Ryan Bresnahan

4/30/2025

Database Systems Design and Management

Project Deliverable 3

**Description of Implementation**

My project is made of two interconnected components: the core local MySQL database (or schema) and a simple user interface (UI) implemented with base Python. Together, these two components form the end application that we have created for the fictional company used for this project – Newark IT.

We will start with the most important aspect of the application: the MySQL database. The main SQL code that is used to generate the database is comprised of three distinct sections: database and table creation, trigger creation, and data generation. When this file is run, these sections are run sequentially to form the database. There are 14 total tables: CUSTOMER\_MEMBERSHIP, CUSTOMER, SHIP\_ADDR, SHIP\_CUST, CRED\_CARD, BASKET, PRODUCT\_ALL, FILLED, PRODUCT\_LAPTOP, PRODUCT\_DESKTOP, PRODUCT\_PRINTER, PRINTER\_SOLUTIONS, OFFER, TRANSACT. These are connected in the same way that is shown in Deliverable 2 (in 3rd Normal Form) and the EER diagram in Deliverable 1.

There are a total of seven triggers that are used, largely to control the logic of the BASKET, FILLED, and TRANSACT tables – which are all, in one way or another, dependent on the other. These triggers are: check\_if\_user\_exists (ensures registration before shopping), create\_basket\_after\_customer (ensures basket is made after registration), set\_filled\_final\_price (updates FILLED final price field based on membership tier), update\_basket\_qtyitem\_after\_insert (updates quantity of items after BASKET insertion), update\_basket\_totamt\_after\_insert (updates total amount [$] after BASKET insertion), update\_basket\_qtyitem\_after\_remove (updates quantity of items after BASKET deletion), update\_basket\_totamt\_after\_insert (updates total amount [$] after BASKET deletion). Combined, these triggers enforce the application logic and allow for a functional shopping experience in the UI.

The last component is the UI, which is in the form of a command line application (CLI). The CLI has six total options for the user: user registration, add a product to basket, place order, view transaction history, and view statistics. These will be further described in the section below, which is the user guide for how to use the CLI.

**User Guide**

There are few instructions that are needed for this app, which is quite simple and easy to use. First, one will need to run the program either in the command line or in an IDE of their choice. As stated above, there are six total options in the main menu that are available to the user: user registration, add a product to basket, place order, view transaction history, and view statistics. Each of these corresponds with a letter (A to C or X). Entering A through C will allow the user to engage in some action, while X will exit the app.

When the user selects an option, they simply need to follow the steps that are presented on the screen. For example, when registering a new customer, the user must enter a first name, surname, phone number, etc. When finished, said customer will be added to the database and a user ID will be generated for them, and shown to the user of the app. For any of the other options, except for E (statistics) and X (exit), the user must simply follow the prompts on the screen.

The one section of the of the CLI which is different from the previous is option E, which allows the user to view select statistics. These statistics correspond to the desired statistics that were described in the functional requirements in the project rubric. Selecting E brings the user to the statistics sub-menu, where there are six options available: view most frequently sold products (A), view products sold to the most unique customers (B), view top 10 customers by spending (C), view top 5 zip codes by shipments (D), view average price per product type (E), and return to the main menu (X).

Apart from this overview, there is no other functionality that the user needs to be aware of. The application is a fully functional shopping system, wherein the user can add a customer, add products to that customer’s basket, place an order based on that basket, then shop again (with the newly generated basket, since the old one was checked out). Value or input errors do not cause the application to crash – the operation will simply fail. Overall, the app is easy to use and fulfills the desired functional requirements.