Contents

[Solution Overview 2](#_Toc168177619)

[Self-diagnosis and Evaluation for question 1: 3](#_Toc168177620)

[Test Evidence: 3](#_Toc168177621)

[Self-diagnosis and Evaluation for question 2: 6](#_Toc168177622)

[Test Evidence: 7](#_Toc168177623)

[Display Sales Performance Graphically 12](#_Toc168177624)

[Discussion of the solution 12](#_Toc168177625)

[Self-diagnosis and Evaluation for question 3: 12](#_Toc168177626)

# Solution Overview

A basic sales management system has been developed for Western Wholesales Pty. Ltd. The initial features of this system include adding new customers and their associated transactions, searching for customer and transaction details, viewing a customer’s transaction history, and deleting customers and transactions. The system’s design organizes related modules into separate files.

The solution necessitates the use of two data structures to store customer and transaction information, establishing a relationship between them. Python offers a variety of data structure options, including lists, dictionaries, and tuples.

For the sales management system’s storage, I’ve opted to use two lists, each containing dictionaries as elements. A dictionary can hold elements in key-value pairs, making it ideal for this task. The methods available with a list and dictionary make operations like adding new records, searching, and deleting straightforward.

When a new customer or transaction is added, the user provides the information. To ensure the storage of valid data, a function validates the user’s input. This function accepts parameters such as the input label, a Boolean indicating whether the input field is required, and the type of data required for the input.

The program can execute a search operation for customers and transactions based on partial data provided. Any search keyword given by the user is compared with the respective data store to retrieve the record containing the keyword. This feature employs nested loops and case conversion to make the search case insensitive.

The horizontal visual layout of the menu items is achieved using Python’s f-string format, which offers various string formatting options.

# Self-diagnosis and Evaluation for question 1:

|  |  |  |
| --- | --- | --- |
| Sl no | Requirement | Status |
| 1 | Loading customer and sales records from two CSV files and store them in a single data structure. | Fully  Completed |
| 2 | Saving customers and sales record from memory. | Fully  Completed |

## Test Evidence:

Feature 1: This feature loads the customers and sales record from csv file.

Testcase for loading customer csv file



Testcase for loading sales record.

A close-up of a computer screen

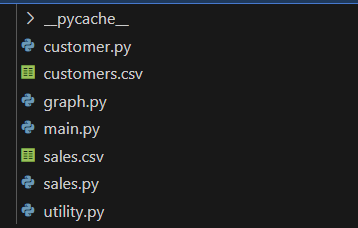
Description automatically generated

Features 2: Saving customers and sales record.

Testcase for saving customer and sales record.

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

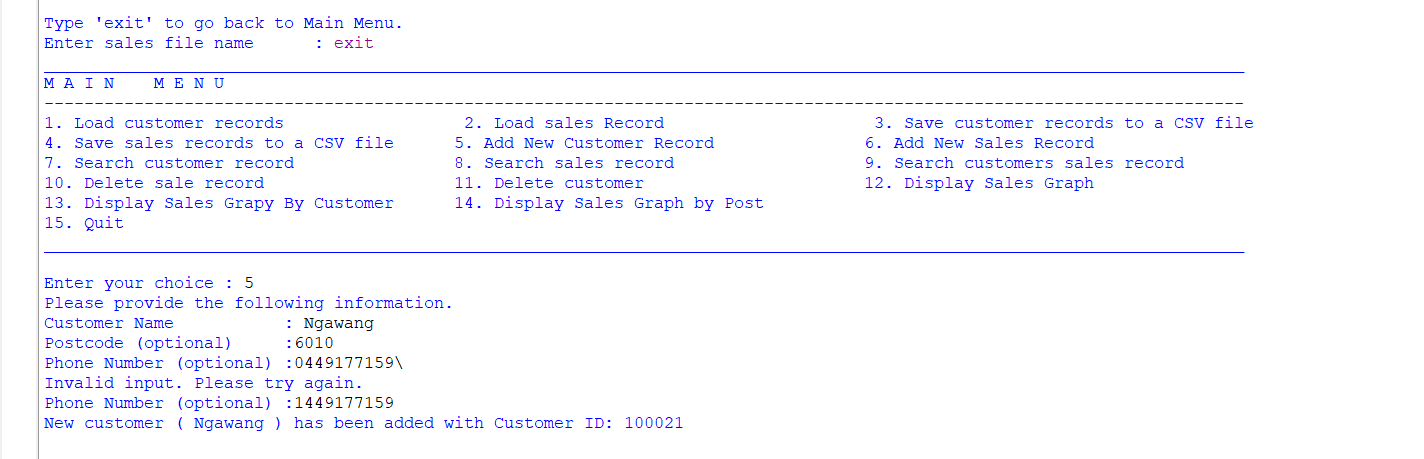
Above two image shows before and after funning the program to show that the feature meets the requirement.

