Import Important Library

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Import DataSet

In [7]:
```
df = pd.read_csv(r"C:\Users\meanu\Downloads\advertising.csv - advertising.csv.csv")
```

In [8]:
```
df.head()
```

Out[8]:

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

Data Understanding and Data Cleaning:-

In [9]:
```
df.shape
```

Out[9]:
```
(200, 4)
```

Find Information

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

Finding Datatype

In [11]:
```
df.dtypes
```

Out[11]:
```
TV           float64
Radio        float64
Newspaper    float64
Sales        float64
dtype: object
```

Checking Duplicates Values Available Or Not:-

In [17]:
```
df.duplicated().sum()
```

Out[17]:
```
0
```

Checking Null Values Available or Not :-

In [18]:
```
df.isna().sum()
```

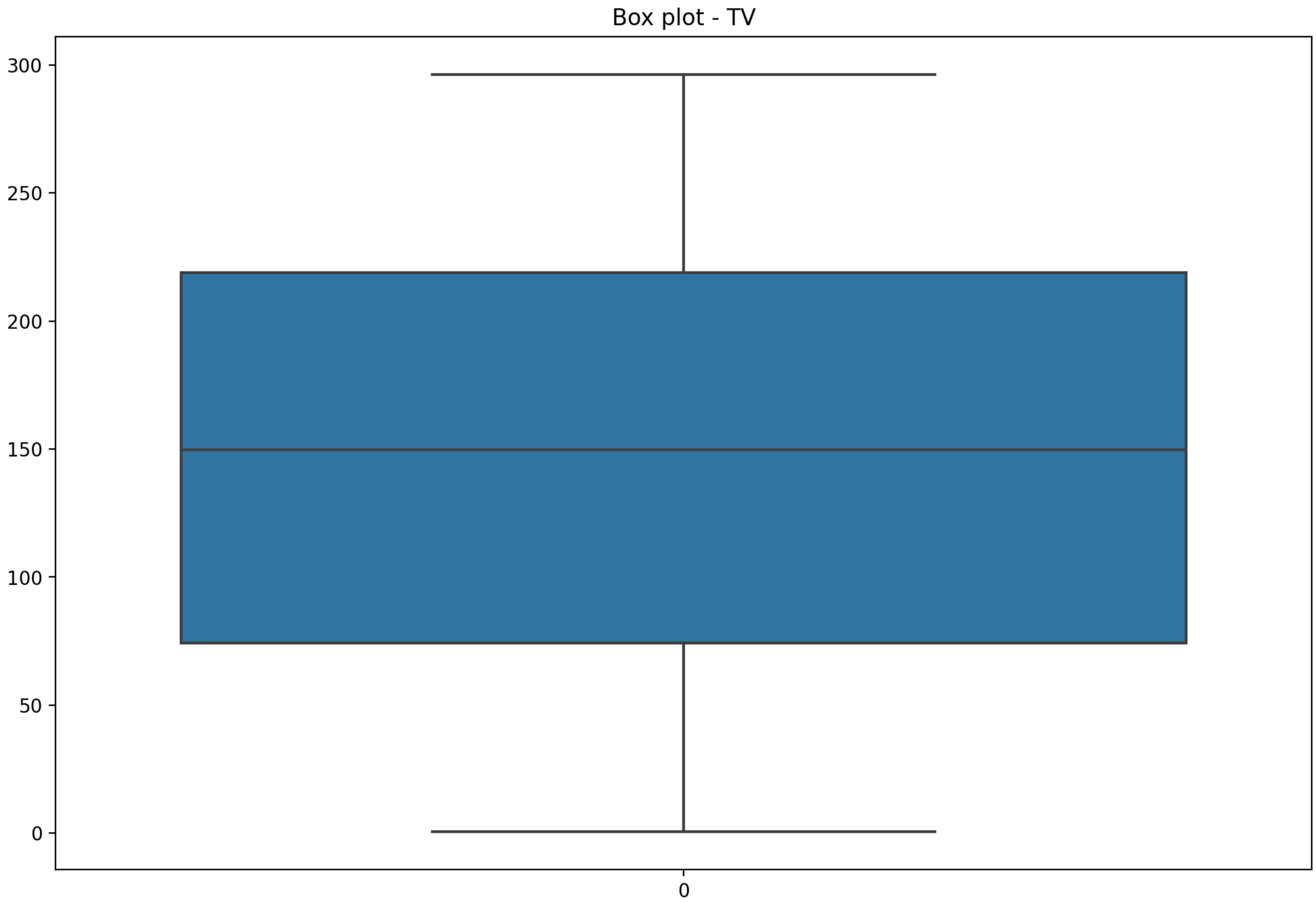
Out[18]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

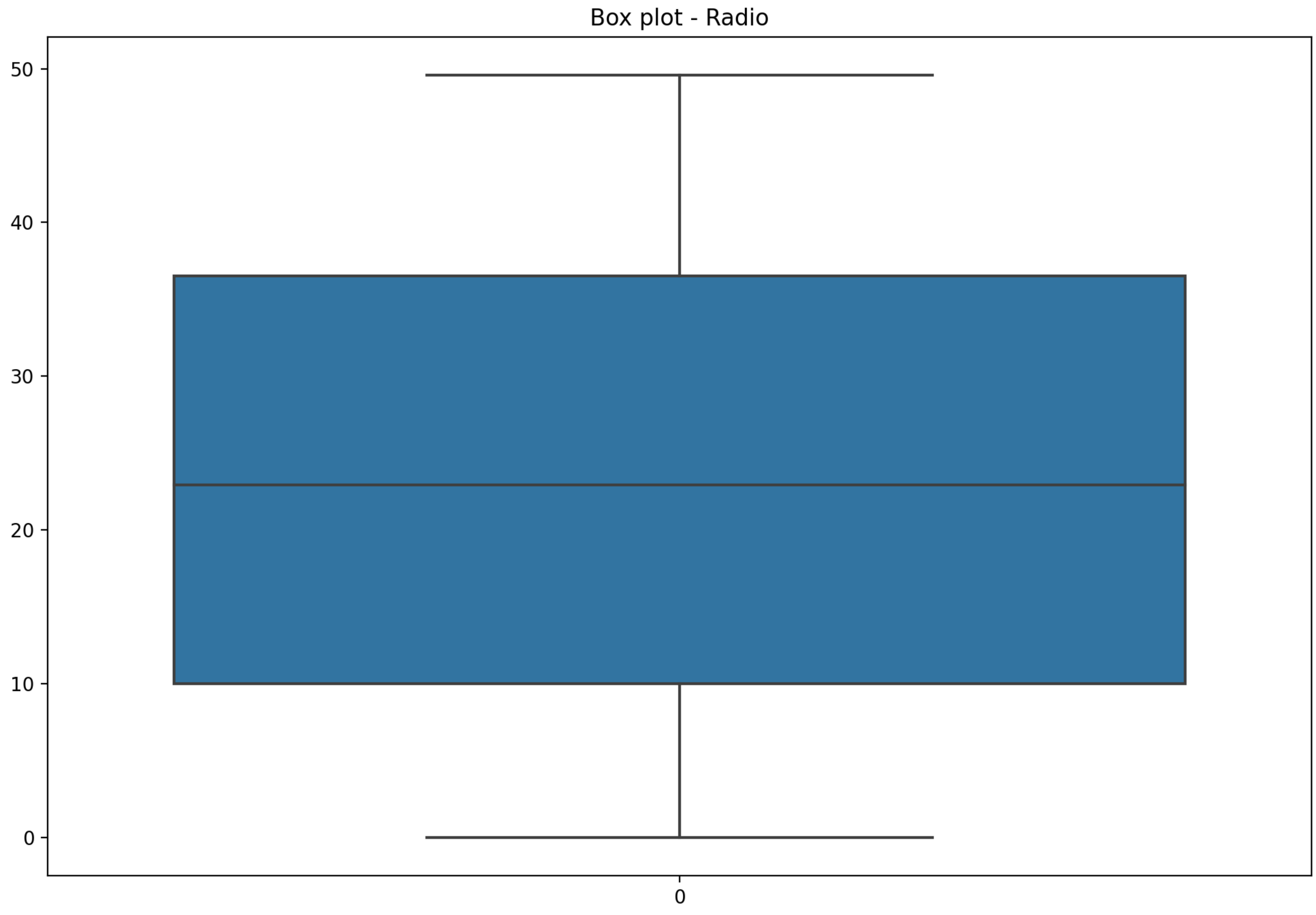
Checking And Showing Outliers In All Column:-

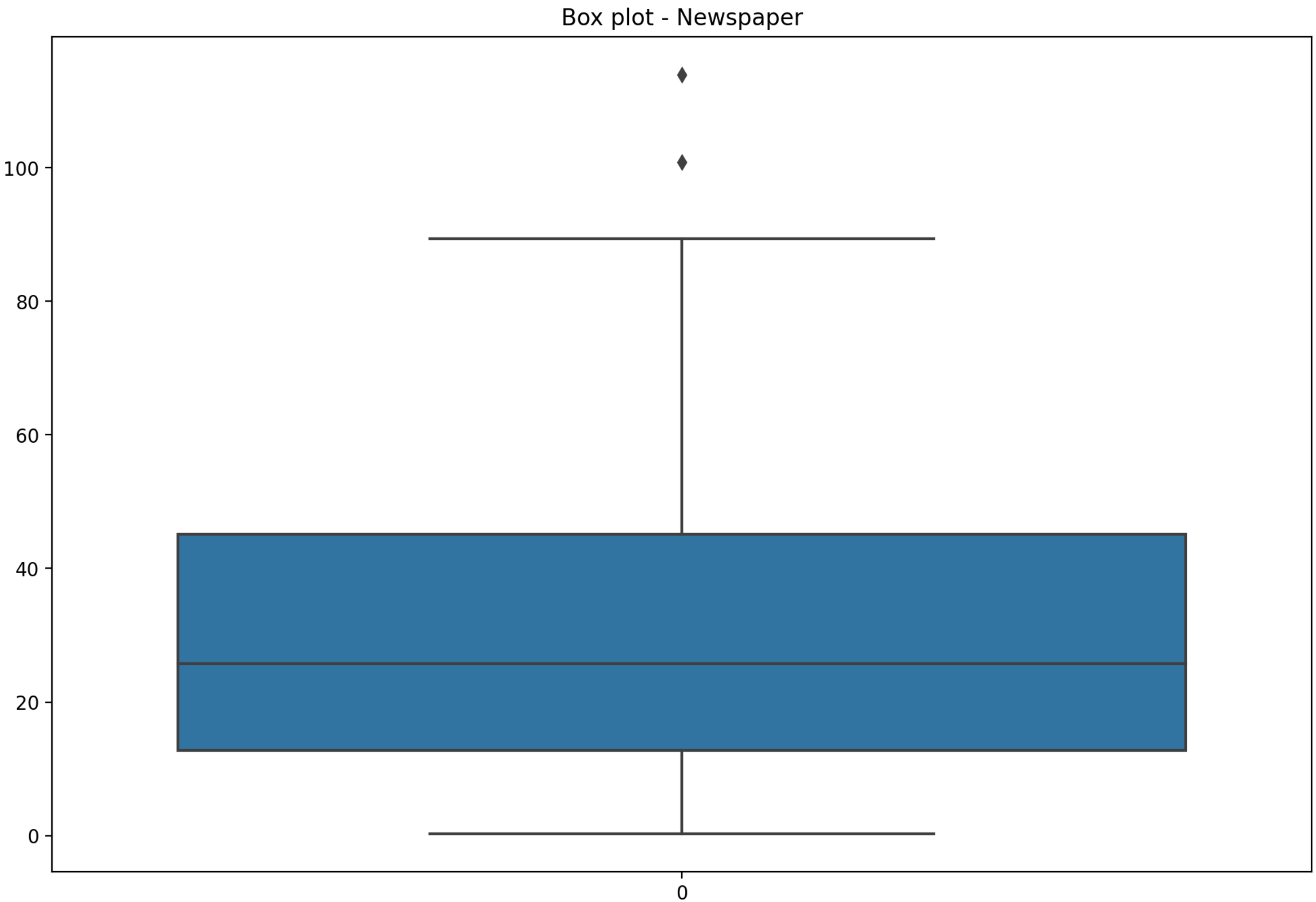
In [19]:
```
df.columns
```

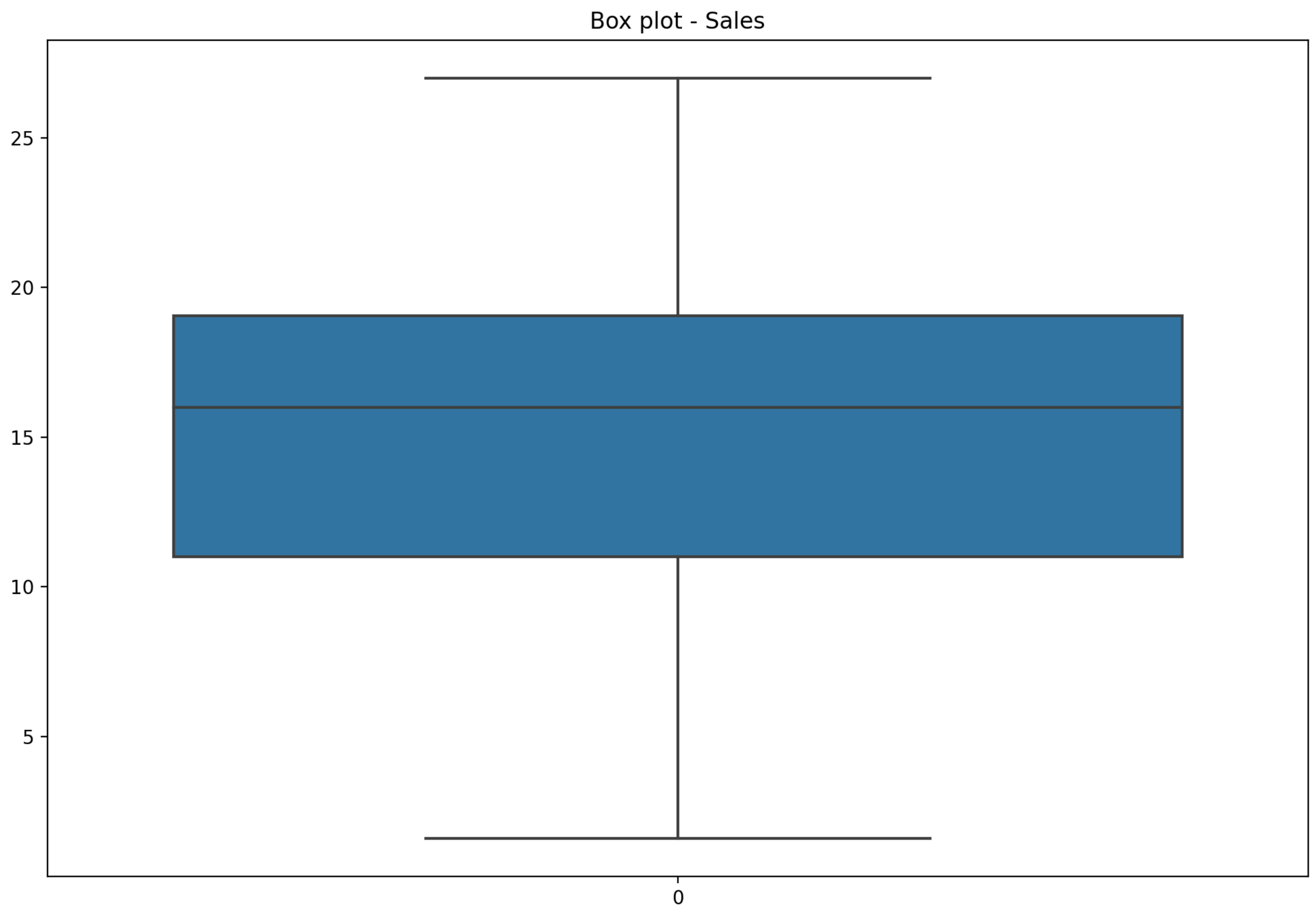
Out[19]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [22]:
```
# Select the column containing numerical data :-
numerical_columns = df.columns

# Create box plots for each numerical column :
for column in numerical_columns:
    plt.figure(figsize=(12,8), dpi = 200)
    sns.boxplot(data= df[column])
    plt.title(f'Box plot - {column}')

    plt.show()
```


Box plot - TV
300
250
200
150
100
50
0
0


Box plot - Radio
50
40
30
20
10
0
0


Box plot - Newspaper
100
80
60
40
20
0
0


Box plot - Sales
25
20
15
10
5
0

Removing Outliers :-

In [23]:
```
# Newspaper column has outliers and removing outliers :
q1, q2, q3 = np.percentile (df["Newspaper"], [25,50,75])
iqr = q3-q1
lower_extreme = q1=1.5*iqr
upper_extreme = q3+1.5*iqr
df= df.loc[(df["Newspaper"]>= lower_extreme) & (df["Newspaper"]<=upper_extreme)]

df
```

Out[23]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
5	8.7	48.9	75.0	7.2
12	23.8	35.1	65.9	9.2
15	195.4	47.7	52.9	22.4
17	281.4	39.6	55.8	24.4
20	218.4	27.7	53.4	18.0
22	13.2	15.9	49.6	5.6
48	227.2	15.8	49.9	19.8
53	182.6	46.2	58.7	21.2
55	198.9	49.4	60.0	23.7
61	261.3	42.7	54.7	24.2
75	16.9	43.7	89.4	8.7
85	193.2	18.4	65.7	20.2
87	110.7	40.6	63.2	16.0
88	88.3	25.5	73.4	12.9
89	109.8	47.8	51.4	16.7
92	217.7	33.5	59.0	19.4
93	250.9	36.5	72.3	22.2
95	163.3	31.6	52.9	16.9
98	289.7	42.3	51.2	25.4
100	222.4	4.3	49.8	16.7
105	137.9	46.4	59.0	15.0

	TV	Radio	Newspaper	Sales
110	225.8	8.2	56.5	18.4
115	75.1	35.0	52.7	12.6
118	125.7	36.9	79.2	15.9
121	18.8	21.7	50.4	7.0
124	229.5	32.3	74.2	19.7
126	7.8	38.9	50.6	6.6
134	36.9	38.6	65.6	10.8
137	273.7	28.9	59.7	20.8
141	193.7	35.4	75.6	19.2
151	121.0	8.4	48.7	11.6
156	93.9	43.5	50.5	15.3
161	85.7	35.8	49.3	13.3
165	234.5	3.4	84.8	16.9
168	215.4	23.6	57.6	17.1
183	287.6	43.0	71.8	26.2
198	283.6	42.0	66.2	25.5

In [24]:

```
df.reset_index(drop = True, inplace = True)
df
```

Out[24]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	17.2	45.9	69.3	12.0
2	151.5	41.3	58.5	16.5
3	180.8	10.8	58.4	17.9
4	8.7	48.9	75.0	7.2
5	23.8	35.1	65.9	9.2
6	195.4	47.7	52.9	22.4
7	281.4	39.6	55.8	24.4
8	218.4	27.7	53.4	18.0
9	13.2	15.9	49.6	5.6
10	227.2	15.8	49.9	19.8
11	182.6	46.2	58.7	21.2
12	198.9	49.4	60.0	23.7
13	261.3	42.7	54.7	24.2
14	16.9	43.7	89.4	8.7
15	193.2	18.4	65.7	20.2
16	110.7	40.6	63.2	16.0
17	88.3	25.5	73.4	12.9
18	109.8	47.8	51.4	16.7
19	217.7	33.5	59.0	19.4
20	250.9	36.5	72.3	22.2
21	163.3	31.6	52.9	16.9
22	289.7	42.3	51.2	25.4
23	222.4	4.3	49.8	16.7
24	137.9	46.4	59.0	15.0

	TV	Radio	Newspaper	Sales
25	225.8	8.2	56.5	18.4
26	75.1	35.0	52.7	12.6
27	125.7	36.9	79.2	15.9
28	18.8	21.7	50.4	7.0
29	229.5	32.3	74.2	19.7
30	7.8	38.9	50.6	6.6
31	36.9	38.6	65.6	10.8
32	273.7	28.9	59.7	20.8
33	193.7	35.4	75.6	19.2
34	121.0	8.4	48.7	11.6
35	93.9	43.5	50.5	15.3
36	85.7	35.8	49.3	13.3
37	234.5	3.4	84.8	16.9
38	215.4	23.6	57.6	17.1
39	287.6	43.0	71.8	26.2
40	283.6	42.0	66.2	25.5

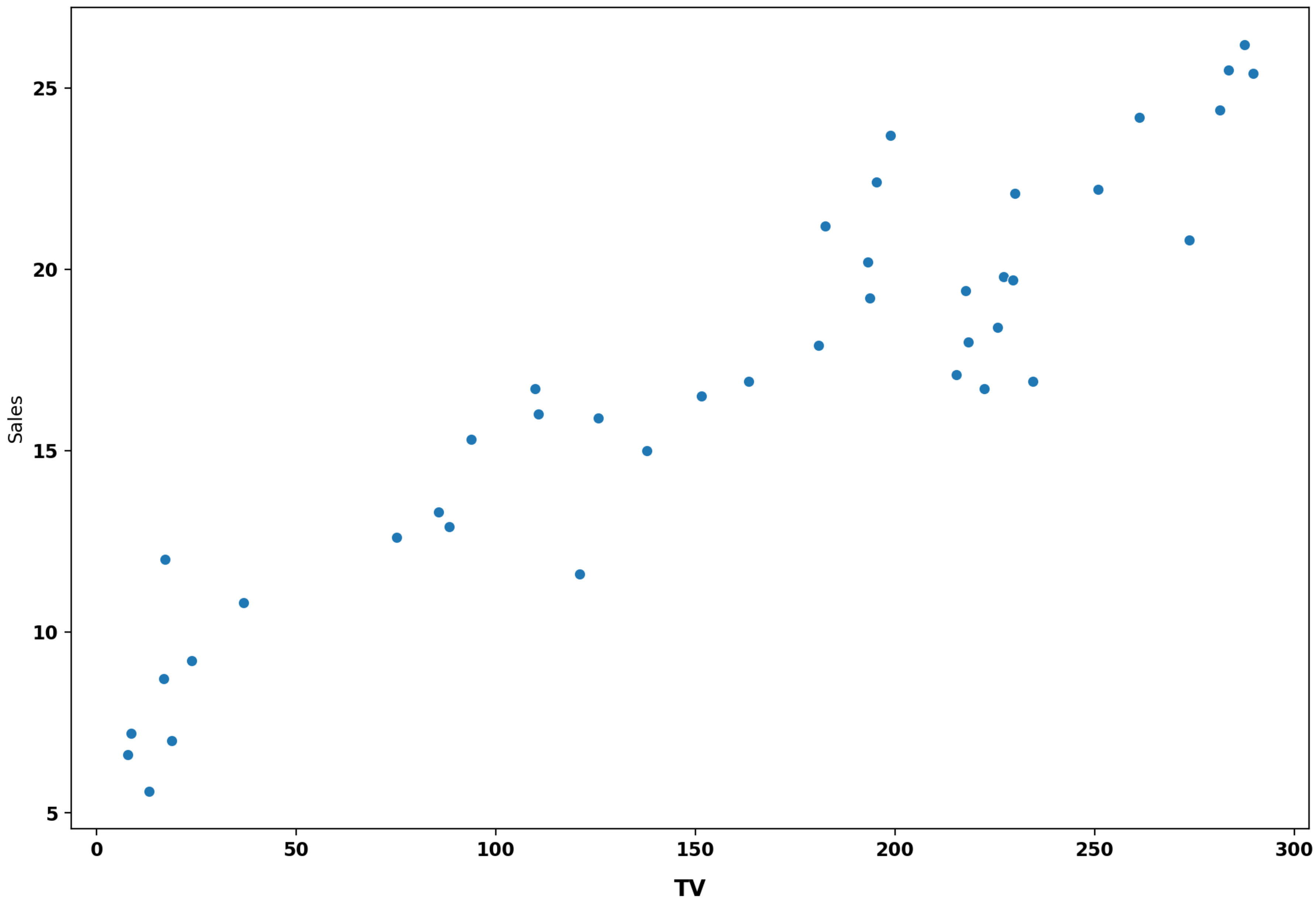
Exploratory Data Analysis :-

In [25]:
```
df.describe()
```

Out[25]:

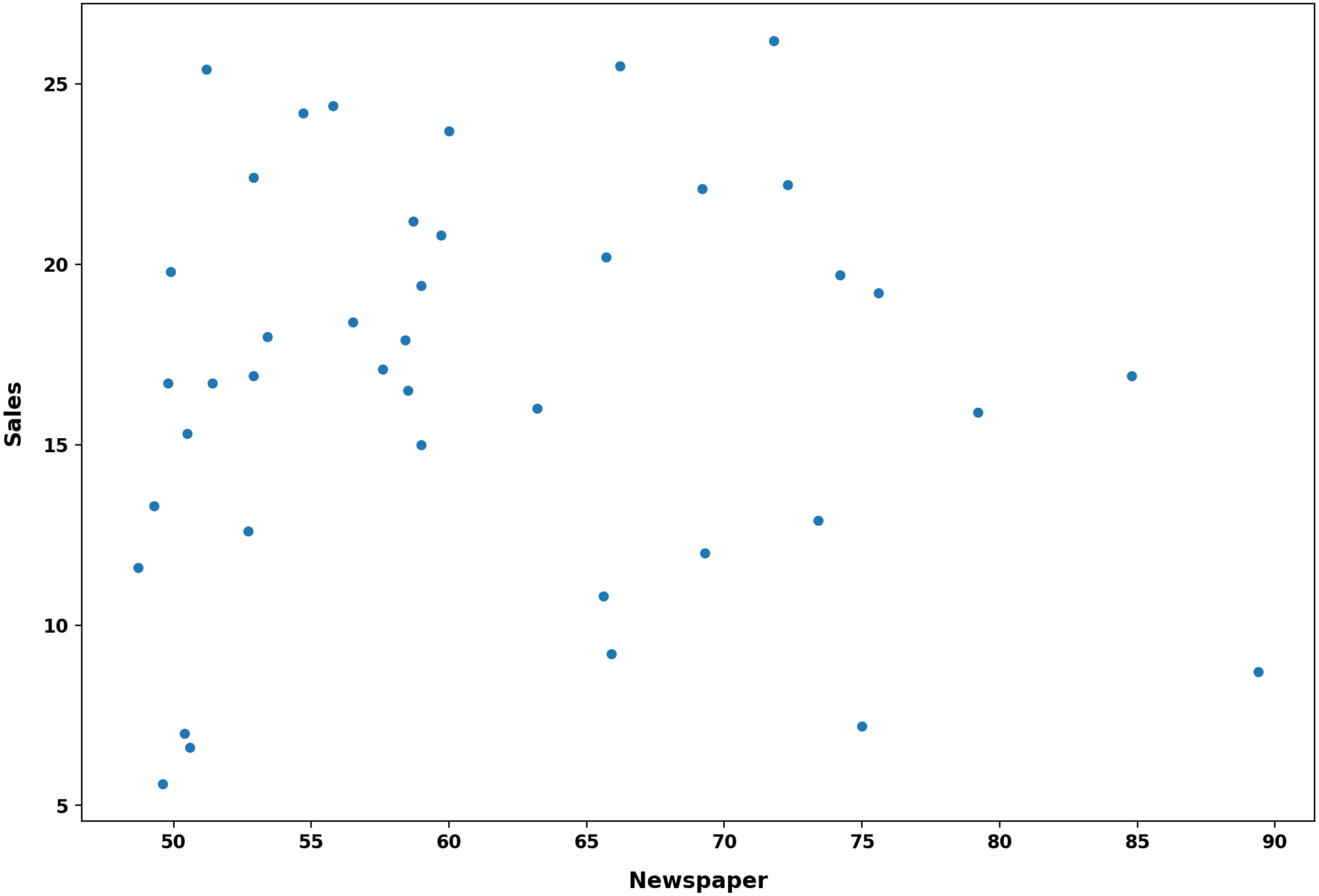
	TV	Radio	Newspaper	Sales
count	41.000000	41.000000	41.000000	41.000000
mean	158.536585	32.951220	61.268293	16.858537
std	90.946343	13.140037	10.593877	5.585829
min	7.800000	3.400000	48.700000	5.600000
25%	88.300000	25.500000	52.700000	12.900000
50%	182.600000	36.500000	58.700000	16.900000
75%	227.200000	42.700000	69.200000	20.800000
max	289.700000	49.400000	89.400000	26.200000

In [27]:
```
plt.figure(figsize=(12,8), dpi = 200)
sns.scatterplot(data = df, x = df['TV'], y = df["Sales"])
plt.xlabel("TV", weight = "bold", fontsize= 12, labelpad= 10)
plt.xticks(weight = "bold")
plt.yticks(weight = "bold")

plt.show()
```



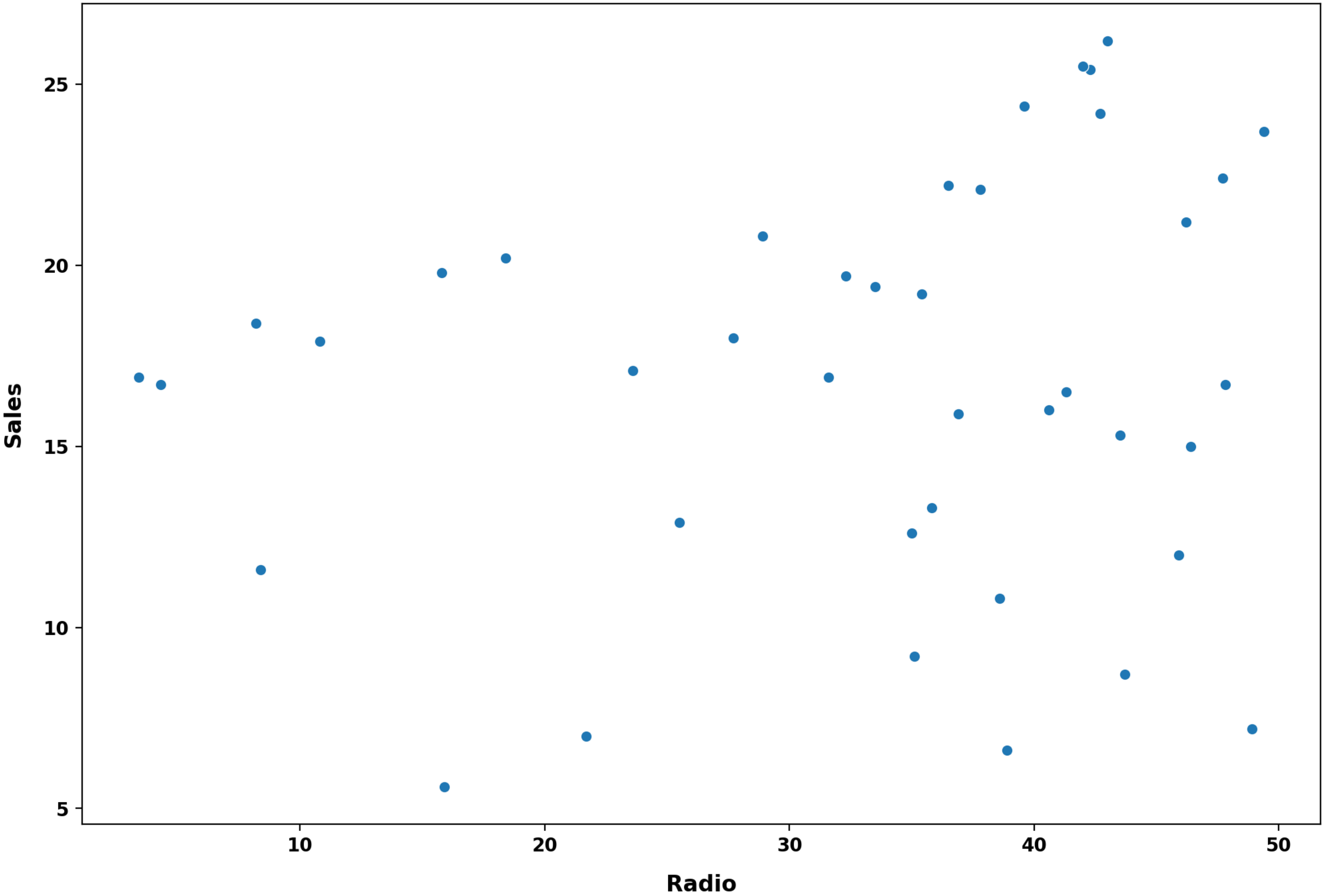
Sales
25
20
15
10
5
0
50
100
150
200
250
300
TV

In [32]:
```
plt.figure(figsize=(12,8), dpi = 200)
sns.scatterplot(data = df, x = df["Newspaper"], y = df["Sales"])
plt.xlabel("Newspaper", weight = "bold", fontsize = 12, labelpad = 10)
plt.ylabel("Sales", weight = "bold", fontsize = 12, labelpad = 10)
plt.xticks(weight = "bold")
plt.yticks(weight = "bold")

plt.show()
```


Sales
Newspaper
25
20
15
10
5
50
55
60
65
70
75
80
85
90

In [34]:

```
plt.figure(figsize=( 12, 8), dpi = 200)
sns.scatterplot(data=df, x =df["Radio"], y = df["Sales"])
plt.xlabel("Radio", weight = "bold", fontsize = 12, labelpad= 10)
plt.ylabel("Sales", weight = "bold", fontsize = 12, labelpad= 10)
plt.xticks(weight = "bold")
plt.yticks(weight = "bold")

plt.show()
```



25
20
15
10
5
Sales
10
20
30
40
50
Radio

Finding Correlation :-

In [35]:

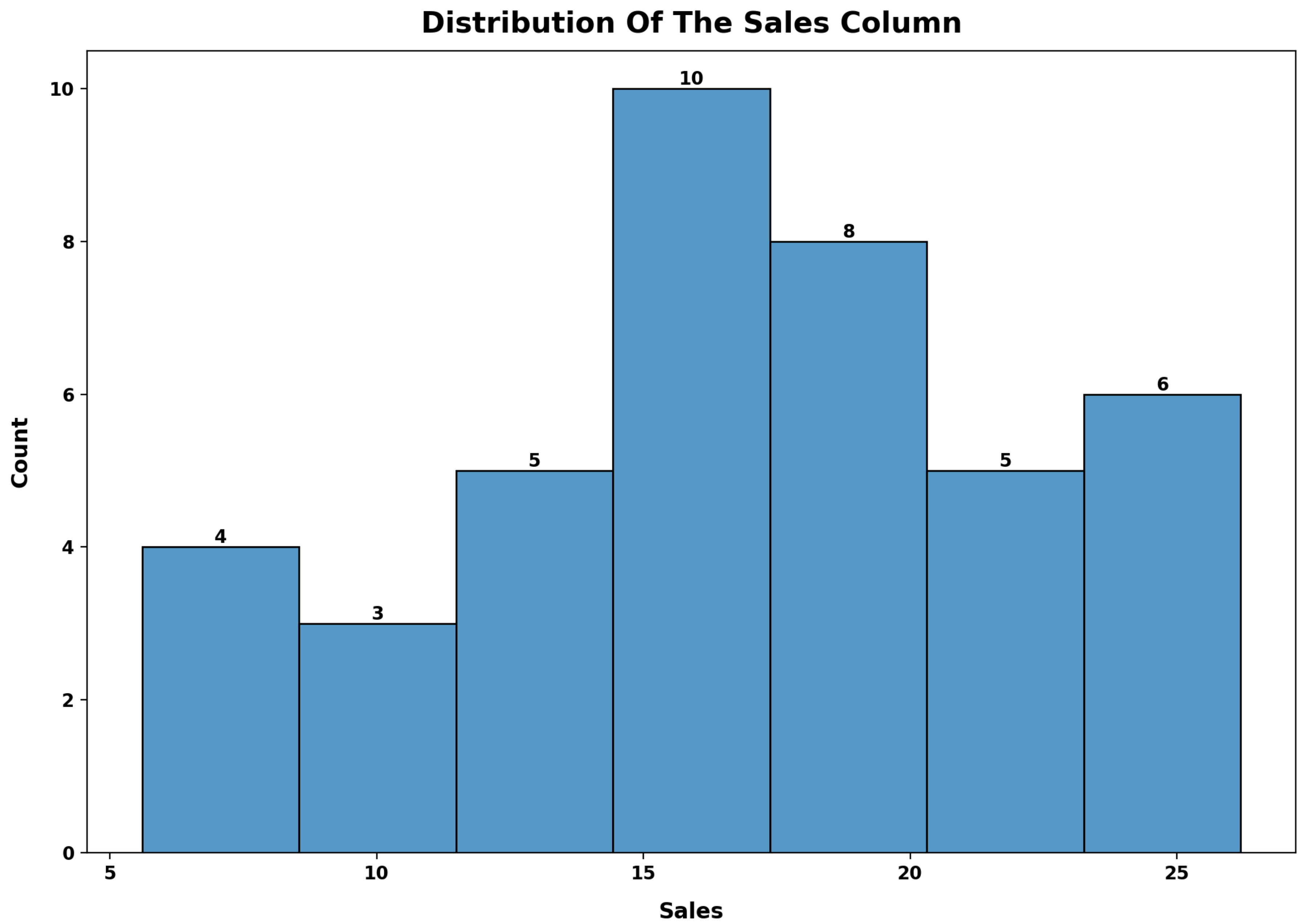
```
plt.figure(figsize= (12,8), dpi=200)
sns.heatmap(data = df.corr(), annot = True)

plt.show()
```


TV
Radio
Newspaper
Sales
1
-0.13
-0.022
0.93
-0.13
1
0.14
0.16
-0.022
0.14
1
0.0037
0.93
0.16
0.0037
1
TV
Radio
Newspaper
Sales
1.0
0.8
0.6
0.4
0.2
0.0

Distribution of the sales column :-

In [36]:

```
plt.figure(figsize=(12,8), dpi = 200)
ax =sns.histplot(data = df, x = df["Sales"])
plt.title("Distribution Of The Sales Column", fontsize = 16 , weight = "bold" , pad= 10)
plt.xlabel("Sales", weight = "bold", fontsize = 12, labelpad = 10)
plt.ylabel("Count", weight = "bold", fontsize = 12, labelpad = 10)
plt.xticks(weight = "bold")
plt.yticks(weight = "bold")


for i in ax.containers:
    i.datavalues
    ax.bar_label(i, weight="bold")


plt.show()
```



Distribution Of The Sales Column
Count
Sales
0
2
4
6
8
10
5
10
15
20
25
4
3
5
10
8
5
6

Model Building :-

Define Dataset

In [38]:
```
x = df.drop(columns = "Sales",  axis = 1)
y = df["Sales"]
```

Train_Test_Split :-

In [39]:
```
x_train ,x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=True)
```

Training Model :-

In [40]:
```
linearRegression = LinearRegression()
linearRegression.fit(x_train, y_train)
```

Out[40]:
```
▾ LinearRegression
LinearRegression()
```

In [42]:
```
y_predict = linearRegression.predict(x_test)
y_predict
```

Out[42]:
```
array([15.53077258, 17.69982501, 17.35459739, 19.21632282, 11.60746587,
        7.60107283, 20.58077151, 11.94840329,  9.14521359])
```

