**Tkinter REPORT**

**PRIYANSH SHAH**

**TABLE CREATION AND INSERTION:**

SQL> create table employee(empid varchar(10),name varchar(20),contact char(10),role varchar(20),constraint emp\_pk primary key(empid));

Table created.

SQL> create table department(dcode varchar(10),name varchar(20),location char(10));

Table created.

SQL> create table project(pid varchar(10),title varchar(20),startdate date,enddate date,constraint pid\_pk primary key(pid));

Table created.

SQL> create table client(cid varchar(10),name varchar(20),contact char(10),constraint cid\_pk primary key(cid));

Table created.

SQL> create table task(tid varchar(10),description varchar(20),constraint tid\_pk primary key(tid));

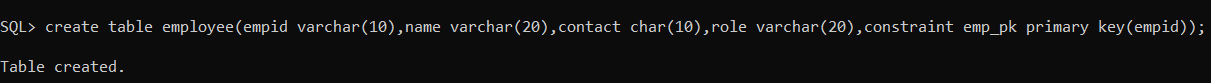
Table created.

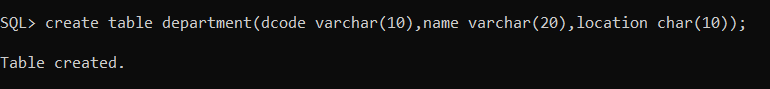
SQL> create table e\_p (empid varchar(10),pid varchar(10),constraint eid\_fk foreign key(empid) references employee,constraint pid\_fk foreign key(pid) references project);

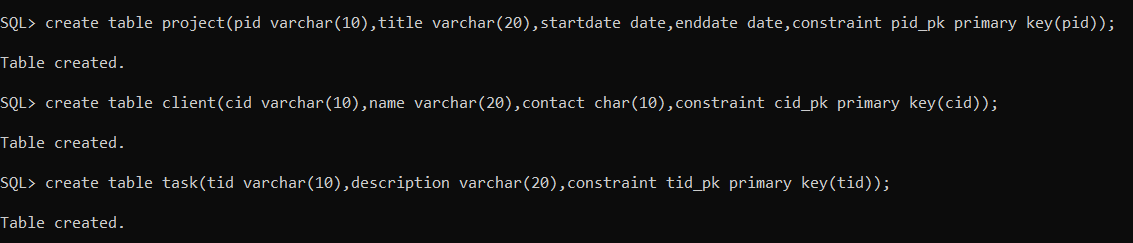
Table created.

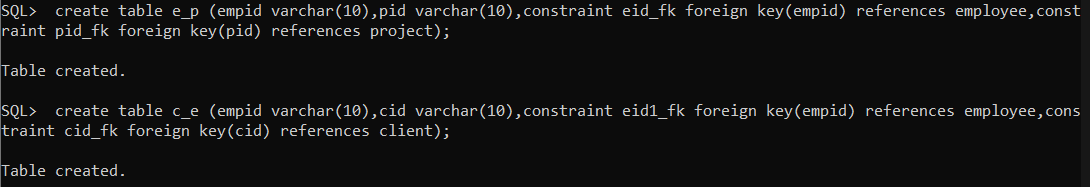
SQL> create table c\_e (empid varchar(10),cid varchar(10),constraint eid1\_fk foreign key(empid) references employee,constraint cid\_fk foreign key(cid) references client);

Table created.









SQL> insert into employee values('e1','priyansh',7434003224,'teacher');

1 row created.

SQL> insert into employee values('e2','samyank',9434003224,'sweeper');

1 row created.

SQL> insert into employee values('e3','niyanta',6434003224,'aka');

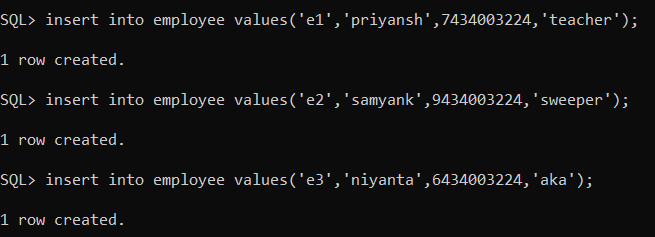
1 row created.

