

**ANL252**

**PYTHON FOR DATA ANALYTICS**

# **Group-Based Assignment**

**July 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Lau Wei Lian** | M2273019 |
| **Lim Kai Wen** | N2270422 |
| **Ng Yi Jian** | E2211293 |
| **Shafina Binte Hazman** | H2311509 |

**Tutorial Group: T03**

**Instructor’s Name: Mr. Munish Kumar**

**Submission Date: 13/10/2023**

**Declaration Page**

We, members of group \_\_\_7\_\_\_, do hereby declare that we each contributed to this assignment and that we collectively agree to a shared grade.

|  |  |  |
| --- | --- | --- |
| Name | Contribution | Signature |
| **Lau Wei Lian** | I did questions 1a & 1b | A black text on a white background  Description automatically generated |
| **Lim Kai Wen** | I did 1d |  |
| **Ng Yi Jian** | I did 1d |  |
| **Shafina Binte Hazman** | I did 1c & 1d |  |

Table of Contents

[Question 1(a) 4](#_Toc148085510)

[Question 1(b) 5](#_Toc148085511)

[Question 1(c) 6](#_Toc148085512)

[Question 1(d) 7](#_Toc148085513)

[References 15](#_Toc148085514)

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Question 1(a)

We should use Pandas as the library to read the dataset with Python.

We could analyse the data quickly with the pre-set functions within the Pandas library.

import pandas as pd

*#importing of .csv file into pandas*

resale = pd.read\_csv('GBA\_HDB.csv')

Once the data is imported successfully into pandas, we can then use the pre-set functions within pandas to identify the dimension of the dataset. There are multiple functions to determine the dimensions of the data set. We would use the .shape function to get the necessary information quickly.

*#identifying the dimension of the data*

print (resale.shape)

The displayed information would be (1250, 11), interpreted as the dimension of the data set as 1250 columns and 11 rows.

[Word count - 117 words ]

# 

# **Question 1(b)**

To identify any missing variables, we would first identify the location of the missing data.

resale.isnull().sum(axis=0)

The output would show that the missing value under the column as follows,

month 0

town 0

flat\_type 40

block 0

street\_name 1

storey\_range 0

floor\_area\_sqm 0

flat\_model 0

lease\_commence\_date 0

remaining\_lease 0

resale\_price 134

dtype: int64

We can also use the below codes to filter the missing values more quickly.

missing\_values = resale.isnull().sum()

columns\_with\_missing = missing\_values[missing\_values > 0]

print(columns\_with\_missing)

flat\_type - 40

street\_name - 1

resales\_price - 134

For statistical functions within pandas, pandas would ignore any missing value in the dataset.

Such would reduce the size of the dataset, affecting any findings from the data and creating a biased result.

Data quality is also affected, as should there be computation on average; the missing data will skew the overall results.

[Word count - 138 words]

# Question 1(c)

Ways to treat missing data in Python:

Removing rows - By using df.dropna(), we can remove rows with missing values. It is a suitable approach when there are a few and random missing values. This ensures that only complete cases are used in the analysis, maintaining data integrity.

Mean or Median Imputation - Replace missing values with the mean or median of the variable using df.fillna(df.mean()) or df.fillna(df.median()). This approach retains the data structure and minimises the impact on central tendencies.

Mode Imputation - For categorical variables, replace missing values with the mode using df.fillna(df.mode().iloc[0]). It preserves the most common category, maintaining the distribution.

Constant Imputation - Replace missing values with a predefined constant using ‘df.fillna(value). This approach is suitable when we have domain knowledge indicating a specific value for missing data.

Predictive Modelling - We can predict missing values using machine learning algorithms. Missing values can be estimated using the relationships in the data to provide more accurate replacements.

Data Transformation - Transform categorical variables into numerical ones to handle missing values. This is useful when the nature of the missing data is related to the data type.

Method selection depends on the dataset, the nature of missing data, and the impact of missing data. A careful consideration of each method is essential to maintain data quality and accuracy.

[Word count - 228 words]

# Question 1(d)

**Flat\_type**

We use "fillna()" to fill in the blank values in a column to correct the missing data in column "flat type". The "pad" approach was employed, which fills in the missing values in the column with the most recent non-missing value. When "inplace = true," the modifications will be made immediately to the DataFrame.

*#Filling the flat type based on the previous data flat type*

resale["flat\_type"].fillna(method= 'pad',inplace= True)

resale

**Resale\_Price**

We will determine the mean price for each flat type and insert the mean number into the empty entries to fill in the missing data in the "resale price" column.

We use Boolean masking to filter out the entire 3-room apartment. From there, we will determine the average cost of the three-room apartment. After that, we will filter out the 3-room apartment without pricing and create a new data frame by using "fillna" to fill in the missing values with the mean price. The original data frame and the new data frame will then be combined.

*#finding mean price of 3-room flat*

three\_room = (resale["flat\_type"] == "3 ROOM")

no\_price\_threeroom = (resale["resale\_price"].isnull())

condition = (three\_room & ~ no\_price\_threeroom)

three\_with\_price = resale[condition]

three\_with\_price

three\_with\_price["resale\_price"].mean()

*#finding all 3-room flat with null value*

condition = three\_room& no\_price\_threeroom

three\_with\_noprice = resale[condition]

three\_with\_noprice

*#filling null values with mean price of 3-room flat*

three\_with\_noprice["resale\_price"] = three\_with\_noprice["resale\_price"].fillna(value=322243.0)

three\_with\_noprice

*#Combine both data frame*

newdf = pd.concat([resale,three\_with\_noprice])

newdf

The same would be done with all the other flat types. Once done, we will drop all the roles with missing values in the specific “resale\_price” column. Finally, we will sort the data back to its original state.

*#dropping rows that has NaN value in resale price*

newdf5 = newdf4.dropna(axis=0, how="any", subset=["resale\_price"])

*#Sorting town based on alphabetical order*

new\_resale = newdf5.sort\_values('town')

new\_resale

**Street\_name**

We will choose to ignore the empty value in street name as there is only one missing and it has minor impact on our analysis.

**Table 1.1: Frequency of Town**

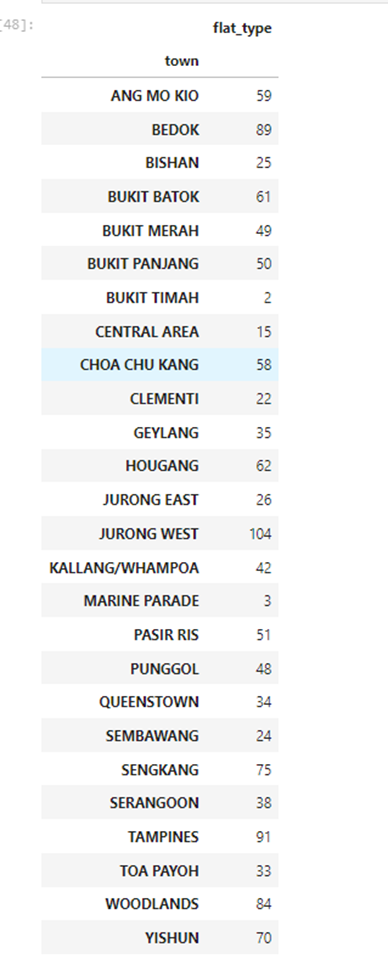
**

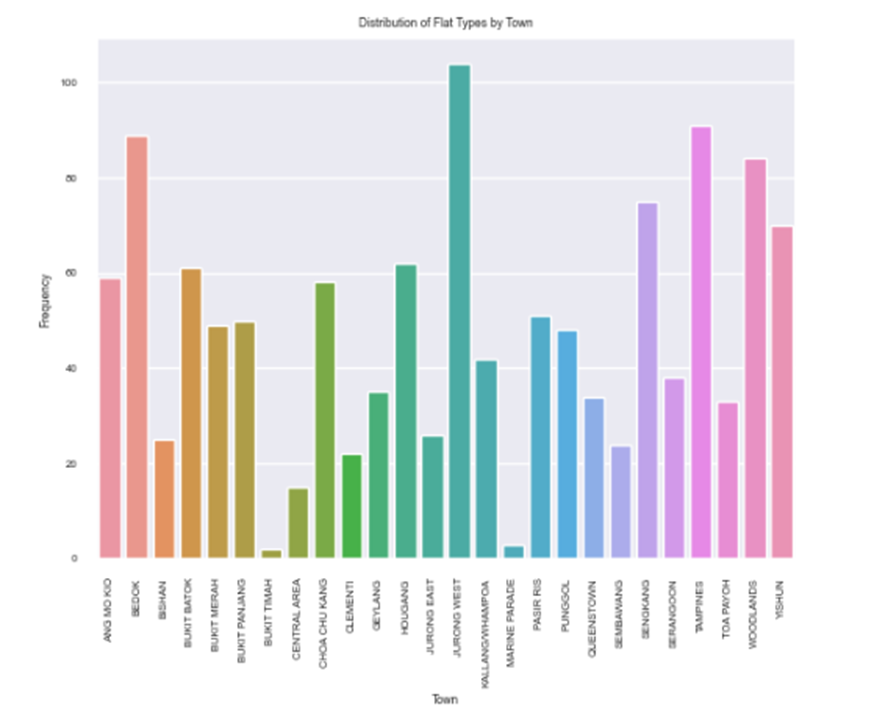
Table 1.1 shows the distribution of flat types for each town. This table was created with the following code below:

new\_resale.groupby('town')['flat\_type'].count()

town\_flat = pd.DataFrame(new\_resale.groupby('town')['flat\_type'].count())

town\_flat

**Graph 1.1: Bar chart of the distribution of flat types by town**



Graph 1.1 is then created using table 1.1 to visualize the frequency of data collected in accordance with the town. From graph 1.1, we can use the Jurong West has the highest frequency of data with 104 and the lowest frequency of data is Bukit Timah and Marine Parade with 2 and 3 respectively. This could mean that there is more housing estate in an Urban area like Jurong West as compared to rural areas like Bukit Timah. Another possibility could also be that areas like Bukit Timah and Marine Parade has more private estate than housing estate. The codes that created graph 1.1 is shown below:

import seaborn as sns

import matplotlib.pyplot as plt

sns.barplot(x=town\_flat.index, y=town\_flat.loc[:, "flat\_type"])

plt.xlabel("Town")

plt.ylabel("Frequency")

plt.title("Distribution of Flat Types by Town")

plt.xticks(rotation=90)

plt.show()

**Table 1.2: Resale prices by flat type**

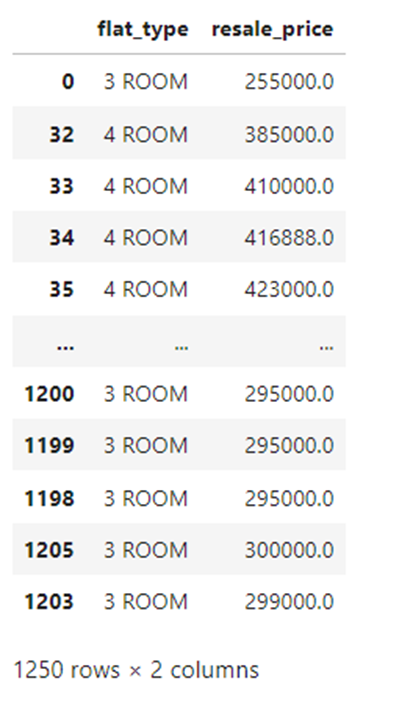
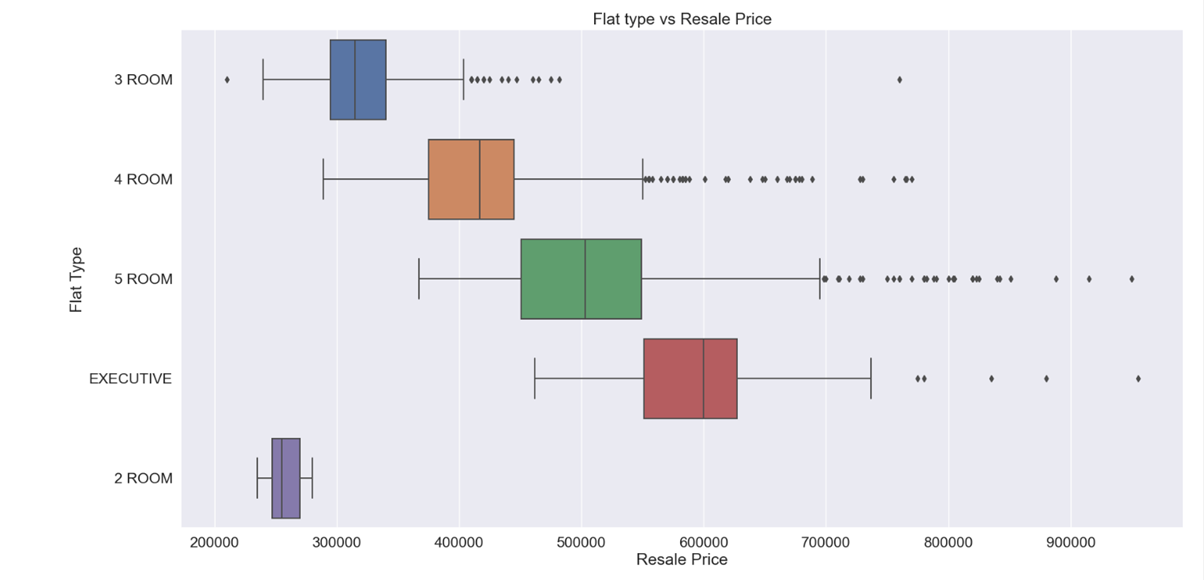


Table 1.2 shows the resale price for each of the data provided in the dataset as long as the flat type. This table was created with the following code below:

new\_resale[['flat\_type','resale\_price']]

**Graph 1.2: Box Plot of the flat types vs. Resale Price**

To examine the selling price of each type of flat, graph 1.2 is then made using table 1.2. The median price for a three-room, four-room, five-room, and executive home is shown in Graph 1.2 to be approximately $300,000, $400,000, $500,000, and $600,000. We may argue that the median price state is the best price point for anyone looking to purchase second-hand apartments to consider. The third quartile pricing point is another option. A person who is willing to spend the 3rd quartile price point for a 4-room flat, for instance, can also look at 5-room flats because the 2nd quartile price point for a 5-room flat is around the same price as the 3rd quartile price point for a 4-room flat. The maximum amount that should be spent on a particular flat is shown by the line at the end of the box plot. Anything that is beyond the norm is an outlier, and those apartments should not be taken into consideration by those looking for housing since they are too pricey. The codes that created graph 1.1 is shown below:

sns.set(font\_scale=1.5)

plt.figure(figsize=(20, 10))

sns.boxplot(data = new\_resale, x = "resale\_price", y = "flat\_type")

plt.xlabel('Resale Price')

plt.ylabel('Flat Type')

plt.title('Flat type vs Resale Price')

**Table 1.3: Scatter plot of the floor area and resale price**

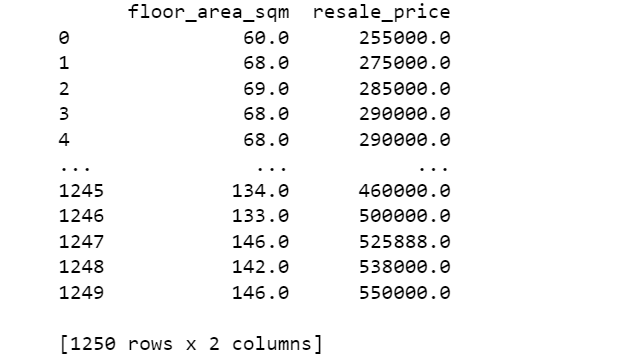


Table 1.3 shows the resale price based on the floor\_area sqm. The table was created using the codes below.

import pandas as pd

*#Read the data from the CSV file into a DataFrame*

df = pd.read\_csv('GBA\_HDB.csv')

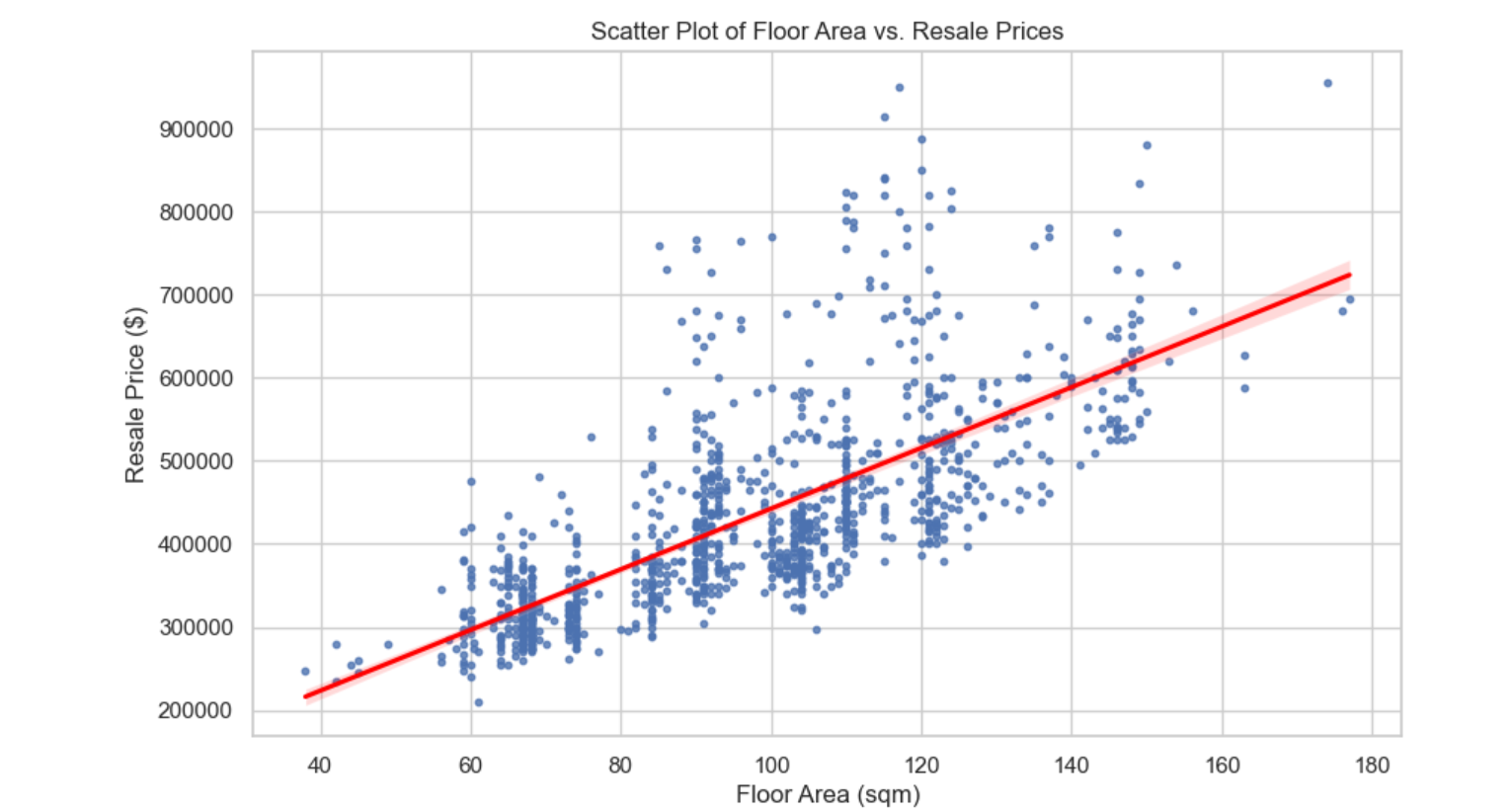
*#Select and display floor area and resale price as a table*

floorarea\_resaleprice = ['floor\_area\_sqm', 'resale\_price']

df\_selected = df[floorarea\_resaleprice]

print(df\_selected)

**Graph 1.3: Scatterplot of the floor area vs resale price**



From the graph 1.3, it shows that there is a strong positive correlation between the floor area and resale price. The data points on the scatter plot are closely clustered together rather than spread out randomly. This indicates that there is a clear relationship between the floor area and resale price. In addition, the data points and the trend line forms an upward sloping line. This also indicates that if floor area per square metre increases, the resale price of the HDB also increases which sounds reasonable as a bigger flat tends to be more expensive than a smaller flat. From the graph, it also shows that flats that are lesser than 80sqm generally falls below 300k to 400k range whereas flats that are between 80sqm and 120 sqm generally falls between 500 - 600k range. However, there are exceptions where some of the flats that are between 100sqm and 120sqm can cost falls in the 800K range and can compete with flat that have a larger square metre One possible reason can be the flats are generally in a popular mature estate. Graph 1.3 is created using the code below.

*#import packages such as panada seaborn and matplotlib*

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

*#use of panda to read csv file*

df = pd.read\_csv("./GBA\_HDB.csv")

*#create the figure size of the graph*

plt.figure(figsize=(10, 6))

*#Create a scatter plot with a custom color for the trend line*

sns.regplot(data=df, x='floor\_area\_sqm', y='resale\_price', scatter\_kws={"s": 10}, line\_kws={"color": "red"})

*#Adding of title and labels*

plt.title('Scatter Plot of Floor Area vs. Resale Prices')

plt.xlabel('Floor Area (sqm)')

plt.ylabel('Resale Price ($)')

*#Display scatterplot*

plt.show()

[Word Count - 360 words]

# References

Code reference for Scatterplot

*Scatter plot in Seaborn*. PYTHON CHARTS | The definitive Python data visualization site. (2022, October 10). https://python-charts.com/correlation/scatter-plot-seaborn/