Submission Number: 1 **Submission Date: March 29, 2021**

Group Number: (C19-S4) 03

Group Members:

Full Legal Name	**Location (Country)**	**E-Mail Address**	**Non-Contributing Member (X)**
Divakaran Karunakaran	India	divakar94@gmail.com	
Moses E.M.K. Tommie	Liberia	tommiemoses1988@gmail.com	
Al Rey Villagracia	Philippines	arcvillagracia@gmail.com	

Introduction

C# is a powerful programming language whose use can be extended to Microsoft Excel which proves to be very helpful in many financial projects. Here we shall be implementing a small project in C# which will help us in automating the task of adding property data to an excel workbook. With the help of functions and classes, our code shall also include basic statistical analysis through user inputs.

Overview

The application works through a user's input commands. Hence the user will have basic capabilities to do the following through a command line interface:

-Add details about a property to an existing workbook (Eg., Area in sq feet, market value and location info)

-Calculate the average market value of properties existing in a workbook

-Calculate the variance of market value of properties existing in a workbook

-Calculate the maximum market value of properties existing in a workbook

-Calculate the minimum market value of properties existing in a workbook

Methodology

The flowchart of the program is shown in Figure 1.

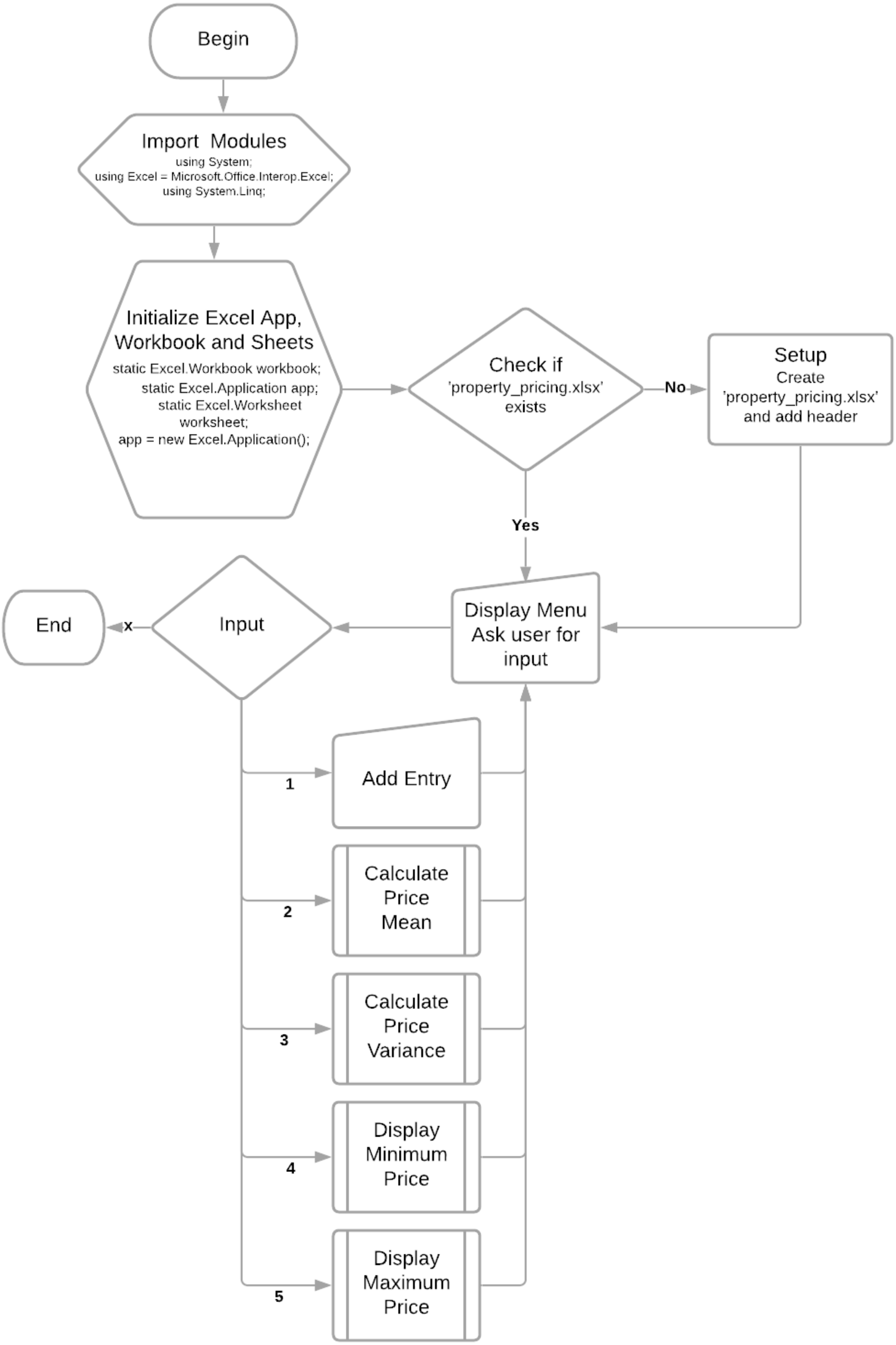
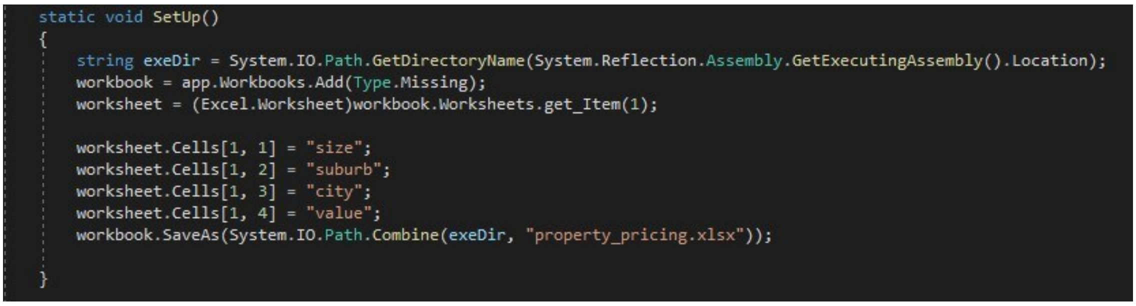

Begin
Import Modules
using System;
using Excel = Microsoft.Office.Interop.Excel;
using System.Linq;
Initialize Excel App, Workbook and Sheets
static Excel.Workbook workbook;
static Excel.Application app;
static Excel.Worksheet worksheet;
app = new Excel.Application();
Check if 'property_pricing.xlsx' exists
No
Setup
Create 'property_pricing.xlsx' and add header
Yes
Display Menu Ask user for input
Input
x
End
1
Add Entry
2
Calculate Price Mean
3
Calculate Price Variance
4
Display Minimum Price
5
Display Maximum Price

Figure 1 Flowchart of the Program

Based on the flowchart, there are 7 main functions and procedures. One procedure is to create the 'Property_pricing.xlsx' if it does not exist. The second one is to display and ask the user for input. Although, we have placed the input inquiry to the main program. The 5 functions are for the five tasks required in this workbook. Aside from the major functions created, two minor functions were created to obtain the last row position in the file and to read the data from the excel file.

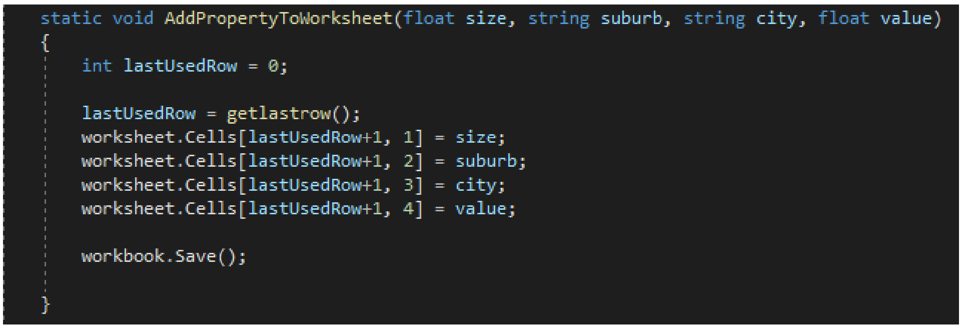
The following section describe in details the different functions and procedures in this program.

Before getting the user input for a required operation on a excel workbook, we write a function in C# that takes care of setting up a workbook. The below code takes care of this operation.



```
static void SetUp()
{
    string exeDir = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
    workbook = app.Workbooks.Add(Type.Missing);
    worksheet = (Excel.Worksheet)workbook.Worksheets.get_Item(1);

    worksheet.Cells[1, 1] = "size";
    worksheet.Cells[1, 2] = "suburb";
    worksheet.Cells[1, 3] = "city";
    worksheet.Cells[1, 4] = "value";
    workbook.SaveAs(System.IO.Path.Combine(exeDir, "property_pricing.xlsx"));

}
```

As can be seen, 4 basic columns are created - 'Size', 'Suburb', 'City' and 'Value'. These relate to the 4 basic information that we shall prompt the user to enter when option 1 is chosen to add new data to the workbook. The code snippet shows a function 'AddProperytToWorkSheet' that enables this.



```
static void AddPropertyToWorksheet(float size, string suburb, string city, float value)
{
    int lastUsedRow = 0;

    lastUsedRow = getlastrow();
    worksheet.Cells[lastUsedRow+1, 1] = size;
    worksheet.Cells[lastUsedRow+1, 2] = suburb;
    worksheet.Cells[lastUsedRow+1, 3] = city;
    worksheet.Cells[lastUsedRow+1, 4] = value;

    workbook.Save();

}
```

Once the workbook has been setup and after data is added to it from user prompts, the user will have the option to add more data by pressing **1**. Should he choose to perform statistical analysis on the existing data, he may choose options **2,3,4 or 5** each of which will redirect to a function that will perform the necessary operation as detailed below.

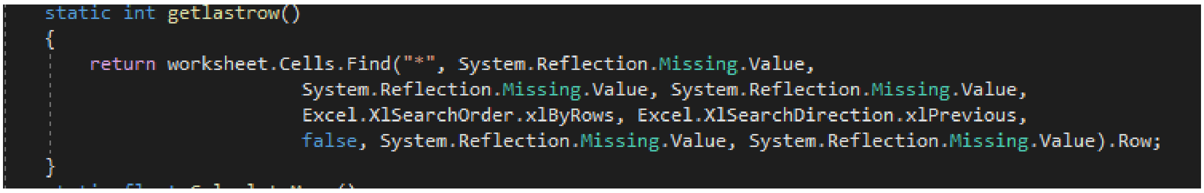
2-> Will redirect to function ***CalculateMean()*** which will help calculate the mean of market values

3-> Will redirect to function ***CalculateVariance()*** which will help calculate the variance of market values of all data in the workbook

4-> Will redirect to a function ***CalculateMinimum()*** which will calculate the minimum value of market values of all data

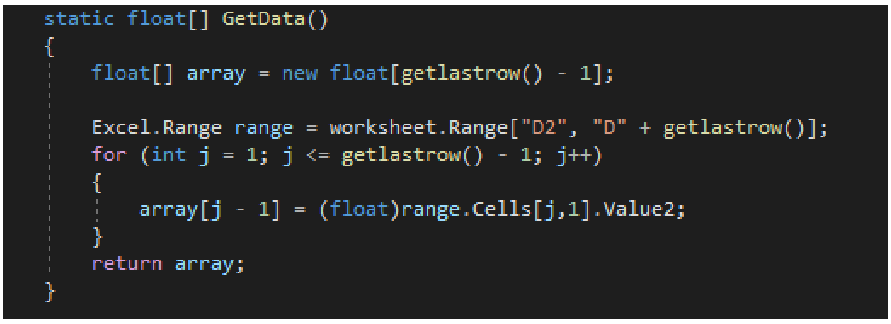
5-> Will redirect to a function ***CalculateMaximum()*** which will calculate the maximum value of market values of all data

Each of the above statistical functions will be required to access the ‘market value’ column of our workbook to perform the calculations. This is an important step in the code and two specific functions have been written exclusively for this to make this possible. The two functions are:



```
static int getlastrow()
{
    return worksheet.Cells.Find("*", System.Reflection.Missing.Value,
                        System.Reflection.Missing.Value, System.Reflection.Missing.Value,
                        Excel.XlSearchOrder.xlByRows, Excel.XlSearchDirection.xlPrevious,
                        false, System.Reflection.Missing.Value, System.Reflection.Missing.Value).Row;
}
```

And,

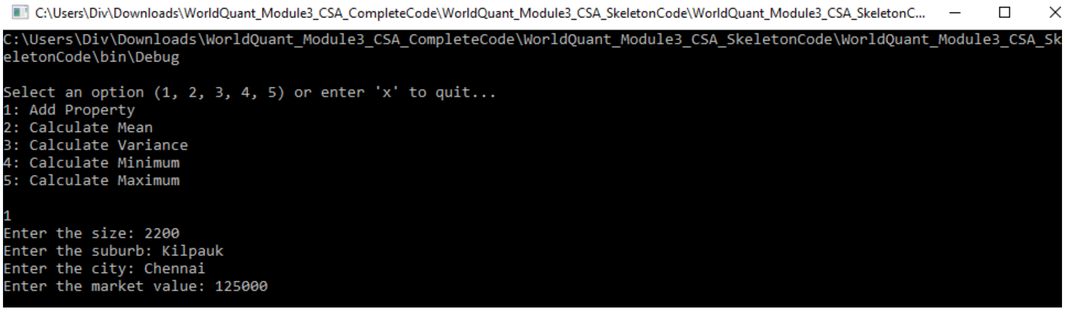


```
static float[] GetData()
{
    float[] array = new float[getlastrow() - 1];

    Excel.Range range = worksheet.Range["D2", "D" + getlastrow()];
    for (int j = 1; j <= getlastrow() - 1; j++)
    {
        array[j - 1] = (float)range.Cells[j,1].Value2;
    }
    return array;
}
```

Getlastrow() is a function that aids in obtaining the last row number of a workbook which has several data in it, whereas ***GetData()*** is a function that aids in extracting the values present in the 'marketvalue' column in the workbook into an array. This array is then used by the other functions to calculate the statistical values.

Results and Discussion

This section will include screenshots of a test run that was performed after the code completion. Screen grabs of various user inputs will be shown here thereby highlighting the working of the written code.

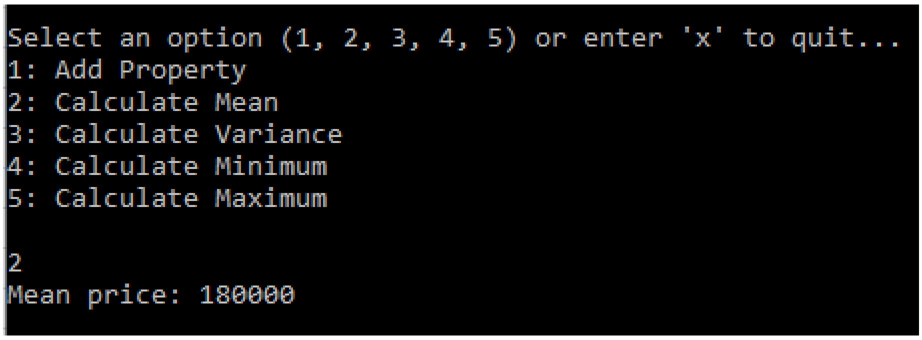


C:\Users\Div\Downloads\WorldQuant_Module3_CSA_CompleteCode\WorldQuant_Module3_CSA_SkeletonCode\WorldQuant_Module3_CSA_SkeletonC...

```
C:\Users\Div\Downloads\WorldQuant_Module3_CSA_CompleteCode\WorldQuant_Module3_CSA_SkeletonCode\WorldQuant_Module3_CSA_Sk
eletonCode\bin\Debug

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

1
Enter the size: 2200
Enter the suburb: Kilpauk
Enter the city: Chennai
Enter the market value: 125000
```

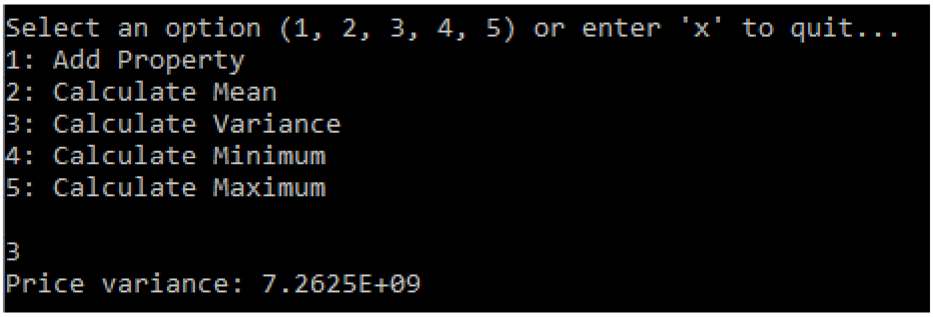
Fig 1: The main command line output which prompts the user to enter the options.

It can be seen from above that choosing 1 as an option further prompts the user to enter the various details of a property which will then be appended to the workbook.



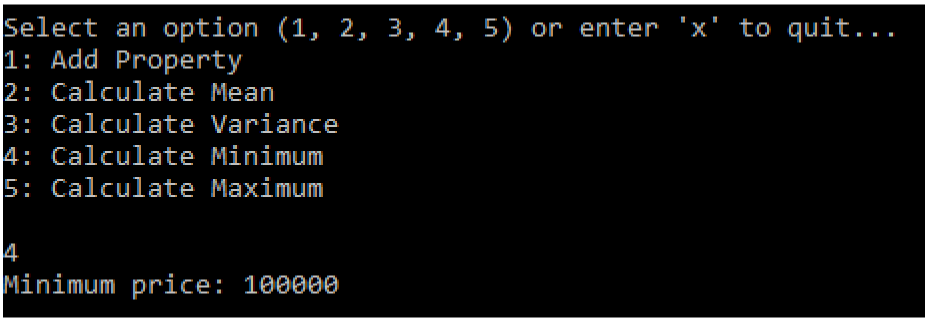
```
Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

2
Mean price: 180000
```

Fig 2: Choosing 2 helps calculate the average market value of the data.



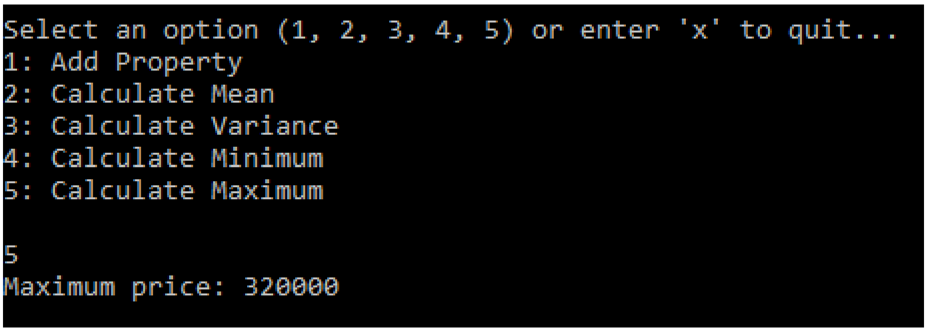
```
Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

3
Price variance: 7.2625E+09
```

Fig 3: Choosing 3 calculates the variance of market value of properties.



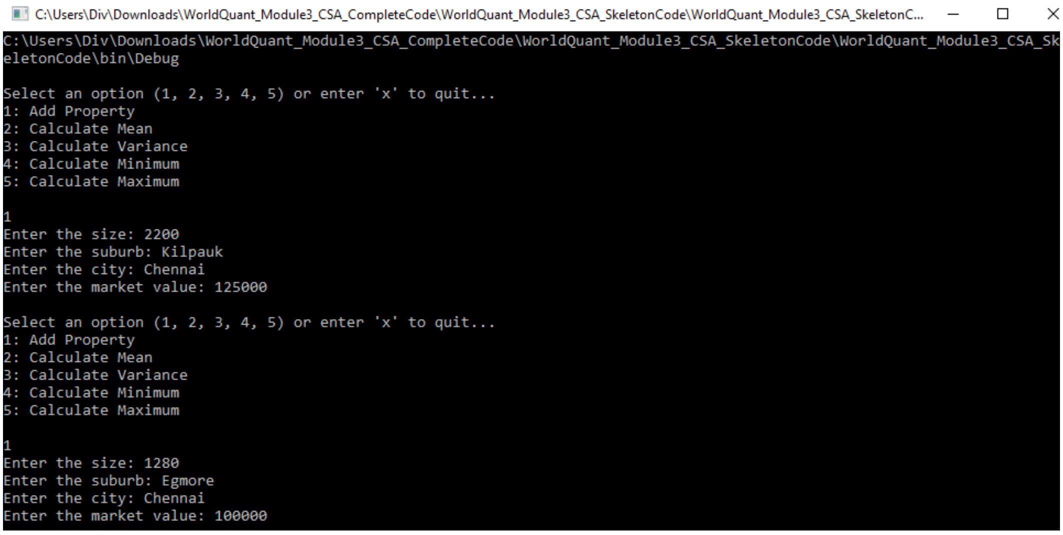
```
Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

4
Minimum price: 100000
```

Fig 4: Choosing 4 shows the minimum of market value of properties.



```
Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

5
Maximum price: 320000
```

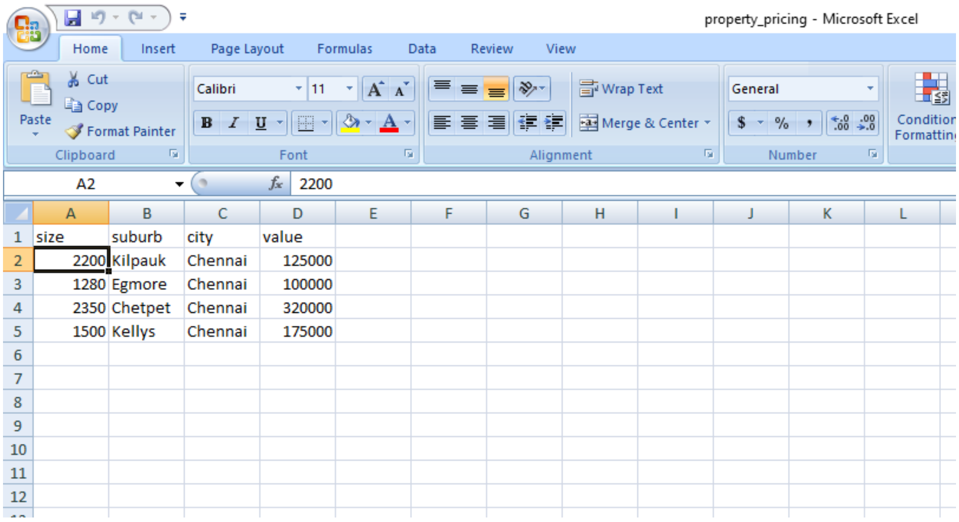
Fig 5: Choosing 5 shows the maximum of market value of properties.



C:\Users\Div\Downloads\WorldQuant_Module3_CSA_CompleteCode\WorldQuant_Module3_CSA_SkeletonCode\WorldQuant_Module3_CSA_SkeletonC...

```
C:\Users\Div\Downloads\WorldQuant_Module3_CSA_CompleteCode\WorldQuant_Module3_CSA_SkeletonCode\WorldQuant_Module3_CSA_Sk
eletonCode\bin\Debug

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

1
Enter the size: 2200
Enter the suburb: Kilpauk
Enter the city: Chennai
Enter the market value: 125000

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

1
Enter the size: 1280
Enter the suburb: Egmore
Enter the city: Chennai
Enter the market value: 100000
```

Fig 6: Adding multiple entries to the workbook.



A2 fx 2200

	A	B	C	D
1	size	suburb	city	value
2	2200	Kilpauk	Chennai	125000
3	1280	Egmore	Chennai	100000
4	2350	Chetpet	Chennai	320000
5	1500	Kellys	Chennai	175000

Fig 7: property_pricing.xlsx which shows all the data entered by the user.

References

n.d.(May 2, 2017). C# Programming guide. Retrieved from https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/

n.d.(March, 2021). World Quant University Data Feeds and Technology Module 2

support.office.com. (2019).Excel function (by category).[online] Available at: https://support.office.com/en-us/article/excel-functions-by-category-5f91f4e9-7b42-46d2-9bd1-63f26a86c0e