NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CE/CZ4123/SC4023 BIG DATA MANAGEMENT

SEMESTER GROUP PROJECT

Name	Matriculation Number
HENDY	U2122559J
GWEE JIA XIANG	U2122287H
GOH PENG AIK	U2022363E

Table Of Contents

Data Storage: Column-Oriented

1. Column Store Implementation

Column store approach was implemented through the ColumnStore class that takes in pandas dataframe obtained from input data source from CSV file located in the directory "Data/ResalePricesSingapore.csv", the program would then iterate over the dataframe columns, extracting each attribute (column) and storing it individually into separate .store files on the disk. Refer to Figure 1 in the appendix.

2. Data Column Design for Efficient Processing

This approach enables efficient storage and retrieval of data that allows direct and selective access to specific attributes by reading the necessary .store file instead of performing full table scans during the query process. The program also allows parallel access to different data attributes, increasing the speed during query execution.

3. Reading and Writing Input/Output Files

The load_data_from_disk(file_path) method allows the reading of input files from the disk and It employs caching using functools.lru_cache(maxsize=None) decorator, which stores previously read data in memory to minimize disk access, this method helps avoid redundant reads, enhancing efficiency in data retrieval. By utilizing caching, the program reduces the overhead associated with repetitive file reads, leading to faster data retrieval.

The save_to_files(directory) method allows the writing of output files to the disk. It organises and stores each column of the dataset in separate files within the specified directory. If the existing attribute files exist, it will overwrite it with the new file.

4. Data anomalies and exception handling

In addressing exceptions like empty qualified entries, the program demonstrated a robust approach to handling data anomalies by making use of a try-catch algorithm during the storing and loading of the data to make it easier to troubleshoot. The program also replaces found empty entries in the table and replace it with “@#NULL@#” and highlights to the user the existence of empty entries. Figure 2 in the appendix shows examples of empty entries in 3 columns “floor_area_sqm”, “flat_model”, and “lease_commence_date”, those entries are replaced and the user will be notified.

The program also checks for the existence of incorrect data format among the data entries of attributes and highlights it to the user. Figure 2 in the appendix shows different data formats in the last column of “resale_price” which have the value of the string, this value will still be stored to ensure the same number of items is the same across all the attributes. however, the user will be notified.

The solution implemented to address empty qualified entries and incorrect data format could result in errors in query computation if those specific entries are selected for computation for the result. However, this solution ensures the correctness of column storage and enables users to easily track the issues in the data entry.

Data Processing: Scanning Process

Firstly, the last 3 numerical digits of the student matriculation number are mapped to the town, month, and year respectively. After mapping, the month and year are appended such that they fit the data format in the month column (e.g. 2014-10). The program will proceed to scan the month column for data with the desired date value, saving the indices into a variable called matching_indices. For each value in matching_indices, the program scans the town column to filter for indices that match the desired town before saving the indices into matching_indices again.

Next, for each index in matching_indices, the program will scan the resale_price column and read each value associated with the index. For each value, the program will compare them to save the minimum price under the variable min_price as well as store the sum of all the values under total_price. Each individual value is also saved under the matching_price_data variable. This step is repeated for the floor_area_sqm column, returning min_area, total_area and matching_area_data. To obtain avg_price and avg_area, total_price and total_area are divided using the length of matching_indices which is obtained via *len(matching_indices)*. Standard Deviation is obtained using the stdev function from the statistics module in Python on matching_price_data or matching_area_data. All the results are rounded to 2 decimal points.

Improving the Efficiency in Column Scans

As the data was already sorted based on the date in the month column and the initial scanning method was done using indices, we did not implement any enhancement with index and sorting. Instead, we implemented Zone Mapping and Shared Scans for enhancements.

1. Zone Mapping

After the program has stored all the values in the database, it calls for the zone mapping function which splits the month column into zones based on the year. (e.g. Zone 1: 2014; min1: 1; max1:16096). Instead of scanning the entire month column for values that fit the query, the program uses the year retrieved from the mapping to get the starting and ending indices of the Zone that the year is associated with. This reduces the number of indices that need to be scanned by the program (Refer to Figure 3 in the appendix).

2. Shared Scanning

The program originally scanned through either the floor_area_sqm and resale_price column three times, once for each query (Min, Avg, S.D). Using Shared Scanning, the program would instead scan through the columns and store all the required variables in a single scan, reducing the time taken (Refer to Figure 4 in the appendix).

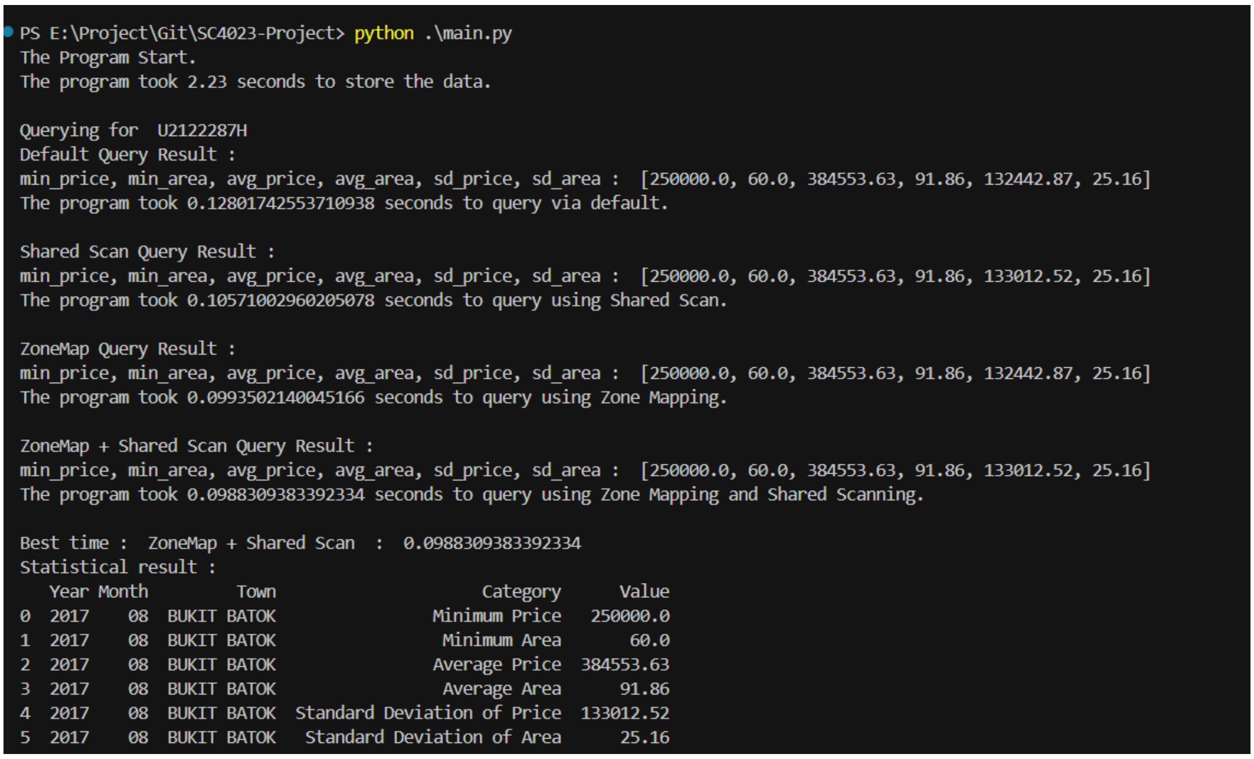
3. Zone Mapping + Shared Scanning

As Zone Mapping and Shared Scanning enhance different parts of the program, the final enhancement combines both methods together to reduce the time taken for column scans by reducing I/O operations, enhancing data locality, and enabling parallel processing (Refer to Figure 5 in the appendix).

Experiment Result

In our solution, query function and result construction are stored in "Output.py" and program flow in "main.py". The program starts by running the Python file "main.py" and it will perform column storage first. It would then perform queries using: Default scan, Shared scan, Zone Map, Zone Map + Shared scan.

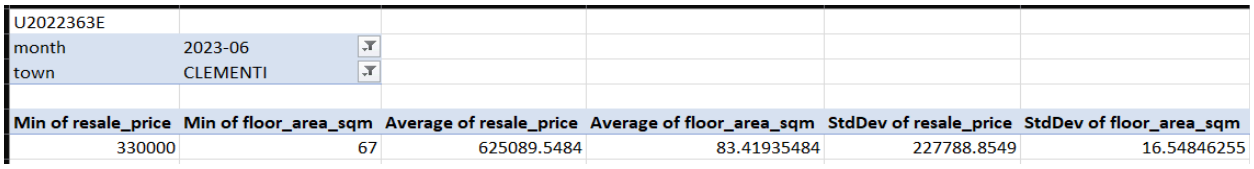
Here is a screenshot of the program execution on one of the IDs using our Python file.



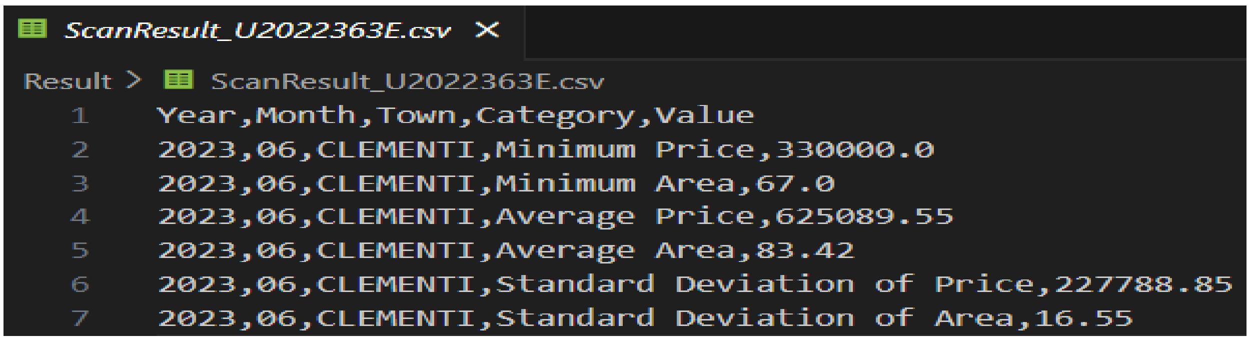
```
PS E:\Project\Git\SC4023-Project> python .\main.py
The Program Start.
The program took 2.23 seconds to store the data.

Querying for  U2122287H
Default Query Result :
min_price, min_area, avg_price, avg_area, sd_price, sd_area :  [250000.0, 60.0, 384553.63, 91.86, 132442.87, 25.16]
The program took 0.12801742553710938 seconds to query via default.

Shared Scan Query Result :
min_price, min_area, avg_price, avg_area, sd_price, sd_area :  [250000.0, 60.0, 384553.63, 91.86, 133012.52, 25.16]
The program took 0.10571002960205078 seconds to query using Shared Scan.

ZoneMap Query Result :
min_price, min_area, avg_price, avg_area, sd_price, sd_area :  [250000.0, 60.0, 384553.63, 91.86, 132442.87, 25.16]
The program took 0.0993502140045166 seconds to query using Zone Mapping.

ZoneMap + Shared Scan Query Result :
min_price, min_area, avg_price, avg_area, sd_price, sd_area :  [250000.0, 60.0, 384553.63, 91.86, 133012.52, 25.16]
The program took 0.0988309383392334 seconds to query using Zone Mapping and Shared Scanning.

Best time :  ZoneMap + Shared Scan  :  0.0988309383392334
Statistical result :
   Year Month        Town                    Category      Value
0  2017    08  BUKIT BATOK             Minimum Price   250000.0
1  2017    08  BUKIT BATOK              Minimum Area       60.0
2  2017    08  BUKIT BATOK             Average Price  384553.63
3  2017    08  BUKIT BATOK              Average Area      91.86
4  2017    08  BUKIT BATOK  Standard Deviation of Price  133012.52
5  2017    08  BUKIT BATOK   Standard Deviation of Area      25.16
```

These are screenshots of the output files and comparisons with Excel pivot table results.

U2022363E: Town= "CLEMENTI" Month=6 Year=2023.



U2022363E					
month	2023-06				
town	CLEMENTI				
Min of resale_price	Min of floor_area_sqm	Average of resale_price	Average of floor_area_sqm	StdDev of resale_price	StdDev of floor_area_sqm
330000	67	625089.5484	83.41935484	227788.8549	16.54846255

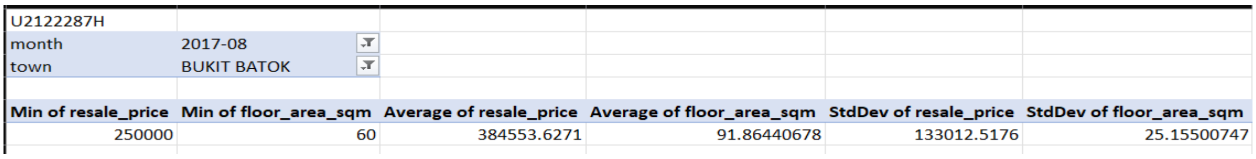


ScanResult_U2022363E.csv

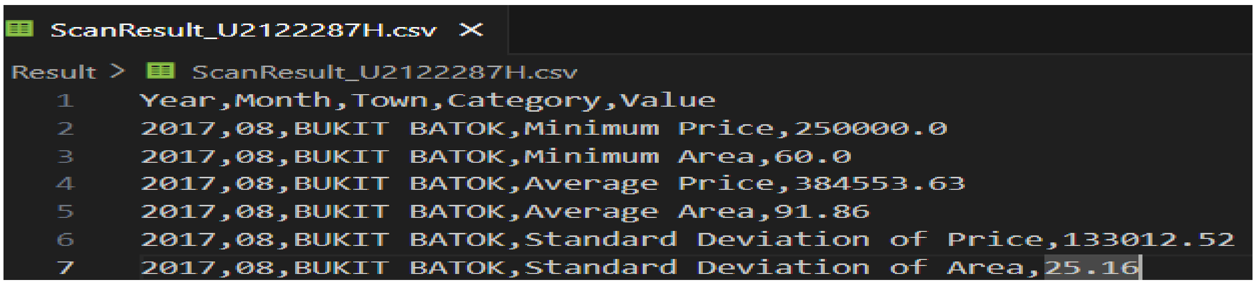
Result > ScanResult_U2022363E.csv

```
Year,Month,Town,Category,Value
2023,06,CLEMENTI,Minimum Price,330000.0
2023,06,CLEMENTI,Minimum Area,67.0
2023,06,CLEMENTI,Average Price,625089.55
2023,06,CLEMENTI,Average Area,83.42
2023,06,CLEMENTI,Standard Deviation of Price,227788.85
2023,06,CLEMENTI,Standard Deviation of Area,16.55
```

U2122287H: Town= ”BUKIT BATOK” Month=8 Year=2017.

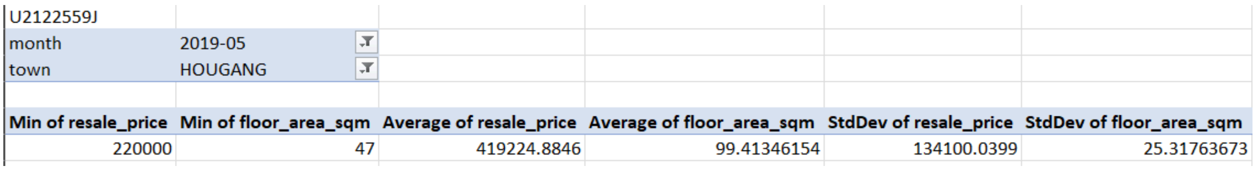


U2122287H					
month	2017-08				
town	BUKIT BATOK				
Min of resale_price	Min of floor_area_sqm	Average of resale_price	Average of floor_area_sqm	StdDev of resale_price	StdDev of floor_area_sqm
250000	60	384553.6271	91.86440678	133012.5176	25.15500747

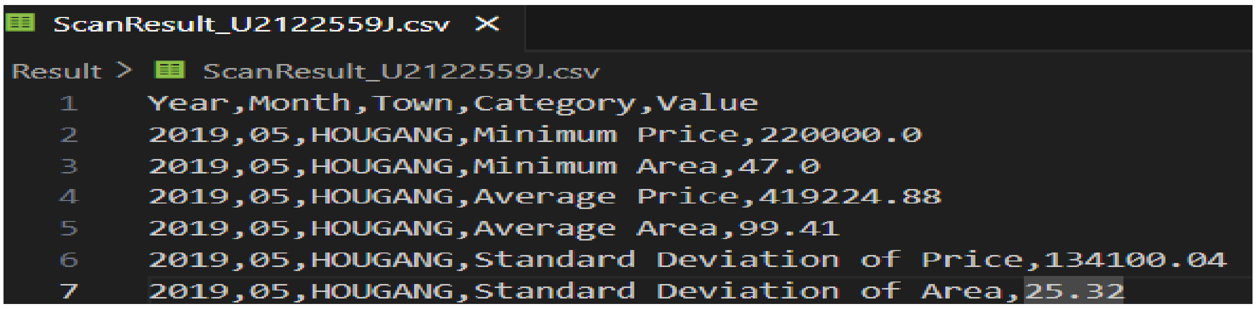


```
ScanResult_U2122287H.csv
Result > ScanResult_U2122287H.csv
Year,Month,Town,Category,Value
2017,08,BUKIT BATOK,Minimum Price,250000.0
2017,08,BUKIT BATOK,Minimum Area,60.0
2017,08,BUKIT BATOK,Average Price,384553.63
2017,08,BUKIT BATOK,Average Area,91.86
2017,08,BUKIT BATOK,Standard Deviation of Price,133012.52
2017,08,BUKIT BATOK,Standard Deviation of Area,25.16
```

U2122559J: Town= ”HOUGANG” Month=5 Year=2019.



U2122559J					
month	2019-05				
town	HOUGANG				
Min of resale_price	Min of floor_area_sqm	Average of resale_price	Average of floor_area_sqm	StdDev of resale_price	StdDev of floor_area_sqm
220000	47	419224.8846	99.41346154	134100.0399	25.31763673

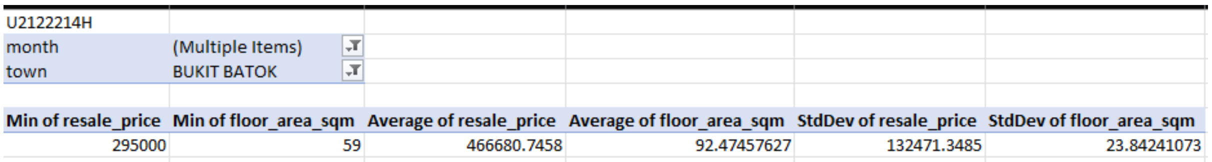


```
ScanResult_U2122559J.csv
Result > ScanResult_U2122559J.csv
Year,Month,Town,Category,Value
2019,05,HOUGANG,Minimum Price,220000.0
2019,05,HOUGANG,Minimum Area,47.0
2019,05,HOUGANG,Average Price,419224.88
2019,05,HOUGANG,Average Area,99.41
2019,05,HOUGANG,Standard Deviation of Price,134100.04
2019,05,HOUGANG,Standard Deviation of Area,25.32
```

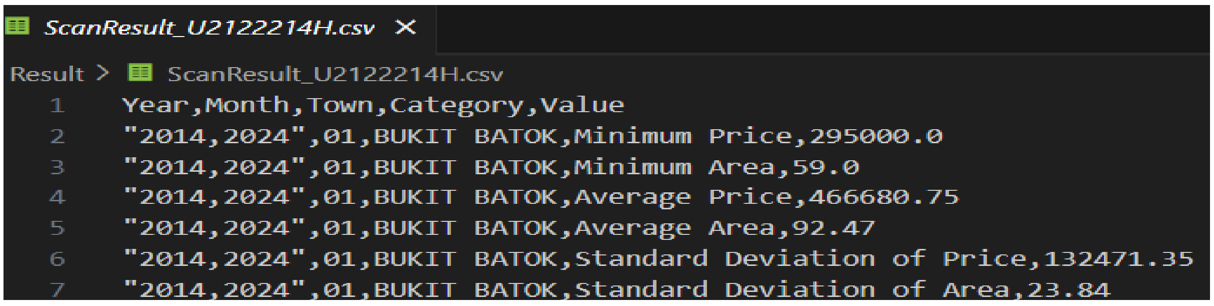
We also tested the program with the possible samples, for example, U212214H covers both “2014-01” and “2024-01”

The program handles the query by combining different years that have the same last digit (refers to our matriculation ID).

U2122214H: Town= ”BUKIT BATOK” Month=1 Year=2014 and 2024.



U2122214H					
month	(Multiple Items)				
town	BUKIT BATOK				
Min of resale_price	Min of floor_area_sqm	Average of resale_price	Average of floor_area_sqm	StdDev of resale_price	StdDev of floor_area_sqm
295000	59	466680.7458	92.47457627	132471.3485	23.84241073



```
ScanResult_U2122214H.csv
Result > ScanResult_U2122214H.csv
Year,Month,Town,Category,Value
"2014,2024",01,BUKIT BATOK,Minimum Price,295000.0
"2014,2024",01,BUKIT BATOK,Minimum Area,59.0
"2014,2024",01,BUKIT BATOK,Average Price,466680.75
"2014,2024",01,BUKIT BATOK,Average Area,92.47
"2014,2024",01,BUKIT BATOK,Standard Deviation of Price,132471.35
"2014,2024",01,BUKIT BATOK,Standard Deviation of Area,23.84
```

Discussion

We verified using the Excel pivot table tool that the implementation of the column stores and results of the queries are accurate as shown in the figures above.

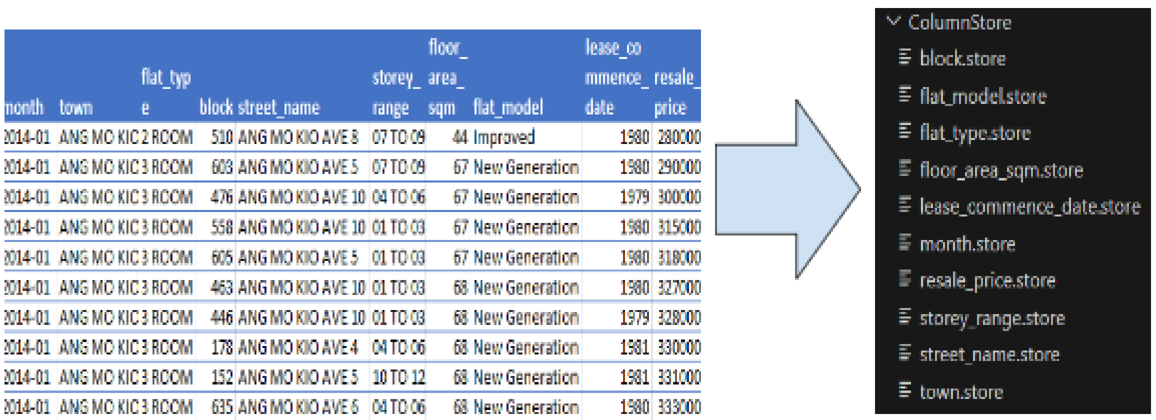
The results demonstrate considerable improvements in query performance through Zone Mapping and Shared Scans, with the combined approach yielding the best results. The default method took 0.1280 seconds for queries, whereas the optimized methods reduced this significantly. Shared Scans dropped it to 0.1057 seconds by consolidating multiple scans into a single operation, and Zone Mapping cut it down to 0.0994 seconds by limiting scans to relevant data segments pre-mapped by year. The integration of both techniques further reduced the time to 0.0988 seconds, showcasing a reduction of about 30% from the default scan.

These improvements can be attributed to more efficient use of computational resources and reduced I/O operations, where Shared Scans decrease the number of redundant data retrieval operations and Zone Mapping minimizes the data scanned by strategically segmenting the database. This optimized scanning approach not only enhances performance by reducing the amount of data to be processed but also speeds up access and retrieval.

Conclusion

In conclusion, our project demonstrated data storage through column-based storage that can handle data anomalies and store each column into individual separate files. Our project also suggested multiple different efficient data scanning and access methods through the use of shared scans and zone maps to retrieve data from the information that we had stored. The implementation of these techniques significantly enhanced query performance, reducing both the time and computational resources required for data retrieval. By leveraging shared scans, we minimized the number of read operations needed, while zone mapping allowed us to focus these operations on relevant segments of data, thus avoiding unnecessary scans of the entire database. These optimizations not only improved the speed and efficiency of our data retrieval processes but also demonstrated the potential for scalable solutions in environments with larger datasets.

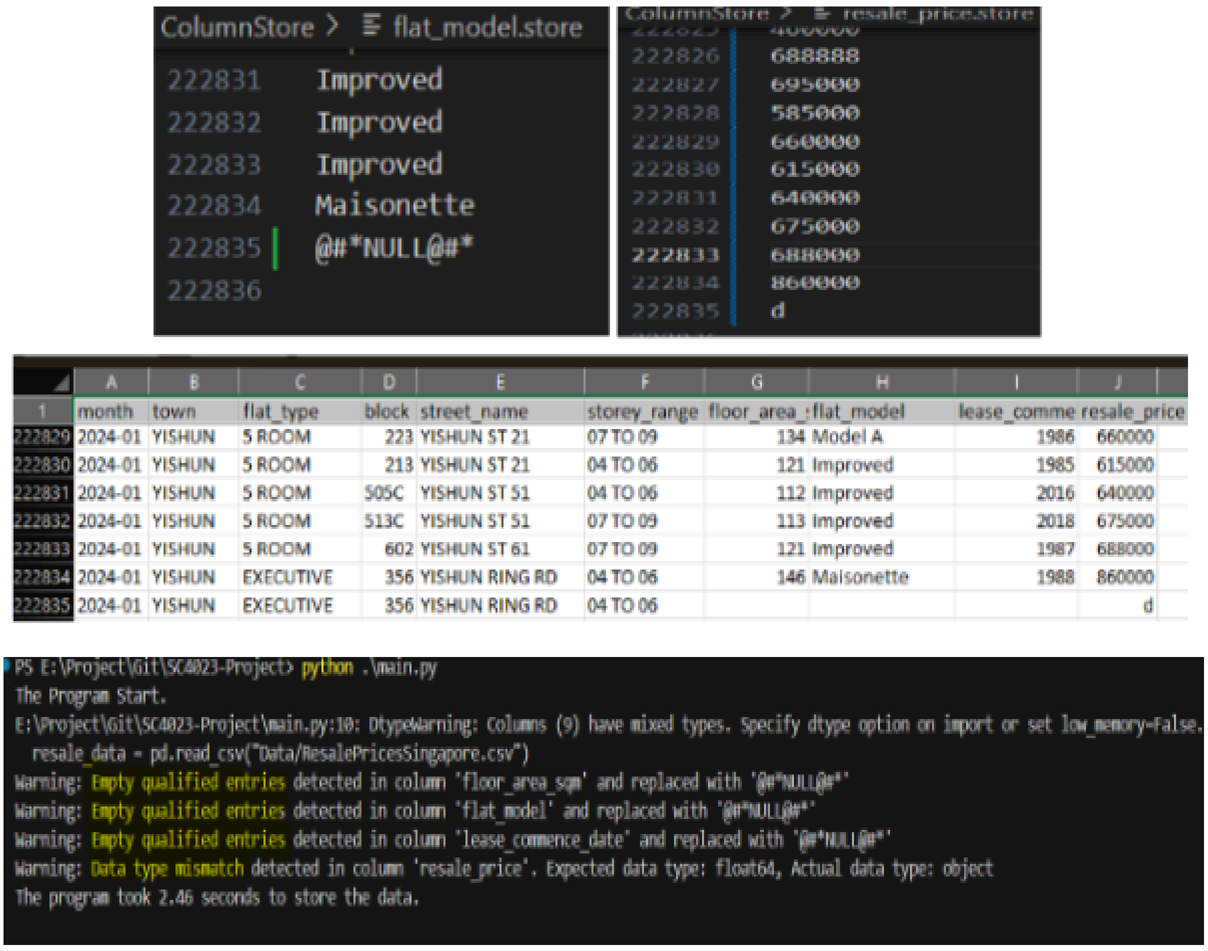
Appendix



month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	resale_price
2014-01	ANG MO KIO	2 ROOM	510	ANG MO KIO AVE 8	07 TO 09	44	Improved	1980	280000
2014-01	ANG MO KIO	3 ROOM	603	ANG MO KIO AVE 5	07 TO 09	67	New Generation	1980	290000
2014-01	ANG MO KIO	3 ROOM	476	ANG MO KIO AVE 10	04 TO 06	67	New Generation	1979	300000
2014-01	ANG MO KIO	3 ROOM	558	ANG MO KIO AVE 10	01 TO 03	67	New Generation	1980	315000
2014-01	ANG MO KIO	3 ROOM	605	ANG MO KIO AVE 5	01 TO 03	67	New Generation	1980	318000
2014-01	ANG MO KIO	3 ROOM	463	ANG MO KIO AVE 10	01 TO 03	68	New Generation	1980	327000
2014-01	ANG MO KIO	3 ROOM	446	ANG MO KIO AVE 10	01 TO 03	68	New Generation	1979	328000
2014-01	ANG MO KIO	3 ROOM	178	ANG MO KIO AVE 4	04 TO 06	68	New Generation	1981	330000
2014-01	ANG MO KIO	3 ROOM	152	ANG MO KIO AVE 5	10 TO 12	68	New Generation	1981	331000
2014-01	ANG MO KIO	3 ROOM	635	ANG MO KIO AVE 6	04 TO 06	68	New Generation	1980	333000

```
ColumnStore
  block.store
  flat_model.store
  flat_type.store
  floor_area_sqm.store
  lease_commence_date.store
  month.store
  resale_price.store
  storey_range.store
  street_name.store
  town.store
```

Figure 1



```
ColumnStore > flat_model.store
222831    Improved
222832    Improved
222833    Improved
222834    Maisonette
222835    @#*NULL@#*
222836
```

```
ColumnStore > resale_price.store
222826    688888
222827    695000
222828    585000
222829    660000
222830    615000
222831    640000
222832    675000
222833    688000
222834    860000
222835    d
```

	A	B	C	D	E	F	G	H	I	J
1	month	town	flat_type	block	street_name	storey_range	floor_area_s	flat_model	lease_comme	resale_price
222829	2024-01	YISHUN	5 ROOM	223	YISHUN ST 21	07 TO 09	134	Model A	1986	660000
222830	2024-01	YISHUN	5 ROOM	213	YISHUN ST 21	04 TO 06	121	Improved	1985	615000
222831	2024-01	YISHUN	5 ROOM	505C	YISHUN ST 51	04 TO 06	112	Improved	2016	640000
222832	2024-01	YISHUN	5 ROOM	513C	YISHUN ST 51	07 TO 09	113	Improved	2018	675000
222833	2024-01	YISHUN	5 ROOM	602	YISHUN ST 61	07 TO 09	121	Improved	1987	688000
222834	2024-01	YISHUN	EXECUTIVE	356	YISHUN RING RD	04 TO 06	146	Maisonette	1988	860000
222835	2024-01	YISHUN	EXECUTIVE	356	YISHUN RING RD	04 TO 06				d

```
PS E:\Project\Git\SC4023-Project> python .\main.py
The Program Start.
E:\Project\Git\SC4023-Project\main.py:10: DtypeWarning: Columns (9) have mixed types. Specify dtype option on import or set low_memory=False.
  resale_data = pd.read_csv("Data/ResalePricesSingapore.csv")
Warning: Empty qualified entries detected in column 'floor_area_sqm' and replaced with '@#*NULL@#*'
Warning: Empty qualified entries detected in column 'flat_model' and replaced with '@#*NULL@#*'
Warning: Empty qualified entries detected in column 'lease_commence_date' and replaced with '@#*NULL@#*'
Warning: Data type mismatch detected in column 'resale_price'. Expected data type: float64, Actual data type: object
The program took 2.46 seconds to store the data.
```

Figure 2

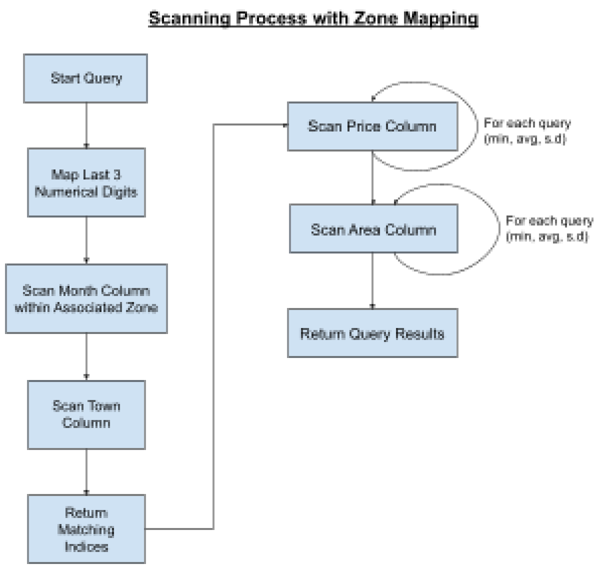

Scanning Process with Zone Mapping
Start Query
Map Last 3 Numerical Digits
Scan Month Column within Associated Zone
Scan Town Column
Return Matching Indices
Scan Price Column
For each query (min, avg, s.d)
Scan Area Column
For each query (min, avg, s.d)
Return Query Results

Figure 3

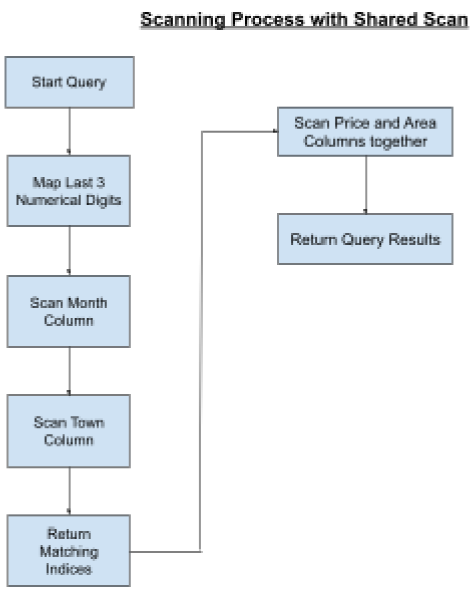

Scanning Process with Shared Scan
Start Query
Map Last 3 Numerical Digits
Scan Month Column
Scan Town Column
Return Matching Indices
Scan Price and Area Columns together
Return Query Results

Figure 4

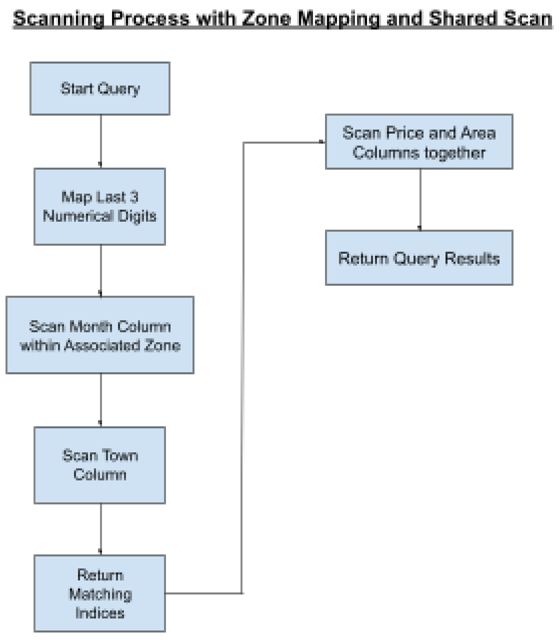

Scanning Process with Zone Mapping and Shared Scan
Start Query
Map Last 3 Numerical Digits
Scan Month Column within Associated Zone
Scan Town Column
Return Matching Indices
Scan Price and Area Columns together
Return Query Results

Figure 5

Contribution Form

Name	**Matriculation Number**	**Detailed Individual Contribution**	**Percentage (100% in total)**
Hendy	U2122559J	● Data Storage system implementation ● Convert querying results into statistics tables and output it as a file ● Report writing	40%
Jia Xiang	U2122287H	● Column Scan Implementation ● Scan Enhancement ● Report writing	40%
Peng Aik	U2022363E	● Scan enhancement ● Results verification ● Report writing	20%

<u>Name and Signature from all group members:</u>

Hendy

[illegible]

Jia Xiang

[illegible]

Peng Aik

[illegible]