# Self-diagnosis and Evaluation for question 2:

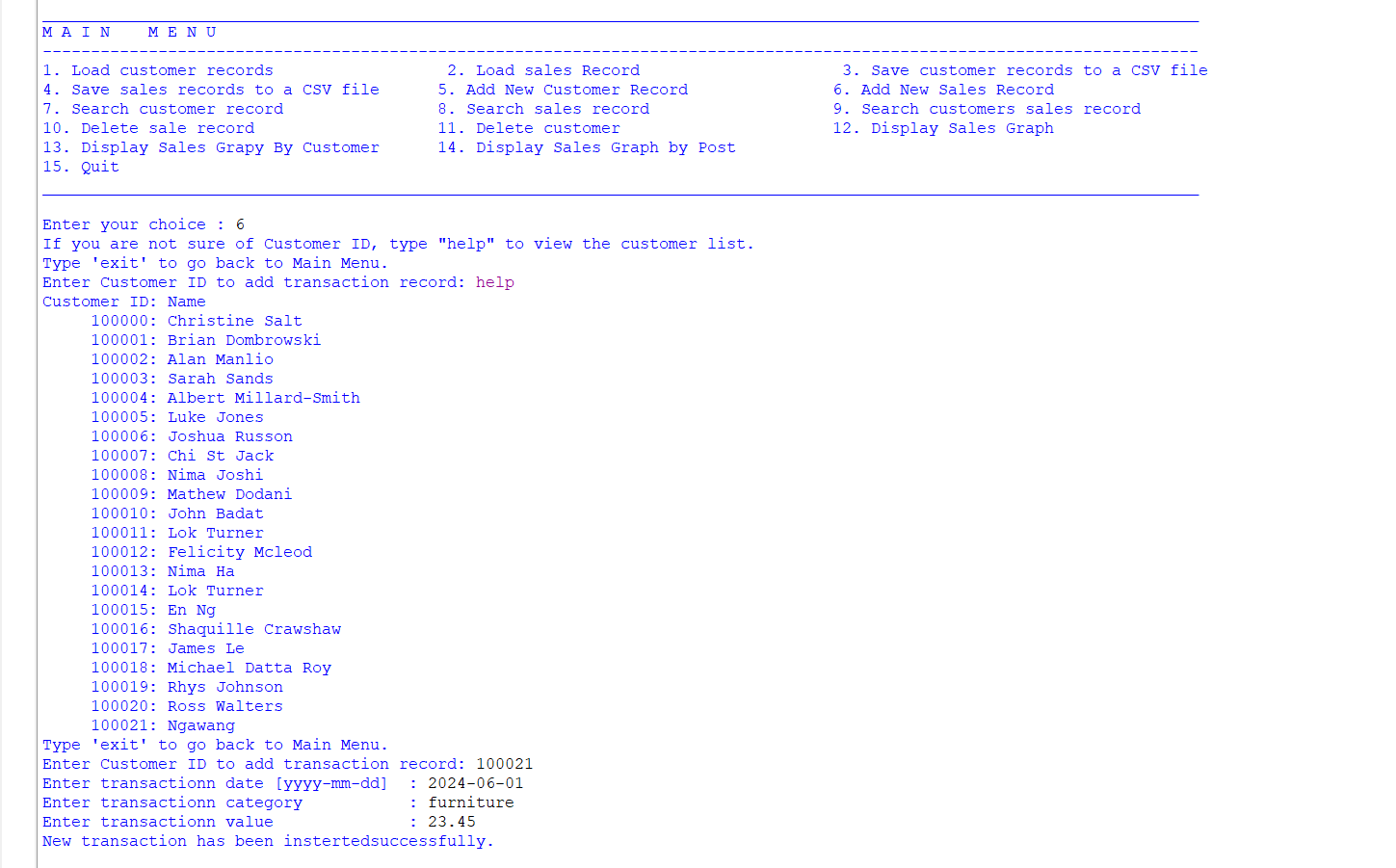
|  |  |  |
| --- | --- | --- |
| Sl no | Requirement | Status |
| 1 | Adding new customer to data structure by getting customer name, postcode and phone number where:   * Customer ID is autogenerated and unique * Customer Name is required. * Postcode and phone no is optional | Fully  Completed |
| 2 | Add a new sales record for an existing customer.   1. First it checks the customer record for existing customer. 2. Transaction id is unique and autogenerated. 3. Date, category and value has to passed by user | Fully  Completed |
| 3 | Search customers using a single search string.   1. Allow partial matches. 2. Compare search string to id, name, postcode, and phone number. | Fully  Completed |
| 4 | Searching record.   1. Compare search string to date,customer id, category and value 2. Allow partial search | Fully  Completed |
| 5 | Display all sales records from a customer using his/her customer id. | Fully  Completed |
| 6 | Delete a sale record with a given transaction ID. | Fully  Completed |
| 7 | Deleting customer record along with its sales record | Fully  Completed |
| 8 | Quiting the program | Fully  Completed |

