**Code-**

#exp-7

def main():

    try:

        num1 = int(input("Enter the first number: "))

        num2 = int(input("Enter the second number: "))

        result = num1 / num2

        print(f"The result is: {result}")

    except ZeroDivisionError:

        print("Error: Cannot divide by zero!")

    except ValueError:

        print("Error: Invalid input! Please enter numbers only.")

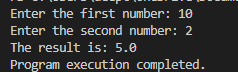
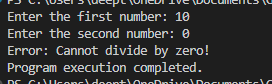
    finally:

        print("Program execution completed.")

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

    main()

OUTPUT-



**CODE-**

#exp-8

def read\_and\_write\_file(input\_file, output\_file):

    try:

        with open(input\_file, 'r') as infile:

            content = infile.read()

        with open(output\_file, 'w') as outfile:

            outfile.write(content)

        print(f"Content from {input\_file} has been written to {output\_file}")

    except FileNotFoundError:

        print(f"Error: {input\_file} not found!")

    except Exception as e:

        print(f"An error occurred: {e}")

def append\_and\_display\_file(file\_name, data\_to\_append):

    try:

        with open(file\_name, 'a') as file:

            file.write(data\_to\_append + '\n')

        with open(file\_name, 'r') as file:

            content = file.read()

            print(f"Updated content of {file\_name}:\n{content}")

    except FileNotFoundError:

        print(f"Error: {file\_name} not found!")

    except Exception as e:

        print(f"An error occurred: {e}")

def count\_file\_content(file\_name):

    try:

        with open(file\_name, 'r') as file:

            lines = file.readlines()

            num\_lines = len(lines)

            num\_words = sum(len(line.split()) for line in lines)

            num\_chars = sum(len(line) for line in lines)

        print(f"Lines: {num\_lines}, Words: {num\_words}, Characters: {num\_chars}")

    except FileNotFoundError:

        print(f"Error: {file\_name} not found!")

    except Exception as e:

        print(f"An error occurred: {e}")

def main():

    input\_file = 'input.txt'

    output\_file = 'output.txt'

    data\_to\_append = 'Appended content'

    read\_and\_write\_file(input\_file, output\_file)

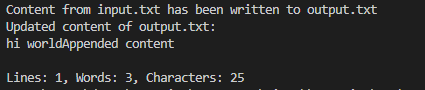
    append\_and\_display\_file(output\_file, data\_to\_append)

    count\_file\_content(output\_file)

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

    main()

OUTPUT-



**Code-**

#exp-9

import os

def count\_file\_content(file\_name):

    try:

        with open(file\_name, 'r') as file:

            lines = file.readlines()

            num\_lines = len(lines)

            num\_words = sum(len(line.split()) for line in lines)

            num\_chars = sum(len(line) for line in lines)

        print(f"Lines: {num\_lines}, Words: {num\_words}, Characters: {num\_chars}")

    except FileNotFoundError:

        print(f"Error: {file\_name} not found!")

    except Exception as e:

        print(f"An error occurred: {e}")

def display\_files\_in\_directory():

    try:

        files = os.listdir('.')

        print("Files in the current directory:")

        for file in files:

            print(file)

    except Exception as e:

        print(f"An error occurred: {e}")

def main():

    file\_name = 'output.txt'

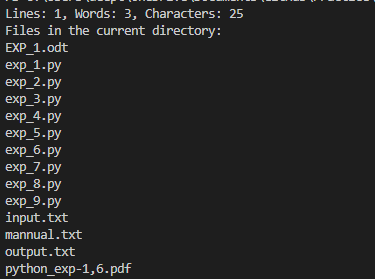
    count\_file\_content(file\_name)

    display\_files\_in\_directory()

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

    main()

OUTPUT-



**Code-**

#exp-10

import sqlite3

# Creating a connection object

conn = sqlite3.connect('student\_database.db')

# Creating a cursor object using the connection

cursor = conn.cursor()

# Creating a table in the database

def create\_table():

    cursor.execute('''CREATE TABLE IF NOT EXISTS students (

                        id INTEGER PRIMARY KEY AUTOINCREMENT,

                        name TEXT NOT NULL,

                        age INTEGER NOT NULL,

                        grade TEXT NOT NULL)''')

    print("Table created successfully!")

