***Advanced Database System Lab***

**Assignment no. 2**

**Date: 04-02-2023**

PRN:2020BTECS00005

Name: Sanket Shivaji Jadhav

* **Title:** Installation, configuration and testing of Oracle 18c XE & MySQL.
* **Aim:** To study the configuration of Oracle 18c XE & MySQL& build Python GUI Application.
* **Introduction:**

*Oracle 18c XE:*

* Connect Oracle Database to your favorite programming languages and dev environments including Java, .NET, Python, Node.js, Go, PHP, C/C++ and more.
* Learn SQL on the world's leading relational database, or experiment with Oracle's native support for JSON documents and spatial & graph data.
* Use free dev tools and IDEs from Oracle including SQL Developer, SQLcl, and SQL Developer Data Modeler.
* Install free Oracle REST Data Services (ORDS) to REST-enable your database.
* For low-code app development, run Oracle APEX on top of ORDS and XE at no extra cost to rapidly build data-centric web apps that look beautiful in mobile and desktop browsers.

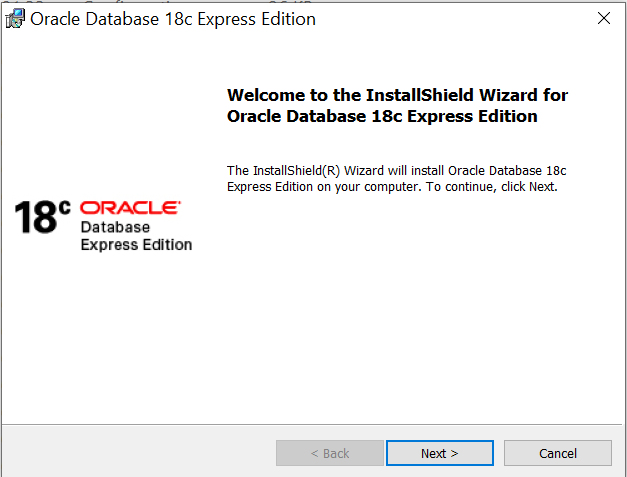
*MySQL:*

MySQL is an open-source relational database management system (RDBMS) .Its name is a combination of "My", the name of co-founder Michael Widenius's daughter My, and "SQL", the acronym for Structured Query Language. A relational database organizes data into one or more data tables in which data may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

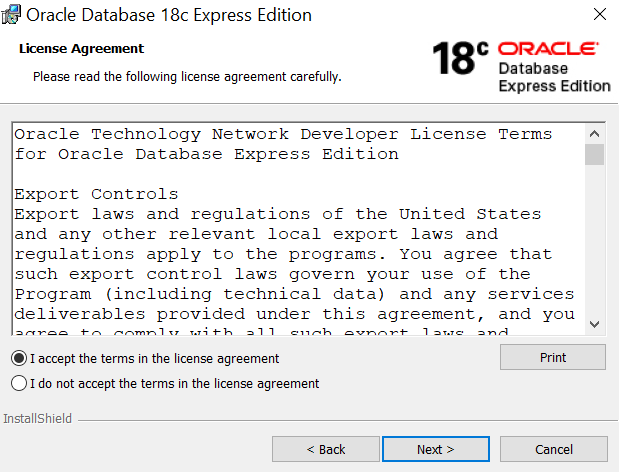
* **Procedure**:
* ***Oracle 18c XE***

**Oracle Server Installation:**

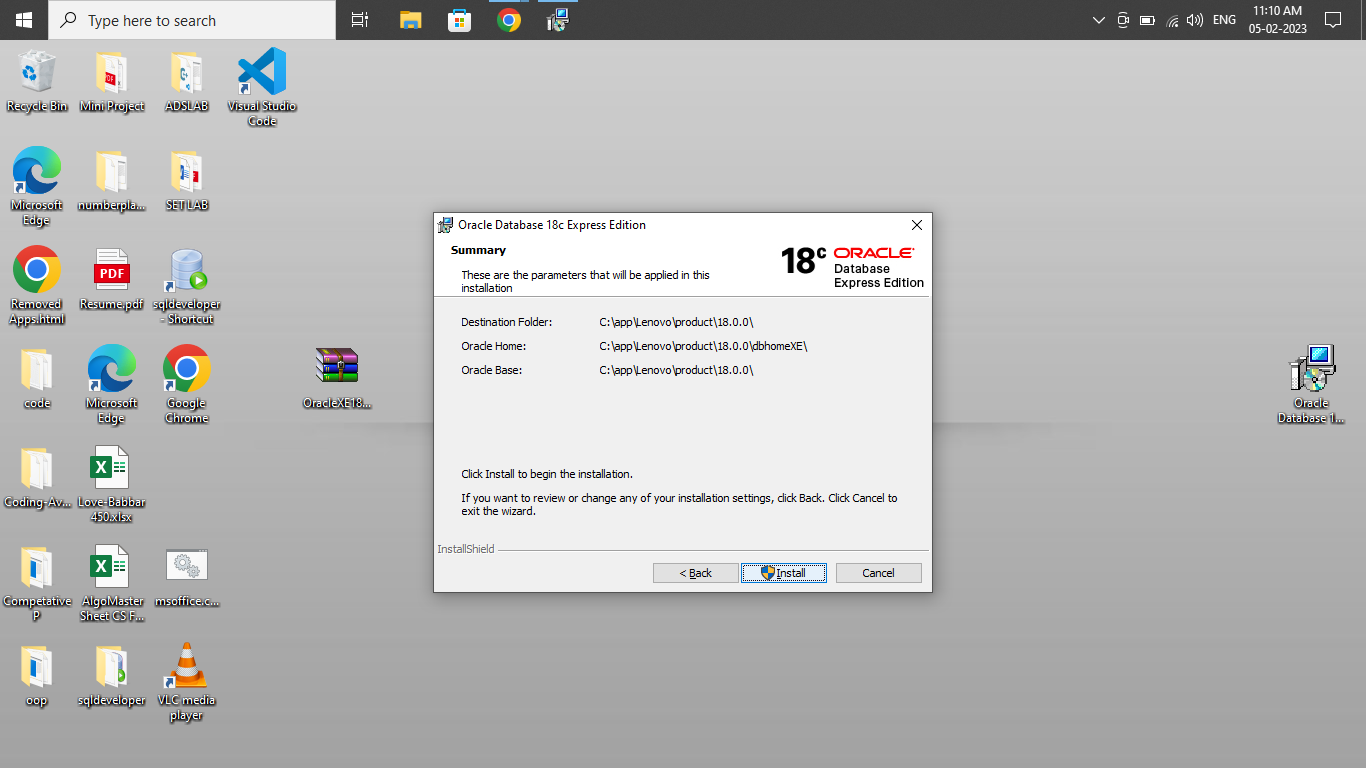
1. Download the Oracle 18c XE file from [oracle website](https://www.oracle.com/database/technologies/xe-downloads.html) for your OS. Extract the zip file and open the setup.exe file.



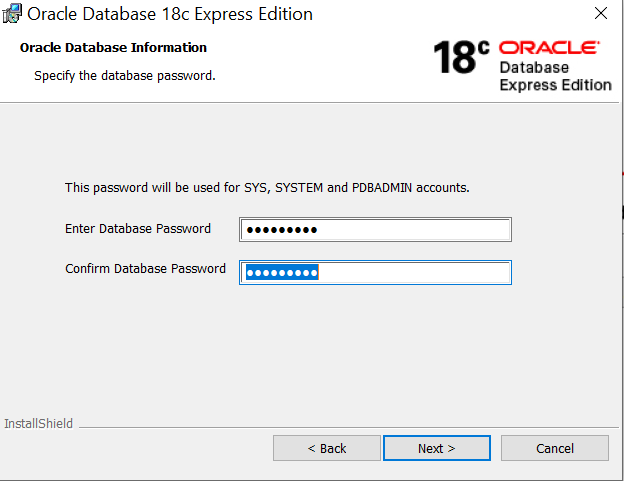
1. Read & accept the License Agreement.



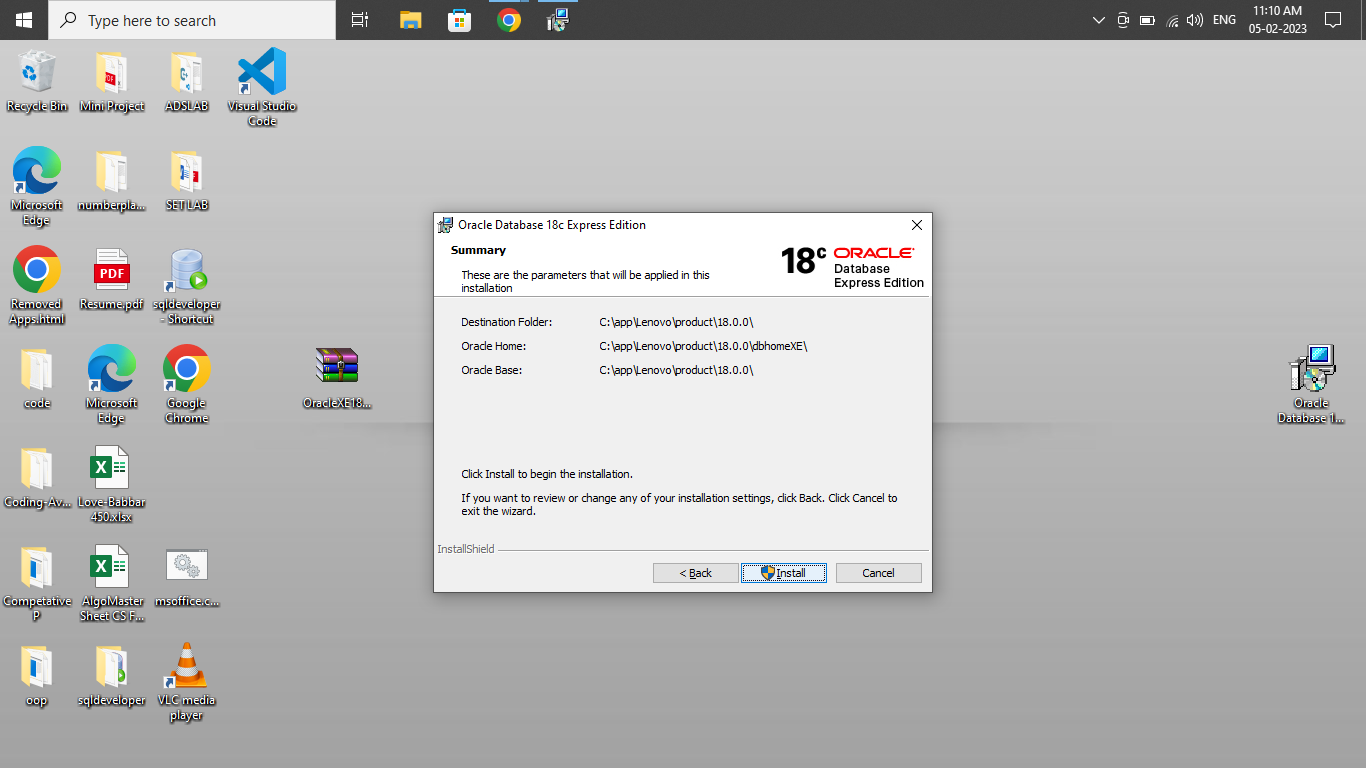
1. Choose the destination to install the Oracle 18c Database.



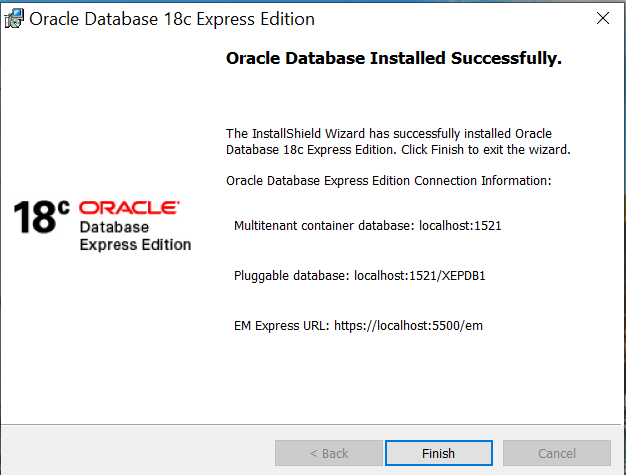
1. Enter the password for the database.



1. Verify the selected parameters.

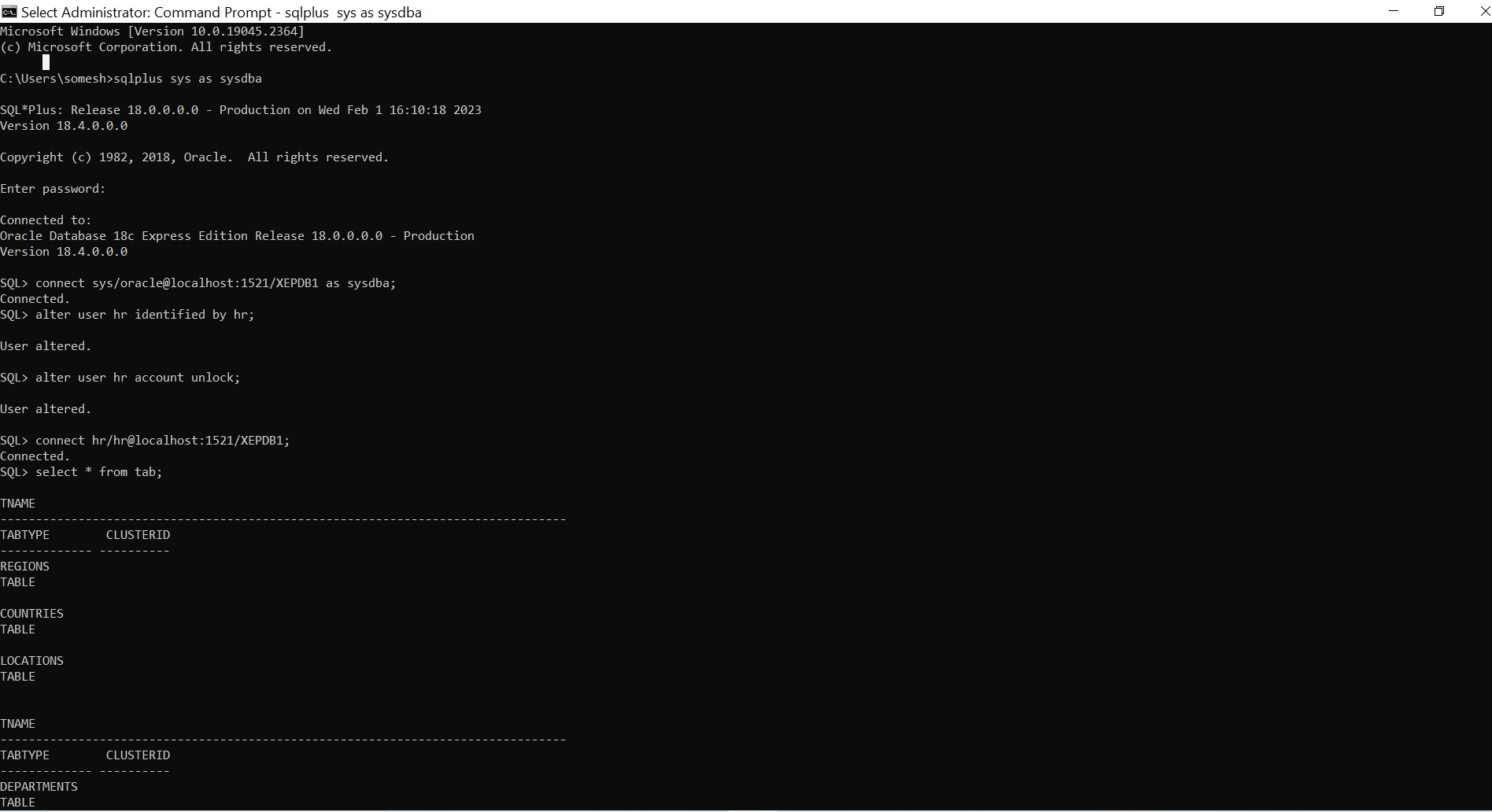


1. Wait for installation to be completed.
2. When the installation is completed, note down the connection information.



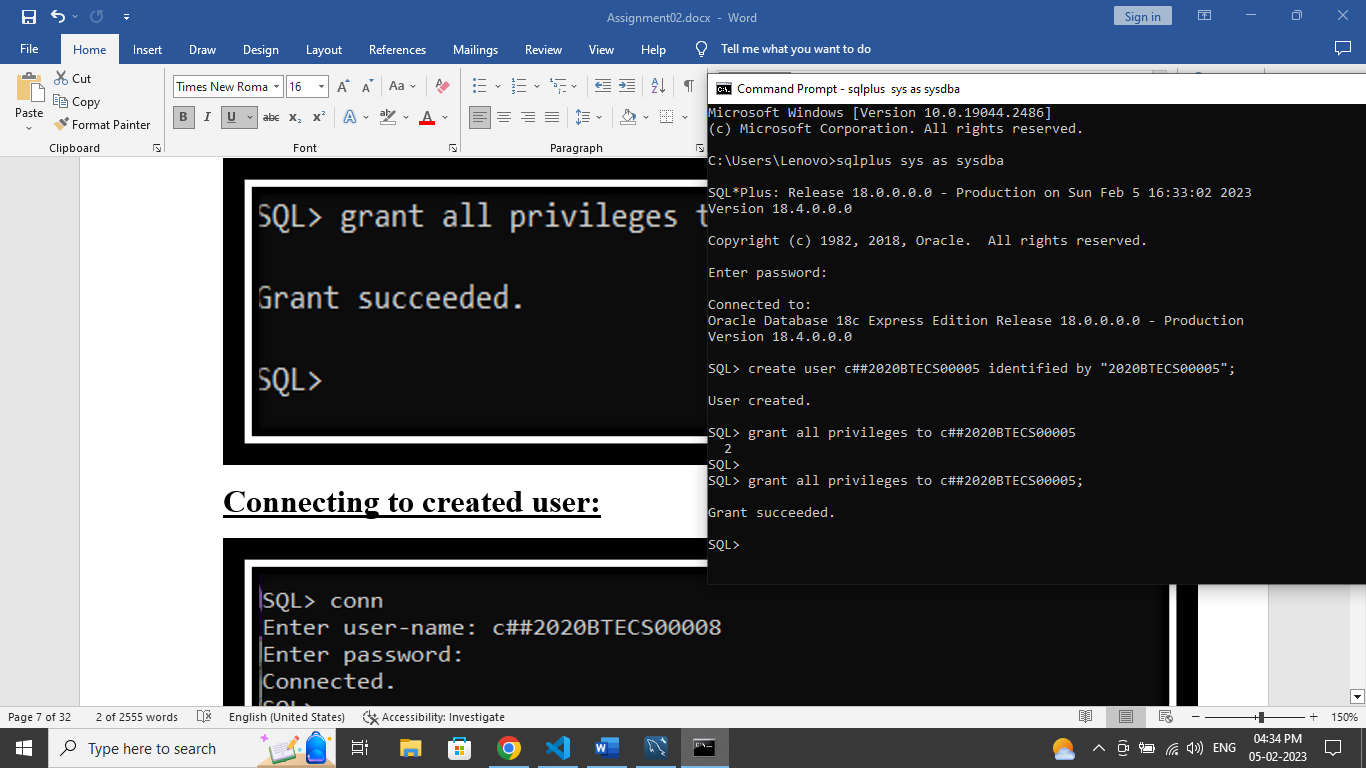
1. Hence, the Oracle Server (18c Express Edition) is installed successfully.

**Testing the connectivity:**

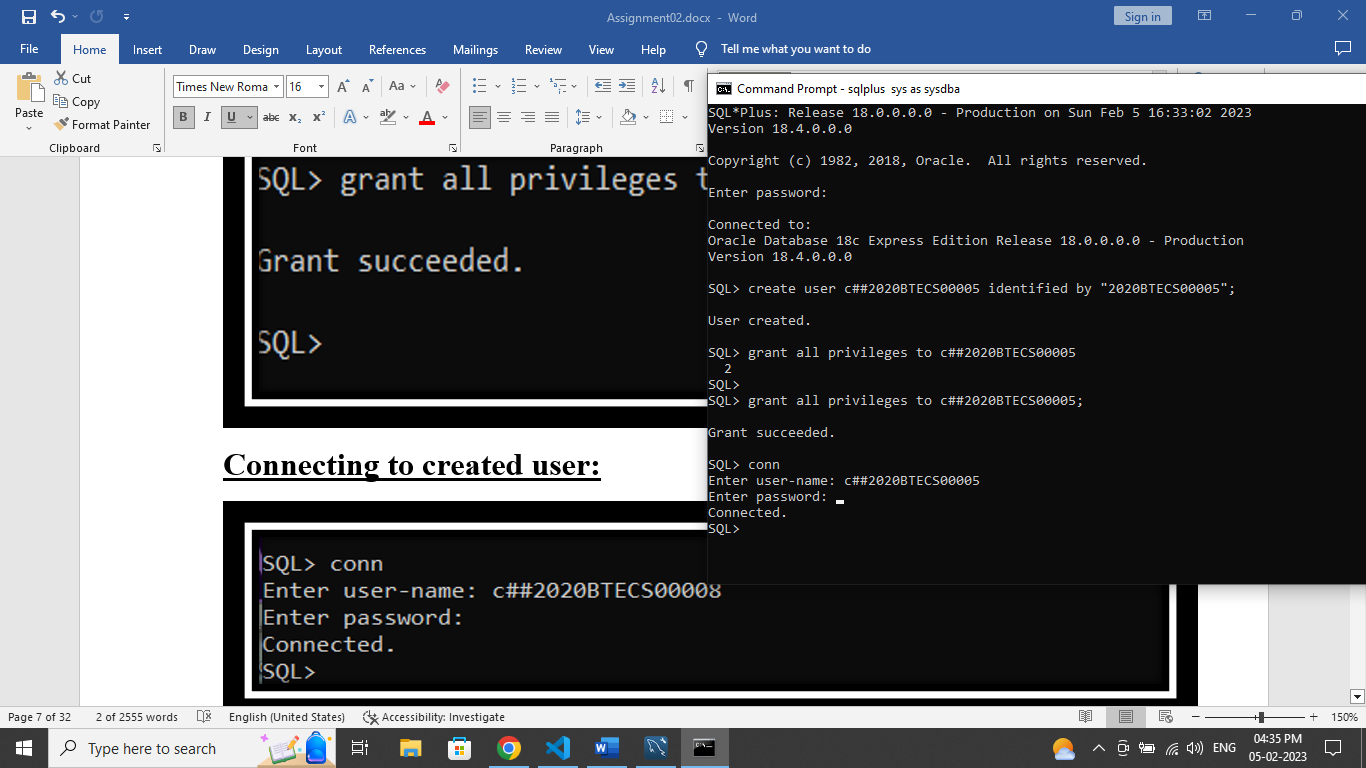


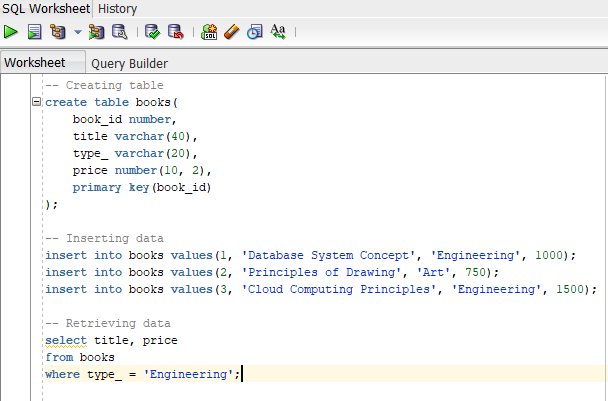


**Creating User:**



**Connecting to created user:**





**Python GUI Application:**

from tkinter import \*

from tkinter import ttk

from tkinter import simpledialog

import tkinter, tkinter.messagebox

import cx\_Oracle

# Connecting to DB

dsn\_tns = cx\_Oracle.makedsn('localhost', '1521', service\_name='XEPDB1')

conn = cx\_Oracle.connect(user='c##2020BTECS00005', password='2020BTECS00005', dsn=dsn\_tns)

c = conn.cursor()

# Initializing Window

window = Tk()

window.title("Oracle Database Connectivity") # Title of window

window.geometry('900x900') # Size of window (width X height)

window.configure(background = "orange"); # Background color of window

window.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

usr\_name = Label(window ,text = f"Connected to DB as: c##2020BTECS00005", background="orange").grid(row = 0, column = 1, pady=20)

# Getting the table names

c.execute('select table\_name from user\_tables')

DB\_NAMES = [a[0] for a in c]

variable = StringVar(window)

variable.set(DB\_NAMES[0]) # default value

selected\_tb = DB\_NAMES[0]

tb\_select = Label(window ,text = "Select the table: ", background="orange").grid(row = 1, column = 0, columnspan=1, padx=10, pady=10)

tb\_dropdown = OptionMenu(window, variable, \*DB\_NAMES).grid(row = 1, column = 0, columnspan=2, padx=15)

def confirm\_tb():

    global selected\_tb

    tkinter.messagebox.showinfo("SUCCESS", f"Table {variable.get()} is selected!")

tb\_btn = Button(window, text="Confirm", command=confirm\_tb, background="green", foreground="white", border=5).grid(row = 1, column = 1)

# CRUD Functions

# 1. View

def view\_tb():

        newWindow = Toplevel(window)

        newWindow.title("VIEW Table")

        newWindow.geometry('1500x900')

        newWindow.configure(background = "orange"); # Background color of window

        newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

        global selected\_tb

        Label(newWindow ,text = f"Viewing Table - {selected\_tb}", background="orange").grid(row = 0, column = 0, padx=10, pady=10)

        # Getting the primary key

        c.execute(f'''select a.column\_name

                       from all\_cons\_columns a

                       inner join all\_constraints c

                       on a.constraint\_name=c.constraint\_name

                       where c.table\_name='{selected\_tb}' and c.constraint\_type='P'

                        ''')

        for a in c:

            pk = a[0]

        # Getting all column names from table

        c.execute(f'''SELECT column\_name

                FROM USER\_TAB\_COLUMNS

                WHERE table\_name = '{selected\_tb}'

                ''')

        columns = [a[0] for a in c]

        tree = ttk.Treeview(newWindow, height=20, columns=columns, show='headings')

        tree.grid(row=1, column=0, sticky='news', padx=10, pady=10)

        # setup columns attributes

        for col in columns:

            tree.heading(col, text=col)

            tree.column(col, width=100, anchor=tkinter.CENTER)

        # populate data to treeview

        c.execute(f'SELECT \* FROM {selected\_tb} ORDER BY {pk}')

        for a in c:

            tree.insert('', 'end', value=a)

        # scrollbar

        sb = tkinter.Scrollbar(newWindow, orient=tkinter.VERTICAL, command=tree.yview)

        sb.grid(row=1, column=1, sticky='ns', padx=0, pady=10)

        tree.config(yscrollcommand=sb.set)

        sbx = tkinter.Scrollbar(newWindow, orient=tkinter.HORIZONTAL, command=tree.xview)

        sbx.grid(row=2, column=0, sticky='ew', padx=10, pady=0)

        tree.config(xscrollcommand=sbx.set)

# 2. Insert

def insert\_tb():

    newWindow = Toplevel(window)

    newWindow.title("INSERT into Table")

    newWindow.geometry('900x900')

    newWindow.configure(background = "orange"); # Background color of window

    newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

    global selected\_tb

    Label(newWindow ,text = f"Insert values in table: {selected\_tb}", background="orange").grid(row = 0, column = 0, padx=10, pady=10)

    c.execute(f'''SELECT column\_name

                FROM USER\_TAB\_COLUMNS

                WHERE table\_name = '{selected\_tb}'

                ''')

    # Getting columns names

    columns = [a[0] for a in c]

    ent\_ref = [] # For storing the Entry references

    # Populating Labels and Entries

    for ind, nm in enumerate(columns):

        Label(newWindow ,text = nm, background="orange").grid(row = ind+1, column = 0, padx=10, pady=10)

        ent = Entry(newWindow)

        ent.grid(row = ind+1,column = 1)

        ent\_ref.append(ent)

    def insert\_val():

        val = []

        is\_empty = False

        # Getting value from each entry field

        for r in ent\_ref:

            if len(r.get()) > 0:

                val.append(r.get())

            else:

                tkinter.messagebox.showerror("ERROR", "All the fields are required!")

                is\_empty = True

                break

        # Checking if all fields are filled, before inserting

        if not is\_empty:

            v = []

            # Typecasting values (int, float & string)

            for x in val:

                try:

                    v.append(int(x))

                except ValueError:

                    try:

                        v.append(float(x))

                    except ValueError:

                        v.append(x)

            # Inserting values

            s = f'insert into {selected\_tb}('+','.join(['?']\*len(v))+')'+ ' values('+','.join([':?']\*len(v))+')'

            for a in columns:

                s = s.replace('?', a, 1)

            for a in columns:

                s = s.replace('?', a, 1)

            try:

                c.execute(s, v)

                conn.commit()

                for r in ent\_ref:

                    r.delete(0, END)

                tkinter.messagebox.showinfo("SUCCESS", "Values inserted into table successfully!")

            except Exception as e:

                tkinter.messagebox.showerror("ERROR", e)

    Button(newWindow, text="Insert Values", command=insert\_val, background="green", foreground="white").grid(row = ind+2, column = 1, pady=20, sticky='ew')

# 3. Update

def update\_tb():

    global selected\_tb

    try:

        c.execute(f'''select a.column\_name

                       from all\_cons\_columns a

                       inner join all\_constraints c

                       on a.constraint\_name=c.constraint\_name

                       where c.table\_name='{selected\_tb}' and c.constraint\_type='P'

                        ''')

        for a in c:

            pk = a[0]

        id = simpledialog.askinteger(title="UPDATE", prompt="Enter the ID to be updated: ")

        if id is not None:

            c.execute(f'select \* from {selected\_tb} where {pk}={id}')

            if len(c.fetchall()) == 0:

                tkinter.messagebox.showerror("ERROR", "No record was found with the given ID !")

            else:

                newWindow = Toplevel(window)

                newWindow.title("UPDATE Table")

                newWindow.geometry('900x900')

                newWindow.configure(background = "orange"); # Background color of window

                newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

                Label(newWindow ,text = "Update values in table:", background="orange").grid(row = 0, column = 0, padx=10, pady=10)

                c.execute(f'''SELECT column\_name

                            FROM USER\_TAB\_COLUMNS

                            WHERE table\_name = '{selected\_tb}'

                            ''')

                columns = [a[0] for a in c]

                ent\_ref = []

                c.execute(f'select \* from {selected\_tb} where {pk}={id}')

                val = []

                for a in c:

                    val.append(a)

                val = [str(item) for t in val for item in t]

                for ind, nm in enumerate(columns):

                    Label(newWindow ,text = nm, background="orange").grid(row = ind+1, column = 0, padx=10, pady=10)

                    ent = Entry(newWindow)

                    ent.grid(row = ind+1,column = 1)

                    ent.insert(0, val[ind])

                    ent\_ref.append(ent)

                def update\_val():

                    upd\_val = []

                    is\_empty = False

                    for r in ent\_ref:

                        if len(r.get()) > 0:

                            upd\_val.append(r.get())

                        else:

                            tkinter.messagebox.showerror("ERROR", "All the fields are required!")

                            is\_empty = True

                            break

                    if not is\_empty:

                        v = []

                        for x in upd\_val:

                            try:

                                v.append(int(x))

                            except ValueError:

                                try:

                                    v.append(float(x))

                                except ValueError:

                                    v.append(x)

                        s = f'update {selected\_tb} set '+','.join(['? = :?']\*len(v))+f' where {pk}={id}'

                        for a in columns:

                            s = s.replace('?', a, 2)

                        try:

                            c.execute(s, v)

                            conn.commit()

                            newWindow.destroy()

                            tkinter.messagebox.showinfo("SUCCESS", "Values updated successfully!")

                        except Exception as e:

                            tkinter.messagebox.showerror("ERROR", e)

                Button(newWindow, text="Update Values", command=update\_val, background="blue", foreground="white").grid(row = ind+2, column = 1, pady=20, sticky='ew')

    except Exception as e:

        tkinter.messagebox.showerror("ERROR", e)

# 4. Delete

def delete\_tb():

    global selected\_tb

    try:

        c.execute(f'''select a.column\_name

                       from all\_cons\_columns a

                       inner join all\_constraints c

                       on a.constraint\_name=c.constraint\_name

                       where c.table\_name='{selected\_tb}' and c.constraint\_type='P'

                        ''')

        for a in c:

            pk = a[0]

        id = simpledialog.askinteger(title="DELETE", prompt="Enter the ID to be deleted: ")

        if id is not None:

            c.execute(f'delete from {selected\_tb} where {pk}={id}')

            if c.rowcount == 0:

                tkinter.messagebox.showerror("ERROR", "Cannot DELETE!\nNo record was found with the given ID !")

            else:

                conn.commit()

                tkinter.messagebox.showinfo("SUCCESS", "Deleted record from table successfully!")

    except Exception as e:

        tkinter.messagebox.showerror("ERROR", e)

# CRUD operation buttons

if selected\_tb is not None:

    Label(window ,text = "Operations on selected table:", background="orange", font='Helvetica 18 bold').grid(row = 3, column = 0, padx=10, pady=60)

    view\_btn = Button(window, text="View", command=view\_tb, background="#9629ff", foreground="white", border=3).grid(row = 4, column = 0)

    insert\_btn = Button(window, text="Insert", command=insert\_tb, background="green", foreground="white", border=3).grid(row = 4, column = 1, sticky='w', columnspan=1)

    update\_btn = Button(window, text="Update", command=update\_tb, background="blue", foreground="white", border=3).grid(row = 4, column = 1, columnspan=2)

    delete\_btn = Button(window, text="Delete", command=delete\_tb, background="red", foreground="white", border=3).grid(row = 4, column = 2)

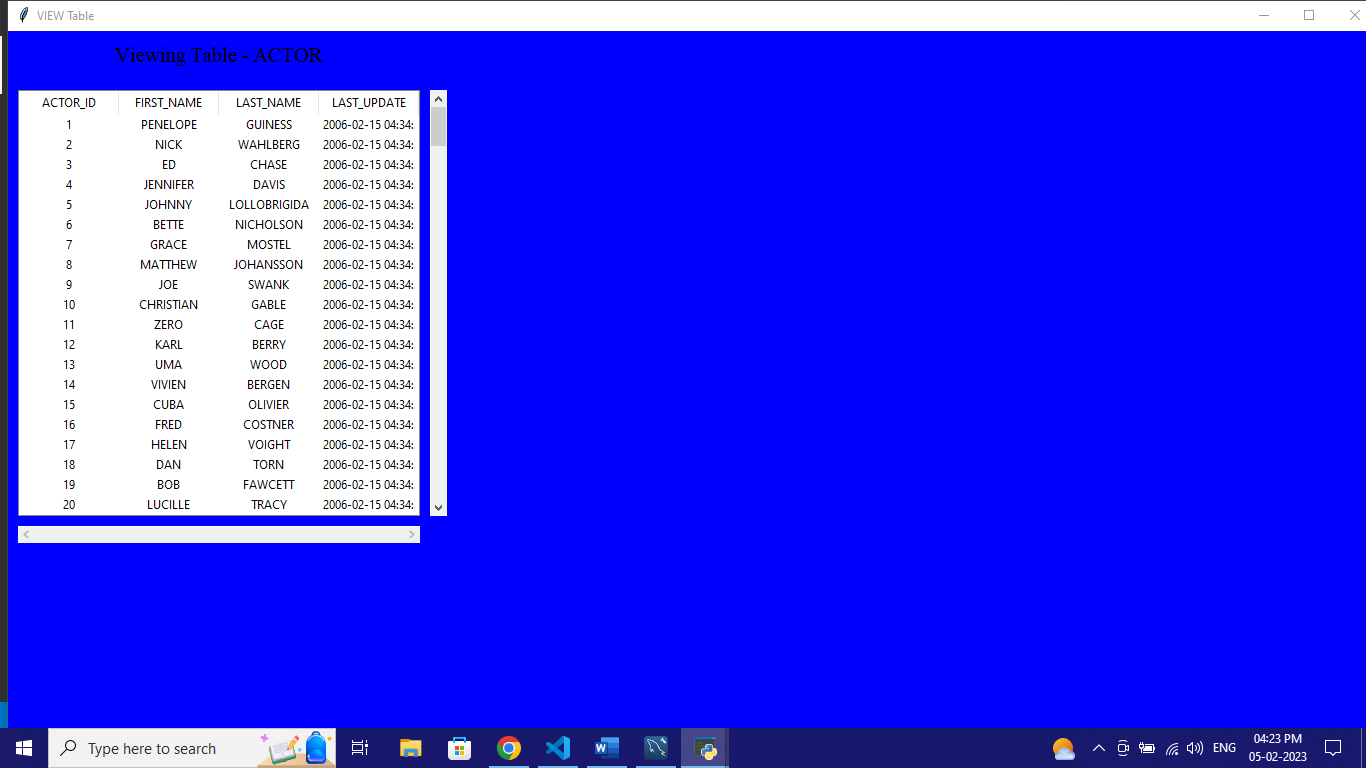
window.mainloop() # window remains until user closes it

conn.close() # Closing the connection to database

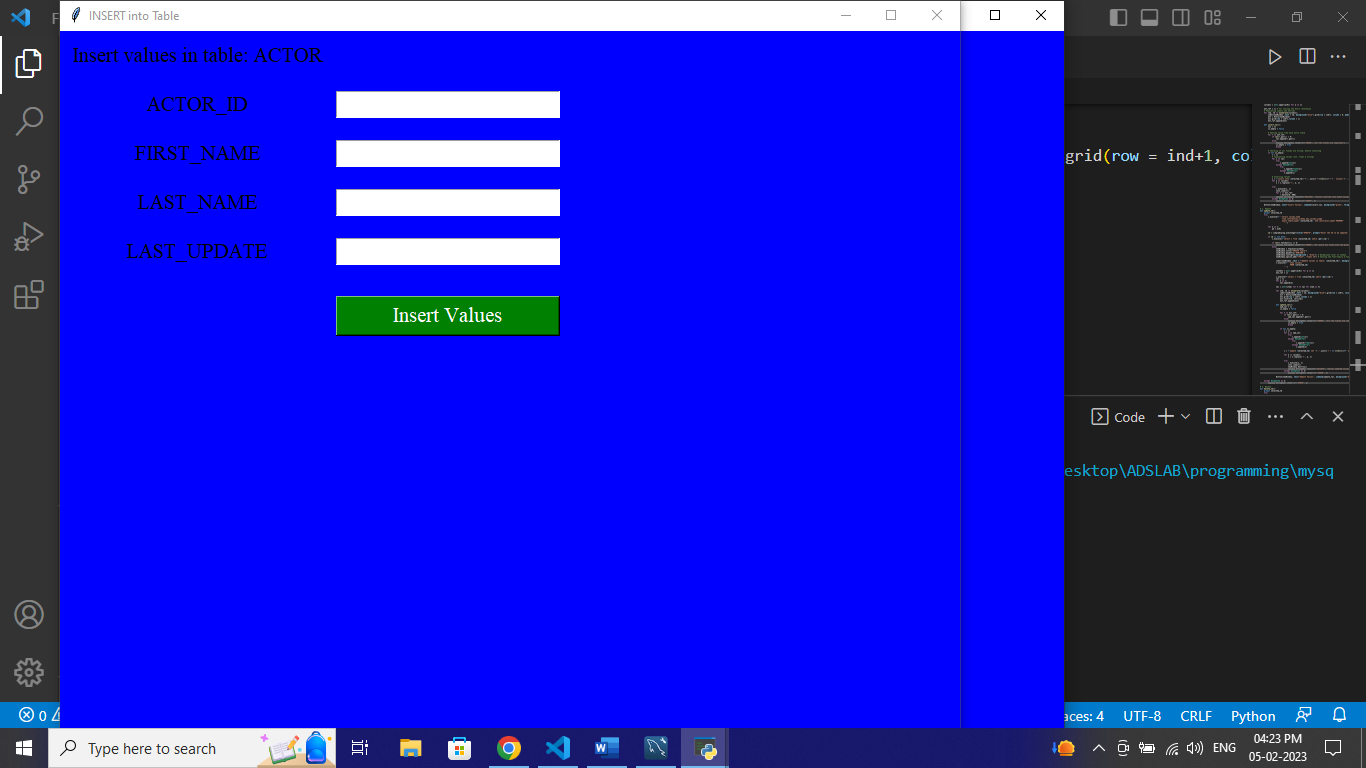
* **Result:**

****

1. **View:**

****

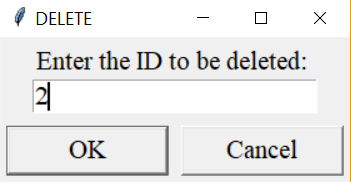
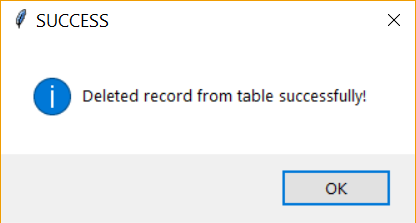
1. **Insert:**

****

1. **Update:**

****

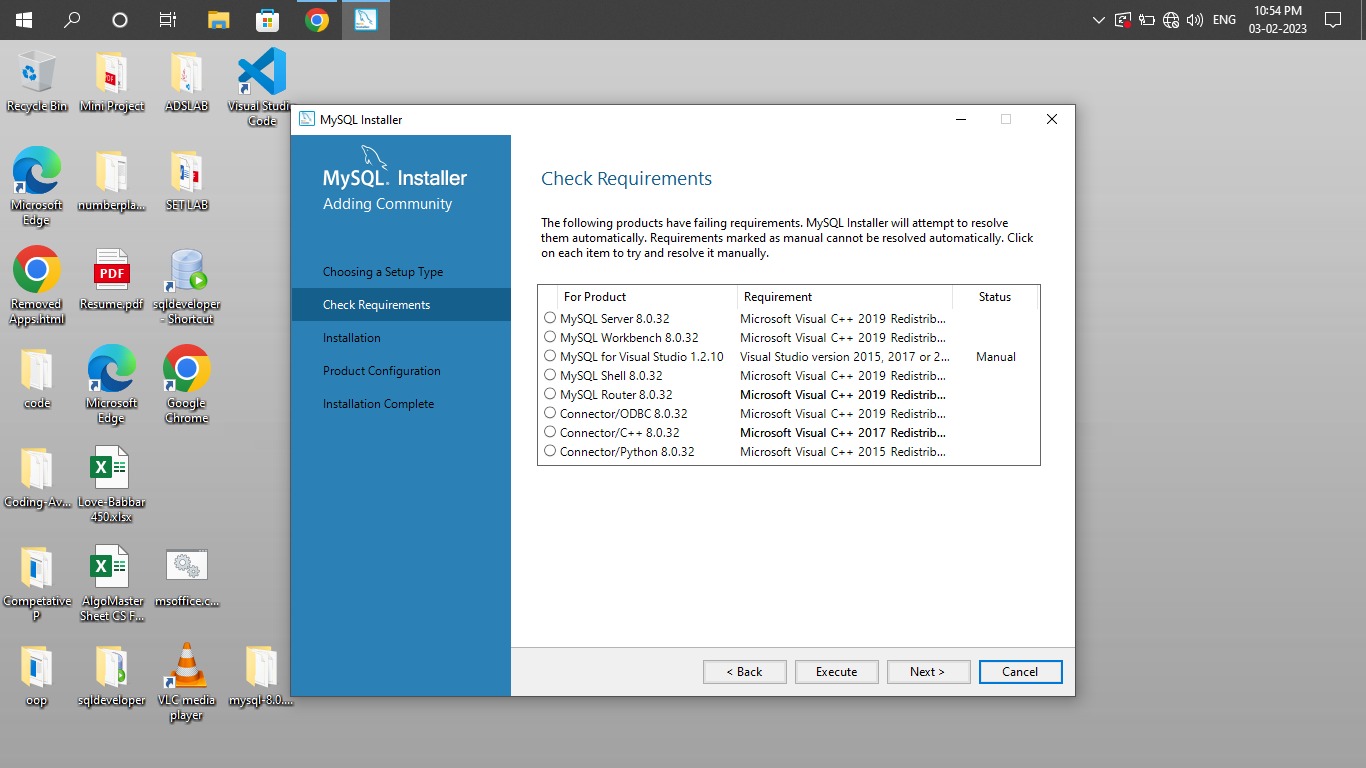
1. **Delete:**



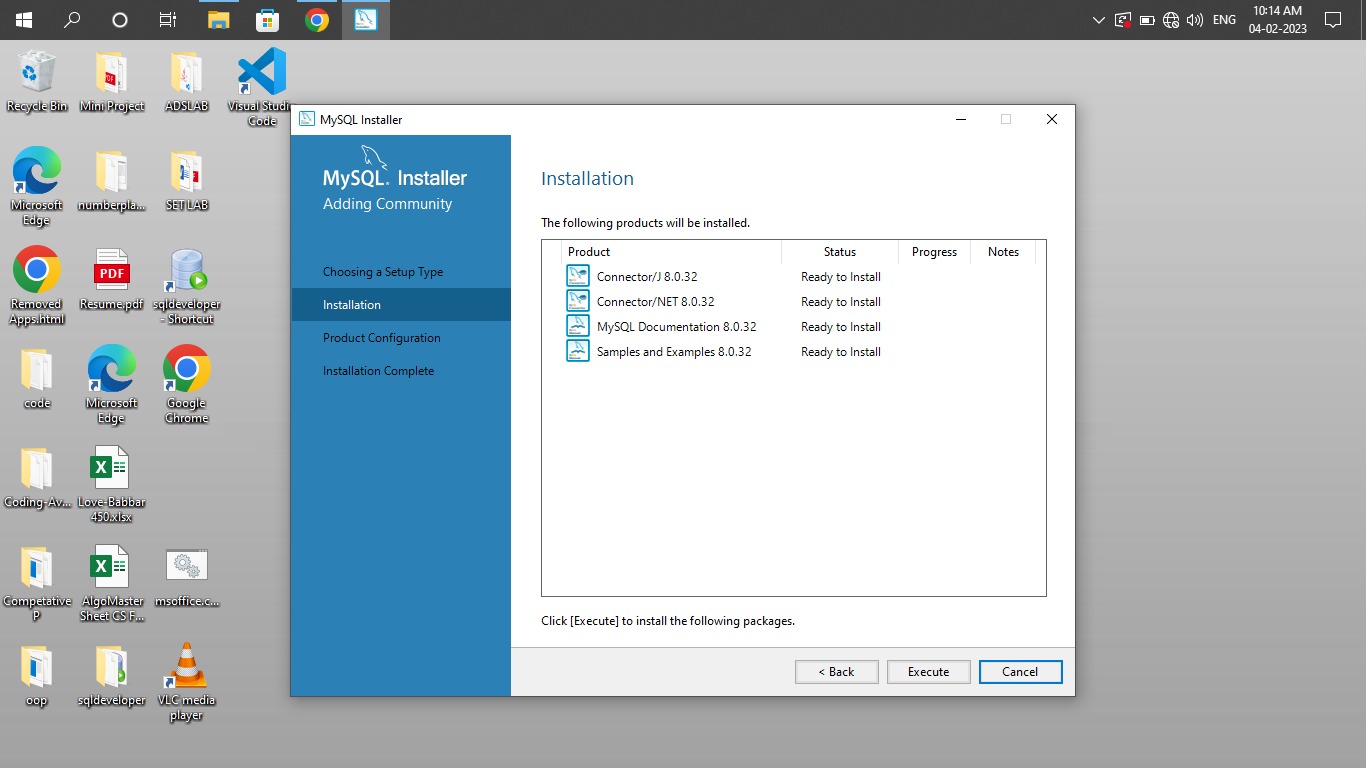
* ***MySQL***

**MySQL installation:**

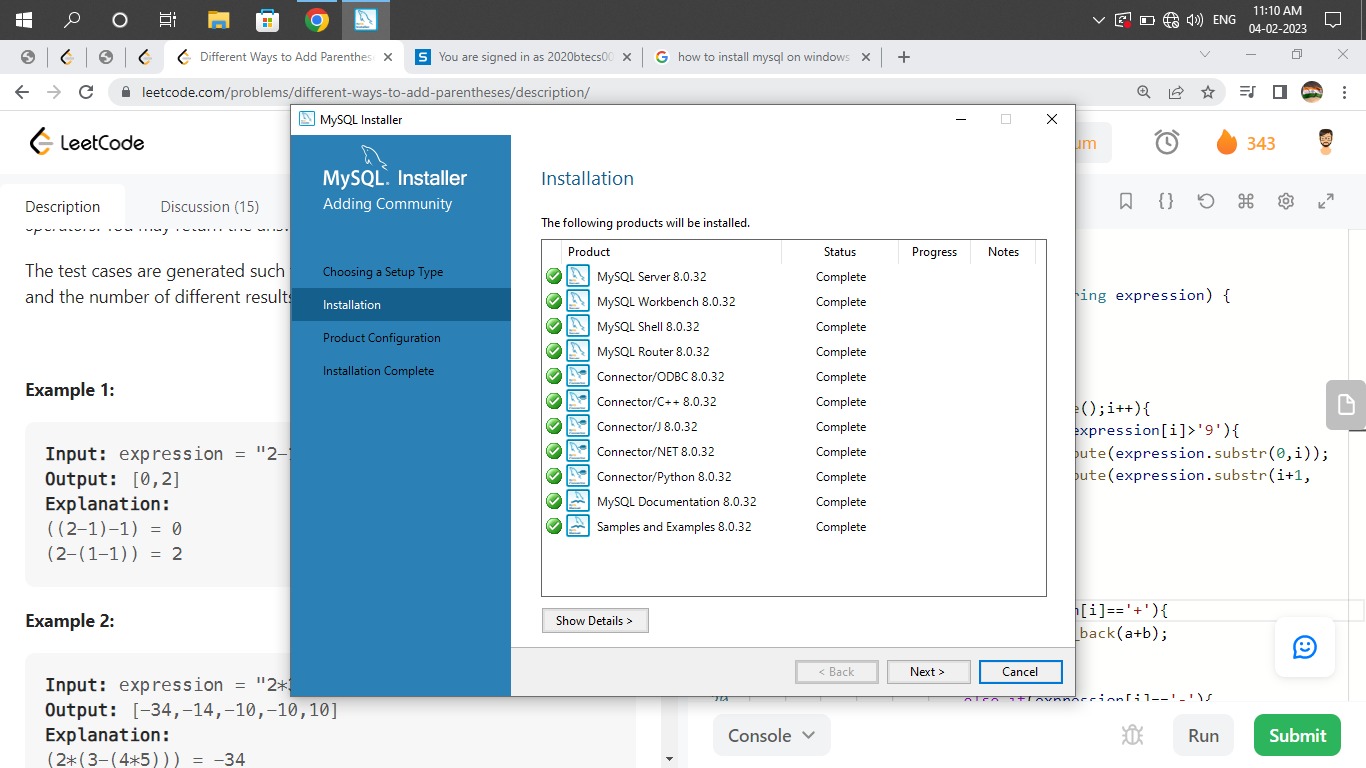
1. Install the setup for MySQL from [MySQL website](https://dev.mysql.com/downloads/mysql/). Run the setup.exe. Choose installation type as ‘Full’.
2. It will check whether your computer satisfies the necessary requirements. Install the necessary requirements (select requirements and click Execute) and click next.



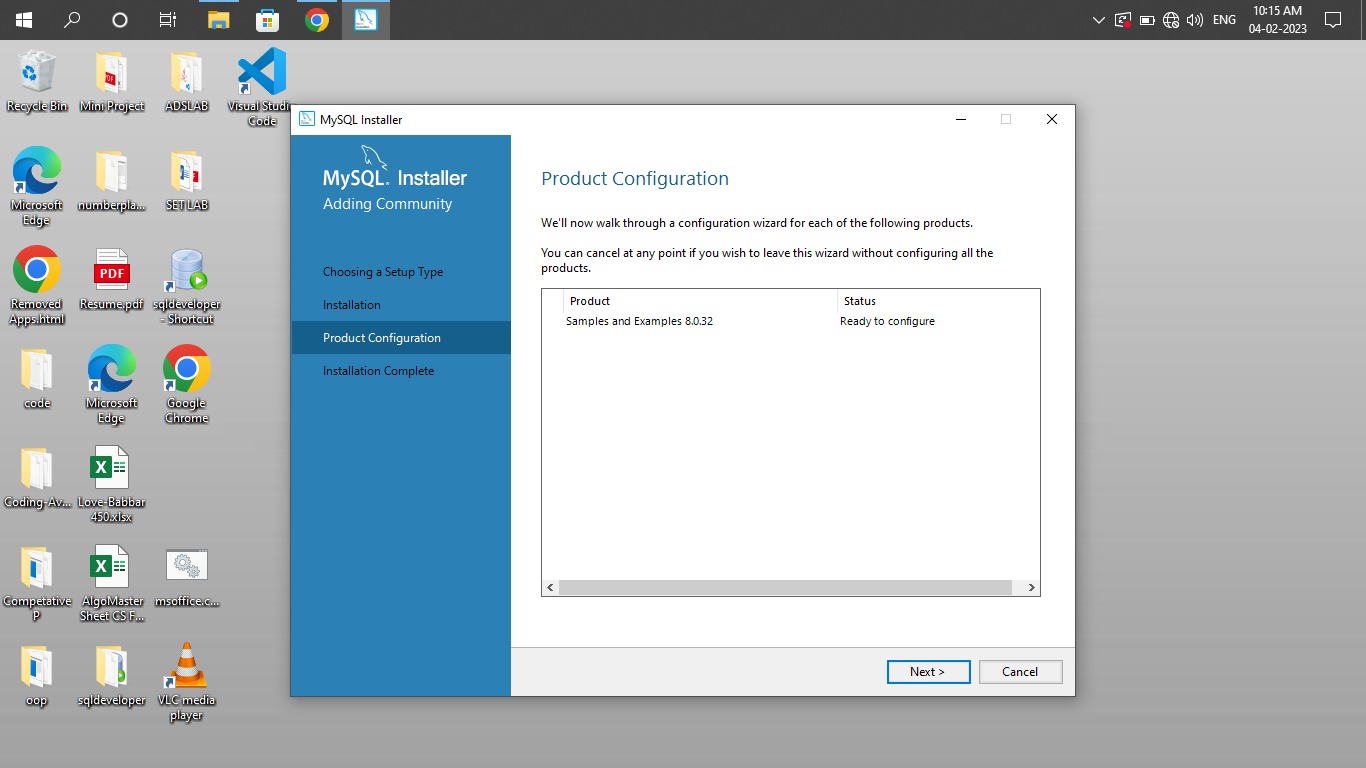
3) Click execute to install the required setups.



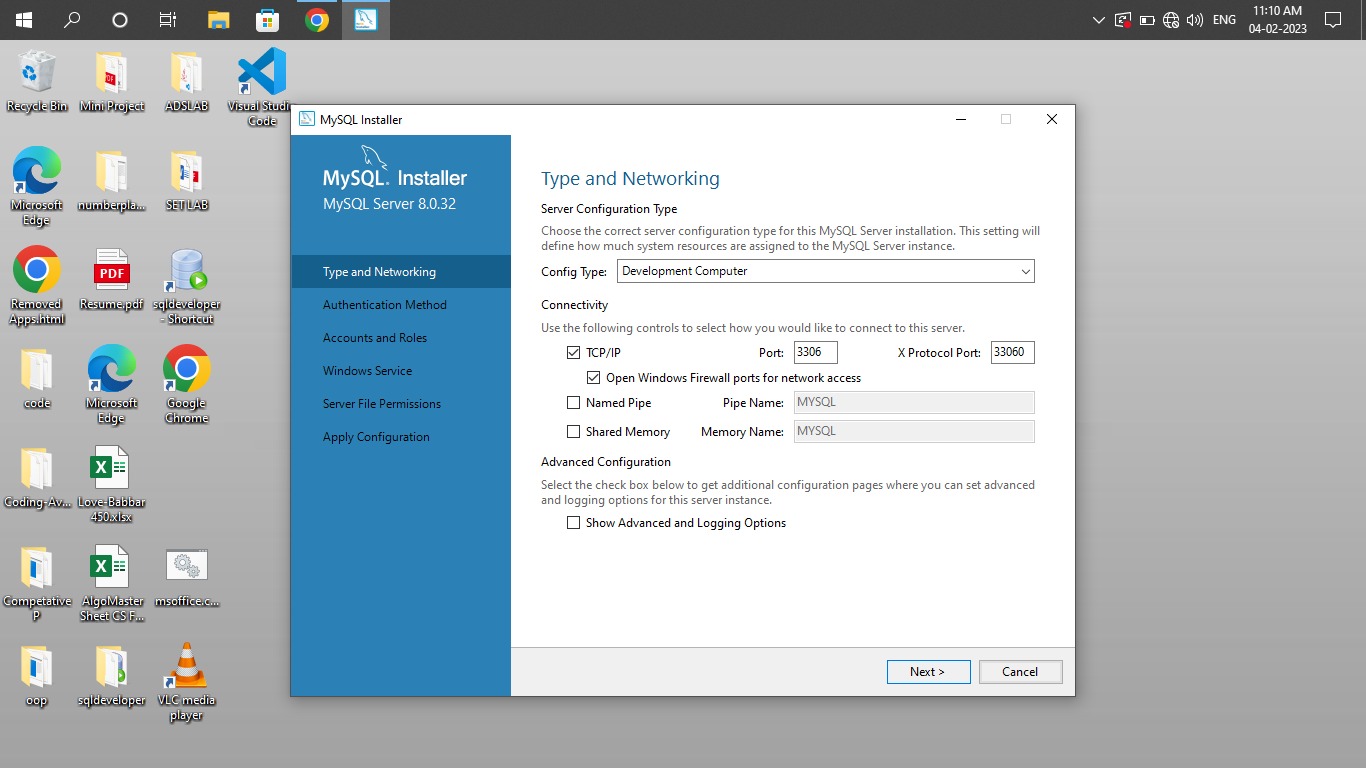
4) Wait until all the files are installed.



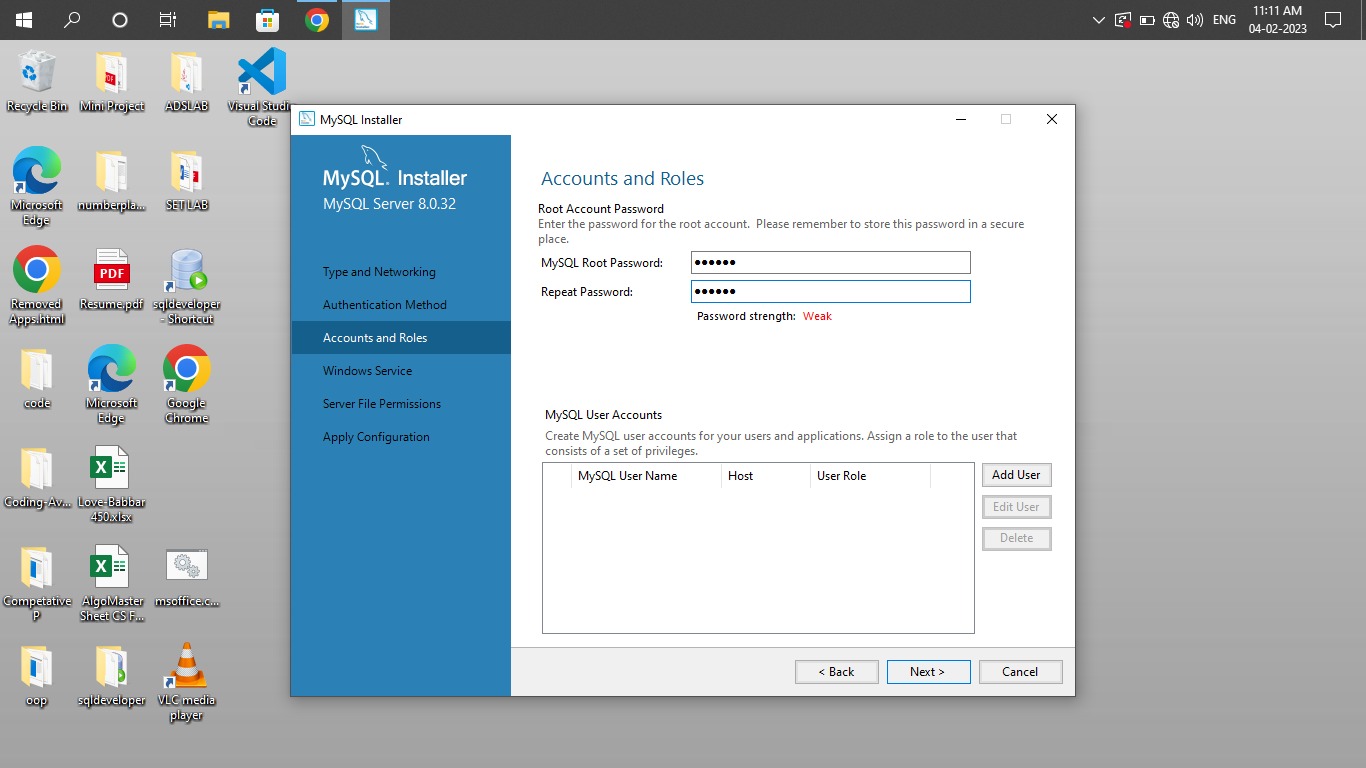
5) Configure the product by clicking next.



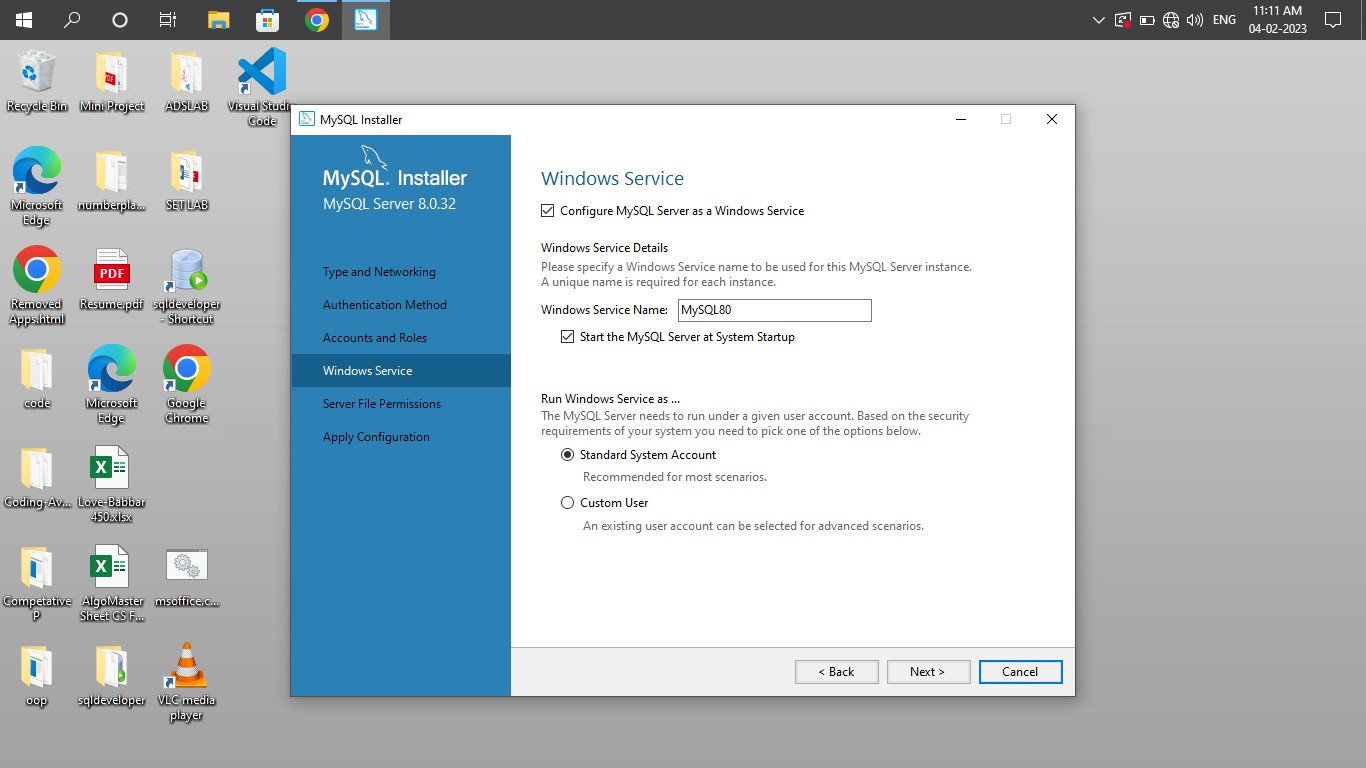
6) Configure the network and click next.

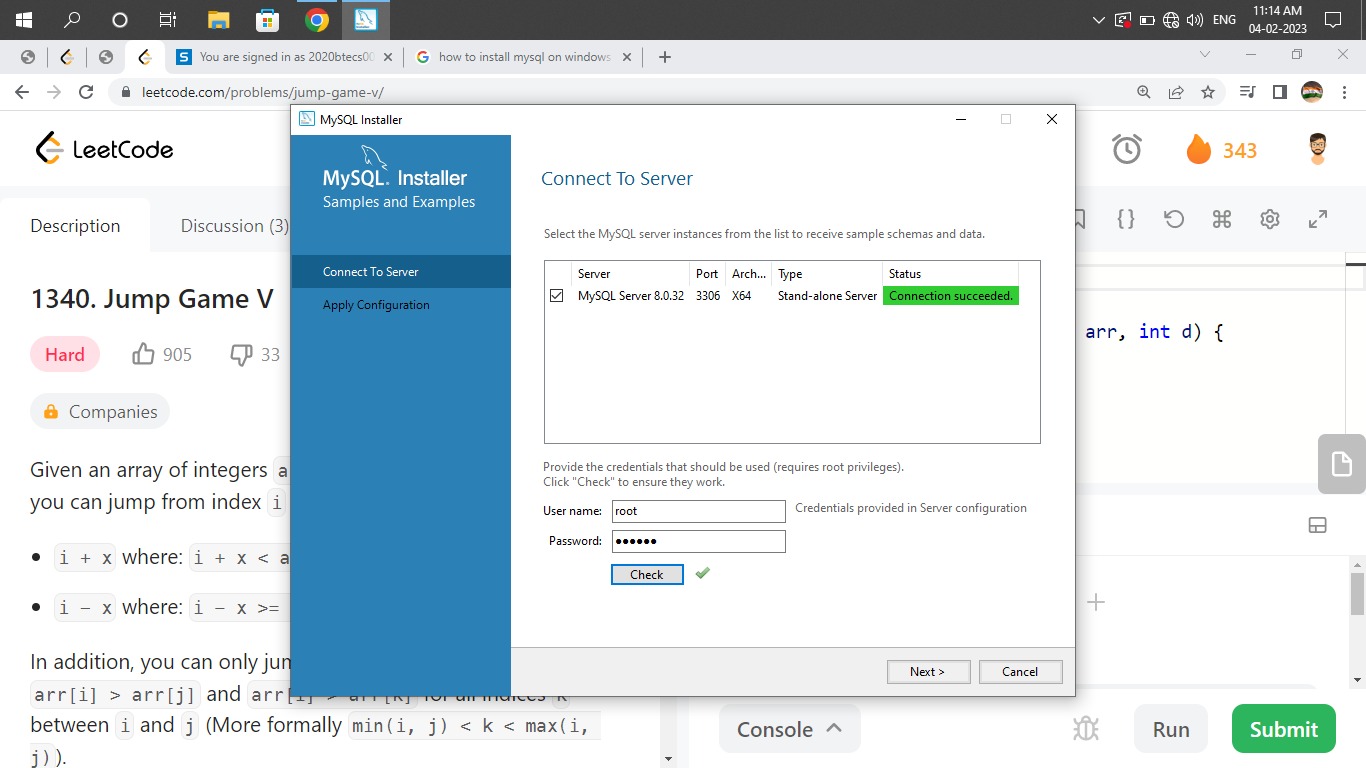


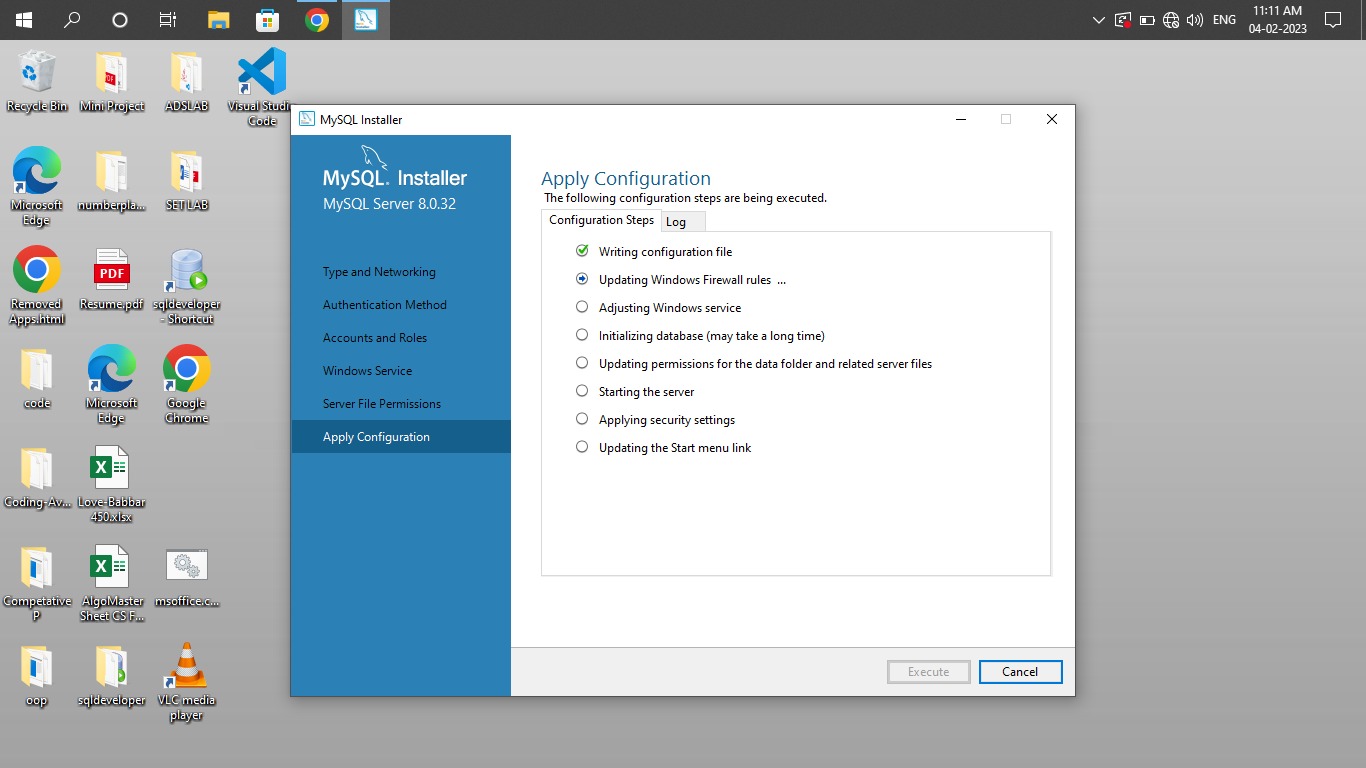
7) Set the password for ‘root’ (admin) and click next.



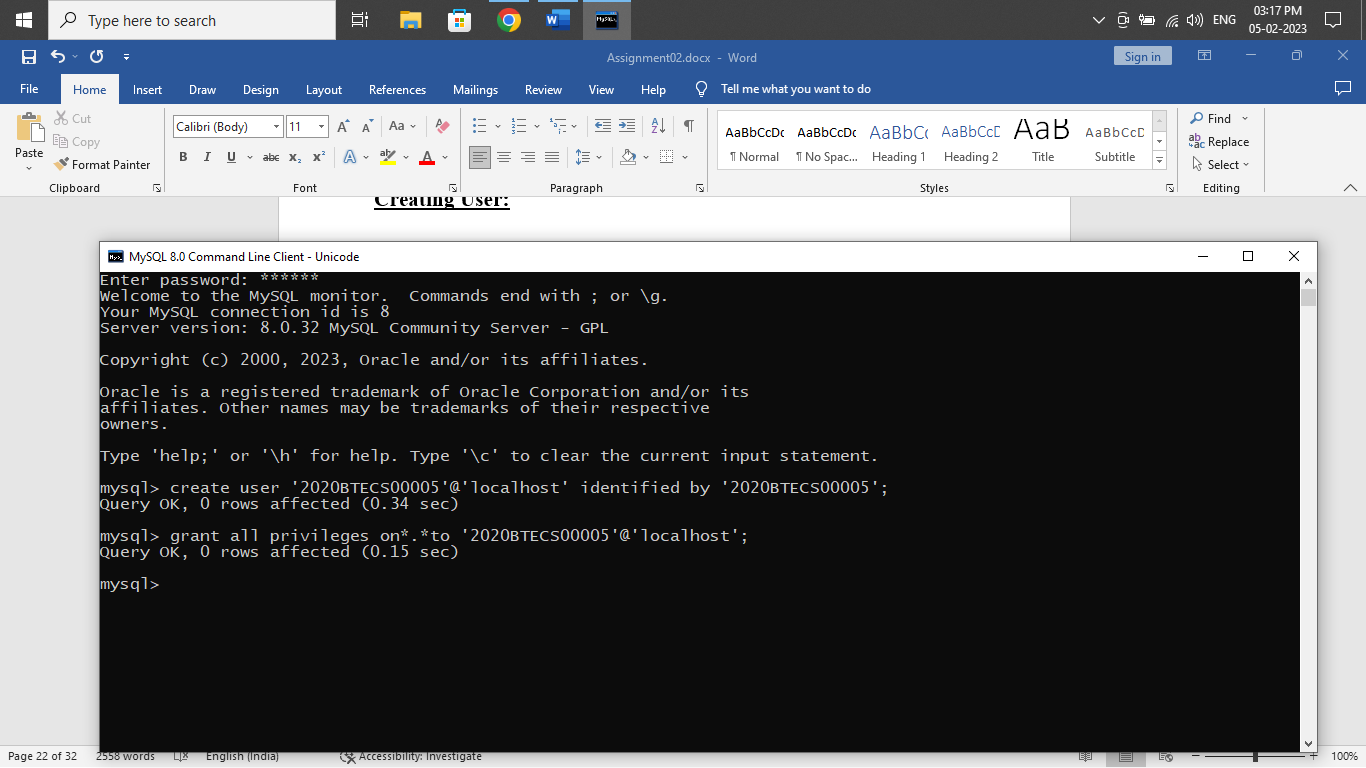
1. Configure the windows service to start. Click next.



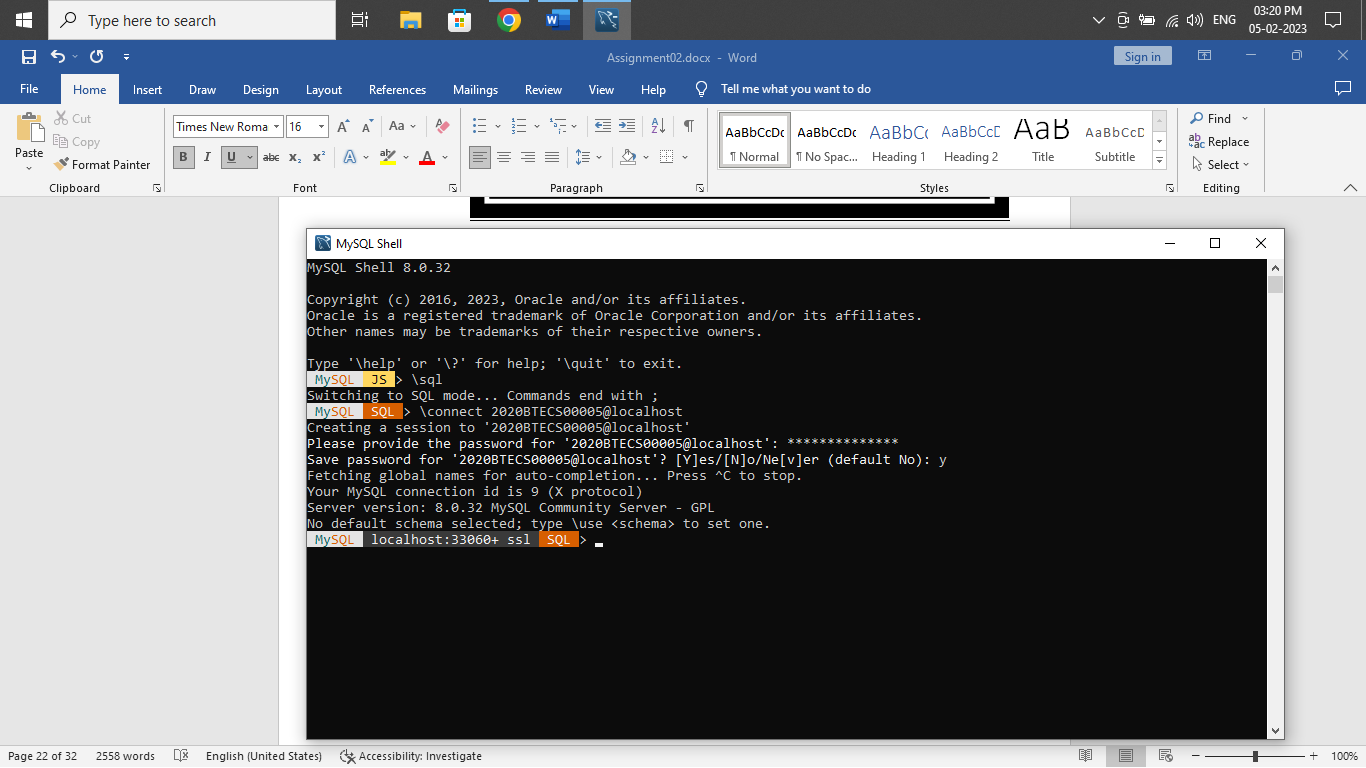
1. Connect to server by logging in with the created credentials. Check the connectivity. 
2. Apply the configuration by clicking on execute. Wait for all changes to be applied. MySQL is installed successfully.



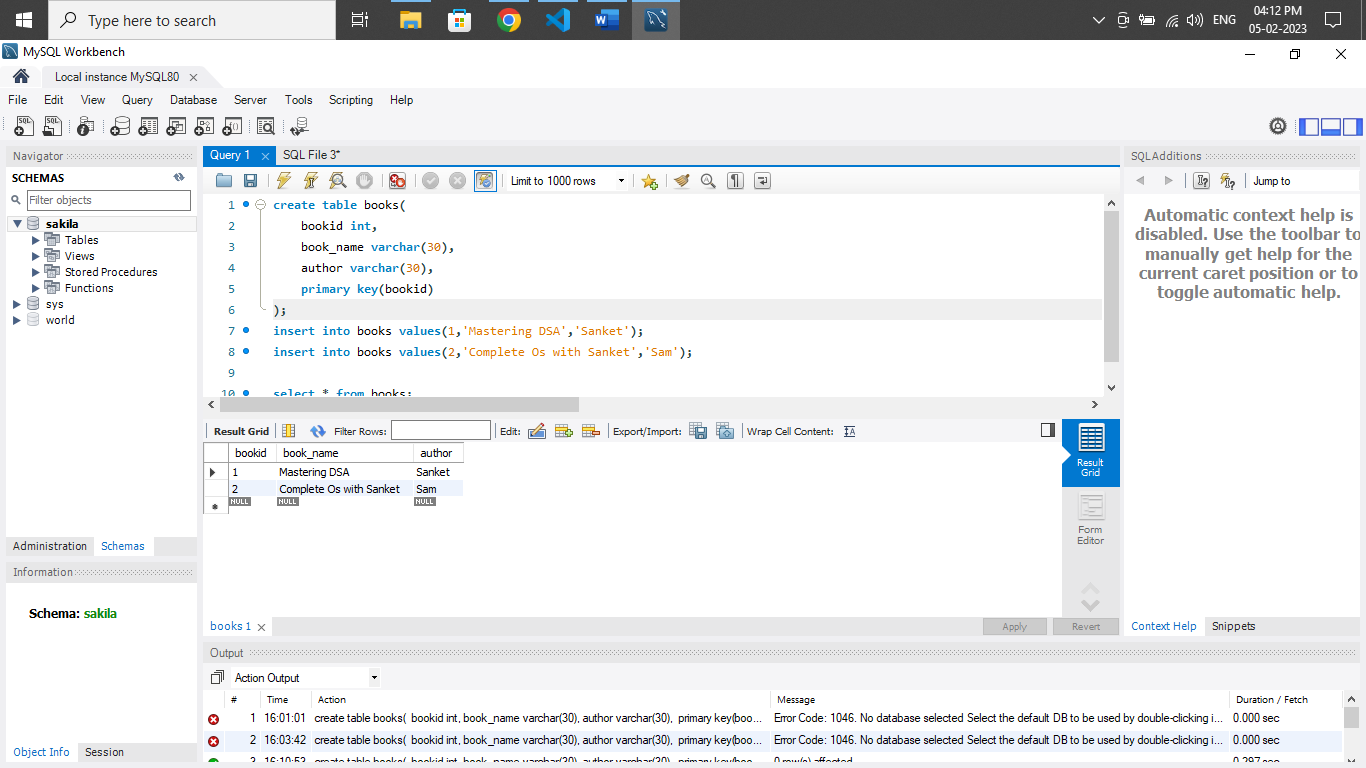
**Creating User:**



**Connecting to created user:**



**Creating Sample Tables:**



**Python GUI Application:**

from tkinter import \*

from tkinter import ttk

from tkinter import simpledialog

import tkinter, tkinter.messagebox

import mysql.connector as connector

# Connecting to DB

conn = connector.connect(host='localhost', user='2020BTECS00005', password='2020BTECS00005', database='sakila')

c = conn.cursor()

# Initializing Window

window = Tk()

window.title("MySQL Database Connectivity") # Title of window

window.geometry('900x900') # Size of window (width X height)

window.configure(background = "blue"); # Background color of window

window.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

usr\_name = Label(window ,text = f"Connected to DB as: 2020BTECS00005", background="blue").grid(row = 0, column = 1, pady=20)

# Getting the table names

c.execute('show tables')

DB\_NAMES = [str.upper(a[0]) for a in c]

variable = StringVar(window)

variable.set(DB\_NAMES[0]) # default value

selected\_tb = DB\_NAMES[0]

tb\_select = Label(window ,text = "Select the table: ", background="blue").grid(row = 1, column = 0, columnspan=1, padx=10, pady=10)

tb\_dropdown = OptionMenu(window, variable, \*DB\_NAMES).grid(row = 1, column = 0, columnspan=2, padx=15)

def confirm\_tb():

    global selected\_tb

    selected\_tb = variable.get()

    tkinter.messagebox.showinfo("SUCCESS", f"Table {selected\_tb} is selected!")

tb\_btn = Button(window, text="Confirm", command=confirm\_tb, background="green", foreground="white", border=5).grid(row = 1, column = 1)

# CRUD Functions

# 1. View

def view\_tb():

        newWindow = Toplevel(window)

        newWindow.title("VIEW Table")

        newWindow.geometry('1500x900')

        newWindow.configure(background = "blue"); # Background color of window

        newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

        global selected\_tb

        Label(newWindow ,text = f"Viewing Table - {selected\_tb}", background="blue").grid(row = 0, column = 0, padx=10, pady=10)

        # Getting the primary key

        c.execute(f'''select column\_name

                      from information\_schema.key\_column\_usage

                      where table\_name='{selected\_tb}' and constraint\_name='PRIMARY'

                        ''')

        for a in c:

            pk = a[0]

        # Getting all column names from table

        c.execute(f'''show columns

                      FROM {selected\_tb}

                ''')

        columns = [str.upper(a[0]) for a in c]

        tree = ttk.Treeview(newWindow, height=20, columns=columns, show='headings')

        tree.grid(row=1, column=0, sticky='news', padx=10, pady=10)

        # setup columns attributes

        for col in columns:

            tree.heading(col, text=col)

            tree.column(col, width=100, anchor=tkinter.CENTER)

        # populate data to treeview

        c.execute(f'SELECT \* FROM {selected\_tb} ORDER BY {pk}')

        for a in c:

            tree.insert('', 'end', value=a)

        # scrollbar

        sb = tkinter.Scrollbar(newWindow, orient=tkinter.VERTICAL, command=tree.yview)

        sb.grid(row=1, column=1, sticky='ns', padx=0, pady=10)

        tree.config(yscrollcommand=sb.set)

        sbx = tkinter.Scrollbar(newWindow, orient=tkinter.HORIZONTAL, command=tree.xview)

        sbx.grid(row=2, column=0, sticky='ew', padx=10, pady=0)

        tree.config(xscrollcommand=sbx.set)

# 2. Insert

def insert\_tb():

    newWindow = Toplevel(window)

    newWindow.title("INSERT into Table")

    newWindow.geometry('900x900')

    newWindow.configure(background = "blue"); # Background color of window

    newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

    global selected\_tb

    Label(newWindow ,text = f"Insert values in table: {selected\_tb}", background="blue").grid(row = 0, column = 0, padx=10, pady=10)

    c.execute(f'''show columns

                  FROM {selected\_tb}

                ''')

    # Getting columns names

    columns = [str.upper(a[0]) for a in c]

    ent\_ref = [] # For storing the Entry references

    # Populating Labels and Entries

    for ind, nm in enumerate(columns):

        Label(newWindow ,text = nm, background="blue").grid(row = ind+1, column = 0, padx=10, pady=10)

        ent = Entry(newWindow)

        ent.grid(row = ind+1,column = 1)

        ent\_ref.append(ent)

    def insert\_val():

        val = []

        is\_empty = False

        # Getting value from each entry field

        for r in ent\_ref:

            if len(r.get()) > 0:

                val.append(r.get())

            else:

                tkinter.messagebox.showerror("ERROR", "All the fields are required!")

                is\_empty = True

                break

        # Checking if all fields are filled, before inserting

        if not is\_empty:

            v = []

            # Typecasting values (int, float & string)

            for x in val:

                try:

                    v.append(int(x))

                except ValueError:

                    try:

                        v.append(float(x))

                    except ValueError:

                        v.append(x)

            # Inserting values

            s = f'insert into {selected\_tb}('+','.join(['?']\*len(v))+')'+ ' values('+','.join(['%s']\*len(v))+')'

            for a in columns:

                s = s.replace('?', a, 1)

            try:

                c.execute(s, v)

                conn.commit()

                for r in ent\_ref:

                    r.delete(0, END)

                tkinter.messagebox.showinfo("SUCCESS", "Values inserted into table successfully!")

            except Exception as e:

                tkinter.messagebox.showerror("ERROR", e)

    Button(newWindow, text="Insert Values", command=insert\_val, background="green", foreground="white").grid(row = ind+2, column = 1, pady=20, sticky='ew')

# 3. Update

def update\_tb():

    global selected\_tb

    try:

        c.execute(f'''select column\_name

                      from information\_schema.key\_column\_usage

                      where table\_name='{selected\_tb}' and constraint\_name='PRIMARY'

                        ''')

        for a in c:

            pk = a[0]

        id = simpledialog.askinteger(title="UPDATE", prompt="Enter the ID to be updated: ")

        if id is not None:

            c.execute(f'select \* from {selected\_tb} where {pk}={id}')

            if len(c.fetchall()) == 0:

                tkinter.messagebox.showerror("ERROR", "No record was found with the given ID !")

            else:

                newWindow = Toplevel(window)

                newWindow.title("UPDATE Table")

                newWindow.geometry('900x900')

                newWindow.configure(background = "blue"); # Background color of window

                newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

                Label(newWindow ,text = f"Update values in table: {selected\_tb}", background="blue").grid(row = 0, column = 0, padx=10, pady=10)

                c.execute(f'''show columns

                              FROM {selected\_tb}

                        ''')

                columns = [str.upper(a[0]) for a in c]

                ent\_ref = []

                c.execute(f'select \* from {selected\_tb} where {pk}={id}')

                val = []

                for a in c:

                    val.append(a)

                val = [str(item) for t in val for item in t]

                for ind, nm in enumerate(columns):

                    Label(newWindow ,text = nm, background="blue").grid(row = ind+1, column = 0, padx=10, pady=10)

                    ent = Entry(newWindow)

                    ent.grid(row = ind+1,column = 1)

                    ent.insert(0, val[ind])

                    ent\_ref.append(ent)

                def update\_val():

                    upd\_val = []

                    is\_empty = False

                    for r in ent\_ref:

                        if len(r.get()) > 0:

                            upd\_val.append(r.get())

                        else:

                            tkinter.messagebox.showerror("ERROR", "All the fields are required!")

                            is\_empty = True

                            break

                    if not is\_empty:

                        v = []

                        for x in upd\_val:

                            try:

                                v.append(int(x))

                            except ValueError:

                                try:

                                    v.append(float(x))

                                except ValueError:

                                    v.append(x)

                        s = f'update {selected\_tb} set '+','.join(['? = %s']\*len(v))+f' where {pk}={id}'

                        for a in columns:

                            s = s.replace('?', a, 1)

                        try:

                            c.execute(s, v)

                            conn.commit()

                            newWindow.destroy()

                            tkinter.messagebox.showinfo("SUCCESS", "Values updated successfully!")

                        except Exception as e:

                            tkinter.messagebox.showerror("ERROR", e)

                Button(newWindow, text="Update Values", command=update\_val, background="blue", foreground="white").grid(row = ind+2, column = 1, pady=20, sticky='ew')

    except Exception as e:

        tkinter.messagebox.showerror("ERROR", e)

# 4. Delete

def delete\_tb():

    global selected\_tb

    try:

        c.execute(f'''select column\_name

                      from information\_schema.key\_column\_usage

                      where table\_name='{selected\_tb}' and constraint\_name='PRIMARY'

                        ''')

        for a in c:

            pk = a[0]

        id = simpledialog.askinteger(title="DELETE", prompt="Enter the ID to be deleted: ")

        if id is not None:

            c.execute(f'delete from {selected\_tb} where {pk}={id}')

            if c.rowcount == 0:

                tkinter.messagebox.showerror("ERROR", "Cannot DELETE!\nNo record was found with the given ID !")

            else:

                conn.commit()

                tkinter.messagebox.showinfo("SUCCESS", "Deleted record from table successfully!")

    except Exception as e:

        tkinter.messagebox.showerror("ERROR", e)

# CRUD operation buttons

if selected\_tb is not None:

    Label(window ,text = "Operations on selected table:", background="blue", font='Helvetica 18 bold').grid(row = 3, column = 0, padx=10, pady=60)

    view\_btn = Button(window, text="View", command=view\_tb, background="#9629ff", foreground="white", border=3).grid(row = 4, column = 0)

    insert\_btn = Button(window, text="Insert", command=insert\_tb, background="green", foreground="white", border=3).grid(row = 4, column = 1, sticky='w', columnspan=1)

    update\_btn = Button(window, text="Update", command=update\_tb, background="blue", foreground="white", border=3).grid(row = 4, column = 1, columnspan=2)

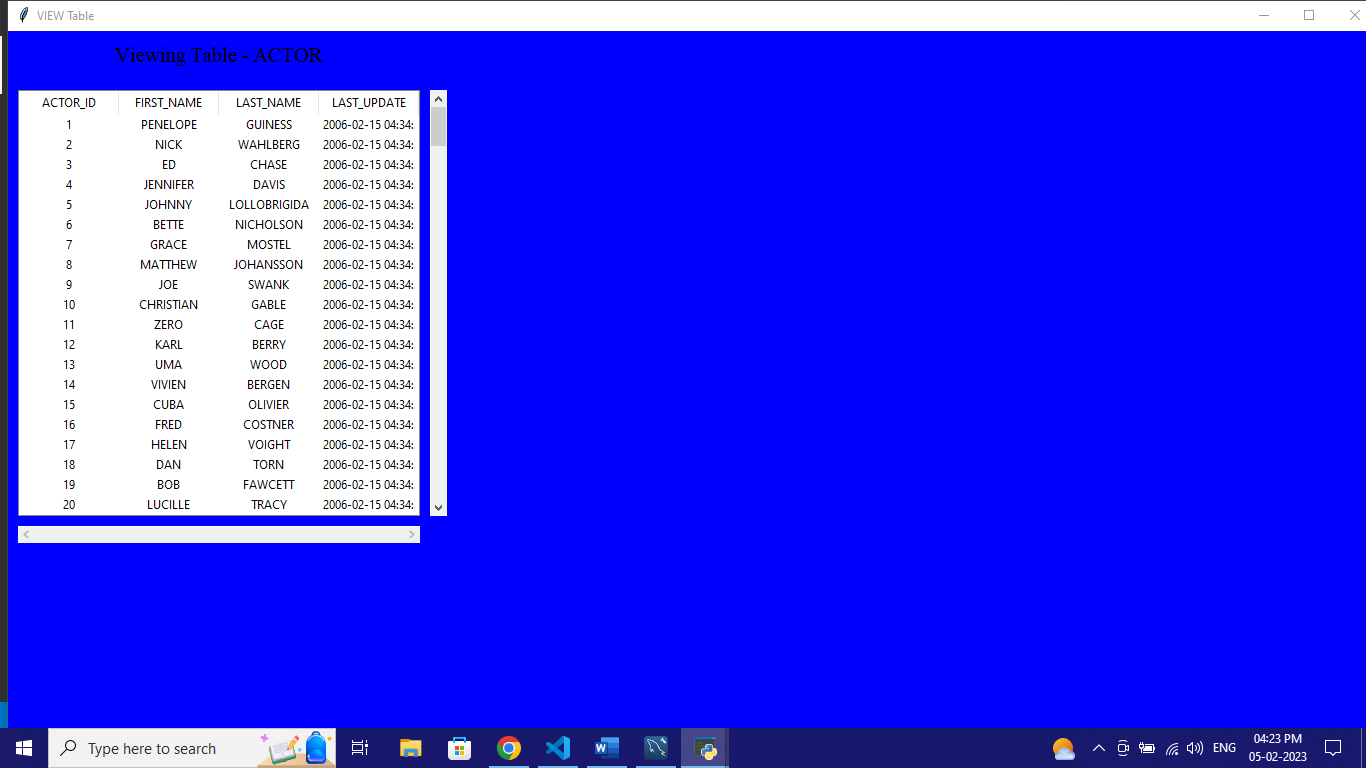
    delete\_btn = Button(window, text="Delete", command=delete\_tb, background="red", foreground="white", border=3).grid(row = 4, column = 2)

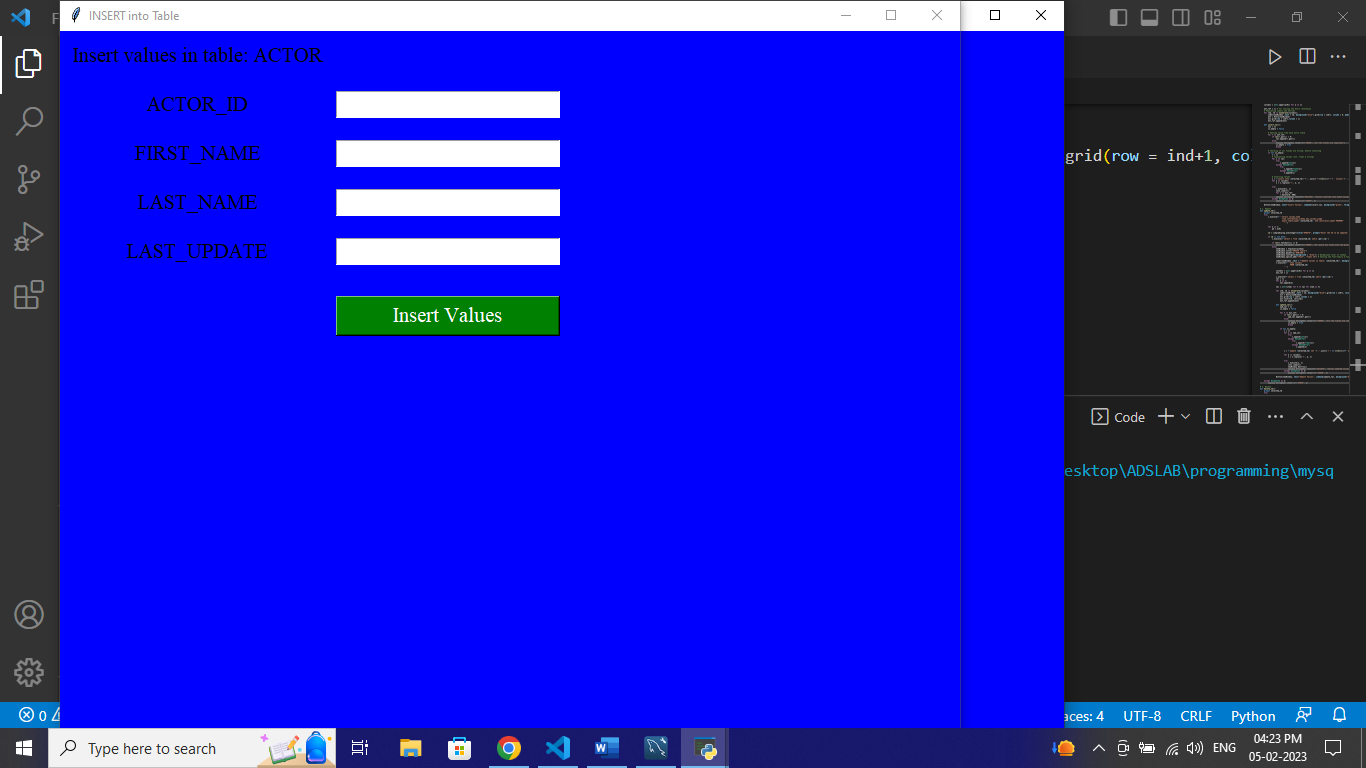
window.mainloop() # window remains until user closes it

conn.close()

* **Result:**



1. **View:**
2. **Insert:**

****

1. **Update:**

****

1. **Delete:**