Mean Squared Error

In [43]:
```
math.sqrt(mean_squared_error(y_predict, y_test))
```

Out[43]:
```
1.562966908776184
```

Prediction Value

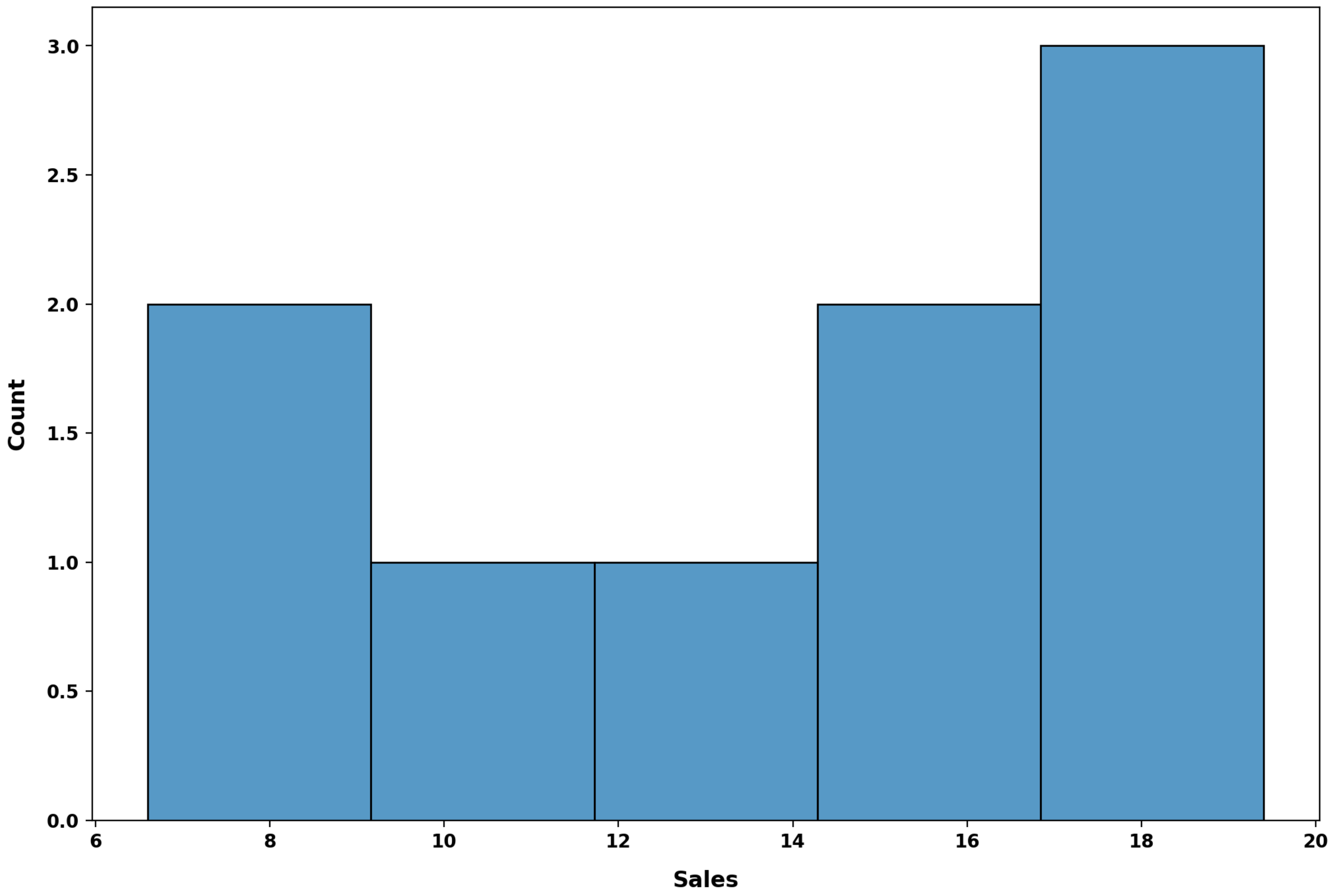
In [44]:
```
linearRegression.predict([[230.1, 37.8, 69.2]])
```

```
D:\rsvp movies\New folder\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, bu
t LinearRegression was fitted with feature names
  warnings.warn(
```

