Welcome to the EDA stage of Zameen.gone

Data Cleaning:

In [71]:
```
import pandas as pd
import numpy as np
import re
import seaborn as sns
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import sklearn.linear_model as sl
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import LassoCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import log_loss
from sklearn.preprocessing import StandardScaler
from mpl_toolkits.mplot3d import Axes3D
```

In [72]:
```
# First, we must merge the CSV files of every city:
df1 = pd.read_csv("islamabad.csv")
df2 = pd.read_csv("lahore.csv")
df3 = pd.read_csv("karachi.csv")
df4 = pd.read_csv("eleven.csv") #  9 smaller cities together
df5 = pd.read_csv("nine.csv")   # 11 smaller cities together
df = pd.concat([df1, df2, df3, df4, df5], ignore_index=True)
df["city"].size
```

Out[72]: 87013

We have **23 cities** in total Abottabad, Attock, Bahawalpur, Dera Ghazi Khan, Faisalabad, Gujranwala, Gujrat, Hyderabad, Islamabad, Jhelum, Karachi, Lahore, Multan, Murree, Okara, Peshawar, Quetta, Rawalpindi, Sahiwal, Sargodha, Sheikhupura, Sialkot, Wah.

`city` : (str) The city where the property is located

`location` : (str) Sub-area in city

`price` : (int) Price of the property

`bedrooms` : (int) No. of bedrooms

`baths` : (int) No. of bathrooms

`size` : (int) area of property in square-feet

`title` : (str) Title shown on the listing

`page_url` : (str) Link to the ad (some NaN values)

`date_added` : (string) Date ad was uploaded. Will convert to Unix time during EDA.

property_type: (str) House/flat/etc

latitude: (float) lat coordinate

longitude: (float) lng coordinate

number_of_photos: (int) How many pics added

province: (str) Punjab/Sindh/etc.

agency: (str) Real estate agency managing it

purpose: (str) Rent or Buy?

Zameen.com does not display the exact day/time a listing was added, but rather just how long ago e.g., "1 hour ago", or "3 days ago". So we converted it to Unix timestamp, relative to when we ran our web scraper.

Helper function for converting time:

In [73]:
```
# converts into seconds
def clean_date(added_str):
    time_str = added_str.replace("Added: ", "").replace(" ago", "").strip()
    number, period = time_str.split()
    if "day" in period:
        return pd.Timedelta(days=int(number)).total_seconds()
    elif "week" in period:
        return pd.Timedelta(weeks=int(number)).total_seconds()
    elif "month" in period:
        return pd.Timedelta(days=int(number) * 30).total_seconds()
    elif "hour" in period:
        return pd.Timedelta(hours=int(number)).total_seconds()
    elif "minute" in period:
        return pd.Timedelta(minutes=int(number)).total_seconds()
    else:
        raise ValueError(f"Unknown time period: {period}")
```

Helper function to calculate title's length:

In [74]:
```
def string_length(title):
    return len(title.split())
```

In [75]:
```
print(df["purpose"].unique())
```

```
['Buy']
```

It turns out Zameen.com does not list properties for Rent by default, which is why our Web-scraper ended up only gathering properties available to Buy. Which makes the column `purpose` of no use anymore:

In [76]:
```
df = df.drop(columns = 'purpose')
```

Dropping NaN and duplicate rows:

In [77]:
```
df = df.dropna()
df = df.drop_duplicates()
```

Also dropping `page_url` as we no longer have any use for it:

In [78]:
```
df = df.drop(columns = 'page_url')
```

We also decided to make `property_id` the dataframe index as it is unique to every listing on Zameen.com.

In [79]:
```
df.reset_index(drop = True, inplace = True)
df.index = df.index + 1
df.index.name = "property_id"
```

Simplifying sub-area names: removing commas.

In [80]:
```
df['location'] = df['location'].apply(lambda x: x.split(',')[0] if ',' in x else x)
```

Reformat date_added:

In [81]:
```
df['date_added'] = df['date_added'].apply(clean_date)
```

In [82]:
```
# POSIX timestamp for 10th Nov, 2023
# the day and time we extracted our data
posix_time = 1699646400

# Adjust each value in each column
df['timestamp_adjusted'] = posix_time - df['date_added']
df['new_time'] = pd.to_datetime(df['date_added'], unit='s')
df.sample(3)
```

Out[82]:

property_id	city	location	price	bedrooms	baths	size	title	date_added	property_type	latitude	longitude	number_of_photos	province	agency	ti
31252	Karachi	Bahria Town - Precinct 10-A	26000000	3	4	1800.0	200 Square Yard Villa For Sale In Precinct 10-...	345600.0	Houses_Property	25.054734	67.309664	14	Sindh	UPN (Universal Property Network)	
22351	Lahore	DHA Phase 6 - Block M	95000000	5	6	4500.0	1 kanal Brand new House in phase 6 near Golf c...	2592000.0	Houses_Property	31.478672	74.473186	28	Punjab	ESTATE LINES	
19357	Lahore	Military Accounts Housing Society	14800000	3	4	900.0	Beautiful 4 Marla Double Story House For Sale	518400.0	Houses_Property	31.421266	74.283972	25	Punjab	Punjab Property Point Real Estate Builders	

Add a column storing each title's length (to be used in a later analysis):

In [83]:
```
df['title_length'] = df['title'].apply(string_length)
```

Renaming columns to better reflect their new data:

In [84]:
```
columns_renamed = {
    'property_id': 'property_id',
    'city': 'city',
    'location': 'location',
    'price': 'price',
    'bedrooms': 'bedrooms',
    'baths': 'bathrooms',
    'size': 'size_sqft',
    'title': 'title_text',
    'page_url': 'page_url',
    'date_added': 'timestamp',
    'property_type': 'property_type',
    'latitude': 'latitude',
    'longitude': 'longitude',
    'number_of_photes': 'num_photos',
    'province': 'province',
    'new_time': 'formatted_time',
    'agency': 'agency'
}

df.rename(columns = columns_renamed, inplace = True)

df.sample()
```

Out[84]:

property_id	city	location	price	bedrooms	bathrooms	size_sqft	title_text	timestamp	property_type	latitude	longitude	number_of_photos	province
50719	Bahawalpur	DHA Defence - Villa Community	19500000	3	3	2700.0	Get In Touch Now To Buy A House In Bahawalpur	14400.0	Houses_Property	29.320482	71.665174	6	Punjab

Encoding `property_type`, `city` and `province` as integers:

In [85]:
```
# df['property_type'] = df['property_type'].map({
#     'Houses_Property': 0,
#     'Flats_Apartments': 1,
#     'Farm_Houses': 2,
#     'Lower_Portions': 3,
#     'Penthouse': 4,
#     'Rooms': 5,
#     'Upper_Portions': 6
# })

# df['city'] = df['city'].map({
#     'Islamabad': 0,
#     'Lahore': 1,
#     'Karachi': 2,
#     'Rawalpindi': 3,
#     'Sheikhupura': 4,
#     'Murree': 5,
#     'Gujranwala': 6,
#     'Abbottabad': 7,
#     'Wah': 8,
#     'Okara': 9,
#     'Sialkot': 10,
#     'Sargodha': 11,
#     'Sahiwal': 12,
#     'Attock': 13,
#     'Multan': 14,
#     'Faisalabad': 15,
#     'Peshawar': 16,
#     'Quetta': 17,
#     'Jhelum': 18,
#     'Gujrat': 19,
#     'Bahawalpur': 20,
#     'Dera_Gazi_Khan': 21,
#     'Hyderabad': 22
# })

# df['province'] = df['province'].map({
#     'Islamabad Capital': 0,
#     'Punjab': 1,
#     'Sindh': 2,
#     'Khyber Pakhtunkhwa': 3,
#     'Balochistan': 4
# })
```

In [86]:
```
df.columns
```

Out[86]:
```
Index(['city', 'location', 'price', 'bedrooms', 'bathrooms', 'size_sqft',
       'title_text', 'timestamp', 'property_type', 'latitude', 'longitude',
       'number_of_photos', 'province', 'agency', 'timestamp_adjusted',
       'formatted_time', 'title_length'],
      dtype='object')
```

In [87]:
```
df.size
```

Out[87]:
```
867017
```

This concludes our Data cleaning process.

EDA starts here:

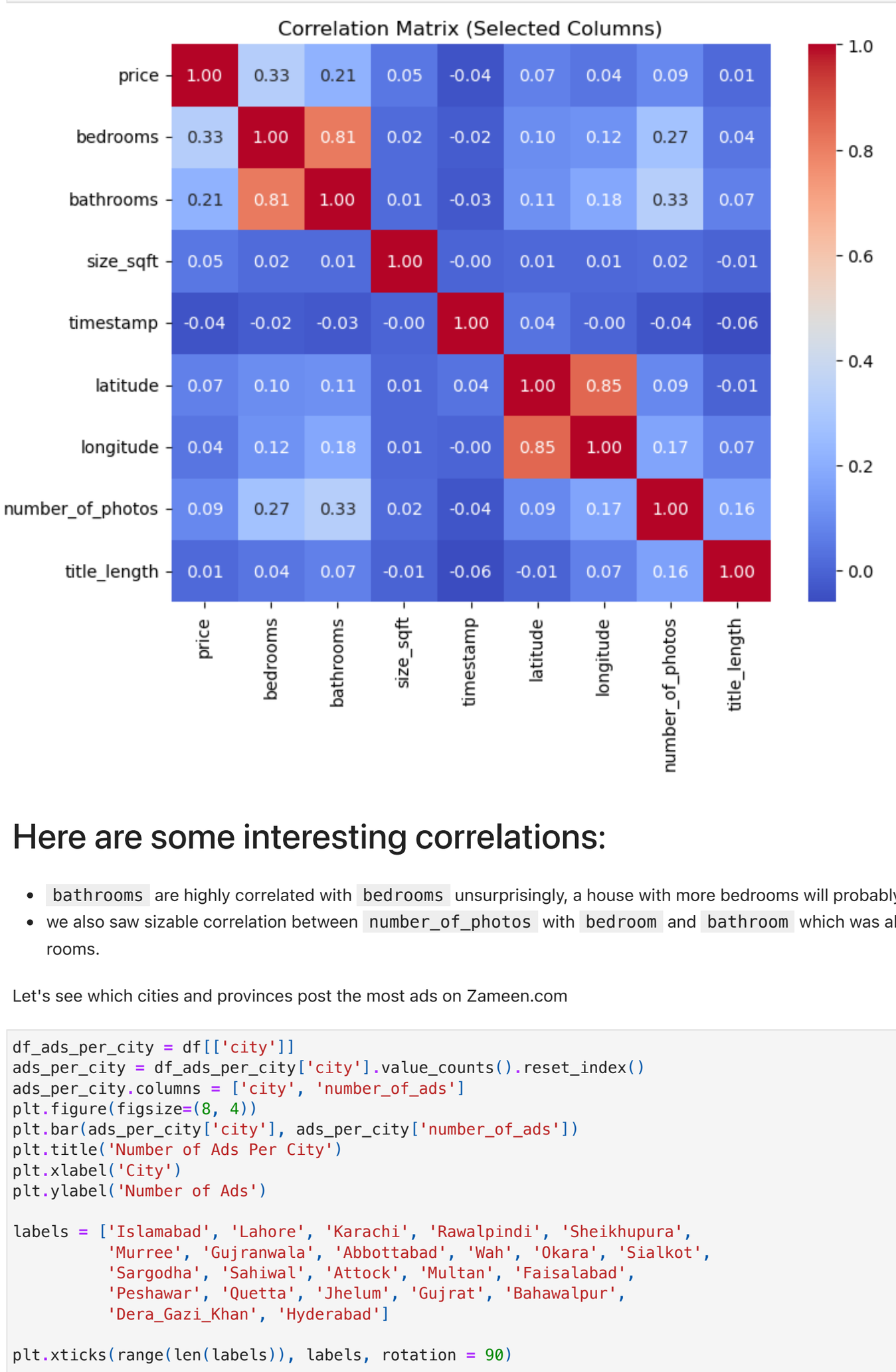
Let's take a look at our data:

In [88]:
```
summary_stats = df.describe()
print(summary_stats)
```

```
              price      bedrooms     bathrooms     size_sqft     timestamp  \
count  5.100100e+04  51001.000000  51001.000000  5.100100e+04  5.100100e+04
mean   5.461383e+07      3.835160      4.010490  3.163835e+03  1.059290e+06
std    1.017878e+08      1.907864      1.917992  5.591487e+04  1.492023e+06
min    1.230000e+03      0.000000      0.000000  0.000000e+00  6.000000e+01
25%    1.500000e+07      3.000000      3.000000  1.125000e+03  1.728000e+05
50%    2.950000e+07      4.000000      4.000000  1.800000e+03  6.048000e+05
75%    5.900000e+07      5.000000      6.000000  2.898000e+03  1.814400e+06
max    8.500000e+09     11.000000     10.000000  1.237500e+07  1.036800e+08

           latitude     longitude  number_of_photos  timestamp_adjusted  \
count  51001.000000  51001.000000      51001.000000        5.100100e+04
mean      30.448573     71.784330         16.647027        1.698587e+09
std        3.534901      2.956753         12.639265        1.492023e+06
min       21.595588     24.784237          0.000000        1.595966e+09
25%       25.030686     67.310472          7.000000        1.697832e+09
50%       31.471571     73.083930         14.000000        1.699042e+09
75%       33.539071     74.203652         24.000000        1.699474e+09
max       74.393259     74.689150         50.000000        1.699646e+09

       title_length
count  51001.000000
mean      10.590302
std        4.174095
min        1.000000
25%        8.000000
50%       10.000000
75%       13.000000
max       48.000000
```

Let's make a correlation matrix for all our numerical data:

In [89]:
```
selected_columns = ['price', 'bedrooms', 'bathrooms', 'size_sqft','timestamp', 'latitude', 'longitude','number_of_photos','title_length']
selected_df = df[selected_columns]

correlation_matrix = selected_df.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix (Selected Columns)')
plt.show()
```

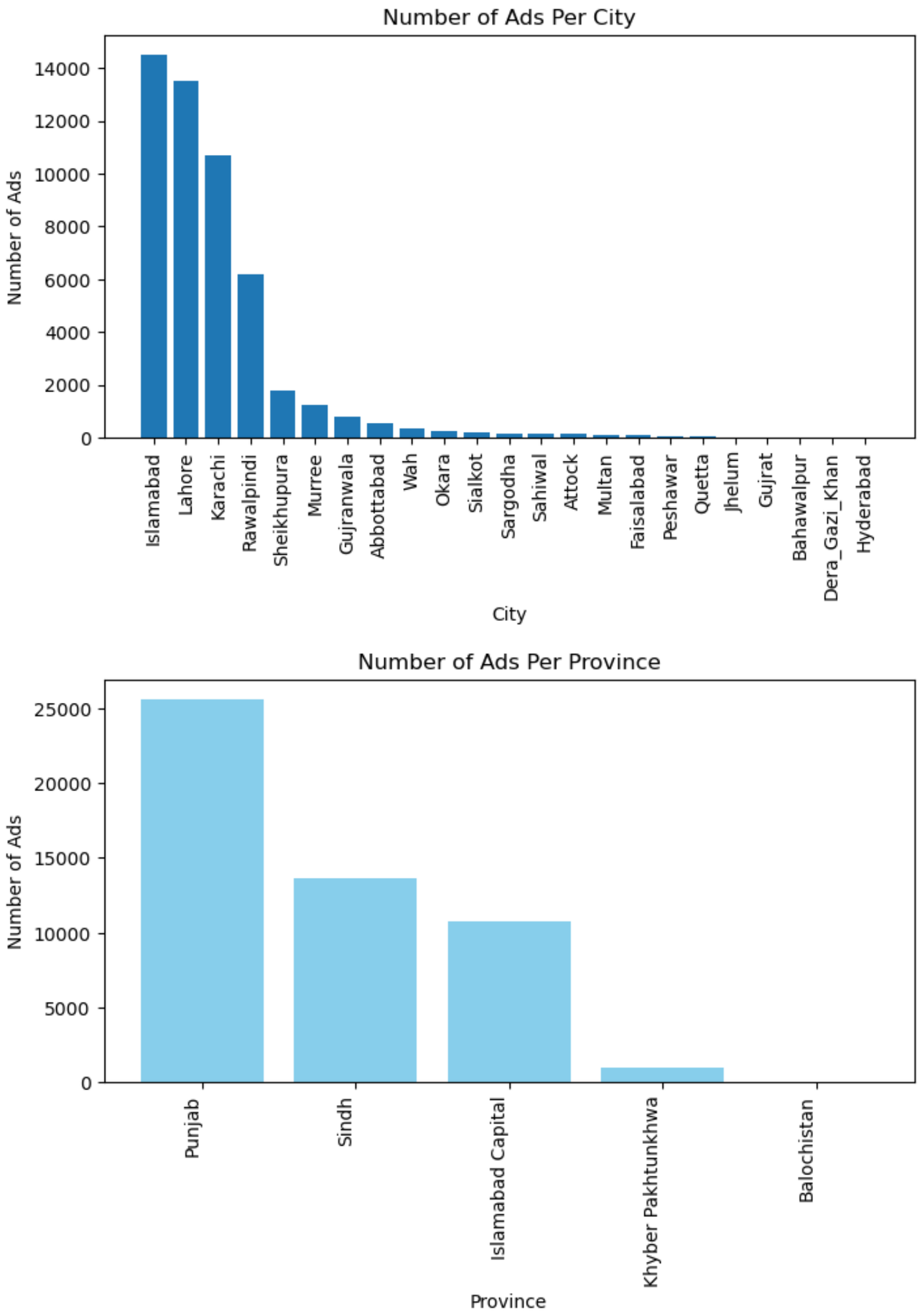


Here are some interesting correlations:

- `bathrooms` are highly correlated with `bedrooms` unsurprisingly, a house with more bedrooms will probably have more bathrooms to accompany them.
- we also saw sizable correlation between `number_of_photos` with `bedroom` and `bathroom` which was also expected, as they would want more pictures to showcase more rooms.

Let's see which cities and provinces post the most ads on Zameen.com

In [90]:
```
df_ads_per_city = df[['city']]
ads_per_city = df_ads_per_city['city'].value_counts().reset_index()
ads_per_city.columns = ['city', 'number_of_ads']
plt.figure(figsize=(8, 4))
plt.bar(ads_per_city['city'], ads_per_city['number_of_ads'])
plt.title('Number of Ads Per City')
plt.xlabel('City')
plt.ylabel('Number of Ads')

labels = ['Islamabad', 'Lahore', 'Karachi', 'Rawalpindi', 'Sheikhupura',
          'Murree', 'Gujranwala', 'Abbottabad', 'Wah', 'Okara', 'Sialkot',
          'Sargodha', 'Sahiwal', 'Attock', 'Multan', 'Faisalabad',
          'Peshawar', 'Quetta', 'Jhelum', 'Gujrat', 'Bahawalpur',
          'Dera_Gazi_Khan', 'Hyderabad']


plt.xticks(range(len(labels)), labels, rotation = 90)

plt.show()

df_ads_per_province = df[['province']]
ads_per_province = df_ads_per_province['province'].value_counts().reset_index()
ads_per_province.columns = ['province', 'number_of_ads']

plt.figure(figsize=(8, 4))
plt.bar(ads_per_province['province'], ads_per_province['number_of_ads'], color='skyblue')
plt.title('Number of Ads Per Province')
plt.xlabel('Province')
plt.ylabel('Number of Ads')

ads_per_province = ads_per_province.sort_values(by='number_of_ads', ascending=False)

plt.xticks(range(len(ads_per_province['province'])), ads_per_province['province'], rotation=90, ha='right')
plt.show()
```

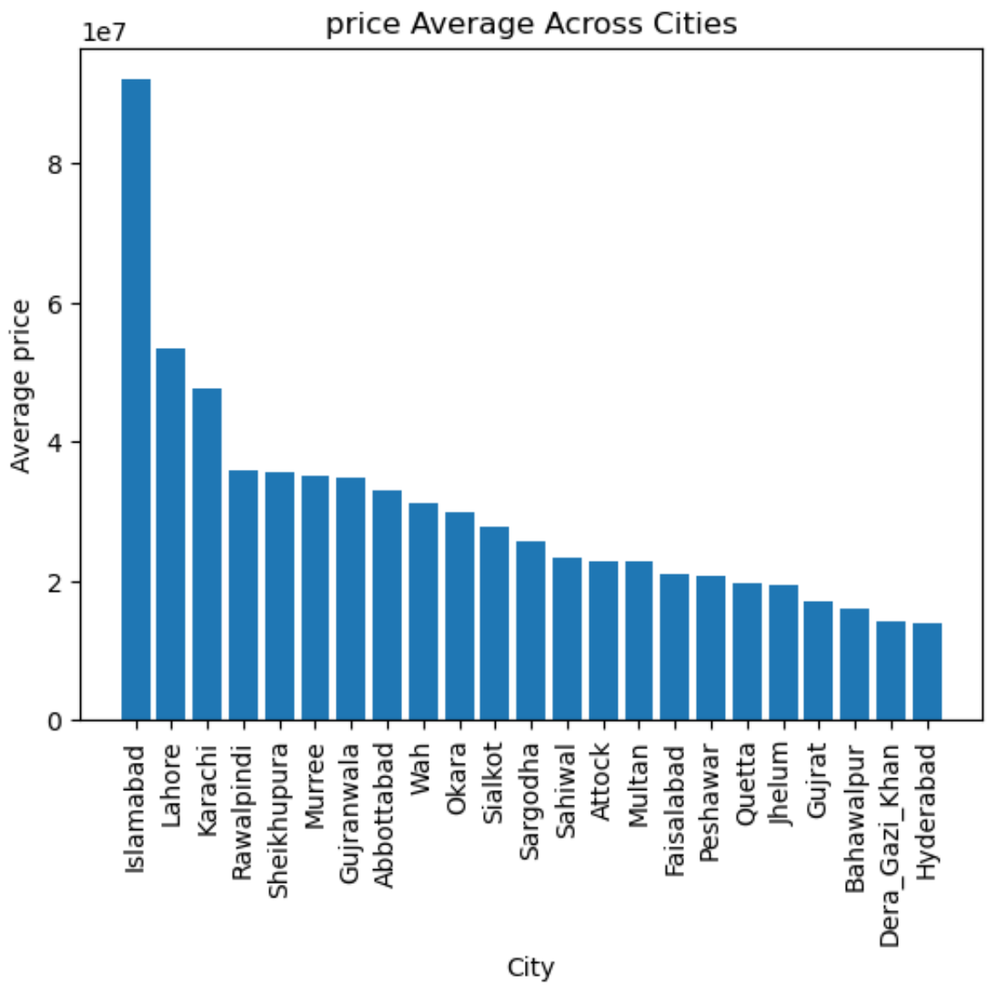


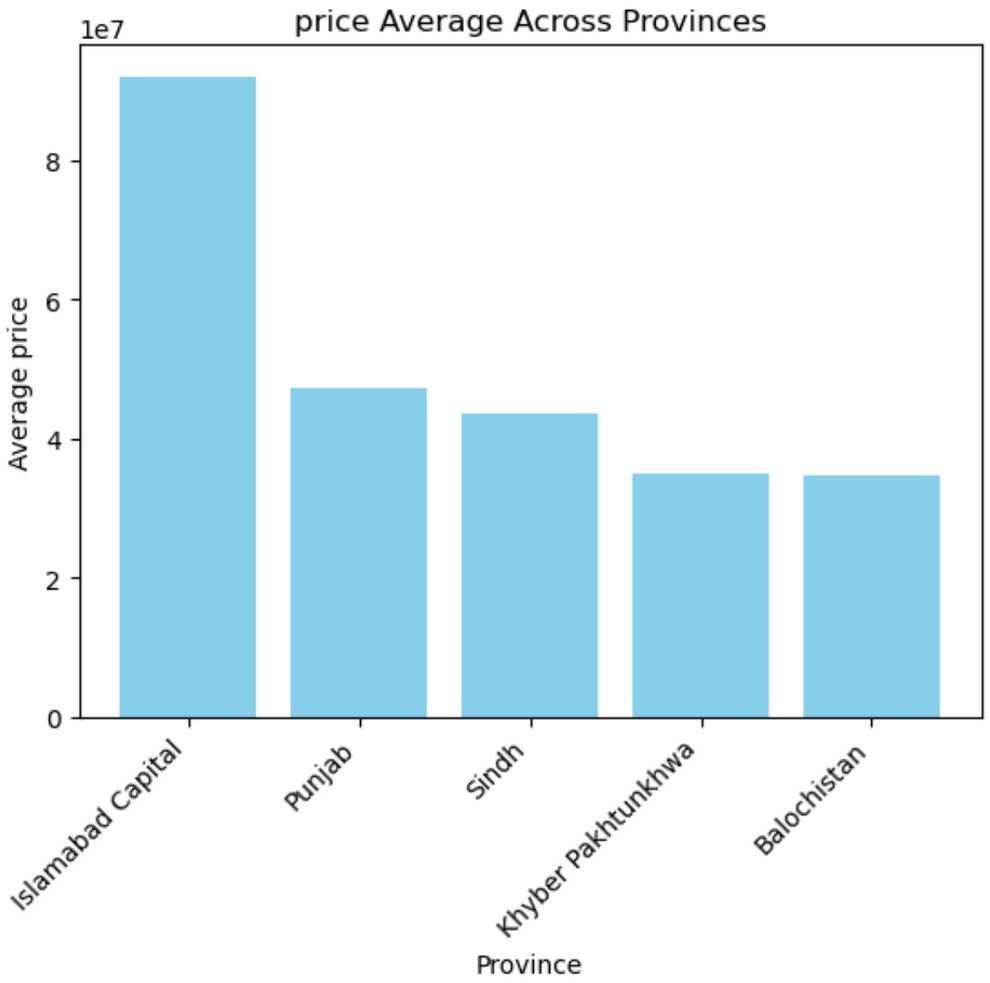
Analysis:

Our initial guess was that Karachi or Lahore would have the most ads, proportional with their population, however to our surprise Islamabad took the crown. Punjab being on top is about what we expected.

Let's see which cities are most expensive on average, in terms of property listings:

In [91]:
```
def plot_column_against_cities(data, col, x_labels=None):
    avg_values = data.groupby('city')[col].mean().sort_values(ascending=False)

    plt.bar(avg_values.index, avg_values)
    plt.title(f'{col} Average Across Cities')
    plt.xlabel('City')
    if x_labels:
        plt.xticks(range(len(x_labels)), x_labels, rotation='horizontal')
    plt.xticks(rotation=90)
    plt.ylabel(f'Average {col}')
    plt.show()

desired_column = 'price'
plot_column_against_cities(df, 'price', x_labels=['Islamabad', 'Lahore', 'Karachi', 'Rawalpindi', 'Sheikhupura',
       'Murree', 'Gujranwala', 'Abbottabad', 'Wah', 'Okara', 'Sialkot',
       'Sargodha', 'Sahiwal', 'Attock', 'Multan', 'Faisalabad',
       'Peshawar', 'Quetta', 'Jhelum', 'Gujrat', 'Bahawalpur',
       'Dera_Gazi_Khan', 'Hyderabad'])


def plot_column_against_provinces(data, col, x_labels=None):
    avg_values = data.groupby('province')[col].mean().sort_values(ascending=False)

    plt.bar(avg_values.index, avg_values, color='skyblue')
    plt.title(f'{col} Average Across Provinces')
    plt.xlabel('Province')
    if x_labels:
        plt.xticks(range(len(x_labels)), x_labels, rotation='horizontal')
    plt.xticks(rotation=45, ha='right')
    plt.ylabel(f'Average {col}')
    plt.show()

desired_column = 'price'
plot_column_against_provinces(df, desired_column, x_labels=['Islamabad Capital', 'Punjab', 'Sindh', 'Khyber Pakhtunkhwa', 'Balochistan'])
```



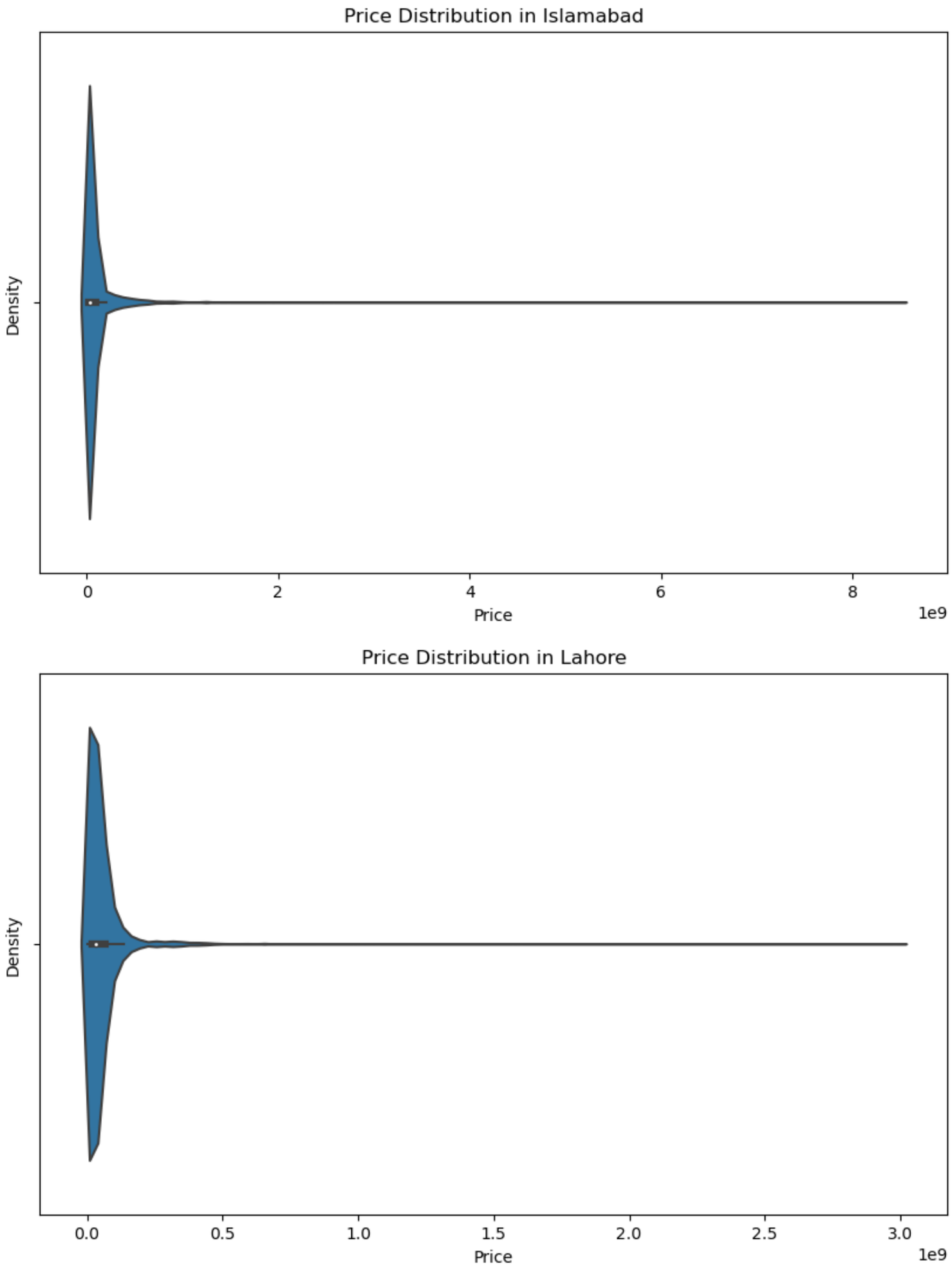


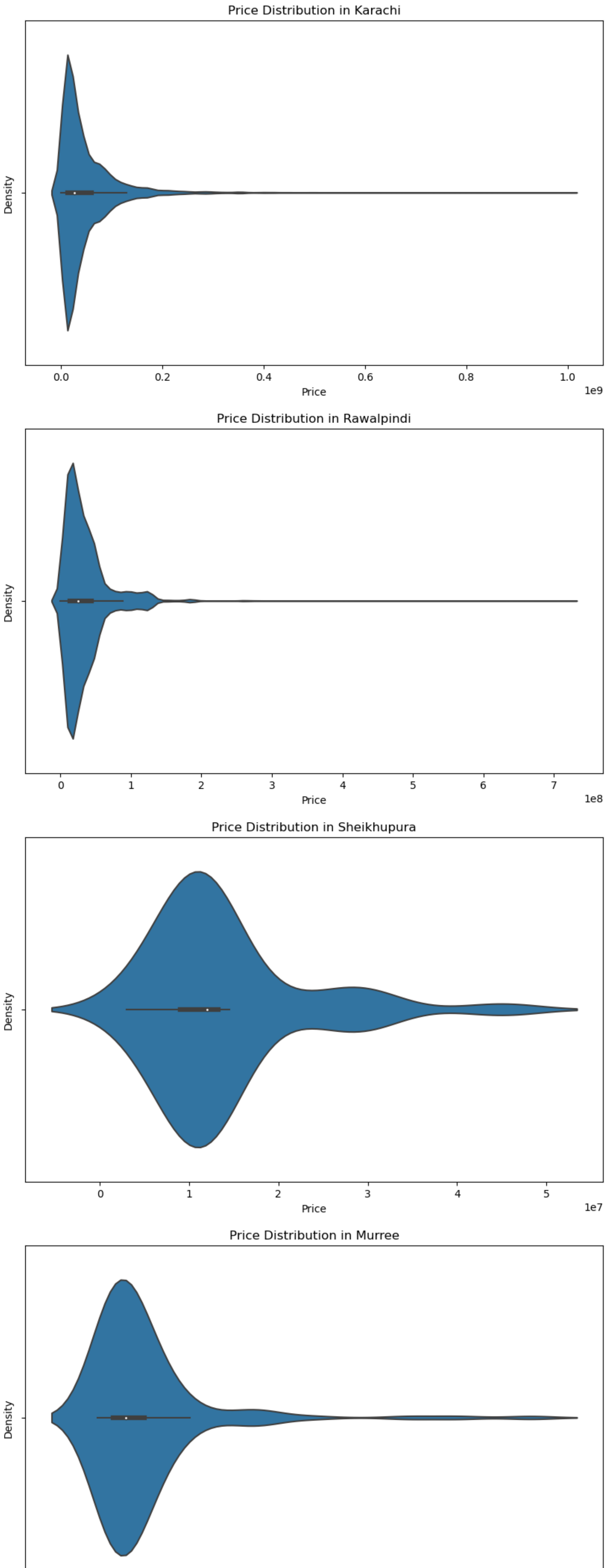
Analysis:

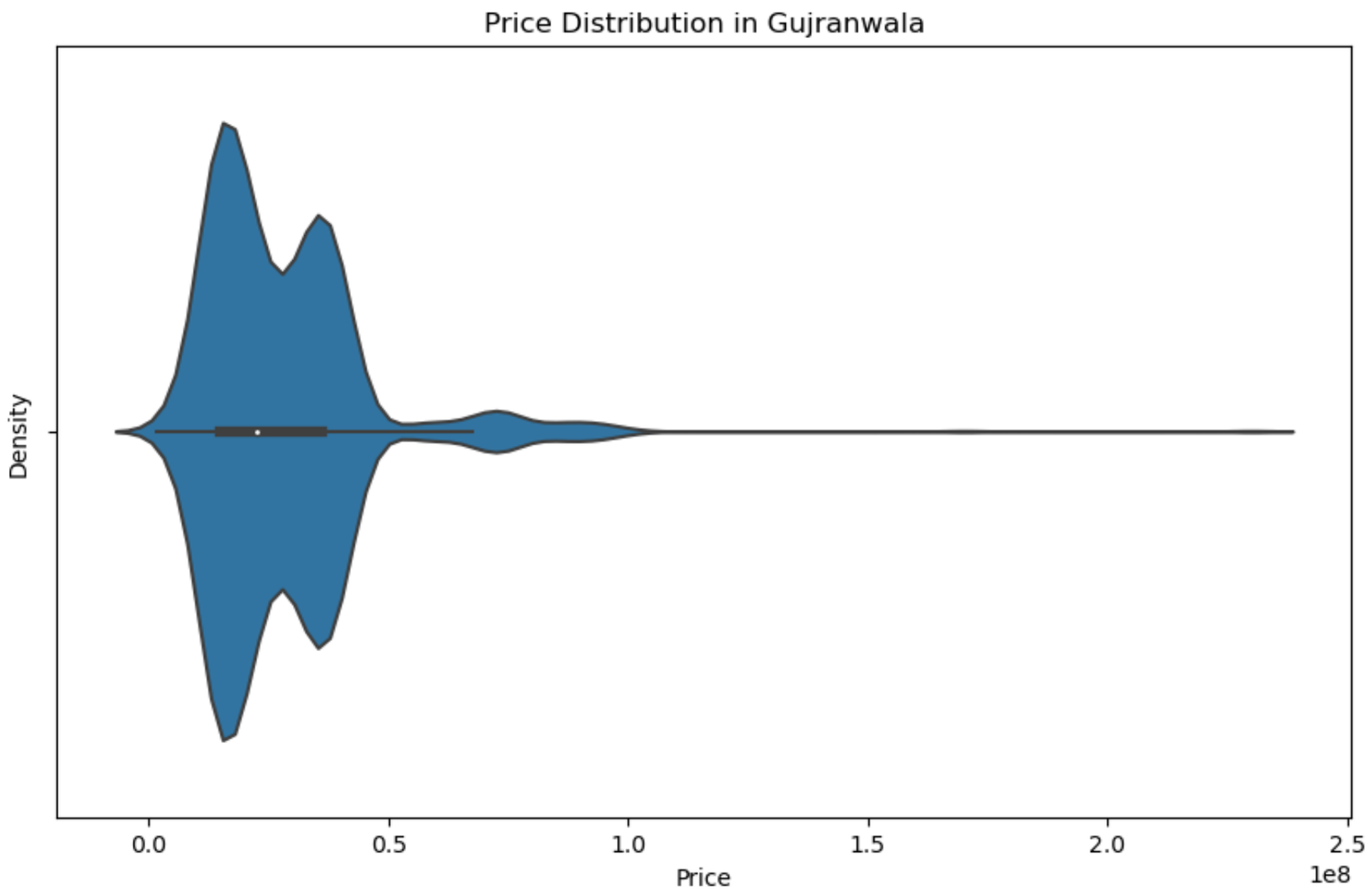
Islamabad came out on top by a large margin, again defying our guess of Karachi. However, when we thought more about it, a higher average in Islamabad makes sense as it has

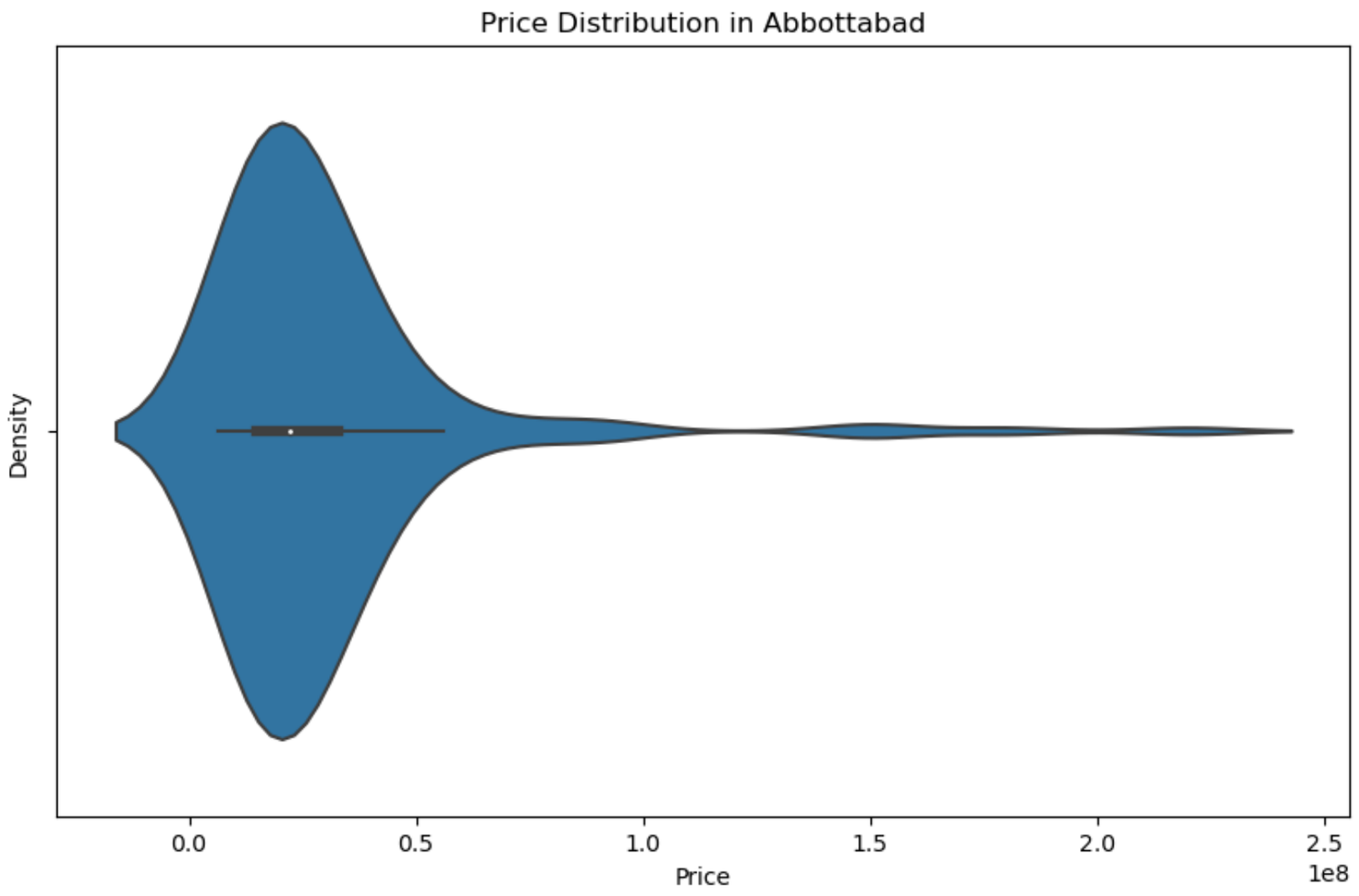
Now let's study the spread and skew of the data within each city:

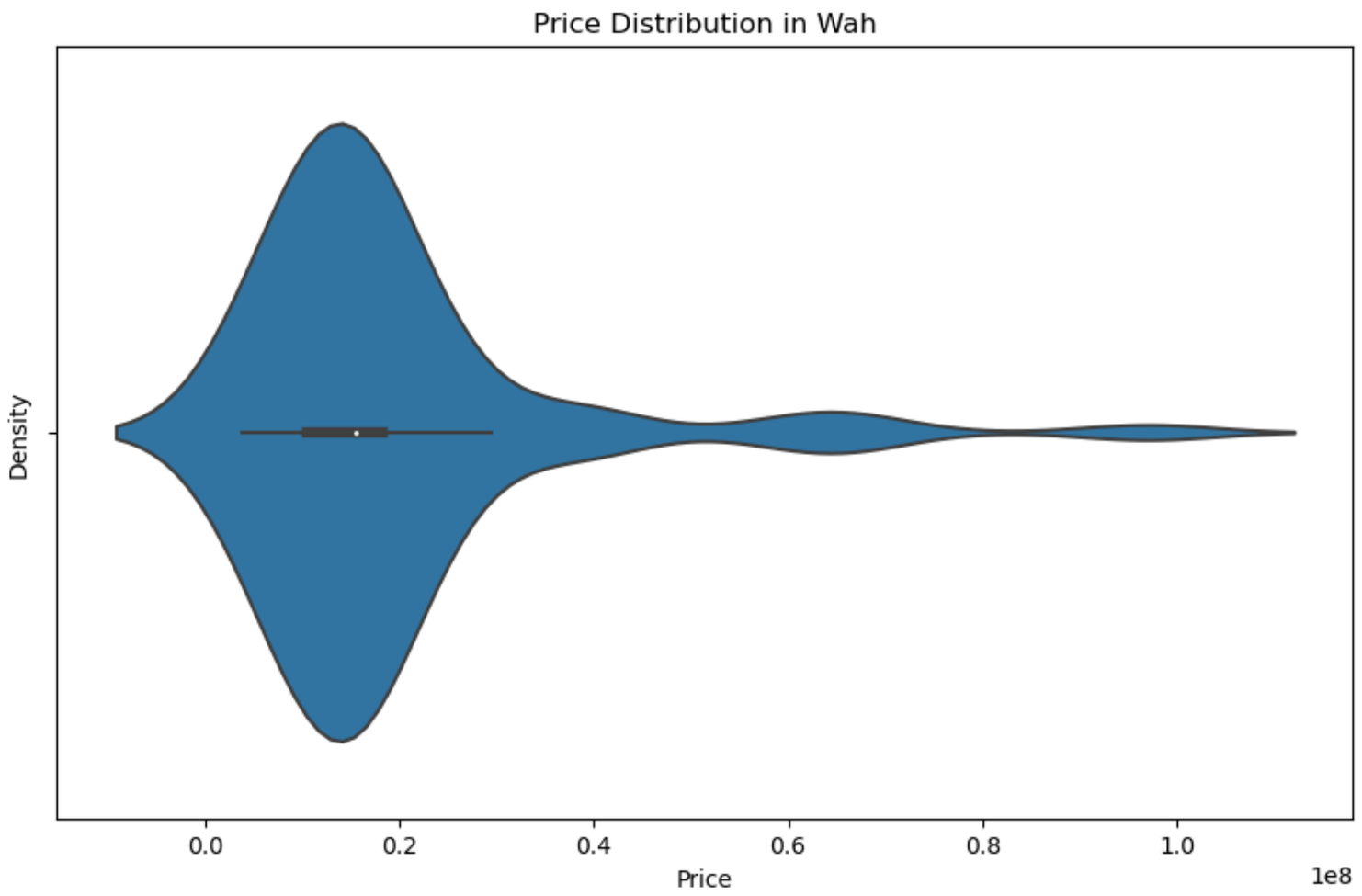
In [92]:
```
df = df.dropna(subset=['price'])
df = df[df['price'] > 0]  # Ensure that price is greater than zero
cities = df['city'].unique()

# iterating over each city and create a violin plot for each
for city in cities:
    city_df = df[df['city'] == city]

    plt.figure(figsize=(10, 6))
    sns.violinplot(x=city_df['price'])
    plt.title(f'Price Distribution in {city}')
    plt.xlabel('Price')
    plt.ylabel('Density')

    # want a separate plot for each city
    plt.show()
```

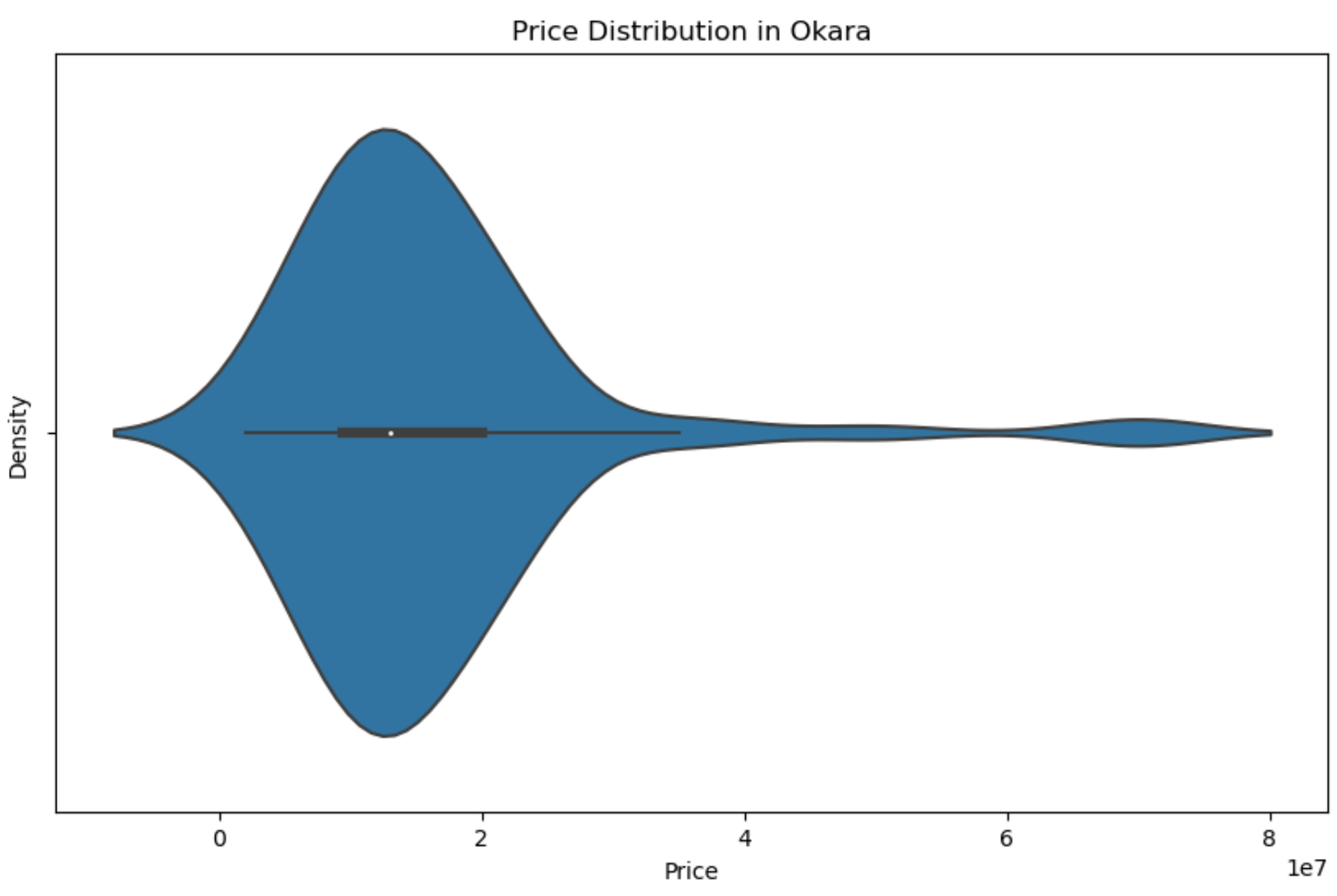


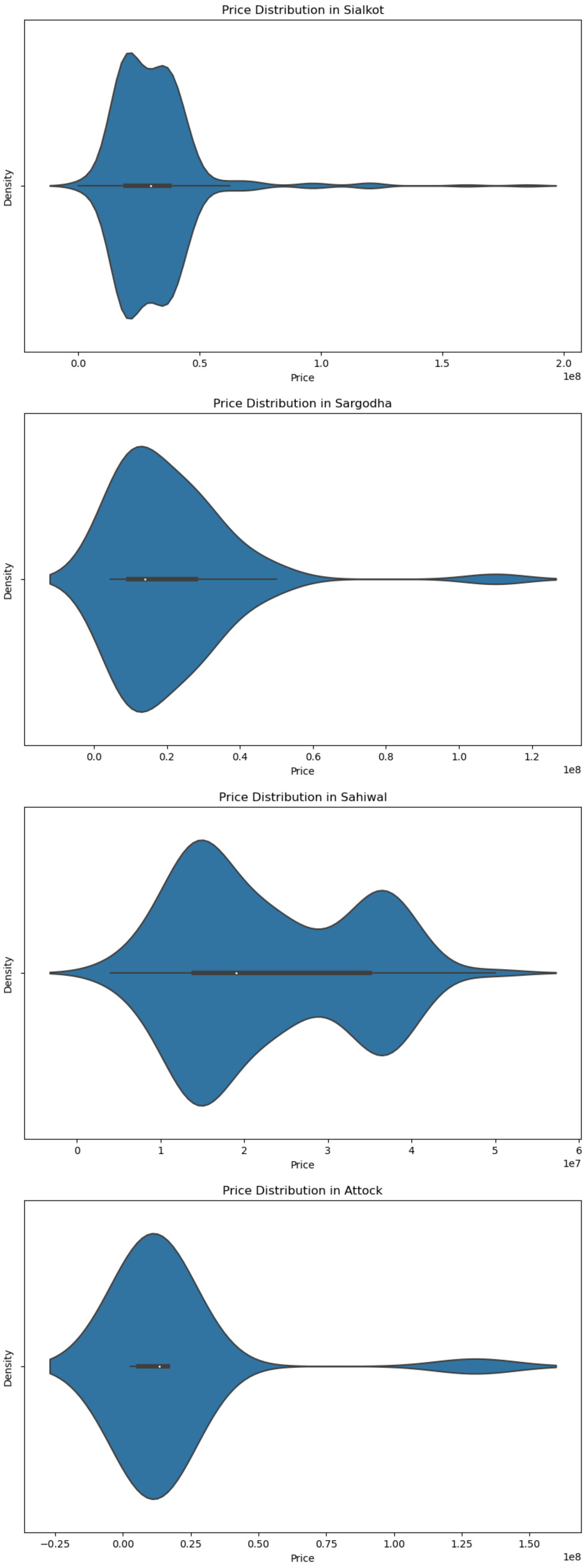

Price Distribution in Karachi
Density
0.0
0.2
0.4
0.6
0.8
1.0
Price
1e9
Price Distribution in Rawalpindi
Density
0
1
2
3
4
5
6
7
Price
1e8
Price Distribution in Sheikhupura
Density
0
1
2
3
4
5
Price
1e7
Price Distribution in Murree
Density
0.0
0.5
1.0
1.5
2.0
Price
1e8

