In [49]:
```
#SAAD SIDDIQUI
#23250068
```

In [1]:
```
%pip install seaborn
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
df = pd.read_csv("Melbourne_housing.csv")
```

In [4]:
```
sns.set_theme(style="darkgrid")
```

In [5]:
```
pd.pandas.set_option("display.max_columns",None)
```

In [ ]:
```
# Understanding the type of Data
# Data Loading and Initial Exploration (20 points):
# Provide information on the dataset, including the number of rows and columns.
```

In [6]:
```
df.shape
```

Out[6]:
```
(34857, 22)
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Suburb             0
Address            0
Rooms              0
Type               0
Method             0
SellerG            0
Date               0
Distance           1
Postcode           1
Bedroom         8217
Bathroom        8226
Car             8728
Landsize       11810
BuildingArea   21097
YearBuilt      19306
CouncilArea        3
Latitude        7976
Longtitude      7976
Regionname         0
Propertycount      3
ParkingArea        0
Price           7610
dtype: int64
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Suburb         34857 non-null  object 
 1   Address        34857 non-null  object 
 2   Rooms          34857 non-null  int64  
 3   Type           34857 non-null  object 
 4   Method         34857 non-null  object 
 5   SellerG        34857 non-null  object 
 6   Date           34857 non-null  object 
 7   Distance       34856 non-null  float64
 8   Postcode       34856 non-null  float64
 9   Bedroom        26640 non-null  float64
 10  Bathroom       26631 non-null  float64
 11  Car            26129 non-null  float64
 12  Landsize       23047 non-null  float64
 13  BuildingArea   13760 non-null  object 
 14  YearBuilt      15551 non-null  float64
 15  CouncilArea    34854 non-null  object 
 16  Latitude       26881 non-null  float64
 17  Longtitude     26881 non-null  float64
 18  Regionname     34857 non-null  object 
 19  Propertycount  34854 non-null  float64
 20  ParkingArea    34857 non-null  object 
 21  Price          27247 non-null  float64
dtypes: float64(11), int64(1), object(10)
memory usage: 4.5+ MB
```

In [9]:
```
datatype = df.dtypes
num_col = datatype[(datatype == 'float64') | (datatype == 'int64')].index.tolist()
cat_col = datatype[(datatype == 'object')].index.tolist()
print("Categorical Columns :",cat_col)
print("Numerical Columns :",num_col)
```

```
Categorical Columns : ['Suburb', 'Address', 'Type', 'Method', 'SellerG', 'Date',
'BuildingArea', 'CouncilArea', 'Regionname', 'ParkingArea']
Numerical Columns : ['Rooms', 'Distance', 'Postcode', 'Bedroom', 'Bathroom', 'Ca
r', 'Landsize', 'YearBuilt', 'Latitude', 'Longtitude', 'Propertycount', 'Price']
```

In [10]:
```
df['Rooms'] = df['Rooms'].astype('object')
df['Postcode'] = df['Postcode'].astype('object')
df['Distance'] = df['Distance'].astype('object')
df['Bedroom'] = df['Bedroom'].astype('object')
df['Bathroom'] = df['Bathroom'].astype('object')
df['Car'] = df['Car'].astype('object')
df['Landsize'] = df['Landsize'].astype('object')
df['YearBuilt'] = df['YearBuilt'].astype('object')
df['Latitude'] = df['Latitude'].astype('object')
df['Propertycount'] = df['Propertycount'].astype('object')
```

In [11]:
```
datatype = df.dtypes
num_col = datatype[(datatype == 'float64') | (datatype == 'int64')].index.tolist()
cat_col = datatype[(datatype == 'object')].index.tolist()
print("Categorical Columns :",cat_col)
print("Numerical Columns :",num_col)
```

```
Categorical Columns : ['Suburb', 'Address', 'Rooms', 'Type', 'Method', 'SellerG',
'Date', 'Distance', 'Postcode', 'Bedroom', 'Bathroom', 'Car', 'Landsize', 'Buildin
gArea', 'YearBuilt', 'CouncilArea', 'Latitude', 'Regionname', 'Propertycount', 'Pa
rkingArea']
Numerical Columns : ['Longtitude', 'Price']
```

In [ ]:
```
# Understanding the target Variables with different Target variables
```

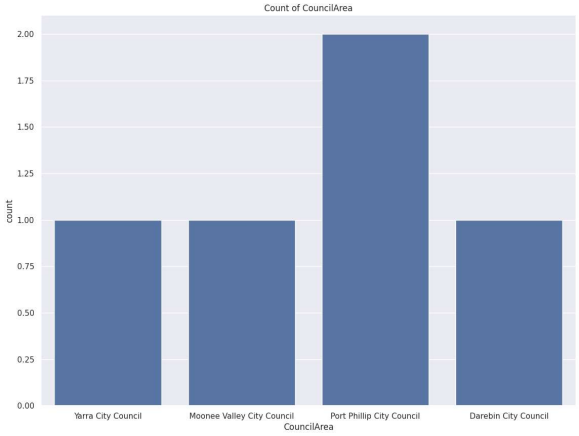
In [12]:
```
df["YearBuilt"].unique()
```

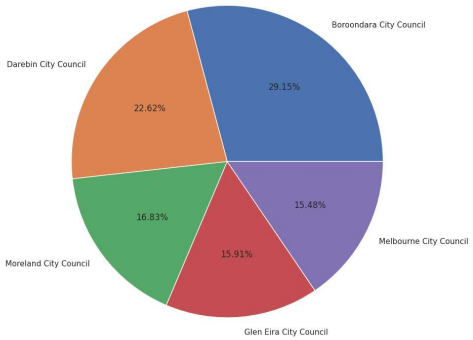
Out[12]:
```
array([nan, 2016.0, 1900.0, 2003.0, 1930.0, 2013.0, 2015.0, 1950.0,
       1965.0, 1970.0, 1964.0, 1910.0, 1960.0, 1980.0, 1920.0, 1985.0,
       2000.0, 1974.0, 1997.0, 2008.0, 1940.0, 1915.0, 1998.0, 1905.0,
       1957.0, 1890.0, 1935.0, 2004.0, 1989.0, 2009.0, 1955.0, 1993.0,
       1830.0, 1975.0, 1925.0, 2010.0, 1880.0, 2007.0, 2001.0, 2005.0,
       1904.0, 1971.0, 1995.0, 1990.0, 1983.0, 2012.0, 2014.0, 1857.0,
       1978.0, 1889.0, 2017.0, 1999.0, 1977.0, 2006.0, 2011.0, 2002.0,
       1938.0, 1919.0, 1969.0, 1863.0, 1893.0, 1945.0, 1912.0, 1968.0,
       1972.0, 1943.0, 1951.0, 1922.0, 1996.0, 1992.0, 1958.0, 1962.0,
       1967.0, 1934.0, 1887.0, 1976.0, 1973.0, 1994.0, 1982.0, 1907.0,
       1918.0, 1986.0, 1953.0, 1928.0, 1948.0, 1931.0, 1941.0, 1949.0,
       1988.0, 1991.0, 1924.0, 1923.0, 1897.0, 1984.0, 1987.0, 1856.0,
       1860.0, 2106.0, 1902.0, 1966.0, 1961.0, 1929.0, 1952.0, 1885.0,
       1881.0, 1870.0, 1959.0, 1926.0, 1939.0, 1903.0, 1911.0, 1906.0,
       1868.0, 1963.0, 1927.0, 1956.0, 1979.0, 1884.0, 1917.0, 1898.0,
       1888.0, 1954.0, 1909.0, 1894.0, 1854.0, 1946.0, 1937.0, 1871.0,
       1981.0, 1895.0, 1914.0, 1908.0, 1921.0, 1913.0, 1872.0, 1936.0,
       1875.0, 1947.0, 1932.0, 1891.0, 1862.0, 2018.0, 1899.0, 1855.0,
       1869.0, 1933.0, 1916.0, 1886.0, 2019.0, 1892.0, 1877.0, 1901.0,
       1850.0, 1896.0, 1800.0, 1879.0, 1196.0, 1883.0, 1942.0, 1876.0,
       1820.0], dtype=object)
```

