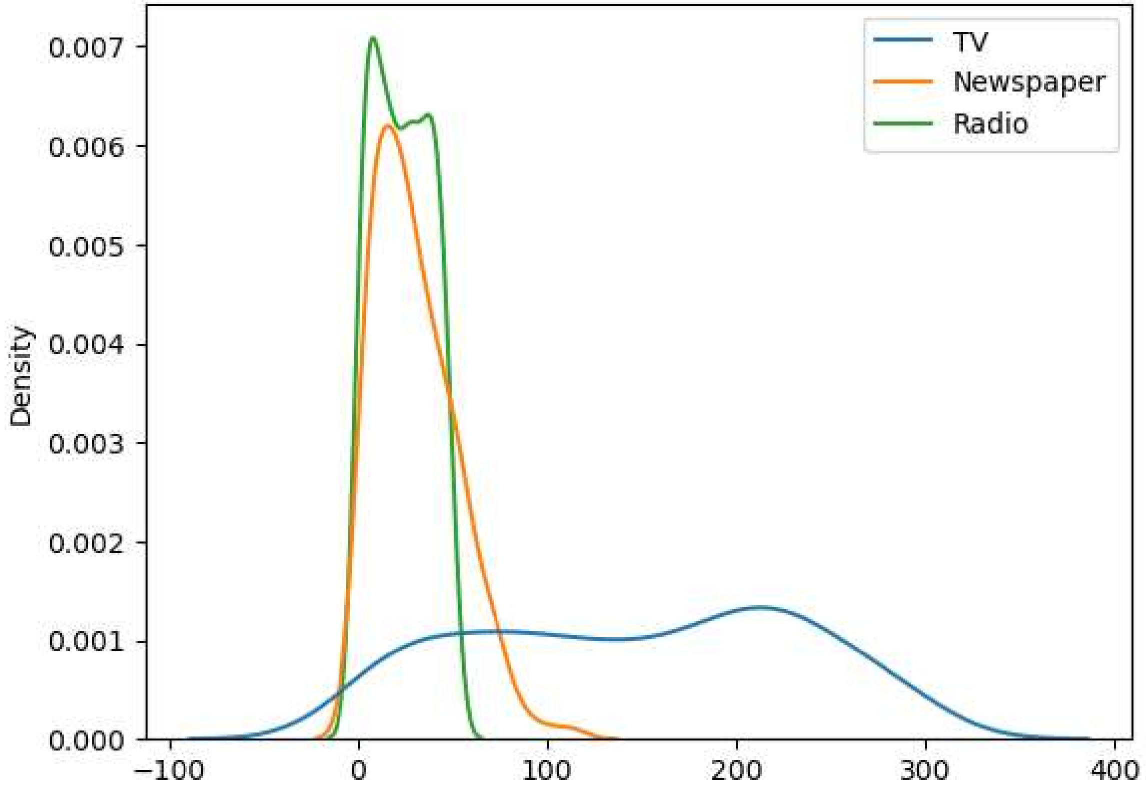
```
<matplotlib.legend.Legend at 0x1bcc5d06b90>
```



In [31]:

```
sn.kdeplot(data=[df['TV'],df['Newspaper'],df['Radio']])
plt.figsize=(10,100)

#sn.scatterplot(data=df['Radio'])
```



In [51]:

```
a=df.iloc[:,:3]
```

In [52]:

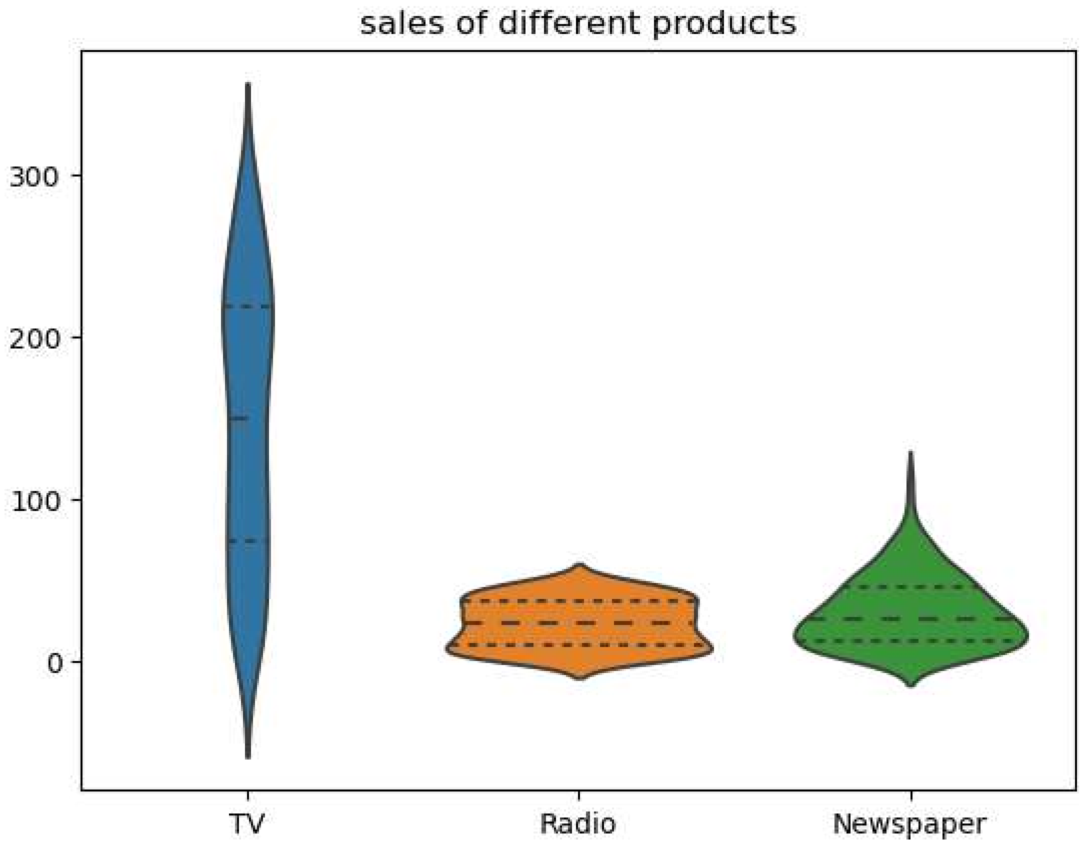
```
print(a)
```

```
        TV  Radio  Newspaper
0    230.1   37.8       69.2
1     44.5   39.3       45.1
2     17.2   45.9       69.3
3    151.5   41.3       58.5
4    180.8   10.8       58.4
..     ...    ...        ...
195   38.2    3.7       13.8
196   94.2    4.9        8.1
197  177.0    9.3        6.4
198  283.6   42.0       66.2
199  232.1    8.6        8.7

[200 rows x 3 columns]
```

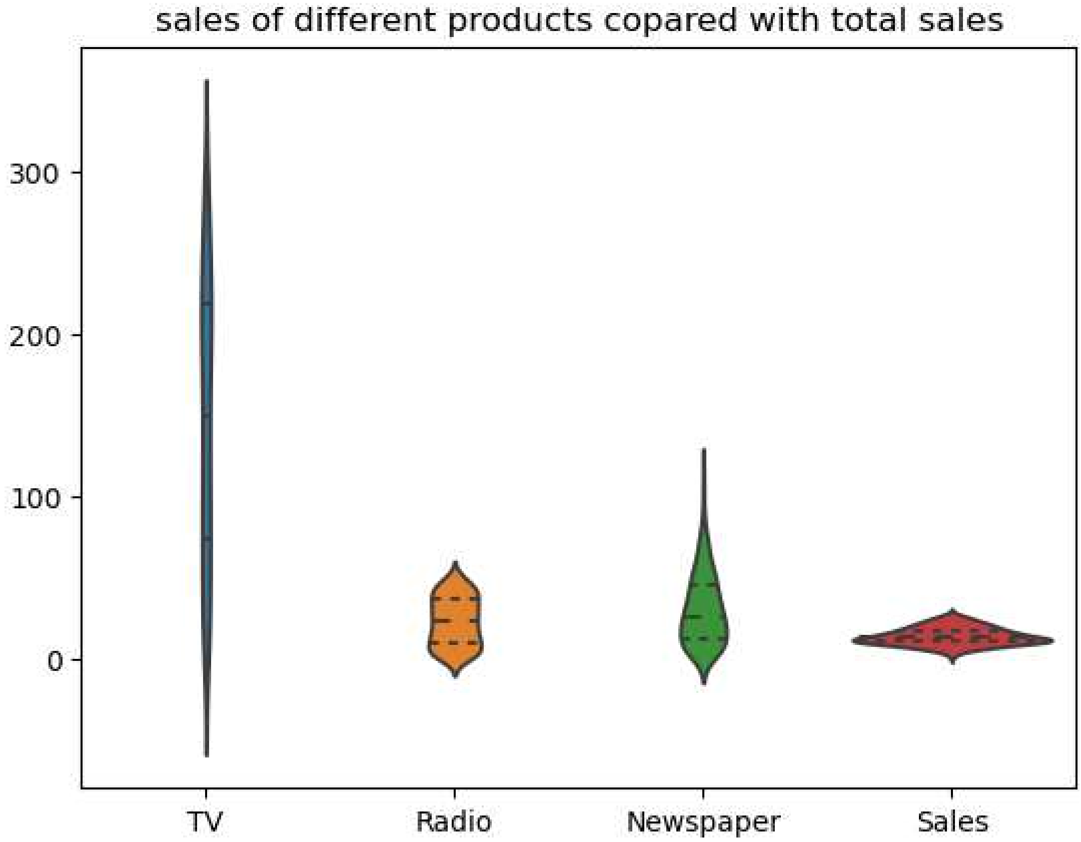
In [70]:

```
sn.violinplot(a,inner='quartile')
plt.title('sales of different products')
plt.show()
```



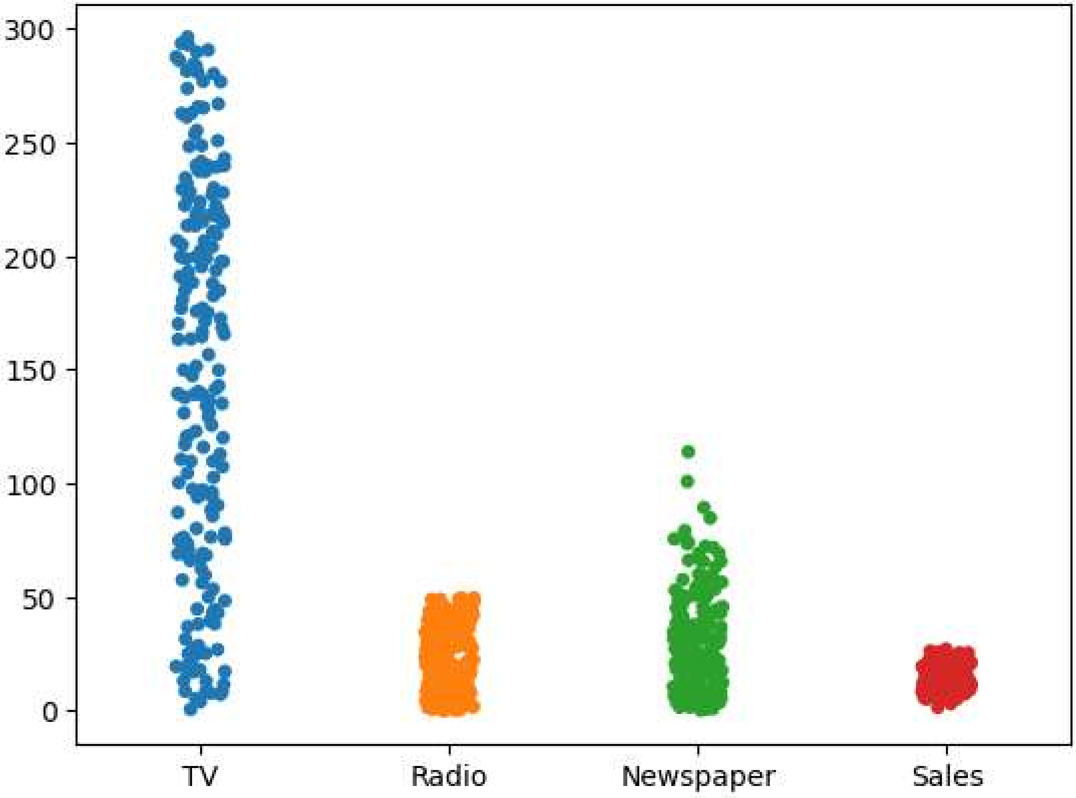
```
In [79]: sn.violinplot(df,inner='quartile')

         plt.title('sales of different products copared with total sales')
         plt.show()
```



In [72]:
```
sn.stripplot(df)
```

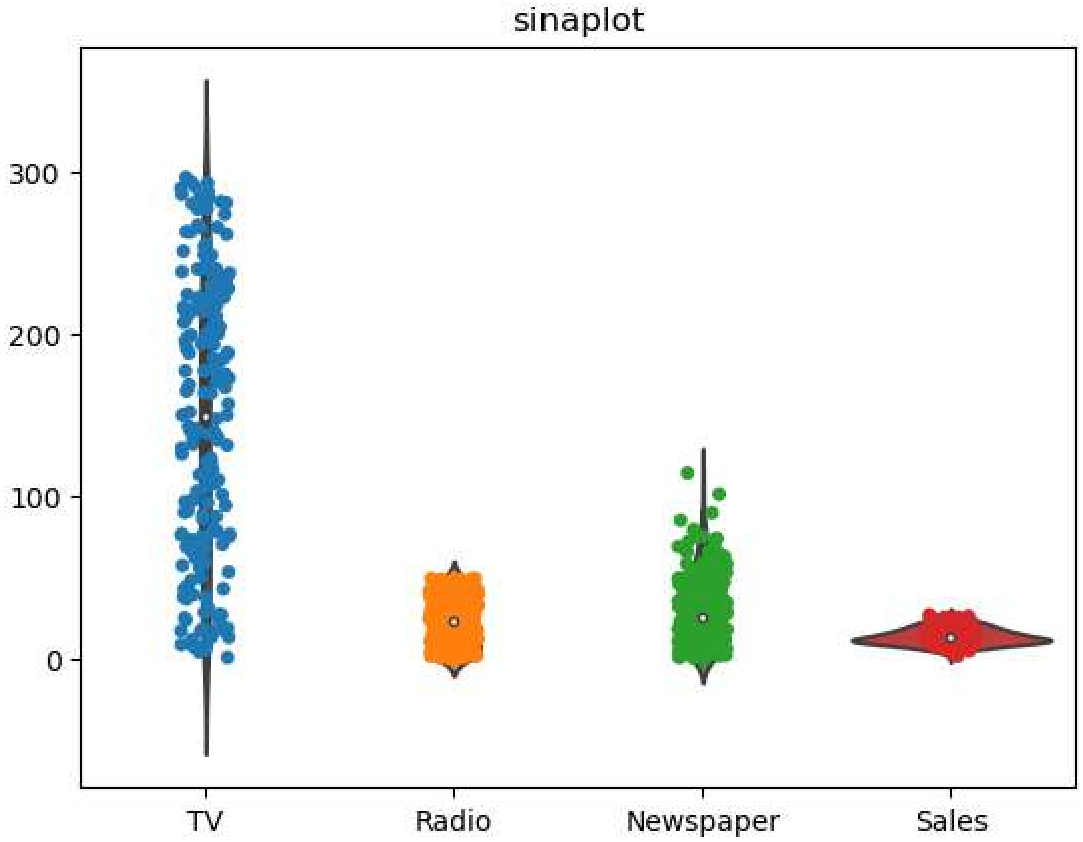
Out[72]:
```
<Axes: >
```



****combination of the both strip plot and violinplot provides a better visualization known as

> "sinaplot"

```
In [76]: sn.stripplot(df)
         sn.violinplot(df)
         plt.title("sinaplot")
         plt.figure(figsize=(10, 8))
         plt.show()
```

