```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data = pd.read_csv('E:/DOWNLOAD/IceCreamData.csv')

data.head(5)
```

```
   Temperature     Revenue
0    24.566884  534.799028
1    26.005191  625.190122
2    27.790554  660.632289
3    20.595335  487.706960
4    11.503498  316.240194
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Temperature  500 non-null    float64
 1   Revenue      500 non-null    float64
dtypes: float64(2)
memory usage: 7.9 KB
```

```
data.describe()
```

```
       Temperature      Revenue
count   500.000000   500.000000
mean     22.232225   521.570777
std       8.096388   175.404751
min       0.000000    10.000000
25%      17.122258   405.558681
50%      22.392791   529.368565
75%      27.740674   642.257922
max      45.000000  1000.000000
```

```
data.shape
```

```
(500, 2)
```

```
data['Revenue'].min()
data['Temperature'].min()
data['Revenue'].max()
data['Temperature'].max()
```

```
45.0
```

```
data.groupby('Revenue').describe()
```

```
             Temperature
\
                   count       mean std        min        25%
50%
Revenue

10.000000            1.0   0.000000 NaN   0.000000   0.000000
0.000000
32.546619            1.0   0.267028 NaN   0.267028   0.267028
0.267028
55.390338            1.0   0.976870 NaN   0.976870   0.976870
0.976870
71.160153            1.0   3.664670 NaN   3.664670   3.664670
3.664670
118.812150           1.0   4.236465 NaN   4.236465   4.236465
4.236465
...                  ...        ...  ..        ...        ...         ..
.
926.067153           1.0  40.303768 NaN  40.303768  40.303768
40.303768
935.717291           1.0  39.764129 NaN  39.764129  39.764129
39.764129
965.493040           1.0  41.924446 NaN  41.924446  41.924446
41.924446
969.291630           1.0  41.766589 NaN  41.766589  41.766589
41.766589
1000.000000          1.0  45.000000 NaN  45.000000  45.000000
45.000000


                     75%        max
Revenue
10.000000       0.000000   0.000000
32.546619       0.267028   0.267028
55.390338       0.976870   0.976870
71.160153       3.664670   3.664670
118.812150      4.236465   4.236465
...                  ...        ...
926.067153     40.303768  40.303768
935.717291     39.764129  39.764129
965.493040     41.924446  41.924446
969.291630     41.766589  41.766589
1000.000000    45.000000  45.000000

[500 rows x 8 columns]
```

```
data.groupby('Temperature').describe()
```

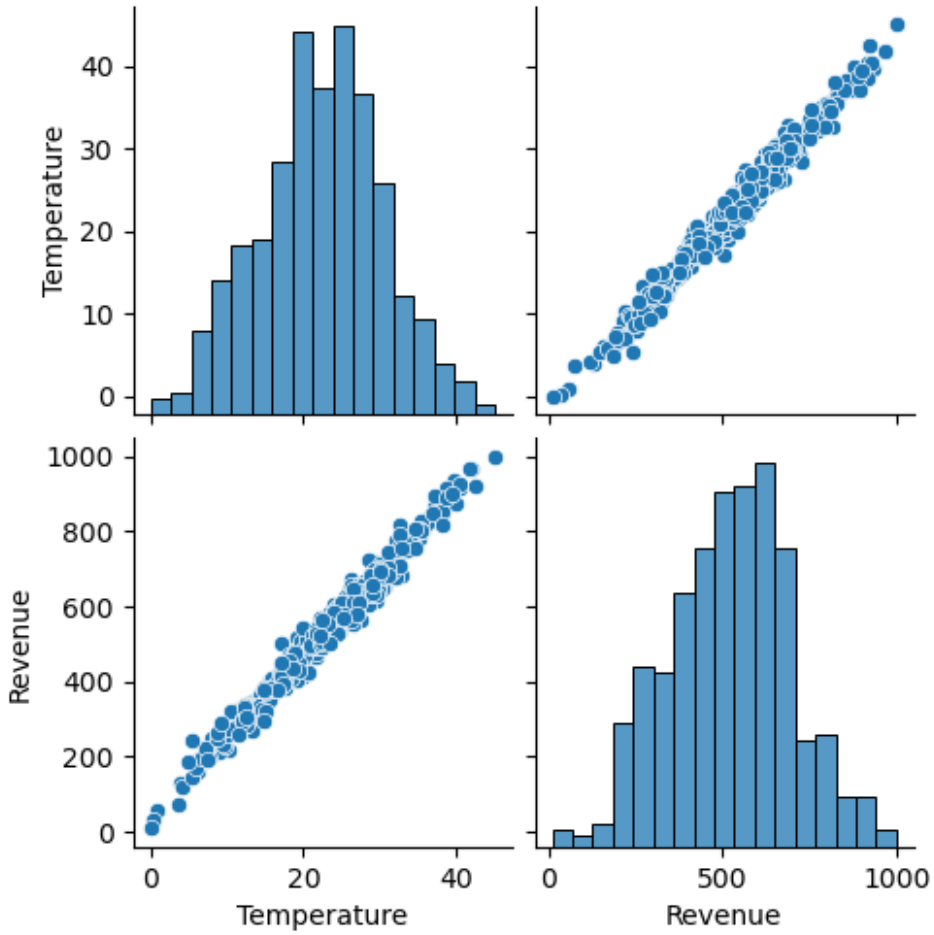
```
             Revenue
\
               count         mean std          min          25%
50%
Temperature

0.000000         1.0    10.000000 NaN    10.000000    10.000000
10.000000
0.267028         1.0    32.546619 NaN    32.546619    32.546619
32.546619
0.976870         1.0    55.390338 NaN    55.390338    55.390338
