**Mridul Das**

**Vocational Training**

**C&IT Dept.**

**Project: - Use python to convert excel, text or word files to database files and vice versa while also creating functionalities such as search, view, database modification and generate reports from database by providing a graphical interface.**

**Menu:**

It is the main menu which offers user various options to choose from depending on their needs:

1. Excel to database
2. Text or docx to database
3. Modify database
4. Database viewer
5. Search database
6. Database to excel
7. Generate Report
8. Exit



To convert Excel to Database:

User is asked to enter the location of Excel files then also to select the location of database file (if present) or else they can create a new file.

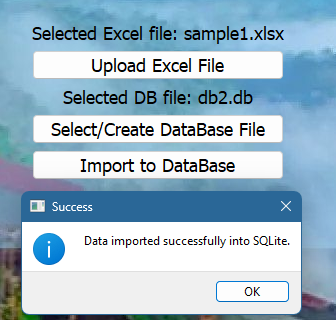


Then user is asked select the location of database file (if present click no ) or else they can create a new file

A screenshot of a computer

Description automatically generated

Then the user has to select import to DB and they will be provided with confirmation.



To convert .txt or .docx to Database:

The user can upload text or docx file by selecting the first option and then they can select or create the database file using the second option.

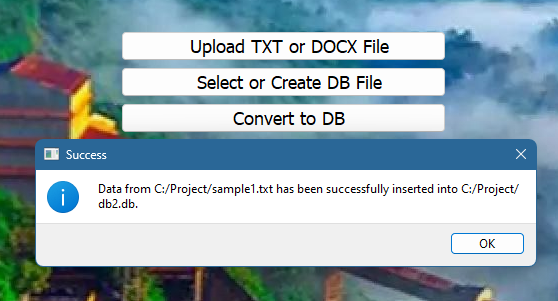


Then user is asked select the location of database file (if present click no ) or else they can create a new file

A screenshot of a computer

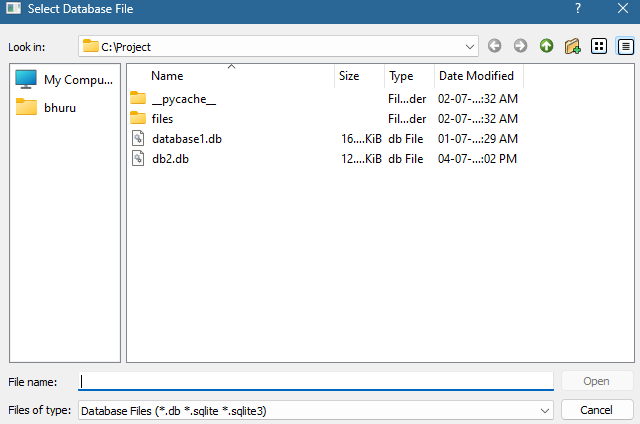
Description automatically generated

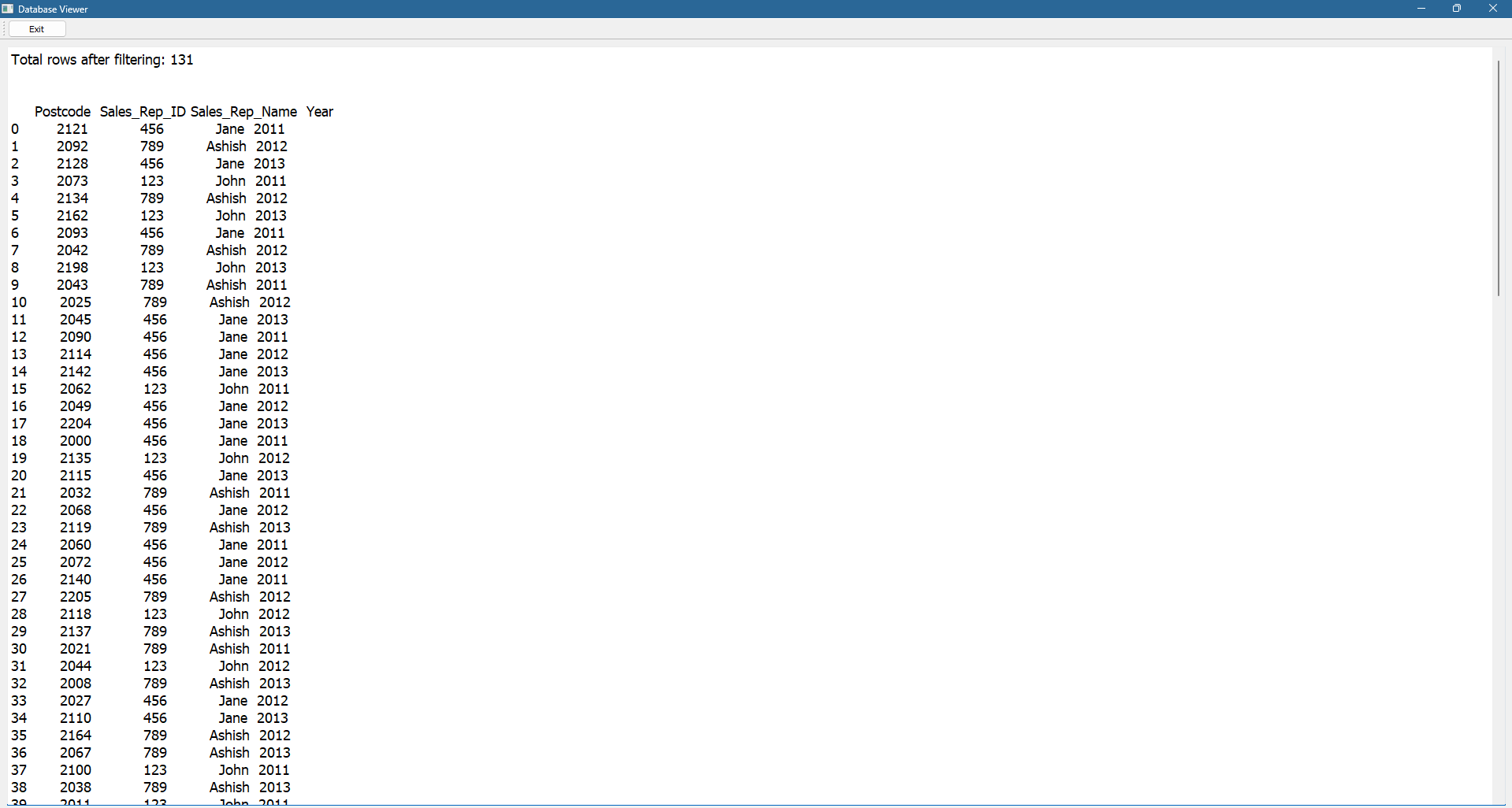
At last, The user has to click on Convert to DB After which they are provided with confirmation



To View the Database:

The user is asked to browse the location for the database file and soon as they select the file, the database is viewed in a fullscreen window with all the headers.

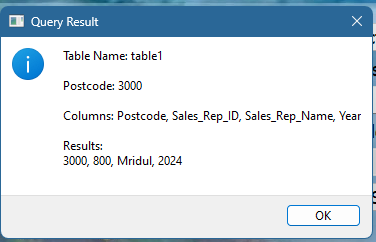




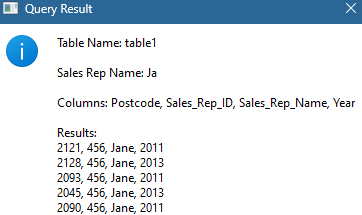
To Search a Specific Postcode or using keywords from the Name:

In the program the user is asked for the database location along with the choice whether they want to search with postcode or they wants to search using Name. The search is done the result is displayed in the other windows and if the user wants they can again search up another postcode or name.





There is not a need that the words entered should be in order they can be in the middle of the name and still the result can be displayed.

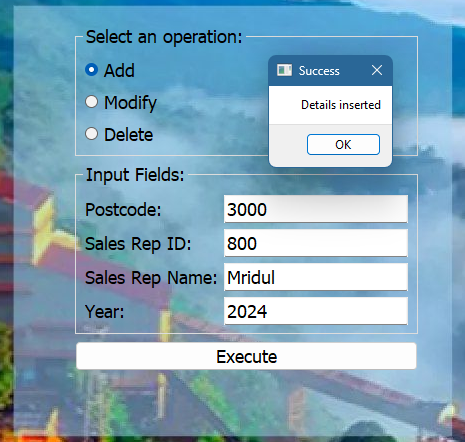


To modify a database:

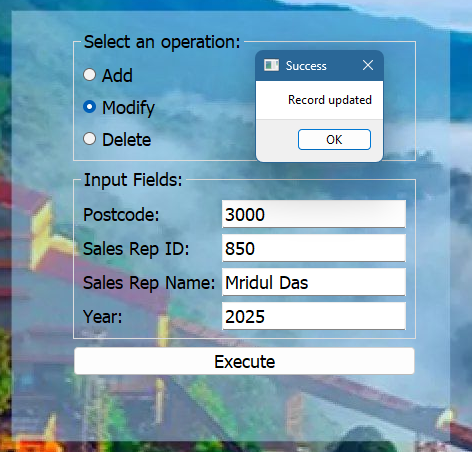
First the user is asked to choose a database file and then there are three operations that the user can perform. Until all the fields are filled user cannot add and user can update data using modify data and also while deleting the user is asked for confirmation while at the same time showing which row is going to be deleted by just entering the postcode in the input field.



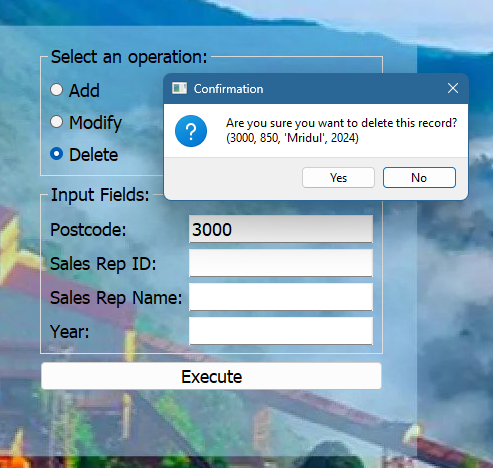
Add:



Modify:



Delete:



If a Postcode doesn’t exist it would give error and same way if we try to add a postcode which already exists then too it would return a error on the screen that ‘The postcode already exist try again with a different postcode’.

Convert Database file to Excel

The main motive is to convert .db file back to .xlsx incase the user wants to migrate the files. For this you just need to select db file (Location can be selected in the explorer window) that you want to import and click Export to Excel.

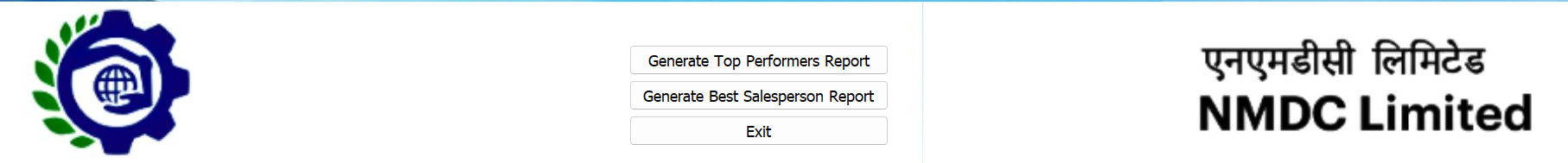


A screenshot of a computer

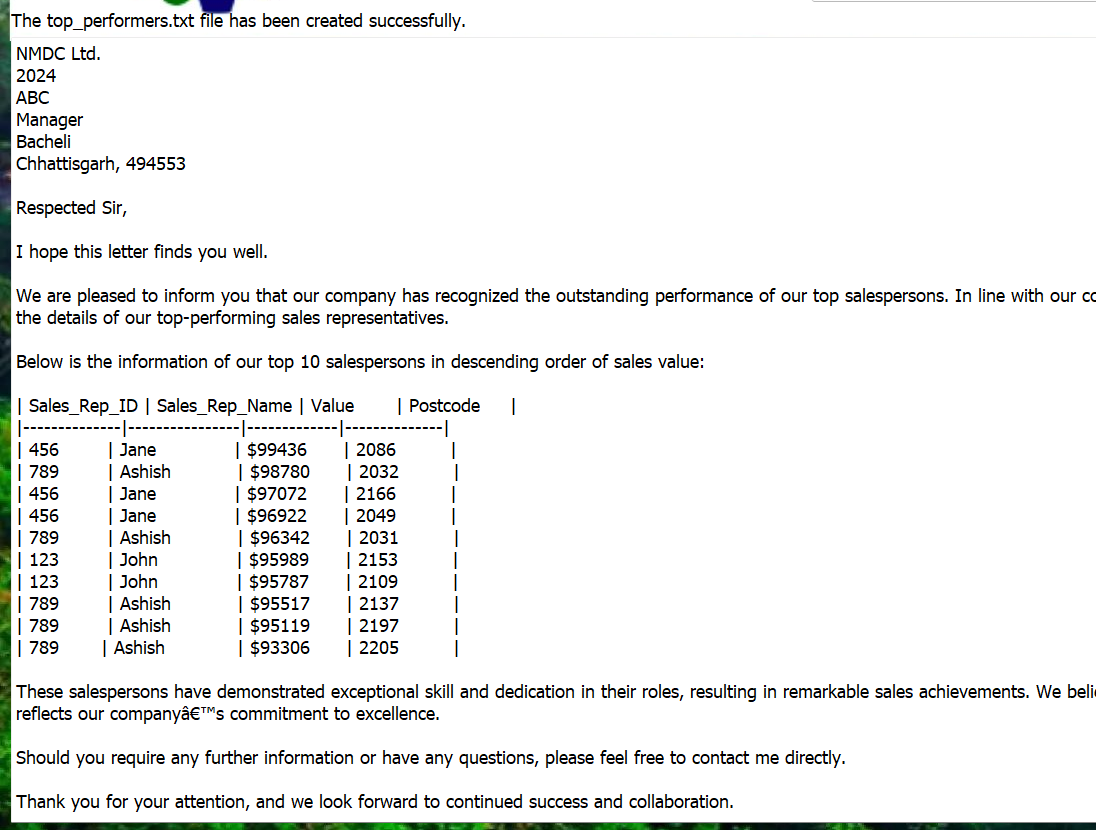
Description automatically generated

Generate Report

User can generate letter from pre-existing drafts by just running the program and then the program itself would fill by replacing the column headers.



Generate Top Performers report:



Generate Best Salesperson Report:

A close-up of a message

Description automatically generated

**In-Depth Documentation of Code**

**Overview**

app.py is a PyQt5 application that provides various database operations, such as importing data from Excel to SQLite, executing scripts, performing CRUD operations on the database, viewing database contents, exporting data to Excel, fetching data, and generating letters.

**Dependencies**

* PyQt5
* sqlite3
* pandas
* subprocess
* sys
* os

**Main Classes and Functions**

**1. ExcelToSQLiteApp(QWidget)**

This class handles the import of data from an Excel file to a SQLite database.

**Methods:**

* \_\_init\_\_(self): Initializes the UI components for the Excel to SQLite import functionality.
* browse\_file(self): Opens a file dialog to select an Excel file.
* load\_to\_db(self): Loads data from the selected Excel file to the specified SQLite table.
* show\_message(self, title, message): Displays a message box with the given title and message.

**2. DatabaseApp(QWidget)**

This class handles CRUD operations (Create, Read, Update, Delete) on the database.

**Methods:**

* \_\_init\_\_(self): Initializes the UI components for database operations.
* keyPressEvent(self, event): Handles the Enter key press to trigger the button click event.
* button\_click(self): Handles the button click event to perform the selected CRUD operation.
* add\_data(self, postcode, sales\_rep\_id, sales\_rep\_name, year): Adds a new record to the database.
* modify\_data(self, column\_name, new\_value, postcode): Modifies an existing record in the database.
* delete\_data(self, postcode): Deletes a record from the database.
* show\_message(self, title, message): Displays a message box with the given title and message.

**3. DatabaseViewer(QWidget)**

This class provides a UI to view the contents of the database.

**Methods:**

* \_\_init\_\_(self): Initializes the UI components for viewing the database.
* load\_data(self): Loads data from the database and displays it in a table.

**4. FetchDataApp(QWidget)**

This class allows the user to fetch data from the database.

**Methods:**

* \_\_init\_\_(self): Initializes the UI components for fetching data.
* fetch\_data(self): Fetches data from the database based on the provided query and displays it in a table.

**5. DBToExcel(QWidget)**

This class handles exporting data from the database to an Excel file.

**Methods:**

* \_\_init\_\_(self): Initializes the UI components for exporting data to Excel.
* export\_to\_excel(self): Exports the data from the specified SQLite table to an Excel file.
* show\_message(self, title, message): Displays a message box with the given title and message.

**6. MainWindow(QMainWindow)**

This class provides the main application window with various functionalities accessible through buttons.

**Methods:**

* \_\_init\_\_(self): Initializes the main application window and sets up the UI components.
* set\_background\_image(self, image\_path): Sets the background image for the main window.
* open\_excel\_to\_db(self): Opens the ExcelToSQLiteApp window.
* open\_db\_operations(self): Opens the DatabaseApp window.
* open\_db\_viewer(self): Opens the DatabaseViewer window.
* open\_db\_to\_excel(self): Opens the DBToExcel window.
* open\_fetch\_data(self): Opens the FetchDataApp window.
* open\_letter\_generation(self): Opens the letter generation application.
* execute\_external\_script(self): Executes an external script (wordortxt.py).

**7. Main Execution**

The application entry point initializes the PyQt application and displays the main window.

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

demo = MainApp()

demo.show()

sys.exit(app.exec\_())

**Usage**

1. Run the script: python app.py
2. The main application window will appear with various buttons for different functionalities.
3. Click on the respective button to perform the desired operation.

**Note**

* Ensure that the required dependencies are installed.
* Modify the paths and database/table names as per your requirements.
* Handle exceptions and errors appropriately to improve the robustness of the application.