please create mysql and python application on gamingZone

features like types of games and charges to play that, membership offers (yearly,monthly,daily) on hour basis and members table should have hour spent and hours left

1) List the games available in zone

2) list the members registered

3) list yearly, monthly, daily members separately

4) show the name of member with membership type and hours left

5) How many members registered monthly basis

6) how many people played perticular game

7) how many hours perticular game was played

8) which game was played most

import mysql.connector

def connect\_db():

    return mysql.connector.connect(

        host="localhost",

        user="root",

        password="Rathi@123",

        database="gaming\_zone"

    )

def create\_tables():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        CREATE TABLE IF NOT EXISTS games (

            id INT AUTO\_INCREMENT PRIMARY KEY,

            name VARCHAR(100),

            charge\_per\_hour DECIMAL(6,2)

        )

    """)

    cursor.execute("""

        CREATE TABLE IF NOT EXISTS members (

            id INT AUTO\_INCREMENT PRIMARY KEY,

            name VARCHAR(100),

            membership\_type ENUM('yearly', 'monthly', 'daily'),

            hours\_spent INT DEFAULT 0,

            hours\_left INT

        )

    """)

    cursor.execute("""

        CREATE TABLE IF NOT EXISTS game\_play\_log (

            id INT AUTO\_INCREMENT PRIMARY KEY,

            member\_id INT,

            game\_id INT,

            hours\_played INT,

            FOREIGN KEY (member\_id) REFERENCES members(id),

            FOREIGN KEY (game\_id) REFERENCES games(id)

        )

    """)

    conn.commit()

    conn.close()

    print("Tables created.")

def insert\_game(name, charge):

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("INSERT INTO games (name, charge\_per\_hour) VALUES (%s, %s)", (name, charge))

    conn.commit()

    conn.close()

def insert\_member(name, membership\_type):

    hours\_dict = {'yearly': 100, 'monthly': 30, 'daily': 2}

    hours = hours\_dict.get(membership\_type.lower())

    if not hours:

        print("Invalid membership type.")

        return

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("INSERT INTO members (name, membership\_type, hours\_left) VALUES (%s, %s, %s)",

                   (name, membership\_type.lower(), hours))

    conn.commit()

    conn.close()

def log\_game\_play(member\_id, game\_id, hours):

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT hours\_left FROM members WHERE id = %s", (member\_id,))

    result = cursor.fetchone()

    if result and result[0] >= hours:

        cursor.execute("INSERT INTO game\_play\_log (member\_id, game\_id, hours\_played) VALUES (%s, %s, %s)",

                       (member\_id, game\_id, hours))

        cursor.execute("UPDATE members SET hours\_spent = hours\_spent + %s, hours\_left = hours\_left - %s WHERE id = %s",

                       (hours, hours, member\_id))

        conn.commit()

        print("Gameplay logged.")

    else:

        print("Not enough hours.")

    conn.close()

# Queries

def list\_games():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT \* FROM games")

    for row in cursor.fetchall():

        print(row)

    conn.close()

def list\_members():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT \* FROM members")

    for row in cursor.fetchall():

        print(row)

    conn.close()

def list\_by\_membership(mtype):

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT \* FROM members WHERE membership\_type = %s", (mtype,))

    for row in cursor.fetchall():

        print(row)

    conn.close()

def member\_hours\_info():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT name, membership\_type, hours\_left FROM members")

    for row in cursor.fetchall():

        print(row)

    conn.close()

def count\_membership(mtype):

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT COUNT(\*) FROM members WHERE membership\_type = %s", (mtype,))

    print(f"{mtype.capitalize()} members count:", cursor.fetchone()[0])

    conn.close()

def players\_per\_game():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT g.name, COUNT(DISTINCT gl.member\_id) as players

        FROM games g

        JOIN game\_play\_log gl ON g.id = gl.game\_id

        GROUP BY g.name

    """)

    for row in cursor.fetchall():

        print(row)

    conn.close()

def hours\_per\_game():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT g.name, SUM(gl.hours\_played) as total\_hours

        FROM games g

        JOIN game\_play\_log gl ON g.id = gl.game\_id

        GROUP BY g.name

    """)

    for row in cursor.fetchall():

        print(row)

    conn.close()

def most\_played\_game():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT g.name, SUM(gl.hours\_played) as total

        FROM games g

        JOIN game\_play\_log gl ON g.id = gl.game\_id

        GROUP BY g.name

        ORDER BY total DESC

        LIMIT 1

    """)

    print("Most played game:", cursor.fetchone())

    conn.close()

def add\_game():

    name = input("Enter game name: ")

    charge = float(input("Enter game charge: "))

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("INSERT INTO games (name, charge\_per\_hour) VALUES (%s, %s)", (name, charge))

    conn.commit()

    conn.close()

    print("Game added successfully.")

def register\_member():

    name = input("Enter member name: ")

    mtype = input("Enter membership type (yearly/monthly/daily): ").lower()

    insert\_member(name, mtype)

def log\_gameplay():

    member\_id = int(input("Enter member ID: "))

    game\_id = int(input("Enter game ID: "))

    hours = int(input("Enter hours played: "))

    log\_game\_play(member\_id, game\_id, hours)

def delete\_member\_if\_inactive():

    member\_id = int(input("Enter member ID to delete: "))

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT \* FROM game\_play\_log WHERE member\_id = %s", (member\_id,))

    if not cursor.fetchone():

        cursor.execute("DELETE FROM members WHERE id = %s", (member\_id,))

        conn.commit()

        print("Member deleted.")

    else:

        print("Member has gameplay records.")

    conn.close()

def list\_expensive\_games():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT \* FROM games WHERE charge\_per\_hour > 100")

    for row in cursor.fetchall():

        print(row)

    conn.close()

# def count\_games\_by\_type():

#     conn = connect\_db()

#     cursor = conn.cursor()

#     cursor.execute("SELECT game\_type, COUNT(\*) FROM games GROUP BY game\_type")

#     for row in cursor.fetchall():

#         print(f"{row[0]}: {row[1]}")

#     conn.close()

def members\_with\_low\_hours():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT \* FROM members WHERE hours\_left < 10")

    for row in cursor.fetchall():

        print(row)

    conn.close()

def members\_played\_multiple\_games():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT member\_id, COUNT(DISTINCT game\_id)

        FROM game\_play\_log

        GROUP BY member\_id

        HAVING COUNT(DISTINCT game\_id) > 2

    """)

    for row in cursor.fetchall():

        print(row)

    conn.close()

def hours\_remaining\_by\_membership():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT membership\_type, SUM(hours\_left) FROM members GROUP BY membership\_type")

    for row in cursor.fetchall():

        print(row)

    conn.close()

def calculate\_total\_income():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT SUM(g.charge\_per\_hour \* gp.hours\_played)

        FROM game\_play\_log gp

        JOIN games g ON gp.game\_id = g.id

    """)

    result = cursor.fetchone()

    print(f"Total Income: ₹{result[0] if result[0] else 0}")

    conn.close()

def most\_active\_member():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT m.name, SUM(gp.hours\_played) AS total

        FROM game\_play\_log gp

        JOIN members m ON gp.member\_id = m.id

        GROUP BY m.name

        ORDER BY total DESC LIMIT 1

    """)

    print(cursor.fetchone())

    conn.close()

def top\_3\_games():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT g.name, SUM(gp.hours\_played)

        FROM game\_play\_log gp

        JOIN games g ON gp.game\_id = g.id

        GROUP BY g.name

        ORDER BY SUM(gp.hours\_played) DESC LIMIT 3

    """)

    for row in cursor.fetchall():

        print(row)

    conn.close()

def member\_game\_hours\_report():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT m.name, g.name, SUM(gp.hours\_played)

        FROM game\_play\_log gp

        JOIN members m ON gp.member\_id = m.id

        JOIN games g ON gp.game\_id = g.id

        GROUP BY m.name, g.name

    """)

    for row in cursor.fetchall():

        print(row)

    conn.close()

def log\_gameplay\_with\_check():

    name = input("Enter member name: ")

    game = input("Enter game name: ")

    hours = int(input("Enter hours: "))

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT id, hours\_left FROM members WHERE name = %s", (name,))

    m = cursor.fetchone()

    cursor.execute("SELECT id FROM games WHERE name = %s", (game,))

    g = cursor.fetchone()

    if m and g and m[1] >= hours:

        cursor.execute("INSERT INTO game\_play\_log (member\_id, game\_id, hours\_played) VALUES (%s, %s, %s)", (m[0], g[0], hours))

        cursor.execute("UPDATE members SET hours\_left = hours\_left - %s, hours\_spent = hours\_spent + %s WHERE id = %s", (hours, hours, m[0]))

        conn.commit()

        print("Gameplay logged.")

    else:

        print("Invalid or insufficient hours.")

    conn.close()

def members\_used\_75\_percent():

    hours\_dict = {'yearly': 100, 'monthly': 30, 'daily': 2}

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("SELECT name, membership\_type, hours\_left FROM members")

    for name, mtype, left in cursor.fetchall():

        allowed = hours\_dict[mtype]

        used = allowed - left

        if used / allowed > 0.75:

            print(f"{name} used {used}/{allowed} hours")

    conn.close()

def detailed\_member\_report():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT m.name, m.membership\_type,

               COUNT(DISTINCT gp.game\_id) AS games\_played,

               SUM(gp.hours\_played) AS hours\_played,

               m.hours\_left

        FROM members m

        LEFT JOIN game\_play\_log gp ON m.id = gp.member\_id

        GROUP BY m.id

    """)

    for row in cursor.fetchall():

        print(row)

    conn.close()

def inactive\_members():

    conn = connect\_db()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT name FROM members m

        LEFT JOIN game\_play\_log gp ON m.id = gp.member\_id

        WHERE gp.member\_id IS NULL

    """)

    for row in cursor.fetchall():

        print(row[0])

    conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

    create\_tables()

    # Sample Data

    insert\_game("Car Racing", 100)

    insert\_game("Shooting", 120)

    insert\_member("Arjun", "monthly")

    insert\_member("Neha", "daily")

    insert\_member("Ravi", "yearly")

    log\_game\_play(1, 1, 2)

    log\_game\_play(2, 2, 1)

    log\_game\_play(3, 2, 3)

    print("\n1. Games Available:")

    list\_games()

    print("\n2. Members Registered:")

    list\_members()

    print("\n3. Yearly Members:")

    list\_by\_membership("yearly")

    print("\n4. Member Hours Info:")

    member\_hours\_info()

    print("\n5. Monthly Member Count:")

    count\_membership("monthly")

    print("\n6. Players per Game:")

    players\_per\_game()

    print("\n7. Hours per Game:")

    hours\_per\_game()

    print("\n8. Most Played Game:")

    most\_played\_game()

    print("\n9. Add Game")

    add\_game()

    print("\n10. Register Member")

    register\_member()

    print("\n11. Log Gameplay")

    log\_gameplay()

    print("\n12. Delete Inactive Member")

    delete\_member\_if\_inactive()

    print("\n13. List Expensive Games")

    list\_expensive\_games()

    # print("\n14. Count Games by Type")

    # count\_games\_by\_type()

    print("\n15. Members with <10 Hours")

    members\_with\_low\_hours()

    print("\n16. Members Played >2 Games")

    members\_played\_multiple\_games()

    print("\n17. Hours Remaining by Membership")

    hours\_remaining\_by\_membership()

    print("\n18. Total Income")

    calculate\_total\_income()

    print("\n19. Most Active Member")

    most\_active\_member()

    print("\n20. Top 3 Games")

    top\_3\_games()

    print("\n21. Member-Game Report")

    member\_game\_hours\_report()

    print("\n22. Log Gameplay With Check")

    log\_gameplay\_with\_check()

    print("\n23. Members Used >75%")

    members\_used\_75\_percent()

    print("\n24. Detailed Member Report")

    detailed\_member\_report()

    print("\n25. Inactive Members")

    inactive\_members()