# Programming and Data Science for the Professions: Group project specification

In this project we will be considering a fictional company called CityCorp, which is considering where to open a number of shops in the UK. Currently they are trading online, and your task is to use the trading data to determine the best locations to open each of the new stores. CityCorp plans to open three new stores: for Men, for Women, and for Children.

On Moodle you have access to the following files:

personal.xlsx

This file contains a record of all of the personal accounts with CityCorp. Each account consists of an account number, the account holder’s name, and the postcode of their address.

orders.xlsx

This file contains a record of every order that has been made. Each order consists of the order number, and then a list of the items bought and the number of each. Each order is listed over one or more lines of the file, with the start of each order indicated by the next order number.

catalogue.xlsx

This file contains a list of every item that is sold by CityCorp. For each item the file lists the item code, which category of store would sell it, and the value of each item. The categories are Man, Woman, Adult, and Child, where Adult indicates that the item would be on sale in both of the Men and Women stores.

master.xlsx

This fill contains a record of each of the account numbers, followed by a list of the orders made by that account.

postcodes.csv

This file is based on the Office for National Statistics’ record of the location of every postcode in the UK. The original csv file is over 1Gb, so you have a simplified version, which only lists the latitude and longitude of each postcode. Even this reduced version is too large to open completely in Excel (as there are approximately 1.8 million postcodes in the UK!), although if you try and open it you will be able to see the initial part of the file. This example illustrates why csv files are important: they can handle datasets that are too big to save as Excel files. All code which uses this file will take a little time to work through the postcodes, so do not be concerned if your programmes take a little time to run.

locations.xlsx

This file gives possible locations where the new stores could be opened. For each location the file lists the location name and the latitude and longitude.

Backup.xlsx

This file can be used if you are unable to complete Parts 1 and 2 of the project, so that you have data to use in Parts 3 and 4.

## Overview of the task

**Part 1:** Your first task will be to use the above files to make a pair of xlsx files linking data from the initial datasets. The first file will use the orders and catalogue files to determine the value of each order with respect to the different types of store. The second file will aggregate this order data for each of the individual accounts.

**Part 2:** The next task is to link the postcode data associated to each account with the latitude and longitude of the associated address. This sounds quite straightforward - however the postcode datafile contains over 1.8 million records, and so the code has to be very carefully constructed so that it will run in a reasonable time.

**Part 3:** Next you will generate scatter plots to illustrate the spread of the account sales for each type, with the size of each point corresponding to the value.

**Part 4:** Finally, you will use a proximity algorithm (described below) to determine how successful each type of store would be at each of the given locations, and hence choose the best location for each type. This will require you to create a function to calculate the distance between two points given their latitude and longitude, and then use this function to give a weighting to each sale in terms of the distance from the candidate location.

Each Part can be attempted separately. **If you are not able to complete all of the requirements for Parts 1 and 2 then you can use the file called Backup.xlsx which contains a version of the data from Parts 1 and 2 which you can use in Parts 2 and 3.** Note that this data is different from the data that you will obtain from Parts 1 and 2, and should only be used if you are unable to complete those Parts.

It may help you to know that there are 9999 orders, and 1000 accounts, and the catalogue contains 499 items. You should examine each file to understand the format (for the postcode file you will only be able to look at some of the data as it is too big to open in Excel!). You may find it helpful to know that the orders file ends with the word END in the first column.

## Details for Part 1

1. Write a python programme which creates a file called OrderValue.xlsx with one row per order. The first cell in each row should contain the order number, and the second to fourth cells should contain the total value of the items in that order that can be sold respectively in the Men’s, Women’s, and Children’s shops. For some orders items will count to more than one of these totals. You will need to use the data in the orders and catalogue files to do this. To do this:
   1. First create the answer sheet with order numbers in column 1 and zeros in the next three columns.
   2. Set test = True and both ordernumber and currentrow to be 1. These will keep track of which order number we are on, and which row of the orders file we are looking at.
   3. While test is true, carry out steps d-j as follows.
   4. Check if the entry in the first column of the row below the current row of the orders file equals “END”, and if it does then set test to be false.
   5. Set item\_number to be the value in the current row of the orders file converted into an integer (you will need to remove “CITY” from the beginning to do this).
   6. Set item\_price to be the value in the third column of row item\_number of the catalogue.
   7. Calculate the value of that part of the order by multiplying the item price by the quantity in column 3 of the orders file.
   8. Check whether the given item is of type Man, Woman, Child, or Adult, and add the value calculated in step g to the relevant columns of your answer sheet.
   9. If the entry in the first column of the row below the current row of the orders file is non-empty then increase the order number by one. (To test whether a certain cell.value is empty use cell.value = None which is true if the cell is empty and false otherwise.)
   10. Increase the current row number by one.
   11. Save your answer sheet as the required xlsx file.
2. Write a python programme which uses the data in the file constructed in step 1 together with the master file to construct a file called AccountValue.xlsx with one row per account. The first cell in each row should contain the account number, and the second to fourth cells should contain the total value of all items from all orders from that account that can be sold respectively in the Men’s, Women’s, and Children’s shops. The master file uses a different format for order numbers than the Order file (for example the same order is referred to as CO-123 in the Orders file, but just as 123 in the master file).

## Details for Part 2

The main aim of this part is to write a python programme which creates a file called AccountLocation.xlsx with one row per account. The first cell in each row should contain the account number, and the second and third cells should contain the latitude and longitude respectively of the postal address associated to that account. You will need to use the data in the personal and postcode files to do this. However, if you try to do this in a direct way you may end up looking at 1.8 million postcodes 1000 times. This is far to slow a process!

When trying to solve this Part it is easy for your code to run for a very long time (or never terminate if you make a mistake). Control+C in the Console pane will stop your code if it is running for too long. You may also want to try to carry out this task for a small sample of 10 accounts first, rather than for all 1000, until you get your code to work.

We will speed up our code by creating a list called tosort whose elements will each be a list consisting of a row number i and the corresponding postcode from row i of the file personal.xlsx. We will then sort this list by the postcode values.

Once we have done this we can use the csv module to loop through the rows of the postcode file, and check to see if the postcode in the first entry of tosort (ie the value of tosort[0][1]) matches the postcode in the first column of that file. As the postcode file is in alphabetical order, this will be the first postcode that we need to find. We will then add the corresponding latitude and longitude to our answer worksheet and remove the first entry from tosort. The row we need to add these to is given by the first element of the first entry from tosort, ie by the value tosort[0][0].

The only complication is that we could have a repeated postcode in tosort, so our code will then have to check that the same row is not also the one we need for the new first element of tosort.

We keep going through the rows, removing the first element of tosort as we find it, until tosort is empty when we can stop.

This process means that we only cycle through the lines of the postcode file once, and do not have to check all 1000 accounts for every line of the postcode file. This reduces the time we take by a factor of about 1000 over the method you might first think of trying.

If you want to check your code has worked, typing a postcode followed by the word latitude into Google will return the associated latitude and longitude.

1. Write a python programme which creates a file called AccountLocation.xlsx with one row per account. The first cell in each row should contain the account number, the second cell the corresponding postcode, and the third and fourth cells should contain the latitude and longitude respectively of the postal address associated to that account. Cells five to seven should contain the values calculated in step 2 for men, women, and children for each account from AccountValue.xlsx.

The above method can be implemented as follows:

* 1. Use openpyxl to create a workbook containing a worksheet where you store the account numbers in column 1 and the corresponding postcodes in column 2.
  2. Add the corresponding values for men, women and children in columns 5 to 7.
  3. Create a list called tosort where each element of the list is itself a list of the form

[i, postcode\_i] where postcode\_i is the postcode in row i of the personal file.

* 1. Use the command tosort.sort(key=lambda x: x[1]) to sort this list by the postcodes.
  2. Now define a variable called nonempty to be True, and use the csv reader to go through the rows of the postcode file. For each row set a new variable test = True, and while test and nonempty are True do steps f to h below.
  3. If the entry in position zero in the row matches the entry tosort[0][1] (ie the postcode in the row matches the postcode in the first pair in tosort) then write to the cell in your worksheet in position (tosort[0][0], 3) the latitude and to the cell in position (tosort[0][0],4) the longitude of the postcode, and then delete tosort[0].
  4. Else set test = False.
  5. If the length of tosort is 0 then set nonempty = False.
  6. Save the resulting workbook as AccountLocation.xlsx.

## Details for Part 3

In this part we are going to visually review our data. We will draw a scatter plot for each of the three types of shop. We may see that there is a problem with our data (depending on which postcodes we had in our initial file)! We will then consider how this data can be cleaned up.

1. Use Pandas to read AccountLocation.xlsx (or Backup.xlsx if you were unable to complete Part 1) as a dataframe, adding suitable one word column names at the same time. Use the pandas .plot.scatter method to plot longitude against latitude, where the size of the dots corresponds to the value of the order for either Men, Women, or Children. You will have two problems: the dots are too large and there are a few isolated points far away from the rest.

Looking at the latitude and longitude values in your file you will find several unexpected values. All latitudes in the UK are between 49 and 61, but you may find numbers that are outside this range. It turns out that the postcode file we used from the Office for National Statistics contains a number of postcodes with no associated geographic data (corresponding to postcodes that are no longer in use), which have all been assigned the same values.

1. Use a suitable mask to remove the data corresponding to these bad postcodes, and then plot another scatterplot. Find out how to scale the dot size so that you can divide all values by 100, and how to make the plot square with a side length of 20 inches using the figsize option. Modify your code for part 4, replot your data and save this as as png file.

## Details for Part 4

Now that we have a file containing the values for each type of shop by order location, we can determine the optimal placement of our three stores. For this we will use a simple model where the probability that a customer will travel to the store is based on the straight-line distance between the two points. Clearly this is a simplification of real life, as it ignores the actual distance that a customer would travel (or the time that it would take them). We will use the following model for our probabilities.

We will assume that a customer will visit the shop with the following probabilities:

| Distance from shop | Probability of visiting |
| --- | --- |
| <= 5 km | 1 |
| > 5 km but <= 10 km | 0.7 |
| >10 km but <= 15 km | 0.4 |
| >15 km but <=25 km | 0.1 |
| >25 km | 0 |

In order to calculate distances we will use the **Haversine formula** which given a pair of points in terms of their latitudes and longitudes determines the distance between them.

Details of this formula can be found at

<https://www.movable-type.co.uk/scripts/latlong.html>

1. Using the formula described on this webpage write a Python function which converts a pair of points given by latitudes and longitudes into a distance, and check that your function works using the online tool at the above link. You will need to use a suitable module for your trigonometric functions, and will need to investigate how to use the atan2 function.
2. Using your distance formula and the file AccountLocation.xlsx (or Backup.xlsx if you were unable to complete Part 1) calculate the estimated income for each store location by type multiplying the value by type for each location by the probability that such a customer will visit the given store location, and summing over all accounts. If you have an error involving ints and strings you may need to convert the values for latitudes and longitudes from AccountLocation.xlsx into floats. Your code should produce a file called Income.xlsx which has a row for each possible store locations from locations.xlsx, where the first column contains the location name, and columns two to four contain the estimated income for Men, respectively Women, respectively Children.
3. The three stores must be in different locations. Write some code that uses Income.xlsx to determine where the three stores should be located to maximise the total estimated income, and prints a suitable message to the console.

Your final submission should consist of the Python code that you wrote for steps 1-8 above, together with the files

* AccountLocation.xlsx
* OrderValue.xlsx
* AccountValue.xlsx
* Income.xlsx

and a word document containing your final scatter plot and the three locations that you have determined in step 8, along with the type of shop to be located at each one.