Setup

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import Ridge, Lasso
from sklearn.model_selection import KFold

from xgboost import XGBRegressor
```

In [2]:
```
df = pd.read_csv('../data/tod-on-main.csv')
```

In [3]:
```
y1 = df["n_movers_out"]
y2 = df["n_movers_out_Lowincome"]
```

In [4]:
```
before_df = df.filter(regex=r'^(Before)', axis=1)
after_df = df.filter(regex=r'^(After)', axis=1)
```

In [5]:
```
def calculating_average_bedrooms(df, before = True):
    word = "Before" if before else "After"
    null_rows = df[df[f"{word} Average number of bedrooms per dwelling"].isr
    weighted_sum = (0.5 * null_rows[f"{word} 0 to 1 bedroom"] + 2 * null_row
    total = (null_rows[[f"{word} 0 to 1 bedroom", f"{word} 2 bedrooms", f"{w
    null_rows[f"{word} Average number of bedrooms per dwelling"] = weighted_
    df.loc[null_rows.index, f"{word} Average number of bedrooms per dwelling
```

In [6]:
```
calculating_average_bedrooms(before_df)
calculating_average_bedrooms(after_df, False)
```

```
/var/folders/qd/w0jg42cx76sc6s81d9zfb7t80000gn/T/ipykernel_40568/1411688805.
py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  null_rows[f"{word} Average number of bedrooms per dwelling"] = weighted_su
m / total
/var/folders/qd/w0jg42cx76sc6s81d9zfb7t80000gn/T/ipykernel_40568/1411688805.
py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.loc[null_rows.index, f"{word} Average number of bedrooms per dwelling"]
= null_rows[f"{word} Average number of bedrooms per dwelling"]
/var/folders/qd/w0jg42cx76sc6s81d9zfb7t80000gn/T/ipykernel_40568/1411688805.
py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  null_rows[f"{word} Average number of bedrooms per dwelling"] = weighted_su
m / total
/var/folders/qd/w0jg42cx76sc6s81d9zfb7t80000gn/T/ipykernel_40568/1411688805.
py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.loc[null_rows.index, f"{word} Average number of bedrooms per dwelling"]
= null_rows[f"{word} Average number of bedrooms per dwelling"]
```

In [7]:
```
# Drop all the individual bedroom columns

cols_to_drop = before_df.filter(regex='bedrooms?$', axis=1).columns
before_df = before_df.drop(columns = cols_to_drop)
cols_to_drop = after_df.filter(regex='bedrooms?$', axis=1).columns
after_df = after_df.drop(columns = cols_to_drop)
```

In [8]:
```
# Removing values that either introduce collinearity or have many missing va

before_df = before_df.drop(["Before Apartment", "Before Other dwelling", "Be
after_df = after_df.drop(["After Other dwelling", "After Other attached dwel
```

Dwelling Characteristics Before and After($X_{D,t=A}$ and $X_{D,t=B}$)

Basic Information

In [9]:
```
print("Shape is", before_df.shape)
print(before_df.info())
```

```
Shape is (70, 13)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 13 columns):
 #   Column                                                Non-Null C
ount  Dtype
---  ------                                                ----------
----  -----
 0   Before Population Density per square kilometre        70 non-nul
l     float64
 1   Before Dwellings                                      42 non-nul
l     float64
 2   Before Total Occupied Private Dwellings               70 non-nul
l     float64
 3   Before Single-detached house                          70 non-nul
l     float64
 4   Before Semi-detached house                            68 non-nul
l     float64
 5   Before Row house                                      68 non-nul
l     float64
 6   Before Apartment, duplex                              68 non-nul
l     float64
 7   Before Apartment, building that has fewer than five storeys  67 non-nul
l     float64
 8   Before Apartment, building that has five or more storeys     69 non-nul
l     float64
 9   Before Movable dwelling                               70 non-nul
l     float64
 10  Before Average number of bedrooms per dwelling        67 non-nul
l     float64
 11  Before Owned                                          70 non-nul
l     float64
 12  Before Rented                                         70 non-nul
l     float64
dtypes: float64(13)
memory usage: 7.2 KB
None
```

In [10]:
```
print("Shape is", after_df.shape)
print(after_df.info())
```

```
Shape is (70, 13)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 13 columns):
 #   Column                                                Non-Null Co
unt  Dtype
---  ------                                                ----------
---  -----
 0   After Population Density per square kilometre         70 non-null
float64
 1   After Dwellings                                       67 non-null
float64
 2   After Total Occupied Private Dwellings                70 non-null
float64
 3   After Single-detached house                           70 non-null
float64
 4   After Semi-detached house                             69 non-null
float64
 5   After Row house                                       69 non-null
float64
 6   After Apartment, duplex                               69 non-null
float64
 7   After Apartment, building that has fewer than five storeys  69 non-null
float64
 8   After Apartment, building that has five or more storeys     70 non-null
float64
 9   After Movable dwelling                                70 non-null
float64
 10  After Average number of bedrooms per dwelling         69 non-null
float64
 11  After Owned                                           70 non-null
float64
 12  After Rented                                          70 non-null
float64
dtypes: float64(13)
memory usage: 7.2 KB
None
```

Descriptive Statistics

In [11]:
```
before_df.describe()
```

Out[11]:

	Before Population Density per square kilometre	Before Dwellings	Before Total Occupied Private Dwellings	Before Single-detached house	Before Semi-detached house	B Row h
count	70.000000	42.000000	70.000000	70.000000	68.000000	68.0(
mean	57324.865634	3098.206985	2377.092768	631.330071	106.456011	122.8
std	113216.390124	3194.219870	2443.470807	475.698510	289.957670	116.3(
min	11.416007	25.342501	23.201693	5.000000	0.000000	0.0(
25%	1466.122390	1312.711342	753.392018	291.053291	11.674445	46.9(
50%	12583.949928	2102.862406	1651.476544	507.219473	36.670702	83.1
75%	49266.737562	3958.129184	2363.522913	913.379267	77.453149	171.4(
max	646168.395259	15586.510605	13367.434463	2680.843792	2184.042565	530.0(

In [12]:
```
after_df.describe()
```

Out[12]:

	After Population Density per square kilometre	After Dwellings	After Total Occupied Private Dwellings	After Single-detached house	After Semi-detached house	After h
count	70.000000	67.000000	70.000000	70.000000	69.000000	69.00(
mean	83624.500274	3454.156110	3071.057221	549.088128	100.673967	165.75(
std	142428.398970	3622.260092	3172.590281	395.692437	186.987633	147.98
min	43.457565	120.044240	114.496750	0.000000	0.000000	0.00(
25%	14117.821398	1364.931211	1330.202504	199.576745	11.158394	63.85
50%	36723.365589	2047.696021	1897.681001	487.073987	37.753862	138.4(
75%	72629.650488	4011.115867	3434.903365	772.408622	108.807448	202.67
max	908077.330189	20128.428244	18201.958223	1806.899375	1049.457765	729.97

Correlations

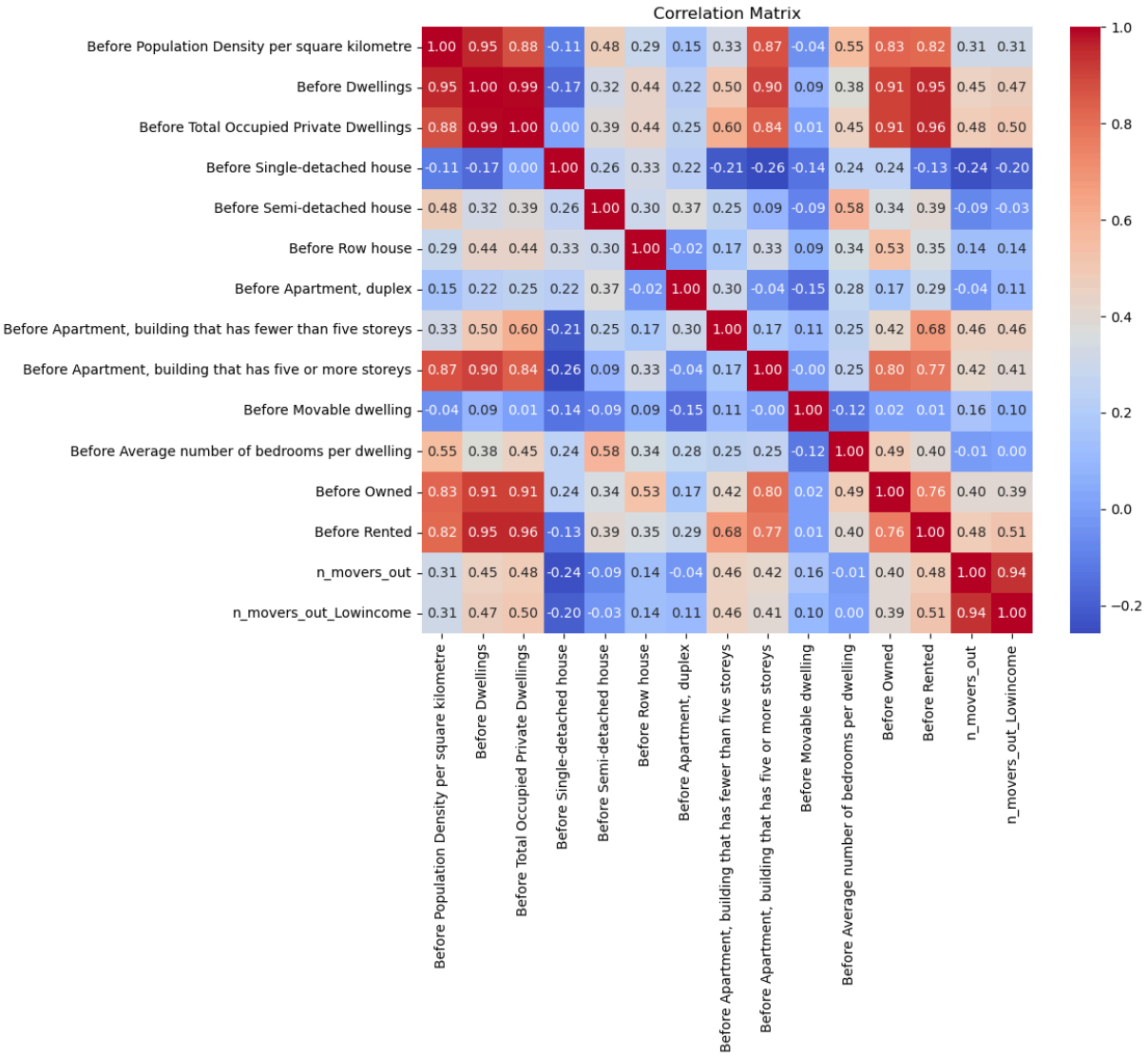
In [13]:
```
temp_before_df = before_df.copy()
temp_after_df = after_df.copy()
```

In [14]:
```
temp_before_df["n_movers_out"] = y1
temp_before_df["n_movers_out_Lowincome"] = y2
```

```
temp_after_df["n_movers_out"] = y1
temp_after_df["n_movers_out_Lowincome"] = y2
```

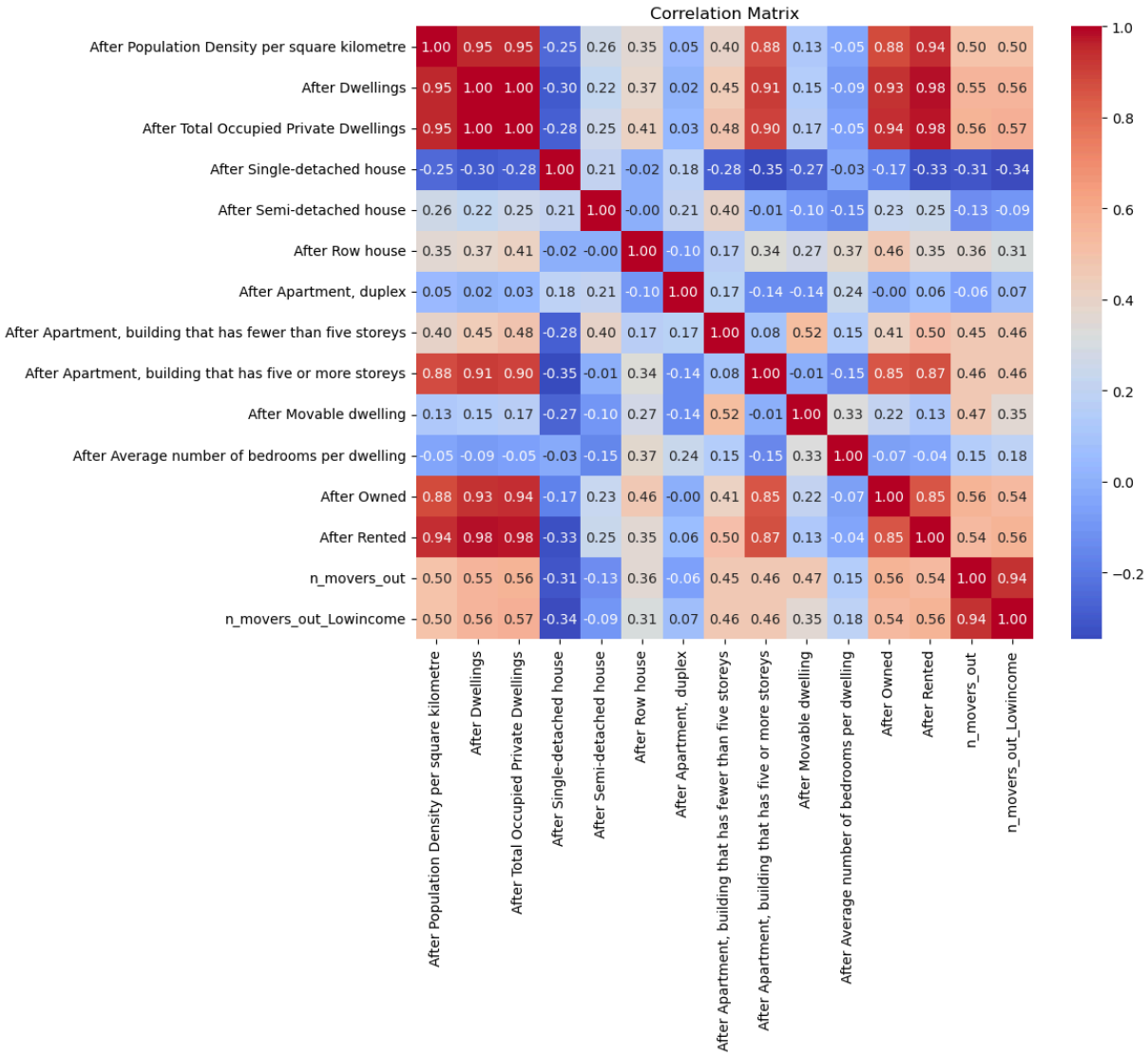
In [15]:
```
b_corr = temp_before_df.corr()
a_corr = temp_after_df.corr()
```

In [16]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(b_corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



In [ ]:
```
```

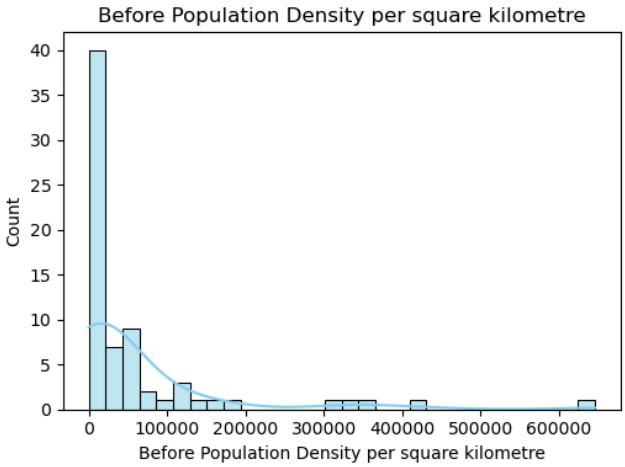
In [17]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(a_corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

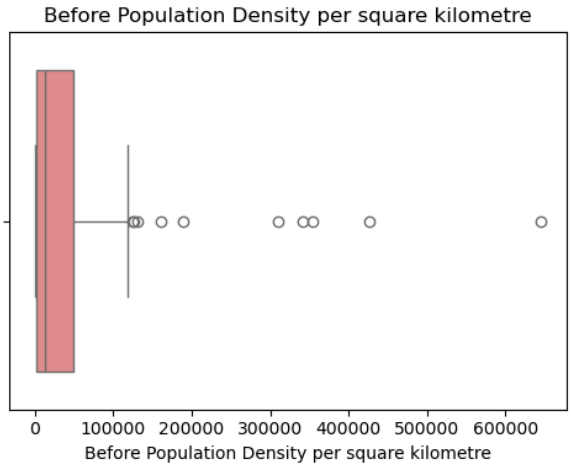


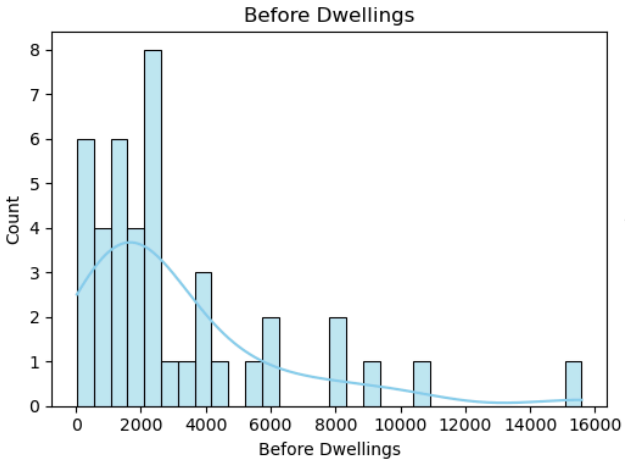
Visualizations

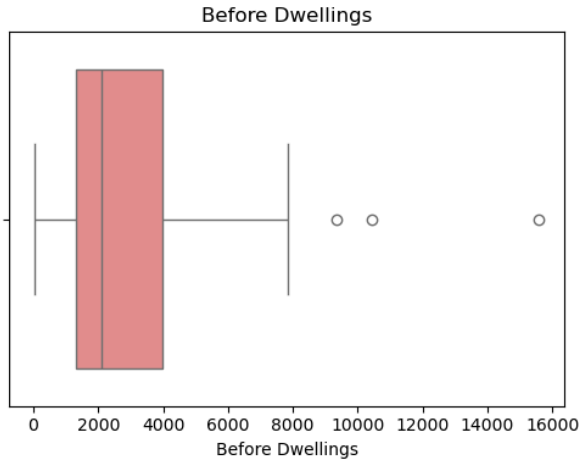
In [18]:

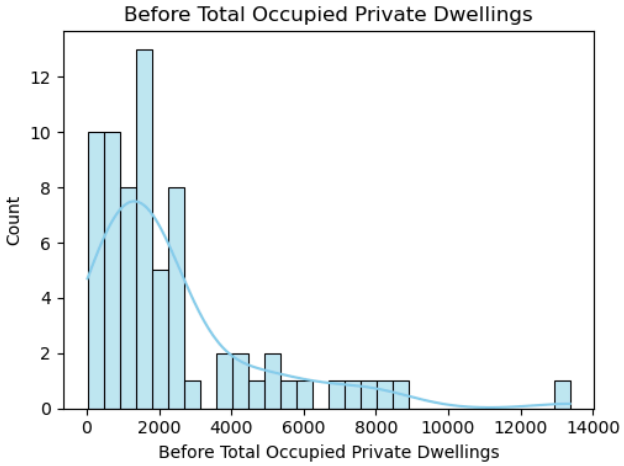
```
for col in before_df.columns:
    plt.figure(figsize=(10, 4))

    # Histogram
    plt.subplot(1, 2, 1)
    sns.histplot(before_df[col], kde=True, bins=30, color='skyblue')
    plt.title(f'{col}')
    plt.xlabel(col)

    # Boxplot
    plt.subplot(1, 2, 2)
    sns.boxplot(x=before_df[col], color='lightcoral')
    plt.title(f'{col}')
    plt.xlabel(col)

    plt.tight_layout()
    plt.show()
```

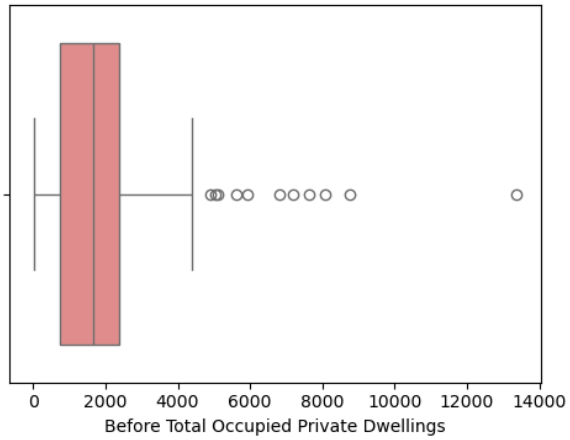

Before Population Density per square kilometre
Count
Before Population Density per square kilometre

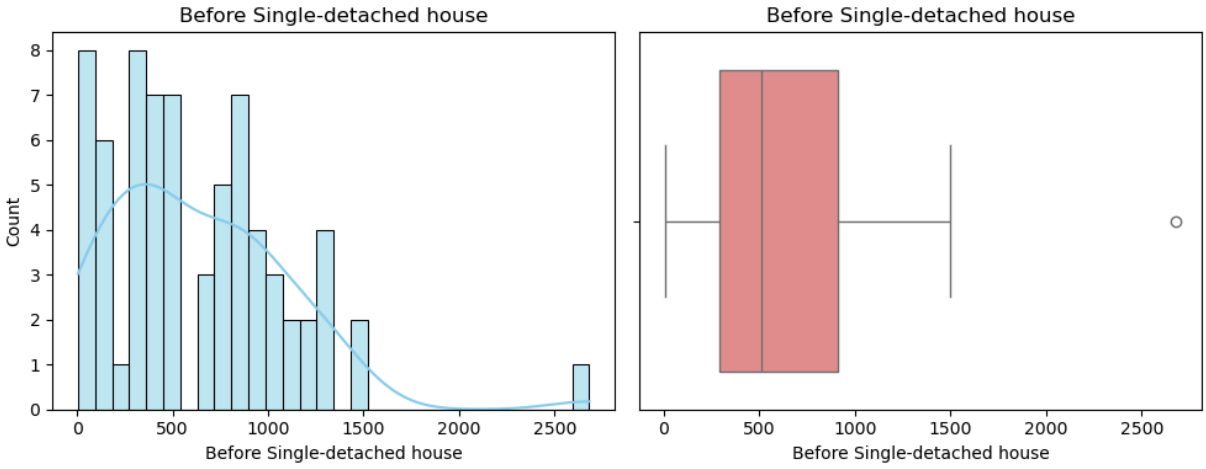

Before Population Density per square kilometre
Before Population Density per square kilometre


Before Dwellings
Count
Before Dwellings

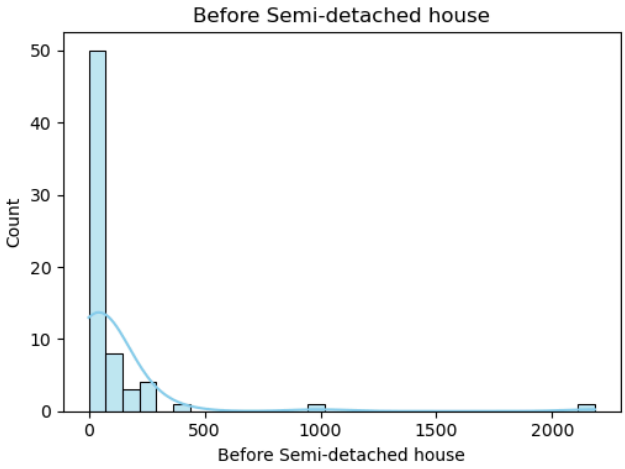

Before Dwellings
Before Dwellings

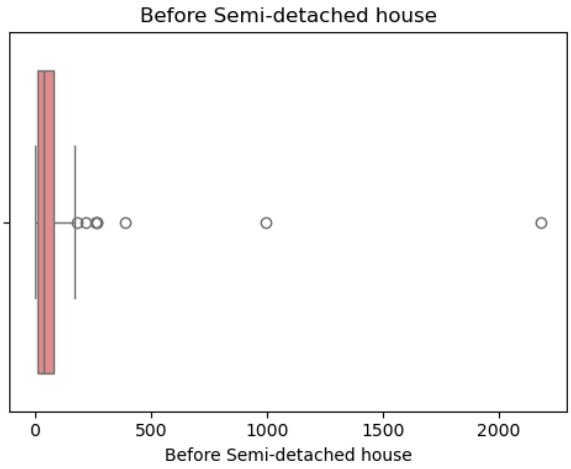

Before Total Occupied Private Dwellings
Count
Before Total Occupied Private Dwellings

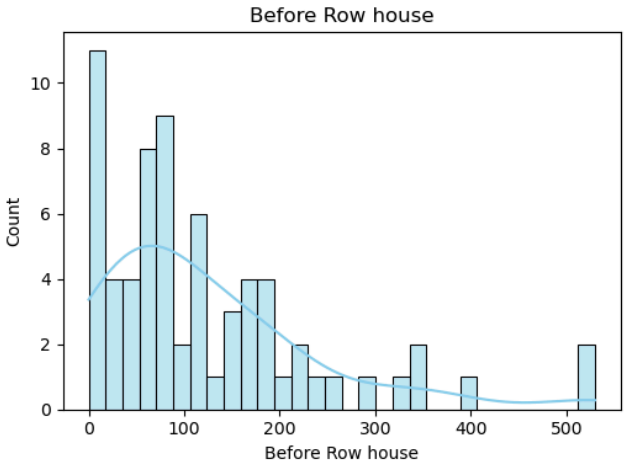

Before Total Occupied Private Dwellings
Before Total Occupied Private Dwellings

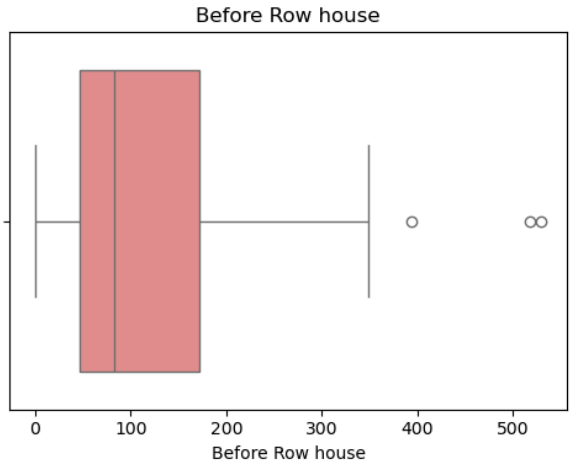

Before Single-detached house
Count
Before Single-detached house

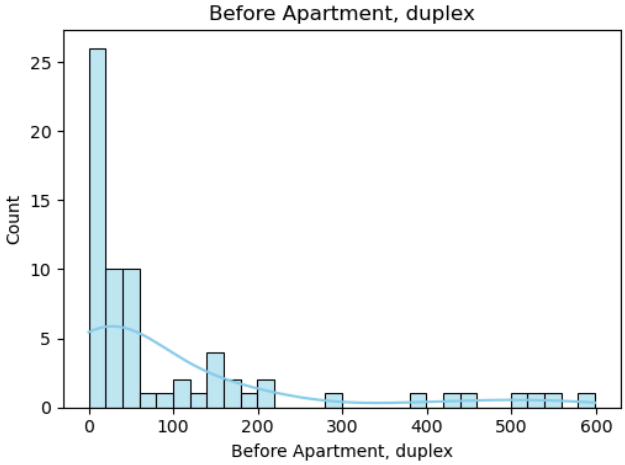
Before Single-detached house
Before Single-detached house

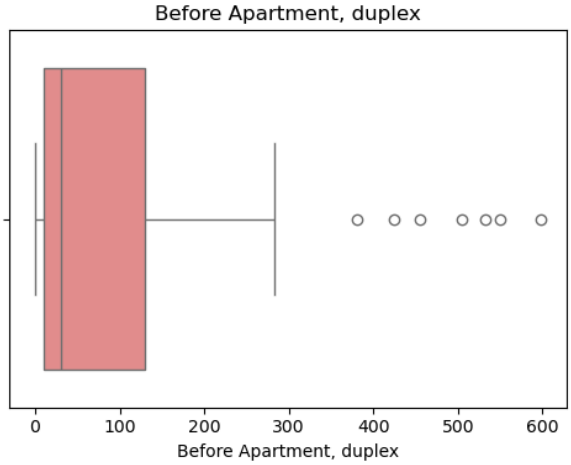

Before Semi-detached house
Count
Before Semi-detached house

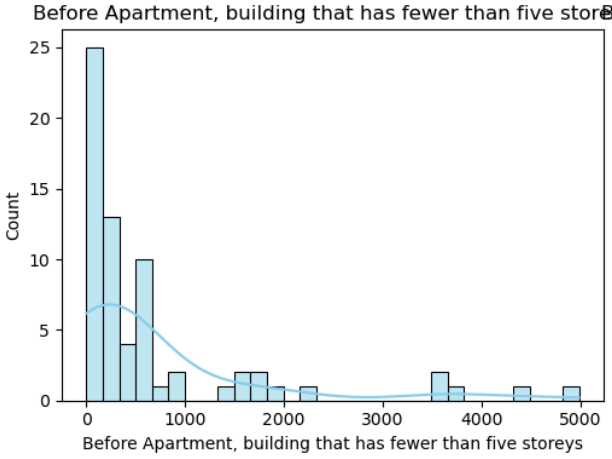

Before Semi-detached house
Before Semi-detached house

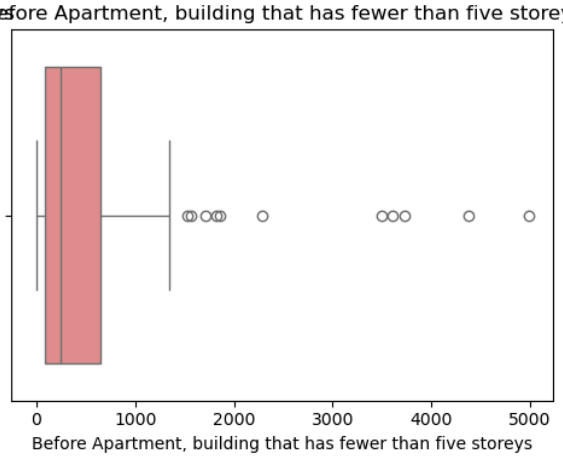

Before Row house
Count
Before Row house

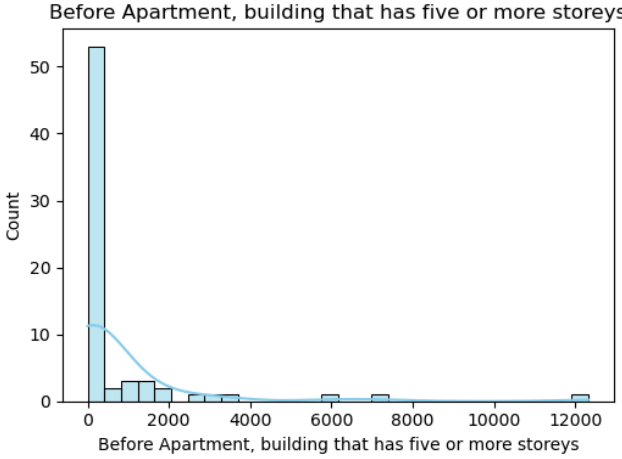

Before Row house
Before Row house

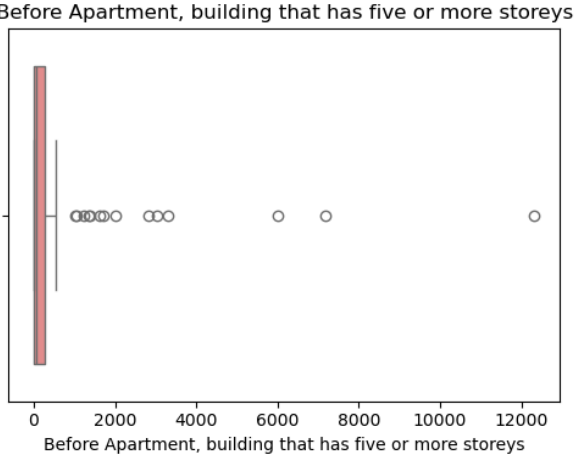

Before Apartment, duplex
Count
Before Apartment, duplex

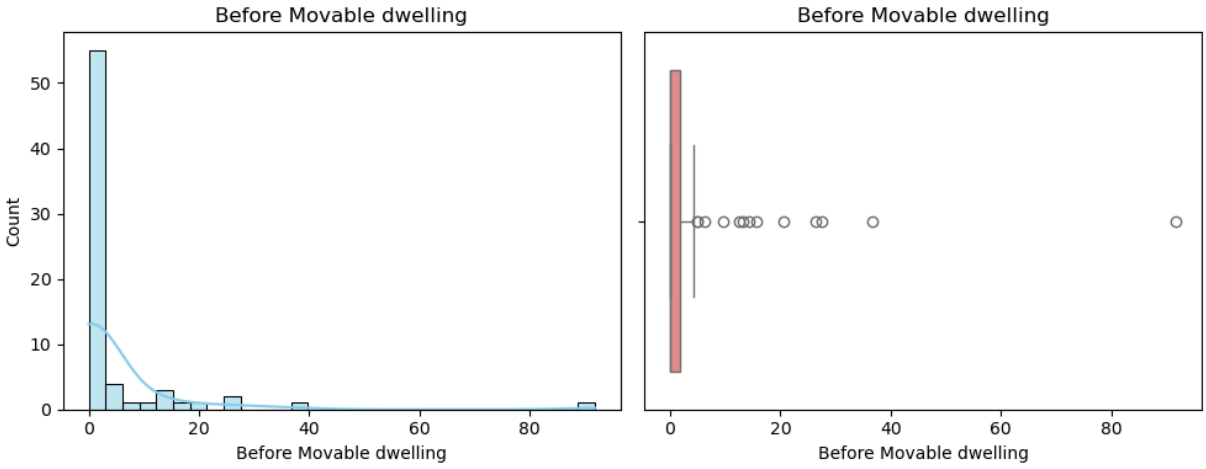

Before Apartment, duplex
Before Apartment, duplex


Before Apartment, building that has fewer than five storeys
Count
Before Apartment, building that has fewer than five storeys

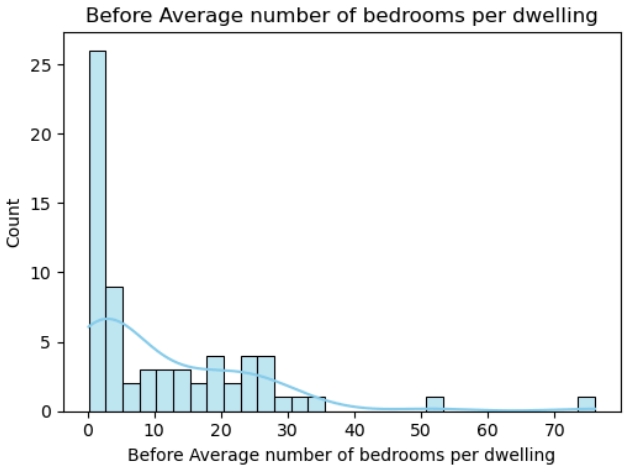

Before Apartment, building that has fewer than five storeys
Before Apartment, building that has fewer than five storeys

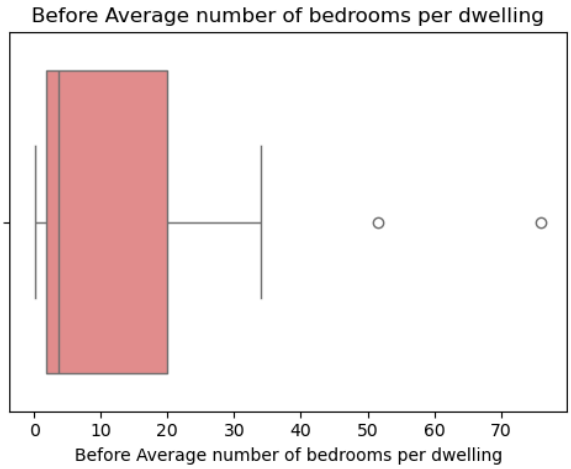

Before Apartment, building that has five or more storeys
Count
Before Apartment, building that has five or more storeys

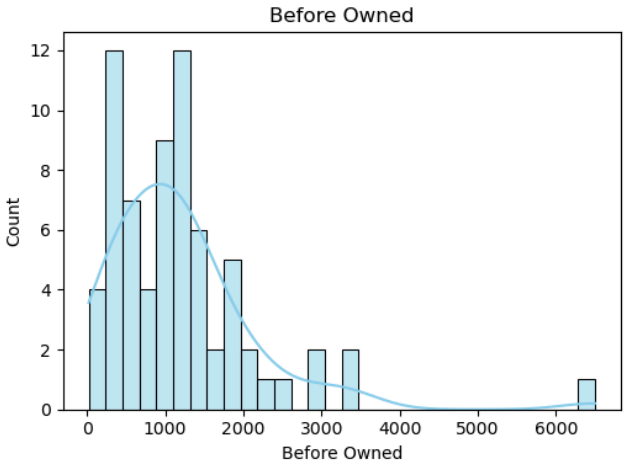

Before Apartment, building that has five or more storeys
Before Apartment, building that has five or more storeys

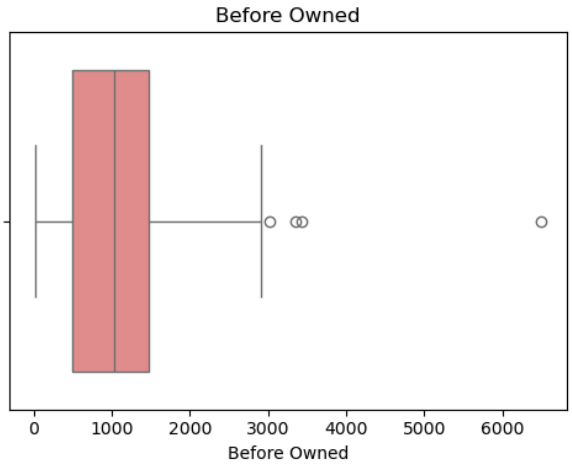

Before Movable dwelling
Count
Before Movable dwelling

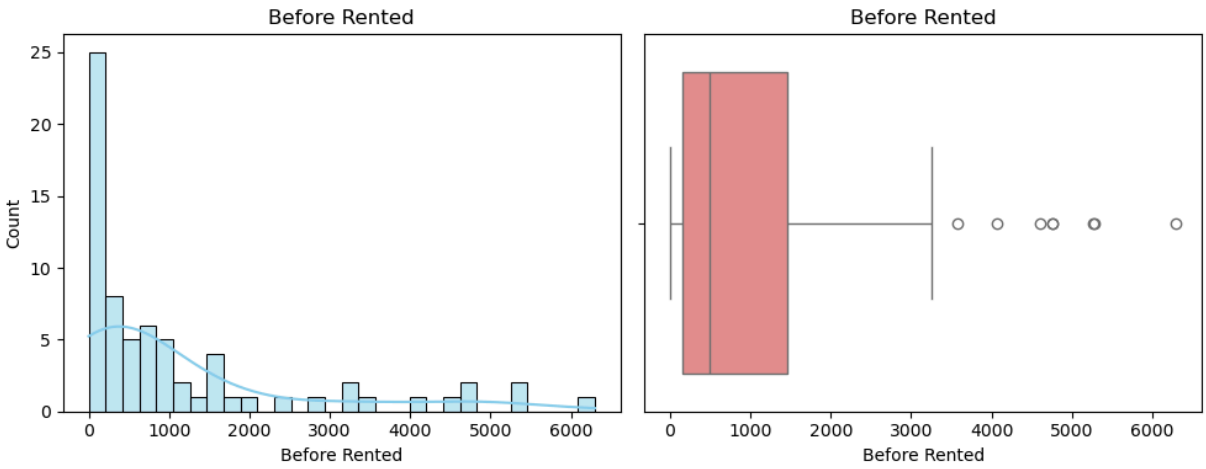
Before Movable dwelling
Before Movable dwelling


Before Average number of bedrooms per dwelling
Count
Before Average number of bedrooms per dwelling

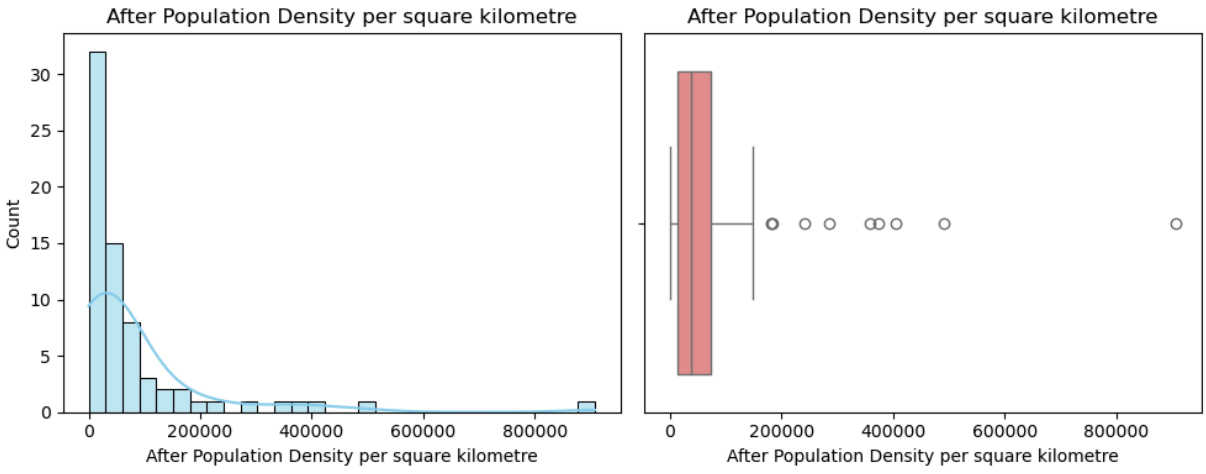

Before Average number of bedrooms per dwelling
Before Average number of bedrooms per dwelling

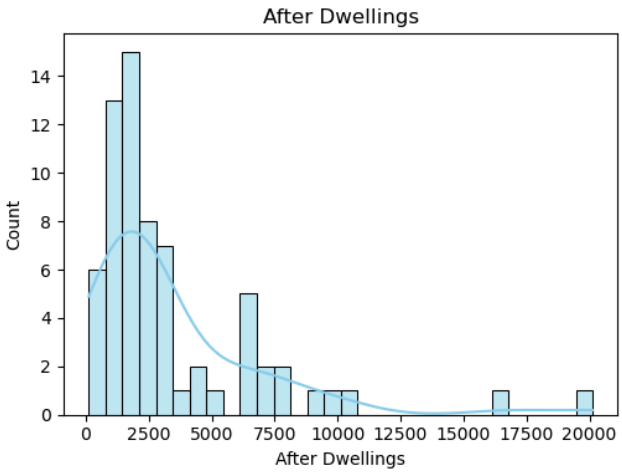

Before Owned
Count
Before Owned

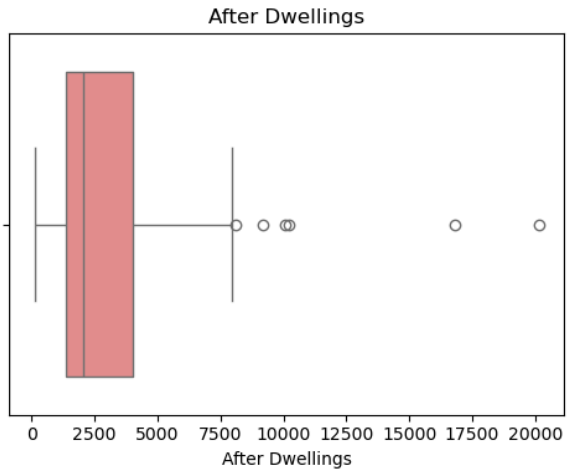

Before Owned
Before Owned

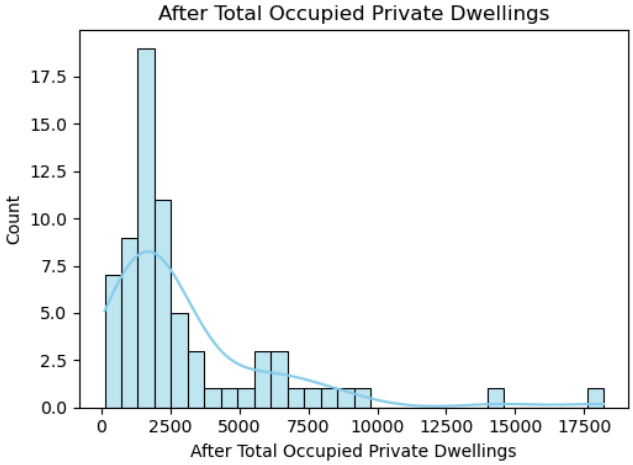


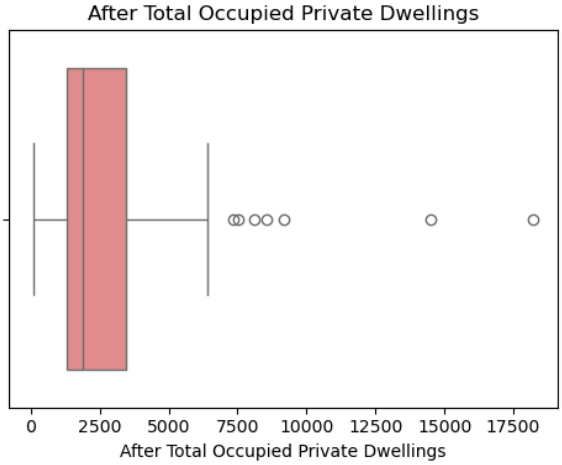
In [19]:
```python
for col in after_df.columns:
    plt.figure(figsize=(10, 4))

    # Histogram
    plt.subplot(1, 2, 1)
    sns.histplot(after_df[col], kde=True, bins=30, color='skyblue')
    plt.title(f'{col}')
    plt.xlabel(col)

    # Boxplot
    plt.subplot(1, 2, 2)
    sns.boxplot(x=after_df[col], color='lightcoral')
    plt.title(f'{col}')
    plt.xlabel(col)

    plt.tight_layout()
    plt.show()
```

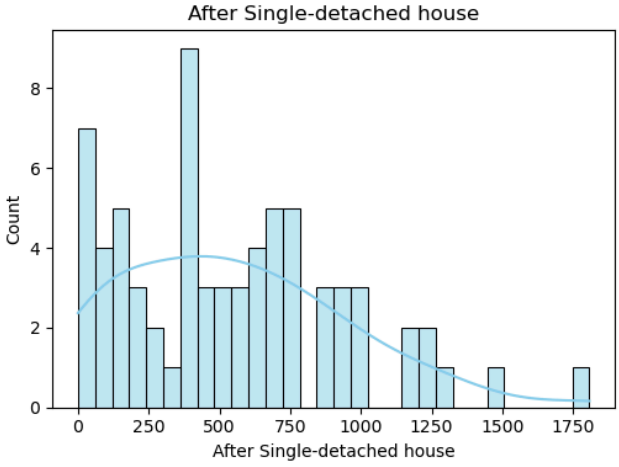


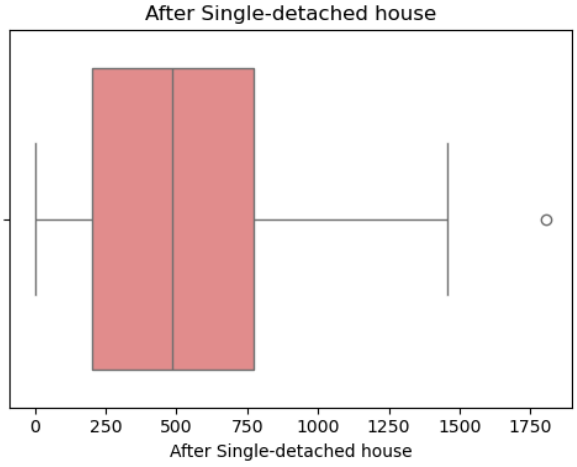

After Dwellings
Count
After Dwellings

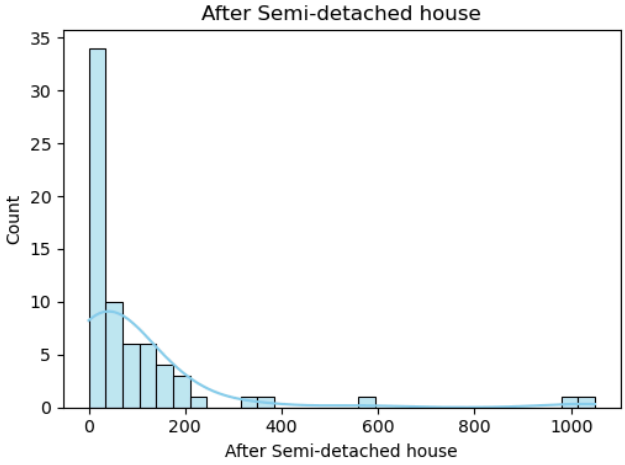

After Dwellings
After Dwellings

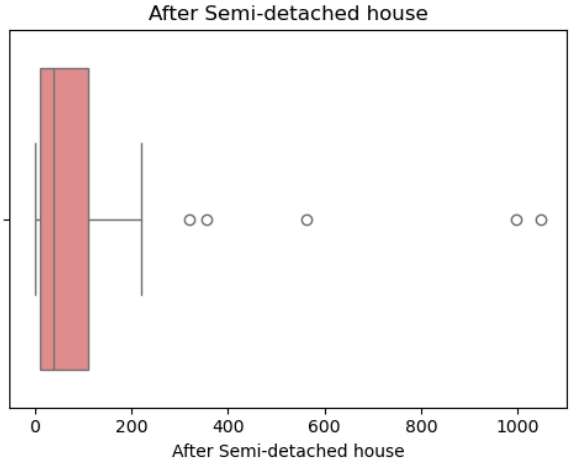

After Total Occupied Private Dwellings
Count
After Total Occupied Private Dwellings

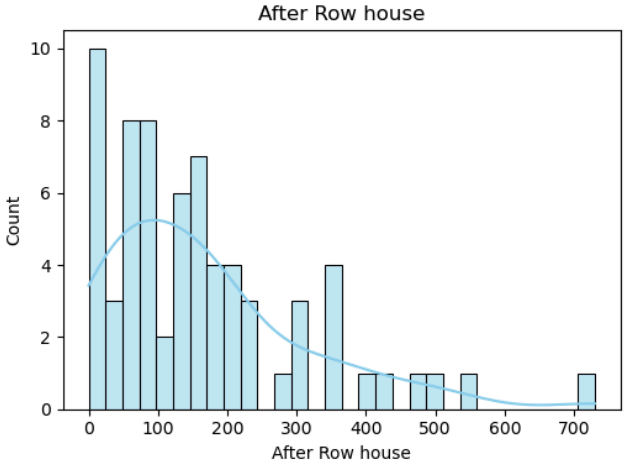

After Total Occupied Private Dwellings
After Total Occupied Private Dwellings

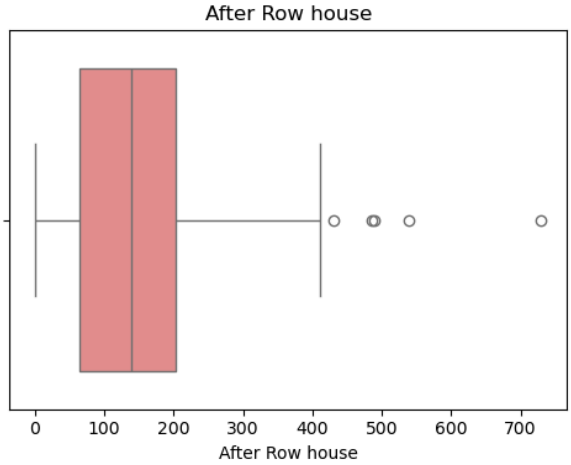

After Single-detached house
Count
After Single-detached house

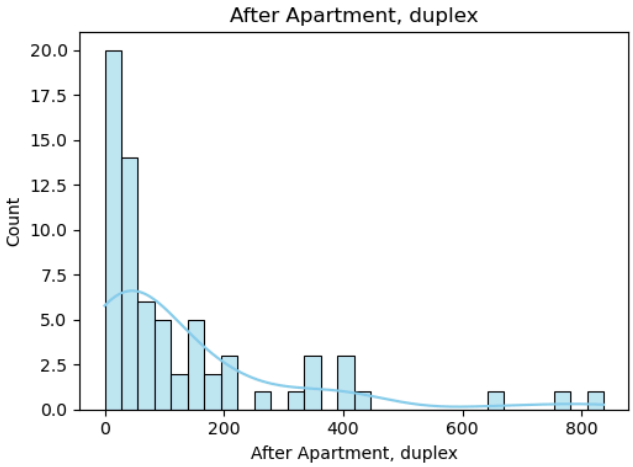

After Single-detached house
After Single-detached house

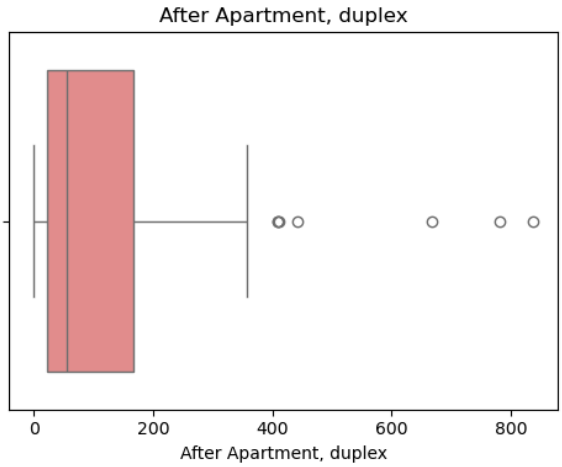

After Semi-detached house
Count
After Semi-detached house

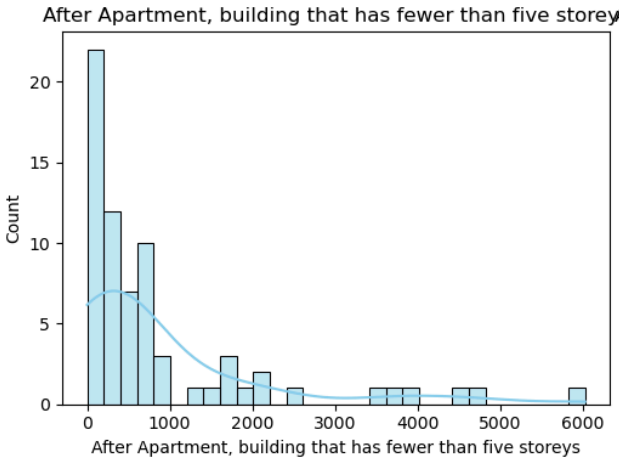

After Semi-detached house
After Semi-detached house

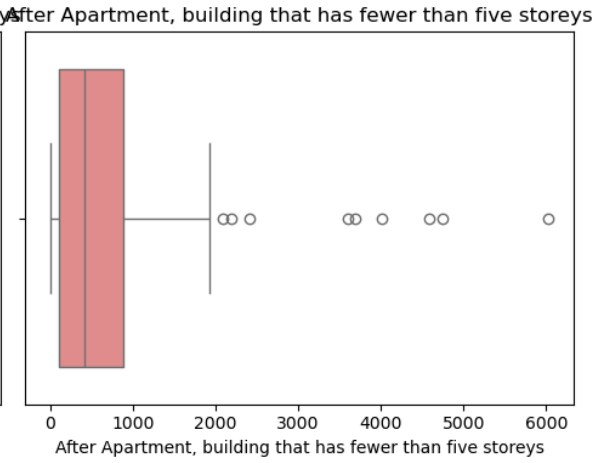

After Row house
Count
After Row house

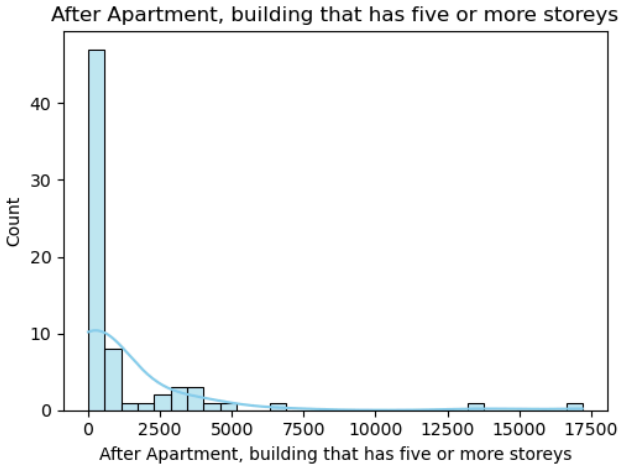

After Row house
After Row house

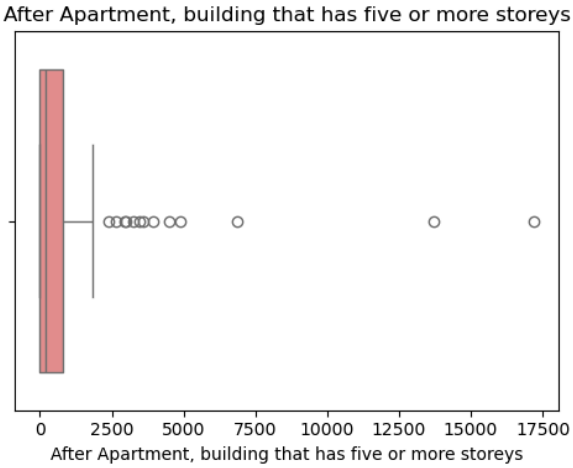

After Apartment, duplex
Count
After Apartment, duplex

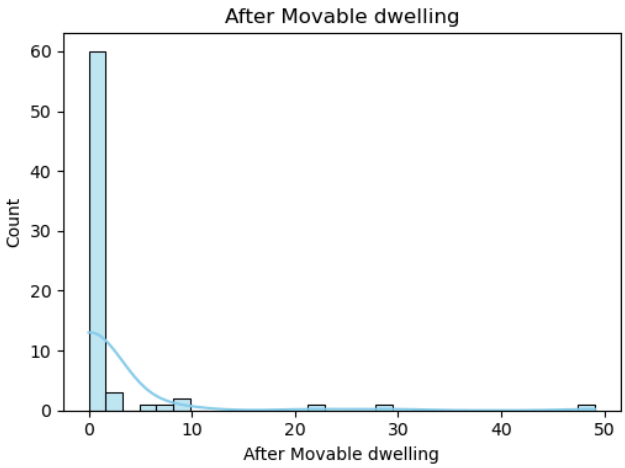

After Apartment, duplex
After Apartment, duplex

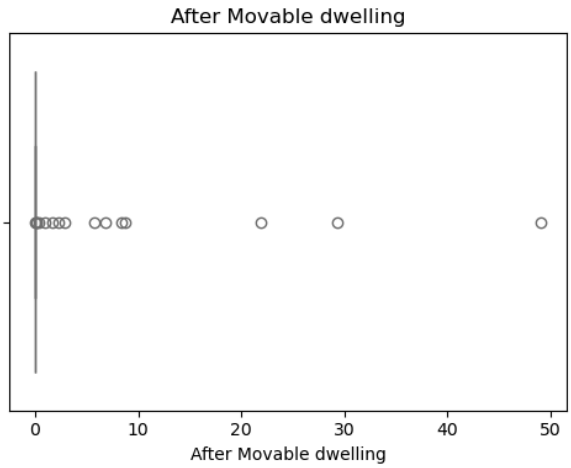

After Apartment, building that has fewer than five storeys
Count
After Apartment, building that has fewer than five storeys

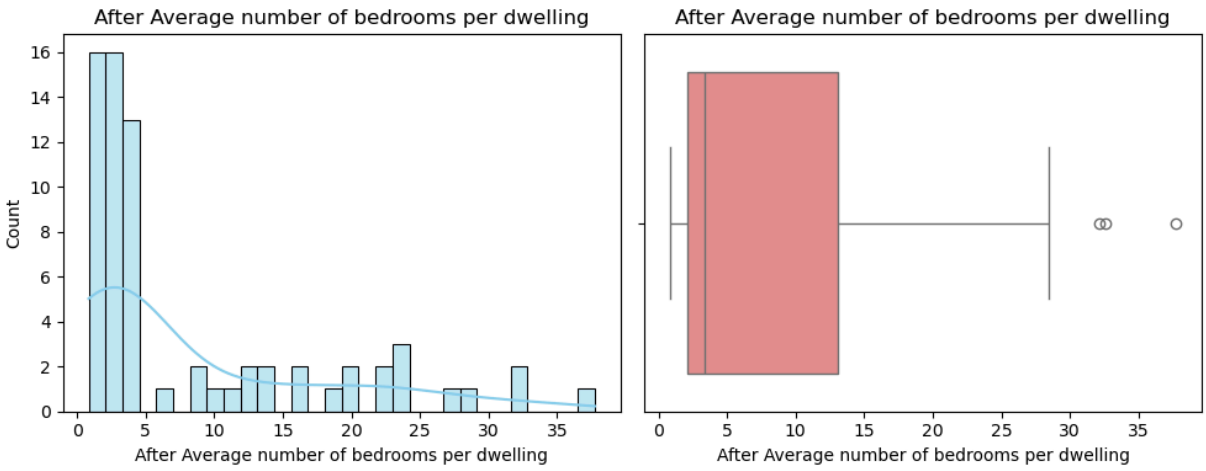

After Apartment, building that has fewer than five storeys
After Apartment, building that has fewer than five storeys

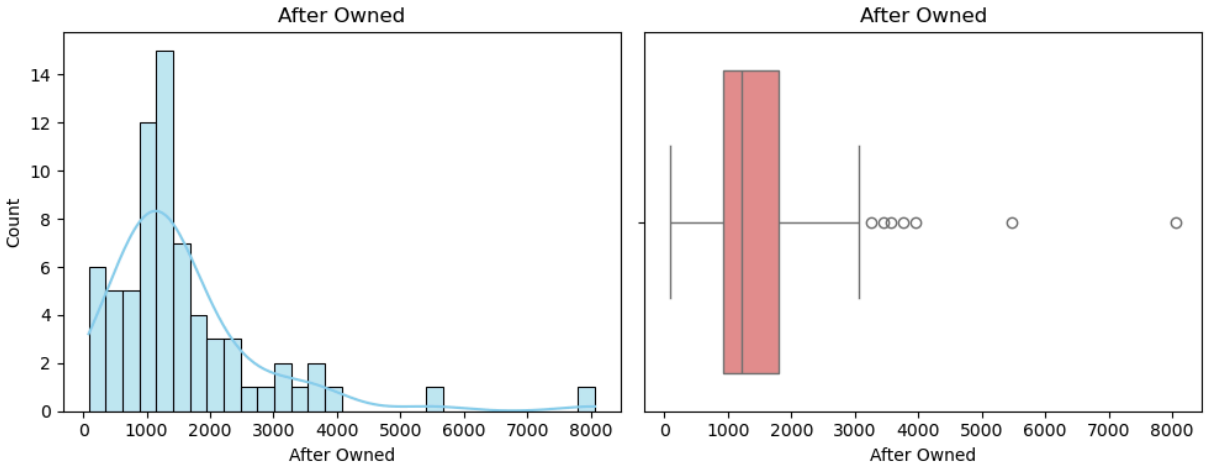

After Apartment, building that has five or more storeys
Count
After Apartment, building that has five or more storeys

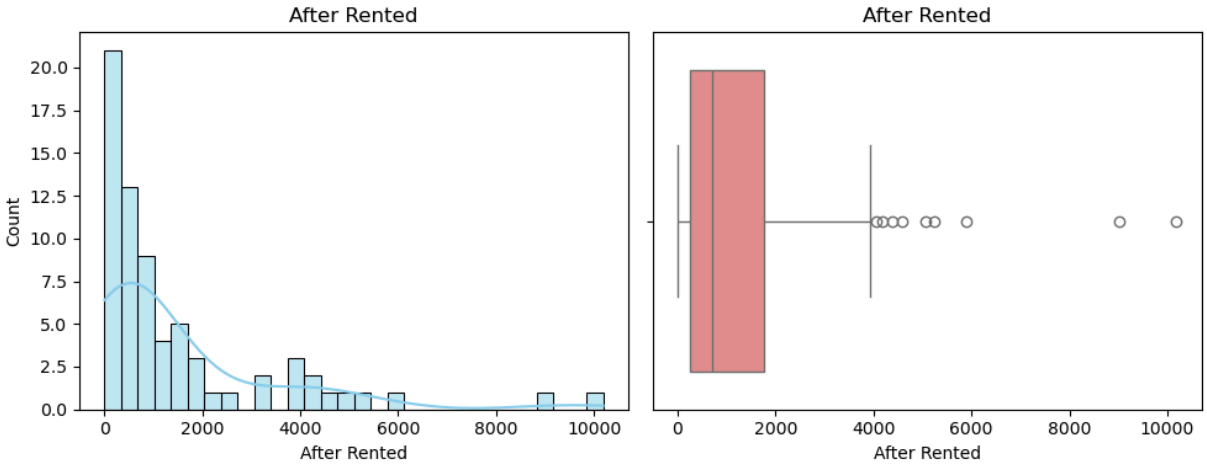

After Apartment, building that has five or more storeys
After Apartment, building that has five or more storeys


After Movable dwelling
Count
After Movable dwelling


After Movable dwelling
After Movable dwelling







Change in Dwelling Characteristics ($X_{D,t=B} - X_{D,t=A}$)

In [20]:
```
before_combied_df = before_df.copy()
after_combined_df = after_df.copy()
```

In [21]:
```
before_combied_df.columns = before_combied_df.columns.str.replace(r'^\w+\s+'
after_combined_df.columns = after_combined_df.columns.str.replace(r'^\w+\s+'
```

In [22]:
```
change_df = after_combined_df - before_combied_df
```

Basic Information

In [23]:
```
print("Shape is", change_df.shape)
print(change_df.info())
```

```
Shape is (70, 13)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 13 columns):
 #   Column                                                  Non-Null C
ount  Dtype
---  ------                                                  ----------
----  -----
 0   Change Population Density per square kilometre          70 non-nul
l     float64
 1   Change Dwellings                                        42 non-nul
l     float64
 2   Change Total Occupied Private Dwellings                 70 non-nul
l     float64
 3   Change Single-detached house                            70 non-nul
l     float64
 4   Change Semi-detached house                              67 non-nul
l     float64
 5   Change Row house                                        67 non-nul
l     float64
 6   Change Apartment, duplex                                67 non-nul
l     float64
 7   Change Apartment, building that has fewer than five storeys  67 non-nul
l     float64
 8   Change Apartment, building that has five or more storeys     69 non-nul
l     float64
 9   Change Movable dwelling                                 70 non-nul
l     float64
 10  Change Average number of bedrooms per dwelling          67 non-nul
l     float64
 11  Change Owned                                            70 non-nul
l     float64
 12  Change Rented                                           70 non-nul
l     float64
dtypes: float64(13)
memory usage: 7.2 KB
None
```

Descriptive Statistics

In [24]:
```
change_df.describe()
```

Out[24]:

	Change Population Density per square kilometre	Change Dwellings	Change Total Occupied Private Dwellings	Change Single-detached house	Change Semi-detached house	C Row
count	70.000000	42.000000	70.000000	70.000000	67.000000	67.0
mean	26299.634640	883.319497	693.964453	-82.241943	-0.833926	52.6
std	60371.696622	1336.536040	1270.095669	312.538167	150.808506	102.
min	-114328.566760	-134.520004	-3232.366353	-1744.971817	-1134.584800	-182.3
25%	323.224019	101.644461	64.381509	-111.097820	-7.404296	-4.9
50%	12068.849501	308.709050	308.536828	-42.882578	0.732230	14.3
75%	24253.407984	866.104584	889.228913	2.040854	17.634644	97.
max	265403.516472	6360.438617	6448.940354	717.692927	298.465238	414.3

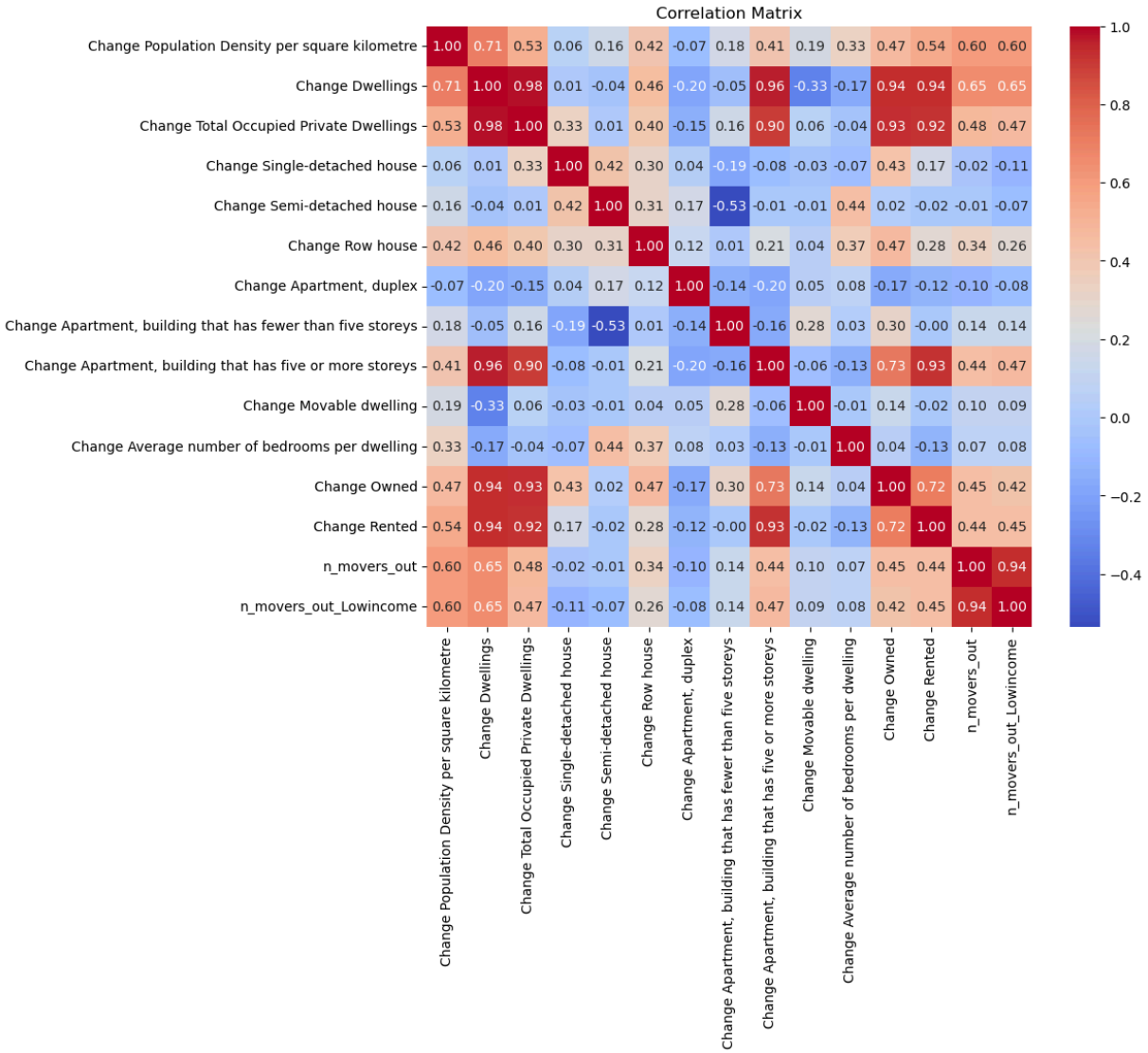
Correlations

In [25]:
```
temp_change_df = change_df.copy()
```

In [26]:
```
temp_change_df["n_movers_out"] = y1
temp_change_df["n_movers_out_Lowincome"] = y2
```

In [27]:
```
c_corr = temp_change_df.corr()
```

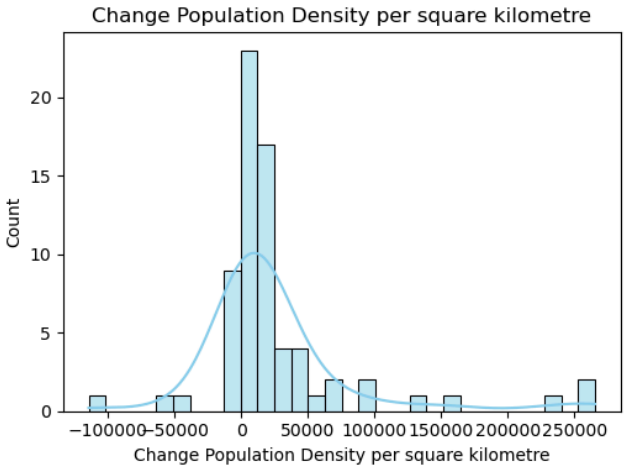
In [28]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(c_corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

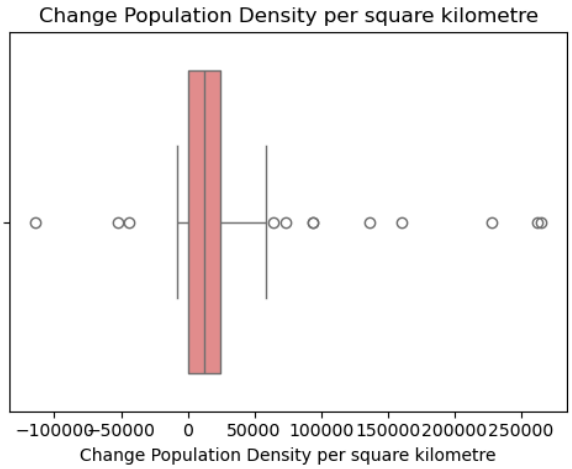


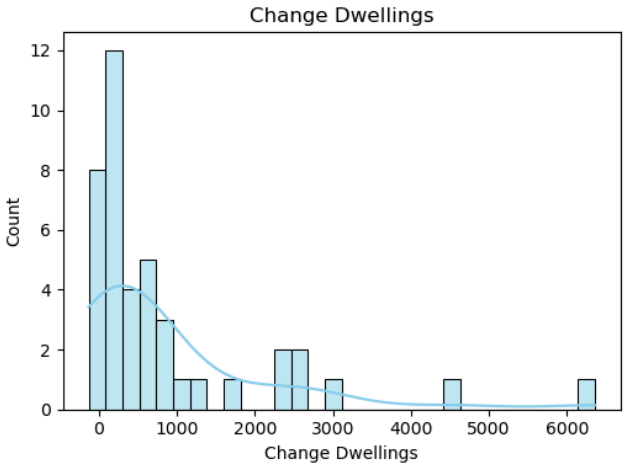
Visualizations

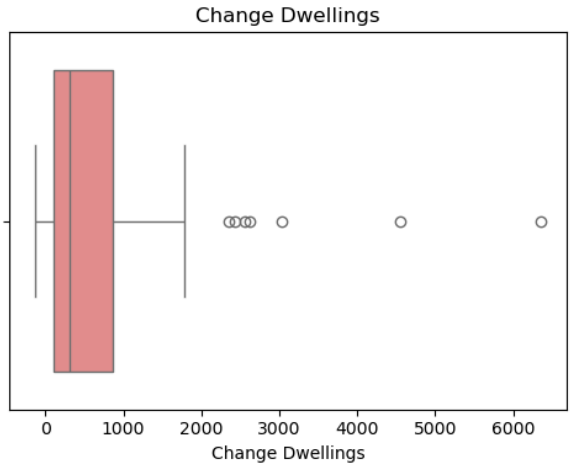
In [29]:

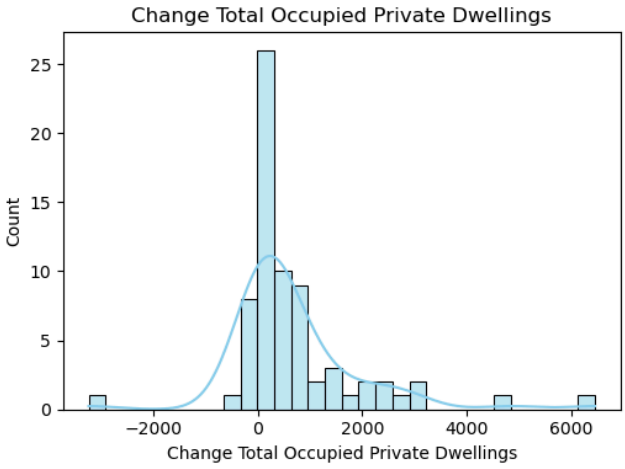
```
for col in change_df.columns:
    plt.figure(figsize=(10, 4))

    # Histogram
    plt.subplot(1, 2, 1)
    sns.histplot(change_df[col], kde=True, bins=30, color='skyblue')
    plt.title(f'{col}')
    plt.xlabel(col)

    # Boxplot
    plt.subplot(1, 2, 2)
    sns.boxplot(x=change_df[col], color='lightcoral')
    plt.title(f'{col}')
    plt.xlabel(col)
    plt.tight_layout()
    plt.show()
```

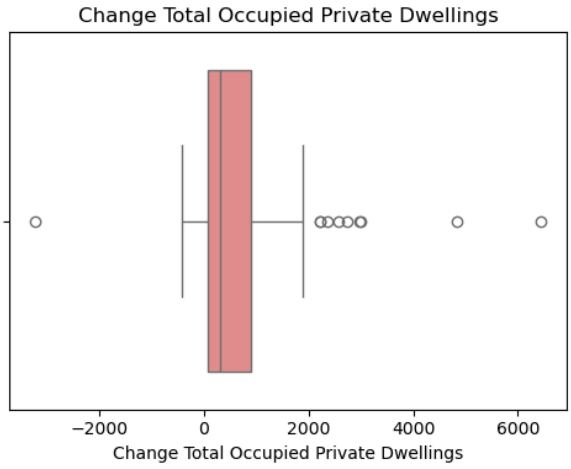

Change Population Density per square kilometre
Count
Change Population Density per square kilometre

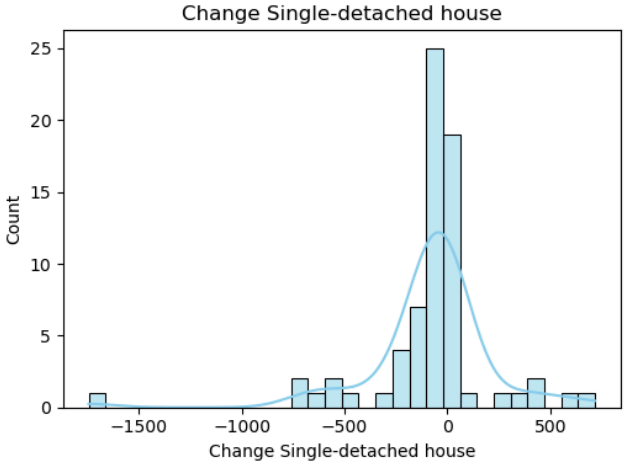

Change Population Density per square kilometre
Change Population Density per square kilometre

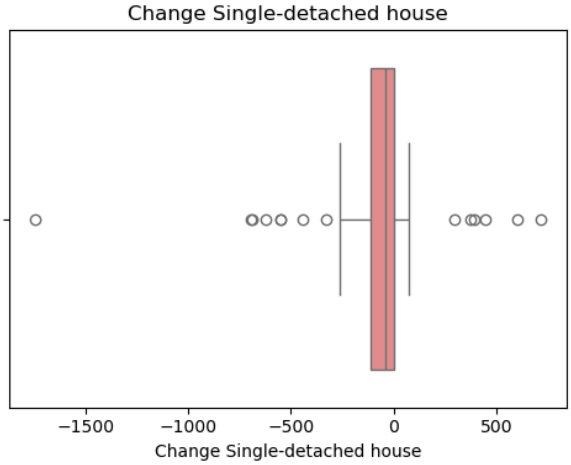

Change Dwellings
Count
Change Dwellings

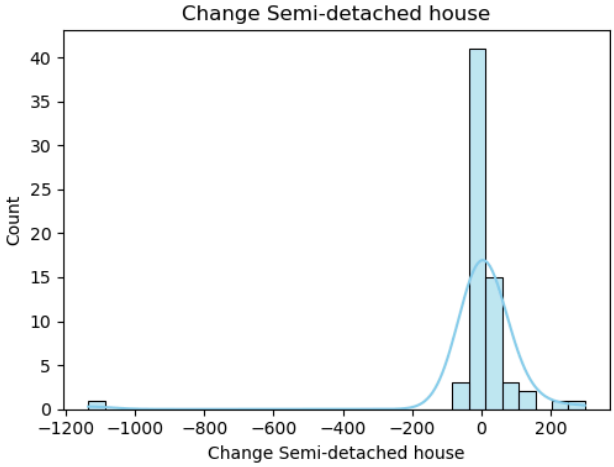

Change Dwellings
Change Dwellings

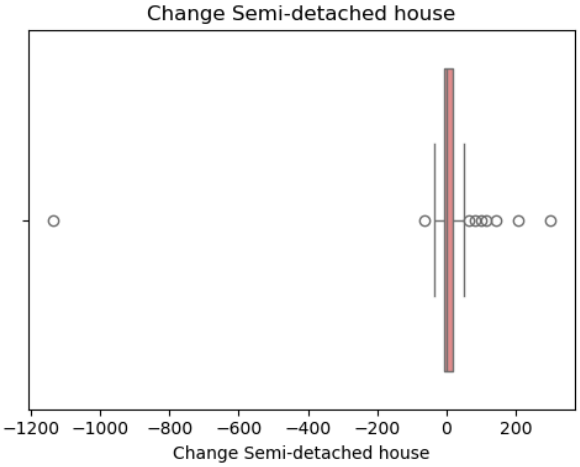

Change Total Occupied Private Dwellings
Count
Change Total Occupied Private Dwellings

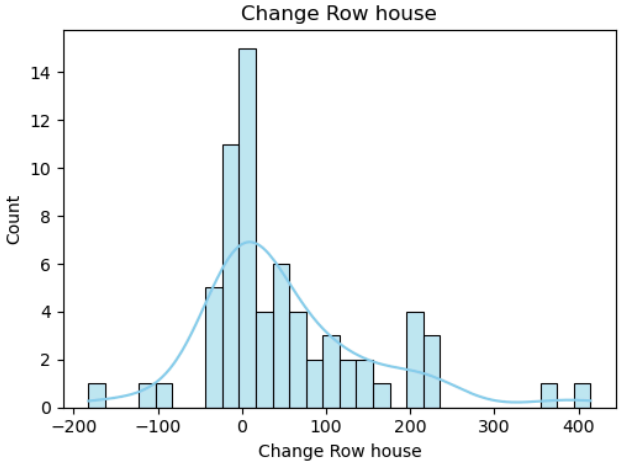

Change Total Occupied Private Dwellings
Change Total Occupied Private Dwellings

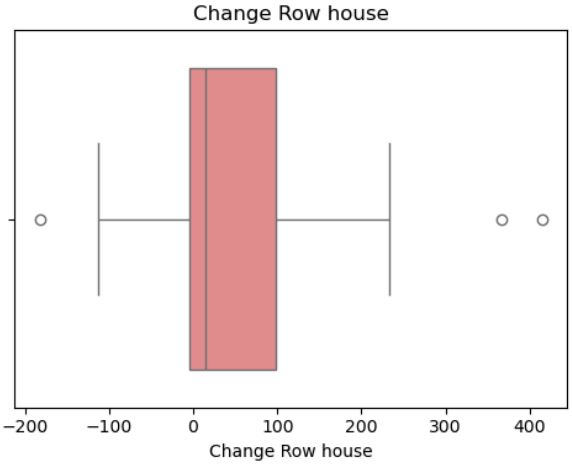

Change Single-detached house
Count
Change Single-detached house

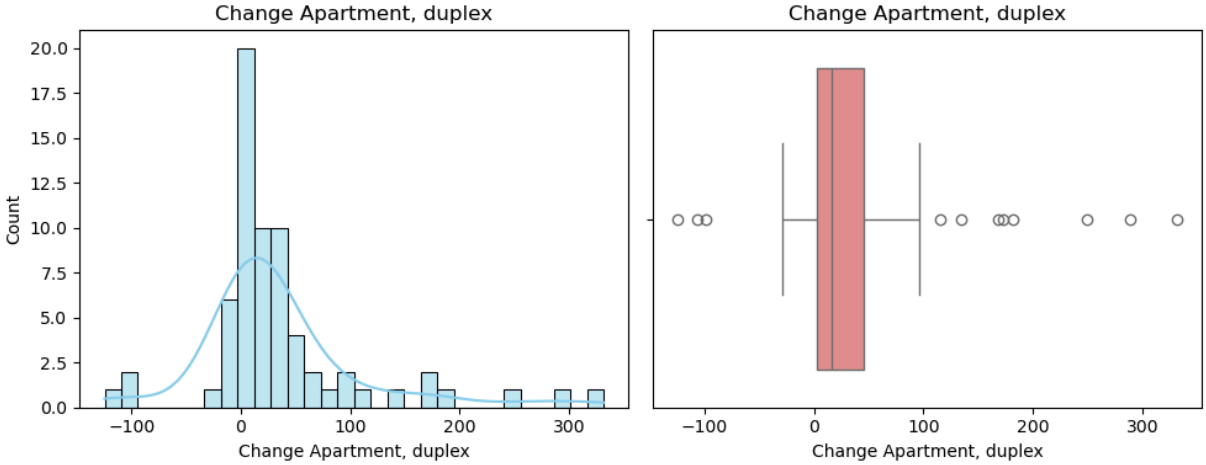

Change Single-detached house
Change Single-detached house

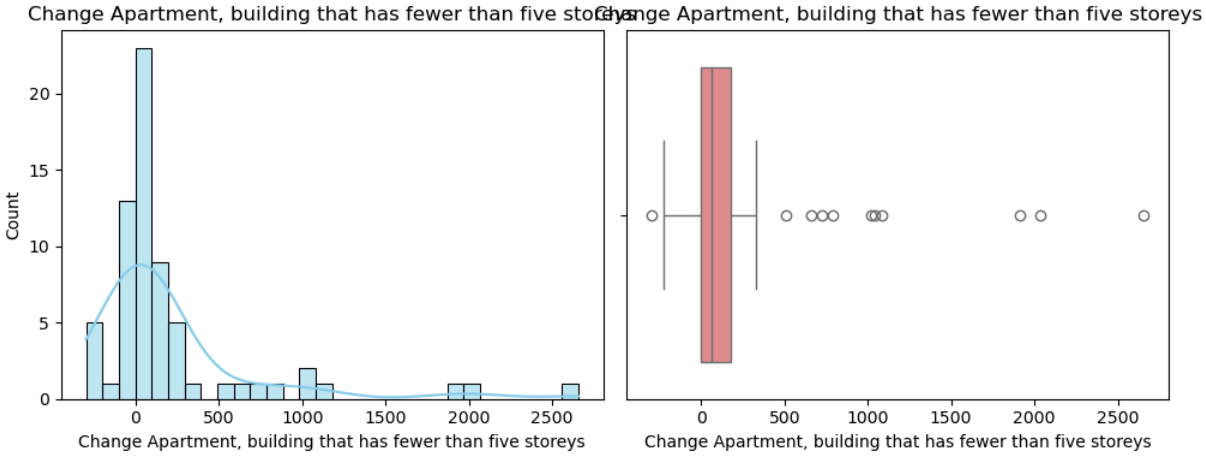

Change Semi-detached house
Count
Change Semi-detached house

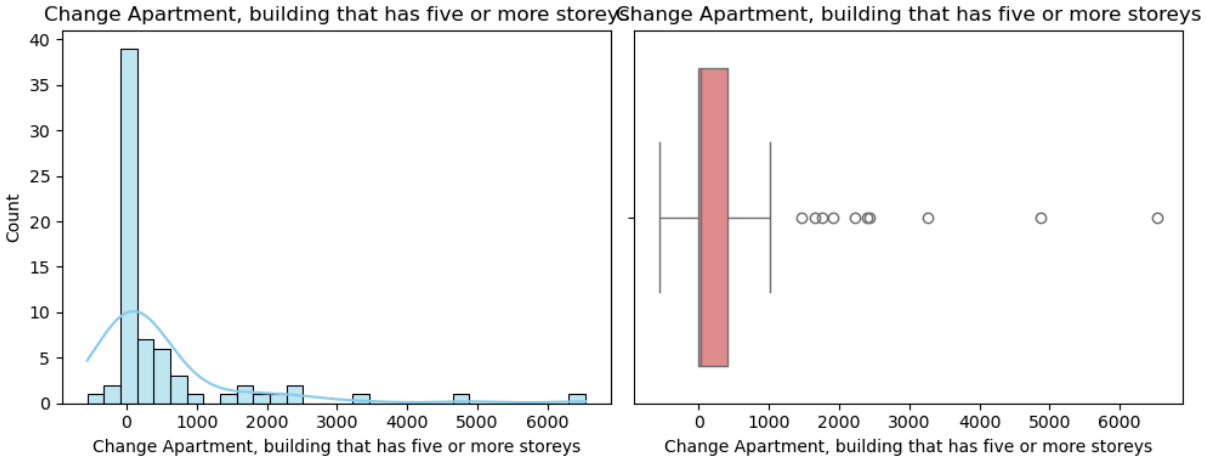

Change Semi-detached house
Change Semi-detached house

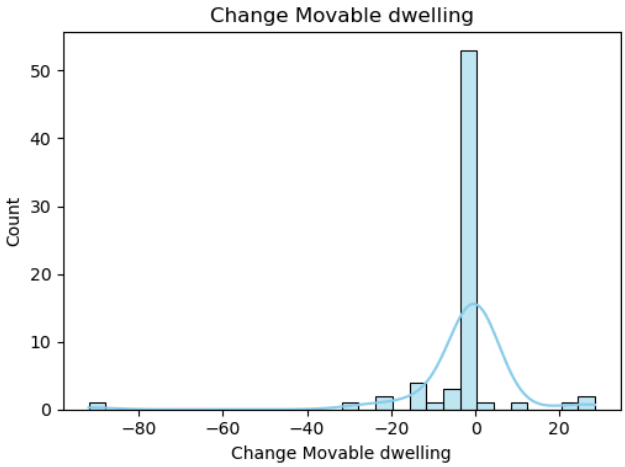

Change Row house
Count
Change Row house

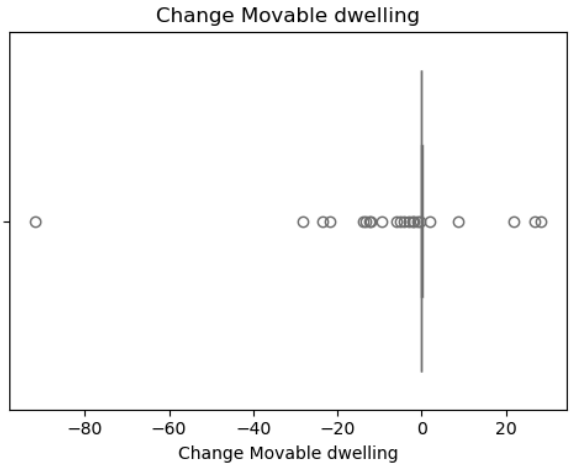

Change Row house
Change Row house

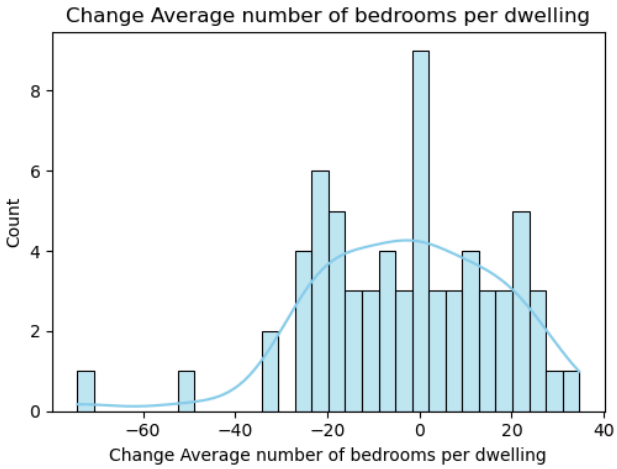

Change Apartment, duplex
Count
Change Apartment, duplex
Change Apartment, duplex
Change Apartment, duplex

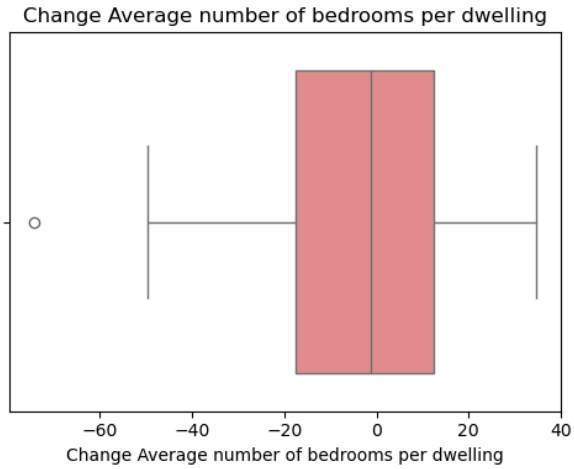

Change Apartment, building that has fewer than five storeys
Count
Change Apartment, building that has fewer than five storeys
Change Apartment, building that has fewer than five storeys
Change Apartment, building that has fewer than five storeys

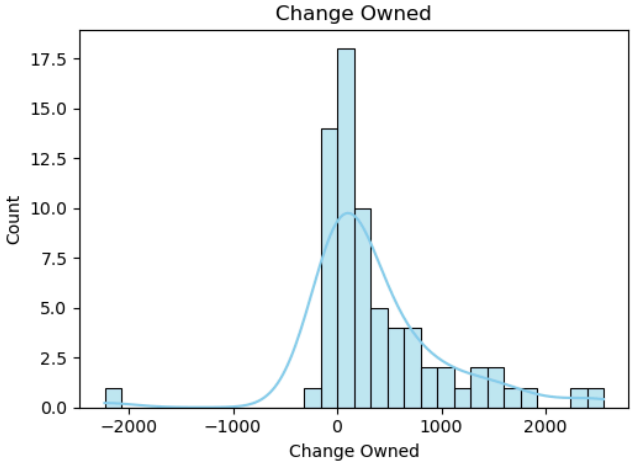

Change Apartment, building that has five or more storeys
Count
Change Apartment, building that has five or more storeys
Change Apartment, building that has five or more storeys
Change Apartment, building that has five or more storeys

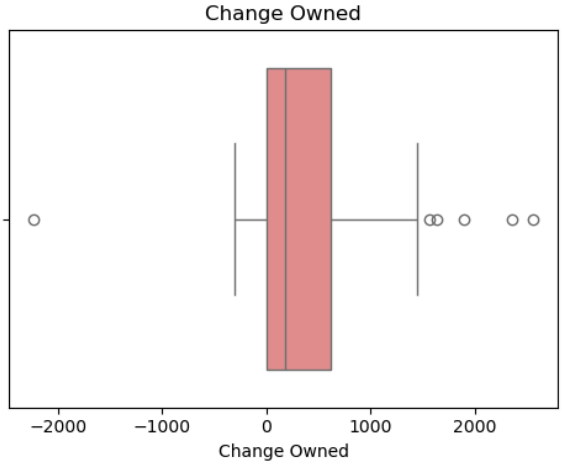

Change Movable dwelling
Count
Change Movable dwelling

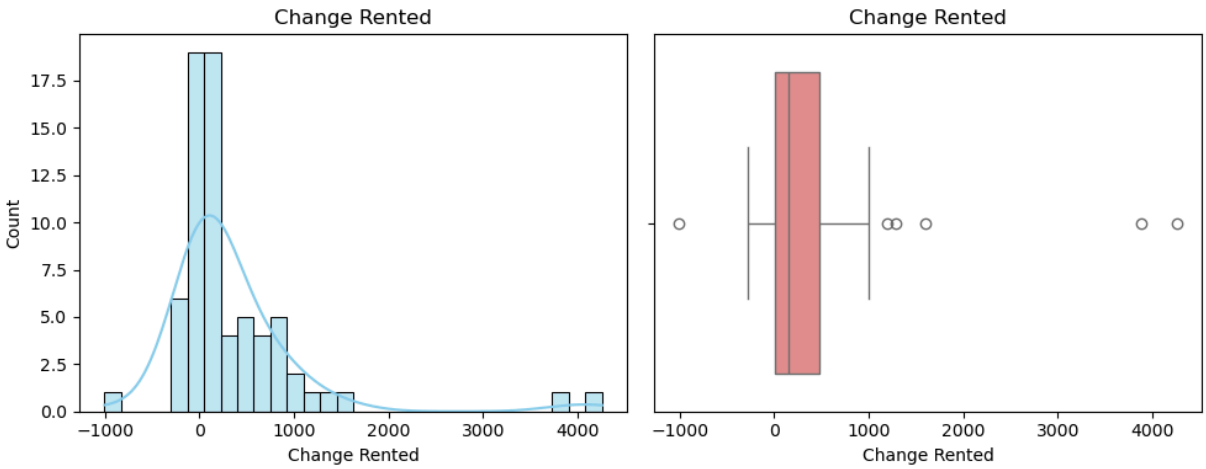

Change Movable dwelling
Change Movable dwelling


Change Average number of bedrooms per dwelling
Count
Change Average number of bedrooms per dwelling


Change Average number of bedrooms per dwelling
Change Average number of bedrooms per dwelling


Change Owned
Count
Change Owned


Change Owned
Change Owned



Dependent Variables

The 2 dependent variables we are looking at are:

1. Y_1 = `n_movers_out`
2. Y_2 = `n_movers_out_Lowincome`
3. Y_1' = `%n_movers_out`
4. Y_2' = `%n_movers_out_Lowincome`

In [30]:
```
original_population = df["n_movers_out"] + df["n_stayers"]
original_low_income_population = df["n_movers_out_Lowincome"] + df["n_stayer
```

In [31]:
```
y1_percent = y1 / original_population
y2_percent = y2 / original_low_income_population
```

In [32]:
```
y1_percent.name = "%n_movers_out"
y2_percent.name = "%n_movers_out_Lowincome"
```

In [33]:
```
dependent_variables = pd.concat([y1, y2, y1_percent, y2_percent], axis=1)
```

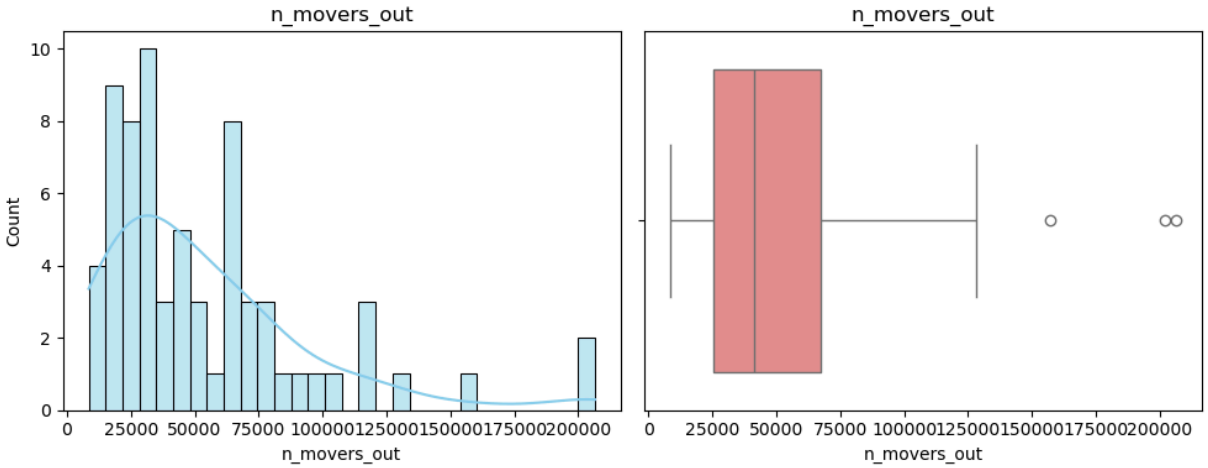
In [34]:
```
dependent_variables.describe()
```

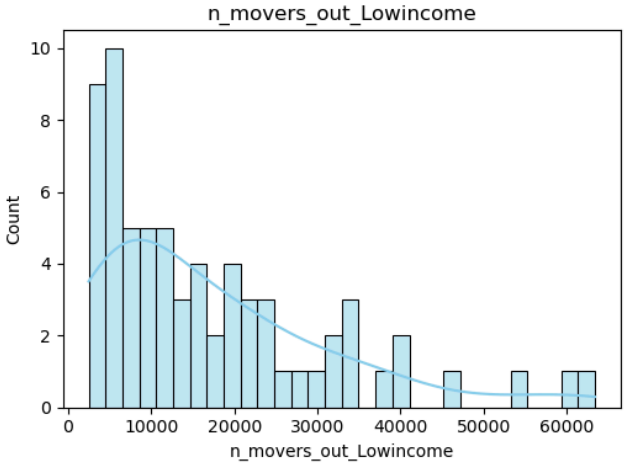
Out[34]:

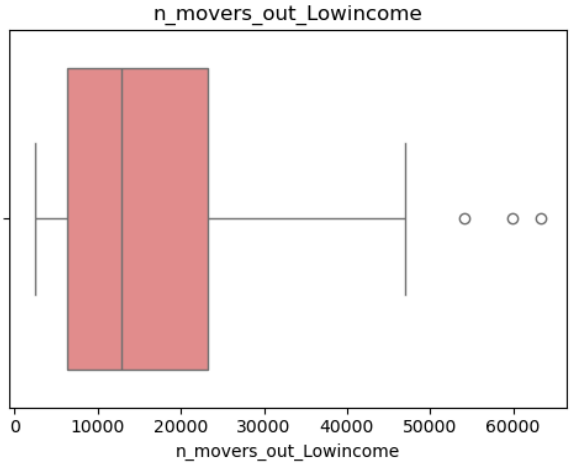
	n_movers_out	n_movers_out_Lowincome	%n_movers_out	%n_movers_out_L
count	68.000000	68.000000	54.000000	
mean	53754.044118	17288.161765	0.607632	
std	41481.434485	14208.638609	0.160230	
min	8495.000000	2465.000000	0.328788	
25%	25452.500000	6362.500000	0.465190	
50%	41452.500000	12812.500000	0.612212	
75%	67363.750000	23213.750000	0.723530	
max	206545.000000	63385.000000	0.968148	

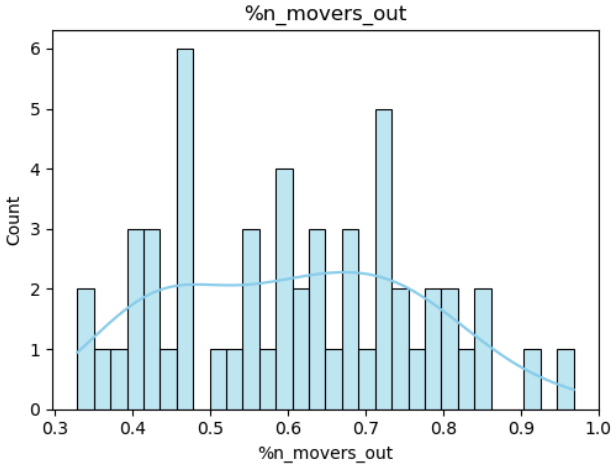
In [35]:

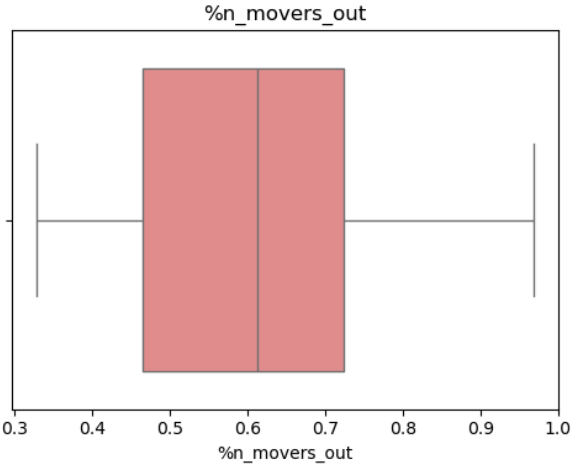
```
for col in dependent_variables.columns:
    plt.figure(figsize=(10, 4))

    # Histogram
    plt.subplot(1, 2, 1)
    sns.histplot(dependent_variables[col], kde=True, bins=30, color='skyblue
    plt.title(f'{col}')
    plt.xlabel(col)

    # Boxplot
    plt.subplot(1, 2, 2)
    sns.boxplot(x=dependent_variables[col], color='lightcoral')
    plt.title(f'{col}')
    plt.xlabel(col)
    plt.tight_layout()
    plt.show()
```

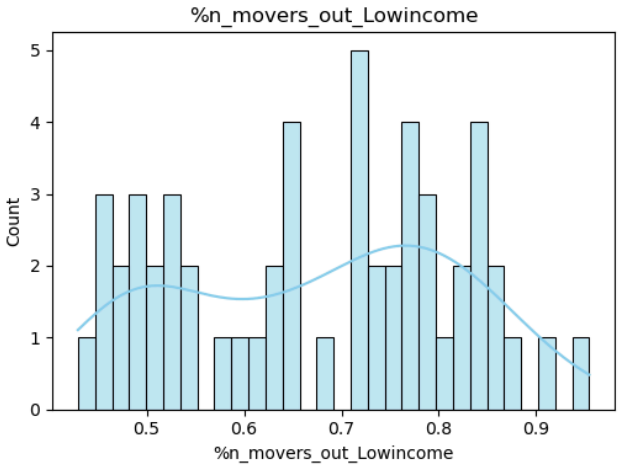


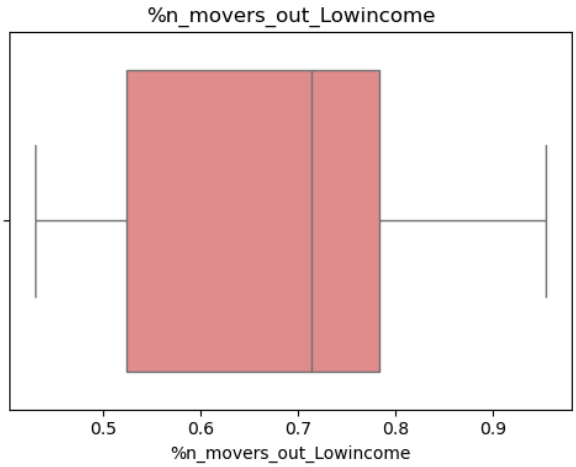

n_movers_out_Lowincome
Count
n_movers_out_Lowincome


n_movers_out_Lowincome
n_movers_out_Lowincome


%n_movers_out
Count
%n_movers_out


%n_movers_out
%n_movers_out


%n_movers_out_Lowincome
Count
%n_movers_out_Lowincome


%n_movers_out_Lowincome
%n_movers_out_Lowincome