Applied Data Science

Assignment 3

Name: Harsh Chawla

Register No: 20BCE0424

Email-id: harsh.chawla2020@vitstudent.ac.in

Campus: Vellore

1. Load the dataset into the tool

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
df = pd.read_csv('Housing.csv')
```

In [3]:
```
df.tail()
```

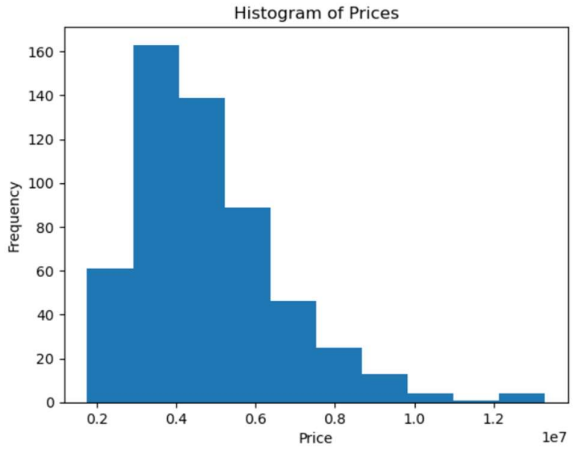
Out[3]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	unfurnished
541	1767150	2400	3	1	1	no	no	no	no	no	0	semi-furnished
542	1750000	3620	2	1	1	yes	no	no	no	no	0	unfurnished
543	1750000	2910	3	1	1	no	no	no	no	no	0	furnished
544	1750000	3850	3	1	2	yes	no	no	no	no	0	unfurnished

2. Perform Below Visualizations.

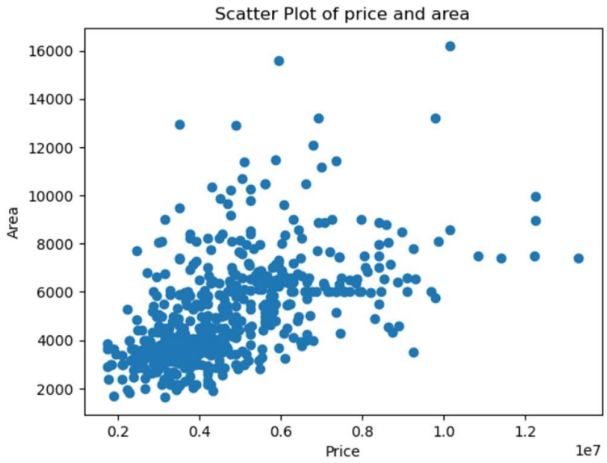
   a. Univariate Analysis

In [4]:
```
plt.hist(df['price'], bins=10)
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.title('Histogram of Prices')
plt.show()
```



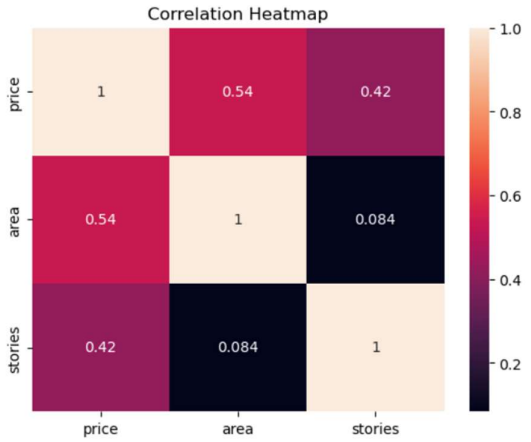
b. Bi-Variate Analysis

In [5]:
```
plt.scatter(df['price'], df['area'])
plt.xlabel('Price')
plt.ylabel('Area')
plt.title('Scatter Plot of price and area')
plt.show()
```



c. Multi-Variate Analysis

In [6]:
```
correlation_matrix = df[['price', 'area', 'stories']].corr()
sns.heatmap(correlation_matrix, annot=True)
plt.title('Correlation Heatmap')
plt.show()
```



3. Perform descriptive statistics on the dataset.

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

Out[7]:

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

4. Check for Missing values and deal with them.

```
In [8]: print(df.isnull().sum())
```

```
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
furnishingstatus  0
dtype: int64
```

No null values found

5. Find the outliers and replace them outliers

```
In [9]: z_scores = np.abs((df['price'] - df['price'].mean()) / df['price'].std())
        threshold = 3
        print(df[z_scores > threshold])
```

```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes
5  10850000  7500         3          3        1      yes        no      yes

  hotwaterheating airconditioning  parking furnishingstatus
0              no             yes        2        furnished
1              no             yes        3        furnished
2              no              no        2   semi-furnished
3              no             yes        3        furnished
4              no             yes        2        furnished
5              no             yes        2   semi-furnished
```

6. Check for Categorical columns and perform encoding.

```
In [10]: label_encoder = LabelEncoder()

         for column in ['mainroad','guestroom','basement','hotwaterheating','airconditioning','furnishingstatus']:
             df[column] = label_encoder.fit_transform(df[column])
```

7. Split the data into dependent and independent variables.

```
In [11]: X = df.drop('price', axis=1)
         y = df['price']
```

8. Scale the independent variables

```
In [12]: scaler = StandardScaler()
         scaled_data = scaler.fit_transform(X)
         scaled_dataframe = pd.DataFrame(scaled_data, columns=X.columns)
```

9. Split the data into training and testing

```
In [13]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

10. Build the Model

```
In [14]: model = LinearRegression()
```

11. Train the Model

```
In [15]: model.fit(X_train, y_train)

Out[15]: LinearRegression()
```

12. Test the Model

```
In [16]: y_pred = model.predict(X_test)
```

13. Measure the performance using Metrics.

```
In [17]: mse = mean_squared_error(y_test, y_pred)
         r2 = r2_score(y_test, y_pred)
         print("Mean Squared Error:", mse)
         print("R-squared:", r2)

         Mean Squared Error: 1653258099588.681
         R-squared: 0.6160919855618678
```

```
In [18]: import matplotlib.pyplot as plt

         plt.scatter(y_test, y_pred)
         plt.xlabel("Actual Values")
         plt.ylabel("Predicted Values")
         plt.title("Scatter Plot of Actual vs. Predicted Values")
         plt.show()
```