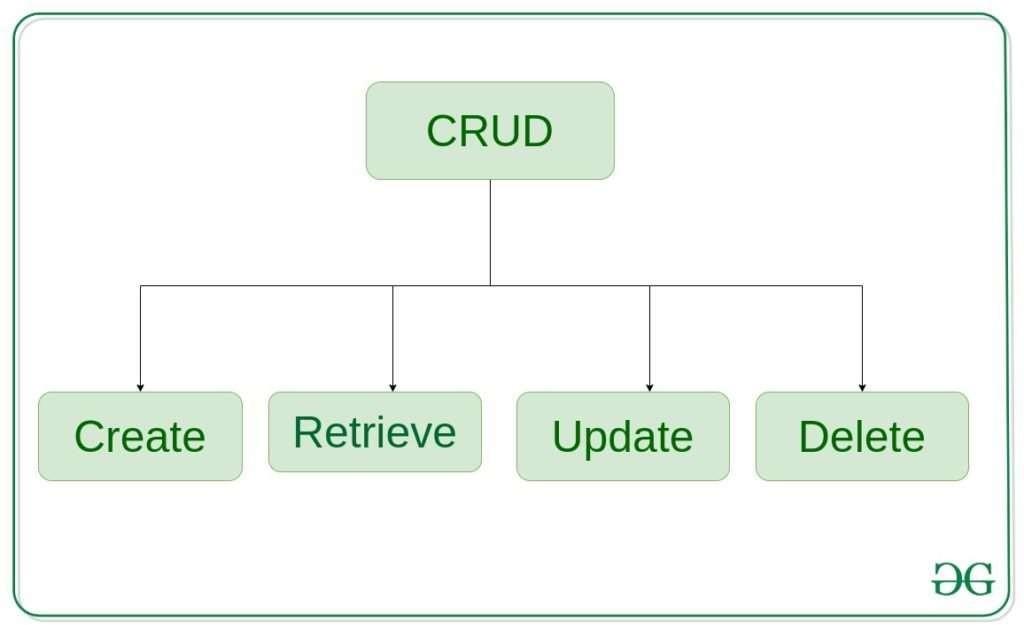
| Experiment No. 13 |
| --- |
| Program to demonstrate CRUD operations on database (SQLite/MySQL using Python) |
| Date of Performace: 10/04/2024 |
| Date of Submission: 17/04/2024 |











Code:

import mysql.connector

# Connect to the MySQL database

conn = mysql.connector.connect(

host="localhost",

user="advait",

password="advait",

database="C\_46"

)

cursor = conn.cursor()

def create\_table():

# Create a table if it doesn't exist

cursor.execute('''CREATE TABLE IF NOT EXISTS mytable

(id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), age INT)''')

conn.commit()

print("Table created successfully")

def insert\_data(name, age):

# Insert data into the table

cursor.execute('''INSERT INTO mytable (name, age) VALUES (%s, %s)''', (name, age))

conn.commit()

print("Data inserted successfully")

def retrieve\_data():

# Retrieve all data from the table

cursor.execute('''SELECT \* FROM mytable''')

rows = cursor.fetchall()

print("ID\tName\tAge")

for row in rows:

print(f"{row[0]}\t{row[1]}\t{row[2]}")

def update\_data(id, new\_name, new\_age):

# Update data in the table

cursor.execute('''UPDATE mytable SET name = %s, age = %s WHERE id = %s''', (new\_name, new\_age, id))

conn.commit()

print("Data updated successfully")

def delete\_data(id):

# Delete data from the table

cursor.execute('''DELETE FROM mytable WHERE id = %s''', (id,))

conn.commit()

print("Data deleted successfully")

# Example usage

create\_table()

insert\_data("John", 30)

insert\_data("Alice", 25)

retrieve\_data()

update\_data(1, "Johnny", 31)

retrieve\_data()

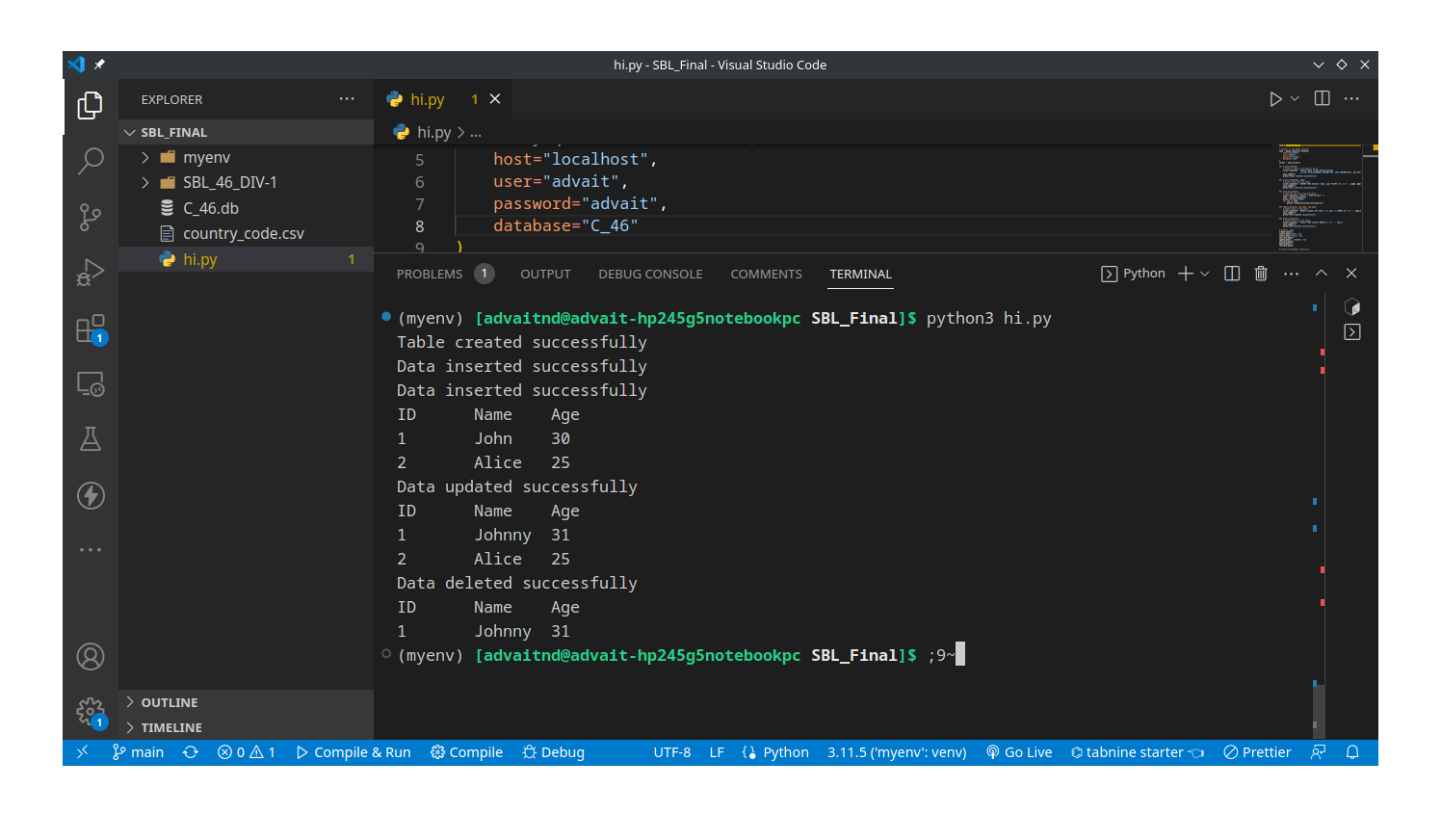
delete\_data(2)

retrieve\_data()

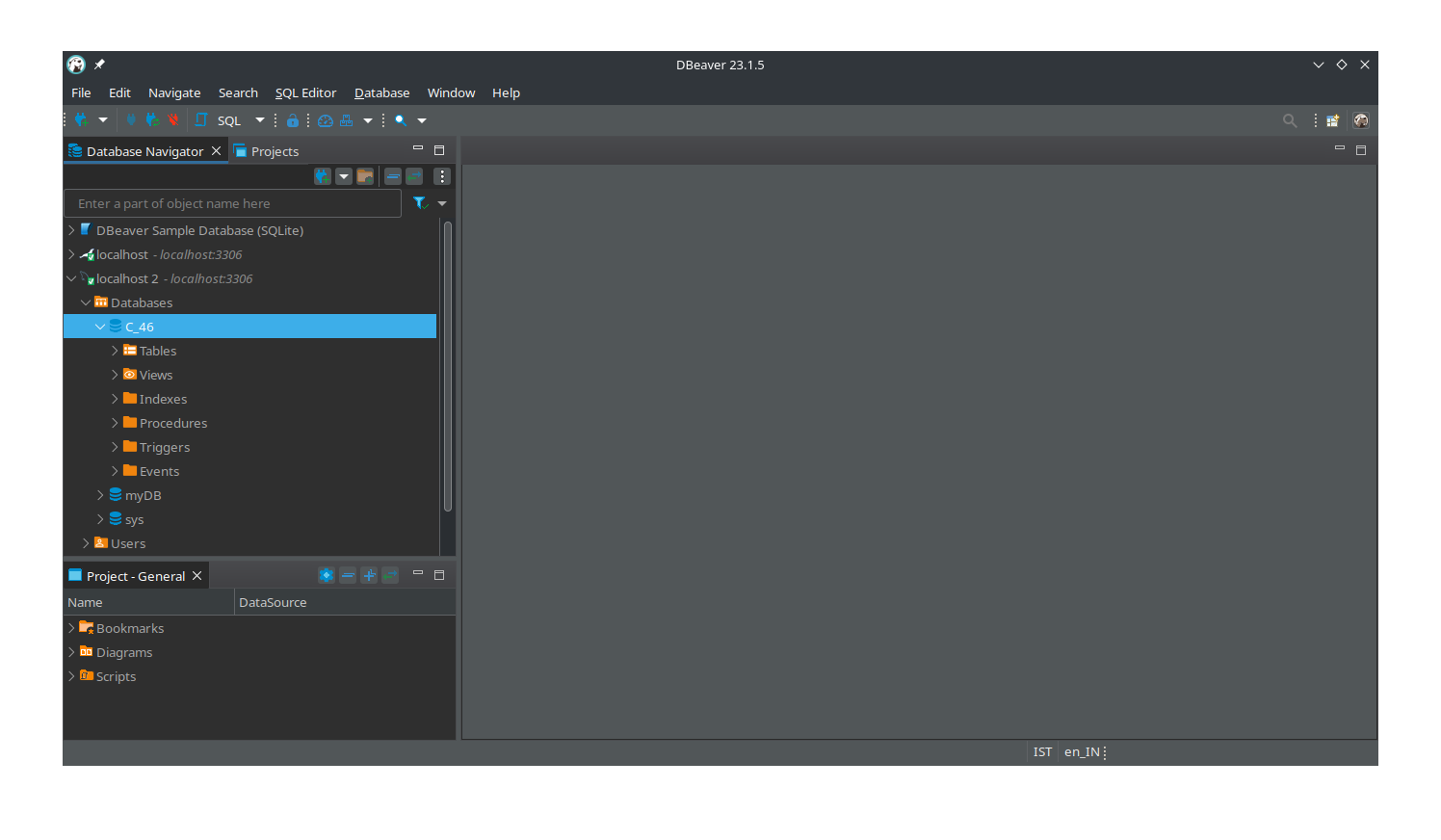
# Close the database connection

conn.close()

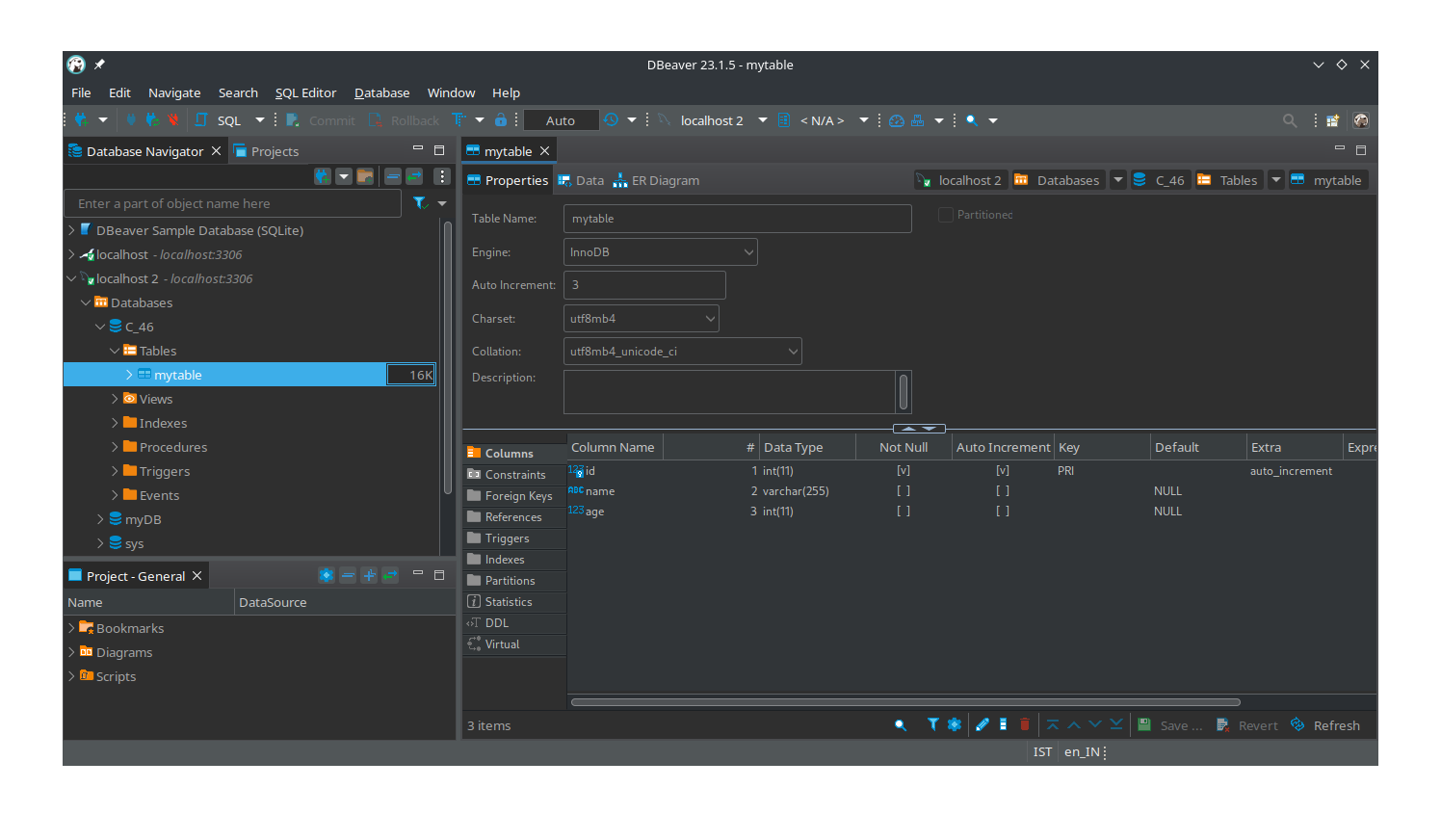
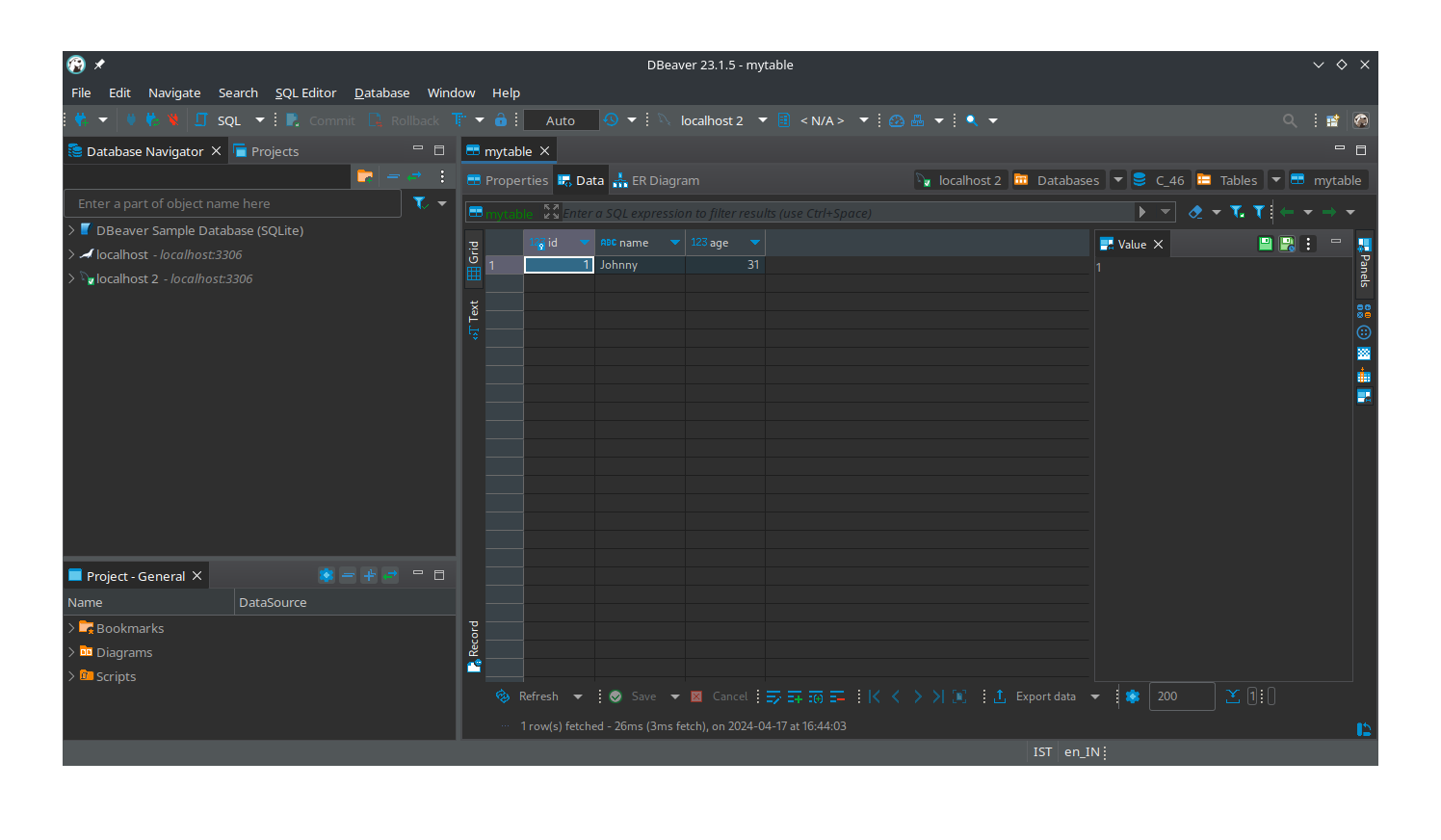
**Output:**



**Before Updation:**

****

**After Updation:**

****

**Conclusion:**

In this Python script, we showcase the integration of `mysql-connector-python` to connect to a MySQL database, implementing CRUD operations: Create, Read, Update, and Delete. Through establishing a connection to the MySQL database by specifying the host, username, password, and database name, the script defines functions for table creation, data insertion, retrieval, update, and deletion. Utilizing SQL queries executed with `cursor.execute()` and committing transactions with `conn.commit()`, the script ensures changes are persisted to the database. This example provides a versatile foundation for Python developers to manage MySQL databases, demonstrating seamless data interaction and manipulation, ideal for diverse applications from simple data management systems to more complex projects requiring robust database functionality.