In [1]:
```
import pandas as pd
import numpy as np
from time import sleep
from random import uniform
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import seaborn as sns
import matplotlib.pyplot as plt
from selenium.webdriver.chrome.options import Options
import undetected_chromedriver as uc
from bs4 import BeautifulSoup
import re
import asyncio
import nest_asyncio
import aiohttp
from multiprocessing import Pool
import os
import glob
from ast import literal_eval
from collections import Counter
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, MultiLabelBinarize
from sklearn.ensemble import RandomForestRegressor
import keras_tuner as kt
import joblib
from sklearn.preprocessing import RobustScaler, PolynomialFeatures
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest, f_regression
from tensorflow.keras import regularizers
from scipy.stats import zscore
from sklearn.decomposition import PCA
from keras_tuner.tuners import RandomSearch
from tensorflow.keras.optimizers import Adam
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MultiLabelBinarizer, PolynomialFe
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import StackingRegressor
from xgboost import XGBRegressor
```

```
from tensorflow.keras import layers, models
import keras_tuner as kt
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import EarlyStopping, LearningRateScheduler
import tensorflow as tf
from scipy.stats import zscore

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [2]:
```
house_types = {
    'Apartment': 'apartment',
    'Flat': 'flat',
    'Condominium': 'condominium',
    'Serviced Residence': 'serviced-residence',
    '1-sty Terrace/Link House': '1-sty-terrace-link-house',
    '2-sty Terrace/Link House': '2-sty-terrace-link-house',
    '3-sty Terrace/Link House': '3-sty-terrace-link-house',
    '4-sty Terrace/Link House': '4-sty-terrace-link-house',
    '1.5-sty Terrace/Link House': '1-5-sty-terrace-link-house',
    '2.5-sty Terrace/Link House': '2-5-sty-terrace-link-house',
    '3.5-sty Terrace/Link House': '3-5-sty-terrace-link-house',
    '4.5-sty Terrace/Link House': '4-5-sty-terrace-link-house',
    'Townhouse': 'townhouse',
    'Cluster House': 'cluster-house',
    'Bungalow': 'bungalow',
    'Semi-detached House': 'semi-detached-house'
}
```

In [3]:
```
options = Options()
options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
driver = uc.Chrome(options=options)

for house_type, house_type_url in house_types.items():
    data = []
    existing_records = set()
    max_properties = 1500
    page_number = 0

    while True:
        driver.get(f'https://www.iproperty.com.my/sale/{house_type_url}/?page={page
        print(f"Scraping {house_type}, Page: {page_number}")

        try:
            WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.XPATH, "//div[contains(@class, '
            )
        except Exception:
            print(f"Failed to load data on page {page_number} for {house_type}. End
            page_number += 1
            continue

        houses = driver.find_elements(By.XPATH, "//li[contains(@class, 'ListingsLis
        print(f"Found {len(houses)} houses on page {page_number}")

        for house in houses:
            try:
                # Skip if it's a false house
                try:
                    false_house = house.find_element(By.XPATH, ".//li[contains(@cla
                    if false_house:
                        continue
                except:
                    pass

                # Title
                try:
                    title = house.find_element(By.XPATH, ".//h2[contains(@class, 'T
                except:
                    continue

                # Price
                try:
                    price_text = house.find_element(By.XPATH, ".//li[contains(@clas
                    price = re.sub(r"[^\d]", "", price_text)
                except:
                    continue

                # Price per square foot
                try:
                    price_per_sqft_text = house.find_element(By.XPATH, ".//div[cont
                    price_per_sqft_match = re.search(r"\d+(\.\d+)?", price_per_sqft
                    price_per_sqft = float(price_per_sqft_match.group()) if price_p
                except:
                    continue
```

```
        # Size
        try:
            details_text = house.find_element(By.XPATH, ".//div[contains(@c
            built_up_match = re.search(r"Built-up\s*:\s*([\d,]+)\s*sq\. ft\
            size = int(built_up_match.group(1).replace(",", "")) if built_u
        except:
            continue

        # District, State
        try:
            location = house.find_element(By.XPATH, ".//div[contains(@class
            district = location.split(",")[0].title()
            state = location.split(",")[1].title()
        except:
            continue

        # Facilities: Bedrooms, Bathrooms, Car Slots
        try:
            bed_number_text = house.find_element(By.XPATH, ".//li[contains(
            bed_number = sum(int(num.strip()) for num in bed_number_text.sp
        except:
            bed_number = None

        try:
            bath_number_text = house.find_element(By.XPATH, ".//li[contains
            bath_number = sum(int(num.strip()) for num in bath_number_text.
        except:
            bath_number = None

        try:
            car_number_text = house.find_element(By.XPATH, ".//li[contains(
            car_number = sum(int(num.strip()) for num in car_number_text.sp
        except:
            car_number = None

        try:
            link = house.find_elements(By.XPATH, ".//a[@class='depth-listin
            driver.get(link)
            WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.TAG_NAME, "main"))
            )

            # Location
            try:
                location = driver.find_element(By.XPATH, ".//h3[@class='sc-
                location = location.lstrip('-').strip()
            except:
                location = None

            property_types = driver.find_elements(By.XPATH, ".//div[@class=

            # Furnished Type
            furnished_type = "Unfurnished"
            try:
                for property_type in property_types:
```

```
                key = property_type.find_element(By.XPATH, ".//div[@cla
                if key == "Furnishing":
                    furnished_type = property_type.find_element(By.XPAT
                    break
        except:
            pass

        # Tenure
        tenure = "Freehold"
        try:
            for property_type in property_types:
                key = property_type.find_element(By.XPATH, ".//div[@cla
                if key == "Tenure":
                    tenure = property_type.find_element(By.XPATH, ".//d
                    break
        except:
            pass

        # Facilities
        try:
            facilities = [
                facility.get_attribute("textContent").strip().title()
                for facility in driver.find_elements(By.XPATH, './/div[
            ]
        except:
            facilities = None

        # Images
        try:
            img_urls = set()
            modal_button = WebDriverWait(driver, 10).until(
                EC.element_to_be_clickable((By.XPATH, '//div[contains(@
            )
            modal_button.click()
            WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.XPATH, '//div[contai
            )

            images = driver.find_elements(By.XPATH, '//div[contains(@cl
            for img in images:
                src = img.get_attribute('src')
                if src:
                    img_urls.add(src)

            close_button = driver.find_element(By.XPATH, '//div[contain
            close_button.click()
        except Exception as e:
            # print(f"Error extracting images: {e}")
            img_urls = []

    except:
        continue

    finally:
        driver.back()
```

```
                record = (
                    title, price, district, state, bed_number, location, furnished_
                )

                if record in existing_records:
                    continue

                existing_records.add(record)

                data.append({
                    "Title": title,
                    "Price": price,
                    "District": district,
                    "State": state,
                    "Bedrooms": bed_number,
                    "Location": location,
                    "Tenure": tenure,
                    "Furnished Type": furnished_type,
                    "Size": size,
                    "Facilities": facilities,
                    "Bathrooms": bath_number,
                    "Car Slots": car_number,
                    "House Type": house_type,
                    "Price per sqft": price_per_sqft,
                    "Images": img_urls,
                })
            except Exception as e:
                print(f"Error extracting house details: {e}")

        if len(data) >= max_properties:
            print(f"Reached max properties limit ({max_properties}) for {house_type
            break

        try:
            next_page_button = driver.find_element(By.XPATH, "//a[@aria-label='Go t
            if next_page_button:
                page_number += 1
                print(len(data))
        except Exception:
            print(f"No 'next page' button found on page {page_number}. Ending scrap
            break

    output_directory = "data"
    if not os.path.exists(output_directory):
        os.makedirs(output_directory)

    file_name = f"{output_directory}/property-{house_type_url}.csv"
    df = pd.DataFrame(data[:max_properties])
    df.to_csv(file_name, index=False, encoding="utf-8")
    print(f"Data saved to {file_name}")

driver.quit()
```

In [4]:

```
folder_path = 'data'
all_files = []
for house_type in house_types.values():
```

```
    pattern = os.path.join(folder_path, f'property-{house_type}.csv')
    all_files.extend(glob.glob(pattern))

combined_df = pd.concat([pd.read_csv(file) for file in all_files], ignore_index=Tru
combined_file_path = 'data/combined_property_data.csv'
combined_df.to_csv(combined_file_path, index=False)
```

In [5]:
```
df = combined_df
df.head()
```

Out[5]:

	Title	Price	District	State	Bedrooms	Location	Tenure	Furnished Type	S
0	Austin Regency (Pangsapuri Austin Perdana), Ta...	680000	Tebrau	Johor	4.0	Jalan Austin Perdana Utama, Taman Austin Perda...	Freehold	Fully Furnished	131
1	Pangsapuri Seri Baiduri, Perling	218000	Perling	Johor	3.0	Jalan Baiduri 3, 80100, Johor	Freehold	Unfurnished	75
2	Sri Intan 1, Jalan Kuching	258000	Jalan Kuching	Kuala Lumpur	3.0	No.2 Jalan Trolak 6, 51200, Kuala Lumpur	Freehold	Partly Furnished	90
3	Mentari Court, Petaling Jaya	290000	Petaling Jaya	Selangor	3.0	Jalan PJS8/9, 46150, Selangor	Leasehold	Partly Furnished	77
4	Bistari Impian Apartment, Taman Dato Onn, Joho...	350000	Johor Bahru	Johor	3.0	0 Jalan Serantau, Taman Dato Onn, 80350, Johor	Leasehold	Unfurnished	107

In [6]:
```
len(df)
```

Out[6]: 19462

In [7]:
```
df = df.drop_duplicates(subset=df.columns.difference(['Facilities']))
```

In [8]:
```
print(len(df))
```

```
19462
```

In [9]:
```
# Identify missing values in critical columns
critical_columns = ['Price', 'Size', 'Bedrooms', 'Bathrooms', 'Car Slots', 'Distric
missing_summary_critical = df[critical_columns].isnull().sum()
missing_summary_critical
```

Out[9]:
```
Price               0
Size              113
Bedrooms          153
Bathrooms          56
Car Slots        6319
District            0
State               0
Price per sqft      8
Furnished Type      0
House Type          0
Facilities          0
dtype: int64
```

In [10]:
```
# Drop rows with missing critical values
df = df.dropna(subset=['Price', 'Size', 'Bedrooms', 'Bathrooms', 'Price per sqft'],

# Drop Unnecessary Columns
columns_to_drop = ['Location', 'Title', 'Images']
df = df.drop(columns=columns_to_drop, errors='ignore')
```

In [11]:
```
# Impute missing numerical values
numerical_critical = ['Bedrooms', 'Bathrooms', 'Car Slots']
df[numerical_critical] = df[numerical_critical].fillna(df[numerical_critical].media
```

In [12]:
```
# Address Outliers for Numerical Data
numerical_columns = ['Price', 'Size', 'Price per sqft']
for col in numerical_columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
```

In [13]:
```
# Fix Inconsistent Formatting
categorical_columns = ['State', 'Tenure', 'Furnished Type', 'House Type', 'District
df[categorical_columns] = df[categorical_columns].apply(lambda x: x.str.strip().str
```

In [14]:
```
df.columns
```

Out[14]:
```
Index(['Price', 'District', 'State', 'Bedrooms', 'Tenure', 'Furnished Type',
       'Size', 'Facilities', 'Bathrooms', 'Car Slots', 'House Type',
       'Price per sqft'],
      dtype='object')
```

In [15]:
```
# Create Derived Features
```

```
df['Price per Bedroom'] = df['Price'] / df['Bedrooms']
df['Room Density'] = df['Bedrooms'] / df['Size']
```

In [16]:
```
df.to_csv('data/cleaned_combined_property_data.csv')
```

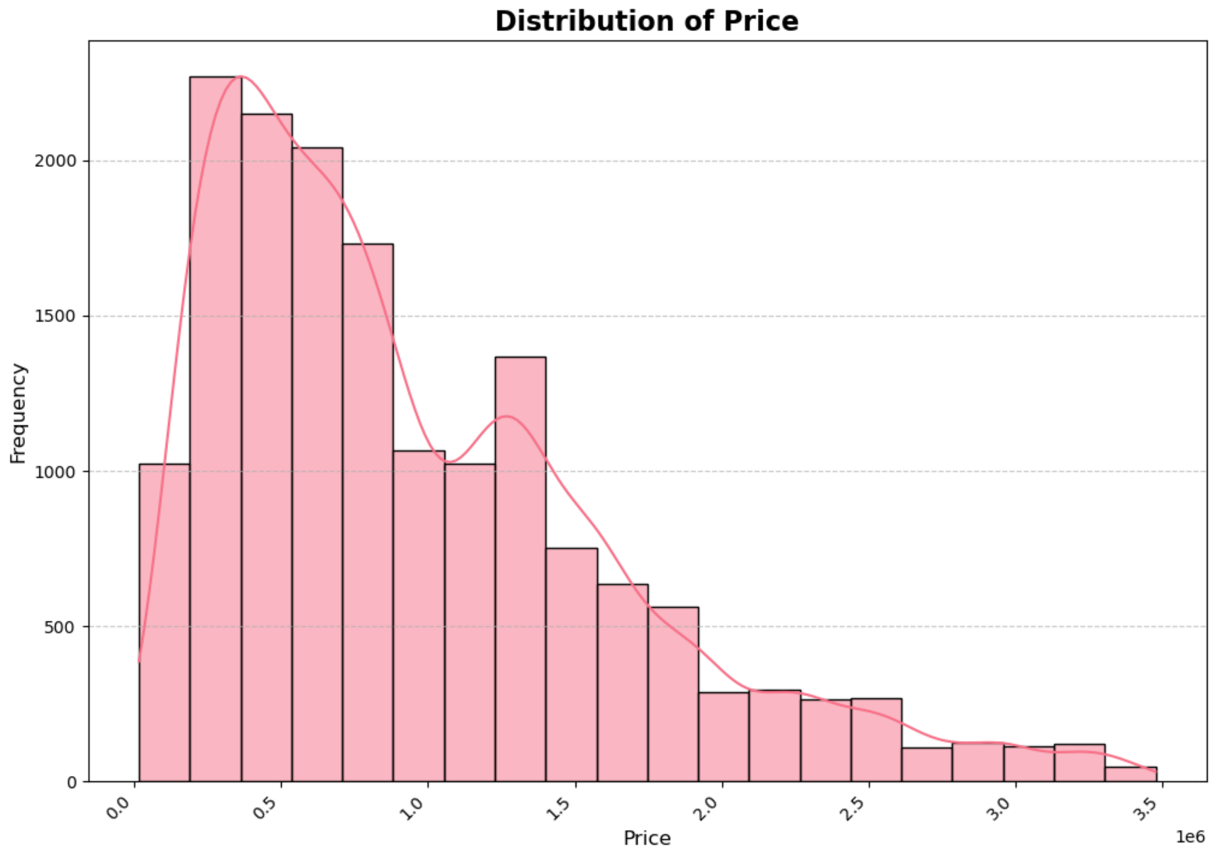
In [17]:
```
# Generate and interpret summary statistics and Validate Data Quality
summary_statistics = df.describe()
summary_statistics
```

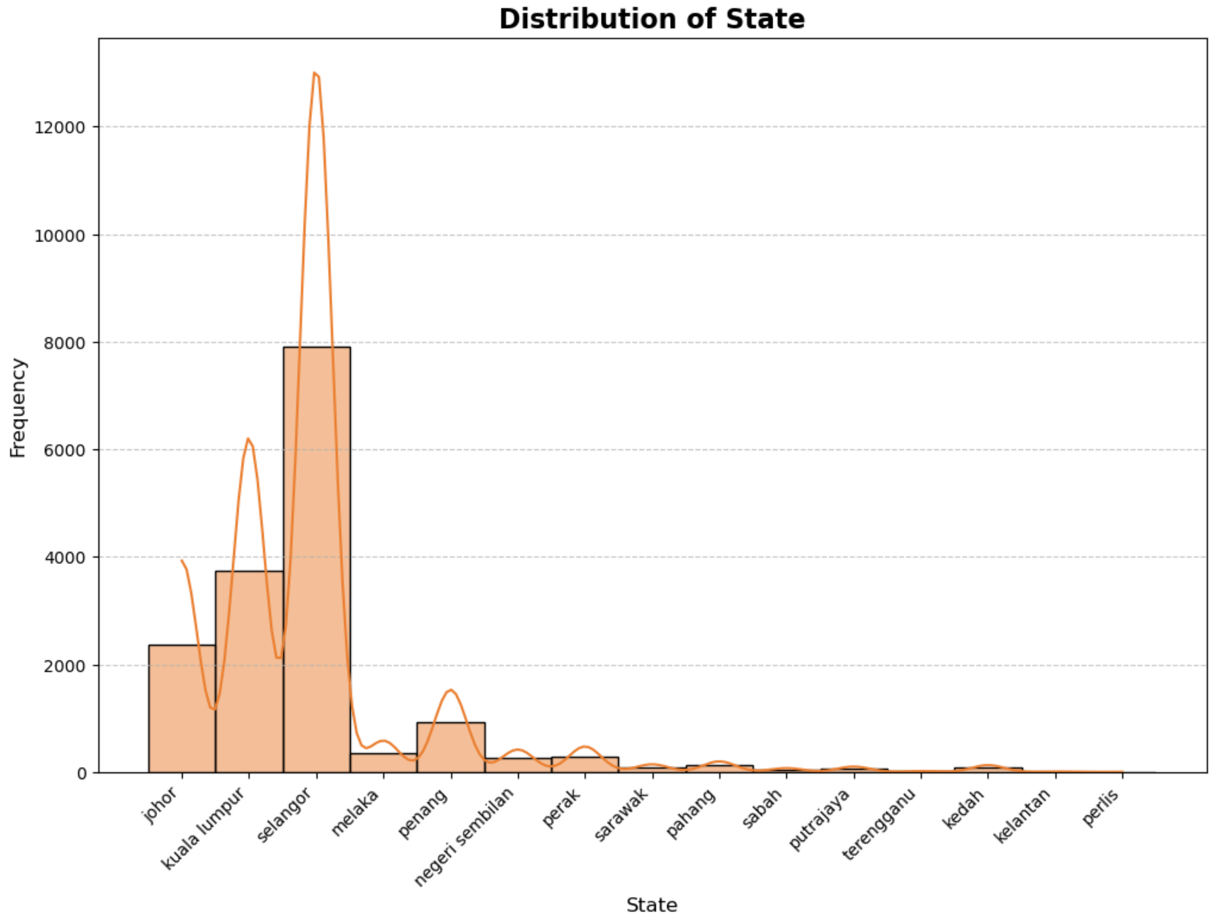
Out[17]:

	Price	Bedrooms	Size	Bathrooms	Car Slots	Price per sqft
count	1.625100e+04	16251.000000	16251.000000	16251.000000	16251.000000	16251.000000
mean	9.627161e+05	4.072672	2090.864439	3.255184	2.193219	442.764500
std	6.936631e+05	1.304835	1142.383706	1.465852	0.976628	173.587615
min	1.548800e+04	1.000000	140.000000	1.000000	1.000000	4.160000
25%	4.200000e+05	3.000000	1100.000000	2.000000	2.000000	314.875000
50%	7.680000e+05	4.000000	1900.000000	3.000000	2.000000	414.140000
75%	1.350000e+06	5.000000	2900.000000	4.000000	2.000000	542.045000
max	3.478000e+06	19.000000	5523.000000	15.000000	12.000000	965.000000

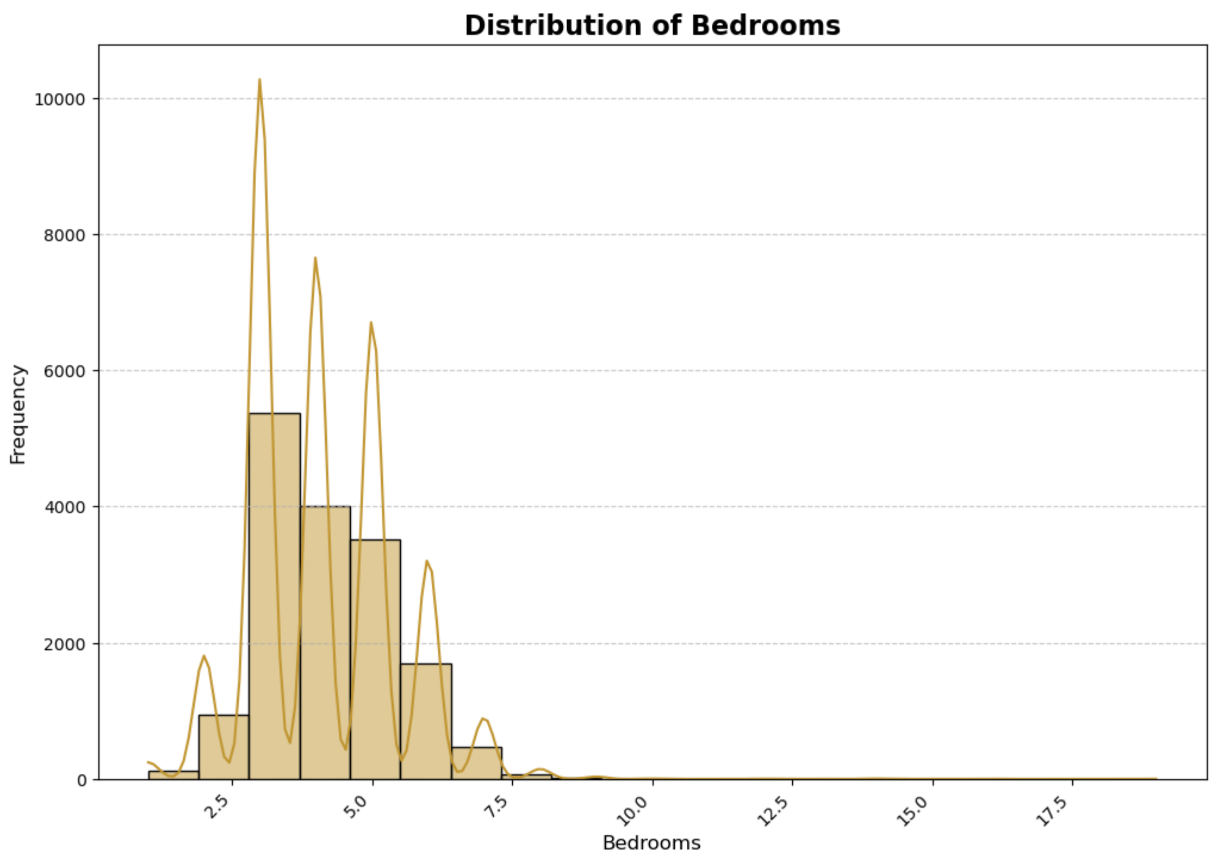
In [18]:
```
# Analyze data distributions for key variables
columns_to_plot = [
    col for col in df.columns
    if col not in ['Facilities', 'State_code', 'Tenure_code', 'Furnished Type_code'
]

colors = sns.color_palette("husl", n_colors=len(columns_to_plot) + 1)

for idx, column in enumerate(columns_to_plot):
    plt.figure(figsize=(12, 8))
    sns.histplot(data=df, x=column, bins=20, kde=True, color=colors[idx], edgecolor
    plt.title(f'Distribution of {column}', fontsize=16, fontweight='bold')
    plt.xlabel(column, fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()

df['Facilities_list'] = df['Facilities'].apply(lambda x: eval(x) if isinstance(x, s
all_facilities = [facility for sublist in df['Facilities_list'] for facility in sub
facility_expanded_df = pd.DataFrame({'Facility': all_facilities})


plt.figure(figsize=(12, 8))
sns.histplot(data=facility_expanded_df, x='Facility', kde=True, bins=20, color=colo
plt.title('Histogram of Facilities Across Houses', fontsize=16, fontweight='bold')
plt.xlabel('Facilities', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
```

