Код самої програми:

```
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
import pymysql.cursors
```

- **tkinter**: Tkinter — це фактично стандартний GUI (графічний інтерфейс користувача) пакет Python. Він використовується для створення додатків графічного інтерфейсу користувача.

- **pymysql.cursors**: Цей модуль надає класи курсорів для доступу та керування даними з баз даних у mysql .

```
def execute_sql():
    command = sql_entry.get()
    try:
        with connection.cursor() as cursor:
            cursor.execute(command)
            if cursor.description:
                columns = [column[0] for column in cursor.description]
                rows = cursor.fetchall()
                update_treeview(columns, rows)
                messagebox.showinfo("Success", "Command executed successfully.")
            else:
                messagebox.showinfo("Success", "Command executed successfully. No results to
display.")
    except pymysql.err.ProgrammingError as err:
        messagebox.showerror("Error", str(err))
    except Exception as ex:
        messagebox.showerror("Error", str(ex))
```

- Ця функція викликається при натисканні кнопки «Виконати».

- Він отримує команду SQL із **sql_entry**віджета.

- Потім він намагається виконати команду SQL за допомогою підключення до бази даних.

- У разі успіху він отримує стовпці та рядки, повернуті запитом, і оновлює **result_tree**віджет даними.

- Якщо є **ProgrammingError**, це відображає помилку за допомогою вікна повідомлень.

- Якщо виникає будь-який інший виняток, він також показує помилку за допомогою вікна повідомлень.

```
def update_treeview(columns, rows):
    result_tree.delete(*result_tree.get_children())
    result_tree["columns"] = columns
    for col in columns:
        result_tree.heading(col, text=col)
        result_tree.column(col, anchor=tk.CENTER)
    for i, row in enumerate(rows, start=1):
        values = [row[column] for column in columns]
        result_tree.insert("", tk.END, text=str(i), values=values)
```

- Ця функція оновлює **result_tree** віджет заданими стовпцями та рядками.

- Спочатку очищає наявні дані в **result_tree**.

- Потім він налаштовує стовпці **result_tree**на основі наданих імен стовпців.
- Після цього він перебирає рядки, витягує значення для кожного рядка та вставляє їх у **result_tree**.

```
def close_connection():
    try:
        connection.close()
        root.destroy()
    except Exception as ex:
        messagebox.showerror("Error", str(ex))
```

- Ця функція викликається при натисканні кнопки «Закрити підключення».
- Він намагається закрити з’єднання з базою даних, а потім знищує кореневе вікно Tkinter.
- Якщо під час процесу виникає будь-який виняток, він показує помилку за допомогою вікна повідомлень.

```
try:
    connection = pymysql.connect(
        host="localhost",
        port=43000,
        user="root",
        password="1234",
        database="to_do_list3",
        cursorclass=pymysql.cursors.DictCursor
    )
except Exception as ex:
    print("Connection refused...")
    print(ex)
```

- Цей блок намагається встановити з’єднання з базою даних MySQL, що працює на локальному хості, із зазначеними обліковими даними.
- Якщо підключення успішне, створюється об’єкт підключення (**connection**).
- Якщо підключення не вдається, друкується повідомлення про помилку.

```
root = tk.Tk()
root.title("SQL Query Executor")

sql_label = tk.Label(root, text="Enter SQL Query:")
sql_label.pack()

sql_entry = tk.Entry(root, width=50)
sql_entry.pack()

execute_button = tk.Button(root, text="Execute", command=execute_sql)
execute_button.pack()

result_tree = ttk.Treeview(root)
result_tree["show"] = "headings"
result_tree.pack()

close_button = tk.Button(root, text="Close Connection", command=close_connection)
close_button.pack()

root.mainloop()
```

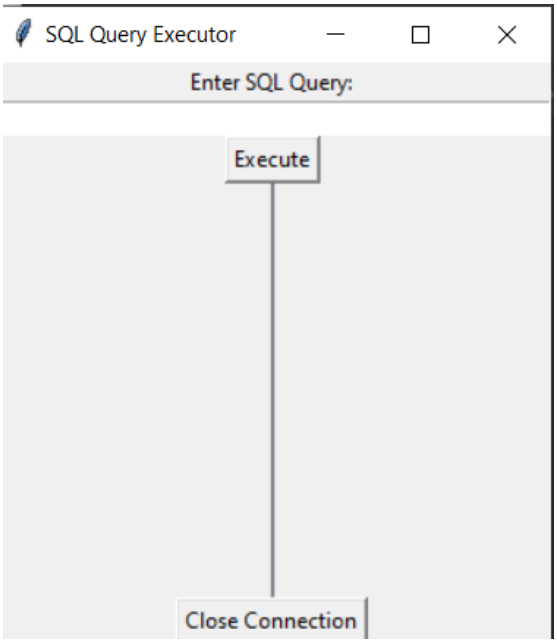
- Цей блок налаштовує GUI (графічний інтерфейс користувача) за допомогою віджетів Tkinter.
- Він створює кореневе вікно Tkinter (**root**) із назвою «Виконавець запиту SQL».
- Він додає мітку, віджет запису, кнопку для виконання команд SQL, віджет Treeview для відображення результатів запиту та кнопку для закриття з'єднання з базою даних.
- Нарешті, він запускає цикл подій Tkinter, викликаючи **root.mainloop()**, який очікує взаємодії користувача.

Перевірка роботи:

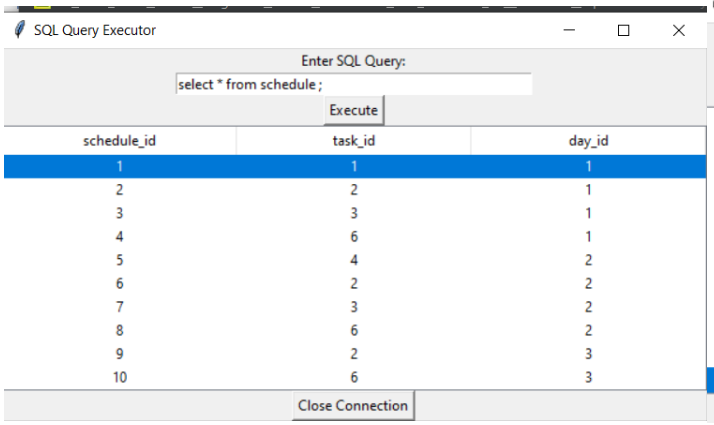
Якщо якісь проблеми з підключенням про це буде повідомленно в консолі – ось приклад з штучно створенною проблемою, невірним паролем

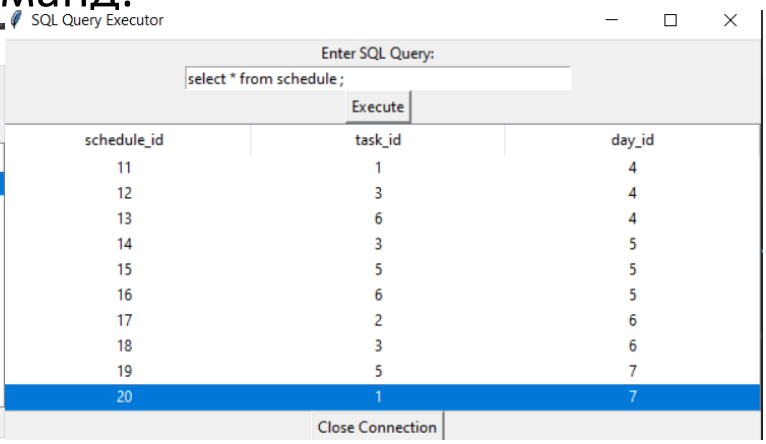
```
Connection refused...
(1045, "Access denied for user 'root'@'192.168.65.1' (using password: YES)")
```

Якщо все добре то вилазить віконце для роботи з базою даних



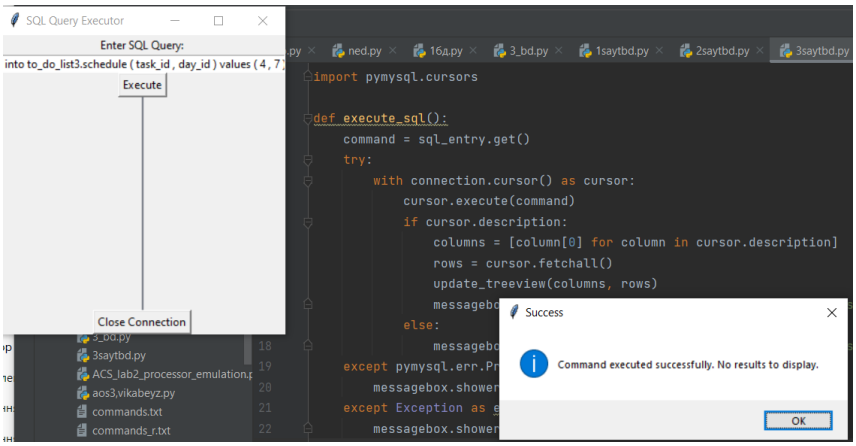
Введемо декілька цікавих команд:





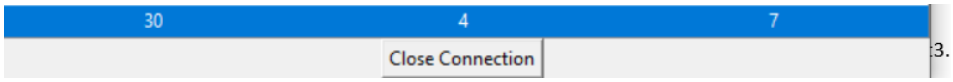
Якщо результтів багато їх можна прогортать.
Можна додати(видалити, оновити і т.д) щось :

insert into to_do_list3.schedule (task_id , day_id) values (4, 7) ; Вилазить віконце в якому



вказано, що команда виконана успішно. Зараз можна зробити select таблиці «розклад»

і перевірити це



Отже впевнилися що все працює, зробимо якийсь складний гарний запит: