How many properties are listed for sale, and in which cities?

In [5]:
```
properties_for_sale = df[df['purpose'] == 'For Sale']

total_properties_for_sale = len(properties_for_sale)
print("\nProperties listed for sale:")
print(f"Total properties for sale: {total_properties_for_sale}")

properties_by_city = properties_for_sale.groupby('city').size().reset_inde
print("\nProperties for sale in each city:")
print(properties_by_city)

```

```
Properties listed for sale:
Total properties for sale: 70947

Properties for sale in each city:
         city  properties_count
0  Faisalabad              1611
1   Islamabad              8794
2     Karachi             27210
3      Lahore             26221
4  Rawalpindi              7111
```

Location Analysis:

Which locations have the highest and lowest average property prices?

In [6]:
```
average_price_by_location = df.groupby('location')['price'].mean().sort_va
highest_avg_price_location = average_price_by_location.idxmax()
lowest_avg_price_location = average_price_by_location.idxmin()

print(f"Highest average price location: {highest_avg_price_location} (Aver
print(f"Lowest average price location: {lowest_avg_price_location} (Averag


```

```
Highest average price location: TECH Society (Average Price: Rs42500000.00)
Lowest average price location: Beaumont Road (Average Price: Rs16000.00)
```

Price Analysis

Are there outliers or high-value properties in the dataset?

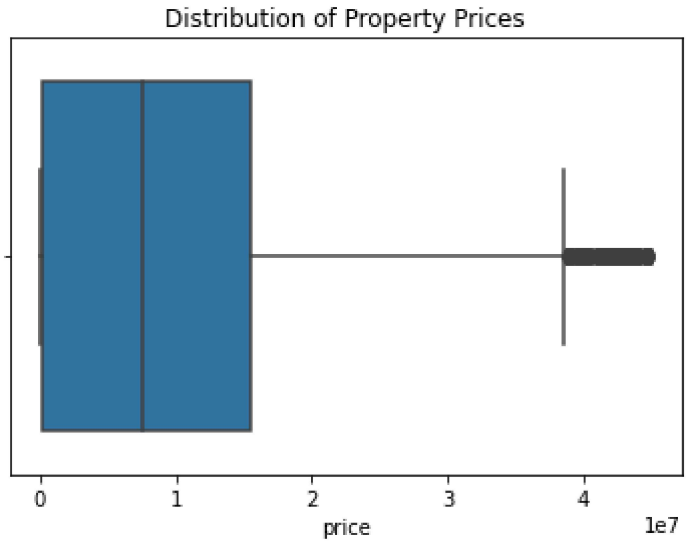
In [10]:
```
# outliers
sns.boxplot(x='price', data=df)
plt.title("Distribution of Property Prices")
plt.show()

correlation_matrix = df[['price', 'bedrooms', 'baths']].corr()

print("\nCorrelation Matrix:")
print(correlation_matrix)


sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix: Property Price, Bedrooms, and Bathrooms")
plt.show()

```

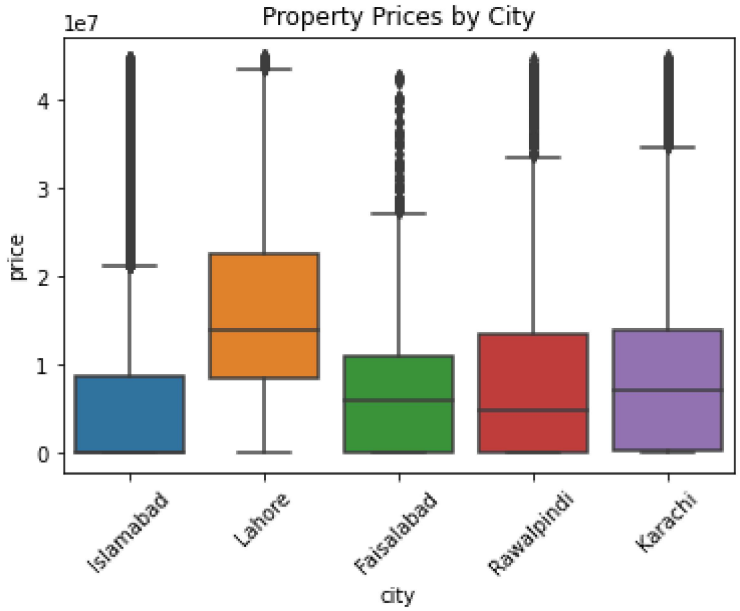



```
Correlation Matrix:
              price  bedrooms     baths
price     1.000000  0.464393  0.496222
bedrooms  0.464393  1.000000  0.863885
baths     0.496222  0.863885  1.000000
```

City Comparison

Question: How do property prices differ between cities?

