**Solent University**

Advanced Database Systems

**Module Leader**:

Dr. Muhammad Akram

Project with Report

**Student ID**: [Your ID Number]

**Student Name**: [Your Full Name]

**Submission Date**: 4th February 2025

**Word Count**: ???

**Table of content**

[Introduction 3](#_Toc188978162)

[Part 1: Client Requirements and Database Design outline 4](#_Toc188978163)

[**1.1. Business Case for the Chosen Restaurant** 4](#_Toc188978164)

[**1.2. Seven Business Requirement** 4](#_Toc188978165)

[**2. Flat File Creation with Collected Data (Excel)** 5](#_Toc188978166)

[**3. Relational Schema and Database Design** 6](#_Toc188978167)

[**4. Develop SQLite Server database system** 8](#_Toc188978168)

[Part 2: Database design development 11](#_Toc188978169)

[**1.Triggers** 11](#_Toc188978170)

[**2.Views** 12](#_Toc188978171)

[**3.Queries Meeting Business Requirements** 14](#_Toc188978172)

[**4.Testing and reviewing the database.** 17](#_Toc188978173)

[Part 3: Programming for Database and testing 19](#_Toc188978174)

[**1.Communitation with SQLite by Python GUI** 19](#_Toc188978175)

[**2.Validation or verification of input data** 35](#_Toc188978176)

[**3.Stroing images in database using BLOB data type** 37](#_Toc188978177)

[**4. XML Development (and Encryption)** 41](#_Toc188978178)

[**5.Innovative features in the application** 48](#_Toc188978179)

**Introduction**

This report details the development of a database system for a restaurant, aimed at creating a highly efficient and user-friendly application. The system streamlines daily operations such as table reservations, sales tracking, and more to ensure smooth functionality.

Managing data in restaurants remains a major challenge, especially for small and medium-sized businesses relying on manual or outdated systems. This often leads to errors and duplication, risking the loss of crucial customer information and missed revenue opportunities.

The objectives of this report are to describe the client’s requirements, design an efficient relational database, implement a proof-of-concept system and evaluate its performance for the solution of the problems defined above. SQLite is used as a lightweight but powerful database management system that will be encapsulated by Tkinter, Python’s standard GUI package. XML will be used for storage and manipulation of arbitrary structured data.

The database is designed with third normal form (3NF) to make data consistet and prevent data for redundancy, relational features like Triggers, Views are also constructed to make system works effectively and efficiently as business wants, as well as usability. User testing confirms alignment with the selected functional requirements.

# **Part 1: Client Requirements and Database Design outline**

This section outlines the essential client requirements for the database system, followed by the process of designing and creating the database using SQLite. The goal is to create a system that meets the needs of the restaurant, ensuring it can handle customer data, inventory, employee information, and order details effectively.

## **1.1. Business Case for the Chosen Restaurant**

The restaurant is a small-sized bistro specializing in Italian food. It employs approximately 10 staff members and serves around 50 customers daily.

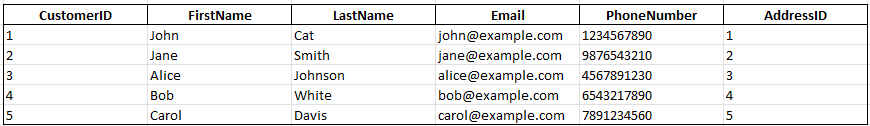
## **1.2. Seven Business Requirement**

1. **Customer Management** – The system should allow for storing and retrieving customer information, including personal details like address and phone number.
2. **Inventory Tracking** – The system should track inventory levels for food ingredients and supplies.
3. **Employee Records** – The database should maintain employee information, such as contact details, job title, and payroll information.
4. **Reservation Management** – The system must handle customer reservations, ensuring no double-booking occurs.
5. **Financial Records** – The database should track sales, revenue, and expenses, including generating financial reports.
6. **Menu Management** – The system should allow for managing and updating menu items, including prices and availability.
7. **Order Management** – The system should manage customer orders, order details and accurate billing.

## 

## **2. Flat File Creation with Collected Data (Excel)**

For data collection, a flat file was created using Microsoft Excel. Below is a screenshot of the flat file created for some of the tables.

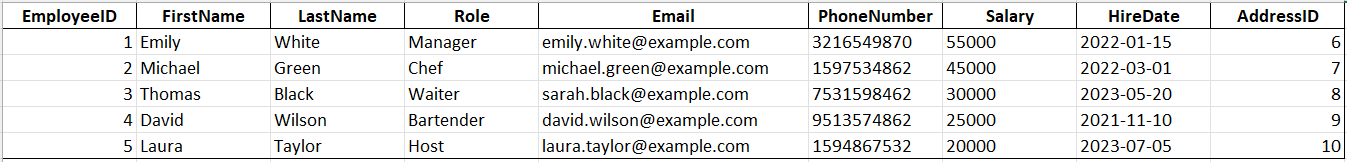


**Customers**

Obraz zawierający tekst, Czcionka, numer, linia

Opis wygenerowany automatycznie

**Suppliers**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

**Reservations**

**Staff**

## **3. Relational Schema and Database Design**

This section presents the relational schema for the restaurant management database, shown through an Entity-Relationship Diagram (ERD).

The ERD below illustrates the key entities, their attributes, and relationships:

Obraz zawierający tekst, zrzut ekranu, numer, design

Opis wygenerowany automatycznie

**Justification for 3NF Compliance**

**First Normal Form (1NF)**

**Criteria:**

* Each column must contain atomic values (no multiple values in one column).
* Each row must be unique, and there must be a primary key.

**Analysis:**

* All tables have primary keys (e.g., CustomerID, OrderID).
* Columns store atomic values without repeating groups of data.

**Second Normal Form (2NF)**

**Criteria:**

* Meets 1NF requirements.
* Non-primary attributes are fully dependent on the entire primary key (no partial dependency).

**Analysis:**

* For composite keys (e.g., MenuItemID + IngredientID), non-primary attributes depend on the primary key.
* Single-column primary keys (e.g., Orders, Customers) fully determine non-key attributes.

**Third Normal Form (3NF)**

**Criteria:**

* Meets 2NF requirements.
* No transitive dependencies (non-key attributes do not depend on other non-key attributes).

**Analysis:**

* Derived data (e.g., order totals) is not stored redundantly.
* Attributes like CustomerName are queried dynamically rather than stored redundantly.

## **4. Develop SQLite Server database system**

**4.1 Database Implementation**

The database was implemented using SQLite. The steps taken include:

1. Obraz zawierający tekst, Czcionka, zrzut ekranu

   Opis wygenerowany automatycznie**Database Creation**:  
   A new SQLite database file was created and named restaurant\_management.db.
2. **Table Creation**:  
   The following SQL commands were used to create the tables as per the relational schema

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

1. **Referential Integrity**:  
   Relationships between tables were established using foreign key constraints to ensure data integrity. For example:

* AddressID in the Customers table references the AddressID in the Address table.

**4.2 Data Insertion**

Initial sample data was inserted into the database to validate its functionality.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**4.3 Database Testing**

The database was tested to ensure all relationships and constraints functioned correctly. Examples of tests:

1. **Foreign Key Constraint Test**: Attempting to delete a CustomerID that has existing Orders results in an error, demonstrating that the foreign key constraint is working.
2. **Data Integrity Test**: Trying to insert a duplicate Email into the Customers table results in error.

**4.4 Database Diagram**

A data diagram was generated to visualize the relationships and structure of the tables. This diagram matches the previously defined ERD and reflects the implemented system.

Obraz zawierający tekst, diagram, zrzut ekranu, Równolegle

Opis wygenerowany automatycznie

# **Part 2: Database design development**

This section focuses on the development and testing of the SQLite database system. The goal is to implement advanced database features such as triggers and views.

## **1.Triggers**

Triggers are used automate specific actions within the database

**Trigger 1: Automating Order Processing and Payment Recording**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznieThis trigger ensures that whenever a new order is placed, the corresponding order details and payment records are automatically generated.

**Trigger 2: Real-Time Inventory Management**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**This trigger is activated after new order details are added. It automatically updates the stock levels of ingredients used to prepare the ordered menu items.

**Trigger 3: Financial Tracking Automation**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie**This trigger is executed whenever a new payment is recorded. It logs the transaction into the FinancialRecords table.

## **2.Views**

**1. Orders Summary**

**Obraz zawierający tekst, zrzut ekranu, wyświetlacz, Czcionka

Opis wygenerowany automatycznie**This view provides a detailed summary of all orders.

**2. Ingredient Stock Levels View**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**The IngredientStockLevels view shows the stock levels of all ingredients, including their unit of measure and associated supplier details.

**3. Menu Items with Ingredients View**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznieThe MenuItemsWithIngredients view lists all menu items alongside their respective ingredients, quantities, and menu names.

**4. Financial Summary View**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**The FinancialSummary view provides a concise overview of revenue and transactions grouped by record type.

**5. Financial Details View**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznie**The FinancialDetails view provides a detailed record of all financial transactions, sorted by date.

## 

## **3.Queries Meeting Business Requirements**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie**1. Query with a Message and JOIN**

**Input**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Output**

**2. Query with GROUP BY and HAVING**

**Input**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie**

**Output**

**3. Query Using SQLite Functions**

**Input**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie**

**Output**

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

**4. Query with a Message and JOIN**

**Input**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

**Output**

**5. Query that combine multiply features**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Output**

**Input**

## 

## **4.Testing and reviewing the database.**

To ensure that the database functions as intended, various tests were conducted to verify the integrity of the schema, relationships, and additional features such as triggers and views. The testing process included the following steps:

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznieObraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie**1.Trigger validation**

**Output**

**Input**

**Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie2.View validation**

**Input**

**Output**

Obraz zawierający tekst, Czcionka, zrzut ekranu, design

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie**3. Error Handling**

**Output**

**Input**

**4. Integrity Testing**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie**

**Input**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

**Output**

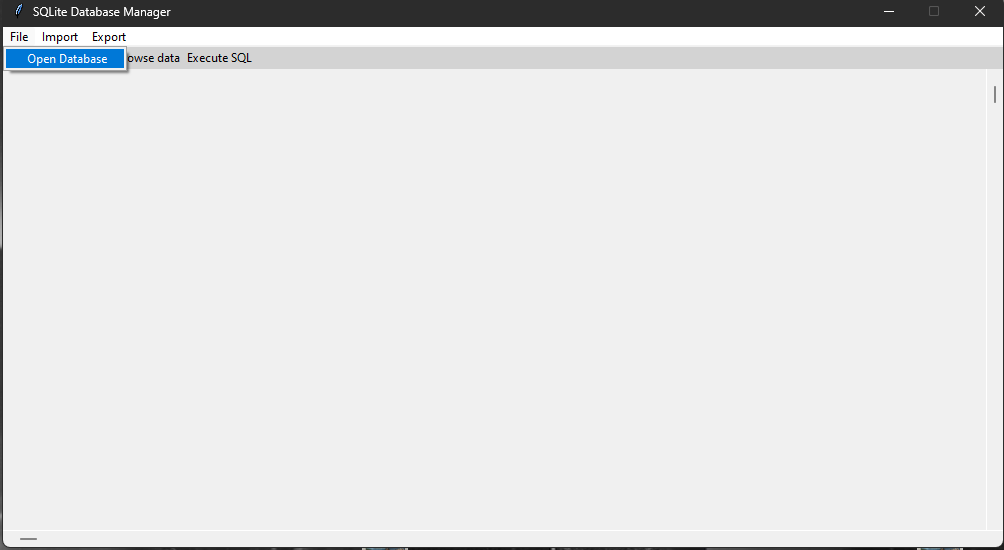
# **Part 3: Programming for Database and testing**

This section provides the user with a Python Tkinter software application for managing the database. It allows user to view, insert, update, delete data in the database and create reports.

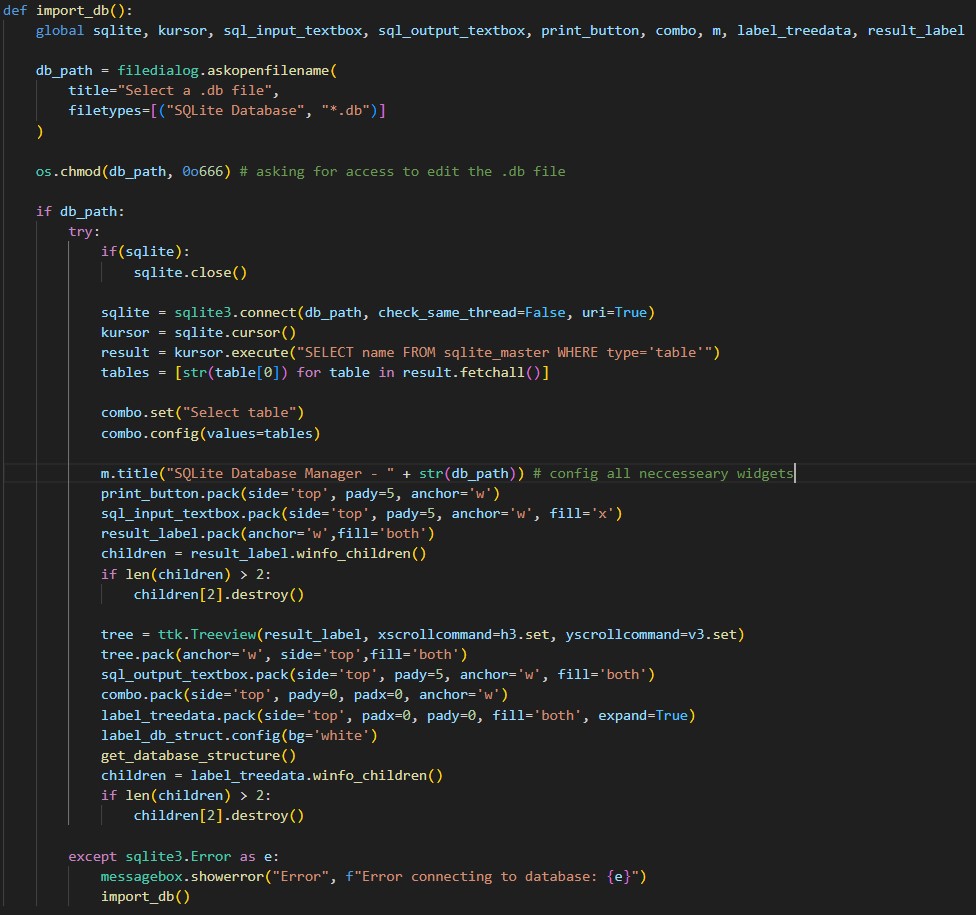
## **1.Communitation with SQLite by Python GUI**

After starting the application, the user sees a menu bar and a navigation bar with various options like:

**1. Opening database from SQLite file**

****

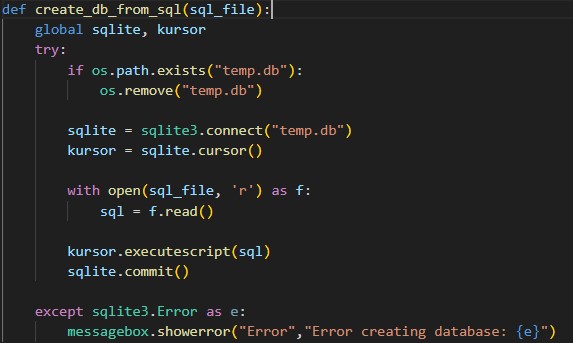
The “Open Database” command allows user to open the SQLite file with database.

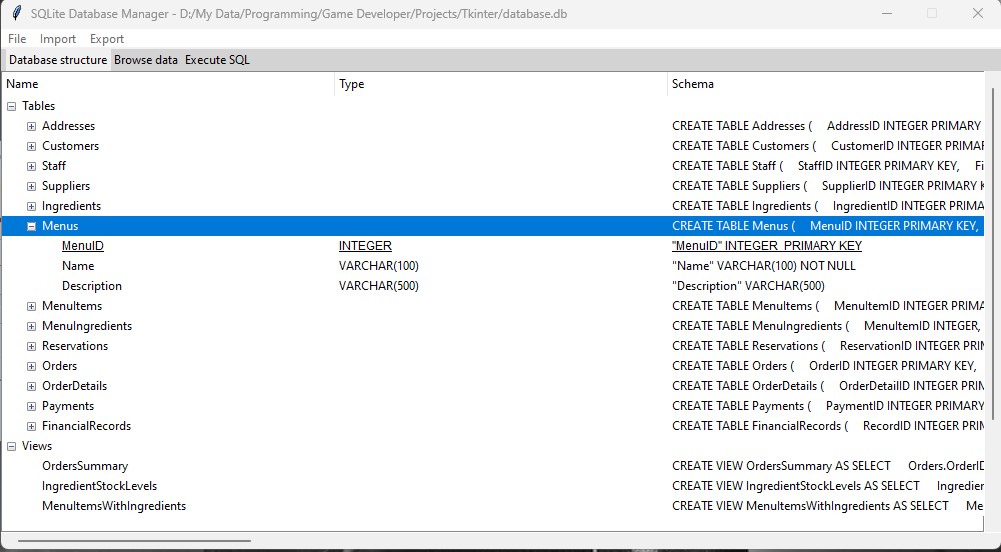

**2. Import database from SQL or XML File**

****

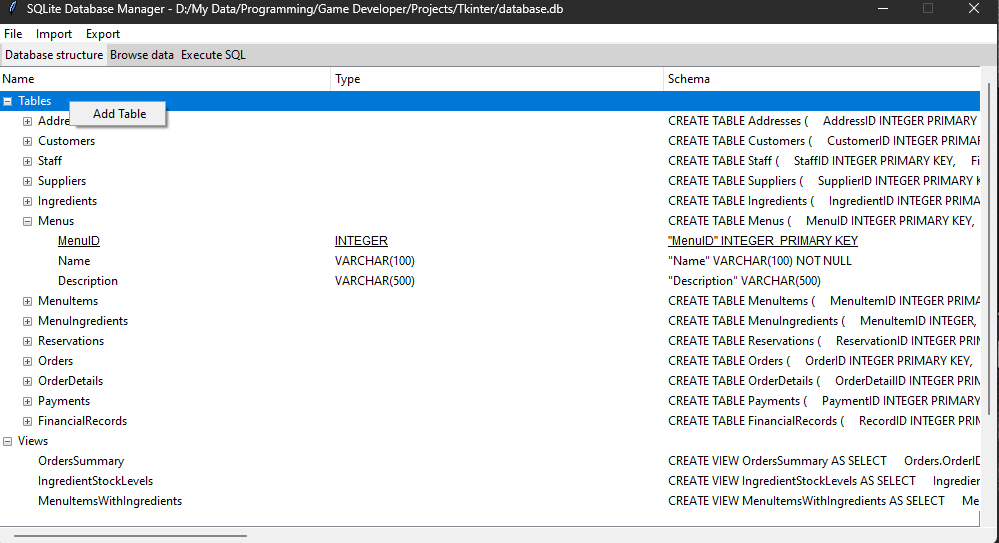
Commands under “Import” allow user to Import database either from SQL file or XML file.  
Note: If the database is imported from a non-SQLite file, the program creates a temporary SQLite database in the directory. If the user needs to preserve the changes to the database, it is necessary to export the database.

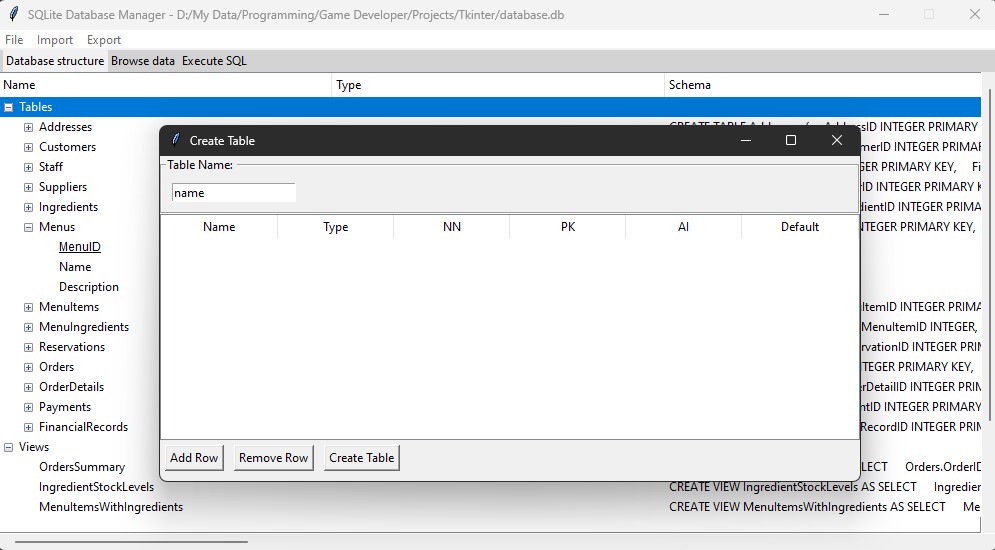


**3.Preview database structure**

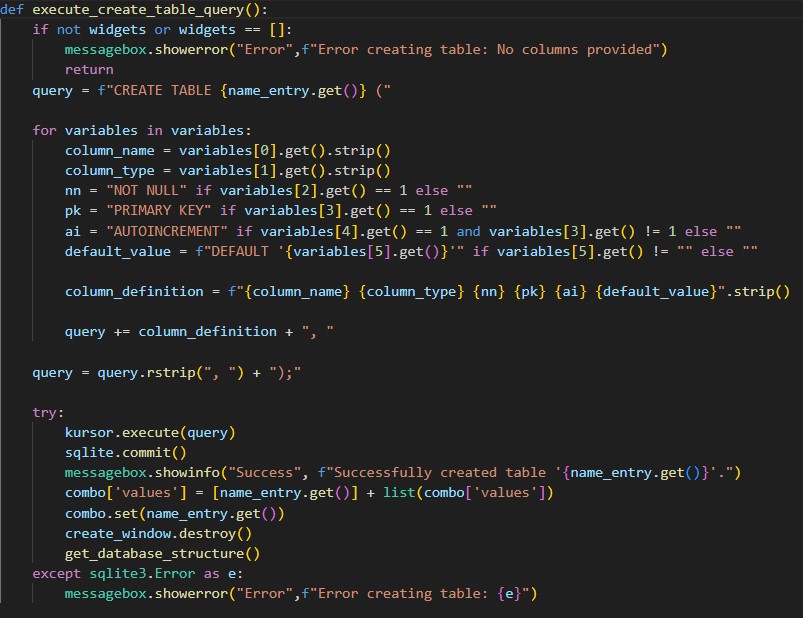
After opening or importing the database, the user has access to the database structure. This provides knowledge about all the tables, triggers, views in one place.

Additionally in this view, the user can insert a new table into the database by right-clicking on the “tables” node.



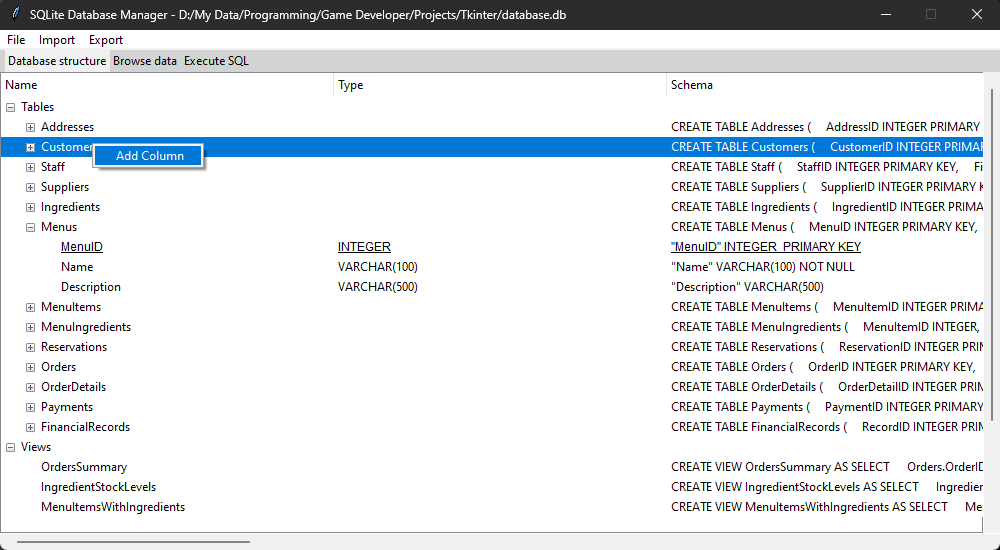


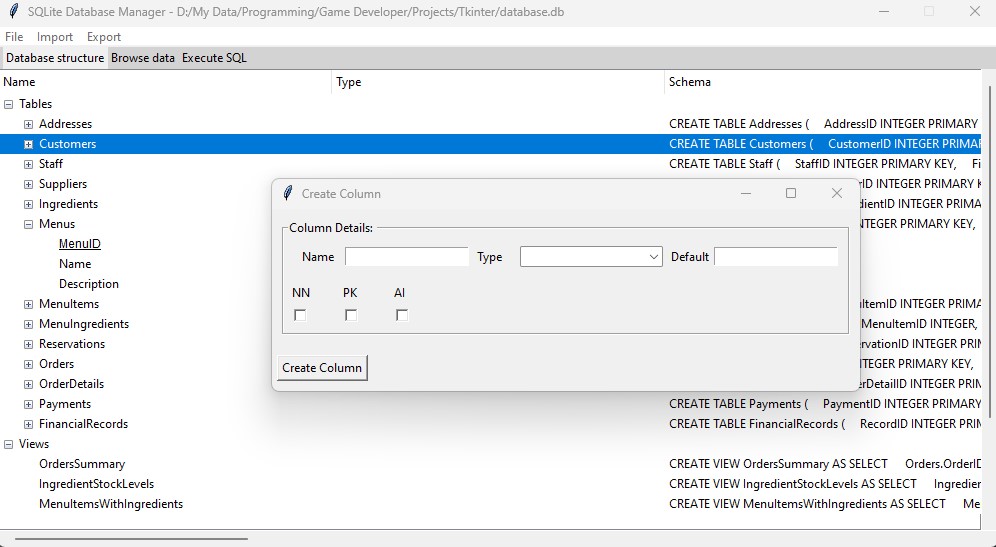
The user must provide a table name, add columns, name them and assign constraints attributes.



This is the code that interprets user-entered data and executes the query that creates a new table.

User is also given the option to add a column to an existing table:

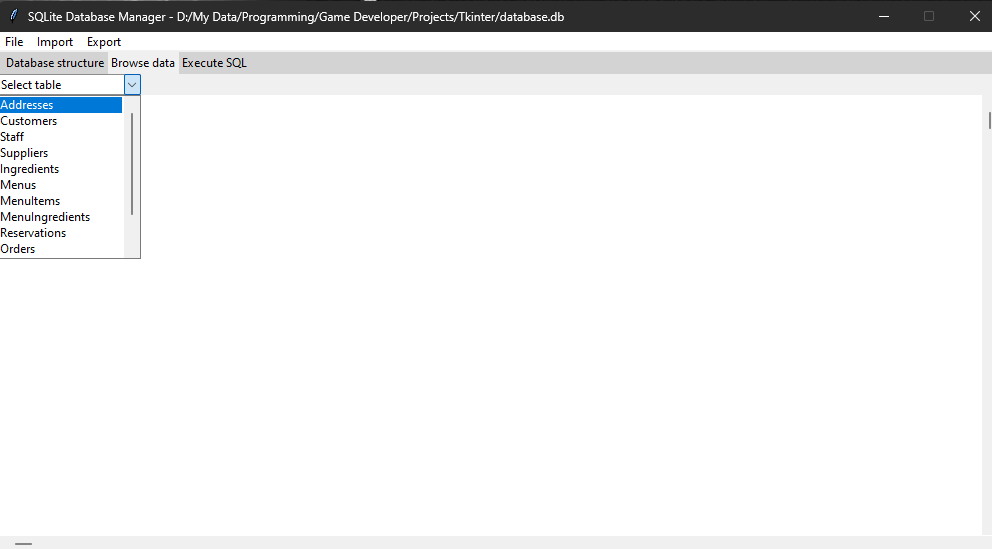


****

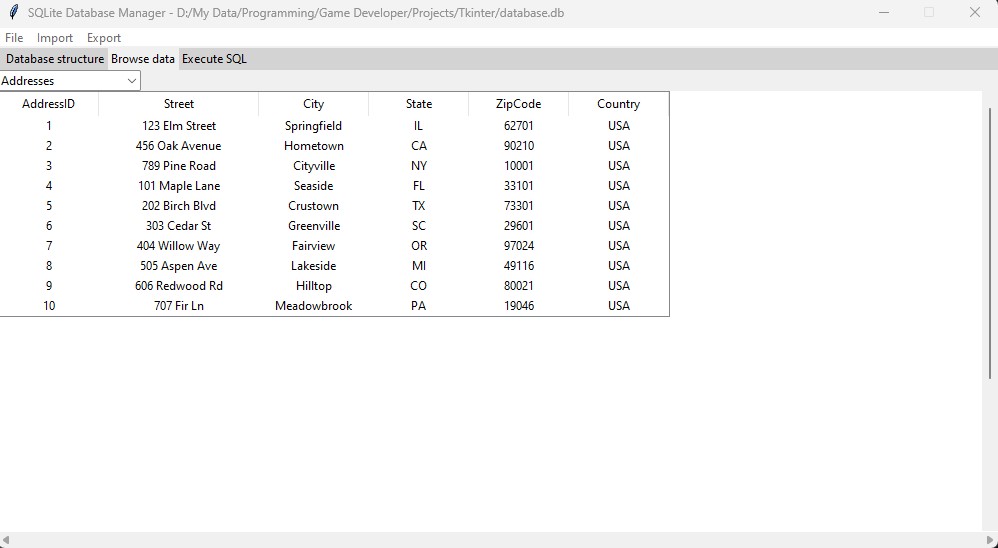
****

**4.Browse all data in the database**

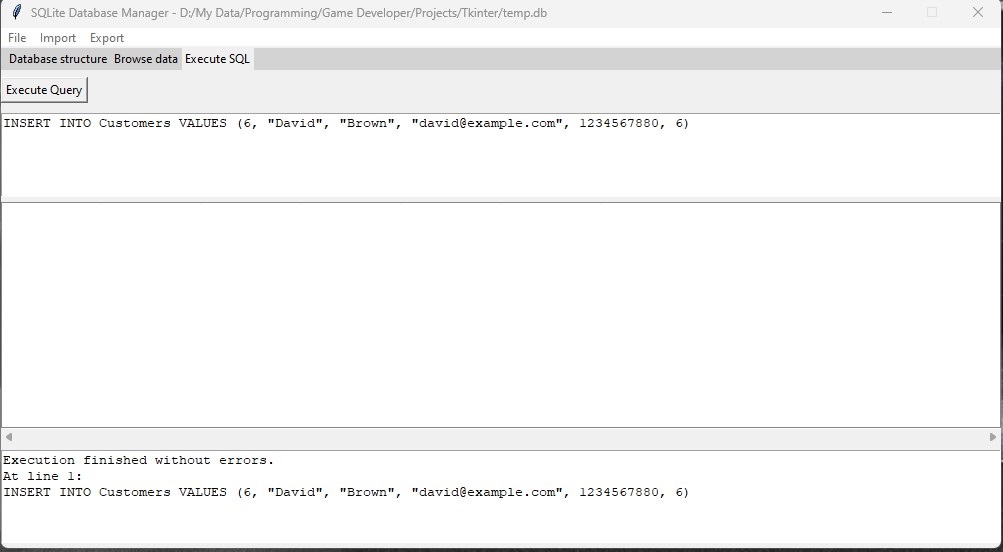
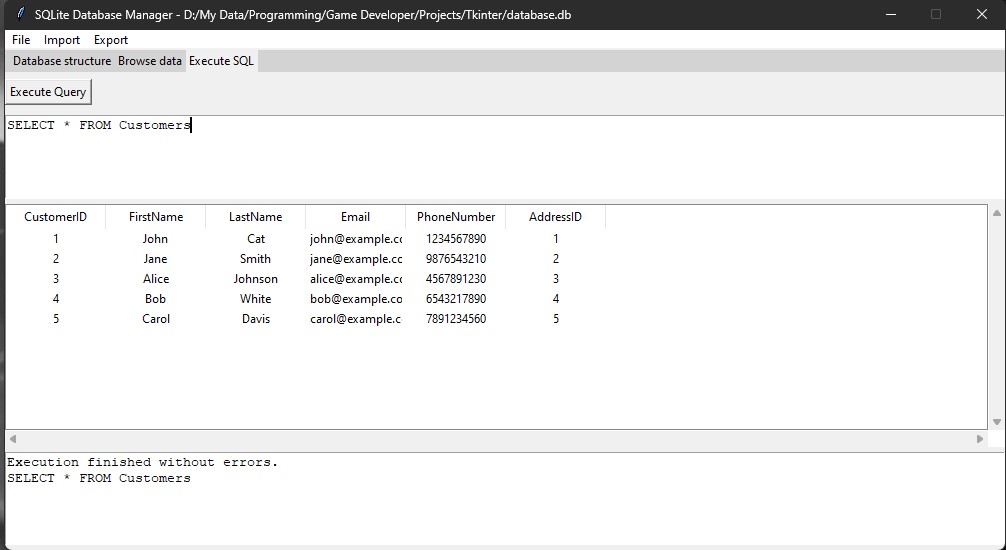
The user can also see all the tables in the database. By selecting one in the combobox, he has a preview of the table:



Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

1. **Executing SQL queries**

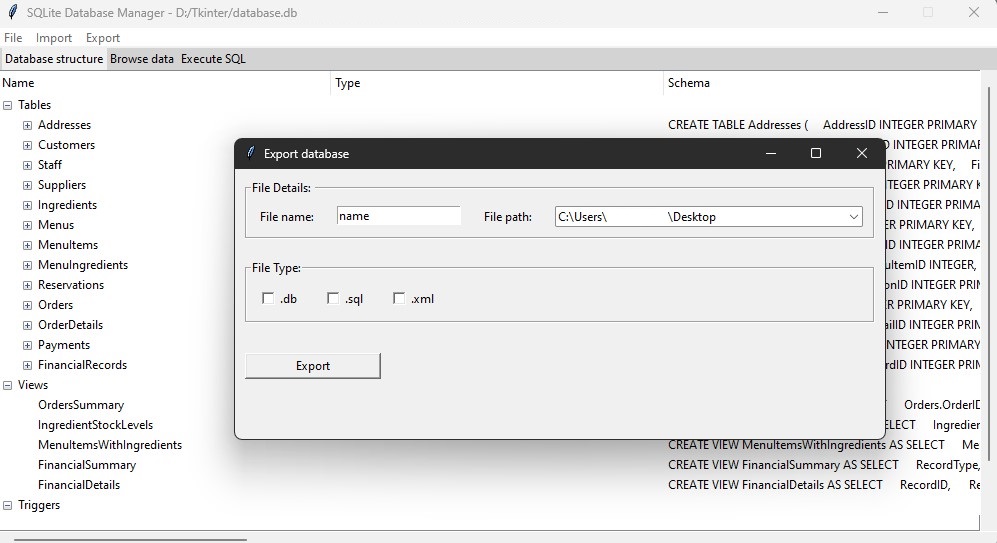
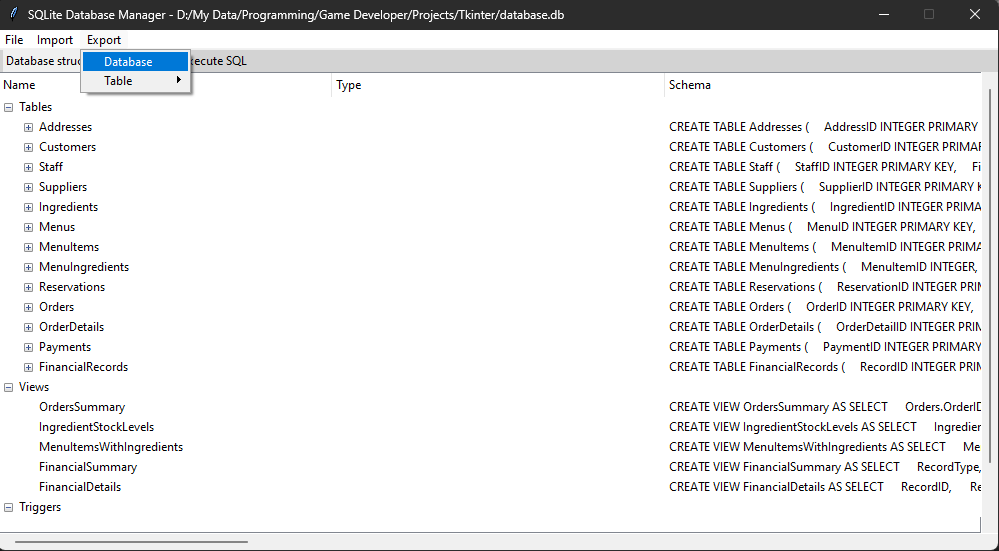
In the Execute SQL tab, the user can enter SQL queries and execute them. When the query syntax is correct, the operations are executed and a positive message is returned in the text box. If the query (e.g. SELECT) returns a result, a table with the appropriate records will be visible in the field between the input textbox and the output textbox.



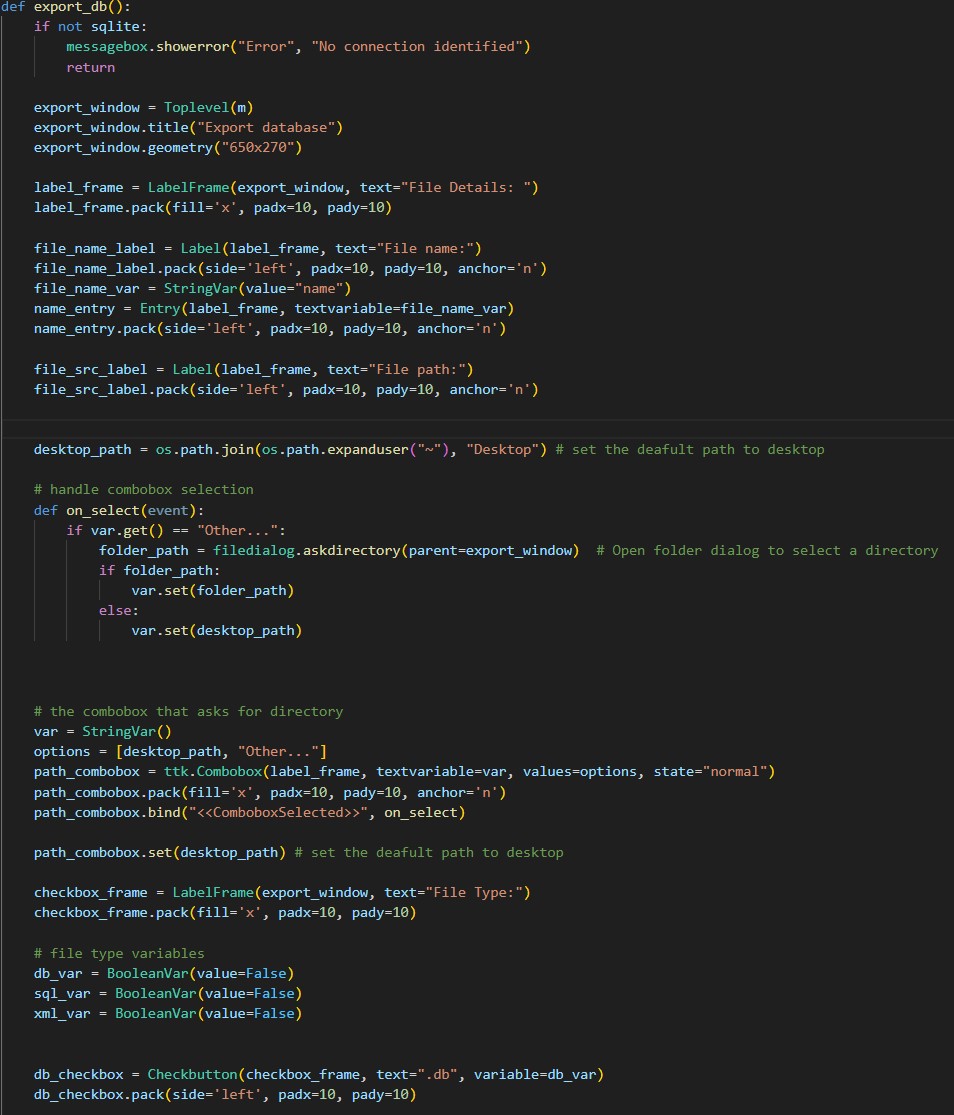
Obraz zawierający tekst, zrzut ekranu

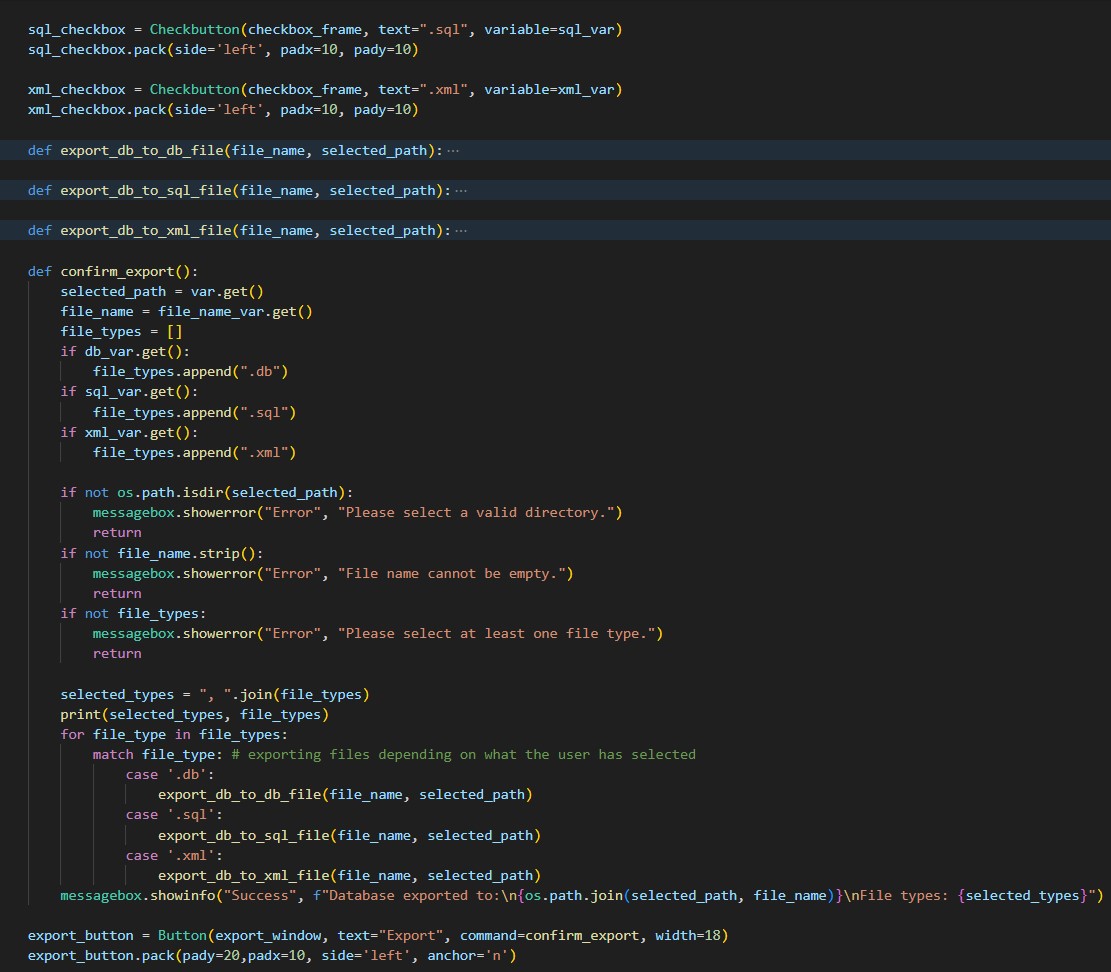
Opis wygenerowany automatycznie

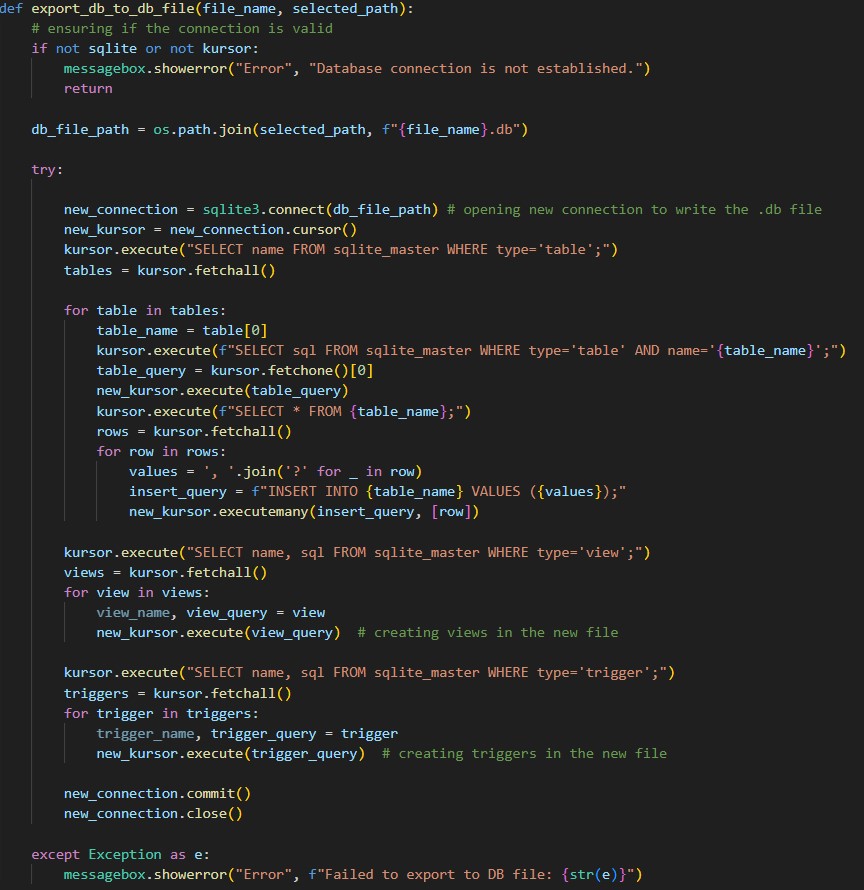
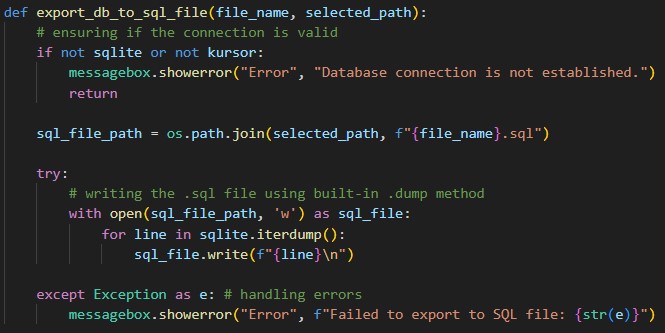
1. **Export database to SQLite, SQL or XML file:**

By clicking “Database” in the “Export” user can export database to either SQLite, SQL or XML file or multiple at once. 

The user must provide the file name, path where the file or files would be saved and the extension. After clicking the export button, a message will be displayed whether the export was successful.

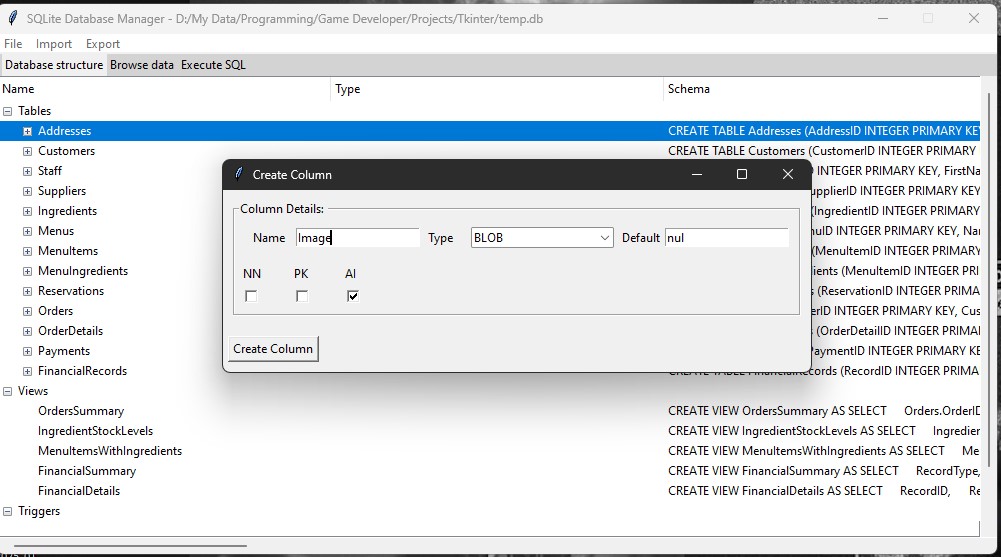


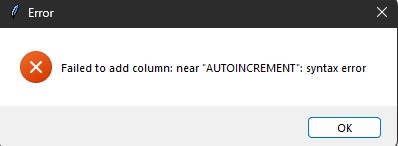


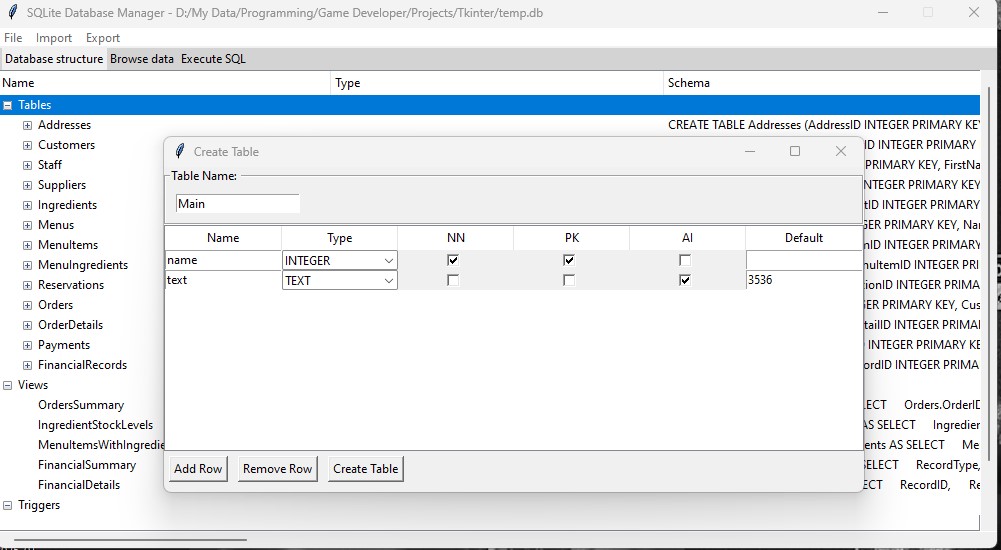
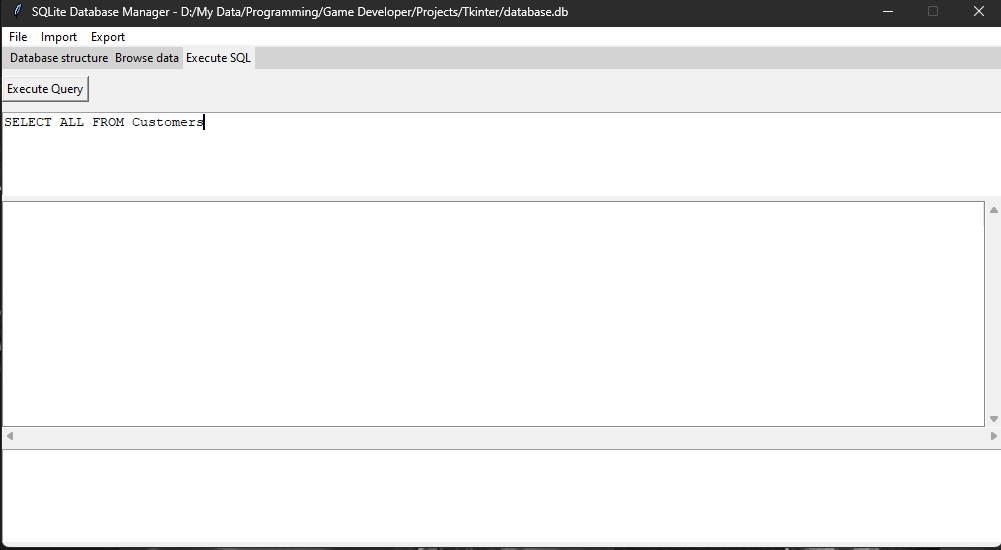
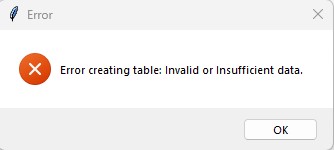


## **2.Validation or verification of input data**

As mentioned before, if the user enters incorrect data for creating a table or column, or enters a query with incorrect syntax, the program will inform the user about the error and will not perform the operation until the data is correct.

Examples:

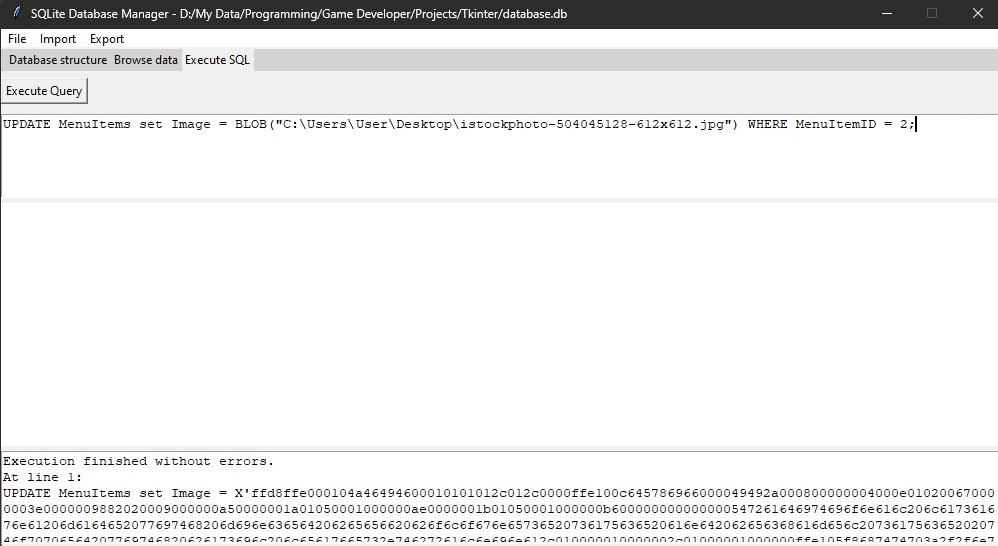




## **3.Stroing images in database using BLOB data type**

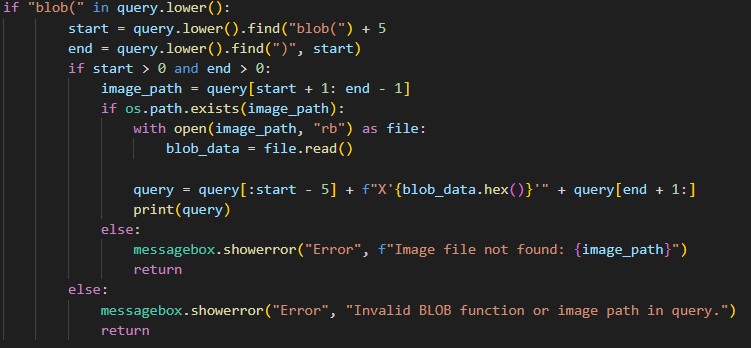
When entering an UPDATE query for a column with the BLOB data type, the new value assigned by the user should use the BLOB function. Its specifically designed for program usability and simplicity, and takes the file path as an argument.

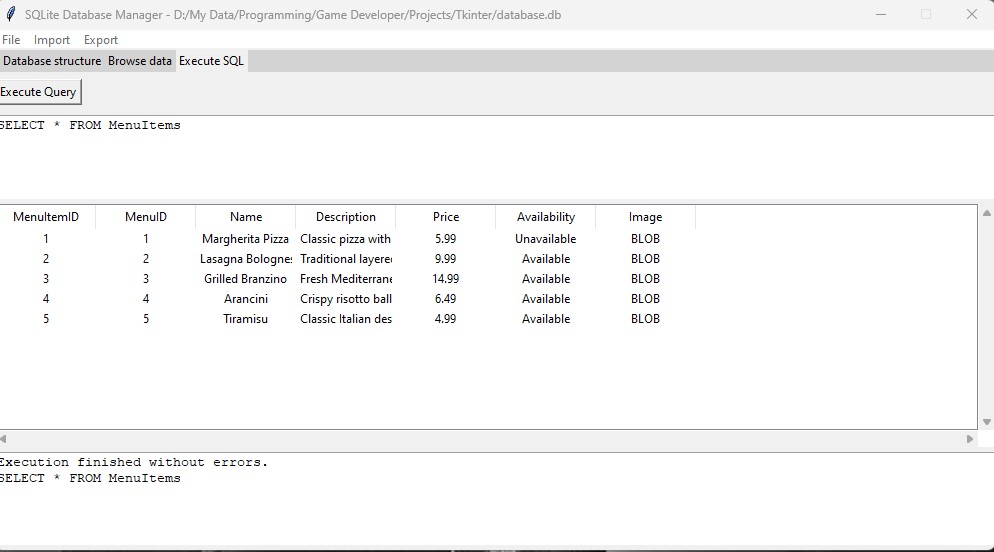
Here is an example of correct usage:



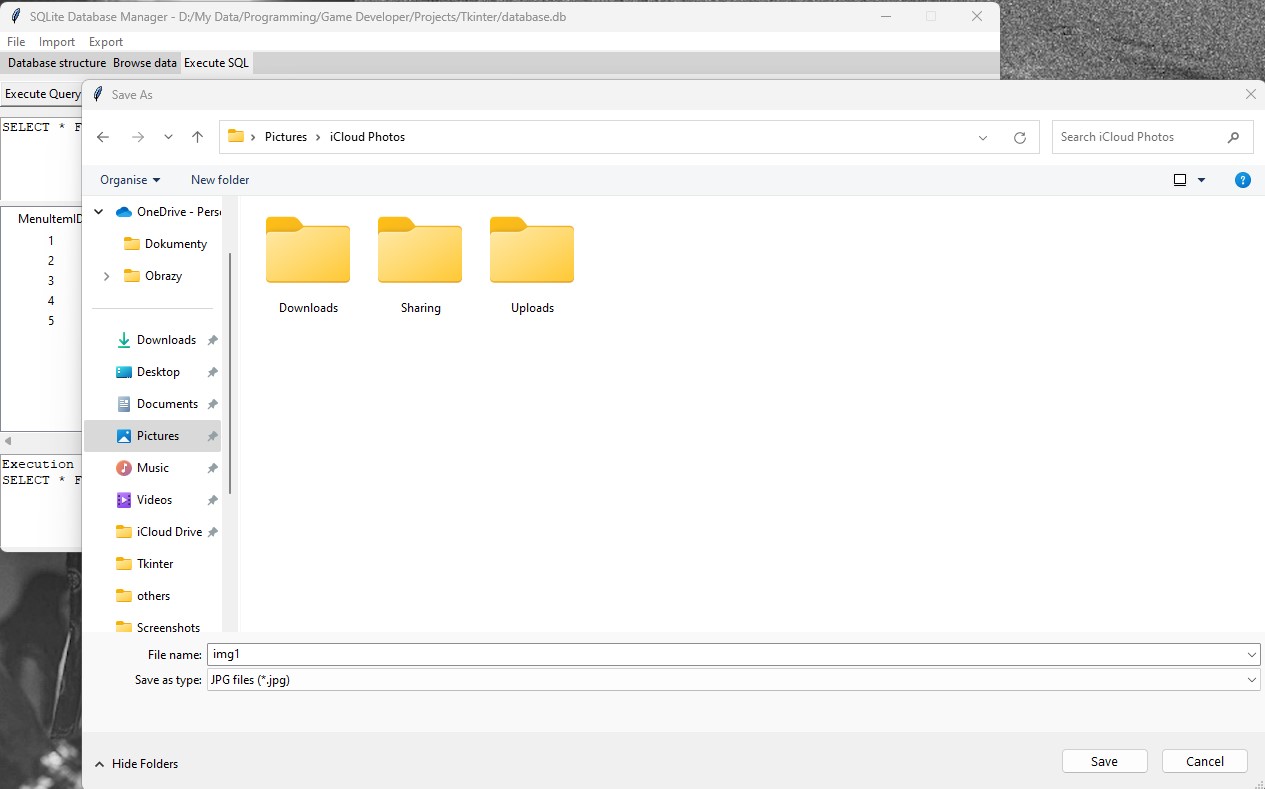
In this example, in the output textbox, the user can see the actual query that was compiled by the program. The user inserted a blob object into the Image column.

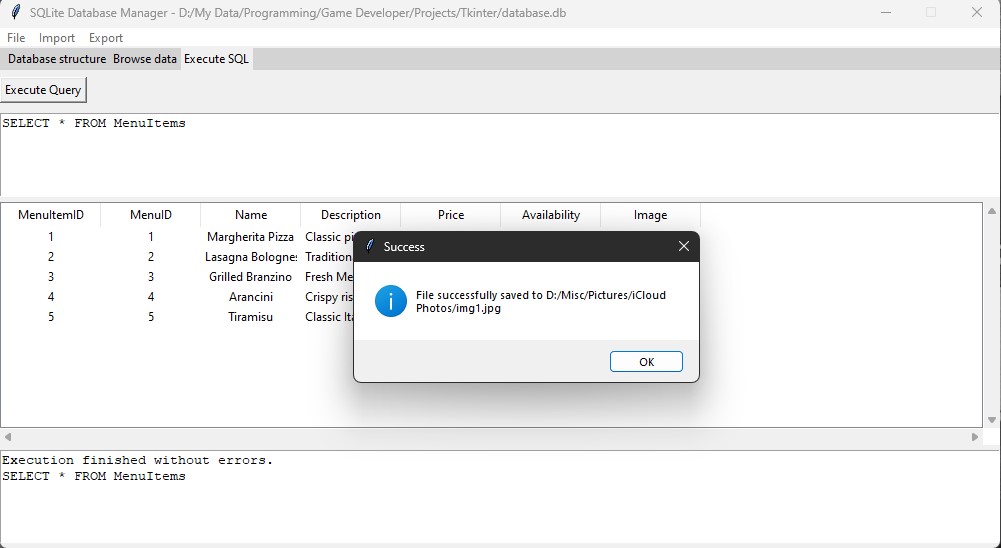
Here is the code that is inserting the BLOB file to the database:

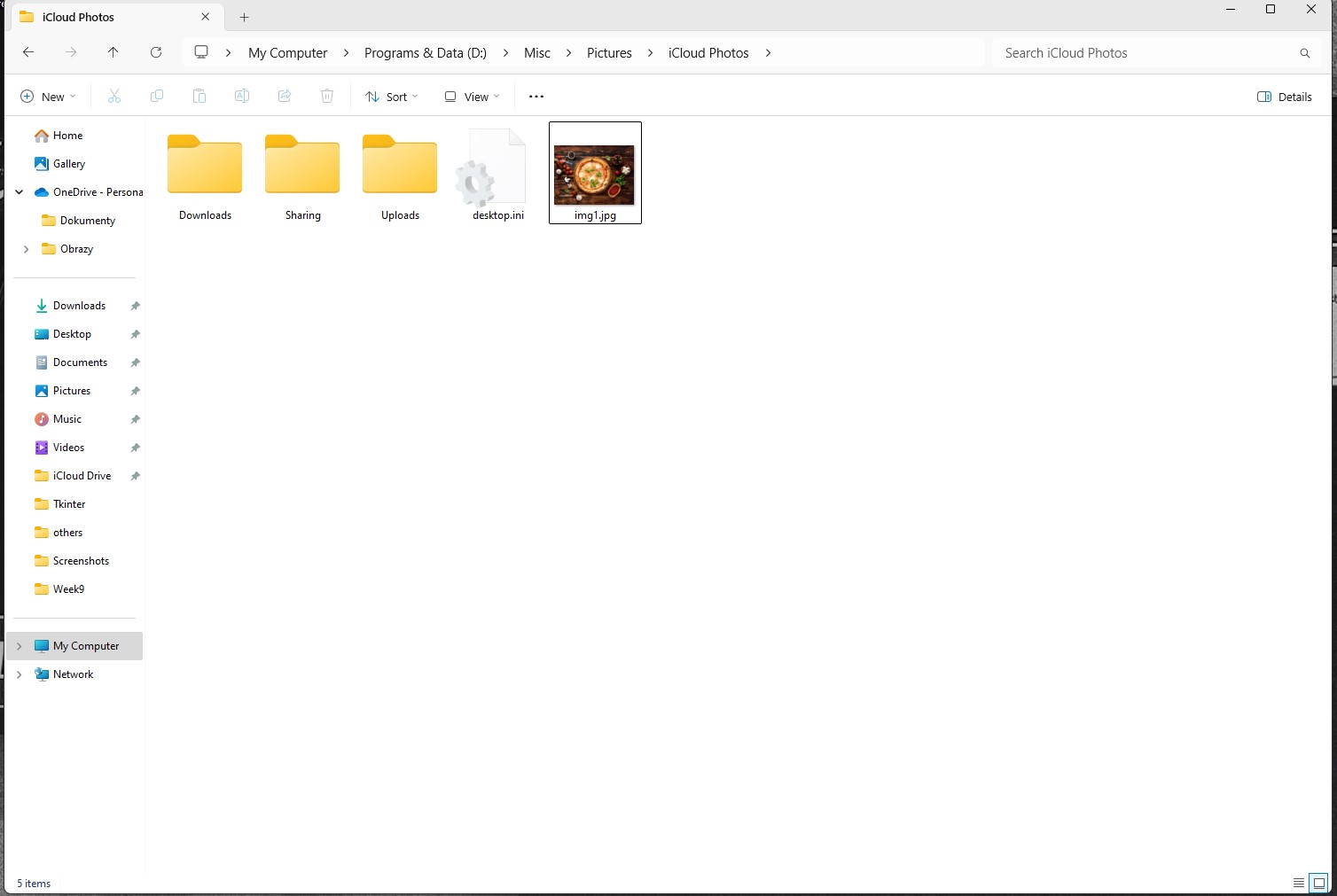
  
  
When a user would like to read a blob file, SELECT query

that has this data type can be executed:

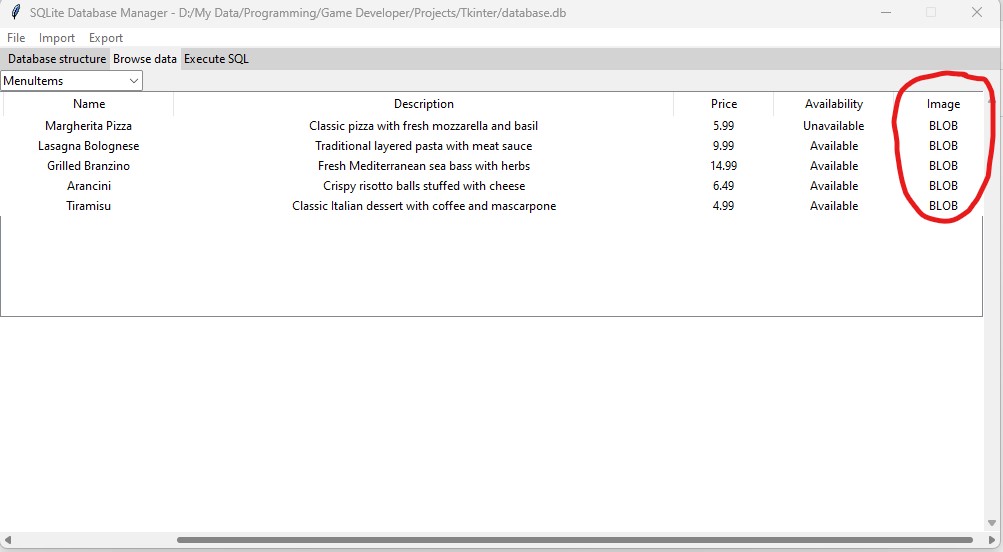
In the data place there is a BLOB placeholder which, when clicked with left mouse button, downloads the blob file that is saved in the given cell.

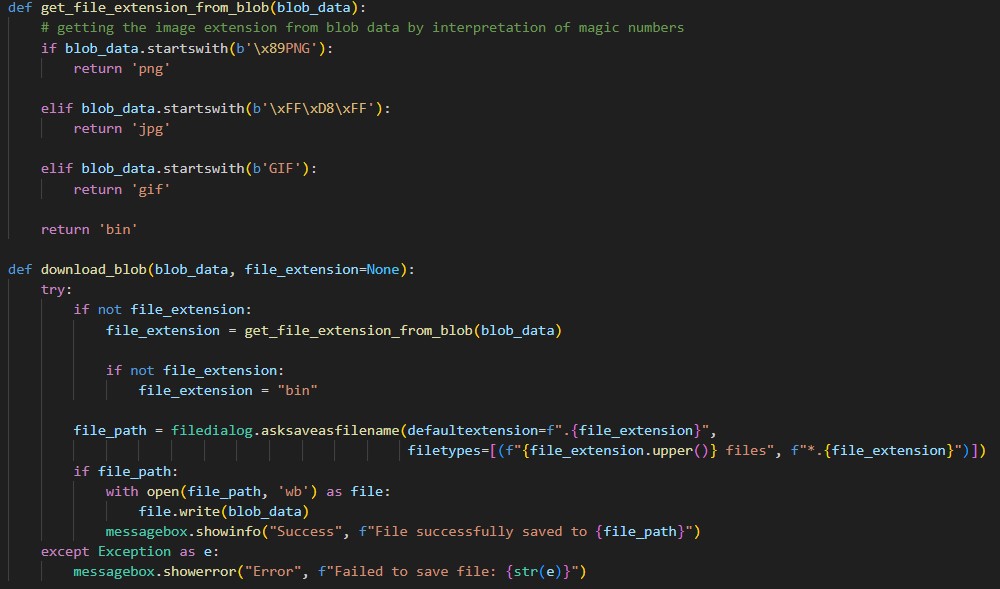






Same goes for tables in “Browse data”:

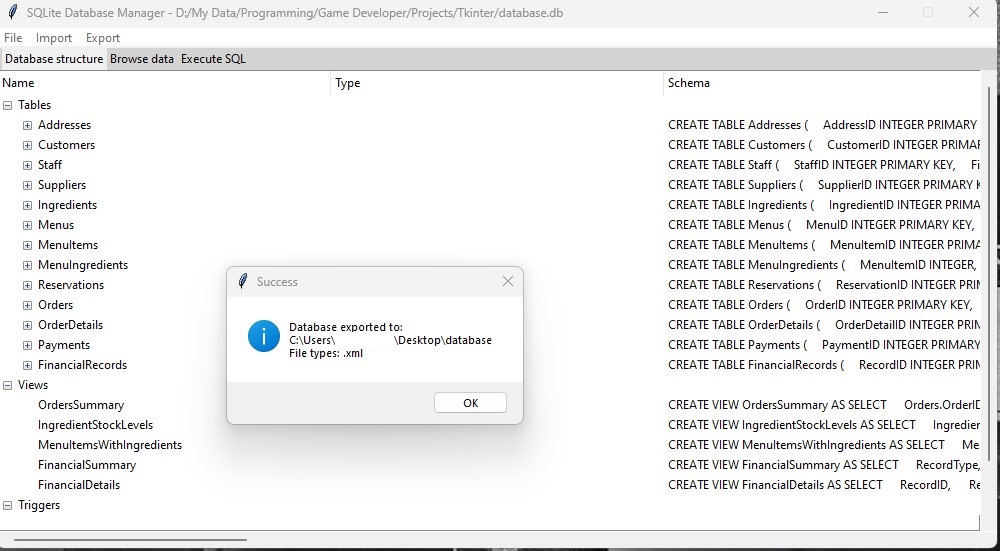
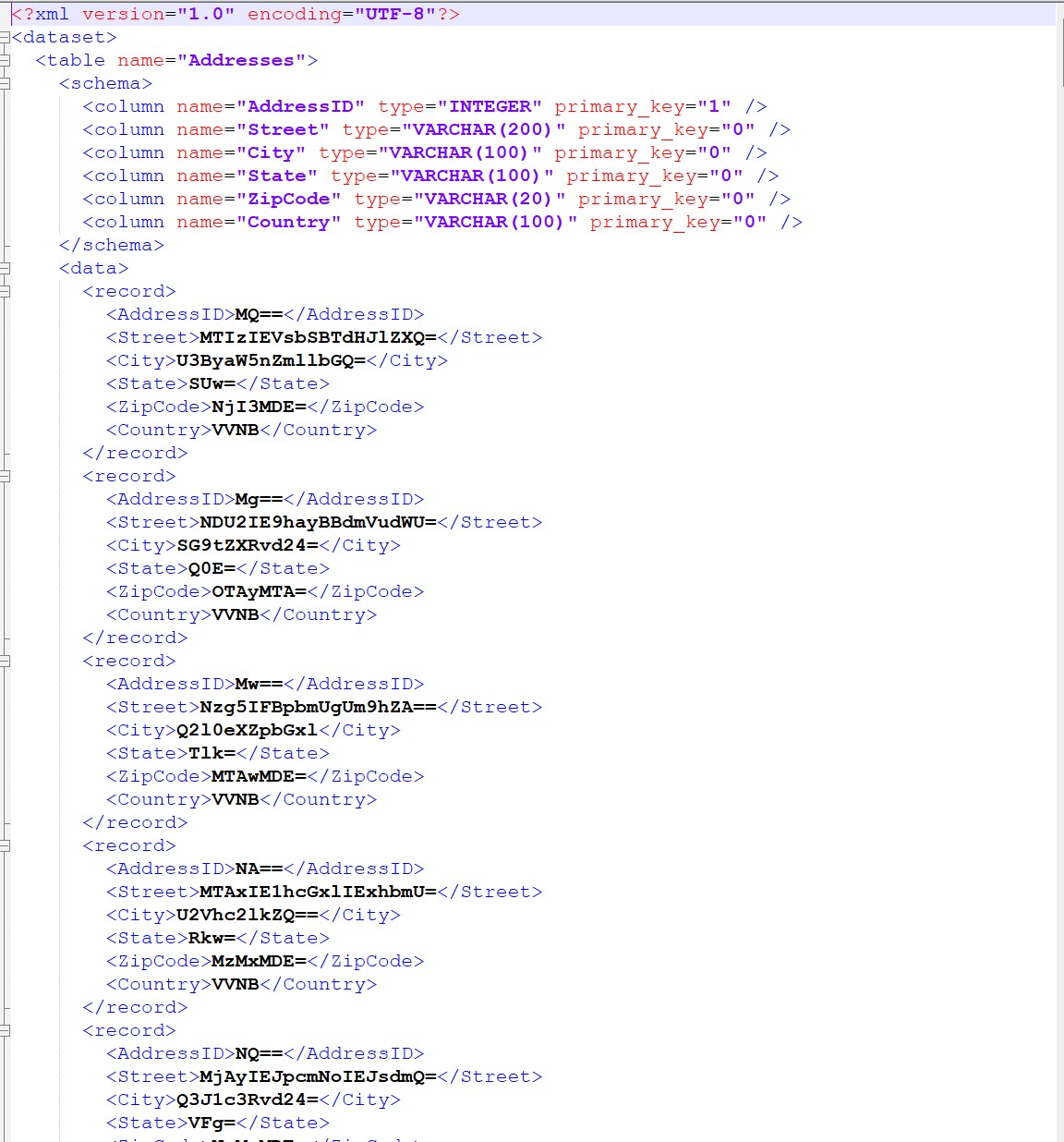




## **4. XML Development (and Encryption)**

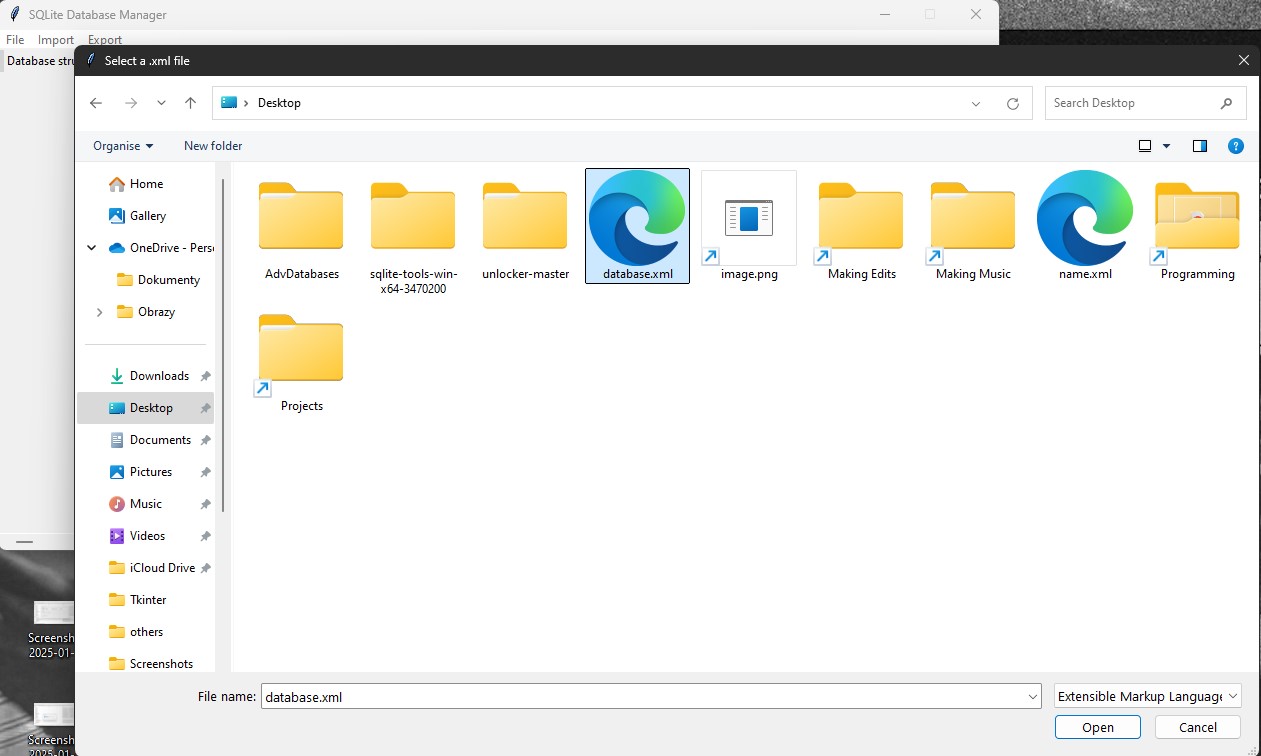
The application supports exporting the database in XML format importing the database in this format which is limited only to XML databases that have been exported by this application.

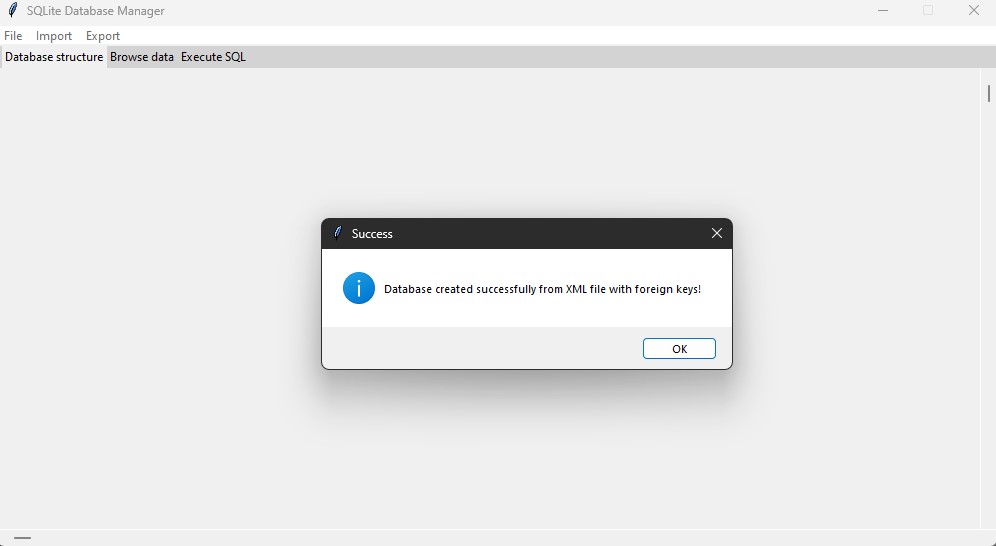


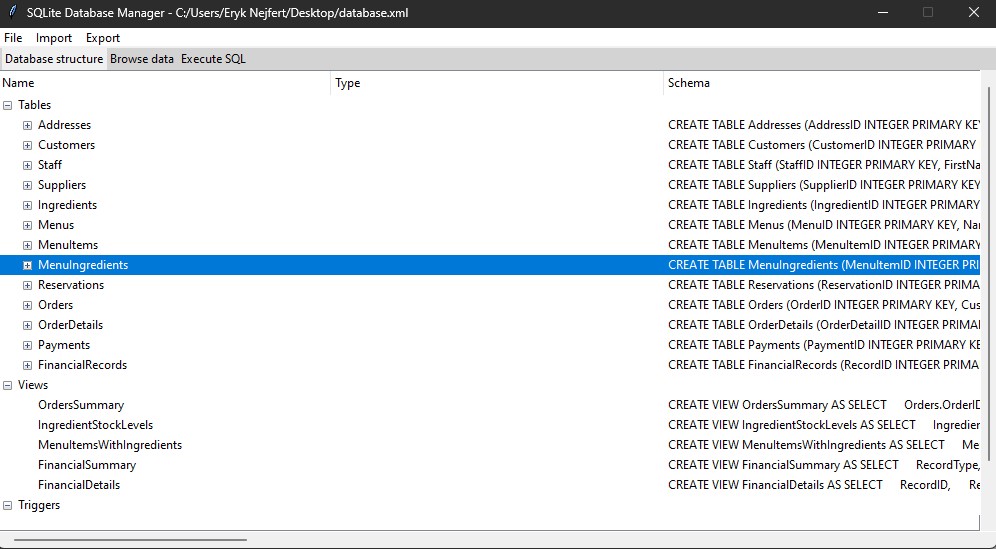
In database.xml:



When the user would like to import the database.xml:





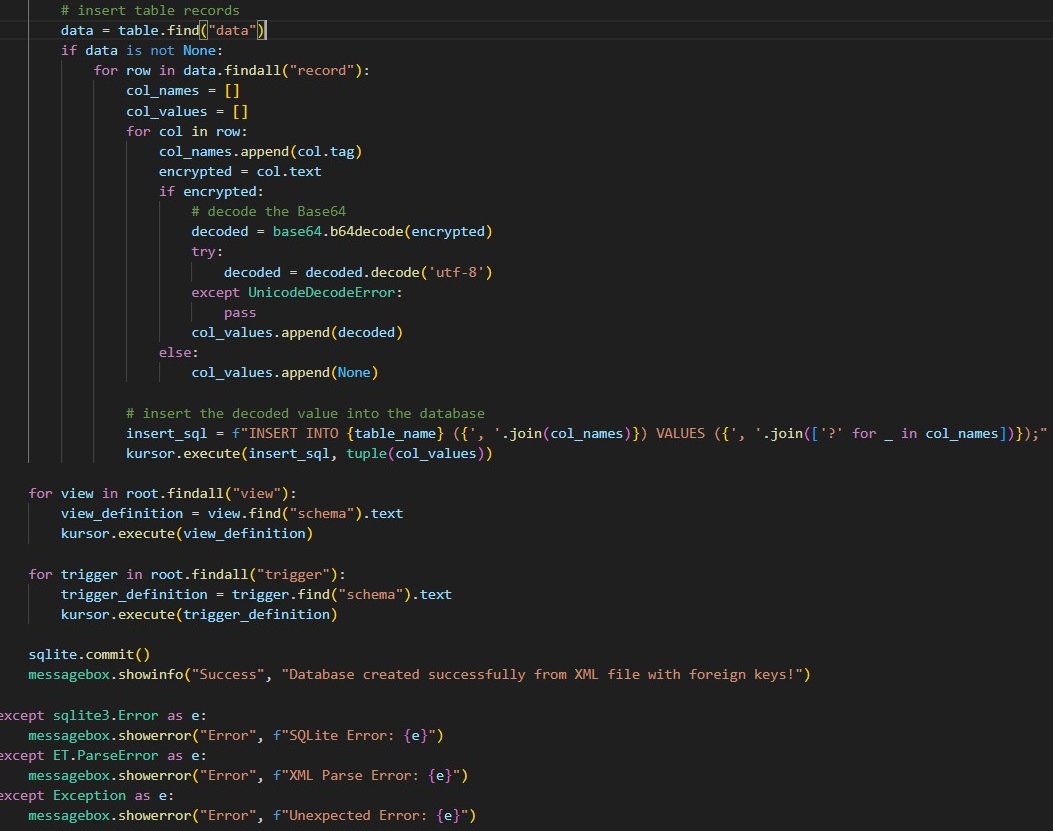


The program encodes the data in Base64 and then writes data in new XML file, and decodes data from XML file and reads data, then writes them in temporary SQLite database:



Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie



## **5.Innovative features in the application**

The application has many innovative functions and solutions that improve user comfort, such as inserting images into the database, adding tables and columns from the database structure view and the ability to download images from the database.

In addition, the application supports all the most popular data saving methods and efficiently operates between them.

Exporting table to CSV file:

