PROJECT NAME = SALES PREDICTION

Importing Library

In [123]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

Data

In [124]:
```
df=pd.read_csv("C:\\Users\\DISHA__COMPUTERS\\Desktop\\Internship\\4) adver
```

In [125]:
```
df
```

Out[125]:

	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
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

EDA

In [126]:
```
df.head()
```

Out[126]:

	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

In [127]:
```
df.tail()
```

Out[127]:

	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

In [128]:
```
df.describe()
```

Out[128]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

In [129]:
```
type(df)
```

Out[129]:
```
pandas.core.frame.DataFrame
```

In [130]:
```
df.shape
```

Out[130]:
```
(200, 4)
```

In [132]:
```
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
```

Data Cleaning

In [133]:
```
df.isna().sum()
```

Out[133]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [134]:
```
df.duplicated()
```

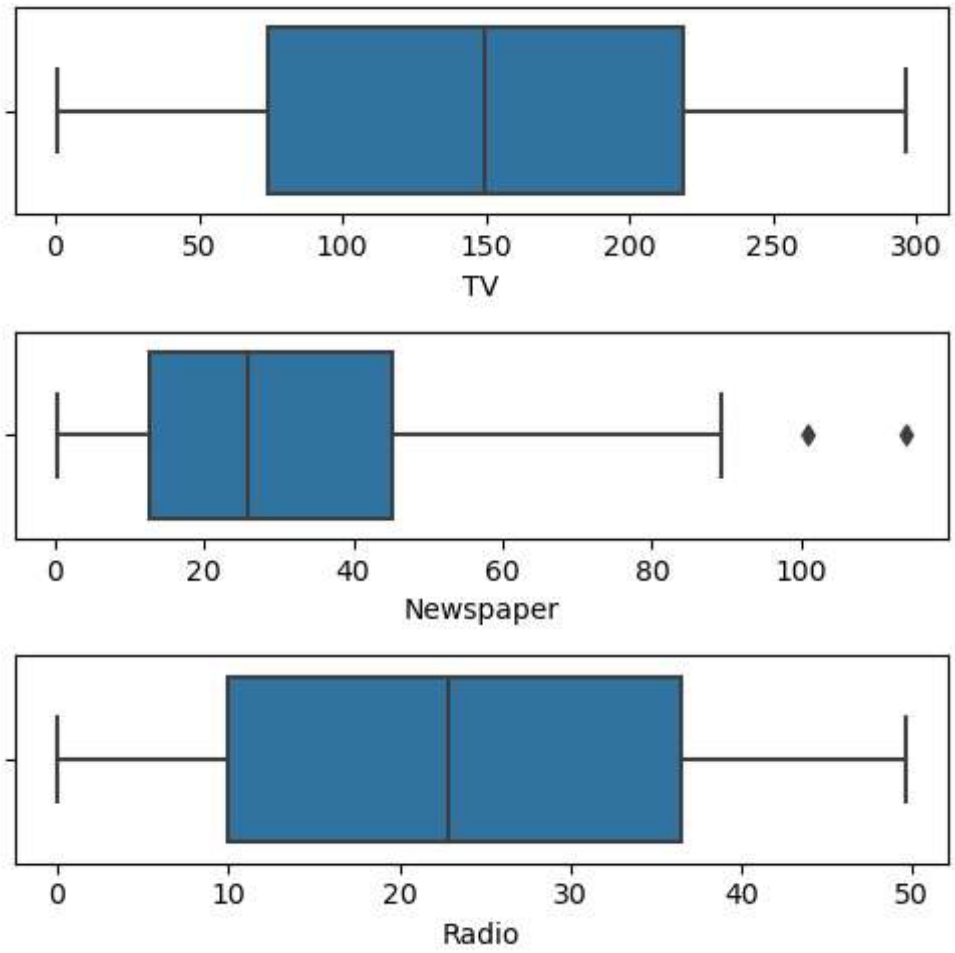
Out[134]:
```
0      False
1      False
2      False
3      False
4      False
       ...
195    False
196    False
197    False
198    False
199    False
Length: 200, dtype: bool
```

In [135]:
```
df.duplicated().sum()
```

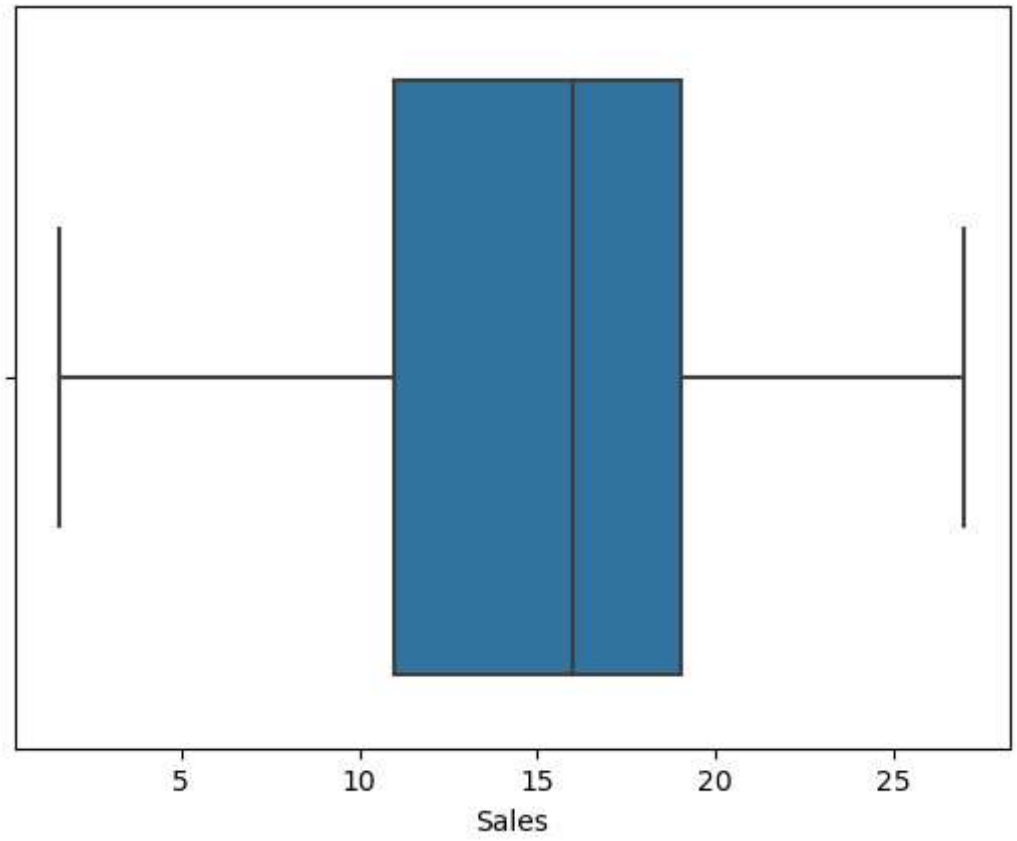
Out[135]:
```
0
```

Data Visualization

In [136]:
```
fig, axs = plt.subplots(3, figsize = (5,5))
plt1 = sns.boxplot(df['TV'],ax = axs[0])
plt2 = sns.boxplot(df['Newspaper'], ax = axs[1])
plt3 = sns.boxplot(df['Radio'], ax = axs[2])
plt.tight_layout()
```

