horizontal line

Automotive Inventory Management System

# INTRODUCTION

In this project, the group was expected to develop an application that interacted with Python and SQL. Objectives expected from this project consisted of building at least three tables, importing data into the database using python/SQL, and carrying out data analysis, in addition to explaining the added value of this application once deployed. By applying the techniques and skills learned in the course, we were able to meet the project goals utilizing the code and earlier methodologies.

The project team deliberated on a database topic that would meet all the needs and pique everyone's curiosity. We ended up going with an Automotive Inventory Management System. This project allows users to perform various operations such as adding/removing cars, managing buyers, handling repairs, importing data from CSV files, displaying inventory, and performing VIN lookups. The GUI provides an interface for users to interact with the database, updating and retrieving information related to cars, buyers, and repairs. It's designed to facilitate the management of automotive inventory for a business or dealership.

Three tables were created and categorized by columns;

**Table 1 “inventory”: id, make, model, year, price, VIN**

**Table 2 “buyer”: id, name, price, credit\_score, payment\_type, VIN**

**Table 3 “repairs”: id, ticket\_id, model, problem, repair\_cost, VIN**

The inventory table keeps track of important vehicle details including the make, model, year, price, and VIN. This table facilitates inventory management and sales activities by acting as the main source of information about the cars that are available. The buyer's table stores information on possible buyers, including name, price, payment type, credit score, and VIN. It facilitates credit assessments and sales processes by helping to organize buyer data. Lastly, the repairs database contains information about repairs, including ticket\_id, model, problem, repair\_cost, and VIN. This table offers information on the history and expenses of the vehicle and aids in the tracking of auto maintenance and repair costs. When combined, these tables offer a thorough overview of the sales, maintenance, and inventory components of the automotive industry.

# DATA SOURCES and ASSUMPTIONS

The online automotive data compilation was executed in an attempt to emulate standard inventory management practices within the automotive industry. Using an Excel sheet was similar to the common method used by many smaller dealerships to track inventory. The compiled data was then exported into a.csv file, which was selected because of its simplicity of use and effectiveness in data import and export processes. The application was created with particular features in mind to make it easier to import this CSV file and process its contents into the appropriate inventory tables in the system.

While developing the application, the following assumptions were made:

* Unique VINs: We assumed that each vehicle possessed a unique Vehicle Identification Number (VIN), serving as the primary identifier.
* Accuracy of Data: We assumed that imported data from diverse sources was accurate and consistently reliable.
* Inventory Consistency: We assumed that the inventory remained consistently updated and synchronized across all records and sales activities.
* Repair and Maintenance: We assumed that all necessary repairs and maintenance were duly recorded and accurately accounted for.

# PROCEDURES

To begin, the user should launch the application, selecting either the graphical user interface (GUI) or the terminal access option, depending on their preference. Upon launch, the system requests authentication through the input of the assigned password, guaranteeing safe access to the system interface.

Next, for inventory management, the user can add new cars to the inventory by providing specific details like make, model, year, price, and Vehicle Identification Number (VIN) using the dedicated interface. Removal of existing cars from the inventory is facilitated by specifying the car's identification number (ID). The system also provides a thorough inventory display that gives users a close-up look at the cars that are currently available.

Moving to buyer management, adding new buyers involves inputting essential information such as the buyer's name, price, credit score, payment type, and associated VIN. Buyer removal from records is executed by providing the buyer's identification number. Additionally, the system streamlines buyer management procedures by offering an easily accessible list of current buyers.

For repair and maintenance tracking, users can record new repairs by entering details like the ticket ID, model, problem description, repair cost, and relevant VIN. Removing past repairs from the system is facilitated by specifying the repair's identification number. Furthermore, the system enables users to retrieve and display the repair history, aiding in meticulous maintenance tracking.

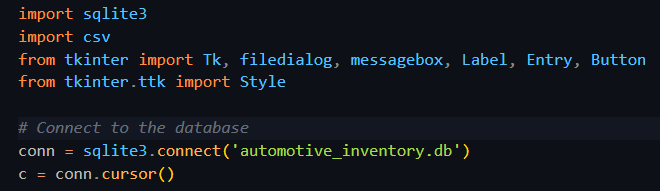
Among the other features of the system is the simplified data import from CSV files, which makes it possible to quickly enter several car details at once. Furthermore, users can search for specific VINs using the VIN lookup feature, which retrieves related buyer, repair, and vehicle data to enable thorough analysis and insights.

To end the session, users can securely terminate the application by utilizing the system's provided "Exit" function, making sure the database connection is securely closed and successfully wrapping up the operational session.

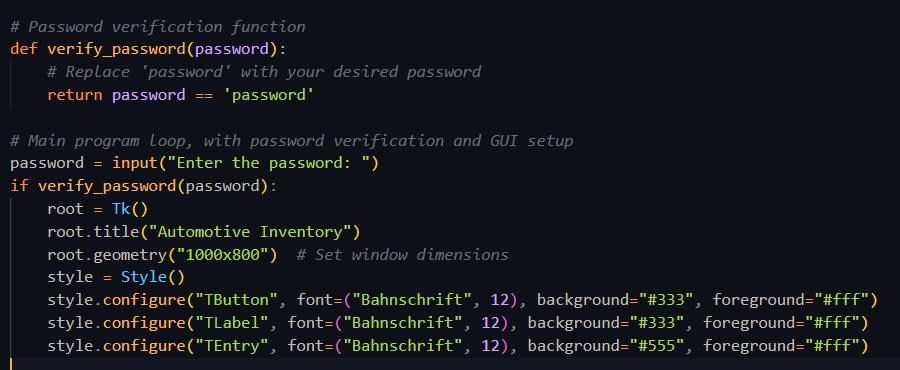
# IMPLEMENTATION

## Connection to SQL and Initialization

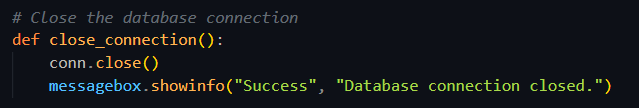
The below segments cover how the code was designed and implemented. As this program works jointly with SQL and Python, a bridge between them was required. Below is the code in regard to that connection.

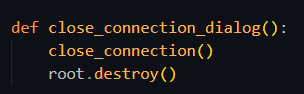


The program here imports the SQLite module for getting the SQL functionality, and then connects to the database named automotive\_inventory after starting an instance. Additionally, there is a password request in the main loop of this program to make sure that only authorized users can access this, as shown in the code snippet below.



The program also exits when prompted to by closing the SQL connection.

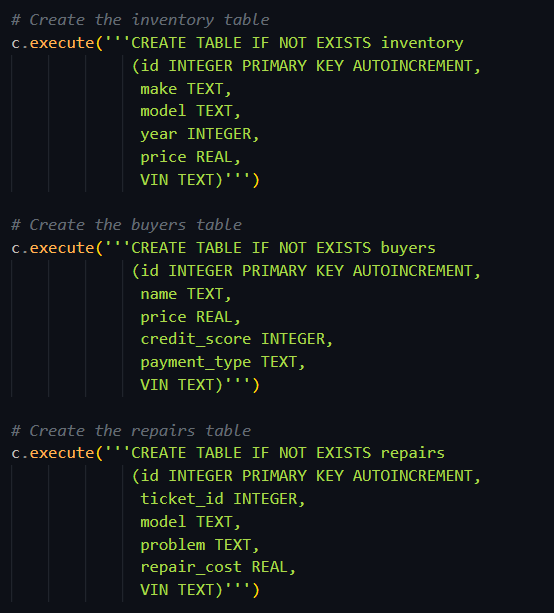




Note that the root.destroy() function involves the Python Tkinter module, which we have used in this program to provide a simple GUI interface.

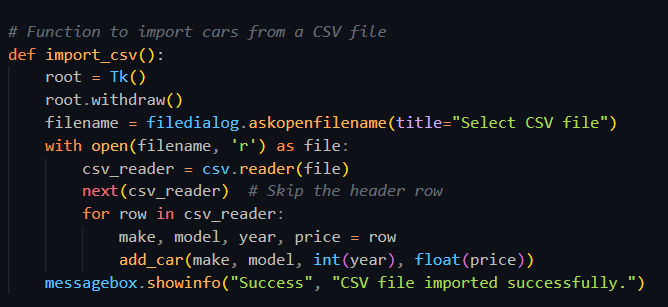
## Creating Tables

The team created three tables as mentioned in the introduction. The codes below explain and demonstrate how each table is created jointly by Python and SQL. The program checks for any previously existing databases, and if there are none, it executes the following:



## Importing Data

Importing data is a major aspect of our program, as we do not randomly generate data. Our team decided that random data generation would be inapplicable towards this project as it is with regard to the automotive industry. Below is the code regarding code input.



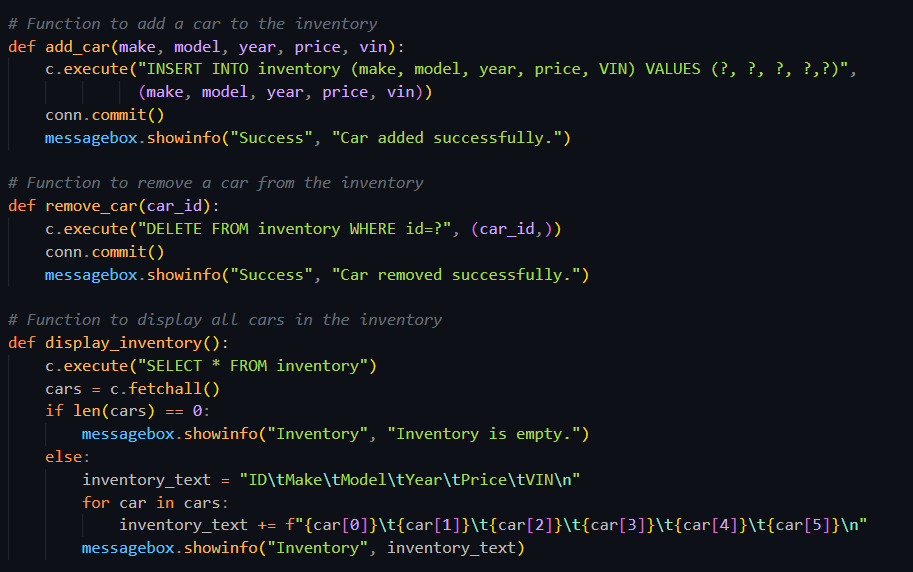
## Data Management

As for the user manually inputting information, the program has been coded to modify all three tables in a very dynamic manner, jointly with the Tkinter module. Along with adding data, the user can also remove and display entire tables.

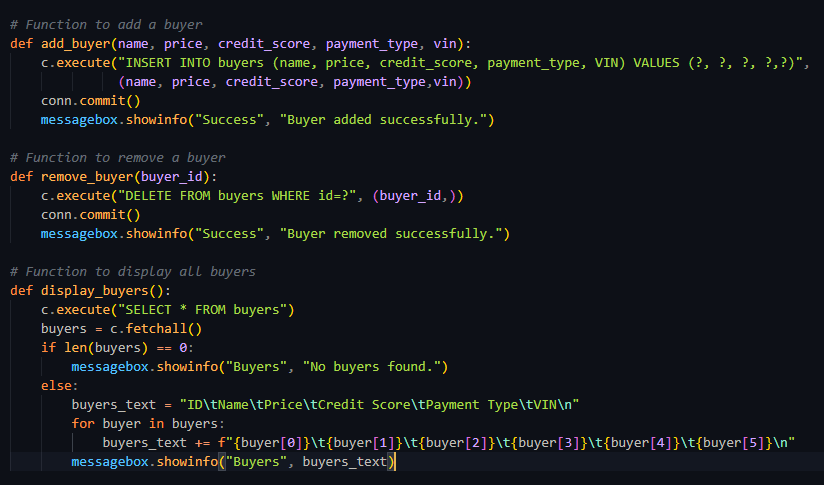


Above is the GUI interface which displays the multiple options the users are provided when using the program.

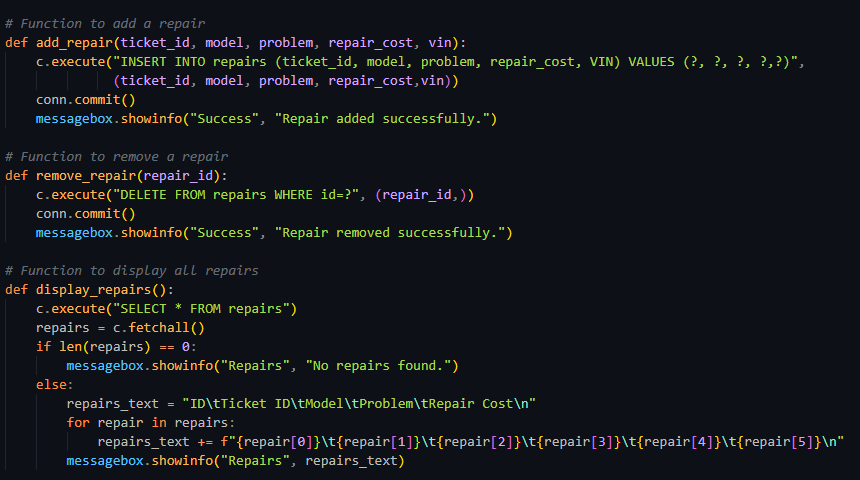
Below is the functionality for table one, inventory:



Functionality for table two, buyers:



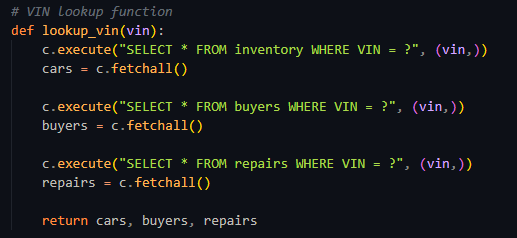
Functionality for table three, repairs:



All three tables are equipped with add, remove, and view options, which are all interactive with the user.

## Data Analysis and Lookups

The program also analyzes the data and returns common VINs that are requested by the user. This is especially useful as one customer could have multiple entries in a database, or one vehicle could have multiple entries. This helps the user to establish vehicle records. Below is the code for the VIN Lookup.



# CONCLUSION

In summary, the completion of the automotive inventory management system stands as a pivotal achievement in optimizing operations within the automotive industry. Through the utilization of a robust SQLite database and a user-friendly interface built with Tkinter in Python, the system proficiently handles inventory control, buyer management, and repair tracking. Its functionality spans from seamless addition and removal of car details to comprehensive buyer information and meticulous repair records. Notably, the system's capabilities extend to importing data from CSV files and real-time VIN lookups, presenting a comprehensive solution for the multifaceted needs of automotive inventory management.

Developing this system is important because it increases productivity, organizes data, and guides automakers' decisions. This technologically advanced tool helps the auto industry improve customer service and inventory control.

Essentially, this project enabled us to understand how Python and SQL work together to allow developers to design reliable, adaptable, and effective applications. It highlights the benefits of combining a strong programming language with an organized query language, showing how well these two tools work together to create complex, scalable database-driven systems.