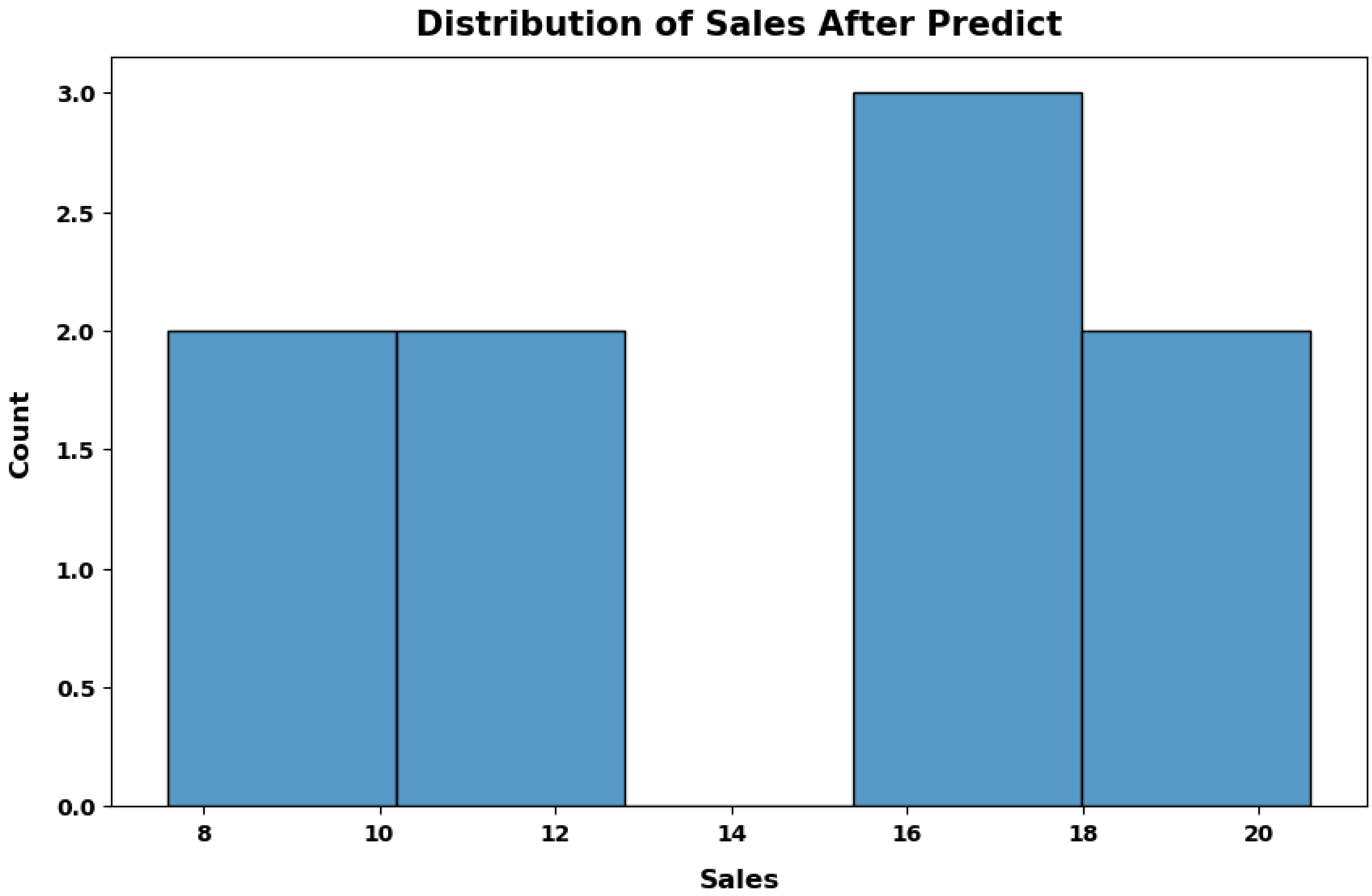
Out[44]:
```
array([21.60913036])
```

Evaluate Model Performance :-

In [45]:
```
plt.figure(figsize= (12,8), dpi = 200)
sns.histplot(x=y_test)
plt.xlabel('Sales', weight ="bold", fontsize = 12, labelpad= 10)
plt.ylabel('Count', weight= "bold", fontsize = 12, labelpad = 10)
plt.xticks(weight = "bold")
plt.yticks (weight= "bold")

plt.show()
```

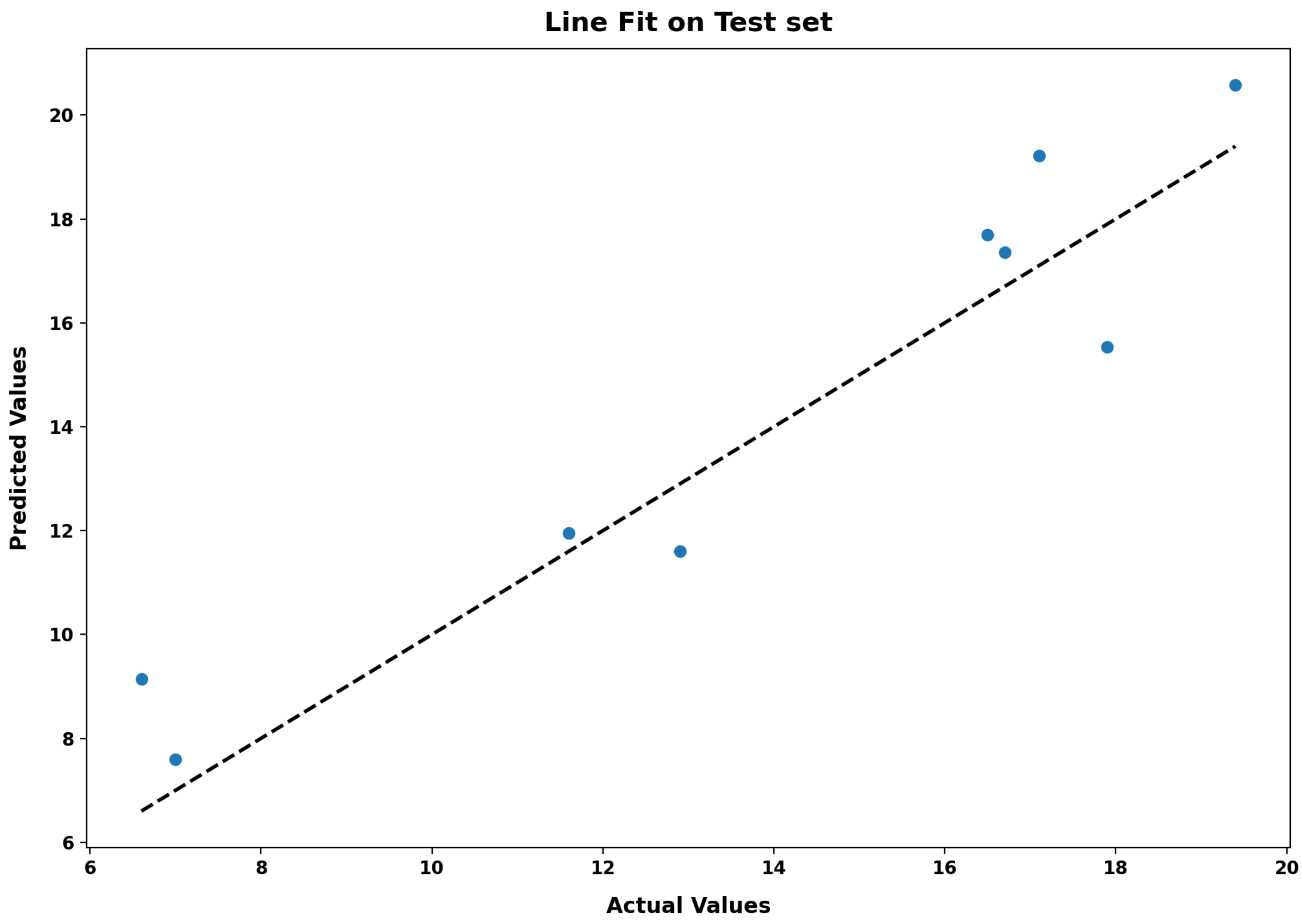

Count
3.0
2.5
2.0
1.5
1.0
0.5
0.0
6
8
10
12
14
16
18
20
Sales

In [52]:
```
plt.figure(figsize =(10,6))
sns.histplot(x=y_predict)
plt.xlabel('Sales', weight ="bold", fontsize= 12, labelpad = 10)
plt.ylabel("Count", weight = "bold", fontsize= 12, labelpad = 10)
plt.title("Distribution of Sales After Predict", fontsize = 15, weight ="bold", pad = 10)
plt.xticks(weight = "bold")
plt.yticks(weight = "bold")

plt.show()
```


Distribution of Sales After Predict
Count
Sales
0.0
0.5
1.0
1.5
2.0
2.5
3.0
8
10
12
14
16
18
20

Visualize Fit Of The On The Test Set :-

In [57]:
```
# The predicted values against the actual values :
plt.figure(figsize=(12,8), dpi = 200)
plt.scatter(x = y_test, y = y_predict)
plt.plot([y_test.min(), y_test.max()], [y_test.min(),y_test.max()], 'k--', lw=2)
plt.xlabel('Actual Values', weight = "bold", fontsize=12, labelpad =10)
plt.ylabel('Predicted Values', weight = "bold", fontsize=12, labelpad=10)
plt.title('Line Fit on Test set', fontsize = 15, weight ='bold', pad=10)
plt.xticks(weight ="bold")
plt.yticks(weight = "bold")

plt.show()
```


Line Fit on Test set
Predicted Values
Actual Values
6
8
10
12
14
16
18
20
6
8
10
12
14
16
18
20

In [ ]: