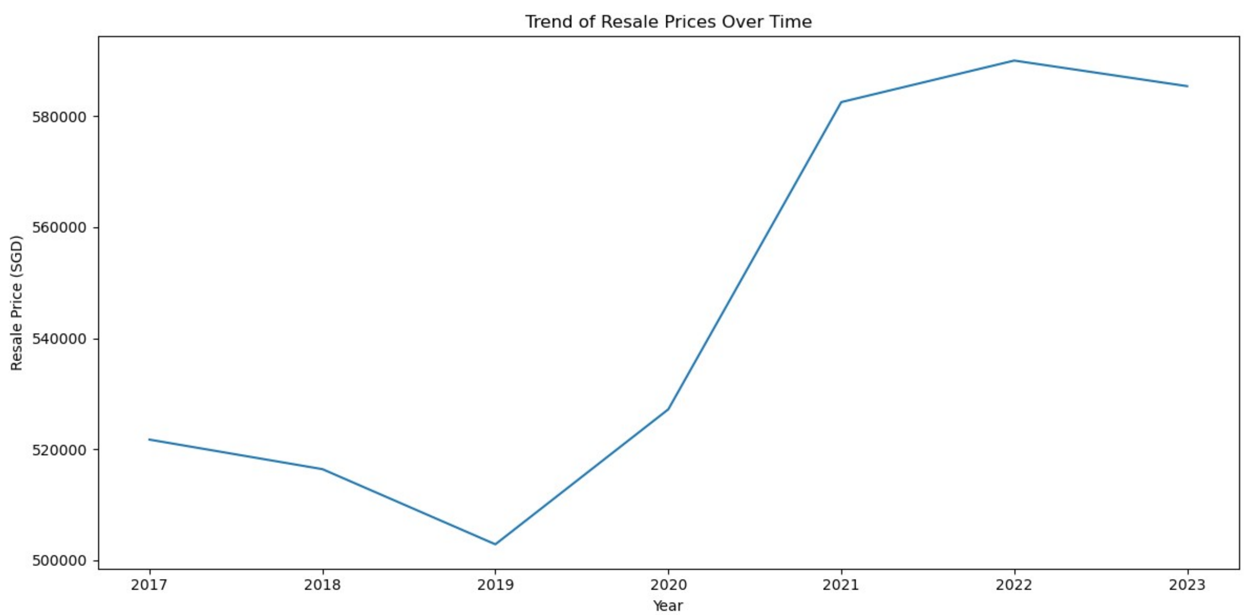
Predicting HDB Resale Prices Beyond Popular Belief

Students: Sie Khai Hinn Crystaline, U2430737L & Chua Jun Xiang, U2430808E

Practical Motivation

In Singapore, the resale market under the Housing and Development Board (HDB) plays a pivotal role in housing the nation. However, there exists a pressing need to leverage advanced analytical techniques to unravel the underlying dynamics necessary to make informed decisions.

```
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x='year', y='resale_price_adj', errorbar=None)
plt.title("Trend of Resale Prices Over Time")
plt.xlabel("Year")
plt.ylabel("Resale Price (SGD)")
plt.tight_layout()
plt.show()
```



The graph depicting Trend of Resale Prices Over Time shows how the resale price changed tremendously over the years, not just due to the Covid-19 pandemic, but also due to inflation.

In an article on Straits Times, 2024, house-buying sentiment is cautious. A significant number of Singaporeans are feeling uncertain about making big financial decisions—like buying a home—because they are worried about rising prices, in other words, inflation. Demand may temporarily dip too. If many potential buyers delay their home purchases, the short-term housing demand might decrease, which could slow down resale volume or price growth—especially in the private market or higher-end HDB flats.

Problem Statement

Therefore, we decided to leverage machine learning—not only to demystify **pricing myths** but to **empower** homeowners with accurate, data-driven insights. We hope this helps sellers price fairly and effectively, while making homes more accessible for buyers.

What are some of these pricing myths?

Larger floor areas tend to fetch higher resale prices. Older flats (40+ years old) are assumed to be less desirable and cheaper. Premium flat models (eg: Premium Apartment, DBSS, Improved) tend to fetch higher prices. Larger flat types (5-room) are expected to have higher prices, because they offer more space. Higher floors (19 to 21, 22 to 24) are believed to have higher prices, due to better views, privacy, less noise compared to lower floors. The nearer the distance to the nearest MRT, the higher the resale price due to greater accessibility.

Data Cleaning

- extracted 2017 to the first month of 2024
- accounted for Inflation that increases every year
- converted storey_range to avg_storey
- converted lease_commence_date to remaining_lease_months

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import requests
import time
from geopy.distance import geodesic
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier,
RandomForestRegressor
from sklearn.cluster import KMeans
from sklearn.ensemble import IsolationForest
from sklearn.metrics import accuracy_score, mean_absolute_error,
silhouette_score, confusion_matrix, classification_report, roc_curve,
auc

df =
pd.read_csv("C:/Users/Crystaline/Downloads/Resaleflatpricesbasedonregi
strationdatefromJan2017onwards.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169584 entries, 0 to 169583
Data columns (total 11 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   month                169584 non-null  object
 1   town                 169584 non-null  object
 2   flat_type            169584 non-null  object
 3   block                169584 non-null  object
 4   street_name          169584 non-null  object
 5   storey_range         169584 non-null  object
 6   floor_area_sqm       169584 non-null  float64
 7   flat_model           169584 non-null  object
 8   lease_commence_date  169584 non-null  int64
 9   remaining_lease      169584 non-null  object
 10  resale_price         169584 non-null  float64
dtypes: float64(2), int64(1), object(8)
memory usage: 14.2+ MB
```

```
df.head()
```

```
     month        town flat_type block        street_name storey_range
\
0  2017-01  ANG MO KIO    2 ROOM   406  ANG MO KIO AVE 10     10 TO 12

1  2017-01  ANG MO KIO    3 ROOM   108   ANG MO KIO AVE 4     01 TO 03

2  2017-01  ANG MO KIO    3 ROOM   602   ANG MO KIO AVE 5     01 TO 03

3  2017-01  ANG MO KIO    3 ROOM   465  ANG MO KIO AVE 10     04 TO 06

4  2017-01  ANG MO KIO    3 ROOM   601   ANG MO KIO AVE 5     01 TO 03


   floor_area_sqm      flat_model  lease_commence_date
remaining_lease  \
0              44        Improved                 1979  61 years 04
months
1              67  New Generation                 1978  60 years 07
months
2              67  New Generation                 1980  62 years 05
months
3              68  New Generation                 1980   62 years 01
month
4              67  New Generation                 1980  62 years 05
months

   resale_price
0       232,000
1       250,000
```

```
2        262,000
3        265,000
4        265,000
```

Account for Inflation

resale_price_adj refers to resale_price having already accounted for Inflation as indicated by CPI

```
df["resale_price"] = df["resale_price"].astype(int)
cpi_dict = {
    2017: 85.084,
    2018: 85.457,
    2019: 85.942,
    2020: 85.794,
    2021: 87.781,
    2022: 93.163,
    2023: 97.666,
    2024: 100
}
df['year'] = pd.to_datetime(df['month']).dt.year
df['cpi'] = df['year'].map(cpi_dict)
df['resale_price_adj'] = df['resale_price'] * (cpi_dict[2024] /
df['cpi'])
df.loc[df['year'] == 2024, 'resale_price_adj'] = df.loc[df['year'] ==
2024, 'resale_price']
pd.set_option('display.float_format', '{:,.0f}'.format)
```

Convert storey_range To avg_storey

Reason: storey_range are categorical-like strings, models cannot easily understand ranges as strings. Hence, we take the midpoint of the range, turning a vague range into a precise numeric feature. This will allow models to capture trends like: higher floor units may be more desirable.

```
def get_average_storey(s):
    try:
        parts = s.split(" TO ")
        return (int(parts[0]) + int(parts[1])) / 2
    except Exception as e:
        print(f"Error processing storey range: {s}, Error: {e}")
        return None
df["avg_storey"] = df["storey_range"].apply(get_average_storey)
```

Convert lease_commence_date To remaining_lease_months

Reason: lease_commence_date does not directly reflect how much lease is left. Whereas, remaining lease directly affects valuation and demand. Numerical values all expressed in months, will also provide more predictive power.

```
def lease_to_months(lease_str):
    try:
        years, months = 0, 0
        if 'years' in lease_str:
            years = int(lease_str.split('years')[0].strip())
        if 'months' in lease_str:
            months = int(lease_str.split('years')[1].split('months')
[0].strip())
        return years * 12 + months
    except:
        return None
df['remaining_lease_months'] =
df['remaining_lease'].apply(lease_to_months)
```

Summary of Data Cleaning

```
for col in df.columns:
    print(f"� Column: {col}")
    print(df[col].unique())
    print("-" * 80)
    print("\n")

� Column: month
['2017-01' '2017-02' '2017-03' '2017-04' '2017-05' '2017-06' '2017-07'
 '2017-08' '2017-09' '2017-10' '2017-11' '2017-12' '2018-01' '2018-02'
 '2018-03' '2018-04' '2018-05' '2018-06' '2018-07' '2018-08' '2018-09'
 '2018-10' '2018-11' '2018-12' '2019-01' '2019-02' '2019-03' '2019-04'
 '2019-05' '2019-06' '2019-07' '2019-08' '2019-09' '2019-10' '2019-11'
 '2019-12' '2020-01' '2020-02' '2020-03' '2020-04' '2020-05' '2020-06'
 '2020-07' '2020-08' '2020-09' '2020-10' '2020-11' '2020-12' '2021-01'
 '2021-02' '2021-03' '2021-04' '2021-05' '2021-06' '2021-11' '2021-07'
 '2021-08' '2021-09' '2021-10' '2021-12' '2022-01' '2022-02' '2022-03'
 '2022-04' '2022-05' '2022-06' '2022-07' '2022-08' '2022-09' '2022-10'
 '2022-11' '2022-12' '2023-01' '2023-02' '2023-03' '2023-04' '2023-05'
 '2023-06' '2023-07' '2023-08' '2023-09' '2023-10' '2023-11' '2023-12'
 '2024-01']
----------------------------------------------------------------------
----------


� Column: town
['ANG MO KIO' 'BEDOK' 'BISHAN' 'BUKIT BATOK' 'BUKIT MERAH' 'BUKIT
```

```
PANJANG'
 'BUKIT TIMAH' 'CENTRAL AREA' 'CHOA CHU KANG' 'CLEMENTI' 'GEYLANG'
 'HOUGANG' 'JURONG EAST' 'JURONG WEST' 'KALLANG/WHAMPOA' 'MARINE
PARADE'
 'PASIR RIS' 'PUNGGOL' 'QUEENSTOWN' 'SEMBAWANG' 'SENGKANG' 'SERANGOON'
 'TAMPINES' 'TOA PAYOH' 'WOODLANDS' 'YISHUN']
----------------------------------------------------------------------
----------


□ Column: flat_type
['2 ROOM' '3 ROOM' '4 ROOM' '5 ROOM' 'EXECUTIVE' '1 ROOM'
 'MULTI-GENERATION']
----------------------------------------------------------------------
----------


□ Column: block
['406' '108' '602' ... '605A' '605C' '460A']
----------------------------------------------------------------------
----------


□ Column: street_name
['ANG MO KIO AVE 10' 'ANG MO KIO AVE 4' 'ANG MO KIO AVE 5'
 'ANG MO KIO AVE 1' 'ANG MO KIO AVE 3' 'ANG MO KIO AVE 9'
 'ANG MO KIO AVE 8' 'ANG MO KIO AVE 6' 'ANG MO KIO ST 52'
 'BEDOK NTH AVE 4' 'BEDOK NTH AVE 1' 'BEDOK NTH RD' 'BEDOK STH AVE 1'
 'BEDOK RESERVOIR RD' 'CHAI CHEE ST' 'BEDOK NTH ST 3' 'BEDOK STH RD'
 'CHAI CHEE AVE' 'NEW UPP CHANGI RD' 'CHAI CHEE DR' 'BEDOK STH AVE 2'
 'BEDOK NTH AVE 3' 'BEDOK RESERVOIR VIEW' 'CHAI CHEE RD' 'LENGKONG
TIGA'
 'BEDOK CTRL' 'JLN DAMAI' 'BEDOK NTH AVE 2' 'BEDOK STH AVE 3'
 'SIN MING RD' 'SIN MING AVE' 'BISHAN ST 12' 'BISHAN ST 13' 'BISHAN ST
22'
 'BISHAN ST 24' 'BISHAN ST 23' 'BRIGHT HILL DR' 'SHUNFU RD'
 'BT BATOK ST 34' 'BT BATOK ST 51' 'BT BATOK ST 11' 'BT BATOK ST 52'
 'BT BATOK ST 21' 'BT BATOK EAST AVE 5' 'BT BATOK WEST AVE 6'
 'BT BATOK CTRL' 'BT BATOK WEST AVE 8' 'BT BATOK EAST AVE 4'
 'BT BATOK ST 31' 'BT BATOK ST 25' 'BT BATOK EAST AVE 3'
 'BT BATOK WEST AVE 5' 'BT BATOK ST 24' 'JLN BT HO SWEE'
 'TELOK BLANGAH DR' 'BEO CRES' 'TELOK BLANGAH CRES' 'TAMAN HO SWEE'
 'TELOK BLANGAH RISE' 'TELOK BLANGAH WAY' 'JLN BT MERAH' 'JLN KLINIK'
 'TELOK BLANGAH HTS' 'BT MERAH VIEW' 'INDUS RD' 'BT MERAH LANE 1'
 'TELOK BLANGAH ST 31' 'MOH GUAN TER' 'HAVELOCK RD' 'HENDERSON CRES'
 'BT PURMEI RD' 'KIM TIAN RD' 'DEPOT RD' 'JLN RUMAH TINGGI' 'DELTA
AVE'
 'JLN MEMBINA' 'REDHILL RD' 'LENGKOK BAHRU' 'ZION RD' 'PETIR RD'
 'PENDING RD' 'BANGKIT RD' 'SEGAR RD' 'JELAPANG RD' 'SENJA RD' 'FAJAR
RD'
```

```
 'BT PANJANG RING RD' 'SENJA LINK' 'LOMPANG RD' 'GANGSA RD' 'TOH YI
DR'
 'FARRER RD' 'JLN KUKOH' 'ROWELL RD' 'WATERLOO ST' 'NEW MKT RD'
 'TG PAGAR PLAZA' 'QUEEN ST' 'BAIN ST' 'CANTONMENT RD' 'TECK WHYE
LANE'
 'CHOA CHU KANG AVE 4' 'CHOA CHU KANG AVE 3' 'CHOA CHU KANG CRES'
 'CHOA CHU KANG ST 54' 'CHOA CHU KANG CTRL' 'JLN TECK WHYE'
 'CHOA CHU KANG ST 62' 'CHOA CHU KANG NTH 6' 'CHOA CHU KANG DR'
 'CHOA CHU KANG NTH 5' 'CHOA CHU KANG ST 52' 'CHOA CHU KANG AVE 2'
 'CLEMENTI WEST ST 2' 'WEST COAST RD' 'CLEMENTI WEST ST 1'
 'CLEMENTI AVE 4' 'CLEMENTI AVE 5' 'CLEMENTI ST 11' 'CLEMENTI AVE 2'
 'CLEMENTI AVE 3' 'CLEMENTI AVE 1' "C'WEALTH AVE WEST" 'CIRCUIT RD'
 'BALAM RD' 'MACPHERSON LANE' 'EUNOS CRES' 'UBI AVE 1' 'HAIG RD'
 'OLD AIRPORT RD' 'GEYLANG EAST AVE 1' 'SIMS DR' 'PIPIT RD'
 'GEYLANG EAST CTRL' 'EUNOS RD 5' 'CASSIA CRES' 'BUANGKOK CRES'
 'HOUGANG AVE 3' 'HOUGANG AVE 8' 'HOUGANG AVE 1' 'HOUGANG AVE 5'
 'HOUGANG ST 61' 'HOUGANG ST 11' 'HOUGANG AVE 7' 'HOUGANG AVE 4'
 'HOUGANG AVE 2' 'LOR AH SOO' 'HOUGANG ST 92' 'HOUGANG ST 52'
 'HOUGANG AVE 10' 'HOUGANG ST 51' 'UPP SERANGOON RD' 'HOUGANG CTRL'
 'HOUGANG ST 91' 'BUANGKOK LINK' 'HOUGANG ST 31' 'PANDAN GDNS'
 'TEBAN GDNS RD' 'JURONG EAST ST 24' 'JURONG EAST ST 21'
 'JURONG EAST AVE 1' 'JURONG EAST ST 13' 'JURONG EAST ST 32' 'TOH GUAN
RD'
 'JURONG WEST ST 93' 'BOON LAY AVE' 'HO CHING RD' 'BOON LAY DR'
 'TAO CHING RD' 'JURONG WEST ST 91' 'JURONG WEST ST 42'
 'JURONG WEST ST 92' 'BOON LAY PL' 'JURONG WEST ST 52' 'TAH CHING RD'
 'JURONG WEST ST 81' 'YUNG SHENG RD' 'JURONG WEST ST 25'
 'JURONG WEST ST 73' 'JURONG WEST ST 72' 'JURONG WEST AVE 3'
 'JURONG WEST AVE 5' 'YUNG HO RD' 'JURONG WEST ST 74' 'JURONG WEST AVE
1'
 'JURONG WEST ST 71' 'JURONG WEST ST 61' 'JURONG WEST ST 65'
 'JURONG WEST CTRL 1' 'JURONG WEST ST 64' 'JURONG WEST ST 62'
 'JURONG WEST ST 41' 'JURONG WEST ST 24' 'JLN BATU' 'JLN BAHAGIA'
 'LOR LIMAU' "ST. GEORGE'S RD" 'KALLANG BAHRU' 'DORSET RD' 'GEYLANG
BAHRU'
 'BENDEMEER RD' 'WHAMPOA DR' 'UPP BOON KENG RD' 'RACE COURSE RD' 'OWEN
RD'
 'NTH BRIDGE RD' 'TOWNER RD' 'FARRER PK RD' 'MCNAIR RD' 'JLN TENTERAM'
 'BOON KENG RD' 'JLN RAJAH' 'MARINE DR' 'MARINE CRES' 'MARINE TER'
 'CHANGI VILLAGE RD' 'PASIR RIS ST 71' 'PASIR RIS ST 11' 'PASIR RIS DR
3'
 'PASIR RIS DR 6' 'PASIR RIS ST 21' 'PASIR RIS DR 4' 'PASIR RIS ST 53'
 'PASIR RIS DR 10' 'PASIR RIS ST 52' 'PASIR RIS ST 12' 'PASIR RIS ST
51'
 'PASIR RIS ST 72' 'PASIR RIS DR 1' 'PUNGGOL FIELD' 'EDGEDALE PLAINS'
 'PUNGGOL FIELD WALK' 'EDGEFIELD PLAINS' 'PUNGGOL RD' 'PUNGGOL EAST'
 'PUNGGOL DR' 'PUNGGOL CTRL' 'PUNGGOL PL' "C'WEALTH CL" 'STIRLING RD'
 'MEI LING ST' "C'WEALTH CRES" "C'WEALTH DR" 'GHIM MOH RD' 'DOVER RD'
 'HOLLAND AVE' 'STRATHMORE AVE' 'HOLLAND DR' 'GHIM MOH LINK'
```

```
 'CLARENCE LANE' 'DOVER CRES' 'SEMBAWANG DR' 'SEMBAWANG CL' 'MONTREAL
DR'
 'ADMIRALTY LINK' 'ADMIRALTY DR' 'SEMBAWANG CRES' 'CANBERRA RD'
 'FERNVALE RD' 'COMPASSVALE LANE' 'ANCHORVALE RD' 'RIVERVALE DR'
 'RIVERVALE CRES' 'SENGKANG EAST WAY' 'RIVERVALE ST' 'RIVERVALE WALK'
 'FERNVALE LANE' 'ANCHORVALE LINK' 'COMPASSVALE RD' 'COMPASSVALE CRES'
 'JLN KAYU' 'COMPASSVALE WALK' 'COMPASSVALE DR' 'COMPASSVALE LINK'
 'COMPASSVALE BOW' 'SENGKANG CTRL' 'ANCHORVALE LANE' 'ANCHORVALE DR'
 'COMPASSVALE ST' 'SERANGOON AVE 4' 'LOR LEW LIAN' 'SERANGOON AVE 2'
 'SERANGOON NTH AVE 1' 'SERANGOON AVE 1' 'SERANGOON CTRL'
 'SERANGOON NTH AVE 4' 'TAMPINES ST 22' 'TAMPINES ST 41' 'TAMPINES AVE
4'
 'TAMPINES ST 44' 'TAMPINES ST 81' 'TAMPINES ST 11' 'TAMPINES ST 23'
 'TAMPINES ST 91' 'TAMPINES ST 21' 'TAMPINES ST 83' 'TAMPINES ST 42'
 'TAMPINES ST 71' 'TAMPINES ST 45' 'TAMPINES ST 34' 'TAMPINES ST 82'
 'TAMPINES AVE 9' 'SIMEI ST 1' 'SIMEI ST 5' 'TAMPINES ST 72'
 'TAMPINES ST 84' 'SIMEI ST 2' 'TAMPINES CTRL 7' 'TAMPINES ST 33'
 'TAMPINES ST 32' 'TAMPINES AVE 5' 'LOR 5 TOA PAYOH' 'LOR 7 TOA PAYOH'
 'LOR 4 TOA PAYOH' 'LOR 1 TOA PAYOH' 'TOA PAYOH EAST' 'POTONG PASIR
AVE 1'
 'TOA PAYOH NTH' 'LOR 8 TOA PAYOH' 'LOR 3 TOA PAYOH' 'POTONG PASIR AVE
3'
 'JOO SENG RD' 'LOR 2 TOA PAYOH' 'TOA PAYOH CTRL' 'MARSILING DR'
 'WOODLANDS ST 13' 'WOODLANDS DR 52' 'WOODLANDS ST 41' 'MARSILING
CRES'
 'WOODLANDS ST 83' 'WOODLANDS CIRCLE' 'WOODLANDS DR 40' 'WOODLANDS ST
31'
 'WOODLANDS DR 16' 'WOODLANDS ST 81' 'WOODLANDS RING RD' 'WOODLANDS DR
53'
 'WOODLANDS DR 62' 'WOODLANDS DR 70' 'WOODLANDS DR 42' 'WOODLANDS DR
50'
 'WOODLANDS AVE 6' 'WOODLANDS DR 14' 'WOODLANDS AVE 1' 'WOODLANDS AVE
5'
 'MARSILING RISE' 'WOODLANDS CRES' 'WOODLANDS DR 73' 'WOODLANDS DR 44'
 'YISHUN RING RD' 'YISHUN AVE 3' 'YISHUN ST 11' 'YISHUN AVE 4'
 'YISHUN ST 22' 'YISHUN ST 71' 'YISHUN AVE 5' 'YISHUN ST 21'
 'YISHUN ST 41' 'YISHUN ST 61' 'YISHUN AVE 6' 'YISHUN AVE 11'
 'YISHUN CTRL' 'YISHUN ST 81' 'YISHUN ST 72' 'YISHUN AVE 2'
 'ANG MO KIO ST 32' 'ANG MO KIO ST 31' 'BEDOK NTH ST 2' 'BEDOK NTH ST
1'
 'JLN TENAGA' 'BEDOK NTH ST 4' 'BT BATOK WEST AVE 4' 'CANTONMENT CL'
 'BOON TIONG RD' 'SPOTTISWOODE PK RD' 'REDHILL CL' 'KIM TIAN PL'
 'CASHEW RD' "QUEEN'S RD" 'CHANDER RD' 'KELANTAN RD' 'SAGO LANE'
 'UPP CROSS ST' 'CHIN SWEE RD' 'SMITH ST' 'TECK WHYE AVE'
 'CHOA CHU KANG ST 51' 'CHOA CHU KANG AVE 5' 'CHOA CHU KANG AVE 1'
 'WEST COAST DR' 'PAYA LEBAR WAY' 'ALJUNIED CRES' 'JOO CHIAT RD' 'PINE
CL'
 'HOUGANG ST 22' 'HOUGANG AVE 9' 'HOUGANG AVE 6' 'HOUGANG ST 21'
 'JURONG WEST ST 75' 'KANG CHING RD' 'KG KAYU RD' 'CRAWFORD LANE'
```

```
 'WHAMPOA WEST' 'BEACH RD' 'CAMBRIDGE RD' "ST. GEORGE'S LANE"
 'JELLICOE RD' 'ELIAS RD' 'HOLLAND CL' 'TANGLIN HALT RD' "C'WEALTH
AVE"
 'WELLINGTON CIRCLE' 'CANBERRA LINK' 'SENGKANG WEST AVE'
 'SENGKANG EAST RD' 'SERANGOON CTRL DR' 'SERANGOON AVE 3'
 'SERANGOON NTH AVE 3' 'TAMPINES AVE 8' 'TAMPINES ST 24' 'TAMPINES ST
12'
 'SIMEI LANE' 'SIMEI ST 4' 'LOR 6 TOA PAYOH' 'KIM KEAT LINK'
 'MARSILING LANE' 'WOODLANDS ST 82' 'WOODLANDS DR 60' 'WOODLANDS AVE
3'
 'WOODLANDS DR 75' 'WOODLANDS AVE 4' 'WOODLANDS ST 32' 'YISHUN AVE 7'
 'ANG MO KIO ST 11' 'BISHAN ST 11' 'BT BATOK WEST AVE 2' 'BT BATOK ST
32'
 'BT BATOK ST 33' 'BT BATOK ST 22' 'BT BATOK WEST AVE 7' 'HOY FATT RD'
 'SILAT AVE' 'EVERTON PK' 'BT MERAH CTRL' 'JELEBU RD' 'EMPRESS RD'
 'VEERASAMY RD' 'CHOA CHU KANG ST 64' 'CHOA CHU KANG ST 53'
 'CHOA CHU KANG NTH 7' 'CLEMENTI AVE 6' 'CLEMENTI ST 13' 'GEYLANG
SERAI'
 'JLN TIGA' 'ALJUNIED RD' 'YUNG LOH RD' 'YUNG AN RD' "JLN MA'MOR"
 'WHAMPOA RD' 'LOR 3 GEYLANG' 'PASIR RIS ST 13' "QUEEN'S CL"
 'DOVER CL EAST' 'SEMBAWANG VISTA' 'TAMPINES ST 43' 'SIMEI RD'
 'KIM KEAT AVE' 'UPP ALJUNIED LANE' 'POTONG PASIR AVE 2' 'WOODLANDS DR
72'
 'MARSILING RD' 'WOODLANDS DR 71' 'YISHUN AVE 9' 'YISHUN ST 20'
 'ANG MO KIO ST 21' 'TIONG BAHRU RD' 'KLANG LANE' 'CHOA CHU KANG LOOP'
 'CLEMENTI ST 14' 'SIMS PL' 'JURONG EAST ST 31' 'YUAN CHING RD'
 'CORPORATION DR' 'YUNG PING RD' 'WHAMPOA STH' 'TESSENSOHN RD' 'JLN
DUSUN'
 'QUEENSWAY' 'FERNVALE LINK' 'KIM PONG RD' 'KIM CHENG ST' 'SAUJANA RD'
 'BUFFALO RD' 'CLEMENTI ST 12' 'DAKOTA CRES' 'JURONG WEST ST 51'
 'FRENCH RD' 'GLOUCESTER RD' 'KG ARANG RD' 'MOULMEIN RD' 'KENT RD'
 'AH HOOD RD' 'SERANGOON NTH AVE 2' 'TAMPINES CTRL 1' 'TAMPINES AVE 7'
 'LOR 1A TOA PAYOH' 'WOODLANDS AVE 9' 'YISHUN CTRL 1' 'LOWER DELTA RD'
 'JLN DUA' 'WOODLANDS ST 11' 'ANG MO KIO AVE 2' 'SELEGIE RD' 'SIMS
AVE'
 'REDHILL LANE' "KING GEORGE'S AVE" 'PASIR RIS ST 41' 'PUNGGOL WALK'
 'LIM LIAK ST' 'JLN BERSEH' 'SENGKANG WEST WAY' 'BUANGKOK GREEN'
 'SEMBAWANG WAY' 'PUNGGOL WAY' 'YISHUN ST 31' 'TECK WHYE CRES'
 'KRETA AYER RD' 'MONTREAL LINK' 'UPP SERANGOON CRES' 'SUMANG LINK'
 'SENGKANG EAST AVE' 'YISHUN AVE 1' 'ANCHORVALE CRES' 'YUNG KUANG RD'
 'ANCHORVALE ST' 'TAMPINES CTRL 8' 'YISHUN ST 51' 'UPP SERANGOON VIEW'
 'TAMPINES AVE 1' 'BEDOK RESERVOIR CRES' 'ANG MO KIO ST 61' 'DAWSON
RD'
 'FERNVALE ST' 'SENG POH RD' 'HOUGANG ST 32' 'TAMPINES ST 86'
 'HENDERSON RD' 'SUMANG WALK' 'CHOA CHU KANG AVE 7' 'KEAT HONG CL'
 'JURONG WEST CTRL 3' 'KEAT HONG LINK' 'ALJUNIED AVE 2' 'SUMANG LANE'
 'CANBERRA CRES' 'CANBERRA ST' 'ANG MO KIO ST 44' 'WOODLANDS RISE'
 'CANBERRA WALK' 'ANG MO KIO ST 51' 'GEYLANG EAST AVE 2'
 'BT BATOK EAST AVE 6' 'BT BATOK WEST AVE 9' 'MARINE PARADE CTRL'
```

```
 'MARGARET DR' 'TAMPINES ST 61' 'YISHUN ST 43']
----------------------------------------------------------------------
----------


 Column: storey_range
['10 TO 12' '01 TO 03' '04 TO 06' '07 TO 09' '13 TO 15' '19 TO 21'
 '22 TO 24' '16 TO 18' '34 TO 36' '28 TO 30' '37 TO 39' '49 TO 51'
 '25 TO 27' '40 TO 42' '31 TO 33' '46 TO 48' '43 TO 45']
----------------------------------------------------------------------
----------


 Column: floor_area_sqm
[ 44.   67.   68.   73.   74.   82.   81.   92.   91.   94.   98.
97.
  99.   90.  117.  119.  118.  112.  121.  147.   45.   59.   63.
70.
  60.   65.   75.   66.   84.   93.  104.  105.  120.  130.  132.
115.
 122.  137.  139.  143.  146.  145.  141.   64.   83.  108.   95.
123.
  69.  103.  102.  100.  107.   86.  101.  150.  155.  144.   34.
51.
  54.   58.   76.   88.   77.  106.   85.   89.  134.  110.  111.
151.
  55.  113.  126.  124.  131.  142.   42.   46.   56.   61.   57.
72.
 109.   47.   96.  116.  128.  140.  148.  156.  157.   71.   52.
79.
 129.  133.  125.   48.   62.  114.   87.  127.  161.  165.   50.
153.
  43.  138.  164.  163.  136.  149.   80.  154.  152.   37.   78.
135.
 170.  192.  182.   31.   49.   53.   60.3 176.  177.  189.   40.
166.
 184.  173.  169.  181.  158.   41.  159.  215.  174.   63.1 179.
162.
  83.1 172.  168.  160.  249.  185.   38.  178.  171.  237.  183.
190.
 175.  188.  187.   35.  186.   39.  243.  199.  222.  210.  241.
167.
 180.  100.2]
----------------------------------------------------------------------
----------


 Column: flat_model
['Improved' 'New Generation' 'DBSS' 'Standard' 'Apartment'
'Simplified'
```

```
 'Model A' 'Premium Apartment' 'Adjoined flat' 'Model A-Maisonette'
 'Maisonette' 'Type S1' 'Type S2' 'Model A2' 'Terrace'
 'Improved-Maisonette' 'Premium Maisonette' 'Multi Generation'
 'Premium Apartment Loft' '2-room' '3Gen']
----------------------------------------------------------------------
----------


□ Column: lease_commence_date
[1979 1978 1980 1981 1976 1977 2011 2012 1996 1988 1985 1986 1974 1984
 1983 1987 1982 2000 2001 2005 1989 2010 1972 1993 1973 1992 1990 1998
 2004 1997 1971 1975 1970 1969 2013 2008 1999 2003 2002 1995 2006 1967
 1968 2007 1991 1966 2009 1994 2014 2015 2016 2017 2018 2019 2022
2020]
----------------------------------------------------------------------
----------


□ Column: remaining_lease
['61 years 04 months' '60 years 07 months' '62 years 05 months'
 '62 years 01 month' '63 years' '61 years 06 months' '58 years 04
months'
 '59 years 08 months' '59 years 06 months' '60 years' '62 years 08
months'
 '61 years' '60 years 10 months' '59 years 03 months' '61 years 05
months'
 '60 years 04 months' '62 years' '60 years 03 months' '63 years 09
months'
 '61 years 01 month' '61 years 10 months' '58 years 06 months'
 '59 years 04 months' '62 years 11 months' '60 years 08 months'
 '93 years 08 months' '93 years 07 months' '60 years 01 month'
 '94 years 08 months' '78 years 04 months' '60 years 06 months'
 '62 years 06 months' '58 years' '70 years 08 months' '63 years 04
months'
 '63 years 06 months' '67 years 07 months' '61 years 07 months'
 '68 years 02 months' '68 years 03 months' '56 years' '67 years 09
months'
 '67 years 05 months' '63 years 07 months' '66 years 03 months'
 '65 years 04 months' '69 years 05 months' '59 years 11 months'
 '60 years 05 months' '69 years 02 months' '69 years 03 months'
 '68 years 10 months' '62 years 10 months' '64 years 04 months'
 '66 years 01 month' '83 years' '83 years 01 month' '87 years 11
months'
 '71 years 02 months' '92 years 04 months' '54 years 06 months'
 '78 years 06 months' '82 years 11 months' '75 years 04 months'
 '66 years 07 months' '66 years 06 months' '75 years 11 months'
 '68 years 04 months' '55 years 09 months' '68 years 07 months'
 '67 years 11 months' '68 years' '69 years 01 month' '69 years 11
months'
 '74 years 06 months' '74 years 04 months' '69 years 06 months'
```

```
 '72 years 03 months' '67 years 02 months' '66 years 05 months'
 '69 years 04 months' '66 years 11 months' '66 years 10 months' '80
years'
 '69 years 08 months' '66 years 09 months' '67 years 10 months'
 '80 years 01 month' '67 years 06 months' '86 years 08 months'
 '71 years 06 months' '71 years 03 months' '67 years 04 months'
 '86 years 11 months' '86 years 10 months' '79 years 05 months'
 '65 years 10 months' '67 years 03 months' '79 years 11 months'
 '53 years 06 months' '57 years 02 months' '52 years 01 month'
 '58 years 03 months' '51 years 06 months' '58 years 08 months'
 '56 years 02 months' '53 years 08 months' '64 years 08 months'
 '55 years 06 months' '95 years 07 months' '55 years 01 month' '55
years'
 '95 years 04 months' '52 years 06 months' '57 years 04 months' '57
years'
 '82 years 06 months' '67 years 08 months' '79 years' '95 years 08
months'
 '90 years 11 months' '87 years 10 months' '82 years 08 months'
 '68 years 06 months' '81 years 02 months' '56 years 05 months'
 '65 years 05 months' '70 years 09 months' '71 years 01 month'
 '94 years 10 months' '80 years 03 months' '83 years 11 months'
 '70 years 10 months' '81 years 01 month' '70 years 06 months' '85
years'
 '81 years 05 months' '80 years 07 months' '80 years 10 months'
 '83 years 10 months' '86 years 09 months' '79 years 09 months'
 '84 years 10 months' '80 years 11 months' '70 years 11 months'
 '84 years 04 months' '83 years 09 months' '81 years 04 months'
 '79 years 07 months' '81 years 03 months' '70 years 01 month'
 '70 years 05 months' '70 years 07 months' '56 years 03 months' '64
years'
 '68 years 09 months' '65 years 03 months' '59 years 01 month'
 '61 years 02 months' '62 years 04 months' '93 years' '71 years 08
months'
 '82 years 03 months' '71 years 04 months' '75 years 05 months'
 '85 years 01 month' '81 years 09 months' '72 years 08 months'
 '75 years 06 months' '78 years 03 months' '82 years 01 month'
 '84 years 03 months' '77 years 05 months' '77 years 07 months'
 '85 years 04 months' '79 years 04 months' '78 years 09 months'
 '81 years 06 months' '78 years' '63 years 08 months' '62 years 07
months'
 '65 years 09 months' '60 years 02 months' '88 years 10 months' '53
years'
 '49 years' '51 years' '50 years 05 months' '54 years 11 months'
 '59 years 05 months' '82 years 04 months' '65 years 07 months' '89
years'
 '72 years 05 months' '78 years 05 months' '58 years 07 months'
 '80 years 05 months' '94 years 04 months' '69 years' '65 years'
 '65 years 02 months' '71 years 07 months' '75 years 03 months'
 '65 years 01 month' '79 years 10 months' '73 years 04 months'
```

```
 '72 years 04 months' '74 years 08 months' '74 years 09 months'
 '80 years 02 months' '81 years 10 months' '86 years 05 months'
 '85 years 06 months' '79 years 01 month' '80 years 06 months'
 '74 years 11 months' '68 years 08 months' '68 years 05 months'
 '48 years 09 months' '63 years 05 months' '65 years 06 months'
 '66 years 04 months' '67 years 01 month' '95 years 09 months'
 '80 years 09 months' '82 years 02 months' '61 years 09 months'
 '90 years 07 months' '57 years 09 months' '54 years' '53 years 05
months'
 '71 years' '58 years 10 months' '92 years 03 months' '66 years 02
months'
 '64 years 03 months' '72 years 02 months' '83 years 04 months'
 '78 years 07 months' '84 years 01 month' '78 years 08 months'
 '72 years 10 months' '71 years 11 months' '76 years' '78 years 02
months'
 '71 years 05 months' '72 years 07 months' '83 years 03 months' '67
years'
 '72 years 11 months' '90 years 08 months' '84 years 11 months'
 '84 years 09 months' '82 years 07 months' '66 years 08 months'
 '88 years 04 months' '85 years 10 months' '90 years 05 months'
 '90 years 06 months' '83 years 05 months' '84 years 02 months'
 '82 years 09 months' '82 years 05 months' '73 years' '84 years 06
months'
 '83 years 07 months' '52 years 05 months' '56 years 08 months'
 '56 years 06 months' '68 years 01 month' '64 years 10 months'
 '57 years 01 month' '86 years 06 months' '88 years 07 months'
 '93 years 11 months' '58 years 02 months' '78 years 10 months'
 '77 years 08 months' '75 years 07 months' '77 years 09 months'
 '75 years 08 months' '77 years 04 months' '85 years 07 months' '95
years'
 '89 years 07 months' '91 years 02 months' '94 years 11 months'
 '85 years 11 months' '94 years 06 months' '94 years 07 months'
 '95 years 01 month' '95 years 02 months' '95 years 06 months'
 '93 years 02 months' '86 years 03 months' '87 years' '86 years 04
months'
 '85 years 03 months' '86 years 02 months' '85 years 09 months' '52
years'
 '49 years 01 month' '57 years 06 months' '56 years 04 months'
 '95 years 03 months' '88 years 11 months' '59 years' '85 years 08
months'
 '92 years 07 months' '83 years 06 months' '83 years 02 months'
 '81 years 11 months' '84 years' '94 years 02 months' '94 years 01
month'
 '85 years 05 months' '85 years 02 months' '83 years 08 months'
 '95 years 10 months' '93 years 10 months' '81 years' '95 years 05
months'
 '88 years 01 month' '94 years 09 months' '92 years 02 months'
 '90 years 04 months' '91 years 11 months' '91 years 07 months'
 '84 years 08 months' '84 years 05 months' '71 years 10 months'
```

```
 '74 years 10 months' '69 years 09 months' '79 years 02 months'
 '64 years 09 months' '79 years 03 months' '79 years 06 months'
 '79 years 08 months' '90 years' '69 years 10 months' '77 years 10
months'
 '49 years 05 months' '52 years 07 months' '54 years 07 months' '88
years'
 '91 years 03 months' '59 years 10 months' '62 years 09 months'
 '78 years 01 month' '71 years 09 months' '81 years 08 months'
 '78 years 11 months' '74 years 05 months' '66 years' '70 years'
 '70 years 02 months' '77 years 02 months' '70 years 03 months'
 '69 years 07 months' '63 years 02 months' '59 years 02 months'
 '61 years 11 months' '61 years 03 months' '60 years 11 months'
 '75 years 02 months' '76 years 10 months' '57 years 11 months'
 '54 years 05 months' '62 years 03 months' '59 years 07 months'
 '62 years 02 months' '64 years 05 months' '63 years 03 months'
 '82 years 10 months' '64 years 11 months' '72 years' '74 years 03
months'
 '80 years 04 months' '93 years 06 months' '68 years 11 months'
 '70 years 04 months' '86 years 07 months' '57 years 03 months'
 '49 years 04 months' '51 years 05 months' '52 years 11 months'
 '55 years 05 months' '56 years 11 months' '87 years 08 months'
 '61 years 08 months' '76 years 11 months' '77 years 06 months'
 '84 years 07 months' '87 years 01 month' '59 years 09 months'
 '92 years 11 months' '77 years' '76 years 09 months' '88 years 05
months'
 '50 years 11 months' '48 years 11 months' '58 years 05 months'
 '57 years 05 months' '94 years 03 months' '74 years 07 months'
 '64 years 07 months' '64 years 01 month' '80 years 08 months'
 '57 years 08 months' '58 years 09 months' '53 years 11 months'
 '55 years 11 months' '56 years 01 month' '88 years 08 months'
 '88 years 06 months' '92 years' '58 years 11 months' '76 years 02
months'
 '76 years 08 months' '75 years 09 months' '77 years 01 month'
 '89 years 10 months' '94 years 05 months' '92 years 05 months'
 '86 years 01 month' '92 years 06 months' '92 years 01 month' '94
years'
 '91 years 04 months' '89 years 06 months' '90 years 03 months'
 '91 years 06 months' '86 years' '90 years 10 months' '77 years 11
months'
 '50 years 04 months' '51 years 11 months' '58 years 01 month'
 '76 years 04 months' '82 years' '81 years 07 months' '65 years 11
months'
 '63 years 01 month' '63 years 10 months' '93 years 05 months'
 '93 years 04 months' '57 years 10 months' '55 years 10 months'
 '64 years 06 months' '87 years 09 months' '64 years 02 months'
 '55 years 07 months' '55 years 08 months' '74 years 01 month'
 '65 years 08 months' '75 years' '51 years 04 months' '54 years 10
months'
 '55 years 04 months' '51 years 10 months' '92 years 10 months'
```

```
 '56 years 10 months' '90 years 09 months' '87 years 07 months'
 '53 years 03 months' '77 years 03 months' '50 years 10 months'
 '48 years 10 months' '52 years 10 months' '93 years 09 months'
 '87 years 03 months' '63 years 11 months' '75 years 01 month'
 '72 years 01 month' '56 years 07 months' '57 years 07 months'
 '53 years 10 months' '72 years 09 months' '75 years 10 months'
 '88 years 02 months' '52 years 03 months' '54 years 04 months'
 '76 years 01 month' '72 years 06 months' '89 years 05 months'
 '52 years 04 months' '91 years 09 months' '90 years 02 months'
 '91 years 05 months' '50 years 03 months' '51 years 03 months'
 '49 years 03 months' '91 years 01 month' '76 years 03 months'
 '60 years 09 months' '74 years' '54 years 03 months' '52 years 09
months'
 '55 years 03 months' '54 years 09 months' '87 years 06 months'
 '92 years 09 months' '76 years 07 months' '88 years 09 months'
 '50 years 02 months' '53 years 09 months' '52 years 02 months'
 '50 years 09 months' '56 years 09 months' '76 years 06 months'
 '89 years 08 months' '51 years 09 months' '89 years 03 months'
 '91 years 08 months' '89 years 04 months' '90 years 01 month'
 '51 years 02 months' '49 years 02 months' '74 years 02 months'
 '93 years 03 months' '54 years 02 months' '73 years 11 months'
 '51 years 08 months' '54 years 08 months' '87 years 05 months'
 '92 years 08 months' '52 years 08 months' '48 years 08 months'
 '50 years 08 months' '50 years 01 month' '73 years 02 months'
 '55 years 02 months' '88 years 03 months' '89 years 02 months'
 '91 years 10 months' '53 years 02 months' '51 years 01 month'
 '54 years 01 month' '73 years 10 months' '50 years 07 months'
 '53 years 01 month' '87 years 04 months' '48 years 07 months'
 '51 years 07 months' '96 years 08 months' '89 years 01 month' '91
years'
 '76 years 05 months' '53 years 07 months' '50 years' '93 years 01
month'
 '48 years 06 months' '50 years 06 months' '73 years 01 month'
 '89 years 11 months' '96 years 07 months' '73 years 09 months'
 '73 years 08 months' '87 years 02 months' '49 years 10 months'
 '48 years 05 months' '96 years 06 months' '89 years 09 months'
 '49 years 11 months' '73 years 07 months' '48 years 04 months'
 '49 years 09 months' '53 years 04 months' '96 years 05 months'
 '73 years 06 months' '48 years 03 months' '49 years 08 months' '48
years'
 '96 years 04 months' '49 years 07 months' '48 years 02 months'
 '96 years 03 months' '73 years 05 months' '48 years 01 month'
 '49 years 06 months' '96 years 02 months' '47 years 09 months'
 '96 years 01 month' '47 years 11 months' '96 years' '95 years 11
months'
 '47 years 10 months' '73 years 03 months' '47 years 08 months'
 '47 years 04 months' '47 years 07 months' '47 years 06 months'
 '47 years 05 months' '47 years 02 months' '47 years 03 months'
 '47 years 01 month' '47 years' '46 years 11 months' '46 years 10
months'
```

```
 '46 years 09 months' '46 years 08 months' '66 years 0 months'
 '95 years 0 months' '46 years 07 months' '46 years 06 months'
 '46 years 03 months' '93 years 0 months' '46 years 05 months'
 '46 years 04 months' '46 years 01 month' '46 years 02 months' '46
years'
 '45 years 09 months' '45 years 11 months' '45 years 10 months'
 '45 years 07 months' '45 years 08 months' '97 years 01 month'
 '45 years 06 months' '97 years 09 months' '45 years 05 months'
 '45 years 04 months' '97 years' '97 years 07 months' '45 years 03
months'
 '45 years 02 months' '96 years 09 months' '45 years 01 month'
 '44 years 10 months' '97 years 05 months' '97 years 04 months'
 '97 years 03 months' '97 years 02 months' '45 years' '96 years 11
months'
 '96 years 10 months' '44 years 09 months' '44 years 08 months'
 '44 years 07 months' '44 years 11 months' '44 years 06 months'
 '44 years 05 months' '44 years 04 months' '44 years 03 months'
 '44 years 02 months' '44 years 01 month' '44 years' '43 years 11
months'
 '43 years 10 months' '43 years 09 months' '43 years 08 months'
 '43 years 07 months' '43 years 06 months' '43 years 05 months'
 '43 years 04 months' '43 years 03 months' '43 years 01 month'
 '43 years 02 months' '43 years' '42 years 11 months' '42 years 08
months'
 '42 years 07 months' '42 years 10 months' '42 years 09 months'
 '42 years 06 months' '42 years 05 months' '42 years 04 months'
 '42 years 03 months' '42 years 02 months' '42 years 01 month']
----------------------------------------------------------------------
----------


 Column: resale_price
[ 232000  250000  262000 ...  680800  473600 1000088]
----------------------------------------------------------------------
----------


 Column: year
[2017 2018 2019 2020 2021 2022 2023 2024]
----------------------------------------------------------------------
----------


 Column: cpi
[ 85.084  85.457  85.942  85.794  87.781  93.163  97.666 100.   ]
----------------------------------------------------------------------
----------


 Column: resale_price_adj
```

```
[272671.71266043 293827.27657374 307930.98584928 ... 660000.
 675000.         688000.        ]
----------------------------------------------------------------------
----------


□ Column: avg_storey
[11.  2.  5.  8. 14. 20. 23. 17. 35. 29. 38. 50. 26. 41. 32. 47. 44.]
----------------------------------------------------------------------
----------


□ Column: remaining_lease_months
[ 736  727  749  744  756  738  700  716  714  720  752  732  730  711
  737  724  723  765  742  702  712  755  728 1124 1123 1136  940  726
  750  696  848  760  762  811  739  818  819  672  813  809  763  795
  784  833  719  725  830  831  826  754  772  792  996 1055  854 1108
  654  942  995  904  799  798  911  820  669  823  815  816  828  839
  894  892  834  867  806  797  832  803  802  960  836  801  814  810
 1040  858  855  808 1043 1042  953  790  807  959  642  686  624  699
  618  704  674  644  776  666 1147  660 1144  630  688  684  990  812
  948 1148 1091 1054  992  822  974  677  785  849  852 1138  963 1007
  850  972  846 1020  977  967  970 1006 1041  957 1018  971  851 1012
 1005  976  955  975  840  845  847  675  768  825  783  708  734  748
 1116  860  987  856  905  981  872  906  939  984 1011  929  931 1024
  952  945  978  936  764  751  789  722 1066  636  588  612  605  659
  713  988  787 1068  869  941  703  965 1132  780  782  859  903  958
  880  868  896  897  962  982 1037 1026  966  899  824  821  585  761
  786  796  804 1149  969  986  741 1087  693  648  641  706 1107  794
  771  866 1000  943 1008  944  874  863  912  938  857  871  999  875
 1088 1019 1017  991  800 1060 1030 1085 1086 1001 1010  993  989  876
 1014 1003  629  680  678  778 1038 1063 1127  698  946  932  907  933
  908  928 1027 1140 1075 1094 1139 1031 1134 1135 1142 1146 1118 1035
 1044 1036 1023 1034 1029  690  676 1143 1067 1028 1111 1002  998  983
 1130 1128 1025 1022 1004 1150 1126 1145 1056 1137 1106 1084 1103 1099
 1016 1013  862  898  837  950  777  951  954  956 1080  838  934  593
  631  655 1095  718  753  861  980  947  893  842  926  843  835  758
  710  743  735  731  902  922  695  653  747  715  746  773  759  994
  779  864  891  964 1122  827  844 1039  687  592  617  635  665  683
 1052  740  923  930 1015  717 1115  924  921 1061  611  587  701  689
 1131  895  775  968  692  705  647  671 1064 1062 1104  707  914  920
  909 1078 1133 1109 1032 1110 1096 1074 1083 1098 1090  935  604  623
  916  979  791  766 1121 1120  694  670  774 1053  770  667  668  888
  788  900  616  658  664  622 1114  682 1089 1051  639  927  610  586
  634 1125 1047  767  679  691  646  873  910 1058  627  652  870 1073
  628 1101 1082 1097  603  615  591 1092  915  729  651  633  663  657
 1050 1113  919 1065  602  645  626  609  681  918 1076  621 1071 1100
 1072  614  590  890 1119  650  887  620  656 1049 1112  632  584  608
  600  878  662 1059 1070 1102  638  886  607 1048  583  619 1160  917
  643  582  606 1079 1159  885  884 1046  598  581 1158 1077  599  883
```

```
  580  597  640 1157  882  579  596  576 1156  595  578 1155  881  594
 1154  573 1152  575 1151  574  879  572  568  571  570  569  566  567
  564  563  562  561  560  559  558  555  557  556  552  554  549  551
  550  547  548 1164  546 1173  545  544 1171  543  542 1161  540  538
 1169 1168 1167 1166 1163 1162  537  536  535  539  534  533  532  531
  530  528  527  526  525  524  523  522  521  520  519  516  518  515
  512  511  514  513  510  509  508  507  506  504]
----------------------------------------------------------------------
----------
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169584 entries, 0 to 169583
Data columns (total 16 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   month                   169584 non-null  object
 1   town                    169584 non-null  object
 2   flat_type               169584 non-null  object
 3   block                   169584 non-null  object
 4   street_name             169584 non-null  object
 5   storey_range            169584 non-null  object
 6   floor_area_sqm          169584 non-null  float64
 7   flat_model              169584 non-null  object
 8   lease_commence_date     169584 non-null  int64
 9   remaining_lease         169584 non-null  object
 10  resale_price            169584 non-null  int32
 11  year                    169584 non-null  int32
 12  cpi                     169584 non-null  float64
 13  resale_price_adj        169584 non-null  float64
 14  avg_storey              169584 non-null  float64
 15  remaining_lease_months  169584 non-null  int64
dtypes: float64(4), int32(2), int64(2), object(8)
memory usage: 19.4+ MB
```

```
df.describe()
```

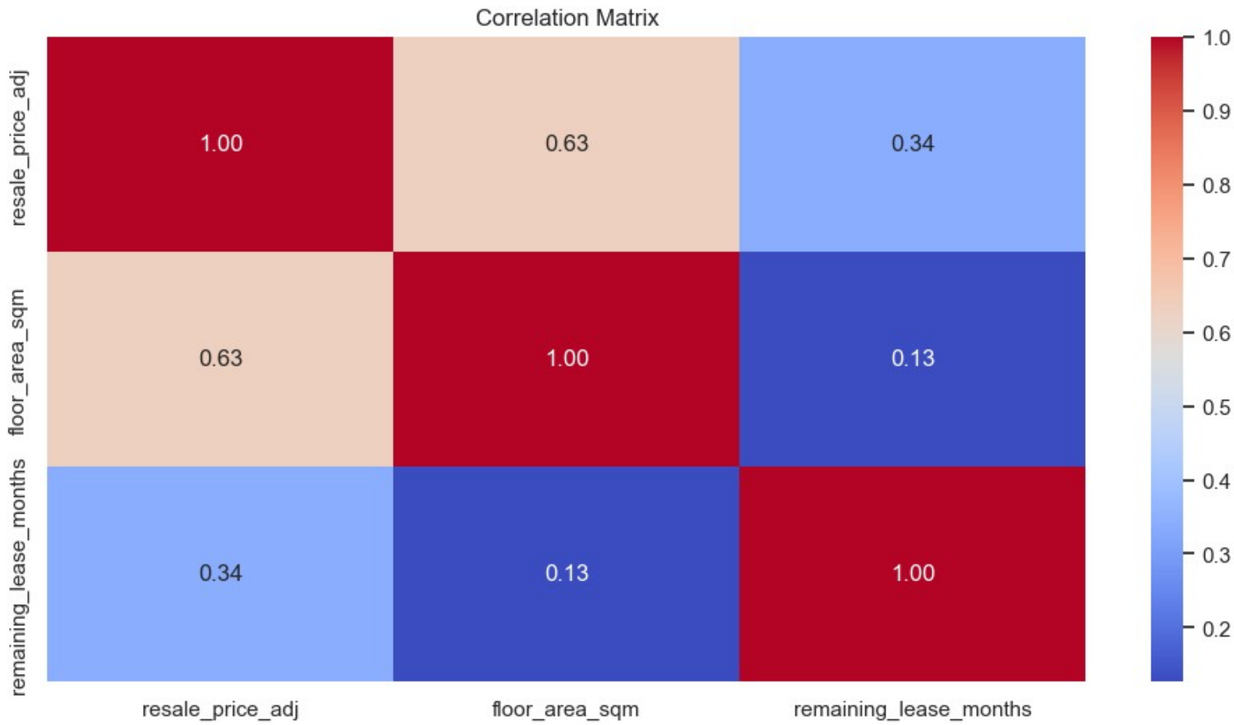
```
       floor_area_sqm  lease_commence_date  resale_price    year
cpi  \
count         169,584              169,584       169,584 169,584
169,584
mean               97                1,996       491,214   2,020
89
std                24                   14       169,917       2
5
min                31                1,966       140,000   2,017
85
```

```
25%                 82                 1,985       365,000   2,019
86
50%                 93                 1,996       460,000   2,020
86
75%                112                 2,009       585,000   2,022
93
max                249                 2,022     1,500,000   2,024
100

       resale_price_adj  avg_storey  remaining_lease_months
count           169,584     169,584                 169,584
mean            550,367           9                     896
std             184,304           6                     166
min             153,585           2                     504
25%             415,808           5                     761
50%             518,180           8                     895
75%             645,988          11                   1,056
max           1,549,310          50                   1,173
```

EDA Numeric Data

The numerical variables for EDA are: resale_price_adj, floor_area_sqm, remaining_lease_months.

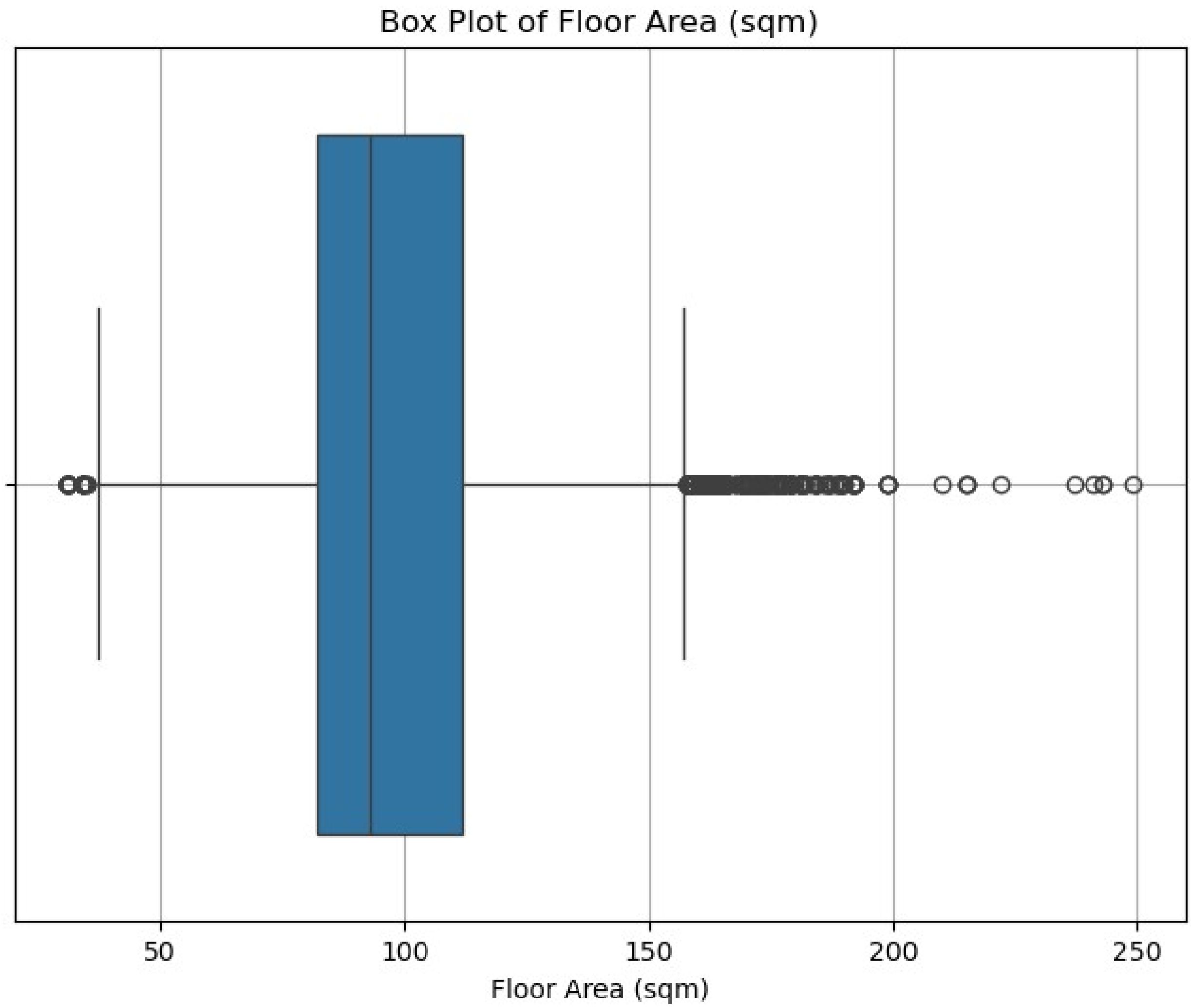
```
df_numeric = df[['resale_price_adj', 'floor_area_sqm',
'remaining_lease_months']]
plt.figure(figsize=(12, 6))
sns.heatmap(df_numeric.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Matrix")
plt.show()
```



Insights: Positive Correlations
floor_area_sqm and resale_price (0.60): Larger floor areas tend to have higher resale prices.
remaining_lease_months and resale_price (0.33): Newer properties generally have higher resale prices.

EDA: floor_area_sqm Uni-Variate

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
sns.boxplot(x=df['floor_area_sqm'])

plt.title('Box Plot of Floor Area (sqm)')
plt.xlabel('Floor Area (sqm)')
plt.grid(True)
plt.show()
```



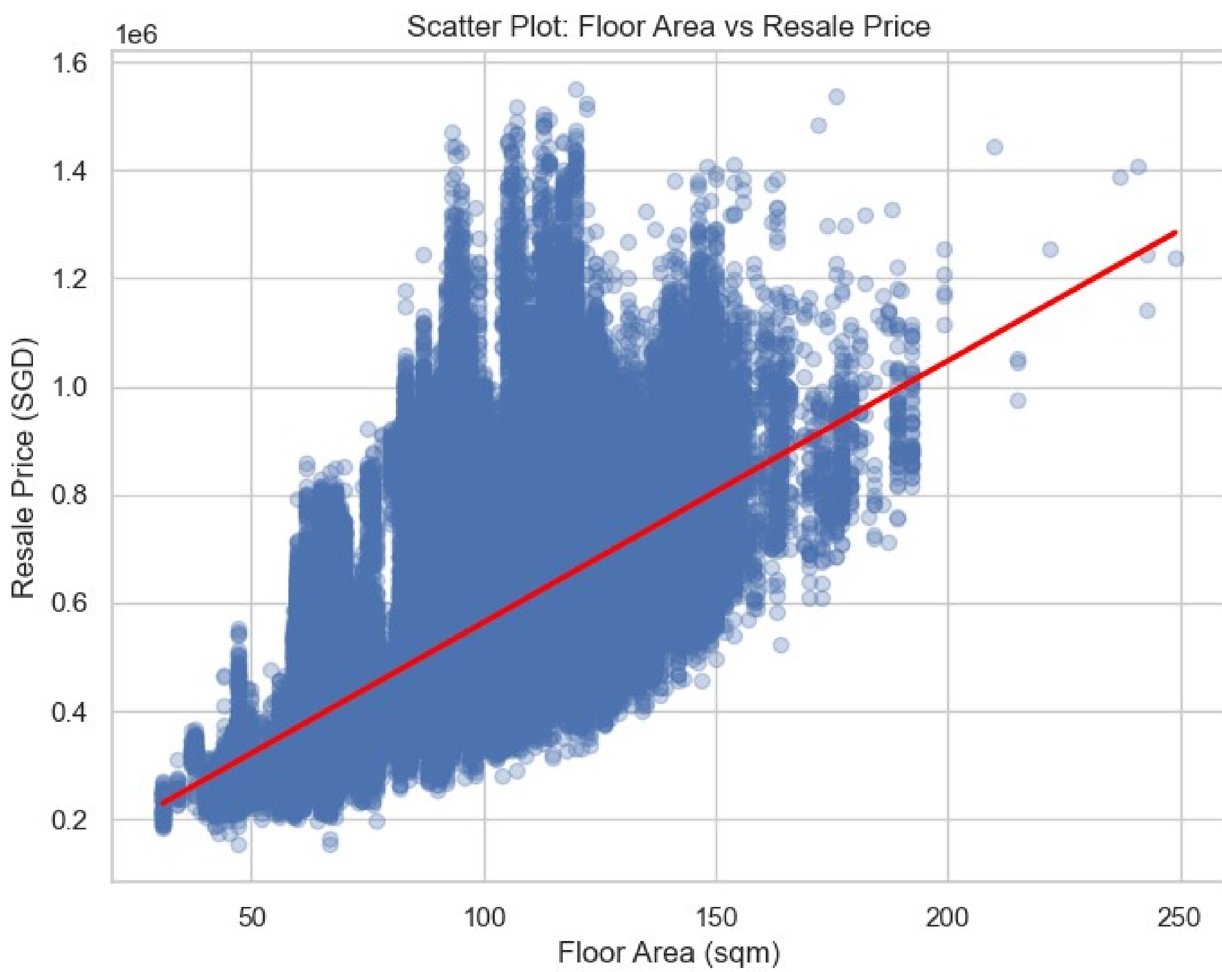
EDA: floor_area_sqm Bi-Variate and Multi-Variate

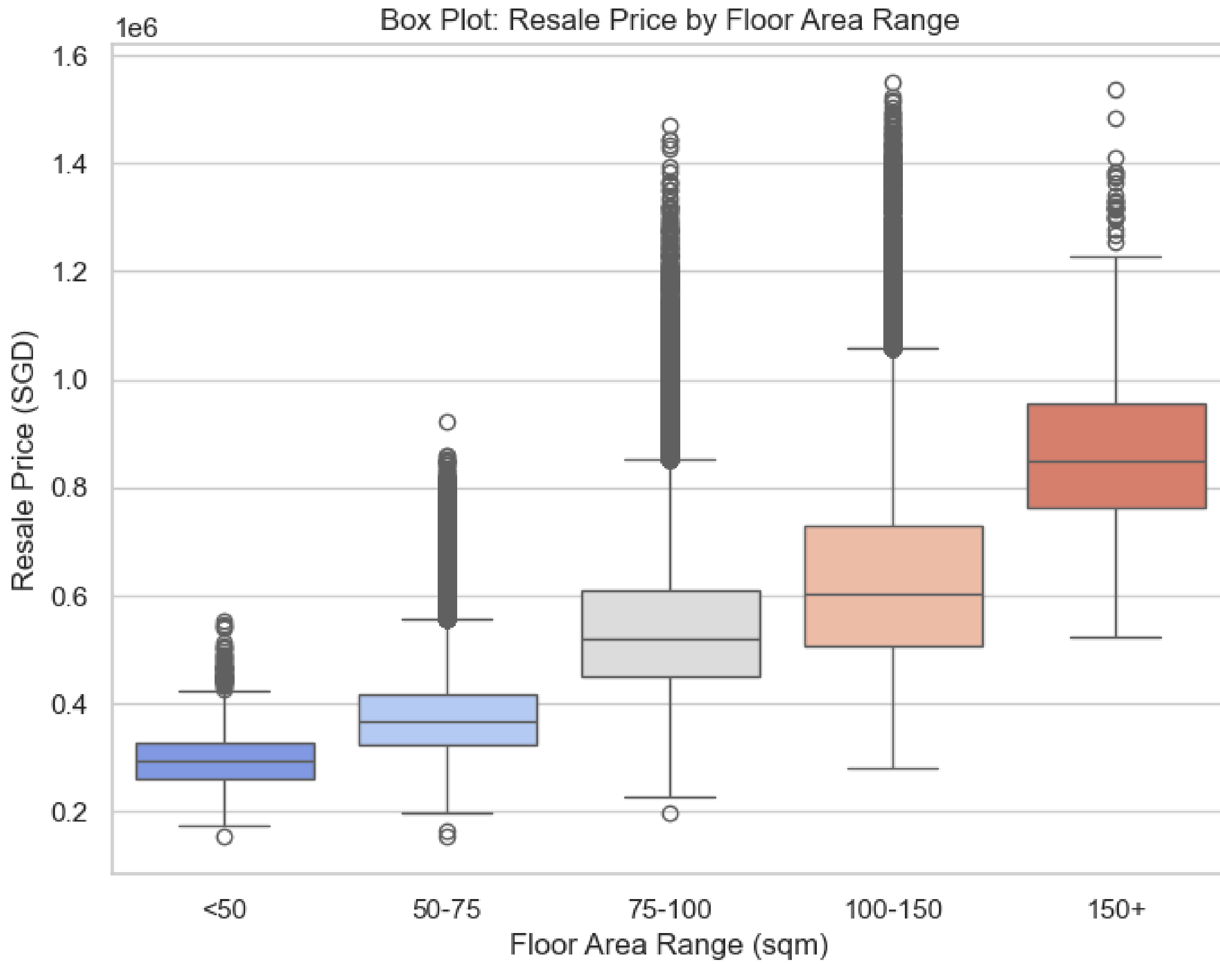
The Histograms are Uni-Variate
Dual Y-Axis Histogram is the multi-variate eda on floor_area_sqm

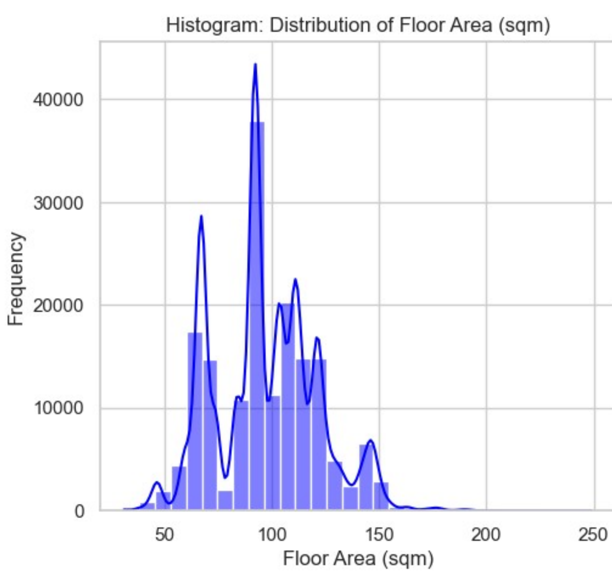
```
sns.set_style("whitegrid")

# Scatter Plot
plt.figure(figsize=(8, 6))
sns.regplot(x=df["floor_area_sqm"], y=df["resale_price_adj"],
scatter_kws={"alpha": 0.3}, line_kws={"color": "red"})
plt.title("Scatter Plot: Floor Area vs Resale Price")
plt.xlabel("Floor Area (sqm)")
plt.ylabel("Resale Price (SGD)")
plt.show()

# Box Plot
df["floor_area_range"] = pd.cut(df["floor_area_sqm"], bins=[0, 50, 75,
```

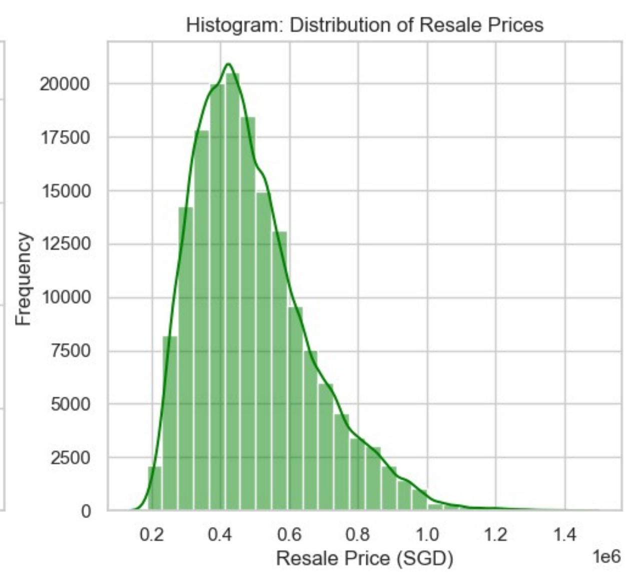
```
100, 150, 200], labels=["<50", "50-75", "75-100", "100-150", "150+"])
plt.figure(figsize=(8, 6))
sns.boxplot(x="floor_area_range", y="resale_price_adj", data=df,
hue="floor_area_range", palette="coolwarm", legend=False)
plt.title("Box Plot: Resale Price by Floor Area Range")
plt.xlabel("Floor Area Range (sqm)")
plt.ylabel("Resale Price (SGD)")
plt.show()

# Histograms
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
sns.histplot(df["floor_area_sqm"], bins=30, kde=True, ax=ax[0],
color="blue")
ax[0].set_title("Histogram: Distribution of Floor Area (sqm)")
ax[0].set_xlabel("Floor Area (sqm)")
ax[0].set_ylabel("Frequency")
sns.histplot(df["resale_price"], bins=30, kde=True, ax=ax[1],
color="green")
ax[1].set_title("Histogram: Distribution of Resale Prices")
ax[1].set_xlabel("Resale Price (SGD)")
ax[1].set_ylabel("Frequency")
plt.show()

# Floor Area stats
floor_min = df["floor_area_sqm"].min()
floor_max = df["floor_area_sqm"].max()
floor_mode = df["floor_area_sqm"].mode()[0]
floor_mode_count = df["floor_area_sqm"].value_counts().max()

print("□ Floor Area (sqm):")
print(f" - Min: {floor_min}")
print(f" - Max: {floor_max}")
print(f" - Most frequent value: {floor_mode} sqm (appears
{floor_mode_count} times)\n")

# Resale Price stats
price_min = df["resale_price"].min()
price_max = df["resale_price"].max()
price_mode = df["resale_price"].mode()[0]
price_mode_count = df["resale_price"].value_counts().max()

print("□ Resale Price (SGD):")
print(f" - Min: ${price_min:,.0f}")
print(f" - Max: ${price_max:,.0f}")
print(f" - Most frequent value: ${price_mode:,.0f} (appears
{price_mode_count} times)")

# Dual Y-Axis Histogram, Combining the Above
fig, ax1 = plt.subplots(figsize=(8, 6))
ax2 = ax1.twinx()
```

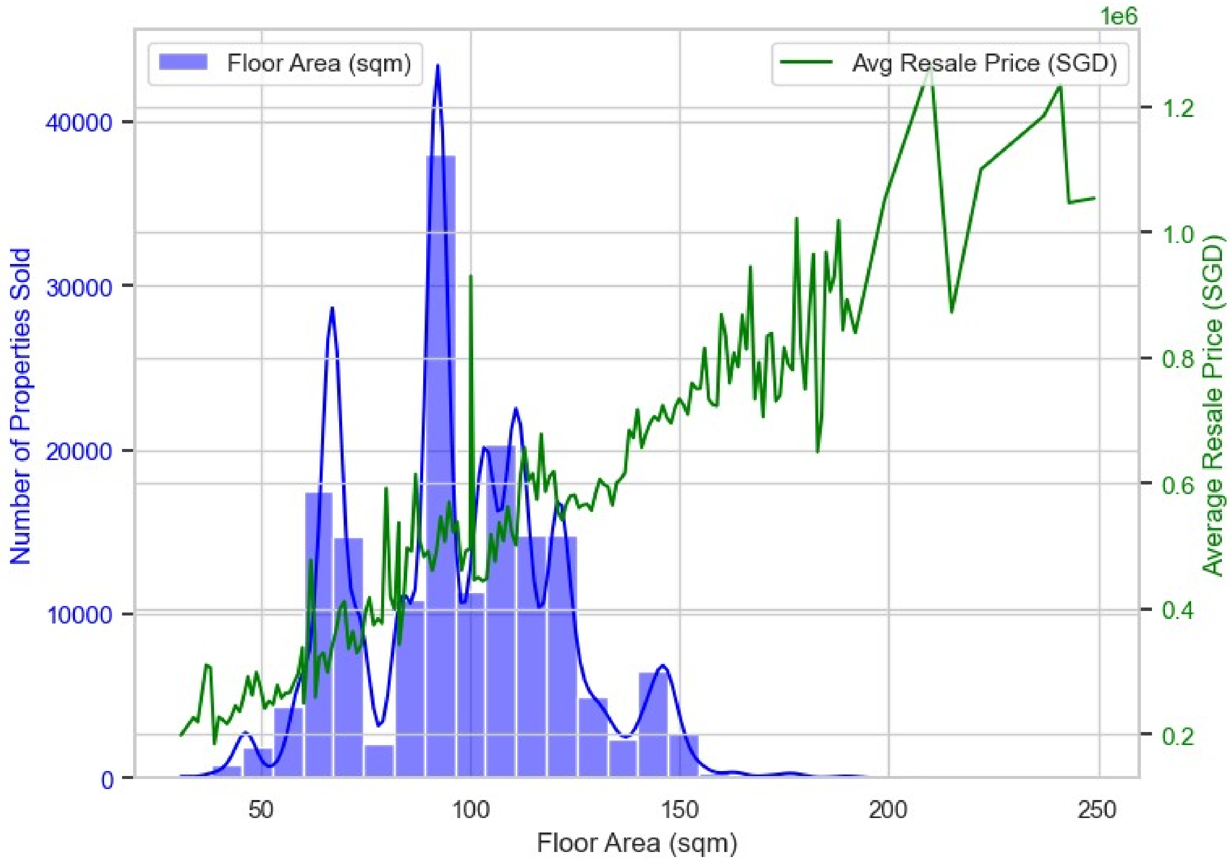
```
sns.histplot(df["floor_area_sqm"], bins=30, kde=True, color="blue",
label="Floor Area (sqm)", alpha=0.5, ax=ax1)
ax1.set_xlabel("Floor Area (sqm)")
ax1.set_ylabel("Number of Properties Sold", color="blue")
ax1.tick_params(axis='y', labelcolor="blue")

sns.lineplot(x=df["floor_area_sqm"], y=df.groupby("floor_area_sqm")
["resale_price"].transform("mean"), color="green", label="Avg Resale
Price (SGD)", ax=ax2)
ax2.set_ylabel("Average Resale Price (SGD)", color="green")
ax2.tick_params(axis='y', labelcolor="green")

ax1.legend(loc="upper left")
ax2.legend(loc="upper right")
plt.show()
```




Box Plot: Resale Price by Floor Area Range
1e6
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
Resale Price (SGD)
<50
50-75
75-100
100-150
150+
Floor Area Range (sqm)


Histogram: Distribution of Floor Area (sqm)
40000
30000
20000
10000
0
Frequency
50
100
150
200
250
Floor Area (sqm)


Histogram: Distribution of Resale Prices
20000
17500
15000
12500
10000
7500
5000
2500
0
Frequency
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6
Resale Price (SGD)



Insights:
The average resale price for the dual-axis histrogram = the mean resale price of all transactions with that specific size.

Volume of Sales vs. Floor Area (Blue Bars & Line)
Most Common Floor Areas:
The majority of resale flats fall between 80–110 sqm, with strong peaks around 90 sqm and 100 sqm — likely representing standard [*4-room*](https://www.dollarsandsense.sg/hdb-4-room-flats-singapore/#:~:text=90%20(2000%20%2DPresent)%20%E2%80%93%20100%20(1998%20%E2%80%93%202000) and 5-room flats.

Sharp Drop-Off:
After 130–140 sqm, the number of units sold drops significantly. Larger flats (e.g., *executive flats, maisonettes*) are less common in the market, possibly due to limited supply or niche demand.

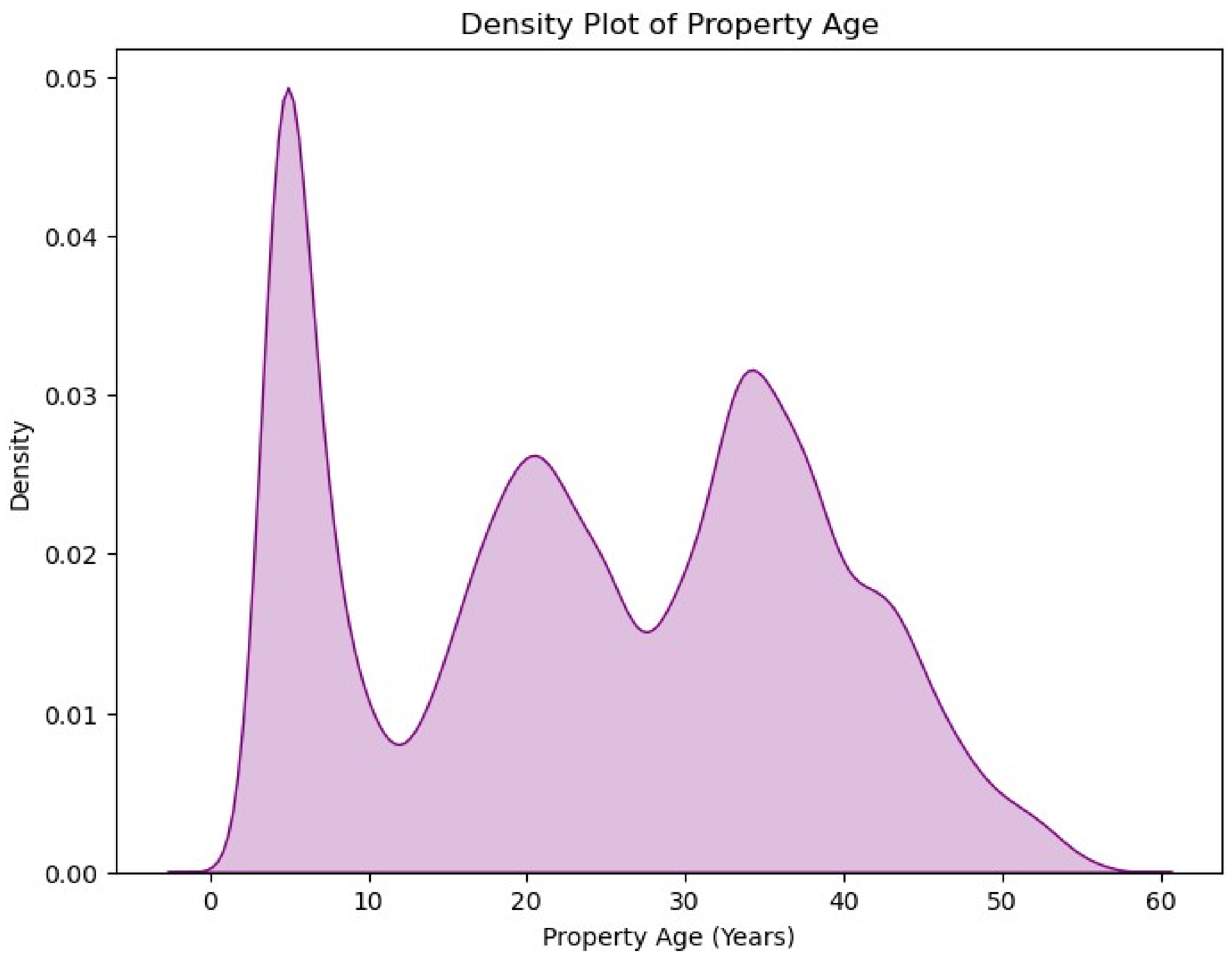
Very Low Sales for Very Small Units (<40 sqm):
These are likely 1-room or studio flats, which are rarely transacted and cater to a smaller population group (e.g., singles or elderly).

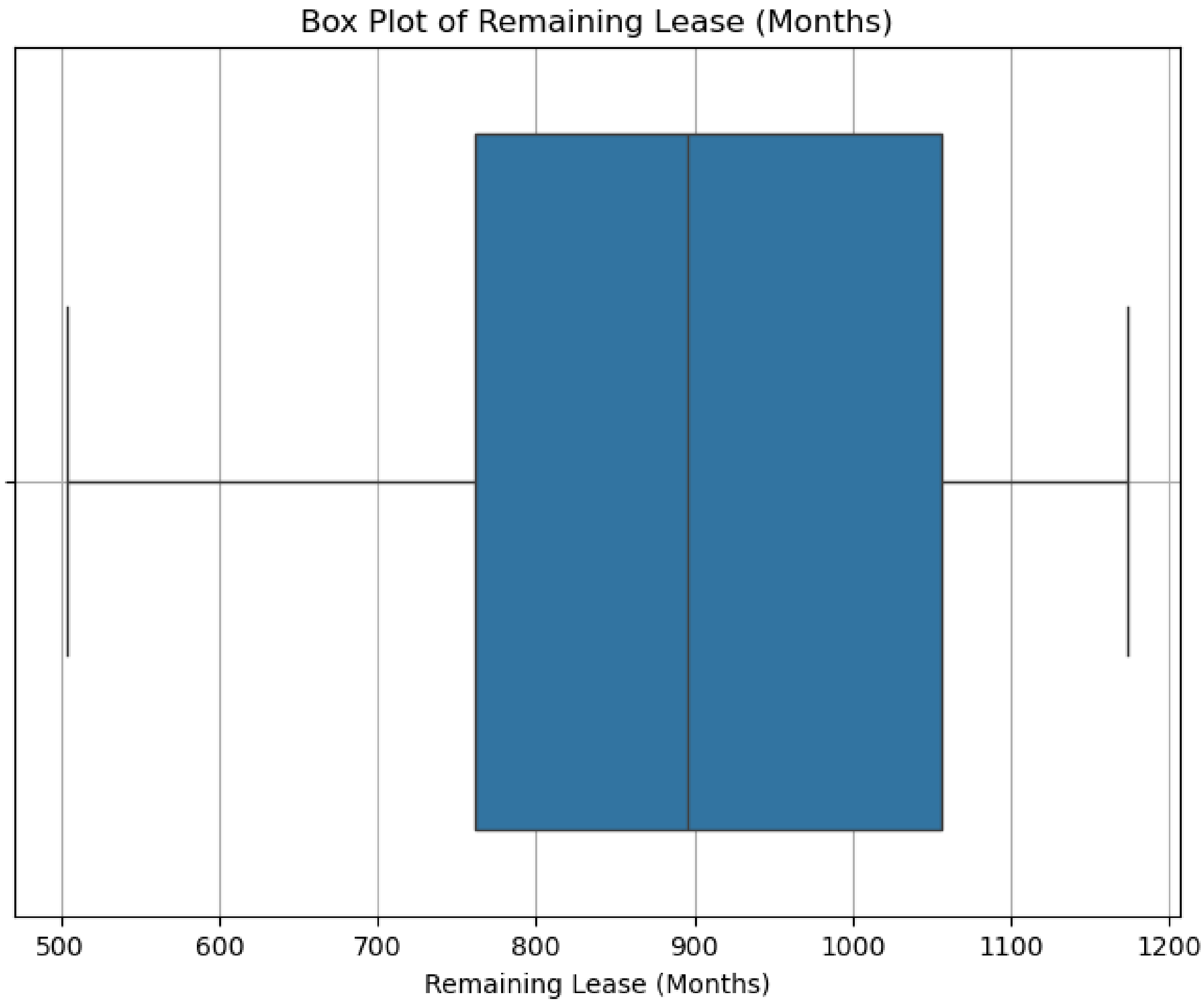
Interesting Interplay
High Preference for More Space ≠ Highest Price:
The floor area with the most transactions (around 90–100 sqm) does not correspond with the highest average prices, suggesting *demand-driven* volume rather than value-per-unit-size.

Scarcity Commands Price:
Even though larger flats are rare, they command higher average prices — possibly due to exclusive flat types, low supply, or niche buyer segments (e.g., multi-gen families).

EDA: lease_commence_date -> property age Uni-Variate

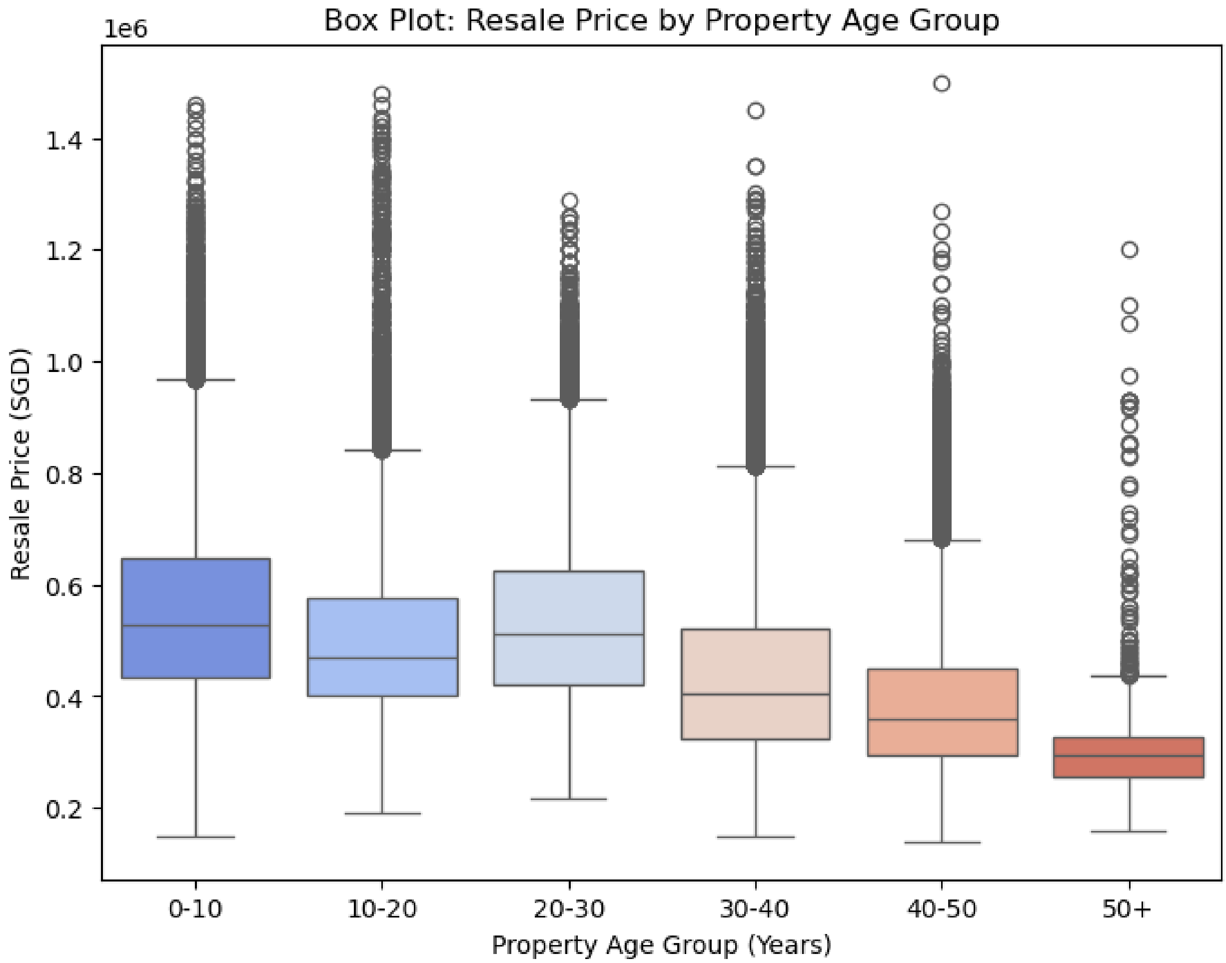
```
df["property_age"] = df['year'] - df["lease_commence_date"]

df["property_age_group"] = pd.cut(df["property_age"], bins=[0, 10, 20, 30, 40, 50, 60],
                                   labels=["0-10", "10-20", "20-30", "30-40", "40-50", "50+"])
plt.figure(figsize=(8, 6))
sns.kdeplot(df["property_age"], fill=True, color="purple")
plt.title("Density Plot of Property Age")
plt.xlabel("Property Age (Years)")
plt.show()

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
sns.boxplot(x=df['remaining_lease_months'])

plt.title('Box Plot of Remaining Lease (Months)')
plt.xlabel('Remaining Lease (Months)')
plt.grid(True)
plt.show()
```


Density Plot of Property Age
0.05
0.04
0.03
0.02
0.01
0.00
Density
0
10
20
30
40
50
60
Property Age (Years)



EDA: Property Age, Bi-Variate

```
plt.figure(figsize=(8, 6))
sns.boxplot(x="property_age_group", y="resale_price",
hue="property_age_group", data=df, palette="coolwarm", legend=False)
plt.title("Box Plot: Resale Price by Property Age Group")
plt.xlabel("Property Age Group (Years)")
plt.ylabel("Resale Price (SGD)")
plt.show()
```




```
plt.figure(figsize=(8, 6))
sns.regplot(x=df["property_age"], y=df["resale_price_adj"],
scatter_kws={"alpha": 0.3}, line_kws={"color": "red"})
plt.title("Scatter Plot: Property Age vs Resale Price")
plt.xlabel("Property Age (Years)")
plt.ylabel("Resale Price (SGD)")
plt.show()
```

Insights:

1. Resale Prices Decline with Property Age (RED LINE)
   - The red line (right y-axis) shows that **average resale prices tend to decrease as property age increases**.

   - Newer properties (0-10 years old) have significantly higher prices, while older properties (40+ years) have lower resale values.
2. Number of Properties Sold is High for Both New & Old Properties (BLUE BARS)
   - The blue bars (left y-axis) show **high transaction volumes for very new properties (0-10 years)** and **older properties (30-50 years)**.

   - This suggests that **new properties are in demand**, and **many old flats are being resold, possibly due to lease expiry concerns**.
3. Volatility in Price Trends for Younger Properties
   - The resale price trend (red line) fluctuates heavily in the **0-20 year range**, which could indicate **variability in demand for newer properties** due to factors like location, condition, or government policies.
4. Steady Decline in Prices for Older Properties

- Beyond 30 years, prices show a steady downward trend, likely due to factors like **lease decay (for leasehold properties)** and **aging infrastructure**. Actually, our EDA seems to support lease decay, but upon further analysis through linear and non-linear regression, the factor of remaining lease turns out not to be as strong as we thought.

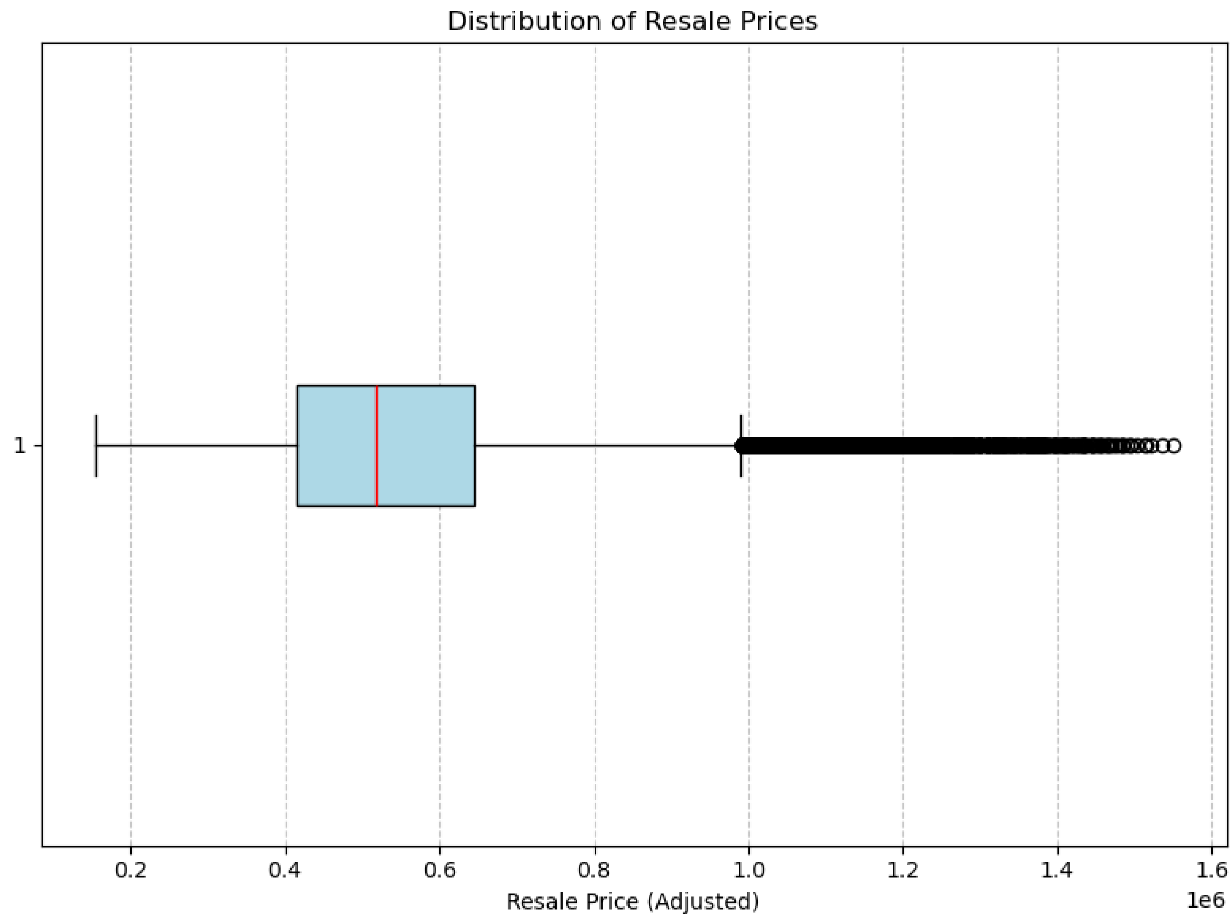
Conclusion:

- Buyers may prefer newer properties for **higher value retention**, while older properties continue to be transacted despite declining prices.

- The **spike in transactions for old properties** suggests many homeowners may be selling before lease decay significantly impacts value.

EDA: Resale Price, Uni-Variate

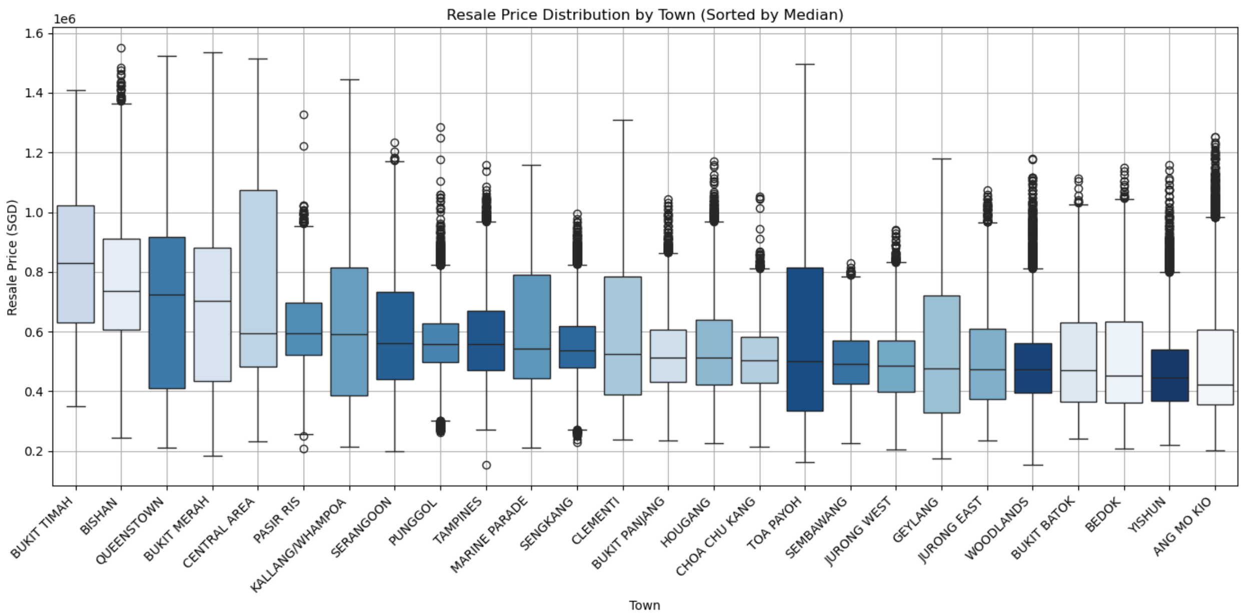
```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.boxplot(df['resale_price_adj'], vert=False, patch_artist=True,
            boxprops=dict(facecolor='lightblue', color='black'),
            medianprops=dict(color='red'))

plt.xlabel("Resale Price (Adjusted)")
plt.title("Distribution of Resale Prices")
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



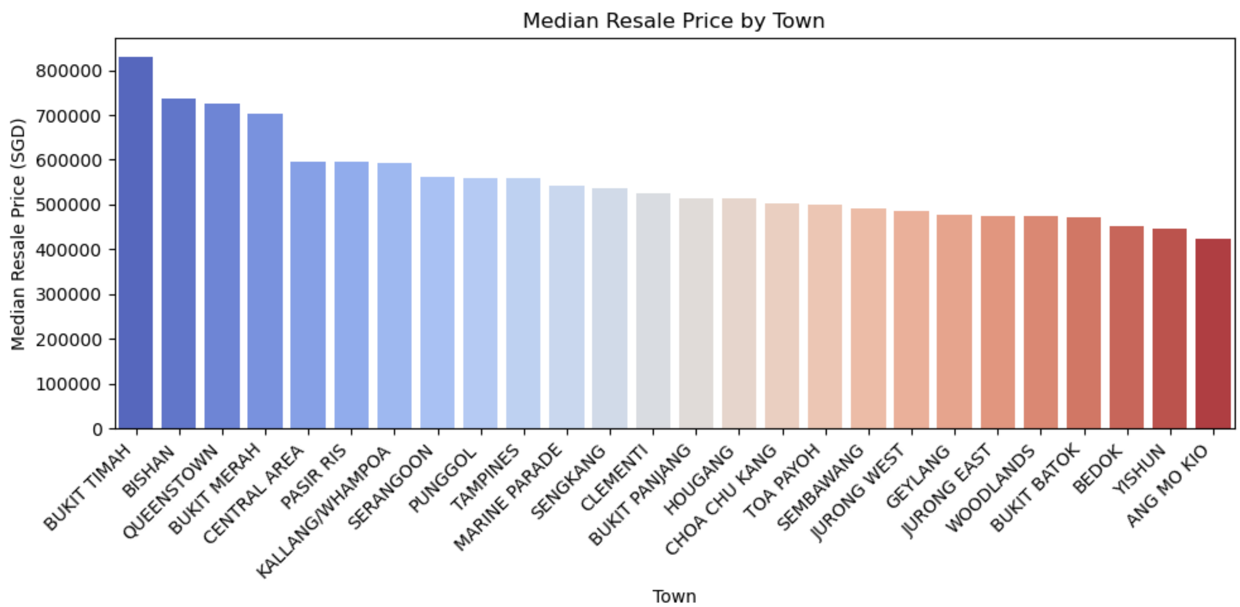
EDA: Resale Price, Bi-Variate

```
df["town"] = df["town"].astype(str).str.upper()
median_prices = df.groupby("town")
["resale_price_adj"].median().sort_values(ascending=False)

# Boxplot
plt.figure(figsize=(14, 7))
sns.boxplot(
    data=df,
    x="town",
    y="resale_price_adj",
    order=median_prices.index,
    hue="town",
    palette="Blues",
    legend=False
)
plt.xticks(rotation=45, ha='right')
plt.title("Resale Price Distribution by Town (Sorted by Median)")
plt.xlabel("Town")
plt.ylabel("Resale Price (SGD)")
plt.grid(True)
```

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

highest_towns = median_prices.head(5)
lowest_towns = median_prices.tail(5)

print("Top 5 Most Expensive Towns (Median Resale Price):")
print(highest_towns)

print("\nTop 5 Most Affordable Towns (Median Resale Price):")
print(lowest_towns)

price_range = df.groupby("town")["resale_price_adj"].agg(lambda x:
x.max() - x.min()).sort_values(ascending=False)

print("\nTowns with the Largest Resale Price Range:")
print(price_range.head(5))

median_prices_df = median_prices.reset_index()
median_prices_df.columns = ['town', 'median_resale_price']

plt.figure(figsize=(10, 5))
sns.barplot(
    data=median_prices_df,
    x="town",
    y="median_resale_price",
    hue="town",
    palette="coolwarm",
    legend=False
)
plt.xticks(rotation=45, ha='right')
plt.title("Median Resale Price by Town")
plt.xlabel("Town")
plt.ylabel("Median Resale Price (SGD)")
plt.tight_layout()
plt.show()
```




```
Top 5 Most Expensive Towns (Median Resale Price):
town
BUKIT TIMAH    829,748
BISHAN         737,206
QUEENSTOWN     724,537
BUKIT MERAH    703,069
CENTRAL AREA   595,091
Name: resale_price_adj, dtype: float64

Top 5 Most Affordable Towns (Median Resale Price):
town
WOODLANDS     473,915
BUKIT BATOK   470,993
BEDOK         452,246
YISHUN        445,426
ANG MO KIO    423,111
Name: resale_price_adj, dtype: float64

Towns with the Largest Resale Price Range:
town
BUKIT MERAH    1,352,850
TOA PAYOH      1,331,709
QUEENSTOWN     1,309,927
BISHAN         1,304,538
CENTRAL AREA   1,282,252
Name: resale_price_adj, dtype: float64
```



Insights from the Box Plot of Resale Prices by Town:

1. Price Variability Across Towns
   - Some towns (e.g., Bukit Timah, Central Area, Queenstown) have much higher median resale prices compared to others. These areas likely contain premium properties or more desirable housing.
2. Presence of Outliers
   - Almost all towns have resale price outliers, particularly on the higher end. This suggests that while most properties fall within a typical range, there are some high-value transactions.
3. Towns with Lower Median Prices
   - Some towns (e.g., Bukit Panjang, Choa Chu Kang, Woodlands, Yishun) have relatively lower median prices. These areas might offer more affordable housing options.
4. Wider Price Range in Certain Towns
   - The Central Area and Bukit Timah have a very large range of resale prices. This suggests that these locations may have a mix of high-end and mid-range properties.
5. Towns with More Stable Pricing
   - Locations like Sengkang, Punggol, and Pasir Ris show a narrower price range, meaning resale prices are more stable in these areas.

Potential Conclusions

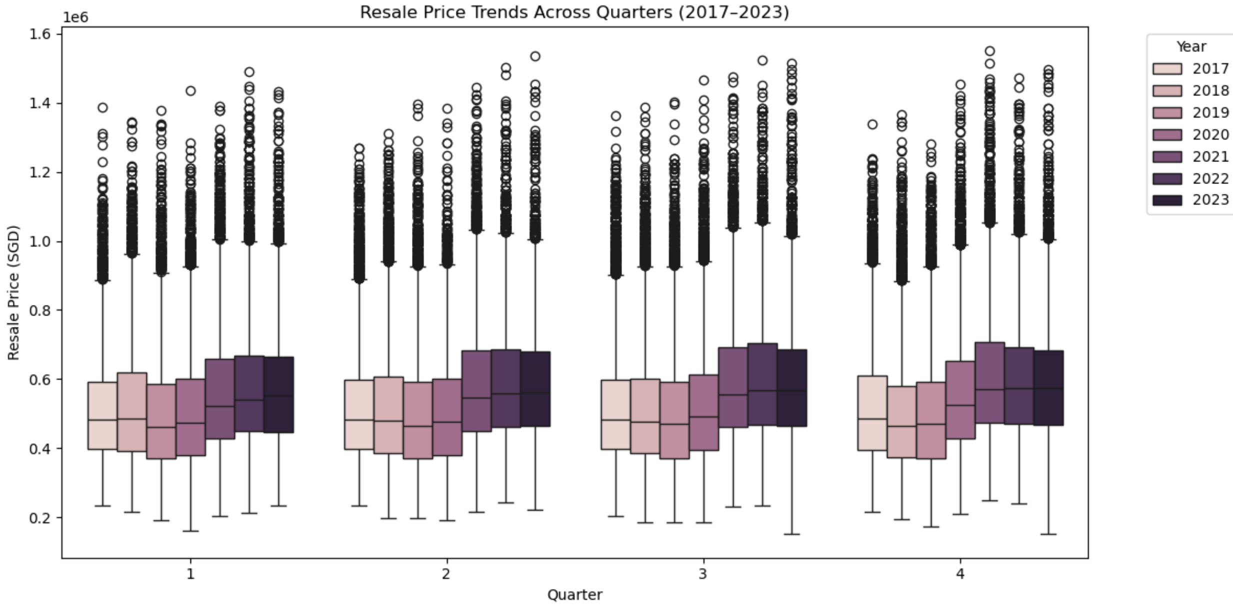
- Affluent areas (e.g., Bukit Timah, Queenstown, Central Area) have high resale prices due to demand and exclusivity.

- Some towns offer affordable housing options (e.g., Woodlands, Bukit Panjang, Choa Chu Kang).

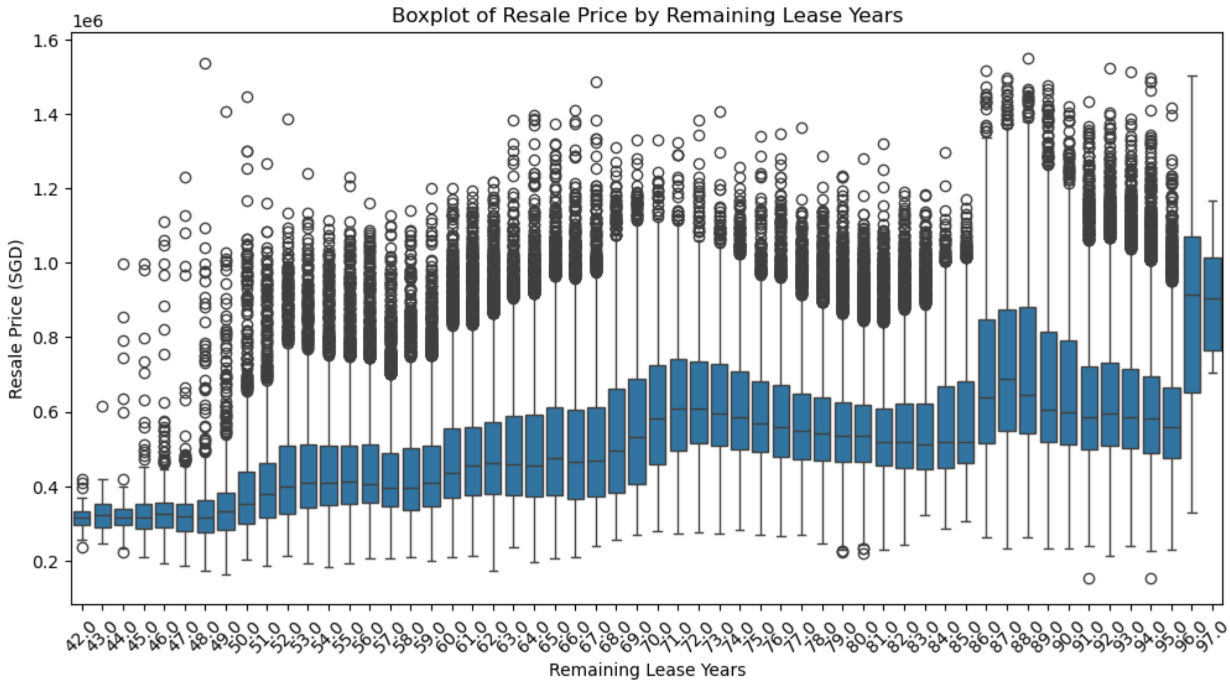
- High-price outliers may indicate luxury developments or recently renovated properties in some areas.

EDA : Time and Resale Price

```
# Quarterly
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df['month'] = pd.to_datetime(df['month'], errors='coerce')
df['Year'] = df['month'].dt.year
df['Quarter'] = df['month'].dt.quarter
df_filtered = df[df['Year'].between(2017, 2023)]
df_filtered = df_filtered.dropna(subset=['resale_price_adj', 'Year',
'Quarter'])
plt.figure(figsize=(12, 6))
plot = sns.boxplot(x='Quarter', y='resale_price_adj', hue='Year',
data=df_filtered)
handles, labels = plot.get_legend_handles_labels()
if handles:
    plt.legend(title="Year", bbox_to_anchor=(1.05, 1), loc='upper
left')
else:
    print("Warning: No legend entries found.")

plt.title("Resale Price Trends Across Quarters (2017–2023)")
plt.xlabel("Quarter")
plt.ylabel("Resale Price (SGD)")
plt.tight_layout()
plt.show()
```

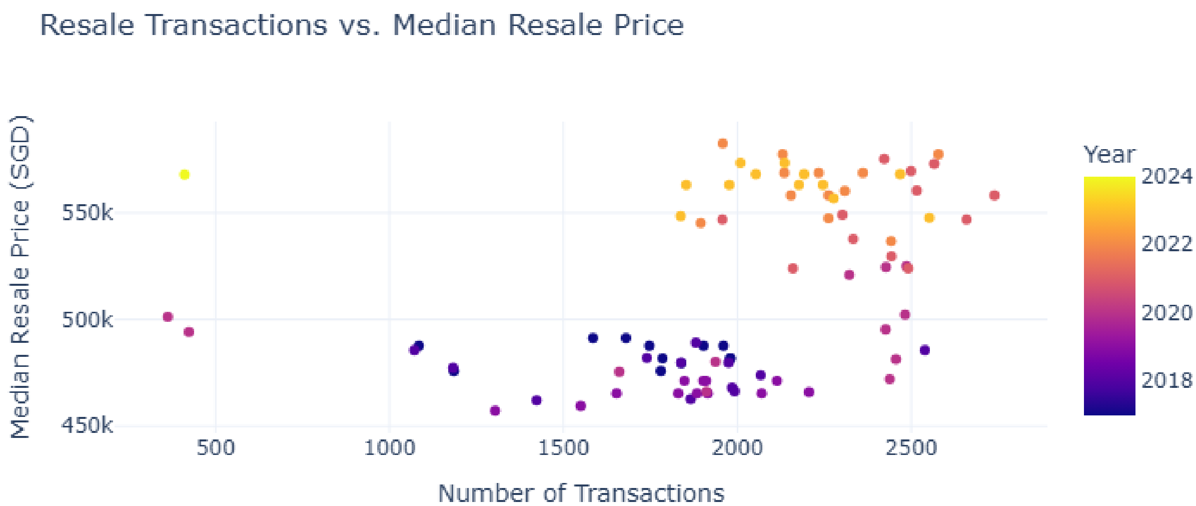


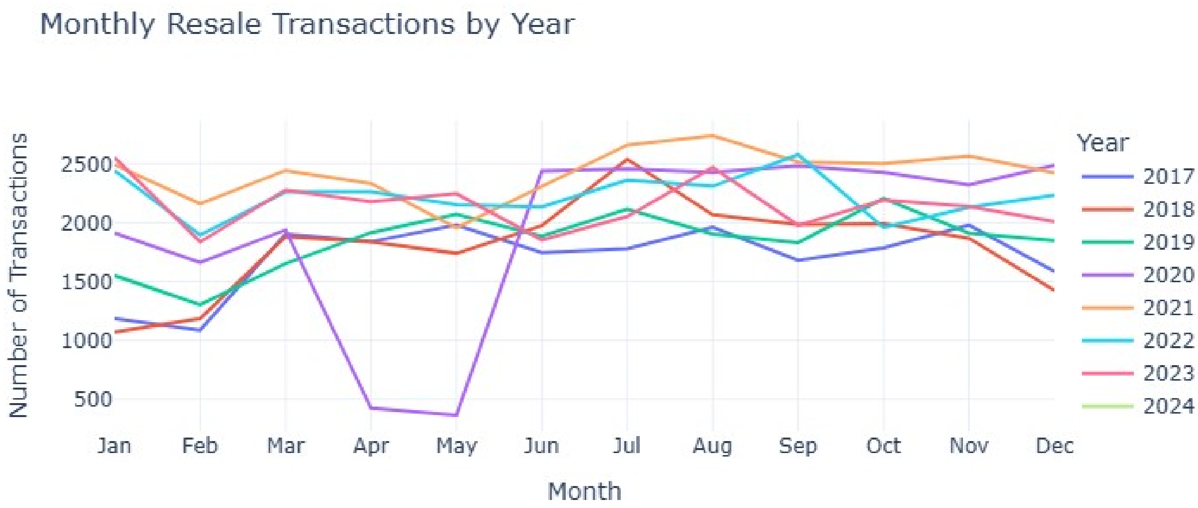

```python
# Boxplot:`remaining_lease_years`
df['remaining_lease_years'] = df['remaining_lease'].str.extract(r'(\
d+)\s+years').astype(float)

plt.figure(figsize=(12, 6))
sns.boxplot(x='remaining_lease_years', y='resale_price_adj', data=df)
plt.title("Boxplot of Resale Price by Remaining Lease Years")
plt.xlabel("Remaining Lease Years")
plt.ylabel("Resale Price (SGD)")
plt.xticks(rotation=45)
plt.show()
```

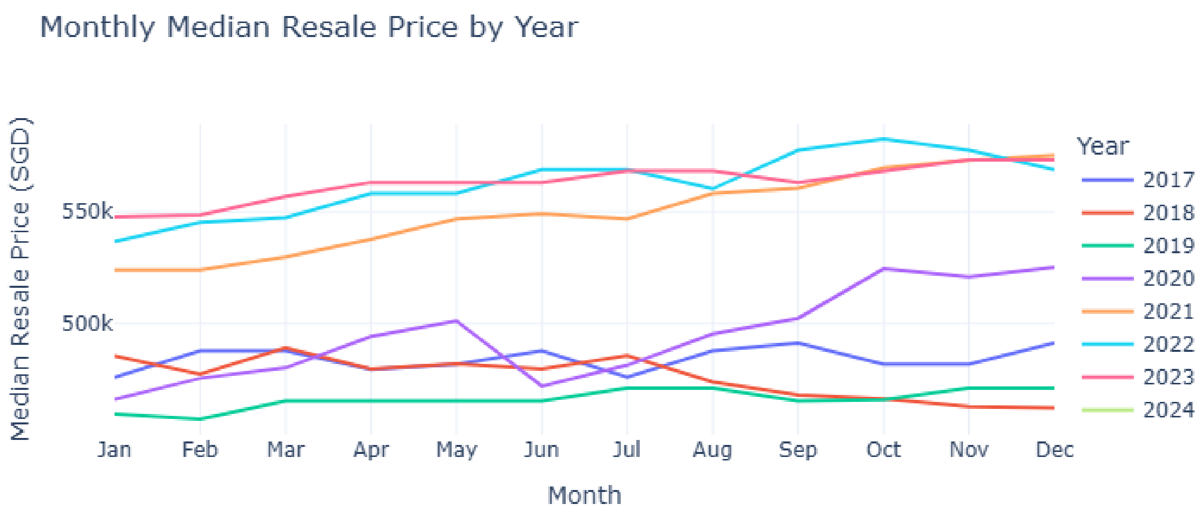



```python
df['Month'] = df['month'].dt.month
monthly_data = df.groupby(["Year", "Month"]).agg(
    resale_transactions=("resale_price_adj", "count"),
    median_resale_price=("resale_price_adj", "median")
).reset_index()
monthly_data["month_name"] = monthly_data["Month"].apply(lambda x:
pd.to_datetime(str(x), format='%m').strftime('%b'))

# scatter plot
df_scatter = px.scatter(
    monthly_data, x="resale_transactions", y="median_resale_price",
color="Year",
    title="Resale Transactions vs. Median Resale Price",
    labels={"resale_transactions": "Number of Transactions",
"median_resale_price": "Median Resale Price (SGD)", "Year": "Year"},
    template="plotly_white"
)

# separate line plots
fig_transactions = px.line(
    monthly_data, x="month_name", y="resale_transactions",
color="Year",
    title="Monthly Resale Transactions by Year",
    labels={"resale_transactions": "Number of Transactions",
"month_name": "Month", "Year": "Year"},
    template="plotly_white"
)
```

```
fig_price = px.line(
    monthly_data, x="month_name", y="median_resale_price",
color="Year",
    title="Monthly Median Resale Price by Year",
    labels={"median_resale_price": "Median Resale Price (SGD)",
"month_name": "Month", "Year": "Year"},
    template="plotly_white"
)

df_scatter.show()
fig_transactions.show()
fig_price.show()
```







Insights:

1. Resale Transactions vs. Median Resale Price (Top Chart)

- There is a positive correlation between the number of transactions and the median resale price, meaning as the number of transactions increases, resale prices also tend to be higher.
- Data points from earlier years (2018, 2019) tend to cluster at lower transaction counts and lower median prices.
- More recent years (2022) show higher median prices and higher transaction counts.
- The color gradient indicates that resale prices have generally increased over time, with newer years (yellow) having higher prices than older years (purple).

2. Monthly Resale Transactions by Year (Middle Chart)

- The number of transactions fluctuates throughout the year, often peaking around mid-year (June - September).
- 2020 shows an unusual dip around April and May, likely due to disruptions (possibly COVID-19 lockdowns).
- Transactions tend to be lower at the beginning and end of the year.
- The years 2022 and 2023 have higher transaction counts compared to earlier years like 2017 and 2018.

3. Monthly Median Resale Price by Year (Bottom Chart)

- Median resale prices have shown an increasing trend over the years.
- The years 2022 and 2023 have significantly higher median resale prices compared to earlier years.
- There is a steady increase in resale prices across all months, with minor fluctuations.
- 2020 shows a temporary dip, aligning with the transaction drop in the middle chart.

Overall Observations

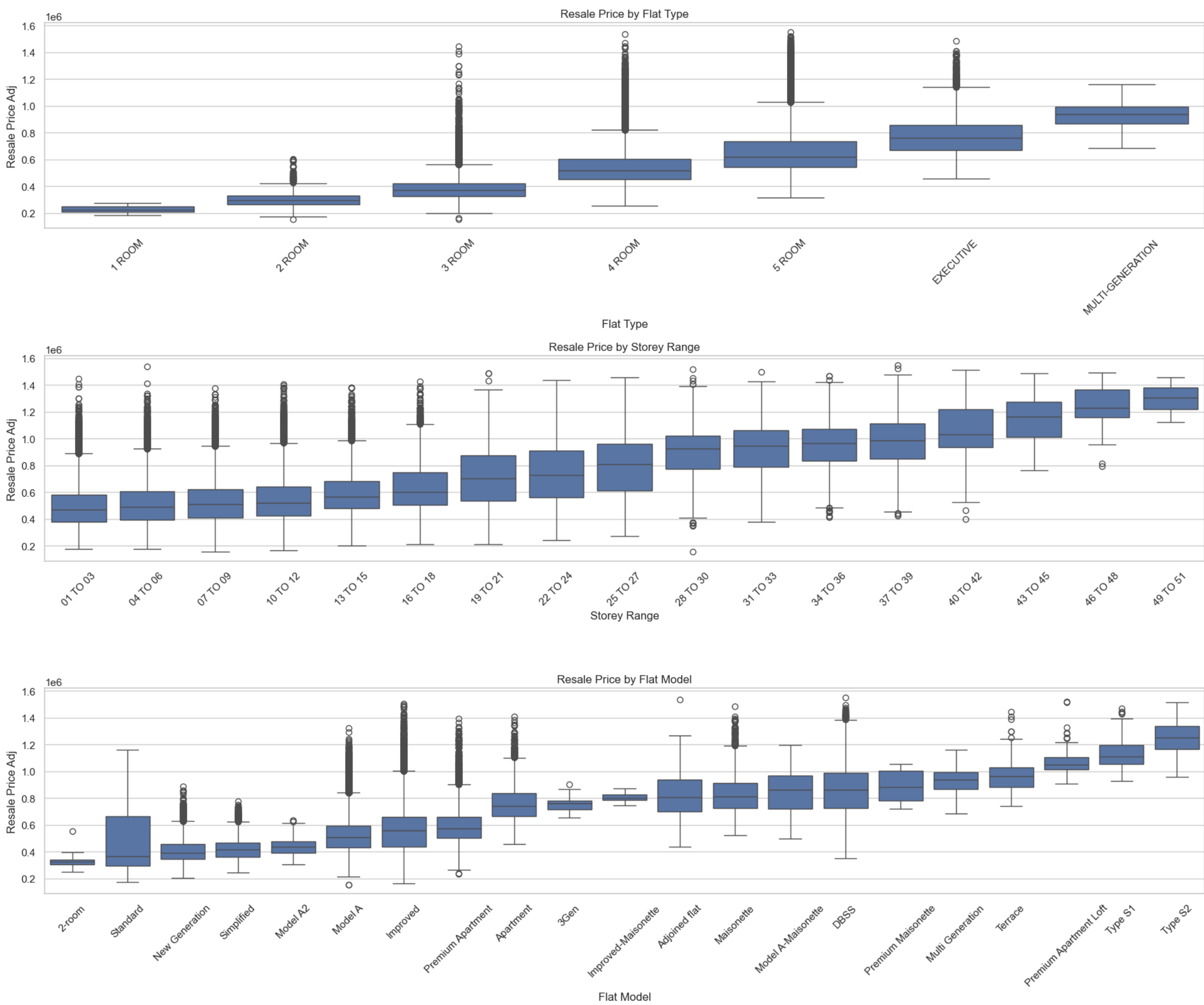
- Over the years, the median resale price has been steadily increasing.
- Transaction volumes show seasonal trends, often peaking in mid-year.
- The impact of external events (e.g., COVID-19 in 2020) is visible in transaction data but did not have a long-term negative effect on prices.
- More recent years (2022 and 2023) have seen both higher resale prices and higher transaction volumes, indicating a strong market.

EDA Multi-Variate

Flat Type, Storey Range, Flat Model VERSUS Resale Price

```
sns.set(style="whitegrid")

def ordered_boxplot(x, y, data, ax, title):
    order = data.groupby(x)[y].median().sort_values().index
    sns.boxplot(x=x, y=y, data=data, order=order, ax=ax)
    ax.set_title(title)
    ax.set_xlabel(x.replace('_', ' ').title())
    ax.set_ylabel(y.replace('_', ' ').title())
    ax.tick_params(axis='x', rotation=45)

fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(18, 15))
ordered_boxplot(x="flat_type", y="resale_price_adj", data=df, 
ax=axes[0], title="Resale Price by Flat Type")
ordered_boxplot(x="storey_range", y="resale_price_adj", data=df, 
ax=axes[1], title="Resale Price by Storey Range")
ordered_boxplot(x="flat_model", y="resale_price_adj", data=df, 
ax=axes[2], title="Resale Price by Flat Model")
```

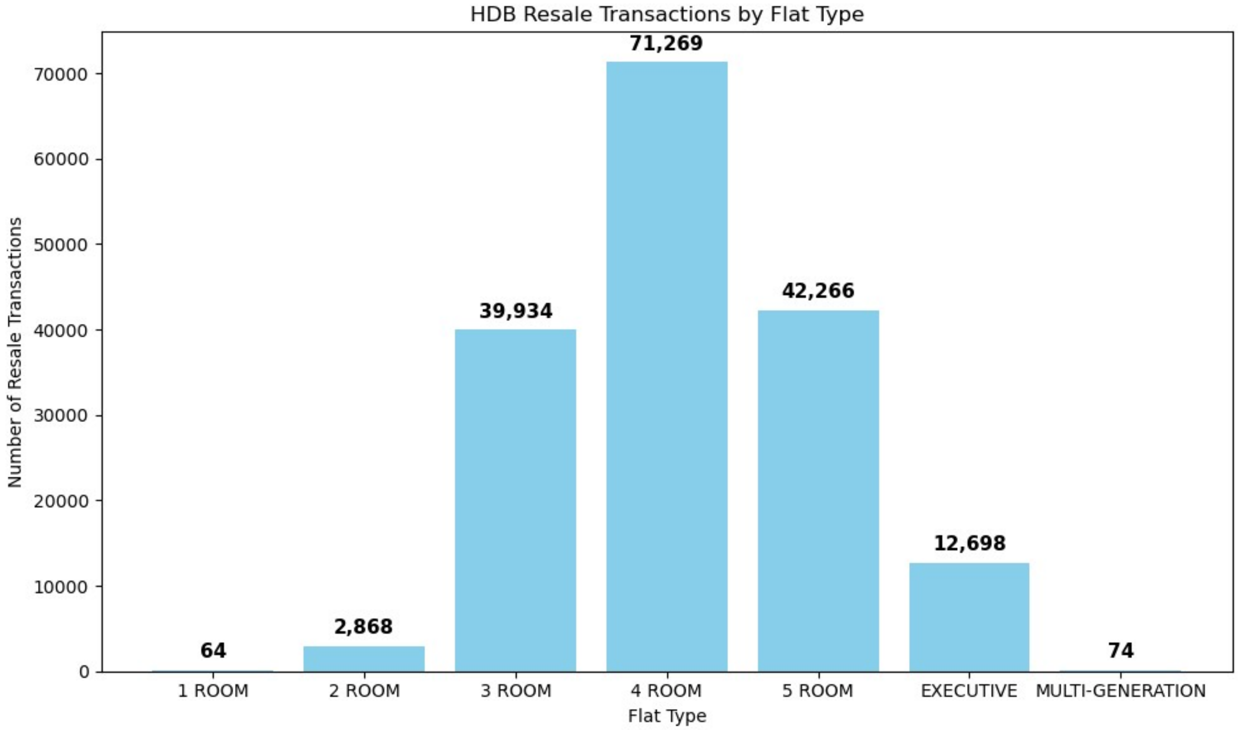
```
plt.tight_layout()
plt.show()
```

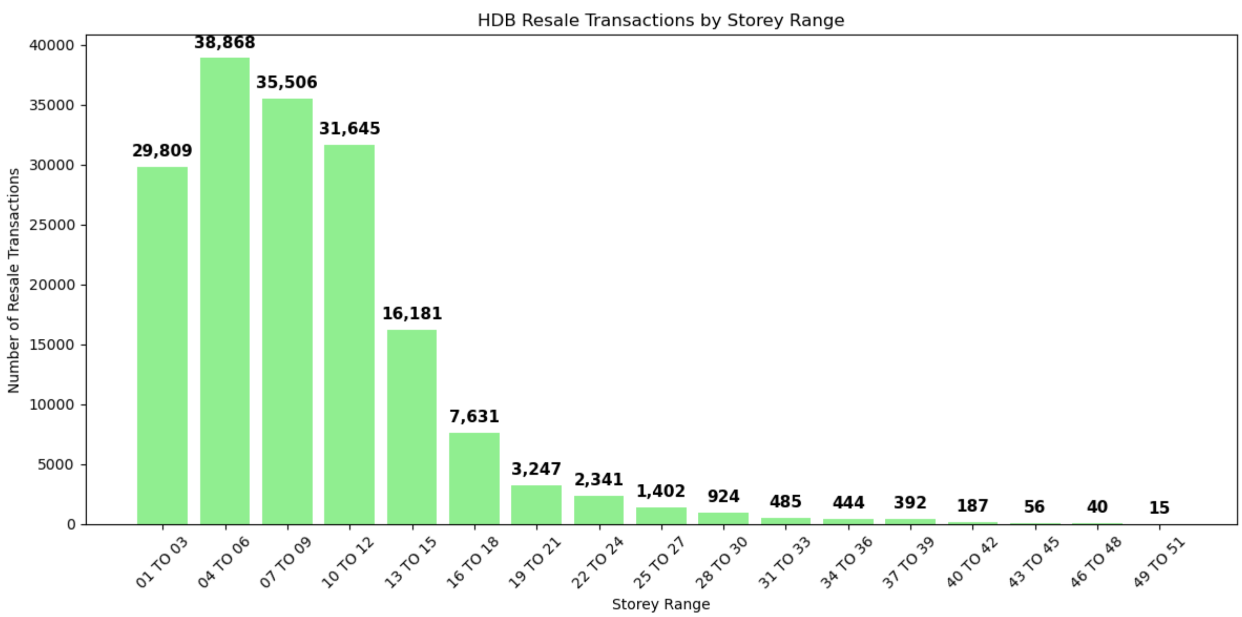


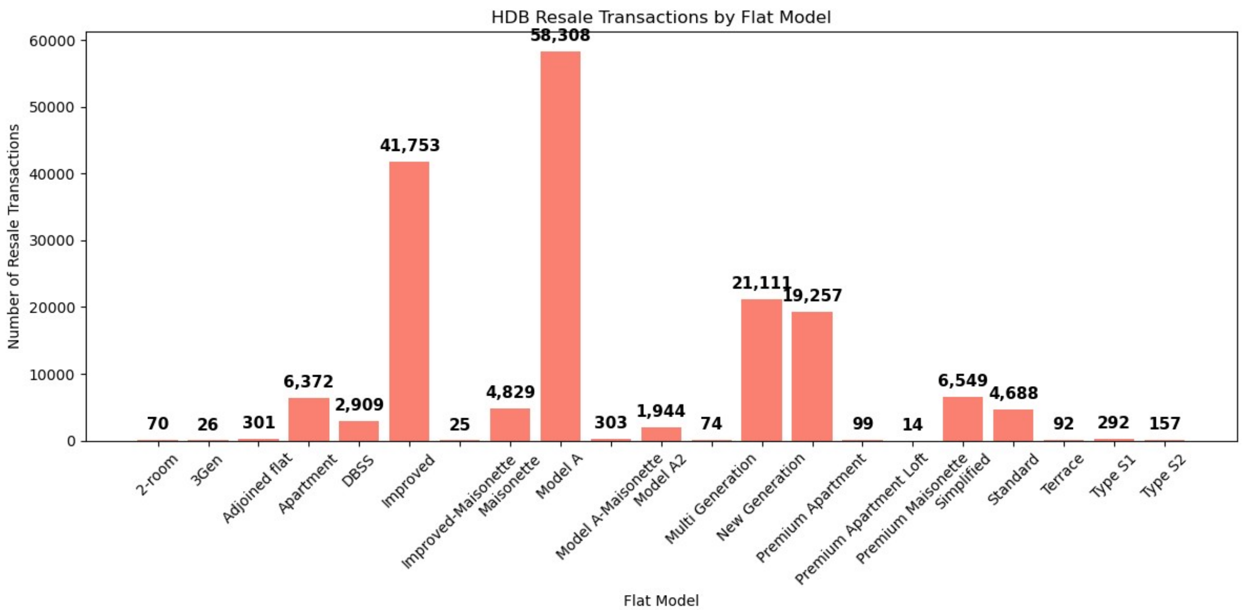

```python
import matplotlib.pyplot as plt

flat_type_counts = df['flat_type'].value_counts().sort_index()

fig, ax = plt.subplots(figsize=(10, 6))
bars = ax.bar(flat_type_counts.index, flat_type_counts.values,
color='skyblue')

for bar in bars:
    height = bar.get_height()
    ax.annotate(f'{height:,}',
                xy=(bar.get_x() + bar.get_width() / 2, height),
                xytext=(0, 5),
                textcoords='offset points',
                ha='center', va='bottom',
```

```
                fontsize=11, fontweight='bold', color='black')

ax.set_xlabel("Flat Type")
ax.set_ylabel("Number of Resale Transactions")
ax.set_title("HDB Resale Transactions by Flat Type")
plt.tight_layout()
plt.show()


storey_counts = df['storey_range'].value_counts().sort_index()

fig, ax = plt.subplots(figsize=(12, 6))
bars = ax.bar(storey_counts.index, storey_counts.values,
color='lightgreen')

for bar in bars:
    height = bar.get_height()
    ax.annotate(f'{height:,}',
                xy=(bar.get_x() + bar.get_width() / 2, height),
                xytext=(0, 5),
                textcoords='offset points',
                ha='center', va='bottom',
                fontsize=11, fontweight='bold')

ax.set_xlabel("Storey Range")
ax.set_ylabel("Number of Resale Transactions")
ax.set_title("HDB Resale Transactions by Storey Range")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

flat_model_counts = df['flat_model'].value_counts().sort_index()

fig, ax = plt.subplots(figsize=(12, 6))
bars = ax.bar(flat_model_counts.index, flat_model_counts.values,
color='salmon')

for bar in bars:
    height = bar.get_height()
    ax.annotate(f'{height:,}',
                xy=(bar.get_x() + bar.get_width() / 2, height),
                xytext=(0, 5),
                textcoords='offset points',
                ha='center', va='bottom',
                fontsize=11, fontweight='bold')

ax.set_xlabel("Flat Model")
ax.set_ylabel("Number of Resale Transactions")
ax.set_title("HDB Resale Transactions by Flat Model")
plt.xticks(rotation=45)
```

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



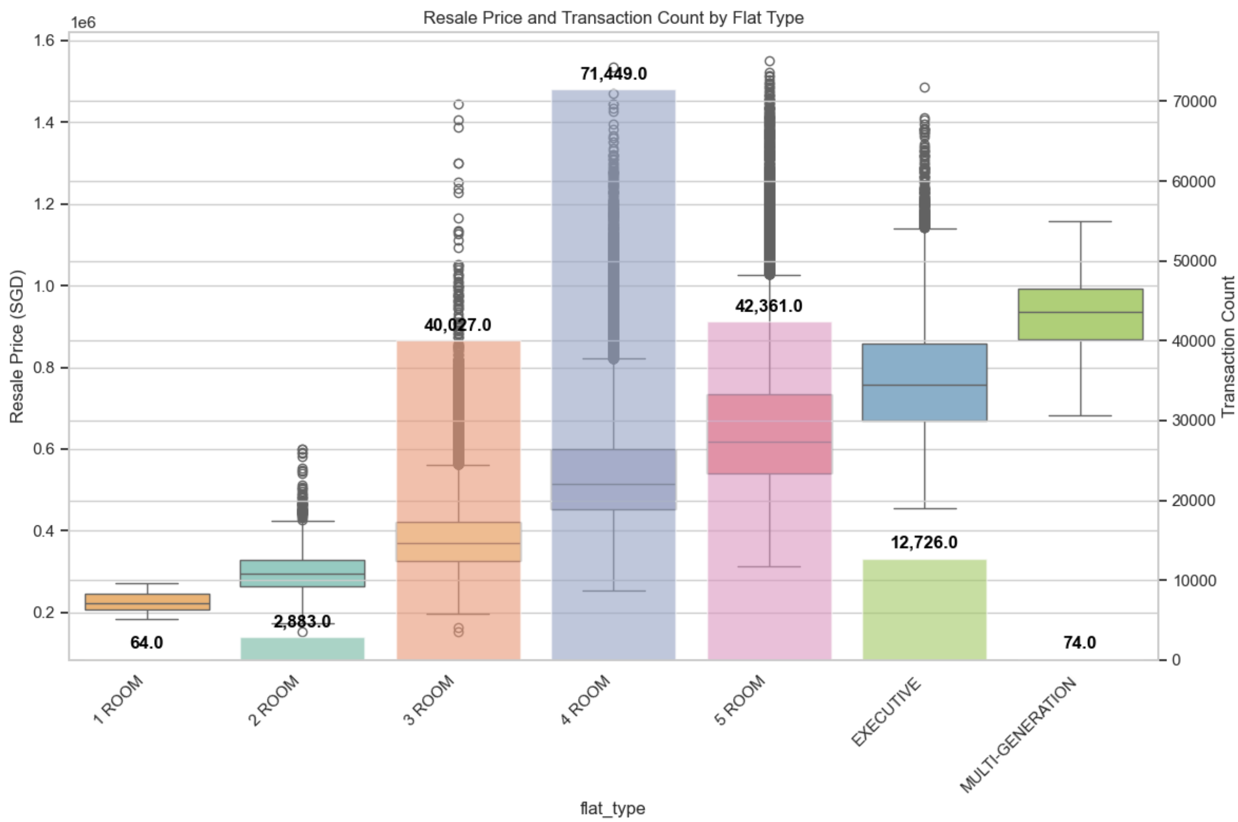


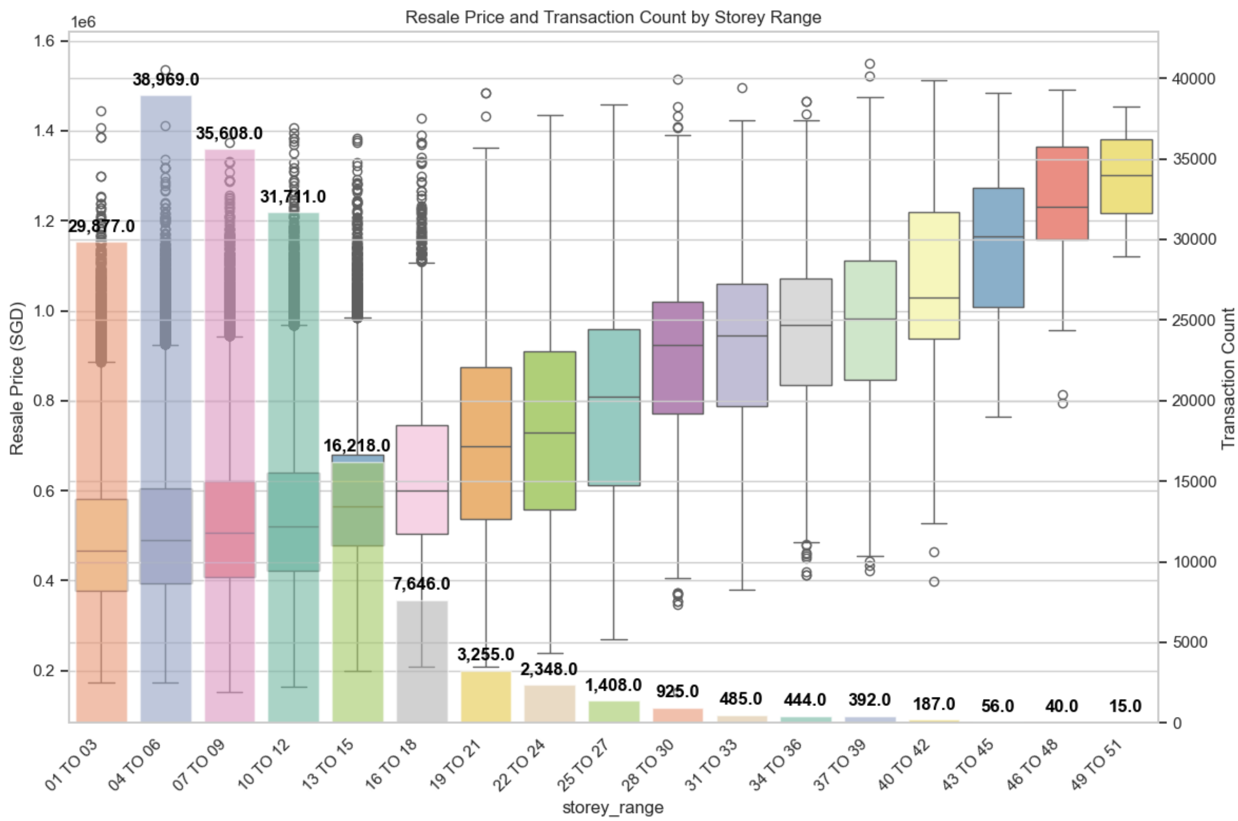


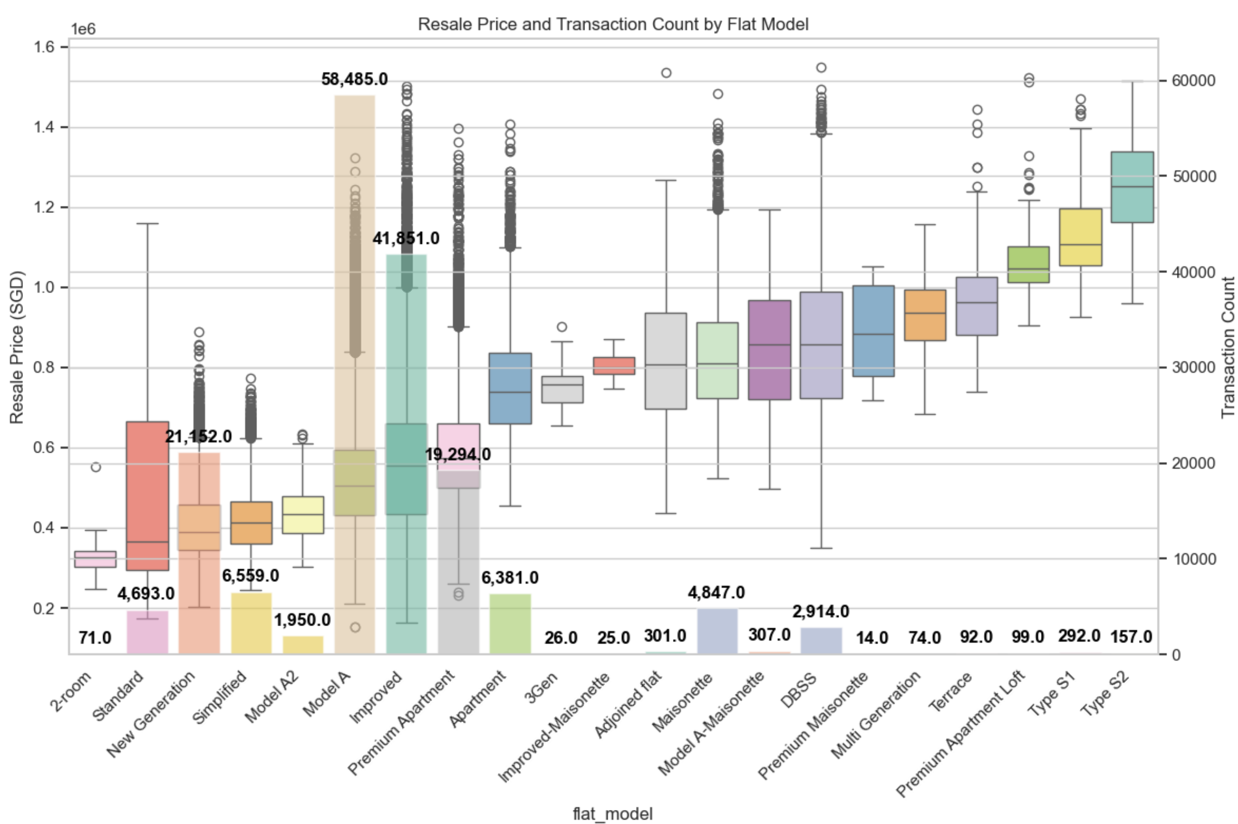
Flat Type, Storey Range, Flat Model VERSUS combined Transaction Counts and Resale Price

```
def combined_box_count_plot(data, x, y, title, min_bar_height_cm=1):
    import matplotlib.pyplot as plt
    import seaborn as sns
    order = data.groupby(x)[y].median().sort_values().index
    fig, ax1 = plt.subplots(figsize=(12, 8))
    sns.boxplot(x=x, y=y, hue=x, data=data, order=order, ax=ax1,
palette="Set3", legend=False)
    ax1.set_ylabel('Resale Price (SGD)')
    ax1.set_title(title)

    ax2 = ax1.twinx()
    sns.countplot(x=x, hue=x, data=data, order=order, ax=ax2,
palette="Set2", alpha=0.6, legend=False)
    ax2.set_ylabel('Transaction Count')

    for bar in ax2.patches:
        height = bar.get_height()
        ax2.annotate(f'{height:,}',
                     xy=(bar.get_x() + bar.get_width() / 2, height),
                     xytext=(0, 5),
                     textcoords='offset points',
                     ha='center', va='bottom',
                     fontsize=12, fontweight='bold', color='black')

    plt.setp(ax1.get_xticklabels(), rotation=45, ha="right")
```

```
    ax2.set_ylim(0, max([bar.get_height() for bar in ax2.patches]) *
1.1)
    plt.tight_layout()
    plt.show()

combined_box_count_plot(df, x='flat_type', y='resale_price_adj',
title='Resale Price and Transaction Count by Flat Type')
combined_box_count_plot(df, x='storey_range', y='resale_price_adj',
title='Resale Price and Transaction Count by Storey Range')
combined_box_count_plot(df, x='flat_model', y='resale_price_adj',
title='Resale Price and Transaction Count by Flat Model')
```




Resale Price and Transaction Count by Storey Range
1e6
Resale Price (SGD)
Transaction Count
storey_range
29,877.0
38,969.0
35,608.0
31,711.0
16,218.0
7,646.0
3,255.0
2,348.0
1,408.0
925.0
485.0
444.0
392.0
187.0
56.0
40.0
15.0
01 TO 03
04 TO 06
07 TO 09
10 TO 12
13 TO 15
16 TO 18
19 TO 21
22 TO 24
25 TO 27
28 TO 30
31 TO 33
34 TO 36
37 TO 39
40 TO 42
43 TO 45
46 TO 48
49 TO 51


Resale Price and Transaction Count by Flat Model
1e6
Resale Price (SGD)
Transaction Count
flat_model
71.0
4,693.0
21,152.0
6,559.0
1,950.0
58,485.0
41,851.0
19,294.0
6,381.0
26.0
25.0
301.0
4,847.0
307.0
2,914.0
14.0
74.0
92.0
99.0
292.0
157.0
2-room
Standard
New Generation
Simplified
Model A2
Model A
Improved
Premium Apartment
Apartment
3Gen
Improved-Maisonette
Adjoined flat
Maisonette
Model A-Maisonette
DBSS
Premium Maisonette
Multi Generation
Terrace
Premium Apartment Loft
Type S1
Type S2

1. By Flat Type
Chart: Resale Price and Transaction Count by Flat Type

- 4-room flats dominate in both resale price and transaction count — making them the most actively traded and middle-ground choice for many buyers.
- 5-room flats are priced slightly higher on average than 4-room flats but have fewer transactions, suggesting they may be less accessible or needed by smaller households.
- ⚖ Executive and Multi-Generation flats show higher resale price variance, but they are traded far less frequently, indicating they serve niche markets.
- Insight: The 3- to 5-room flats form the core of HDB resale activity, reinforcing their importance in housing policy planning.

---

2. By Storey Range
Chart: Resale Price and Transaction Count by Storey Range

- Higher floors (31 and above) tend to command higher resale prices consistently — confirming that "sky view premium" is real in Singapore.
- However, most transactions occur between the 4th and 12th storey, indicating buyers' practical preferences (e.g. accessibility, lift access, price) outweigh premium pricing.
- Lower storeys (1–3) and mid-range storeys (13–18) have notably lower prices, but they still capture significant transaction counts — possibly due to affordability or elderly buyers.
- Insight: While higher floors may be more expensive, buyers still gravitate toward lower-mid floors, balancing price, accessibility, and preference.

---

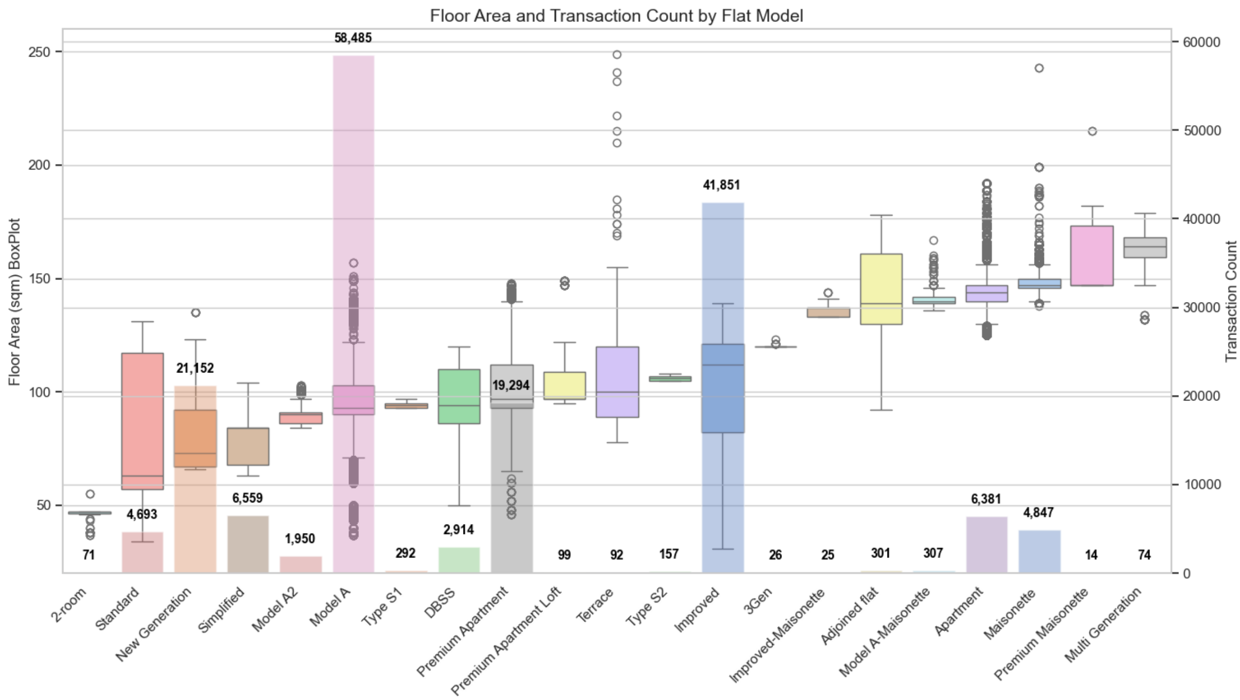
3. By Flat Model
Chart: Resale Price and Transaction Count by Flat Model

- Maisonettes, DBSS, and Premium Apartment Loft models have some of the highest resale prices, despite lower transaction volumes — signaling their luxury or rarity status.
- Improved and New Generation models have high transaction counts, suggesting they're more commonly built and traded, possibly across older estates.
- Old models like "Model A" and "Simplified" have lower prices and are traded much less now — indicating gradual phasing out of outdated layouts.
- Insight: Flat model plays a huge role in pricing, beyond just size or location. Premium/rare designs draw higher prices even with low supply.

□ Big Picture Takeaways

- □ The HDB resale market is shaped by a complex interaction of flat type, floor level, and model — pricing isn't determined by one factor alone.
- □ Although higher floors and premium models fetch higher prices, demand remains focused around practical, middle-tier options (4-room flats, mid-storeys, standard models).
- □ These charts support your modeling conclusion that no single feature dominates resale pricing — which alignversion formatted for a slide or a report!how — I can help summarize them nicely!

```
def var_floor_count(data, x, y, title):
    order = data.groupby(x)[y].median().sort_values().index
    fig, ax1 = plt.subplots(figsize=(14, 8))

    # Boxplot for floor area
    sns.boxplot(x=x, y=y, data=data, order=order, ax=ax1,
palette="pastel", hue=x, legend=False)
    ax1.set_ylabel('Floor Area (sqm) BoxPlot', fontsize=12)
    ax1.set_xlabel('')
    ax1.set_title(title, fontsize=14)

    # Countplot on secondary axis
    ax2 = ax1.twinx()
    sns.countplot(x=x, data=data, order=order, ax=ax2,
palette="muted", alpha=0.4, hue=x, legend=False)
    ax2.set_ylabel('Transaction Count', fontsize=12)

    counts = data[x].value_counts().reindex(order)
    for i, count in enumerate(counts):
        ax2.text(i, count + max(counts) * 0.02, f'{int(count):,}',
ha='center', va='bottom',
                 fontsize=10, weight='bold', color='black')
    ax1.set_xticks(range(len(order)))
    ax1.set_xticklabels(order, rotation=45, ha="right")
    plt.tight_layout()
    plt.show()

var_floor_count(df, x='flat_model', y='floor_area_sqm', title='Floor
Area and Transaction Count by Flat Model')
```



Insights:
Model A is the most transacted flat type, followed by Improved, Premium Apartment, New Generation and Simplified. Rare/legacy types like Premium Maisonette, Maisonette, 3Gen, and Multi Generation have very low counts (< 100), possibly phased out or niche.

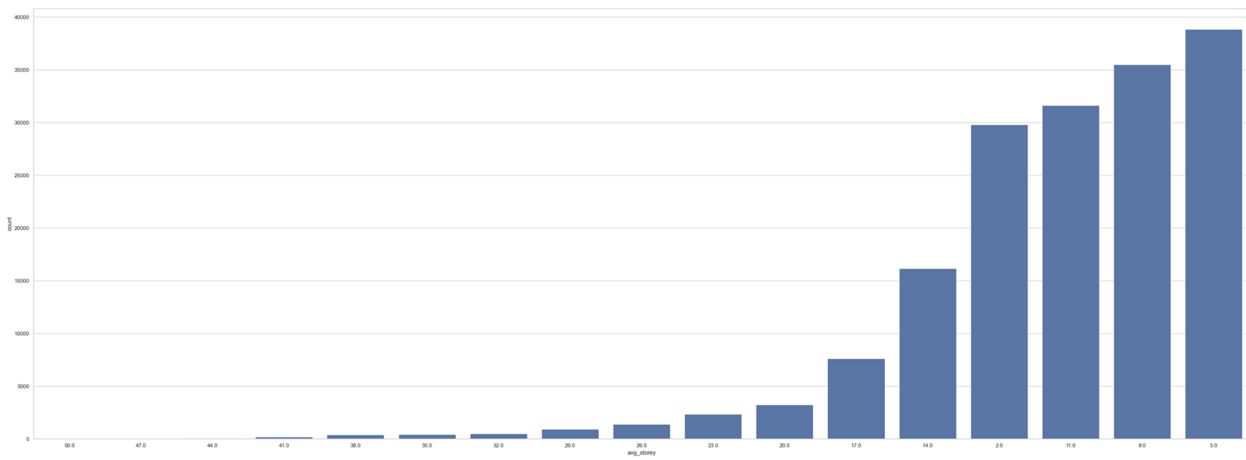
Premium Apartment has a moderate-large floor area and very high transaction count, suggesting it's a popular mid-high range option. Maisonettes offer large spaces (≥120 sqm) but are rarely transacted — possibly due to their age or limited stock. DBSS, Terrace, and Type S1/S2 have larger areas but relatively fewer transactions — possibly because they are rare, premium-priced, or phased out.

Model A is the sweet spot: moderate size (~100 sqm), highest count, likely reflecting standard family units in peak supply years. Premium Apartment Loft and Terrace units stand out for size, indicating luxury-tier living within HDB constraints.

EDA: Avg Storey, Uni-Variate

```
sns.set(style="whitegrid")
Average_Storey_order = 
df['avg_storey'].value_counts().sort_values(ascending=True).index
fig, axes = plt.subplots(figsize=(45, 16))
sns.countplot(data=df, x="avg_storey", order=Average_Storey_order, 
ax=axes)
```

```
<Axes: xlabel='avg_storey', ylabel='count'>
```

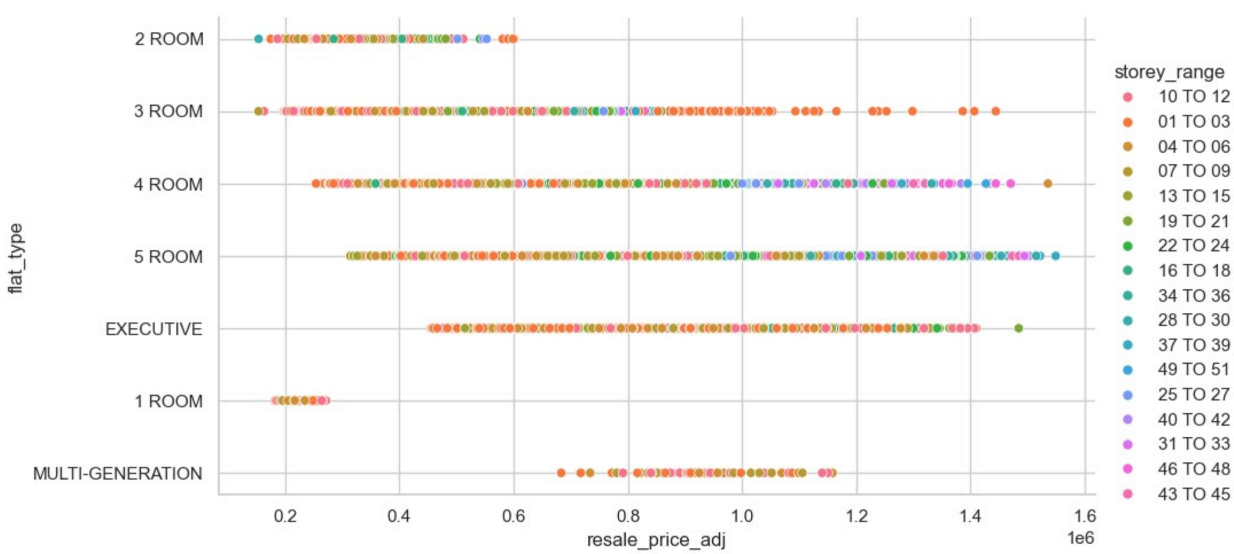


EDA On Flat_Type & Storey_range, Multi-Variate

According to the visualisations, it seems that even though storey tend to not affect much of the resale prices, it seems that **3 room** flats catch higher prices at a very low floors.

```
sns.relplot(x='resale_price_adj',y='flat_type', hue='storey_range', data=df,aspect=2 )
```

```
<seaborn.axisgrid.FacetGrid at 0x304f40440>
```



Insights
Storey does not always dictate prices.

Price Distribution by Flat Type: There is a clear progression in resale prices as the number of rooms increases. 1-room flats have the lowest prices (clustered around 200,000), while 5-room

and Executive flats command higher prices.
Multi-Generation Flats: These appear exclusively in the higher price ranges (800,000+), suggesting they're premium properties.
Price Ranges:

1-room: Tightly clustered around 200,000
2-room: Mostly 200,000-400,000
3-room: Wide range from 200,000 to 1,200,000+
4-room: Broad distribution from 300,000 to 1,600,000
5-room: Higher concentration in 400,000-1,400,000
Executive: Mostly 600,000-1,400,000

Storey Impact: While there is no strong correlation, higher storey ranges (such as 43-51) tend to appear more frequently at higher price points, particularly for 4 and 5-room flats.
Price Ceiling Variation: The maximum prices vary by flat type, with 4 and 5-room flats reaching the highest resale values (up to 1.6 million). Outliers: There are notable outliers in each category, particularly in the 4 and 5-room types that command prices well above the typical range for their category.
Distribution Density: 3, 4, and 5-room flats show the densest distribution, suggesting they're the most commonly traded types in this market.

Maximum Price Housing Characteristics

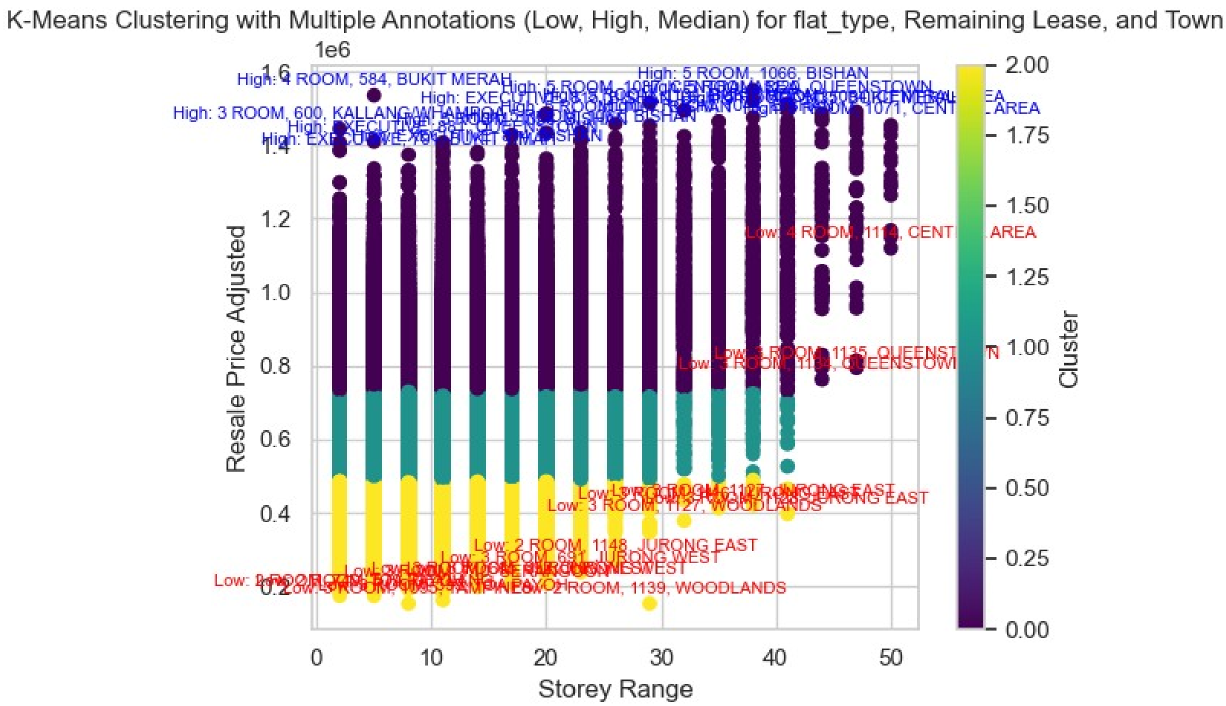
```
maxpriceHouse=df.iloc[df.resale_price_adj.argmax()]
maxpriceHouse
```

```
month                                2021-12
town                                  BISHAN
flat_type                             5 ROOM
block                                   273B
street_name                     BISHAN ST 24
storey_range                        37 TO 39
floor_area_sqm                           120
flat_model                              DBSS
lease_commence_date                     2011
remaining_lease           88 years 10 months
resale_price                         1360000
year                                    2021
cpi                                       88
resale_price_adj                   1,549,310
avg_storey                                38
remaining_lease_months                  1066
floor_area_range                     100-150
property_age                              14
property_age_group                     10-20
latitude                                   1
longitude                                104
Name: 90829, dtype: object
```

Clustering of Storey

This is something new we learnt.
According to the visualisation, it shows that storey do play a small range in determining the price, however it seems that other factors play a part as well. For instance, **Central Area** seems to provide higher resales price as compared to **Seng Kang** irregardless of the storey the HDB is at.

```
from sklearn.cluster import KMeans
X = df[['avg_storey', 'resale_price_adj']]
df['avg_storey'] = df['avg_storey'].astype(float)
kmeans = KMeans(n_clusters=3, random_state=42)
y_kmeans = kmeans.fit_predict(X)
plt.scatter(X['avg_storey'], X['resale_price_adj'], c=y_kmeans,
cmap='viridis')
for storey in df['avg_storey'].unique():
    storey_data = df[df['avg_storey'] == storey]

    lowest_point =
storey_data.loc[storey_data['resale_price_adj'].idxmin()]
    plt.annotate(f'Low: {lowest_point["flat_type"]},
{lowest_point["remaining_lease_months"]}, {lowest_point["town"]}',
                 (lowest_point['avg_storey'],
lowest_point['resale_price_adj']),
                 textcoords="offset points", xytext=(0, 5),
ha='center', fontsize=8, color='red')

    highest_point =
storey_data.loc[storey_data['resale_price_adj'].idxmax()]
    plt.annotate(f'High: {highest_point["flat_type"]},
{highest_point["remaining_lease_months"]}, {highest_point["town"]}',
                 (highest_point['avg_storey'],
highest_point['resale_price_adj']),
                 textcoords="offset points", xytext=(0, 5),
ha='center', fontsize=8, color='blue')

plt.title("K-Means Clustering with Multiple Annotations (Low, High,
Median) for flat_type, Remaining Lease, and Town")
plt.xlabel("Storey Range")
plt.ylabel("Resale Price Adjusted")
plt.colorbar(label='Cluster')
plt.show()
```



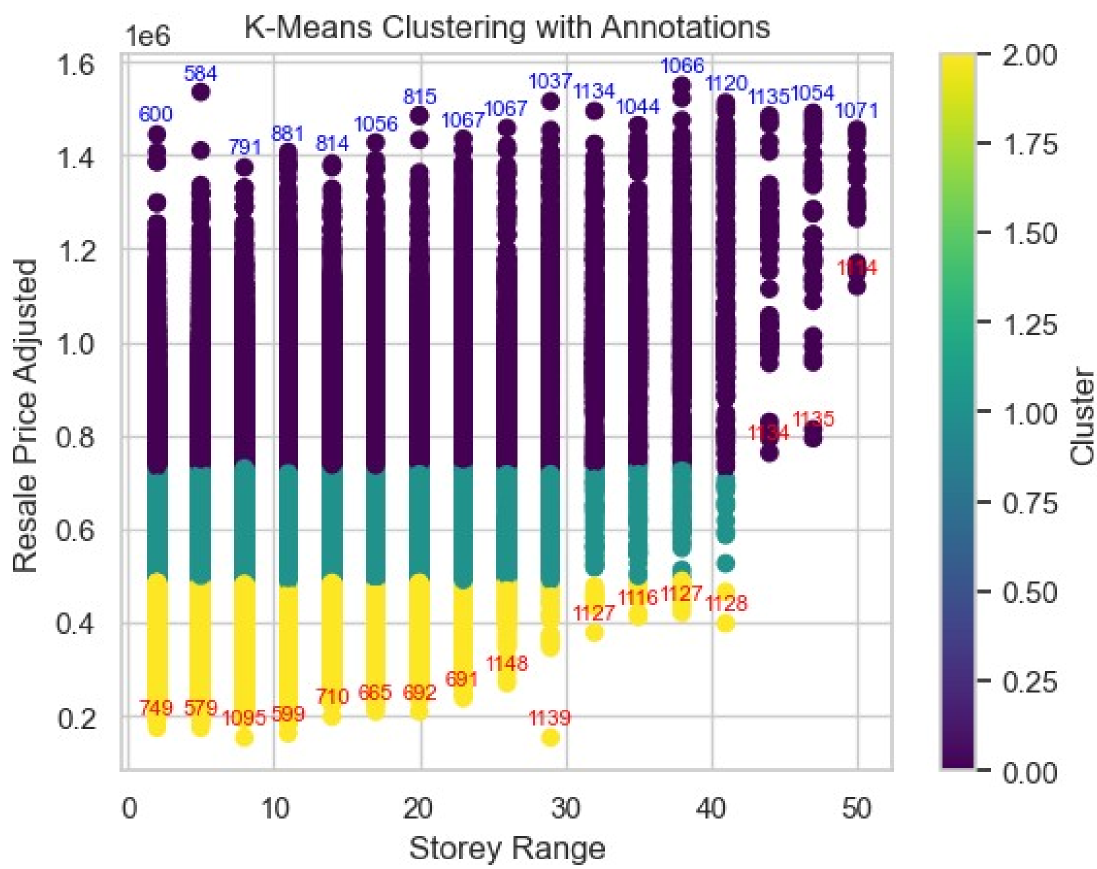
Insights:

Interestingly, it shows that houses with higher lease remaining tend to have higher storey range, this is expected because of how new generation flats tend to have higher storey compared to older generation flats.

Higher storeys, means higher resale price. but even with lower storeys, resale price can still be high. cluster yellow: likely represents less desirable flats cluster green: average locations cluster purple: likely premium, possibly newer, well-located, or in high demand (e.g., city fringe, near MRT, well-renovated)

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
X = df[['avg_storey', 'resale_price_adj']]

kmeans = KMeans(n_clusters=3, random_state=42)
y_kmeans = kmeans.fit_predict(X)

plt.scatter(X['avg_storey'], X['resale_price_adj'], c=y_kmeans,
cmap='viridis')

for storey in df['storey_range'].unique():
    storey_data = df[df['storey_range'] == storey]
    lowest_point =
storey_data.loc[storey_data['resale_price_adj'].idxmin()]
    plt.annotate(f'{lowest_point["remaining_lease_months"]}',
                 (lowest_point['avg_storey'],
lowest_point['resale_price_adj']),
```

```
                    textcoords="offset points", xytext=(0, 5),
ha='center', fontsize=8, color='red')
    highest_point =
storey_data.loc[storey_data['resale_price_adj'].idxmax()]
    plt.annotate(f'{highest_point["remaining_lease_months"]}',
                    (highest_point['avg_storey'],
highest_point['resale_price_adj']),
                    textcoords="offset points", xytext=(0, 5),
ha='center', fontsize=8, color='blue')
plt.title("K-Means Clustering with Annotations")
plt.xlabel("Storey Range")
plt.ylabel("Resale Price Adjusted")
plt.colorbar(label='Cluster')
plt.show()
```



Insights
Clear Price Stratification: The data shows three distinct price clusters (colored yellow, teal, and purple).
Height Premium Pattern: There's a consistent upward trend in the maximum resale prices as storey range increases, confirming the "higher floor premium" effect in real estate pricing.
Cluster Boundaries:

Low cluster (yellow): ~200,000-500,000
Mid cluster (teal): ~500,000-750,000
High cluster (purple): ~750,000-1,600,000

Price Ceiling Trend: The highest prices (annotated in blue) show an upward trajectory from storeys 1-40, peaking around the 1,550,000 mark at storeys 35-40.
Price Floor Pattern: The lowest prices (annotated in red) gradually increase with storey height until around storey 40, then they jump significantly higher for the uppermost floors.
Diminishing Range at Highest Floors: Above storey 40, the price spread narrows considerably, with fewer properties in the lowest cluster.
Data Density: The visualization shows more data points in the middle storey ranges (10-40), suggesting these are more common in the housing stock.
Outliers: Several numbered annotations highlight specific properties that represent maximum/minimum values within their storey ranges.
High-Rise Premium Cap: The price premium for height appears to plateau or slightly decrease beyond storey 40, suggesting there may be a ceiling to how much premium buyers will pay for extreme heights.

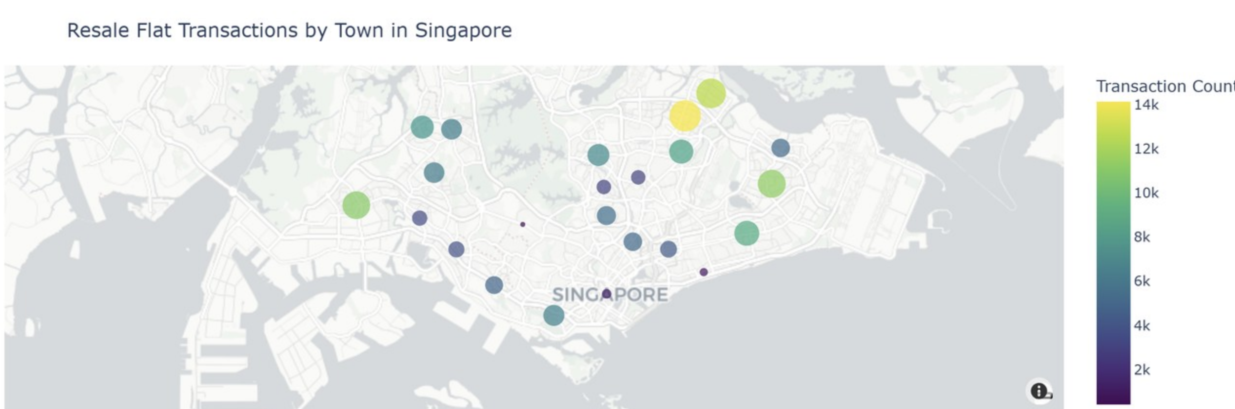
This analysis confirms that while higher floors generally command higher prices, the relationship is not purely linear, and other factors likely influence clustering beyond just height of the storeys.

EDA: Mapping towns

conda install geopandas
This is something new we learnt.

```
town_counts = df["town"].value_counts().reset_index()
town_counts.columns = ["town", "count"]
town_coords = {
    "ANG MO KIO": [1.3691, 103.8454],
    "BEDOK": [1.3243, 103.9303],
    "BISHAN": [1.3508, 103.8485],
    "BUKIT BATOK": [1.3590, 103.7513],
    "BUKIT MERAH": [1.2774, 103.8199],
    "BUKIT PANJANG": [1.3838, 103.7613],
    "BUKIT TIMAH": [1.3294, 103.8021],
    "CENTRAL AREA": [1.2897, 103.8501],
    "CHOA CHU KANG": [1.3850, 103.7446],
    "CLEMENTI": [1.3151, 103.7641],
    "GEYLANG": [1.3152, 103.8855],
    "HOUGANG": [1.3710, 103.8928],
    "JURONG EAST": [1.3330, 103.7431],
    "JURONG WEST": [1.3404, 103.7068],
    "KALLANG/WHAMPOA": [1.3195, 103.8651],
    "MARINE PARADE": [1.3021, 103.9057],
    "PASIR RIS": [1.3731, 103.9497],
    "PUNGGOL": [1.4043, 103.9099],
```

```
    "QUEENSTOWN": [1.2947, 103.7857],
    "SEMBAWANG": [1.4471, 103.8200],
    "SENGKANG": [1.3915, 103.8950],
    "SERANGOON": [1.3563, 103.8682],
    "TAMPINES": [1.3526, 103.9446],
    "TOA PAYOH": [1.3344, 103.8500],
    "WOODLANDS": [1.4381, 103.7890],
    "YISHUN": [1.4294, 103.8354]
}
town_counts["lat"] = town_counts["town"].map(lambda x:
town_coords.get(x, [None, None])[0])
town_counts["lon"] = town_counts["town"].map(lambda x:
town_coords.get(x, [None, None])[1])
town_counts = town_counts.dropna()
town_counts["scaled_count"] = (town_counts["count"] /
town_counts["count"].max()) * 120

fig = px.scatter_mapbox(
    town_counts,
    lat="lat",
    lon="lon",
    size="scaled_count",
    color="count",
    hover_name="town",
    hover_data={"count": True},
    color_continuous_scale="Viridis",
    title="Resale Flat Transactions by Town in Singapore",
    zoom=11,
    height=700
)
fig.update_layout(
    mapbox_style="open-street-nao",
    margin={"r": 0, "t": 50, "l": 0, "b": 0},
    coloraxis_colorbar=dict(title="Transaction Count")
)
fig.show()
```



Getting addresses of hdb

```
import requests
import pandas as pd
import time
unique_streets = df['street_name'].unique()
street_coords = {}

def get_coordinates(street):
    url = f"https://www.onemap.gov.sg/api/common/elastic/search?
searchVal={street} SINGAPORE&returnGeom=Y&getAddrDetails=Y&pageNum=1"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        if data["found"] > 0:
            lat = data["results"][0]["LATITUDE"]
            lon = data["results"][0]["LONGITUDE"]
            return float(lat), float(lon)
    return None, None

for street in unique_streets:
    lat, lon = get_coordinates(street)
    street_coords[street] = (lat, lon)
    print(f"{street} → ({lat}, {lon})")
    time.sleep(0.2)

df['latitude'] = df['street_name'].map(lambda x: street_coords[x][0])
df['longitude'] = df['street_name'].map(lambda x: street_coords[x][1])

df.to_csv("lon_lat_17aprilWith2024", index=False)
```

```
ANG MO KIO AVE 10 → (1.35968882523895, 103.854575321899)
ANG MO KIO AVE 4 → (1.37231328287072, 103.837601112788)
ANG MO KIO AVE 5 → (1.37634218416684, 103.840158984505)
ANG MO KIO AVE 1 → (1.36013579518176, 103.855025577808)
ANG MO KIO AVE 3 → (1.36619690818888, 103.847756649677)
ANG MO KIO AVE 9 → (1.38403248432085, 103.840356912671)
ANG MO KIO AVE 8 → (1.37742633496685, 103.848699347584)
ANG MO KIO AVE 6 → (1.37048118793194, 103.844805800791)
ANG MO KIO ST 52 → (1.37196445720109, 103.851763531681)
BEDOK NTH AVE 4 → (1.33497991918563, 103.949579950322)
BEDOK NTH AVE 1 → (1.32744916555918, 103.927184430479)
BEDOK NTH RD → (1.33145574028188, 103.935458644142)
BEDOK STH AVE 1 → (1.32085208689731, 103.933721091441)
BEDOK RESERVOIR RD → (1.32808857939361, 103.909877642545)
CHAI CHEE ST → (1.32856339985122, 103.923222454401)
BEDOK NTH ST 3 → (1.33168385606168, 103.922360842039)
BEDOK STH RD → (1.32123665768589, 103.92867661181)
CHAI CHEE AVE → (1.32510368063361, 103.924874310792)
NEW UPP CHANGI RD → (1.32529103316339, 103.930353017872)
```

```
CHAI CHEE DR → (1.32370195143182, 103.91981743872)
BEDOK STH AVE 2 → (1.32258561622604, 103.936849426204)
BEDOK NTH AVE 3 → (1.3269699402053, 103.93585007622)
BEDOK RESERVOIR VIEW → (1.33781310073984, 103.935679722167)
CHAI CHEE RD → (1.32727572366021, 103.924803599281)
LENGKONG TIGA → (1.32388941155006, 103.912757798349)
BEDOK CTRL → (1.32628149646546, 103.934180410915)
JLN DAMAI → (1.33324475634461, 103.90994321469)
BEDOK NTH AVE 2 → (1.32810291260758, 103.932938099482)
BEDOK STH AVE 3 → (1.32191533930821, 103.933180861675)
SIN MING RD → (1.35371665672148, 103.836115880148)
SIN MING AVE → (1.36386365291723, 103.833840920046)
BISHAN ST 12 → (1.34755435145843, 103.849896077145)
BISHAN ST 13 → (1.34759165594904, 103.855034433114)
BISHAN ST 22 → (1.35900490923943, 103.847006575058)
BISHAN ST 24 → (1.35865897891792, 103.842049817206)
BISHAN ST 23 → (1.35506686853913, 103.846946060532)
BRIGHT HILL DR → (1.35593097385551, 103.832217240299)
SHUNFU RD → (1.35176163249001, 103.839046663239)
BT BATOK ST 34 → (1.36348296381773, 103.749393361059)
BT BATOK ST 51 → (1.3571882058688, 103.751123943023)
BT BATOK ST 11 → (1.35044171803071, 103.743212010475)
BT BATOK ST 52 → (1.35606274589378, 103.752766962762)
BT BATOK ST 21 → (1.34586253909028, 103.747905760698)
BT BATOK EAST AVE 5 → (1.34904853042117, 103.755828558538)
BT BATOK WEST AVE 6 → (1.34670636093149, 103.746920663038)
BT BATOK CTRL → (1.34822368204242, 103.748034526478)
BT BATOK WEST AVE 8 → (1.35148188938294, 103.738386015338)
BT BATOK EAST AVE 4 → (1.34979707822416, 103.758148314499)
BT BATOK ST 31 → (1.35676942740178, 103.748774260571)
BT BATOK ST 25 → (1.34235172835604, 103.760159353985)
BT BATOK EAST AVE 3 → (1.34745649001205, 103.754710439758)
BT BATOK WEST AVE 5 → (1.3603313804856, 103.741236536443)
BT BATOK ST 24 → (1.34326273087543, 103.756531451656)
JLN BT HO SWEE → (1.28757615073911, 103.829412797188)
TELOK BLANGAH DR → (1.27325111010743, 103.811719770241)
BEO CRES → (1.28852535171166, 103.828458597142)
TELOK BLANGAH CRES → (1.27773153190559, 103.819769431353)
TAMAN HO SWEE → (1.28780753147299, 103.832780576669)
TELOK BLANGAH RISE → (1.27394761788206, 103.820866934319)
TELOK BLANGAH WAY → (1.27446866093398, 103.8214740879)
JLN BT MERAH → (1.28730052130543, 103.807988268738)
JLN KLINIK → (1.28811451810467, 103.829040334276)
TELOK BLANGAH HTS → (1.27492292747431, 103.813408066986)
BT MERAH VIEW → (1.28357870670139, 103.823819212586)
INDUS RD → (1.29130442807028, 103.827447793798)
BT MERAH LANE 1 → (1.28593893475633, 103.804715808324)
TELOK BLANGAH ST 31 → (1.27439946314347, 103.807281423949)
MOH GUAN TER → (1.28488193684795, 103.83104242125)
```

```
HAVELOCK RD → (1.28825972041875, 103.836657784125)
HENDERSON CRES → (1.29071799751539, 103.821625207737)
BT PURMEI RD → (1.27146256556443, 103.825683545567)
KIM TIAN RD → (1.28400968071886, 103.827421421303)
DEPOT RD → (1.28019628238676, 103.815076215856)
JLN RUMAH TINGGI → (1.28797300869683, 103.808839267852)
DELTA AVE → (1.2920752508431, 103.828584077626)
JLN MEMBINA → (1.28532736402595, 103.828385533878)
REDHILL RD → (1.28793296437158, 103.817298929577)
LENGKOK BAHRU → (1.2867006427442, 103.810819756429)
ZION RD → (1.28859939872156, 103.83339469918)
PETIR RD → (1.37792694616982, 103.763102931112)
PENDING RD → (1.37855560391305, 103.768748798796)
BANGKIT RD → (1.38137152811663, 103.773245009924)
SEGAR RD → (1.38676223534087, 103.772523384527)
JELAPANG RD → (1.38288226683648, 103.76684712949)
SENJA RD → (1.38191601013855, 103.762499853661)
FAJAR RD → (1.38333404683843, 103.769408076179)
BT PANJANG RING RD → (1.37853337682238, 103.771789071508)
SENJA LINK → (1.38691070954118, 103.763310183378)
LOMPANG RD → (1.38021544120199, 103.766302173684)
GANGSA RD → (1.37656096521362, 103.765952443882)
TOH YI DR → (1.33820236785945, 103.774522626293)
FARRER RD → (1.31198097962561, 103.854748154489)
JLN KUKOH → (1.28683793317038, 103.838674084243)
ROWELL RD → (1.30939779202524, 103.854053574201)
WATERLOO ST → (1.29853916273521, 103.851856012904)
NEW MKT RD → (1.2839640518753, 103.843141945391)
TG PAGAR PLAZA → (1.27523683085708, 103.842605335418)
QUEEN ST → (1.29711316857746, 103.85196484973)
BAIN ST → (1.29681525928856, 103.853619482206)
CANTONMENT RD → (1.27937182115543, 103.840030648473)
TECK WHYE LANE → (1.37788964623982, 103.754239909547)
CHOA CHU KANG AVE 4 → (1.38237840752771, 103.739203543708)
CHOA CHU KANG AVE 3 → (1.3808318992022, 103.74293968344)
CHOA CHU KANG CRES → (1.40021628497652, 103.750393021939)
CHOA CHU KANG ST 54 → (1.39381403543098, 103.749398690897)
CHOA CHU KANG CTRL → (1.38092680230382, 103.748559601544)
JLN TECK WHYE → (1.37820351130765, 103.756450878128)
CHOA CHU KANG ST 62 → (1.39693363078609, 103.744921736632)
CHOA CHU KANG NTH 6 → (1.3958713396608, 103.743519194378)
CHOA CHU KANG DR → (1.38582271319419, 103.746674275849)
CHOA CHU KANG NTH 5 → (1.39246581726118, 103.747906889864)
CHOA CHU KANG ST 52 → (1.39447144920635, 103.744054352225)
CHOA CHU KANG AVE 2 → (1.37803539434682, 103.745020692529)
CLEMENTI WEST ST 2 → (1.30234147977998, 103.764495663452)
WEST COAST RD → (1.29220562998617, 103.76889162075)
CLEMENTI WEST ST 1 → (1.30477461299303, 103.765253754328)
CLEMENTI AVE 4 → (1.30891243803731, 103.76705513582)
```

```
CLEMENTI AVE 5 → (1.31513509080723, 103.766054474051)
CLEMENTI ST 11 → (1.32212392900686, 103.770265903384)
CLEMENTI AVE 2 → (1.31399161476536, 103.767089003977)
CLEMENTI AVE 3 → (1.31158040643011, 103.764211279777)
CLEMENTI AVE 1 → (1.30924251597431, 103.767106335451)
C'WEALTH AVE WEST → (1.32300684559737, 103.761868274259)
CIRCUIT RD → (1.32880277340318, 103.888599094688)
BALAM RD → (1.33038182281246, 103.885788977269)
MACPHERSON LANE → (1.33255923011716, 103.885892533665)
EUNOS CRES → (1.3206576452482, 103.90246344511)
UBI AVE 1 → (1.33068802490987, 103.901953040703)
HAIG RD → (1.31129091152627, 103.897703432652)
OLD AIRPORT RD → (1.30707415731033, 103.883582973686)
GEYLANG EAST AVE 1 → (1.31585561966299, 103.885271443327)
SIMS DR → (1.31556694222653, 103.876391006571)
PIPIT RD → (1.32399305603529, 103.886692720017)
GEYLANG EAST CTRL → (1.31821153053878, 103.884752333398)
EUNOS RD 5 → (1.31844042901657, 103.899852177666)
CASSIA CRES → (1.30894452962256, 103.883530101124)
BUANGKOK CRES → (1.3803190835418, 103.878911441386)
HOUGANG AVE 3 → (1.36413097349222, 103.89300132385)
HOUGANG AVE 8 → (1.37647221404827, 103.893822113477)
HOUGANG AVE 1 → (1.36413097349222, 103.89300132385)
HOUGANG AVE 5 → (1.36778751494032, 103.89392989782)
HOUGANG ST 61 → (1.37518092664617, 103.886022660348)
HOUGANG ST 11 → (1.35202640347162, 103.879078569569)
HOUGANG AVE 7 → (1.36497584742043, 103.896809883055)
HOUGANG AVE 4 → (1.37123297855584, 103.88691800304)
HOUGANG AVE 2 → (1.36596036249795, 103.888646477925)
LOR AH SOO → (1.35087034387766, 103.883677420501)
HOUGANG ST 92 → (1.37457017021599, 103.879877813491)
HOUGANG ST 52 → (1.3771741721863, 103.889474443456)
HOUGANG AVE 10 → (1.3732611475703, 103.894631713049)
HOUGANG ST 51 → (1.37971113367172, 103.889950967432)
UPP SERANGOON RD → (1.36292693490267, 103.887685723603)
HOUGANG CTRL → (1.37115128124187, 103.89447367673)
HOUGANG ST 91 → (1.37844828552468, 103.882856903411)
BUANGKOK LINK → (1.38198270653355, 103.881217675808)
HOUGANG ST 31 → (1.36382073428889, 103.889572770625)
PANDAN GDNS → (1.3212933695135, 103.747648912115)
TEBAN GDNS RD → (1.32349635256112, 103.737884227151)
JURONG EAST ST 24 → (1.34263187196449, 103.742133807093)
JURONG EAST ST 21 → (1.33673823981604, 103.74381241167)
JURONG EAST AVE 1 → (1.35062884013196, 103.728878030806)
JURONG EAST ST 13 → (1.33762415333866, 103.738354894451)
JURONG EAST ST 32 → (1.34404144956674, 103.735500080389)
TOH GUAN RD → (1.33583470103276, 103.748281616371)
JURONG WEST ST 93 → (1.33701267246131, 103.693517778242)
BOON LAY AVE → (1.34459584223718, 103.708190688947)
```

```
HO CHING RD → (1.33388939555115, 103.723306118882)
BOON LAY DR → (1.34624968110858, 103.709387013109)
TAO CHING RD → (1.33318449829266, 103.724354324887)
JURONG WEST ST 91 → (1.34298940506671, 103.691864154487)
JURONG WEST ST 42 → (1.35328785595113, 103.723331769563)
JURONG WEST ST 92 → (1.34079304701319, 103.690470022063)
BOON LAY PL → (1.34834916643319, 103.711381619147)
JURONG WEST ST 52 → (1.34935036475827, 103.717947920796)
TAH CHING RD → (1.33727166109701, 103.724355525718)
JURONG WEST ST 81 → (1.34883617453427, 103.695003812477)
YUNG SHENG RD → (1.33441342907315, 103.722221199754)
JURONG WEST ST 25 → (1.3547096550712, 103.702641921539)
JURONG WEST ST 73 → (1.34584119966198, 103.699605445991)
JURONG WEST ST 72 → (1.34504744920415, 103.698147806019)
JURONG WEST AVE 3 → (1.34987064861736, 103.703027085952)
JURONG WEST AVE 5 → (1.34988154179299, 103.702687484208)
YUNG HO RD → (1.32676126001247, 103.721288499183)
JURONG WEST ST 74 → (1.34852762746799, 103.699428834965)
JURONG WEST AVE 1 → (1.34988154179299, 103.702687484208)
JURONG WEST ST 71 → (1.34231547534586, 103.69528112377)
JURONG WEST ST 61 → (1.33641298743234, 103.69970173931)
JURONG WEST ST 65 → (1.34190096421275, 103.700263646805)
JURONG WEST CTRL 1 → (1.34426725618628, 103.705833695408)
JURONG WEST ST 64 → (1.33620559481206, 103.704177389573)
JURONG WEST ST 62 → (1.33966177435104, 103.70013784164)
JURONG WEST ST 41 → (1.35069822477174, 103.720905064522)
JURONG WEST ST 24 → (1.35088144577666, 103.704062250377)
JLN BATU → (1.30315437968651, 103.883912023247)
JLN BAHAGIA → (1.32540758450301, 103.858565503017)
LOR LIMAU → (1.32379607125113, 103.855670620709)
ST. GEORGE'S RD → (1.32211971964107, 103.860820092196)
KALLANG BAHRU → (1.32036884714802, 103.868340795574)
DORSET RD → (1.31344485985076, 103.850086370216)
GEYLANG BAHRU → (1.32370763354213, 103.871683051486)
BENDEMEER RD → (1.31525572704872, 103.860149969633)
WHAMPOA DR → (1.3225053606235, 103.853439457391)
UPP BOON KENG RD → (1.31264014497827, 103.872491873561)
RACE COURSE RD → (1.30938416717193, 103.85198385069)
OWEN RD → (1.31369813219166, 103.853259087477)
NTH BRIDGE RD → (1.29578999991756, 103.854041485994)
TOWNER RD → (1.31942539040726, 103.861087462608)
FARRER PK RD → (1.31198097962561, 103.854748154489)
MCNAIR RD → (1.31861240808264, 103.85609494219)
JLN TENTERAM → (1.32649026652942, 103.860572066608)
BOON KENG RD → (1.31569896370315, 103.859785170022)
JLN RAJAH → (1.32699075133711, 103.847403697576)
MARINE DR → (1.30338982213302, 103.908553539684)
MARINE CRES → (1.30402508947699, 103.913501458863)
MARINE TER → (1.30509642100259, 103.917733747552)
```

```
CHANGI VILLAGE RD → (1.38854656229022, 103.987804503483)
PASIR RIS ST 71 → (1.37790740617351, 103.935984430467)
PASIR RIS ST 11 → (1.3684636800666, 103.956009070114)
PASIR RIS DR 3 → (1.37091699784346, 103.955849482219)
PASIR RIS DR 6 → (1.37091699784346, 103.955849482219)
PASIR RIS ST 21 → (1.36933055696636, 103.962762942764)
PASIR RIS DR 4 → (1.36858695162235, 103.959204695763)
PASIR RIS ST 53 → (1.37276905622506, 103.946510452269)
PASIR RIS DR 10 → (1.37804865266679, 103.937574532149)
PASIR RIS ST 52 → (1.37505696315817, 103.945289416873)
PASIR RIS ST 12 → (1.36705021516612, 103.958181857039)
PASIR RIS ST 51 → (1.36842577055313, 103.950648583066)
PASIR RIS ST 72 → (1.3819775730856, 103.937637737839)
PASIR RIS DR 1 → (1.37538513507624, 103.939356106886)
PUNGGOL FIELD → (1.3979461677694, 103.905945136259)
EDGEDALE PLAINS → (1.39472186392182, 103.909873807721)
PUNGGOL FIELD WALK → (1.39330438365601, 103.912130617791)
EDGEFIELD PLAINS → (1.39747339816869, 103.904821824666)
PUNGGOL RD → (1.42068279153158, 103.908229394415)
PUNGGOL EAST → (1.38604163686578, 103.902900571161)
PUNGGOL DR → (1.40717643218, 103.904437061575)
PUNGGOL CTRL → (1.39540050236203, 103.91523939182)
PUNGGOL PL → (1.40672876870245, 103.905515357793)
C'WEALTH CL → (1.36895143913237, 103.773340678215)
STIRLING RD → (1.29707664144646, 103.800924785913)
MEI LING ST → (1.29348201306177, 103.8049167409)
C'WEALTH CRES → (1.30805159231327, 103.800842814413)
C'WEALTH DR → (1.33598647881104, 103.814300881432)
GHIM MOH RD → (1.31181075710414, 103.785967251779)
DOVER RD → (1.30797570313315, 103.777733041672)
HOLLAND AVE → (1.30934498335733, 103.795559353753)
STRATHMORE AVE → (1.29338294925716, 103.809330719416)
HOLLAND DR → (1.31499297470317, 103.782787128582)
GHIM MOH LINK → (1.30869473784381, 103.784048768238)
CLARENCE LANE → (1.29234264245612, 103.815172818084)
DOVER CRES → (1.30406012764311, 103.782637200936)
SEMBAWANG DR → (1.4457453096024, 103.821152676215)
SEMBAWANG CL → (1.43494011882681, 103.802910592969)
MONTREAL DR → (1.45089251057067, 103.823678171092)
ADMIRALTY LINK → (1.45427018021651, 103.815812639222)
ADMIRALTY DR → (1.45123288382933, 103.81588941317)
SEMBAWANG CRES → (1.44482697682231, 103.81737852086)
CANBERRA RD → (1.44585216960767, 103.822896239346)
FERNVALE RD → (1.38984421252451, 103.875022286615)
COMPASSVALE LANE → (1.38536458815848, 103.900817493136)
ANCHORVALE RD → (1.39033147569227, 103.886768124827)
RIVERVALE DR → (1.38246038039285, 103.902238681998)
RIVERVALE CRES → (1.39250410994833, 103.904483668323)
SENGKANG EAST WAY → (1.38625978931626, 103.90610810756)
```

```
RIVERVALE ST → (1.38989921548704, 103.902189187567)
RIVERVALE WALK → (1.38215139024135, 103.900075103796)
FERNVALE LANE → (1.39030227963797, 103.874445458032)
ANCHORVALE LINK → (1.38809684996098, 103.891441268138)
COMPASSVALE RD → (1.38700703200166, 103.891964770037)
COMPASSVALE CRES → (1.39907215975311, 103.896456458461)
JLN KAYU → (1.3981475833764, 103.872190046897)
COMPASSVALE WALK → (1.38813209503832, 103.89725765372)
COMPASSVALE DR → (1.38785624696615, 103.894681035814)
COMPASSVALE LINK → (1.38515662404444, 103.895055060286)
COMPASSVALE BOW → (1.38283357281231, 103.890243161685)
SENGKANG CTRL → (1.38407497672692, 103.892522060726)
ANCHORVALE LANE → (1.39131627514676, 103.884776554591)
ANCHORVALE DR → (1.39052785130704, 103.890470348151)
COMPASSVALE ST → (1.39368190554213, 103.900770045348)
SERANGOON AVE 4 → (1.3733188039986, 103.868758532925)
LOR LEW LIAN → (1.35105750010477, 103.876835728912)
SERANGOON AVE 2 → (1.3723953045272, 103.869166235443)
SERANGOON NTH AVE 1 → (1.37587606900586, 103.871197626608)
SERANGOON AVE 1 → (1.37616135805867, 103.871395487426)
SERANGOON CTRL → (1.35097779933054, 103.873595480638)
SERANGOON NTH AVE 4 → (1.37463845670095, 103.873687116712)
TAMPINES ST 22 → (1.34781303677538, 103.950409276356)
TAMPINES ST 41 → (1.3573452482067, 103.944611453168)
TAMPINES AVE 4 → (1.34495328242502, 103.939408365081)
TAMPINES ST 44 → (1.3588395485276, 103.954351586053)
TAMPINES ST 81 → (1.34841976685509, 103.934649103154)
TAMPINES ST 11 → (1.34740034584157, 103.945294915705)
TAMPINES ST 23 → (1.35292162536007, 103.953858335355)
TAMPINES ST 91 → (1.34711275884527, 103.939451879381)
TAMPINES ST 21 → (1.35289506312913, 103.953585683185)
TAMPINES ST 83 → (1.35036280478376, 103.934554541156)
TAMPINES ST 42 → (1.35756791894077, 103.953020775042)
TAMPINES ST 71 → (1.3561725371375, 103.937479923002)
TAMPINES ST 45 → (1.36026524882674, 103.958372350188)
TAMPINES ST 34 → (1.35625122843632, 103.961576641177)
TAMPINES ST 82 → (1.34978359073264, 103.93561470672)
TAMPINES AVE 9 → (1.35989315339663, 103.955756570936)
SIMEI ST 1 → (1.34134724976252, 103.951355074928)
SIMEI ST 5 → (1.34315890383485, 103.954188106526)
TAMPINES ST 72 → (1.3588121878635, 103.935212702172)
TAMPINES ST 84 → (1.35440347555796, 103.93270229399)
SIMEI ST 2 → (1.34585135529578, 103.955069793124)
TAMPINES CTRL 7 → (1.35892339007701, 103.940498383792)
TAMPINES ST 33 → (1.35330429046842, 103.957434964486)
TAMPINES ST 32 → (1.35336372874342, 103.955477865389)
TAMPINES AVE 5 → (1.37059544561685, 103.927380834135)
LOR 5 TOA PAYOH → (1.33865954756919, 103.855806480101)
LOR 7 TOA PAYOH → (1.33865954756919, 103.855806480101)
```

```
LOR 4 TOA PAYOH → (1.33233882778404, 103.851277086862)
LOR 1 TOA PAYOH → (1.33965126704796, 103.845485151552)
TOA PAYOH EAST → (1.33356606706026, 103.856981751253)
POTONG PASIR AVE 1 → (1.33315710123237, 103.868380398922)
TOA PAYOH NTH → (1.341790284928, 103.852280842345)
LOR 8 TOA PAYOH → (1.34074251155363, 103.857801875294)
LOR 3 TOA PAYOH → (1.33933820986766, 103.851224736347)
POTONG PASIR AVE 3 → (1.33447489186627, 103.866286539376)
JOO SENG RD → (1.33461833377097, 103.88016502252)
LOR 2 TOA PAYOH → (1.33669904090409, 103.846407081525)
TOA PAYOH CTRL → (1.3325746127759, 103.848267915547)
MARSILING DR → (1.44207297644105, 103.776353990531)
WOODLANDS ST 13 → (1.43598268540185, 103.781932747463)
WOODLANDS DR 52 → (1.43343677091965, 103.79805544789)
WOODLANDS ST 41 → (1.43044908253351, 103.772775923524)
MARSILING CRES → (1.44620708744186, 103.7736803529)
WOODLANDS ST 83 → (1.43947828856474, 103.790258027127)
WOODLANDS CIRCLE → (1.44409699510211, 103.80040459108)
WOODLANDS DR 40 → (1.44061701971721, 103.79664662375)
WOODLANDS ST 31 → (1.42958660649594, 103.774850125046)
WOODLANDS DR 16 → (1.43061338987334, 103.798862387555)
WOODLANDS ST 81 → (1.44168228349553, 103.785984689784)
WOODLANDS RING RD → (1.43672576257891, 103.797147612705)
WOODLANDS DR 53 → (1.43210416367253, 103.796490812112)
WOODLANDS DR 62 → (1.43953354132813, 103.803797276395)
WOODLANDS DR 70 → (1.44111273367565, 103.798010950745)
WOODLANDS DR 42 → (1.43864718901423, 103.796518065481)
WOODLANDS DR 50 → (1.43856288054116, 103.793634541557)
WOODLANDS AVE 6 → (1.44011341244035, 103.8021435578)
WOODLANDS DR 14 → (1.43318008766271, 103.791790437361)
WOODLANDS AVE 1 → (1.4561494204255, 103.801273599229)
WOODLANDS AVE 5 → (1.432723484346, 103.801172745628)
MARSILING RISE → (1.43872152075601, 103.781287430018)
WOODLANDS CRES → (1.44700426493785, 103.801889031334)
WOODLANDS DR 73 → (1.44075549422421, 103.804264107553)
WOODLANDS DR 44 → (1.43178572126544, 103.79448461206)
YISHUN RING RD → (1.4322213507358, 103.827555124846)
YISHUN AVE 3 → (1.42347167139153, 103.830623267044)
YISHUN ST 11 → (1.43080205677888, 103.833011168832)
YISHUN AVE 4 → (1.42053932360735, 103.840690129891)
YISHUN ST 22 → (1.43498006358563, 103.8389647186)
YISHUN ST 71 → (1.42621096565013, 103.827704108295)
YISHUN AVE 5 → (1.43113228990206, 103.828367142634)
YISHUN ST 21 → (1.43068689542817, 103.83657443377)
YISHUN ST 41 → (1.42033791627017, 103.844936184411)
YISHUN ST 61 → (1.41922603586175, 103.837390782849)
YISHUN AVE 6 → (1.43839562897006, 103.839309173817)
YISHUN AVE 11 → (1.4290305000563, 103.84430354208)
YISHUN CTRL → (1.42767464529376, 103.837731455467)
```

```
YISHUN ST 81 → (1.41177094390456, 103.83336826951)
YISHUN ST 72 → (1.42688859174215, 103.834067042338)
YISHUN AVE 2 → (1.44227547723252, 103.839701515305)
ANG MO KIO ST 32 → (1.36440439747367, 103.851810911935)
ANG MO KIO ST 31 → (1.36562478329901, 103.847011474539)
BEDOK NTH ST 2 → (1.33036165024598, 103.935081017472)
BEDOK NTH ST 1 → (1.33553203648306, 103.94919449094)
JLN TENAGA → (1.33167117872807, 103.906537912581)
BEDOK NTH ST 4 → (1.33016612297745, 103.940100315843)
BT BATOK WEST AVE 4 → (1.36485171911506, 103.745814559435)
CANTONMENT CL → (1.27547252623201, 103.839962631748)
BOON TIONG RD → (1.28652962097246, 103.833163087554)
SPOTTISWOODE PK RD → (1.27559528527257, 103.837559859708)
REDHILL CL → (1.28501998646245, 103.818577722823)
KIM TIAN PL → (1.28392508534814, 103.828263774087)
CASHEW RD → (1.37002390815881, 103.764505614295)
QUEEN'S RD → (1.32280086346268, 103.811023492628)
CHANDER RD → (1.30774014690879, 103.850826426805)
KELANTAN RD → (1.30613509115259, 103.856130240976)
SAGO LANE → (1.28180062242257, 103.842908159495)
UPP CROSS ST → (1.2862352148315, 103.842303340846)
CHIN SWEE RD → (1.28764537765146, 103.840270100813)
SMITH ST → (1.28211630828089, 103.842845624261)
TECK WHYE AVE → (1.38194642527068, 103.751888178683)
CHOA CHU KANG ST 51 → (1.39150729488856, 103.742746434118)
CHOA CHU KANG AVE 5 → (1.37584148701172, 103.737757429932)
CHOA CHU KANG AVE 1 → (1.3812909121038, 103.750296649613)
WEST COAST DR → (1.31129566365301, 103.758864434999)
PAYA LEBAR WAY → (1.32259415243858, 103.884296105282)
ALJUNIED CRES → (1.32095901223871, 103.884129395369)
JOO CHIAT RD → (1.30718803828074, 103.903829065777)
PINE CL → (1.30787056219408, 103.883192164927)
HOUGANG ST 22 → (1.35876258437852, 103.891460074318)
HOUGANG AVE 9 → (1.35981497140512, 103.888955970393)
HOUGANG AVE 6 → (1.36271446291047, 103.893980503167)
HOUGANG ST 21 → (1.35954177325561, 103.885167372726)
JURONG WEST ST 75 → (1.3461552094198, 103.701169369401)
KANG CHING RD → (1.33927933228555, 103.723441613129)
KG KAYU RD → (1.3955134347833, 103.873284976735)
CRAWFORD LANE → (1.30498797633055, 103.861150299806)
WHAMPOA WEST → (1.32050169556608, 103.863341271367)
BEACH RD → (1.29971949507987, 103.860829893192)
CAMBRIDGE RD → (1.31337582481667, 103.846416916995)
ST. GEORGE'S LANE → (1.32211971964107, 103.860820092196)
JELLICOE RD → (1.30889576732658, 103.863068603426)
ELIAS RD → (1.3727072529948, 103.939314655088)
HOLLAND CL → (1.30921937771543, 103.796732386696)
TANGLIN HALT RD → (1.29767708730608, 103.799636799806)
C'WEALTH AVE → (1.32300684559737, 103.761868274259)
```

```
WELLINGTON CIRCLE → (1.45218571290718, 103.821369887884)
CANBERRA LINK → (1.44778956988449, 103.824418843373)
SENGKANG WEST AVE → (1.39172552064567, 103.877000446561)
SENGKANG EAST RD → (1.38700703200166, 103.891964770037)
SERANGOON CTRL DR → (1.35403939780038, 103.870851534185)
SERANGOON AVE 3 → (1.37312860171966, 103.867345646638)
SERANGOON NTH AVE 3 → (1.3733324748714, 103.869907162546)
TAMPINES AVE 8 → (1.34972244708859, 103.927836018905)
TAMPINES ST 24 → (1.35443550096726, 103.952286197035)
TAMPINES ST 12 → (1.3477319115079, 103.94412402405)
SIMEI LANE → (1.3437195631364, 103.958424286264)
SIMEI ST 4 → (1.34120980441122, 103.956288849924)
LOR 6 TOA PAYOH → (1.3325746127759, 103.848267915547)
KIM KEAT LINK → (1.33050536301596, 103.855776378989)
MARSILING LANE → (1.44280576638619, 103.779298836184)
WOODLANDS ST 82 → (1.44202662604033, 103.789905781681)
WOODLANDS DR 60 → (1.44646333383129, 103.798737609119)
WOODLANDS AVE 3 → (1.43319140316565, 103.781594917057)
WOODLANDS DR 75 → (1.44100430753825, 103.807223345894)
WOODLANDS AVE 4 → (1.43389740955882, 103.795730926437)
WOODLANDS ST 32 → (1.43151585350574, 103.778881591946)
YISHUN AVE 7 → (1.4403210014766, 103.840575731125)
ANG MO KIO ST 11 → (1.37065009249377, 103.839410348811)
BISHAN ST 11 → (1.34455983329518, 103.855737419596)
BT BATOK WEST AVE 2 → (1.3626534376553, 103.744120311371)
BT BATOK ST 32 → (1.35964071966908, 103.748829089659)
BT BATOK ST 33 → (1.36139508011177, 103.746516655571)
BT BATOK ST 22 → (1.34461889355755, 103.749789011042)
BT BATOK WEST AVE 7 → (1.36470350565862, 103.744246776207)
HOY FATT RD → (1.28703560089104, 103.809955080725)
SILAT AVE → (1.27727696508456, 103.831093355841)
EVERTON PK → (1.37936398445683, 103.871804055428)
BT MERAH CTRL → (1.28347723784604, 103.813718091863)
JELEBU RD → (1.37925788735519, 103.762997101416)
EMPRESS RD → (1.3172499273924, 103.805978893931)
VEERASAMY RD → (1.30754470402275, 103.852892643539)
CHOA CHU KANG ST 64 → (1.39672603497195, 103.751817992962)
CHOA CHU KANG ST 53 → (1.39179994537682, 103.745621182307)
CHOA CHU KANG NTH 7 → (1.40047053163504, 103.746343470361)
CLEMENTI AVE 6 → (1.32014419030202, 103.763253531405)
CLEMENTI ST 13 → (1.32370061941892, 103.769671092663)
GEYLANG SERAI → (1.31649443408571, 103.896941669743)
JLN TIGA → (1.34138492547159, 103.959558271424)
ALJUNIED RD → (1.31370804131105, 103.881829577914)
YUNG LOH RD → (1.32767173847341, 103.722407547175)
YUNG AN RD → (1.33682896786791, 103.721117649246)
JLN MA'MOR → (1.32827853904357, 103.856097361404)
WHAMPOA RD → (1.32612552903646, 103.855672661254)
LOR 3 GEYLANG → (1.31266416584241, 103.876931602503)
```

```
PASIR RIS ST 13 → (1.36226599374262, 103.962078643998)
QUEEN'S CL → (1.29238272764571, 103.800196306719)
DOVER CL EAST → (1.30304255193392, 103.785358436408)
SEMBAWANG VISTA → (1.44686608385044, 103.82060863266)
TAMPINES ST 43 → (1.36103454930396, 103.952164422603)
SIMEI RD → (1.34469945127151, 103.958210105786)
KIM KEAT AVE → (1.33107979498474, 103.858966444055)
UPP ALJUNIED LANE → (1.33429242412799, 103.878746207181)
POTONG PASIR AVE 2 → (1.33290762111116, 103.866168845292)
WOODLANDS DR 72 → (1.44411840891704, 103.804444768882)
MARSILING RD → (1.43911200532501, 103.782256593639)
WOODLANDS DR 71 → (1.43882635430604, 103.799297445551)
YISHUN AVE 9 → (1.43163452931638, 103.83986734931)
YISHUN ST 20 → (1.43539224579712, 103.835500981499)
ANG MO KIO ST 21 → (1.3694605445784, 103.834659046016)
TIONG BAHRU RD → (1.28613693561234, 103.833027874167)
KLANG LANE → (1.30882069575537, 103.852065291651)
CHOA CHU KANG LOOP → (1.38409830722396, 103.74549899545)
CLEMENTI ST 14 → (1.32208936757694, 103.76893341249)
SIMS PL → (1.31659507864197, 103.879545641729)
JURONG EAST ST 31 → (1.34649072704441, 103.729053872207)
YUAN CHING RD → (1.34083828325568, 103.725098139838)
CORPORATION DR → (1.33538115974338, 103.722906605661)
YUNG PING RD → (1.32928599309236, 103.72238572895)
WHAMPOA STH → (1.32380442520685, 103.866839885662)
TESSENSOHN RD → (1.31649300843498, 103.856941466502)
JLN DUSUN → (1.3278687431847, 103.844209602863)
QUEENSWAY → (1.30205082145355, 103.800264607379)
FERNVALE LINK → (1.38943818402848, 103.87792381251)
KIM PONG RD → (1.28486466867448, 103.830500488707)
KIM CHENG ST → (1.28489922215081, 103.831584184636)
SAUJANA RD → (1.38233492640361, 103.767677663417)
BUFFALO RD → (1.30644303053168, 103.851349085178)
CLEMENTI ST 12 → (1.32271179028812, 103.76988810369)
DAKOTA CRES → (1.30634114614887, 103.884458183612)
JURONG WEST ST 51 → (1.34882313460497, 103.71923879406)
FRENCH RD → (1.30720611783037, 103.860733875195)
GLOUCESTER RD → (1.3135861663611, 103.85169992386)
KG ARANG RD → (1.34678772565959, 103.870408458138)
MOULMEIN RD → (1.31840833197495, 103.84861811451)
KENT RD → (1.29455312871495, 103.779693997471)
AH HOOD RD → (1.32717562161128, 103.846650410884)
SERANGOON NTH AVE 2 → (1.36725606646986, 103.874813498992)
TAMPINES CTRL 1 → (1.35264347882967, 103.942942106476)
TAMPINES AVE 7 → (1.3564470487275, 103.954542098775)
LOR 1A TOA PAYOH → (1.33669264885385, 103.844796096205)
WOODLANDS AVE 9 → (1.44427220470164, 103.783638071171)
YISHUN CTRL 1 → (1.42767464529376, 103.837731455467)
LOWER DELTA RD → (1.28083419778829, 103.823318065486)
JLN DUA → (1.30868190843585, 103.887123739313)
```

```
WOODLANDS ST 11 → (1.43359370586711, 103.775884057946)
ANG MO KIO AVE 2 → (1.37216802661963, 103.86656908416)
SELEGIE RD → (1.30376527977766, 103.849948910168)
SIMS AVE → (1.31429764072389, 103.877800568232)
REDHILL LANE → (1.29234264245612, 103.815172818084)
KING GEORGE'S AVE → (1.30856889161962, 103.85921493175)
PASIR RIS ST 41 → (1.37258743090674, 103.958120451767)
PUNGGOL WALK → (1.39184675107192, 103.913039039387)
LIM LIAK ST → (1.28549378363945, 103.832599980169)
JLN BERSEH → (1.30778509510261, 103.857010982845)
SENGKANG WEST WAY → (1.39527159096969, 103.880343031203)
BUANGKOK GREEN → (1.38170892235511, 103.887284803847)
SEMBAWANG WAY → (1.44954485621129, 103.81872759237)
PUNGGOL WAY → (1.40541353588128, 103.896823897352)
YISHUN ST 31 → (1.43208985750645, 103.846198333971)
TECK WHYE CRES → (1.38319029263484, 103.751878290333)
KRETA AYER RD → (1.28004979441591, 103.842481877778)
MONTREAL LINK → (1.45063212743043, 103.826554055194)
UPP SERANGOON CRES → (1.37811995342948, 103.90229625161)
SUMANG LINK → (1.41090728064356, 103.901186089898)
SENGKANG EAST AVE → (1.3809657561666, 103.900820298646)
YISHUN AVE 1 → (1.4403210014766, 103.840575731125)
ANCHORVALE CRES → (1.39901716420001, 103.89098971955)
YUNG KUANG RD → (1.33193786368443, 103.721561508725)
ANCHORVALE ST → (1.39635987610915, 103.889317071957)
TAMPINES CTRL 8 → (1.35673316454402, 103.940165250463)
YISHUN ST 51 → (1.41812574615209, 103.845416491236)
UPP SERANGOON VIEW → (1.37577531879135, 103.902561278398)
TAMPINES AVE 1 → (1.34342966240204, 103.932354333699)
BEDOK RESERVOIR CRES → (1.33534200645768, 103.920335116785)
ANG MO KIO ST 61 → (1.38095848597775, 103.841554486324)
DAWSON RD → (1.29386475464004, 103.810092969298)
FERNVALE ST → (1.39604094734393, 103.880735419984)
SENG POH RD → (1.28529691796456, 103.833235327463)
HOUGANG ST 32 → (1.36380223576131, 103.89378970225)
TAMPINES ST 86 → (1.35021013028761, 103.927274573665)
HENDERSON RD → (1.28408880599015, 103.819324172554)
SUMANG WALK → (1.40509726671603, 103.895362277221)
CHOA CHU KANG AVE 7 → (1.38224243360043, 103.738361221869)
KEAT HONG CL → (1.37524979404042, 103.743448796988)
JURONG WEST CTRL 3 → (1.34051876831483, 103.704549717701)
KEAT HONG LINK → (1.37480118403433, 103.749537925078)
ALJUNIED AVE 2 → (1.32052595935167, 103.886159306934)
SUMANG LANE → (1.4007698842262, 103.895484648396)
CANBERRA CRES → (1.4456906155288, 103.830296155027)
CANBERRA ST → (1.45209602014253, 103.830626737591)
ANG MO KIO ST 44 → (1.36866971395026, 103.857658686543)
WOODLANDS RISE → (1.44520176239086, 103.804828558584)
CANBERRA WALK → (1.44776611412274, 103.83222301056)
```

```
ANG MO KIO ST 51 → (1.37090382885863, 103.854421757506)
GEYLANG EAST AVE 2 → (1.31797271874983, 103.888778294506)
BT BATOK EAST AVE 6 → (1.3475199299187, 103.768367337416)
BT BATOK WEST AVE 9 → (1.35387904859721, 103.740677712222)
MARINE PARADE CTRL → (1.30134714783304, 103.907183612906)
MARGARET DR → (1.29482148558901, 103.813000857523)
TAMPINES ST 61 → (1.36162838057513, 103.938434393074)
YISHUN ST 43 → (1.42438879653642, 103.852083098516)
```

distance_to_mrt_m

is a continuous numerical variable. First, we will be loading data from mrt_Stations.csv which contains all the mrt stations existing in 2024 first month, and their longitudes latitudes coordinates. These coordinates will be used to obtain the distance between nearest MRT and the HDB flat.

```
from geopy.distance import geodesic
mrt_df = pd.read_csv(r"C:\Users\Crystaline\Downloads\
mrt_stations.csv")
mrt_df.columns = mrt_df.columns.str.strip()
mrt_df.rename(columns={'Station Name': 'station', 'latitude':
'mrt_lat', 'longitude': 'mrt_lon'}, inplace=True)

def get_nearest_mrt_distance(hdb_lat, hdb_lon, mrt_df):
    hdb_point = (hdb_lat, hdb_lon)
    distances = mrt_df.apply(lambda row: geodesic(hdb_point,
(row['mrt_lat'], row['mrt_lon'])).meters, axis=1)
    return distances.min()

df['distance_from_mrt_m'] = df.apply(
    lambda row: get_nearest_mrt_distance(row['latitude'],
row['longitude'], mrt_df), axis=1
)

df.rename(columns={'distance_from_mrt_m': 'distance_to_mrt_m'},
inplace=True)

for col in df.columns:
    print(f"□ Column: {col}")
    print(df[col].unique())
    print("-" * 80)
    print("\n")

□ Column: month
<DatetimeArray>
['2017-01-01 00:00:00', '2017-02-01 00:00:00', '2017-03-01 00:00:00',
 '2017-04-01 00:00:00', '2017-05-01 00:00:00', '2017-06-01 00:00:00',
 '2017-07-01 00:00:00', '2017-08-01 00:00:00', '2017-09-01 00:00:00',
```

```
 '2017-10-01 00:00:00', '2017-11-01 00:00:00', '2017-12-01 00:00:00',
 '2018-01-01 00:00:00', '2018-02-01 00:00:00', '2018-03-01 00:00:00',
 '2018-04-01 00:00:00', '2018-05-01 00:00:00', '2018-06-01 00:00:00',
 '2018-07-01 00:00:00', '2018-08-01 00:00:00', '2018-09-01 00:00:00',
 '2018-10-01 00:00:00', '2018-11-01 00:00:00', '2018-12-01 00:00:00',
 '2019-01-01 00:00:00', '2019-02-01 00:00:00', '2019-03-01 00:00:00',
 '2019-04-01 00:00:00', '2019-05-01 00:00:00', '2019-06-01 00:00:00',
 '2019-07-01 00:00:00', '2019-08-01 00:00:00', '2019-09-01 00:00:00',
 '2019-10-01 00:00:00', '2019-11-01 00:00:00', '2019-12-01 00:00:00',
 '2020-01-01 00:00:00', '2020-02-01 00:00:00', '2020-03-01 00:00:00',
 '2020-04-01 00:00:00', '2020-05-01 00:00:00', '2020-06-01 00:00:00',
 '2020-07-01 00:00:00', '2020-08-01 00:00:00', '2020-09-01 00:00:00',
 '2020-10-01 00:00:00', '2020-11-01 00:00:00', '2020-12-01 00:00:00',
 '2021-01-01 00:00:00', '2021-02-01 00:00:00', '2021-03-01 00:00:00',
 '2021-04-01 00:00:00', '2021-05-01 00:00:00', '2021-06-01 00:00:00',
 '2021-11-01 00:00:00', '2021-07-01 00:00:00', '2021-08-01 00:00:00',
 '2021-09-01 00:00:00', '2021-10-01 00:00:00', '2021-12-01 00:00:00',
 '2022-01-01 00:00:00', '2022-02-01 00:00:00', '2022-03-01 00:00:00',
 '2022-04-01 00:00:00', '2022-05-01 00:00:00', '2022-06-01 00:00:00',
 '2022-07-01 00:00:00', '2022-08-01 00:00:00', '2022-09-01 00:00:00',
 '2022-10-01 00:00:00', '2022-11-01 00:00:00', '2022-12-01 00:00:00',
 '2023-01-01 00:00:00', '2023-02-01 00:00:00', '2023-03-01 00:00:00',
 '2023-04-01 00:00:00', '2023-05-01 00:00:00', '2023-06-01 00:00:00',
 '2023-07-01 00:00:00', '2023-08-01 00:00:00', '2023-09-01 00:00:00',
 '2023-10-01 00:00:00', '2023-11-01 00:00:00', '2023-12-01 00:00:00']
Length: 84, dtype: datetime64[ns]
----------------------------------------------------------------------
----------


□ Column: town
['ANG MO KIO' 'BEDOK' 'BISHAN' 'BUKIT BATOK' 'BUKIT MERAH' 'BUKIT
PANJANG'
 'BUKIT TIMAH' 'CENTRAL AREA' 'CHOA CHU KANG' 'CLEMENTI' 'GEYLANG'
 'HOUGANG' 'JURONG EAST' 'JURONG WEST' 'KALLANG/WHAMPOA' 'MARINE
PARADE'
 'PASIR RIS' 'PUNGGOL' 'QUEENSTOWN' 'SEMBAWANG' 'SENGKANG' 'SERANGOON'
 'TAMPINES' 'TOA PAYOH' 'WOODLANDS' 'YISHUN']
----------------------------------------------------------------------
----------


□ Column: flat_type
['2 ROOM' '3 ROOM' '4 ROOM' '5 ROOM' 'EXECUTIVE' '1 ROOM'
 'MULTI-GENERATION']
----------------------------------------------------------------------
----------


□ Column: block
```

```
['406' '108' '602' ... '860B' '605A' '605C']
----------------------------------------------------------------------
----------


□ Column: street_name
['ANG MO KIO AVE 10' 'ANG MO KIO AVE 4' 'ANG MO KIO AVE 5'
 'ANG MO KIO AVE 1' 'ANG MO KIO AVE 3' 'ANG MO KIO AVE 9'
 'ANG MO KIO AVE 8' 'ANG MO KIO AVE 6' 'ANG MO KIO ST 52'
 'BEDOK NTH AVE 4' 'BEDOK NTH AVE 1' 'BEDOK NTH RD' 'BEDOK STH AVE 1'
 'BEDOK RESERVOIR RD' 'CHAI CHEE ST' 'BEDOK NTH ST 3' 'BEDOK STH RD'
 'CHAI CHEE AVE' 'NEW UPP CHANGI RD' 'CHAI CHEE DR' 'BEDOK STH AVE 2'
 'BEDOK NTH AVE 3' 'BEDOK RESERVOIR VIEW' 'CHAI CHEE RD' 'LENGKONG
TIGA'
 'BEDOK CTRL' 'JLN DAMAI' 'BEDOK NTH AVE 2' 'BEDOK STH AVE 3'
 'SIN MING RD' 'SIN MING AVE' 'BISHAN ST 12' 'BISHAN ST 13' 'BISHAN ST
22'
 'BISHAN ST 24' 'BISHAN ST 23' 'BRIGHT HILL DR' 'SHUNFU RD'
 'BT BATOK ST 34' 'BT BATOK ST 51' 'BT BATOK ST 11' 'BT BATOK ST 52'
 'BT BATOK ST 21' 'BT BATOK EAST AVE 5' 'BT BATOK WEST AVE 6'
 'BT BATOK CTRL' 'BT BATOK WEST AVE 8' 'BT BATOK EAST AVE 4'
 'BT BATOK ST 31' 'BT BATOK ST 25' 'BT BATOK EAST AVE 3'
 'BT BATOK WEST AVE 5' 'BT BATOK ST 24' 'JLN BT HO SWEE'
 'TELOK BLANGAH DR' 'BEO CRES' 'TELOK BLANGAH CRES' 'TAMAN HO SWEE'
 'TELOK BLANGAH RISE' 'TELOK BLANGAH WAY' 'JLN BT MERAH' 'JLN KLINIK'
 'TELOK BLANGAH HTS' 'BT MERAH VIEW' 'INDUS RD' 'BT MERAH LANE 1'
 'TELOK BLANGAH ST 31' 'MOH GUAN TER' 'HAVELOCK RD' 'HENDERSON CRES'
 'BT PURMEI RD' 'KIM TIAN RD' 'DEPOT RD' 'JLN RUMAH TINGGI' 'DELTA
AVE'
 'JLN MEMBINA' 'REDHILL RD' 'LENGKOK BAHRU' 'ZION RD' 'PETIR RD'
 'PENDING RD' 'BANGKIT RD' 'SEGAR RD' 'JELAPANG RD' 'SENJA RD' 'FAJAR
RD'
 'BT PANJANG RING RD' 'SENJA LINK' 'LOMPANG RD' 'GANGSA RD' 'TOH YI
DR'
 'FARRER RD' 'JLN KUKOH' 'ROWELL RD' 'WATERLOO ST' 'NEW MKT RD'
 'TG PAGAR PLAZA' 'QUEEN ST' 'BAIN ST' 'CANTONMENT RD' 'TECK WHYE
LANE'
 'CHOA CHU KANG AVE 4' 'CHOA CHU KANG AVE 3' 'CHOA CHU KANG CRES'
 'CHOA CHU KANG ST 54' 'CHOA CHU KANG CTRL' 'JLN TECK WHYE'
 'CHOA CHU KANG ST 62' 'CHOA CHU KANG NTH 6' 'CHOA CHU KANG DR'
 'CHOA CHU KANG NTH 5' 'CHOA CHU KANG ST 52' 'CHOA CHU KANG AVE 2'
 'CLEMENTI WEST ST 2' 'WEST COAST RD' 'CLEMENTI WEST ST 1'
 'CLEMENTI AVE 4' 'CLEMENTI AVE 5' 'CLEMENTI ST 11' 'CLEMENTI AVE 2'
 'CLEMENTI AVE 3' 'CLEMENTI AVE 1' "C'WEALTH AVE WEST" 'CIRCUIT RD'
 'BALAM RD' 'MACPHERSON LANE' 'EUNOS CRES' 'UBI AVE 1' 'HAIG RD'
 'OLD AIRPORT RD' 'GEYLANG EAST AVE 1' 'SIMS DR' 'PIPIT RD'
 'GEYLANG EAST CTRL' 'EUNOS RD 5' 'CASSIA CRES' 'BUANGKOK CRES'
 'HOUGANG AVE 3' 'HOUGANG AVE 8' 'HOUGANG AVE 1' 'HOUGANG AVE 5'
 'HOUGANG ST 61' 'HOUGANG ST 11' 'HOUGANG AVE 7' 'HOUGANG AVE 4'
```

```
 'HOUGANG AVE 2' 'LOR AH SOO' 'HOUGANG ST 92' 'HOUGANG ST 52'
 'HOUGANG AVE 10' 'HOUGANG ST 51' 'UPP SERANGOON RD' 'HOUGANG CTRL'
 'HOUGANG ST 91' 'BUANGKOK LINK' 'HOUGANG ST 31' 'PANDAN GDNS'
 'TEBAN GDNS RD' 'JURONG EAST ST 24' 'JURONG EAST ST 21'
 'JURONG EAST AVE 1' 'JURONG EAST ST 13' 'JURONG EAST ST 32' 'TOH GUAN
RD'
 'JURONG WEST ST 93' 'BOON LAY AVE' 'HO CHING RD' 'BOON LAY DR'
 'TAO CHING RD' 'JURONG WEST ST 91' 'JURONG WEST ST 42'
 'JURONG WEST ST 92' 'BOON LAY PL' 'JURONG WEST ST 52' 'TAH CHING RD'
 'JURONG WEST ST 81' 'YUNG SHENG RD' 'JURONG WEST ST 25'
 'JURONG WEST ST 73' 'JURONG WEST ST 72' 'JURONG WEST AVE 3'
 'JURONG WEST AVE 5' 'YUNG HO RD' 'JURONG WEST ST 74' 'JURONG WEST AVE
1'
 'JURONG WEST ST 71' 'JURONG WEST ST 61' 'JURONG WEST ST 65'
 'JURONG WEST CTRL 1' 'JURONG WEST ST 64' 'JURONG WEST ST 62'
 'JURONG WEST ST 41' 'JURONG WEST ST 24' 'JLN BATU' 'JLN BAHAGIA'
 'LOR LIMAU' "ST. GEORGE'S RD" 'KALLANG BAHRU' 'DORSET RD' 'GEYLANG
BAHRU'
 'BENDEMEER RD' 'WHAMPOA DR' 'UPP BOON KENG RD' 'RACE COURSE RD' 'OWEN
RD'
 'NTH BRIDGE RD' 'TOWNER RD' 'FARRER PK RD' 'MCNAIR RD' 'JLN TENTERAM'
 'BOON KENG RD' 'JLN RAJAH' 'MARINE DR' 'MARINE CRES' 'MARINE TER'
 'CHANGI VILLAGE RD' 'PASIR RIS ST 71' 'PASIR RIS ST 11' 'PASIR RIS DR
3'
 'PASIR RIS DR 6' 'PASIR RIS ST 21' 'PASIR RIS DR 4' 'PASIR RIS ST 53'
 'PASIR RIS DR 10' 'PASIR RIS ST 52' 'PASIR RIS ST 12' 'PASIR RIS ST
51'
 'PASIR RIS ST 72' 'PASIR RIS DR 1' 'PUNGGOL FIELD' 'EDGEDALE PLAINS'
 'PUNGGOL FIELD WALK' 'EDGEFIELD PLAINS' 'PUNGGOL RD' 'PUNGGOL EAST'
 'PUNGGOL DR' 'PUNGGOL CTRL' 'PUNGGOL PL' "C'WEALTH CL" 'STIRLING RD'
 'MEI LING ST' "C'WEALTH CRES" "C'WEALTH DR" 'GHIM MOH RD' 'DOVER RD'
 'HOLLAND AVE' 'STRATHMORE AVE' 'HOLLAND DR' 'GHIM MOH LINK'
 'CLARENCE LANE' 'DOVER CRES' 'SEMBAWANG DR' 'SEMBAWANG CL' 'MONTREAL
DR'
 'ADMIRALTY LINK' 'ADMIRALTY DR' 'SEMBAWANG CRES' 'CANBERRA RD'
 'FERNVALE RD' 'COMPASSVALE LANE' 'ANCHORVALE RD' 'RIVERVALE DR'
 'RIVERVALE CRES' 'SENGKANG EAST WAY' 'RIVERVALE ST' 'RIVERVALE WALK'
 'FERNVALE LANE' 'ANCHORVALE LINK' 'COMPASSVALE RD' 'COMPASSVALE CRES'
 'JLN KAYU' 'COMPASSVALE WALK' 'COMPASSVALE DR' 'COMPASSVALE LINK'
 'COMPASSVALE BOW' 'SENGKANG CTRL' 'ANCHORVALE LANE' 'ANCHORVALE DR'
 'COMPASSVALE ST' 'SERANGOON AVE 4' 'LOR LEW LIAN' 'SERANGOON AVE 2'
 'SERANGOON NTH AVE 1' 'SERANGOON AVE 1' 'SERANGOON CTRL'
 'SERANGOON NTH AVE 4' 'TAMPINES ST 22' 'TAMPINES ST 41' 'TAMPINES AVE
4'
 'TAMPINES ST 44' 'TAMPINES ST 81' 'TAMPINES ST 11' 'TAMPINES ST 23'
 'TAMPINES ST 91' 'TAMPINES ST 21' 'TAMPINES ST 83' 'TAMPINES ST 42'
 'TAMPINES ST 71' 'TAMPINES ST 45' 'TAMPINES ST 34' 'TAMPINES ST 82'
 'TAMPINES AVE 9' 'SIMEI ST 1' 'SIMEI ST 5' 'TAMPINES ST 72'
 'TAMPINES ST 84' 'SIMEI ST 2' 'TAMPINES CTRL 7' 'TAMPINES ST 33'
```

```
 'TAMPINES ST 32' 'TAMPINES AVE 5' 'LOR 5 TOA PAYOH' 'LOR 7 TOA PAYOH'
 'LOR 4 TOA PAYOH' 'LOR 1 TOA PAYOH' 'TOA PAYOH EAST' 'POTONG PASIR
AVE 1'
 'TOA PAYOH NTH' 'LOR 8 TOA PAYOH' 'LOR 3 TOA PAYOH' 'POTONG PASIR AVE
3'
 'JOO SENG RD' 'LOR 2 TOA PAYOH' 'TOA PAYOH CTRL' 'MARSILING DR'
 'WOODLANDS ST 13' 'WOODLANDS DR 52' 'WOODLANDS ST 41' 'MARSILING
CRES'
 'WOODLANDS ST 83' 'WOODLANDS CIRCLE' 'WOODLANDS DR 40' 'WOODLANDS ST
31'
 'WOODLANDS DR 16' 'WOODLANDS ST 81' 'WOODLANDS RING RD' 'WOODLANDS DR
53'
 'WOODLANDS DR 62' 'WOODLANDS DR 70' 'WOODLANDS DR 42' 'WOODLANDS DR
50'
 'WOODLANDS AVE 6' 'WOODLANDS DR 14' 'WOODLANDS AVE 1' 'WOODLANDS AVE
5'
 'MARSILING RISE' 'WOODLANDS CRES' 'WOODLANDS DR 73' 'WOODLANDS DR 44'
 'YISHUN RING RD' 'YISHUN AVE 3' 'YISHUN ST 11' 'YISHUN AVE 4'
 'YISHUN ST 22' 'YISHUN ST 71' 'YISHUN AVE 5' 'YISHUN ST 21'
 'YISHUN ST 41' 'YISHUN ST 61' 'YISHUN AVE 6' 'YISHUN AVE 11'
 'YISHUN CTRL' 'YISHUN ST 81' 'YISHUN ST 72' 'YISHUN AVE 2'
 'ANG MO KIO ST 32' 'ANG MO KIO ST 31' 'BEDOK NTH ST 2' 'BEDOK NTH ST
1'
 'JLN TENAGA' 'BEDOK NTH ST 4' 'BT BATOK WEST AVE 4' 'CANTONMENT CL'
 'BOON TIONG RD' 'SPOTTISWOODE PK RD' 'REDHILL CL' 'KIM TIAN PL'
 'CASHEW RD' "QUEEN'S RD" 'CHANDER RD' 'KELANTAN RD' 'SAGO LANE'
 'UPP CROSS ST' 'CHIN SWEE RD' 'SMITH ST' 'TECK WHYE AVE'
 'CHOA CHU KANG ST 51' 'CHOA CHU KANG AVE 5' 'CHOA CHU KANG AVE 1'
 'WEST COAST DR' 'PAYA LEBAR WAY' 'ALJUNIED CRES' 'JOO CHIAT RD' 'PINE
CL'
 'HOUGANG ST 22' 'HOUGANG AVE 9' 'HOUGANG AVE 6' 'HOUGANG ST 21'
 'JURONG WEST ST 75' 'KANG CHING RD' 'KG KAYU RD' 'CRAWFORD LANE'
 'WHAMPOA WEST' 'BEACH RD' 'CAMBRIDGE RD' "ST. GEORGE'S LANE"
 'JELLICOE RD' 'ELIAS RD' 'HOLLAND CL' 'TANGLIN HALT RD' "C'WEALTH
AVE"
 'WELLINGTON CIRCLE' 'CANBERRA LINK' 'SENGKANG WEST AVE'
 'SENGKANG EAST RD' 'SERANGOON CTRL DR' 'SERANGOON AVE 3'
 'SERANGOON NTH AVE 3' 'TAMPINES AVE 8' 'TAMPINES ST 24' 'TAMPINES ST
12'
 'SIMEI LANE' 'SIMEI ST 4' 'LOR 6 TOA PAYOH' 'KIM KEAT LINK'
 'MARSILING LANE' 'WOODLANDS ST 82' 'WOODLANDS DR 60' 'WOODLANDS AVE
3'
 'WOODLANDS DR 75' 'WOODLANDS AVE 4' 'WOODLANDS ST 32' 'YISHUN AVE 7'
 'ANG MO KIO ST 11' 'BISHAN ST 11' 'BT BATOK WEST AVE 2' 'BT BATOK ST
32'
 'BT BATOK ST 33' 'BT BATOK ST 22' 'BT BATOK WEST AVE 7' 'HOY FATT RD'
 'SILAT AVE' 'EVERTON PK' 'BT MERAH CTRL' 'JELEBU RD' 'EMPRESS RD'
 'VEERASAMY RD' 'CHOA CHU KANG ST 64' 'CHOA CHU KANG ST 53'
 'CHOA CHU KANG NTH 7' 'CLEMENTI AVE 6' 'CLEMENTI ST 13' 'GEYLANG
```

```
SERAI'
 'JLN TIGA' 'ALJUNIED RD' 'YUNG LOH RD' 'YUNG AN RD' "JLN MA'MOR"
 'WHAMPOA RD' 'LOR 3 GEYLANG' 'PASIR RIS ST 13' "QUEEN'S CL"
 'DOVER CL EAST' 'SEMBAWANG VISTA' 'TAMPINES ST 43' 'SIMEI RD'
 'KIM KEAT AVE' 'UPP ALJUNIED LANE' 'POTONG PASIR AVE 2' 'WOODLANDS DR
72'
 'MARSILING RD' 'WOODLANDS DR 71' 'YISHUN AVE 9' 'YISHUN ST 20'
 'ANG MO KIO ST 21' 'TIONG BAHRU RD' 'KLANG LANE' 'CHOA CHU KANG LOOP'
 'CLEMENTI ST 14' 'SIMS PL' 'JURONG EAST ST 31' 'YUAN CHING RD'
 'CORPORATION DR' 'YUNG PING RD' 'WHAMPOA STH' 'TESSENSOHN RD' 'JLN
DUSUN'
 'QUEENSWAY' 'FERNVALE LINK' 'KIM PONG RD' 'KIM CHENG ST' 'SAUJANA RD'
 'BUFFALO RD' 'CLEMENTI ST 12' 'DAKOTA CRES' 'JURONG WEST ST 51'
 'FRENCH RD' 'GLOUCESTER RD' 'KG ARANG RD' 'MOULMEIN RD' 'KENT RD'
 'AH HOOD RD' 'SERANGOON NTH AVE 2' 'TAMPINES CTRL 1' 'TAMPINES AVE 7'
 'LOR 1A TOA PAYOH' 'WOODLANDS AVE 9' 'YISHUN CTRL 1' 'LOWER DELTA RD'
 'JLN DUA' 'WOODLANDS ST 11' 'ANG MO KIO AVE 2' 'SELEGIE RD' 'SIMS
AVE'
 'REDHILL LANE' "KING GEORGE'S AVE" 'PASIR RIS ST 41' 'PUNGGOL WALK'
 'LIM LIAK ST' 'JLN BERSEH' 'SENGKANG WEST WAY' 'BUANGKOK GREEN'
 'SEMBAWANG WAY' 'PUNGGOL WAY' 'YISHUN ST 31' 'TECK WHYE CRES'
 'KRETA AYER RD' 'MONTREAL LINK' 'UPP SERANGOON CRES' 'SUMANG LINK'
 'SENGKANG EAST AVE' 'YISHUN AVE 1' 'ANCHORVALE CRES' 'YUNG KUANG RD'
 'ANCHORVALE ST' 'TAMPINES CTRL 8' 'YISHUN ST 51' 'UPP SERANGOON VIEW'
 'TAMPINES AVE 1' 'BEDOK RESERVOIR CRES' 'ANG MO KIO ST 61' 'DAWSON
RD'
 'FERNVALE ST' 'SENG POH RD' 'HOUGANG ST 32' 'TAMPINES ST 86'
 'HENDERSON RD' 'SUMANG WALK' 'CHOA CHU KANG AVE 7' 'KEAT HONG CL'
 'JURONG WEST CTRL 3' 'KEAT HONG LINK' 'ALJUNIED AVE 2' 'SUMANG LANE'
 'CANBERRA CRES' 'CANBERRA ST' 'ANG MO KIO ST 44' 'WOODLANDS RISE'
 'CANBERRA WALK' 'ANG MO KIO ST 51' 'GEYLANG EAST AVE 2'
 'BT BATOK EAST AVE 6' 'BT BATOK WEST AVE 9' 'MARINE PARADE CTRL'
 'MARGARET DR' 'TAMPINES ST 61' 'YISHUN ST 43']
----------------------------------------------------------------------
----------


 Column: storey_range
['10 TO 12' '01 TO 03' '04 TO 06' '07 TO 09' '13 TO 15' '19 TO 21'
 '22 TO 24' '16 TO 18' '34 TO 36' '28 TO 30' '37 TO 39' '49 TO 51'
 '25 TO 27' '40 TO 42' '31 TO 33' '46 TO 48' '43 TO 45']
----------------------------------------------------------------------
----------


 Column: floor_area_sqm
[ 44.   67.   68.   73.   74.   82.   81.   92.   91.   94.   98.
97.
  99.   90.  117.  119.  118.  112.  121.  147.   45.   59.   63.
70.
```

```
  60.   65.   75.   66.   84.   93.  104.  105.  120.  130.  132.
115.
 122.  137.  139.  143.  146.  145.  141.   64.   83.  108.   95.
123.
  69.  103.  102.  100.  107.   86.  101.  150.  155.  144.   34.
51.
  54.   58.   76.   88.   77.  106.   85.   89.  134.  110.  111.
151.
  55.  113.  126.  124.  131.  142.   42.   46.   56.   61.   57.
72.
 109.   47.   96.  116.  128.  140.  148.  156.  157.   71.   52.
79.
 129.  133.  125.   48.   62.  114.   87.  127.  161.  165.   50.
153.
  43.  138.  164.  163.  136.  149.   80.  154.  152.   37.   78.
135.
 170.  192.  182.   31.   49.   53.   60.3 176.  177.  189.   40.
166.
 184.  173.  169.  181.  158.   41.  159.  215.  174.   63.1 179.
162.
  83.1 172.  168.  160.  249.  185.   38.  178.  171.  237.  183.
190.
 175.  188.  187.   35.  186.   39.  243.  199.  222.  210.  241.
167.
 180.  100.2]
----------------------------------------------------------------------
----------


 Column: flat_model
['Improved' 'New Generation' 'DBSS' 'Standard' 'Apartment'
'Simplified'
 'Model A' 'Premium Apartment' 'Adjoined flat' 'Model A-Maisonette'
 'Maisonette' 'Type S1' 'Type S2' 'Model A2' 'Terrace'
 'Improved-Maisonette' 'Premium Maisonette' 'Multi Generation'
 'Premium Apartment Loft' '2-room' '3Gen']
----------------------------------------------------------------------
----------


 Column: lease_commence_date
[1979 1978 1980 1981 1976 1977 2011 2012 1996 1988 1985 1986 1974 1984
 1983 1987 1982 2000 2001 2005 1989 2010 1972 1993 1973 1992 1990 1998
 2004 1997 1971 1975 1970 1969 2013 2008 1999 2003 2002 1995 2006 1967
 1968 2007 1991 1966 2009 1994 2014 2015 2016 2017 2018 2019 2022
2020]
----------------------------------------------------------------------
----------
```

```
□ Column: remaining_lease
['61 years 04 months' '60 years 07 months' '62 years 05 months'
 '62 years 01 month' '63 years' '61 years 06 months' '58 years 04
months'
 '59 years 08 months' '59 years 06 months' '60 years' '62 years 08
months'
 '61 years' '60 years 10 months' '59 years 03 months' '61 years 05
months'
 '60 years 04 months' '62 years' '60 years 03 months' '63 years 09
months'
 '61 years 01 month' '61 years 10 months' '58 years 06 months'
 '59 years 04 months' '62 years 11 months' '60 years 08 months'
 '93 years 08 months' '93 years 07 months' '60 years 01 month'
 '94 years 08 months' '78 years 04 months' '60 years 06 months'
 '62 years 06 months' '58 years' '70 years 08 months' '63 years 04
months'
 '63 years 06 months' '67 years 07 months' '61 years 07 months'
 '68 years 02 months' '68 years 03 months' '56 years' '67 years 09
months'
 '67 years 05 months' '63 years 07 months' '66 years 03 months'
 '65 years 04 months' '69 years 05 months' '59 years 11 months'
 '60 years 05 months' '69 years 02 months' '69 years 03 months'
 '68 years 10 months' '62 years 10 months' '64 years 04 months'
 '66 years 01 month' '83 years' '83 years 01 month' '87 years 11
months'
 '71 years 02 months' '92 years 04 months' '54 years 06 months'
 '78 years 06 months' '82 years 11 months' '75 years 04 months'
 '66 years 07 months' '66 years 06 months' '75 years 11 months'
 '68 years 04 months' '55 years 09 months' '68 years 07 months'
 '67 years 11 months' '68 years' '69 years 01 month' '69 years 11
months'
 '74 years 06 months' '74 years 04 months' '69 years 06 months'
 '72 years 03 months' '67 years 02 months' '66 years 05 months'
 '69 years 04 months' '66 years 11 months' '66 years 10 months' '80
years'
 '69 years 08 months' '66 years 09 months' '67 years 10 months'
 '80 years 01 month' '67 years 06 months' '86 years 08 months'
 '71 years 06 months' '71 years 03 months' '67 years 04 months'
 '86 years 11 months' '86 years 10 months' '79 years 05 months'
 '65 years 10 months' '67 years 03 months' '79 years 11 months'
 '53 years 06 months' '57 years 02 months' '52 years 01 month'
 '58 years 03 months' '51 years 06 months' '58 years 08 months'
 '56 years 02 months' '53 years 08 months' '64 years 08 months'
 '55 years 06 months' '95 years 07 months' '55 years 01 month' '55
years'
 '95 years 04 months' '52 years 06 months' '57 years 04 months' '57
years'
 '82 years 06 months' '67 years 08 months' '79 years' '95 years 08
months'
```

```
 '90 years 11 months' '87 years 10 months' '82 years 08 months'
 '68 years 06 months' '81 years 02 months' '56 years 05 months'
 '65 years 05 months' '70 years 09 months' '71 years 01 month'
 '94 years 10 months' '80 years 03 months' '83 years 11 months'
 '70 years 10 months' '81 years 01 month' '70 years 06 months' '85
years'
 '81 years 05 months' '80 years 07 months' '80 years 10 months'
 '83 years 10 months' '86 years 09 months' '79 years 09 months'
 '84 years 10 months' '80 years 11 months' '70 years 11 months'
 '84 years 04 months' '83 years 09 months' '81 years 04 months'
 '79 years 07 months' '81 years 03 months' '70 years 01 month'
 '70 years 05 months' '70 years 07 months' '56 years 03 months' '64
years'
 '68 years 09 months' '65 years 03 months' '59 years 01 month'
 '61 years 02 months' '62 years 04 months' '93 years' '71 years 08
months'
 '82 years 03 months' '71 years 04 months' '75 years 05 months'
 '85 years 01 month' '81 years 09 months' '72 years 08 months'
 '75 years 06 months' '78 years 03 months' '82 years 01 month'
 '84 years 03 months' '77 years 05 months' '77 years 07 months'
 '85 years 04 months' '79 years 04 months' '78 years 09 months'
 '81 years 06 months' '78 years' '63 years 08 months' '62 years 07
months'
 '65 years 09 months' '60 years 02 months' '88 years 10 months' '53
years'
 '49 years' '51 years' '50 years 05 months' '54 years 11 months'
 '59 years 05 months' '82 years 04 months' '65 years 07 months' '89
years'
 '72 years 05 months' '78 years 05 months' '58 years 07 months'
 '80 years 05 months' '94 years 04 months' '69 years' '65 years'
 '65 years 02 months' '71 years 07 months' '75 years 03 months'
 '65 years 01 month' '79 years 10 months' '73 years 04 months'
 '72 years 04 months' '74 years 08 months' '74 years 09 months'
 '80 years 02 months' '81 years 10 months' '86 years 05 months'
 '85 years 06 months' '79 years 01 month' '80 years 06 months'
 '74 years 11 months' '68 years 08 months' '68 years 05 months'
 '48 years 09 months' '63 years 05 months' '65 years 06 months'
 '66 years 04 months' '67 years 01 month' '95 years 09 months'
 '80 years 09 months' '82 years 02 months' '61 years 09 months'
 '90 years 07 months' '57 years 09 months' '54 years' '53 years 05
months'
 '71 years' '58 years 10 months' '92 years 03 months' '66 years 02
months'
 '64 years 03 months' '72 years 02 months' '83 years 04 months'
 '78 years 07 months' '84 years 01 month' '78 years 08 months'
 '72 years 10 months' '71 years 11 months' '76 years' '78 years 02
months'
 '71 years 05 months' '72 years 07 months' '83 years 03 months' '67
years'
```

```
 '72 years 11 months' '90 years 08 months' '84 years 11 months'
 '84 years 09 months' '82 years 07 months' '66 years 08 months'
 '88 years 04 months' '85 years 10 months' '90 years 05 months'
 '90 years 06 months' '83 years 05 months' '84 years 02 months'
 '82 years 09 months' '82 years 05 months' '73 years' '84 years 06
months'
 '83 years 07 months' '52 years 05 months' '56 years 08 months'
 '56 years 06 months' '68 years 01 month' '64 years 10 months'
 '57 years 01 month' '86 years 06 months' '88 years 07 months'
 '93 years 11 months' '58 years 02 months' '78 years 10 months'
 '77 years 08 months' '75 years 07 months' '77 years 09 months'
 '75 years 08 months' '77 years 04 months' '85 years 07 months' '95
years'
 '89 years 07 months' '91 years 02 months' '94 years 11 months'
 '85 years 11 months' '94 years 06 months' '94 years 07 months'
 '95 years 01 month' '95 years 02 months' '95 years 06 months'
 '93 years 02 months' '86 years 03 months' '87 years' '86 years 04
months'
 '85 years 03 months' '86 years 02 months' '85 years 09 months' '52
years'
 '49 years 01 month' '57 years 06 months' '56 years 04 months'
 '95 years 03 months' '88 years 11 months' '59 years' '85 years 08
months'
 '92 years 07 months' '83 years 06 months' '83 years 02 months'
 '81 years 11 months' '84 years' '94 years 02 months' '94 years 01
month'
 '85 years 05 months' '85 years 02 months' '83 years 08 months'
 '95 years 10 months' '93 years 10 months' '81 years' '95 years 05
months'
 '88 years 01 month' '94 years 09 months' '92 years 02 months'
 '90 years 04 months' '91 years 11 months' '91 years 07 months'
 '84 years 08 months' '84 years 05 months' '71 years 10 months'
 '74 years 10 months' '69 years 09 months' '79 years 02 months'
 '64 years 09 months' '79 years 03 months' '79 years 06 months'
 '79 years 08 months' '90 years' '69 years 10 months' '77 years 10
months'
 '49 years 05 months' '52 years 07 months' '54 years 07 months' '88
years'
 '91 years 03 months' '59 years 10 months' '62 years 09 months'
 '78 years 01 month' '71 years 09 months' '81 years 08 months'
 '78 years 11 months' '74 years 05 months' '66 years' '70 years'
 '70 years 02 months' '77 years 02 months' '70 years 03 months'
 '69 years 07 months' '63 years 02 months' '59 years 02 months'
 '61 years 11 months' '61 years 03 months' '60 years 11 months'
 '75 years 02 months' '76 years 10 months' '57 years 11 months'
 '54 years 05 months' '62 years 03 months' '59 years 07 months'
 '62 years 02 months' '64 years 05 months' '63 years 03 months'
 '82 years 10 months' '64 years 11 months' '72 years' '74 years 03
months'
```

```
 '80 years 04 months' '93 years 06 months' '68 years 11 months'
 '70 years 04 months' '86 years 07 months' '57 years 03 months'
 '49 years 04 months' '51 years 05 months' '52 years 11 months'
 '55 years 05 months' '56 years 11 months' '87 years 08 months'
 '61 years 08 months' '76 years 11 months' '77 years 06 months'
 '84 years 07 months' '87 years 01 month' '59 years 09 months'
 '92 years 11 months' '77 years' '76 years 09 months' '88 years 05
months'
 '50 years 11 months' '48 years 11 months' '58 years 05 months'
 '57 years 05 months' '94 years 03 months' '74 years 07 months'
 '64 years 07 months' '64 years 01 month' '80 years 08 months'
 '57 years 08 months' '58 years 09 months' '53 years 11 months'
 '55 years 11 months' '56 years 01 month' '88 years 08 months'
 '88 years 06 months' '92 years' '58 years 11 months' '76 years 02
months'
 '76 years 08 months' '75 years 09 months' '77 years 01 month'
 '89 years 10 months' '94 years 05 months' '92 years 05 months'
 '86 years 01 month' '92 years 06 months' '92 years 01 month' '94
years'
 '91 years 04 months' '89 years 06 months' '90 years 03 months'
 '91 years 06 months' '86 years' '90 years 10 months' '77 years 11
months'
 '50 years 04 months' '51 years 11 months' '58 years 01 month'
 '76 years 04 months' '82 years' '81 years 07 months' '65 years 11
months'
 '63 years 01 month' '63 years 10 months' '93 years 05 months'
 '93 years 04 months' '57 years 10 months' '55 years 10 months'
 '64 years 06 months' '87 years 09 months' '64 years 02 months'
 '55 years 07 months' '55 years 08 months' '74 years 01 month'
 '65 years 08 months' '75 years' '51 years 04 months' '54 years 10
months'
 '55 years 04 months' '51 years 10 months' '92 years 10 months'
 '56 years 10 months' '90 years 09 months' '87 years 07 months'
 '53 years 03 months' '77 years 03 months' '50 years 10 months'
 '48 years 10 months' '52 years 10 months' '93 years 09 months'
 '87 years 03 months' '63 years 11 months' '75 years 01 month'
 '72 years 01 month' '56 years 07 months' '57 years 07 months'
 '53 years 10 months' '72 years 09 months' '75 years 10 months'
 '88 years 02 months' '52 years 03 months' '54 years 04 months'
 '76 years 01 month' '72 years 06 months' '89 years 05 months'
 '52 years 04 months' '91 years 09 months' '90 years 02 months'
 '91 years 05 months' '50 years 03 months' '51 years 03 months'
 '49 years 03 months' '91 years 01 month' '76 years 03 months'
 '60 years 09 months' '74 years' '54 years 03 months' '52 years 09
months'
 '55 years 03 months' '54 years 09 months' '87 years 06 months'
 '92 years 09 months' '76 years 07 months' '88 years 09 months'
 '50 years 02 months' '53 years 09 months' '52 years 02 months'
 '50 years 09 months' '56 years 09 months' '76 years 06 months'
 '89 years 08 months' '51 years 09 months' '89 years 03 months'
```

```
 '91 years 08 months' '89 years 04 months' '90 years 01 month'
 '51 years 02 months' '49 years 02 months' '74 years 02 months'
 '93 years 03 months' '54 years 02 months' '73 years 11 months'
 '51 years 08 months' '54 years 08 months' '87 years 05 months'
 '92 years 08 months' '52 years 08 months' '48 years 08 months'
 '50 years 08 months' '50 years 01 month' '73 years 02 months'
 '55 years 02 months' '88 years 03 months' '89 years 02 months'
 '91 years 10 months' '53 years 02 months' '51 years 01 month'
 '54 years 01 month' '73 years 10 months' '50 years 07 months'
 '53 years 01 month' '87 years 04 months' '48 years 07 months'
 '51 years 07 months' '96 years 08 months' '89 years 01 month' '91
years'
 '76 years 05 months' '53 years 07 months' '50 years' '93 years 01
month'
 '48 years 06 months' '50 years 06 months' '73 years 01 month'
 '89 years 11 months' '96 years 07 months' '73 years 09 months'
 '73 years 08 months' '87 years 02 months' '49 years 10 months'
 '48 years 05 months' '96 years 06 months' '89 years 09 months'
 '49 years 11 months' '73 years 07 months' '48 years 04 months'
 '49 years 09 months' '53 years 04 months' '96 years 05 months'
 '73 years 06 months' '48 years 03 months' '49 years 08 months' '48
years'
 '96 years 04 months' '49 years 07 months' '48 years 02 months'
 '96 years 03 months' '73 years 05 months' '48 years 01 month'
 '49 years 06 months' '96 years 02 months' '47 years 09 months'
 '96 years 01 month' '47 years 11 months' '96 years' '95 years 11
months'
 '47 years 10 months' '73 years 03 months' '47 years 08 months'
 '47 years 04 months' '47 years 07 months' '47 years 06 months'
 '47 years 05 months' '47 years 02 months' '47 years 03 months'
 '47 years 01 month' '47 years' '46 years 11 months' '46 years 10
months'
 '46 years 09 months' '46 years 08 months' '66 years 0 months'
 '95 years 0 months' '46 years 07 months' '46 years 06 months'
 '46 years 03 months' '93 years 0 months' '46 years 05 months'
 '46 years 04 months' '46 years 01 month' '46 years 02 months' '46
years'
 '45 years 09 months' '45 years 11 months' '45 years 10 months'
 '45 years 07 months' '45 years 08 months' '97 years 01 month'
 '45 years 06 months' '97 years 09 months' '45 years 05 months'
 '45 years 04 months' '97 years' '97 years 07 months' '45 years 03
months'
 '45 years 02 months' '96 years 09 months' '45 years 01 month'
 '44 years 10 months' '97 years 05 months' '97 years 04 months'
 '97 years 03 months' '97 years 02 months' '45 years' '96 years 11
months'
 '96 years 10 months' '44 years 09 months' '44 years 08 months'
 '44 years 07 months' '44 years 11 months' '44 years 06 months'
 '44 years 05 months' '44 years 04 months' '44 years 03 months'
```

```
 '44 years 02 months' '44 years 01 month' '44 years' '43 years 11
months'
 '43 years 10 months' '43 years 09 months' '43 years 08 months'
 '43 years 07 months' '43 years 06 months' '43 years 05 months'
 '43 years 04 months' '43 years 03 months' '43 years 01 month'
 '43 years 02 months' '43 years' '42 years 11 months' '42 years 08
months'
 '42 years 07 months' '42 years 10 months' '42 years 09 months'
 '42 years 06 months' '42 years 05 months' '42 years 04 months'
 '42 years 03 months' '42 years 02 months' '42 years 01 month']
----------------------------------------------------------------------
----------


□ Column: resale_price
[232000. 250000. 262000. ... 403388. 751688. 300088.]
----------------------------------------------------------------------
----------


□ Column: Year
[2017 2018 2019 2020 2021 2022 2023]
----------------------------------------------------------------------
----------


□ Column: Month
[ 1  2  3  4  5  6  7  8  9 10 11 12]
----------------------------------------------------------------------
----------


□ Column: latitude
[1.35968883 1.37231328 1.37634218 1.3601358  1.36619691 1.38403248
 1.37742633 1.37048119 1.37196446 1.33497992 1.32744917 1.33145574
 1.32085209 1.32808858 1.3285634  1.33168386 1.32123666 1.32510368
 1.32529103 1.32370195 1.32258562 1.32696994 1.3378131  1.32727572
 1.32388941 1.3262815  1.33324476 1.32810291 1.32191534 1.35371666
 1.36386365 1.34755435 1.34759166 1.35900491 1.35865898 1.35506687
 1.35593097 1.35176163 1.36348296 1.35718821 1.35044172 1.35606275
 1.34586254 1.34904853 1.34670636 1.34822368 1.35148189 1.34979708
 1.35676943 1.34235173 1.34745649 1.36033138 1.34326273 1.28757615
 1.27325111 1.28852535 1.27773153 1.28780753 1.27394762 1.27446866
 1.28730052 1.28811452 1.27492293 1.28357871 1.29130443 1.28593893
 1.27439946 1.28488194 1.28825972 1.290718   1.27146257 1.28400968
 1.28019628 1.28797301 1.29207525 1.28532736 1.28793296 1.28670064
 1.2885994  1.37792695 1.3785556  1.38137153 1.38676224 1.38288227
 1.38191601 1.38333405 1.37853338 1.38691071 1.38021544 1.37656097
 1.33820237 1.31198098 1.28683793 1.30939779 1.29853916 1.28396405
 1.27523683 1.29711317 1.29681526 1.27937182 1.37788965 1.38237841
```

1.3808319	1.40021628	1.39381404	1.3809268	1.37820351	1.39693363
1.39587134	1.38582271	1.39246582	1.39447145	1.37803539	1.30234148
1.29220563	1.30477461	1.30891244	1.31513509	1.32212393	1.31399161
1.31158041	1.30924252	1.32300685	1.32880277	1.33038182	1.33255923
1.32065765	1.33068802	1.31129091	1.30707416	1.31585562	1.31556694
1.32399306	1.31821153	1.31844043	1.30894453	1.38031908	1.36413097
1.37647221	1.36778751	1.37518093	1.3520264	1.36497585	1.37123298
1.36596036	1.35087034	1.37457017	1.37717417	1.37326115	1.37971113
1.36292693	1.37115128	1.37844829	1.38198271	1.36382073	1.32129337
1.32349635	1.34263187	1.33673824	1.35062884	1.33762415	1.34404145
1.3358347	1.33701267	1.34459584	1.3338894	1.34624968	1.3331845
1.34298941	1.35328786	1.34079305	1.34834917	1.34935036	1.33727166
1.34883617	1.33441343	1.35470966	1.3458412	1.34504745	1.34987065
1.34988154	1.32676126	1.34852763	1.34231548	1.33641299	1.34190096
1.34426726	1.33620559	1.33966177	1.35069822	1.35088145	1.30315438
1.32540758	1.32379607	1.32211972	1.32036885	1.31344486	1.32370763
1.31525573	1.32250536	1.31264014	1.30938417	1.31369813	1.29579
1.31942539	1.31861241	1.32649027	1.31569896	1.32699075	1.30338982
1.30402509	1.30509642	1.38854656	1.37790741	1.36846368	1.370917
1.36933056	1.36858695	1.37276906	1.37804865	1.37505696	1.36705022
1.36842577	1.38197757	1.37538514	1.39794617	1.39472186	1.39330438
1.3974734	1.42068279	1.38604164	1.40717643	1.3954005	1.40672877
1.36895144	1.29707664	1.29348201	1.30805159	1.33598648	1.31181076
1.3079757	1.30934498	1.29338295	1.31499297	1.30869474	1.29234264
1.30406013	1.44574531	1.43494012	1.45089251	1.45427018	1.45123288
1.44482698	1.44585217	1.38984421	1.38536459	1.39033148	1.38246038
1.39250411	1.38625979	1.38989922	1.38215139	1.39030228	1.38809685
1.38700703	1.39907216	1.39814758	1.3881321	1.38785625	1.38515662
1.38283357	1.38407498	1.39131628	1.39052785	1.39368191	1.3733188
1.3510575	1.3723953	1.37587607	1.37616136	1.3509778	1.37463846
1.34781304	1.35734525	1.34495328	1.35883955	1.34841977	1.34740035
1.35292163	1.34711276	1.35289506	1.3503628	1.35756792	1.35617254
1.36026525	1.35625123	1.34978359	1.35989315	1.34134725	1.3431589
1.35881219	1.35440348	1.34585136	1.35892339	1.35330429	1.35336373
1.37059545	1.33865955	1.33233883	1.33965127	1.33356607	1.3331571
1.34179028	1.34074251	1.33933821	1.33447489	1.33461833	1.33669904
1.33257461	1.44207298	1.43598269	1.43343677	1.43044908	1.44620709
1.43947829	1.444097	1.44061702	1.42958661	1.43061339	1.44168228
1.43672576	1.43210416	1.43953354	1.44111273	1.43864719	1.43856288
1.44011341	1.43318009	1.45614942	1.43272348	1.43872152	1.44700426
1.44075549	1.43178572	1.43222135	1.42347167	1.43080206	1.42053932
1.43498006	1.42621097	1.43113229	1.4306869	1.42033792	1.41922604
1.43839563	1.4290305	1.42767465	1.41177094	1.42688859	1.44227548
1.3644044	1.36562478	1.33036165	1.33553204	1.33167118	1.33016612
1.36485172	1.27547253	1.28652962	1.27559529	1.28501999	1.28392509
1.37002391	1.32280086	1.30774015	1.30613509	1.28180062	1.28623521
1.28764538	1.28211631	1.38194643	1.39150729	1.37584149	1.38129091
1.31129566	1.32259415	1.32095901	1.30718804	1.30787056	1.35876258
1.35981497	1.36271446	1.35954177	1.34615521	1.33927933	1.39551343

```
 1.30498798 1.3205017  1.2997195  1.31337582 1.30889577 1.37270725
 1.30921938 1.29767709 1.45218571 1.44778957 1.39172552 1.3540394
 1.3731286  1.37333247 1.34972245 1.3544355  1.34773191 1.34371956
 1.3412098  1.33050536 1.44280577 1.44202663 1.44646333 1.4331914
 1.44100431 1.43389741 1.43151585 1.440321   1.37065009 1.34455983
 1.36265344 1.35964072 1.36139508 1.34461889 1.36470351 1.2870356
 1.27727697 1.37936398 1.28347724 1.37925789 1.31724993 1.3075447
 1.39672603 1.39179995 1.40047053 1.32014419 1.32370062 1.31649443
 1.34138493 1.31370804 1.32767174 1.33682897 1.32827854 1.32612553
 1.31266417 1.36226599 1.29238273 1.30304255 1.44686608 1.36103455
 1.34469945 1.33107979 1.33429242 1.33290762 1.44411841 1.43911201
 1.43882635 1.43163453 1.43539225 1.36946054 1.28613694 1.3088207
 1.38409831 1.32208937 1.31659508 1.34649073 1.34083828 1.33538116
 1.32928599 1.32380443 1.31649301 1.32786874 1.30205082 1.38943818
 1.28486467 1.28489922 1.38233493 1.30644303 1.32271179 1.30634115
 1.34882313 1.30720612 1.31358617 1.34678773 1.31840833 1.29455313
 1.32717562 1.36725607 1.35264348 1.35644705 1.33669265 1.4442722
 1.2808342  1.30868191 1.43359371 1.37216803 1.30376528 1.31429764
 1.30856889 1.37258743 1.39184675 1.28549378 1.3077851  1.39527159
 1.38170892 1.44954486 1.40541354 1.43208986 1.38319029 1.28004979
 1.45063213 1.37811995 1.41090728 1.38096576 1.39901716 1.33193786
 1.39635988 1.35673316 1.41812575 1.37577532 1.34342966 1.33534201
 1.38095849 1.29386475 1.39604095 1.28529692 1.36380224 1.35021013
 1.28408881 1.40509727 1.38224243 1.37524979 1.34051877 1.37480118
 1.32052596 1.40076988 1.44569062 1.45209602 1.36866971 1.44520176
 1.44776611 1.37090383 1.31797272 1.34751993 1.35387905 1.30134715
 1.29482149 1.36162838 1.4243888 ]
----------------------------------------------------------------------
----------


 Column: longitude
[103.85457532 103.83760111 103.84015898 103.85502558 103.84775665
 103.84035691 103.84869935 103.8448058  103.85176353 103.94957995
 103.92718443 103.93545864 103.93372109 103.90987764 103.92322245
 103.92236084 103.92867661 103.92487431 103.93035302 103.91981744
 103.93684943 103.93585008 103.93567972 103.9248036  103.9127578
 103.93418041 103.90994321 103.9329381  103.93318086 103.83611588
 103.83384092 103.84989608 103.85503443 103.84700658 103.84204982
 103.84694606 103.83221724 103.83904666 103.74939336 103.75112394
 103.74321201 103.75276696 103.74790576 103.75582856 103.74692066
 103.74803453 103.73838602 103.75814831 103.74877426 103.76015935
 103.75471044 103.74123654 103.75653145 103.8294128  103.81171977
 103.8284586  103.81976943 103.83278058 103.82086693 103.82147409
 103.80798827 103.82904033 103.81340807 103.82381921 103.82744779
 103.80471581 103.80728142 103.83104242 103.83665778 103.82162521
 103.82568355 103.82742142 103.81507622 103.80883927 103.82858408
 103.82838553 103.81729893 103.81081976 103.8333947  103.76310293
 103.7687488  103.77324501 103.77252338 103.76684713 103.76249985
```

103.76940808	103.77178907	103.76331018	103.76630217	103.76595244
103.77452263	103.85474815	103.83867408	103.85405357	103.85185601
103.84314195	103.84260534	103.85196485	103.85361948	103.84003065
103.75423991	103.73920354	103.74293968	103.75039302	103.74939869
103.7485596	103.75645088	103.74492174	103.74351919	103.74667428
103.74790689	103.74405435	103.74502069	103.76449566	103.76889162
103.76525375	103.76705514	103.76605447	103.7702659	103.767089
103.76421128	103.76710634	103.76186827	103.88859909	103.88578898
103.88589253	103.90246345	103.90195304	103.89770343	103.88358297
103.88527144	103.87639101	103.88669272	103.88475233	103.89985218
103.8835301	103.87891144	103.89300132	103.89382211	103.8939299
103.88602266	103.87907857	103.89680988	103.886918	103.88864648
103.88367742	103.87987781	103.88947444	103.89463171	103.88995097
103.88768572	103.89447368	103.8828569	103.88121768	103.88957277
103.74764891	103.73788423	103.74213381	103.74381241	103.72887803
103.73835489	103.73550008	103.74828162	103.69351778	103.70819069
103.72330612	103.70938701	103.72435432	103.69186415	103.72333177
103.69047002	103.71138162	103.71794792	103.72435553	103.69500381
103.7222212	103.70264192	103.69960545	103.69814781	103.70302709
103.70268748	103.7212885	103.69942883	103.69528112	103.69970174
103.70026365	103.7058337	103.70417739	103.70013784	103.72090506
103.70406225	103.88391202	103.8585655	103.85567062	103.86082009
103.8683408	103.85008637	103.87168305	103.86014997	103.85343946
103.87249187	103.85198385	103.85325909	103.85404149	103.86108746
103.85609494	103.86057207	103.85978517	103.8474037	103.90855354
103.91350146	103.91773375	103.9878045	103.93598443	103.95600907
103.95584948	103.96276294	103.9592047	103.94651045	103.93757453
103.94528942	103.95818186	103.95064858	103.93763774	103.93935611
103.90594514	103.90987381	103.91213062	103.90482182	103.90822939
103.90290057	103.90443706	103.91523939	103.90551536	103.77334068
103.80092479	103.80491674	103.80084281	103.81430088	103.78596725
103.77773304	103.79555935	103.80933072	103.78278713	103.78404877
103.81517282	103.7826372	103.82115268	103.80291059	103.82367817
103.81581264	103.81588941	103.81737852	103.82289624	103.87502229
103.90081749	103.88676812	103.90223868	103.90448367	103.90610811
103.90218919	103.9000751	103.87444546	103.89144127	103.89196477
103.89645646	103.87219005	103.89725765	103.89468104	103.89505506
103.89024316	103.89252206	103.88477655	103.89047035	103.90077005
103.86875853	103.87683573	103.86916624	103.87119763	103.87139549
103.87359548	103.87368712	103.95040928	103.94461145	103.93940837
103.95435159	103.9346491	103.94529492	103.95385834	103.93945188
103.95358568	103.93455454	103.95302078	103.93747992	103.95837235
103.96157664	103.93561471	103.95575657	103.95135507	103.95418811
103.9352127	103.93270229	103.95506979	103.94049838	103.95743496
103.95547787	103.92738083	103.85580648	103.85127709	103.84548515
103.85698175	103.8683804	103.85228084	103.85780188	103.85122474
103.86628654	103.88016502	103.84640708	103.84826792	103.77635399
103.78193275	103.79805545	103.77277592	103.77368035	103.79025803
103.80040459	103.79664662	103.77485013	103.79886239	103.78598469

```
 103.79714761 103.79649081 103.80379728 103.79801095 103.79651807
 103.79363454 103.80214356 103.79179044 103.8012736  103.80117275
 103.78128743 103.80188903 103.80426411 103.79448461 103.82755512
 103.83062327 103.83301117 103.84069013 103.83896472 103.82770411
 103.82836714 103.83657443 103.84493618 103.83739078 103.83930917
 103.84430354 103.83773146 103.83336827 103.83406704 103.83970152
 103.85181091 103.84701147 103.93508102 103.94919449 103.90653791
 103.94010032 103.74581456 103.83996263 103.83316309 103.83755986
 103.81857772 103.82826377 103.76450561 103.81102349 103.85082643
 103.85613024 103.84290816 103.84230334 103.8402701  103.84284562
 103.75188818 103.74274643 103.73775743 103.75029665 103.75886443
 103.88429611 103.8841294  103.90382907 103.88319216 103.89146007
 103.88895597 103.8939805  103.88516737 103.70116937 103.72344161
 103.87328498 103.8611503  103.86334127 103.86082989 103.84641692
 103.8630686  103.93931466 103.79673239 103.7996368  103.82136989
 103.82441884 103.87700045 103.87085153 103.86734565 103.86990716
 103.92783602 103.9522862  103.94412402 103.95842429 103.95628885
 103.85577638 103.77929884 103.78990578 103.79873761 103.78159492
 103.80722335 103.79573093 103.77888159 103.84057573 103.83941035
 103.85573742 103.74412031 103.74882909 103.74651666 103.74978901
 103.74424678 103.80995508 103.83109336 103.87180406 103.81371809
 103.7629971  103.80597889 103.85289264 103.75181799 103.74562118
 103.74634347 103.76325353 103.76967109 103.89694167 103.95955827
 103.88182958 103.72240755 103.72111765 103.85609736 103.85567266
 103.8769316  103.96207864 103.80019631 103.78535844 103.82060863
 103.95216442 103.95821011 103.85896644 103.87874621 103.86616885
 103.80444477 103.78225659 103.79929745 103.83986735 103.83550098
 103.83465905 103.83302787 103.85206529 103.745499   103.76893341
 103.87954564 103.72905387 103.72509814 103.72290661 103.72238573
 103.86683989 103.85694147 103.8442096  103.80026461 103.87792381
 103.83050049 103.83158418 103.76767766 103.85134909 103.7698881
 103.88445818 103.71923879 103.86073388 103.85169992 103.87040846
 103.84861811 103.779694   103.84665041 103.8748135  103.94294211
 103.9545421  103.8447961  103.78363807 103.82331807 103.88712374
 103.77588406 103.86656908 103.84994891 103.87780057 103.85921493
 103.95812045 103.91303904 103.83259998 103.85701098 103.88034303
 103.8872848  103.81872759 103.8968239  103.84619833 103.75187829
 103.84248188 103.82655406 103.90229625 103.90118609 103.9008203
 103.89098972 103.72156151 103.88931707 103.94016525 103.84541649
 103.90256128 103.93235433 103.92033512 103.84155449 103.81009297
 103.88073542 103.83323533 103.8937897  103.92727457 103.81932417
 103.89536228 103.73836122 103.7434488  103.70454972 103.74953793
 103.88615931 103.89548465 103.83029616 103.83062674 103.85765869
 103.80482856 103.83222301 103.85442176 103.88877829 103.76836734
 103.74067771 103.90718361 103.81300086 103.93843439 103.8520831 ]
----------------------------------------------------------------------
----------
```

```
□ Column: distance_to_mrt_m
[1178.77415552 1381.79277587 1321.05757848 1160.64064608  380.95983888
 1945.35995134  926.1236253   554.96473176  397.86291358 2051.32326658
  398.90450815 1053.31110797  646.83268247 2177.7877227   799.87475911
 1100.5393264   352.94552681  476.6701779   170.74299144 1035.93549532
  885.46155772  803.16712192 1654.06681717  574.22695343 1819.59911644
  602.4605082  2345.57600615  591.69120968  530.79482136 1361.32240974
 1846.90431414  416.32463745  859.48616251  913.9683057  1092.5489532
  486.16931018 1845.80122516 1002.06319248 1535.14247057  863.26864499
  683.8879539   812.12420173  441.43765917  729.11006999  415.32225915
  207.35139594 1232.30034518  984.9595106   794.91484589 1450.13836165
  647.09215609 1487.73163447 1067.10475113  477.98736374 1843.3208568
  526.06186702 1062.06566717  768.84900105 1304.38160674 1220.23457367
  817.95346323  508.48040831 1619.82322649  357.04736117  809.22195458
  962.91662945 1923.49951665  460.33610814 1173.5069718   623.80613337
 1394.04500534   46.91291828 1014.38168834  775.07628938  910.15660657
  212.88372892  194.44005957  664.18714551  874.71980484  196.60120312
  818.01034652 1358.53430576 1545.1081875   783.01980484  407.59488558
 1044.93687715 1156.2677581   964.7821547   581.3112021   545.91748705
  335.435345   1109.7777549  1336.58783554 1385.87521467 2454.67739355
  876.59868108  346.87908577 2296.69363565 2248.49287435  691.73205948
  798.82440834  656.85402977  492.51319177 1805.83333153 1116.72165422
  637.93519762  551.20375803 1320.39478993 1207.04291151  258.5049358
  908.43410866 1048.47843815  772.28771796 1414.35693761 2561.19137917
 1141.84418524  708.55081933   72.93993441  946.82005109  224.39591944
  411.04919996  674.95521082  958.58392847 1291.02591679 1586.9828934
 1793.70948263 1093.41887389 1720.79157636  907.93594585 1046.14386306
  369.02254427  630.98668732  965.33068225  365.98979972  764.13388946
  841.79100582 1839.43780107  793.03100769  583.11916012  408.24067425
  867.7703162   734.83944843  829.54852007  654.63659215  749.84697953
 1184.21842883 1482.84547743  747.5736239   297.62265947  982.61930611
 1086.80280931  186.98436672 1359.85474076 1748.36122875  901.63778878
 1448.99800714 1156.49570843 1054.09080033  450.03051014 1210.75651248
  644.06466614 1409.61068213  761.6752777  1545.88325158  789.5408297
 1208.52227275  992.44463571 1321.1746693  1832.42543662 1060.74727612
 1918.98860779 1058.48916838  603.33550165  915.93155187 1863.9850699
 1120.80595907 1975.25531789 1266.15799518 1325.40796617 1451.87454785
 1466.10035185 1944.63823641 1507.92493969 1450.00640897  865.12515835
  931.43306023  768.32332547  386.14785596  842.80923282  714.61699343
 1525.56495769 1480.00952911  676.666626    739.60841639  257.28293355
  819.37605939 1403.21804957 1264.99932134  511.30978771  893.00860402
  175.09975047 1527.51871245 1094.25782058 2133.59887117   42.5506886
  561.45753349  741.30672662  473.19670536  632.89609458 2367.72662351
 2755.65855465 2481.16100705 4571.60144887 1616.28923258  869.36610256
  725.68490826 1512.31809034 1169.56442227  356.83891446 1456.14007956
  536.42571179 1156.93554483  527.52312213 1663.00736489 1178.55337093
  914.38932765 1452.6561203  1732.3964796   910.2532129  1847.04119517
 1060.2377421   348.30589385 1828.91991088  426.17087119 1690.3835835
  662.59701342  190.70806268 1616.86565984 3405.56821272 2317.65451276
```

1582.55296475	2030.58913742	359.19769133	1935.0561147	2192.95390938
355.33075372	2273.8096362	392.49772163	1830.00595424	454.7860661
737.54704596	514.66121821	555.29476652	482.54994976	2229.90403884
929.79461717	923.68774651	1275.1760459	1062.44556792	1360.59359641
817.09192732	1168.2611727	2290.66969994	538.75076431	591.65153449
863.69361018	2645.82479701	440.11124352	393.45471894	690.38803673
1085.1203268	855.62055797	1137.77310364	513.23282167	689.92550438
2182.54804127	464.19160669	2207.7610672	2456.75178763	2441.95819138
223.079775	2158.8253042	815.50519196	484.91921077	1098.68969548
1205.90723524	1278.76402849	619.27125036	963.61354378	912.68518442
933.30361515	1220.08003885	1006.30736311	928.01539881	1671.62547133
1857.65196701	1124.46154358	1400.4204536	1459.26347977	1478.06780596
1283.90403345	1399.48368233	1353.25734318	838.31519154	1362.02840493
1144.51669048	2499.24209952	1181.00955173	477.66766116	786.90660376
1114.98492995	1689.98466533	1104.29957175	1495.31960213	871.68956059
1726.03983177	1769.10357747	447.08965068	141.78530933	1313.69444014
508.26922793	1316.81020579	1646.00515099	1819.09743348	568.17818334
1787.72194141	1239.21259359	1477.73664905	1499.14910192	630.92989942
1187.63136579	1192.36337381	1964.17595441	1400.19652229	1152.64385699
843.02496556	1799.3251962	666.21898653	2224.95911128	1672.56526401
653.49222525	2028.7428964	1979.37330422	1003.3446889	902.40867316
838.89302808	282.95232701	1164.05424175	736.68818704	910.01396823
781.5222498	201.54176851	1483.34935157	1161.90943784	1084.73442367
1014.41778745	346.55619138	1970.95924331	315.08213754	1498.84138867
574.56479177	480.26052472	936.81084541	1982.0143121	2134.27906862
1380.3047835	1730.49020729	624.74024316	740.74170956	888.14331851
542.21837253	140.89133093	988.57002532	3173.20946393	1749.34028272
1605.24925297	663.18883119	1144.23866018	1372.93791096	697.70503227
888.83982228	745.53757078	1260.40064683	764.43034212	840.23229814
731.0332905	557.03589706	1697.5670717	952.42781338	1394.32030445
1340.07330105	958.39213898	1553.04340162	1181.84106123	674.46444221
2459.02715928	1348.95606393	271.86747499	1755.17829059	1771.60533583
970.89309516	1156.57727109	1944.95088991	820.56108521	373.71011203
512.65155505	2003.43750951	615.47878576	2024.49154926	2307.89035517
1966.10641125	804.42901429	594.69843586	1794.15715087	1795.14681023
1006.41514791	1099.33406272	766.63578532	1786.97010007	628.02263134
1680.17316243	1053.24284619	982.16066127	1337.75053093	1146.86523286
1103.44102908	1554.23166596	1111.48772509	1340.51823645	553.46738109
1762.21002549	738.73355719	871.87937256	2500.94613241	700.59911252
201.46749867	2515.82368704	1628.02461817	1531.15251633	762.18576397
1722.91084104	606.76279722	1063.18562305	469.19693081	2050.08667345
298.26658851	1858.08745486	835.41979174	1084.72412943	916.91269696
628.92952234	1825.62226426	718.50056638	2514.75497419	256.13286535
1178.9856487	1714.28462274	1267.62782159	1739.47999911	1559.35394733
1816.58078448	584.30229493	1458.04749902	570.51794696	652.39529572
1663.2255009	711.24982269	1569.26974548	149.33513318	867.13767627
273.99669172	1036.30619896	684.41630359	1037.89464951	1681.26878578
786.4219998	581.1411799	617.91821697	1071.57410849	1912.69772477
401.13632472	519.78071441	823.0384733	1826.18729107	978.72115655

```
 1145.4065161   507.26872376 1278.47321371 1242.31231293  395.4165921
 1386.54805356 2773.52665309  612.0951462  2026.63324499  254.35159541
 1107.33563329  505.04306476  968.56022202  538.93538727 1026.39766475
 1210.98568657 1918.68259707 2157.83878633  521.98014582 1257.66757617
  938.80555472 1920.77031241  644.74798712 1400.75614893 1686.37066577
 1304.39226922  149.89950513  576.51268747 1257.00547     845.14146678
  525.06862269  748.77931232 1298.28731262  637.549197   1323.14859057
  953.19987443 1377.94375767  837.11281757  695.95009959 1695.42749195
 1193.71467805 1778.64225846 1554.14616034 1587.78469115  427.92872006
 1668.34424777  708.60313059  836.3488236  2018.60923476  672.99978902
  738.77455105  748.12630168 1084.45661733  460.48018113 1259.4123071
  649.93133479  875.03634838 1206.22547782 1228.11252423  898.09576472
 1742.4984696  1368.20077122  572.4740648   469.84428899 1223.64225132
 1069.87748462 2425.36107424  706.76603307 1215.39404149
1962.00265383]
----------------------------------------------------------------------
----------
```

Visualising distance_to_mrt_m

This is something new we learnt:
Folium Map
pip install folium

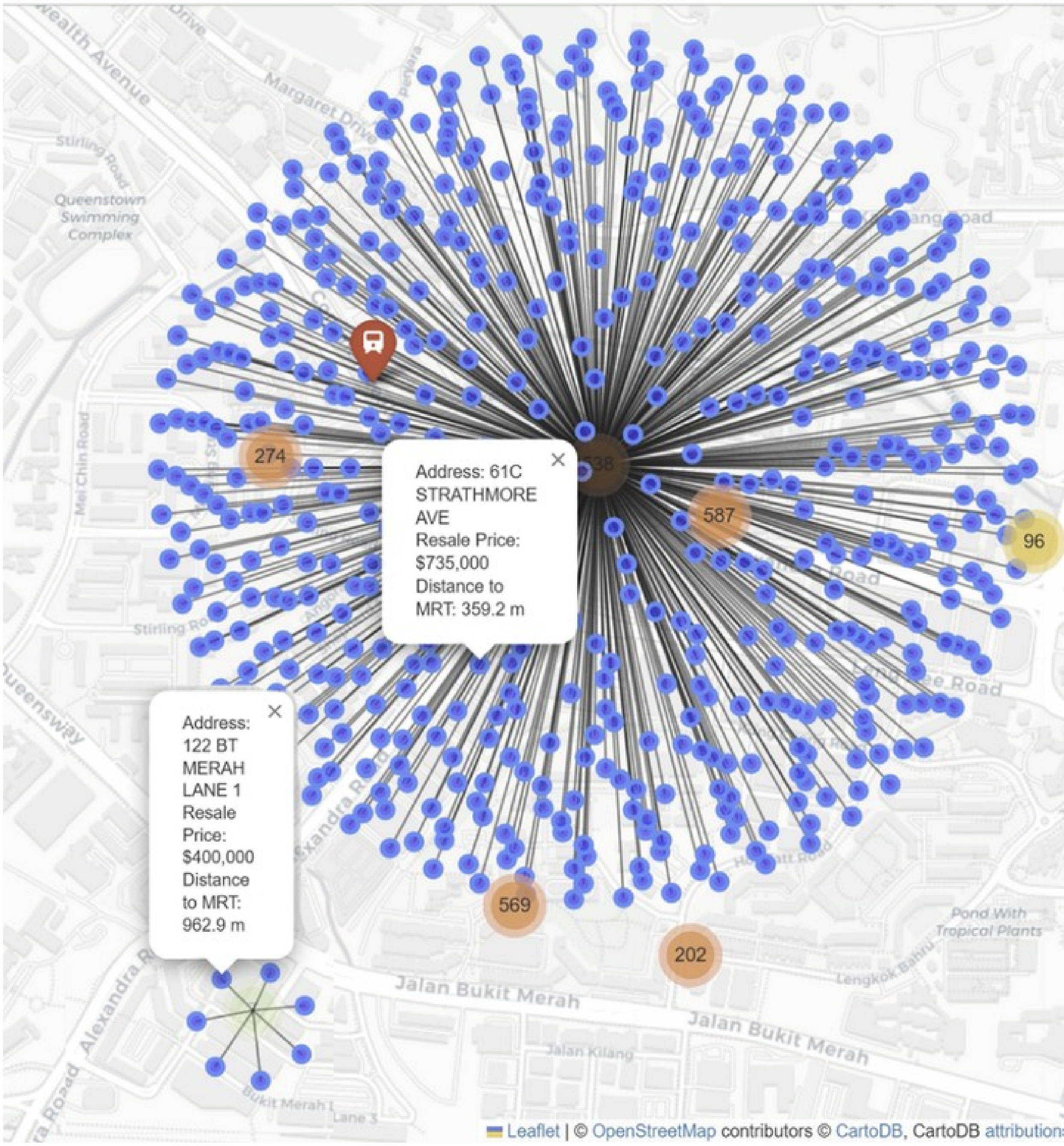
```
import pandas as pd
from geopy.distance import geodesic
mrt_df = pd.read_csv("C:\\Users\\Crystaline\\Downloads\\
mrt_stations.csv")
mrt_coords = list(zip(mrt_df['latitude'], mrt_df['longitude']))

def calculate_min_distance(hdb_lat, hdb_lon, mrt_coords):
    return min(geodesic((hdb_lat, hdb_lon), mrt_coord).meters for
mrt_coord in mrt_coords)

df['distance_to_mrt_m'] = df.apply(lambda row:
calculate_min_distance(row['latitude'], row['longitude'], mrt_coords),
axis=1)
df.to_csv("hdb_with_mrt_distance.csv", index=False)
print("□ Distances to MRT stations computed and saved.")
```

```
KeyboardInterrupt
```

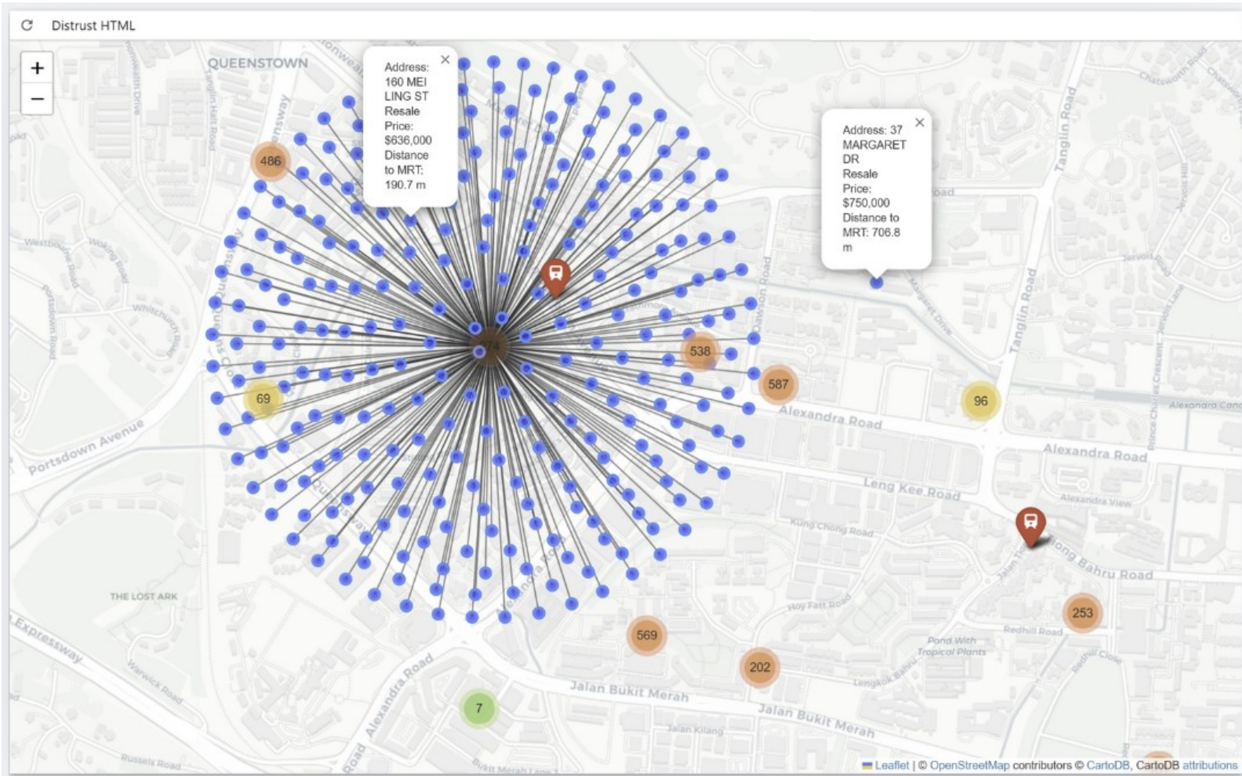
Now we are going to explore the claim: "Distance to nearest MRT and resale price has a correlation.", using evidence from our data visualise to determine whether the claim is true or false.

```python
# map
import pandas as pd
from geopy.distance import geodesic
import folium
from folium.plugins import MarkerCluster

mrt_df = pd.read_csv("C:\\Users\\Crystaline\\Downloads\\
mrt_stations.csv")
mrt_coords = list(zip(mrt_df['latitude'], mrt_df['longitude']))
def nearest_mrt_distance(hdb_point):
    hdb_coords = (hdb_point['latitude'], hdb_point['longitude'])
    return min(geodesic(hdb_coords, mrt).meters for mrt in mrt_coords)

df['distance_to_mrt_m'] = df.apply(nearest_mrt_distance, axis=1)
sg_center = [1.3521, 103.8198]
m = folium.Map(location=sg_center, zoom_start=12, tiles='CartoDB
positron')
for _, row in mrt_df.iterrows():
    folium.Marker(
        location=[row['latitude'], row['longitude']],
        popup=row['Station Name'],
        icon=folium.Icon(color='red', icon='train', prefix='fa')
    ).add_to(m)
marker_cluster = MarkerCluster().add_to(m)

for _, row in df.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=5,
        fill=True,
        fill_color='blue',
        fill_opacity=0.7,
        popup=(
            f"Address: {row['block']} {row['street_name']}<br>"
            f"Resale Price: ${row['resale_price']:,.0f}<br>"
            f"Distance to MRT: {row['distance_to_mrt_m']:.1f} m"
        )
    ).add_to(marker_cluster)
m.save('hdb_resale_map.html')

import branca.colormap as cm
colormap = cm.linear.YlOrRd_09.scale(df['resale_price'].min(),
df['resale_price'].max())
fill_color = colormap(row['resale_price'])  # or
row['distance_to_mrt_m']
```

The following images are taken from 'hdb_resale_map.html'



The nearer to MRT, the higher the resale price of that hdb. But this is not always the case. There are exceptions.

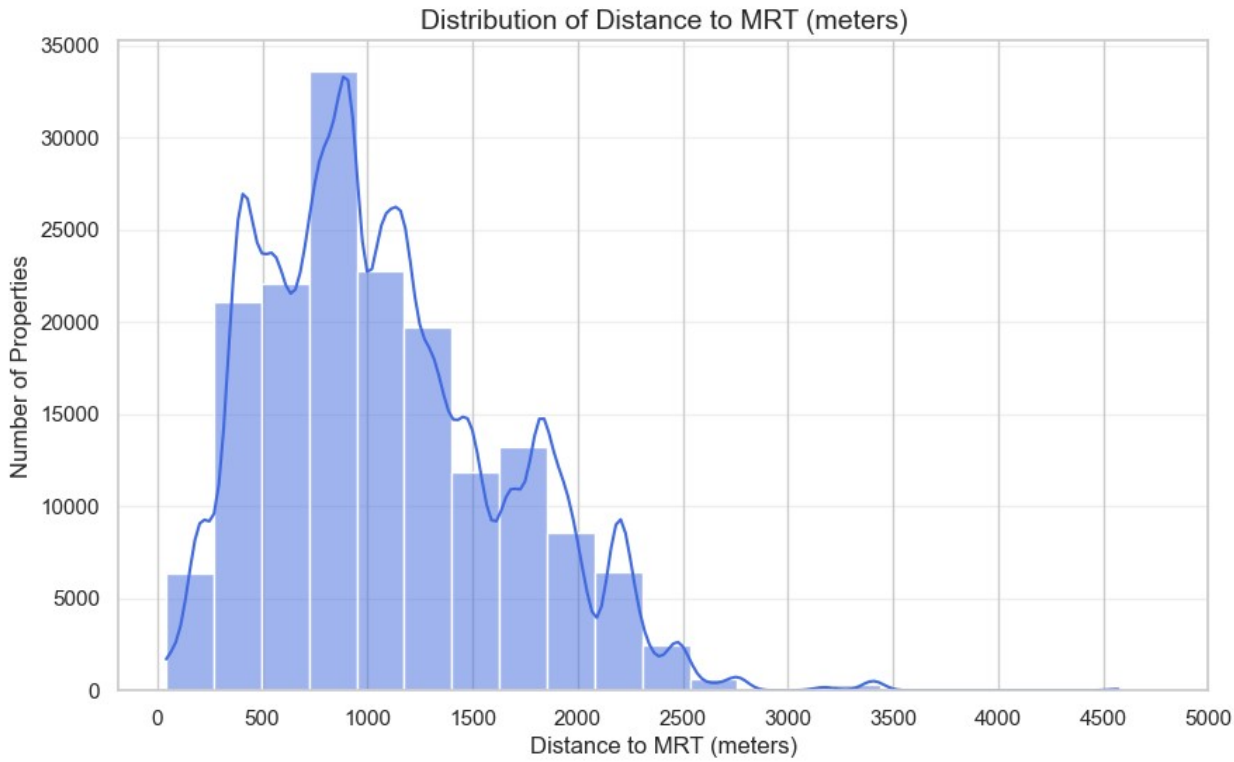


The second example shows: The nearer to MRT, sometimes the lower the resale price.

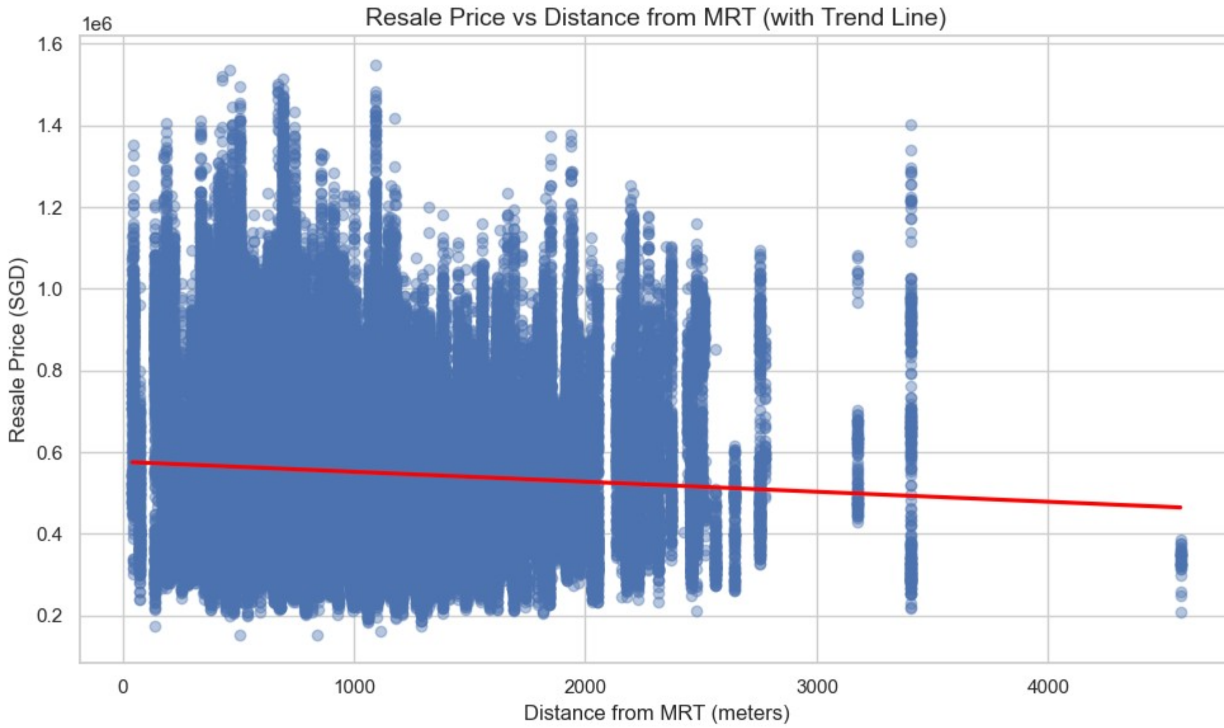
Overall Distribution of distance_to_mrt_m:

Bi-Variate

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='distance_to_mrt_m', bins=20, kde=True,
color='royalblue')

plt.title('Distribution of Distance to MRT (meters)', fontsize=14)
plt.xlabel('Distance to MRT (meters)', fontsize=12)
plt.ylabel('Number of Properties', fontsize=12)
max_dist = df['distance_to_mrt_m'].max()
plt.xticks(range(0, int(max_dist)+500, 500))
plt.grid(axis='y', alpha=0.3)
plt.show()
```

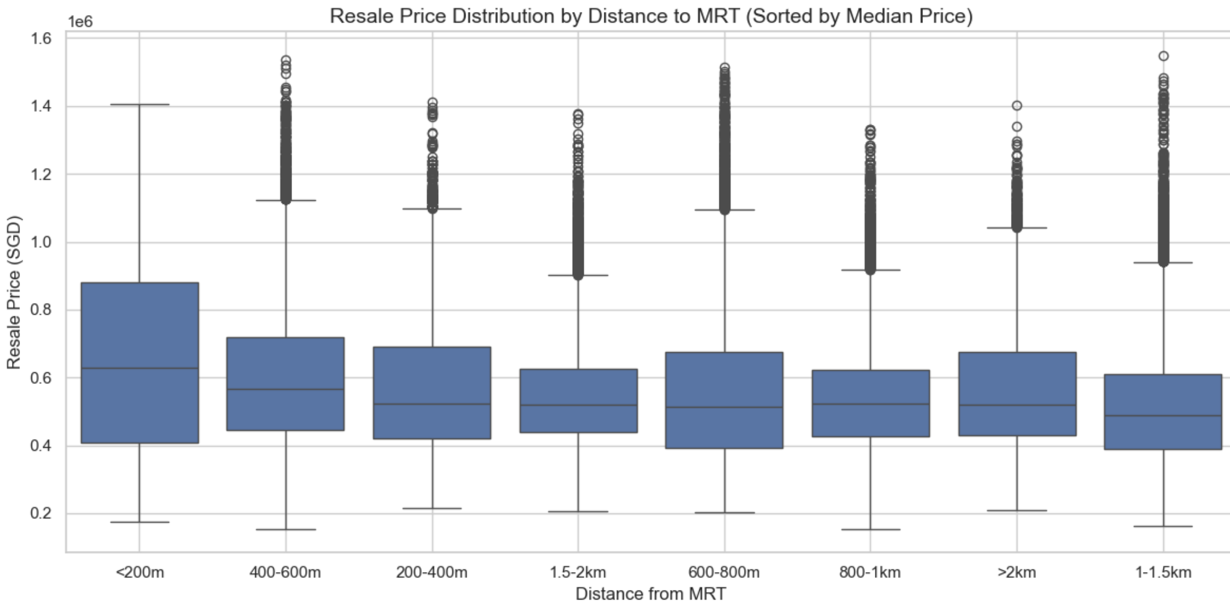



```python
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.regplot(x='distance_to_mrt_m', y='resale_price_adj', data=df,
scatter_kws={'alpha':0.4}, line_kws={"color": "red"})
plt.title('Resale Price vs Distance from MRT (with Trend Line)',
fontsize=14)
plt.xlabel('Distance from MRT (meters)')
plt.ylabel('Resale Price (SGD)')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df['distance_bin'] = pd.cut(
    df['distance_to_mrt_m'],
    bins=[0, 200, 400, 600, 800, 1000, 1500, 2000, float('inf')],
    labels=['<200m', '200-400m', '400-600m', '600-800m', '800-1km',
'1-1.5km', '1.5-2km', '>2km']
)

median_prices = df.groupby('distance_bin')
['resale_price'].median().sort_values(ascending=False)

sorted_bins = median_prices.index.tolist()

plt.figure(figsize=(12, 6))
sns.boxplot(x='distance_bin', y='resale_price_adj', data=df,
order=sorted_bins)
plt.title('Resale Price Distribution by Distance to MRT (Sorted by
Median Price)', fontsize=14)
plt.xlabel('Distance from MRT')
plt.ylabel('Resale Price (SGD)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
C:\Users\Crystaline\AppData\Local\Temp\
ipykernel_11268\741279116.py:13: FutureWarning:

The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain
current behavior or observed=True to adopt the future default and
silence this warning.
```



Insights

1. **Property Distribution by Distance**: The top histogram shows most properties in the dataset are within 2000 meters of an MRT station, with the highest concentration between 500-1500 meters. This suggests Singapore's urban planning effectively places most housing within reasonable distance of public transit.

2. **Price-Distance Correlation**: The middle scatter plot with trend line demonstrates a clear negative correlation - as distance from MRT increases, property prices generally decrease. However, the shallow slope of the trend line indicates this effect is moderate rather than dramatic.

3. **Price Variability**: Despite the downward trend, there remains significant price variability at all distances, shown by the vertical spread of points at each distance interval. This suggests other factors beyond MRT proximity significantly influence pricing.

4. **Premium for Extreme Proximity**: Properties closest to MRT stations (<200m) show the highest median prices in the boxplot, confirming a premium for extreme convenience.

5. **Non-Linear Relationship**: The boxplot (bottom chart) reveals the relationship isn't purely linear - median prices drop most significantly in the first few distance bands, then stabilize somewhat after 600-800m.

6. **Outliers and Luxury Properties**: High-value outliers exist across all distance bands (visible in both scatter plot and boxplots), indicating premium properties can command high prices regardless of MRT proximity.

7. **Price Floor Consistency**: The lower bounds of prices remain fairly stable across distances, suggesting a baseline housing value independent of MRT access.

8. **Distribution Shape**: The histogram's right-skewed distribution shows fewer properties at extreme distances (>3000m), reflecting Singapore's comprehensive public transit coverage.

These findings confirm that while MRT proximity is a value driver for Singapore properties, it is just one of several factors influencing resale prices, with diminishing impact beyond certain distance thresholds.

Regression Machine Learning Models

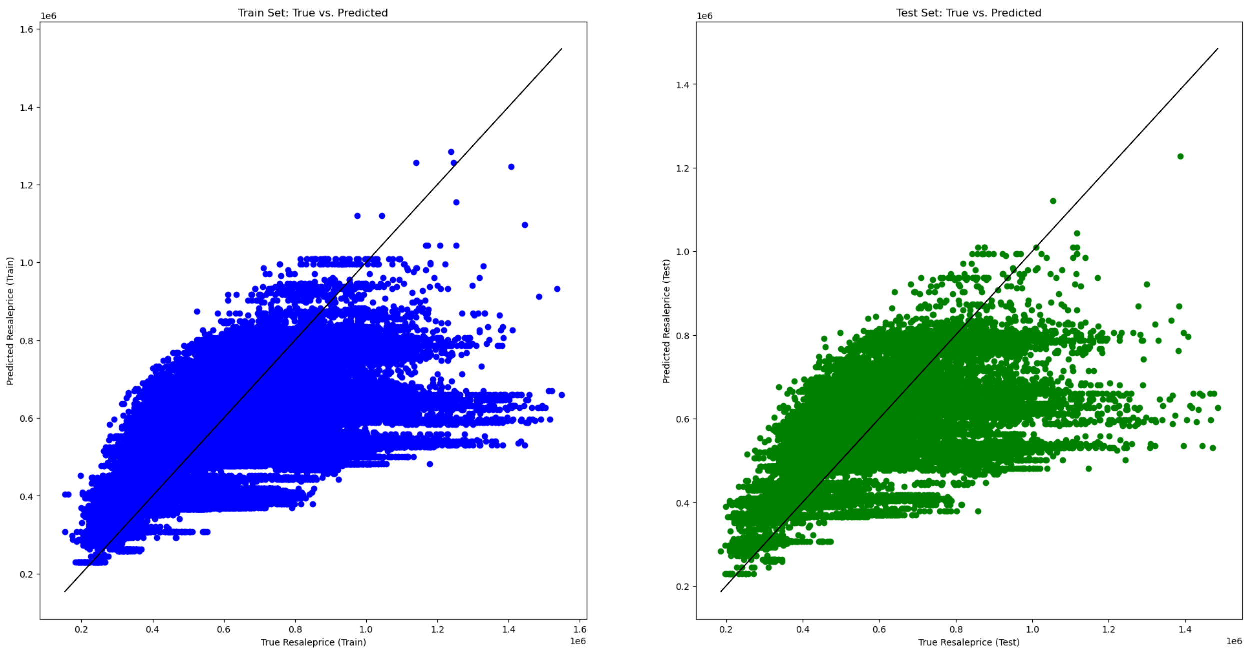
Single Variate

```
from sklearn.model_selection import train_test_split
def mean_sq_err(actual, predicted):
    return np.mean(np.square(np.array(actual) - np.array(predicted)))

def regression_random_split(feature):
    X = pd.DataFrame(df[feature])
    y = pd.DataFrame(df['resale_price_adj'])
    X_train, X_test, y_train, y_test = train_test_split(X, y, 
test_size=0.2)
    linreg = LinearRegression()
    linreg.fit(X_train, y_train)
    print(f"\nRegression for {feature}:")
    print('Intercept \t: b =', linreg.intercept_)
    print("Coefficient \t: a =", linreg.coef_)
    print()
    y_train_pred = linreg.predict(X_train)
    y_test_pred = linreg.predict(X_test)
    print("Goodness of Fit \t\t Train Dataset")
    print("Explained Variance (R²) \t:", linreg.score(X_train, 
y_train))
    print("Mean Squared Error \t\t:", mean_sq_err(y_train, 
y_train_pred))
    print()
    print("Goodness of Fit \t\t Test Dataset")
    print("Explained Variance (R²) \t:", linreg.score(X_test, y_test))
```

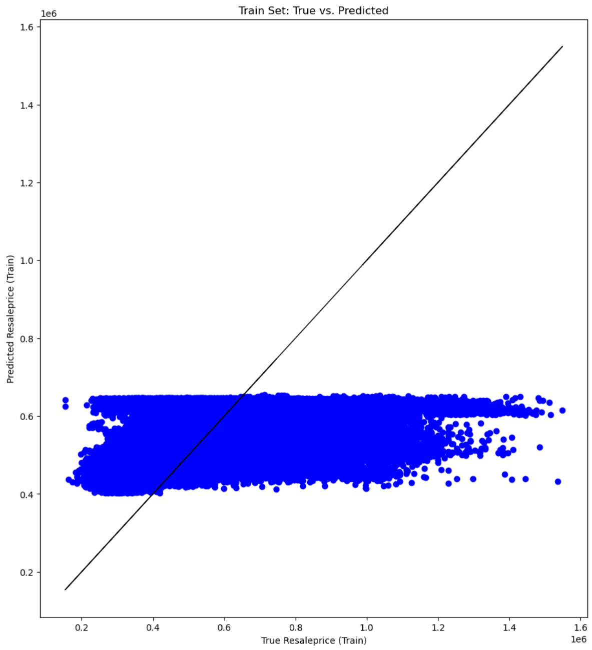
```
    print("Mean Squared Error \t\t:", mean_sq_err(y_test,
y_test_pred))
    print()

    r2_train = linreg.score(X_train, y_train)
    mse_train = mean_sq_err(y_train, y_train_pred)
    r2_test = linreg.score(X_test, y_test)
    mse_test = mean_sq_err(y_test, y_test_pred)

    if r2_train > 0.95:
        print(f'Train graph for {feature} is overfitted!')
    else:
        print(f'Train graph for {feature} is NOT overfitted!')
    f, axes = plt.subplots(1, 2, figsize=(24, 12))
    axes[0].scatter(y_train, y_train_pred, color='blue')
    axes[0].plot(y_train, y_train, 'k-', linewidth=1)
    axes[0].set_xlabel("True Resaleprice (Train)")
    axes[0].set_ylabel("Predicted Resaleprice (Train)")
    axes[0].set_title("Train Set: True vs. Predicted")

    axes[1].scatter(y_test, y_test_pred, color='green')
    axes[1].plot(y_test, y_test, 'k-', linewidth=1)
    axes[1].set_xlabel("True Resaleprice (Test)")
    axes[1].set_ylabel("Predicted Resaleprice (Test)")
    axes[1].set_title("Test Set: True vs. Predicted")

    plt.show()

    r2_test = linreg.score(X_test, y_test)
    mse_test = mean_sq_err(y_test, y_test_pred)

    return r2_test, mse_test
features = ['floor_area_sqm',
'remaining_lease_months','distance_to_mrt_m']
r2_graph_test = []
mse_graph_test = []

for feature in features:
    r2_test, mse_test = regression_random_split(feature)
    r2_graph_test.append(r2_test)
    mse_graph_test.append(mse_test)
```

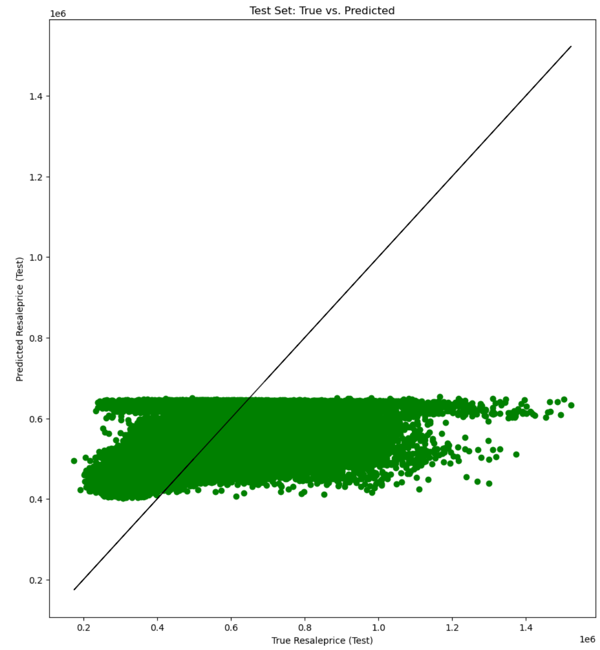
```
Regression for floor_area_sqm:
Intercept  : b = [79252.90731962]
Coefficient     : a = [[4844.14109573]]

Goodness of Fit        Train Dataset
Explained Variance (R²)     : 0.39673942514529204
Mean Squared Error          : 20524505664.646736
```

```
Goodness of Fit        Test Dataset
Explained Variance (R²)    : 0.3997721788880477
Mean Squared Error         : 20255394668.43385

Train graph for floor_area_sqm is NOT overfitted!
```

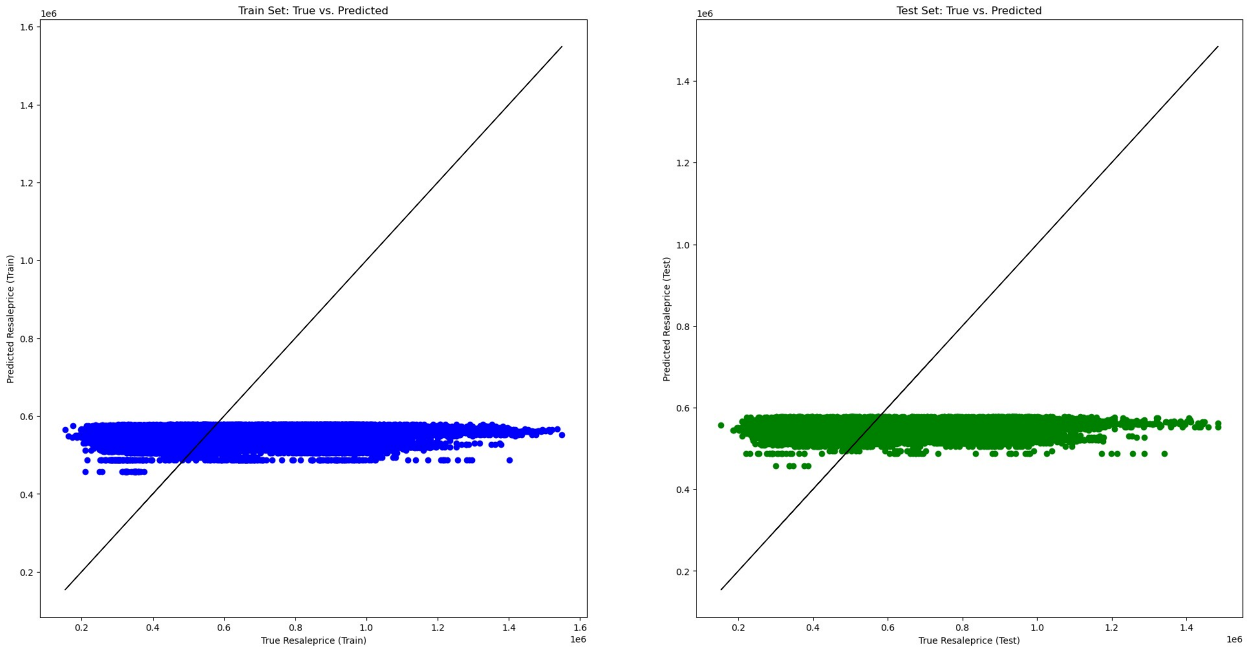



```
Regression for remaining_lease_months:
Intercept  : b = [211077.35215309]
Coefficient     : a = [[378.47807039]]

Goodness of Fit        Train Dataset
Explained Variance (R²)    : 0.1166808477617216
Mean Squared Error         : 30019525070.7819

Goodness of Fit        Test Dataset
Explained Variance (R²)    : 0.11110764306140031
Mean Squared Error         : 30131506025.25585

Train graph for remaining_lease_months is NOT overfitted!
```

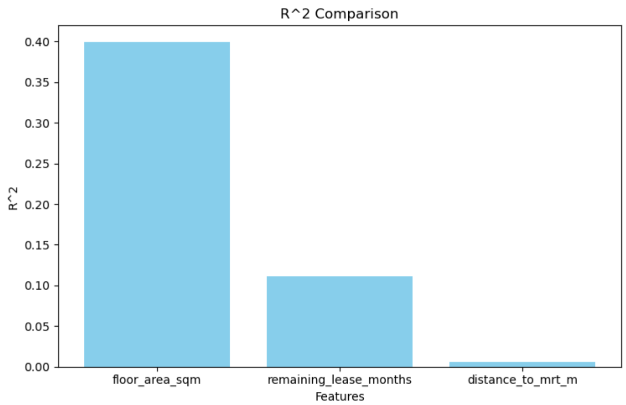


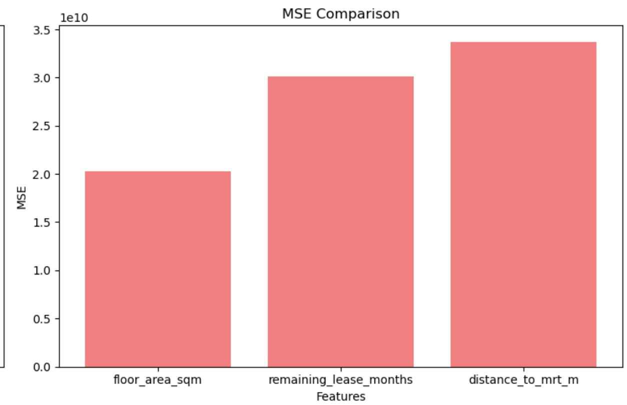



```
Regression for distance_to_mrt_m:
Intercept  : b = [579271.65111906]
Coefficient     : a = [[-26.82873532]]

Goodness of Fit       Train Dataset
Explained Variance (R²)    : 0.00694901556537697
Mean Squared Error         : 33741241560.297737

Goodness of Fit       Test Dataset
Explained Variance (R²)    : 0.005989570769253438
Mean Squared Error         : 33725117063.860012

Train graph for distance_to_mrt_m is NOT overfitted!
```




```
fig, axes = plt.subplots(1, 2, figsize=(15, 5))
axes[0].bar(features, r2_graph_test, color='skyblue')
axes[0].set_xlabel('Features')
axes[0].set_ylabel('R^2')
axes[0].set_title('R^2 Comparison')

axes[1].bar(features, mse_graph_test, color='lightcoral')
axes[1].set_xlabel('Features')
axes[1].set_ylabel('MSE')
axes[1].set_title('MSE Comparison')

plt.tight_layout()
plt.show()

best_r2 = np.argmax(r2_graph_test)
best_r2 = features[best_r2]
print(f'Best model based on highest R² score: {best_r2}')

best_mse = np.argmin(mse_graph_test)
best_mse = features[best_mse]
print(f'Best model based on Prediction Accuracy aka lowest MSE :
{best_mse}')
```






```
Best model based on highest R² score: floor_area_sqm
Best model based on Prediction Accuracy aka lowest MSE :
floor_area_sqm
```

Insight:
floor_area_sqm out of the three numerical variables has the highest R^2 score and lowest mean squared error.

remaining lease in terms of months
The linear regression between the remaining_lease in terms of months and resale price has a poor predictive power, R^2 is 0.1, suggesting lease decay alone does not drive resale prices. Contrary to public perception, the age of flats, or in other words the remaining lease, weakly affects the resale prices. Lease decay does affect resale prices to some extent—but not as strongly as many people assume. Here are some possible reasons:

1. Market Sentiment Remains Resilient Despite concerns, data shows that resale prices of older HDB flats are not crashing. In fact, the number of million-dollar resale flats —including older ones—is rising. This indicates that other factors like size, location, and scarcity of large units matter more to buyers.

2. Government Support Helps Mitigate Concerns Policies like the Voluntary En-bloc Redevelopment Scheme (VERS) and ongoing discussions about lease buyback and upgrading offer homeowners greater security and keep confidence in the resale market relatively stable.

distance_to_mrt_m
Here are possible reasons for its weak predictive power on resale price in our model:

1. **Non-linear or Threshold Effects**

- Buyers might only care whether a flat is near an MRT (e.g., within 500m or 1km), not the exact distance.

- So whether it is 200m or 400m might not matter much — both are close.

- Beyond a certain distance (say >1km), price sensitivity drops significantly.

1. **Measurement Noise**

- The distance computed was using straight-line distance instead of actual walking paths, in order to minimise the computational time in retrieving the distance_to_mrt_m. However, euclidean distance can be misleading — a flat might be 300m away but separated by highways or walls.

last words
The low accuracy across these three factors suggests that no single feature could sufficiently explain resale prices, affirming the complexity of real estate valuation.

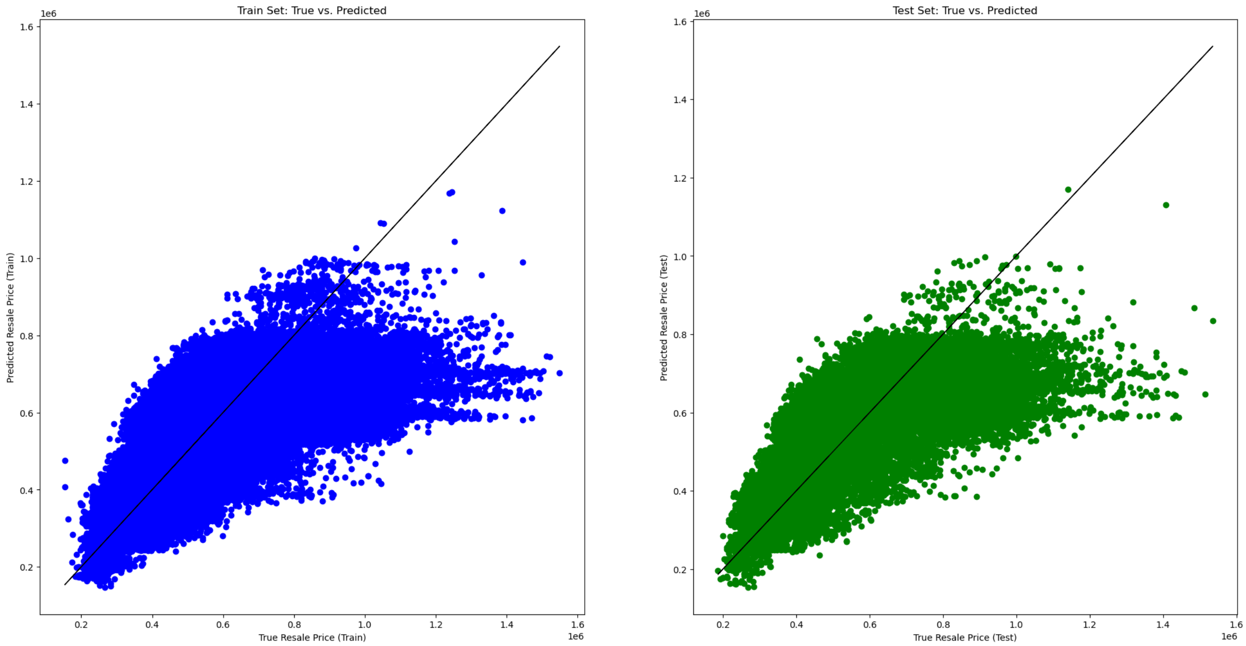
Regression

Multi Variate

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

def mean_sq_err(actual, predicted):
    return np.mean(np.square(np.array(actual) - np.array(predicted)))

def multivariate_regression(features):
    X = df[features]
    y = df['resale_price_adj']
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
    scaler_X = StandardScaler()
    X_train_scaled = scaler_X.fit_transform(X_train)
    X_test_scaled = scaler_X.transform(X_test)
    linreg = LinearRegression()
    linreg.fit(X_train_scaled, y_train)

    y_train_pred = linreg.predict(X_train_scaled)
    y_test_pred = linreg.predict(X_test_scaled)

    print("\nMultivariate Regression:")
    print('Intercept \t:', linreg.intercept_)
    print("Coefficients \t:", linreg.coef_)

    print("\nGoodness of Fit (Train Dataset):")
    print("Explained Variance (R²):", r2_score(y_train, y_train_pred))
    print("Mean Squared Error (MSE):", mean_sq_err(y_train,
y_train_pred))
```

```
    print("\nGoodness of Fit (Test Dataset):")
    print("Explained Variance (R²):", r2_score(y_test, y_test_pred))
    print("Mean Squared Error (MSE):", mean_sq_err(y_test,
y_test_pred))

    f, axes = plt.subplots(1, 2, figsize=(24, 12))
    axes[0].scatter(y_train, y_train_pred, color='blue')
    axes[0].plot(y_train, y_train, 'k-', linewidth=1)
    axes[0].set_xlabel("True Resale Price (Train)")
    axes[0].set_ylabel("Predicted Resale Price (Train)")
    axes[0].set_title("Train Set: True vs. Predicted")

    axes[1].scatter(y_test, y_test_pred, color='green')
    axes[1].plot(y_test, y_test, 'k-', linewidth=1)
    axes[1].set_xlabel("True Resale Price (Test)")
    axes[1].set_ylabel("Predicted Resale Price (Test)")
    axes[1].set_title("Test Set: True vs. Predicted")

    plt.show()
    return linreg.coef_, linreg.intercept_
features = ['floor_area_sqm', 'remaining_lease_months',
'distance_to_mrt_m']
coefficients, intercept = multivariate_regression(features)
```

```
Multivariate Regression:
Intercept  : 549735.7566362439
Coefficients    : [109624.50961208  48761.2109983  -15803.64776443]

Goodness of Fit (Train Dataset):
Explained Variance (R²): 0.47188090061527566
Mean Squared Error (MSE): 17851517220.14021

Goodness of Fit (Test Dataset):
Explained Variance (R²): 0.47750607481515206
Mean Squared Error (MSE): 18116175343.93269
```



Insights:
Multi-Variate Regression Accuracy in terms of R square score and MSE are higher than all of the Single-Variate Regresion, showing that in predicting resale prices, it is an interplacy of factors, and not solely dependent on just one numerical variable. Therefore, Multi-Variate is preferred over to Single-Variate Regression Model.

Also, this revealed a key insight: resale prices are influenced by a combination of factors, **not isolated attributes**. However, the linear model was still limited in capturing non-linear relationships.

Regression

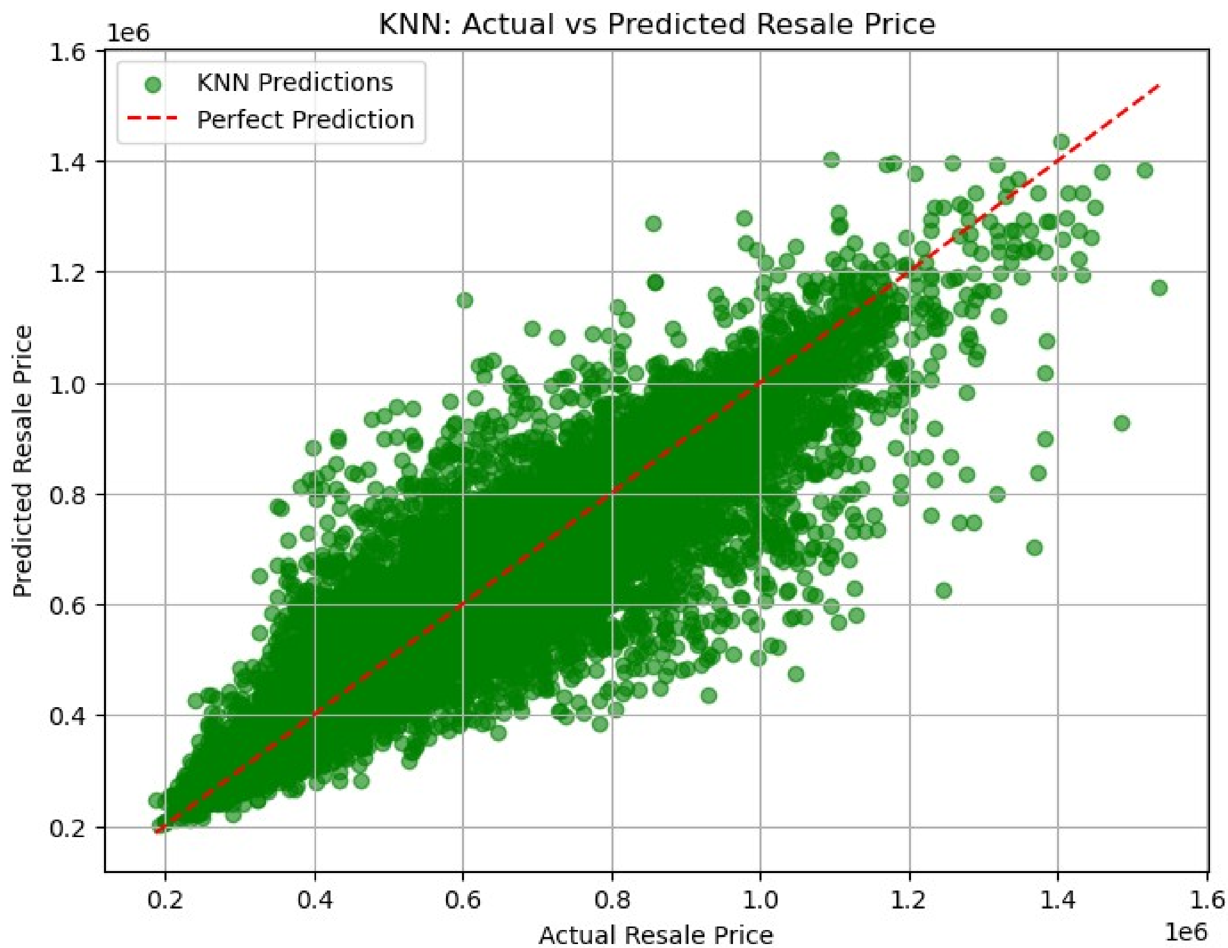
K-Nearest Neighbors KNN

a regression ML, predicts the resale price of HDB flats by averaging the prices of the 5 nearest neighbors in the feature space, after scaling the input features.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

X = df[['floor_area_sqm',
'remaining_lease_months','distance_to_mrt_m']].values
y = df['resale_price_adj'].values
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
scaler_X = StandardScaler()
X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
y_pred = knn.predict(X_test_scaled)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse:.2f}")
print(f"R² Score: {r2:.4f}")

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.6, color='green', label='KNN
Predictions')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
         'r--', label='Perfect Prediction')
plt.xlabel('Actual Resale Price')
plt.ylabel('Predicted Resale Price')
plt.title('KNN: Actual vs Predicted Resale Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\Crystaline\anaconda3\Lib\site-packages\sklearn\metrics\
_regression.py:492: FutureWarning:

'squared' is deprecated in version 1.4 and will be removed in 1.6. To
calculate the root mean squared error, use the
function'root_mean_squared_error'.


RMSE: 68624.99
R² Score: 0.8642
```



Insights:
This is something new we learnt:
To improve model performance, we turned to K-Nearest Neighbors (KNN) regression. Unlike traditional models, KNN is instance-based and non-parametric. It does not make prior assumptions about the data distribution, but instead makes predictions by referencing nearby points in the training set.
Features Used: Floor Area, Remaining Lease, Distance to MR T Preprocessing: Data scaling appli ed Evaluation Metrics: RMSE and R^2 Sc ore

KNN significantly outperformed the previous models with an R^2 score of 86%, capturing non-linear patterns that linear models missed. This model highlighted the strength of memory-based learning for dynamic real-world data.

Support Vector Regressor (SVR)

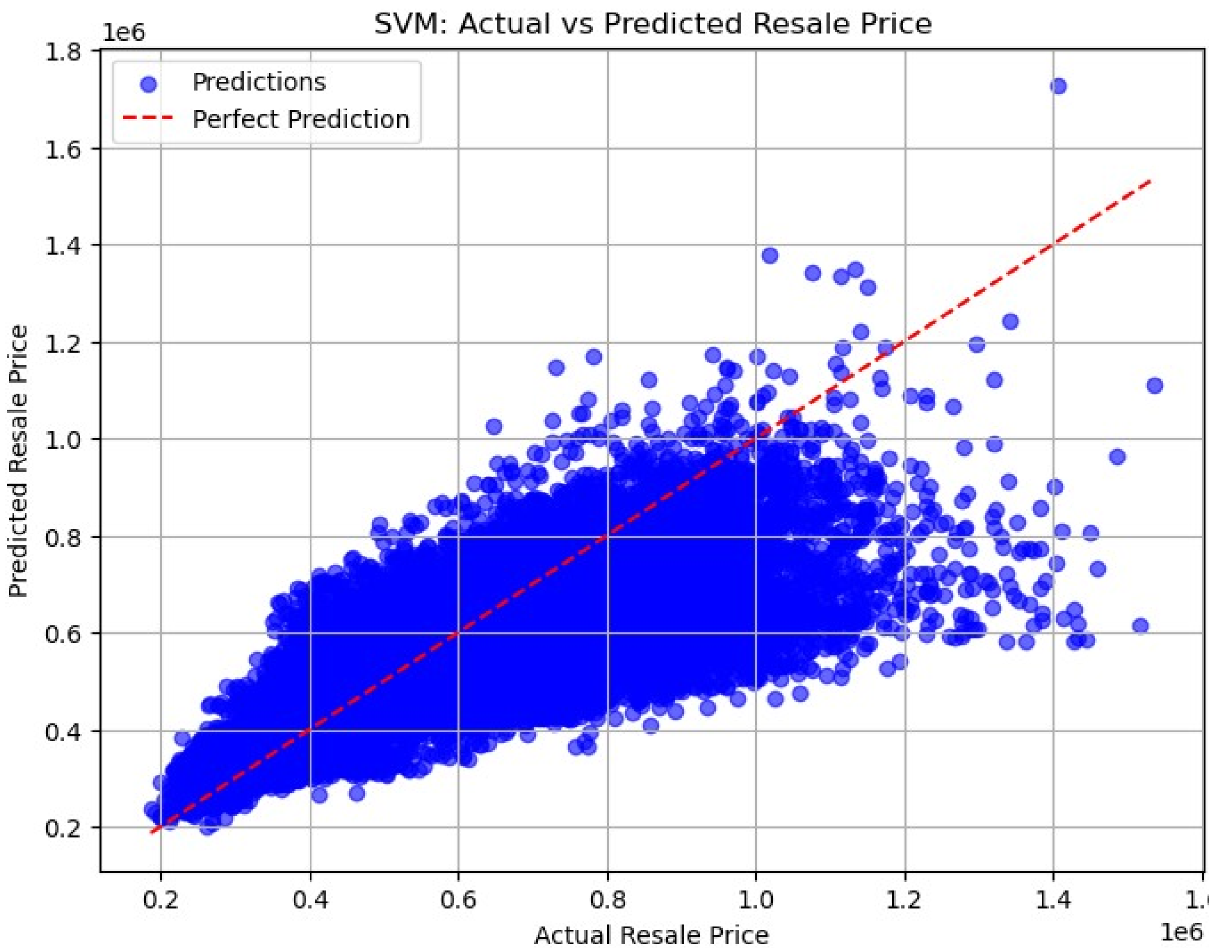
a non-linear svm regression model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVR
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error

features = ['floor_area_sqm', 'remaining_lease_months',
'distance_to_mrt_m']
target = 'resale_price_adj'

X = df[features].values
y = df[target].values
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
scaler_X = StandardScaler()
scaler_y = StandardScaler()

X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)
y_train_scaled = scaler_y.fit_transform(y_train.reshape(-1,
1)).flatten()

svr = SVR(kernel='rbf', C=100, epsilon=0.1)
svr.fit(X_train_scaled, y_train_scaled)

y_train_pred_scaled = svr.predict(X_train_scaled)
y_train_pred =
scaler_y.inverse_transform(y_train_pred_scaled.reshape(-1,
1)).flatten()
y_pred_scaled = svr.predict(X_test_scaled)
y_pred = scaler_y.inverse_transform(y_pred_scaled.reshape(-1,
1)).flatten()
r2_train = r2_score(y_train, y_train_pred)
mse_train = mean_squared_error(y_train, y_train_pred)

r2_test = r2_score(y_test, y_pred)
mse_test = mean_squared_error(y_test, y_pred)

print("Goodness of Fit \t\t Train Dataset")
print(f"Explained Variance (R²) \t: {r2_train}")
print(f"Mean Squared Error \t\t: {mse_train:.4f}\n")

print("Goodness of Fit \t\t Test Dataset")
print(f"Explained Variance (R²) \t: {r2_test}")
print(f"Mean Squared Error \t\t: {mse_test:.4f}")

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.6, color='blue',
label='Predictions')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
         'r--', label='Perfect Prediction')
plt.xlabel('Actual Resale Price')
```

```
plt.ylabel('Predicted Resale Price')
plt.title('SVM: Actual vs Predicted Resale Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
Goodness of Fit        Train Dataset
Explained Variance (R²)     : 0.6009640455066032
Mean Squared Error          : 13488240098.4796

Goodness of Fit        Test Dataset
Explained Variance (R²)     : 0.5995907454744882
Mean Squared Error          : 13883193496.9412
```



Insights:
Moderate Accuracy: The model captures general pricing trends, but predictions show noticeable variance, especially for outliers.
Systematic Bias: SVR tends to overestimate lower-value properties and underestimate higher-value ones. Error Spread: A relatively high Mean Squared Error (13.4M–13.8M) indicates significant unexplained variance.

Outlier Sensitivity: Predictions are least reliable for top-tier properties (1.5M SGD).
Best Performance Range: Predictions are most stable in the 600k–1M SGD range, where data is denser.

While SVR improves on linear regression (48% R^2), it lags behind KNN's ability to capture complex, non-linear patterns. Still, its stable performance across datasets makes it a viable option for moderate-precision use cases, particularly where model interpretability and consistency are valued. The best model to use will be: KNN.

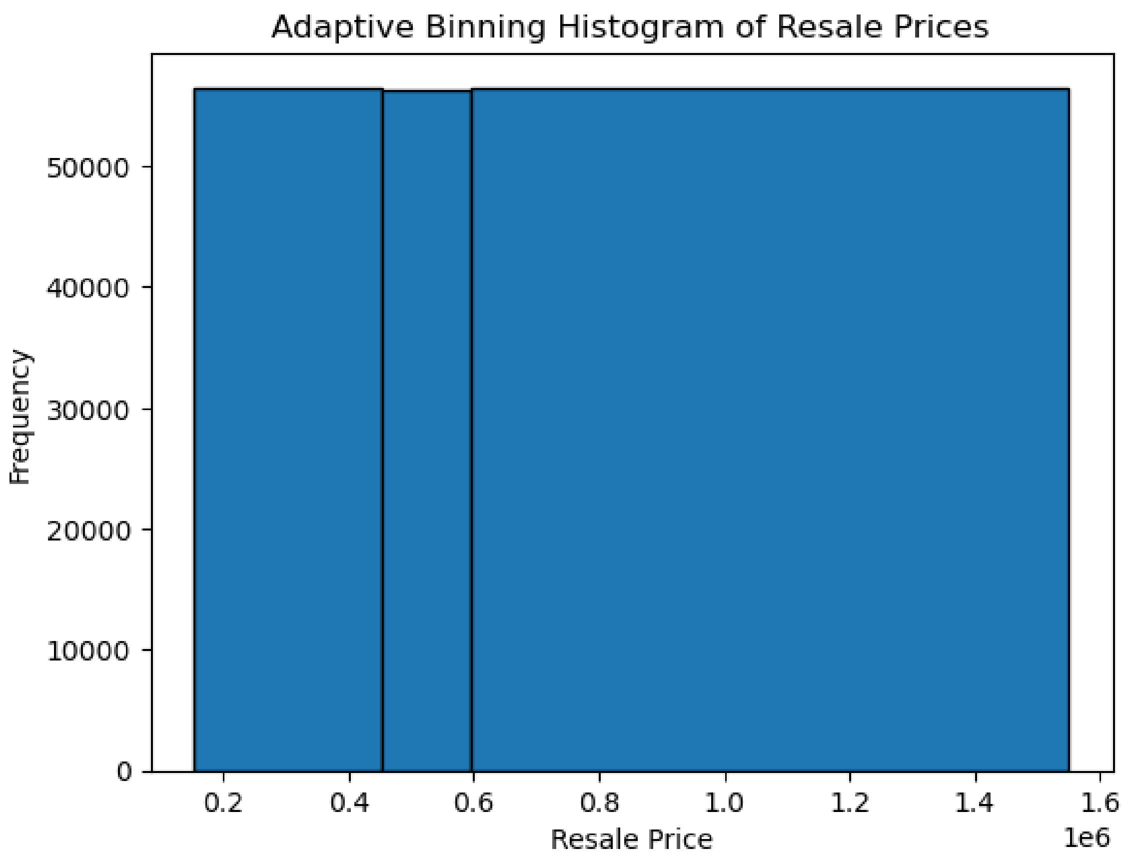
Classification ML

CatBoost

3 Bin

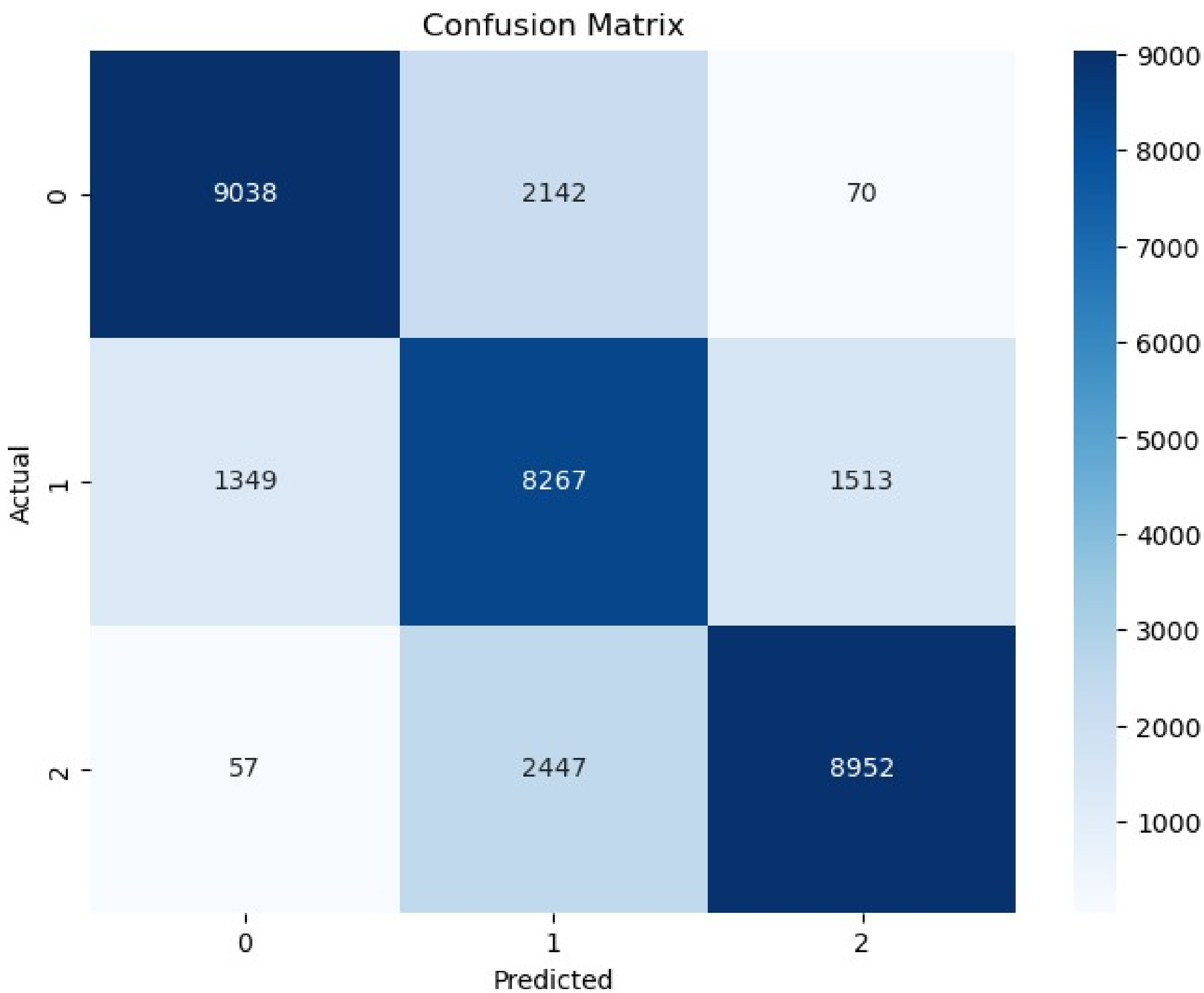
```
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df_categ = df[['town', 'flat_type', 'storey_range',
'flat_model']].copy()
n_bins = 3
bin_edges = np.percentile(df['resale_price_adj'], np.linspace(0, 100,
n_bins + 1))
df_categ['resale_price_category'] = pd.cut(df['resale_price_adj'],
bins=bin_edges, labels=[f'Bin {i+1}' for i in range(n_bins)],
include_lowest=True).astype(str)

plt.hist(df['resale_price_adj'], bins=bin_edges, edgecolor='black',
align='mid')
plt.title("Adaptive Binning Histogram of Resale Prices")
plt.xlabel("Resale Price")
plt.ylabel("Frequency")
plt.show()

print(df_categ['resale_price_category'].value_counts())
print("Bin edges (resale price percentiles):")
for i, edge in enumerate(bin_edges):
    print(f"{int(i)}: {edge:,.2f}")

def train_and_evaluate_catboost(df, target_col, cat_features):
    df = df.copy()
    for col in cat_features:
        if df[col].dtype == 'object':
            df.loc[:, col] = df[col].str.extract(r'(\d+)')
```

```
[0].fillna(df[col]).astype(str)
    for col in cat_features:
        df.loc[:, col] = df[col].astype(str)
    X = df.drop(columns=[target_col])
    y = df[target_col].astype(str)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
    model = CatBoostClassifier(iterations=500, depth=6,
learning_rate=0.05, loss_function='MultiClass', verbose=100)
    model.fit(X_train, y_train, cat_features=cat_features)
    y_pred = model.predict(X_test).ravel()
    print("Classification Report:\n", classification_report(y_test,
y_pred))
    print(f"Accuracy: {(y_pred == y_test).mean():.4f}")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    plt.figure(figsize=(8, 6))
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d",
cmap="Blues")
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

cat_features = ['town', 'flat_type', 'storey_range', 'flat_model']
train_and_evaluate_catboost(df_categ,
target_col='resale_price_category', cat_features=cat_features)
```




```
resale_price_category
Bin 2    56439
Bin 1    56408
Bin 3    56326
Name: count, dtype: int64
Bin edges (resale price percentiles):
0: 153,584.67
1: 452,562.82
2: 594,447.16
3: 1,549,310.22
0:	learn: 1.0701790	total: 530ms	remaining: 4m 24s
100:	learn: 0.5100136	total: 36.4s	remaining: 2m 23s
200:	learn: 0.4935290	total: 1m 12s	remaining: 1m 48s
300:	learn: 0.4885061	total: 1m 50s	remaining: 1m 13s
400:	learn: 0.4857181	total: 2m 27s	remaining: 36.3s
499:	learn: 0.4838671	total: 3m 3s	remaining: 0us
Classification Report:
              precision    recall  f1-score   support

       Bin 1       0.87      0.80      0.83     11250
       Bin 2       0.64      0.74      0.69     11129
       Bin 3       0.85      0.78      0.81     11456
```

```
    accuracy                           0.78     33835
   macro avg       0.79      0.78      0.78     33835
weighted avg       0.79      0.78      0.78     33835

Accuracy: 0.7760
Confusion Matrix:
 [[9038 2142   70]
 [1349 8267 1513]
 [  57 2447 8952]]
```

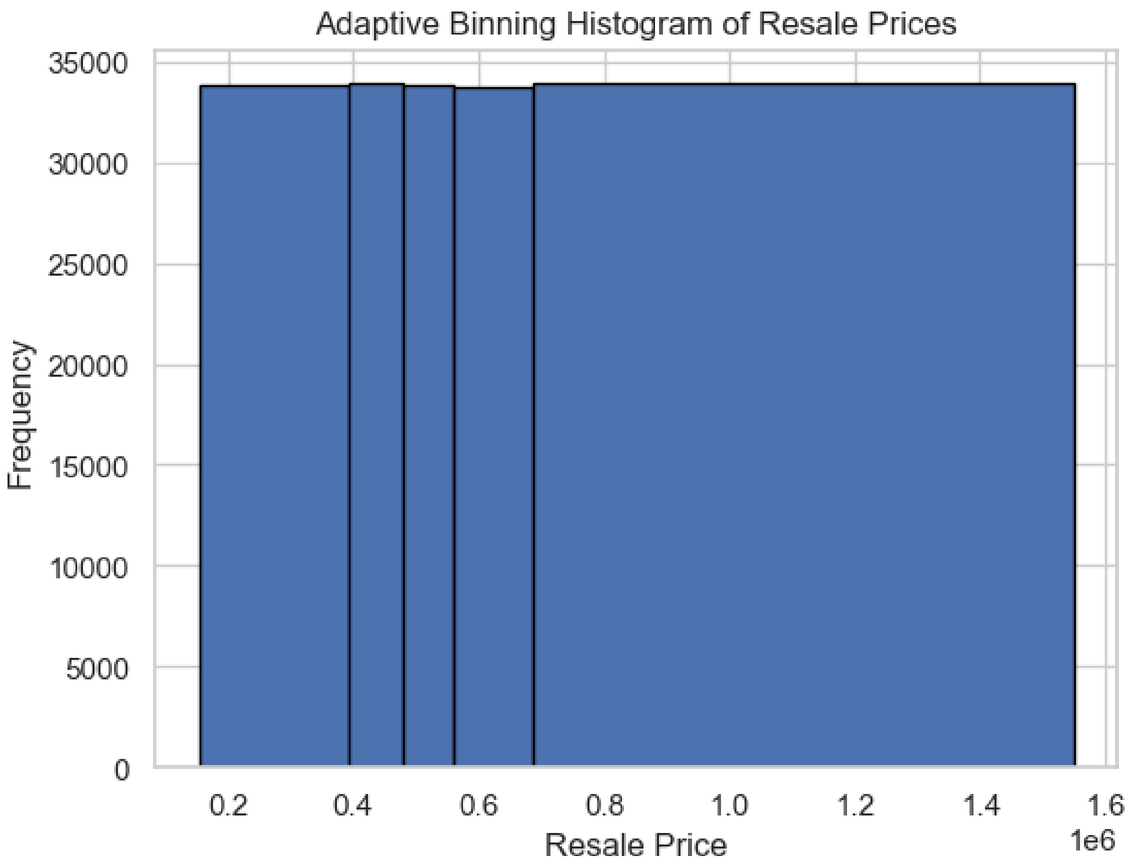


5 Bin

```
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

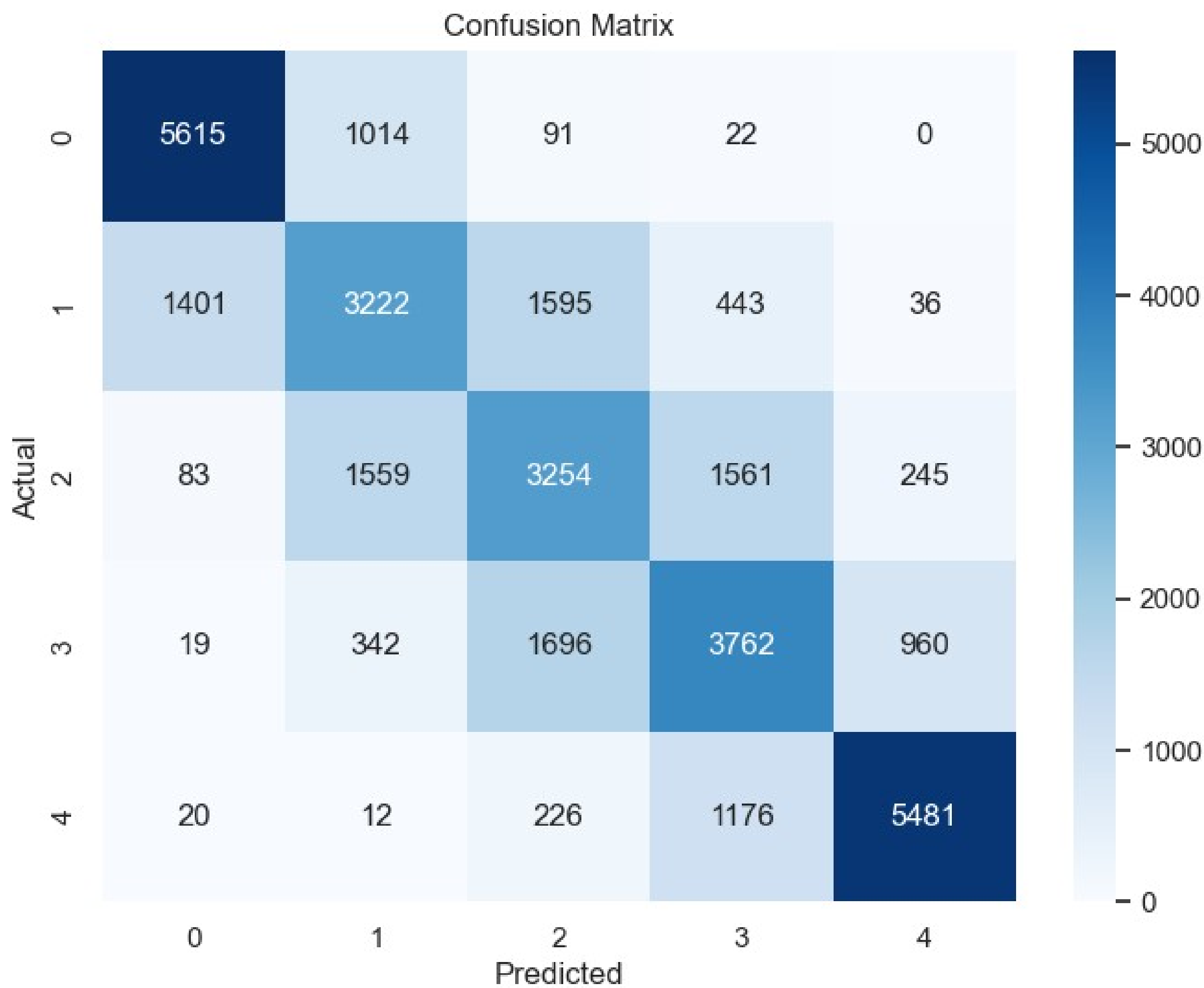
```
df_categ = df[['town', 'flat_type', 'storey_range',
'flat_model']].copy()
n_bins = 5
bin_edges = np.percentile(df['resale_price_adj'], np.linspace(0, 100,
n_bins + 1))
df_categ['resale_price_category'] = pd.cut(df['resale_price_adj'],
bins=bin_edges, labels=[f'Bin {i+1}' for i in range(n_bins)],
include_lowest=True).astype(str)

plt.hist(df['resale_price_adj'], bins=bin_edges, edgecolor='black',
align='mid')
plt.title("Adaptive Binning Histogram of Resale Prices")
plt.xlabel("Resale Price")
plt.ylabel("Frequency")
plt.show()

print(df_categ['resale_price_category'].value_counts())
print("Bin edges (resale price percentiles):")
for i, edge in enumerate(bin_edges):
    print(f"{int(i)}: {edge:,.2f}")

def train_and_evaluate_catboost(df, target_col, cat_features):
    df = df.copy()
    for col in cat_features:
        if df[col].dtype == 'object':
            df.loc[:, col] = df[col].str.extract(r'(\d+)')
[0].fillna(df[col]).astype(str)
    for col in cat_features:
        df.loc[:, col] = df[col].astype(str)
    X = df.drop(columns=[target_col])
    y = df[target_col].astype(str)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
    model = CatBoostClassifier(iterations=500, depth=6,
learning_rate=0.05, loss_function='MultiClass', verbose=100)
    model.fit(X_train, y_train, cat_features=cat_features)
    y_pred = model.predict(X_test).ravel()
    print("Classification Report:\n", classification_report(y_test,
y_pred))
    print(f"Accuracy: {(y_pred == y_test).mean():.4f}")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    plt.figure(figsize=(8, 6))
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d",
cmap="Blues")
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

```python
cat_features = ['town', 'flat_type', 'storey_range', 'flat_model']
train_and_evaluate_catboost(df_categ,
target_col='resale_price_category', cat_features=cat_features)
```




```
resale_price_category
Bin 1    33899
Bin 4    33875
Bin 3    33832
Bin 5    33794
Bin 2    33773
Name: count, dtype: int64
Bin edges (resale price percentiles):
0: 153,584.67
1: 391,786.44
2: 479,602.65
3: 560,514.94
4: 687,693.78
5: 1,549,310.22
0:    learn: 1.5532086 total: 676ms    remaining: 5m 37s
100:  learn: 0.8573067 total: 1m 14s   remaining: 4m 52s
200:  learn: 0.8282262 total: 2m 27s   remaining: 3m 39s
300:  learn: 0.8186798 total: 3m 40s   remaining: 2m 25s
```

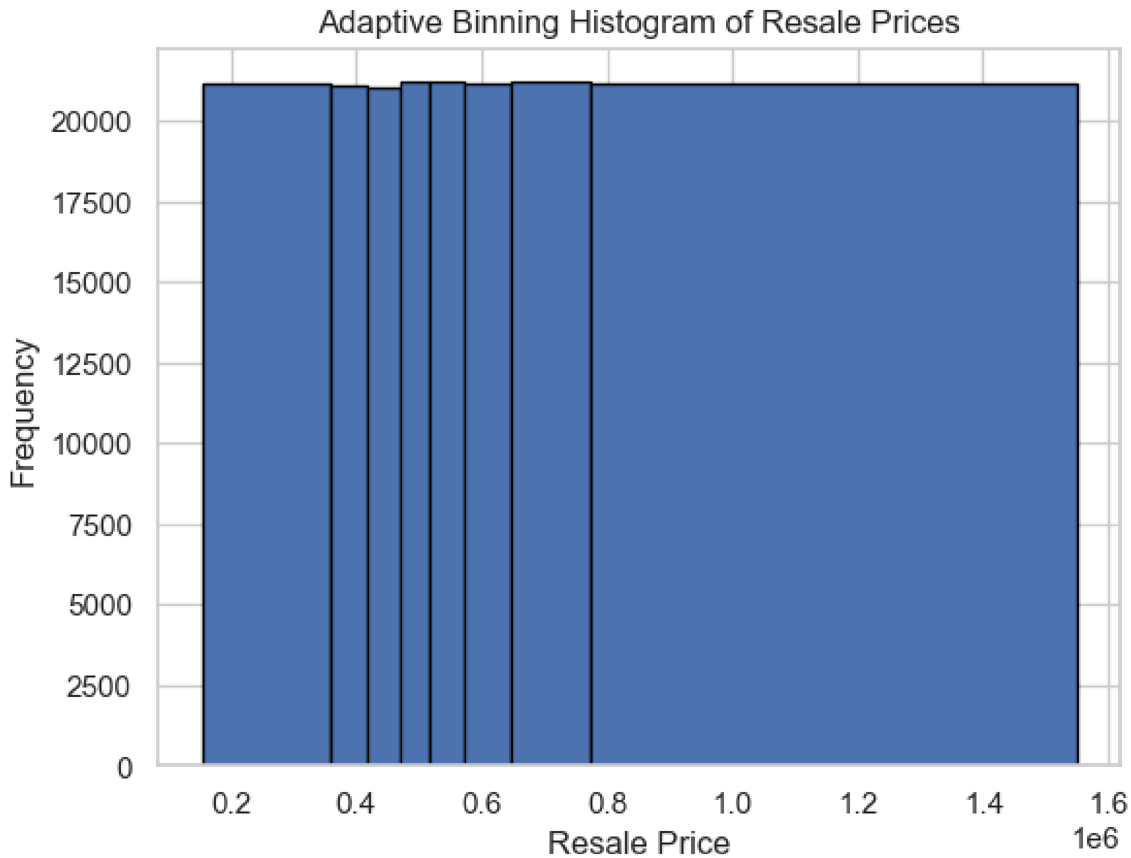
```
400:	learn: 0.8137683	total: 4m 54s	remaining: 1m 12s
499:	learn: 0.8102013	total: 6m 7s	remaining: 0us
Classification Report:
               precision    recall  f1-score   support

       Bin 1       0.79      0.83      0.81      6742
       Bin 2       0.52      0.48      0.50      6697
       Bin 3       0.47      0.49      0.48      6702
       Bin 4       0.54      0.55      0.55      6779
       Bin 5       0.82      0.79      0.80      6915

    accuracy                           0.63     33835
   macro avg       0.63      0.63      0.63     33835
weighted avg       0.63      0.63      0.63     33835

Accuracy: 0.6305
Confusion Matrix:
 [[5615 1014   91   22    0]
 [1401 3222 1595  443   36]
 [  83 1559 3254 1561  245]
 [  19  342 1696 3762  960]
 [  20   12  226 1176 5481]]
```



8 Bin

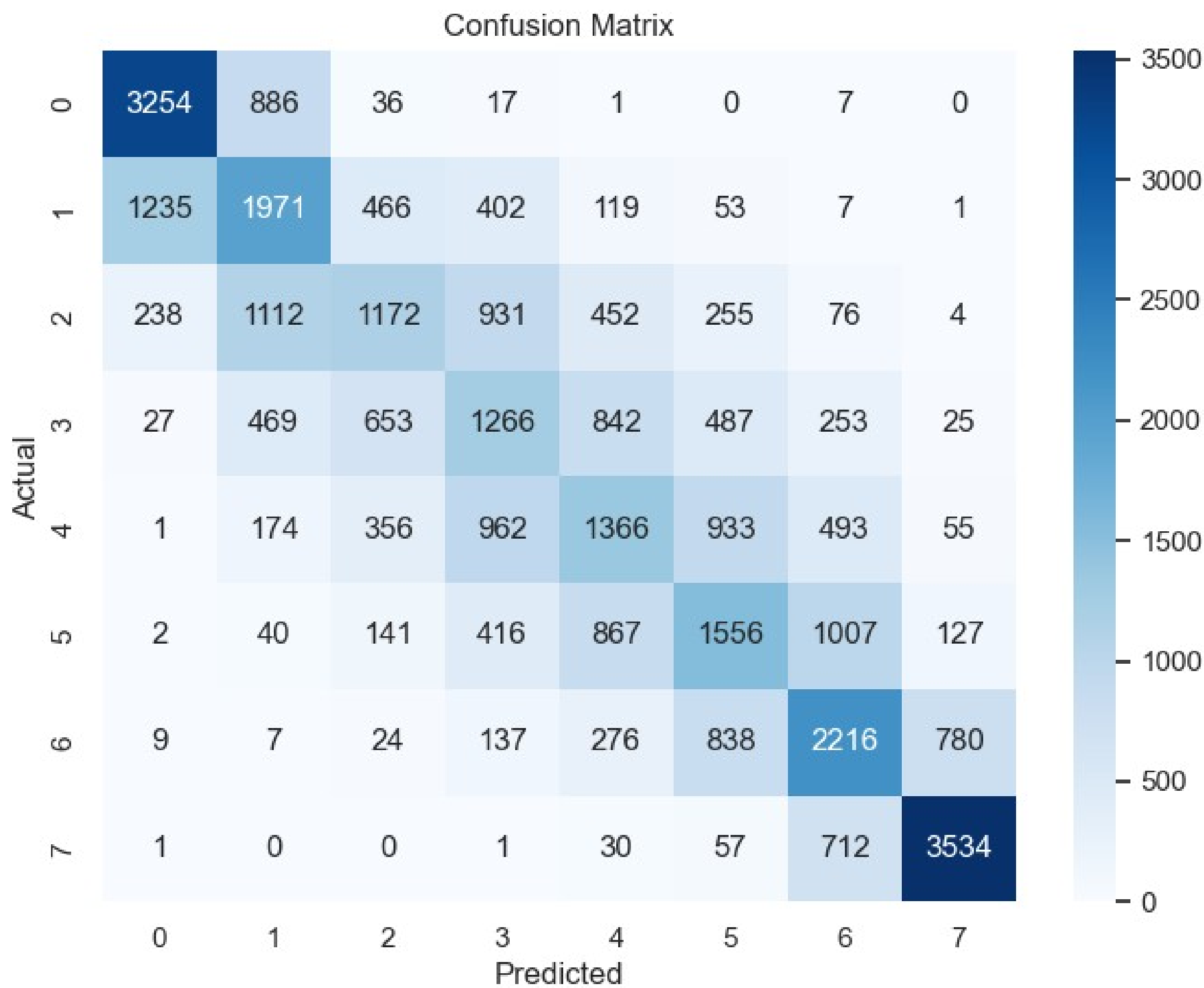
```
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df_categ = df[['town', 'flat_type', 'storey_range',
'flat_model']].copy()
n_bins = 8
bin_edges = np.percentile(df['resale_price_adj'], np.linspace(0, 100,
n_bins + 1))
df_categ['resale_price_category'] = pd.cut(df['resale_price_adj'],
bins=bin_edges, labels=[f'Bin {i+1}' for i in range(n_bins)],
include_lowest=True).astype(str)
```

```
plt.hist(df['resale_price_adj'], bins=bin_edges, edgecolor='black',
align='mid')
plt.title("Adaptive Binning Histogram of Resale Prices")
plt.xlabel("Resale Price")
plt.ylabel("Frequency")
plt.show()

print(df_categ['resale_price_category'].value_counts())
print("Bin edges (resale price percentiles):")
for i, edge in enumerate(bin_edges):
    print(f"{int(i)}: {edge:,.2f}")

def train_and_evaluate_catboost(df, target_col, cat_features):
    df = df.copy()
    for col in cat_features:
        if df[col].dtype == 'object':
            df.loc[:, col] = df[col].str.extract(r'(\d+)')
[0].fillna(df[col]).astype(str)
    for col in cat_features:
        df.loc[:, col] = df[col].astype(str)
    X = df.drop(columns=[target_col])
    y = df[target_col].astype(str)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
    model = CatBoostClassifier(iterations=500, depth=6,
learning_rate=0.05, loss_function='MultiClass', verbose=100)
    model.fit(X_train, y_train, cat_features=cat_features)
    y_pred = model.predict(X_test).ravel()
    print("Classification Report:\n", classification_report(y_test,
y_pred))
    print(f"Accuracy: {(y_pred == y_test).mean():.4f}")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    plt.figure(figsize=(8, 6))
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d",
cmap="Blues")
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

cat_features = ['town', 'flat_type', 'storey_range', 'flat_model']
train_and_evaluate_catboost(df_categ,
target_col='resale_price_category', cat_features=cat_features)
```




```
resale_price_category
Bin 5    21362
Bin 7    21276
Bin 2    21223
Bin 3    21217
Bin 1    21150
Bin 4    21026
Bin 8    21013
Bin 6    20906
Name: count, dtype: int64
Bin edges (resale price percentiles):
0: 153,584.67
1: 357,217.66
2: 415,807.52
3: 470,123.64
4: 517,791.07
5: 573,382.75
6: 645,784.37
7: 772,839.00
8: 1,549,310.22
0:    learn: 2.0108435 total: 1.67s    remaining: 13m 55s
100:  learn: 1.2390975 total: 2m 50s   remaining: 11m 13s
200:  learn: 1.2029291 total: 5m 33s   remaining: 8m 16s
```

```
300:	learn: 1.1891126	total: 8m 17s	remaining: 5m 29s
400:	learn: 1.1818553	total: 11m 2s	remaining: 2m 43s
499:	learn: 1.1766565	total: 13m 43s	remaining: 0us
Classification Report:
               precision    recall  f1-score   support

       Bin 1       0.68      0.77      0.73      4201
       Bin 2       0.42      0.46      0.44      4254
       Bin 3       0.41      0.28      0.33      4240
       Bin 4       0.31      0.31      0.31      4022
       Bin 5       0.35      0.31      0.33      4340
       Bin 6       0.37      0.37      0.37      4156
       Bin 7       0.46      0.52      0.49      4287
       Bin 8       0.78      0.82      0.80      4335

    accuracy                           0.48     33835
   macro avg       0.47      0.48      0.47     33835
weighted avg       0.47      0.48      0.48     33835

Accuracy: 0.4828
Confusion Matrix:
 [[3254  886   36   17    1    0    7    0]
 [1235 1971  466  402  119   53    7    1]
 [ 238 1112 1172  931  452  255   76    4]
 [  27  469  653 1266  842  487  253   25]
 [   1  174  356  962 1366  933  493   55]
 [   2   40  141  416  867 1556 1007  127]
 [   9    7   24  137  276  838 2216  780]
 [   1    0    0    1   30   57  712 3534]]
```



Insights:
Adaptive binning is something new we learnt and through adaptive binning of price range, we effectively prevent imbalance of data. We also know that the prices goes as low as 160k to as high as 1.6 million, so binning needs to be effective. We Trial and error with 3,5 and 8 bins respectively together with variables like Town, Flat type, models and Storey. Looking at the results, we can see that 3 bins worked the best but comes at a limitation of over generalising of the results, while 8 being more precise in the price range prediction gave a lower accuracy. This means that 5 bins seems a better one.

XGBoost

Classification ML

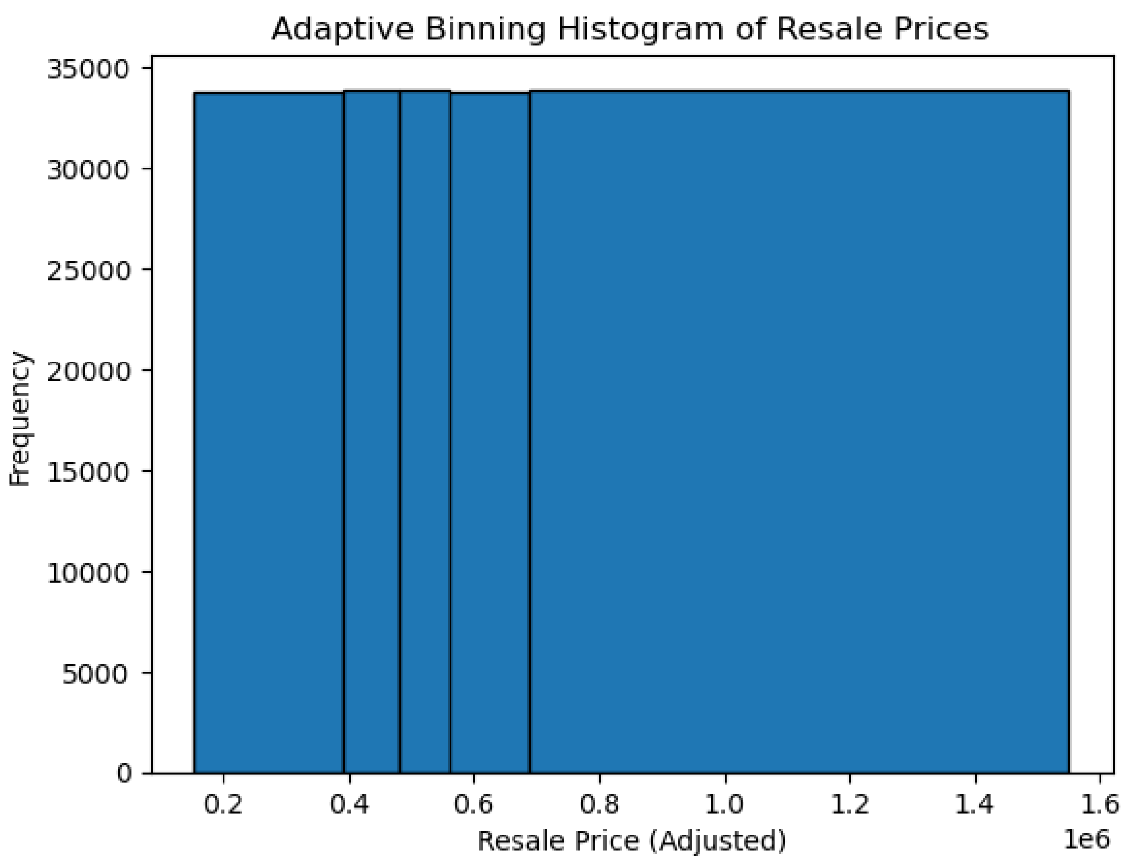
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
```

```python
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from xgboost import XGBClassifier

n_bins = 5
bin_edges = np.percentile(df['resale_price_adj'], np.linspace(0, 100,
n_bins + 1))

df['resale_price_bin'] = pd.cut(
    df['resale_price_adj'],
    bins=bin_edges,
    labels=False,
    include_lowest=True
)

df = df.dropna(subset=['resale_price_bin'])
df['resale_price_bin'] = df['resale_price_bin'].astype(int)

features = ['town', 'flat_type', 'avg_storey', 'flat_model']
X = df[features]
y = df['resale_price_bin']

print("Class distribution:\n", y.value_counts().sort_index())

categorical_features = ['town', 'flat_type', 'avg_storey',
'flat_model']
preprocessor = ColumnTransformer(
    transformers=[('cat', OneHotEncoder(handle_unknown='ignore'),
categorical_features)]
)
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', XGBClassifier(
        use_label_encoder=False,
        eval_metric='mlogloss',
        colsample_bytree=0.8,
        learning_rate=0.05,
        max_depth=10,
        n_estimators=300,
        subsample=0.8
    ))
])
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

start = time.time()
pipeline.fit(X_train, y_train)
```

```
print("Fit time:", time.time() - start)

start = time.time()
y_pred = pipeline.predict(X_test)
print("Predict time:", time.time() - start)

print("Classification Report:\n", classification_report(y_test,
y_pred))
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(f"Classification Accuracy: {accuracy:.4f}")
plt.hist(df['resale_price_adj'], bins=bin_edges, edgecolor='black')
plt.title("Adaptive Binning Histogram of Resale Prices")
plt.xlabel("Resale Price (Adjusted)")
plt.ylabel("Frequency")
plt.show()
print("Bin edges (resale price percentiles):")
for i, edge in enumerate(bin_edges):
    print(f"{i}: {edge:,.2f}")
import seaborn as sns
labels = [f'Bin {i+1}' for i in range(n_bins)]

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=labels, yticklabels=labels)

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix (CatBoost)')
plt.tight_layout()
plt.show()
```

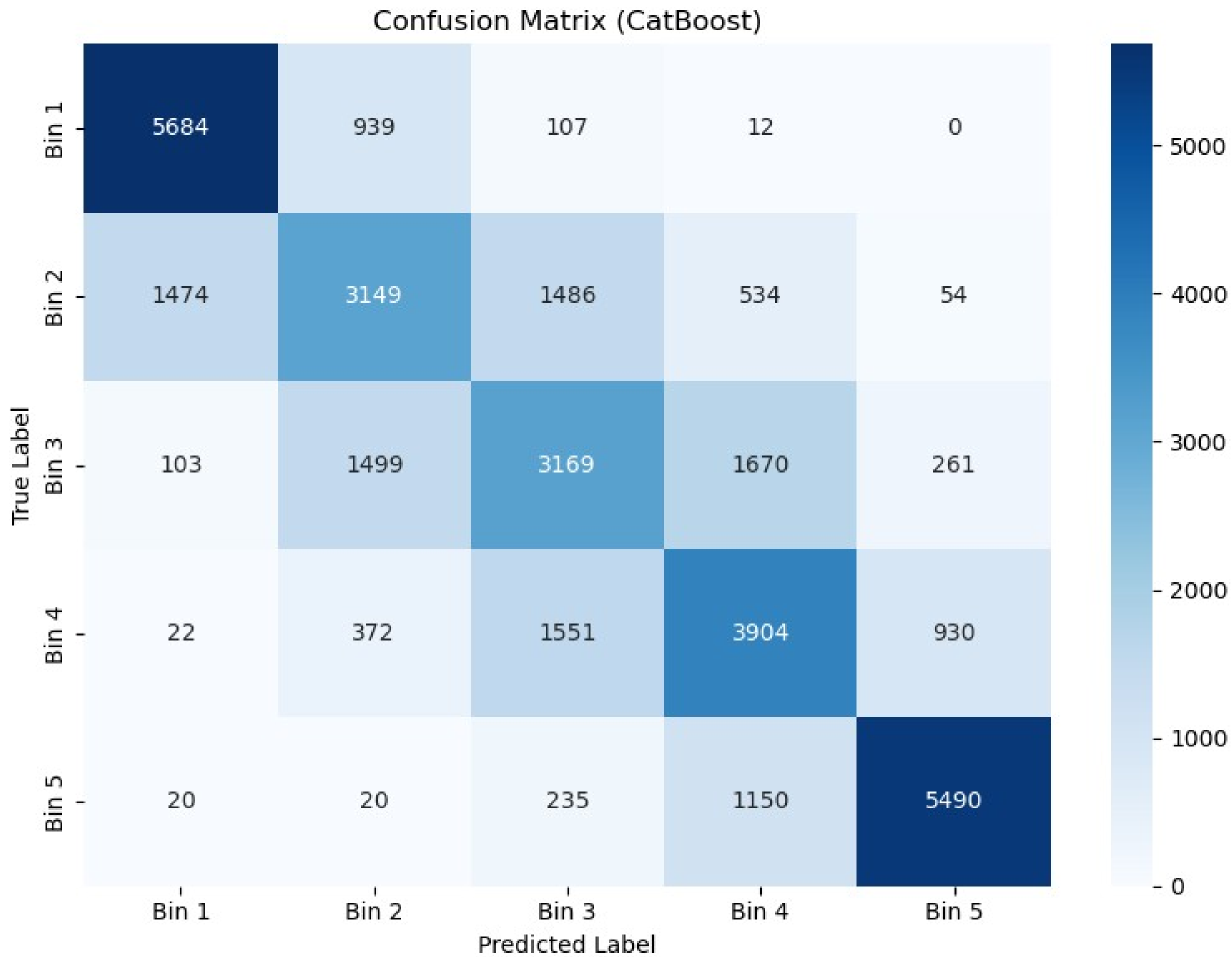
```
Class distribution:
 resale_price_bin
0    33899
1    33773
2    33832
3    33875
4    33794
Name: count, dtype: int64

C:\Users\Crystaline\anaconda3\Lib\site-packages\xgboost\core.py:158:
UserWarning:

[01:25:30] WARNING: D:\bld\xgboost-split_1737531313485\work\src\
learner.cc:740:
Parameters: { "use_label_encoder" } are not used.
```

```
Fit time: 36.759888887405396
Predict time: 3.4898886680603027
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.84      0.81      6742
           1       0.53      0.47      0.50      6697
           2       0.48      0.47      0.48      6702
           3       0.54      0.58      0.56      6779
           4       0.82      0.79      0.80      6915

    accuracy                           0.63     33835
   macro avg       0.63      0.63      0.63     33835
weighted avg       0.63      0.63      0.63     33835

Classification Accuracy: 0.6324
```




```
Bin edges (resale price percentiles):
0: 153,584.67
1: 391,786.44
2: 479,602.65
3: 560,514.94
```

```
4: 687,693.78
5: 1,549,310.22
```



Insights:
This is something new we learnt:
To further validate our approach, we employed XGBoost, another gradient boosting model that excels at producing well-calibrated class probabilities. Both CatBoost and XGBoost consistently delivered accuracies 3x higher than random guessing (20%), reinforcing their reliability. The combination of structured inputs and effective binning was instrumental in reaching these results.

Conclusion of Overall Insights

While higher floor above a certain range does lead to higher pricings, having a flat on a lower floor does not mean the HDB flat will be undervalued. Lease decay does not play that strong as a factor as the public perceives; remaining lease of the flat has a weak predictive power for resale prices. Our models prove that pricing is affected by multiple factors, and not solely on one of them. Based on our findings, K-Nearest Neighbors (KNN) for regression yields the most accurate predictions for resale prices. For classification, both CatBoost and XGBoost models perform with comparable accuracy.

We hope that these insights can contribute to a more data-driven understanding of the HDB resale market and help buyers and sellers make more informed decisions. By highlighting the multi-factor nature of pricing, we aim to encourage a more nuanced conversation around property valuation. Ultimately, we believe our findings can support fairer pricing expectations and reduce misconceptions in the public discourse.