In [13]:
```
df["CouncilArea"].unique()
```

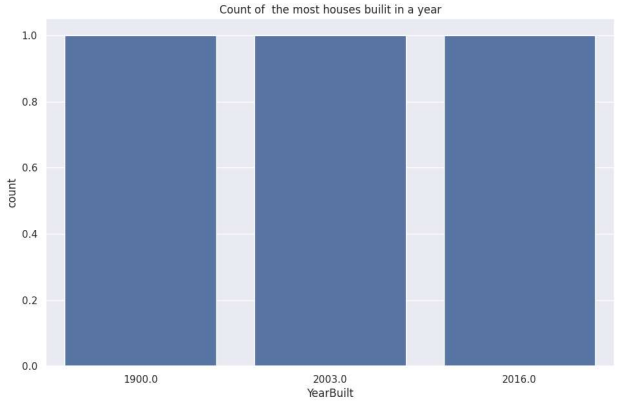
Out[13]:
```
array(['Yarra City Council', 'Moonee Valley City Council',
       'Port Phillip City Council', 'Darebin City Council',
       'Hobsons Bay City Council', 'Stonnington City Council',
       'Boroondara City Council', 'Monash City Council',
       'Glen Eira City Council', 'Whitehorse City Council',
       'Maribyrnong City Council', 'Bayside City Council',
       'Moreland City Council', 'Manningham City Council',
       'Melbourne City Council', 'Banyule City Council',
       'Brimbank City Council', 'Kingston City Council',
       'Hume City Council', 'Knox City Council', 'Maroondah City Council',
       'Casey City Council', 'Melton City Council',
       'Greater Dandenong City Council', 'Nillumbik Shire Council',
       'Cardinia Shire Council', 'Whittlesea City Council',
       'Frankston City Council', 'Macedon Ranges Shire Council',
       'Yarra Ranges Shire Council', 'Wyndham City Council',
       'Mitchell Shire Council', 'Moorabool Shire Council', nan],
      dtype=object)
```

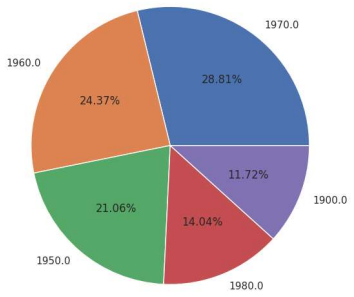
In [14]:
```
fig, ax = plt.subplots(1, 2, figsize = (30, 10))
ax[0].set_title("Count of CouncilArea")
percentage = df["CouncilArea"].value_counts().head(5)
labels = list(df["CouncilArea"].value_counts().head(5).index)

sns.countplot(x = df["CouncilArea"].head(5), ax = ax[0])
plt.pie(percentage,labels = labels, autopct= "%0.2f%%")

plt.show()
```



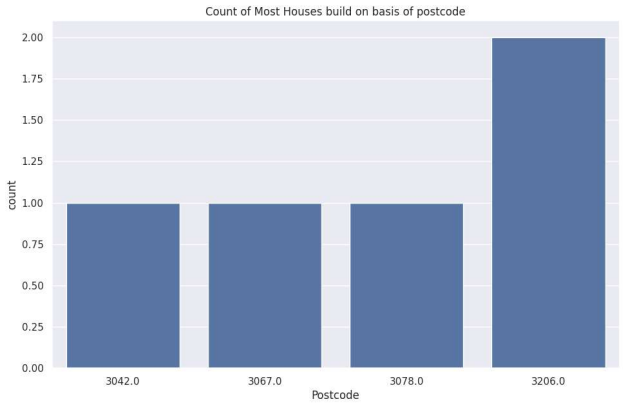


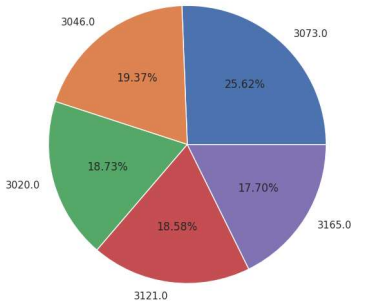
In [15]:
```
fig, ax = plt.subplots(1, 2, figsize = (25, 7))
ax[0].set_title("Count of  the most houses builit in a year")
percentage = df["YearBuilt"].value_counts().head(5)
labels = list(df["YearBuilt"].value_counts().head(5).index)

sns.countplot(x = df["YearBuilt"].head(5), ax = ax[0])
plt.pie(percentage,labels = labels, autopct= "%0.2f%%")

plt.show()
```





In [16]:
```
fig, ax = plt.subplots(1, 2, figsize = (25, 7))
ax[0].set_title("Count of Most Houses build on basis of postcode")
percentage = df["Postcode"].value_counts().head(5)
labels = list(df["Postcode"].value_counts().head(5).index)

sns.countplot(x = df["Postcode"].head(5), ax = ax[0])
plt.pie(percentage,labels = labels, autopct= "%0.2f%%")

plt.show()
```



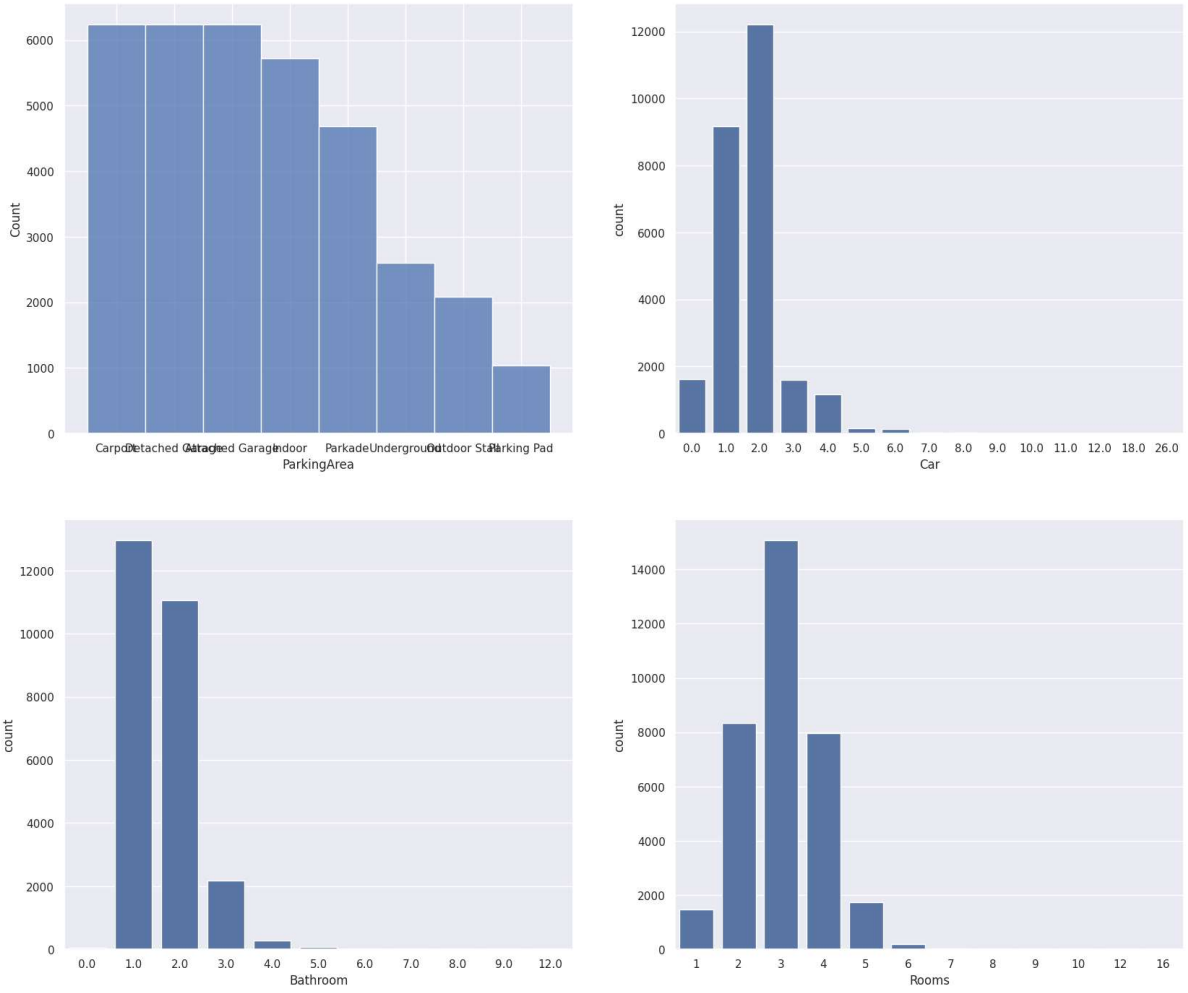


In [17]:
```
# Through these analysis we were trying to figure out three major KPIs for the hous
# YearBuilt: There might be government scheme or house shortage so more houses were
```

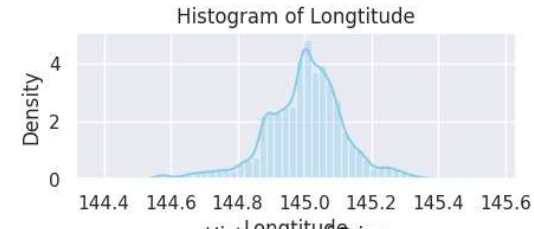
```
# PostCode: Some new area might be made of use or might be made an exclusive area c
# CouncilArea: Chechking out the area where the count of houses is greater as it co
```

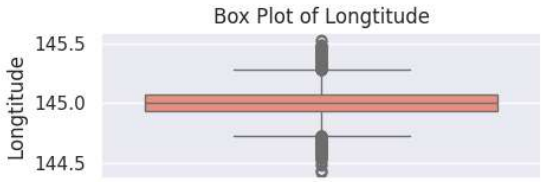
In [18]:
```
fig,ax = plt.subplots(2,2,figsize = (20,17))
sns.histplot(x = df["ParkingArea"],ax=ax[0,0])
sns.countplot(x = df["Car"],ax=ax[0,1])
sns.countplot(x = df["Bathroom"],ax=ax[1,0])
sns.countplot(x = df["Rooms"],ax=ax[1,1])
```

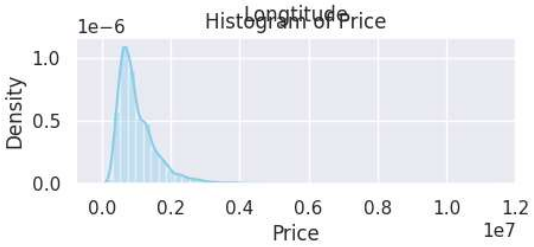
Out[18]:
```
<AxesSubplot:xlabel='Rooms', ylabel='count'>
```

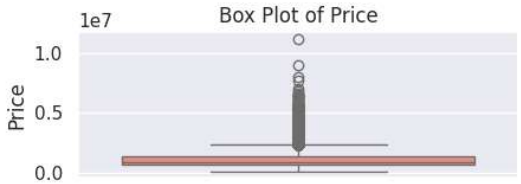


In [20]:
```
numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns

fig, axes = plt.subplots(nrows=len(numeric_columns), ncols=2, figsize=(12, 2 * len(
plt.subplots_adjust(wspace=0.5, hspace=0.5)

for i, col in enumerate(numeric_columns):
    # Histogram
    sns.distplot(df[col], kde=True, ax=axes[i, 0], color='skyblue')
    axes[i, 0].set_title('Histogram of ' + col)

    # Box plot
    sns.boxplot(df[col], ax=axes[i, 1], color='salmon')
    axes[i, 1].set_title('Box Plot of ' + col)


plt.show()
```



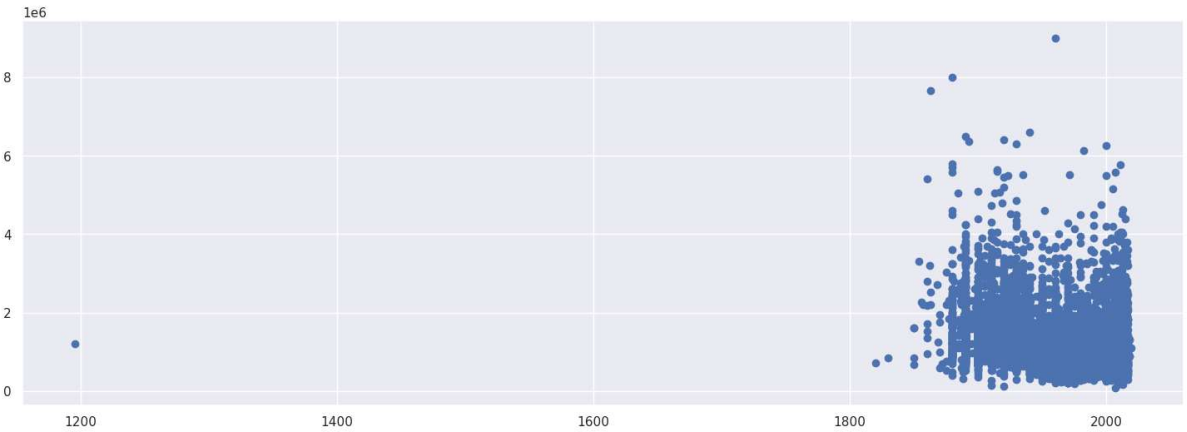






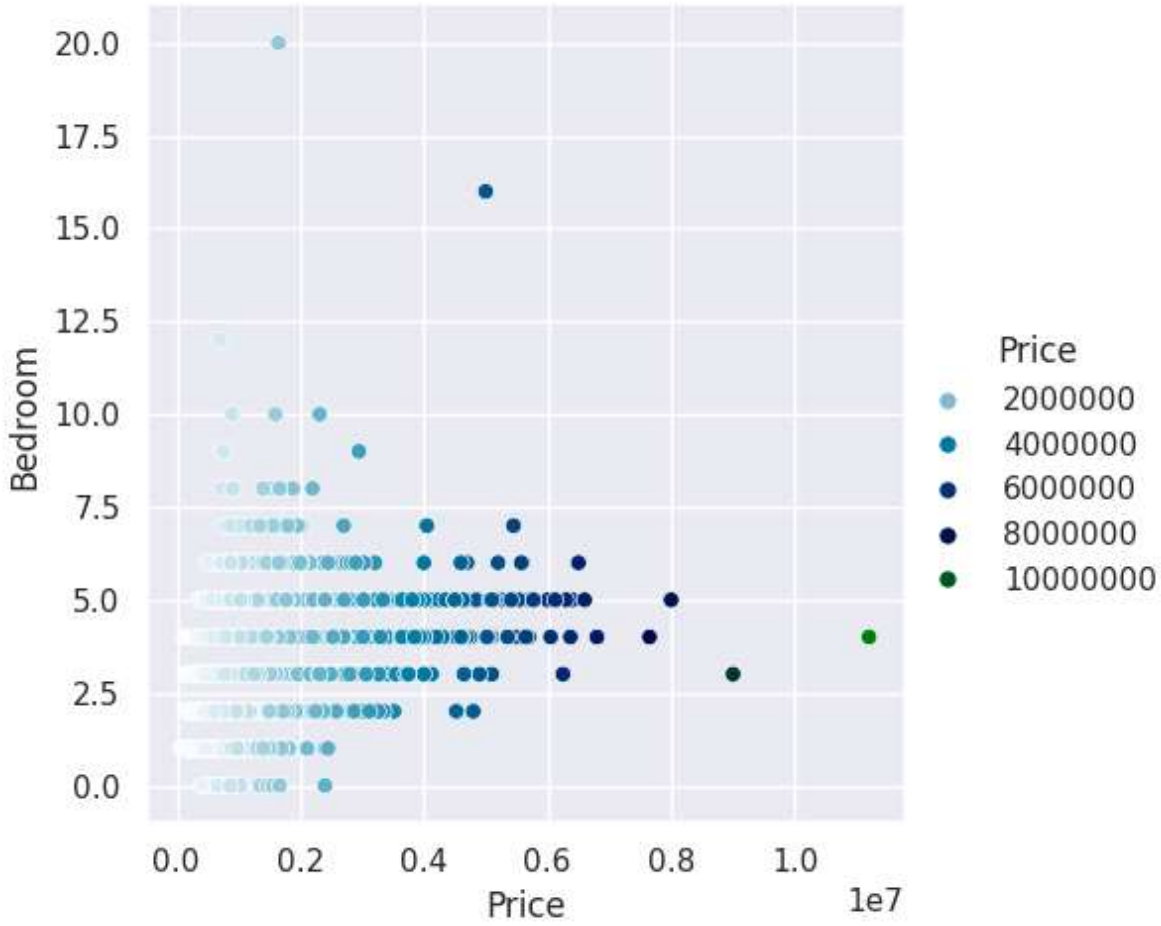
In [21]:
```
plt.figure(figsize=(18,6))
plt.scatter(x=df.YearBuilt ,y=df.Price)
```

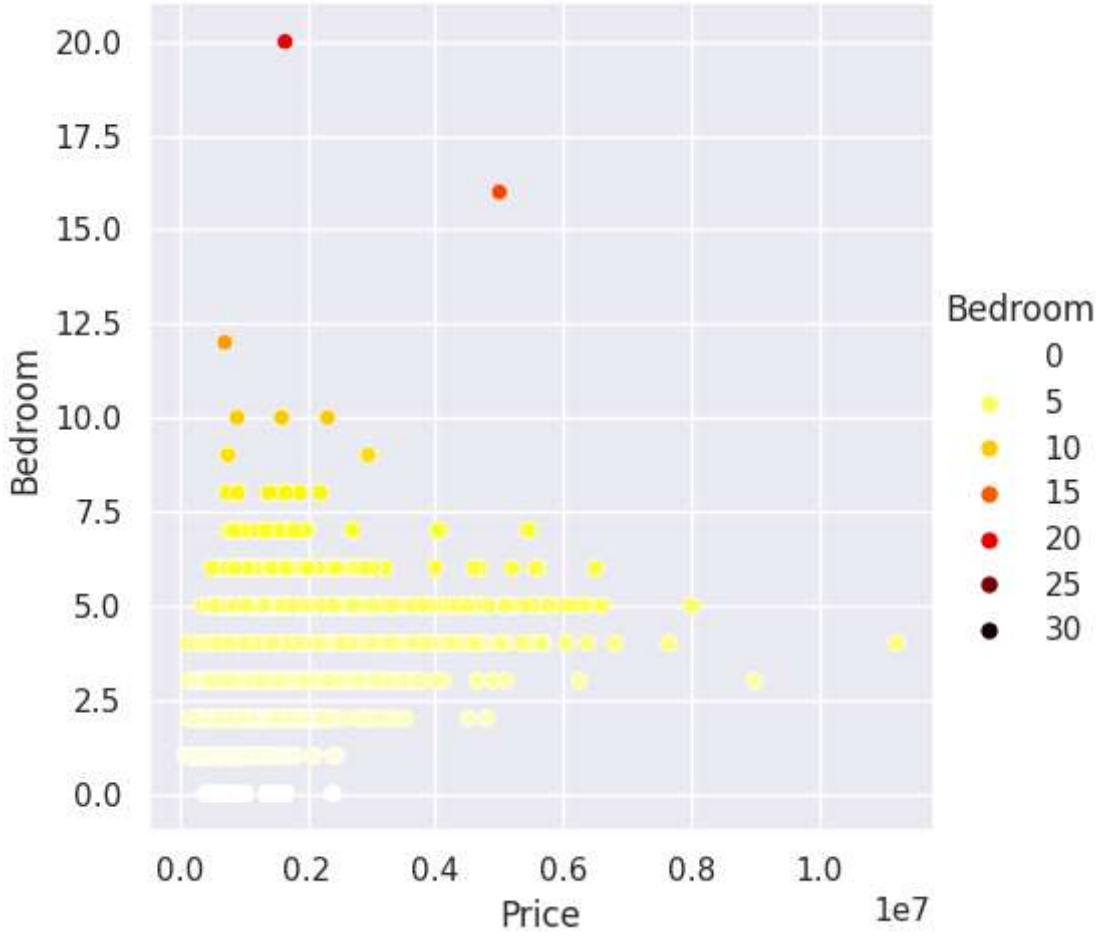
Out[21]:
```
<matplotlib.collections.PathCollection at 0x7514860>
```



In [24]:
```
sns.relplot(data = df,x = "Price"
            ,y = "Bedroom"
           ,hue = "Price",ax=ax[0],palette = "ocean_r")
sns.relplot(data = df,x = "Price"
            ,y = "Bedroom"
           ,hue = "Bedroom",ax=ax[1],palette = "hot_r")
```

Out[24]:
```
<seaborn.axisgrid.FacetGrid at 0x7131b10>
```





In [26]:
```
columns_to_fill_0 = ['Car', 'Bathroom', 'Bedroom', 'Distance', 'Propertycount']
df[columns_to_fill_0] = df[columns_to_fill_0].fillna(0)
df.isna().sum()
```

Out[26]:
```
Suburb              0
Address             0
Rooms               0
Type                0
Method              0
SellerG             0
Date                0
Distance            0
Postcode            1
Bedroom             0
Bathroom            0
Car                 0
Landsize        11810
BuildingArea    21097
YearBuilt       19306
CouncilArea         3
Latitude         7976
Longtitude       7976
Regionname          0
Propertycount       0
ParkingArea         0
Price            7610
dtype: int64
```

In [35]:
```
# Now lets fill the columns named landsize and building area with mean of the whole
df['Landsize'] = pd.to_numeric(df['Landsize'], errors='coerce') # the numbers were
df['BuildingArea'] = pd.to_numeric(df['BuildingArea'], errors='coerce')
df['Landsize'] = df['Landsize'].fillna(df.Landsize.mean())
df['BuildingArea'] = df['BuildingArea'].fillna(df.BuildingArea.mean())
df = df.replace([np.inf, -np.inf], np.nan).dropna()
# in the buildingArea column there are some infitly large valuea and the model was
# I had to come back here and drop those inf values as well.
```

In [37]:
```
df.isna().sum()
```

Out[37]:
```
Suburb          0
Address         0
Rooms           0
Type            0
Method          0
SellerG         0
Date            0
Distance        0
Postcode        0
Bedroom         0
Bathroom        0
Car             0
Landsize        0
BuildingArea    0
YearBuilt       0
CouncilArea     0
Latitude        0
Longtitude      0
Regionname      0
Propertycount   0
ParkingArea     0
Price           0
dtype: int64
```

In [38]:
```
 #with the help of min max scaler we are normalizing the value of distance and pric
#there is a huge difference between both of them in the output we can compare the v
from sklearn.preprocessing import MinMaxScaler
Scaling=MinMaxScaler()
```

```
normalised_values=Scaling.fit_transform(df[['Distance','Price']])
print(normalised_values)
```

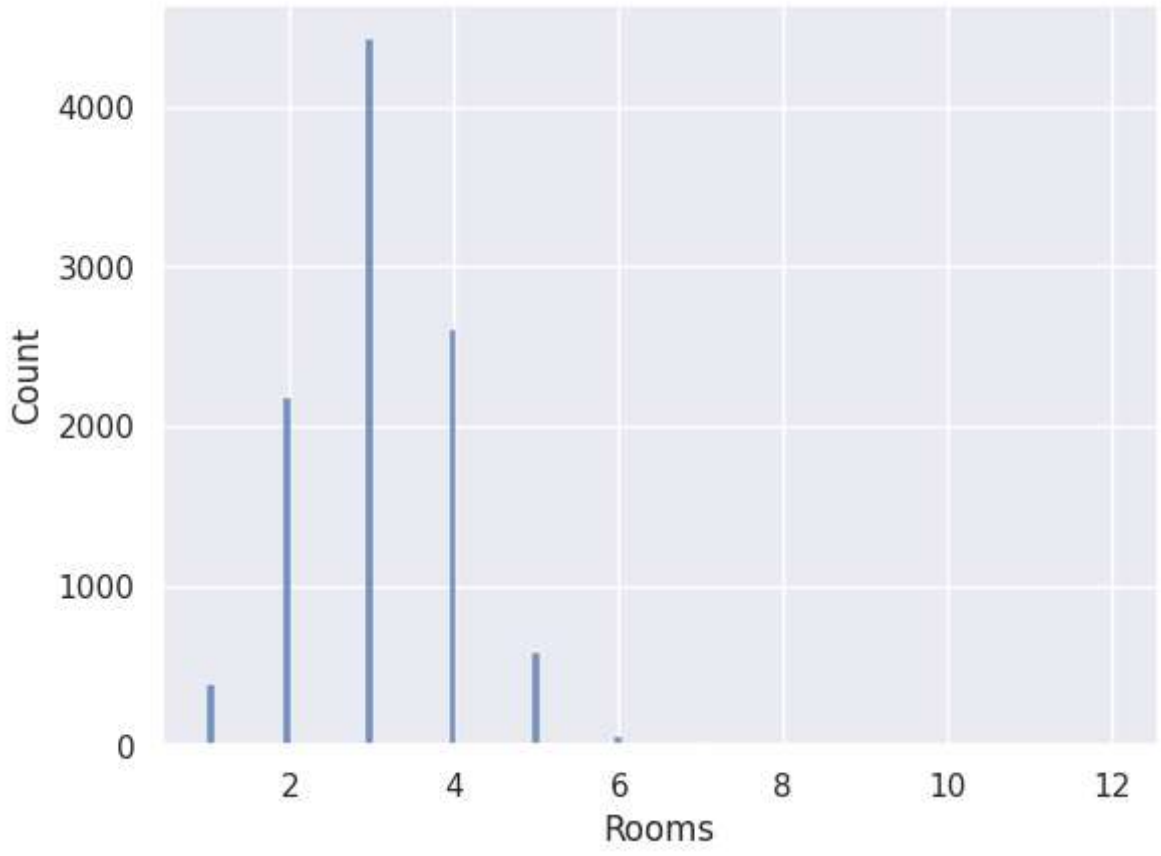
```
[[0.28066528 0.07994137]
 [0.06860707 0.12898861]
 [0.13305613 0.21073402]
 ...
 [0.24948025 0.03878679]
 [0.42827443 0.05186605]
 [0.34303534 0.07441651]]
```

In [39]:
```
df['Rooms'].describe()
```

Out[39]:
```
count    10241.000000
mean         3.107509
std          0.960684
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max         12.000000
Name: Rooms, dtype: float64
```

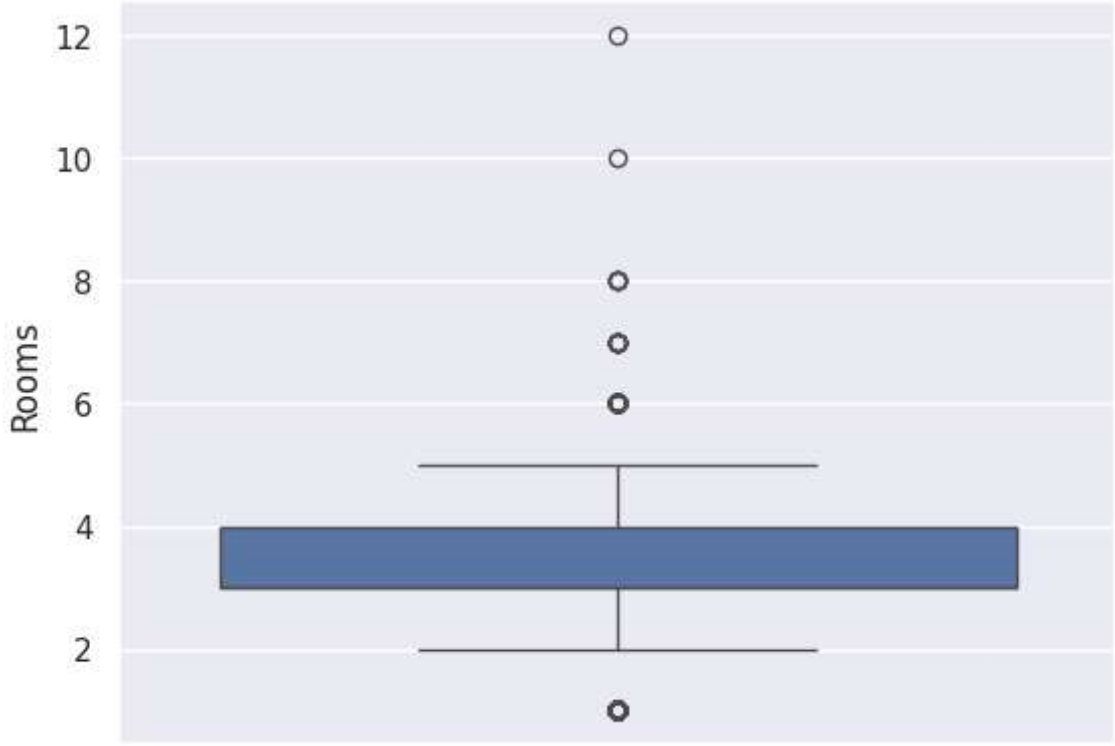
In [40]:
```
sns.histplot(df['Rooms']) # has normal curve
```

Out[40]:
```
<AxesSubplot:xlabel='Rooms', ylabel='Count'>
```



In [41]:
```
sns.boxplot(df['Rooms']) # in this we can see we have Outlier like 8 16 14
```

Out[41]:
```
<AxesSubplot:ylabel='Rooms'>
```



In [42]:
```
upper_limit = df['Rooms'].quantile(0.99)
upper_limit
```

Out[42]:
```
5.0
```

In [43]:
```
lower_limit = df['Rooms'].quantile(0.01)
lower_limit
```

Out[43]:
```
1.0
```

In [45]:
```
new_df = df[(df['Rooms'] <= 5.0) & (df['Rooms'] >=1.0)] #triming and creating new d
print(new_df['Rooms'])
```

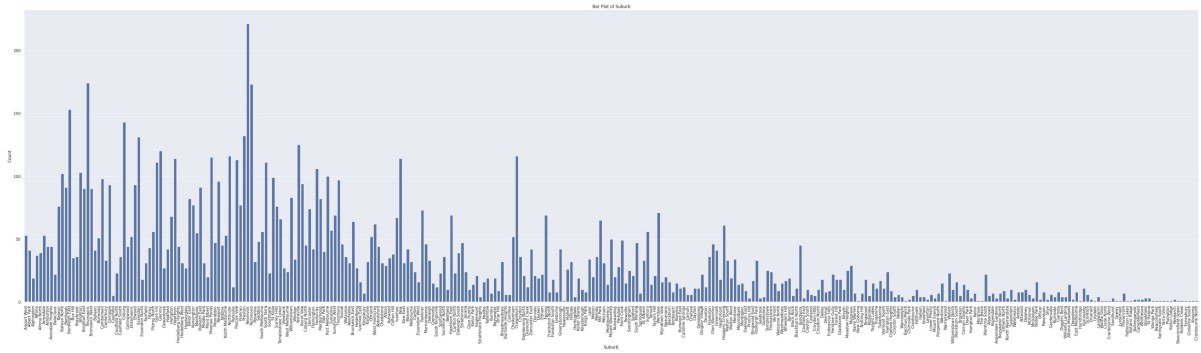
```
1        3
2        2
5        4
7        3
9        5
       ..
34848    2
34851    3
34852    3
34853    4
34856    4
Name: Rooms, Length: 10166, dtype: int64
```

In [46]:
```
sns.boxplot(new_df['Rooms'])# in this we can see there are none outliners
```

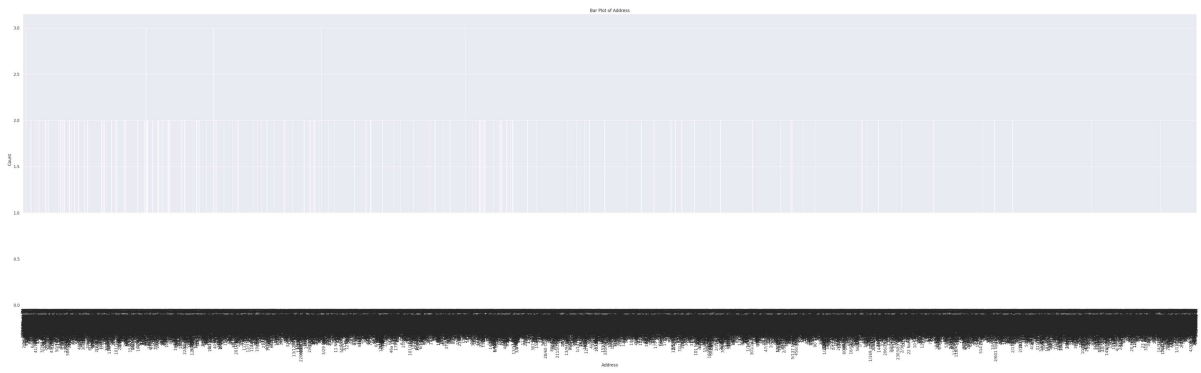
Out[46]:
```
<AxesSubplot:ylabel='Rooms'>
```

In [47]:
```
 # Select categorical variables
cat_values = df.select_dtypes(include=['object'])
print(cat_values)
for i in cat_values.columns:
    plt.figure(figsize=(60,15))
    print(i)
    sns.countplot(data=df, x=i)
    plt.xlabel(i)
    plt.ylabel('Count')
    plt.title(f'Bar Plot of {i}')
    plt.xticks(rotation=90)
    plt.show()
```

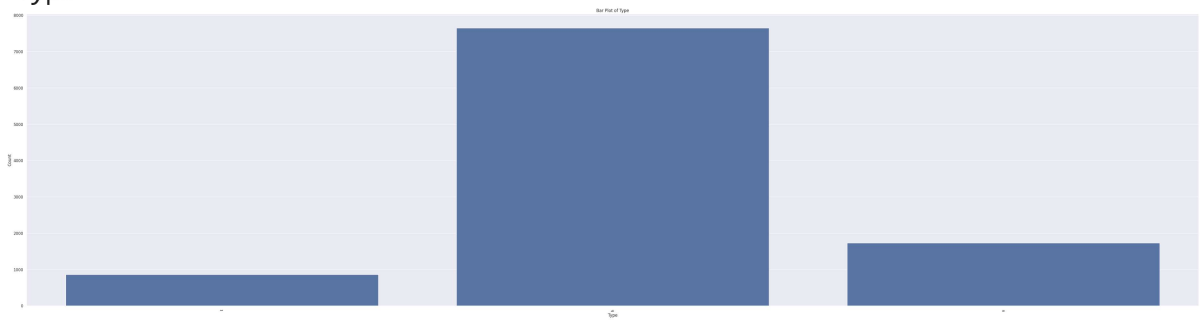
```
              Suburb         Address Type Method        SellerG        Date  \
1       Airport West   154 Halsey Rd    t     PI         Nelson    3/9/2016
2        Albert Park  105 Kerferd Rd    h      S  hockingstuart    3/9/2016
5         Alphington      6 Smith St    h      S          Brace    3/9/2016
7             Altona    158 Queen St    h     VB           Greg    3/9/2016
9       Altona North     45 Hearn St    h      S             FN    3/9/2016
...              ...             ...  ...    ...            ...         ...
34848      Maidstone  15 Ballarat Rd    h     SP         Biggin  30/09/2017
34851     Noble Park      5 Blaby St    h     PI            C21  30/09/2017
34852      Reservoir    18 Elinda Pl    u     SP             RW  30/09/2017
34853  Roxburgh Park  14 Stainsby Cr    h      S          Raine  30/09/2017
34856    Westmeadows    42 Pascoe St    h      S          Barry  30/09/2017

                        CouncilArea               Regionname  \
1          Moonee Valley City Council       Western Metropolitan
2           Port Phillip City Council      Southern Metropolitan
5                Darebin City Council      Northern Metropolitan
7             Hobsons Bay City Council       Western Metropolitan
9             Hobsons Bay City Council       Western Metropolitan
...                                ...                        ...
34848        Maribyrnong City Council       Western Metropolitan
34851  Greater Dandenong City Council  South-Eastern Metropolitan
34852            Darebin City Council      Northern Metropolitan
34853               Hume City Council      Northern Metropolitan
34856               Hume City Council      Northern Metropolitan

          ParkingArea
1      Detached Garage
2      Attached Garage
5          Underground
7          Parking Pad
9      Detached Garage
...                ...
34848          Carport
34851           Indoor
34852          Parkade
34853      Underground
34856  Attached Garage

[10241 rows x 9 columns]
Suburb
```



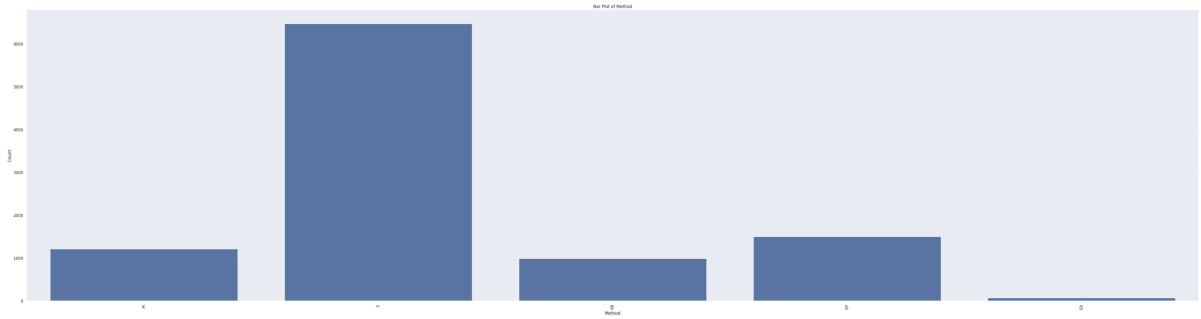
```
Address
```



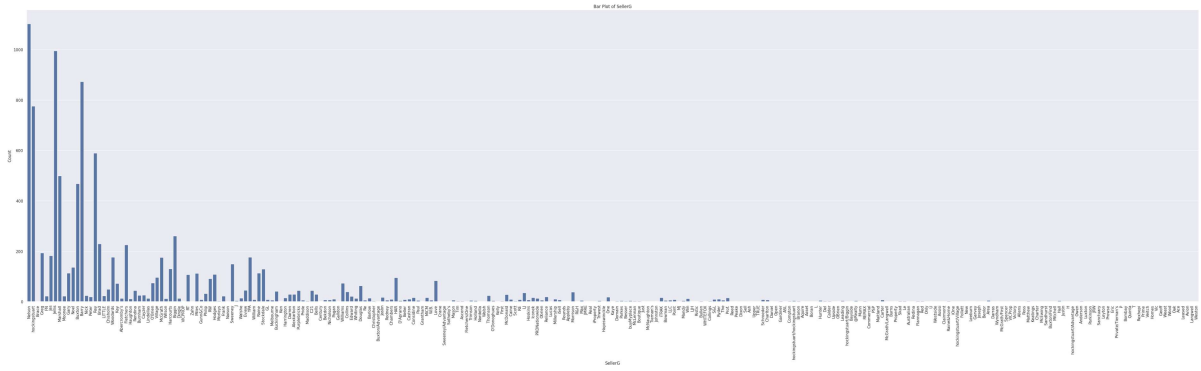
Type



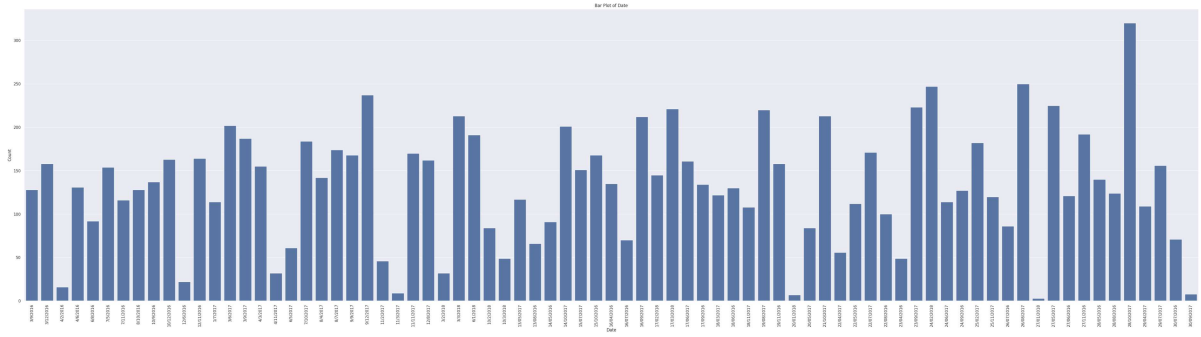
Method



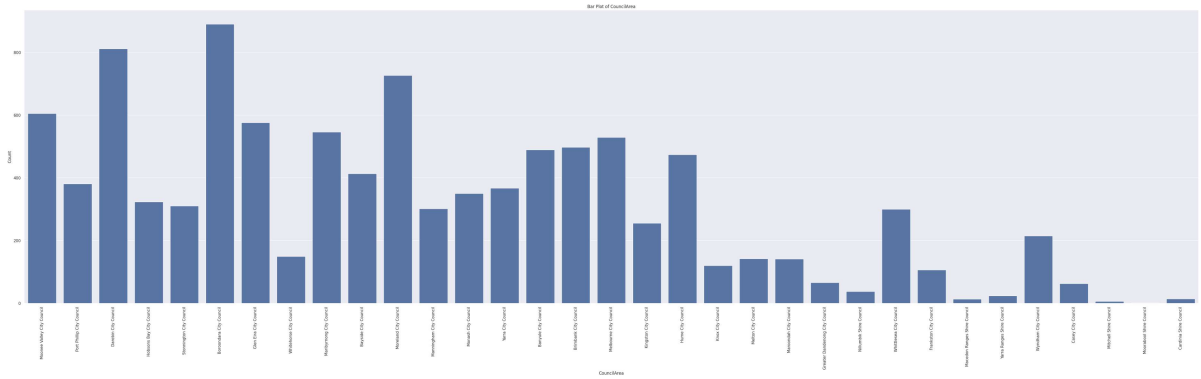
SellerG



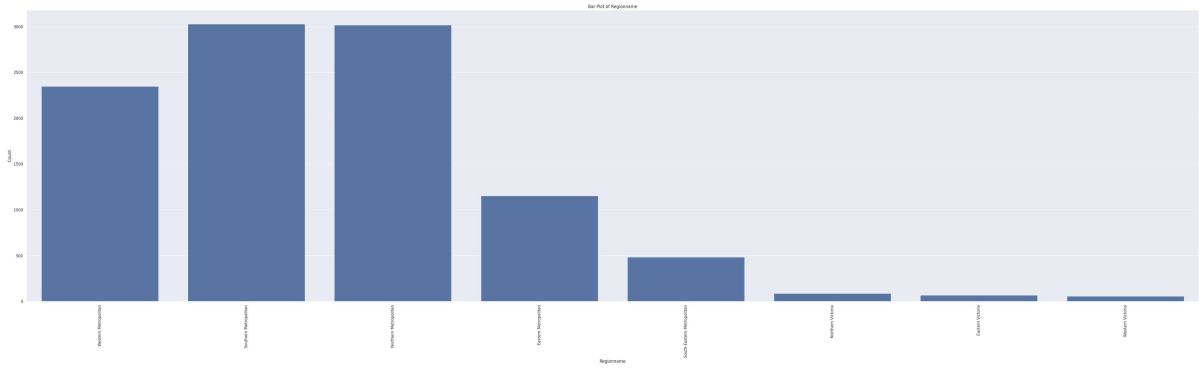
Date



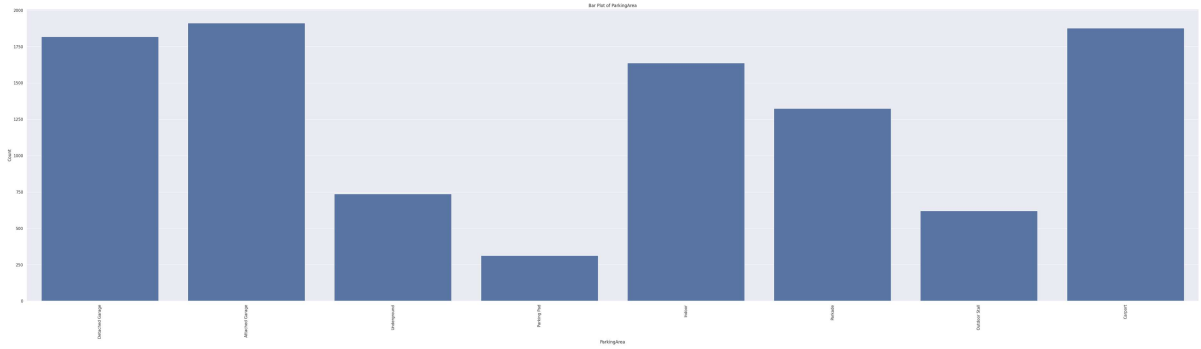
CouncilArea



Regionname



ParkingArea



Data Analysis Summary

The analysis of the dataset reveals critical insights into the distribution, characteristics, and relationships of its variables.

Distribution of Price: The examination of the 'Price' variable indicates a right-skewed pattern, signifying a concentration of lower-priced properties with notable outliers of considerably higher prices.

Numeric Variables: A thorough exploration of numeric attributes identifies potential outliers in the 'Room' and 'Price' variables. These outliers may significantly impact model performance and necessitate further scrutiny.

Categorical Variables: Categorical features like 'Rooms' and 'SellerG' were assessed using bar plots and frequency tables, offering valuable information about category distributions.

Correlation Analysis: The correlation analysis highlights a positive yet weak correlation between 'Room' and 'Price.' As the number of rooms increases, prices tend to rise, though the relationship is not strong.

Recommended Actions

Missing Data Imputation: To address missing values in 'Room' and 'Price,' consider imputation strategies. For 'Room,' mean or median imputation may suffice, while 'Price' may require more advanced methods like predictive imputation.

Outlier Handling: Devise an appropriate approach for managing outliers in 'Room' and 'Price.' Based on domain expertise and data characteristics, opt for outlier trimming or transformation.

Feature Engineering: Explore the creation of interaction or polynomial features, particularly for variables exhibiting non-linear associations with the target variable, such as 'Room' and 'Price.'

Model Development: Progress with machine learning model development, with a focus on robustness against outliers and accuracy in predicting both lower and higher-priced properties.

Further Analysis: Continue investigating other factors that potentially influence property prices, such as neighborhood attributes, property size, or additional variables not present in the current dataset.

Conclusion

In conclusion, the exploratory data analysis (EDA) and feature engineering stages have furnished invaluable insights for the dataset. Tackling missing data, managing outliers, and engineering pertinent features are pivotal steps toward constructing a resilient predictive model for property prices. Additionally, ongoing analysis should aim to uncover additional determinants of property prices, ensuring a comprehensive understanding of this complex domain.