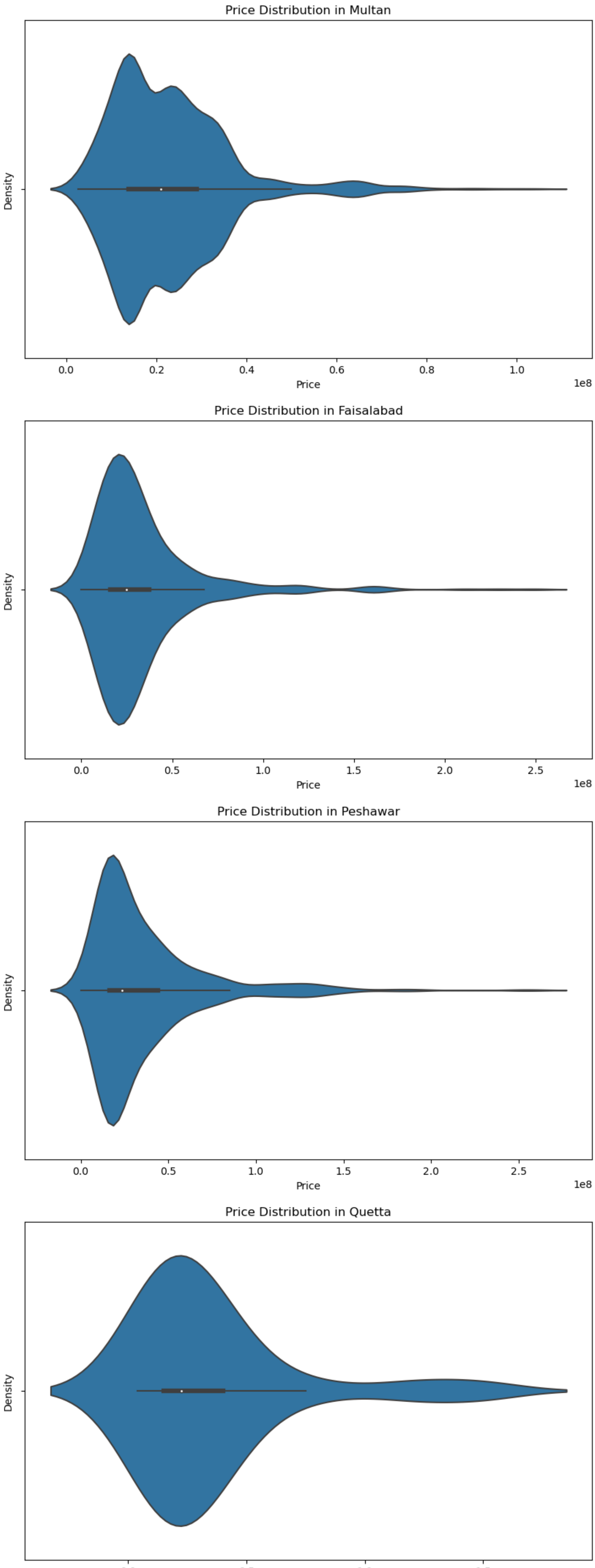

Price Distribution in Gujranwala
Density
0.0
0.5
1.0
1.5
2.0
2.5
Price
1e8

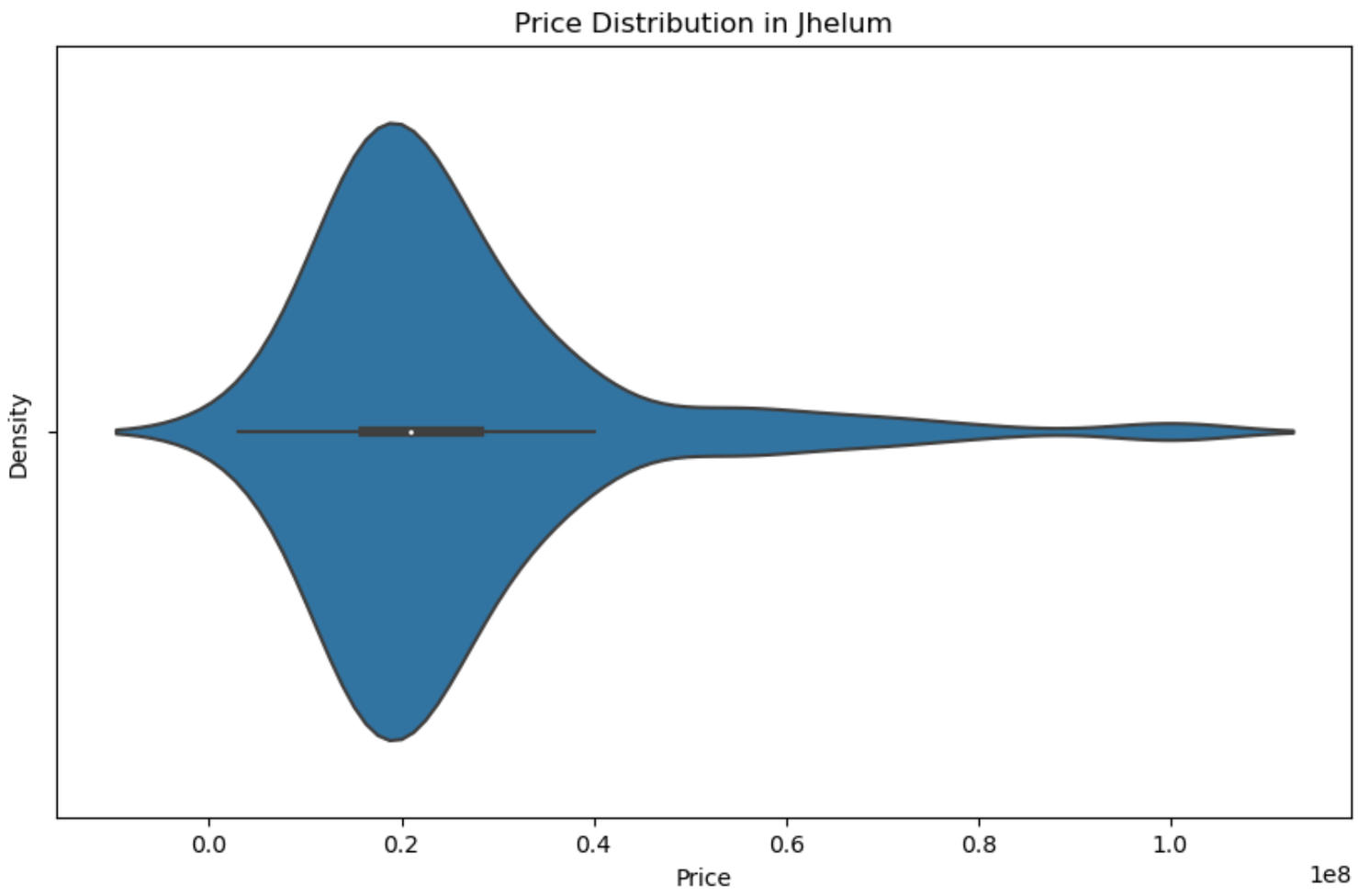

Price Distribution in Abbottabad
Density
0.0
0.5
1.0
1.5
2.0
2.5
Price
1e8

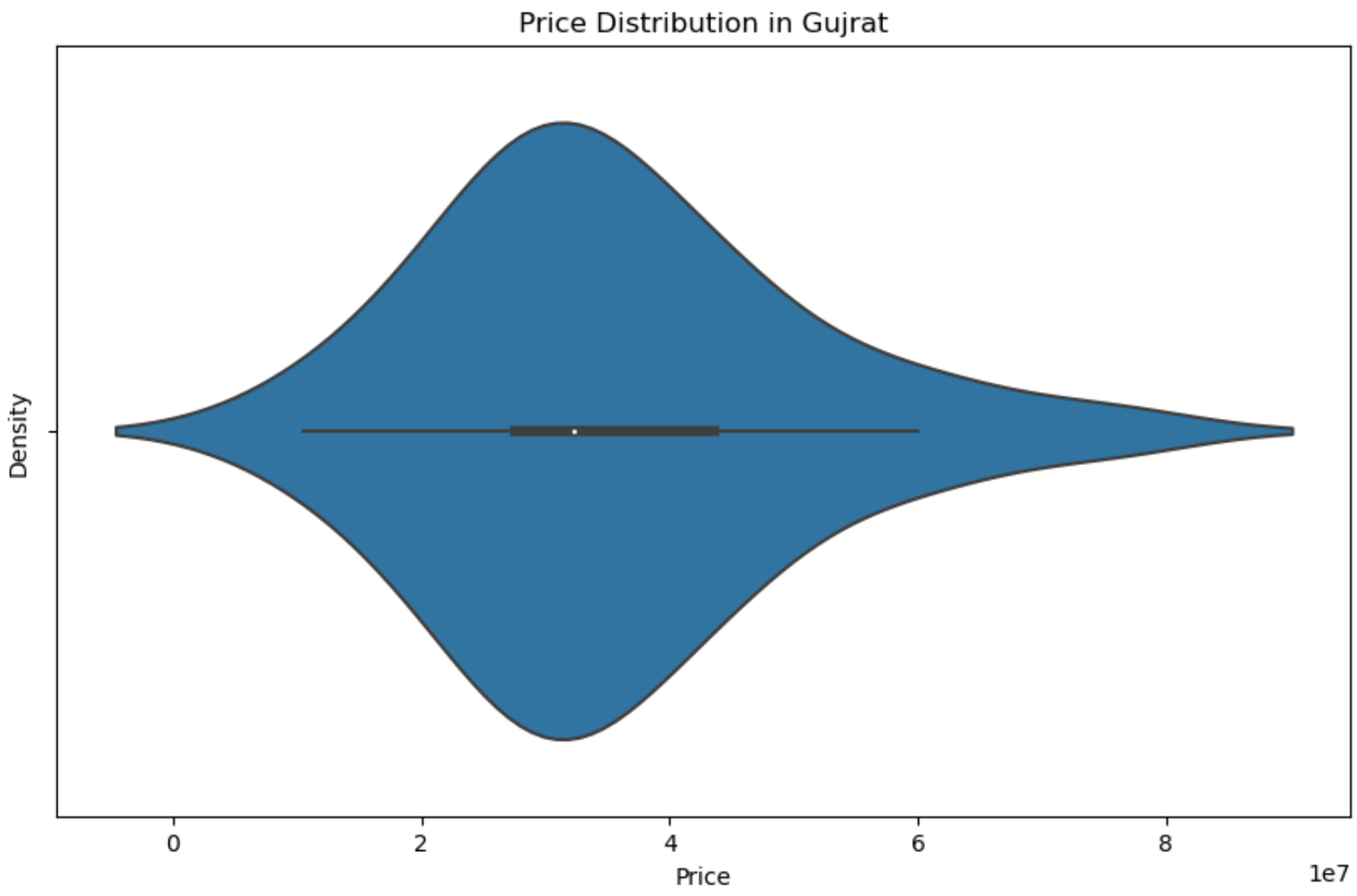

Price Distribution in Wah
Density
0.0
0.2
0.4
0.6
0.8
1.0
Price
1e8

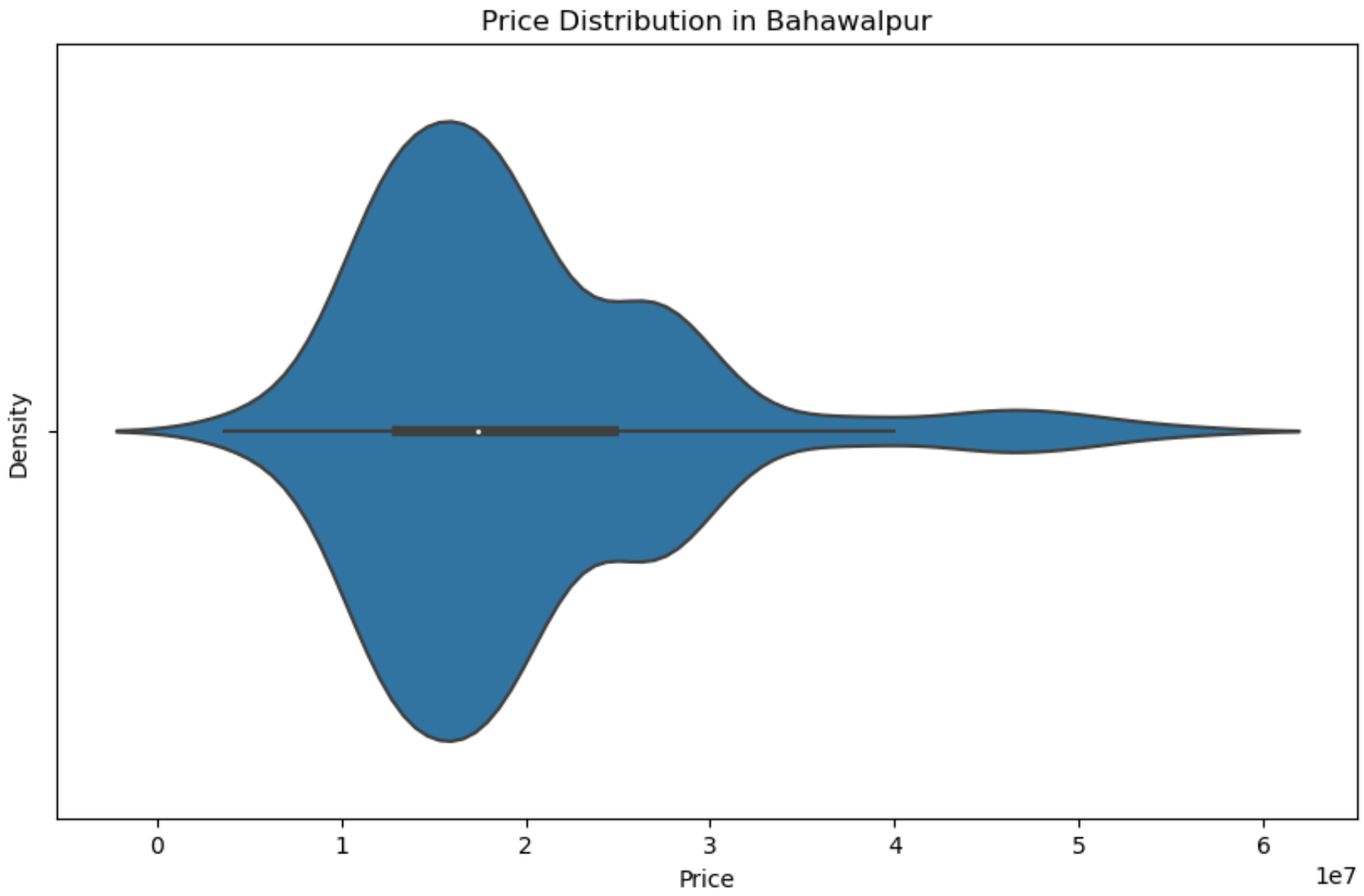

Price Distribution in Okara
Density
0
2
4
6
8
Price
1e7

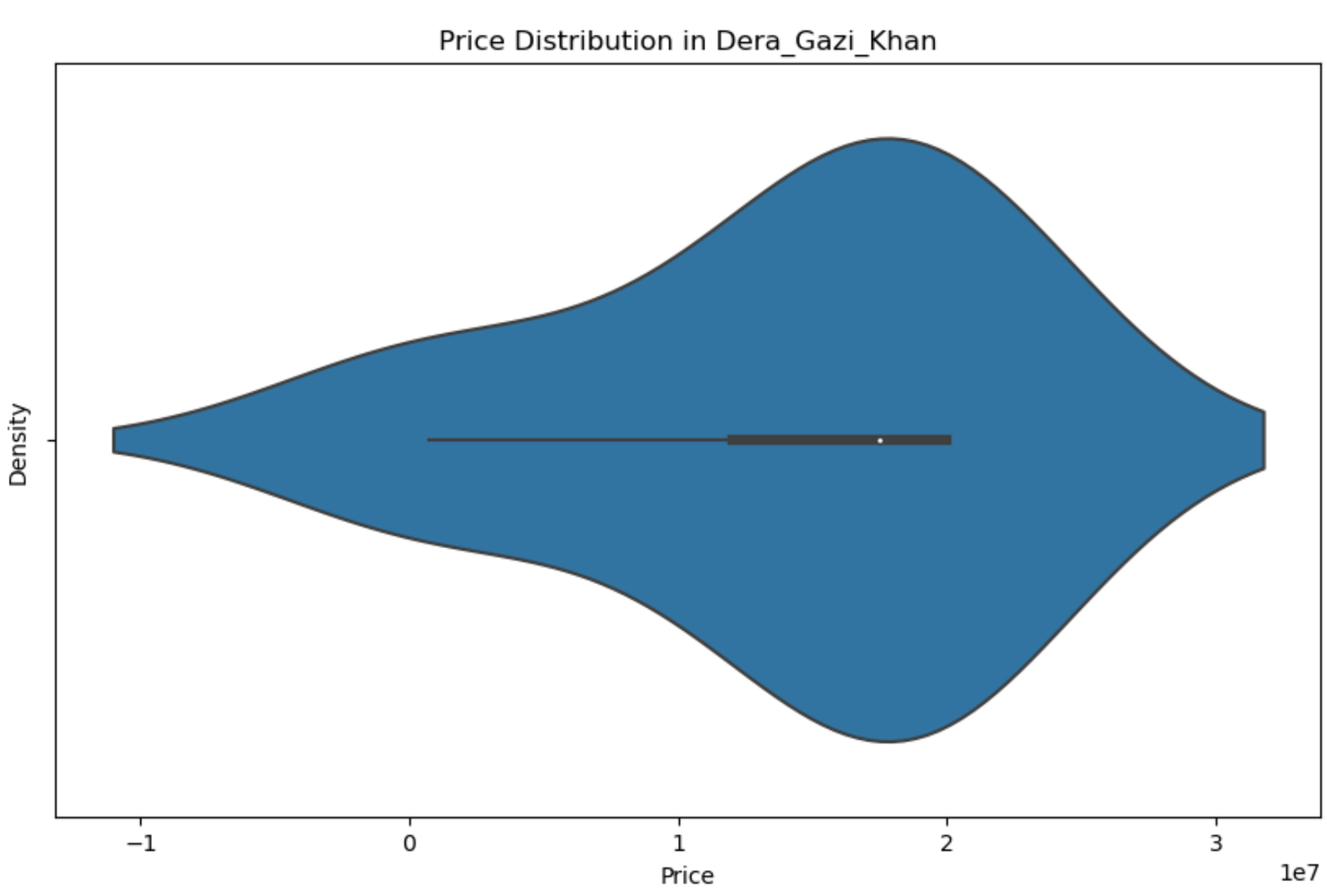

Price Distribution in Sialkot
Density
0.0
0.5
1.0
1.5
2.0
Price
1e8
Price Distribution in Sargodha
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Price
1e8
Price Distribution in Sahiwal
Density
0
1
2
3
4
5
6
Price
1e7
Price Distribution in Attock
Density
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
Price
1e8

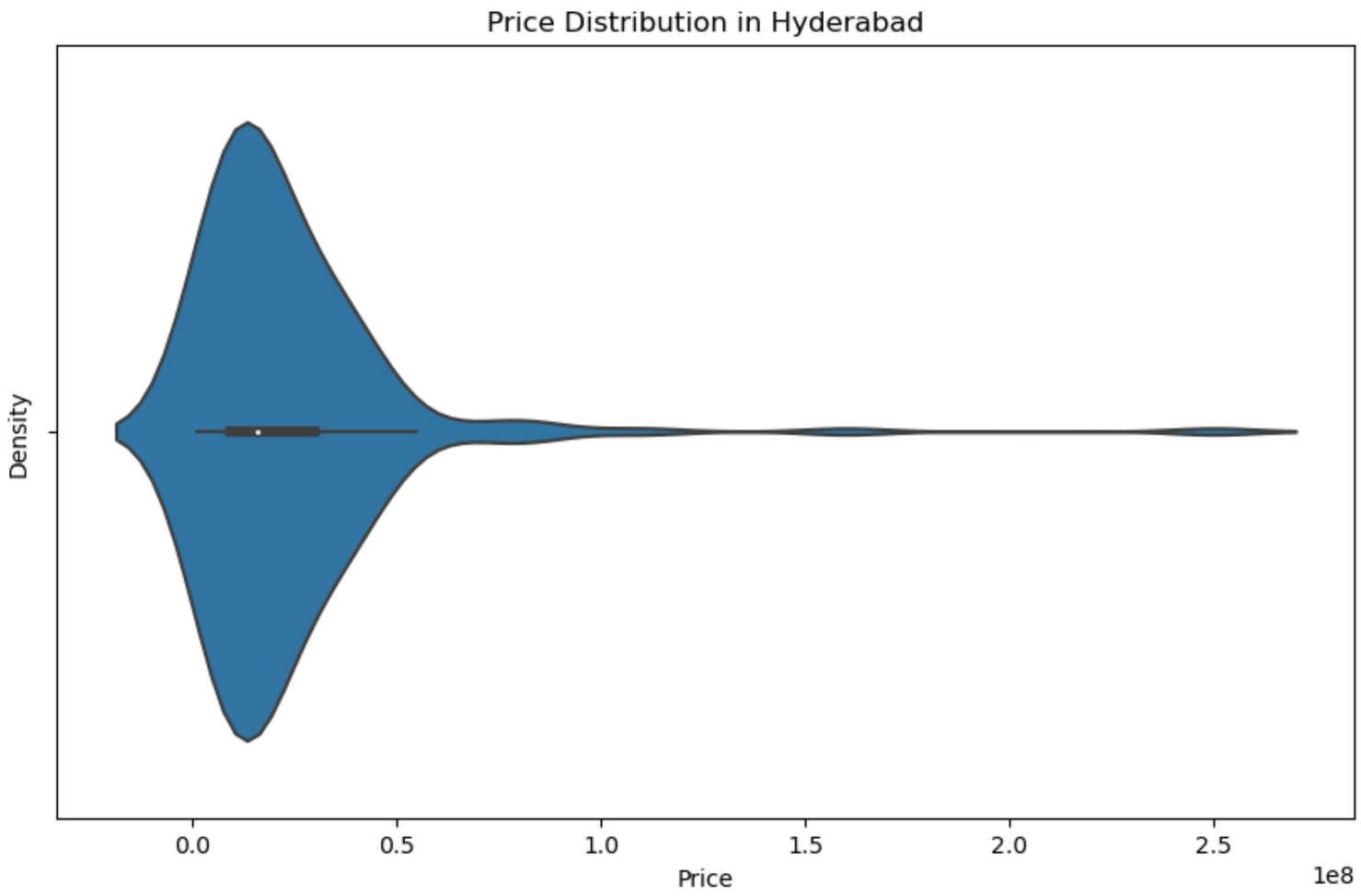

Price Distribution in Multan
Density
0.0
0.2
0.4
0.6
0.8
1.0
Price
1e8
Price Distribution in Faisalabad
Density
0.0
0.5
1.0
1.5
2.0
2.5
Price
1e8
Price Distribution in Peshawar
Density
0.0
0.5
1.0
1.5
2.0
2.5
Price
1e8
Price Distribution in Quetta
Density
0.0
0.5
1.0
1.5
Price
1e8


Price Distribution in Jhelum
Density
0.0
0.2
0.4
0.6
0.8
1.0
Price
1e8


Price Distribution in Gujrat
Density
0
2
4
6
8
Price
1e7


Price Distribution in Bahawalpur
Density
0
1
2
3
4
5
6
Price
1e7

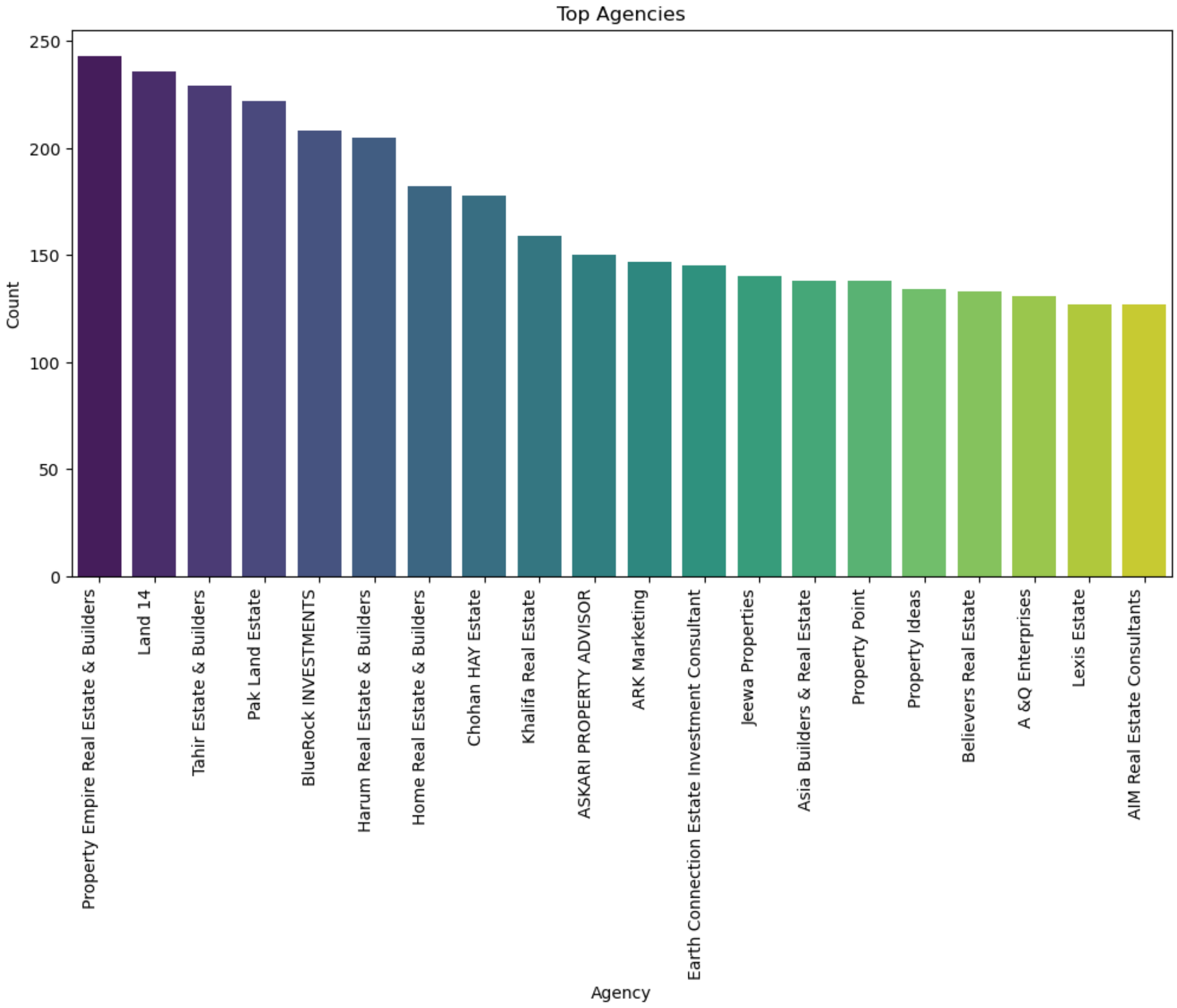

Price Distribution in Dera_Gazi_Khan
Density
−1
0
1
2
3
Price
1e7



We really wanted to uncover trends with agencies, and with over 4,000 agencies, here are the ones appearing in the most listings:

In [93]:
```
print(len(df['agency'].unique()))
unique_agencies = df['agency'].unique()
agency_counts = df['agency'].value_counts()
top_agencies = agency_counts.head(20)
plt.figure(figsize=(12, 6))
sns.barplot(x=top_agencies.index, y = top_agencies.values, palette = 'viridis')
plt.title('Top Agencies')
plt.xlabel('Agency')
plt.ylabel('Count')
plt.xticks(rotation=90, ha='right')
plt.show()
```

4184



Analysis

In [94]:
```
df.head(5)
```

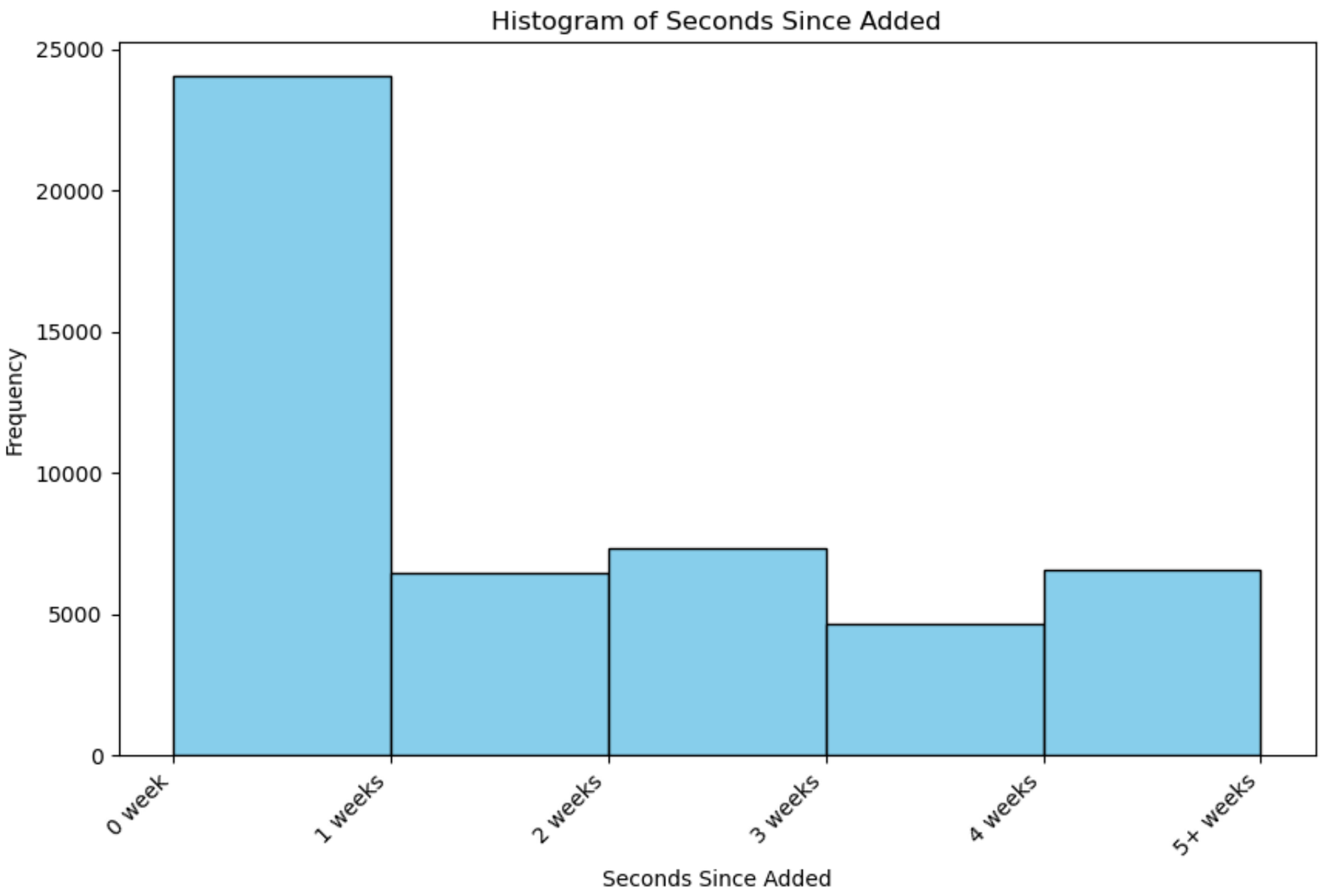
Out[94]:

property_id	city	location	price	bedrooms	bathrooms	size_sqft	title_text	timestamp	property_type	latitude	longitude	number_of_photos	province
1	Islamabad	DHA Defence Phase 2	110000000	6	6	4500.0	Luxurious 1 kanal Brand New Designer House For...	10800.0	Houses_Property	33.527944	73.161392	48	Islamabad Capital
2	Islamabad	F-7	320000000	4	4	5850.0	Fully Renovated House For Sale In F-7 Islamabad	32400.0	Houses_Property	33.720413	73.056493	12	Islamabad Capital
3	Islamabad	B-17	20800000	1	1	562.0	1 Studio Bed Apartment 566 Sq Feet For Sale In...	54000.0	Flats_Apartments	33.680640	72.822189	19	Islamabad Capital
4	Islamabad	Bahria Enclave - Sector C	51000000	5	6	2250.0	Amazing 10Marla House is Available for sale	57600.0	Houses_Property	33.689546	73.219049	24	Islamabad Capital
5	Islamabad	Shams Colony	42000000	6	5	2250.0	10 Marla Commercial House For Sale	61200.0	Houses_Property	33.629736	72.974367	6	Islamabad Capital

In [95]:
```
plt.figure(figsize=(10, 6))
```

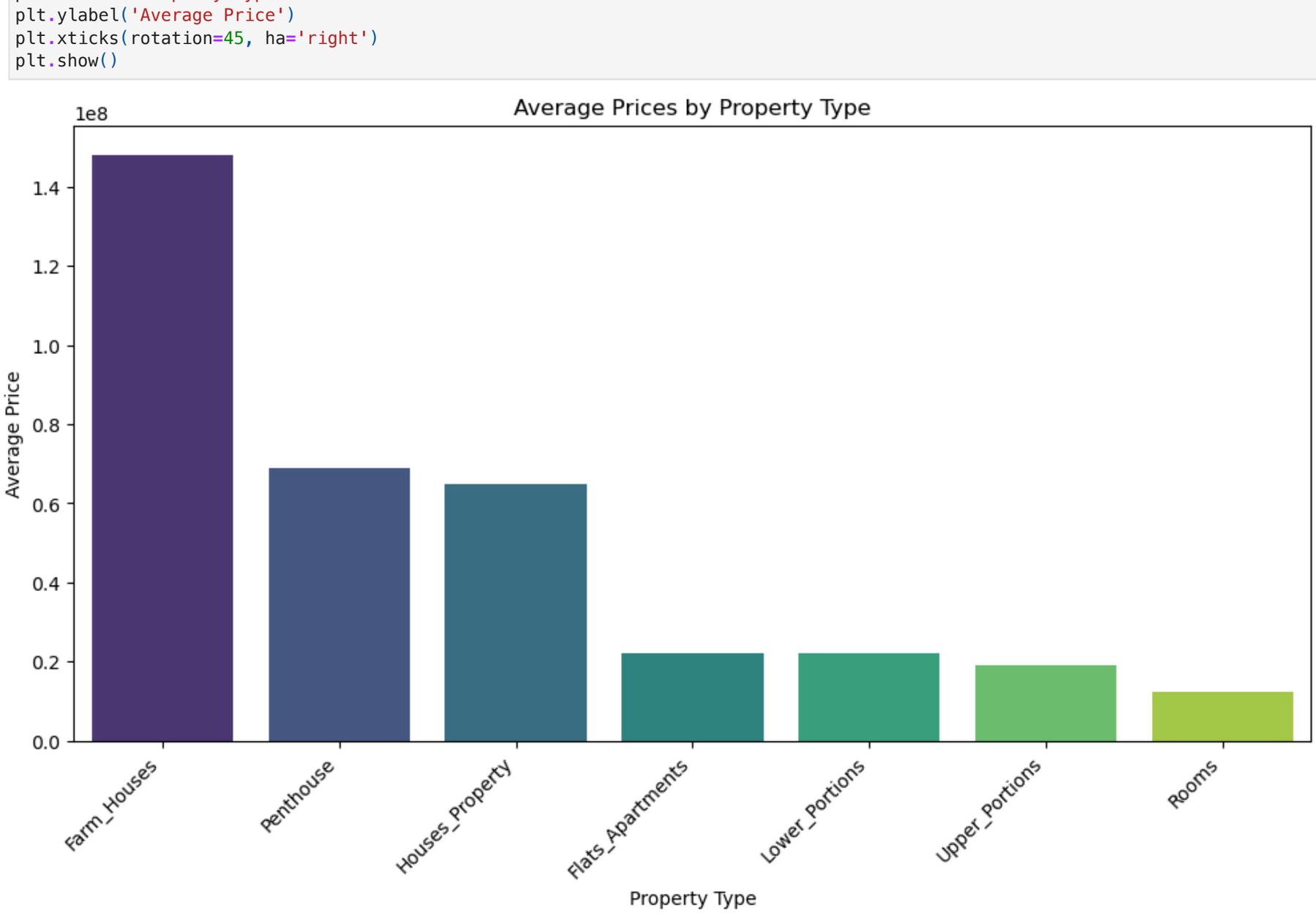
```
plt.hist(df['timestamp'], bins=[0, 604800, 1209600, 1814400, 2419200, 3024000, float('inf')], edgecolor='black', color='skyblue')
plt.title('Histogram of Seconds Since Added')
plt.xlabel('Seconds Since Added')
plt.ylabel('Frequency')
plt.xticks([0, 604800, 1209600, 1814400, 2419200, 3024000],
           ['0 week', '1 weeks', '2 weeks', '3 weeks', '4 weeks', '5+ weeks'], rotation=45, ha='right')
plt.show()
```

```
/Users/knatiq/anaconda3/lib/python3.11/site-packages/matplotlib/axes/_axes.py:6826: RuntimeWarning: invalid value encountered in multiply
  boffset = -0.5 * dr * totwidth * (1 - 1 / nx)
```



Let's see which types of property are most expensive, on average:

In [96]:
```
average_prices_by_type = df.groupby('property_type')['price'].mean().sort_values(ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x=average_prices_by_type.index, y=average_prices_by_type.values, palette='viridis')
plt.title('Average Prices by Property Type')
plt.xlabel('Property Type')
plt.ylabel('Average Price')
plt.xticks(rotation=45, ha='right')
plt.show()
```

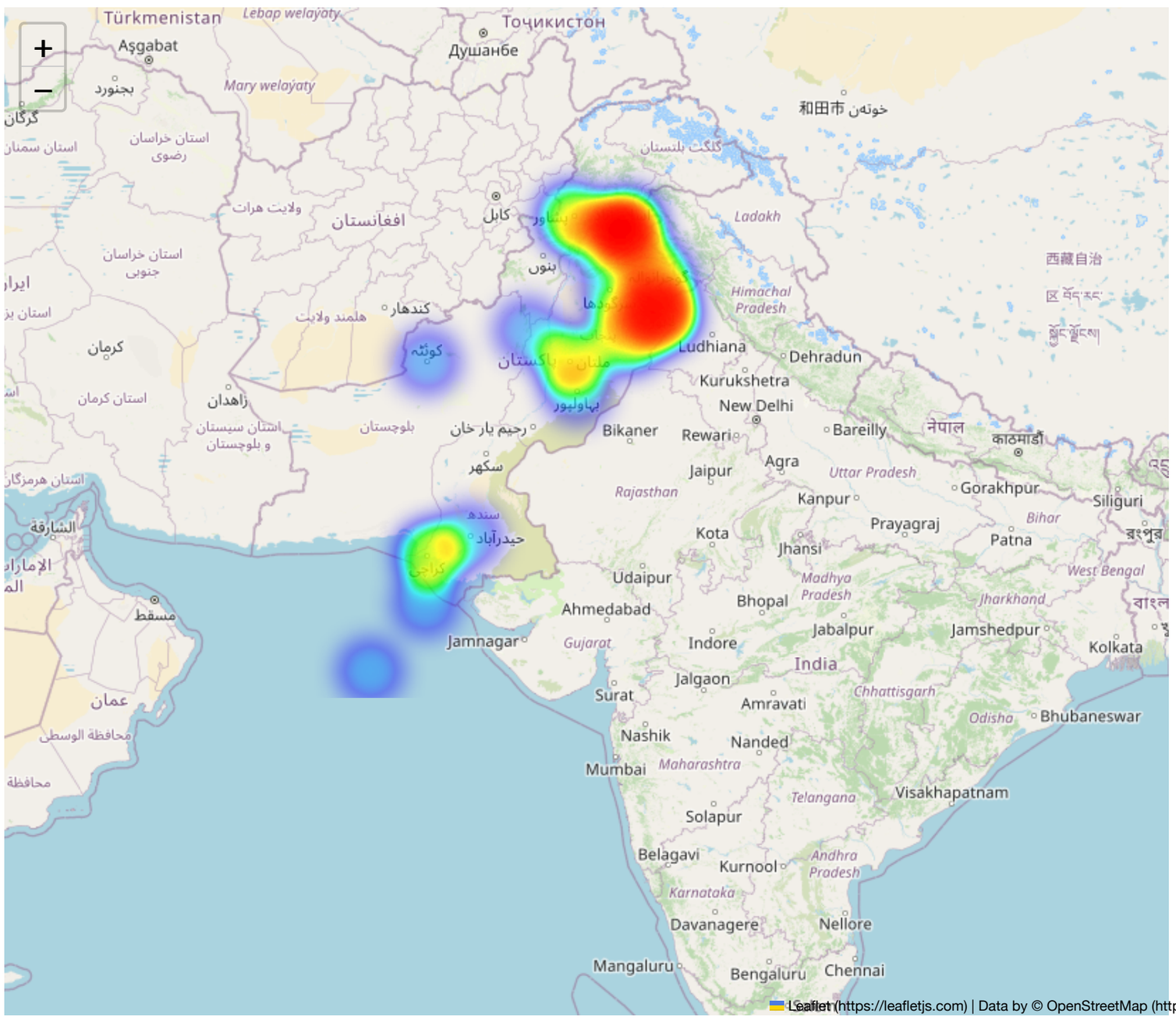


Let's plot a heatmap about prices across Pakistan:

In [97]:
```
import pandas as pd
import folium
from folium.plugins import HeatMap

# Create a base map
map_center = [df['latitude'].mean(), df['longitude'].mean()]
map = folium.Map(location=map_center, zoom_start=5)

heatmap_data = df[['latitude', 'longitude', 'price']].values.tolist()
heatmap = HeatMap(heatmap_data)
map.add_child(heatmap)

print("Property listing prices across Pakistan:")
map
```

```
Property listing prices across Pakistan:
```

Out[97]:



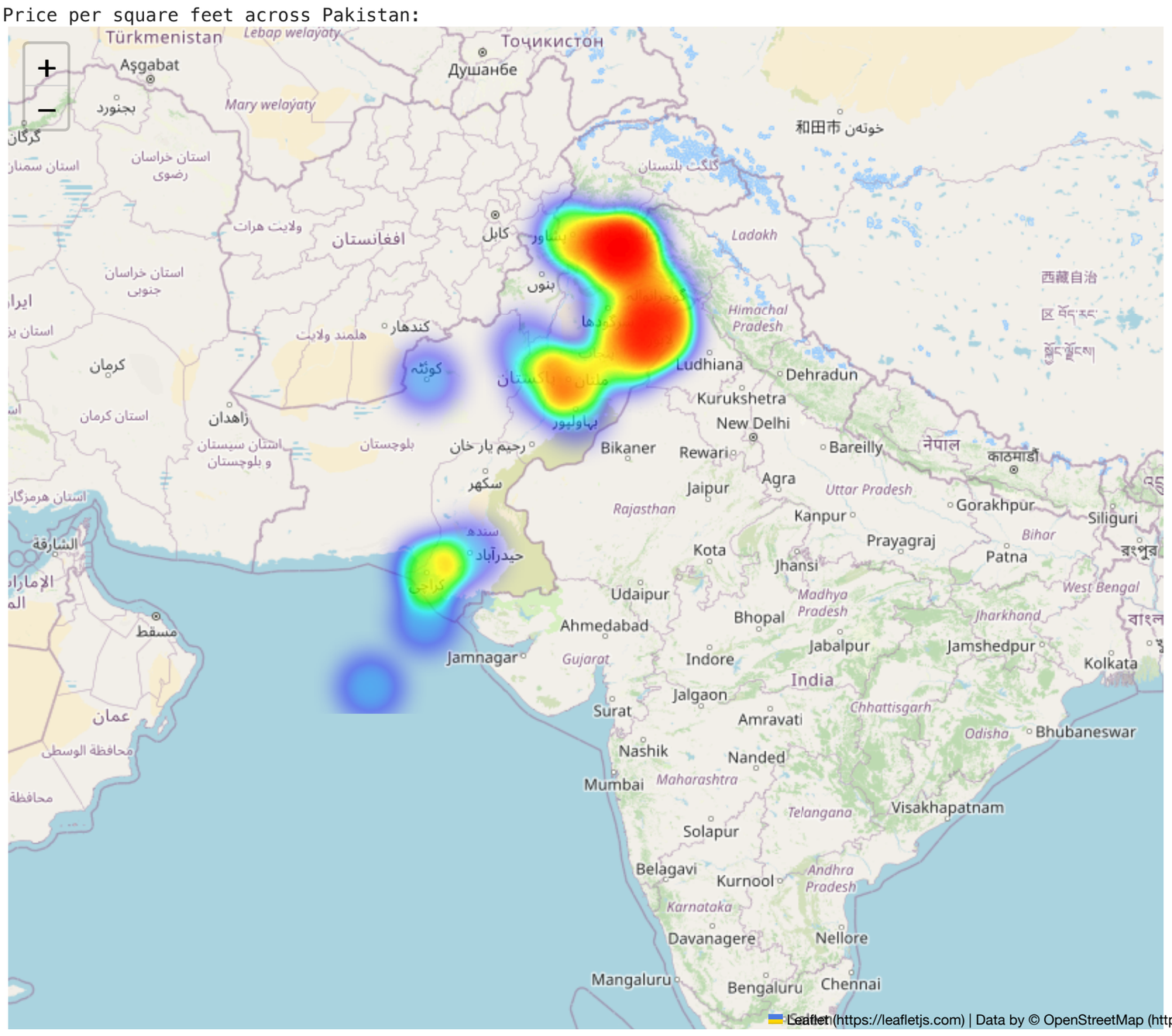
As expected, the most expensive property listing are concentrated around major cities, with the most around Islamabad and Lahore, and all major cities of Pakistan.

However, these were absolute prices and we were curious to know how price per square feet is different across the map. Luckily we had the size of every property and we can calculate this:

In [98]:
```
df['price_per_sqft'] = df['price'] / df['size_sqft']
df = df.dropna(subset=['latitude', 'longitude', 'price_per_sqft', 'size_sqft'])
df = df[df['size_sqft'] > 0]  # Ensure square footage is greater than zero

# Create a base map
map_center = [df['latitude'].mean(), df['longitude'].mean()]
map = folium.Map(location=map_center, zoom_start=5)
heatmap_data = df[['latitude', 'longitude', 'price_per_sqft']].values.tolist()
heatmap = HeatMap(heatmap_data)
map.add_child(heatmap)
print("Price per square feet across Pakistan:")
map
```

```
Price per square feet across Pakistan:
```

Out[98]:



Despite our excitement, this change did not reveal any new discoveries, other than the fact that expensive properties are usually also expensive by square feet.

Our 3 main Research Questions were:

"What is the average price of properties in different areas of Pakistan?"

Our first question is more descriptive in nature. For example, what's the standard dev, is pricing left-skew or right-skewed? Which cities have more spread-out pricings and which ones have narrower pricing? What if instead of comparing absolute price we compare per-square-feet pricing? And so on... Our violin plots answered these questions for the most part. I feel like this does not need further analysis, and so we will be consdidering this question answered here and moving forward with our next two questions in our further analysis.

"What are the hotspots for expensive properties in Pakistan, and the hotspots within each city?"

Once we realized that every Zameen.com listing includes its latitude and longitude coordinates, we just knew we had to make a hotspot map. It would be interesting to visualize, for example, which parts of Islamabad are considered more expensive than others (as if entirety of Islamabad isn't expensive enough already).

Our heatmap answers these questions extensively. However, there are still a few things we would like to learn more about each city's sub-area more extensively, moreover we still want to learn whether the presence of other listing influences the prices.

"Can we predict the price of a property based on its characteristics such as location, size, and number of bedrooms/bathrooms?"

Probably our most practical question, we would like to know which factors are the best at predicting the price of a property, and then based off it we can create a model which takes parameters and can output an appropriate price for it. We could also use this to find out if a property is overpriced/underpriced.

In order to answer this question we will need to develop a predictive Machine Learning model, or use linear regression etc. This will need further work in future deliverables.

Our analysis will focus on these points:

1. To what extent do qualitative and quantitive features such as area size, bedrooms, location, and property type, influence property prices in different cities and provinces.
2. Can we develop a predictive model to recommend an optimal pricing strategy for property listing?

Price Predictive Model

In [99]:
```
df['price_in_crore'] = df['price'] / 1e7  # Convert price to crore
df['size_marla'] = df['size_sqft'] / 225
df['log_size'] = np.log1p(df['size_marla'])
features_columns = ['log_size', 'bedrooms', 'bathrooms', 'city_Attock', 'city_Bahawalpur', 'city_Dera_Gazi_Khan',
                    'city_Faisalabad', 'city_Gujranwala', 'city_Gujrat', 'city_Hyderabad',
                    'city_Islamabad', 'city_Jhelum', 'city_Karachi', 'city_Lahore',
                    'city_Multan', 'city_Murree', 'city_Okara', 'city_Peshawar',
                    'city_Quetta', 'city_Rawalpindi', 'city_Sahiwal', 'city_Sargodha',
                    'city_Sheikhupura', 'city_Sialkot', 'city_Wah', 'property_type_Flats_Apartments',
                   'property_type_Houses_Property', 'property_type_Lower_Portions', 'property_type_Penthouse',
                   'property_type_Rooms', 'property_type_Upper_Portions', 'price_in_crore']
df_encoded = pd.get_dummies(df, columns=['city'], drop_first=True)
df_encoded2 = pd.get_dummies(df_encoded, columns=['property_type'], drop_first=True)
df_features = df_encoded2[features_columns]
X = df_features.drop('price_in_crore', axis = 1)
Y = df_features['price_in_crore']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state = 100, test_size = 0.30, shuffle = True)
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
ols_model = sm.OLS(Y_train, X_train).fit()
print(ols_model.summary())

df_features
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         price_in_crore   R-squared:                       0.356
Model:                            OLS   Adj. R-squared:                  0.356
Method:                 Least Squares   F-statistic:                     631.7
Date:                Tue, 11 Jun 2024   Prob (F-statistic):               0.00
Time:                        19:51:50   Log-Likelihood:            -1.2586e+05
No. Observations:               35424   AIC:                         2.518e+05
Df Residuals:                   35392   BIC:                         2.521e+05
Df Model:                          31
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                          -28.2831      1.063    -26.605      0.000     -30.367     -26.199
log_size                         8.7145      0.085    102.865      0.000       8.548       8.881
bedrooms                         0.8375      0.045     18.781      0.000       0.750       0.925
bathrooms                       -0.9574      0.042    -22.537      0.000      -1.041      -0.874
city_Attock                     -0.4883      2.426     -0.201      0.840      -5.243       4.266
city_Bahawalpur                 -1.6742      1.080     -1.551      0.121      -3.791       0.442
city_Dera_Gazi_Khan              1.4652      4.959      0.295      0.768      -8.255      11.185
city_Faisalabad                  1.7682      0.973      1.817      0.069      -0.139       3.675
city_Gujranwala                  0.7811      0.925      0.844      0.399      -1.032       2.595
city_Gujrat                     -0.4121      2.292     -0.180      0.857      -4.904       4.079
city_Hyderabad                   0.7412      1.213      0.611      0.541      -1.637       3.119
city_Islamabad                   5.3218      0.883      6.025      0.000       3.591       7.053
city_Jhelum                      0.7017      1.277      0.549      0.583      -1.801       3.204
city_Karachi                     2.3478      0.884      2.657      0.008       0.616       4.080
city_Lahore                      1.5458      0.881      1.754      0.079      -0.182       3.273
city_Multan                      0.1489      0.910      0.164      0.870      -1.635       1.933
city_Murree                      5.2089      1.226      4.248      0.000       2.805       7.612
city_Okara                       2.4851      1.415      1.756      0.079      -0.289       5.259
city_Peshawar                    0.4414      0.949      0.465      0.642      -1.419       2.301
city_Quetta                      1.5553      2.353      0.661      0.509      -3.057       6.167
city_Rawalpindi                  1.2628      0.887      1.424      0.154      -0.475       3.000
city_Sahiwal                     1.3953      1.134      1.231      0.218      -0.827       3.618
city_Sargodha                    1.5396      1.798      0.856      0.392      -1.985       5.064
city_Sheikhupura                 0.4422      1.775      0.249      0.803      -3.037       3.922
city_Sialkot                     1.4000      1.034      1.354      0.176      -0.627       3.427
city_Wah                         2.1732      1.294      1.679      0.093      -0.364       4.710
property_type_Flats_Apartments  11.5196      0.551     20.904      0.000      10.439      12.600
property_type_Houses_Property   12.6003      0.536     23.522      0.000      11.550      13.650
property_type_Lower_Portions     9.9798      0.787     12.675      0.000       8.437      11.523
property_type_Penthouse         11.7229      0.981     11.951      0.000       9.800      13.646
property_type_Rooms             17.4249      2.507      6.952      0.000      12.512      22.338
property_type_Upper_Portions    10.4353      0.639     16.336      0.000       9.183      11.687
==============================================================================
Omnibus:                    96759.510   Durbin-Watson:                   2.015
Prob(Omnibus):                  0.000   Jarque-Bera (JB):      12032452287.272
Skew:                          33.584   Prob(JB):                         0.00
Kurtosis:                    2857.392   Cond. No.                         756.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Out[99]:

property_id	log_size	bedrooms	bathrooms	city_Attock	city_Bahawalpur	city_Dera_Gazi_Khan	city_Faisalabad	city_Gujranwala	city_Gujrat	city_Hyderabad	...	city_Sheik
1	3.044522	6	6	0	0	0	0	0	0	0	...	
2	3.295837	4	4	0	0	0	0	0	0	0	...	
3	1.252128	1	1	0	0	0	0	0	0	0	...	
4	2.397895	5	6	0	0	0	0	0	0	0	...	
5	2.397895	6	5	0	0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
50997	2.197225	4	4	0	0	0	0	0	0	1	...	
50998	2.555676	4	4	0	0	0	0	0	0	1	...	
50999	1.175573	0	1	0	0	0	0	0	0	1	...	
51000	2.197225	4	6	0	0	0	0	0	0	1	...	
51001	1.771557	3	3	0	0	0	0	0	0	1	...	

50607 rows × 32 columns

Analysis:

From this summary we can determine which factors are most influential in determining a property's price

We used OLS Regression using Statsmodel and devised a model combining both numeric and categorical data. The categorical data types `city` (with Abbotabad as our baseline) and `property_type` (with Farmhouse as baseline) were encoded using One-hot encoding. And size was transformed using a log scale in order to better establish a linear relation.

The adjusted-R-squared of our model came out to be 0.356, meaning that using our parameters of `log_size`, `bedrooms`, `bathrooms`, `city` and `property_type` our model can explain approximately 35.6% of the observed variation in `price` for these property listings.

Key inferences:

We can study the coefficient and p-values of each parameter to determine their significance and influence on our target `price`. The most influential factors came out to be:

- **`log_size`, `bedrooms`, `bathrooms`, `city_Islamabad`, and all `property_type` columns had p-value of 0.000 meaning they are very significant in their effect and carry strong predictive ability for `price`. If we study their coefficient values we can see `log_size` has some of the highest weight, at 8.7145. Meaning that every PERCENT increase in `log_size` of 1 unit there is an associated approximate 0.0087145 increase in the price (in crore).**
- It makes sense for bathroom and bedroom to both be significant factors, as it would be in line with the domain knowledge as well. Zameen.com includes these two metrics in EVERY listing because they too know from their domain expertise that these are important determinants in a property's pricing.
- Surprisingly the coefficient for `bathrooms` is negative and the coefficient for `bedrooms` is positive, and both are significant. We did not expect to see this, considering the strong correlation we saw between them earlier. However, both coefficients are near to zero so they are not as influential in predicting price.
- Interestingly, `city_Islamabad` is a significant predictor, with a coefficient of 5.3218. This means that relative to our baseline of Abottabad, a property in Islamabad is associated with about 5.2 crore higher price taking all other things constant. Similar trend was seen for `city_Murree` as well.
- With farmhouse as our baseline property type, all `property_type` variables came out to be siginficant predictors, with the following coefficients:

```
property_type_Flats_Apartments    11.5196
property_type_Houses_Property     12.6003
property_type_Lower_Portions       9.9798
property_type_Penthouse           11.7229
property_type_Rooms               17.4249
property_type_Upper_Portions      10.4353
```

- This means that, for exampel properties which are Flats/Apartments would be approx 11.52 crore more expensive than Farmhouse properties taking all else constant.

Predictive model

In [100…

```
def RMSE(actual, predicted):
    mse = np.mean((actual - predicted)**2)
    rmse = np.sqrt(mse)
    return rmse

thetas = ols_model.params
Y_test_pred = np.dot(X_test, thetas)
rmse_statsmodels = RMSE(Y_test, Y_test_pred)

print("RMSE for Statsmodels OLS model:", rmse_statsmodels)
sample_value = thetas * [1, np.log1p(10), 4, 4, 0, 0, 0,
       0, 0, 0, 0,
       0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0]

# print(np.sum(sample_value))
# df_features.columns
# df_features
# print(thetas)
```

```
RMSE for Statsmodels OLS model: 6.990326116389657
```

The RMSE of our model comes out to be about 7, that is a mean squared error of 7 crore. This is a relatively large unit of error, which could imply that there are still more influencing factors that we do not have data on.

In [101…

```
sample_value = thetas * [1, np.log1p(10), 6, 6, 0, 0, 0,
       0, 0, 0, 0,
       1, 0, 0, 0,
       0, 0, 0, 0,
       0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0]

print(np.sum(sample_value))
```

```
9.816251168113098
```

We implemented a user-friendly UI to input the parameters of our model:

In [102…

```
import ipywidgets as widgets
from IPython.display import display

# Create input widgets
size_input = widgets.FloatSlider(min=1, max=100, step=1, description='Size (Marla)', value=10)
bedroom_input = widgets.IntSlider(min=1, max=15, step=1, description='Bedrooms', value=3)
bathroom_input = widgets.IntSlider(min=1, max=15, step=1, description='Bathrooms', value=2)
city_input = widgets.Dropdown(options=['Attock', 'Bahawalpur', 'Dera_Gazi_Khan',
                'Faisalabad', 'Gujranwala', 'Gujrat', 'Hyderabad',
                'Islamabad', 'Jhelum', 'Karachi', 'Lahore',
                'Multan', 'Murree', 'Okara', 'Peshawar',
                'Quetta', 'Rawalpindi', 'Sahiwal', 'Sargodha',
                'Sheikhupura', 'Sialkot', 'Wah'], description='City')
property_type_input = widgets.Dropdown(options=['Flat/Apartment', 'House', 'Lower Portion', 'Penthouse', 'Room', 'Upper_Portion'], description='Property

# Create a button to trigger the prediction
predict_button = widgets.Button(description='Predict')

# Function to handle button click and show prediction
def on_predict_button_click(b):
    # Get input values
    size = size_input.value
    bedrooms = bedroom_input.value
    bathrooms = bathroom_input.value
    city = city_input.value
    property_type = property_type_input.value

    cities_order = ['Attock', 'Bahawalpur', 'Dera_Gazi_Khan',
                'Faisalabad', 'Gujranwala', 'Gujrat', 'Hyderabad',
                'Islamabad', 'Jhelum', 'Karachi', 'Lahore',
                'Multan', 'Murree', 'Okara', 'Peshawar',
                'Quetta', 'Rawalpindi', 'Sahiwal', 'Sargodha',
                'Sheikhupura', 'Sialkot', 'Wah']
    city_vector = np.zeros(len(cities_order))
    city_index = cities_order.index(city)
    city_vector[city_index] = 1

    properties = ['Flat/Apartment', 'House', 'Lower Portion', 'Penthouse', 'Room', 'Upper_Portion']
    property_type_vector = np.zeros(len(properties))
    property_type_index = properties.index(property_type)
    property_type_vector[property_type_index] = 1

    feature_vector = [1, np.log1p(size), bedrooms, bathrooms] + list(city_vector) + list(property_type_vector)

    predicted_price = np.dot(feature_vector, thetas)

    # Show the predicted price
    output_label.value = f'Predicted Price: {predicted_price:.2f} Crores'

# Attach the button click event
predict_button.on_click(on_predict_button_click)

# Output label to display the prediction
output_label = widgets.Label()

# Display the widgets
display(size_input, bedroom_input, bathroom_input, city_input, property_type_input, predict_button, output_label)
```

```
FloatSlider(value=10.0, description='Size (Marla)', min=1.0, step=1.0)
IntSlider(value=3, description='Bedrooms', max=15, min=1)
IntSlider(value=2, description='Bathrooms', max=15, min=1)
Dropdown(description='City', options=('Attock', 'Bahawalpur', 'Dera_Gazi_Khan', 'Faisalabad', 'Gujranwala', 'G…
Dropdown(description='Property Type', options=('Flat/Apartment', 'House', 'Lower Portion', 'Penthouse', 'Room'…
Button(description='Predict', style=ButtonStyle())
Label(value='')
```

Conclusion:

Research Q1 answered:

From our OLS model we learned that `log_size`, `bedrooms`, `bathrooms`, and all `property_type` were the most influential factors in determining price of a property. And it would make sense as these features are some of the first things Zameen.com highlights when you open a Property's webpage.

As for `city_Islamabad`, we saw earlier that it has the most number of ads and the highest average price for ads. It would make sense for it to come out as an influential factor. However, its influence on Zameen.com might just be inflated compared to real life, as we don't know how many of these listings are actually being sold at the price. Nobody's stopping anyone from putting up an expensive ad.

Research Q2 answered:

We were able to succesfully train and deploy a predictive model for predicting property pricings. Although not perfect, it can be helpful to people looking to find comparable pricing for

properties of their interest.

Limitations:

Despite collecting, cleaning, exploring and finally modelling, our model is still far from perfect. The adjusted R-sqaure of 0.356 is not very good and we do not expect it to be very accurate at predicting listing prices based on these parameters. Our data and model still do not account for a number of different factors such as a measure of popularity for neighborhoods/colonies etc.

Moreover, `log_size`, while helpful in establishing linear trends, seems to have skewed the predictions our model makes. From our testing we found that it has better accuracy when predicting more expensive/larger properties while is very inconsistent at lower cost properties. Our theory is that the log transformation skewed the low-price data points more, resulting in more noise in that portion, hence affecting our model's accuracy.

Another drawback of the model is built-into the nature of our data itself. While our data tries to generalize for all of Pakistan, not all cities can be fitted into the same trend. For example, if you try predicting the price of a very large property in a relativley small city (like 20 bathrooms in Swat) the model has no point of reference as there never existed any listing in Swat with such a combination. In cases like this, our model outputs completely arbitrary predictions. If you add very absurd combinations you can even get negative values... for example, try a 50 marla Room in Attock

In the future:

It is quite clear that there are still many more factors out there that play a role in a property's price, be it popularity of the location, news about future developments, the agencies and social dynamics of each area, brands (e.g., DHA/Bahria) and so much more that determines a property's price. We know from domain knowledge that nobody is perfect at "predicting" property prices and most prices are determined by the equilibrium decided by market forces.

In order to better capture the social aspect of property dealing, we could carry out sentiment analyses on textual data relating to each property, both on Zameen.com or on the wider internet. We could include a lot more features otehr than bathrooms and bedrooms, such as proximity to certain kinds of facilities, security ratings, etc. Of course, we also realize the fact that the lens we view Pakistani property listing prices will be through Zameen.com listings, which does not cover every property currently for sale in Pakistan. We do not expect to create a perfect model, however there is still room for improvement.

In [ ]: