

## ICT582 ASSIGNMENT CHECKLIST

**Surname (Family Name):** Phuntsho

**Given Names:** Tashi

**Student Number:** 34907934

**Tutor's Name:** Aaron Thomson

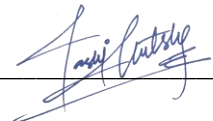
**Tutorial group:** Day: Tuesday Start time 11:30AM Venue: 245.3.062 Group ID \_\_\_\_\_

**Assignment Due Date:** 4/11/2023 **Date Submitted:** 4/11/2023

**Your assignment should meet the following requirements. Please confirm this (by ticking boxes) before submitting your assignment.**

- ☒ All details above are complete.
- ☒ The zip file contains the file `Assignment.pdf`. The documentation was prepared according to Documentation Requirements specified in Assignment's question sheet.
- ☒ The zip file includes the code for the three questions. The code for each question is stored under a separate directory named q1, q2 and q3 respectively.
- ☒ You understand that the zip file must be submitted to ICT582 Unit LMS.
- ☒ You have kept a copy of this assignment, including the zip file.

**I declare that the work included in this submission is completed by myself independently.**

Signature  Date 4/11/2023

## Contents

Question 1: A Simple Sales Management System .....	3
Discussion of Solution: .....	3
Self-diagnosis and Evaluation:.....	4
Test Evidence: .....	5
Feature 1: .....	5
Feature 2: .....	6
Feature 3: .....	8
Feature 4: .....	9
Feature 5: .....	9
Feature 6: .....	10
Feature 7: .....	11
Feature 8: .....	12
List of File Names:.....	13
Question 2: Load Sales Records from Files and Saves Them to Files.....	13
Discussion of Solution: .....	13
Self-diagnosis and Evaluation:.....	13
Test Evidence: .....	14
Feature 1: .....	14
Feature 2: .....	15
Feature 3: .....	17
Feature 4: .....	18
List of File Names:.....	20
Question 3: Display Sales Performance Graphically .....	20
Discussion of Solution: .....	20
Self-diagnosis and Evaluation:.....	20
Test Evidence: .....	21
Feature 1: .....	21
Feature 2: .....	22
Feature 3: .....	22
Feature 4: .....	23
List of File Names:.....	26

## Question 1: A Simple Sales Management System

### Discussion of Solution:

A simple sales management system for Western Wholesales Pty. Ltd. Initially consists of features such as adding new customers and transactions associated with the customers, searching for customer details and transactions, viewing the transaction history of a customer, and deleting customers and transactions. The system has been designed by grouping related modules into different files.

The solution requires the utilization of 2 data structures that can be used to store customer and transaction information by forming a relationship between them. Python provides many options for data structures, such as lists, dictionaries, and tuples.

As a storage for the sales management system, I have chosen to use two lists, which will have dictionaries as their elements. A dictionary can hold elements with key and value pairs, making it suitable for achieving the required task. With the methods available with a list and dictionary, performing operations such as adding new records, searching, and deleting can be easily accomplished.

When adding a new customer or a new transaction, information is provided by the user. To ensure valid data is stored, the input from the user is validated using a function. This function takes parameters like label for the input, Boolean for whether the input field is required or not, and type of data required for the input.

The program can perform a search operation for customers and transactions based on provided partial data. Any search keyword provided by the user is checked with the respective data store to retrieve the record that contains the keyword. The feature utilizes the concept of nested loops and case conversion to make the search case-insensitive.

The horizontal visual layout of the menu items is achieved using the f string format in Python, which provides various options to format a string.

The above-discussed approaches and concepts provide enough ability to meet and fulfill all the requirements of question 1.

### Self-diagnosis and Evaluation:

#	Requirements for Question 1	Status
1	<p>Adding new customers by getting the customer name, postcode, and mobile number from the user.</p> <ul style="list-style-type: none"><li>• Customer ID is unique and autogenerated.</li><li>• Customer name is required.</li><li>• Postcode and mobile number can be optional.</li></ul>	Fully Completed
2	<p>Adding new transactions for a given customer:</p> <ul style="list-style-type: none"><li>• Check if the customer exists.</li><li>• Transaction ID unique and autogenerated.</li><li>• Record date, category, and value of transaction</li></ul>	Fully Completed
3	<p>Search customers using single search string:</p> <ul style="list-style-type: none"><li>• Case-insensitive search</li><li>• Allow partial matches.</li><li>• Compare search string to id, name, postcode, and phone number.</li></ul>	Fully Completed
4	<p>Search transactions using single search string:</p> <ul style="list-style-type: none"><li>• Case-insensitive search</li><li>• Allow partial matches.</li><li>• Compare search string to customer id, date, category, and value</li></ul>	Fully Completed
5	<p>Display the sales transactions for a given customer using the id of that customer.</p>	Fully Completed
6	<p>Delete a transaction record with a given transaction ID.</p>	Fully Completed
7	<p>Delete a customer with a given customer ID together with all transaction records due to this customer.</p>	Fully Completed
8	<p>Quit to quit program</p>	Fully Completed

## Test Evidence:

### Feature 1:

Adding new customers by getting the customer name, postcode, and mobile number from the user.

#### Testcase 1:

Customer Name is a required field while adding a new customer.

In addition to being required, the program also ensures that the customer's name meets following requirements:

1. Can only contain alphabets, spaces, and hyphen.
2. Cannot start with spaces or hyphen.
3. Cannot end with hyphen.

#### Evidence:

```
Add New Customer
-----
Add new customer by providing the following details.
Customer Name:
Customer Name is required to proceed further.
Customer Name: l23test
Customer Name can only contain alphabets, spaces and hyphen. It can not begin with space or hyphen and can not end with hyphen.
Customer Name: tashi -
Customer Name can only contain alphabets, spaces and hyphen. It can not begin with space or hyphen and can not end with hyphen.
Customer Name: - astds
Customer Name can only contain alphabets, spaces and hyphen. It can not begin with space or hyphen and can not end with hyphen.
Customer Name: Tashi Phuntsho
Postcode: |
```

Status: The evidence proves that tested feature meets the requirement.

#### Testcase 2:

Postcode and Mobile Number can be optional. The program also validates that if postcode or mobile number is provided, they must be of numeric value.

#### Evidence:

---

Add New Customer

---

Add new customer by providing the following details.

Customer Name: Tashi Phuntsho

Postcode: test

Postcode needs to be a numeric values.

Postcode:

Phone Number: test2

Phone Number needs to be a numeric values.

Phone Number: 045122222

Status: The evidence proves that tested feature meets the requirement.

### Testcase 3:

Customer ID is unique and autogenerated. The program autogenerates an incrementing positive integer starting from 100000 as customer ID and assign it to the new customer.

#### Evidence:

---

Add New Customer

---

Add new customer by providing the following details.

Customer Name: Tashi

Postcode:

Phone Number:

[system notification]: New customer ( Tashi ) has been added with Customer ID: 100000

---

M A I N M E N U

---

1. Add New Customer

2. Record New Transaction

3. Search Customer

4. Search Transaction

5. Display Transaction History

6. Delete Transaction Record

7. Delete Customer

8. Quit

---

Please select your option from [1-8] or type 'quit' to exit the program.

Your Option: 1

---

Add New Customer

---

Add new customer by providing the following details.

Customer Name: Phuntsho

Postcode: 6060

Phone Number: 0412343343

[system notification]: New customer ( Phuntsho ) has been added with Customer ID: 100001

Status: The evidence proves that tested feature meets the requirement.

### Feature 2:

Adding new transactions for a given customer:

### Testcase 1:

Check if the customer exists in the customer database by getting the customer ID from the user. In addition, user can use 'help' to view the ID and Name of all the customers.

### Evidence:

#### Record New Transaction

```
-----
If you are not sure of Customer ID, type "help" to view the customer list.
Type 'exit' to go back to Main Menu.
Enter Customer ID to add transaction record: 100
[system notification]: The entered Customer ID was not found in the Customer Database. Please check the Customer ID.
Type 'exit' to go back to Main Menu.
Enter Customer ID to add transaction record:
[system notification]: Please enter valid Customer ID.

Type 'exit' to go back to Main Menu.
Enter Customer ID to add transaction record: help
Viewing Customer(s) From Customer Database
Customer ID: Name
    100000: Tashi
    100001: Phuntsho
Type 'exit' to go back to Main Menu.
Enter Customer ID to add transaction record: 100000
Transaction Date [yyyy-mm-dd]: |
```

Status: The evidence proves that tested feature meets the requirement.

### Testcase 2:

Record date, category, and value of transaction.

- All the fields are required.
- Date should be in 'yyyy-mm-dd' format.
- Category should be one of the 6 available categories.
- Value should be numeric or decimal.

### Evidence:

```
Transaction Date [yyyy-mm-dd]: 10/10/2023
Transaction Date [yyyy-mm-dd] needs to be a valid date in specified format.
Transaction Date [yyyy-mm-dd]: 2023-11-01
Transaction Category: test
Transaction Category must be one of these available category ['food', 'alcohol and beverage', 'apparel', 'furniture',
'household appliances','computer equipment']
Transaction Category: Food
Transaction Value: test
Transaction Value needs to be a numeric or decimal value.
Transaction Value:
Transaction Value is required to proceed further.
Transaction Value: 123.12
[system notification]: New transaction has been recorded in the transaction database successfully.
.
```

Status: The evidence proves that tested feature meets the requirement.

### Testcase 3:

Transaction ID unique and autogenerated. The system generates Transaction ID starting from 100000000 (as per the data in Transactions.csv file)

Evidence:

ID	Customer ID	Date	Category	Value
100000000	100000	2023-11-01	Food	123.12
100000001	100000	2023-11-02	apparel	3212.0

Status: The evidence proves that tested feature meets the requirement.

Feature 3:

Search customers using single search string.

Testcase 1:

Customer search feature should support:

- Case-insensitive search
- Allow partial matches.
- Compare search string to id, name, postcode, and phone number.
- If keyword is not provided, the program will retrieve information of all the customers.

Evidence:

```
Customer Database Search
-----
Type in a keyword to search the customer database.

Type 'exit' to go back to Main Menu.
Search Keyword: tetet
[system notification]: Your search keyword did not match anything in the customer database. Please try with a different keyword.

Type 'exit' to go back to Main Menu.
Search Keyword: ta
Search Result

ID      Name      Postcode  Mobile Number
100000  Tashi      6060      0455434567

Type 'exit' to go back to Main Menu.
Search Keyword: 707
Search Result

ID      Name      Postcode  Mobile Number
100001  Phuntsho  7072      0438274653

Type 'exit' to go back to Main Menu.
Search Keyword:
Search Result

ID      Name      Postcode  Mobile Number
100000  Tashi      6060      0455434567
100001  Phuntsho  7072      0438274653

Type 'exit' to go back to Main Menu.
Search Keyword: |
```



Status: The evidence proves that tested feature meets the requirement.

#### Feature 4:

Search customers using single search string.

##### Testcase 1:

Search transactions using single search string:

- Case-insensitive search
- Allow partial matches.
- Compare search string to customer id, date, category, and value
- If keyword is not provided, the program will retrieve information of all the transactions.

##### Evidence:

```
Transaction Database Search
-----
Type in a keyword to search the transaction database.

Type 'exit' to go back to Main Menu.
Search Keyword: sadsad
[system notification]: Your search keyword did not match anything in the transaction database. Please try with a different keyword.

Type 'exit' to go back to Main Menu.
Search Keyword: foo
Search Result

ID          Customer ID      Date Category      Value
1000000000  100000      2023-10-01 food      1234.0

Type 'exit' to go back to Main Menu.
Search Keyword: 2023-10
Search Result

ID          Customer ID      Date Category      Value
1000000000  100000      2023-10-01 food      1234.0

Type 'exit' to go back to Main Menu.
Search Keyword: 100000
Search Result

ID          Customer ID      Date Category      Value
1000000000  100000      2023-10-01 food      1234.0
1000000001  100000      2023-08-10 computer equipment  4000.0
1000000002  100001      2023-11-01 furniture    432.0

Type 'exit' to go back to Main Menu.
Search Keyword: f
Search Result

ID          Customer ID      Date Category      Value
1000000000  100000      2023-10-01 food      1234.0
1000000002  100001      2023-11-01 furniture    432.0
```

Status: The evidence proves that tested feature meets the requirement.

#### Feature 5:

Display the sale transactions for a given customer using the id of that customer.

Testcase 1:
<p>Display the sale transactions for a given customer using the id of that customer.:</p> <ul style="list-style-type: none"> <li>• Get the Customer ID from the user.</li> <li>• Check if the customer exists.</li> <li>• Display transaction history if found.</li> </ul>
<p>Evidence:</p> <pre> Display Transaction History ----- View transaction history of a customer by entering Customer ID. If you are not sure of Customer ID, type "help" to view the customer list.  Type 'exit' to go back to Main Menu. Customer ID: 1000 [system notification]: The Customer ID did not match with any customer in the database.  Type 'exit' to go back to Main Menu. Customer ID: [system notification]: The Customer ID did not match with any customer in the database.  Type 'exit' to go back to Main Menu. Customer ID: 100000  Customer Information ----- Customer ID: 100000 Name: Tashi Postcode: Mobile Number:  Transaction History ----- ID          Date       Category  Value 1000000000  2023-11-01  Food      123.12 1000000001  2023-11-02  apparel   3212.0 </pre>
Status: The evidence proves that tested feature meets the requirement.

#### Feature 6:

Delete a transaction record with a given transaction ID.

Testcase 1:
<p>Delete a transaction record with a given transaction ID.</p> <ul style="list-style-type: none"> <li>• Ask user for the transaction ID.</li> </ul>

- Asks for user confirmation if they really intend to delete the Transaction record.

#### Evidence:

```

Delete Transaction Record
-----
Delete transaction from the transaction list by entering Transaction ID.
If you are not sure of Transaction ID, type "help" to view the transaction list.

Type 'exit' to go back to Main Menu.
Enter the Transaction ID to Delete:
[system notification]: Please enter valid Transaction ID.

Type 'exit' to go back to Main Menu.
Enter the Transaction ID to Delete: 100
[system notification]: Delete operation canceled.

Type 'exit' to go back to Main Menu.
Enter the Transaction ID to Delete: 100000000
[system notification]: You are about to delete following transaction:
[system notification]: {'transaction_id': 100000000, 'customer_id': 100000, 'date': '2023-11-01', 'category': 'Food', 'value': 123.12}
Are you sure? [y | yes for yes, anything else for no]: yes
[system notification]: Transaction has been deleted successfully.

Type 'exit' to go back to Main Menu.
Enter the Transaction ID to Delete: 1000000001
[system notification]: You are about to delete following transaction:
[system notification]: {'transaction_id': 1000000001, 'customer_id': 100000, 'date': '2023-11-02', 'category': 'apparel', 'value': 3212.0}
Are you sure? [y | yes for yes, anything else for no]: no
[system notification]: Delete operation canceled.

```

Status: The evidence proves that tested feature meets the requirement.

#### Feature 7:

Delete a customer with a given customer ID together with all transaction records due to this customer.

#### Testcase 1:

Delete a customer with a given customer ID together with all transaction records due to this customer.

- Ask user for the Customer ID.
- Asks for user confirmation if they really intend to delete the customer record.
- Remove the customer and transactions if confirmed.

#### Evidence:

```
-----
Delete customer from customer database and remove all the transaction associated with the customer.
If you are not sure of Customer ID, type "help" to view the customer list.
```

```
Type 'exit' to go back to Main Menu.
```

```
Enter the Customer ID to Delete: 100000
```

```
[system notification]: You are about to delete following customer and transaction details:
```

```
Customer Information
```

```
-----
Customer ID: 100000
```

```
Name: Tashi
```

```
Postcode:
```

```
Mobile Number:
```

```
Transaction Details
```

```
-----
ID      Date      Category
```

```
100000001 2023-11-02      apparel
```

```
Are you sure? [y | yes for yes, anything else for no]: no
```

```
[system notification]: Delete operation has been canceled.
```

```
Type 'exit' to go back to Main Menu.
```

```
Enter the Customer ID to Delete: 100000
```

```
[system notification]: You are about to delete following customer and transaction details:
```

```
Customer Information
```

```
-----
Customer ID: 100000
```

```
Name: Tashi
```

```
Postcode:
```

```
Mobile Number:
```

```
Transaction Details
```

```
-----
ID      Date      Category
```

```
100000001 2023-11-02      apparel
```

```
Are you sure? [y | yes for yes, anything else for no]: yes
```

```
[system notification]: Customer has been deleted successfully.
```

```
-----
Status: The evidence proves that tested feature meets the requirement.
```

Feature 8:

Quit to quit program

Testcase 1:

The program will exit when the user type quit or the associated number.

Evidence:

```
-----
M A I N    M E N U
```

```
1. Add New Customer
```

```
2. Record New Transaction
```

```
3. Search Customer
```

```
4. Search Transaction
```

```
5. Display Transaction History
```

```
6. Delete Transaction Record
```

```
7. Delete Customer
```

```
8. Quit
```

```
-----
Please select your option from [1-8] or type 'quit' to exit the program.
```

```
Your Option: Quit
```

```
-----
Thank you for using the system of Western Wholesales Pty Ltd.
```

```
Logging out.....
```

```
|
```

Status: The evidence proves that tested feature meets the requirement.

### List of File Names:

Under the folder q1, following files are present:

- common.py
- customer.py
- q1.py
- transaction.py

## Question 2: Load Sales Records from Files and Saves Them to Files

### Discussion of Solution:

The system needs features of importing/exporting customer details and transaction records from/to CSV files. When importing the user will specify the location and the file name and the program will check if the csv file exists. If the file exists, the system will also check the size of the specified file before importing. This feature uses os module of python to get the file size. The entries in the csv file are read and validated before being imported. Working with csv file is accomplished through the use of csv module and csv.DictReader is used to read the records of the csv file. This provide a convenient mean of accessing the value as the records are retrieved as dictionaries from the file.

The Export features allow the export of customer details or transactions to a specified file. These features take the path/file name as input from the user and checks if the specified file already exists in that location. These also provide user with an option to overwrite the file, if the file already exists. An additional feature is, when the user specifies a file name, if the program detects that the file name does not have “.csv” in the end, the program will append “.csv” to the file name.

### Self-diagnosis and Evaluation:

#	Requirements for Question 2	Status
1	Loading the customer records from csv file. <ul style="list-style-type: none"><li>• Prompt user to provide the file path.</li><li>• Check for duplicates of customer records.</li></ul>	Fully Completed

	<ul style="list-style-type: none"> <li>• Ignore the duplicates.</li> </ul>	
2	<p>Save all customer records from the memory to a CSV file:</p> <ul style="list-style-type: none"> <li>• Prompt the user to provide a file path.</li> <li>• If the file does not exist, create a new file.</li> <li>• 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.</li> <li>• Save the customer records to CSV file.</li> </ul>	Fully Completed
3	<p>Load transaction records from a CSV file:</p> <ul style="list-style-type: none"> <li>• Add new records to the those in the memory.</li> <li>• The IDs of new transactions to be re-generated to avoid clashes with those of the existing transactions.</li> <li>• For each transaction, check whether the customer exists in the memory.</li> <li>• Reject those transactions from unknown customers.</li> </ul>	Fully Completed
4	<p>Save all transaction records from the memory to a CSV file:</p> <ul style="list-style-type: none"> <li>• Prompt the user to enter a file path.</li> <li>• If the file does not exist, create a new file.</li> <li>• 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.</li> <li>• Save the transactions records to that CSV file.</li> </ul>	Fully Completed

### Test Evidence:

#### Feature 1:

Loading the customer records from csv file.

Testcase 1:
<p>Loading the customer records from csv file.</p> <ul style="list-style-type: none"> <li>• Prompt user to provide the file path.</li> <li>• Check for duplicates of customer records.</li> <li>• Ignore the duplicates.</li> </ul>

### Evidence:

```
-----
Load customer records from a .csv file by specifying file path.
The csv file should contain a header in the sequence provided below:
['customer_id', 'name', 'postcode', 'phone number']

Type 'exit' to go back to Main Menu.
Specify File Location: cus.csv
[system notification]: System could not locate the specified file. Please check the file path and try again.

Type 'exit' to go back to Main Menu.
Specify File Location: Transactions.csv
[system notification]: The format of the customer records does not meet the requirement. Please check the file.

Type 'exit' to go back to Main Menu.
Specify File Location:
[system notification]: System could not locate the specified file. Please check the file path and try again.

Type 'exit' to go back to Main Menu.
Specify File Location: Customers less.csv
[system notification]: Customer with same ID already exist in the database. {'customer_id': '100000', 'name': 'Chris', '43908827'} has been ignored.
[system notification]: Customer with same ID already exist in the database. {'customer_id': '100001', 'name': 'Bria', ': '48531388'} has been ignored.
.....
[system notification]: 12 out of 14 customer record(s) imported successfully from the csv file.
```

Status: The evidence proves that tested feature meets the requirement.

### Feature 2:

Save all customer records from the memory to a CSV file.

#### Testcase 1:

Save all customer records from the memory to a CSV file:

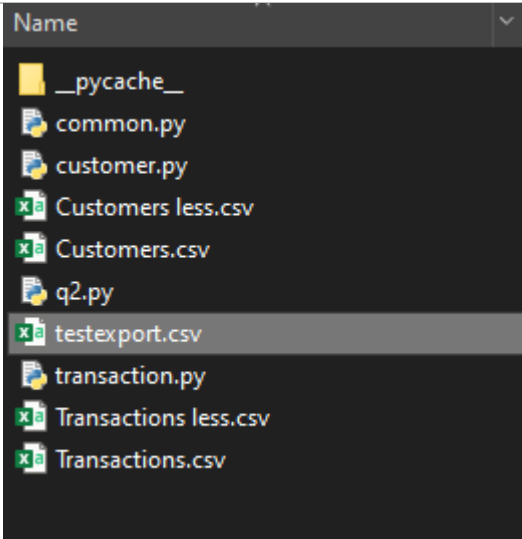
- Prompt the user to provide a file path.
- If the file does not exist, create a new file.
- Save the customer records to CSV file.

### Evidence:

```
Export Customer Records
-----
Export customer records to a .csv file.

Type 'exit' to go back to Main Menu.
Specify File Location: testexport.csv
[system notification]: Customer Records have been exported successfully to testexport.csv

Type 'exit' to go back to Main Menu.
Specify File Location: |
```



Status: The evidence proves that tested feature meets the requirement.

#### Testcase 2:

Save all customer records from the memory to a CSV file:

- Prompt the user to provide a file path.
- 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.
- Save the customer records to CSV file.

#### Evidence:

```
Type 'exit' to go back to Main Menu.  
Specify File Location: testexport.csv
```

```
File with same name already exist at the specified location.  
You can either overwrite the existing file, change the name or location of the file or cancel the export operation  
[OW - overwrite the file]          [CNL - change name or location of the file]      [CE - cancel export]  
Your option: ow  
[system notification]: Customer Records have been exported successfully to testexport.csv
```

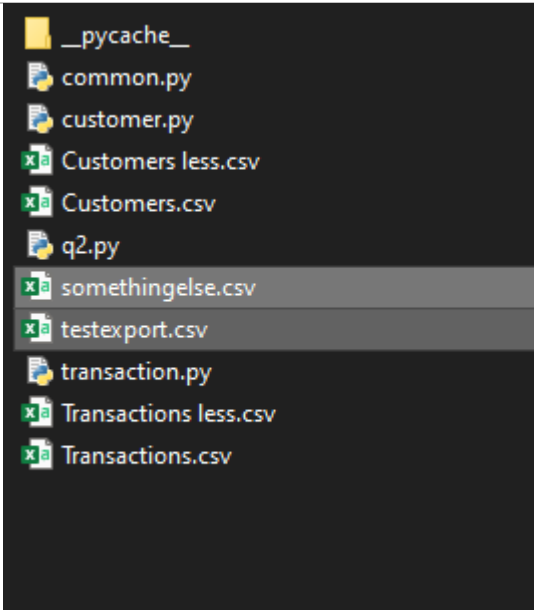
```
Type 'exit' to go back to Main Menu.  
Specify File Location: testexport
```

```
File with same name already exist at the specified location.  
You can either overwrite the existing file, change the name or location of the file or cancel the export operation  
[OW - overwrite the file]          [CNL - change name or location of the file]      [CE - cancel export]  
Your option: cnl  
[system notification]: Please provide a different file name or location.
```

```
Type 'exit' to go back to Main Menu.  
Specify File Location: somethingelse  
[system notification]: Customer Records have been exported successfully to somethingelse.csv
```

```
Type 'exit' to go back to Main Menu.  
Specify File Location: |
```





Status: The evidence proves that tested feature meets the requirement.

### Feature 3:

Load transaction records from a CSV file.

#### Testcase 1:

Load transaction records from a CSV file:

- Add new records to the those in the memory.
- The IDs of new transactions to be re-generated to avoid clashes with those of the existing transactions.
- For each transaction, check whether the customer exists in the memory.
- Reject those transactions from unknown customers.

Evidence:

```

Type 'exit' to go back to Main Menu.
Specify File Location: ttt
[system notification]: System could not locate the specified file. Please check the file path and try again.

Type 'exit' to go back to Main Menu.
Specify File Location: Customers.csv
[system notification]: The format of the transaction records does not meet the requirement. Please check the file.

Type 'exit' to go back to Main Menu.
Specify File Location: Transactions less.csv
.....[system notification]: Customer ID does not exist in the database. {'date': '2/11/2020', 'transaction_id': '100000044', 'customer_id': '100000044', 'category': 'computer equipment', 'value': '4064.88'} has been ignored.
.[system notification]: Customer ID does not exist in the database. {'date': '2/25/2020', 'transaction_id': '100000053', 'customer_id': '100000053', 'category': 'furniture', 'value': '3059.88'} has been ignored.
..[system notification]: Customer ID does not exist in the database. {'date': '3/14/2020', 'transaction_id': '100000068', 'customer_id': '100000068', 'category': 'furniture', 'value': '1989.74'} has been ignored.
....[system notification]: Customer ID does not exist in the database. {'date': '5/2/2020', 'transaction_id': '100000110', 'customer_id': '100000110', 'category': 'computer equipment', 'value': '1313.75'} has been ignored.
..[system notification]: Customer ID does not exist in the database. {'date': '8/31/2020', 'transaction_id': '100000210', 'customer_id': '100000210', 'category': 'computer equipment', 'value': '560.12'} has been ignored.
....[system notification]: Customer ID does not exist in the database. {'date': '11/3/2020', 'transaction_id': '100000276', 'customer_id': '100000276', 'category': 'furniture', 'value': '6744.75'} has been ignored.
.[system notification]: Customer ID does not exist in the database. {'date': '11/20/2020', 'transaction_id': '100000293', 'customer_id': '100000293', 'category': 'alcohol and beverage', 'value': '2748.91'} has been ignored.
.....[system notification]: Customer ID does not exist in the database. {'date': '8/12/2021', 'transaction_id': '10000054', 'customer_id': '10000054', 'category': 'alcohol and beverage', 'value': '6338.39'} has been ignored.
.....[system notification]: Customer ID does not exist in the database. {'date': '6/7/2022', 'transaction_id': '100000044', 'customer_id': '100000044', 'category': 'alcohol and beverage', 'value': '2198.76'} has been ignored.
.[system notification]: Customer ID does not exist in the database. {'date': '6/27/2022', 'transaction_id': '1000000128', 'customer_id': '1000000128', 'category': 'furniture', 'value': '3485.57'} has been ignored.
.[system notification]: Customer ID does not exist in the database. {'date': '8/3/2022', 'transaction_id': '1000000162', 'customer_id': '1000000162', 'category': 'furniture', 'value': '2197.94'} has been ignored.
....[system notification]: Customer ID does not exist in the database. {'date': '10/9/2022', 'transaction_id': '1000000212', 'customer_id': '1000000212', 'category': 'furniture', 'value': '1151.68'} has been ignored.
....
[system notification]: 69 out of 81 transaction record(s) imported successfully from the csv file.

```

Status: The evidence proves that tested feature meets the requirement.

#### Feature 4:

Save all transaction records from the memory to a CSV file:

##### Testcase 1:

Save all transaction records from the memory to a CSV file:

- Prompt the user to enter a file path.
- If the file does not exist, create a new file.
- Save the transactions records to that CSV file.

##### Evidence:

Export Transaction Records

Export transaction records to a .csv file.

```

Type 'exit' to go back to Main Menu.
Specify File Location: transactionsexport.csv
[system notification]: Transaction Records have been exported successfully to transactionsexport.csv

```

```

Type 'exit' to go back to Main Menu.
Specify File Location: |

```

Status: The evidence proves that tested feature meets the requirement.

#### Testcase 2:

Save all transaction records from the memory to a CSV file:

- Prompt the user to enter a file path.
- 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.
- Save the transactions records to that CSV file.

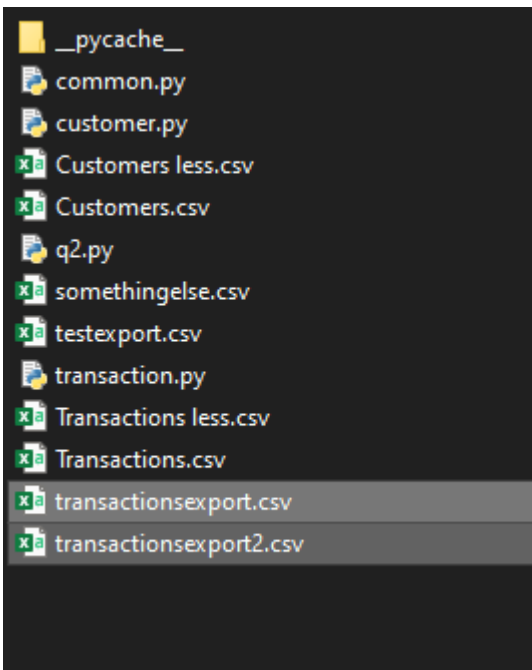
#### Evidence:

```
Type 'exit' to go back to Main Menu.
Specify File Location: transactionsexport.csv

File with same name already exist at the specified location.
You can either overwrite the existing file, change the name or location of the file or cancel the export operation.
[OW - overwrite the file]          [CNL - change name or location of the file]      [CE - cancel export]
Your option: cnl
[system notification]: Please provide a different file name or location.

Type 'exit' to go back to Main Menu.
Specify File Location: transactionsexport2
[system notification]: Transaction Records have been exported successfully to transactionsexport2.csv

Type 'exit' to go back to Main Menu.
Specify File Location: |
```



Status: The evidence proves that tested feature meets the requirement.

### List of File Names:

Under the folder q2, following files are present:

- common.py
- customer.py
- Customers.csv
- q2.py
- transaction.py
- Transactions.csv

## Question 3: Display Sales Performance Graphically

### Discussion of Solution:

To improve the effectiveness of the program, data structures used to store customer records and transactions are replaced with ndarray. I have chosen to implement structured array to store the records of customers and transactions because I can define the type of data that is going to be stored and accessing the values of the structured array is made easier through the use of column names like a dictionary.

The visualization of the sales and transaction counts are accomplished using matplotlib and numpy modules. Since the sale values tend to be very high compared to the transaction counts, the program uses a function to scale down the sale values in orders of 10, 100, or 1000 based on the highest sale value present. When plotting a graph, the transaction database is sent as a parameter to a function (get\_months) which will extract all unique year and month and return an array. This array is used as the x axis for plotting the graph. The corresponding value of y is determined by passing the x to a function get\_Y which calculates the sale values and transaction counts and returns a ndarray for plotting.

### Self-diagnosis and Evaluation:

#	Requirements for Question 3	Status
1	Replace the data structures used in q1 and q2 with ndarray.	Fully Completed
2	Display the monthly sales values and transaction numbers with two line graphs in one axes. The line graph must have: <ul style="list-style-type: none"><li>• Appropriate title.</li></ul>	Fully Completed

	<ul style="list-style-type: none"> <li>Labels for X axis and Y axis</li> <li>Legend</li> </ul>	
3	<p>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:</p> <ul style="list-style-type: none"> <li>Appropriate title.</li> <li>Labels for X axis and Y axis</li> <li>Legend</li> </ul>	Fully Completed
4	<p>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:</p> <ul style="list-style-type: none"> <li>Appropriate title.</li> <li>Labels for X axis and Y axis</li> <li>Legend</li> </ul>	Fully Completed

### Test Evidence:

#### Feature 1:

Replace the data structures used in q1 and q2 with ndarray.

Testcase 1:
<p>Replace the data structures used in q1 and q2 with ndarray.</p> <p>The program has been adjusted to use ndarray (structured array) as the data store.</p>
<p>Evidence:</p> <pre>c_dtype = np.dtype([('customer_id','int64'), ('name','U20'), ('postcode','U20'), ('phone number','U20')]) t_dtype = np.dtype([('transaction_id','int64'), ('customer_id','int64'), ('date','datetime64[D]'), ('category','U20'), ('value', float)]) customer_db = np.empty(0, dtype=c_dtype) transaction_db = np.empty(0, dtype=t_dtype) .....</pre>
Status: The evidence proves that tested feature meets the requirement.

### Feature 2:

Display the monthly sales values and transaction numbers with two line graphs in one axes.

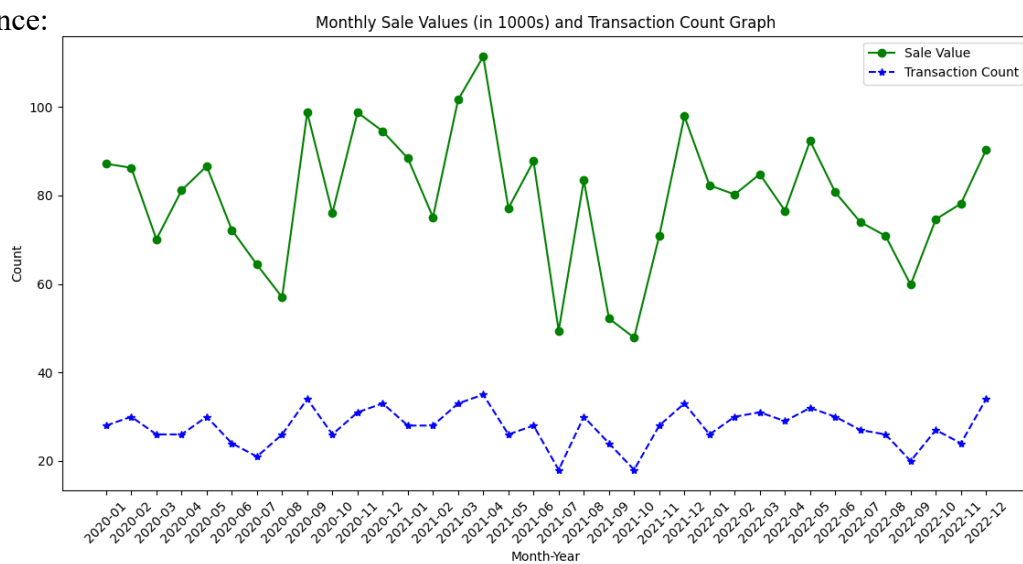
#### Testcase 1:

Display the monthly sales values and transaction numbers with two line graphs in one axes.

The line graph must have:

- Appropriate title.
- Labels for X axis and Y axis
- Legend

#### Evidence:



Status: The evidence proves that tested feature meets the requirement.

### Feature 3:

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.

#### Testcase 1:

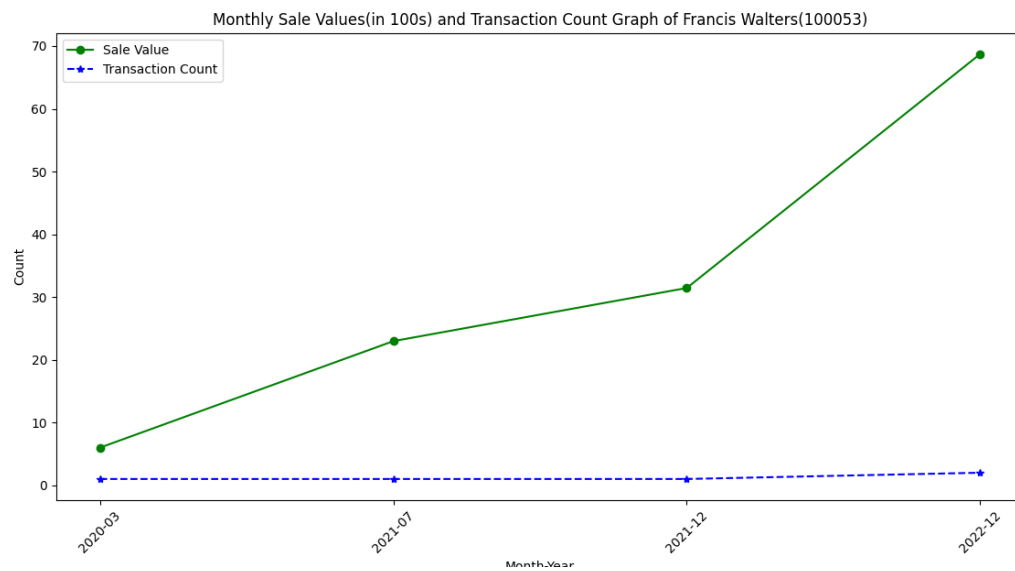
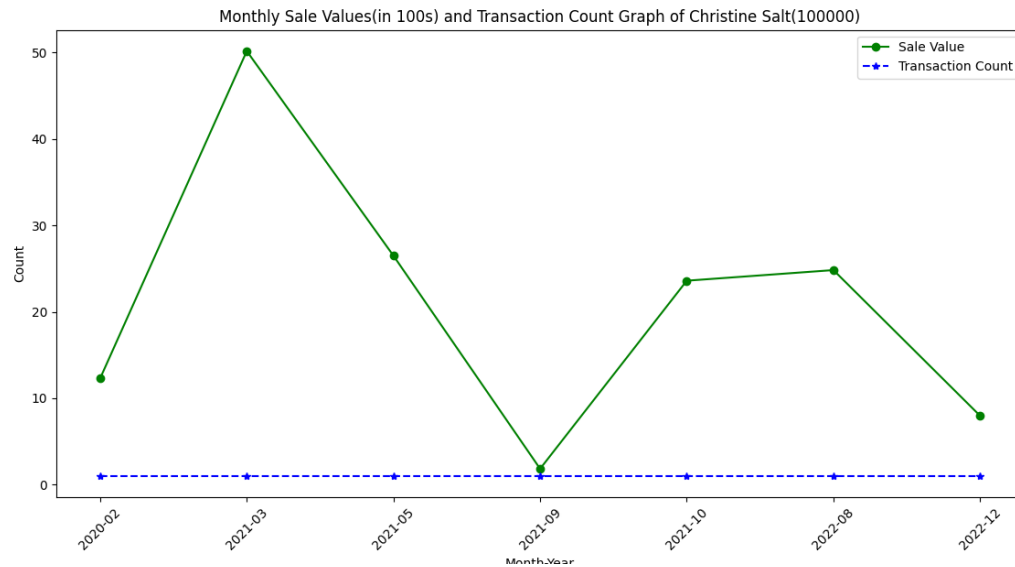
For a given customer, display the monthly sales values and the number of transactions generated by the customer using two line graphs in one axes.

The line graph must have:

- Appropriate title.
- Labels for X axis and Y axis

- Legend

Evidence:



Status: The evidence proves that tested feature meets the requirement.

#### Feature 4:

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.

Testcase 1:

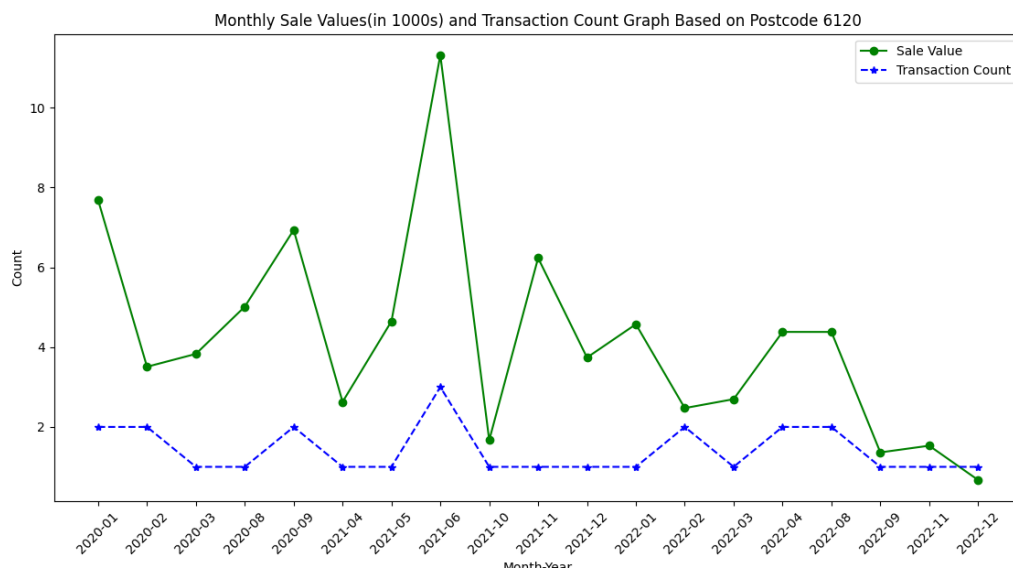
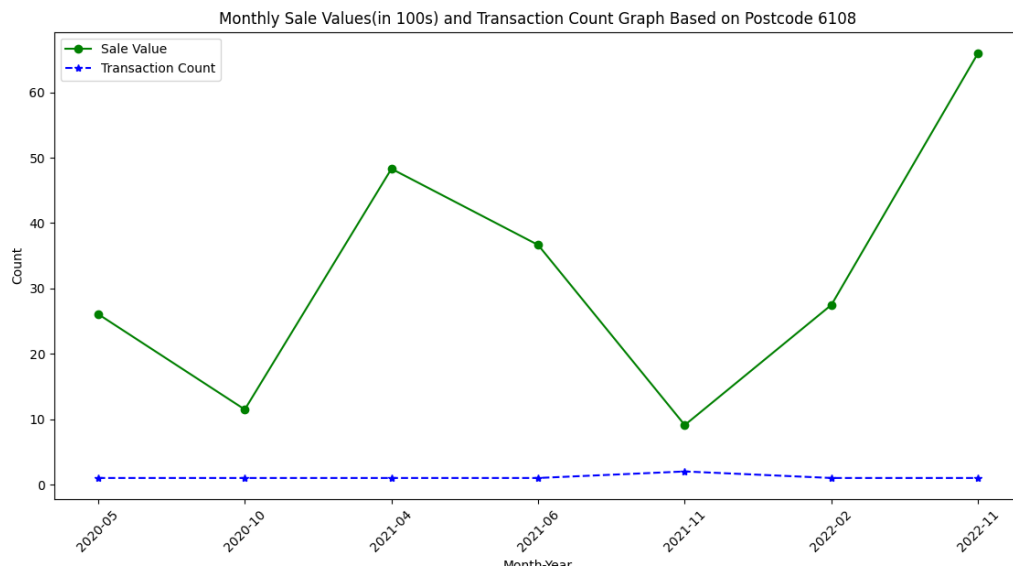
For a given postcode, display the monthly sales values and the number of transactions generated by the customers located in the postcode area using two line graphs in one axes.

The line graph must have:

- Appropriate title.
- Labels for X axis and Y axis
- Legend

Evidence:







### List of File Names:

Under the folder q3, following files are present:

- common.py
- customer.py
- Customers.csv
- q3.py
- transaction.py
- Transactions.csv
- visual.py