SQL> insert into employee values('e4','dhruvi',1434003224,'actress');

1 row created.

SQL> insert into employee values('e5','rashi',3434003224,'hr');

1 row created.



A black background with white text

Description automatically generated

SQL> insert into department values('d1','xyz','gandhidham');

1 row created.

SQL> insert into department values('d2','pro','gandhidham');

1 row created.

SQL> insert into department values('d3','tcs','panvel');

1 row created.

SQL> insert into department values('d4','stt','panvel');

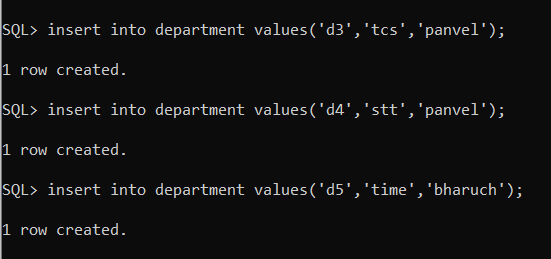
1 row created.

SQL> insert into department values('d5','time','bharuch');

1 row created

A black screen with white text

Description automatically generated



SQL> insert into project values('p1','dbms','12-jan-2024','16-jan-2024');

1 row created.

SQL> insert into project values('p2','sql','12-feb-2024','16-feb-2024');

1 row created.

SQL> insert into project values('p3','movie','12-mar-2024','16-mar-2024');

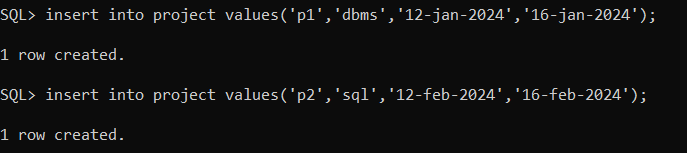
1 row created.

SQL> insert into project values('p4','dsa','11-mar-2024','19-mar-2024');

1 row created.

SQL> insert into project values('p5','dai','12-mar-2024','19-mar-2024');

1 row created.



A screen shot of a computer

Description automatically generated

SQL> insert into client values('c1','harshit','7894563269');

1 row created.

SQL> insert into client values('c2','monish','9894563269');

1 row created.

SQL> insert into client values('c3','rahul','8894563269');

1 row created.

SQL> insert into client values('c4','ram','5894563269');

1 row created.

SQL> insert into client values('c5','rim','3894563269');

1 row created.

A screenshot of a computer

Description automatically generated

SQL> insert into task values('t1','database');

1 row created.

SQL> insert into task values('t2','structures');

1 row created.

SQL> insert into task values('t3','html');

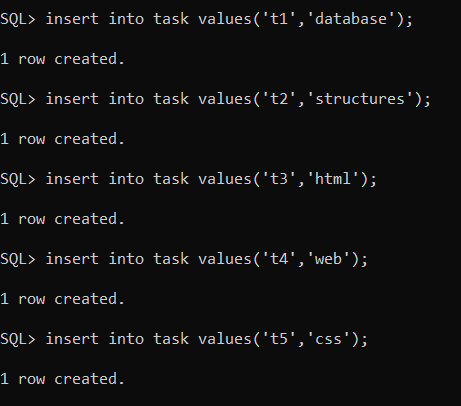
1 row created.

SQL> insert into task values('t4','web');

1 row created.

SQL> insert into task values('t5','css');

1 row created.



SQL> insert into e\_p values('e1','p1');

1 row created.

SQL> insert into e\_p values('e3','p2');

1 row created.

SQL> insert into e\_p values('e4','p2');

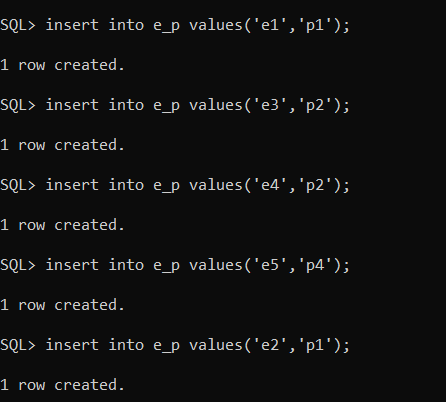
1 row created.

SQL> insert into e\_p values('e5','p4');

1 row created.

SQL> insert into e\_p values('e2','p1');

1 row created.



SQL> insert into c\_e values('e1','c1');

1 row created.

SQL> insert into c\_e values('e2','c3');

1 row created.

SQL> insert into c\_e values('e2','c4');

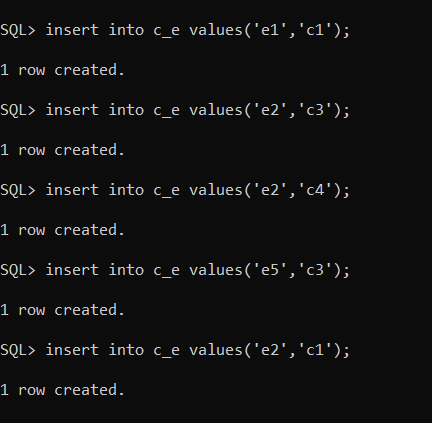
1 row created.

SQL> insert into c\_e values('e5','c3');

1 row created.

SQL> insert into c\_e values('e2','c1');

1 row created.

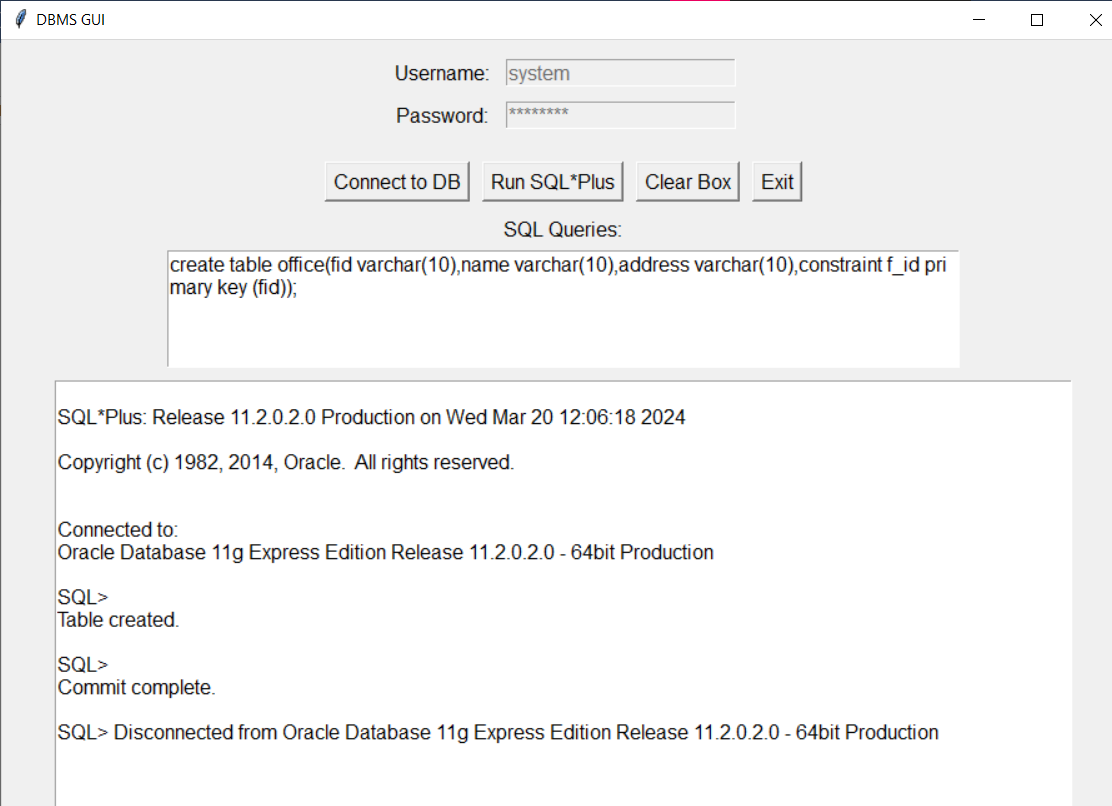


**SQL QUERIES:**

**1.create office table**

create table office(fid varchar(10),name varchar(10),address varchar(10),constraint f\_id primary key (fid));

SQL>

Table created.