## Test Evidence:

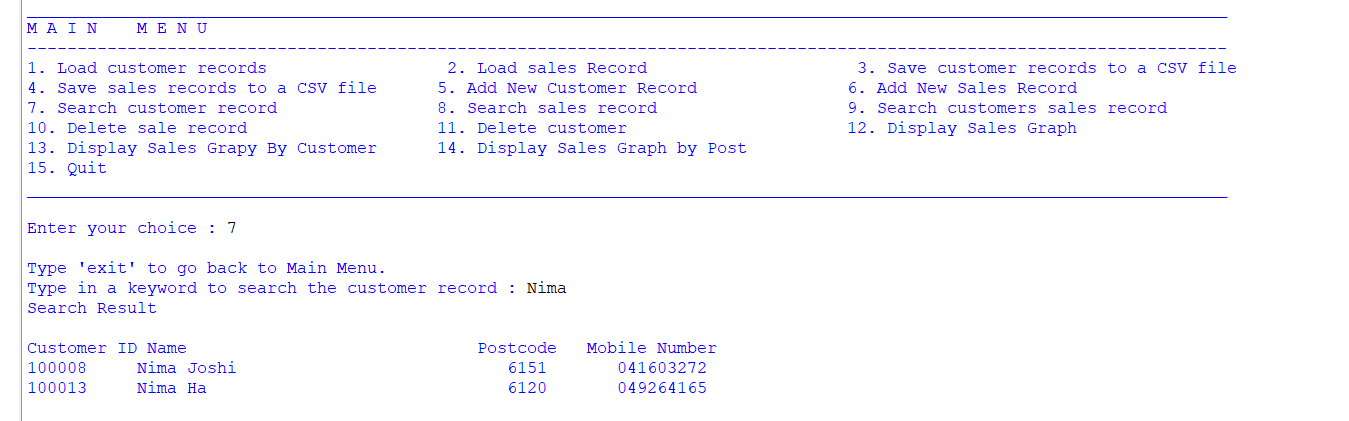
Feature 1: Adding new customers by getting the customer’s name, postcode, and mobile number from the user.



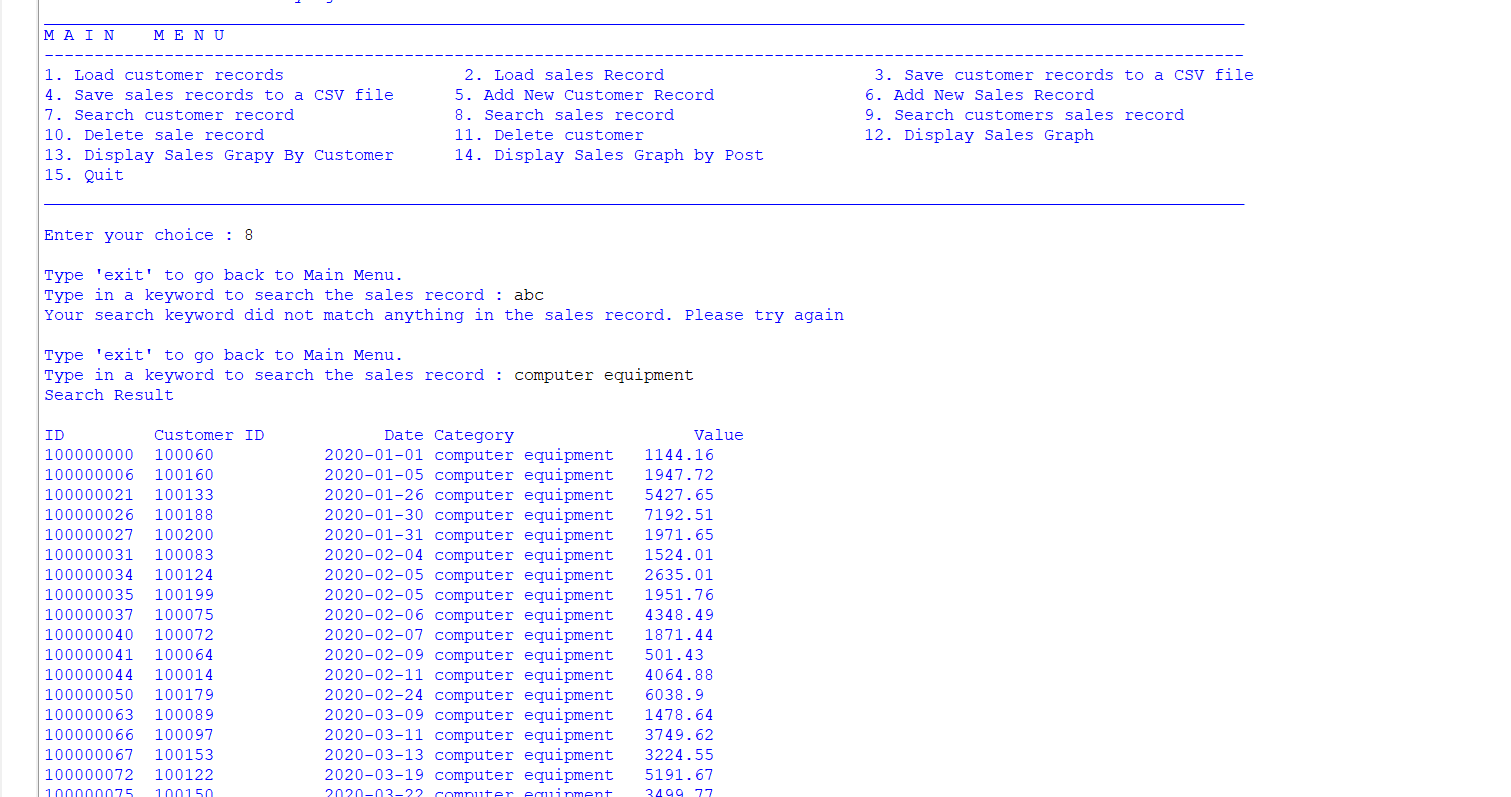
Feature 2: Add a new sales record for an existing customer.



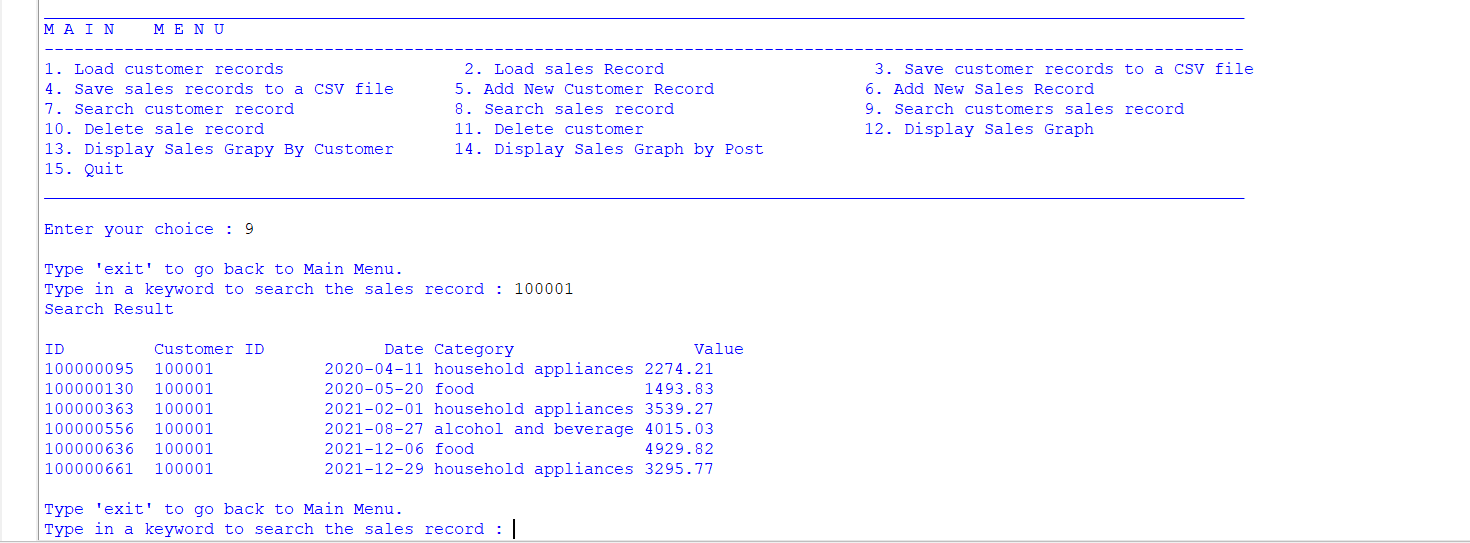
Feature 3: Search customers using single search string.



Feature 4: Display the sale record single search string.



Feature 5: Displaying sale history record for existing customers:

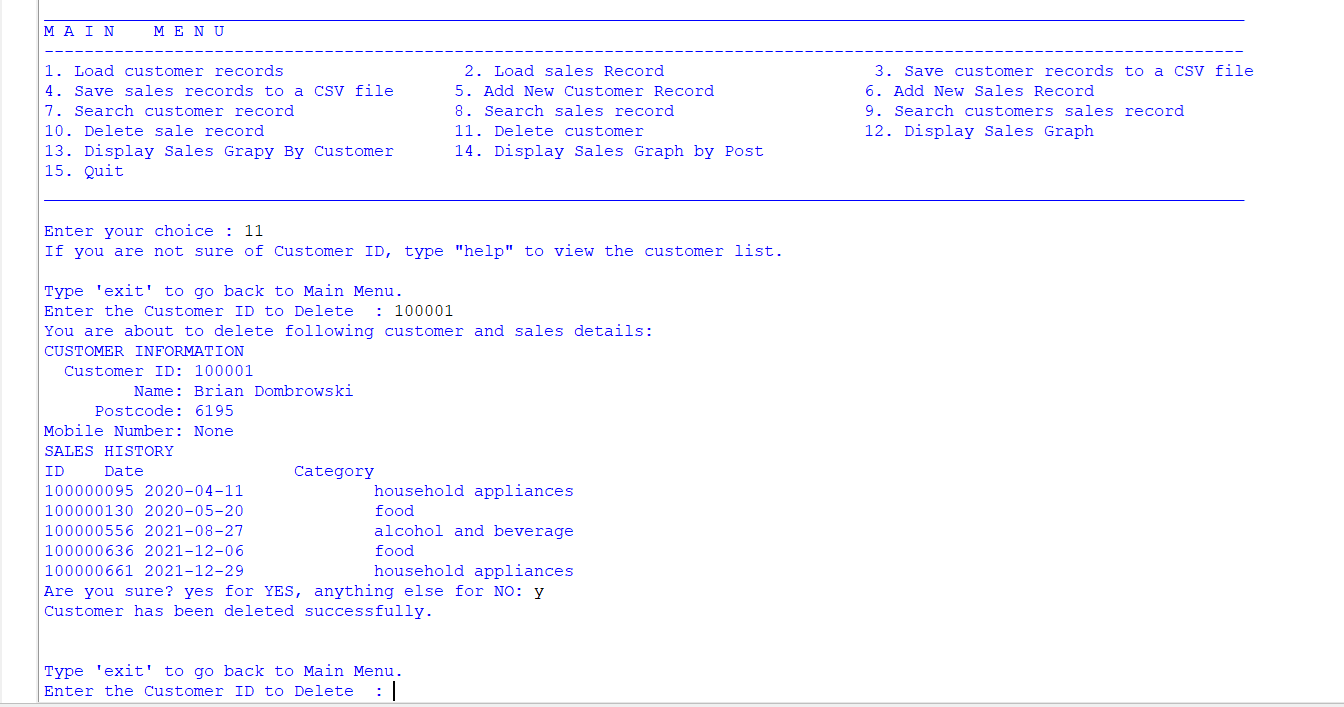


Feature 6: Deleting a sales transaction.

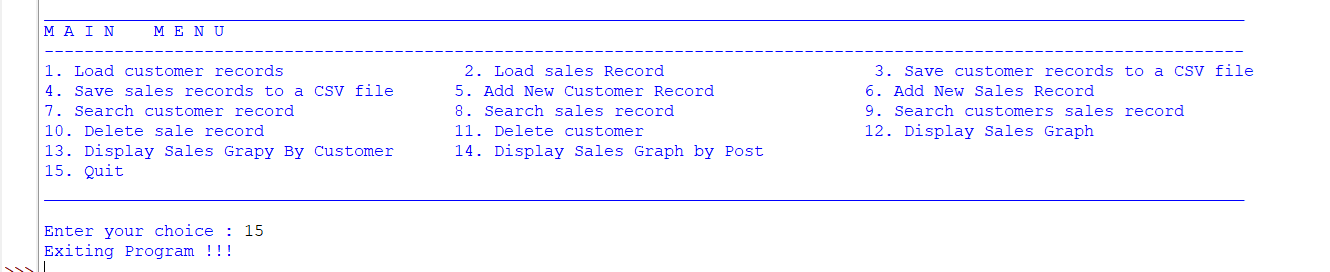
A close-up of a computer screen

Description automatically generated

Feature 7: Deleting a customer record along with their sale record.



Feature 8: Exiting the program.



# Display Sales Performance Graphically

## Discussion of the solution

I’ve opted to use a structured array to store customer and transaction records. This choice is due to its ability to define the data type to be stored and the ease of accessing values through dictionary-like column names.

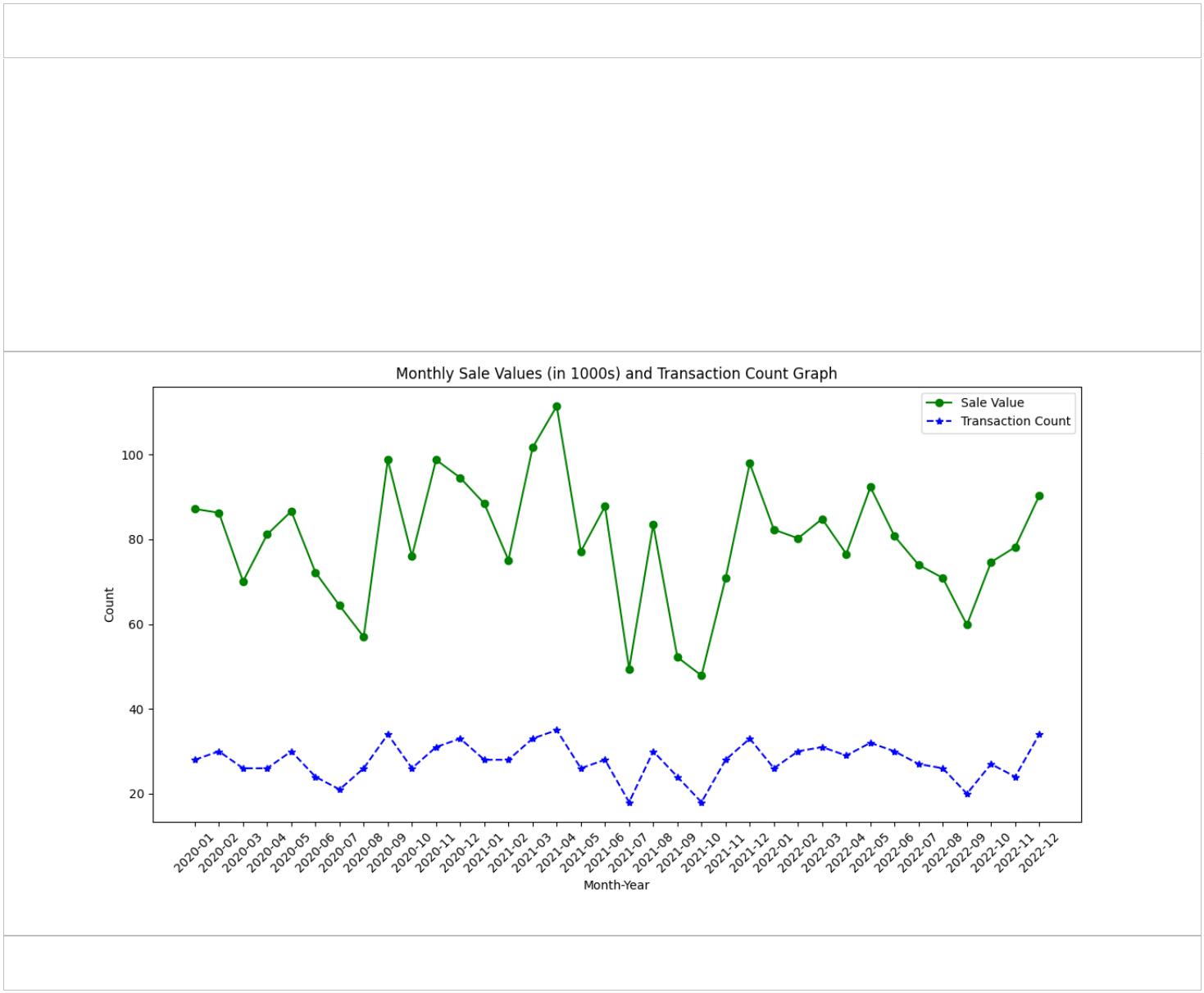
The visualization of sales and transaction counts is achieved using the matplotlib and numpy modules. Given that sales values are typically much higher than transaction counts, the program employs a function to scale down sales values by orders of 10, 100, or 1000, depending on the highest sales value.

When creating a graph, the transaction database is passed to a function (get\_months), which extracts all unique years and months and returns an array. This array serves as the x-axis for the graph. The corresponding y-value is determined by passing the x to a function get\_Y, which computes the sales values and transaction counts and returns a detail for plotting.

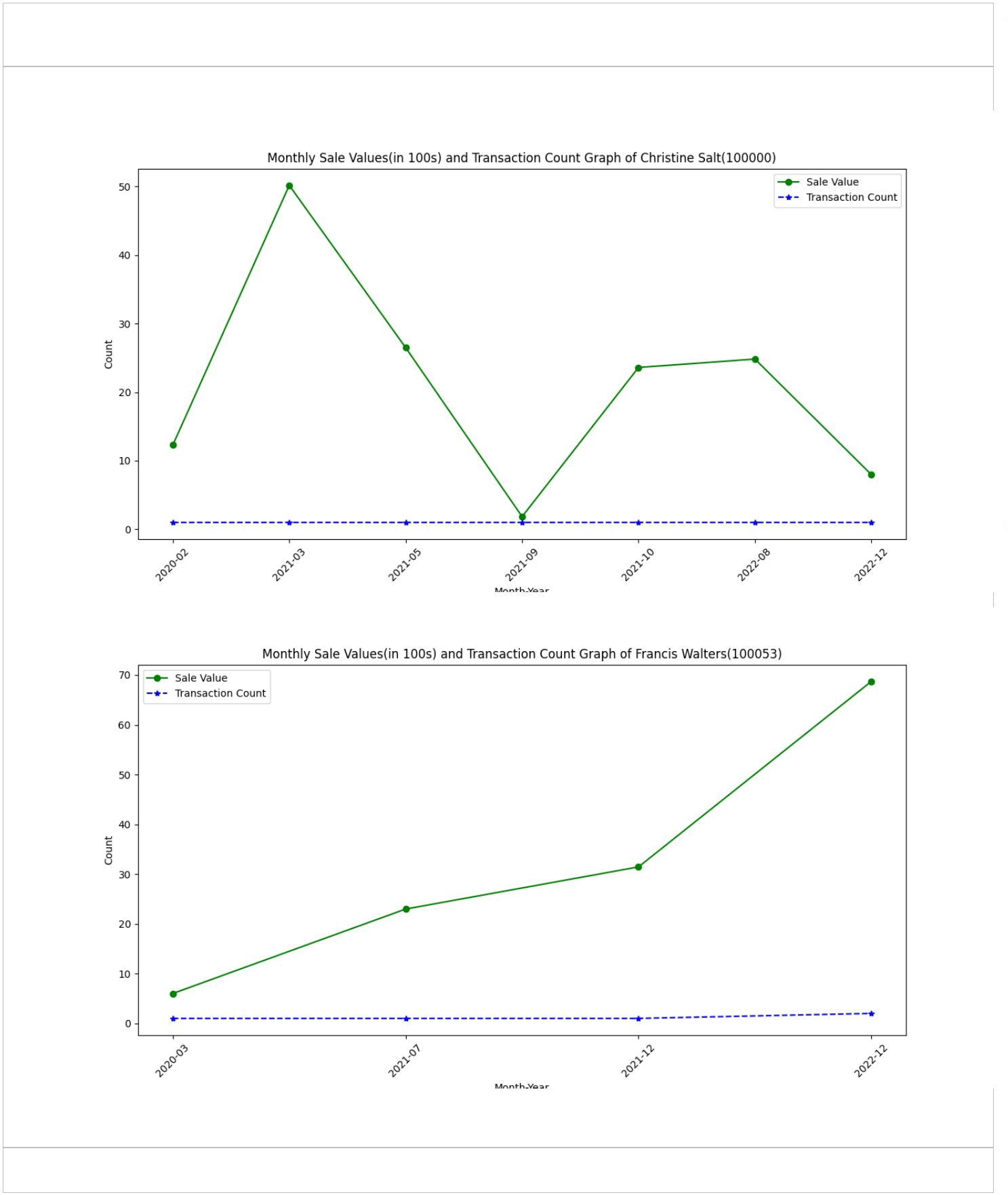
## Self-diagnosis and Evaluation for question 3:

|  |  |  |
| --- | --- | --- |
| Sl no | Requirement | Status |
| 1 | Display the monthly sales values and transaction numbers with two-line graphs in one axis. | Fully Completed |
| 2 | For a specific customer, illustrate the monthly sales figures and the count of sales made by the customer, using two-line charts on the same axes. | Fully Completed |
| 3 | For a specified postcode, illustrate the monthly sales figures and the count of transactions made by customers in that postcode area, using two-line charts on the same axes. | Fully Completed |

Features 1: Display the monthly sales values and transaction numbers with two-line graphs in one axis.



Features 2: Displaying monthly sales figures and the count of sales made by the customer, using two-line charts on the same axes.



Feature 3: Displaying the monthly sales figures and the count of transactions made by customers in that postcode area, using two-line charts on the same axes.

