```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'pakistan-house-price-prediction:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F1753715%2F286

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')



# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a versi
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/pakistan-house-price-prediction/Entities.csv
```

```
import pandas as pd
house_data=pd.read_csv('/kaggle/input/pakistan-house-price-prediction/Entities.csv')
house_df=pd.DataFrame(house_data)
house_df.head()
```

	Unnamed: 0	property_id	location_id	page_
0	0	237062	3325	https://www.zameen.com/Property/g_10_g_10_2_
1	1	346905	3236	https://www.zameen.com/Property/e_11_2_servi
2	2	386513	764	https://www.zameen.com/Property/islamabad_g_
3	3	656161	340	https://www.zameen.com/Property/islamabad_ba
4	4	841645	3226	https://www.zameen.com/Property/dha_valley_d

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
house_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168446 entries, 0 to 168445
Data columns (total 18 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Unnamed: 0     168446 non-null  int64
 1   property_id    168446 non-null  int64
 2   location_id    168446 non-null  int64
 3   page_url       168446 non-null  object
 4   property_type  168446 non-null  object
 5   price          168446 non-null  int64
 6   location       168446 non-null  object
 7   city           168446 non-null  object
 8   province_name  168446 non-null  object
 9   latitude       168446 non-null  float64
 10  longitude      168446 non-null  float64
 11  baths          168446 non-null  int64
 12  purpose        168446 non-null  object
 13  bedrooms       168446 non-null  int64
 14  date_added     168446 non-null  object
 15  agency         124375 non-null  object
 16  agent          124374 non-null  object
 17  Total_Area     168446 non-null  float64
dtypes: float64(3), int64(6), object(9)
memory usage: 23.1+ MB
```

```
house_df.describe()
```

	Unnamed: 0	property_id	location_id	price	latitude	l
count	168446.000000	1.684460e+05	168446.000000	1.684460e+05	168446.000000	168
mean	84222.500000	1.559626e+07	4375.936395	1.776576e+07	29.859519	
std	48626.316059	2.251207e+06	3776.561581	3.531003e+07	3.807870	
min	0.000000	8.657500e+04	1.000000	0.000000e+00	11.052446	
25%	42111.250000	1.488320e+07	1058.000000	1.750000e+05	24.948536	
50%	84222.500000	1.665851e+07	3286.000000	8.500000e+06	31.459784	
75%	126333.750000	1.708662e+07	7220.000000	1.950000e+07	33.560887	
max	168445.000000	1.735772e+07	14220.000000	2.000000e+09	73.184088	

Data cleaning

```
house_df.isnull().sum()
```

```
Unnamed: 0            0
property_id           0
location_id           0
page_url              0
property_type         0
price                 0
location              0
city                  0
province_name         0
latitude              0
longitude             0
baths                 0
purpose               0
bedrooms              0
date_added            0
agency            44071
agent             44072
Total_Area            0
dtype: int64
```

```
house_df.columns
```

```
Index(['Unnamed: 0', 'property_id', 'location_id', 'page_url', 'property_type',
       'price', 'location', 'city', 'province_name', 'latitude', 'longitude',
       'baths', 'purpose', 'bedrooms', 'date_added', 'agency', 'agent',
       'Total_Area'],
      dtype='object')
```

```
table1=house_df.drop(['Unnamed: 0', 'property_id', 'location_id', 'page_url','agency','agent'],axis=1)
```

```
table1.head()
```

	property_type	price	location	city	province_name	latitude	long
0	Flat	10000000	G-10	Islamabad	Islamabad Capital	33.679890	73.0
1	Flat	6900000	E-11	Islamabad	Islamabad Capital	33.700993	72.9
2	House	16500000	G-15	Islamabad	Islamabad Capital	33.631486	72.9
					Islamabad		

```
table1.columns
```

```
Index(['property_type', 'price', 'location', 'city', 'province_name',
       'latitude', 'longitude', 'baths', 'purpose', 'bedrooms', 'date_added',
       'Total_Area'],
      dtype='object')
```

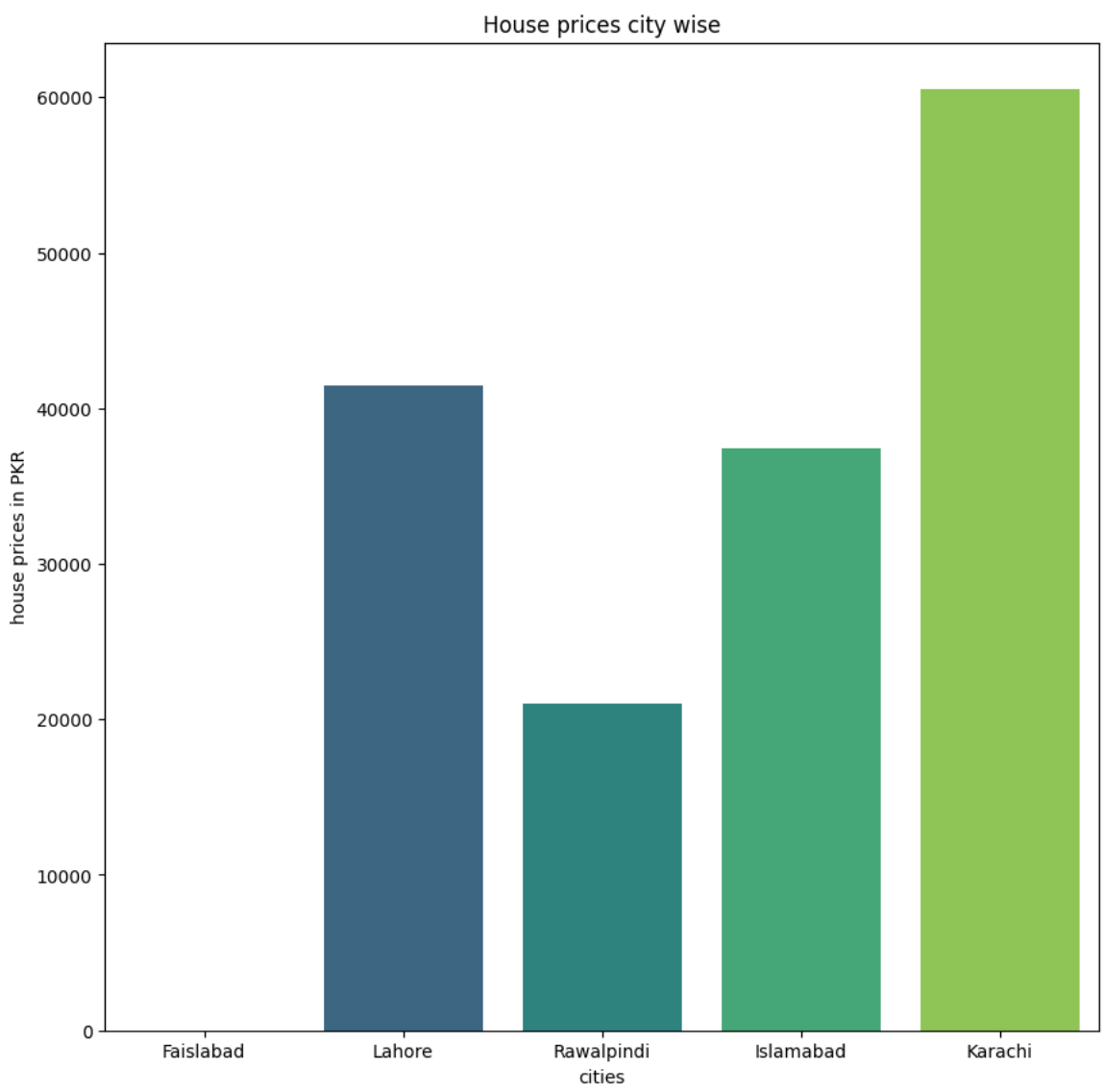
After dropping irrelevant columns we are left with 12 important data columns

Creating a bar chart for house prices according to cities

```
table1["city"].unique()
```

```
array(['Islamabad', 'Lahore', 'Faisalabad', 'Rawalpindi', 'Karachi'],
      dtype=object)
```

```
plt.figure(figsize=(10,10))
sns.countplot(x=table1["city"],palette="viridis",order=["Faislabad","Lahore","Rawalpindi","Islamabad","Karachi"])
plt.title("House prices city wise")
plt.xlabel("cities")
plt.ylabel("house prices in PKR")
plt.show()
```




```
table1.columns
```

```
Index(['property_type', 'price', 'location', 'city', 'province_name',
       'latitude', 'longitude', 'baths', 'purpose', 'bedrooms', 'date_added',
       'Total_Area'],
      dtype='object')
```

Start coding or generate with AI.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
import pandas as pd
new_data=pd.read_csv('/kaggle/input/pakistan-house-price-prediction/Entities.csv')
new_data_df=pd.DataFrame(new_data)
new_data_df.head()
```

	Unnamed: 0	property_id	location_id	page_url	property_type	price	location	city
0	0	237062	3325	https://www.zameen.com/Property/g_10_g_10_2_gr...	Flat	10000000	G-10	Islamabac
1	1	346905	3236	https://www.zameen.com/Property/e_11_2_service...	Flat	6900000	E-11	Islamabac
2	2	386513	764	https://www.zameen.com/Property/islamabad_g_15...	House	16500000	G-15	Islamabac
3	3	656161	340	https://www.zameen.com/Property/islamabad_bani...	House	43500000	Bani Gala	Islamabac
4	4	841645	3226	https://www.zameen.com/Property/dha_valley_dha...	House	7000000	DHA Defence	Islamabac

```
table2=new_data_df.drop(['Unnamed: 0', 'property_id', 'location_id', 'page_url','agency','agent'],axis=1)
```

```
table2.head()
```

	property_type	price	location	city	province_name	latitude	longitude	baths	purpose	bedrooms	date_added	Tot
0	Flat	10000000	G-10	Islamabad	Islamabad Capital	33.679890	73.012640	2	For Sale	2	2/4/2019	
1	Flat	6900000	E-11	Islamabad	Islamabad Capital	33.700993	72.971492	3	For Sale	3	5/4/2019	1
2	House	16500000	G-15	Islamabad	Islamabad Capital	33.631486	72.926559	6	For Sale	5	7/17/2019	
					Islamabad							

```
ordinal_map = {'Faisalabad':1,
               'Rawalpindi':2,
               'Islamabad':3,
               'Karachi':4,
               'Lahore':5

}

table2['city_new'] = table2.city.map(ordinal_map)
table2.head()
```

	property_type	price	location	city	province_name	latitude	longitude	baths	purpose	bedrooms	date_added	Tot
0	Flat	10000000	G-10	Islamabad	Islamabad Capital	33.679890	73.012640	2	For Sale	2	2/4/2019	
1	Flat	6900000	E-11	Islamabad	Islamabad Capital	33.700993	72.971492	3	For Sale	3	5/4/2019	1
2	House	16500000	G-15	Islamabad	Islamabad Capital	33.631486	72.926559	6	For Sale	5	7/17/2019	
3	House	43500000	Bani Gala	Islamabad	Islamabad Capital	33.707573	73.151199	4	For Sale	4	4/5/2019	1
4	House	7000000	DHA Defence	Islamabad	Islamabad Capital	33.492591	73.301339	3	For Sale	3	7/10/2019	

```
ordinal_map={"Flat":1,
             "House":2,
             "Room":3,
             'Lower Portion':4,
             'Upper Portion':5,
             'Penthouse':6,
             'Farm House':7

}
table2["property_encoded"]=table2.property_type.map(ordinal_map)
```

```
ordinal_map={'Islamabad Capital':1,
             'Punjab':2,
              'Sindh':3
```

```
}
table2["province_encoded"]=table2.province_name.map(ordinal_map)



ordinal_map={'For Sale':1,
             'For Rent':2,


}
table2["purpose_encoded"]=table2.purpose.map(ordinal_map)
table2.head()
```

	property_type	price	location	city	province_name	latitude	longitude	baths	purpose	bedrooms	date_added	Tot
0	Flat	10000000	G-10	Islamabad	Islamabad Capital	33.679890	73.012640	2	For Sale	2	2/4/2019	
1	Flat	6900000	E-11	Islamabad	Islamabad Capital	33.700993	72.971492	3	For Sale	3	5/4/2019	1
2	House	16500000	G-15	Islamabad	Islamabad Capital	33.631486	72.926559	6	For Sale	5	7/17/2019	
3	House	43500000	Bani Gala	Islamabad	Islamabad Capital	33.707573	73.151199	4	For Sale	4	4/5/2019	1
4	House	7000000	DHA Defence	Islamabad	Islamabad Capital	33.492591	73.301339	3	For Sale	3	7/10/2019	

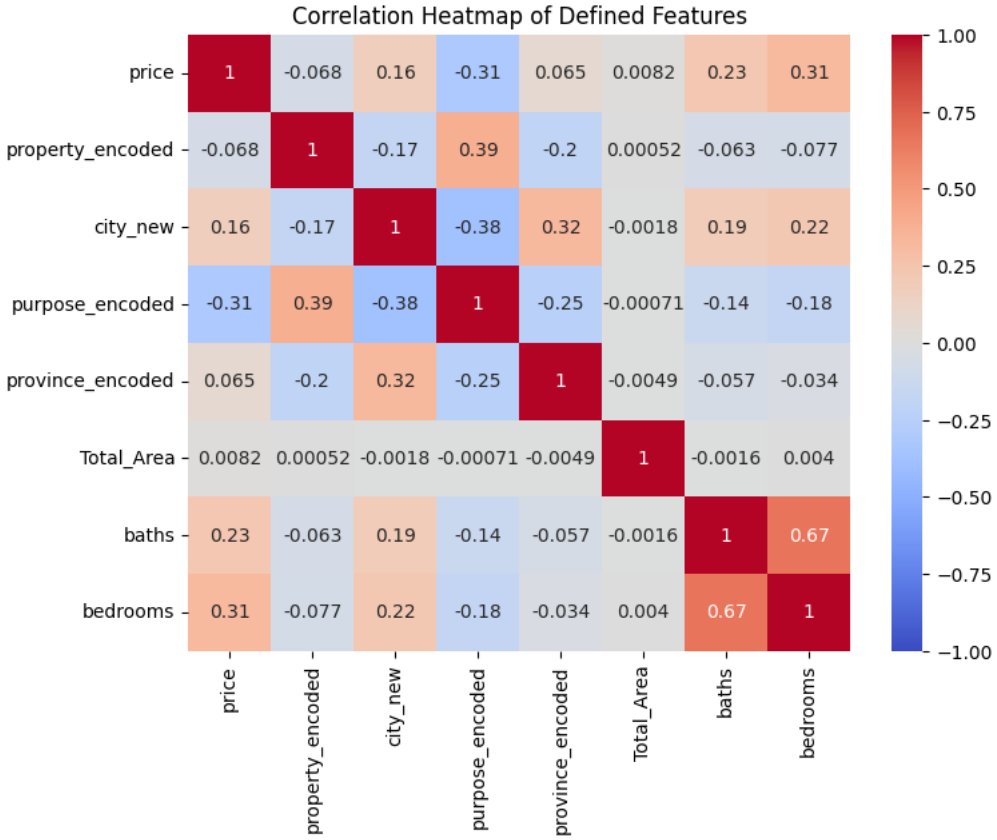
```
table2.columns
```

```
Index(['property_type', 'price', 'location', 'city', 'province_name',
       'latitude', 'longitude', 'baths', 'purpose', 'bedrooms', 'date_added',
       'Total_Area', 'city_new', 'property_encoded', 'province_encoded',
       'purpose_encoded'],
      dtype='object')
```

```
features = ['price', 'property_encoded', 'city_new', 'purpose_encoded', 'province_encoded',  'Total_Area', 'baths', 'bedrooms

# Calculate the correlation matrix
corr_matrix = table2[features].corr()

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap of Defined Features')
plt.show()
```




```
print(corr_matrix['price'].sort_values(ascending=False))
```

```
price                1.000000
bedrooms             0.314145
baths                0.230205
city_new             0.164977
province_encoded     0.064750
Total_Area           0.008168
property_encoded    -0.068345
purpose_encoded     -0.314965
Name: price, dtype: float64
```

So our correlation matrix shows that the factors which affects the prices of property w.r.t to pakistan are foremost, number of bedrooms, no of bathrooms and the city in which they are located.Price hikes are observed usually on these measures , contrary to general practice , totalcovered area , province , type of property and purpose were not found to be the defining features!!

```
# Splitting the data into training and testing sets
x=table2.drop(["price"],axis=1)
y=table2["price"]
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=42)
```

```
table2.columns
```

```
Index(['property_type', 'price', 'location', 'city', 'province_name',
       'latitude', 'longitude', 'baths', 'purpose', 'bedrooms', 'date_added',
       'Total_Area', 'city_new', 'property_encoded', 'province_encoded',
       'purpose_encoded'],
      dtype='object')
```

```
table2_updated=table2.drop(["property_type","city","province_name","purpose","location","date_added","latitude","longitude"]
table2_updated.head()
```

	price	baths	bedrooms	Total_Area	city_new	property_encoded	province_encoded	purpose_encoded
0	10000000	2	2	1089.004	3	1	1	1
1	6900000	3	3	15246.056	3	1	1	1
2	16500000	6	5	2178.008	3	2	1	1
3	43500000	4	4	10890.000	3	2	1	1
4	7000000	3	3	2178.008	3	2	1	1

```
# Splitting the data into training and testing sets
X=table2_updated.drop(["price"],axis=1)
y=table2_updated["price"]
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.2,random_state=42)


from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.fit_transform(X_test)
X_train_scaled.shape,X_test_scaled.shape
```

```
((134756, 7), (33690, 7))
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np


LR=LinearRegression()
LR.fit(X_train_scaled,y_train)
y_pred=LR.predict(X_test_scaled)
mse=mean_squared_error(y_test,y_pred)
rmse=np.sqrt(mse)
r2=r2_score(y_test,y_pred)
mse,rmse,r2
```

```
(953053559694035.5, 30871565.553013917, 0.17777773711957512)
```

```
y_pred
```

```
array([38673053.40711106,  1199406.60791383, 28306949.50050325, ...,
       33331699.62433267,  7322261.19188414, 14500526.36401242])
```

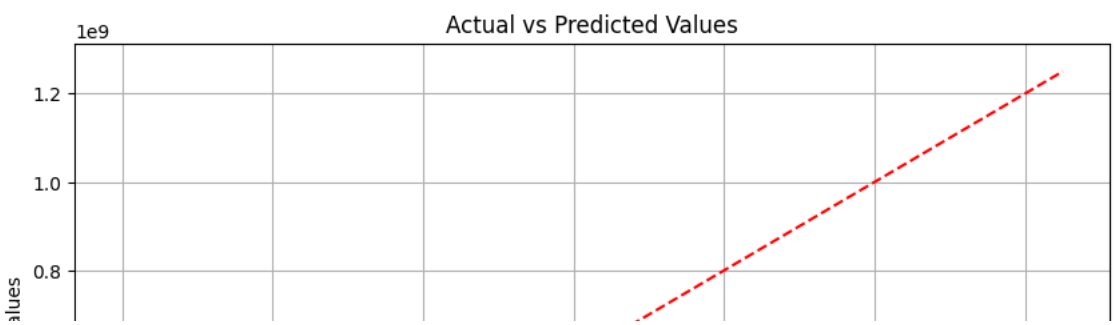
```
y_pred
```

```
array([38673053.40711106,  1199406.60791383, 28306949.50050325, ...,
       33331699.62433267,  7322261.19188414, 14500526.36401242])
```

```
y_test
```

```
49309      87500000
42561         55000
1398       21800000
81279       6500000
18347         75000
            ...
73327      38000000
157728     59000000
156404    175000000
129725       140000
112380       200000
Name: price, Length: 33690, dtype: int64
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.title('Actual vs Predicted Values')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.grid(True)
plt.show()
```




```
table2_updated.columns
```

```
Index(['price', 'baths', 'bedrooms', 'Total_Area', 'city_new',
       'property_encoded', 'province_encoded', 'purpose_encoded'],
      dtype='object')
```

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
import numpy as np

features = ['baths', 'bedrooms', 'Total_Area', 'property_encoded', 'province_encoded', 'purpose_encoded', 'city_new']
target = 'price'

# Example feature matrix X and target vector y
X = table2_updated[features]
y = table2_updated[target]

# Initialize the model
LR = LinearRegression()

# Cross-validation
cv_scores = cross_val_score(LR, X, y, cv=10, scoring='neg_mean_squared_error')

# Calculate the Mean RMSE
mean_rmse = np.sqrt(-cv_scores.mean())
print(f"Mean CV RMSE: {mean_rmse}")
```

```
Mean CV RMSE: 32274280.774695672
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV, cross_val_score
import numpy as np

# Initialize a more complex model (e.g., Random Forest)
model = RandomForestRegressor()

# Define a parameter grid for hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
```