```
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

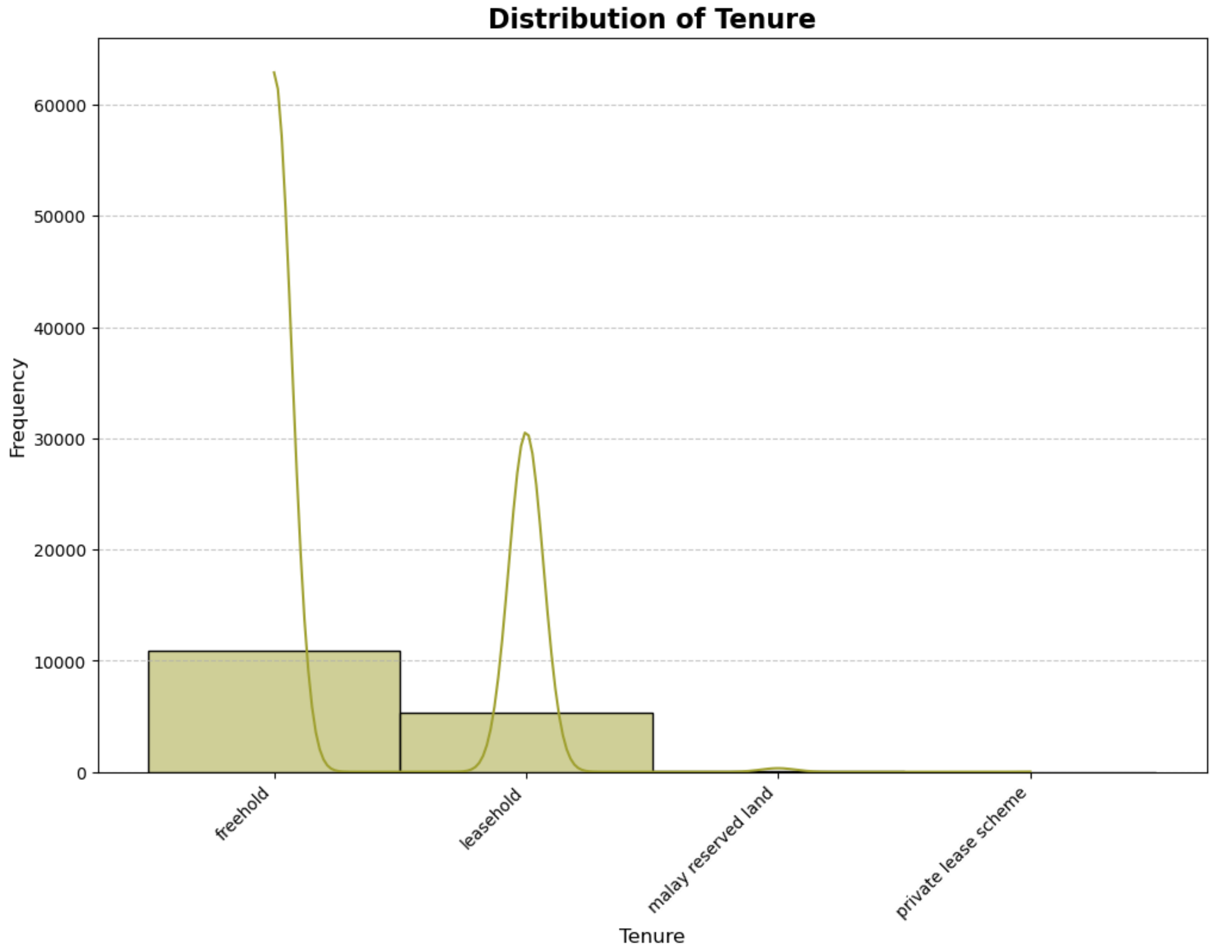


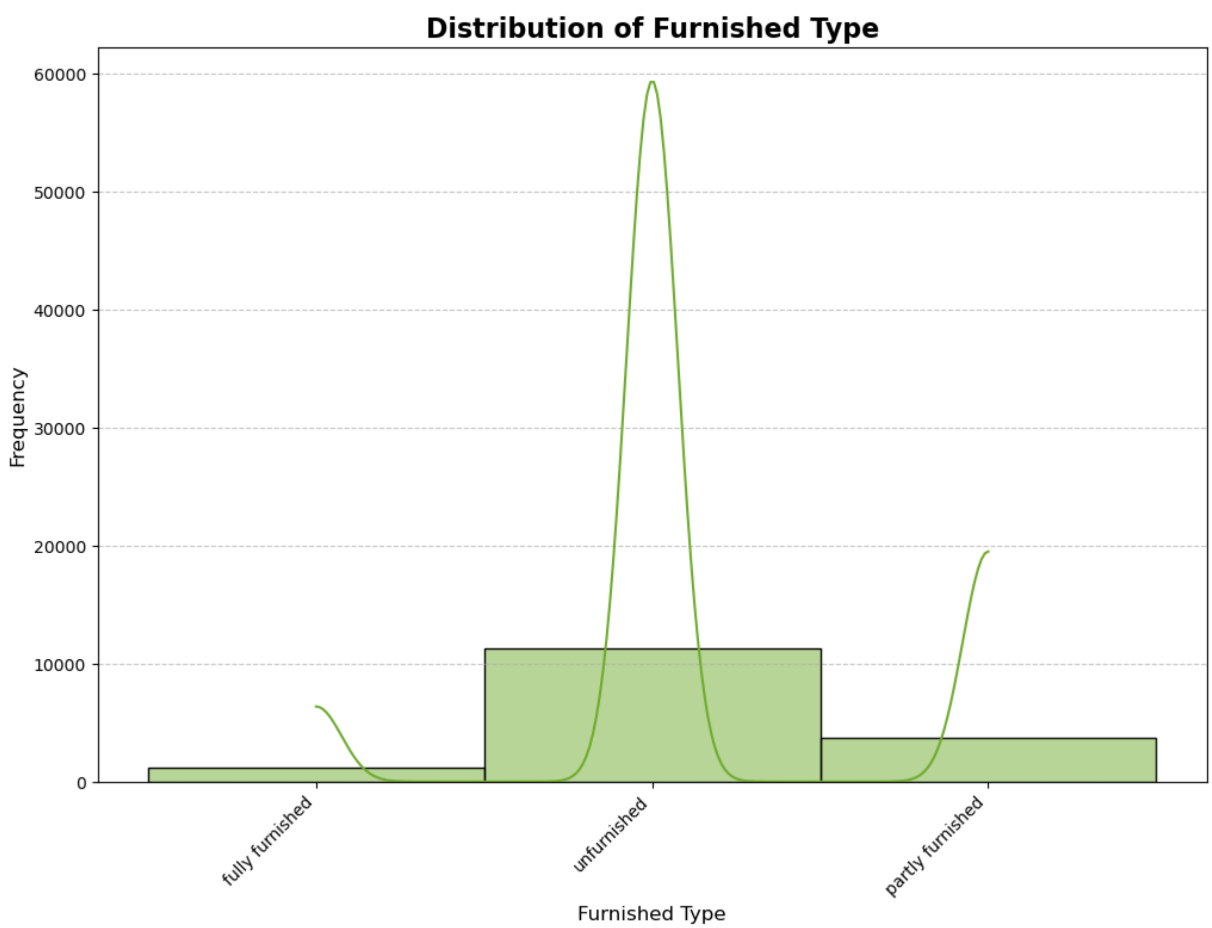


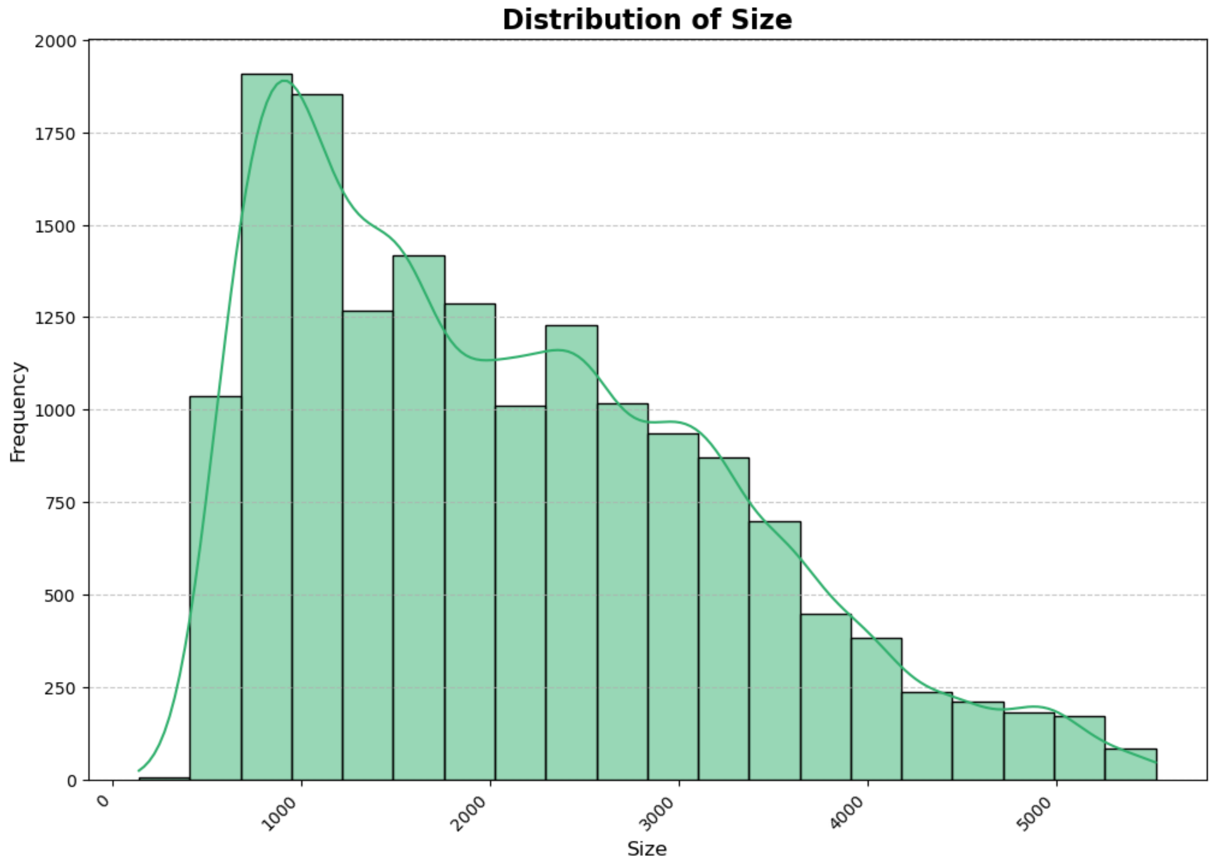
Distribution of State
Frequency
0
2000
4000
6000
8000
10000
12000
johor
kuala lumpur
selangor
melaka
penang
negeri sembilan
perak
sarawak
pahang
sabah
putrajaya
terengganu
kedah
kelantan
perlis
State

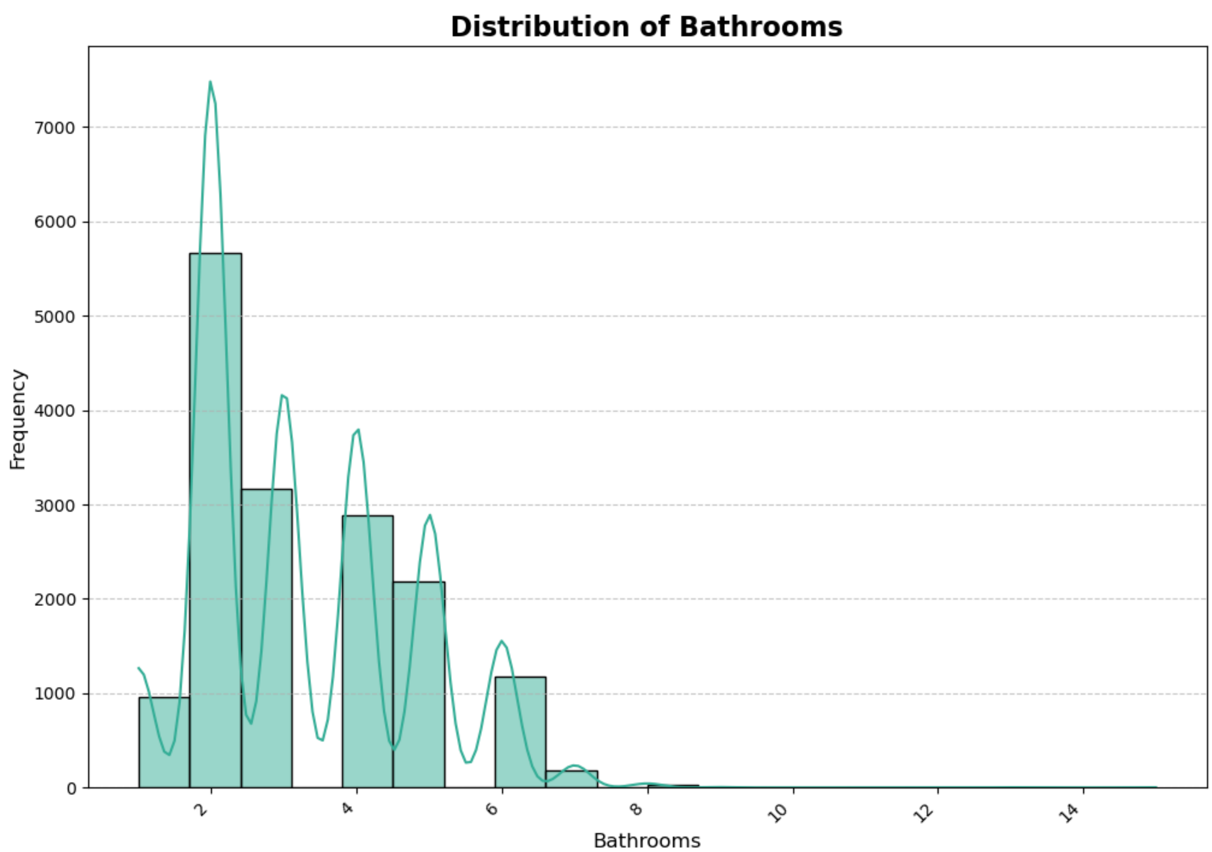


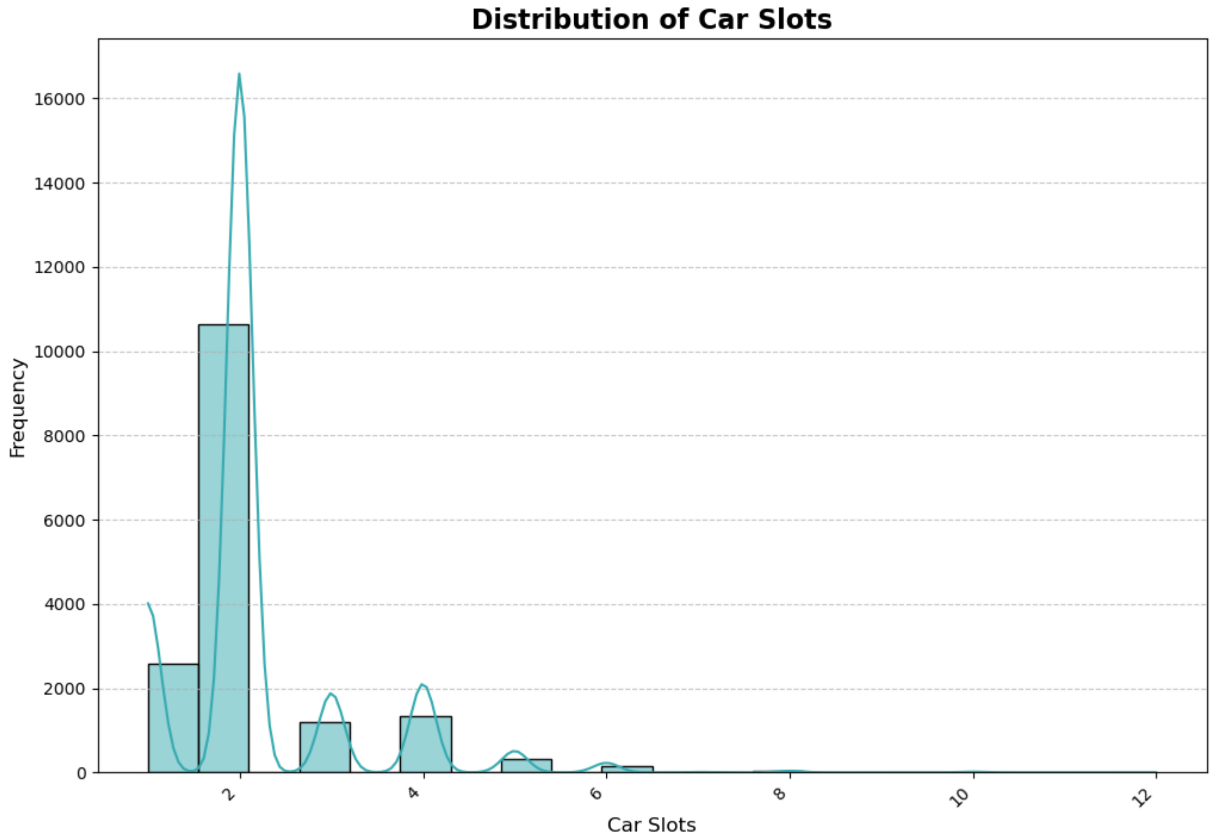
Distribution of Bedrooms
Frequency
0
2000
4000
6000
8000
10000
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Bedrooms

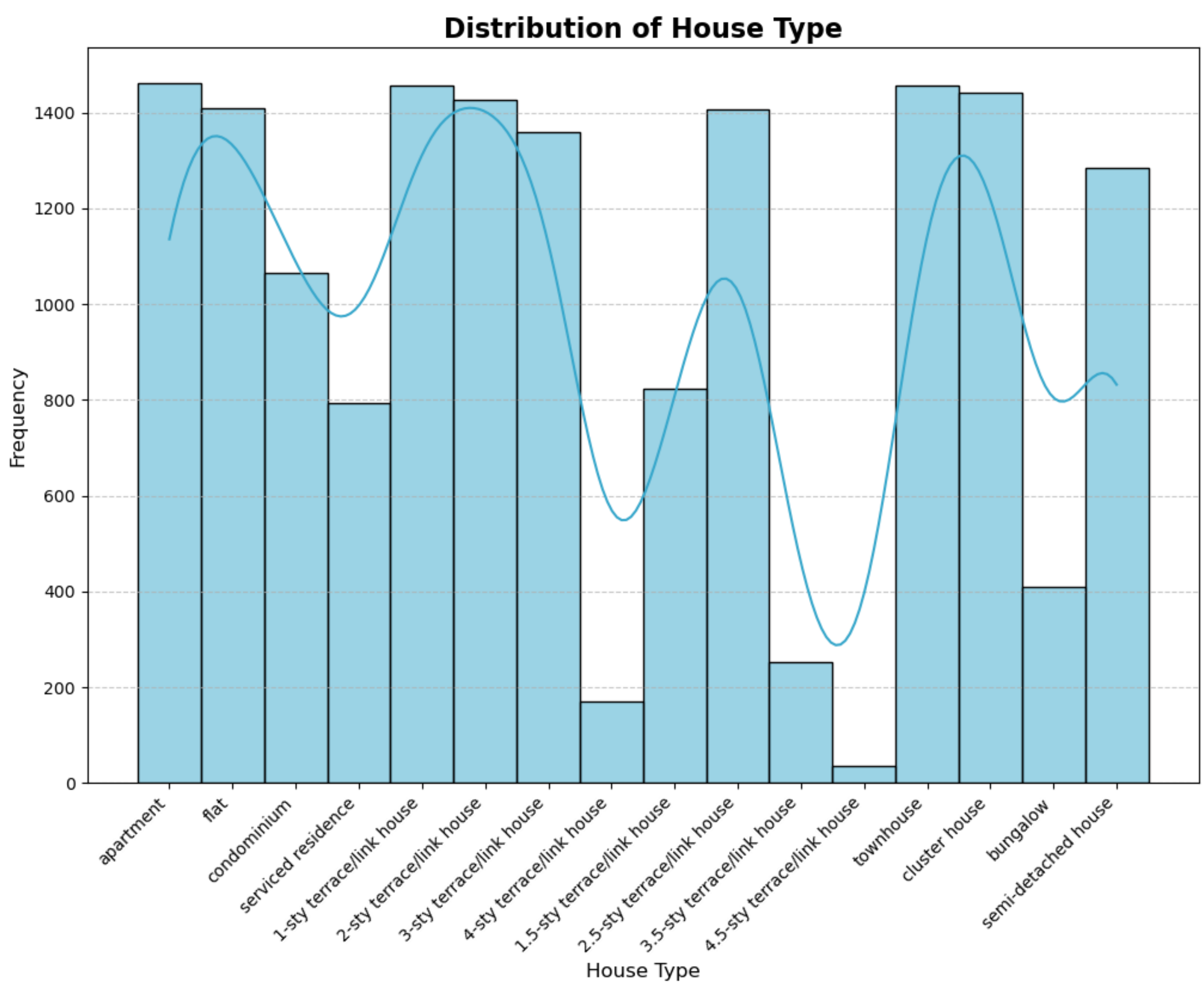

Distribution of Tenure
Frequency
60000
50000
40000
30000
20000
10000
0
freehold
leasehold
malay reserved land
private lease scheme
Tenure

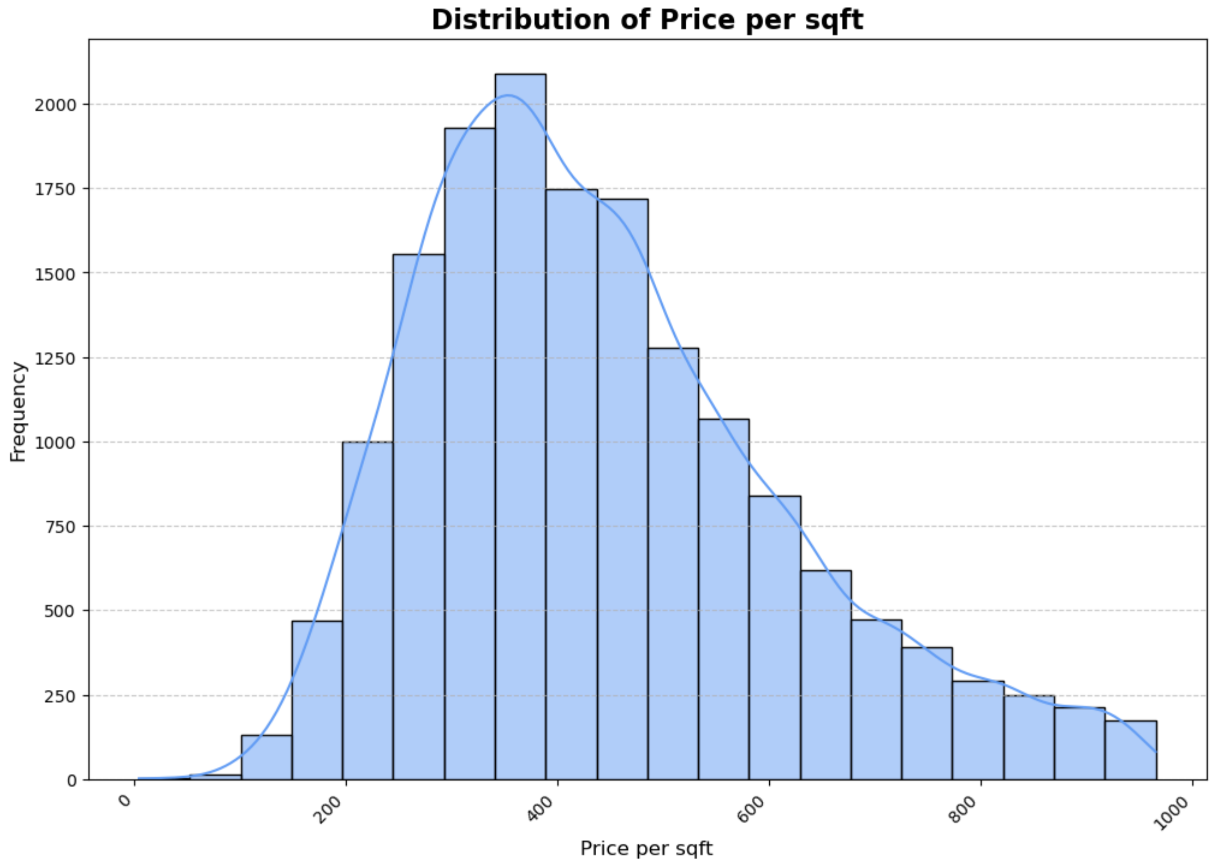

Distribution of Furnished Type
Frequency
60000
50000
40000
30000
20000
10000
0
fully furnished
unfurnished
partly furnished
Furnished Type


Distribution of Size
Frequency
Size
0
250
500
750
1000
1250
1500
1750
2000
0
1000
2000
3000
4000
5000

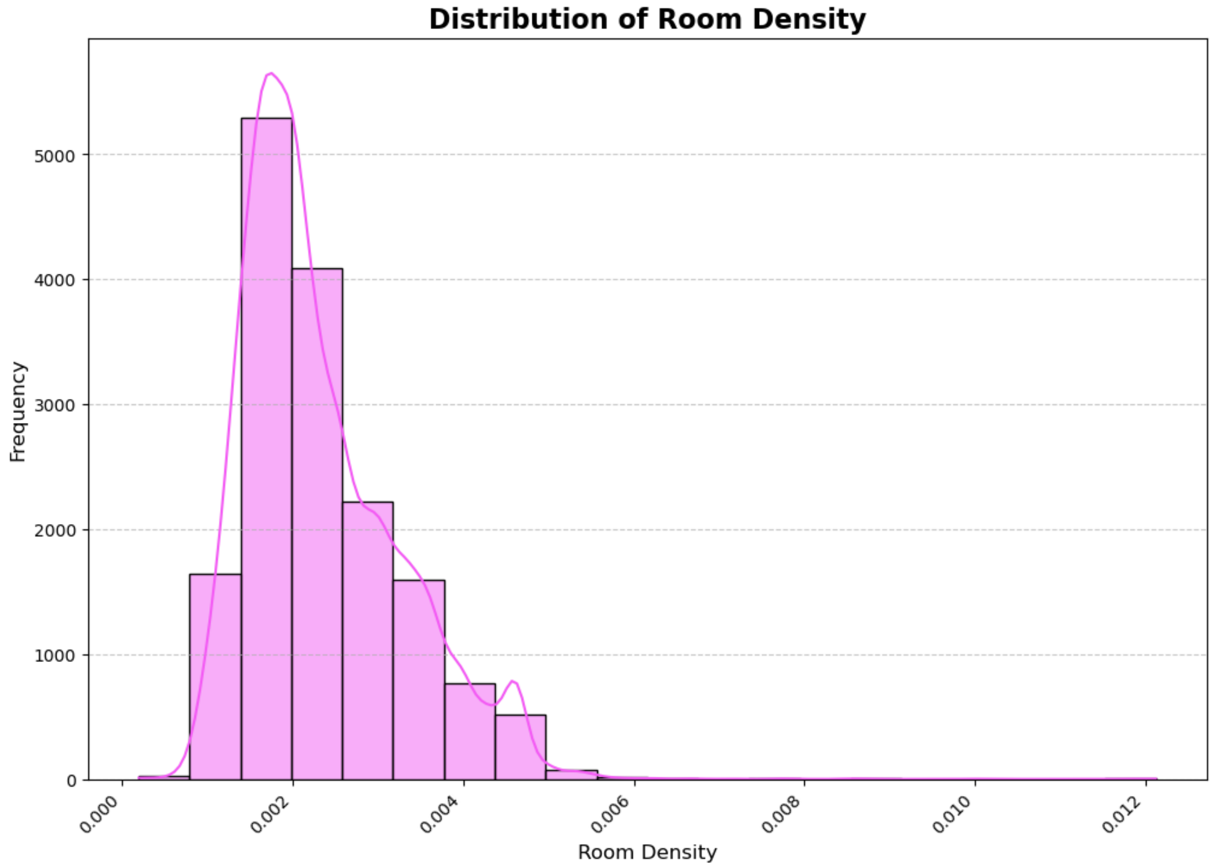

Distribution of Bathrooms
Frequency
Bathrooms
0
1000
2000
3000
4000
5000
6000
7000
2
4
6
8
10
12
14

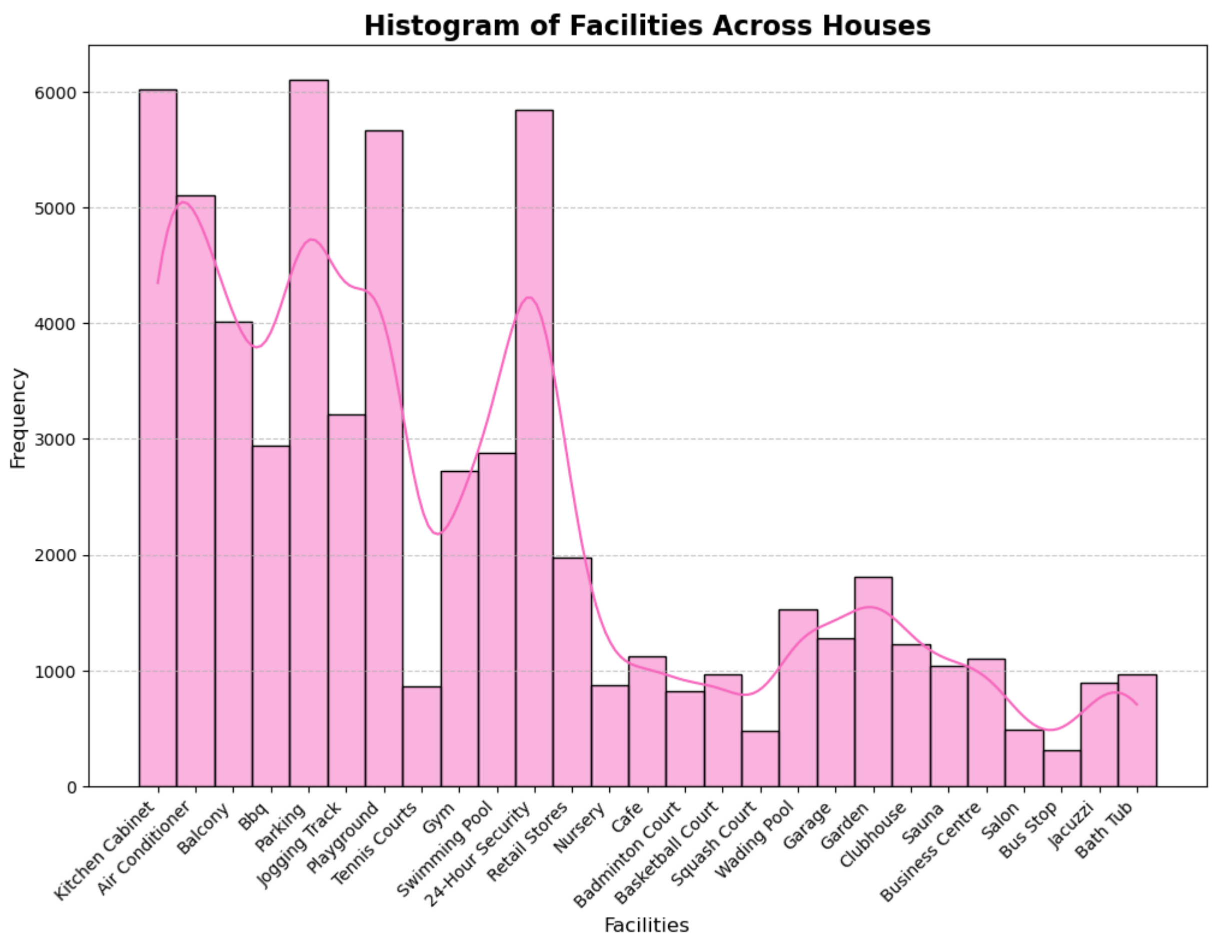

Distribution of Car Slots
Frequency
0
2000
4000
6000
8000
10000
12000
14000
16000
2
4
6
8
10
12
Car Slots


Distribution of House Type
Frequency
0
200
400
600
800
1000
1200
1400
apartment
flat
condominium
serviced residence
1-sty terrace/link house
2-sty terrace/link house
3-sty terrace/link house
4-sty terrace/link house
1.5-sty terrace/link house
2.5-sty terrace/link house
3.5-sty terrace/link house
4.5-sty terrace/link house
townhouse
cluster house
bungalow
semi-detached house
House Type


Distribution of Price per sqft
Frequency
0
250
500
750
1000
1250
1500
1750
2000
0
200
400
600
800
1000
Price per sqft


Distribution of Price per Bedroom
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
0.0
0.5
1.0
1.5
2.0
2.5
1e6
Price per Bedroom

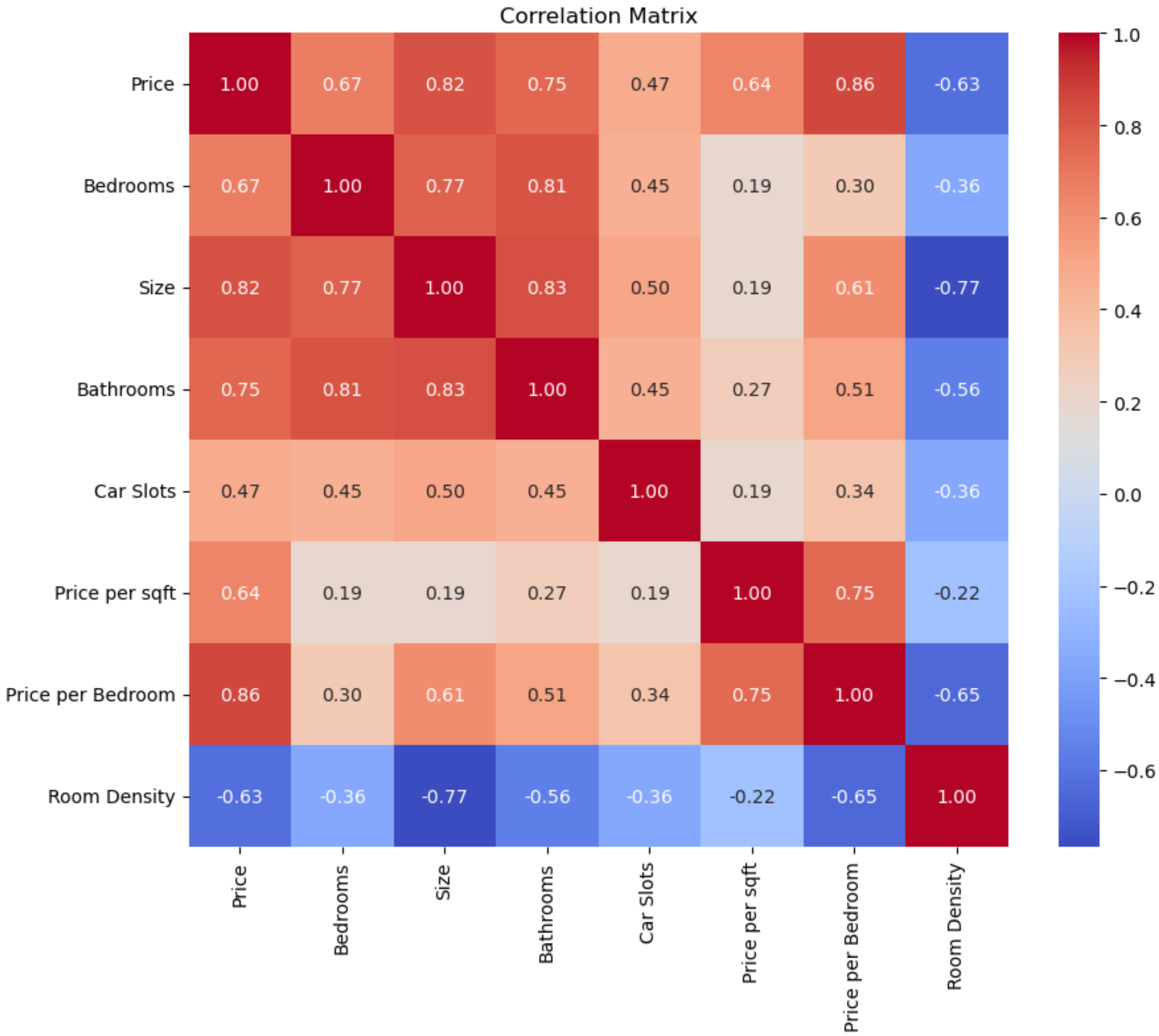




In [19]:
```
# Create correlation analysis
```

```
numerical_df = df.select_dtypes(include=['number']).drop(columns=['Facilities_code'
correlation_matrix = numerical_df.corr()
fig3, ax3 = plt.subplots(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', ax=ax3)
ax3.set_title('Correlation Matrix')
```

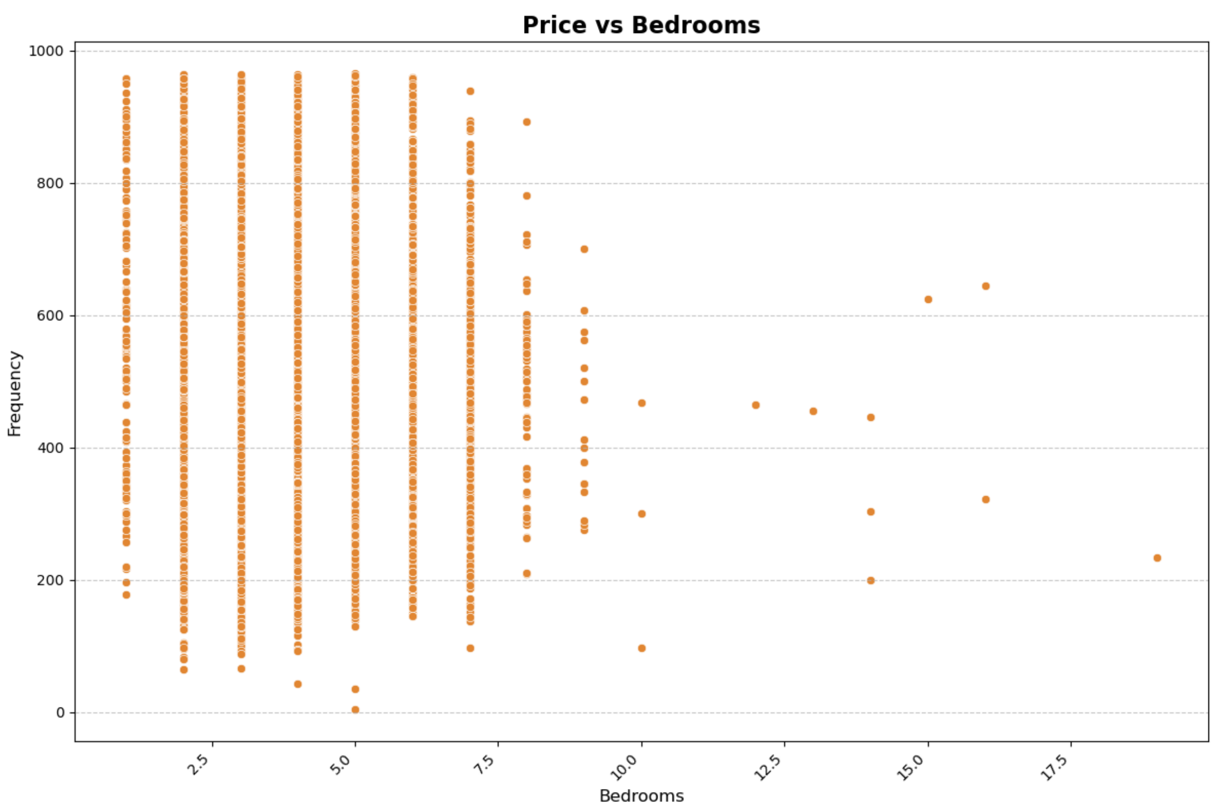
Out[19]: Text(0.5, 1.0, 'Correlation Matrix')

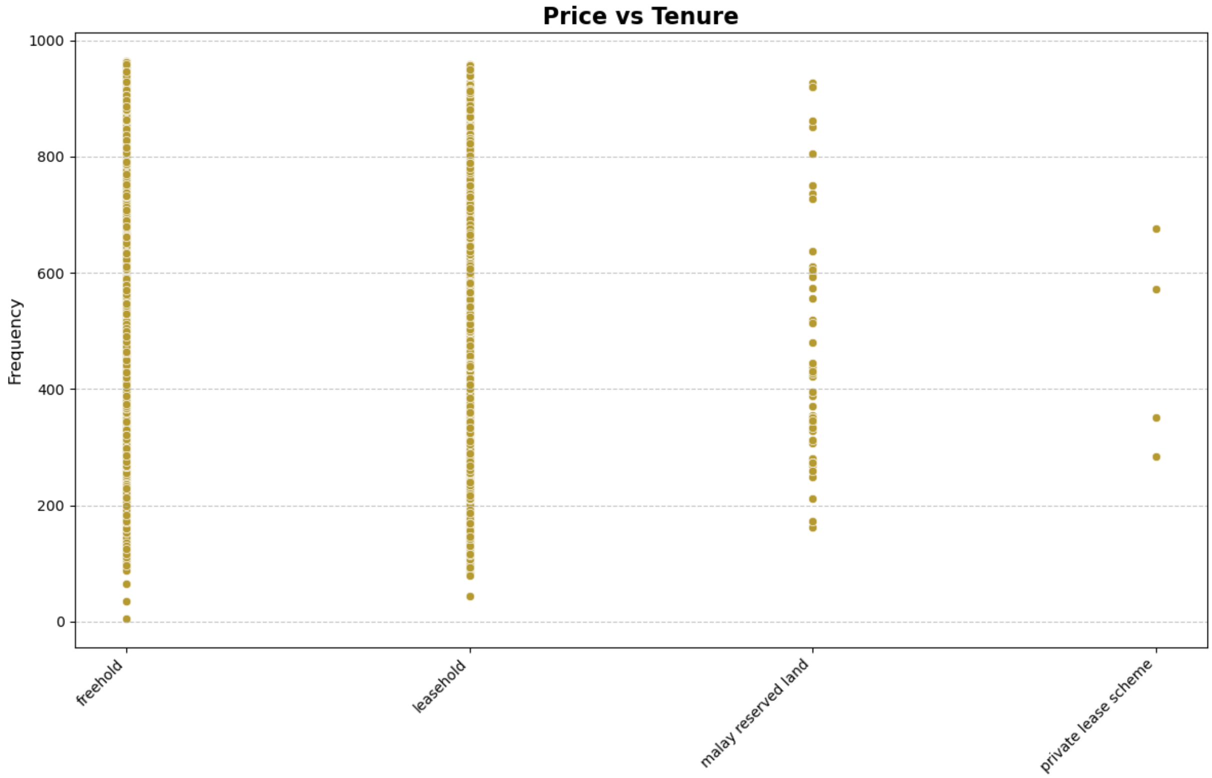


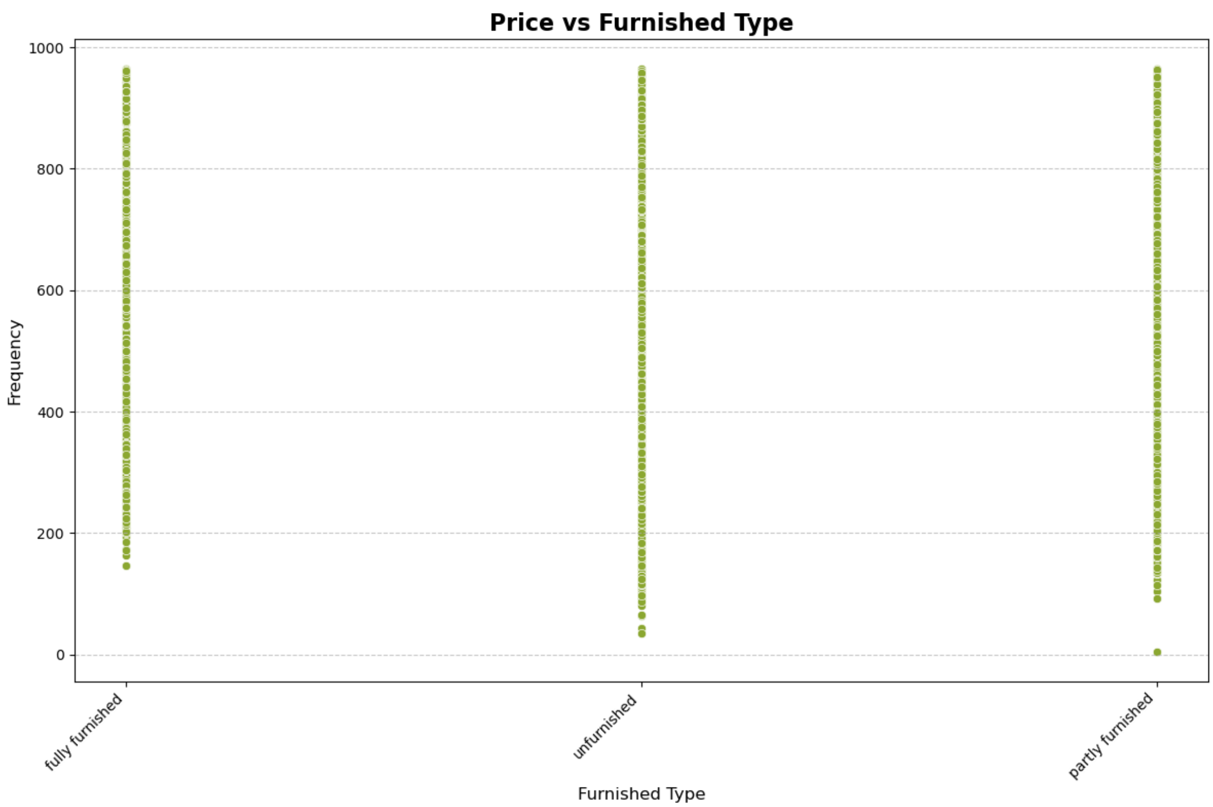
In [20]:
```
# Identify and visualize key trends
columns_to_plot = [
    col for col in df.columns
    if col not in ['Facilities', 'Price', 'State_code', 'Tenure_code', 'Furnished T
]
colors = sns.color_palette("husl", n_colors=len(columns_to_plot))

for idx, column in enumerate(columns_to_plot):
    plt.figure(figsize=(12, 8))
    sns.scatterplot(data=df, x=column, y='Price per sqft', color=colors[idx])
    plt.title(f'Price vs {column}', fontsize=16, fontweight='bold')
    plt.xlabel(column, fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
```

```
plt.show()
```

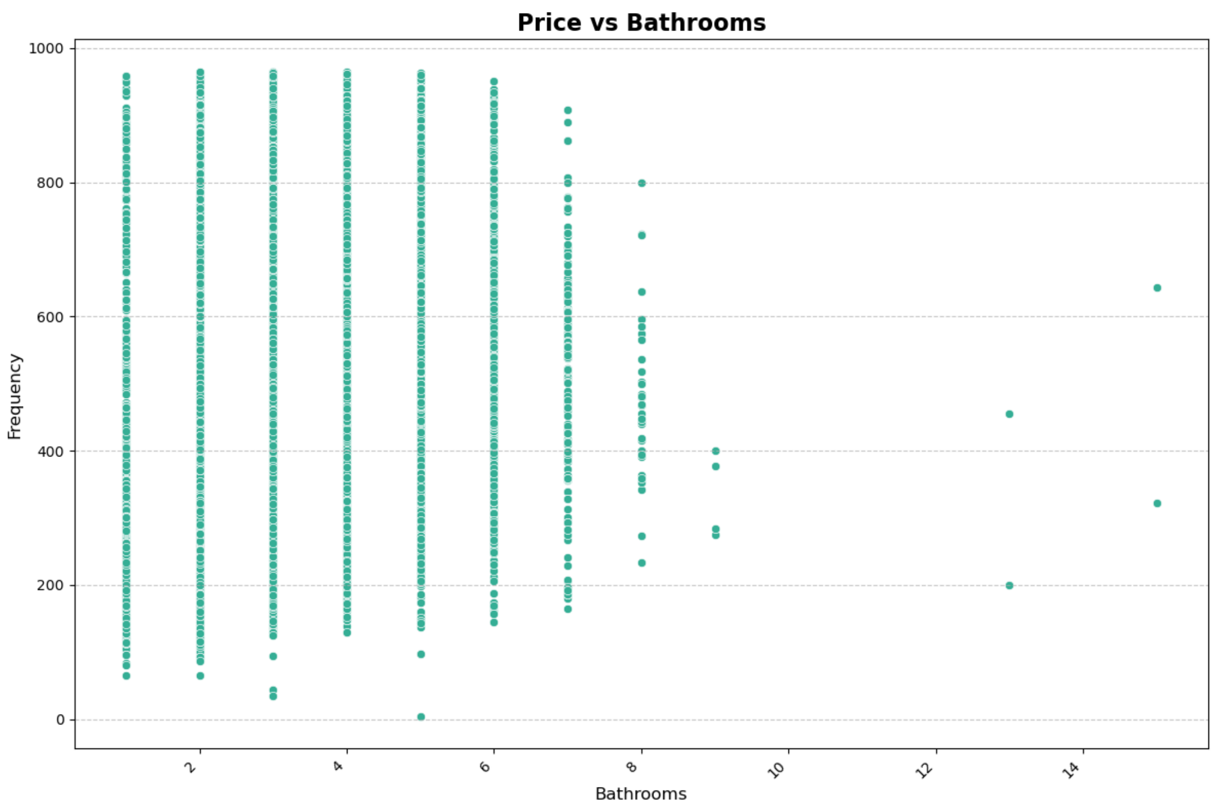

Price vs State
Frequency
State
johor
kuala lumpur
selangor
melaka
penang
negeri sembilan
perak
sarawak
pahang
sabah
putrajaya
terengganu
kedah
kelantan
perlis