In [11]:
```
sns.boxplot(x='city', y='price', data=df)
plt.title("Property Prices by City")
plt.xticks(rotation=45)
plt.show()
```

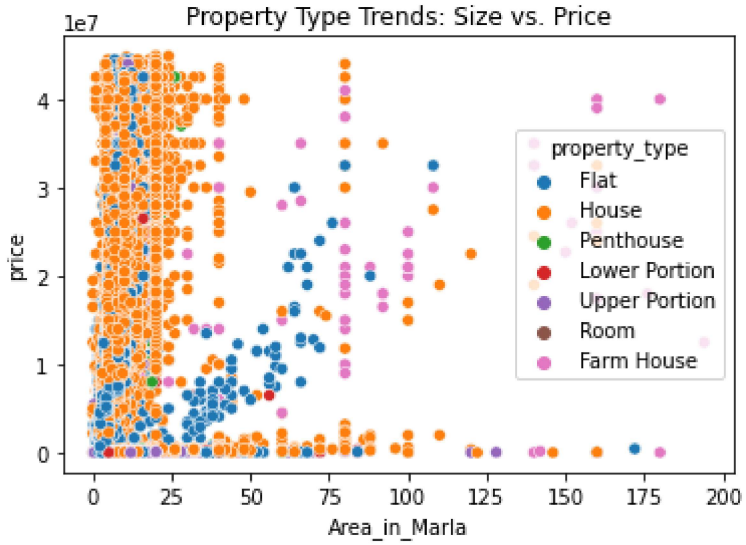


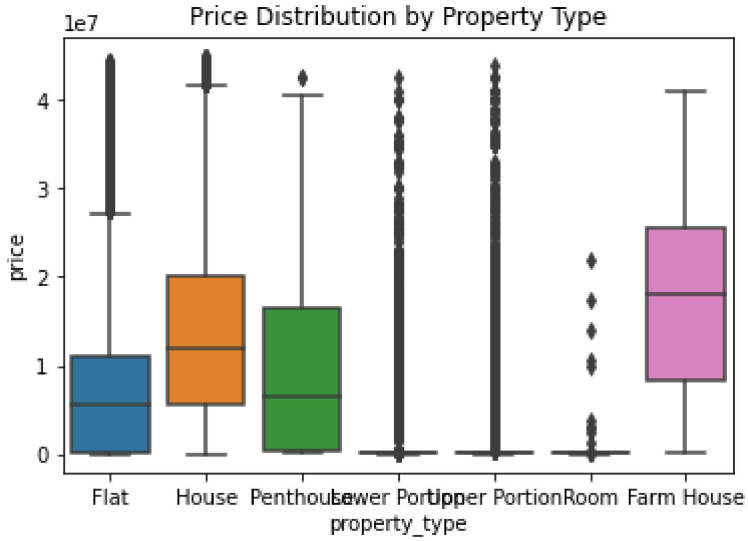
Are there trends or patterns specific to flats, houses, or other property types?

In [16]:
```
sns.scatterplot(x='Area_in_Marla', y='price', hue='property_type', data=df
plt.title("Property Type Trends: Size vs. Price")
plt.show()

sns.boxplot(x='property_type', y='price', data=df)
plt.title("Price Distribution by Property Type")
plt.show()

```





Predictive Modeling:

In [21]:

```
# One-hot encoding
df = pd.get_dummies(df, columns=['location', 'property_type', 'city', 'pur


X = df.drop(columns=['price'])
y = df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r


linear_reg_model = LinearRegression()
linear_reg_model.fit(X_train, y_train)


# Predictions
linear_reg_predictions = linear_reg_model.predict(X_test)


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

df = pd.read_csv('house_prices.csv')

# Select features and target variable
X = df[['location', 'bedrooms', 'Area_in_Marla']]
y = df['price']


X = pd.get_dummies(X, columns=['location'], drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r

model = LinearRegression()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

# Evaluation
r2 = r2_score(y_test, predictions)
rmse = mean_squared_error(y_test, predictions, squared=False)  # RMSE
print(f"R-squared (R2): {r2:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

```

```
R-squared (R2): -922385505381862784.0000
Root Mean Squared Error (RMSE): 10275059879753562.00
```

In [22]:
```
# Random Forest Regression model
random_forest_model = RandomForestRegressor()
random_forest_model.fit(X_train, y_train)

```

Out[22]:
```
RandomForestRegressor()
```

In [23]:
```
random_forest_predictions = random_forest_model.predict(X_test)
```

In [24]:
```
# Evaluate performance
def evaluate_model(predictions, model_name):
    r2 = r2_score(y_test, predictions)
    rmse = sqrt(mean_squared_error(y_test, predictions))
    print(f"\nPerformance of {model_name} Model:")
    print(f"R-squared (R2): {r2:.4f}")
    print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

```

In [25]:
```
evaluate_model(random_forest_predictions, random_forest_model)
```

```
Performance of RandomForestRegressor() Model:
R-squared (R2): 0.5474
Root Mean Squared Error (RMSE): 7197282.44
```