SQL> insert into office values('f1','xyz','gujarat');

1 row created.

SQL> insert into office values('f2','abc','gujarat');

1 row created.

SQL> insert into office values('f3','tcs','mumbai');

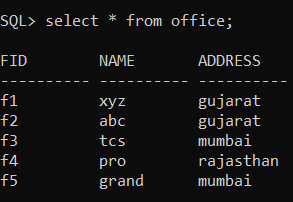
1 row created.

SQL> insert into office values('f4','pro','rajasthan');

1 row created.

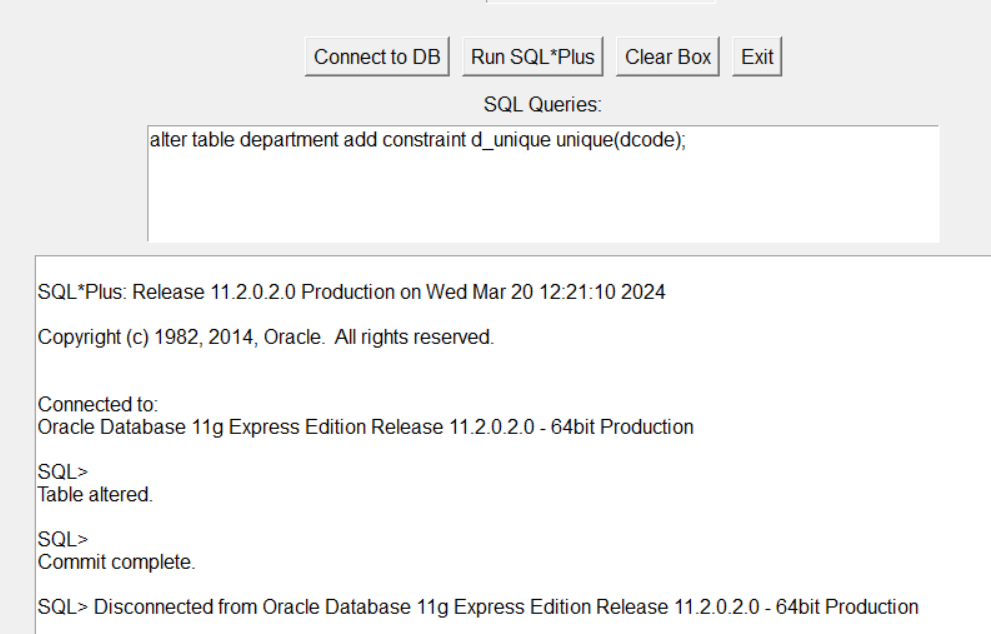
SQL> insert into office values('f5','grand','mumbai');

1 row created.

****

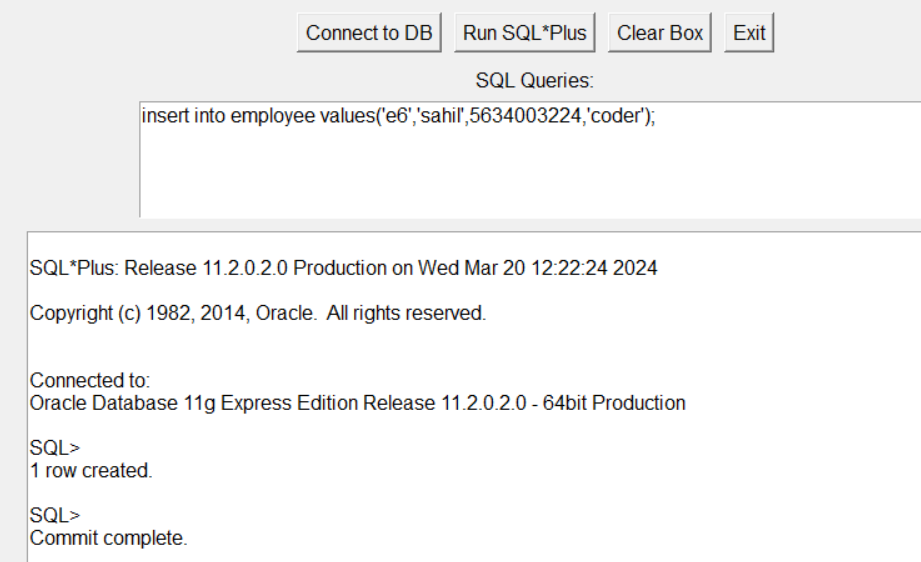
**2.add any constraint to any one identity**

alter table department add constraint d\_unique unique(dcode);