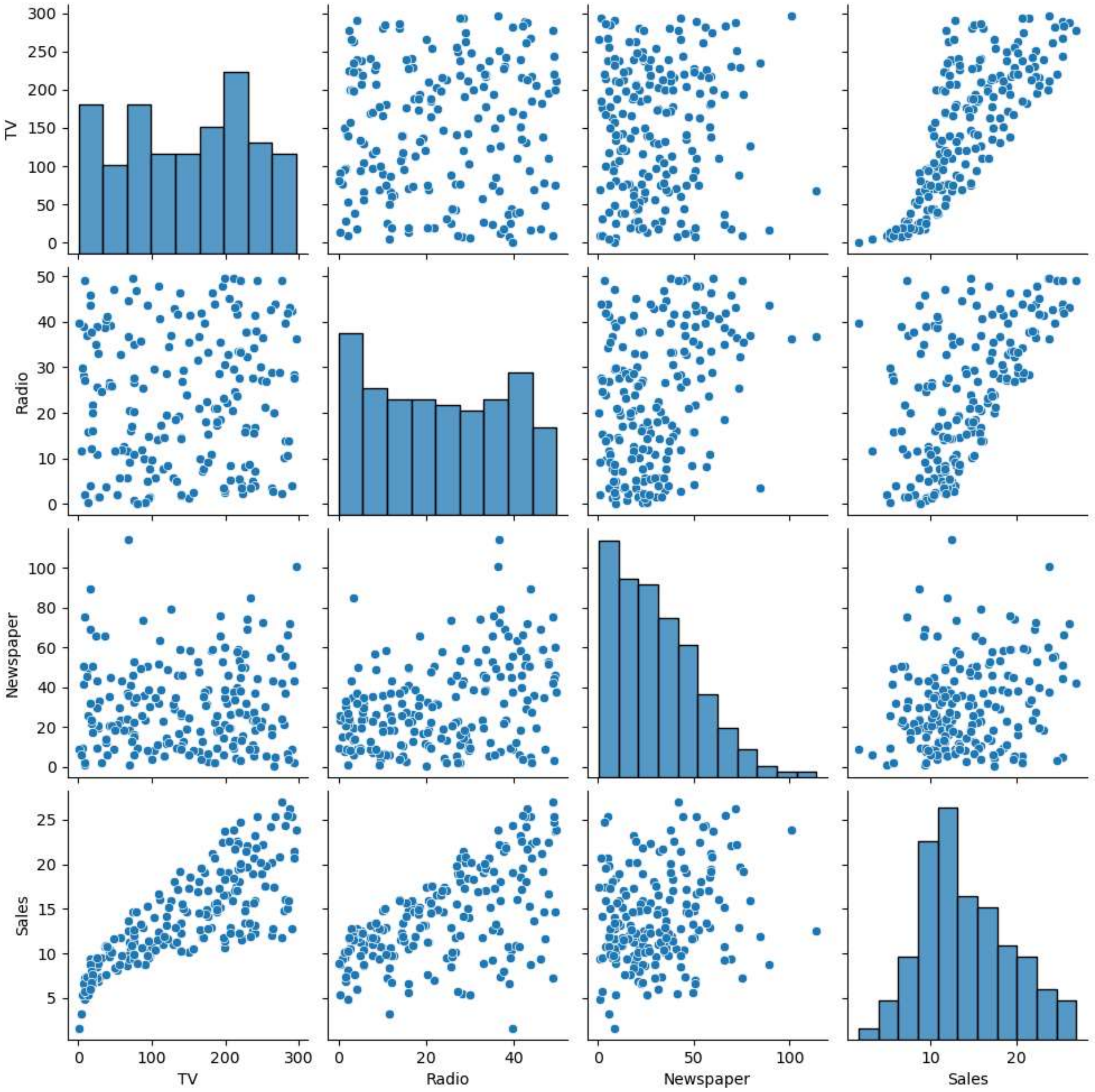


```
<Figure size 1000x800 with 0 Axes>
```

***let's see other insights in data

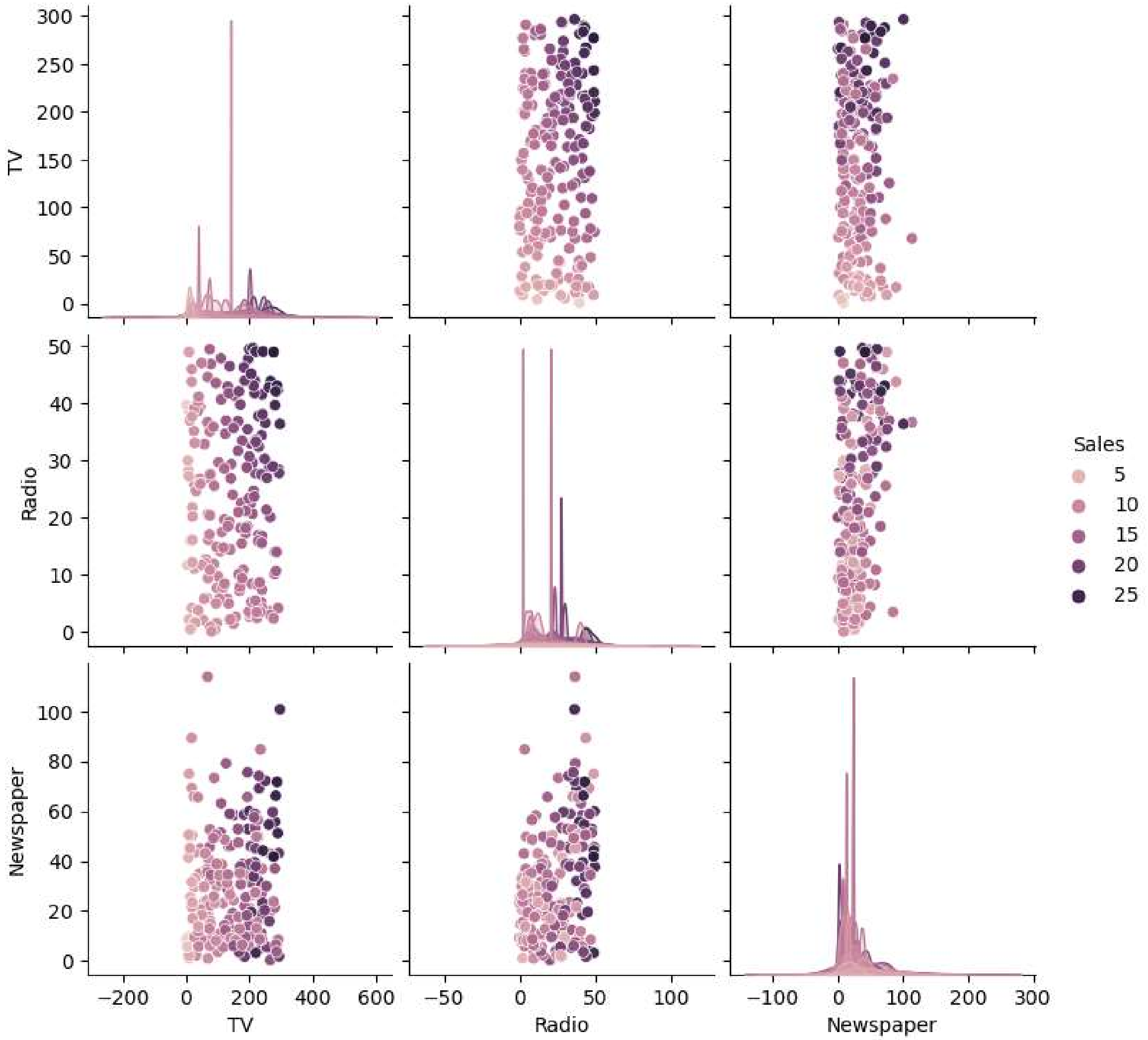
In [91]:

```
sn.pairplot(df)
plt.show()
```



```
In [92]: sn.pairplot(df,hue='Sales')
```

```
Out[92]: <seaborn.axisgrid.PairGrid at 0x1bcd561eda0>
```



```
In [93]: from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error, r2_score
```

****inputvariabls:Tv,Radio,Newspaper ****target variable:sales

In [96]:
```
x1=df.iloc[:,:3]
x1
```

Out[96]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4
...	...	...	...
195	38.2	3.7	13.8
196	94.2	4.9	8.1
197	177.0	9.3	6.4
198	283.6	42.0	66.2
199	232.1	8.6	8.7

200 rows × 3 columns

In [104]:
```
y1=df['Sales']
y1
```

Out[104]:
```
0      22.1
1      10.4
2       9.3
3      18.5
4      12.9
       ... 
195     7.6
196     9.7
197    12.8
198    25.5
199    13.4
Name: Sales, Length: 200, dtype: float64
```

In [105]:
```
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1)
```

In [107]:
```
x1_train
```

Out[107]:

	TV	Radio	Newspaper
124	229.5	32.3	74.2
21	237.4	5.1	23.5
68	237.4	27.5	11.0
141	193.7	35.4	75.6
190	39.5	41.1	5.8
...	...	...	...
187	191.1	28.7	18.2
155	4.1	11.6	5.7
193	166.8	42.0	3.6
101	296.4	36.3	100.9
131	265.2	2.9	43.0

150 rows × 3 columns

In [108]:
```
y1_train
```

Out[108]:
```
124    19.7
21     12.5
68     18.9
141    19.2
190    10.8
       ... 
187    17.3
155     3.2
193    19.6
101    23.8
131    12.7
Name: Sales, Length: 150, dtype: float64
```

In [109]:

```
x1_test
```

Out[109]:

	TV	Radio	Newspaper
109	255.4	26.9	5.5
94	107.4	14.0	10.9
93	250.9	36.5	72.3
110	225.8	8.2	56.5
127	80.2	0.0	9.2
130	0.7	39.6	8.7
42	293.6	27.7	1.8
164	117.2	14.7	5.4
23	228.3	16.9	26.2
181	218.5	5.4	27.4
150	280.7	13.9	37.0
134	36.9	38.6	65.6
25	262.9	3.5	19.5
171	164.5	20.9	47.4
33	265.6	20.0	0.3
180	156.6	2.6	8.3
18	69.2	20.5	18.3
71	109.8	14.3	31.7
188	286.0	13.9	3.7
198	283.6	42.0	66.2
96	197.6	3.5	5.9
104	238.2	34.3	5.3
16	67.8	36.6	114.0
32	97.2	1.5	30.0
35	290.7	4.1	8.5
20	218.4	27.7	53.4
142	220.5	33.2	37.9
30	292.9	28.3	43.2
76	27.5	1.6	20.7
87	110.7	40.6	63.2
107	90.4	0.3	23.2
52	216.4	41.7	39.6
160	172.5	18.1	30.7
12	23.8	35.1	65.9
14	204.1	32.9	46.0
145	140.3	1.9	9.0

	TV	Radio	Newspaper
38	43.1	26.7	35.1
61	261.3	42.7	54.7
121	18.8	21.7	50.4
27	240.1	16.7	22.9
81	239.8	4.1	36.9
57	136.2	19.2	16.6
51	100.4	9.6	3.6
103	187.9	17.2	17.9
126	7.8	38.9	50.6
10	66.1	5.8	24.2
174	222.4	3.4	13.1
64	131.1	42.8	28.9
85	193.2	18.4	65.7
50	199.8	3.1	34.6

In [110]:

```
y1_test
```

Out[110]:

```
109    19.8
94     11.5
93     22.2
110    13.4
127     8.8
130     1.6
42     20.7
164    11.9
23     15.5
181    12.2
150    16.1
134    10.8
25     12.0
171    14.5
33     17.4
180    10.5
18     11.3
71     12.4
188    15.9
198    25.5
96     11.7
104    20.7
16     12.5
32      9.6
35     12.8
20     18.0
142    20.1
30     21.4
76      6.9
87     16.0
107     8.7
52     22.6
160    14.4
12      9.2
14     19.0
145    10.3
38     10.1
61     24.2
121     7.0
27     15.9
81     12.3
57     13.2
51     10.7
103    14.7
126     6.6
10      8.6
174    11.5
64     18.0
85     15.2
50     11.4
Name: Sales, dtype: float64
```

In [114]:

```
model=LinearRegression()
print(model)
```

```
LinearRegression()
```

In [131]:

```
x1_train=x1_train.astype(int)
y1_train=y1_train.astype(int)
x1_test=x1_test.astype(int)
y1_test=y1_test.astype(int)
```

In [138]:

```
scale=StandardScaler()
x1_train_scaled=scale.fit_transform(x1_train)
x1_test_scaled=scale.fit_transform(x1_test)
```

In [140]:

```
model.fit(x1_train_scaled,y1_train)
```

Out[140]:

```
▾ LinearRegression
LinearRegression()
```

In [141]:

```
y1_predi=model.predict(x1_test_scaled)
```

In [144]:

```
print(y1)
```

```
0      22.1
1      10.4
2       9.3
3      18.5
4      12.9
       ... 
195     7.6
196     9.7
197    12.8
198    25.5
199    13.4
Name: Sales, Length: 200, dtype: float64
```

In [145]:

```
print("Accuracy of the data is:",r2_score(y1_test,y1_predi)*100)
```

```
Accuracy of the data is: 83.67322142617478
```

we got the the taccuracy of above 83 %

hence task is completed

In [ ]: