CE/CZ4123/SC4023 BIG DATA MANAGEMENT

SEMESTER GROUP PROJECT

I. MOTIVATION

II. STEP 1: AN EASY PYTHOM IMPLEMENTATION

III. STEP2: PYTHOM IMPLEMENTATION WITH MEMORY ACCESS

IV. CONCLUSION

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING NANYANG TECHNOLOGICAL UNIVERSITY

IKER ROSALES SAIZ **ALEJANDRO PONCE FERNÁNDEZ** **JAIME SALAFRANCA PARDO**

I. MOTIVATION

II. STEP 1: AN EASY PYTHON IMPLEMENTATION

We have to create a program that for a given matric number retrieves a specific query among the proposed ones. In this program, the package time will be used for measuring the time of execution and accessing memory. On the other hand, the package csv will be used to retrieve all the information from the file *ResalePricesSingapore.csv.* As already mentioned the relevant variables in this file are: **town, month, resale_price,** and **flat_area_sqm**.

The first four dictionaries will be created to compare the matric relevant digit to its equivalent information in the query. Once the dictionaries are created the matric number is retrieved and each relevant digit is stored in a variable. It is important to take into account all the possibilities and combinations of months. It is assumed that the combination of starting in 2023-12 is neglected as there is not enough data.

Once the important values of the query are retrieved, we can create the arrays where the data would be stored. It is important to indicate that in this milestone only one single array for all the data is used. When the model would be implemented in Java ZoneMaps would be used for computing the values. This first milestone is just a quick idea of the model. Then we read the csv column by column only for the four important variables, as it will be done in Java we first have to read the array to check the **town** and the **month.**

Once the data is retrieved it is compared to the requirement of the query. Then the correct data is stored in similar arrays to the previous ones. In this case, the index of the data is used to select the correct data. Then the values are computed. In this case, to reduce the computing costs dual scanning is implemented. As for standard deviation, the mean has to be computed in advance.

Finally, the time taken is shown on the screen in this case it is 2.18 seconds. We will compare this value with further implementations.

As told in the assignment implementation doesn't correspond to the Big Data management perspective. To create a more realistic approach we have to simulate that the main memory is not large enough to handle all the data. That is why we will create an array limitation on the main memory and also use file-accessing to simulate disk memory accessing.

III. STEP2: PYTHOM IMPLEMENTATION WITH MEMORY ACCESS

0. Explanation

In the first part of the project, we have seen how easy this query is with just only considering that the data is small enough to be handled by the main memory. As in big Data management, it is not the case. We have to limit the main memory size. The arrays of data that we will use in this second step are limited to MAX_LINE, in our case this variable has a value of 50000. But this value is completely changeable. Moreover, to show all the memory access we will create different files that represent the disk memory.

1. Useful functions

In this part of the script, we create functions that will be very helpful for the rest of the program. We first create dictionaries, those dictionaries map the digits of the matric number to the year the town, and the month. It is important to note that the month mapping is created with the instructions of the project but it is not corresponding to the typical mapping. *(ex: January 10).*

In this project, we will separate the columns into different files. As we have already explained we do this to simulate that the main memory can handle more than the MAX_LINE, and then we will create different files of this size that will represent the disk memory.

That is why we created a zone map to store the information of each zone, the index, and the date. We notice by going through the data set that the column **month** is sorted. This is very interesting as we will not need to go completely through the data to take out the monthly information but just find the first value and the last one. This simplifies a lot of our research as it is an optimizing column search. To do so we create in this part of the work two functions for storing those values when scanning the data and for initializing the index and date values.

2. Preprocessing of the data set and relation of the disk storage

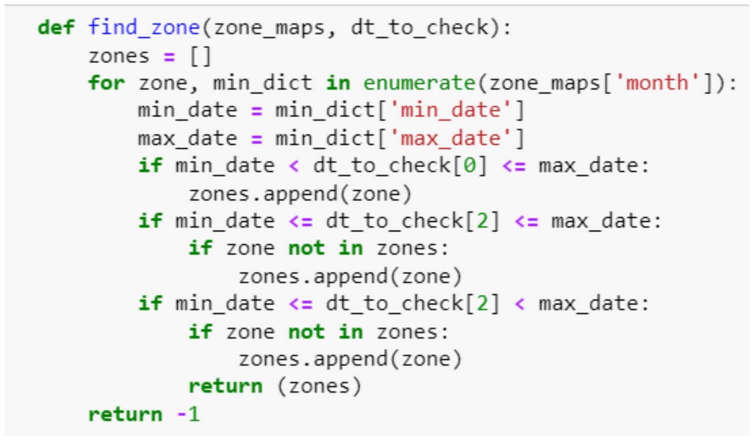
During the preprocessing, a reading of the data will be performed. The objective of this step is to prepare the data for a column analysis. For each relevant column (**month, town, floor_area_sqm, resale_price**) we will create enough files to store the data. Each column has only the value of the column. The number of files will be determined by the maximum line file. We will create a dictionary that will store the maximum and minimum dates and index.

We will also manage the different years and months in the query. If the month entered is December the year should be updated respectively. We will then need to find the indexes of the values of the data that correspond to the year of the query and then among those indexes find the ones that correspond to the town of the query.

This will be done in the processing part.

3. Finding the data indexes matching the year query.

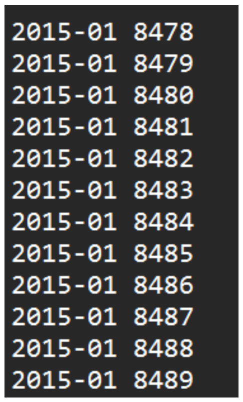
As explained previously we need to retrieve the indexes that correspond to the query. In the case of the matric number used as reference N2303635K the year is 2015 and the months are 01, 02, and 03.



```python
def find_zone(zone_maps, dt_to_check):
    zones = []
    for zone, min_dict in enumerate(zone_maps['month']):
        min_date = min_dict['min_date']
        max_date = min_dict['max_date']
        if min_date < dt_to_check[0] <= max_date:
            zones.append(zone)
        if min_date <= dt_to_check[2] <= max_date:
            if zone not in zones:
                zones.append(zone)
        if min_date <= dt_to_check[2] < max_date:
            if zone not in zones:
                zones.append(zone)
            return (zones)
    return -1
```

A file will be created for each month. Then using the find_zone() method we will easily retrieve the zones where we should look for our data. As the data is sorted by date only checking whether the data is between the maximum and minimum date of each zone is needed.

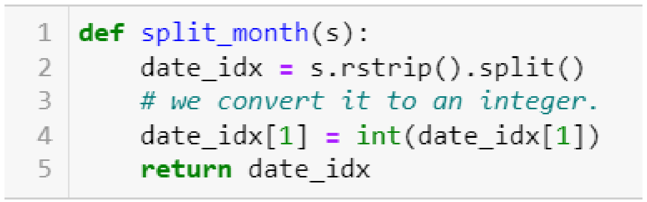
Once the zones are found, only writing the data in the folder created is left. To do so first a binary search among the sone is created to find the first date included in the query. Then all the following date indexes will be written after checking they belong to the data date of the query. The date and the index are retrieved as a single value in the column of data. Here is an example of how the data would look like.



```
2015-01 8478
2015-01 8479
2015-01 8480
2015-01 8481
2015-01 8482
2015-01 8483
2015-01 8484
2015-01 8485
2015-01 8486
2015-01 8487
2015-01 8488
2015-01 8489
```

4. Town Processing

Once the indices of the date are retrieved, the following step is just retrieving the corresponding indexes that match in date and town. With our reference matric number, the town is JURONG WEST. As a tool for this processing, we will use a tool that will enable us to separate the column of months previously created and separate the index and date.



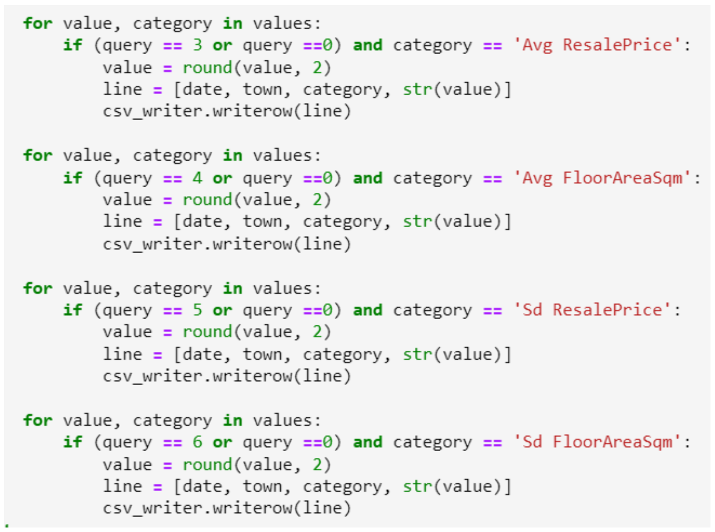
```python
def split_month(s):
    date_idx = s.rstrip().split()
    # we convert it to an integer.
    date_idx[1] = int(date_idx[1])
    return date_idx
```

The rest of the process for taking the indices of the query town is not very complicated. We will create a file for each month as it already was created and while reading the corresponding data from the slitted data we will check if the town value fits the one of the query. It is important to say that here only the indexes are used and the data is also sorted by index as the index here is only the number of the line. The indexing part has been checked with Excel. It is interesting to say that a difference of 2 in the indexes was confusing for us during the research. Finally, it was explained because of the header in the csv doc and because Excel started the index in one and Python in 0.

after this step, all the relevant indexes would be retrieved the only remaining step is computing the query values and creating the result file.

5. Computing values for the query.

To compute the values of the query first all the values will be initialized. Then as has been done previously the town data is read and the index from the town file. Then we will go along the resale price file



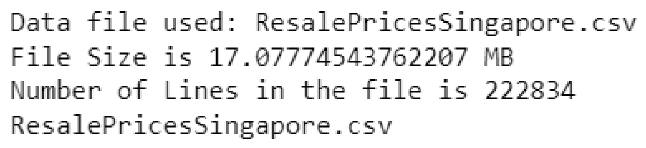
```python
for value, category in values:
    if (query == 3 or query ==0) and category == 'Avg ResalePrice':
        value = round(value, 2)
        line = [date, town, category, str(value)]
        csv_writer.writerow(line)

for value, category in values:
    if (query == 4 or query ==0) and category == 'Avg FloorAreaSqm':
        value = round(value, 2)
        line = [date, town, category, str(value)]
        csv_writer.writerow(line)

for value, category in values:
    if (query == 5 or query ==0) and category == 'Sd ResalePrice':
        value = round(value, 2)
        line = [date, town, category, str(value)]
        csv_writer.writerow(line)

for value, category in values:
    if (query == 6 or query ==0) and category == 'Sd FloorAreaSqm':
        value = round(value, 2)
        line = [date, town, category, str(value)]
        csv_writer.writerow(line)
```

and the floor_area_sqm to compute the value for minimum average and standard deviation. For Average and sminimum we will use the same loop doing a dual scanning. We will need to do a second scanning once the average is found to compute the standard deviation. But as we are going to see in the results this second scanning is not very time-consuming.

Finally, we write the results in Excel using the variable query to adjust the results we write in the CSV file to the requirements of the query.

6. User Interface

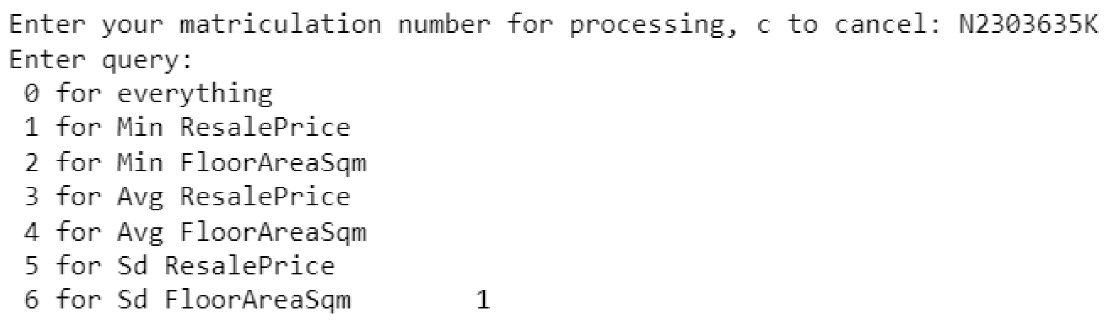
In the user interface first, some information about the data is shown. as the following:



```
Data file used: ResalePricesSingapore.csv
File Size is 17.07774543762207 MB
Number of Lines in the file is 222834
ResalePricesSingapore.csv
```

Information about the data shown to the user

Then the interface asks for the matric number of the user and for the query he wants to do giving different options.



```
Enter your matriculation number for processing, c to cancel: N2303635K
Enter query:
 0 for everything
 1 for Min ResalePrice
 2 for Min FloorAreaSqm
 3 for Avg ResalePrice
 4 for Avg FloorAreaSqm
 5 for Sd ResalePrice
 6 for Sd FloorAreaSqm        1
```

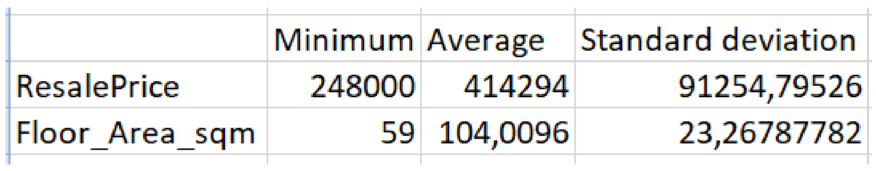
Information about the query shown to the user

Finally, all the program is run and the user gets to know at the end the time taken for all the computations.

7. Performance metrics

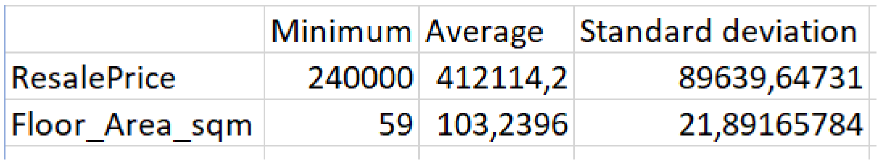
Using Excel we can see the different results we should get for the queries.

For 2015-01:



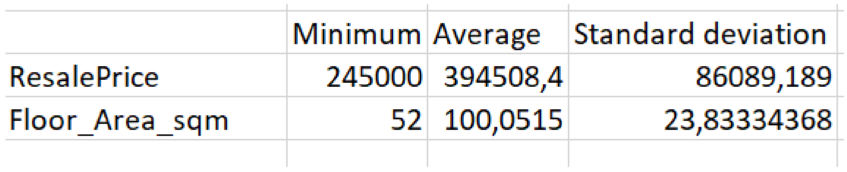
	Minimum	Average	Standard deviation
ResalePrice	248000	414294	91254,79526
Floor_Area_sqm	59	104,0096	23,26787782

For 2015-02:



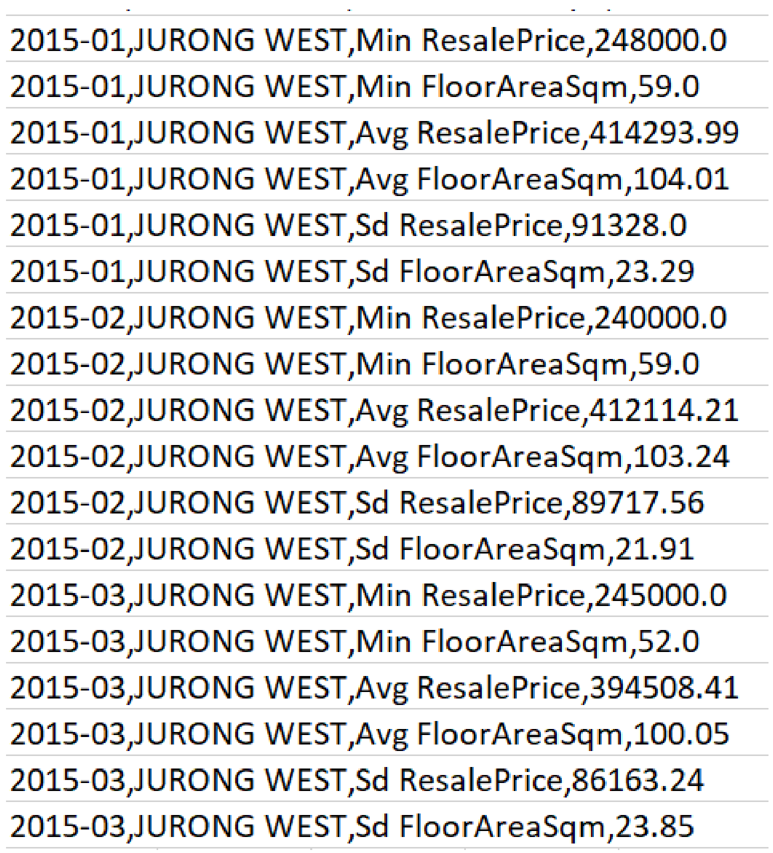
	Minimum	Average	Standard deviation
ResalePrice	240000	412114,2	89639,64731
Floor_Area_sqm	59	103,2396	21,89165784

For 2015 -03:



	Minimum	Average	Standard deviation
ResalePrice	245000	394508,4	86089,189
Floor_Area_sqm	52	100,0515	23,83334368

In comparison, our results are the following:



```
2015-01,JURONG WEST,Min ResalePrice,248000.0
2015-01,JURONG WEST,Min FloorAreaSqm,59.0
2015-01,JURONG WEST,Avg ResalePrice,414293.99
2015-01,JURONG WEST,Avg FloorAreaSqm,104.01
2015-01,JURONG WEST,Sd ResalePrice,91328.0
2015-01,JURONG WEST,Sd FloorAreaSqm,23.29
2015-02,JURONG WEST,Min ResalePrice,240000.0
2015-02,JURONG WEST,Min FloorAreaSqm,59.0
2015-02,JURONG WEST,Avg ResalePrice,412114.21
2015-02,JURONG WEST,Avg FloorAreaSqm,103.24
2015-02,JURONG WEST,Sd ResalePrice,89717.56
2015-02,JURONG WEST,Sd FloorAreaSqm,21.91
2015-03,JURONG WEST,Min ResalePrice,245000.0
2015-03,JURONG WEST,Min FloorAreaSqm,52.0
2015-03,JURONG WEST,Avg ResalePrice,394508.41
2015-03,JURONG WEST,Avg FloorAreaSqm,100.05
2015-03,JURONG WEST,Sd ResalePrice,86163.24
2015-03,JURONG WEST,Sd FloorAreaSqm,23.85
```

We can see how the results match perfectly the expectations.

It is also important to determine the time taken. In step 1 we found a time of 2.16 seconds in this second step we measure a time t = 1.79 seconds and the time to process takes around 0.33 seconds. This makes a performance of 2.12 seconds. This time seems good as it is faster than the easier implementation.

IV. CONCLUSION