# Advanced Database System Lab

## Assignment no. 2

**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:

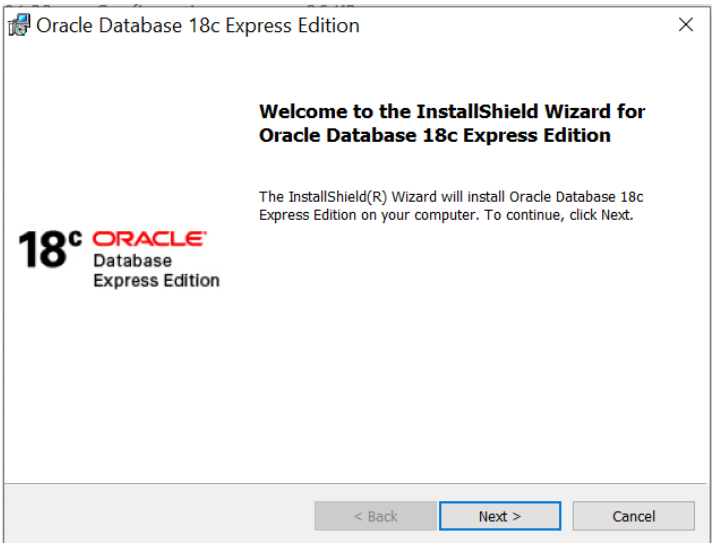
MySQL is an open-source relational database management system (RDBMS) .Its name is a combination of "My", the name of co-founder Michael Widenius's daughter My, and "SQL", the acronym for Structured Query Language. A relational database organizes data into one or more data tables in which data may be 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:

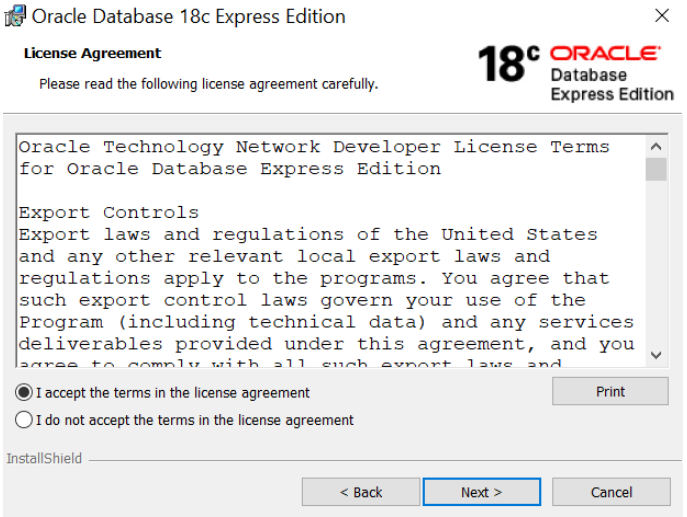
#### Oracle 18c XE

**Oracle Server Installation:**

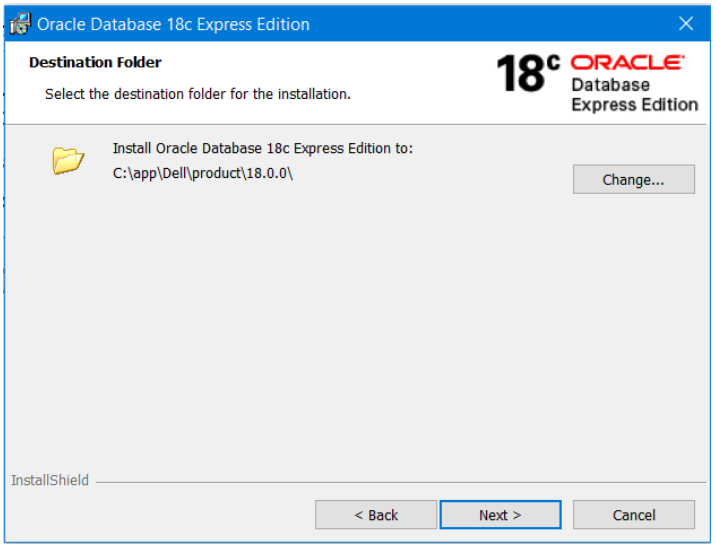
1) Download the Oracle 18c XE file from [oracle website] for your OS. Extract the zip file. Now Run the setup.exe file.



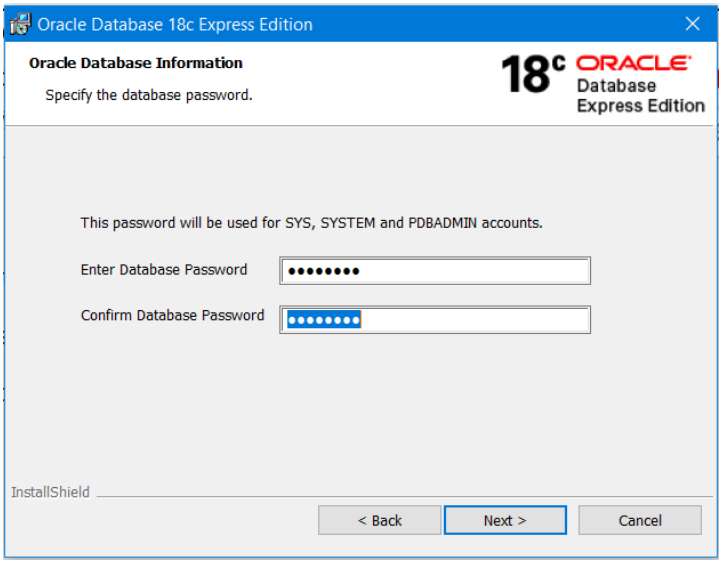

2) Read & accept the License Agreement.



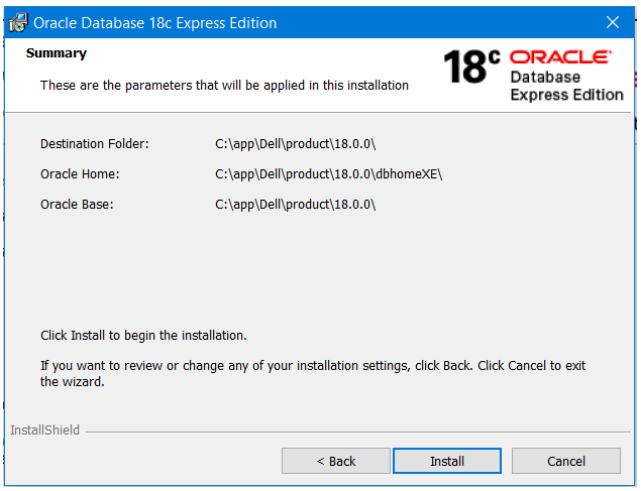

3) Choose the destination folder for the installation.



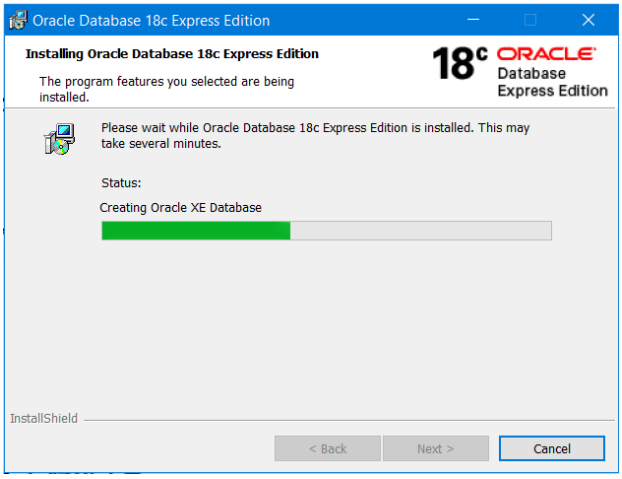

4) Enter a strong password for the database and confirm it.



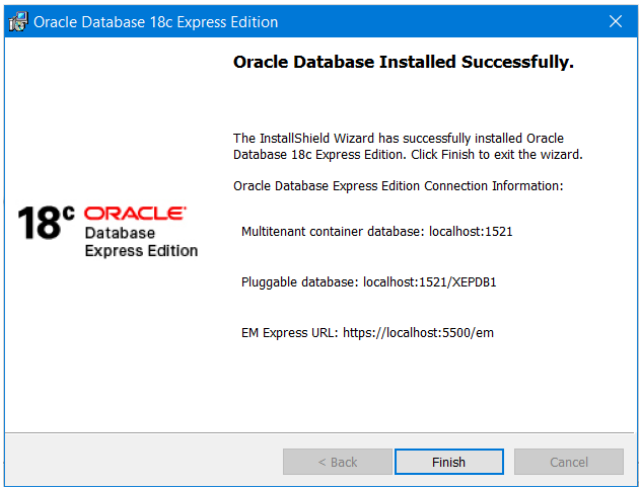

5) Verify the selected parameters.



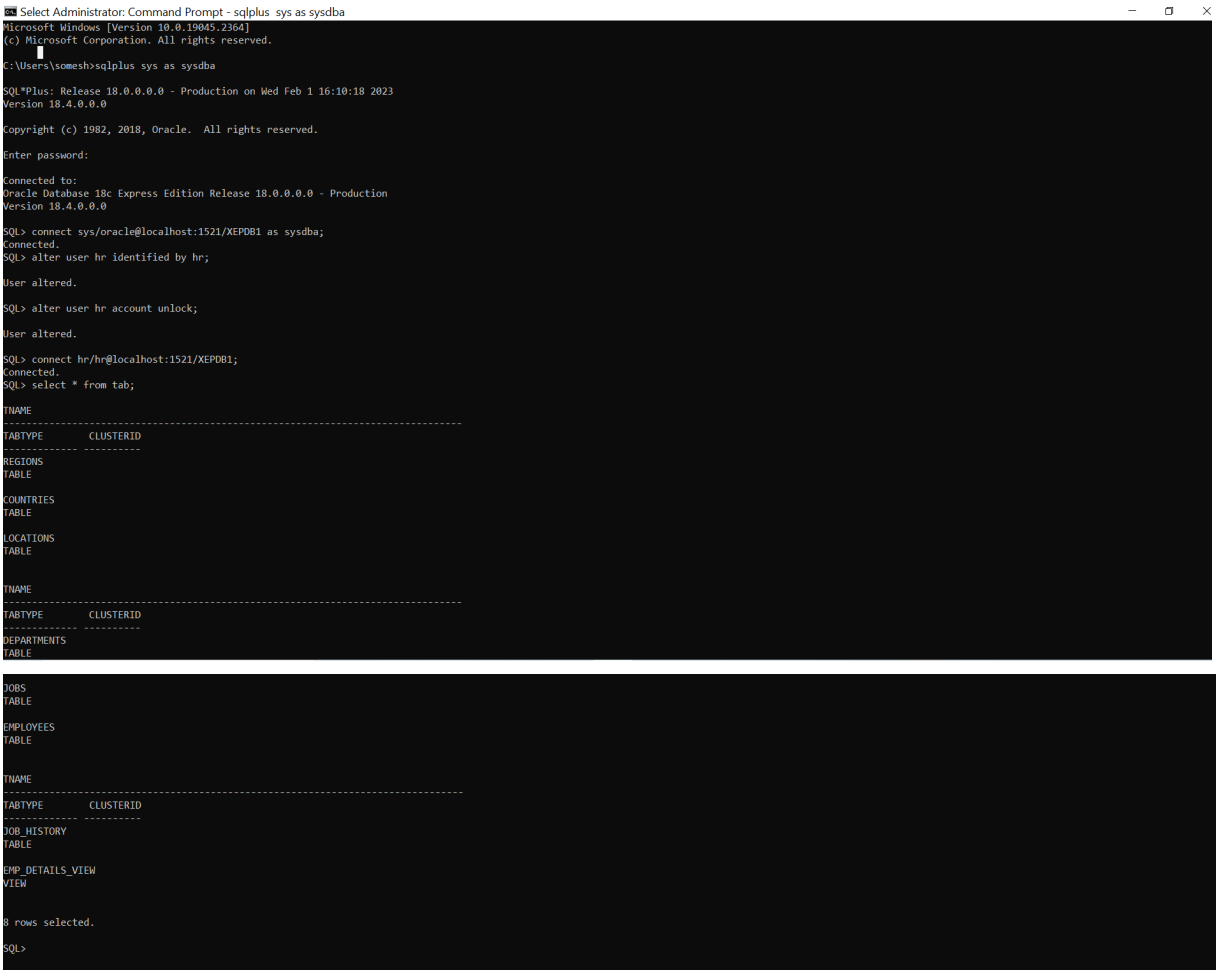

6) Wait for installation to be completed.



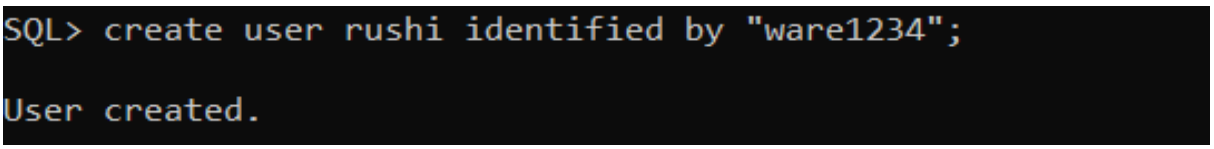

7) When the installation is completed, note down the connection information. Hence, the Oracle Server (18c Express Edition) is installed successfully.

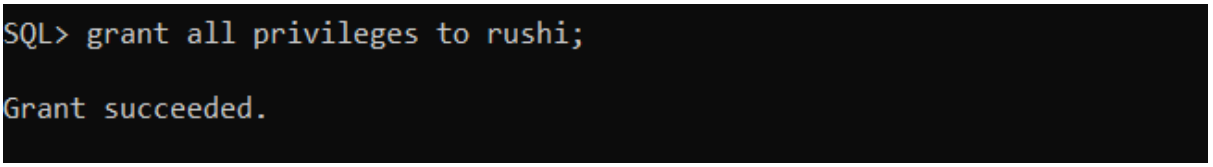


## Testing the connectivity:




## Creating User:



**Connecting to created user:**

```
SQL> conn
Enter user-name: rushi
Enter password:
```

**Creating Sample Tables:**

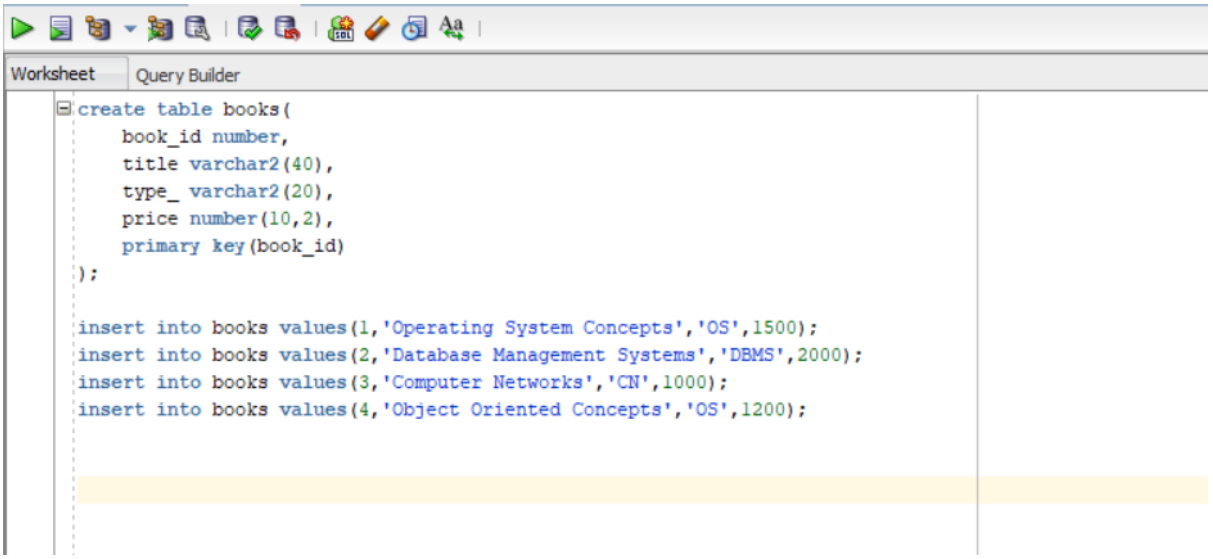
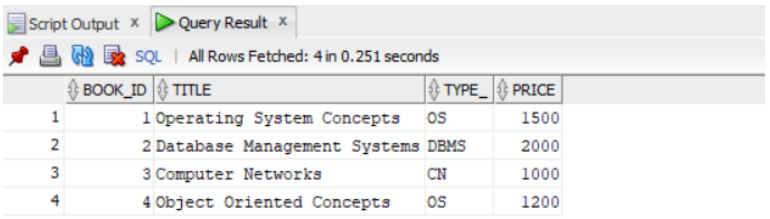



**Table Data:**




| | BOOK_ID | TITLE | TYPE_ | PRICE |
|---|---|---|---|---|
| 1 | 1 | Operating System Concepts | OS | 1500 |
| 2 | 2 | Database Management Systems | DBMS | 2000 |
| 3 | 3 | Computer Networks | CN | 1000 |
| 4 | 4 | Object Oriented Concepts | OS | 1200 |

**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='rushi', password='ware1234', 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 = "green"); # 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: rushi", background="green").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="green").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 = "green"); # 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="green").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 = "green"); # 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="green").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="green").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'
```

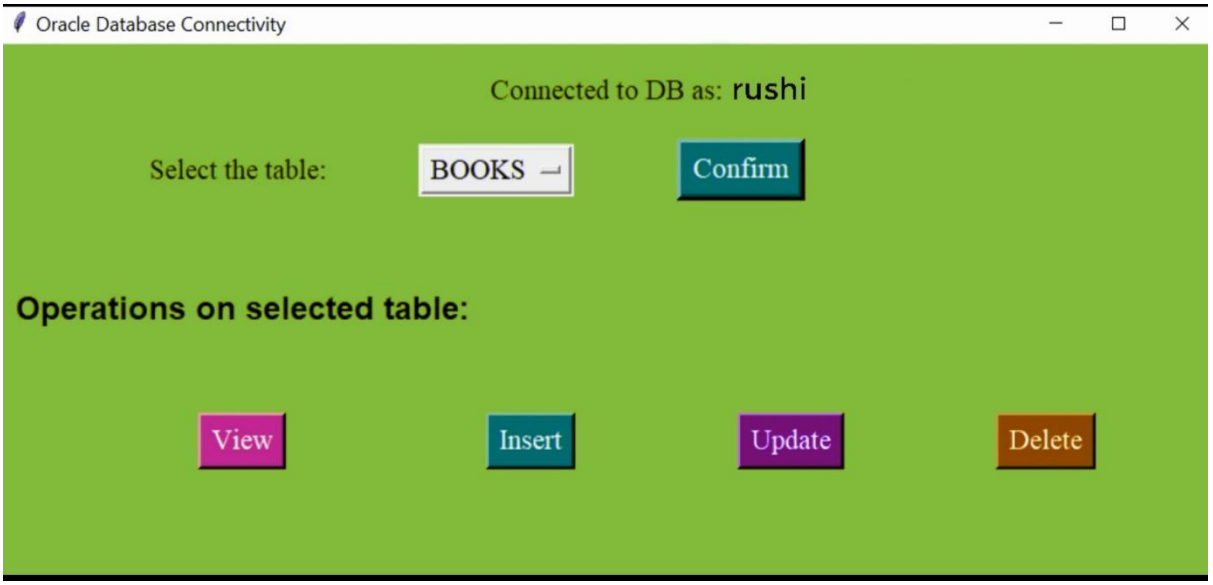
```
                '')

    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 = "green"); # Background color of window
            newWindow.option_add("*Font", "Times 16") # Setting the font-family & font-size

            Label(newWindow ,text = "Update values in table:", background="green").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="green").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="green", 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
```

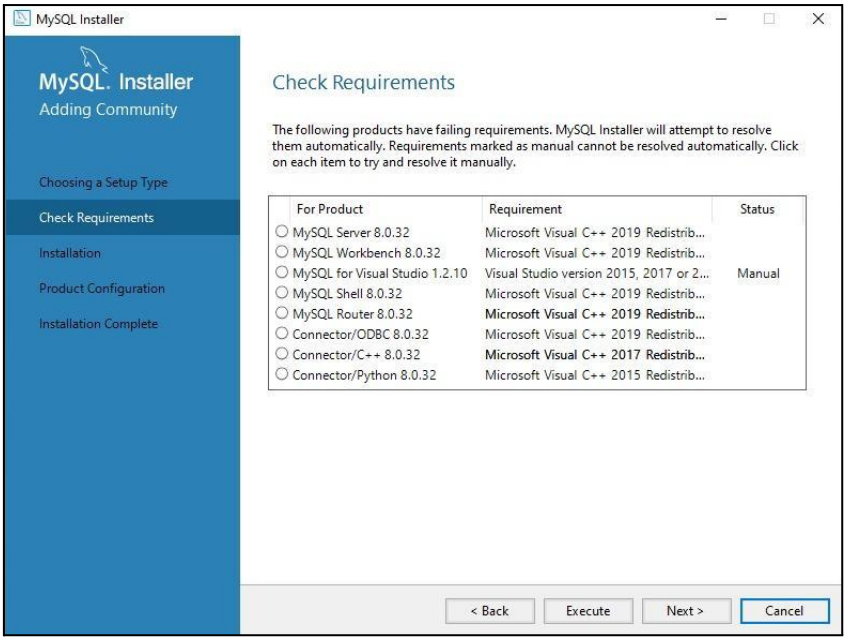
**Output:**



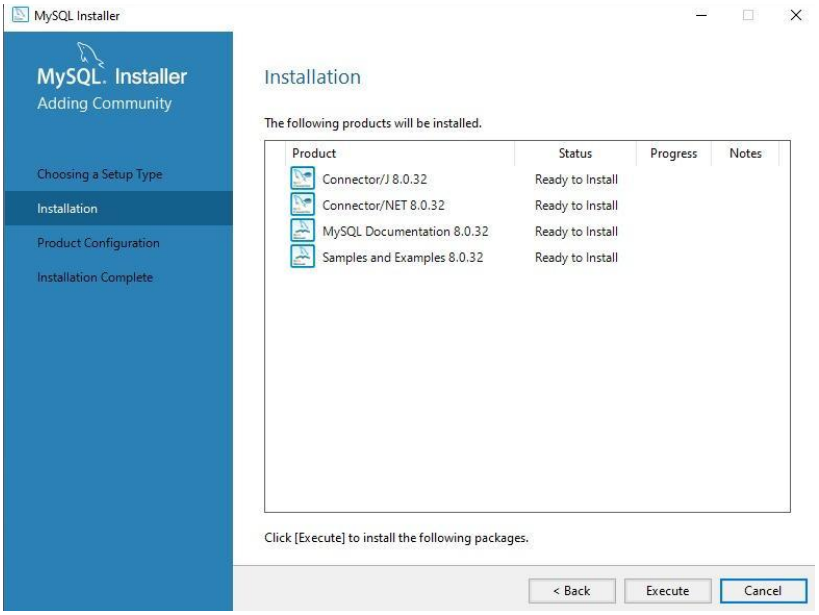

**MySQL**

**MySQL installation:**

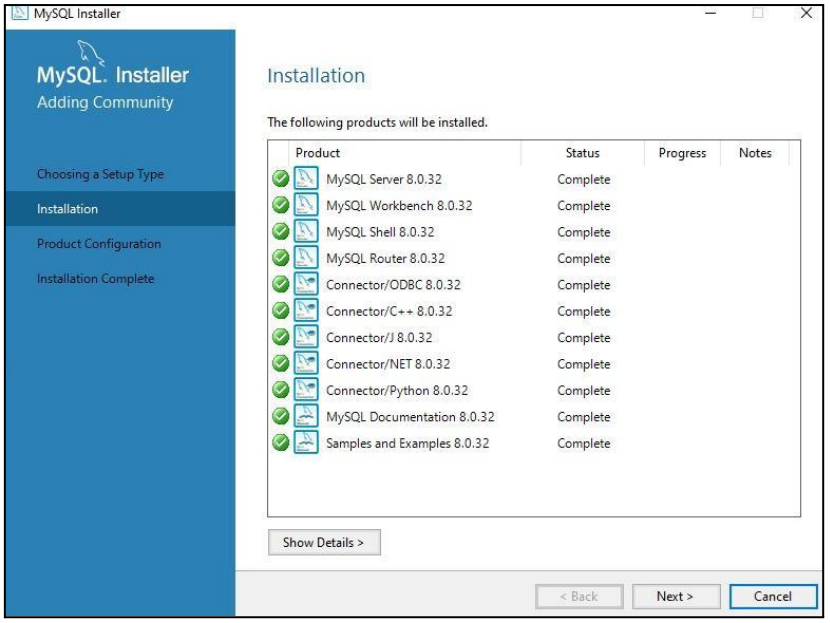
1) Install the setup for MySQL from MySQL website. Run the setup.exe. Choose installation type as 'Full'.
2) It will check whether your computer satisfies the necessary requirements. Install the necessary requirements (select requirements and click Execute) and click next.



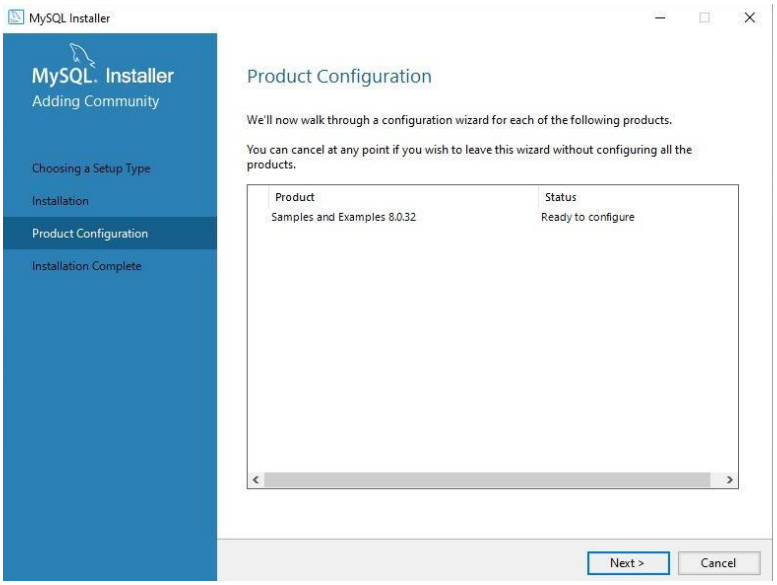

3) Click execute to install the required setups.



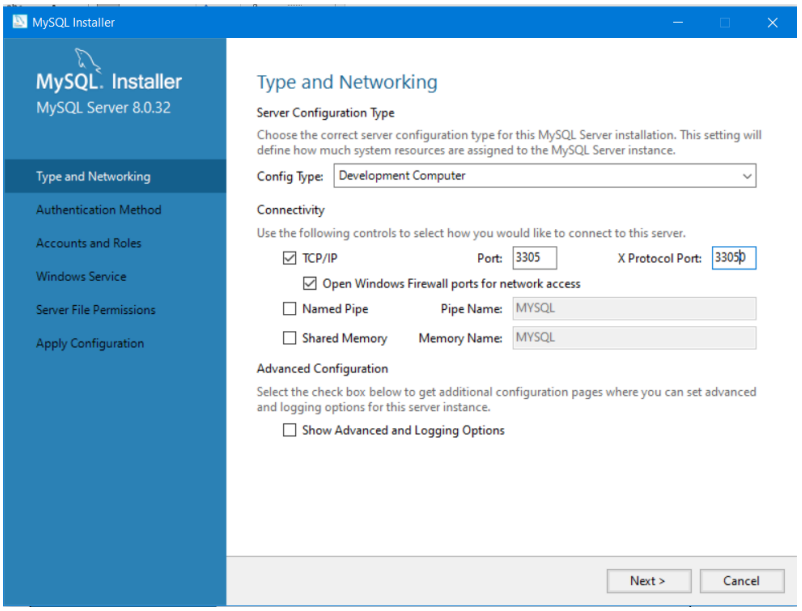

4) Wait until all the files are installed.



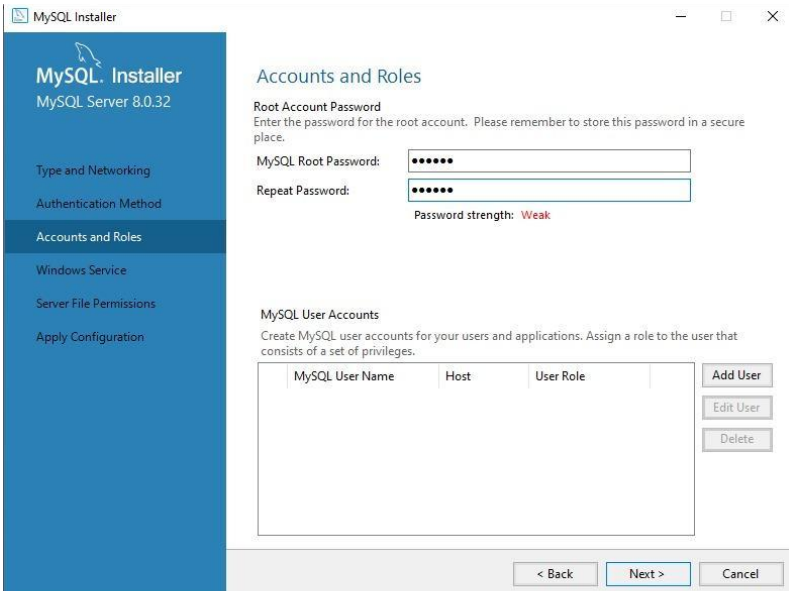

5) Configure the product by clicking next.



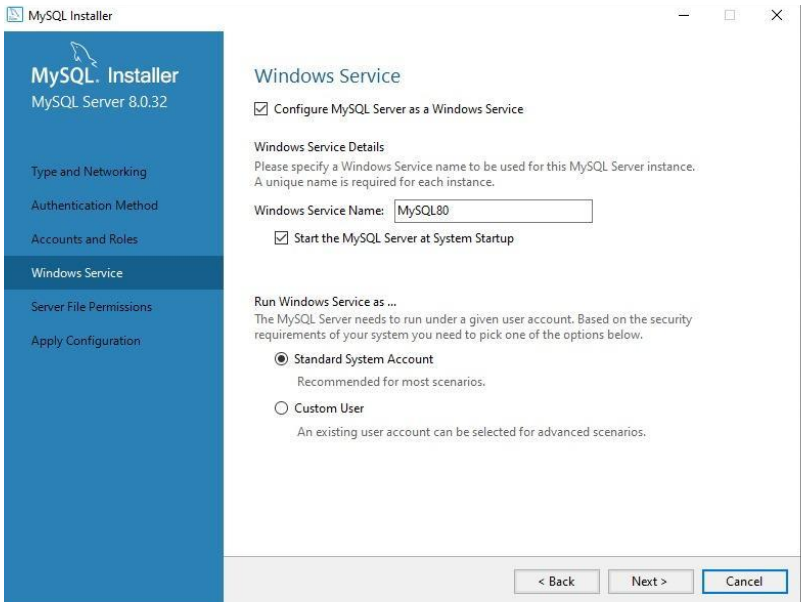

6) Configure the network and click next.



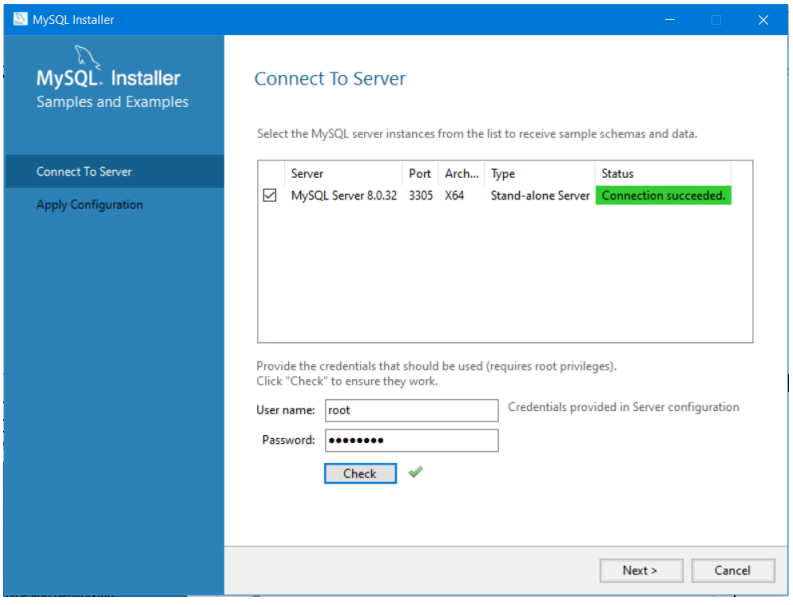

7) Set the password for ‘root’ (admin) and click next.



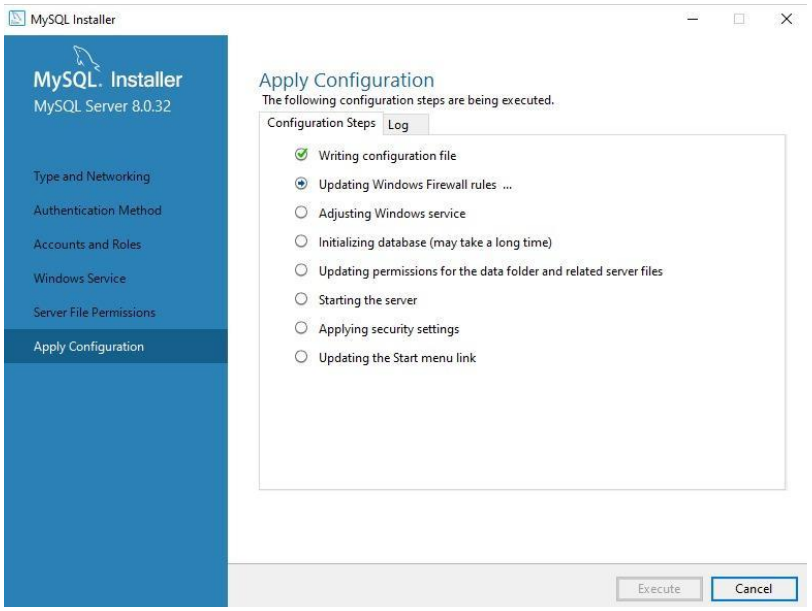
8) Configure the windows service to start. Click next.



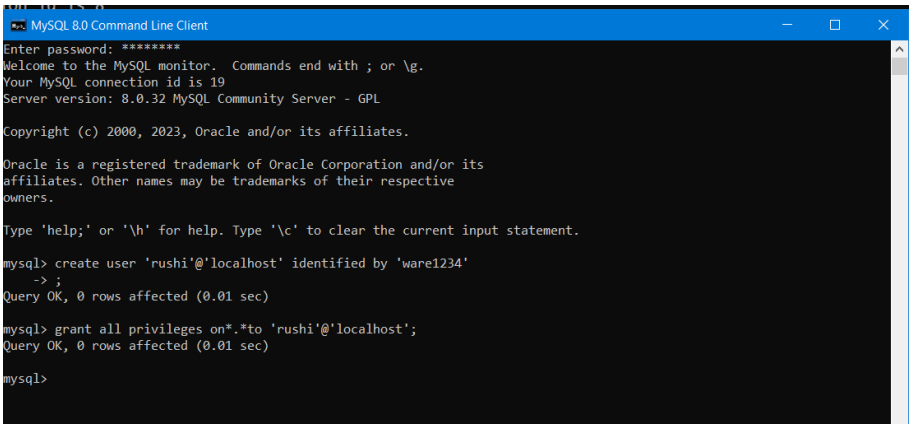

9) Connect to server by logging in with the created credentials. Check the connectivity.



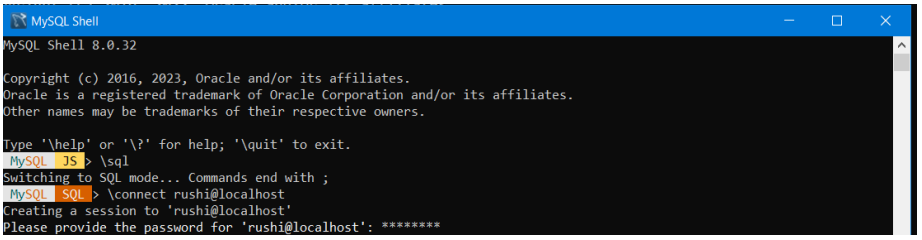
10) Apply the configuration by clicking on execute. Wait for all changes to be applied. MySQL is installed successfully.



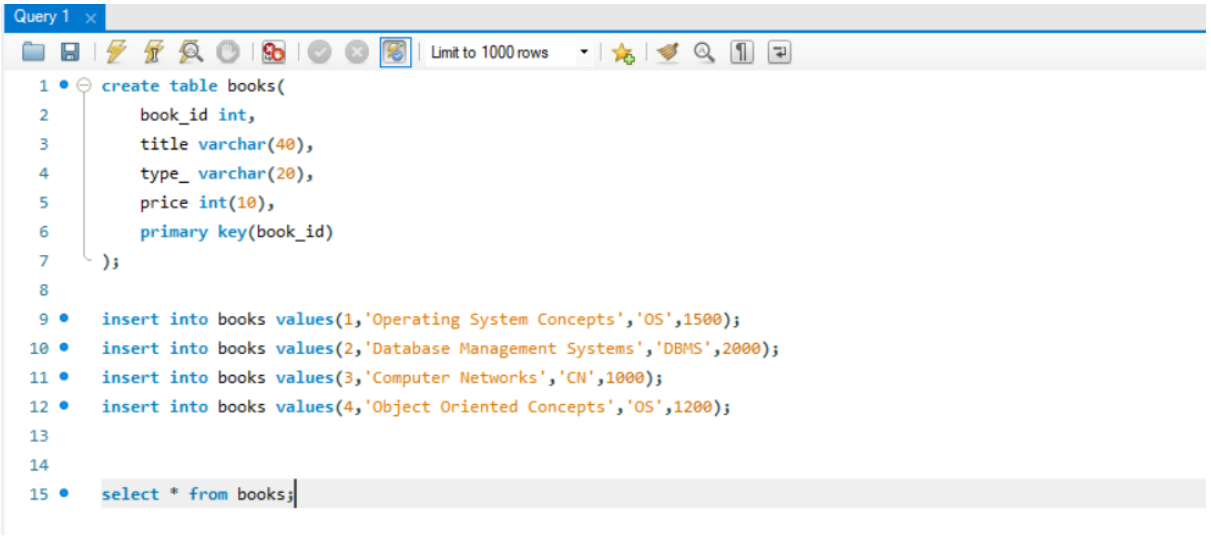

## Creating User:




## Connecting to created user:

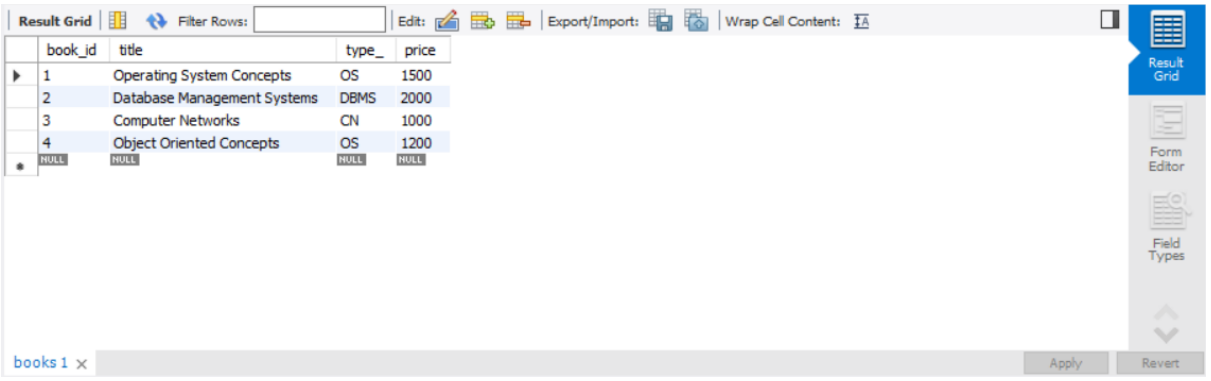


**Creating Sample Tables:**



