Project Name: House Price Prediction using Linear Regression

TASK 01

Description: Implement a linear regression model to predict the prices of houses based on their square footage and the number of bedrooms and bathrooms.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/d
```

```
cd /content/drive/MyDrive/PP/TASK01PDG
```

```
/content/drive/MyDrive/PP/TASK01PDG
```

```
import pandas as pd
```

```
df=pd.read_csv("/content/drive/MyDrive/PP/TASK01PDG/House Prices.csv")
```

Studying Data

```
df.head()
```

	ID	Date	Price	Bedrooms	Bathrooms	Sqft_livi
0	1	20140916T000000	280000.0	6	3.00	2400
1	2	20150422T000000	300000.0	6	3.00	2400
2	3	20140508T000000	647500.0	4	1.75	2060
3	4	20140811T000000	400000.0	3	1.00	1460
4	5	20150401T000000	235000.0	3	1.00	1430

5 rows × 21 columns

```
df.describe()
```

	ID	Price	Bedrooms	Bathrooms	Sq
count	10147.000000	1.014700e+04	10147.000000	10147.000000	10
mean	5074.000000	5.447467e+05	3.352321	2.114418	2(
std	2929.330925	3.719381e+05	0.960354	0.791662	9
min	1.000000	7.500000e+04	0.000000	0.000000	3
25%	2537.500000	3.200000e+05	3.000000	1.500000	14
50%	5074.000000	4.499000e+05	3.000000	2.250000	19
75%	7610.500000	6.500000e+05	4.000000	2.500000	25
max	10147.000000	5.570000e+06	33.000000	8.000000	13

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10147 entries, 0 to 10146
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   ID             10147 non-null  int64  
 1   Date           10147 non-null  object 
 2   Price          10147 non-null  float64
 3   Bedrooms       10147 non-null  int64  
 4   Bathrooms      10147 non-null  float64
 5   Sqft_living    10146 non-null  float64
 6   Sqft_lot       10146 non-null  float64
 7   Floors         10146 non-null  float64
 8   Waterfront     10146 non-null  float64
 9   View           10146 non-null  float64
 10  Condition      10146 non-null  float64
 11  Grade          10146 non-null  float64
 12  Sqft_above     10146 non-null  float64
 13  Sqft_basement  10146 non-null  float64
 14  Yr_built       10146 non-null  float64
 15  Yr_renovated   10146 non-null  float64
 16  zipcode        10146 non-null  float64
 17  Lat            10146 non-null  float64
 18  Long           10146 non-null  float64
 19  Sqft_living15  10146 non-null  float64
 20  Sqft_lot15     10146 non-null  float64
dtypes: float64(18), int64(2), object(1)
memory usage: 1.6+ MB
```

Defining X & Y

```
df.columns
```

```
Index(['ID', 'Date', 'Price', 'Bedrooms', 'Bathrooms', 'Sqft_living',
       'Sqft_lot', 'Floors', 'Waterfront', 'View', 'Condition', 'Grade',
       'Sqft_above', 'Sqft_basement', 'Yr_built', 'Yr_renovated', 'zipcode',
       'Lat', 'Long', 'Sqft_living15', 'Sqft_lot15'],
      dtype='object')
```

```
y=df['Price']
X=df[[ 'Sqft_living','Bedrooms', 'Bathrooms']]
```

```
from sklearn.model_selection import train_test_split
```

Dividing data into Train-Test

```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=2529)
```

```
X.shape,X_train.shape,X_test.shape
```

```
((10147, 3), (7610, 3), (2537, 3))
```

```
y.shape,y_train.shape,y_test.shape
```

```
((10147,), (7610,), (2537,))
```

Removing rows having NULLS

```
X_train = X_train.dropna()
y_train = y_train[X_train.index]
```

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

Training Model

```
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
y_pred=model.predict(X_test)
```

```
# step8: evaluation
from sklearn.metrics import mean_absolute_percentage_error, mean_absolute_error, mean_squared_error
mean_absolute_percentage_error(y_test,y_pred)
```

```
0.35555954248017896
```

```
mean_squared_error(y_test,y_pred)
```

```
57102246405.82373
```

Predicting house price by giving parameters like bedrooms,bathrooms, & Square Footage.

```
def predict_price(bedrooms, bathrooms, sqft_living):
  """
  Predicts the price of a house based on its square footage, number of bedrooms, and number of bathrooms.

  Args:
    bedrooms: The number of bedrooms in the house.
    bathrooms: The number of bathrooms in the house.
    sqft_living: The square footage of the house.

  Returns:
    The predicted price of the house.
  """
  # Create a DataFrame with the input values.
  input_data = pd.DataFrame({
      'Sqft_living': [sqft_living],
      'Bedrooms': [bedrooms],
      'Bathrooms': [bathrooms],

  })

  # Predict the price of the house using the model.
  predicted_price = model.predict(input_data)[0]

  # Return the predicted price.
  return predicted_price

# Example usage:

sqft_living=int(input("Enter Square footage: "))
bedrooms=int(input("Enter number of bedrooms: "))
bathrooms=int(input("Enter number of bathrooms: "))
predicted_price = predict_price(bedrooms, bathrooms, sqft_living)
print(f"Predicted price: {predicted_price}")
```

```
Enter Square footage: 2400
Enter number of bedrooms: 6
Enter number of bathrooms: 3
Predicted price: 523735.83300729224
```

Relation between Sq_ft & House Price accordingly

```
import seaborn as sns
sns.regplot(x='Sqft_living', y='Price' ,data = df,fit_reg=True)
```

<Axes: xlabel='Sqft_living', ylabel='Price'>