55.390338
3.664670         1.0    71.160153 NaN    71.160153    71.160153
71.160153
3.986523         1.0   131.657017 NaN   131.657017   131.657017
131.657017
...              ...          ...  ..          ...          ...
...
40.473989        1.0   918.391232 NaN   918.391232   918.391232
918.391232
41.766589        1.0   969.291630 NaN   969.291630   969.291630
969.291630
41.924446        1.0   965.493040 NaN   965.493040   965.493040
965.493040
42.515280        1.0   921.508275 NaN   921.508275   921.508275
921.508275
45.000000        1.0  1000.000000 NaN  1000.000000  1000.000000
1000.000000


                     75%          max
Temperature
0.000000       10.000000    10.000000
0.267028       32.546619    32.546619
0.976870       55.390338    55.390338
3.664670       71.160153    71.160153
3.986523      131.657017   131.657017
...                  ...          ...
40.473989     918.391232   918.391232
41.766589     969.291630   969.291630
41.924446     965.493040   965.493040
42.515280     921.508275   921.508275
45.000000    1000.000000  1000.000000

[500 rows x 8 columns]
```

```
data.isnull()
```

```
     Temperature  Revenue
0          False    False
1          False    False
2          False    False
3          False    False
4          False    False
..           ...      ...
495        False    False
496        False    False
497        False    False
498        False    False
499        False    False

[500 rows x 2 columns]
```

```
plt.figure(figsize=(10,6))
sns.pairplot(data)
```

```
D:\ANACONDA\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)

<seaborn.axisgrid.PairGrid at 0x1aa88ba4cd0>

<Figure size 1000x600 with 0 Axes>
```

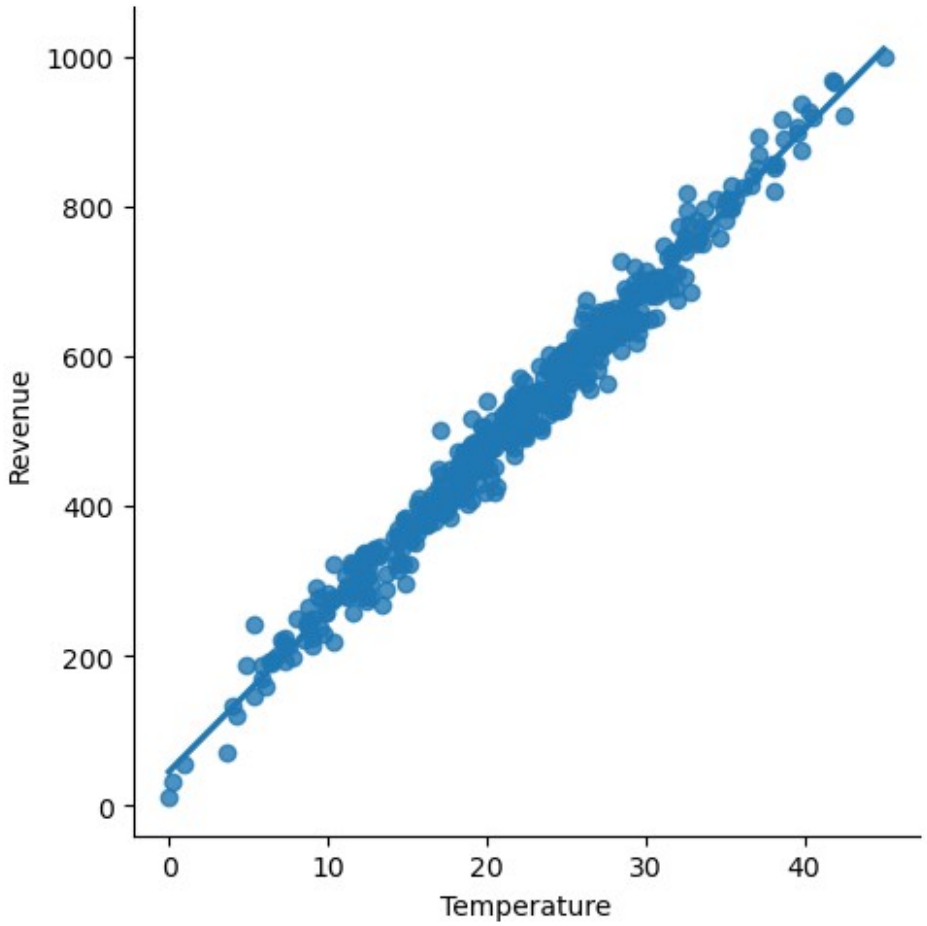



```
plt.figure(figsize=(10,6))
sns.lmplot(x = 'Temperature', y = 'Revenue', data = data )
```

```
D:\ANACONDA\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<seaborn.axisgrid.FacetGrid at 0x1aa87f7eb50>
```

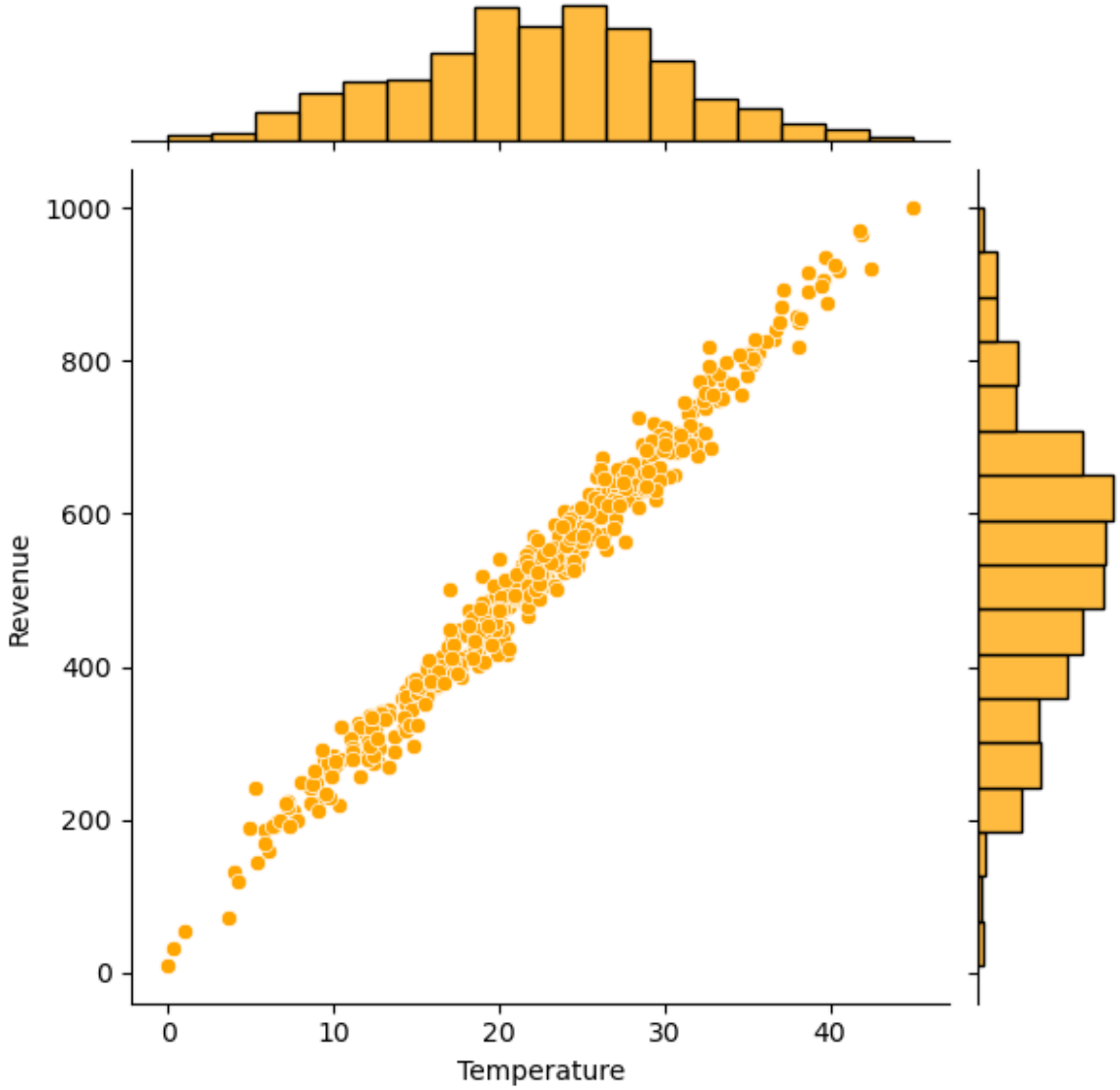
```
<Figure size 1000x600 with 0 Axes>
```



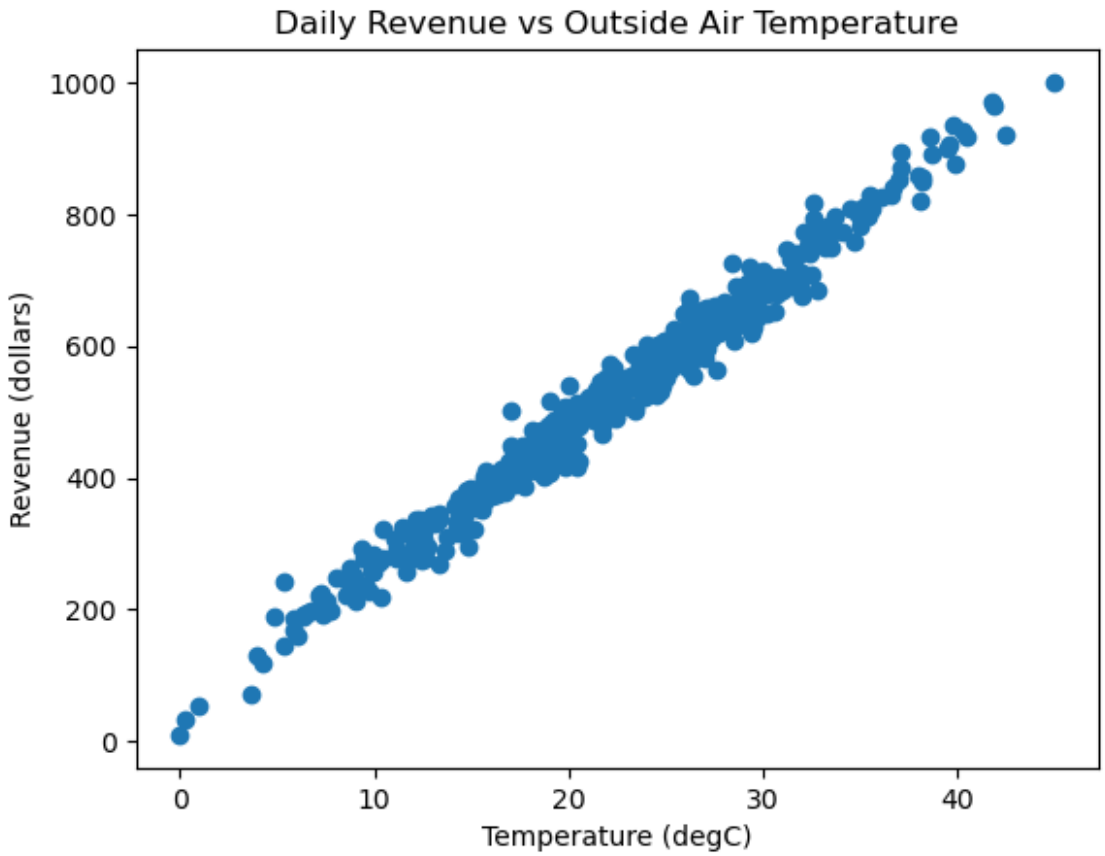

```
plt.figure(figsize=(10,8))
sns.jointplot(x = 'Temperature', y= 'Revenue', data = data, color =
'Orange')
```

```
<seaborn.axisgrid.JointGrid at 0x1aa86ab99d0>
```

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




```
# Plot the data
plt.scatter(data['Temperature'], data['Revenue'])
plt.title('Daily Revenue vs Outside Air Temperature')
plt.xlabel('Temperature (degC)')
plt.ylabel('Revenue (dollars)')
plt.show()
```



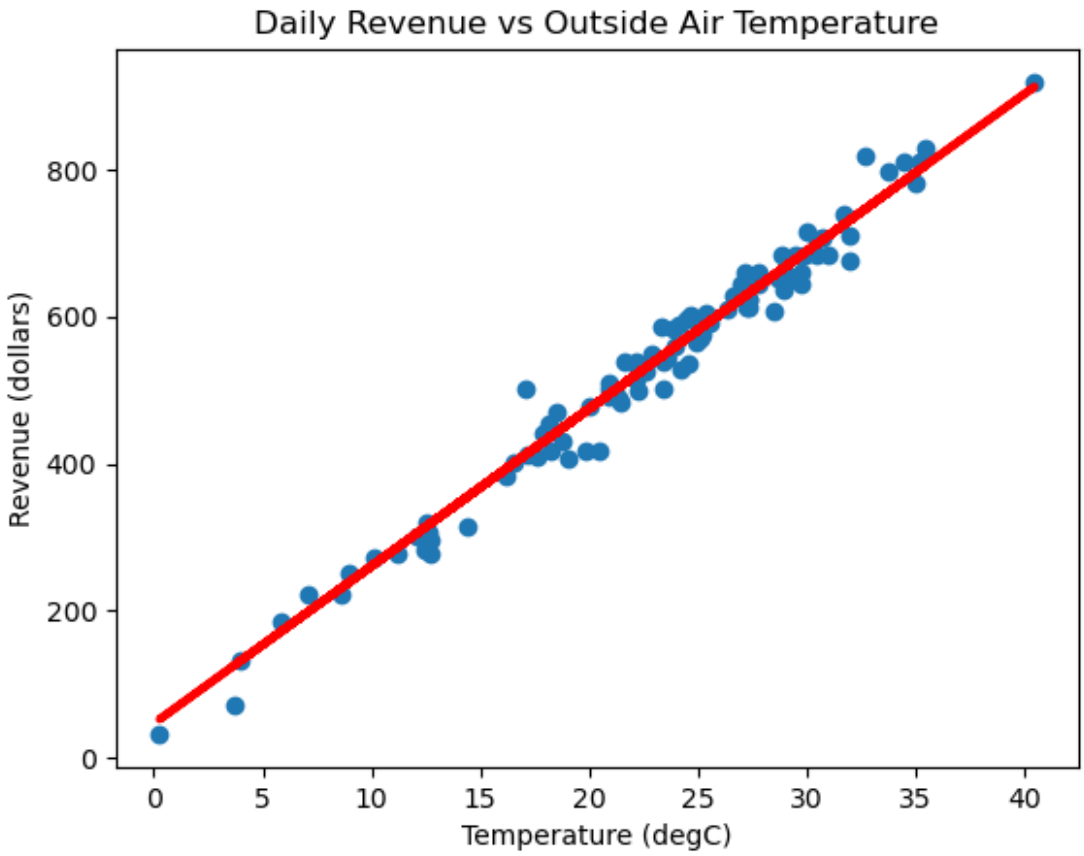

```
#training and testing sets
x = data[['Temperature']]  # Independent variable
y = data['Revenue']        # Dependent variable

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)

#training the Linear Regression model
model = LinearRegression()
model.fit(x_train, y_train)
```

```
LinearRegression()
```

```
# Evaluate the model
y_pred = model.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R-squared Score:', r2)
```

```
Mean Squared Error: 652.5200612979733
R-squared Score: 0.9771532792713993
```

```
# Plot the regression line
plt.scatter(x_test, y_test)
plt.plot(x_test, y_pred, color='red', linewidth=3)
plt.title('Daily Revenue vs Outside Air Temperature')
plt.xlabel('Temperature (degC)')
plt.ylabel('Revenue (dollars)')
plt.show()
```




```
# Make predictions
new_temperature = [[25]]
predicted_revenue = model.predict(new_temperature)
print('Predicted revenue for temperature  25°C:',
predicted_revenue[0])
```

```
Predicted revenue for temperature  25°C: 581.3539876598
```

```
D:\ANACONDA\Lib\site-packages\sklearn\base.py:464: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with
feature names
  warnings.warn(
```

```
# Make predictions
new_temperature = [[70]]
```

```
predicted_revenue = model.predict(new_temperature)
print('Predicted revenue for temperature  70°C:',
predicted_revenue[0])
```

```
Predicted revenue for temperature  70°C: 1543.5428111480774
```

```
D:\ANACONDA\Lib\site-packages\sklearn\base.py:464: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with
feature names
  warnings.warn(
```