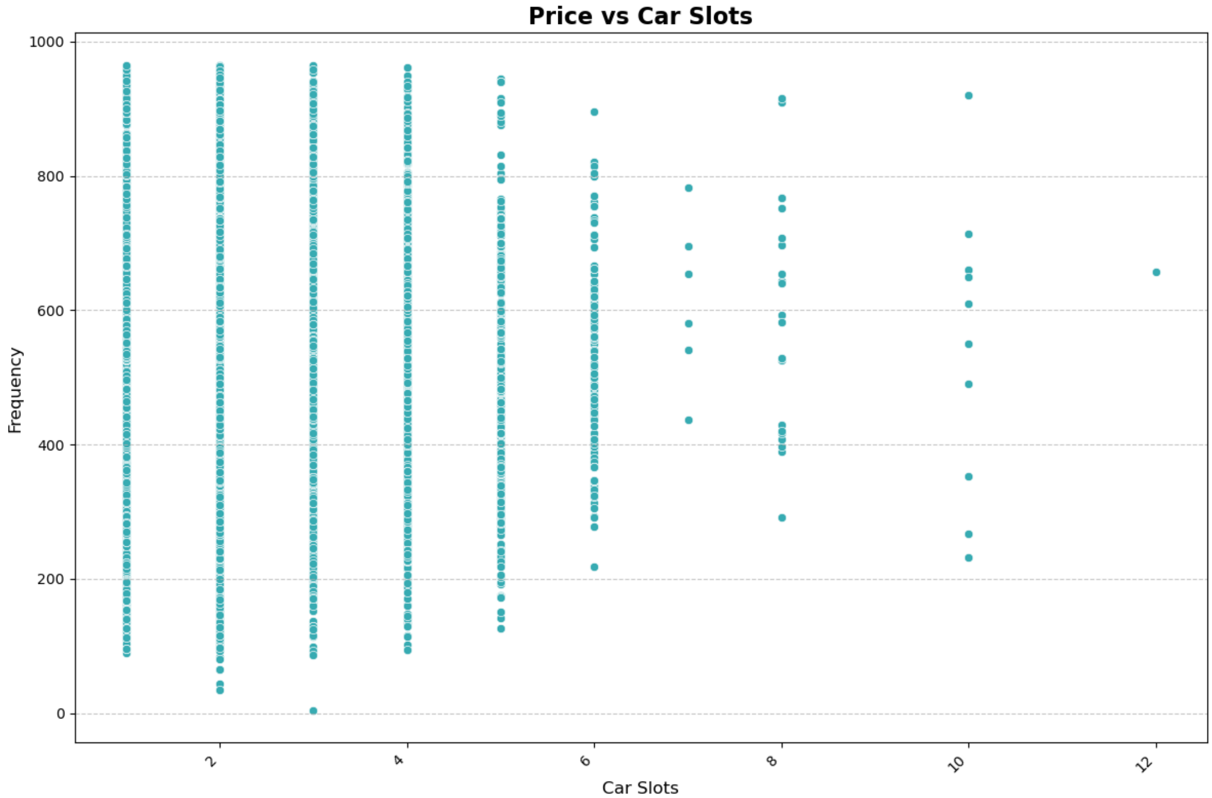

Price vs Bedrooms
Frequency
Bedrooms

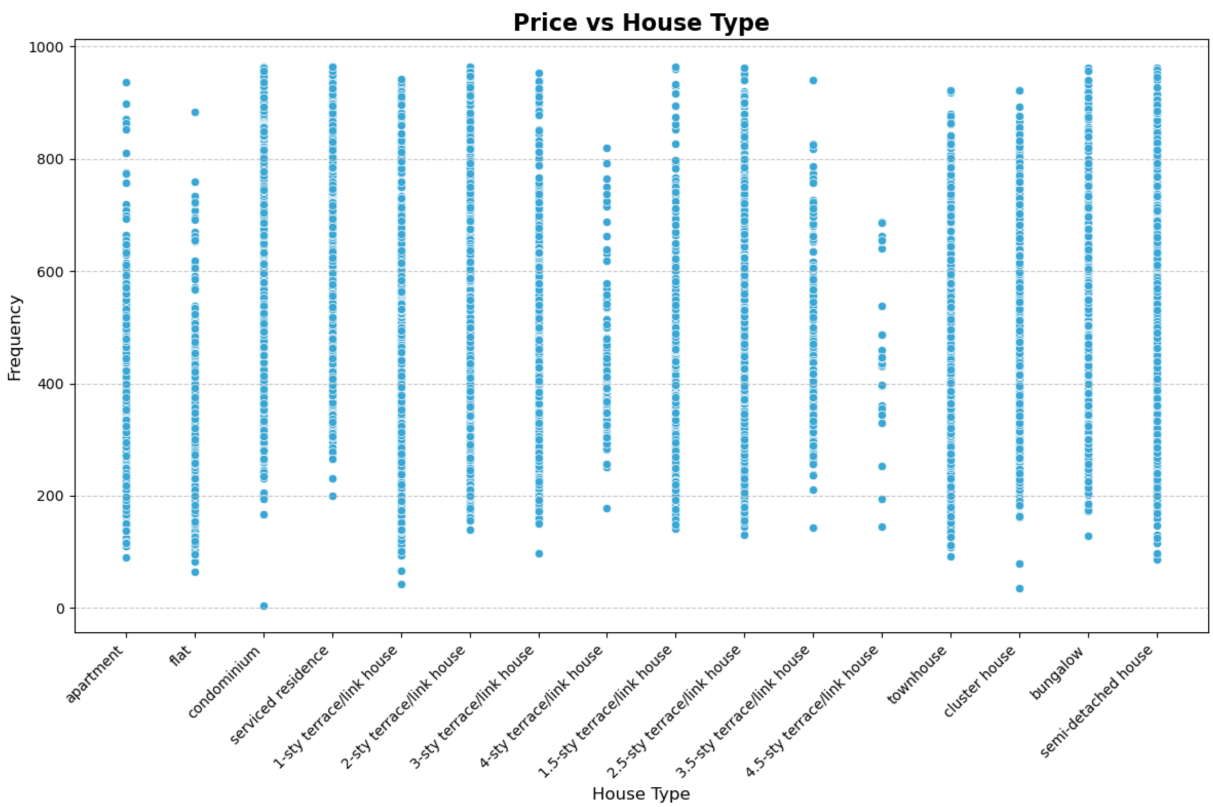

Price vs Tenure
Frequency
1000
800
600
400
200
0
freehold
leasehold
malay reserved land
private lease scheme
Tenure


Price vs Furnished Type
Frequency
1000
800
600
400
200
0
fully furnished
unfurnished
partly furnished
Furnished Type


Price vs Size
Frequency
Size
0
200
400
600
800
1000
0
1000
2000
3000
4000
5000

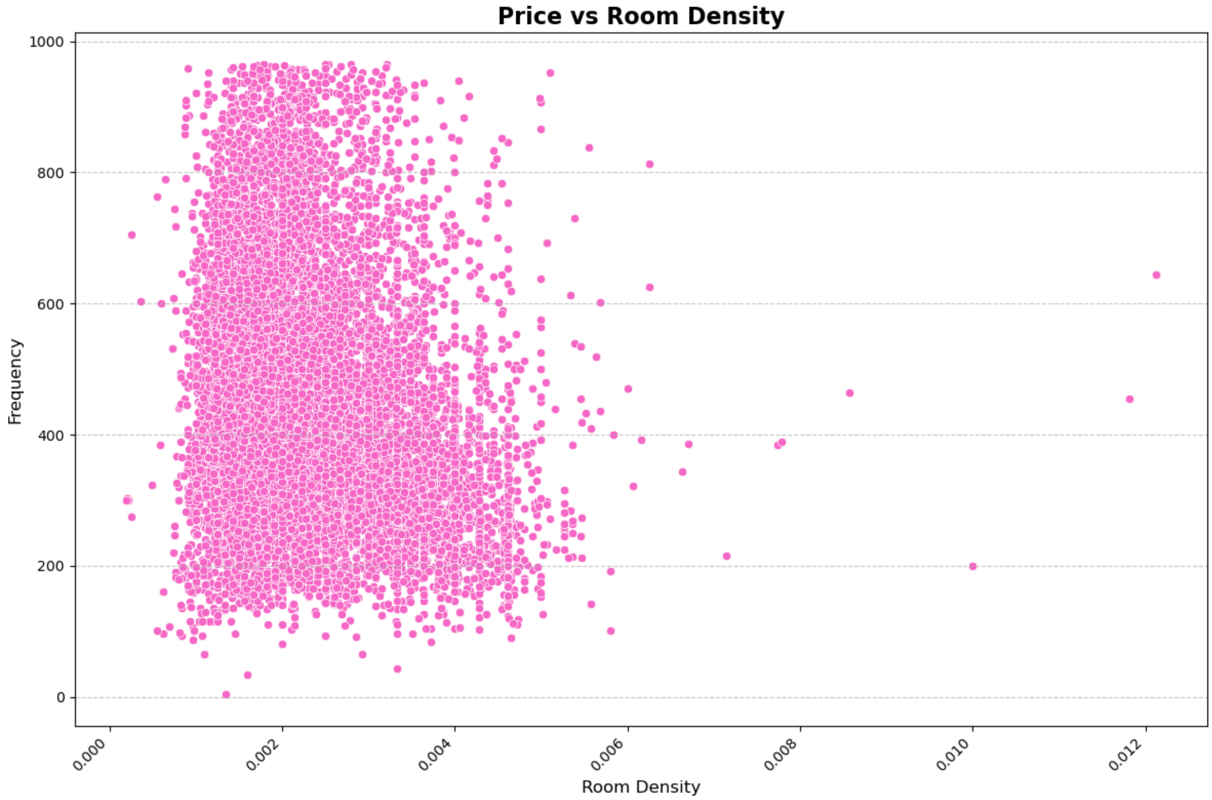

Price vs Bathrooms
Frequency
Bathrooms
0
200
400
600
800
1000
2
4
6
8
10
12
14


Price vs Car Slots
Frequency
0
200
400
600
800
1000
2
4
6
8
10
12
Car Slots


Price vs House Type
Frequency
0
200
400
600
800
1000
apartment
flat
condominium
serviced residence
1-sty terrace/link house
2-sty terrace/link house
3-sty terrace/link house
4-sty terrace/link house
1.5-sty terrace/link house
2.5-sty terrace/link house
3.5-sty terrace/link house
4.5-sty terrace/link house
townhouse
cluster house
bungalow
semi-detached house
House Type

Price vs Price per sqft
Frequency
Price per sqft
0
200
400
600
800
1000

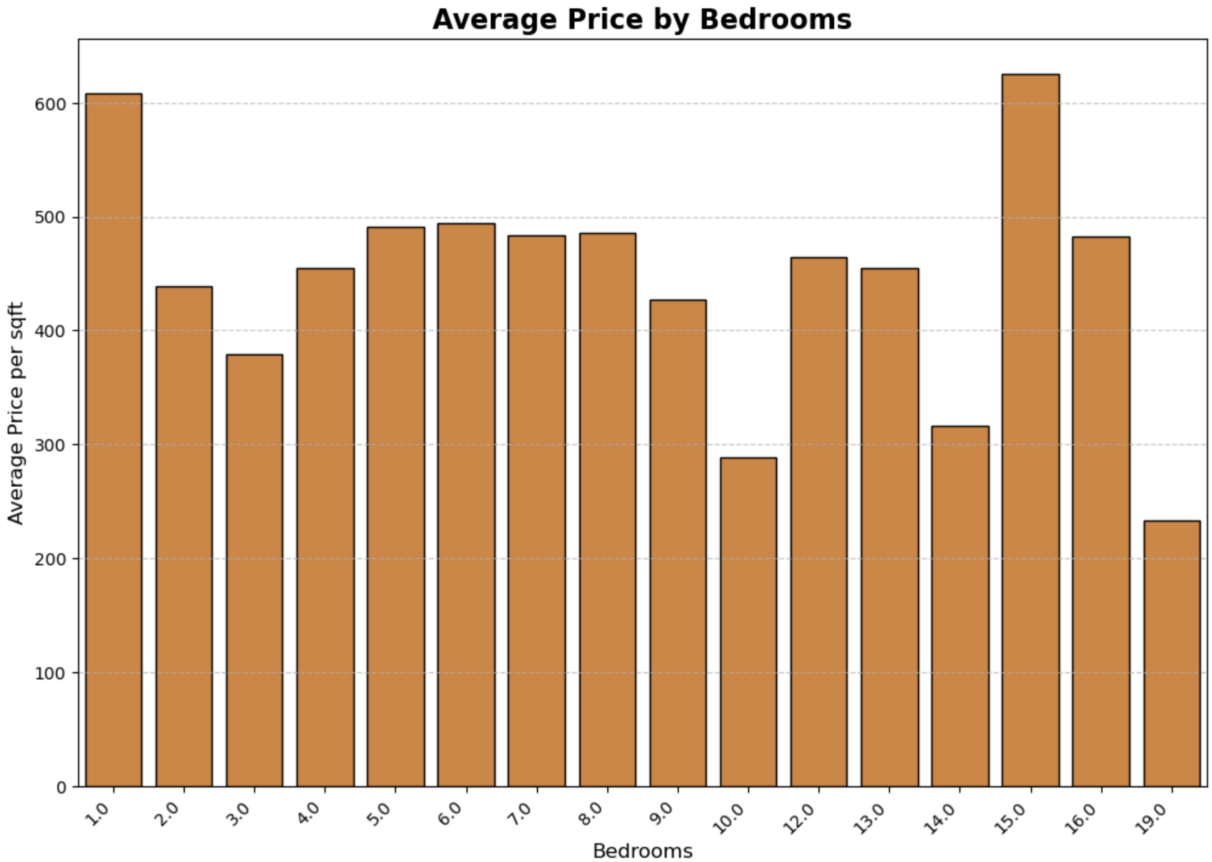
Price vs Price per Bedroom
Frequency
Price per Bedroom
0
200
400
600
800
1000
0.0
0.5
1.0
1.5
2.0
2.5
1e6

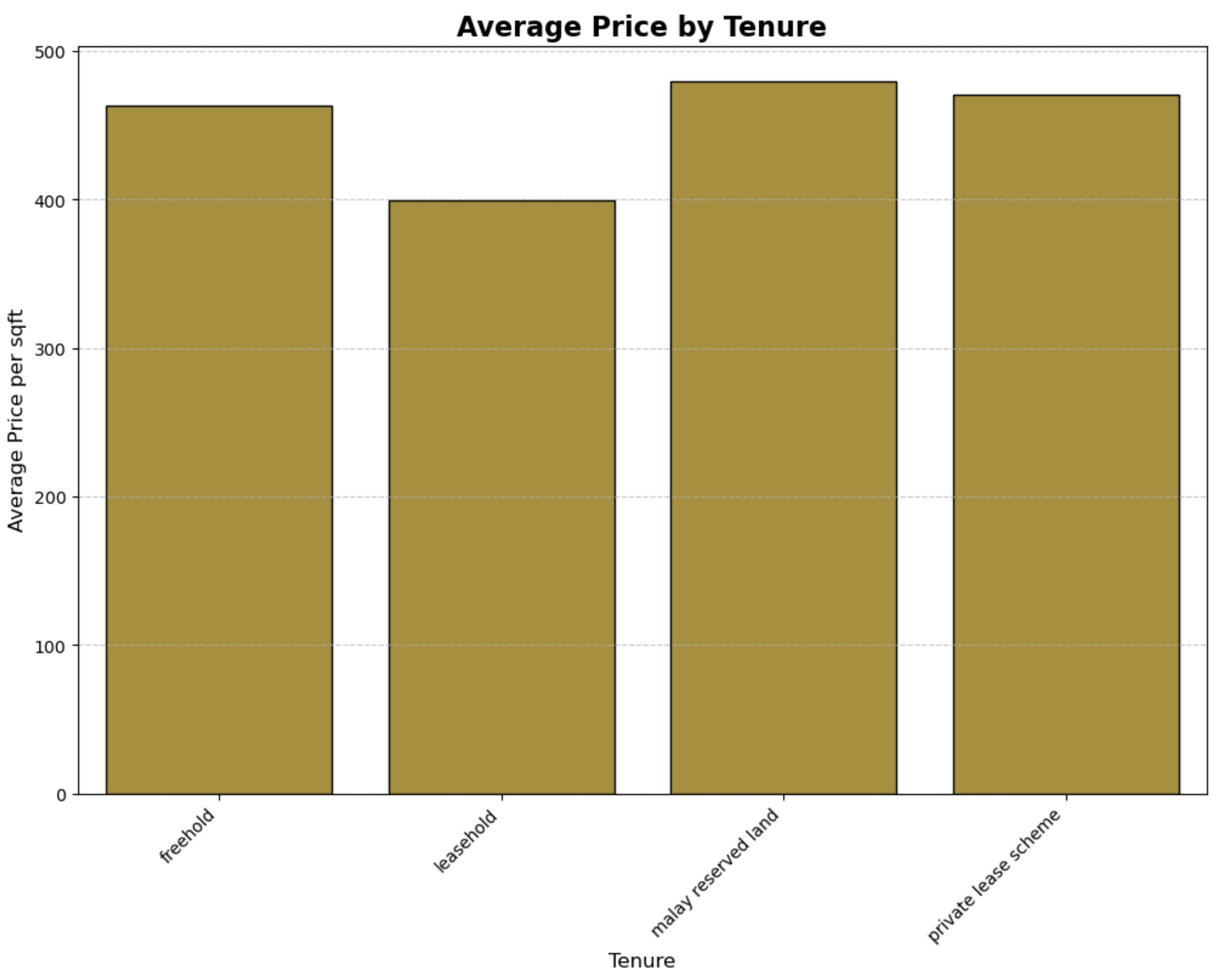


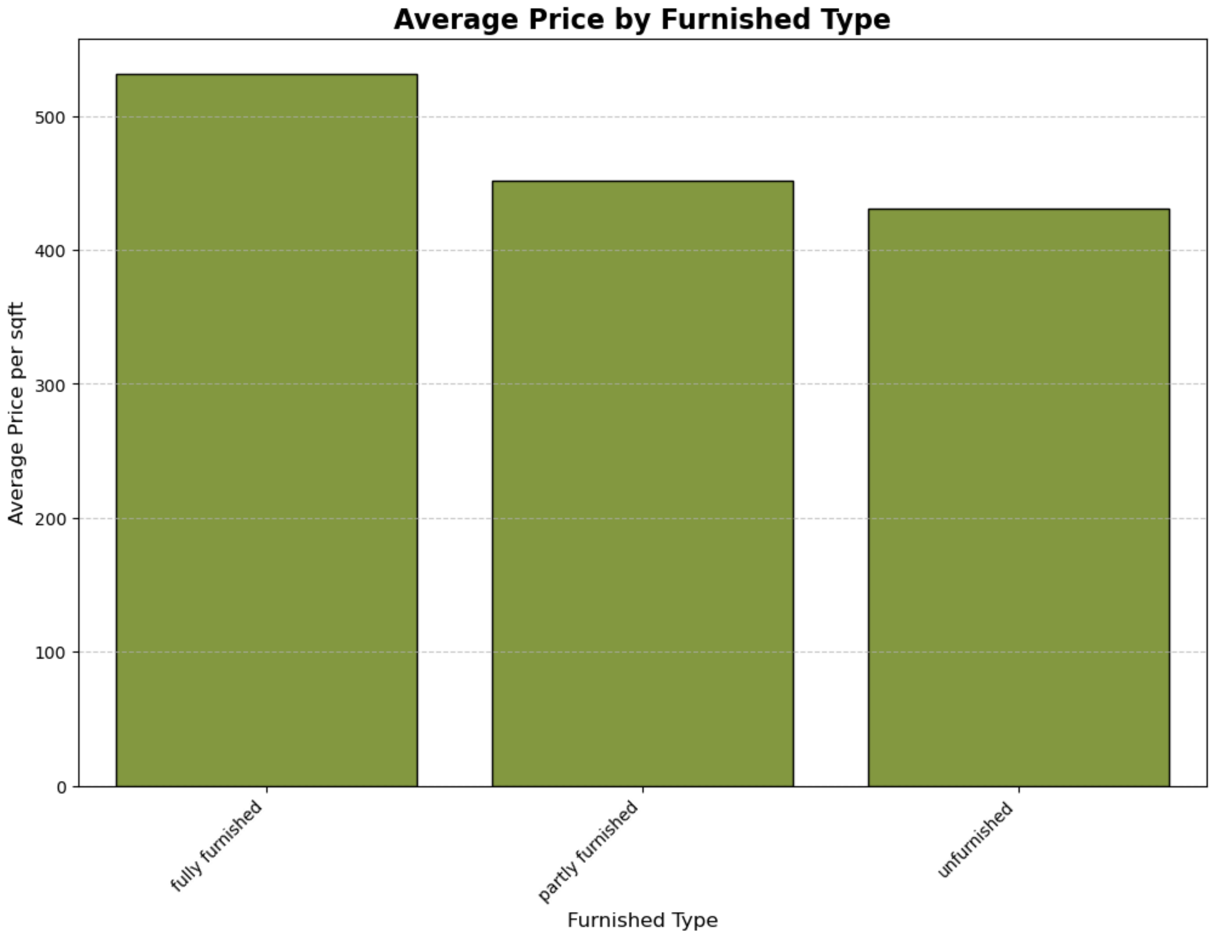
In [21]:
```
size_bins = [0, 500, 1000, 1500, 2000, 2500, df['Size'].max()]
size_labels = ['0-500', '501-1000', '1001-1500', '1501-2000', '2001-2500', '2500+']
df['Size_Bins'] = pd.cut(df['Size'], bins=size_bins, labels=size_labels, include_lo

max_price_per_bedroom = df['Price per Bedroom'].max()
rounded_max_price_per_bedroom = np.ceil(max_price_per_bedroom / 100000) * 100000
bins_bedroom = np.linspace(0, rounded_max_price_per_bedroom, 6)
labels_bedroom = [f"{int(bins_bedroom[i])}-{int(bins_bedroom[i + 1])}" for i in ran

df['Price_per_Bedroom_Bins'] = pd.cut(df['Price per Bedroom'], bins=bins_bedroom, l

binned_columns = ['Size_Bins', 'Price_per_Bedroom_Bins']
binned_labels = ['Size', 'Price per Bedroom']
binned_colors = colors[-3:]
```

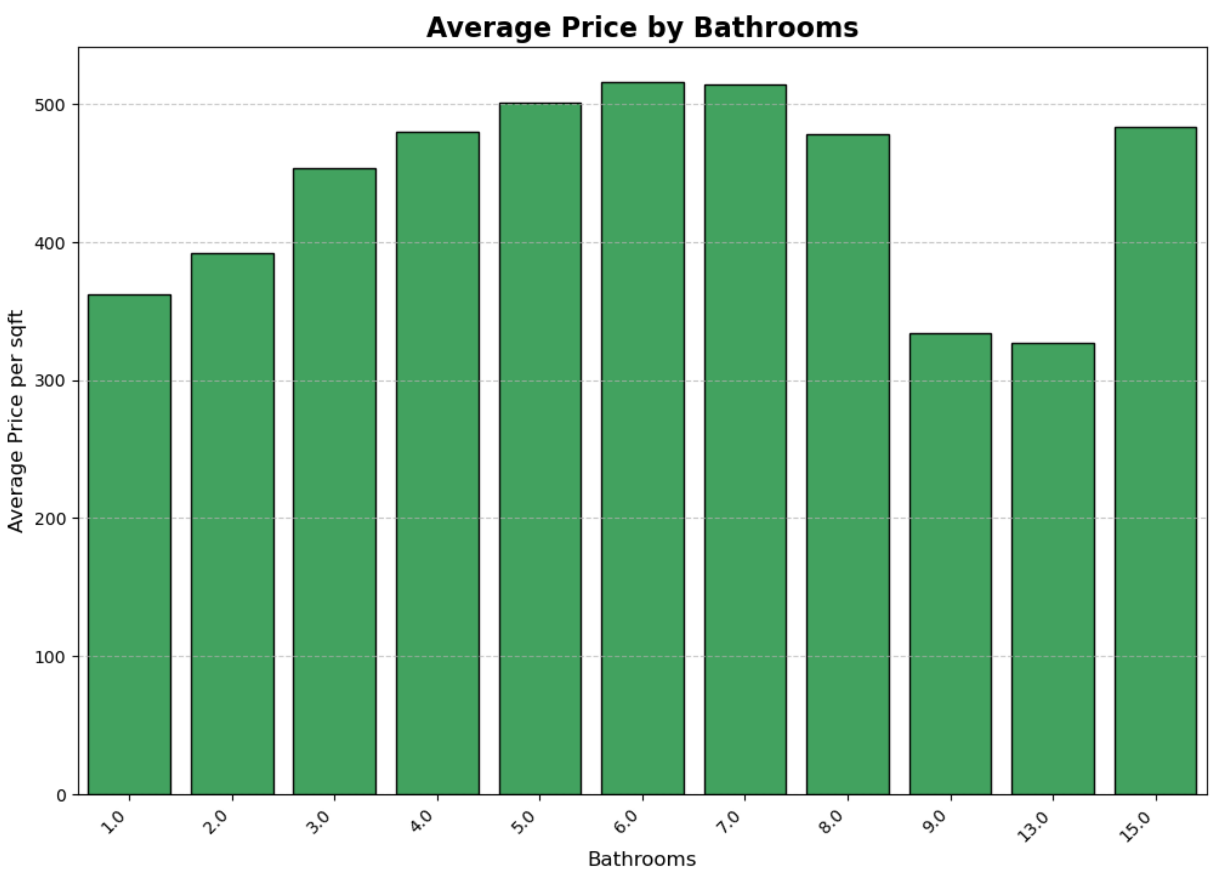
In [22]:
```
# Bar plotting
columns_to_plot = [
    col for col in df.columns
    if col not in ['Facilities', 'Price', 'State_code', 'Tenure_code', 'Furnished T
]
colors = sns.color_palette("husl", n_colors=len(columns_to_plot)+4)

for idx, column in enumerate(columns_to_plot):
    avg_price = df.groupby(column)['Price per sqft'].mean().reset_index()
    plt.figure(figsize=(12, 8))
    sns.barplot(data=avg_price, x=column, y='Price per sqft', color=colors[idx], ed
    plt.title(f'Average Price by {column}', fontsize=16, fontweight='bold')
    plt.xlabel(column, fontsize=12)
    plt.ylabel('Average Price per sqft', fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
```

