Imolementation of a Linear Regression Model to predict the prices of houses based on their square footage and the number of bedrooms and bathrooms.

In [13]:
```
# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import zscore
```

In [14]:
```
# Reading the dataset
df = pd.read_csv('C:/Users/biswa/Downloads/house_price_regression_dataset.c
df.head(3)
```

Out[14]:

	House_Price	Square_Footage	Num_Bedrooms	Num_Bathrooms
0	262382.8523	1360	2	1
1	985260.8545	4272	3	3
2	777977.3901	3592	1	2

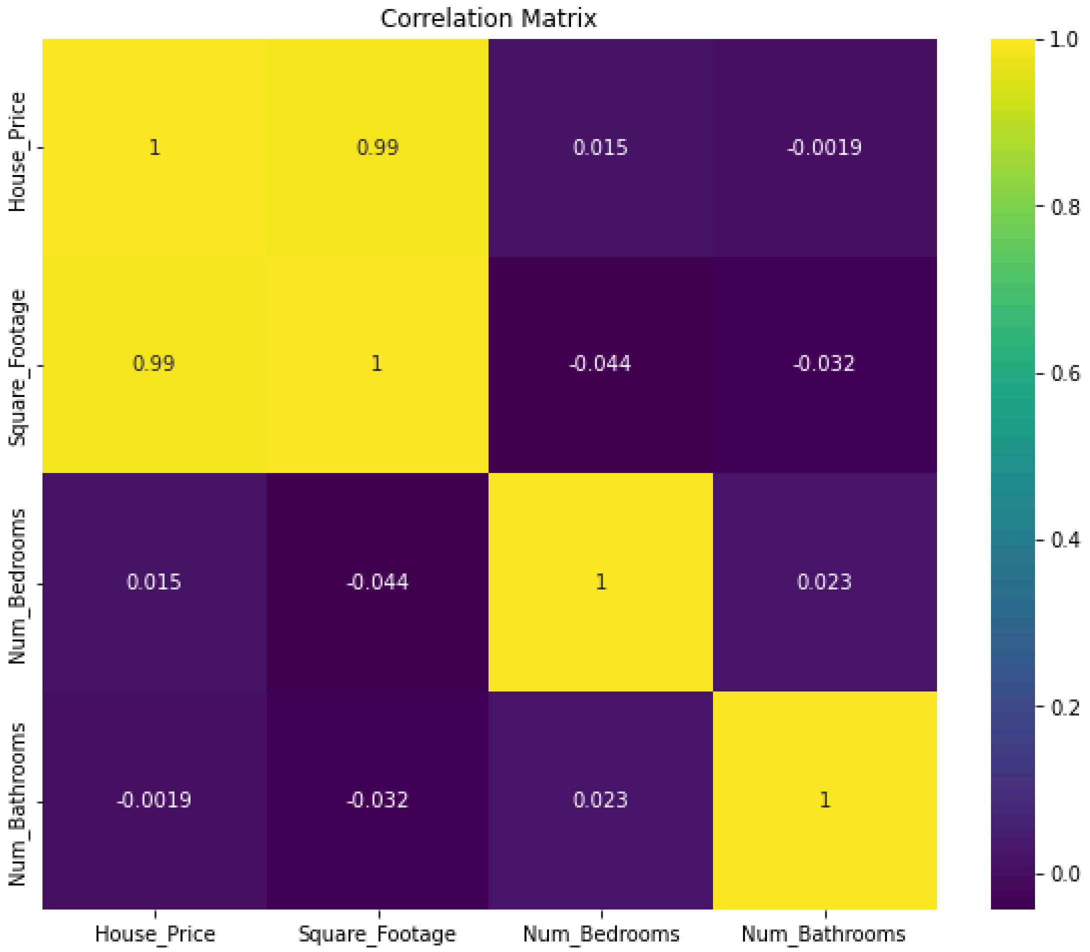
In [15]:
```
# Checking necessary infomation about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   House_Price     1000 non-null   float64
 1   Square_Footage  1000 non-null   int64  
 2   Num_Bedrooms    1000 non-null   int64  
 3   Num_Bathrooms   1000 non-null   int64  
dtypes: float64(1), int64(3)
memory usage: 31.4 KB
```

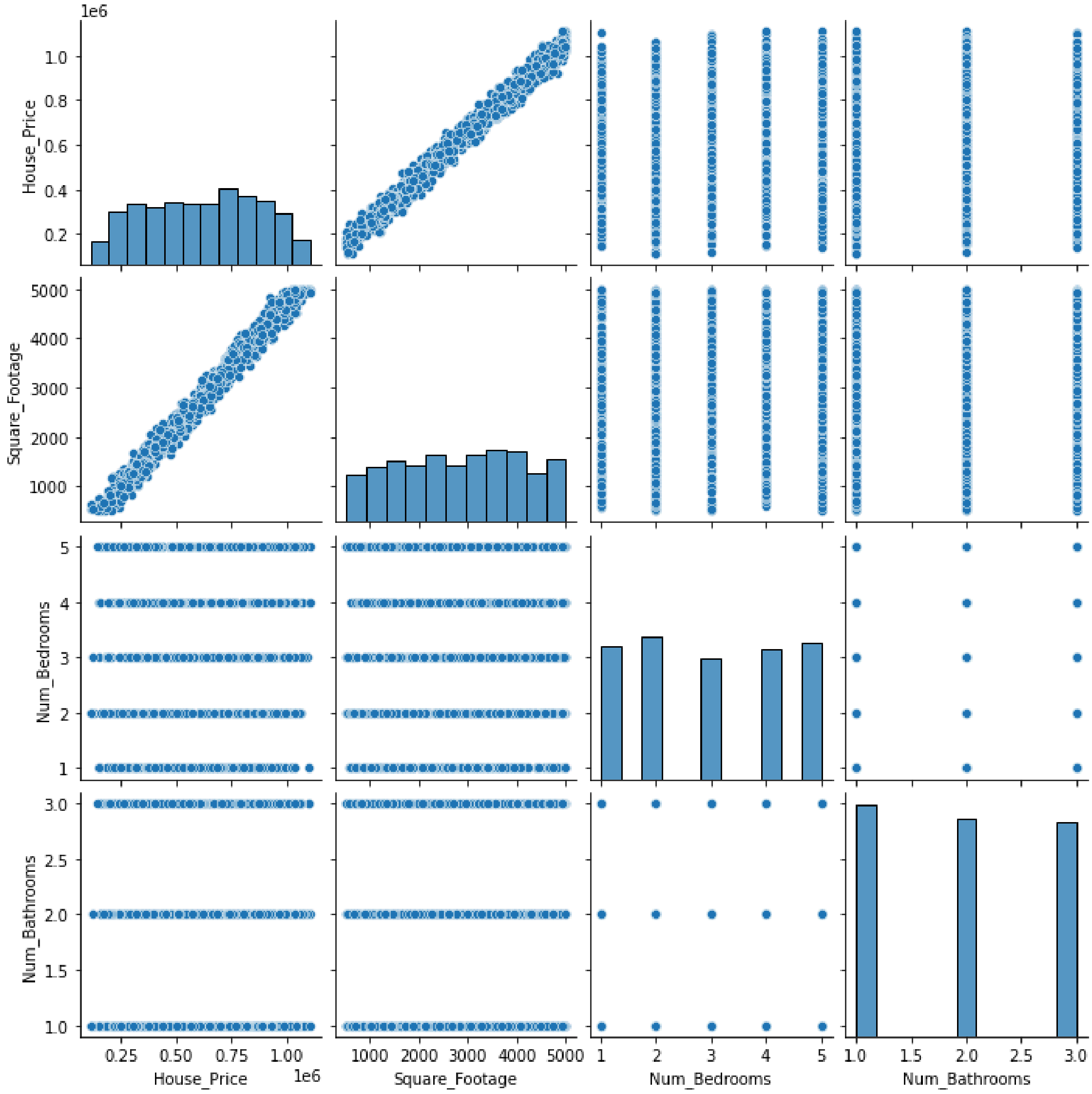
In [16]:
```
#Checking for the null values
df.isnull().sum()
```

Out[16]:
```
House_Price       0
Square_Footage    0
Num_Bedrooms      0
Num_Bathrooms     0
dtype: int64
```

In [17]:
```
# Checking correlation betwwen each variables
correlation_matrix = df.corr()

# heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='viridis')
plt.title('Correlation Matrix')
plt.show()
```

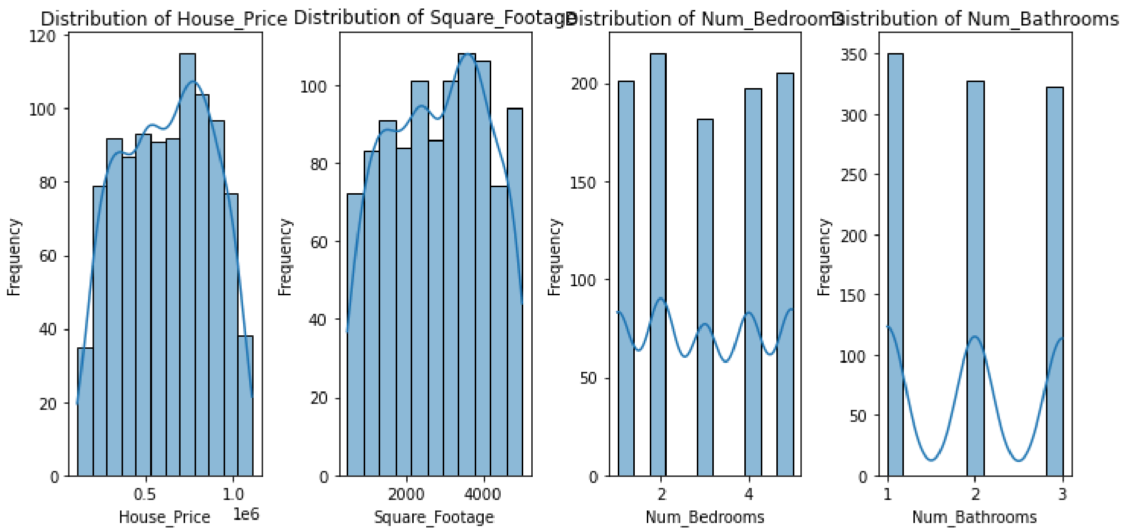


In [18]:
```
# Pair Plots
warnings.filterwarnings('ignore', category=FutureWarning)
sns.pairplot(df)
plt.show()
```



In [19]:
```
# Distributions of each variables
fig, axes = plt.subplots(nrows=int(np.ceil(len(df.columns) / 4)), ncols=4,
for i, col in enumerate(df.columns):
    ax = axes.flatten()[i]

    sns.histplot(data=df, x=col, ax=ax, kde=True)

    skewness = stats.skew(df[col])
    kurtosis = stats.kurtosis(df[col])

    ax.set_title(f"Distribution of {col}")
    ax.set_xlabel(col)
    ax.set_ylabel('Frequency')

    print(f"Column: {col}")
    print(f"Skewness: {skewness:.2f}")
    print(f"Kurtosis: {kurtosis:.2f}")
    print("-" * 20)  # Skeweness here are all tending to zero almost Gauss
plt.tight_layout()
plt.show()
```

```
Column: House_Price
Skewness: -0.06
Kurtosis: -1.09
--------------------
Column: Square_Footage
Skewness: -0.07
Kurtosis: -1.13
--------------------
Column: Num_Bedrooms
Skewness: 0.03
Kurtosis: -1.33
--------------------
Column: Num_Bathrooms
Skewness: 0.05
Kurtosis: -1.51
--------------------
```



In [20]:
```
# Sorting out the target variabl and independent variables
X = df[['Square_Footage', 'Num_Bedrooms', 'Num_Bathrooms']]
y = df['House_Price']

# Spliting the data for the purpose of Training and Testing the Model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [21]:
```
# Standardizing the Data
scaler_X = StandardScaler()
X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)

scaler_y = StandardScaler()
y_train_scaled = scaler_y.fit_transform(y_train.values.reshape(-1, 1)).rave
y_test_scaled = scaler_y.transform(y_test.values.reshape(-1, 1)).ravel()
```

In [22]:
```
# Fitting of Model
model = LinearRegression()
model.fit(X_train_scaled, y_train_scaled)

# Calculating MSE and R^2
y_pred_scaled = model.predict(X_test_scaled)
mse = mean_squared_error(y_test_scaled, y_pred_scaled)
r2 = r2_score(y_test_scaled, y_pred_scaled)

print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")
```

```
Mean Squared Error: 0.013188492093350023
R-squared Score: 0.9868694920703436
```

In [23]:
```
# Model coefficients
print("Model Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Model Coefficients: [0.99417213 0.05669728 0.03008204]
Intercept: 2.1835312917467362e-16
```

It's a good fitted model for this data, siince it has an accuracy of 98.7%

In [7]:
```
```