STOCK MANAGEMENT SYSTEM USING PYTHON 3 AND SQL

Stock Management System using MySQL and Python 3.12

Preface

The Stock Management System is a dynamic solution designed to enhance businesses' inventory control processes. Focused on efficient product tracking, streamlined supplier interactions, and accurate recordkeeping, the system provides a comprehensive platform for effective stock management. Implemented in Python and utilizing either the **sqlite3** or **mysql.connector** library, the system features a modular design, ensuring scalability and ease of maintenance. While the current command-line interface (CLI) offers straightforward interactions, future enhancements may include the development of a graphical user interface (GUI) for broader accessibility. The project's success lies in its ability to address inventory challenges, maintain supplier relationships, and provide an essential tool for businesses seeking to optimize their stock management processes.

1) Introduction

1.1 Project Background

The Stock Management System is a comprehensive solution designed to assist businesses in effectively managing their inventory, suppliers, and stock transactions. The system provides a robust platform for businesses to monitor stock levels, streamline supplier interactions, and maintain accurate records of stock movements.

1.2 Objectives

The primary objectives of the Stock Management System include:

- Efficiently track product inventory.
- Streamline interactions with suppliers.
- Record and manage stock transactions accurately.
- Provide a user-friendly interface for users to interact with the system.
- Enhance overall stock management processes within the business.

2) System Features

2.1 Product Management

- **Display Products:** Users can view a list of products stored in the system, including details such as product name, category, unit price, and current stock quantity.
- **Add/Update Products:** Authorized users can add new products to the system or update existing product information.

2.2 Supplier Management

- **Display Suppliers:** Users can access a comprehensive list of suppliers, including contact details such as supplier name, contact person, contact number, and email.
- **Add/Update Suppliers:** Authorized users can add new suppliers to the system or update existing supplier information.

2.3 Transaction Management

- **Display Transactions:** Users have the ability to view a log of all stock transactions, providing insights into product movements, quantities, and transaction dates.
- **Add Transactions:** Authorized users can record new stock transactions, specifying the product involved, the corresponding supplier, transaction type (in or out), and quantity.

3) Scope and Applications

3.1 Project Scope

a. Efficient Inventory Management:

The Stock Management System provides businesses with an effective tool to monitor and control their inventory. The system enables real-time tracking of product quantities, helping prevent stockouts and overstock situations.

b. Streamlined Supplier Interactions:

Businesses can maintain a comprehensive list of suppliers, including contact details and previous transactions. This streamlines communication and ensures that businesses can easily manage relationships with their suppliers.

c. Accurate Recordkeeping:

The system maintains a detailed log of stock transactions, including transaction types, quantities, and dates. This accurate recordkeeping is essential for auditing, analyzing trends, and making informed business decisions.

d. User-Friendly Interface:

While the current implementation uses a command-line interface (CLI), future enhancements can include the development of a graphical user interface (GUI), making it accessible to users who may not be comfortable with command-line interactions.

e. Modular Design for Scalability:

The system is designed with a modular architecture, allowing for easy scalability. Additional features and functionalities can be integrated without significant disruption to the existing codebase.

f. Cross-Platform Compatibility:

As a Python-based system, it offers cross-platform compatibility, allowing businesses to deploy the Stock Management System on various operating systems without major modifications.

3.1 Applications of the Project

The Stock Management System project can find applications in various industries and business sectors where effective inventory control is essential. Some of the potential applications include:

1. **Retail Businesses:**
   - Manage and track the stock levels of various products in retail stores.
   - Streamline interactions with suppliers for timely restocking of products.
2. **Manufacturing Companies:**
   - Monitor raw material and finished goods inventory.
   - Optimize production schedules based on real-time stock information.
3. **Distribution and Logistics:**
   - Track and manage inventory in warehouses for efficient distribution.
   - Streamline supplier relationships for timely and accurate deliveries.
4. **E-commerce Platforms:**
   - Keep track of diverse product offerings and their availability.
   - Integrate with order processing systems to manage stock levels.
5. **Healthcare Institutions:**
   - Monitor and manage medical supplies and pharmaceutical inventory.
   - Facilitate seamless communication with suppliers for timely restocking.
6. **Hospitality Industry:**
   - Manage stock levels of consumables in hotels and restaurants.
   - Coordinate with suppliers to ensure a continuous supply chain.
7. **Educational Institutions:**
   - Track and manage inventory of educational materials and supplies.
   - Coordinate with suppliers for timely procurement of resources.
8. **Small and Medium-sized Enterprises (SMEs):**
   - Provide an affordable and scalable stock management solution for smaller businesses.
   - Streamline inventory and supplier management for improved operational efficiency.
9. **Automotive Industry:**
   - Monitor and manage spare parts inventory in auto workshops.
   - Facilitate timely procurement of components from suppliers.

10. **Pharmaceutical Companies:**
    - Ensure accurate tracking of pharmaceutical products and their expiration dates.
    - Enhance communication with suppliers for the timely procurement of medicines.
11. **Construction Companies:**
    - Manage and monitor the availability of construction materials.
    - Coordinate with suppliers for the timely delivery of materials to project sites.

4) Architecture and Technologies

4.1 Database

The system utilizes a relational database to store and manage data. The database schema includes tables for Products, Suppliers, and Transactions, establishing relationships between them.

4.1.1. Database Schema

Products Table

- ProductID (Primary Key)
- ProductName
- Category
- UnitPrice
- StockQuantity

Suppliers Table

- SupplierID (Primary Key)
- SupplierName
- ContactPerson
- ContactNumber
- Email

Transactions Table

- TransactionID (Primary Key)
- ProductID (Foreign Key)
- SupplierID (Foreign Key)
- TransactionType (In/Out)
- Quantity
- TransactionDate

4.2 Programming Language

The Stock Management System is implemented in Python, leveraging the capabilities of the sqlite3 module for SQLite databases or the mysql.connector library for MySQL databases.

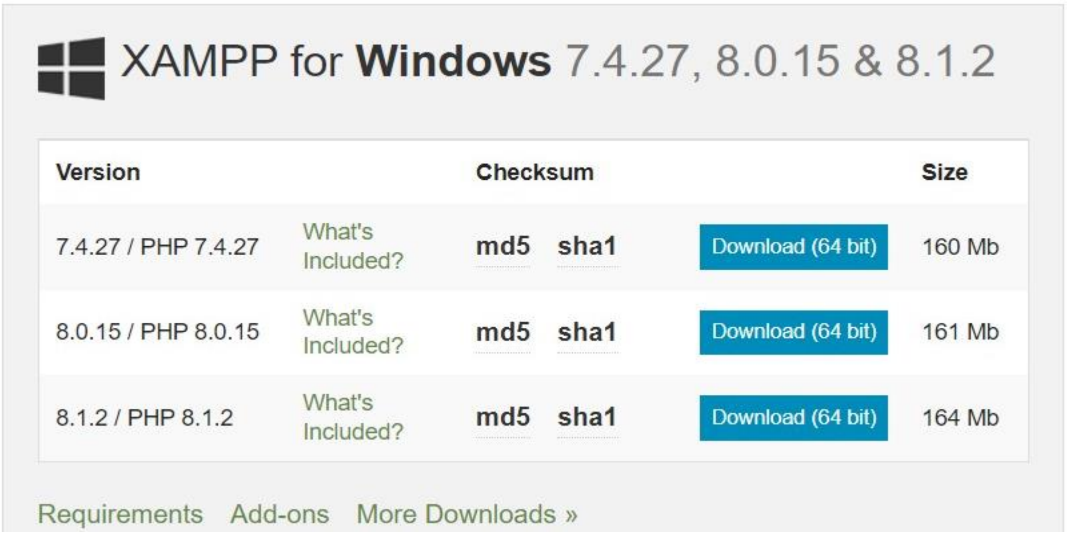
4.3 User Interface

Currently, the system operates through a command-line interface (CLI), providing a straightforward and efficient means of interaction. Future enhancements may include the development of a graphical user interface (GUI) for improved user experience.

5) Installing the Required Software and platform

The XAMPP server is an open source localhost web server platform which consists Apache, MySQL phpMyAdmin features that are help to convert the normal system into a server that can be accessible by those computers whose are connected with them via. Local area network.

First of all, we need to install the XAMPP application form their website – https://www.apachefriends.org/download.html

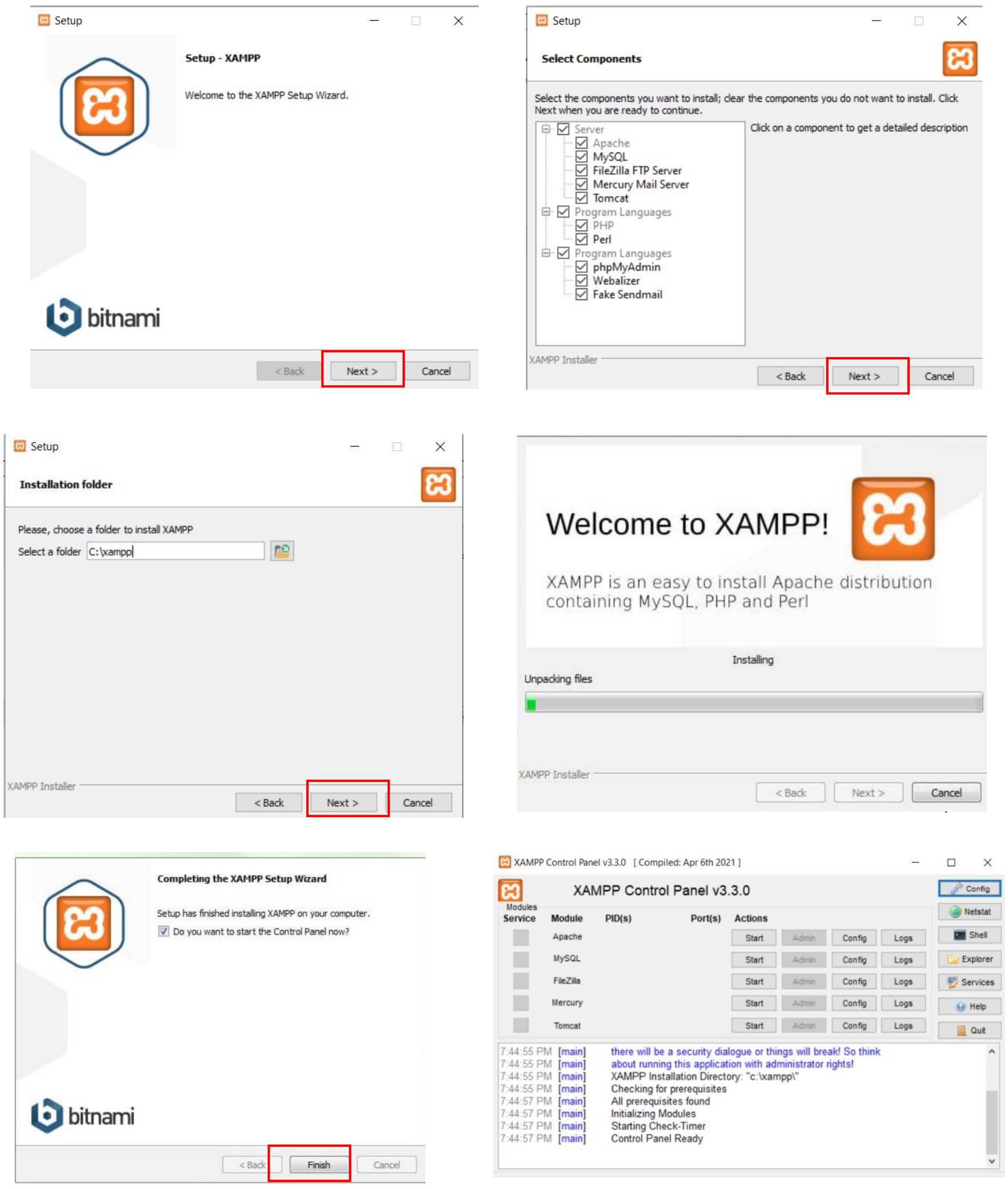

XAMPP for **Windows** 7.4.27, 8.0.15 & 8.1.2

Version		Checksum			Size
7.4.27 / PHP 7.4.27	What's Included?	md5	sha1	Download (64 bit)	160 Mb
8.0.15 / PHP 8.0.15	What's Included?	md5	sha1	Download (64 bit)	161 Mb
8.1.2 / PHP 8.1.2	What's Included?	md5	sha1	Download (64 bit)	164 Mb

Fig. 5.1: Download XAMPP

Download latest version of XAMPP with PHP 8.1.2 (164MB) and Run the file from downloads ‘xamppwindows-x64-8.1.2-2-VS16-installer.exe’ then click on next and accept terms and conditions. Then follow the process of installation.

-


Setup

Setup - XAMPP

Welcome to the XAMPP Setup Wizard.

Setup

Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

Click on a component to get a detailed description

XAMPP Installer

Setup

Installation folder

Please, choose a folder to install XAMPP

Select a folder: C:\xampp

XAMPP Installer

Welcome to XAMPP!

XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl

Installing

Unpacking files

XAMPP Installer

Completing the XAMPP Setup Wizard

Setup has finished installing XAMPP on your computer.

☑ Do you want to start the Control Panel now?

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

XAMPP Control Panel v3.3.0

Service	Module	PID(s)	Port(s)	Actions
	Apache			Start Admin Config Logs
	MySQL			Start Admin Config Logs
	FileZilla			Start Admin Config Logs
	Mercury			Start Admin Config Logs
	Tomcat			Start Admin Config Logs

```
7:44:55 PM  [main]   there will be a security dialogue or things will break! So think
7:44:55 PM  [main]   about running this application with administrator rights!
7:44:55 PM  [main]   XAMPP Installation Directory: "c:\xampp\"
7:44:55 PM  [main]   Checking for prerequisites
7:44:57 PM  [main]   All prerequisites found
7:44:57 PM  [main]   Initializing Modules
7:44:57 PM  [main]   Starting Check-Timer
7:44:57 PM  [main]   Control Panel Ready
```

Fig. 5.2: Installation process of XAMPP

The above screenshot shows the complete installation process of XAMPP server in windows OS. This platform needs APACHE and MySQL server, so we have to start both of these services as we have to use them for this project.

The apache server tends to execute the localhost server and MySQL server, a database service that are used to create and manipulate the database using GUI as well as CUI. The xampp application is a good choice for the developers to work with localhost and the database. The project takes both APACHE localhost and PhpMyAdmin for connecting to the database.

5.1. Starting APACHE and MySQL server

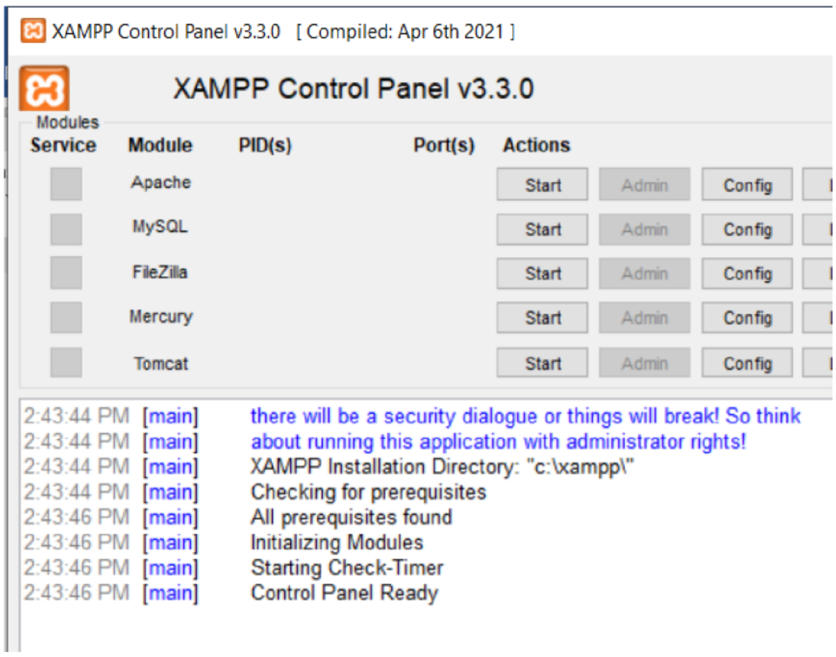
For starting the APACHE and MySQL server, the XAMPP control panel is required. In windows system, we can directly open XAMPP control from the installation path. For Mac OS, access the application menu by pressing command key and type XAMPP control panel. And for Linux, the installation path is must for opening the XAMPP control panel.

7. Python Database Connection

To connect Python to a MySQL database using the **mysql.connector** library, we first need to install the library using the following:

```
pip install mysql-connector-python
```

Once the library installed, we can use the following code as an example to establish a



```
2:43:44 PM  [main]   there will be a security dialogue or things will break! So think
2:43:44 PM  [main]   about running this application with administrator rights!
2:43:44 PM  [main]   XAMPP Installation Directory: "c:\xampp\"
2:43:44 PM  [main]   Checking for prerequisites
2:43:46 PM  [main]   All prerequisites found
2:43:46 PM  [main]   Initializing Modules
2:43:46 PM  [main]   Starting Check-Timer
2:43:46 PM  [main]   Control Panel Ready
```

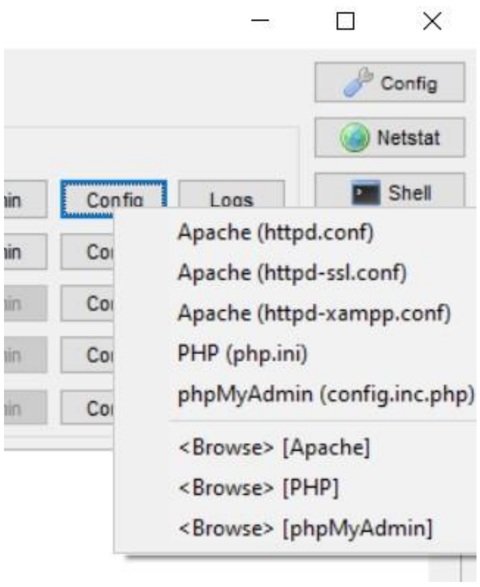


Fig. 5.3

In above figure, there are 5 servers to facilitate the module, i.e. Apache, MySQL, FileZilla, Mercury, Tomcat. The given project uses only two modules that are APACHE and MySQL.

connection to a MySQL database:

```python
import mysql.connector

try:
    # Establishing a connection
    connection = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="StockManagement"
    )

    if connection.is_connected():
        print("Connected to MySQL database")

        # Perform database operations here

except mysql.connector.Error as e:
    print(f"Error: {e}")
```

Output:

```
Connected to MySQL database
```

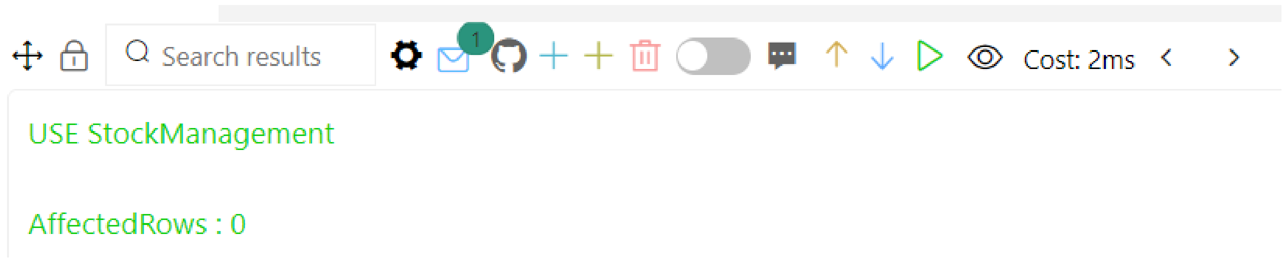
SOURCE CODE:

File://stockManagement.sql

```sql
-- Create a database for stock management
CREATE DATABASE IF NOT EXISTS StockManagement;
```

```
Query OK 0 Row(s) Affected
```

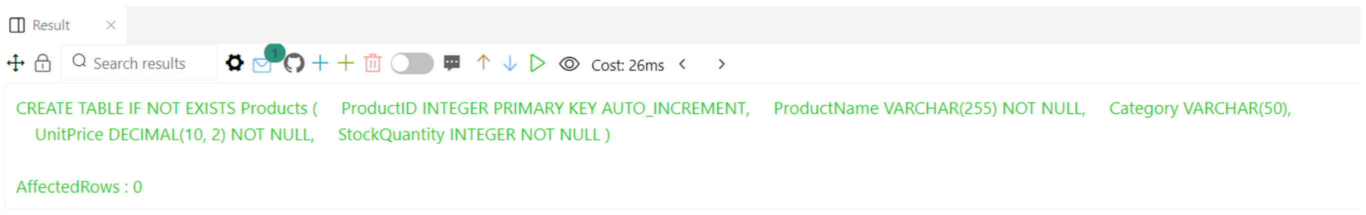
```sql
-- Use the created database
USE StockManagement;
```



```
USE StockManagement

AffectedRows : 0
```

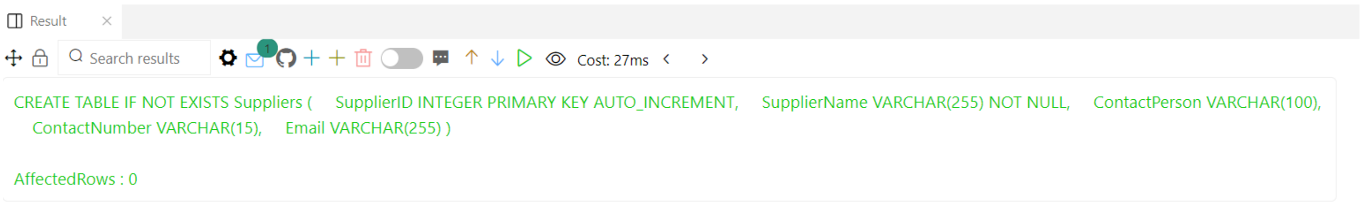
```sql
-- Create a table for products
CREATE TABLE IF NOT EXISTS Products (
    ProductID INTEGER PRIMARY KEY AUTO INCREMENT,
    ProductName VARCHAR(255) NOT NULL,
    Category VARCHAR(50),
    UnitPrice DECIMAL(10, 2) NOT NULL,
    StockQuantity INTEGER NOT NULL
);
```



```
CREATE TABLE IF NOT EXISTS Products (    ProductID INTEGER PRIMARY KEY AUTO_INCREMENT,    ProductName VARCHAR(255) NOT NULL,    Category VARCHAR(50),
    UnitPrice DECIMAL(10, 2) NOT NULL,    StockQuantity INTEGER NOT NULL )

AffectedRows : 0
```

```sql
-- Create a table for suppliers
CREATE TABLE IF NOT EXISTS Suppliers (
    SupplierID INTEGER PRIMARY KEY AUTO INCREMENT,
    SupplierName VARCHAR(255) NOT NULL,
    ContactPerson VARCHAR(100),
    ContactNumber VARCHAR(15),
    Email VARCHAR(255)
);
```

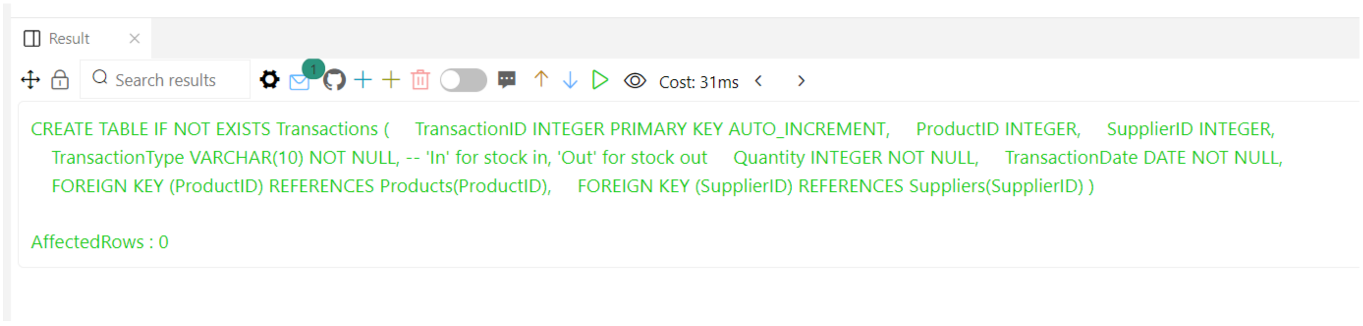


Result

CREATE TABLE IF NOT EXISTS Suppliers (SupplierID INTEGER PRIMARY KEY AUTO_INCREMENT,   SupplierName VARCHAR(255) NOT NULL,   ContactPerson VARCHAR(100),   ContactNumber VARCHAR(15),   Email VARCHAR(255))

AffectedRows : 0

```
-- Create a table for transactions
CREATE TABLE IF NOT EXISTS Transactions (
    TransactionID INTEGER PRIMARY KEY,
    ProductID INTEGER,
    SupplierID INTEGER,
    TransactionType VARCHAR(10) NOT NULL, -- 'In' for stock in, 'Out' for stock out
    Quantity INTEGER NOT NULL,
    TransactionDate DATE NOT NULL,
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID),
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)
);
```

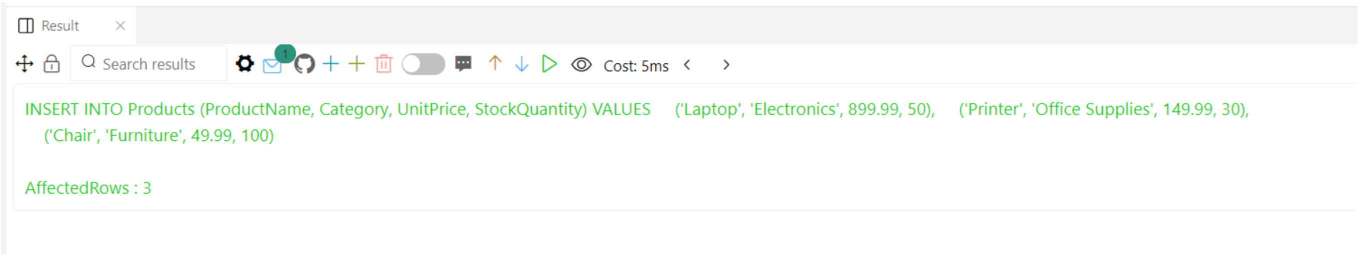


Result

CREATE TABLE IF NOT EXISTS Transactions (TransactionID INTEGER PRIMARY KEY AUTO_INCREMENT,   ProductID INTEGER,   SupplierID INTEGER,   TransactionType VARCHAR(10) NOT NULL, -- 'In' for stock in, 'Out' for stock out   Quantity INTEGER NOT NULL,   TransactionDate DATE NOT NULL,   FOREIGN KEY (ProductID) REFERENCES Products(ProductID),   FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID))

AffectedRows : 0

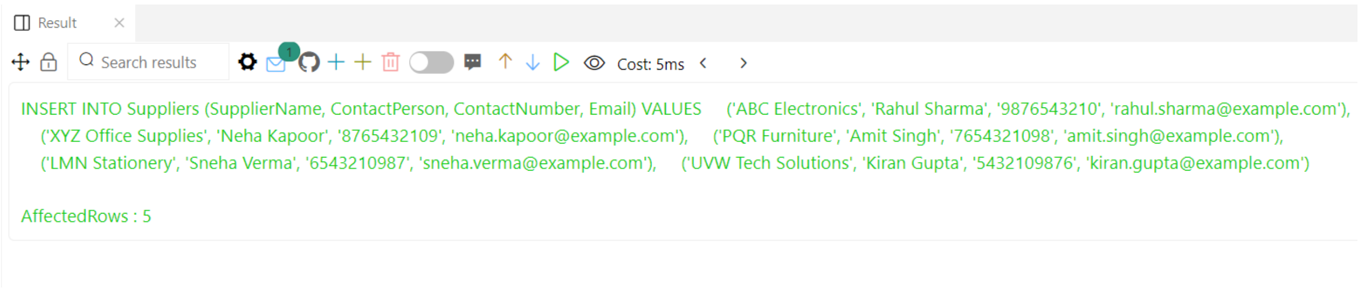
```
-- Insert sample data into the Products table
INSERT INTO Products (ProductName, Category, UnitPrice,
StockQuantity) VALUES
    ('Laptop', 'Electronics', 899.99, 50),
    ('Printer', 'Office Supplies', 149.99, 30),
    ('Chair', 'Furniture', 49.99, 100);
```



```
INSERT INTO Products (ProductName, Category, UnitPrice, StockQuantity) VALUES     ('Laptop', 'Electronics', 899.99, 50),     ('Printer', 'Office Supplies', 149.99, 30),
    ('Chair', 'Furniture', 49.99, 100)

AffectedRows : 3
```

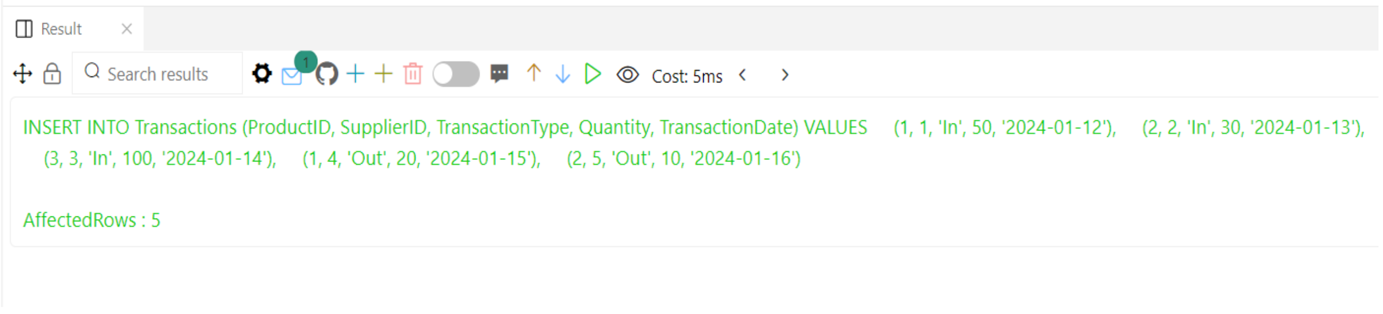
```sql
-- Insert sample data into the Suppliers table
INSERT INTO Suppliers (SupplierName, ContactPerson, ContactNumber,
Email) VALUES
    ('ABC Electronics', 'Rahul Sharma', '9876543210',
'rahul.sharma@example.com'),
    ('XYZ Office Supplies', 'Neha Kapoor', '8765432109',
'neha.kapoor@example.com'),
    ('PQR Furniture', 'Amit Singh', '7654321098',
'amit.singh@example.com'),
    ('LMN Stationery', 'Sneha Verma', '6543210987',
'sneha.verma@example.com'),
    ('UVW Tech Solutions', 'Kiran Gupta', '5432109876',
'kiran.gupta@example.com');
```



```
INSERT INTO Suppliers (SupplierName, ContactPerson, ContactNumber, Email) VALUES     ('ABC Electronics', 'Rahul Sharma', '9876543210', 'rahul.sharma@example.com'),
    ('XYZ Office Supplies', 'Neha Kapoor', '8765432109', 'neha.kapoor@example.com'),     ('PQR Furniture', 'Amit Singh', '7654321098', 'amit.singh@example.com'),
    ('LMN Stationery', 'Sneha Verma', '6543210987', 'sneha.verma@example.com'),     ('UVW Tech Solutions', 'Kiran Gupta', '5432109876', 'kiran.gupta@example.com')

AffectedRows : 5
```

```sql
-- Insert sample data into the Transactions table
INSERT INTO Transactions (ProductID, SupplierID, TransactionType,
Quantity, TransactionDate) VALUES
    (1, 1, 'In', 50, '2024-01-12'),
    (2, 2, 'In', 30, '2024-01-13'),
    (3, 3, 'In', 100, '2024-01-14'),
    (1, 4, 'Out', 20, '2024-01-15'),
    (2, 5, 'Out', 10, '2024-01-16');
```



```
INSERT INTO Transactions (ProductID, SupplierID, TransactionType, Quantity, TransactionDate) VALUES     (1, 1, 'In', 50, '2024-01-12'),     (2, 2, 'In', 30, '2024-01-13'),
    (3, 3, 'In', 100, '2024-01-14'),     (1, 4, 'Out', 20, '2024-01-15'),     (2, 5, 'Out', 10, '2024-01-16')

AffectedRows : 5
```

File://stockManagement.ipynb

```
import mysql.connector
from datetime import date
# Establish a connection to the MySQL database
conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="StockManagement"
    )
cursor = conn.cursor()

def display_products():
    cursor.execute("SELECT * FROM Products")
    products = cursor.fetchall()
    print("\nProducts:")
    for product in products:
        print(product)

def display_suppliers():
    cursor.execute("SELECT * FROM Suppliers")
    suppliers = cursor.fetchall()
    print("\nSuppliers:")
    for supplier in suppliers:
        print(supplier)

def display_transactions():
    cursor.execute("SELECT * FROM Transactions")
    transactions = cursor.fetchall()
    print("\nTransactions:")
    for transaction in transactions:
        print(transaction)

def add_transaction(product_id, supplier_id, transaction_type, quantity):
    today = date.today()
    transaction_date = today.strftime("%Y-%m-%d")
    cursor.execute("INSERT INTO Transactions (ProductID, SupplierID,
TransactionType, Quantity, TransactionDate) VALUES (%s, %s, %s, %s, %s)",
                   (product_id, supplier_id, transaction_type, quantity,
transaction_date))
    conn.commit()
    print("\nTransaction added successfully.")

# Main program loop
while True:
    print("\nStock Management System")
    print("1. Display Products")
    print("2. Display Suppliers")
    print("3. Display Transactions")
    print("4. Add Transaction")
    print("0. Exit")

    choice = input("Enter your choice (0-4): ")

    if choice == '1':
        display_products()
    elif choice == '2':
        display_suppliers()
    elif choice == '3':
        display_transactions()
    elif choice == '4':
        product_id = int(input("Enter Product ID: "))
        supplier_id = int(input("Enter Supplier ID: "))
        transaction_type = input("Enter Transaction Type (In/Out):
").capitalize()
        quantity = int(input("Enter Quantity: "))
        add_transaction(product_id, supplier_id, transaction_type, quantity)
    elif choice == '0':
        break
    else:
        print("Invalid choice. Please enter a valid option.")
```

```python
# Close the connection
cursor.close()
conn.close()
```

OUTPUT:

```
Stock Management System
1. Display Products
2. Display Suppliers
3. Display Transactions
4. Add Transaction
0. Exit
```

Input: 1

```
Products:
(1, 'Laptop', 'Electronics', Decimal('899.99'), 50)
(2, 'Printer', 'Office Supplies', Decimal('149.99'), 30)
(3, 'Chair', 'Furniture', Decimal('49.99'), 100)
```

Input: 2

```
Suppliers:
(1, 'ABC Electronics', 'Rahul Sharma', '9876543210',
'rahul.sharma@example.com')
(2, 'XYZ Office Supplies', 'Neha Kapoor', '8765432109',
'neha.kapoor@example.com')
(3, 'PQR Furniture', 'Amit Singh', '7654321098',
'amit.singh@example.com')
(4, 'LMN Stationery', 'Sneha Verma', '6543210987',
'sneha.verma@example.com')
(5, 'UVW Tech Solutions', 'Kiran Gupta', '5432109876',
'kiran.gupta@example.com')
```

Input: 3

```
Transactions:
(1, 1, 1, 'In', 50, datetime.date(2024, 1, 12))
(2, 2, 2, 'In', 30, datetime.date(2024, 1, 13))
(3, 3, 3, 'In', 100, datetime.date(2024, 1, 14))
(4, 1, 4, 'Out', 20, datetime.date(2024, 1, 15))
(5, 2, 5, 'Out', 10, datetime.date(2024, 1, 16))
```

Input: 4

```
 Enter Product ID: 2
```

```
Enter Supplier ID: 1
Enter Transaction Type (In/Out): In
Enter Quantity: 3
Transaction added successfully.
```