**3.insert new employee record**

insert into employee values('e6','sahil',5634003224,'coder');

****

**4.display name from employee**

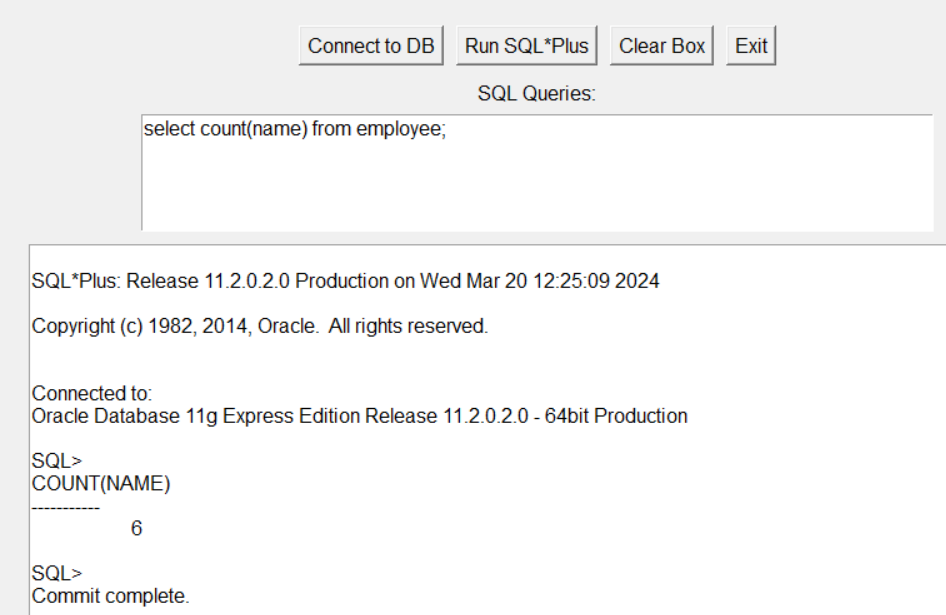
select name from employee;

A screenshot of a computer

Description automatically generated

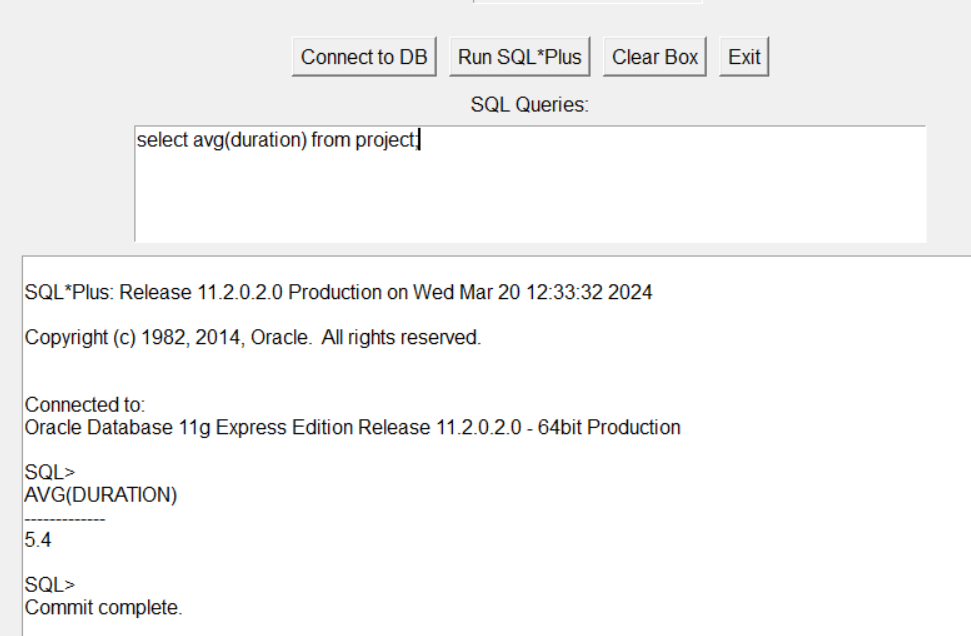
**5.calculate total number of employees**

select count(name) from employee;

