Rashi Ghosh
010699523

CSCE 4523: Database Management Systems

Homework 4

In this assignment, the task was to use a high level language (C++ in this particular case) to interact with a SQL-based database (via ODBC in this particular case). The end user can run the program from the command line and interact with the database to run queries, add, delete, and update from a menu with predefined options.

The overall implementation strategy behind this code was as follows:

First, Dr. Gauch's sample C++ code was used as a template. Then, it was modified by adding menu options to select from 5 different functions that would perform queries/alterations to the database tables.

A couple extra functions (aside from the 5 menu functions) were added. One was processInput, which will take user input and convert it to all caps and remove spaces to make sure the database can query it properly. The second function was queryModify, which is used when deleting data from the database. This function uses **execute** instead of **executeQuery** to make sure the database doesn't throw any errors.

The following are the menu options available to the user, and some details and comments about their function, implementation, and error handling:

1. Find all agents and their associated clients in a given city.
    - Prompts the user for a city. If an agent with clients is found in that city, returns the agent and their associated clients.
    - If a) that city is not found, or, b) there is an agent in that city but does not have any clients yet, then this function will return an empty table. The user is prompted for a different city name until an agent with their clients is returned, or the user enters "0" to return to the main menu.
    - Note: Empty tables are detected in the function by checking resultSet->rowsCount(). If it is 0, then nothing was found, and throws the user into the loop mentioned above. This method for detecting empty tables is used throughout the program.
2. Add a new client, then purchase an available policy from a particular agent in the client's city.
    - Prompts the user for client information. If an agent exists based on the city the user entered, then the agent(s) is/are listed, and the user is prompted again for a type of policy. If a valid policy is entered, then the user is prompted to choose an agent name and enter an amount to purchase the policy.
    - When adding a client, the query inserts the client ID by performing a simple calculation on the max number in the C_ID column in order to properly "increment" the C_ID column. Note: This method for "auto-incrementing" an ID column is used later on in the program when adding an agent.

- If there is no agent in the client's city, the function immediately returns after notifying the user there are no clients in that city.
- If the user enters an invalid policy, the program will continue to prompt the user until a valid policy is entered, or the user enters "0" to return to the main menu..

3. List all policies sold by a particular agent.
    - Prompts the user for an agent name and city. If there exists an agent in that city who has sold policies, the policies they have sold along with the commissions earned from those policies is returned.
    - If the agent name or city name does not exist, or if the agent has not yet sold any policies, an empty table is returned and the user is notified. The user is continuously prompted for a different agent and city until one is valid, or the user enters "0,0" to return to the main menu.
4. Cancel a policy.
    - First displays all policies, and then prompts the user to enter a purchase ID. If the ID is valid, successfully deletes it and shows table.
    - If the purchase ID does not exist, the helper function used to delete the tuple (queryModify) returns a false boolean. It does this in the following way: executeUpdate returns the number of rows affected. If 0 rows are affected, that means nothing was deleted, and the purchase ID was therefore "invalid." The helper function returns false.
    - The user is continuously prompted for a valid purchase ID until one is entered or the user enters "0" to return to the main program.
5. Add a new agent to a city.
    - The user is prompted for agent information. The agent is added to the table (the A_ID is "auto-incremented" in the same way C_ID is incremented in option 1). The updated table is displayed to the user.
6. Quit.
    - Deletes everything from tables & drops all tables. Disconnects the database & bids the user good-bye!

A typsecript with various test cases is included separately in the project folder. This assignment was a really neat way to see how databases can be accessed and interacted with via a high level language. ODBC allowed queries to be easily done by simply using the in-built execute and executeQuery functions. The error checking was a bit tough -- I am not sure I did it in a proper or conventional way. The only way I could think to do error checking was by checking empty tables, etc.

Overall, this was yet another tedious assignment that I got a lot out of. I learned a lot about using a high level language like C++ to interact with a SQL database. I had to learn about many little things such as the difference between execute and executeUpdate, resultSet and checking the rows count, etc. It was a wholesome assignment for me overall!