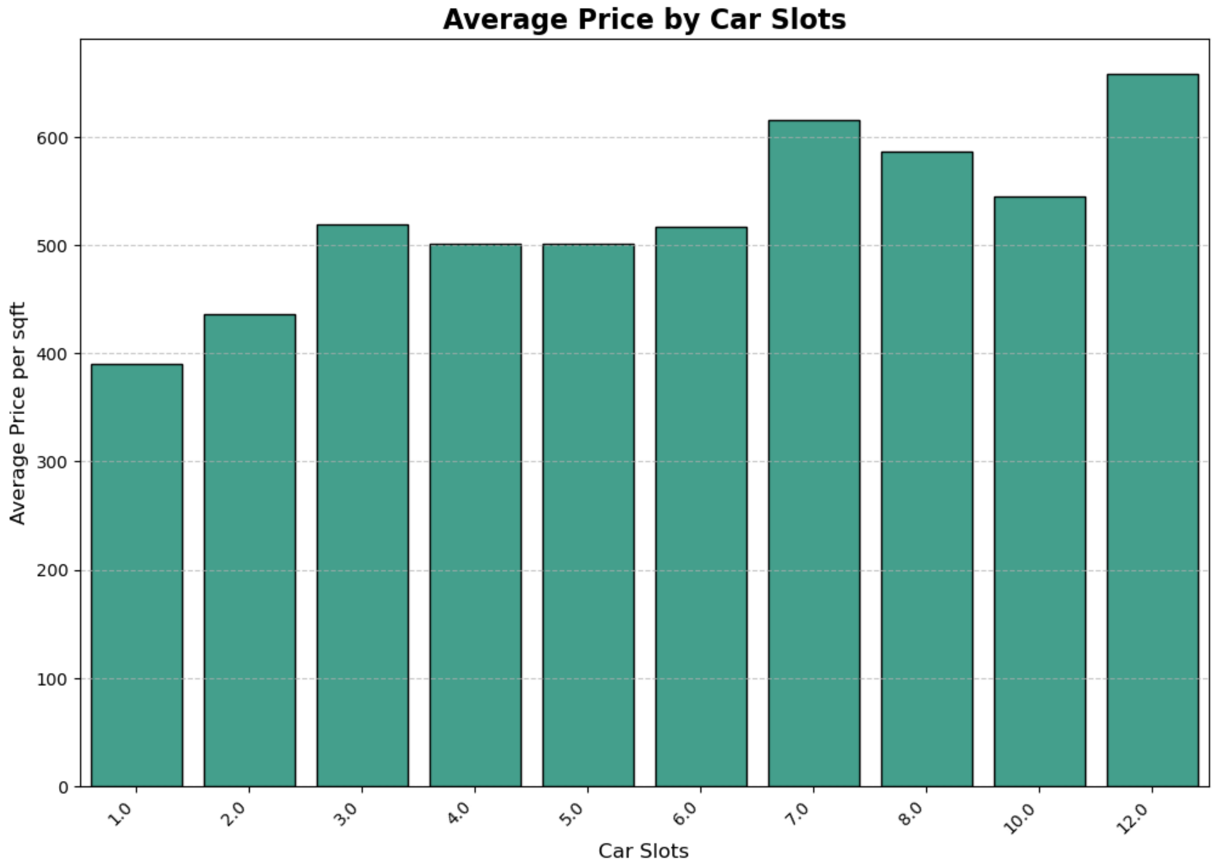
```
    plt.show()

for idx, binned_column in enumerate(binned_columns):
    avg_price = df.groupby(binned_column, observed=True)['Price per sqft'].mean().r
    plt.figure(figsize=(12, 8))
    sns.barplot(data=avg_price, x=binned_column, y='Price per sqft', color=binned_c
    plt.title(f'Average Price by {binned_labels[idx]}', fontsize=16, fontweight='bo
    plt.xlabel(binned_labels[idx], fontsize=12)
    plt.ylabel('Average Price per sqft', fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```


Average Price by Bedrooms
Average Price per sqft
0
100
200
300
400
500
600
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
12.0
13.0
14.0
15.0
16.0
19.0
Bedrooms

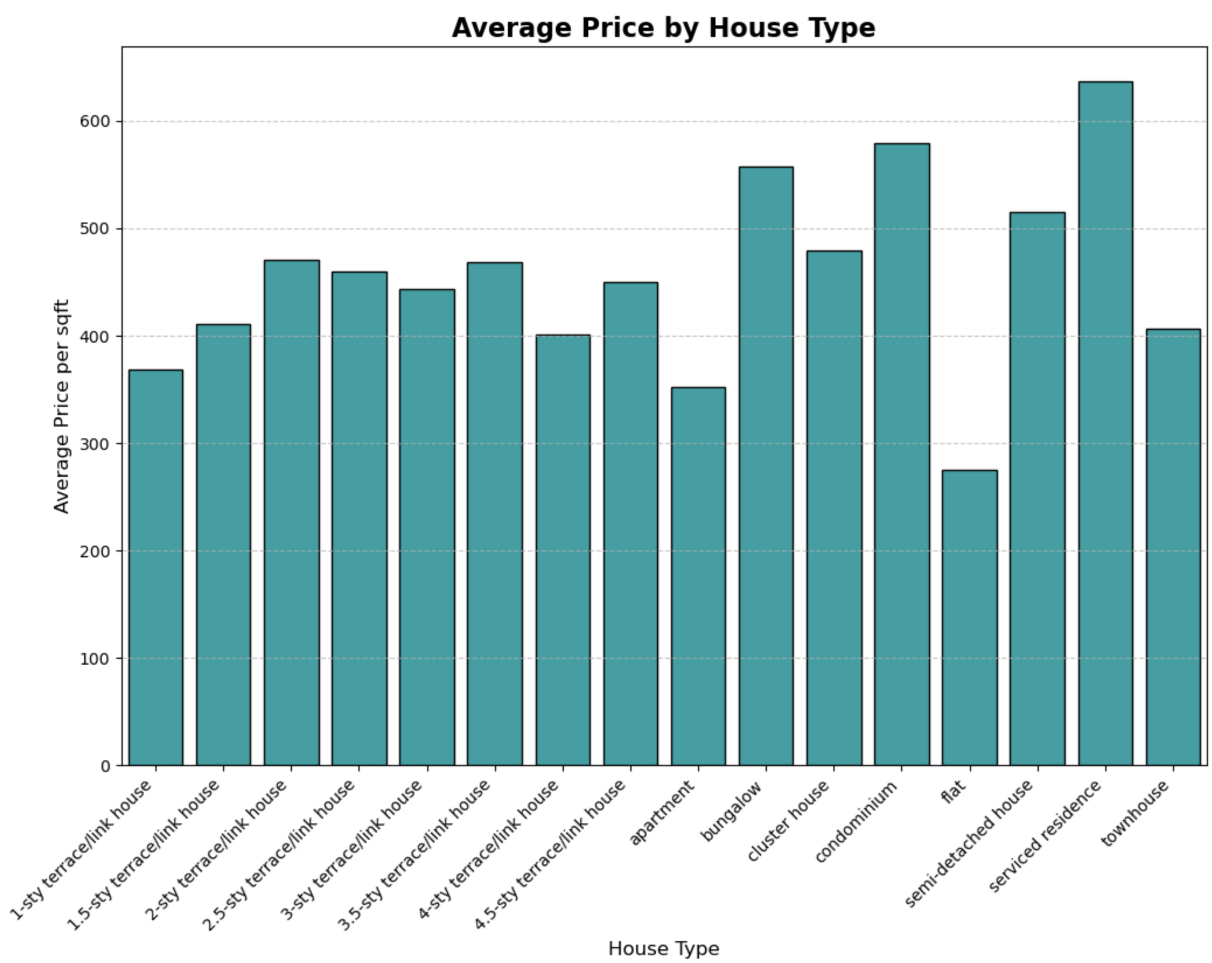

Average Price by Tenure
Average Price per sqft
0
100
200
300
400
500
freehold
leasehold
malay reserved land
private lease scheme
Tenure


Average Price by Furnished Type
Average Price per sqft
0
100
200
300
400
500
fully furnished
partly furnished
unfurnished
Furnished Type

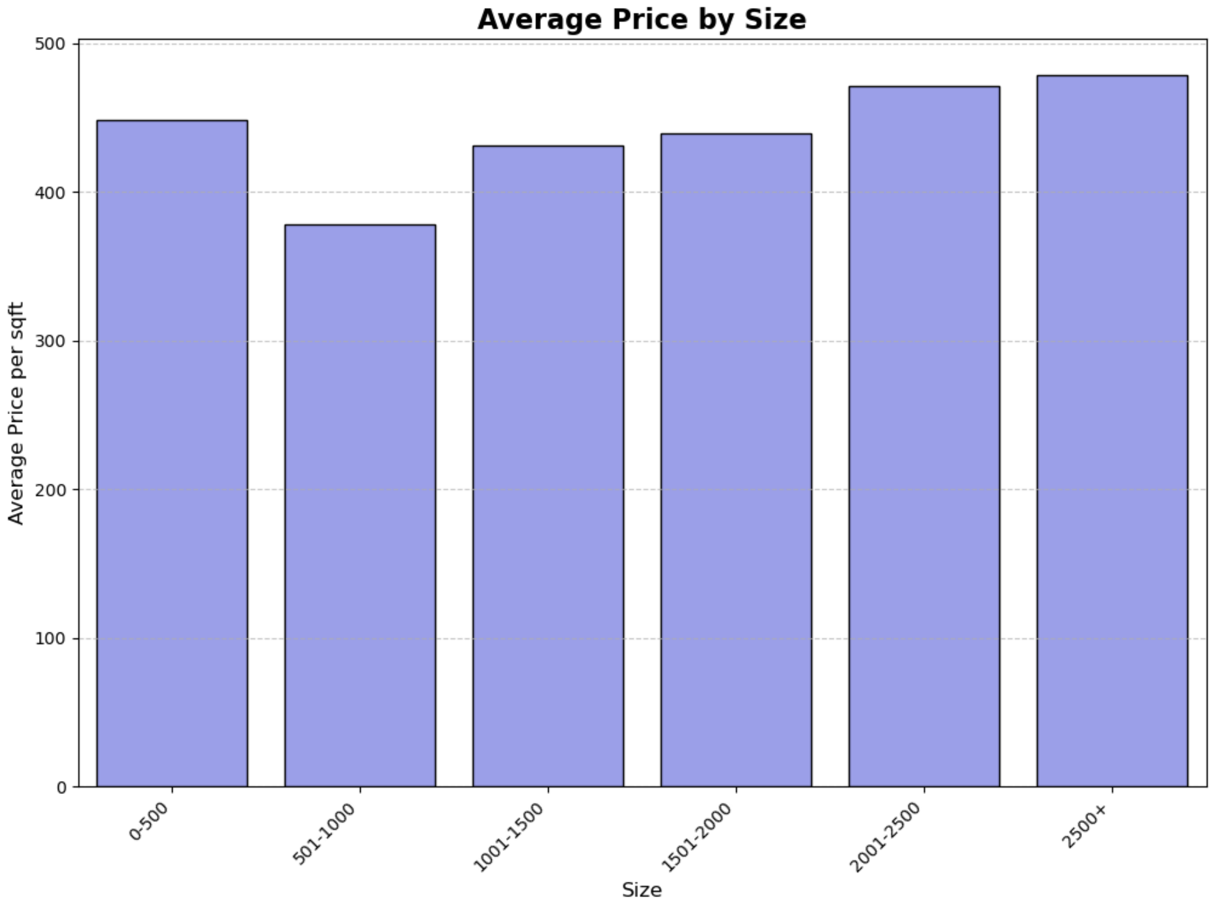

Average Price by Bathrooms
Average Price per sqft
0
100
200
300
400
500
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
13.0
15.0
Bathrooms

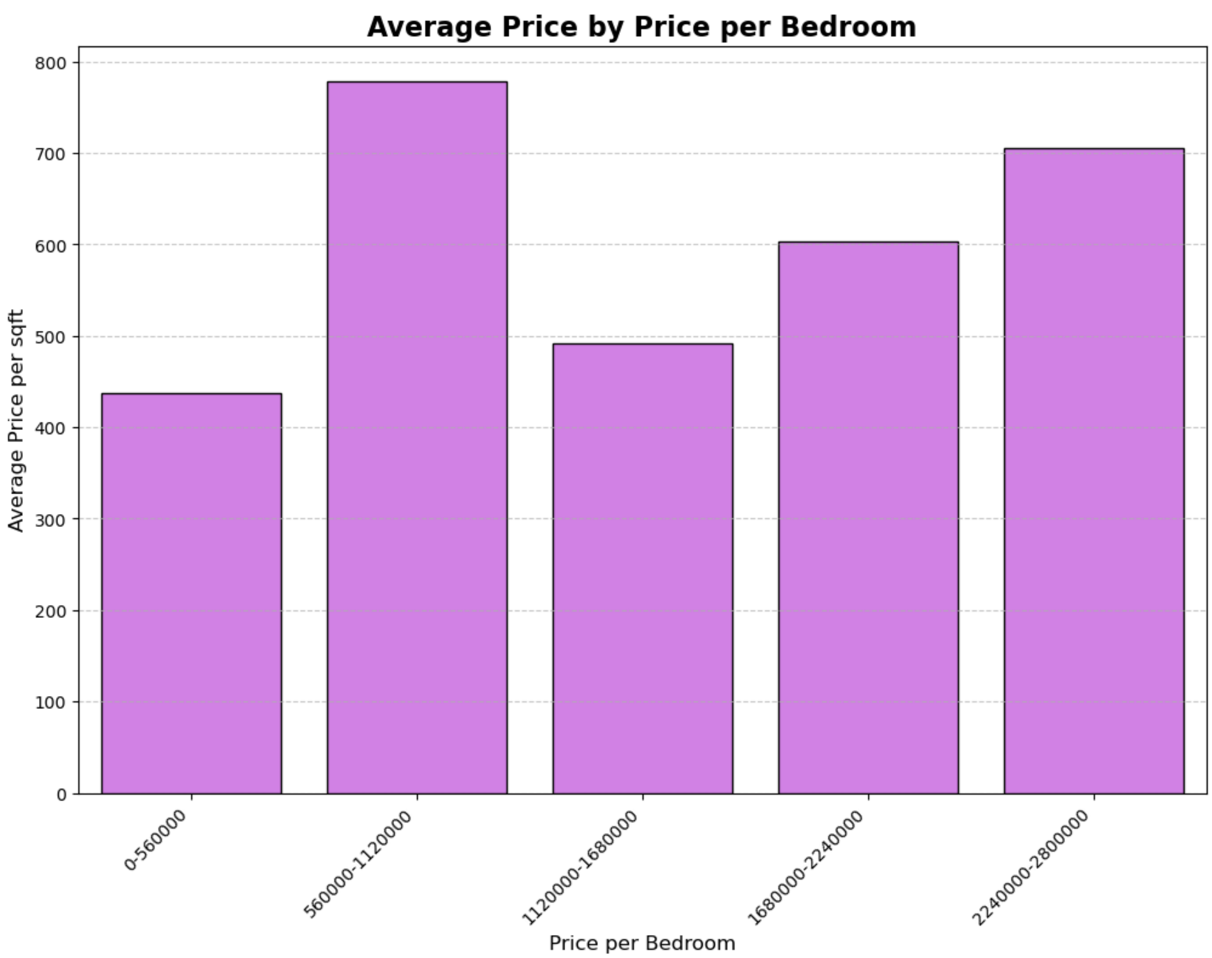


Average Price by Car Slots
Average Price per sqft
0
100
200
300
400
500
600
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
10.0
12.0
Car Slots

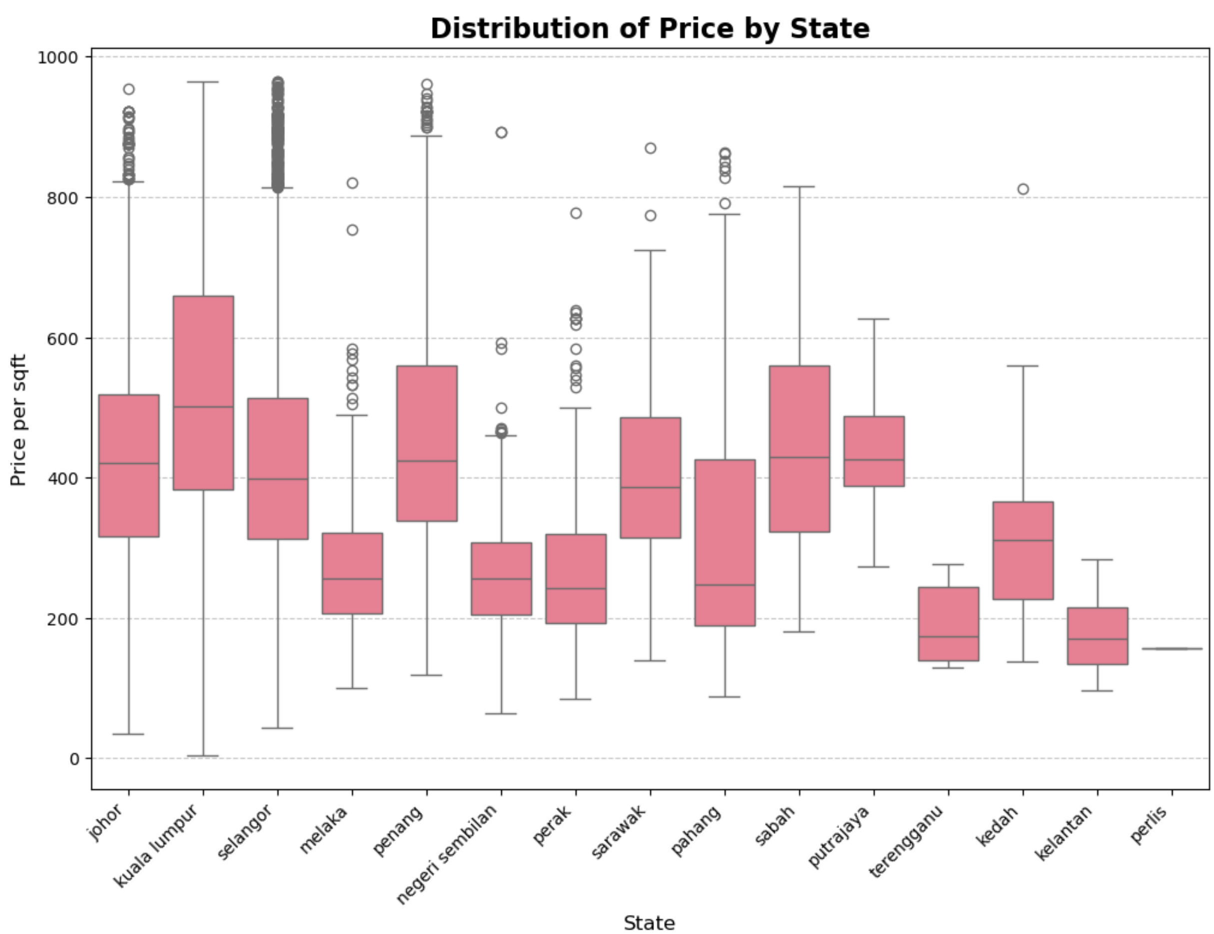


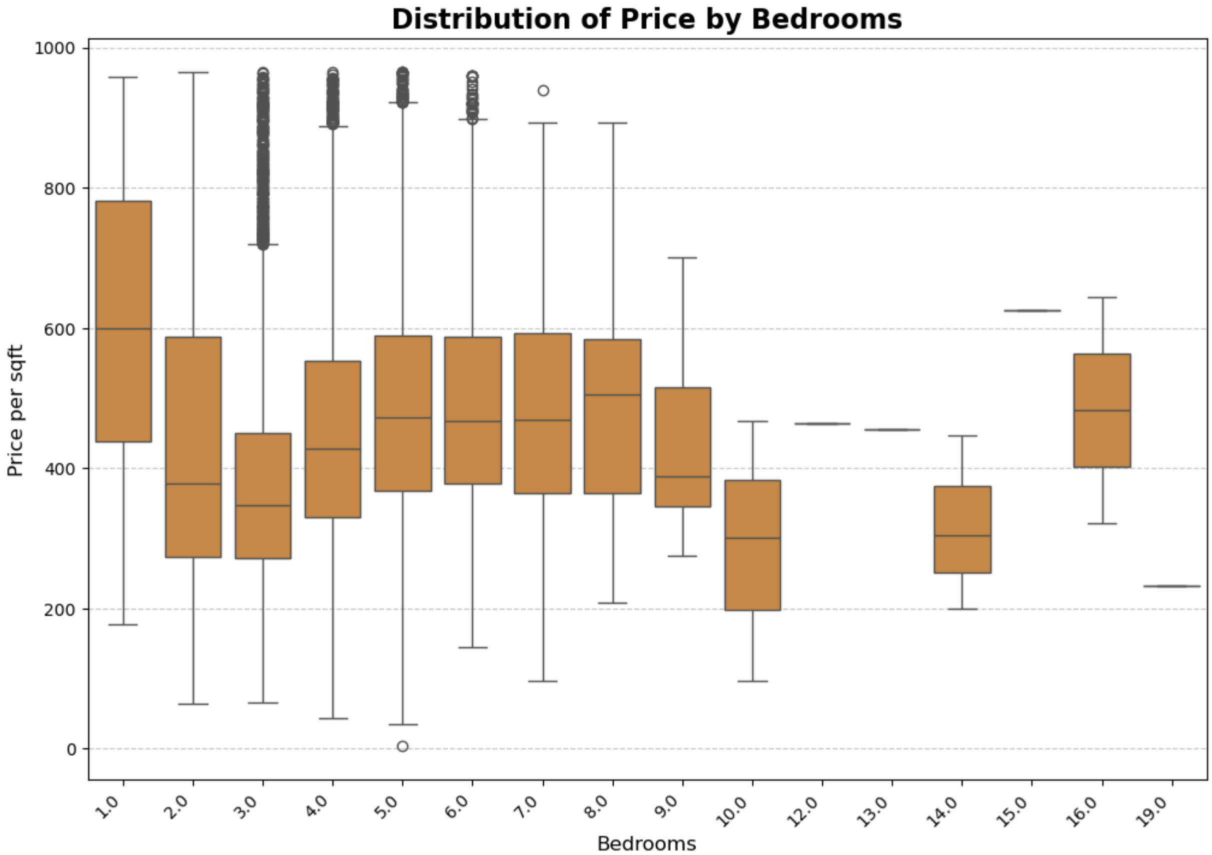
Average Price by House Type
Average Price per sqft
0
100
200
300
400
500
600
1-sty terrace/link house
1.5-sty terrace/link house
2-sty terrace/link house
2.5-sty terrace/link house
3-sty terrace/link house
3.5-sty terrace/link house
4-sty terrace/link house
4.5-sty terrace/link house
apartment
bungalow
cluster house
condominium
flat
semi-detached house
serviced residence
townhouse
House Type

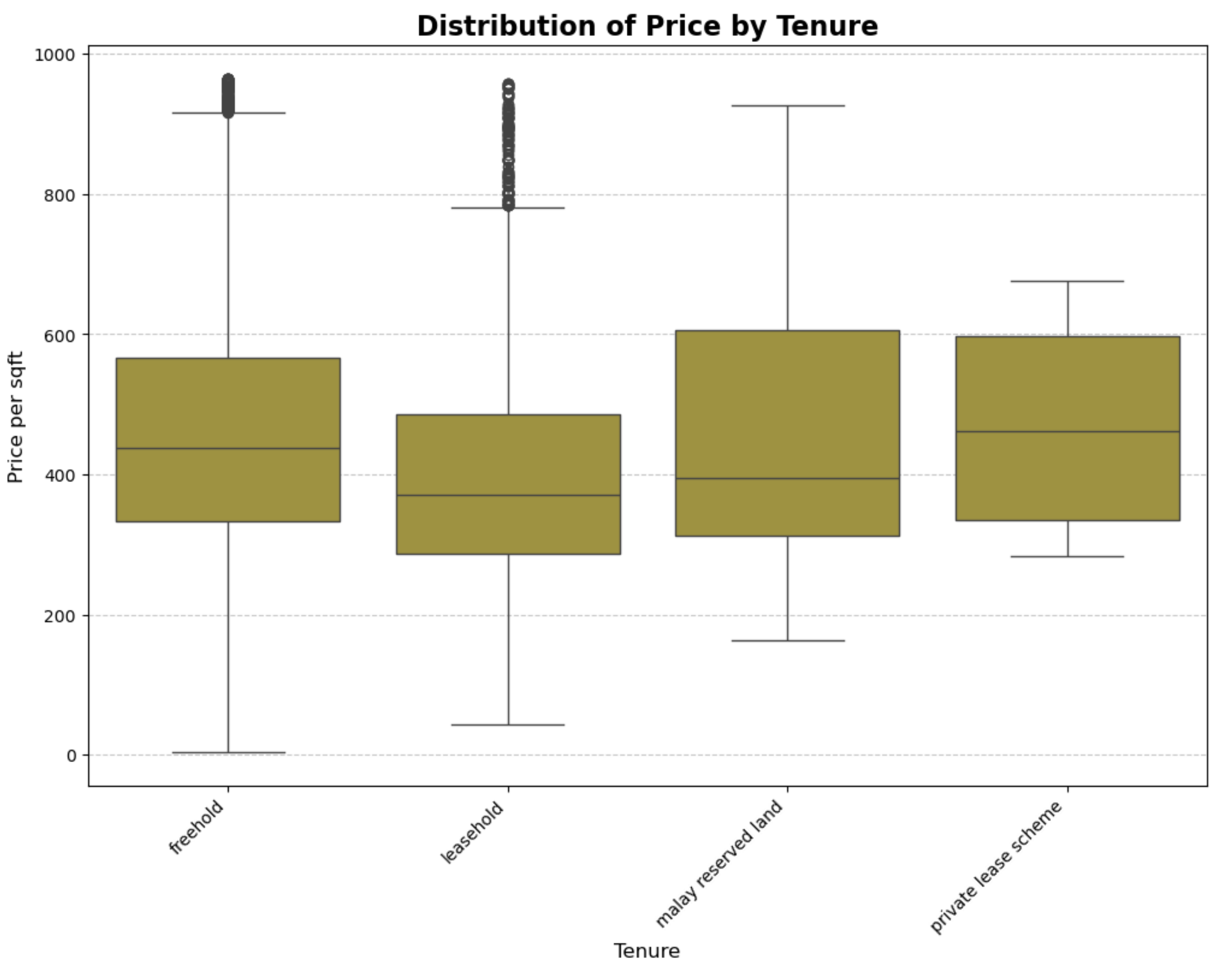

Average Price by Size
Average Price per sqft
0
100
200
300
400
500
0-500
501-1000
1001-1500
1501-2000
2001-2500
2500+
Size

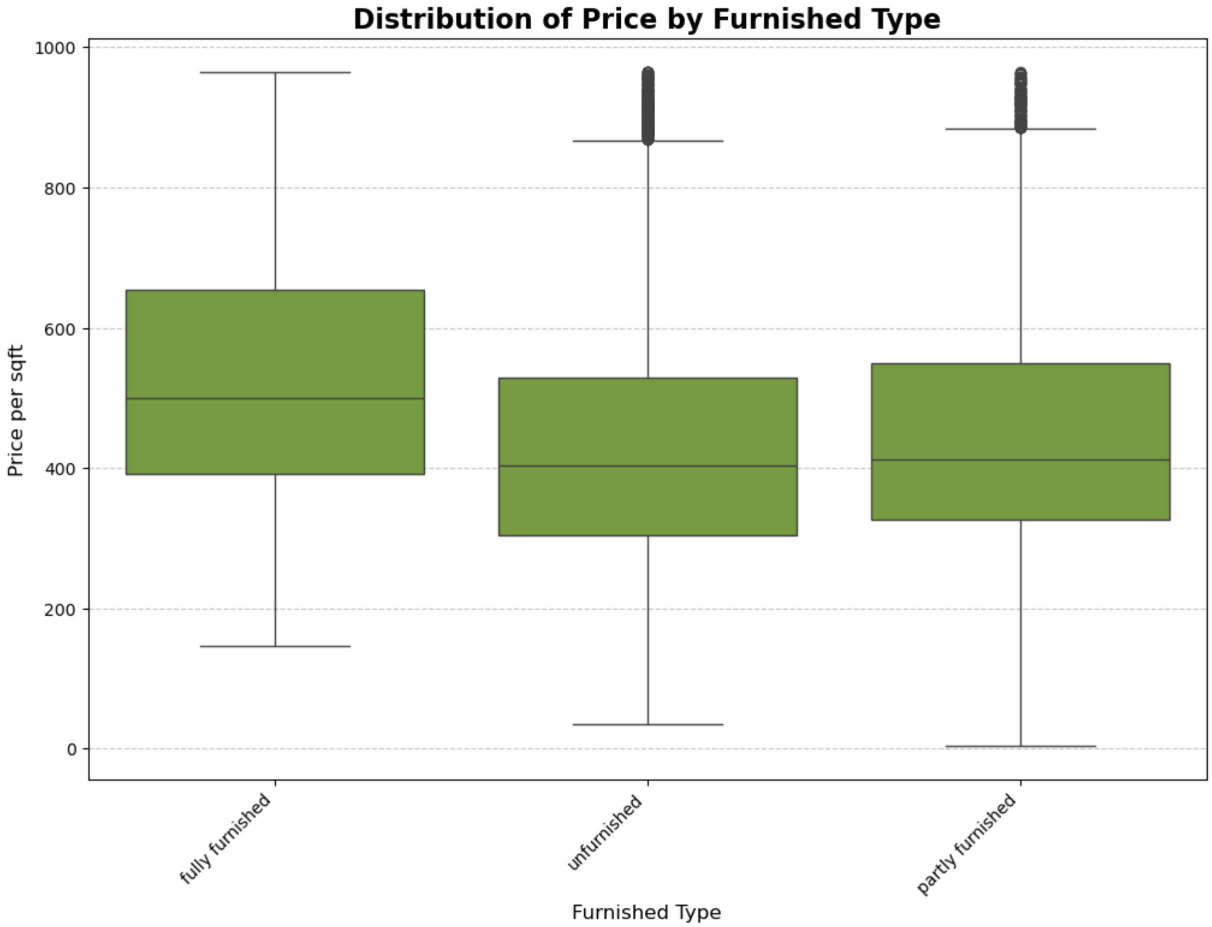

Average Price by Price per Bedroom
Average Price per sqft
0
100
200
300
400
500
600
700
800
0-560000
560000-1120000
1120000-1680000
1680000-2240000
2240000-2800000
Price per Bedroom

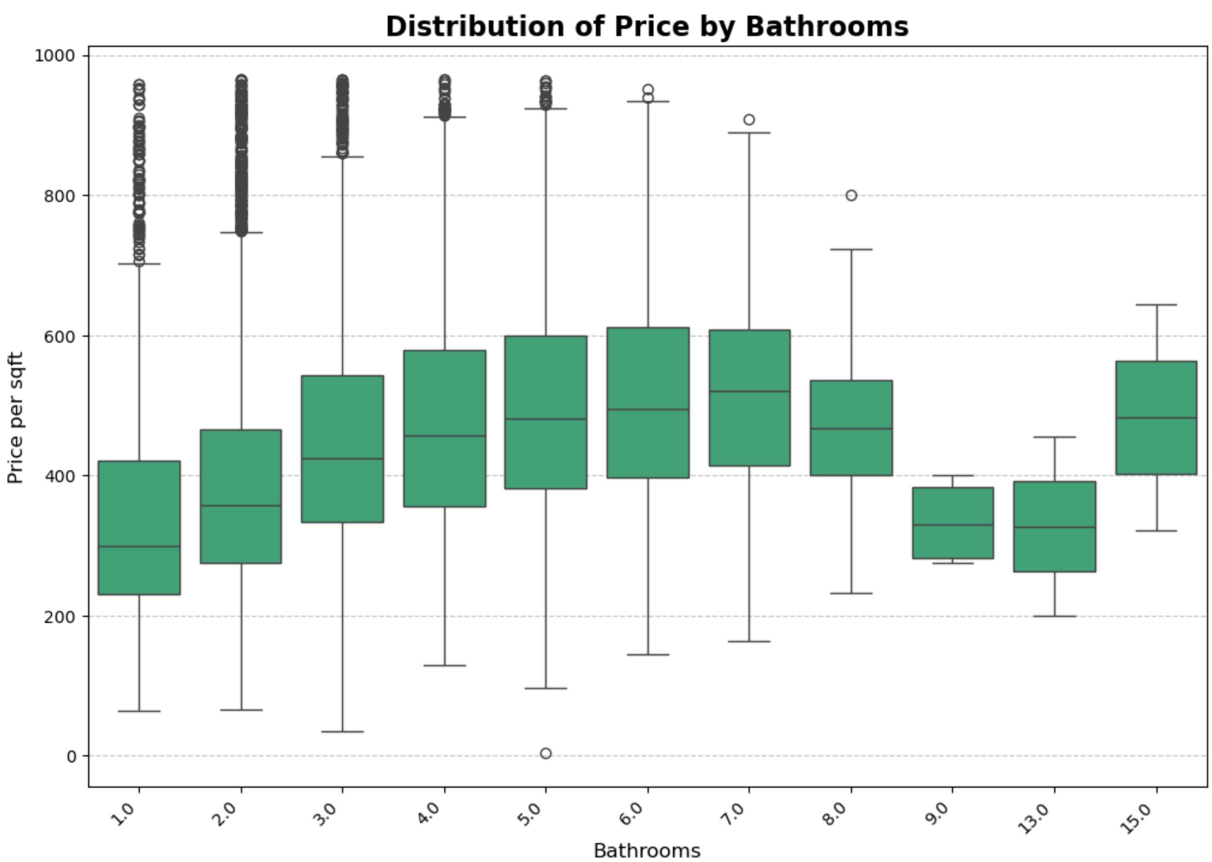
In [23]:
```
# Box plotting
colors = sns.color_palette("husl", n_colors=len(columns_to_plot)+3)

for idx, column in enumerate(columns_to_plot):
    plt.figure(figsize=(12, 8))
    sns.boxplot(data=df, x=column, y='Price per sqft', color=colors[idx])
    plt.title(f'Distribution of Price by {column}', fontsize=16, fontweight='bold')
    plt.xlabel(column, fontsize=12)
    plt.ylabel('Price per sqft', fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()

for idx, binned_column in enumerate(binned_columns):
    plt.figure(figsize=(12, 8))
    sns.boxplot(data=df, x=binned_column, y='Price per sqft', color=binned_colors[i
    plt.title(f'Distribution of Price by {binned_labels[idx]}', fontsize=16, fontwe
    plt.xlabel(binned_labels[idx], fontsize=12)
    plt.ylabel('Price per sqft', fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```

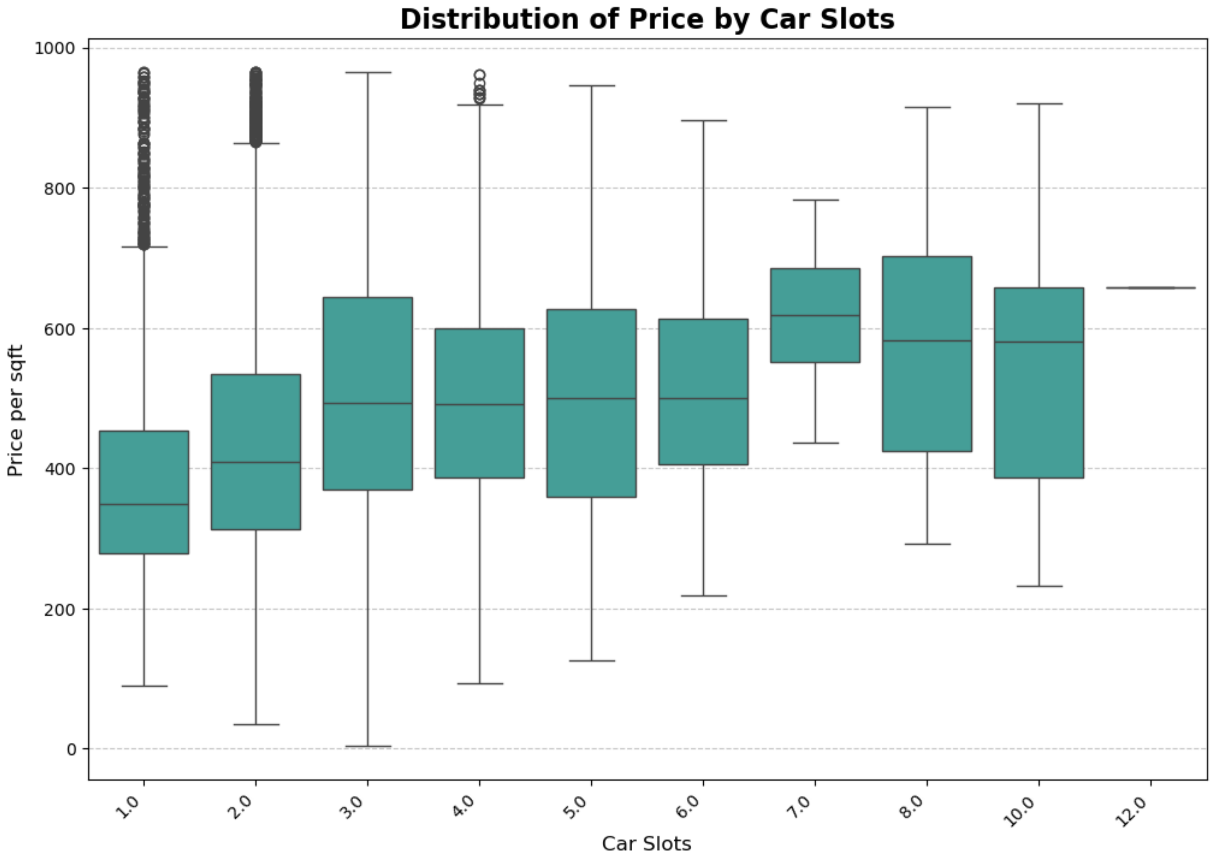


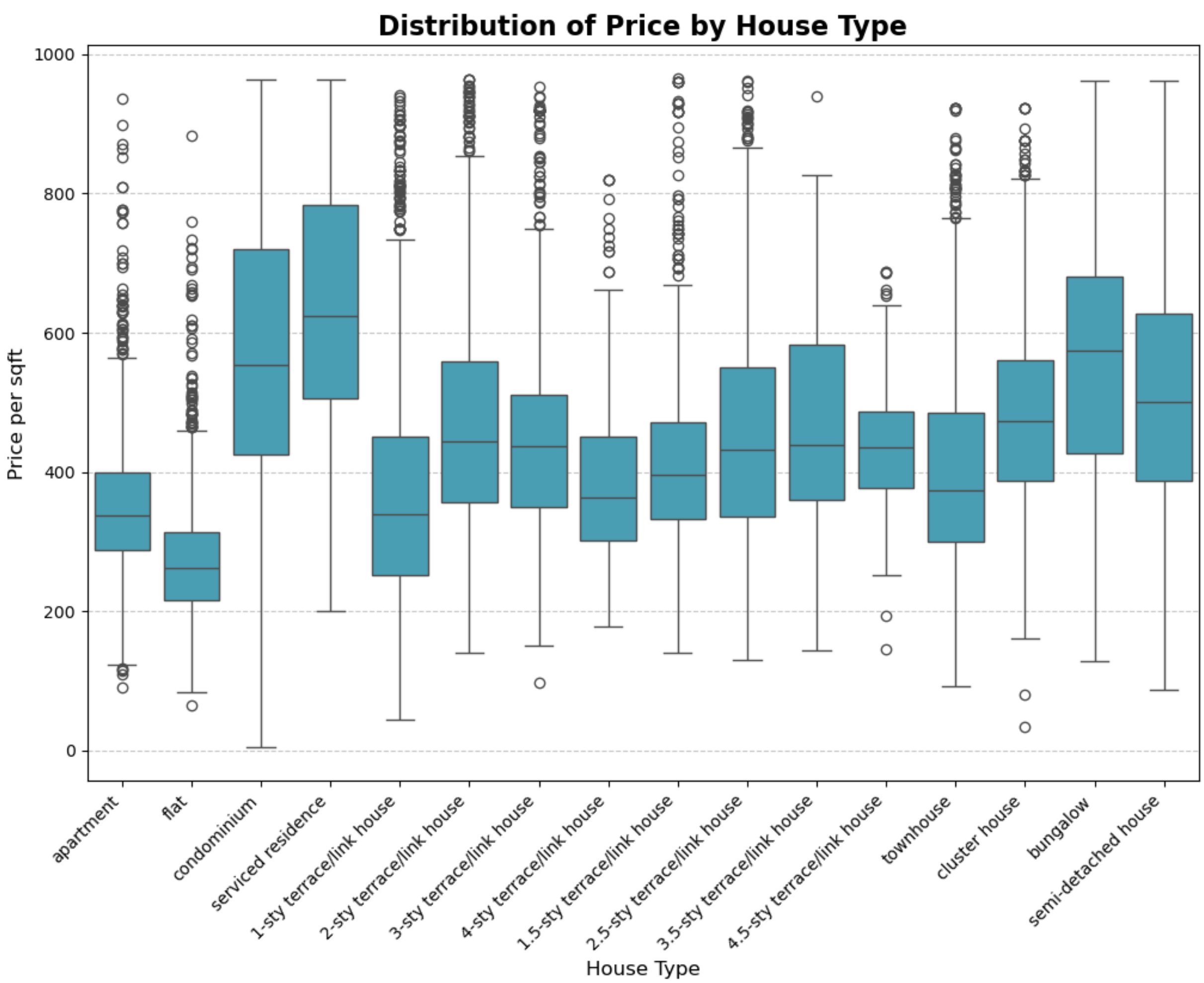

Distribution of Price by Bedrooms
Price per sqft
0
200
400
600
800
1000
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
12.0
13.0
14.0
15.0
16.0
19.0
Bedrooms

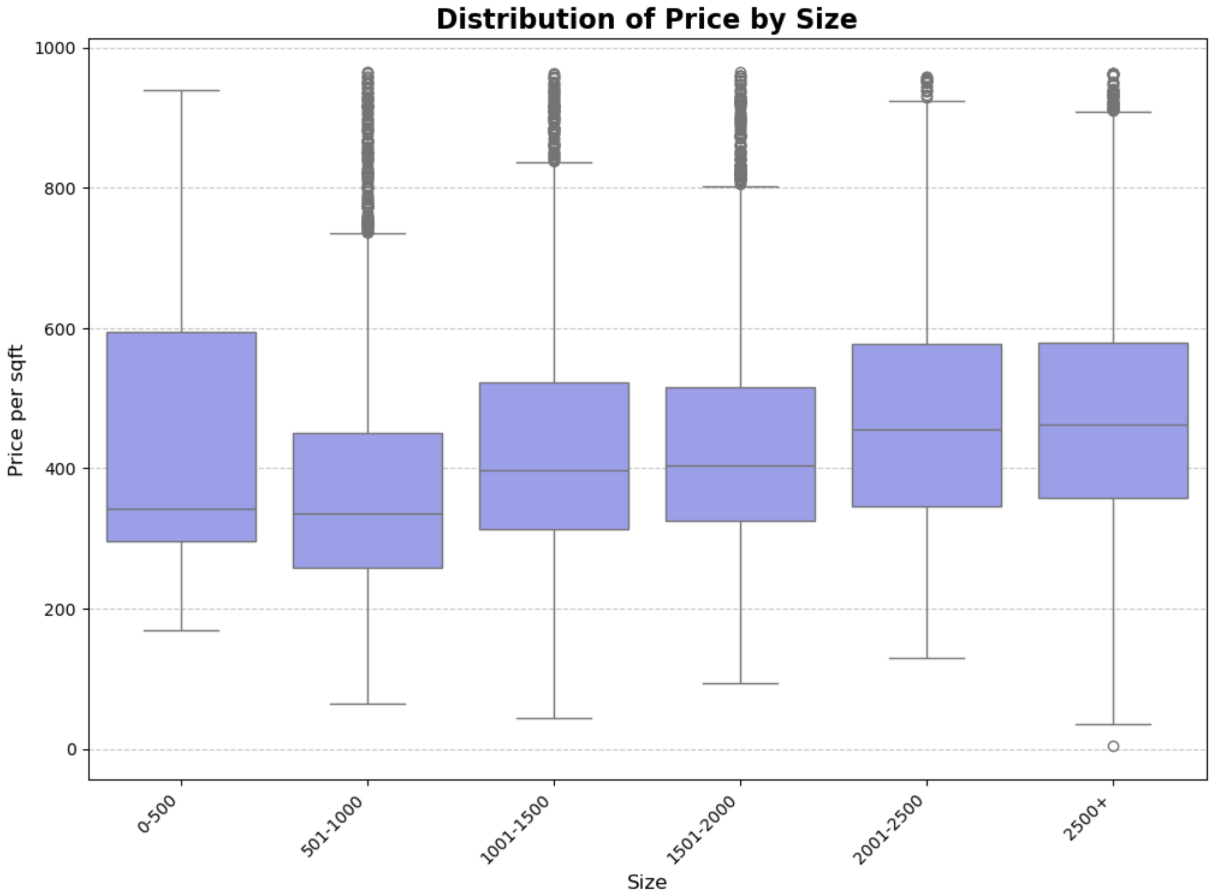

Distribution of Price by Tenure
Price per sqft
0
200
400
600
800
1000
freehold
leasehold
malay reserved land
private lease scheme
Tenure

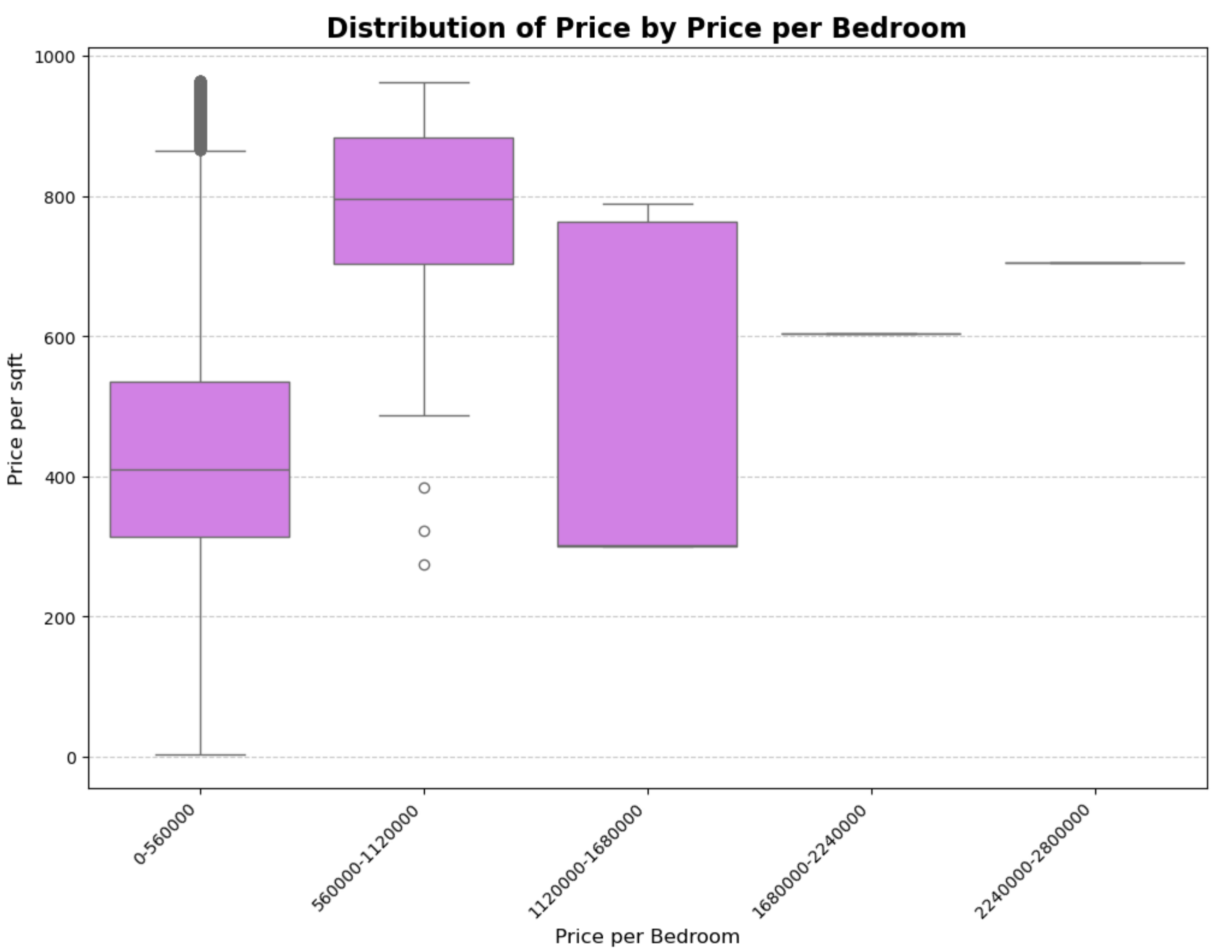

Distribution of Price by Furnished Type
Price per sqft
0
200
400
600
800
1000
fully furnished
unfurnished
partly furnished
Furnished Type


Distribution of Price by Bathrooms
Price per sqft
0
200
400
600
800
1000
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
13.0
15.0
Bathrooms

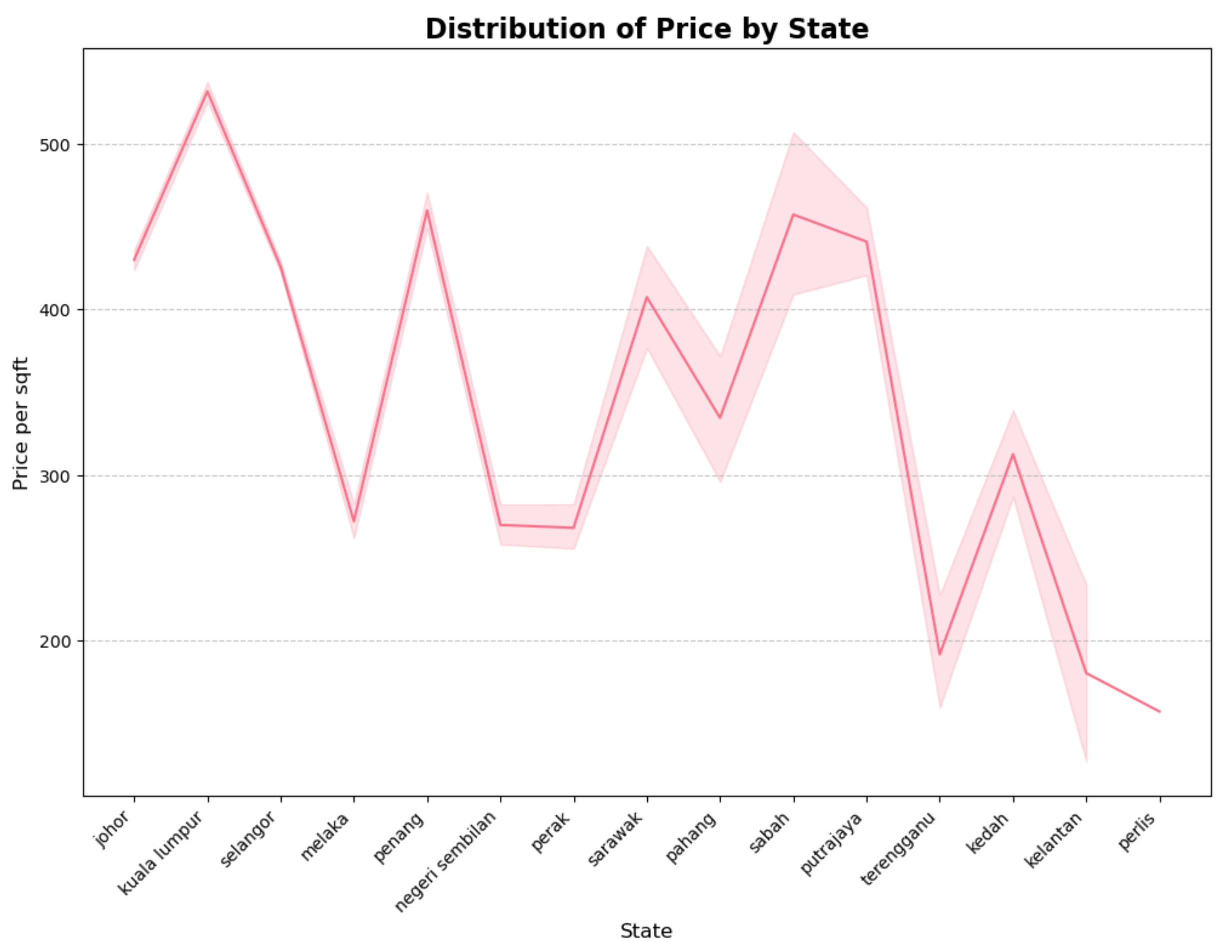

Distribution of Price by Car Slots
Price per sqft
0
200
400
600
800
1000
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
10.0
12.0
Car Slots

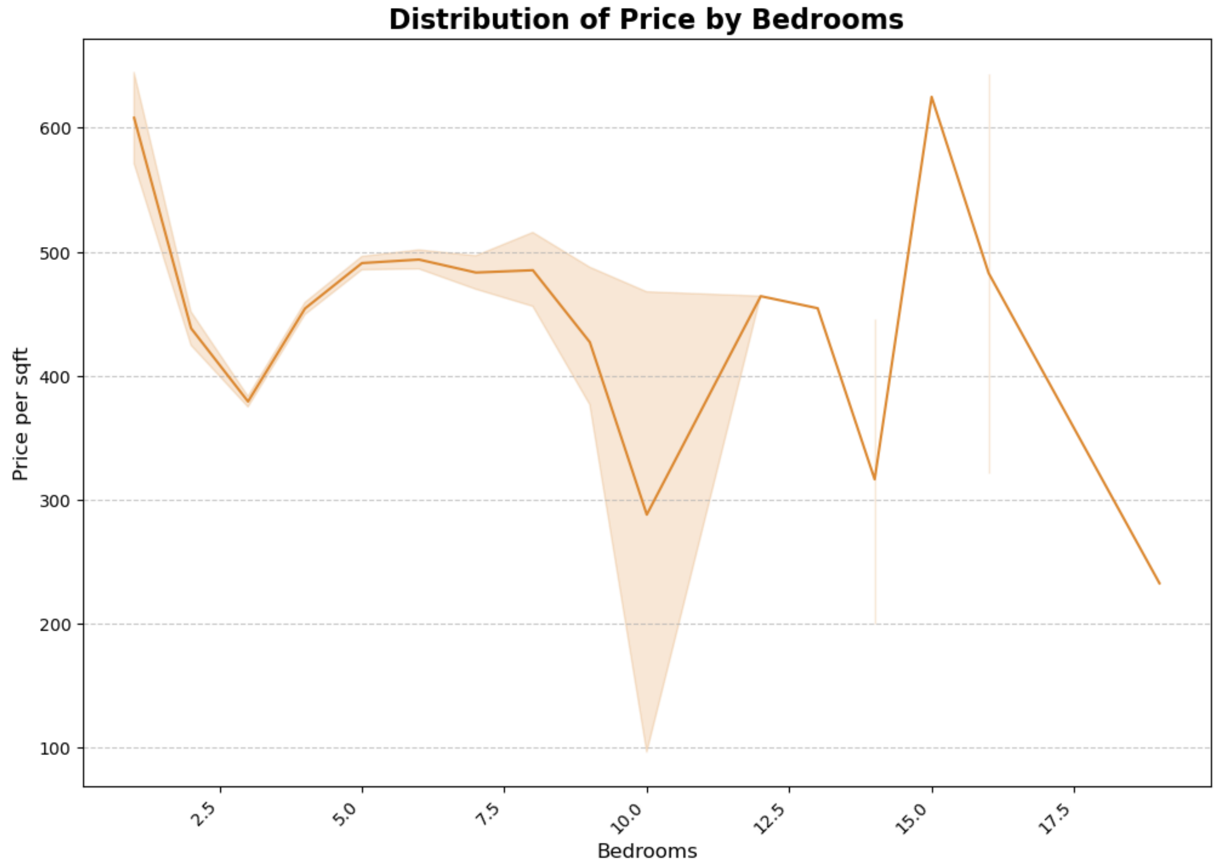

Distribution of Price by House Type
Price per sqft
0
200
400
600
800
1000
apartment
flat
condominium
serviced residence
1-sty terrace/link house
2-sty terrace/link house
3-sty terrace/link house
4-sty terrace/link house
1.5-sty terrace/link house
2.5-sty terrace/link house
3.5-sty terrace/link house
4.5-sty terrace/link house
townhouse
cluster house
bungalow
semi-detached house
House Type

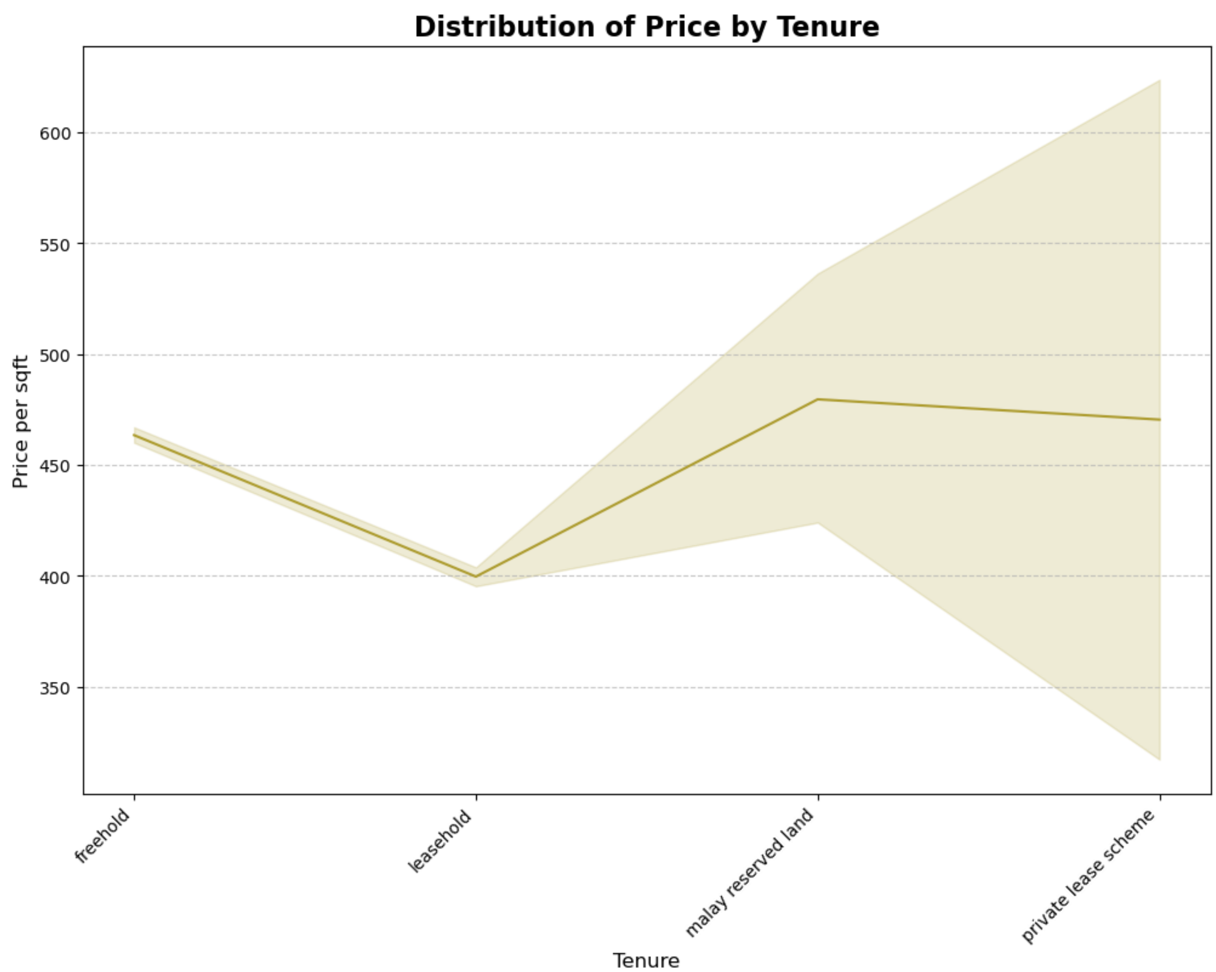

Distribution of Price by Size
Price per sqft
0
200
400
600
800
1000
0-500
501-1000
1001-1500
1501-2000
2001-2500
2500+
Size

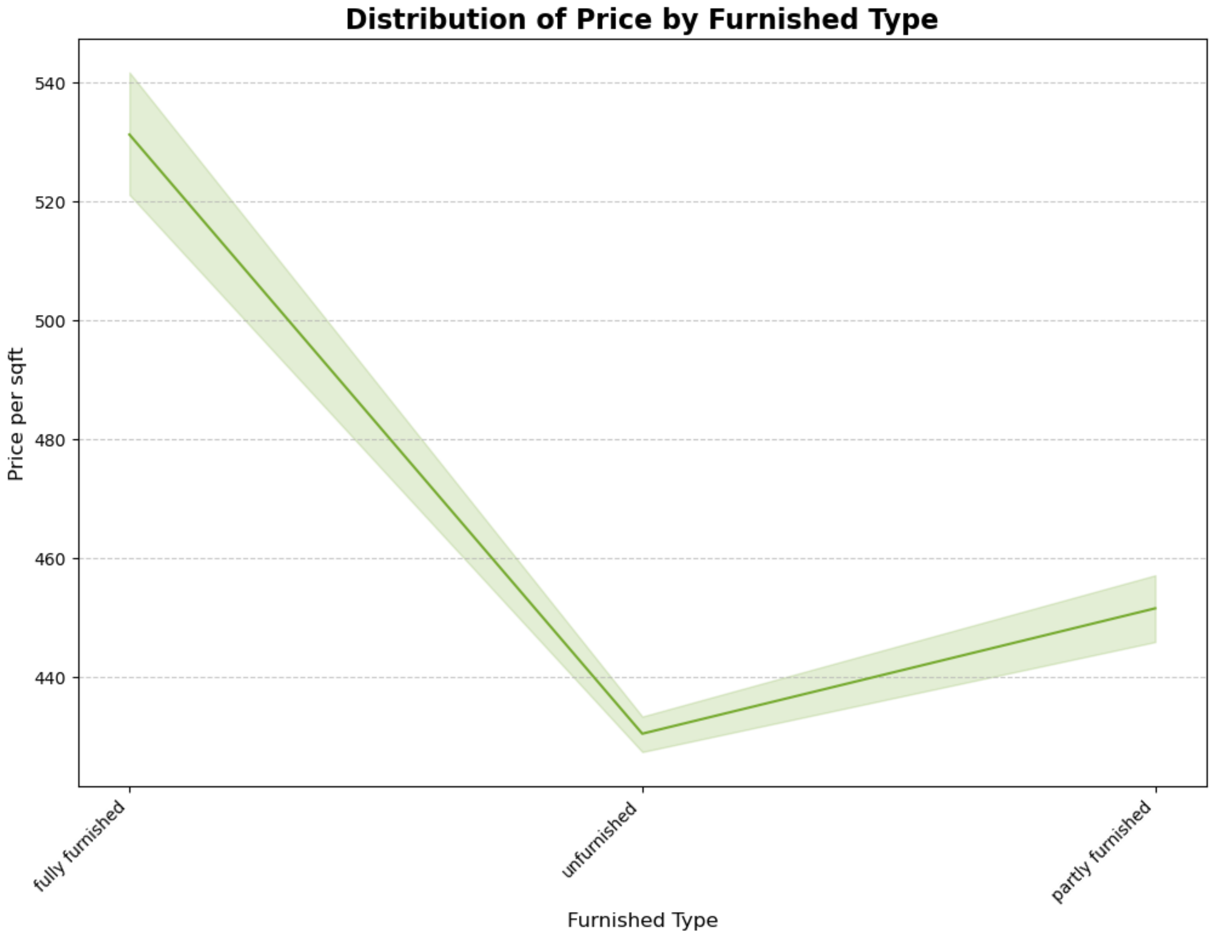

Distribution of Price by Price per Bedroom
Price per sqft
0
200
400
600
800
1000
0-560000
560000-1120000
1120000-1680000
1680000-2240000
2240000-2800000
Price per Bedroom

In [24]:
```
# Line plotting
colors = sns.color_palette("husl", n_colors=len(columns_to_plot)+3)

for idx, column in enumerate(columns_to_plot):
    plt.figure(figsize=(12, 8))
    sns.lineplot(data=df, x=column, y='Price per sqft', color=colors[idx])
    plt.title(f'Distribution of Price by {column}', fontsize=16, fontweight='bold')
    plt.xlabel(column, fontsize=12)
    plt.ylabel('Price per sqft', fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()

for idx, binned_column in enumerate(binned_columns):
    plt.figure(figsize=(12, 8))
    sns.lineplot(data=df, x=binned_column, y='Price per sqft', color=binned_colors[
    plt.title(f'Avg Price by {binned_labels[idx]}', fontsize=16, fontweight='bold')
    plt.xlabel(binned_labels[idx], fontsize=12)
    plt.ylabel('Price per sqft', fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```

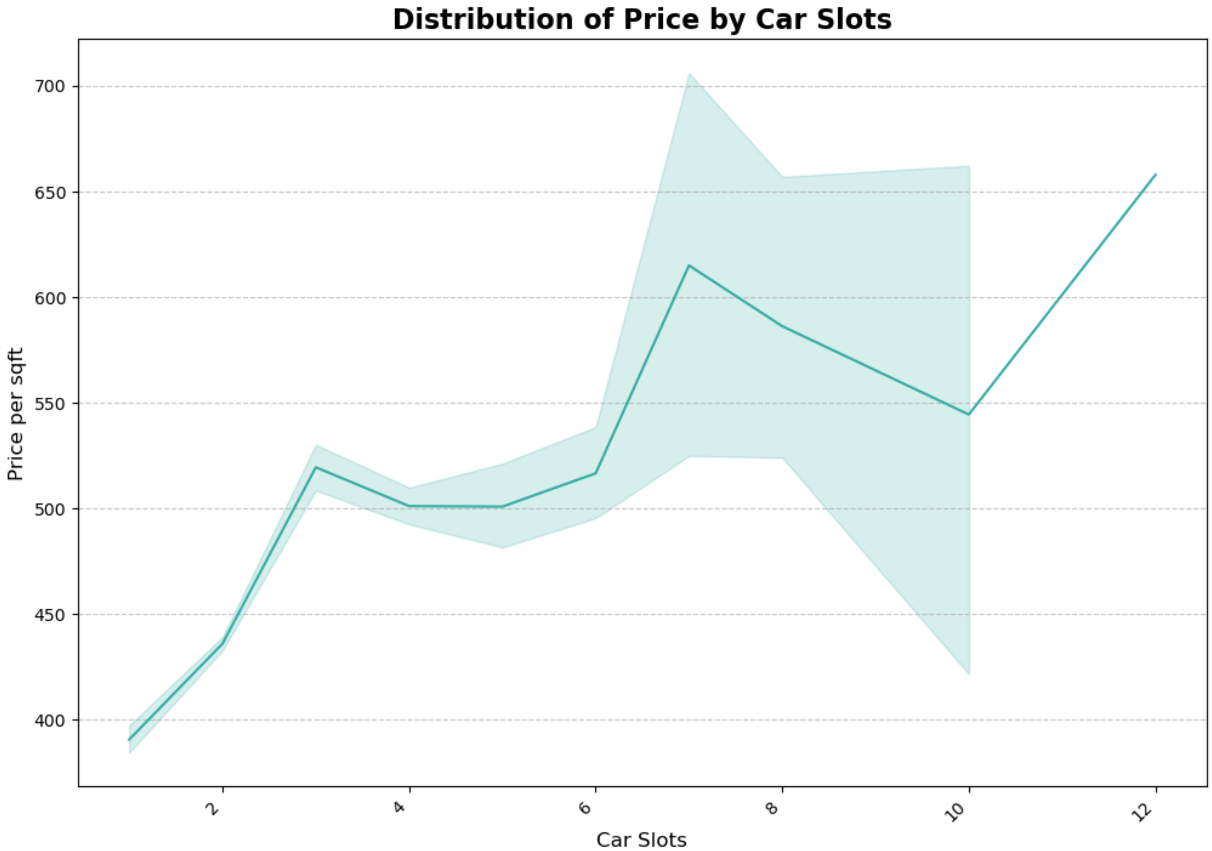


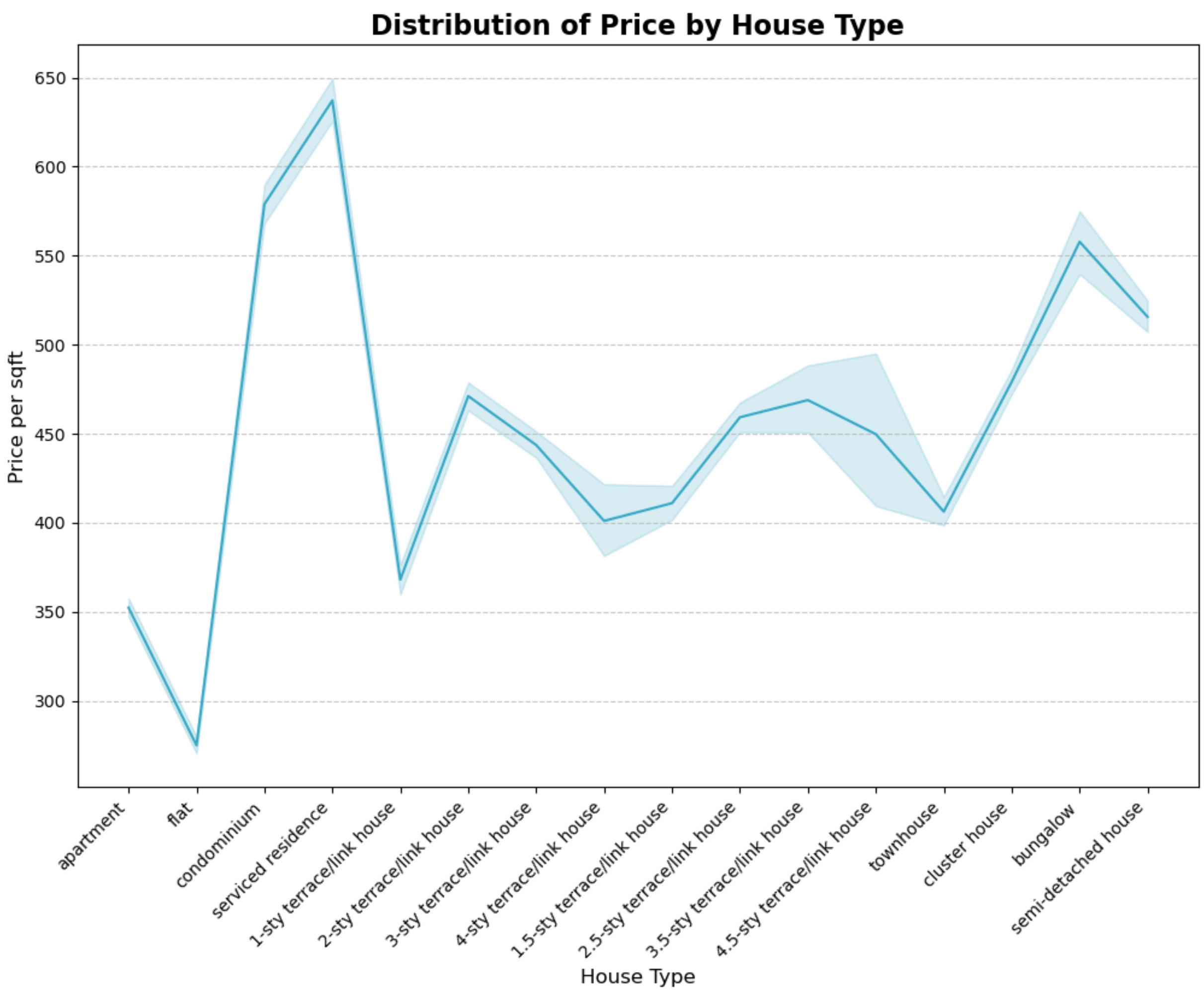

Distribution of Price by Bedrooms
Price per sqft
100
200
300
400
500
600
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Bedrooms

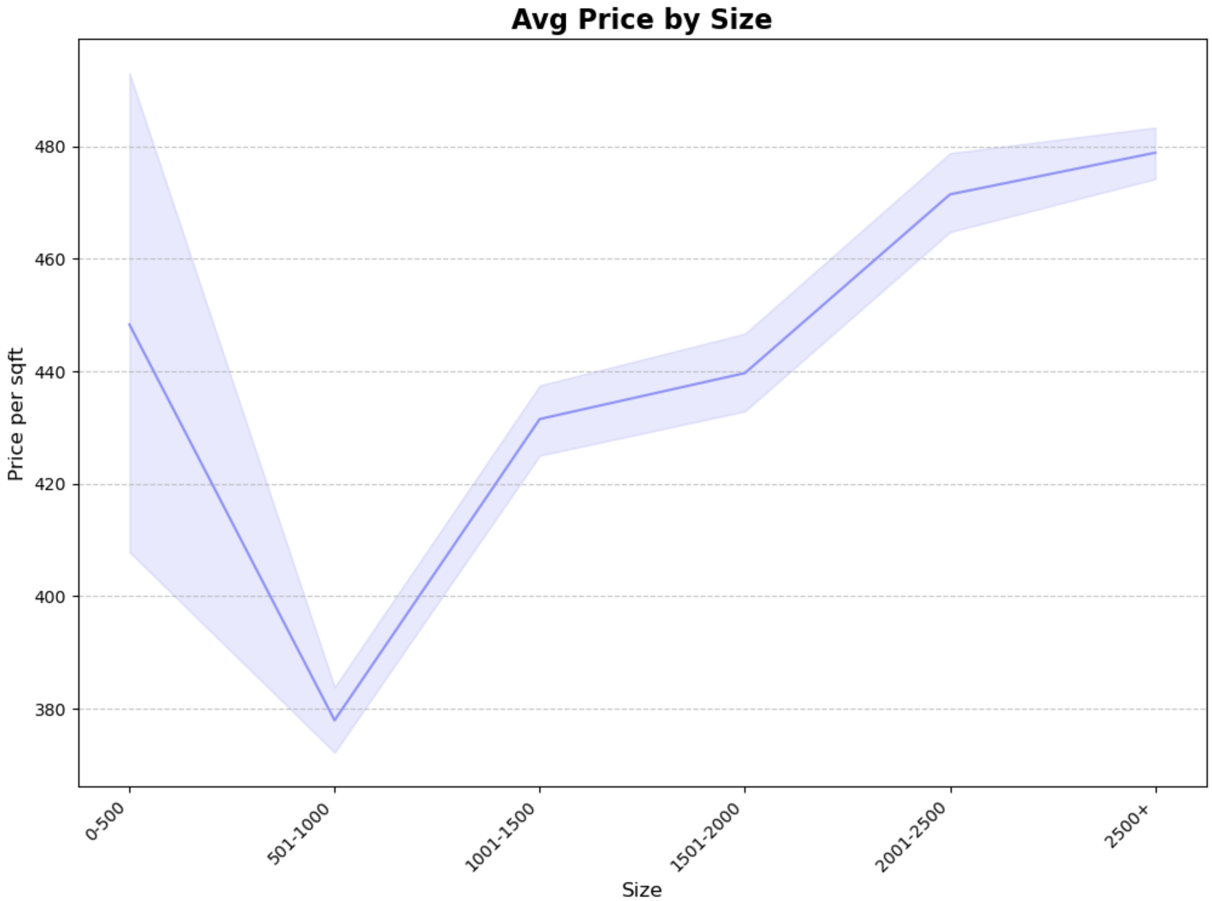

Distribution of Price by Tenure
Price per sqft
350
400
450
500
550
600
freehold
leasehold
malay reserved land
private lease scheme
Tenure


Distribution of Price by Furnished Type
Price per sqft
540
520
500
480
460
440
fully furnished
unfurnished
partly furnished
Furnished Type


Distribution of Price by Bathrooms
Price per sqft
600
500
400
300
200
2
4
6
8
10
12
14
Bathrooms


Distribution of Price by Car Slots
Price per sqft
Car Slots
700
650
600
550
500
450
400
2
4
6
8
10
12


Distribution of Price by House Type
Price per sqft
House Type
650
600
550
500
450
400
350
300
apartment
flat
condominium
serviced residence
1-sty terrace/link house
2-sty terrace/link house
3-sty terrace/link house
4-sty terrace/link house
1.5-sty terrace/link house
2.5-sty terrace/link house
3.5-sty terrace/link house
4.5-sty terrace/link house
townhouse
cluster house
bungalow
semi-detached house


Avg Price by Size
Price per sqft
480
460
440
420
400
380
0-500
501-1000
1001-1500
1501-2000
2001-2500
2500+
Size


Avg Price by Price per Bedroom
Price per sqft
800
700
600
500
400
300
0-560000
560000-1120000
1120000-1680000
1680000-2240000
2240000-2800000
Price per Bedroom

🏡 Capstone Project Documentation

1. Data Collection and Preprocessing

📥 Data Sources and Collection Methods

- Data collected from **iProperty Malaysia** using web scraping.
- House types scraped include:
  - 🏢 Apartments, Flats, Condominiums
  - 🏘️ Terrace/Link Houses (1-sty, 2-sty, etc.)
  - 🏠 Townhouses, Cluster Houses, Semi-Detached Houses, Bungalows, etc.
- Attributes scraped:
  - `Price`, `Size`, `Bedrooms`, `Bathrooms`, `Location`, `Facilities`, and `House Type`.

🧹 Data Cleaning Procedures

1. 🔄 **Remove Duplicates:** Avoided by maintaining a set of existing records.
2. 📉 **Handle Missing Values:**
   - Removed rows missing critical values (`Price`, `Size`).
   - Imputed numerical columns (`Bedrooms`, `Bathrooms`, `Car Slots`) using the median.
3. 🚫 **Address Outliers:** Used the IQR method to clip outliers in `Price`, `Size`, and `Price per Sqft`.
4. 🖊️ **Fix Inconsistent Formatting:** Cleaned categorical values using `.str.strip()` and `.str.lower()`.

🛠️ Feature Engineering

1. 🆕 **Derived Features:**
   - `Price per Bedroom`: Total price divided by the number of bedrooms.
2. 📊 **Binned Features:**
   - **`Size_Bins`:** `0-500`, `501-1000`, ..., `2500+`.
   - **`Price_per_Sqft_Bins`:** `0-200`, `201-400`, ..., `1K+`.
   - **`Price_per_Bedroom_Bins`:** `0-50K`, `51-100K`, ..., `250K+`.

✅ Data Quality Validation

- **Summary Statistics:** Used `df.describe()` for quality checks.
- **Missing Values:** Addressed and validated.

📄 Documentation of Preprocessing Steps

- Code implemented the above cleaning and preprocessing steps systematically.

---

2. Exploratory Data Analysis

📊 Summary Statistics

- Generated metrics for numerical columns:
  - **Mean:** `Price` : RM 1.6M.
  - **Median:** `Bedrooms` : 4.
  - **Range:** `Price` : RM 92K to RM 4.4M.

📈 Distribution Analysis

- Created **Histograms** and **KDE plots**:
  - `Price` is right-skewed; most properties are under RM 2M.
  - `Size` clusters between 1,000–3,000 sq. ft.

🔗 Correlation Analysis

- **Heatmap** findings:
  - `Price` and `Size` are strongly correlated (+0.8).
  - Moderate correlation between `Price per Sqft` and `Bedrooms` .

🔍 Key Trends

- **Scatterplots:**
  - Larger properties generally have higher prices.
  - More bedrooms correlate with higher prices.

📑 Segmentation

- **Size Bins:** Most properties are `1001-1500` sq. ft.
- **Price per Bedroom Bins:** Majority fall in the `51K-100K` range.

📊 Visualizations

1. 📊 **Bar Plots:** Average price by `State` , `Tenure` , and more.
2. 📈 **Scatterplots:** Price trends against numerical variables.
3. 📦 **Box Plots:** Distribution of prices by `Size_Bins` and `Price_per_Sqft_Bins` .
4. 📊 **Histograms:** Distributions of `Price` , `Size` , and other variables.
5. 🌡️ **Heatmaps:** Correlation matrix.

📝 Key Findings

- **Larger properties = Higher prices.**
- **Freehold properties dominate listings.**
- **Price per square foot** varies significantly by state and property type.

🔮 Potential Areas for Deeper Investigation

1. 📍 **Price Variations by Location:** Compare districts within states.
2. 🏊 **Impact of Facilities:** Explore how facilities affect pricing.
3. 📊 **Market Segmentation:** Examine trends in high-demand categories like condominiums or terrace houses.

📝 Notes

- 🎯 **Objective:** The primary goal of this project is to analyze property pricing trends in Malaysia and identify key factors that influence property value.
- 🌏 **Data Scope:**
  - The dataset was scraped from **iProperty Malaysia**.
  - It includes various house types such as 🏢 apartments, 🏠 terrace houses, and 🏡 bungalows.
- ⚠️ **Limitations:**
  - The dataset may not fully represent all property listings in Malaysia.
  - Missing values for certain columns were imputed, which could introduce bias.
  - The analysis assumes the scraped data is accurate and up-to-date.
- 🛠️ **Tools Used:**
  - Python libraries such as:
    - 🔗 **Selenium** for web scraping.
    - 📊 **Pandas** for data cleaning and manipulation.
    - 📈 **Seaborn & Matplotlib** for visualization.
- 🔮 **Future Steps:**
  - Collect more recent data to expand the analysis over time.
  - Include 🏘️ rental properties in the dataset for rental trend analysis.

- Apply 🤖 **machine learning models** to predict property prices based on key features.

Legend

- 📥 Data collection
- 🧹 Cleaning
- 📊 Visuals
- 🔍 Trends
- 🔗 Correlation
- 📝 Findings
- 🔮 Deeper investigation

In [25]:
```
# Display all columns without truncation
pd.set_option('display.max_columns', None)


# Display all rows without truncation
pd.set_option('display.max_rows', None)

# Reset back to default (optional)

# pd.reset_option('display.max_columns')
# pd.reset_option('display.max_rows')
```

In [26]:
```
df = pd.read_csv('data/cleaned_combined_property_data.csv')
```

In [27]:
```
label_encoders = {}
for column in df.columns:
    if df[column].dtype == 'object' and column != 'Facilities':
        label_encoder = LabelEncoder()
        df[column] = label_encoder.fit_transform(df[column])
        label_encoders[column] = label_encoder
```

In [28]:
```
for column, encoder in label_encoders.items():
    print(f"Column: {column}")
    print(f"Mapping: {dict(enumerate(encoder.classes_))}")
```

```
Column: District
Mapping: {0: 'alai', 1: 'alor gajah', 2: 'alor setar', 3: 'ampang', 4: 'ara damansar
a', 5: 'ayer hitam', 6: 'ayer itam', 7: 'ayer keroh', 8: 'ayer molek', 9: 'bachang',
10: 'bachok', 11: 'bagan serai', 12: 'bahau', 13: 'balai panjang', 14: 'balakong', 1
5: 'balik pulau', 16: 'bandar darulaman', 17: 'bandar enstek', 18: 'bandar kinrara',
19: 'bandar menjalara', 20: 'bandar sri damansara', 21: 'bandar sri sendayan', 22: '
bandar sungai long', 23: 'bandar tasik selatan', 24: 'bandar utama', 25: 'banggul',
26: 'bangi', 27: 'bangsar', 28: 'banting', 29: 'batang berjuntai', 30: 'batu arang',
31: 'batu berendam', 32: 'batu caves', 33: 'batu feringghi', 34: 'batu gajah', 35: '
batu kawan', 36: 'batu maung', 37: 'batu pahat', 38: 'bayan baru', 39: 'bayan lepas'
, 40: 'bedong', 41: 'bekenu', 42: 'bemban', 43: 'bentong', 44: 'beranang', 45: 'bert
am', 46: 'beserah', 47: 'bidor', 48: 'bota', 49: 'brickfields', 50: 'bukit baru', 51
: 'bukit jalil', 52: 'bukit jambul', 53: 'bukit katil', 54: 'bukit kayu hitam', 55:
'bukit kepayang', 56: 'bukit kiara', 57: 'bukit lintang', 58: 'bukit mertajam', 59:
'bukit minyak', 60: 'bukit payung', 61: 'bukit raja', 62: 'bukit tunku (kenny hill
s)', 63: 'butterworth', 64: 'cameron highlands', 65: 'chemor', 66: 'cheng', 67: 'che
ras', 68: 'country heights damansara', 69: 'cyberjaya', 70: 'damansara damai', 71: '
damansara heights', 72: 'damansara perdana', 73: 'dengkil', 74: 'desa parkcity', 75:
'desa petaling', 76: 'dungun', 77: 'durian tunggal', 78: 'dutamas', 79: 'duyong', 80
: 'gelang patah', 81: 'gelugor', 82: 'george town', 83: 'glenmarie', 84: 'gombak', 8
5: 'gopeng', 86: 'gurney', 87: 'gurun', 88: 'hulu langat', 89: 'hulu telom', 90: 'ij
ok', 91: 'ipoh', 92: 'iskandar puteri (nusajaya)', 93: 'jalan ipoh', 94: 'jalan klan
g lama (old klang road)', 95: 'jalan kuching', 96: 'jasin', 97: 'jelutong', 98: 'jen
jarom', 99: 'jeram batu', 100: 'jimah', 101: 'jinjang', 102: 'jitra', 103: 'johor ba
hru', 104: 'juru', 105: 'kajang', 106: 'kampar', 107: 'kampung kerinchi (bangsar sou
th)', 108: 'kamunting', 109: 'kangar', 110: 'kapar', 111: 'karak', 112: 'karangan',
113: 'kayu ara', 114: 'kemaman', 115: 'kepala batas', 116: 'kepong', 117: 'keramat',
118: 'kerling', 119: 'kertih', 120: 'kl city centre', 121: 'kl eco city', 122: 'kl s
entral', 123: 'klang', 124: 'klebang', 125: 'kluang', 126: 'kota bharu', 127: 'kota
damansara', 128: 'kota kinabalu', 129: 'kota kuala muda', 130: 'kota tinggi', 131: '
krubong', 132: 'kuah', 133: 'kuala kangsar', 134: 'kuala kedah', 135: 'kuala ketil',
136: 'kuala kurau', 137: 'kuala nerang', 138: 'kuala paka', 139: 'kuala pilah', 140:
'kuala selangor', 141: 'kuala terengganu', 142: 'kuang', 143: 'kuantan', 144: 'kuban
g pasu', 145: 'kubang semang', 146: 'kuchai lama', 147: 'kuching', 148: 'kulai', 149
: 'kulim', 150: 'labis', 151: 'labu', 152: 'lahat', 153: 'lenggeng', 154: 'lukut', 1
55: 'lumut', 156: 'lunas', 157: 'machang', 158: 'mantin', 159: 'masai', 160: 'masjid
tanah', 161: 'melaka city', 162: 'menglembu', 163: 'mentakab', 164: 'merlimau', 165:
'miri', 166: 'mont kiara', 167: 'muadzam shah', 168: 'muar', 169: 'mutiara damansara
', 170: 'nibong tebal', 171: 'nilai', 172: 'padang meha', 173: 'padang serai', 174:
'paloh', 175: 'pantai', 176: 'papar', 177: 'parit buntar', 178: 'pasir gudang', 179:
'pasir panjang', 180: 'paya rumput', 181: 'pekan', 182: 'pekan nenas', 183: 'penampa
ng', 184: 'pengerang', 185: 'penor', 186: 'perai', 187: 'perling', 188: 'permas jaya
', 189: 'petaling jaya', 190: 'pokok sena', 191: 'pontian', 192: 'port dickson', 193
: 'port klang (pelabuhan klang)', 194: 'presint 11', 195: 'presint 18', 196: 'presin
t 8', 197: 'puchong', 198: 'puchong perdana', 199: 'pulau tikus', 200: 'puncak alam'
, 201: 'putrajaya', 202: 'rantau', 203: 'rasah', 204: 'rawang', 205: 'sabak bernam',
206: 'salak selatan', 207: 'salak south', 208: 'samarahan', 209: 'sandakan', 210: 's
aujana', 211: 'saujana utama', 212: 'seberang jaya', 213: 'seberang perai', 214: 'se
denak', 215: 'segambut', 216: 'selama', 217: 'selandar', 218: 'selayang', 219: 'sema
bok', 220: 'semenyih', 221: 'senai', 222: 'senawang', 223: 'sentul', 224: 'sepang',
225: 'seputeh', 226: 'serdang', 227: 'seremban', 228: 'seremban 2', 229: 'seremban j
aya', 230: 'serendah', 231: 'seri iskandar', 232: 'seri kembangan', 233: 'serom', 23
4: 'setapak', 235: 'setia alam', 236: 'setiawangsa', 237: 'shah alam', 238: 'sik', 2
39: 'sikamat', 240: 'simpang', 241: 'simpang ampat', 242: 'simpang pulai', 243: 'sit
iawan', 244: 'skudai', 245: 'sri gading', 246: 'sri hartamas', 247: 'sri petaling',
248: 'subang', 249: 'subang jaya', 250: 'sungai ara', 251: 'sungai besi', 252: 'sung
```

```
ai buloh', 253: 'sungai dua', 254: 'sungai jawi', 255: 'sungai karang', 256: 'sungai
karangan', 257: 'sungai kob', 258: 'sungai lalang', 259: 'sungai petani', 260: 'sung
ai siput', 261: 'sungei baru tengah', 262: 'sungei petai', 263: 'sunway', 264: 'sunw
ay spk', 265: 'taiping', 266: 'taman desa', 267: 'taman tun dr ismail', 268: 'tambun
', 269: 'tampoi', 270: 'tangkak', 271: 'tanjong duabelas', 272: 'tanjong karang', 27
3: 'tanjong minyak', 274: 'tanjong rambutan', 275: 'tanjung bungah', 276: 'tanjung m
alim', 277: 'tanjung tokong', 278: 'tasek gelugor', 279: 'tawau', 280: 'tebrau', 281
: 'telok panglima garang', 282: 'teluk intan', 283: 'teluk kemang', 284: 'teluk kumb
ar', 285: 'temerloh', 286: 'temin', 287: 'titiwangsa', 288: 'tronoh', 289: 'tropican
a', 290: 'tuaran', 291: 'ujong pasir', 292: 'ulu kelang', 293: 'ulu kinta', 294: 'ul
u langat', 295: 'ulu tiram', 296: 'wangsa maju'}
Column: State
Mapping: {0: 'johor', 1: 'kedah', 2: 'kelantan', 3: 'kuala lumpur', 4: 'melaka', 5:
'negeri sembilan', 6: 'pahang', 7: 'penang', 8: 'perak', 9: 'perlis', 10: 'putrajaya
', 11: 'sabah', 12: 'sarawak', 13: 'selangor', 14: 'terengganu'}
Column: Tenure
Mapping: {0: 'freehold', 1: 'leasehold', 2: 'malay reserved land', 3: 'private lease
scheme'}
Column: Furnished Type
Mapping: {0: 'fully furnished', 1: 'partly furnished', 2: 'unfurnished'}
Column: House Type
Mapping: {0: '1-sty terrace/link house', 1: '1.5-sty terrace/link house', 2: '2-sty
terrace/link house', 3: '2.5-sty terrace/link house', 4: '3-sty terrace/link house',
5: '3.5-sty terrace/link house', 6: '4-sty terrace/link house', 7: '4.5-sty terrace/
link house', 8: 'apartment', 9: 'bungalow', 10: 'cluster house', 11: 'condominium',
12: 'flat', 13: 'semi-detached house', 14: 'serviced residence', 15: 'townhouse'}
```

In [29]:
```python
df['Facilities'] = df['Facilities'].apply(lambda x: literal_eval(x) if pd.notna(x)
mlb = MultiLabelBinarizer()
facilities_encoded = mlb.fit_transform(df['Facilities'])
facilities_df = pd.DataFrame(facilities_encoded, columns=mlb.classes_)
```

In [30]:
```python
df = df.drop(columns=['Facilities']).reset_index(drop=True)
df = pd.concat([df, facilities_df], axis=1)
df.head()
```

Out[30]:

	Unnamed: 0	Price	District	State	Bedrooms	Tenure	Furnished Type	Size	Bathrooms	S
0	0	680000	280	0	4.0	0	0	1317.0	3.0	
1	1	218000	187	0	3.0	0	2	750.0	2.0	
2	2	258000	95	3	3.0	0	1	905.0	2.0	
3	3	290000	189	13	3.0	1	1	775.0	2.0	
4	4	350000	103	0	3.0	1	2	1078.0	2.0	

In [31]:
```python
X = df[df.columns.difference(['Unnamed: 0', 'Price per Bedroom', 'Room Density', 'P
```

In [32]:
```python
y = df['Price per sqft']
```

In [33]:
```python
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_stat
```

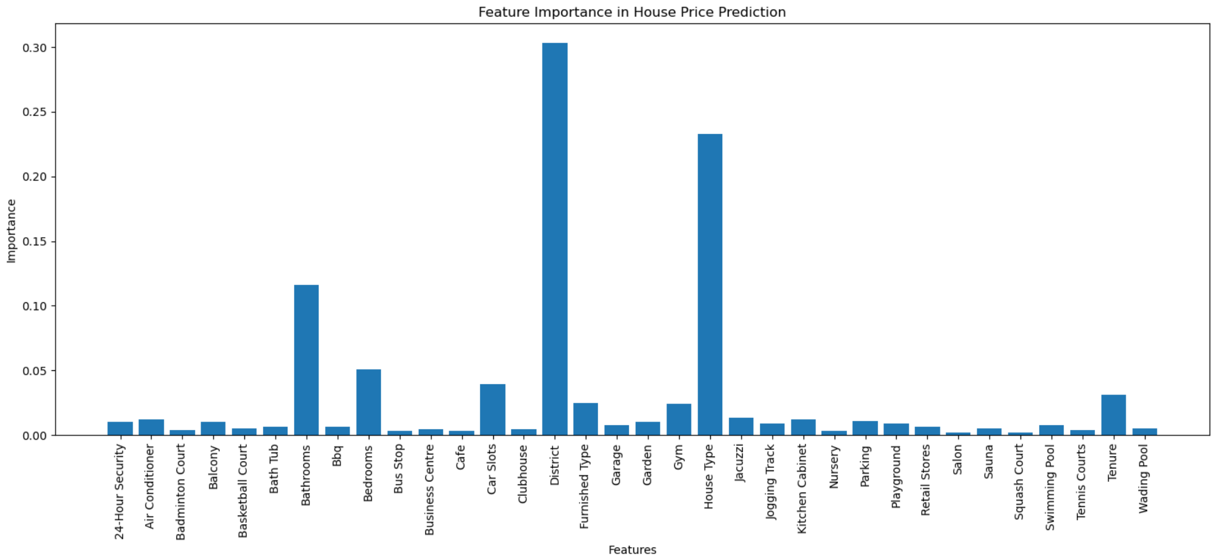
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [42]:
```
param_grid = {
    'n_estimators': [100, 200, 300, 500],
    'max_depth': [None, 10, 20, 30, 40],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 8],
    'max_features': ['sqrt', 'log2', None]
}

grid_search = GridSearchCV(
    estimator=RandomForestRegressor(random_state=42),
    param_grid=param_grid,
    cv=5,
    n_jobs=-1,
    scoring='r2',
    error_score='raise'
)

grid_search.fit(X_train, y_train)

model = grid_search.best_estimator_

y_pred = model.predict(X_test)

# Calculate all metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print results
print("Best Parameters:", grid_search.best_params_)
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-squared Score: {r2}")
```

```
Best Parameters: {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min
_samples_split': 5, 'n_estimators': 500}
Mean Squared Error: 12598.996871330675
Mean Absolute Error: 80.94865966742712
R-squared Score: 0.5956342802611385
```

In [43]:
```
feature_importance = model.feature_importances_

plt.figure(figsize=(15, 7))
plt.bar(X.columns, feature_importance)
plt.title('Feature Importance in House Price Prediction')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

top_features = sorted(zip(X.columns, feature_importance), key=lambda x: x[1], rever
```

```
for feature, importance in top_features:
    print(f"{feature}: {importance * 100:.2f}%")
```




```
District: 30.31%
House Type: 23.30%
Bathrooms: 11.62%
Bedrooms: 5.09%
Car Slots: 3.92%
```

In [44]:
```
joblib.dump(model, 'house_price_model.pkl')
joblib.dump(scaler, 'scaler_price_scaler.pkl')
```

Out[44]:
```
['scaler_price_scaler.pkl']
```

In [45]:
```
X.columns
```

Out[45]:
```
Index(['24-Hour Security', 'Air Conditioner', 'Badminton Court', 'Balcony',
       'Basketball Court', 'Bath Tub', 'Bathrooms', 'Bbq', 'Bedrooms',
       'Bus Stop', 'Business Centre', 'Cafe', 'Car Slots', 'Clubhouse',
       'District', 'Furnished Type', 'Garage', 'Garden', 'Gym', 'House Type',
       'Jacuzzi', 'Jogging Track', 'Kitchen Cabinet', 'Nursery', 'Parking',
       'Playground', 'Retail Stores', 'Salon', 'Sauna', 'Squash Court',
       'Swimming Pool', 'Tennis Courts', 'Tenure', 'Wading Pool'],
      dtype='object')
```