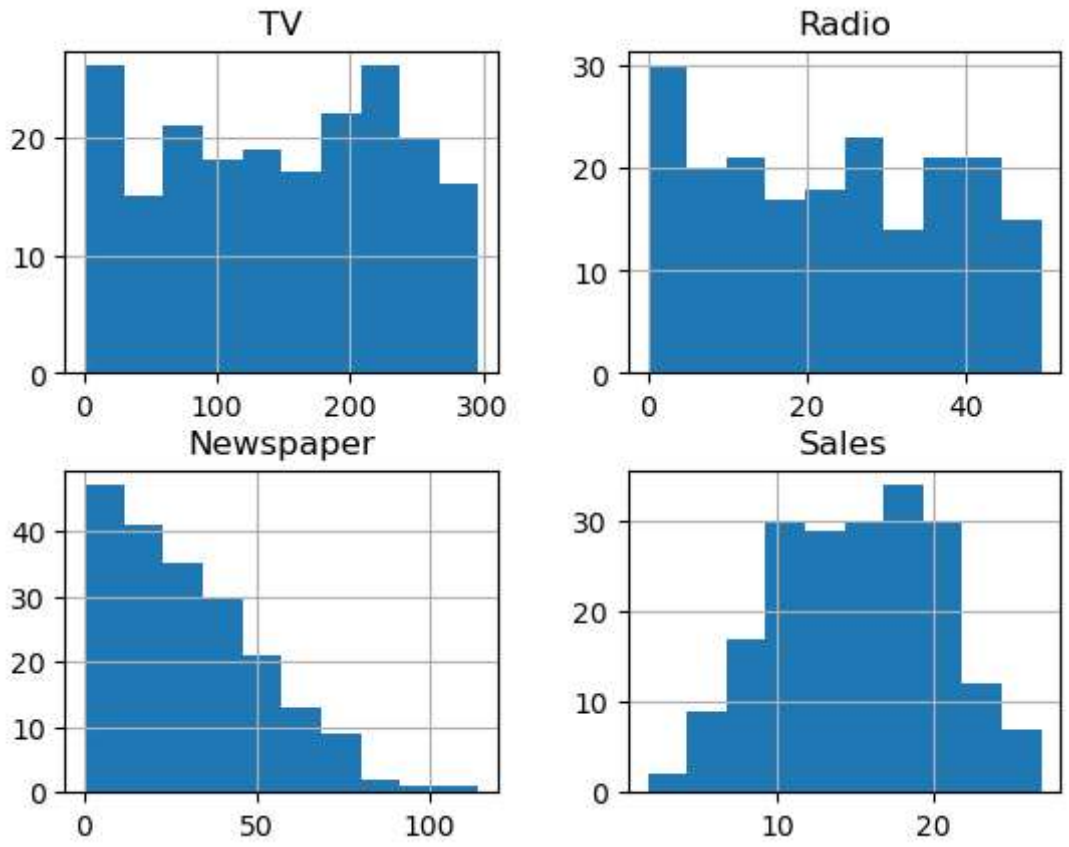


In [137]:
```
sns.boxplot(df['Sales'])
plt.show()
```

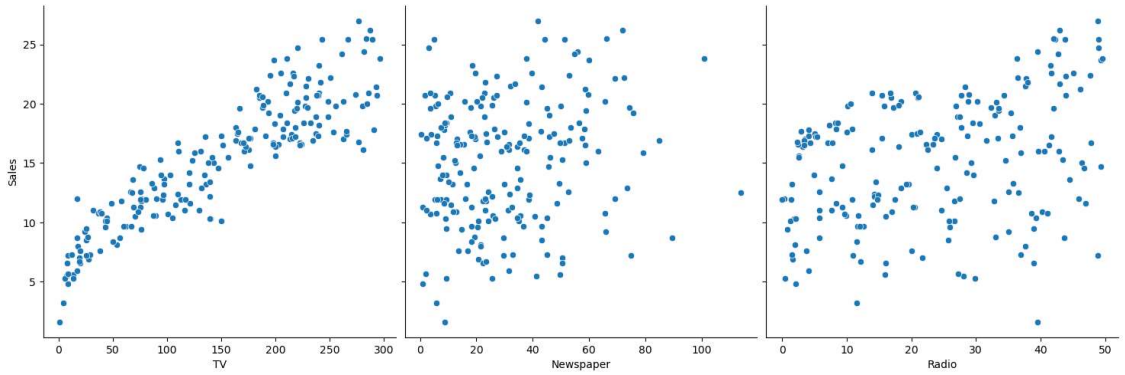


In [138]:
```
df.hist()
```

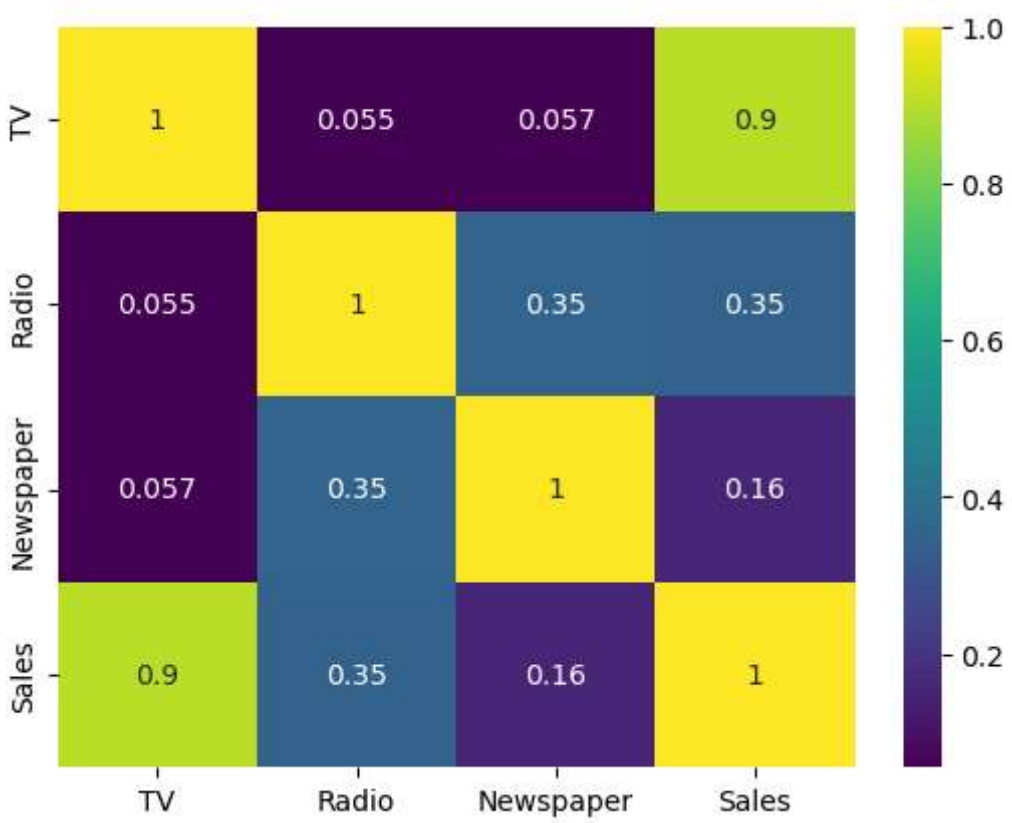
Out[138]:
```
array([[<AxesSubplot:title={'center':'TV'}>,
        <AxesSubplot:title={'center':'Radio'}>],
       [<AxesSubplot:title={'center':'Newspaper'}>,
        <AxesSubplot:title={'center':'Sales'}>]], dtype=object)
```



In [139]:
```
sns.pairplot(df, x_vars =['TV', 'Newspaper', 'Radio'], y_vars='Sales', hei
plt.show()
```



In [140]:
```
sns.heatmap(df.corr(), cmap="viridis", annot = True)
plt.show()
```