****

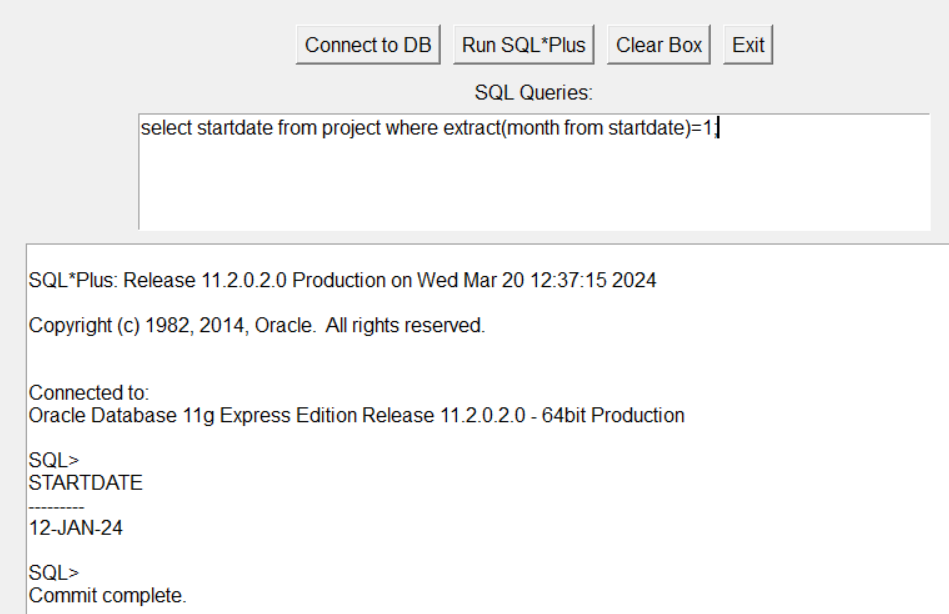
**6.find average duration of project**

select avg(duration) from project;

****

**7.list the project that started in jan**

select startdate from project where extract(month from startdate)=1;

****

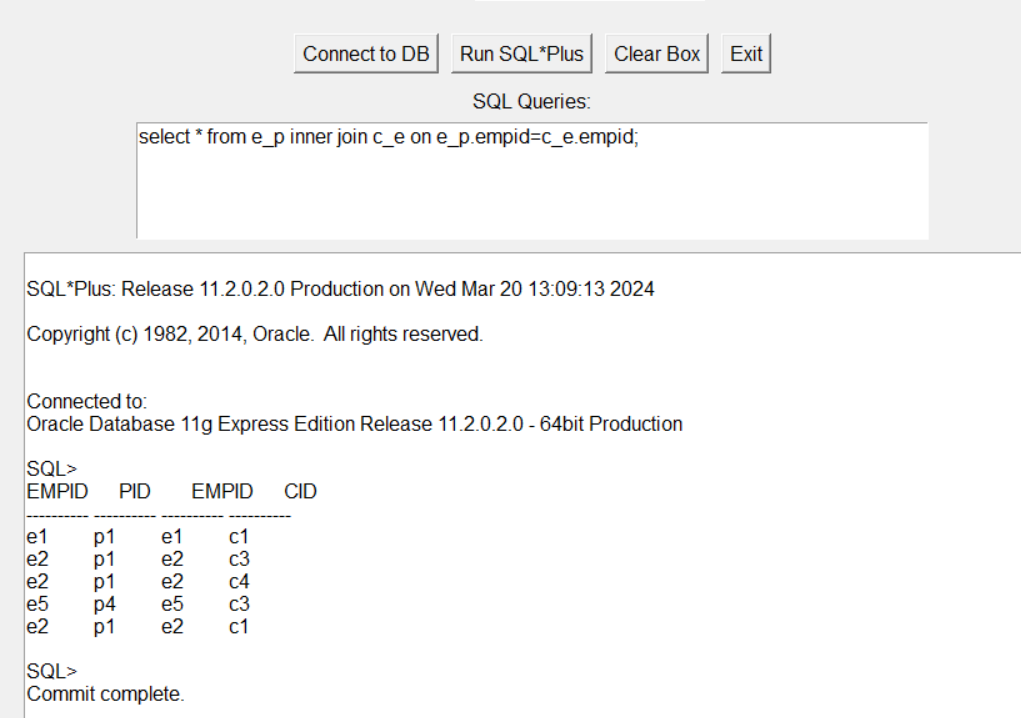
**8.find average duration of project**

**A screenshot of a computer

Description automatically generated**select avg(months\_between(enddate, startdate)) AS avg\_duration from project;

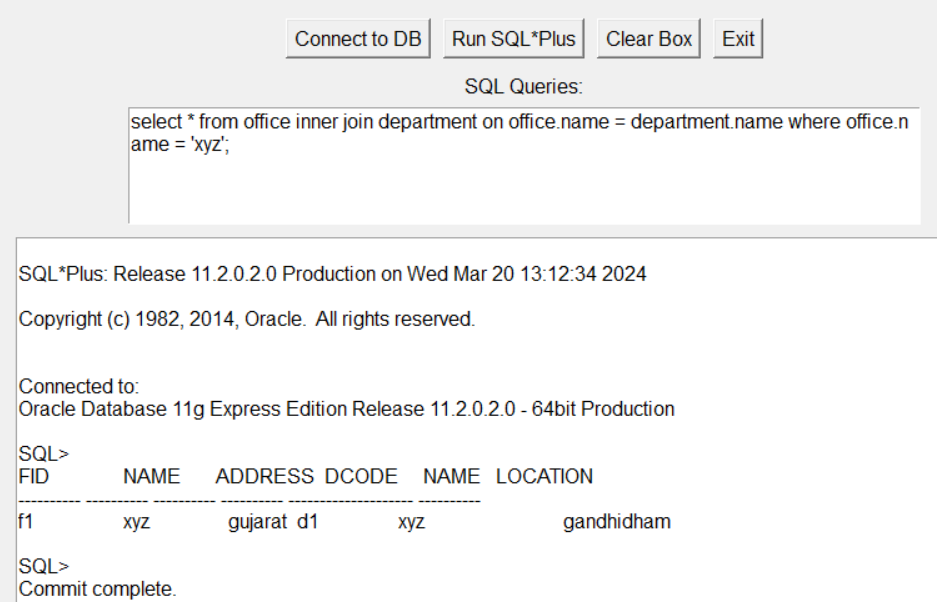
**9.display any inner join**

select \* from e\_p inner join c\_e on e\_p.empid=c\_e.empid;

****

**10.list details of department located in specific office**

select \* from office inner join department on office.name = department.name where office.name = 'xyz';



**PYTHON CODE FOR CONNECTIVITY:**

import subprocess

import tkinter as tk

# Global variables to store connection and cursor objects

con = None

cursor = None

def connect\_to\_db():

    global con, cursor

    # Get connection details from the input fields

    global username, password

    username = username\_entry.get()

    password = password\_entry.get()

    hostname = "asus-pc"  # Default hostname

    port = "1521"  # Default port

    service\_name = "XE"  # Default service name

    # Construct the SQL\*Plus command

    sqlplus\_command = f"sqlplus {username}/{password}@{hostname}:{port}/{service\_name}"

    try:

        # Execute the SQL\*Plus command

        sqlplus\_process = subprocess.Popen(sqlplus\_command, stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)

        # Create connection and cursor objects

        con = sqlplus\_process

        cursor = sqlplus\_process.stdin

        # Disable username and password entry fields

        username\_entry.config(state="disabled")

        password\_entry.config(state="disabled")

        # Enable SQL query entry field and buttons

        sql\_query\_entry.config(state="normal")

        run\_button.config(state="normal")

        clear\_button.config(state="normal")

        exit\_button.config(state="normal")

        # Display message

        output\_text.delete(1.0, tk.END)

        output\_text.insert(tk.END, "Connected to database.")

    except Exception as e:

        error\_text.delete(1.0, tk.END)

        error\_text.insert(tk.END, f"An error occurred: {e}")

def run\_sqlplus():

    global con, cursor

    connect\_to\_db()

    # Check if the connection is established

    if con is None or cursor is None:

        error\_text.delete(1.0, tk.END)

        error\_text.insert(tk.END, "Please connect to the database first.")

        return

    # Get SQL queries from the input field

    queries = sql\_query\_entry.get("1.0", tk.END).strip().split(";")

    try:

        # Execute each SQL query

        for query in queries:

            if query.strip():

                cursor.write((query.strip() + ";\n").encode())

        cursor.write(b"COMMIT;\n")  # Ensure changes are committed

        cursor.flush()  # Flush the buffer

        # Read the output until EOF is reached

        stdout, stderr = con.communicate()

        # Display the output

        output\_text.delete(1.0, tk.END)

        output\_text.insert(tk.END, stdout.decode())

        # Check for errors

        if stderr:

            error\_text.delete(1.0, tk.END)

            error\_text.insert(tk.END, stderr.decode())

    except Exception as e:

        error\_text.delete(1.0, tk.END)

        error\_text.insert(tk.END, f"An error occurred: {e}")

def clear\_text():

    # Clear input and output text boxes

    sql\_query\_entry.delete(1.0, tk.END)

    output\_text.delete(1.0, tk.END)

    error\_text.delete(1.0, tk.END)

def exit\_program():

    # Close the application

    window.destroy()

# Create a Tkinter window

window = tk.Tk()

window.title("DBMS GUI")

# Set window size

window.geometry("900x900")

# Create a frame for the input fields

input\_frame = tk.Frame(window)

input\_frame.pack(pady=10)

# Create input fields

tk.Label(input\_frame, text="Username:", font=("Arial", 12)).grid(row=0, column=0, padx=5, pady=5)

username\_entry = tk.Entry(input\_frame, font=("Arial", 12))

username\_entry.grid(row=0, column=1, padx=5, pady=5)

tk.Label(input\_frame, text="Password:", font=("Arial", 12)).grid(row=1, column=0, padx=5, pady=5)

password\_entry = tk.Entry(input\_frame, show="\*", font=("Arial", 12))

password\_entry.grid(row=1, column=1, padx=5, pady=5)

# Create buttons

button\_frame = tk.Frame(window)

button\_frame.pack(pady=5)

connect\_button = tk.Button(button\_frame, text="Connect to DB", command=connect\_to\_db, font=("Arial", 12))

connect\_button.grid(row=0, column=0, padx=5, pady=5)

run\_button = tk.Button(button\_frame, text="Run SQL\*Plus", command=run\_sqlplus, state="disabled", font=("Arial", 12))

run\_button.grid(row=0, column=1, padx=5, pady=5)

clear\_button = tk.Button(button\_frame, text="Clear Box", command=clear\_text, state="disabled", font=("Arial", 12))

clear\_button.grid(row=0, column=2, padx=5, pady=5)

exit\_button = tk.Button(button\_frame, text="Exit", command=exit\_program, state="disabled", font=("Arial", 12))

exit\_button.grid(row=0, column=3, padx=5, pady=5)

# Create SQL query entry field

tk.Label(window, text="SQL Queries:", font=("Arial", 12)).pack()

sql\_query\_entry = tk.Text(window, height=5, width=70, state="disabled", font=("Arial", 12))

sql\_query\_entry.pack(padx=10, pady=5)

# Create output and error display

output\_text = tk.Text(window, height=35, width=90, font=("Arial", 12))

output\_text.pack(padx=10, pady=(5, 10))

error\_text = tk.Text(window, height=3, width=90, font=("Arial", 12))

error\_text.pack(padx=10, pady=(0, 5))

# Run the Tkinter event loop

window.mainloop()

**GUI IMAGE:**