```
create table books(
    book_id int,
    title varchar(40),
    type_ varchar(20),
    price int(10),
    primary key(book_id)
);

insert into books values(1,'Operating System Concepts','OS',1500);
insert into books values(2,'Database Management Systems','DBMS',2000);
insert into books values(3,'Computer Networks','CN',1000);
insert into books values(4,'Object Oriented Concepts','OS',1200);


select * from books;
```

**Table Data:**



| book_id | title | type_ | price |
|---|---|---|---|
| 1 | Operating System Concepts | OS | 1500 |
| 2 | Database Management Systems | DBMS | 2000 |
| 3 | Computer Networks | CN | 1000 |
| 4 | Object Oriented Concepts | OS | 1200 |

**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='2020BTECS00008', password='2020BTECS00008', database='ads')

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 = "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: 2020BTECS00008", background="orange").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="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

    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 = "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 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 = "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'''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="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(['%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: ")
```

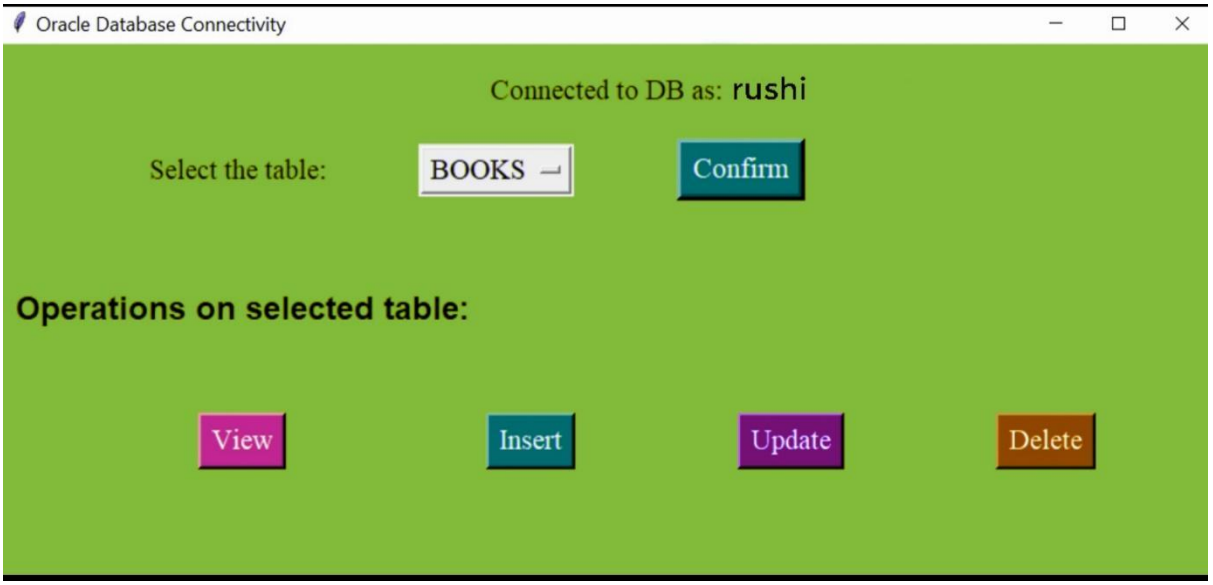
```
    if id is not None:
        c.execute(f'select * from {selected_tb} where {pk}={id}')

        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 = "orange"); # 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="orange").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="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(['? = %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="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
```

**Output:**


Oracle Database Connectivity
Connected to DB as: rushi
Select the table:
BOOKS
Confirm
Operations on selected table:
View
Insert
Update
Delete