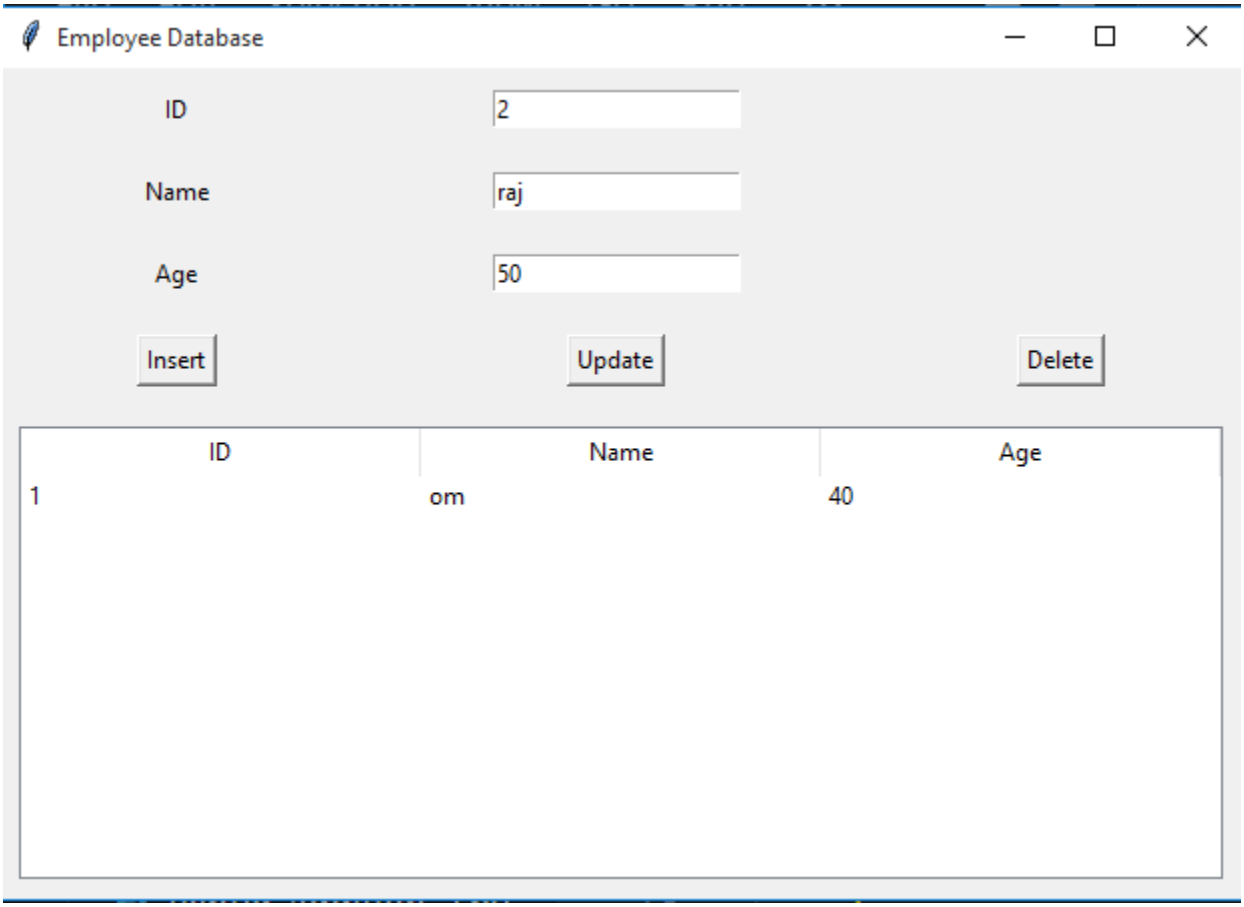
Here is an example of how to create a simple CRUD (Create, Read, Update, Delete) application using `tkinter` and MySQL. This example will show how you can insert, update, delete, and display records from a MySQL database.



Here is an example of how to create a simple CRUD (Create, Read, Update, Delete) application using `tkinter` and MySQL. This example will show how you can insert, update, delete, and display records from a MySQL database.

Before running this example, ensure you have the following:

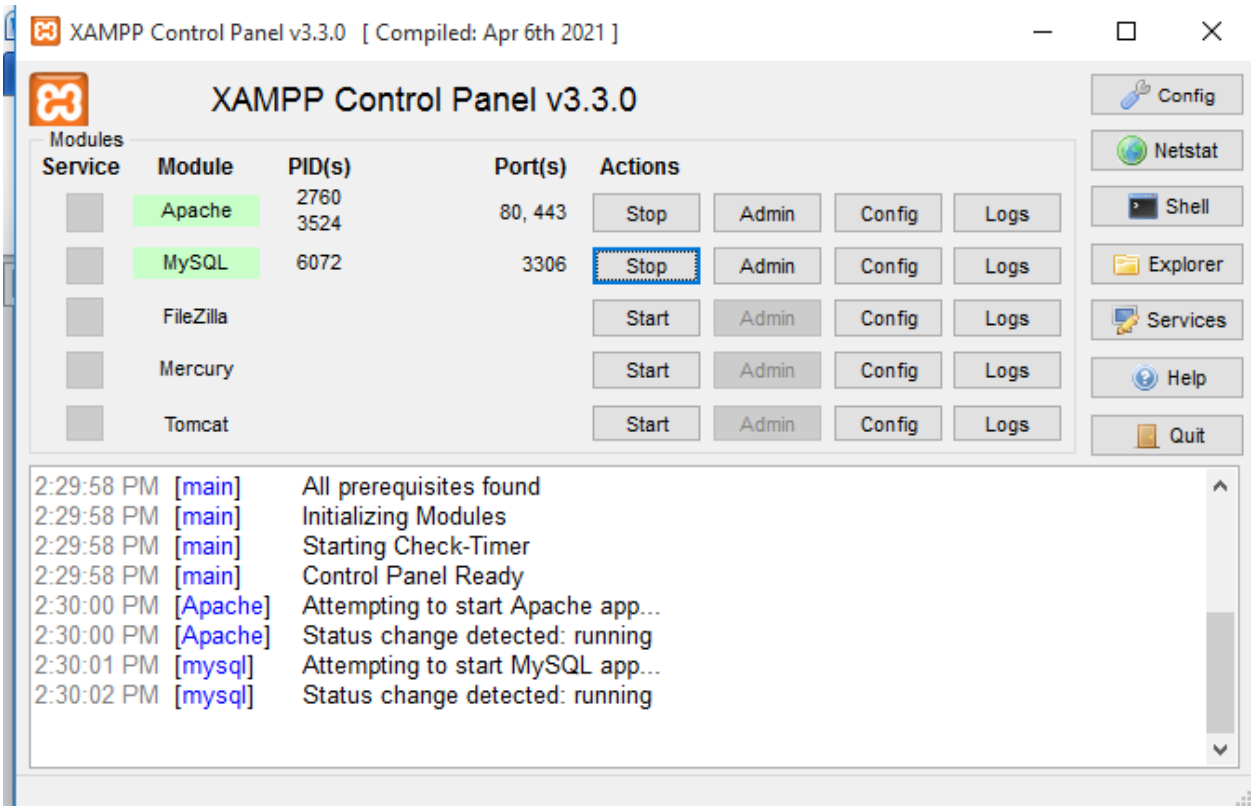
1. Python installed on your system.
2. Install `mysql-connector` for connecting to MySQL:

```
bash
Copy code
pip install mysql-connector
```

3. Create a MySQL database and table for testing.

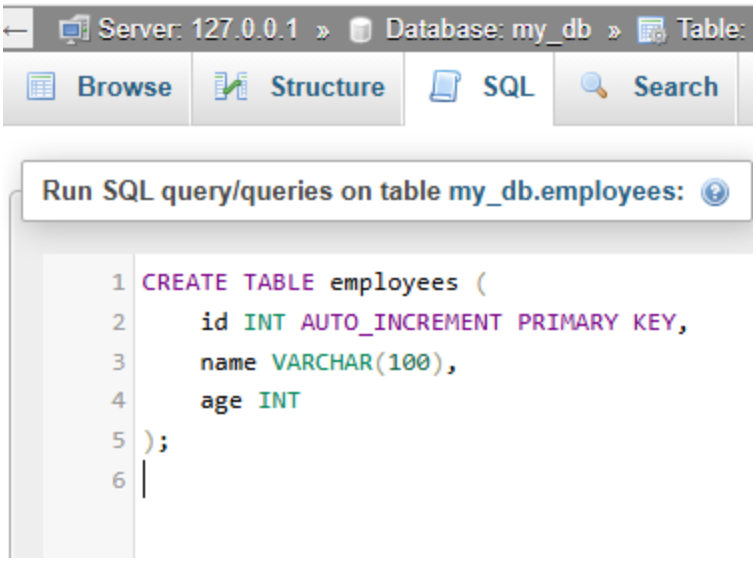
Example MySQL Table:

Install xampp and run it as shown below :-



Sql Code

```
CREATE DATABASE my_db;

USE my_db;

CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    age INT
);

Open

localhost/phpmyadmin
```



Now, here's the Python code using `tkinter` and `mysql-connector`:-

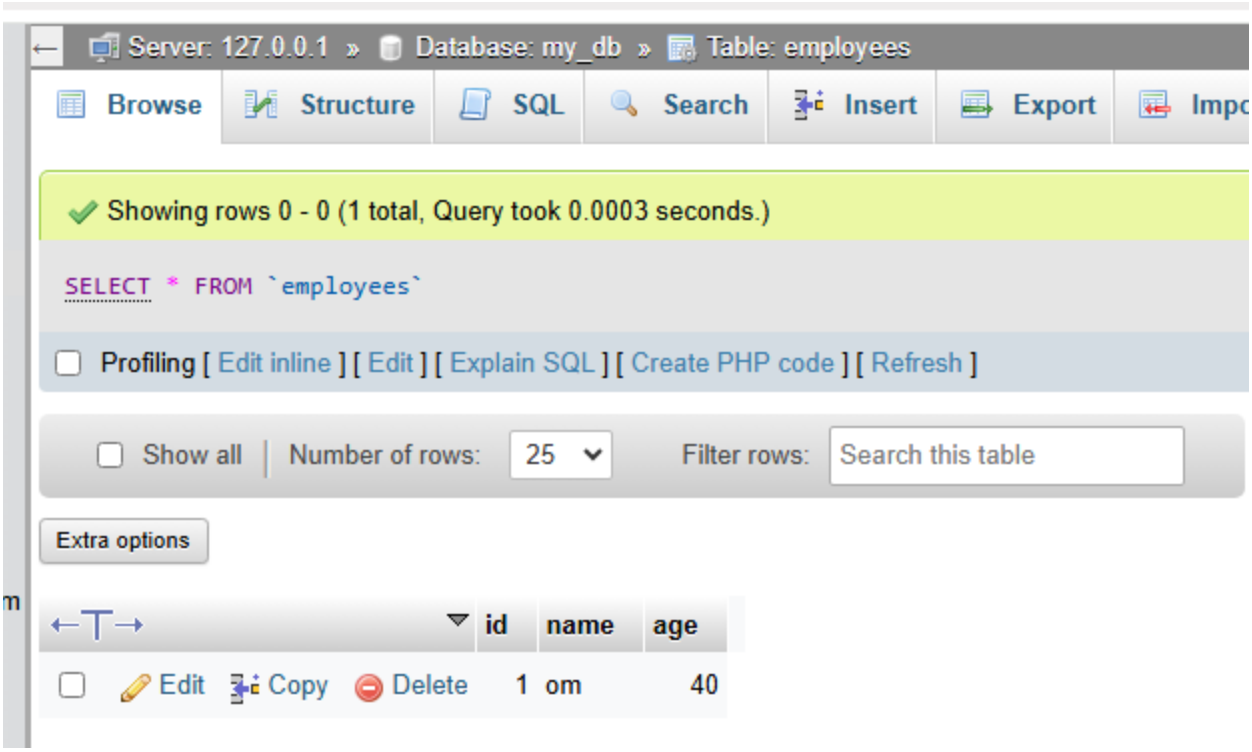
```
import tkinter as tk
from tkinter import messagebox
import mysql.connector
from tkinter import ttk

# MySQL database connection
def connect_db():
    return mysql.connector.connect(
        host="localhost",        # Your MySQL host
        user="root",             # Your MySQL username
        password="",     # Your MySQL password
        database="my_db"       # Your MySQL database
    )

# Insert Data into MySQL
def insert_data():
    name = name_entry.get()
    age = age_entry.get()

    if name == "" or age == "":
        messagebox.showerror("Input Error", "All fields must be filled.")
        return

    conn = connect_db()
```

```
    cursor = conn.cursor()
    cursor.execute("INSERT INTO employees (name, age) VALUES (%s, %s)", (name,
age))
    conn.commit()
    conn.close()

    messagebox.showinfo("Success", "Record inserted successfully.")
    display_data()

# Update Data in MySQL
def update_data():
    try:
        emp_id = int(id_entry.get())
        name = name_entry.get()
        age = age_entry.get()

        if name == "" or age == "":
            messagebox.showerror("Input Error", "All fields must be filled.")
            return

        conn = connect_db()
        cursor = conn.cursor()
        cursor.execute("UPDATE employees SET name = %s, age = %s WHERE id = %s",
(name, age, emp_id))
        conn.commit()
        conn.close()

        messagebox.showinfo("Success", "Record updated successfully.")
        display_data()

    except ValueError:
        messagebox.showerror("Input Error", "Please enter a valid ID.")

# Delete Data from MySQL
def delete_data():
    try:
        emp_id = int(id_entry.get())

        conn = connect_db()
        cursor = conn.cursor()
        cursor.execute("DELETE FROM employees WHERE id = %s", (emp_id,))
        conn.commit()
        conn.close()

        messagebox.showinfo("Success", "Record deleted successfully.")
```

```
        display_data()

    except ValueError:
        messagebox.showerror("Input Error", "Please enter a valid ID.")

# Display Data from MySQL
def display_data():
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM employees")
    records = cursor.fetchall()
    conn.close()

    # Clear previous records
    for row in records_listbox.get_children():
        records_listbox.delete(row)

    # Insert new records
    for record in records:
        records_listbox.insert("", "end", values=(record[0], record[1],
record[2]))

# Setup Tkinter GUI
root = tk.Tk()
root.title("Employee Database")

# Labels
id_label = tk.Label(root, text="ID")
id_label.grid(row=0, column=0, padx=10, pady=10)

name_label = tk.Label(root, text="Name")
name_label.grid(row=1, column=0, padx=10, pady=10)

age_label = tk.Label(root, text="Age")
age_label.grid(row=2, column=0, padx=10, pady=10)

# Entries
id_entry = tk.Entry(root)
id_entry.grid(row=0, column=1, padx=10, pady=10)

name_entry = tk.Entry(root)
name_entry.grid(row=1, column=1, padx=10, pady=10)

age_entry = tk.Entry(root)
age_entry.grid(row=2, column=1, padx=10, pady=10)
```

```
# Buttons
insert_button = tk.Button(root, text="Insert", command=insert_data)
insert_button.grid(row=3, column=0, padx=10, pady=10)

update_button = tk.Button(root, text="Update", command=update_data)
update_button.grid(row=3, column=1, padx=10, pady=10)

delete_button = tk.Button(root, text="Delete", command=delete_data)
delete_button.grid(row=3, column=2, padx=10, pady=10)

# Treeview for displaying records
columns = ("ID", "Name", "Age")
records_listbox = tk.ttk.Treeview(root, columns=columns, show="headings")
records_listbox.grid(row=4, column=0, columnspan=3, padx=10, pady=10)

for col in columns:
    records_listbox.heading(col, text=col)

# Initial data load
display_data()

# Run the app
root.mainloop()
```

check table in localhost/phpmyadmin



Breakdown of the Code:

1. **Connect to MySQL Database:**
   - The `connect_db()` function is used to connect to a MySQL database.
2. **Insert Data:**
   - The `insert_data()` function grabs the input data from the text fields, inserts it into the MySQL database, and refreshes the display.
3. **Update Data:**
   - The `update_data()` function updates an existing record based on the provided `ID`.
4. **Delete Data:**
   - The `delete_data()` function deletes a record based on the provided `ID`.
5. **Display Data:**
   - The `display_data()` function retrieves and displays all records from the database in a `Treeview` widget.
6. **Tkinter Interface:**
   - The GUI uses `tkinter` to create input fields, labels, buttons, and a `Treeview` widget to display records.

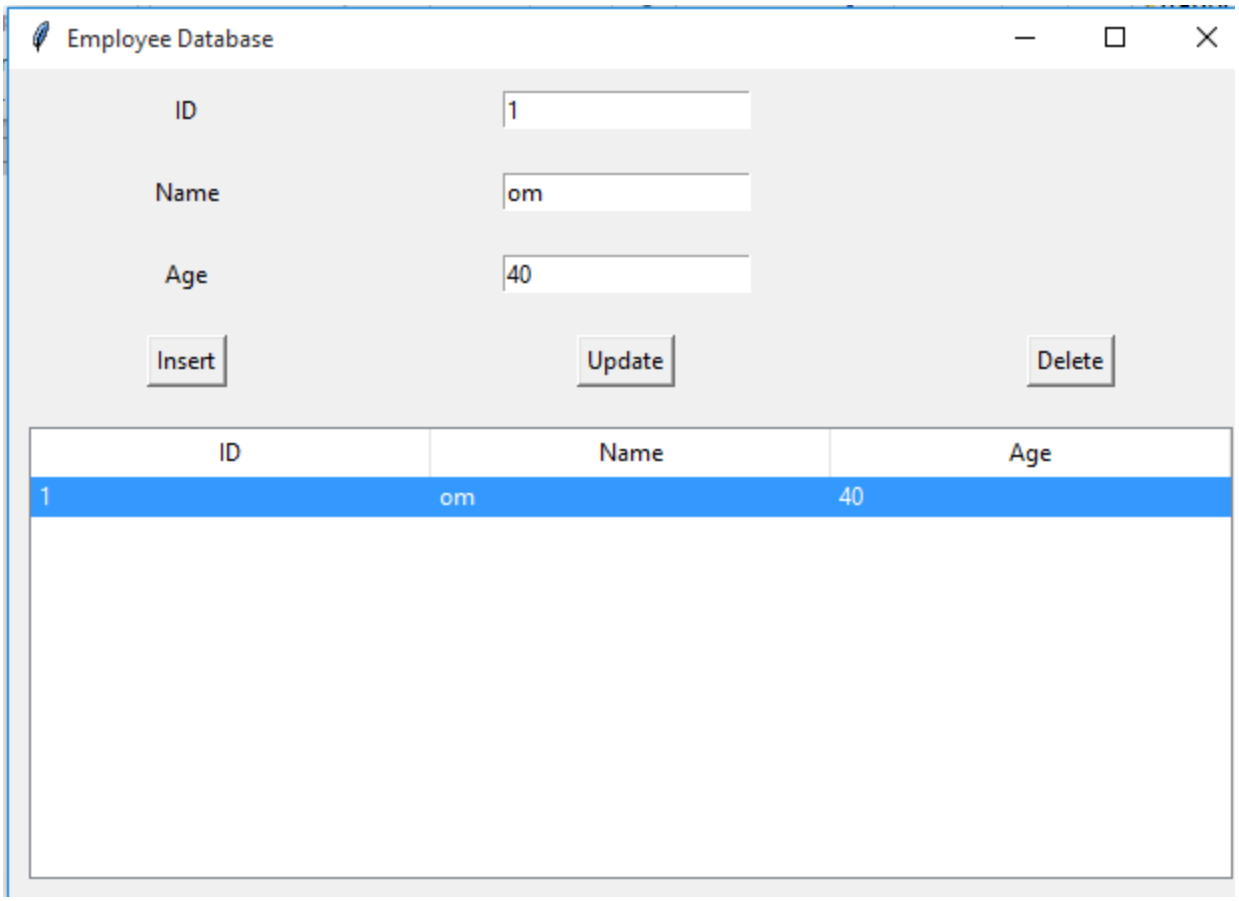
Fully Functional Crud Example:-

```
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk  # Import ttk
import mysql.connector

# MySQL database connection
def connect_db():
    return mysql.connector.connect(
        host="localhost",        # Your MySQL host
        user="root",             # Your MySQL username
        password="",     # Your MySQL password
        database="my_db"       # Your MySQL database
    )

# Insert Data into MySQL
def insert_data():
    name = name_entry.get()
    age = age_entry.get()

    if name == "" or age == "":
        messagebox.showerror("Input Error", "All fields must be filled.")
        return

    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("INSERT INTO employees (name, age) VALUES (%s, %s)", (name,
age))
    conn.commit()
    conn.close()

    messagebox.showinfo("Success", "Record inserted successfully.")
    display_data()

# Update Data in MySQL
def update_data():
    try:
        emp_id = int(id_entry.get())
        name = name_entry.get()
        age = age_entry.get()

        if name == "" or age == "":
            messagebox.showerror("Input Error", "All fields must be filled.")
            return
```

```
        conn = connect_db()
        cursor = conn.cursor()
        cursor.execute("UPDATE employees SET name = %s, age = %s WHERE id = %s",
(name, age, emp_id))
        conn.commit()
        conn.close()

        messagebox.showinfo("Success", "Record updated successfully.")
        display_data()

    except ValueError:
        messagebox.showerror("Input Error", "Please enter a valid ID.")

# Delete Data from MySQL
def delete_data():
    try:
        emp_id = int(id_entry.get())

        conn = connect_db()
        cursor = conn.cursor()
        cursor.execute("DELETE FROM employees WHERE id = %s", (emp_id,))
        conn.commit()
        conn.close()

        messagebox.showinfo("Success", "Record deleted successfully.")
        display_data()

    except ValueError:
        messagebox.showerror("Input Error", "Please enter a valid ID.")

# Display Data from MySQL
def display_data():
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM employees")
    records = cursor.fetchall()
    conn.close()

    # Clear previous records
    for row in records_listbox.get_children():
        records_listbox.delete(row)

    # Insert new records
    for record in records:
```

```
        records_listbox.insert("", "end", values=(record[0], record[1],
record[2]))

# Function to handle row selection
def on_select(event):
    # Get the selected row
    selected_item = records_listbox.selection()
    if selected_item:
        # Get the values of the selected row
        values = records_listbox.item(selected_item)["values"]

        # Set the values in the input fields
        id_entry.delete(0, tk.END)
        id_entry.insert(0, values[0])
        name_entry.delete(0, tk.END)
        name_entry.insert(0, values[1])
        age_entry.delete(0, tk.END)
        age_entry.insert(0, values[2])

# Setup Tkinter GUI
root = tk.Tk()
root.title("Employee Database")

# Labels
id_label = tk.Label(root, text="ID")
id_label.grid(row=0, column=0, padx=10, pady=10)

name_label = tk.Label(root, text="Name")
name_label.grid(row=1, column=0, padx=10, pady=10)

age_label = tk.Label(root, text="Age")
age_label.grid(row=2, column=0, padx=10, pady=10)

# Entries
id_entry = tk.Entry(root)
id_entry.grid(row=0, column=1, padx=10, pady=10)

name_entry = tk.Entry(root)
name_entry.grid(row=1, column=1, padx=10, pady=10)

age_entry = tk.Entry(root)
age_entry.grid(row=2, column=1, padx=10, pady=10)

# Buttons
insert_button = tk.Button(root, text="Insert", command=insert_data)
```

```
insert_button.grid(row=3, column=0, padx=10, pady=10)

update_button = tk.Button(root, text="Update", command=update_data)
update_button.grid(row=3, column=1, padx=10, pady=10)

delete_button = tk.Button(root, text="Delete", command=delete_data)
delete_button.grid(row=3, column=2, padx=10, pady=10)

# Treeview for displaying records
columns = ("ID", "Name", "Age")
records_listbox = ttk.Treeview(root, columns=columns, show="headings")
records_listbox.grid(row=4, column=0, columnspan=3, padx=10, pady=10)

for col in columns:
    records_listbox.heading(col, text=col)

# Bind the select event to the on_select function
records_listbox.bind("<<TreeviewSelect>>", on_select)

# Initial data load
display_data()

# Run the app
root.mainloop()
```


Employee Database
ID
1
Name
om
Age
40
Insert
Update
Delete
ID
Name
Age
1
om
40