```python
import psycopg2
import time

def confirm_action():
    while True:
        user_input = input("Confirm operation? 'y' for yes, 'n' for no: ")
        if user_input == "y":
            return True
        elif user_input == "n":
            return False
        else:
            print("Enter a valid response.\n")

def print_players(cur):
    cursor = cur

    query = """
    SELECT *
    FROM player
    ORDER BY player_id ASC
    """

    cursor.execute(query)
    players = cursor.fetchall()

    print("Current players registered in the casino are as follows:")
    for player in players:
        print(f"ID: {player[0]:<4} | NAME: {player[1]:<15}")
    print("\n")

    answer = input("Display more information? 'y' for yes, 'n' for no: ")
    if answer == "y":
        for player in players:
            print(f"ID: {player[0]:<5} | NAME: {player[1]:<15} | BALANCE: {player[2]:<8} | NET WON: {player[3]:<5}")
        print("\n")

    return 0

def cash_update(conn, cur, player_id, new_balance):
    id = player_id
    balance = new_balance
    cursor = cur

    query = """
    UPDATE player
    SET balance = %s
    WHERE player_id = %s
    """

    try:
        cursor.execute(query, (balance, id))
        conn.commit()

    except psycopg2.Error as e:
        print("Other Error")
        print(e)
        conn.rollback()
        return 0

    finally:
        conn.close()
        return 0

def won_update(conn, cur, player_id, amount):
    id = player_id
    cursor = cur

    query = """
    UPDATE player
    SET net_won = %s
    WHERE player_id = %s
    """

    try:
        cursor.execute(query, (amount, id))
        conn.commit()

    except psycopg2.errors.UndefinedTable:
        print("Error: One or more tables in the query do not exist.")
        conn.rollback()
    except psycopg2.Error as e:
        print("Other Error")
        print(e)
        conn.rollback()
        return 0
    finally:
        conn.close()
        return 0

def cash_out(conn, cur):
    cursor = cur
    player_found = None
    selected_player_id = 0
    cashin_value = 0

    query = """
    SELECT *
    FROM player
    """

    cursor.execute(query)
    players = cursor.fetchall()

    while True:
        while True:
            try:
                selected_player_id = int(input("Player ID: "))
                for player in players:
                    if player[0] == selected_player_id:
                        player_found = True
                        break

                if player_found == True:
                    break
                else:
                    print("Player has not been found.")

            except ValueError:
                    print("Enter a valid ID number.")

        while True :
            try:
                cashin_value = int(input("Enter cash out value (>0): "))
                if cashin_value > 0:
                    balance = player[2]
                    new_balance = balance - cashin_value
                    print(f"\nCURR BALANCE: {balance}\nNEW BALANCE: {new_balance}\n")

                    if new_balance < 0:
                        print("Cashing out more than available.\n")

                    else:
                        confirm = confirm_action()
                        if confirm == True:
                            cash_update(conn, cur, selected_player_id, new_balance)
                            return 0

                        elif confirm == False:
                            continue

                else:
                    print("Enter a number bigger than 0.\n")

            except ValueError:
                print("Enter a valid number.\n")

def cash_in(conn, cur):
    cursor = cur
    player_found = None
    selected_player_id = 0
    cashin_value = 0

    query = """
    SELECT *
    FROM player
    """

    cursor.execute(query)
    players = cursor.fetchall()

    while True:
        while True:
            try:
                selected_player_id = int(input("Player ID: "))
                for player in players:
                    if player[0] == selected_player_id:
                        player_found = True
                        break

                if player_found == True:
                    break
                else:
```

```python
                    print("Player has not been found.")

            except ValueError:
                    print("Enter a valid ID number.")

        while True :
            try:
                cashin_value = int(input("Enter cash out value (>0): "))
                if cashin_value > 0:
                    balance = player[2]
                    new_balance = balance + cashin_value
                    print(f"\nCURR BALANCE: {balance}\nNEW BALANCE: {new_balance}\n")

                    confirm = confirm_action()
                    if confirm == True:
                        cash_update(conn, cur, selected_player_id, new_balance)
                        return 0

                    elif confirm == False:
                        continue

                else:
                    print("Enter a number bigger than 0.\n")

            except ValueError:
                print("Enter a valid number.\n")

def record_result(conn, cur):
    cursor = cur
    player_found = None
    selected_player_id = 0
    amount_won = 0

    query = """
    SELECT *
    FROM player
    """

    cursor.execute(query)
    players = cursor.fetchall()

    while True:
        while True:
            try:
                selected_player_id = int(input("Player ID: "))
                for player in players:
                    if player[0] == selected_player_id:
                        player_found = True
                        break

                if player_found == True:
                    break
                else:
                    print("Player has not been found.")

            except ValueError:
                    print("Enter a valid ID number.")

        while True :
            try:
                amount_won = int(input("Enter won/lost(negative) amount: "))
                balance = player[2]
                new_netwon = balance + amount_won

                confirm = confirm_action()
                if confirm == True:
                    won_update(conn, cur, selected_player_id, new_netwon)
                    return 0

                elif confirm == False:
                    continue

            except ValueError:
                print("Enter a valid number.\n")

def get_top_player(connection, cursor):
    try:
        # Query to find the player with the most sessions
        query = """
        SELECT player.name, COUNT(plays.session_id) AS sessions_played
        FROM plays
        JOIN player ON plays.player_id = player.player_id
        GROUP BY player.player_id, player.name
        ORDER BY sessions_played DESC
        LIMIT 1;
        """
        cursor.execute(query)
        result = cursor.fetchone()

        if result:
            print(f"\nTop Player: {result[0]} with {result[1]} sessions played.")
        else:
            print("No player data available.")

    except psycopg2.errors.UndefinedTable:
        print("Error: One or more tables in the query do not exist.")
        connection.rollback()
    except psycopg2.errors.DatabaseError as e:
        print("Database error occurred.")
        print(e)
        connection.rollback()

    return 0

def get_worst_player(connection, cursor):
    try:
        # Query to find the player with the least sessions
        query = """
        SELECT player.name, COUNT(plays.session_id) AS sessions_played
        FROM plays
        JOIN player ON plays.player_id = player.player_id
        GROUP BY player.player_id, player.name
        ORDER BY sessions_played ASC
        LIMIT 1;
        """
        cursor.execute(query)
        result = cursor.fetchone()

        if result:
            print(f"\nWorst Player: {result[0]} with {result[1]} sessions played.")
        else:
            print("No player data available.")

    except psycopg2.errors.UndefinedTable:
        print("Error: One or more tables in the query do not exist.")
        connection.rollback()
    except psycopg2.errors.DatabaseError as e:
        print("Database error occurred.")
        print(e)
        connection.rollback()

    return 0

about = """
-----------------------------------------------------------------------------------------
Microsoft Casino Database User Interface by team Microsoft

a Python script that runs SQL queries to modify the casinos's database\n
utilizes SQL

(made for CSCI 421, Project 2 Part B)

About page.
-----------------------------------------------------------------------------------------
"""

usage = """
-----------------------------------------------------------------------------------------
USAGE) Enter:
'c' or "cash" to cash in or cash out money for a player.
'p' or "players" for viewing information about players.
    When prompted, "What would you like to view?", enter:
    't' to view the player with the most sessions played.
    'w' to view the player with the least sessions played.
    'p' to view all players and their information if desired.
'r' or "record" to record results of a game for a player.
    Project purpose description: This modifies the "net_won" attribute.

"about" to display more information about the program.

'q' or "quit" to exit the program.

**Not case sensitve.
Usage page.
-----------------------------------------------------------------------------------------
"""

menu = """
-----------------------------------------------------------------------------------------
Microsoft Casino Database User Interface (MCDB UI)

Enter 'u' or 'usage' to learn more about the options.
Enter "about" for more information about the program.
```

```
Menu page.
-----------------------------------------------------------------------------------------
"""

opt_dict = {
    "c":1,
    "cash":1,

    "p":2,
    "players":2,

    "r":3,
    "record":3,

    "about":997,

    "u":998,
    "usage":998,

    "q":999,
    "quit":999
}

def process(conn, cur, opt):
    exit = None
    if opt == 1:
        io = input("Cash in or out? 'i' for in, 'o' for out: ")
        if io == "o":
            exit = cash_out(conn, cur)
        elif io == "i":
            exit = cash_in(conn, cur)

    elif opt == 2:
        popt = None
        while True:
            popt = input("What would you like to view? 'u' for Usage: ")
            if popt == 'p':
                exit = print_players(cur)
                break
            elif popt == 't':
                exit = get_top_player(conn, cur)
                break
            elif popt == 'w':
                exit = get_worst_player(conn, cur)
                break
            elif popt == 'u':
                print(usage)
            else:
                print("You've entered an invalid option. Refer to the usage printed below.")
                time.sleep(1)
                print(usage)

    elif opt == 3:
        record_result(conn, cur)
    elif opt == 4:
        pass
    elif opt == 5:
        pass
    elif opt == 997:
        print(about)
        exit = 0
    elif opt == 998:
        print(usage)
        exit = 0
    elif opt == 999:
        exit = 1
    else:
        exit = 0

    return exit

status = 0
while status == 0:

    process_opt = None
    opt = None
    w1 = True
    w2 = True

    print(menu)


    while w1 == True :
        opt = input("Enter your option after the colon: ").lower()
        if opt in opt_dict:
            w1 == False
            break
```

```python
    else:
        print("Invalid input. Enter a valid option")
        print(usage)

conn = psycopg2.connect(
    dbname="microsoft"
    )
cur = conn.cursor()


while w2 == True:
    status = process(conn, cur, opt_dict[opt])

    if status == -1:
        while True:
            process_opt = input(
                """Retry option? Press 'y' for yes, 'n' for no."""
                ).lower()

            if process_opt == "n":
                w2 = False
                break
            elif process_opt == "y":
                break
            else:
                print("Invalid input. Enter a valid option.")

    else:
        w2 = False

if status == 1:
    cur.close()
    conn.close()
    print("Successfully quit the program.")
```