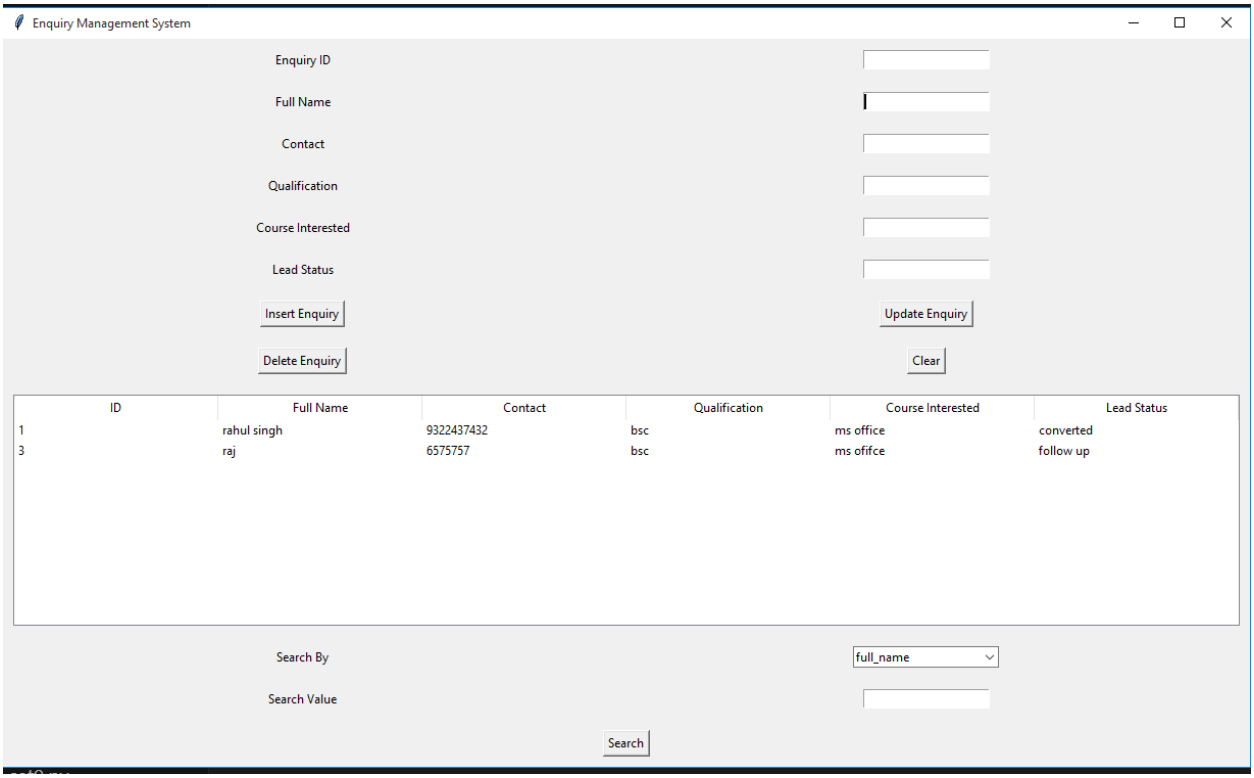
Tkinter Simple task on Enquiry Mangement System:-




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

# MySQL connection
def connect_db():
    return mysql.connector.connect(
        host="localhost",  # Change to your MySQL server
        user="root",       # Your MySQL username
        password="",       # Your MySQL password
        database="enquiries_db"  # Name of your database
    )

# Insert Enquiry
def insert_enquiry():
    full_name = entry_full_name.get()
```

```
    contact = entry_contact.get()
    qualification = entry_qualification.get()
    course_interested = entry_course_interested.get()
    lead_status = entry_lead_status.get()

    if not full_name or not contact or not qualification or not course_interested
or not lead_status:
        messagebox.showwarning("Input Error", "All fields are required.")
        return

    try:
        conn = connect_db()
        cursor = conn.cursor()
        query = "INSERT INTO enquiries (full_name, contact, qualification,
course_interested, lead_status) VALUES (%s, %s, %s, %s, %s)"
        values = (full_name, contact, qualification, course_interested,
lead_status)
        cursor.execute(query, values)
        conn.commit()
        messagebox.showinfo("Success", "Enquiry inserted successfully!")
        conn.close()
        clear_fields()
        display_enquiries()  # Refresh the grid after insertion
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        conn.close()

# Update Enquiry
def update_enquiry():
    enquiry_id = entry_id.get()
    full_name = entry_full_name.get()
    contact = entry_contact.get()
    qualification = entry_qualification.get()
    course_interested = entry_course_interested.get()
    lead_status = entry_lead_status.get()

    if not enquiry_id:
        messagebox.showwarning("Input Error", "Please provide the enquiry ID to
update.")
        return

    try:
        conn = connect_db()
        cursor = conn.cursor()
```

```
        query = "UPDATE enquiries SET full_name=%s, contact=%s, qualification=%s,
course_interested=%s, lead_status=%s WHERE id=%s"
        values = (full_name, contact, qualification, course_interested,
lead_status, enquiry_id)
        cursor.execute(query, values)
        conn.commit()
        messagebox.showinfo("Success", "Enquiry updated successfully!")
        conn.close()
        clear_fields()
        display_enquiries()  # Refresh the grid after update
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        conn.close()

# Delete Enquiry
def delete_enquiry():
    enquiry_id = entry_id.get()

    if not enquiry_id:
        messagebox.showwarning("Input Error", "Please provide the enquiry ID to
delete.")
        return

    try:
        conn = connect_db()
        cursor = conn.cursor()
        query = "DELETE FROM enquiries WHERE id=%s"
        cursor.execute(query, (enquiry_id,))
        conn.commit()

        if cursor.rowcount > 0:
            messagebox.showinfo("Success", f"Enquiry with ID {enquiry_id} deleted
successfully!")
            display_enquiries()  # Refresh the grid after deletion
        else:
            messagebox.showwarning("Not Found", f"No enquiry found with ID
{enquiry_id}")

        conn.close()
        clear_fields()
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        conn.close()

# Search and Display Enquiries in Grid with filter
```

```
def display_enquiries(search_by="", search_value=""):
    # Clear the existing rows in the grid
    for row in tree.get_children():
        tree.delete(row)

    try:
        conn = connect_db()
        cursor = conn.cursor()
        # If search criteria is provided, filter by it
        if search_by and search_value:
            query = f"SELECT * FROM enquiries WHERE {search_by} LIKE %s"
            cursor.execute(query, (f"%{search_value}%",))
        else:
            query = "SELECT * FROM enquiries"
            cursor.execute(query)

        rows = cursor.fetchall()

        for row in rows:
            tree.insert("", "end", values=row)

        conn.close()
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        conn.close()

# Select record from Treeview to update or delete
def on_select_item(event):
    selected_item = tree.selection()[0]
    selected_id = tree.item(selected_item)["values"][0]  # Get the ID of selected
item

    # Fill in the fields with selected record data
    entry_id.delete(0, tk.END)
    entry_full_name.delete(0, tk.END)
    entry_contact.delete(0, tk.END)
    entry_qualification.delete(0, tk.END)
    entry_course_interested.delete(0, tk.END)
    entry_lead_status.delete(0, tk.END)

    entry_id.insert(0, selected_id)
    entry_full_name.insert(0, tree.item(selected_item)["values"][1])
    entry_contact.insert(0, tree.item(selected_item)["values"][2])
    entry_qualification.insert(0, tree.item(selected_item)["values"][3])
    entry_course_interested.insert(0, tree.item(selected_item)["values"][4])
```

```
    entry_lead_status.insert(0, tree.item(selected_item)["values"][5])

# Clear Fields
def clear_fields():
    entry_id.delete(0, tk.END)
    entry_full_name.delete(0, tk.END)
    entry_contact.delete(0, tk.END)
    entry_qualification.delete(0, tk.END)
    entry_course_interested.delete(0, tk.END)
    entry_lead_status.delete(0, tk.END)

# Search Filter Function
def search_filter():
    search_by = search_by_var.get()
    search_value = search_value_entry.get()
    if search_value.strip():
        display_enquiries(search_by, search_value)
    else:
        messagebox.showwarning("Input Error", "Please enter a value to search.")

# Main window
root = tk.Tk()
root.title("Enquiry Management System")

# Labels
tk.Label(root, text="Enquiry ID").grid(row=0, column=0, padx=10, pady=10)
tk.Label(root, text="Full Name").grid(row=1, column=0, padx=10, pady=10)
tk.Label(root, text="Contact").grid(row=2, column=0, padx=10, pady=10)
tk.Label(root, text="Qualification").grid(row=3, column=0, padx=10, pady=10)
tk.Label(root, text="Course Interested").grid(row=4, column=0, padx=10, pady=10)
tk.Label(root, text="Lead Status").grid(row=5, column=0, padx=10, pady=10)

# Entry fields
entry_id = tk.Entry(root)
entry_full_name = tk.Entry(root)
entry_contact = tk.Entry(root)
entry_qualification = tk.Entry(root)
entry_course_interested = tk.Entry(root)
entry_lead_status = tk.Entry(root)

entry_id.grid(row=0, column=1, padx=10, pady=10)
entry_full_name.grid(row=1, column=1, padx=10, pady=10)
entry_contact.grid(row=2, column=1, padx=10, pady=10)
entry_qualification.grid(row=3, column=1, padx=10, pady=10)
entry_course_interested.grid(row=4, column=1, padx=10, pady=10)
```

```
entry_lead_status.grid(row=5, column=1, padx=10, pady=10)

# Buttons
btn_insert = tk.Button(root, text="Insert Enquiry", command=insert_enquiry)
btn_update = tk.Button(root, text="Update Enquiry", command=update_enquiry)
btn_delete = tk.Button(root, text="Delete Enquiry", command=delete_enquiry)
btn_clear = tk.Button(root, text="Clear", command=clear_fields)

btn_insert.grid(row=6, column=0, padx=10, pady=10)
btn_update.grid(row=6, column=1, padx=10, pady=10)
btn_delete.grid(row=7, column=0, padx=10, pady=10)
btn_clear.grid(row=7, column=1, padx=10, pady=10)

# Treeview for displaying enquiries
tree = ttk.Treeview(root, columns=("ID", "Full Name", "Contact", "Qualification",
"Course Interested", "Lead Status"), show="headings")
tree.heading("ID", text="ID")
tree.heading("Full Name", text="Full Name")
tree.heading("Contact", text="Contact")
tree.heading("Qualification", text="Qualification")
tree.heading("Course Interested", text="Course Interested")
tree.heading("Lead Status", text="Lead Status")

tree.grid(row=8, column=0, columnspan=2, padx=10, pady=10)

# Bind the select event to the on_select_item function
tree.bind("<ButtonRelease-1>", on_select_item)

# Search filter UI
search_by_var = tk.StringVar(root)
search_by_var.set("full_name")  # Default search by "Full Name"

tk.Label(root, text="Search By").grid(row=9, column=0, padx=10, pady=10)
search_by_menu = ttk.Combobox(root, textvariable=search_by_var,
values=("full_name", "contact", "course_interested", "lead_status"))
search_by_menu.grid(row=9, column=1, padx=10, pady=10)

tk.Label(root, text="Search Value").grid(row=10, column=0, padx=10, pady=10)
search_value_entry = tk.Entry(root)
search_value_entry.grid(row=10, column=1, padx=10, pady=10)

search_button = tk.Button(root, text="Search", command=search_filter)
search_button.grid(row=11, column=0, columnspan=2, padx=10, pady=10)

# Initial data load
```

```
display_enquiries()  # Display existing enquiries on start

# Run the main loop
root.mainloop()
```

Detailed Explanation of the Program

This program is a simple **Enquiry Management System** built using Python's Tkinter library for the graphical user interface (GUI) and MySQL for database management. The program allows users to **Insert**, **Update**, **Delete**, and **Search** for records in a MySQL database that stores enquiries.

Let’s break down the program step by step:

1. MySQL Database Setup

Before running the program, ensure you have a MySQL database called `enquiries_db` with a table `enquiries`. Here’s a simple SQL query to create such a table:

```
CREATE DATABASE enquiries_db;
USE enquiries_db;

CREATE TABLE enquiries (
    id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(255),
    contact VARCHAR(255),
    qualification VARCHAR(255),
    course_interested VARCHAR(255),
    lead_status VARCHAR(255)
);
```

The table contains the following columns:

- `id`: Auto-incremented primary key for uniquely identifying records.
- `full_name`: Name of the person making the enquiry.
- `contact`: Contact number or email address.
- `qualification`: Qualification of the person.
- `course_interested`: The course the person is interested in.
- `lead_status`: The status of the lead (e.g., "Interested", "Not Interested").

2. Database Connection Function

The function `connect_db()` is used to establish a connection with the MySQL database. It connects to the MySQL server running on `localhost`, using the username `root`, without a password, and selecting the database `enquiries_db`.

```
def connect_db():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="enquiries_db"
    )
```

3. Insert Enquiry Function

The `insert_enquiry()` function handles the insertion of new records into the database. It collects values from the input fields and performs an **INSERT INTO** SQL query to add the data into the `enquiries` table.

```
def insert_enquiry():
    full_name = entry_full_name.get()
    contact = entry_contact.get()
    qualification = entry_qualification.get()
    course_interested = entry_course_interested.get()
    lead_status = entry_lead_status.get()

    # Check if all fields are filled
    if not full_name or not contact or not qualification or not
course_interested or not lead_status:
        messagebox.showwarning("Input Error", "All fields are required.")
        return

    try:
        conn = connect_db()
        cursor = conn.cursor()
        query = "INSERT INTO enquiries (full_name, contact, qualification,
course_interested, lead_status) VALUES (%s, %s, %s, %s, %s)"
        values = (full_name, contact, qualification, course_interested,
lead_status)
        cursor.execute(query, values)
        conn.commit()
        messagebox.showinfo("Success", "Enquiry inserted successfully!")
        conn.close()
        clear_fields()  # Clear input fields
        display_enquiries()  # Refresh the display
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        conn.close()
```

4. Update Enquiry Function

The `update_enquiry()` function is used to update an existing record. It retrieves the `ID` from the input field (`entry_id`) and updates the corresponding record in the database using the **UPDATE** SQL query.

```
def update_enquiry():
    enquiry_id = entry_id.get()
    full_name = entry_full_name.get()
    contact = entry_contact.get()
    qualification = entry_qualification.get()
    course_interested = entry_course_interested.get()
    lead_status = entry_lead_status.get()

    if not enquiry_id:
        messagebox.showwarning("Input Error", "Please provide the enquiry ID
to update.")
        return

    try:
        conn = connect_db()
        cursor = conn.cursor()
        query = "UPDATE enquiries SET full_name=%s, contact=%s,
qualification=%s, course_interested=%s, lead_status=%s WHERE id=%s"
        values = (full_name, contact, qualification, course_interested,
lead_status, enquiry_id)
        cursor.execute(query, values)
        conn.commit()
        messagebox.showinfo("Success", "Enquiry updated successfully!")
        conn.close()
        clear_fields()
        display_enquiries()  # Refresh the grid after update
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        conn.close()
```

5. Delete Enquiry Function

The `delete_enquiry()` function deletes a record based on the `ID` input. It checks if the `ID` is provided, then executes a **DELETE** SQL query to remove the record from the database.

```
def delete_enquiry():
    enquiry_id = entry_id.get()

    if not enquiry_id:
        messagebox.showwarning("Input Error", "Please provide the enquiry ID
to delete.")
        return

    try:
        conn = connect_db()
        cursor = conn.cursor()
        query = "DELETE FROM enquiries WHERE id=%s"
        cursor.execute(query, (enquiry_id,))
        conn.commit()
```

```
        if cursor.rowcount > 0:
            messagebox.showinfo("Success", f"Enquiry with ID {enquiry_id}
deleted successfully!")
            display_enquiries()  # Refresh the grid after deletion
        else:
            messagebox.showwarning("Not Found", f"No enquiry found with ID
{enquiry_id}")

        conn.close()
        clear_fields()
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        conn.close()
```

6. Display Enquiries in Grid

The `display_enquiries()` function fetches all the records from the database and populates a **Treeview** widget (a type of table) in Tkinter. It also supports filtering based on specific fields (e.g., "Full Name", "Contact", etc.).

```
def display_enquiries(search_by="", search_value=""):
    # Clear existing rows
    for row in tree.get_children():
        tree.delete(row)

    try:
        conn = connect_db()
        cursor = conn.cursor()
        if search_by and search_value:
            query = f"SELECT * FROM enquiries WHERE {search_by} LIKE %s"
            cursor.execute(query, (f"%{search_value}%",))
        else:
            query = "SELECT * FROM enquiries"
            cursor.execute(query)

        rows = cursor.fetchall()

        for row in rows:
            tree.insert("", "end", values=row)

        conn.close()
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        conn.close()
```

7. Search Filter Function

The `search_filter()` function is triggered when the user wants to search for a record. The search is done based on a specific column (e.g., "Full Name", "Contact") and its value.

```
def search_filter():
    search_by = search_by_var.get()
    search_value = search_value_entry.get()
```

```
    if search_value.strip():
        display_enquiries(search_by, search_value)
    else:
        messagebox.showwarning("Input Error", "Please enter a value to
search.")
```

8. UI Layout

The program uses **Tkinter** to create the graphical user interface (GUI). It includes:

- **Entry fields** for user inputs: `full_name`, `contact`, `qualification`, `course_interested`, `lead_status`, and `id`.
- **Buttons** for operations: `Insert Enquiry`, `Update Enquiry`, `Delete Enquiry`, and `Clear`.
- **Treeview widget**: Displays the records in a tabular format.
- **Search filter**: Allows filtering records based on selected criteria (like full name, contact, etc.).

9. Selecting a Record from the Grid

The user can click on a record in the grid (Treeview) to fill the form with that record's details for **updating** or **deleting** it. The `on_select_item()` function is responsible for this.

```
def on_select_item(event):
    selected_item = tree.selection()[0]
    selected_id = tree.item(selected_item)["values"][0]  # Get the ID of
selected item

    # Fill in the fields with selected record data
    entry_id.delete(0, tk.END)
    entry_full_name.delete(0, tk.END)
    entry_contact.delete(0, tk.END)
    entry_qualification.delete(0, tk.END)
    entry_course_interested.delete(0, tk.END)
    entry_lead_status.delete(0, tk.END)

    entry_id.insert(0, selected_id)
    entry_full_name.insert(0, tree.item(selected_item)["values"][1])
    entry_contact.insert(0, tree.item(selected_item)["values"][2])
    entry_qualification.insert(0, tree.item(selected_item)["values"][3])
    entry_course_interested.insert(0, tree.item(selected_item)["values"][4])
    entry_lead_status.insert(0, tree.item(selected_item)["values"][5])
```

10. Clear Fields

The `clear_fields()` function clears all the input fields, which is helpful after inserting, updating, or deleting records.

11. Program Flow

- The program starts by displaying all existing records in the Treeview (`display_enquiries()`).
- The user can perform the following actions:
    - **Insert** a new enquiry.
    - **Update** an existing enquiry by selecting a record from the Treeview.
    - **Delete** a record by providing the ID.
    - **Search** for records based on a filter.