Introduction

This is an individual assignment and it must be completed independently. The assignment consists of three questions, covering materials from Topic 1 to Topic 11. These questions are incremental in nature, ie, a later question relies on a previous question. You should complete these questions in their natural order.

Please note that one of the objectives of this unit is to develop skills in self-learning and research. This assignment deliberately includes questions that require some independent research on your part in order to come up with an answer.

The assignment will be marked out of 100 marks, including 10 marks allocated to the quality of presentation (including formatting and layout) and adherence to Documentation and Submission Requirements. To achieve good marks you must strictly adhere to the Documentation and Submission Requirements.

The marks you will receive for this assignment depend on the following two aspects:

- How many required features are completed and are fully working
- The quality of your work compared to other students

You will receive a top grade only if your assignments meet all requirements with the outstanding quality compared to other students.

You should try each question as soon as the relevant topic is covered, not wait until all relevant topics are covered. If you don't you may find that you do not have enough time to complete the assignment. For example, the first question relies on the materials covered in the first 6 topics. The second question relies on the materials covered in the first 7 topics, and the last question relies on the materials covered in the first 11 topics.

Deadline and Penalty for Late Submission

The submission deadline for this assignment is specified on the Unit Information Page of the LMS. If there is any change in the submission deadline, it will be announced in the Announcements page of the LMS.

Assignments submitted on or before the deadline will be marked out of 100%.

Late submission of the assignment will have 10% of the full mark of the assignment deducted per day (including weekend days and public holiday) unless an application for an extension of the submission deadline is granted. Submission that is late by more than 7 days will not be accepted (in fact the LMS will stop accepting assignments 7 days after the published deadline).

Applications for extension of your assignment deadline can only be made via email to the Unit Coordinator (ICT582@murdoch.edu.au), normally prior to the specified due date of the assignment. Before applying for an extension, you must read the section "Application for extension of the deadline" in the Unit Information and

Learning Guide (page 13-14) carefully and understand what are the accepted grounds for granting extension and the procedure of the application.

If an extension is granted (also by email), you must attach a copy of the email to your submission (see Documentation and Submission Requirements). Applications for extension by phone or in person do not count even if granted. The above policy will be rigorously enforced.

Question 1 (40%): A simple sales management system

Western Wholesales Pty Ltd sells variety of products to the re-seller markets. It sells products in the following 6 categories: food, alcohol and beverage, apparel, furniture, household appliances, and computer equipment. Design a program to record and manage the sale transactions of the company and implement the program in Python programming language.

The program maintains the sales records in the computer internal memory and allows the user to manually add the sales records to the computer program and search and delete the sales records. Specifically, the program will display a menu that allows the user to perform different operaions repeatedly. The menu must include the following options:

1. add a new customer to the program. The customer details include the `customer_id`, the customer's `name`, the customer's `postcode`, the customer's `phone number`. When the user enters the details, only the customer's name is required, other details are optional (leaves it empty). Note, the customer id must be unique and be automatically generated. You may use a positive integer number for the customer id. For convenience to the user, the auto-generated customer id should be displayed after the customer record is successfully added.
2. add a new transaction for a given customer. In this case, the user needs to provide the customer id. If the id exists, the user can then enter the details of a new transaction, ie, `date`, `transaction_id`, `customer_id`, `catetegory`, and `value`. Note for each transaction, there must be a unque, auto-generated, transaction id for that transaction record.
3. search customers using a single search string. The search string is compared to the customer's id, name, postcode and phone number. Then the program will display the details of all matching customers. The search must be case-insensitive and must allow for partial match, for example, the search string "john" would match "John Smith" as well as "Elton Johns"
4. search sales transactions using a single string. The search string is compared to customer id, date, category and value. Then the program will display the details of all matching transactions.
5. display the sales transactions for a given customer using the id of that customer. When the customer id is given, the program should display all transactions due to that customer.
6. delete a transaction record with a given transaction id.
7. delete a customer with a given customer id together with all transaction records due to this customer.
8. quit: to quit the program

Restrictions:

- your design must be modular with at least two modules (in addition to the main module) in your implementation, one provides functionality related to customers and the other provides functionality related to the sales transactions.
- in this question, do not use ndarrays to store the customeer records or transaction records.
- in this assignment, do not use any Data Base Management System or module.

Recommendations:

- use two separate data structures, one to store the customer records and the other to store the sales transactions.

Question 2 (25%): Load sales records from files and saves them to files

Revised your solution to Question 1 to add new features in the form of additional menu options:

9. load customer records from a CSV file and add these new records to to those already in the memory. The program should prompt the user to provide a file path. The customer records from the file are added to those already in the memory. Check for duplicates of customer records (ie, records with the same id): in case of duplicate, the one from the file should be ignored.
10. save all customer records from the memory to a CSV file. Your program should prompt the user to provide a file path. If the file does not exist, create a new file. If the file does exist, warn the user that the content of the file would be lost and give the user the option of either changing the file name, overwriting the file, or cancel the operation. Then save the customer records to that CSV file.
11. load transaction records from a CSV file and add these new records to the those in the memory. The ids of these new transactions may need to be re-generated to avoid clashes with those of the existing transactions. Furthermore, for each transaction, you must check whether the customer exists in the memory. Reject those transactions from unknown customers.
12. save all transaction records from the memory to a CSV file. Your program should prompt the user to enter a file path. If the file does not exist, create a new file. If the file does exist, warn the user that the content of the file would be lost and give the user the option of either changing the file name, overwriting the file, or cancel the operation. Then save the transactions records to that CSV file.

You may use Python's builtin module `csv` to handle CSV files.

For your convenience, I have created two test files Customers.csv and Transactions.csv. The customer file contains the details of 200 customers while the transaction file contains 1000 sales records. The information inside the files are completely fititious and random.

You are required to provide evidence in your assignment documentation that your program was tested successfully using the above test files

You should test your program using multiple test data sets. You may create a few small test sets, perhaps containing a dozen customers and several dozens of transaction records, and use these small test data sets to test your program first. Once your program is tested successfully with these small test data sets, you should then test your program using the above large test data set.

When creating your own test files, use the same column names as in Customers.csv and Transactions.csv.

Question 3 (25%): Display sales performance graphically

In Question 1 and Question 2, you may have used two language builtin data structures, such as lists or dictionaries, to store the customer records and transaction records. These builtin data structures provide many useful methods which greatly simplified your programming. For small data size, this is an appropriate design choice. However, if the system is to be used to handle very large data sets, such as thousands of customers and tens of thousands (or even more) of transactions, these data structures would not be the most appropriate to use as they are less efficient compared to the ndarrays.

To improve the system efficiency, in this question, you will revise the program from Question 2 by replacing the two data structures used to store customer records and transaction records with two ndarrays and make the necessary adjustments to the rest of the program.

Furthermore, you will provide the following additional features for analysing the sales performance in the form of the following additional menu options:

13. display the monthly sales values and transaction numbers with two line graphs in one axes. The line graph must have an appropriate title, the labels for X axis and Y axis, and a legend.
14. for a given customer, display the monthly sales values and the number of transactions generated by the customer using two line graphs in one axes. The line graph must have an appropriate title, the labels for X axis and Y axis, and a legend.
15. for a given postcode, display the monthly sales values and the number of transactions generated by the customers located in the postcode area using two line graphs in one axes. The line graph must have an appropriate title, the labels for X axis and Y axis, and a legend.

You should use Python modules `numpy` and `matplotlib` for this question.

Documentation and Submission Requirements

Your assignment must be submitted in the form of a single zip archive file to the LMS. No other format will be accepted. The zip file must be named in the following format:

```
<your last name>_<your first name>_<your student number>.zip
```

For example, the student, John Smith, with student number 12345678, would name his zip file as `Smith_John_12345678.zip`

Your zip archive file must contain the following files/folders:

- One PDF document named `Assignment.pdf`. See the detailed requirements of this file later in this section.
- Three folders named `q1`, `q2` and `q3` for each of the three questions. Each folder must contain all files for the question, including the Python code and data files (if any).

  You must include all files for each question in a separate folder. If you only include one folder containing all the files for one question, you are deemed to have only submitted your solution to one question, even though that solution may have satisfied the requirements of each of the three questions. In that case, the maximum mark you will receive is 50%.

The file `Assignment.pdf`

This file must contain the documents in the following order:

1. Assignment Check List: you must complete this checklist and include it in your PDF file.
2. Extension Granted: if you have been granted an extension, include the email from your Unit Coordinator.
3. Table of Contents: listing of the page number of each question.
4. Documentation for *each* of the three questions. The documentation for each question must include:
    a. ***The question number and the question title***
    b. ***Discussion of your solution***: in particular, you should emphasise any aspect of your solution that is novel or unusual. You should also discuss the technical choices available to you, the consideration behind your technical choice, and whether the result of your solution meets your expectation. You should highlight the strength and weaknesses of your solution and point out what improvements can be made to the solution. This discussion should not exceed one page.
    c. ***Self-diagnosis and evaluation***: a statement giving the following details for each requirement of the question: what features have been fully completed, and are working, and what features have been completed but not fully working, what features have not been fully completed or not attempted. This statement is **essential**. You could lose up to 50% of the mark allocated to the question if this statement is missing, misleading, or too vague!
    d. ***Test evidence***: for each required feature of the question, you must provide a sufficient number of test cases to demonstrate that your solution meets the requirements of the question. The presentation of each test case must include
        - which feature is to be tested,

- the evidence of the test (which can be a copy of your terminal output showing the command and output of the command),
- the conclusion of the test, ie, whether the test evidence proves that the tested feature meets the requirement.

e. ***A list of the file names***: for your Python source code and data files used by your program for the question.

The documentation and submission requirements described above will be strictly enforced. Your assignment will not be marked, or your marks will be significantly reduced if you fail to adhere to the above requirements. Your mark will be substantially based on the test cases you presented. You will not receive a pass mark if the test cases presented do not cover enough required features, and/or do not convince us that your solution satisfies the requirements of the question (regardless of how good your program is).

Policy on the Reuse of the Third-Party Source Code

Please read this section very carefully.

All students are encouraged to complete the assignment independently and write their own source code. I understand, however, that occasionally there may be justifiable reasons to use source code from a third party, including the source code from the Internet and the source code generated by any AI tool such as ChatGPT. Please note that if you have used one or more pieces of third-party source code in your program (this includes the situations where you have made minor modifications to the third-party source code), your assignment will be acceptable only if you have satisfied all of the following conditions:

1. The third party source code is fully identified, including the page numbers and line numbers, in your source code and also in your assignment documentation, and
2. The origin of the third-party code is fully disclosed and acknowledged in your assignment submission, and
3. The third party source code is fully commented in your program listing. All variables, functions and major control structures must be commented to show clearly that you understand the logic of the code, and
4. The third-party source code is less than 10% of your program (in terms of the number of lines), excluding the code specifically allowed to be used, and
5. The source code does not come from a student assignment, whether from Murdoch University or not.

Failure to satisfy any one of the above conditions will result in 0 mark being awarded to your entire assignment.

If unsure, you must seek clarification from your tutor or the unit coordinator, and get his or her confirmation in writing.

The above policy will be rigorously enforced by the Unit Coordinator and Tutors.

Grievance with Assignment Marking

Once you have received your marked assignment, if you have any grievance with the marking, you must raise it with your Unit Coordinator by email, within 7 days of returning of your marked assignment to the LMS, or within the announced deadline in the LMS, whichever is earlier. Otherwise the marks awarded to you will be final and will be used to calculate your final weighted average score for the unit.

Errata

This page is subject to change based on errors found. Any errors found and corrected will be posted in the Announcements page of the LMS. If you print out a copy of this page, please follow the news in the Announcements page of the LMS for any updates.