```
from bs4 import BeautifulSoup
import pandas as pd
import requests
import matplotlib.pyplot as plt
import seaborn as sns


# Function to scrape property data for a city URL
def data_scapper(city_url):
    try:
        results = []

        for page in range(1, 10):
            url = f"{city_url}-{page}.html"
            print(url)
            response = requests.get(url)
            soup = BeautifulSoup(response.content, 'html.parser')

            # Scrape property's title, prices, location, and details
            titles = [title.text.strip() for title in soup.select('h2.c0df3811')]
            prices = [price.text.strip() for price in soup.select('span[aria-label="Price"]')]
            locations = [location.text.strip() for location in soup.select('div._162e6469')]
            details = [detail.text.strip() for detail in soup.select('span.b779b320')]

            # Ensure that lists for each attribute are of the same length
            smallest = min(len(titles), len(locations), len(prices), len(details))
            for i in range(smallest):
                results.append([titles[i], locations[i], prices[i], details[i], city_url])

            print(f"Property data scraped for {city_url}: {len(results)} properties")

        return results
    except Exception as e:
        print(f"Error scraping property data for {city_url}: {str(e)}")
        return []

def main():
    city_urls = [
        "https://www.zameen.com/Homes/Karachi-2",
        "https://www.zameen.com/Homes/Lahore-1",
        "https://www.zameen.com/Homes/Islamabad-3",
        "https://www.zameen.com/Homes/Hunza-1546",
        "https://www.zameen.com/Homes/Quetta-18"
        #doing for 5 cities for now
    ]

    all_results = []

    for city_url in city_urls:
        print(f"Scraping property data for {city_url}...")
        results = data_scapper(city_url)
        all_results.extend(results)

    if all_results:
        columns = ['Title', 'Location', 'Price', 'Details', 'City_URL']
        df = pd.DataFrame(all_results, columns=columns)

        csv_file = "zameenScrappedData.csv"
        with open(csv_file, 'w', newline='') as f:
            df.to_csv(f, index=False)
            print(f"Property data saved to {csv_file}")

if __name__ == '__main__':
    main()
```

```
Property data scraped for https://www.zameen.com/Homes/Islamabad-3: 125 properties
https://www.zameen.com/Homes/Islamabad-3-6.html
Property data scraped for https://www.zameen.com/Homes/Islamabad-3: 150 properties
https://www.zameen.com/Homes/Islamabad-3-7.html
Property data scraped for https://www.zameen.com/Homes/Islamabad-3: 175 properties
https://www.zameen.com/Homes/Islamabad-3-8.html
Property data scraped for https://www.zameen.com/Homes/Islamabad-3: 200 properties
https://www.zameen.com/Homes/Islamabad-3-9.html
Property data scraped for https://www.zameen.com/Homes/Islamabad-3: 225 properties
Scraping property data for https://www.zameen.com/Homes/Hunza-1546...
https://www.zameen.com/Homes/Hunza-1546-1.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
https://www.zameen.com/Homes/Hunza-1546-2.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
https://www.zameen.com/Homes/Hunza-1546-3.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
https://www.zameen.com/Homes/Hunza-1546-4.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
https://www.zameen.com/Homes/Hunza-1546-5.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
https://www.zameen.com/Homes/Hunza-1546-6.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
https://www.zameen.com/Homes/Hunza-1546-7.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
https://www.zameen.com/Homes/Hunza-1546-8.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
https://www.zameen.com/Homes/Hunza-1546-9.html
Property data scraped for https://www.zameen.com/Homes/Hunza-1546: 11 properties
Scraping property data for https://www.zameen.com/Homes/Quetta-18...
https://www.zameen.com/Homes/Quetta-18-1.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 25 properties
https://www.zameen.com/Homes/Quetta-18-2.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 26 properties
https://www.zameen.com/Homes/Quetta-18-3.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 26 properties
https://www.zameen.com/Homes/Quetta-18-4.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 26 properties
https://www.zameen.com/Homes/Quetta-18-5.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 26 properties
https://www.zameen.com/Homes/Quetta-18-6.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 26 properties
https://www.zameen.com/Homes/Quetta-18-7.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 26 properties
https://www.zameen.com/Homes/Quetta-18-8.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 26 properties
https://www.zameen.com/Homes/Quetta-18-9.html
Property data scraped for https://www.zameen.com/Homes/Quetta-18: 26 properties
Property data saved to zameenScrappedData.csv
```

```
data = pd.read_csv("/content/zameenScrappedData.csv")
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 712 entries, 0 to 711
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Title     712 non-null    object
 1   Location  712 non-null    object
 2   Price     712 non-null    object
 3   Details   712 non-null    object
 4   City_URL  712 non-null    object
dtypes: object(5)
memory usage: 27.9+ KB
```

```
data.isna().sum()
```

```
Title       0
Location    0
Price       0
Details     0
City_URL    0
dtype: int64
```

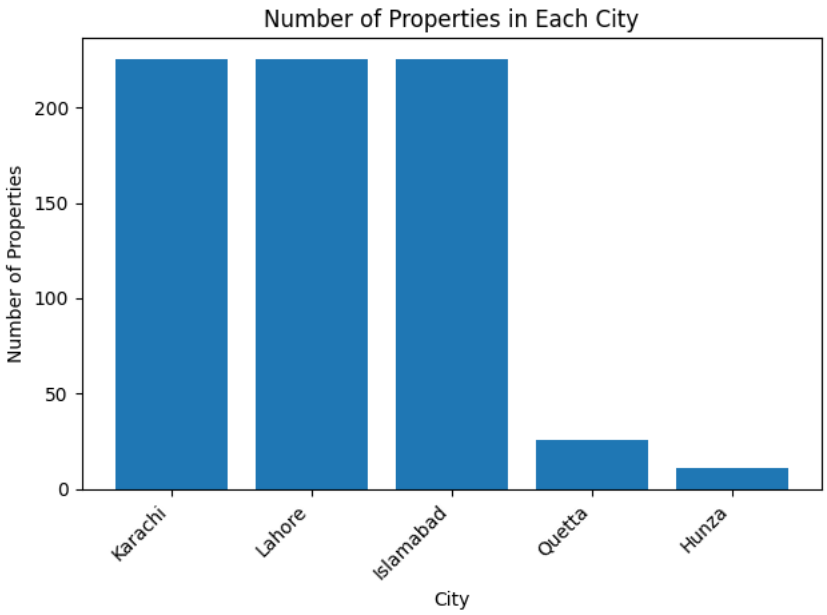
```
# Extract city names from the City_URL column
def extract_city_name(url):
    parts = url.split('/')
    if len(parts) >= 4:
        city_name = parts[4].split('-')[0].capitalize()
```

```
        return city_name
    else:
        return None

data['City'] = data['City_URL'].apply(extract_city_name)

# Count the number of properties for each city
city_counts = data['City'].value_counts()

# Create a bar chart showing the number of properties for each city
plt.bar(city_counts.index, city_counts.values)
plt.xlabel('City')
plt.ylabel('Number of Properties')
plt.title('Number of Properties in Each City')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```




```
data["Price"].unique()
```

```
            '22.88 Lakh', '1.75 Crore', '2.2 Crore', '16.5 Crore', '7.5 Crore',
            '15.5 Crore', '7.95 Crore', '2.5 Crore', '6.4 Crore', '1.35 Crore',
            '5.5 Crore', '3.4 Crore', '65 Lakh', '10.5 Crore', '3.28 Crore',
            '4.9 Crore', '11.5 Crore', '14 Crore', '21 Crore', '2.1 Crore',
            '4.7 Crore', '2 Crore', '7.4 Crore', '6.6 Crore', '3.48 Crore',
            '1.45 Crore', '3.8 Crore', '45.5 Lakh', '4 Crore', '9.5 Crore',
            '2.07 Crore', '3.5 Crore', '1.9 Crore', '2.35 Crore', '8 Crore',
            '9 Crore', '16 Crore', '14.5 Crore', '10 Crore', '17 Crore',
            '2.05 Crore', '9.25 Crore', '7.45 Crore', '21.05 Lakh',
            '5.2 Crore', '9.7 Crore', '5 Crore', '1.28 Crore', '3.15 Crore',
            '1.1 Crore', '12.8 Crore', '2.25 Crore', '13 Crore', '13.4 Crore',
            '4.5 Crore', '7.47 Crore', '4.75 Crore', '1.3 Crore', '60 Lakh',
            '1.8 Crore', '55 Lakh', '2.65 Crore', '4.55 Crore', '1.85 Crore',
            '4.8 Crore', '2.99 Crore', '2.95 Crore', '2.7 Crore', '71.23 Lakh',
            '1.5 Crore', '27.84 Lakh', '20 Crore', '1.05 Crore', '9.3 Crore',
            '3.65 Crore', '2.15 Crore', '1.01 Crore', '84 Lakh', '90 Lakh',
            '2.59 Crore', '4.6 Crore', '2.6 Crore', '1.2 Crore', '3.7 Crore',
            '12 Crore', '8.95 Crore', '3 Crore', '36 Crore', '1.7 Crore',
            '50 Lakh', '2.53 Crore', '1.95 Crore', '19.9 Crore', '13.5 Crore',
            '3.25 Crore', '6.5 Crore', '3.75 Crore', '1.65 Crore',
            '2.23 Crore', '4.68 Crore', '6.25 Crore', '8.7 Crore', '28 Lakh',
            '62 Lakh', '1.6 Crore', '4.2 Crore', '3.9 Crore', '17.54 Lakh',
            '7.5 Lakh', '85 Lakh', '3.85 Crore', '80 Lakh', '6.75 Crore',
            '8.75 Crore', '3.1 Crore', '79.5 Lakh', '80.3 Lakh', '5.99 Crore',
            '4.95 Crore', '18.95 Crore', '5.25 Crore', '6.35 Crore',
            '2.45 Crore', '2.55 Crore', '5.75 Crore', '13.95 Crore',
            '3.35 Crore', '52.7 Lakh', '2.85 Crore', '7.65 Crore',
            '14.25 Crore', '4.25 Crore', '1.99 Crore', '68.9 Lakh', '45 Crore',
            '6.86 Crore', '37 Crore', '7.25 Crore', '3.53 Crore', '81 Lakh',
            '2.3 Crore', '2.9 Crore', '2.75 Crore', '7.41 Crore', '7.35 Crore',
            '5.7 Crore', '2.13 Crore', '4.4 Crore', '1.57 Crore', '1.48 Crore',
            '7.75 Crore', '3.62 Crore', '14.75 Crore', '8.5 Crore',
```

```
       '4.1 Crore', '27.35 Lakh', '2.18 Crore', '46 Lakh', '1 Crore',
       '1.94 Crore', '1.13 Crore', '15 Lakh', '15 Crore', '40 Crore',
       '7.85 Crore', '7.15 Crore', '4.45 Crore', '6.65 Crore',
       '3.99 Crore', '2.34 Crore', '35 Crore', '50 Crore', '1.2 Arab',
       '18 Crore', '22.9 Crore', '19 Crore', '18.25 Crore', '7.7 Crore',
       '1.88 Crore', '1.03 Crore', '7.89 Crore', '2.8 Crore',
       '8.45 Crore', '6.85 Crore', '7.18 Crore', '12.5 Crore',
       '1.96 Crore', '5.65 Crore', '2.28 Crore', '26.39 Lakh', '6 Crore',
       '4.3 Crore', '4.15 Crore', '13.25 Crore', '7 Crore', '66.3 Lakh',
       '2.01 Crore', '9.6 Crore', '86.36 Lakh', '5.95 Crore',
       '8.25 Crore', '5.6 Crore', '9.95 Crore', '10.85 Crore',
       '63.09 Lakh', '43.99 Lakh', '10.8 Crore', '1.18 Crore',
       '11.9 Crore', '30 Crore', '9.4 Crore', '15.05 Lakh', '1.63 Crore',
       '5.1 Crore', '8.4 Crore', '6.2 Crore', '3.3 Crore', '10.2 Crore',
       '4.65 Crore', '83.7 Lakh', '9.9 Crore', '28 Crore', '58.5 Crore',
       '75 Crore', '61.58 Lakh', '11.75 Crore', '1.55 Arab', '64 Crore',
       '47 Crore', '24.5 Crore', '7.8 Crore', '79.79 Lakh', '77 Lakh',
       '19.75 Crore', '48 Crore', '26 Crore', '8.3 Crore', '10.25 Crore',
       '1.21 Crore', '32 Lakh', '75 Lakh', '17.5 Crore', '45.5 Crore',
       '28.1 Crore', '23 Crore', '10.9 Crore', '63 Crore', '11 Crore',
       '42 Crore', '50.5 Crore', '1.37 Crore', '27 Crore', '24 Crore',
       '27.5 Crore', '92 Lakh', '1.86 Crore', '235', '25 Thousand', '787',
       '6.78 Thousand', '456', '91.92 Lakh', '51 Lakh', '17.66 Lakh',
       '48.5 Lakh', '44.9 Lakh', '68.61 Lakh', '39.46 Lakh', '3.2 Crore',
       '40 Lakh'], dtype=object)
```

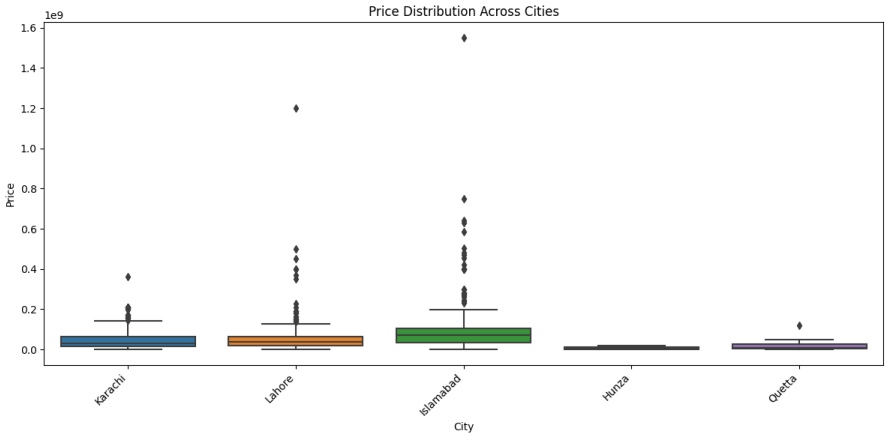
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 712 entries, 0 to 711
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Title     712 non-null    object
 1   Location  712 non-null    object
 2   Price     712 non-null    object
 3   Details   712 non-null    object
 4   City_URL  712 non-null    object
 5   City      712 non-null    object
dtypes: object(6)
memory usage: 33.5+ KB
```

```
def convert_price(price_str):
    if 'Crore' in price_str:
        price_str = price_str.replace(' Crore', '')
        price = float(price_str) * 10000000  # 1 crore = 10 million
    elif 'Lakh' in price_str:
        price_str = price_str.replace(' Lakh', '')
        price = float(price_str) * 100000  # 1 lakh = 100000
    elif 'Thousand' in price_str:
        price_str = price_str.replace(' Thousand', '')
        price = float(price_str) * 1000  # 1 thousand = 1000
    elif 'Arab' in price_str:
        price_str = price_str.replace(' Arab', '')
        price = float(price_str) * 1000000000  # 1 Arab = 1 billion
    else:
        price = float(price_str.replace(',', ''))
    return price

# Apply the function to the Price column
data['Price'] = data['Price'].apply(convert_price)

# Create a box plot to visualize price distribution across cities
plt.figure(figsize=(12, 6))
sns.boxplot(data=data, x='City', y='Price')
plt.xlabel('City')
plt.ylabel('Price')
plt.title('Price Distribution Across Cities')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```




```
data.describe()
```

	Price
count	7.120000e+02
mean	6.879886e+07
std	1.121432e+08
min	2.350000e+02
25%	1.750000e+07
50%	3.700000e+07
75%	7.750000e+07
max	1.550000e+09