Deliverable 5

GROUP 20:

Ankan Mazumdar	A20541357	amazumdar@hawk.iit.edu
Ping-Chun Shih	A20536344	pshih@hawk.iit.edu
Sandra Alrifai	A20554830	salrifai@hawk.iit.edu
Shivani Shrivastav	A20553589	sshrivastav@hawk.iit.edu

The video is recorded in 2 parts, Please watch both-
https://www.loom.com/share/f36681b5c1594a61adb17a0262220df9?sid=f3c9ebfd-491a-44c5-95e3-3ea723ba3a32

https://www.loom.com/share/89562758c65444c38c62ef90af6cae85?sid=a9a11801-e3e8-4387-af1c-8aceae63851d

Python Code-

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

def display_connection_details():
    mydb = connect_mysql()
    if mydb is not None:
        try:
            server_info = mydb.get_server_info()
            host_info = mydb.server_host
            db_info = mydb.database
            messagebox.showinfo("Connection Details", f"Server: {server_info}\nHost: {host_info}\nDatabase:
{db_info}")
        except mysql.connector.Error as e:
            messagebox.showerror("Error", e)
        finally:
            mydb.close()
    else:
        messagebox.showerror("Error", "Database connection was not established.")

def set_background(root, image_path):
    # Load the image
    bg_image = tk.PhotoImage(file=image_path)

    # Create a canvas widget and place it in the root window, covering the entire window
    canvas = tk.Canvas(root, width=root.winfo_screenwidth(), height=root.winfo_screenheight())
    canvas.place(x=0, y=0)
```

```
    # Add a black rectangle to serve as the background
    canvas.create_rectangle(0, 0, root.winfo_screenwidth(), root.winfo_screenheight(), fill="black", outline="")

    # Calculate the coordinates to center the image on the canvas
    x_center = (root.winfo_screenwidth() - bg_image.width()) // 2
    y_center = (root.winfo_screenheight() - bg_image.height()) // 8

    # Add the image at the center of the canvas
    canvas.create_image(x_center, y_center, image=bg_image, anchor="nw")

    # Ensure the image persists
    canvas.image = bg_image

def connect_mysql():
    try:
        mydb = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "########",
        database = "fifa_wc"
        )
        print("Connection successsful")
        return mydb
    except mysql.connector.Error as e:
        print("Error:", e)

def fetch_all_databases():
    mydb = connect_mysql()
    if mydb is not None:
        try:
            mycursor = mydb.cursor()
            mycursor.execute("SHOW DATABASES")
            db_list = [db[0] for db in mycursor.fetchall()]
            mycursor.close()

            # Create a new Toplevel window to display the message
            window = tk.Toplevel()
            window.title("Databases")
            window.geometry("400x300")  # Set the size of the window
            message = tk.Message(window, text="\n".join(db_list), width=350, font=("Helvetica", 14))
            message.pack()

        except mysql.connector.Error as e:
            messagebox.showerror("Error", e)
        finally:
            mydb.close()
    else:
        messagebox.showerror("Error", "Database connection was not established.")

def fetch_all_tables():
```

```
    mydb = connect_mysql()
    if mydb is not None:
        try:
            mycursor = mydb.cursor()
            mycursor.execute("SHOW TABLES")

            # Fetch column names
            columns = ["List of Tables in this DB-"]

            # Fetch all rows
            table_list = [table[0] for table in mycursor.fetchall()]
            mycursor.close()

            # Create a new Toplevel window to display the message
            window = tk.Toplevel()
            window.title("Tables")
            window.geometry("400x300")  # Set the size of the window

            # Create a Text widget to display the data
            result_text = tk.Text(window, width=80, height=20, font=("Helvetica", 16))

            # Create a vertical scrollbar
            scrollbar = tk.Scrollbar(window, command=result_text.yview)
            scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

            result_text.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

            # Configure the Text widget to use the scrollbar
            result_text.config(yscrollcommand=scrollbar.set)

            # Insert column names into the Text widget
            result_text.insert(tk.END, "\t".join(columns) + "\n")

            # Insert the data into the Text widget
            for row in table_list:
                result_text.insert(tk.END, f"{row}\n")

        except mysql.connector.Error as e:
            messagebox.showerror("Error", e)
        finally:
            mydb.close()
    else:
        messagebox.showerror("Error", "Database connection was not established.")


def read_data(table):
    mydb = connect_mysql()
    if mydb is not None:
        try:
            mycursor = mydb.cursor()
            query = f"SELECT * FROM {table}"
```

```
            mycursor.execute(query)

            # Fetch column names
            columns = [desc[0] for desc in mycursor.description]

            # Fetch all rows
            rows = mycursor.fetchall()
            mycursor.close()

            # Create a new Toplevel window to display the message
            window = tk.Toplevel()
            window.title("Data")
            window.geometry("600x400")  # Set the size of the window

            # Create a Text widget to display the data
            result_text = tk.Text(window, width=80, height=20, font=("Helvetica", 10))

            # Create a vertical scrollbar
            scrollbar = tk.Scrollbar(window, command=result_text.yview)
            scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

            result_text.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

            # Configure the Text widget to use the scrollbar
            result_text.config(yscrollcommand=scrollbar.set)

            # Insert column names into the Text widget
            result_text.insert(tk.END, "\t".join(columns) + "\n")

            # Insert the data into the Text widget
            for row in rows:
                result_text.insert(tk.END, "\t".join([str(cell) for cell in row]) + "\n")

        except mysql.connector.Error as e:
            messagebox.showerror("Error", e)
        finally:
            mydb.close()
    else:
        messagebox.showerror("Error", "Database connection was not established.")


def insert_data():
    window_insert = tk.Toplevel()
    window_insert.title("Insert Data")
    window_insert.geometry("400x300")  # Set the size of the window

    tk.Label(window_insert, text="Enter INSERT Query:", font=("Helvetica", 14)).pack()
    query_entry = tk.Text(window_insert, width=50, height=10, font=("Helvetica", 10))
    query_entry.pack()

    def execute_insert():
```

```
        query = query_entry.get("1.0", tk.END)  # Retrieve text content from the Text widget
        try:
            mydb = connect_mysql()
            if mydb:
                mycursor = mydb.cursor()
                mycursor.execute(query)
                mydb.commit()
                messagebox.showinfo("Success", f"{mycursor.rowcount} record(s) inserted.")
                mycursor.close()
                mydb.close()
                window_insert.destroy()
            else:
                messagebox.showerror("Error", "Failed to establish a database connection.")
        except mysql.connector.Error as e:
            messagebox.showerror("Error", e)

    tk.Button(window_insert, text="Insert", command=execute_insert, font=("Helvetica", 10)).pack()

def update_data():
    window_update = tk.Toplevel()
    window_update.title("Update Data")
    window_update.geometry("400x300")  # Set the size of the window

    tk.Label(window_update, text="Enter UPDATE Query:", font=("Helvetica", 14)).pack()
    query_entry = tk.Text(window_update, width=50, height=10, font=("Helvetica", 10))
    query_entry.pack()

    result_text = tk.Text(window_update, width=50, height=10)
    result_text.pack()

    def execute_update():
        query = query_entry.get("1.0", tk.END)
        try:
            mydb = connect_mysql()
            if mydb:
                mycursor = mydb.cursor()
                mycursor.execute(query)
                mydb.commit()
                result_text.insert(tk.END, f"{mycursor.rowcount} record(s) updated.")
                mycursor.close()
                mydb.close()
                window_update.destroy()
            else:
                messagebox.showerror("Error", "Failed to establish a database connection.")
        except mysql.connector.Error as e:
            messagebox.showerror("Error", e)

    tk.Button(window_update, text="Update", command=execute_update, font=("Helvetica", 10)).pack()
    window_query.state('zoomed')

def delete_data():
```

```
    window_delete = tk.Toplevel()
    window_delete.title("Delete Data")
    window_delete.geometry("400x300")  # Set the size of the window

    tk.Label(window_delete, text="Enter DELETE Query:", font=("Helvetica", 14)).pack()
    query_entry = tk.Text(window_delete, width=50, height=10, font=("Helvetica", 10))
    query_entry.pack()

    result_text = tk.Text(window_delete, width=50, height=10)
    result_text.pack()

    def execute_delete():
        query = query_entry.get("1.0", tk.END)
        try:
            mydb = connect_mysql()
            if mydb:
                mycursor = mydb.cursor()
                mycursor.execute(query)
                mydb.commit()
                result_text.insert(tk.END, f"{mycursor.rowcount} record(s) deleted.")
                mycursor.close()
                mydb.close()
                window_delete.destroy()
            else:
                messagebox.showerror("Error", "Failed to establish a database connection.")
        except mysql.connector.Error as e:
            messagebox.showerror("Error", e)

    tk.Button(window_delete, text="Delete", command=execute_delete, font=("Helvetica", 10)).pack()
    window_query.state('zoomed')

def custom_query():
    window_query = tk.Toplevel()
    window_query.title("Custom Query")
    window_query.geometry("800x600")  # Set the size of the window

    tk.Label(window_query, text="Enter SQL Query (semicolon-separated for multiple queries):",
font=("Helvetica", 14)).pack()
    query_entry = tk.Text(window_query, width=100, height=10, font=("Helvetica", 10))
    query_entry.pack()

    result_text = tk.Text(window_query, width=100, height=30)
    result_text.pack()

    def execute_query():
        query = query_entry.get("1.0", tk.END)
        try:
            mydb = connect_mysql()
            if mydb:
                mycursor = mydb.cursor()
                queries = query.split(';')
```

```
            # Clear previous result
            result_text.delete("1.0", tk.END)
            for q in queries:
                if q.strip():
                    mycursor.execute(q)
                    result = mycursor.fetchall()
                    if result:
                        # Fetch column names
                        columns = [desc[0] for desc in mycursor.description]
                        result_text.insert(tk.END, "\t".join(columns) + "\n")
                        # Fetch and display rows
                        for row in result:
                            result_text.insert(tk.END, "\t".join(map(str, row)) + "\n")
                    else:
                        result_text.insert(tk.END, "No results found.")
            mycursor.close()
            mydb.close()
        else:
            messagebox.showerror("Error", "Failed to establish a database connection.")
    except mysql.connector.Error as e:
        messagebox.showerror("Error", e)

tk.Button(window_query, text="Execute Query", command=execute_query, font=("Helvetica", 12)).pack()

window_query.state('zoomed')


def perform_crud():
    window = tk.Tk()
    window.state('zoomed')
    window.title("FIFA Women's World Cup UI")
    set_background(window, r"C:\Users\Ankan Mazumdar\Downloads\CS430\del
4\resize-17135020706358351 42Untitled.png")
    def fetch_databases():
        fetch_all_databases()

    def fetch_tables():
        fetch_all_tables()

    def read_records():
        table_name = input_table.get()
        read_data(table_name)

    def insert_record():
        insert_data()

    def update_record():
        update_data()

    def delete_record():
        delete_data()
```

```
    def show_connection_details():
        display_connection_details()
    tk.Label(window, text="FIFA Women's World Cup Database", font=("Helvetica", 14), fg="red").pack()
  #  tk.Label(window, text="", font=("Helvetica", 10)).pack()
    tk.Label(window, text="Please select any of the below option of your choice", font=("Helvetica", 12)).pack()
    tk.Button(window, text="Show Connection Details", font=("Helvetica", 14), fg="blue", padx=20, pady=10,
command=show_connection_details).pack()
    tk.Label(window, text="", font=("Helvetica", 10)).pack()
    tk.Button(window, text="Display all Databases in the server",font=("Helvetica", 14), fg="blue", padx=10,
pady=10, command=fetch_databases).pack()
    tk.Label(window, text="", font=("Helvetica", 10)).pack()
    tk.Button(window, text="Display all Tables in FIFA_wc DB",   font=("Helvetica", 14), fg="blue", padx=20,
pady=10, command=fetch_tables).pack()
    tk.Label(window, text="", font=("Helvetica", 10)).pack()

    tk.Label(window, text="Enter Table Name & click Read Records to view", font=("Helvetica", 14),
fg="blue").pack()
    input_table = tk.Entry(window, font=("Helvetica", 10), width=30)
    input_table.pack()

    tk.Button(window, text="Read Records", font=("Helvetica", 14), fg="blue", padx=20, pady=10,
command=read_records).pack()
    tk.Label(window, text="", font=("Helvetica", 10)).pack()
    tk.Button(window, text="Insert Record", font=("Helvetica", 14), fg="blue", padx=20,
pady=10,command=insert_record).pack()
    tk.Label(window, text="", font=("Helvetica", 10)).pack()
    tk.Button(window, text="Update Record", font=("Helvetica", 14), fg="blue", padx=20,
pady=10,command=update_record).pack()
    tk.Label(window, text="", font=("Helvetica", 10)).pack()
    tk.Button(window, text="Delete Record", font=("Helvetica", 14), fg="blue", padx=20,
pady=10,command=delete_record).pack()
    tk.Label(window, text="", font=("Helvetica", 10)).pack()
    tk.Button(window, text="Execute Complex Query like Windows, OLAP, Aggregate functions",
font=("Helvetica", 14), fg="blue", padx=20, pady=10,command=custom_query).pack()

    window.mainloop()

if __name__ == "__main__":
    perform_crud()
```

Advanced QUERY used-

1. UNION and OLAP functions:
Find each team's performance over time, showing running total points gained in each subsequent match.

```
SELECT   sub.team,  sub.matchID,  sub.points,
  SUM(sub.points) OVER (PARTITION BY sub.team ORDER BY sub.matchID) AS running_total
FROM (SELECT m.matchID, IF(m.winner = team1, team1, team2) AS team,
    CASE
      WHEN m.winner = 'draw' THEN 1
      WHEN m.winner = team1 THEN 3
```

```
    WHEN m.winner = team2 THEN 3
    ELSE 0
  END AS points FROM Matches m
 UNION ALL
 #Fetch rows where team2 column has winner
 SELECT m.matchID, IF(m.winner = team1, team2, team1) AS team,
  CASE
    WHEN m.winner = 'draw' THEN 1
    ELSE 0
  END AS points FROM Matches m) AS sub ORDER BY sub.team, sub.matchID;
```

2. Show Points table with group wise team rankings with Total Points, number of Wins, Losses, and Draws

```
SELECT g.group_name,
   t.team_name,
   pt.points,
   pt.wins,
   pt.losses,
   pt.draws,
   RANK() OVER (PARTITION BY g.group_name ORDER BY pt.points DESC) as ranking
 FROM
   Teams t
 JOIN
   Groupss g ON t.groupID = g.groupID
 JOIN
   PointsTable pt ON t.team_name = pt.team_name
```

4. Analyze team's wins group wise through percentile rank.

```
select p.team_name, t.groupID, p.losses, p.wins,
ntile(4) over(PARTITION BY groupID order by wins desc) as wins_quartile_rank
from pointsTable p join teams t on p.team_name = t.team_name
join groupss g on t.groupID = g.groupID;
```

WITH Clause:
1. Find the total number of matches played in each stadium:

```
WITH StadiumMatches AS (
SELECT stadiumID, COUNT(*) AS matches_played
FROM Matches
GROUP BY stadiumID
)
SELECT v.stadiumName, sm.matches_played
FROM StadiumMatches sm
JOIN Venues v ON sm.stadiumID = v.stadiumID
order by sm.matches_played desc;
```

2. Top Scorer by Team with CTE:

```
WITH TopScorers AS (
   SELECT
```

```
        t.team_name,
        p.firstName,
        p.lastName,
        p.goals,
        RANK() OVER (PARTITION BY t.team_name ORDER BY p.goals DESC) AS goal_rank
    FROM
        Players p
    JOIN
        Teams t ON p.teamID = t.teamID
)
SELECT
    team_name,
    firstName,
    lastName,
    goals
FROM
    TopScorers
WHERE
    goal_rank = 1;
```

3. Players with Yellow or Red Cards

```
WITH PlayerCards AS (
 SELECT
        CONCAT(p.firstName, ' ', p.lastName) AS player_name,
        pm.YellowCard_flag AS yellow_cards,
        pm.RedCard_flag AS red_cards
    FROM
        Players p
    JOIN
        PlayerMatches pm ON p.PlayerID = pm.PlayerID
    having
        yellow_cards > 0 OR red_cards > 0
)
SELECT
    player_name,
    yellow_cards,
    red_cards
FROM
    PlayerCards order by 3 desc;
```

4. Find the average number of yellow cards per win for each team

```
Select  t.team_name,
  Round(AVG(pm.yellowCard_flag), 2) AS avg_yellow_cards_per_win, pt.wins
From Teams t
Join TeamMatches tm ON t.teamID = tm.teamID
Join Matches m ON tm.matchID = m.matchID
Join PlayerMatches pm ON m.matchID = pm.matchID
Join PointsTable pt ON t.team_name = pt.team_name
WHERE m.winner = t.team_name
```

```
Group by t.team_name, pt.wins
Order by pt.wins DESC;
```