

**ANL252**

**Python for Data Analytics**



**Group-based Assignment**

**July 2023 Presentation**



**Submitted by:**

**Group 9**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Chua Yong Hua** | **H2070924** |
| **Ailn Aik Xiu Jing** | **N2070416** |
| **Ng Xue Er, Ariel** | **J2271307** |

**Tutorial Group: T03**

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

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

**Declaration Page**

We, members of group **9** , do hereby declare that we each contributed to this assignment and that we collectively agree to a shared grade.

|  |  |  |
| --- | --- | --- |
| Name | Contribution | Signature |
| Chua Yong Hua (Team Lead) | I did question 1 (a) (b) (c) |  |
| Ng Xue Er, Ariel | I did question (d) |  |
| Ailn Aik Xiu Jing | I did question (d) |  |

**Question 1(a):**

To read the dataset with Python, we can use different methods depending on the format and structure of dataset. For example, we can use NumPy library to work on numerical data as NumPy is capable of providing multi-dimensional arrays. We can use Pandas library to work on tabular data in a comma-separated values (CSV) file as Pandas library is capable of reading it into a DataFrame object using the pd.read\_CSV() (Sanchhaya Education Private Limited, 2023).

To identify the dimensions (numbers of rows and columns) with Python, we can use different attributes functions depending on the object provided in the dataset. For instance, we can use the .ndim attribute to get the number of dimensions if the data is in a NumPy array. We can use ‘df.shape’ to obtain the dimensions of a DataFrame.

Since the dataset given in this assignment is in a CSV file, it is more appropriate to use Pandas library to read the dataset. Here are the steps:

1. Import Pandas library

2. Load the dataset

3. Explore the dimensions of the dataset

4. The results (1250, 11) tell us that there are 1250 rows and 11 columns in the dataset.

```python

import pandas as pd

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

#To get the number of rows and columns

num\_rows, num\_columns = df.shape

print (df.shape)

```

(1250, 11)

```python

Total word counts: 194.

**Question 1(b):**

The variables with missing values are as follows:

Flat type : 40

Street name : 1

Resale price : 134

```python

import pandas as pd

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

#check for missing value in each column

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

#filter column with missing values

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

print(columns\_with\_missing\_values)

```

flat\_type 40

street\_name 1

resale\_price 134

dtype: int64

```python

Handling missing values are essential for several reasons:

1. **Data integrity**: missing data can lead to inaccurate analysis of dataset causing bias or errors in the results. Proper handling of missing data will reduce the risk of drawing incorrect insights.
2. **Statistical value**: missing data can affect the validity of tests and statistical measures and leading to inaccurate conclusions. By simply removing the missing data may distorts the statistical measures.
3. **Visualization**: missing data can affect the creation of visualisation and produce meaningless charts or diagrams. The misleading interpretations could lead to a wrong decision making.
4. **Model performance**: machine learning algorithms may not be capable in handling the missing data and therefore causing errors in model or suboptimal performance. Handling missing data is crucial to ensure data used in the most appropriate way to make accurate predictions.

In summary, handling missing data is important in the data preprocessing pipeline to ensure the reliability and accuracy of the analysis.

Total word counts: 179.

**Question 1(c):**

Treating missing data is an important step in preprocessing before performing any analysis in Python. There are different techniques to process the missing data depending on the nature and reason of the missing values.

The dataset in this assignment has three columns with missing values. Here are some ways to treat the missing values:

1. **‘flat\_type’**: there are 40 missing values in this column. Hence, it is more appropriate to use the imputation with ‘mode’ (most frequent value) since the missing values are categorical data. This method is suitable when there are missing values at random and doesn’t contain any significant information.

2. **‘street\_name’**: there is only 1 missing value in this column. We may delete the entire row since it is not significant to the findings. Alternatively, we can also impute it with ‘mode’ since the missing value is categorical data.

3. **‘resale\_price’**: we may use the imputation with ‘mean’ or ‘median’ to treat the 134 missing numerical values in this column. However, there are outliers in the dataset. Hence, it is more appropriate to use ‘median’ since it is not influenced by extreme values.

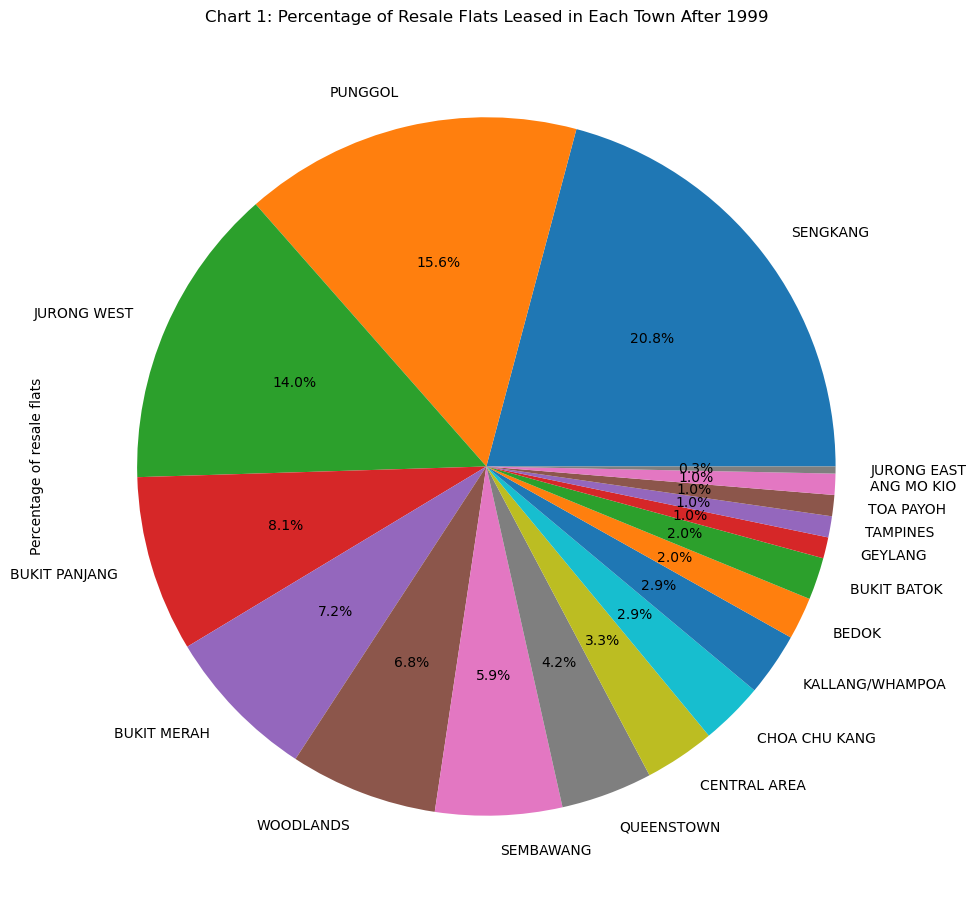
After treating the missing data as described above, the DataFrame has no missing values.

```python  
 import pandas as pd  
  
 df = pd.read\_csv('GBA\_HDB.csv')  
  
 #impute the missing values with the mode of the 'flat\_type' column  
 mode\_flat\_type = df['flat\_type'].mode()[0]  
 df['flat\_type'].fillna(mode\_flat\_type, inplace=True)  
  
 # Drop rows with missing values in the 'street\_name' column  
 df.dropna(subset=['street\_name'], inplace=True)  
  
 #impute the missing values with the median of the 'resale\_price' column  
 median\_resale\_price = df['resale\_price'].median()  
 df['resale\_price'].fillna(median\_resale\_price, inplace=True)  
  
 missing\_values=df.isnull().sum()  
  
 columns\_with\_missing\_values=missing\_values[missing\_values>0]  
  
 print(columns\_with\_missing\_values)

```  
 Series([], dtype: int64)  
  
 ```python

Total word counts: 200.

**Question 1 (d):**



*Chart 1 - Percentage of Resale Flats Leased in Each Town After 1999*

|  |  |  |
| --- | --- | --- |
|  | **town** | **number\_of\_flats** |
| **0** | SENGKANG | 64 |
| **1** | PUNGGOL | 48 |
| **2** | JURONG WEST | 43 |
| **3** | BUKIT PANJANG | 25 |
| **4** | BUKIT MERAH | 22 |
| **5** | WOODLANDS | 21 |
| **6** | SEMBAWANG | 18 |
| **7** | QUEENSTOWN | 13 |
| **8** | CENTRAL AREA | 10 |
| **9** | CHOA CHU KANG | 9 |
| **10** | KALLANG/WHAMPOA | 9 |
| **11** | BEDOK | 6 |
| **12** | BUKIT BATOK | 6 |
| **13** | GEYLANG | 3 |
| **14** | TAMPINES | 3 |
| **15** | TOA PAYOH | 3 |
| **16** | ANG MO KIO | 3 |
| **17** | JURONG EAST | 1 |

***Table 1: Number of flats leased in each town after 1999***

#Chart 1: Percentage of Resale Flats Leased in Each Town After 1999

#Import pandas

import pandas as pd

#Convert GBA\_HDB.csv to pandas DataFrame

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

#Display hdb table for checking

hdb

#Call columns 'town' and 'lease\_commence\_date'

hdb\_towns = hdb[["town", "lease\_commence\_date"]]

#Display table with only 'town' and 'lease\_commence\_date' columns

hdb\_towns

#Remove rows with null values in any column

hdb\_towns.dropna(axis = 0, how = "any")

#Display table without rows with null values

hdb\_towns

#Select and filter flats with lease\_commence\_date that after 1999

hdb\_towns['lease\_commence\_date']>1999

#Boolean masking

hdb\_2000towns = hdb\_towns[hdb\_towns['lease\_commence\_date']>1999]

#Display filtered table

hdb\_2000towns

#Count the number of flats built in each town

town\_counts = hdb\_2000towns['town'].value\_counts()

#Convert series into a table

town\_counts = town\_counts.reset\_index()

town\_counts.columns = ['town', 'number\_of\_flats']

#Display table for plotting

town\_counts

#Import matplotlib

import matplotlib.pyplot as plt

#Plot pie chart based on town\_counts

plt.figure(figsize=(10,100))

#Rename 'number\_of\_flats' to 'Percentage of resale flats'

town\_counts.rename(columns={'number\_of\_flats': 'Percentage of resale flats'}, inplace=True)

#Define x and y

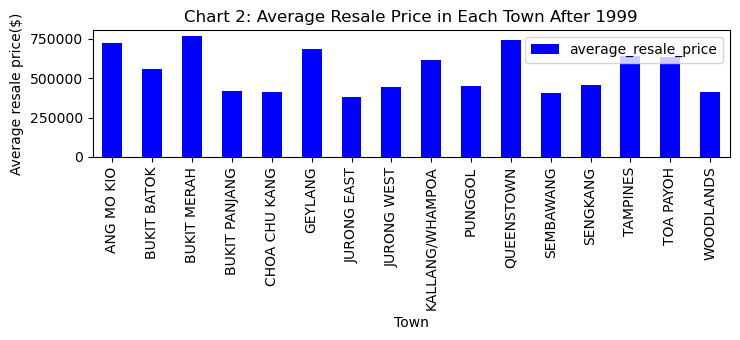
town\_counts['Percentage of resale flats'].plot(kind='pie', labels=town\_counts['town'], autopct='%1.1f%%')

#Label pie chart

plt.title('Chart 1: Percentage of Resale Flats Leased in Each Town After 1999')

#Display pie chart

plt.show()



*Chart 2 - Average Resale Price in Each Town After 1999*

|  |  |  |
| --- | --- | --- |
|  | **town** | **average\_resale\_price** |
| **0** | ANG MO KIO | 725296.000000 |
| **1** | BUKIT BATOK | 557814.666667 |
| **2** | BUKIT MERAH | 765361.000000 |
| **3** | BUKIT PANJANG | 417987.555556 |
| **4** | CHOA CHU KANG | 412654.222222 |
| **5** | GEYLANG | 684333.333333 |
| **6** | JURONG EAST | 382000.000000 |
| **7** | JURONG WEST | 443160.162791 |
| **8** | KALLANG/WHAMPOA | 613666.666667 |
| **9** | PUNGGOL | 451834.500000 |
| **10** | QUEENSTOWN | 740138.461538 |
| **11** | SEMBAWANG | 407765.333333 |
| **12** | SENGKANG | 457586.263750 |
| **13** | TAMPINES | 642000.000000 |
| **14** | TOA PAYOH | 632666.666667 |
| **15** | WOODLANDS | 413084.571429 |

***Table 2: Average Resale Price in Each Town after 1999***

#Chart 2: Average Resale Price in Each Town After 1999

#Import pandas

import pandas as pd

#Convert GBA\_HDB.csv to pandas DataFrame

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

#Display hdb table for checking

hdb

#Call columns 'town', 'lease\_commence\_date and 'resale\_price'

hdb\_resale = hdb[["town", "lease\_commence\_date", "resale\_price"]]

#Display table with only 'flat\_type' and 'resale\_price' columns

hdb\_resale

#Remove rows with null values in any column

hdb\_resale = hdb\_resale.dropna(axis = 0, how = "any")

#Display table without rows with null values

hdb\_resale

#Select and filter flats with lease\_commence\_date that after 1999

hdb\_resale['lease\_commence\_date']>1999

#Boolean masking

hdb\_resale = hdb\_resale[hdb\_resale['lease\_commence\_date']>1999]

#Display filtered table

hdb\_resale

#Group hdb\_prices by flat\_type

hdb\_resale\_group = hdb\_resale.groupby(by = ['town'])

#Calculate the mean resale price of each flat type group

hdb\_resale\_mean = hdb\_resale\_group['resale\_price'].mean()

#Display result

print(hdb\_resale\_mean)

#Convert series into a table

hdb\_resale\_mean = hdb\_resale\_mean.reset\_index()

hdb\_resale\_mean.columns = ['town', 'average\_resale\_price']

#Display table for plotting

hdb\_resale\_mean

#Import matplotlib

import matplotlib.pyplot as plt

#Plot bar chart

hdb\_resale\_mean.plot(x='town', y='average\_resale\_price', kind='bar', color='blue')

#Label bar chart title

plt.title('Chart 2: Average Resale Price in Each Town After 1999')

#Label x

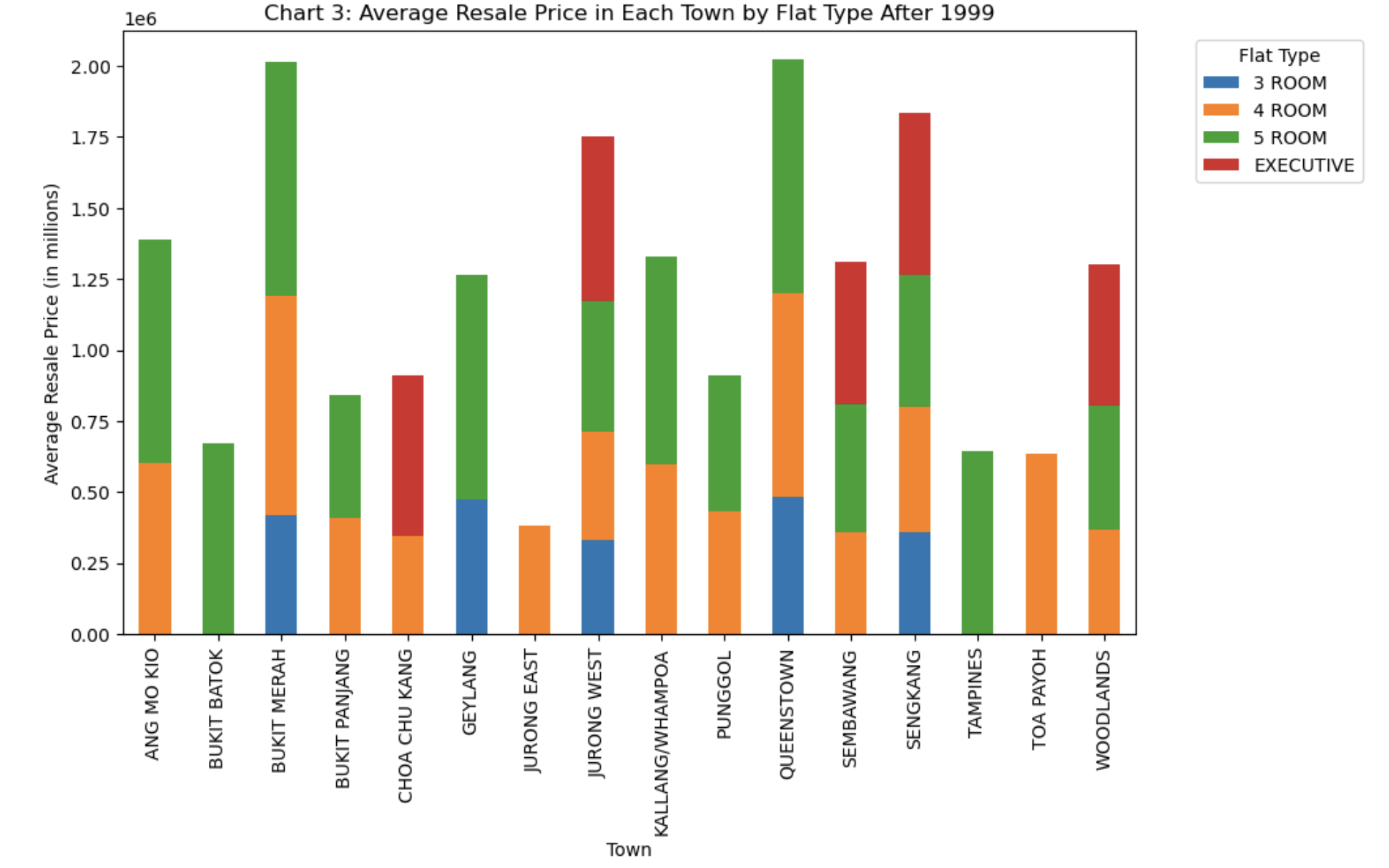
plt.xlabel('Town')

#Label y

plt.ylabel('Average resale price($)')

#Display bar chart

plt.show()



*Chart 3 - Average Resale Price in Each Town by Flat Type After 1999*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Flat Type /**  **Town** | **3 ROOM** | **4 ROOM** | **5 ROOM** | **EXECUTIVE** |
| ANG MO KIO | 0.00 | 600888.00 | 787500.00 | 0.00 |
| BUKIT BATOK | 0.00 | 0.00 | 669629.33 | 0.00 |
| BUKIT MERAH | 420000.00 | 770000.00 | 822148.00 | 0.00 |
| BUKIT PANJANG | 0.00 | 410500.00 | 432962.67 | 0.00 |
| CHOA CHU KANG | 0.00 | 344377.60 | 0.00 | 564000.00 |
| GEYLANG | 475000.00 | 0.00 | 789000.00 | 0.00 |
| JURONG EAST | 0.00 | 382000.00 | 0.00 | 0.00 |
| JURONG WEST | 330000.00 | 384999.92 | 455690.78 | 582000.00 |
| KALLANG / WHAMPOA | 0.00 | 599125.00 | 730000.00 | 0.00 |
| PUNGGOL | 0.00 | 433577.43 | 477394.40 | 0.00 |
| QUEENSTOWN | 482000.00 | 716000.00 | 825560.00 | 0.00 |
| SEMBAWANG | 0.00 | 357977.60 | 452000.00 | 500000.00 |
| SENGKANG | 360666.67 | 438699.61 | 465659.70 | 571333.33 |
| TAMPINES | 0.00 | 0.00 | 642000.00 | 0.00 |
| TOA PAYOH | 0.00 | 632666.67 | 0.00 | 0.00 |
| WOODLANDS | 0.00 | 367361.00 | 436324.00 | 500000.00 |

***Table 3: Pivot table with Average Resale Price in Each Town by Flat Type after 1999***

#Chart 3: Average Resale Price in Each Town by Flat Type After 1999

import pandas as pd

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

import matplotlib.pyplot as plt

hdb\_resale = df[["town", "flat\_type", "lease\_commence\_date", "resale\_price"]]

#Remove rows with null values

hdb\_resale = hdb\_resale.dropna(axis = 0, how = "any")

#Filter flats with lease\_commence\_date after 1999

hdb\_resale = hdb\_resale[hdb\_resale['lease\_commence\_date']>1999]

#Group hdb resale price by town & flat\_type and calculate average resale price

hdb\_flat\_grp = hdb\_resale.groupby(by = ["town","flat\_type"])

avg\_resale = hdb\_flat\_grp['resale\_price'].mean()

#Round the resale price to 2dp

avg\_resale = avg\_resale.round(2)

#Convert series into table

avg\_resale = avg\_resale.reset\_index()

avg\_resale.columns = ['Town','Flat Type', 'Average Resale Price']

# Pivot dataset to prepare for stacked bar chart and display

pivot\_avg\_resale = avg\_resale.pivot(index='Town', columns='Flat Type', values='Average Resale Price').fillna(0)

print(pivot\_avg\_resale)

#Plot the stacked bar chart

ax = pivot\_avg\_resale.plot(kind='bar', stacked=True, figsize=(10, 6))

# Customise the chart's label, title and legend

plt.xlabel('Town')

plt.ylabel('Average Resale Price (in millions)')

plt.title('Chart 3: Average Resale Price in Each Town by Flat Type After 1999')

plt.legend(title='Flat Type', bbox\_to\_anchor=(1.05, 1), loc='upper left')

#Display stacked bar chart

plt.show()

Chart 1: A pie chart was used to display the proportion of resale houses by town, with the size of each segment representing the percentage of resale houses in town in the 2000s. The chart highlights the towns that have the most resale houses with recent leasing commencement in the 2000s. With reference to the chart, we can infer that towns such as SENGKANG, PUNGGOL and JURONG WEST have the most resale houses in the market as compared to the towns like JURONG EAST, ANG MO KIO, TOA PAYOH, TAMPINES and GEYLANG which have a relatively lower number of resale houses from that period.

Chart 2: A vertical bar chart that illustrates the average resale prices for HDB flats in each town, regardless of the types of flat. It highlights the most and least expensive town for purchasing resale HDB flats with leases commencing in the 2000s. With reference to the chart, we can infer that the towns like ANG MO KIO, BUKIT MERAH and QUEENSTOWN are among the costlier options for buyers, whereas JURONG EAST is least expensive.

Chart 3: A stacked bar chart was used to visualise the composition of average resale prices for each town, whereby each segment within the bar represents the average resale price for each flat type. The chart facilitates the comparison of average resale prices for different flat types across town. With reference to the chart, we can infer that 4-room and 5-room flat types are the most commonly available flat sizes in the towns.

Although chart 1 highlights that SENGKANG, PUNGGOL and JURONG WEST have the most resale houses on the market in the 2000s, chart 2 shows that resale prices in these estate are relatively cheaper than the estates with lesser newer resale flats such as ANG MO KIO.

This could be because estates with lesser newer resale flats are mature towns with more amenities as compared to developing estates like SENGKANG, which contributes to the differences in average resale prices of the HDB flats. Buyers can expect to pay more for resale flats in mature estates if they are looking to move into towns with more convenient amenities.

A breakdown of HDB resale prices according to the flat type in each estate as illustrated in chart 3, highlights that 4 to 5 room flats in mature estates like QUEENSTOWN, BUKIT MERAH and ANG MO KIO costs almost twice as much as 4 to 5 room flats in younger estates such as SENGKANG, SEMBAWANG and BUKIT PANJANG. Since 4 to 5 rooms flats are the most common type of resale HDB that are available on the market, prospective buyers can expect their chances of purchasing 4 to 5 rooms flats to be higher than other HDB flat types, with leases commencing in the 2000s. If buyers are looking for smaller flat types like 3 room flats or bigger executive flat type with leases commencing in the 2000s, they will have limited purchase choices in just 5 towns according to chart 3.

Total word counts: 500

**References**

Sanchhaya Education Private Limited. (2023, July 24). *Different between Pandas vs NumPy.* Retrieved from:<https://www.geeksforgeeks.org/difference-between-pandas-vs-numpy/>

Saturn Cloud. (2023, July 10). *Creating Pie Charts with Pandas DataFrame: A Guide | Saturn Cloud Blog*. Saturncloud.io. Retrieved from: <https://saturncloud.io/blog/creating-pie-charts-with-pandas-dataframe-a-comprehensive-guide/#:~:text=We%20plot%20a%20pie%20chart,distribution%20in%20the%20pie%20chart>.

w3resource (2022, August 19). *Pandas DataFrame: Pivot() function*. Retrieved from: <https://www.w3resource.com/pandas/dataframe/dataframe-pivot.php>