In [141]:
```
X = df['TV']
y = df['Sales']
```

In [142]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.9,
```

In [143]:
```
import statsmodels.api as sm
```

In [144]:
```
train = sm.add_constant(X_train)


lr = sm.OLS(y_train, train).fit()
```

In [145]:
```
lr.params
```

Out[145]:
```
const    6.548509
TV       0.058475
dtype: float64
```

In [146]:
```
print(lr.summary())
```

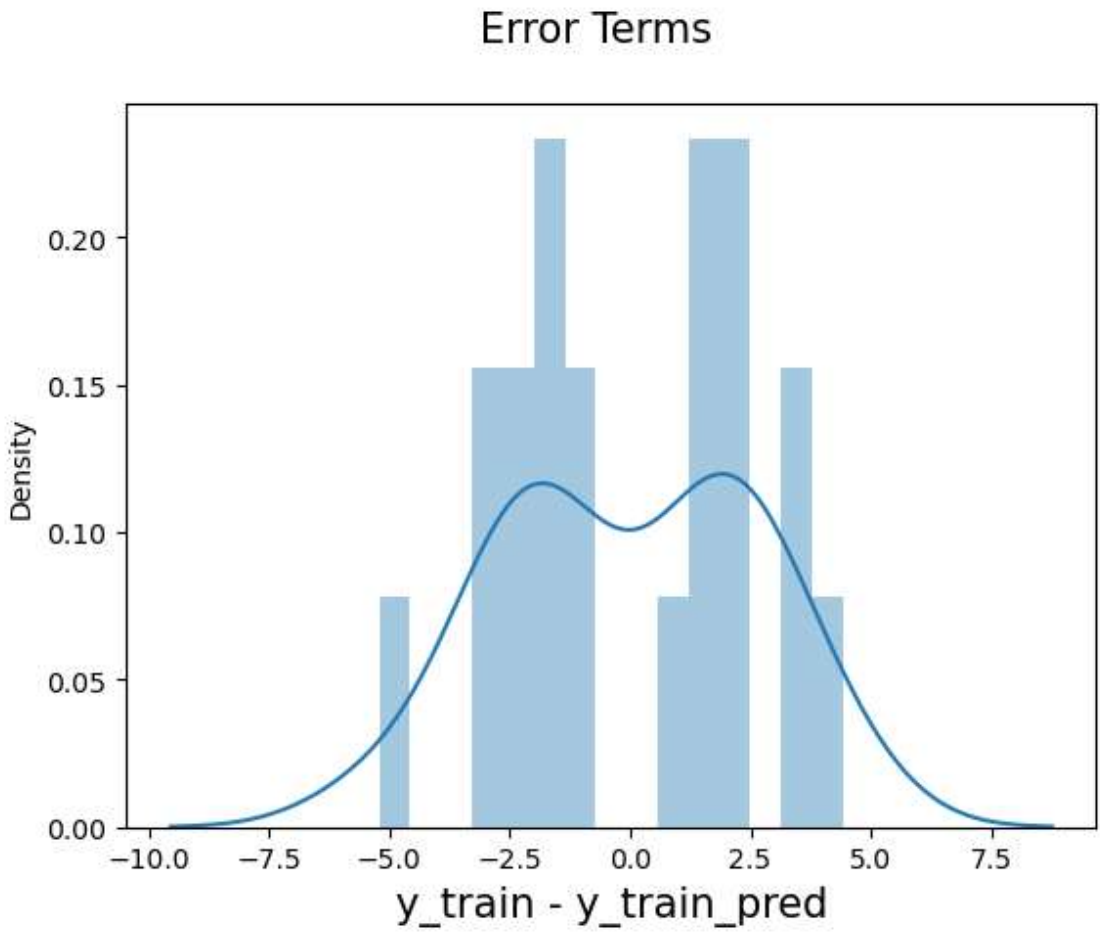
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.708
Model:                            OLS   Adj. R-squared:                  0.692
Method:                 Least Squares   F-statistic:                     43.60
Date:                Sat, 09 Sep 2023   Prob (F-statistic):           3.36e-06
Time:                        11:29:53   Log-Likelihood:                -47.205
No. Observations:                  20   AIC:                             98.41
Df Residuals:                      18   BIC:                             100.4
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.5485      1.386      4.725      0.000       3.637       9.460
TV             0.0585      0.009      6.603      0.000       0.040       0.077
==============================================================================
Omnibus:                        1.544   Durbin-Watson:                   2.139
Prob(Omnibus):                  0.462   Jarque-Bera (JB):                0.944
Skew:                          -0.123   Prob(JB):                        0.624
Kurtosis:                       1.965   Cond. No.                         359.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

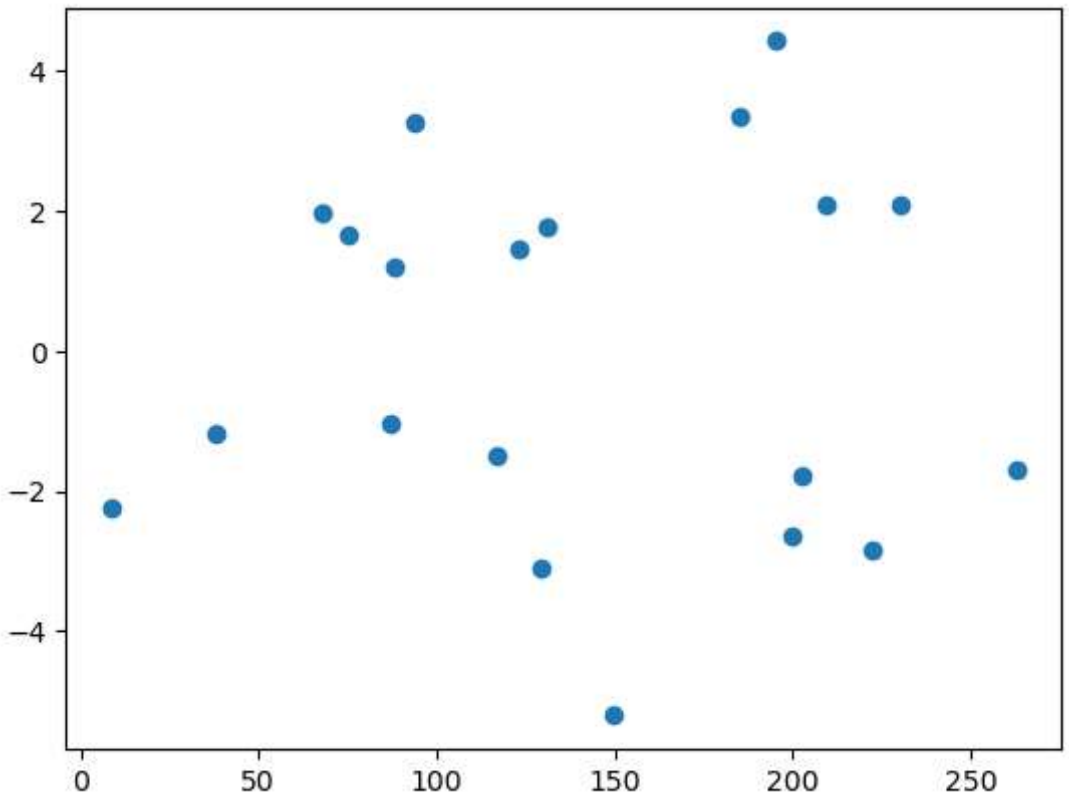
In [147]:
```
y_train_pred = lr.predict(train)
res = (y_train - y_train_pred)
```

In [148]:

```
fig = plt.figure()
sns.distplot(res, bins = 15)
fig.suptitle('Error Terms', fontsize = 15)
plt.xlabel('y_train - y_train_pred', fontsize = 15)
plt.show()
```



In [149]:
```
plt.scatter(X_train,res)
plt.show()
```



In [150]:
```
test = sm.add_constant(X_test)
y_pred = lr.predict(test)
```

In [151]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

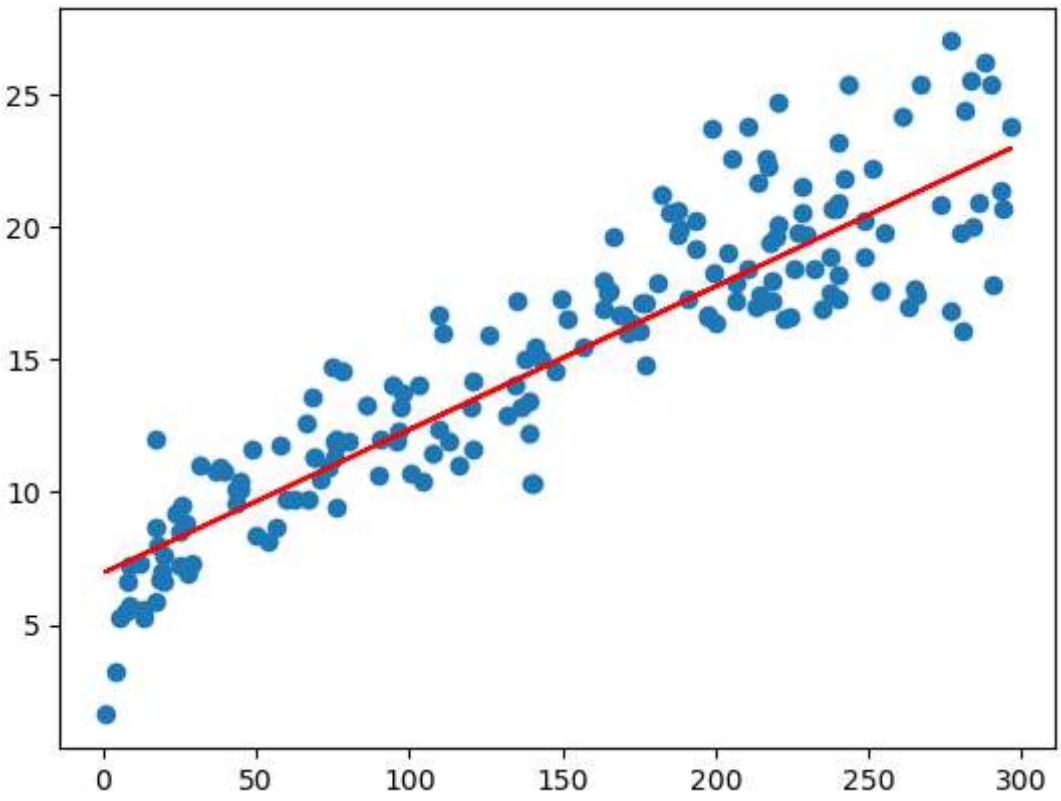
In [152]:
```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[152]: 2.267503601357533

In [153]:
```
r_squared = r2_score(y_test, y_pred)
r_squared
```

Out[153]: 0.8186609418736366

In [154]:
```
plt.scatter(X_test, y_test)
plt.plot(X_test, 6.948 + 0.054 * X_test, 'r')
plt.show()
```



LinearRegression

In [155]:
```
X_train, X_test, y_train, y_test = train_test_split(df.drop('Sales', axis=
```

In [156]:
```
from sklearn.linear_model import LinearRegression
```

In [157]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[157]:
```
LinearRegression()
```

In [158]:
```
print('Coefficients:', lr.coef_)
print('Intercept:', lr.intercept_)
```

```
Coefficients: [0.05450927 0.10094536 0.00433665]
Intercept: 4.714126402214134
```

In [159]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sc
```

In [160]:
```
lr_preds = lr.predict(X_test)
lr_mae = mean_absolute_error(y_test, lr_preds)
lr_mse = mean_squared_error(y_test, lr_preds)
lr_r2 = r2_score(y_test, lr_preds)
```

In [161]:
```
print('Linear Regression - MAE:', lr_mae, 'MSE:', lr_mse, 'R-squared:', lr
```

```
Linear Regression - MAE: 1.274826210954934 MSE: 2.907756910271091 R-squa
red: 0.9059011844150826
```

In [ ]:
```

```