# Inserting values into the table

def insert\_values(name, age, grade):

    cursor.execute('''INSERT INTO students (name, age, grade)

                      VALUES (?, ?, ?)''', (name, age, grade))

    conn.commit()

    print("Values inserted successfully!")

# Displaying values from the table

def display\_values():

    cursor.execute('SELECT \* FROM students')

    rows = cursor.fetchall()

    for row in rows:

        print(row)

# Updating values in the table

def update\_values(student\_id, new\_grade):

    cursor.execute('''UPDATE students SET grade = ? WHERE id = ?''', (new\_grade, student\_id))

    conn.commit()

    print("Values updated successfully!")

# Main function

def main():

    create\_table()

    insert\_values('John Doe', 20, 'A')

    insert\_values('Jane Smith', 22, 'B')

    display\_values()

    update\_values(1, 'A+')

    display\_values()

# Calling the main function

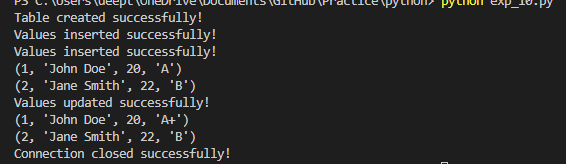
if \_\_name\_\_ == "\_\_main\_\_":

    main()

conn.close()

print("Connection closed successfully!")

OUTPUT-



**Code-**

#exp-11

import tkinter as tk

from tkinter import messagebox

import sqlite3

def create\_table():

    conn = sqlite3.connect('student.db')

    c = conn.cursor()

    c.execute('''CREATE TABLE IF NOT EXISTS students

                 (roll\_no INTEGER PRIMARY KEY, name TEXT, subject1 INTEGER, subject2 INTEGER)''')

    conn.commit()

    conn.close()

def insert\_data():

    conn = sqlite3.connect('student.db')

    c = conn.cursor()

    try:

        roll\_no = int(entry\_roll.get())

        name = entry\_name.get()

        subject1 = int(entry\_subject1.get())

        subject2 = int(entry\_subject2.get())

        c.execute("INSERT INTO students (roll\_no, name, subject1, subject2) VALUES (?, ?, ?, ?)",

                  (roll\_no, name, subject1, subject2))

        conn.commit()

        entry\_roll.delete(0, tk.END)

        entry\_name.delete(0, tk.END)

        entry\_subject1.delete(0, tk.END)

        entry\_subject2.delete(0, tk.END)

        fetch\_data()

    except Exception as e:

        messagebox.showerror("Error", f"Could not insert data: {e}")

    conn.close()

def fetch\_data():

    conn = sqlite3.connect('student.db')

    c = conn.cursor()

    c.execute("SELECT \* FROM students")

    rows = c.fetchall()

    listbox.delete(0, tk.END)  # Clear the listbox before inserting new data

    for row in rows:

        listbox.insert(tk.END, row)

    conn.close()

def delete\_data():

    conn = sqlite3.connect('student.db')

    c = conn.cursor()

    try:

        roll\_no = int(entry\_roll.get())

        c.execute("DELETE FROM students WHERE roll\_no=?", (roll\_no,))

        conn.commit()

        entry\_roll.delete(0, tk.END)

        fetch\_data()

    except Exception as e:

        messagebox.showerror("Error", f"Could not delete data: {e}")

    conn.close()

root = tk.Tk()

root.title("Student Database")

root.geometry("500x500")

tk.Label(root, text="Roll No:").pack()

entry\_roll = tk.Entry(root)

entry\_roll.pack()

tk.Label(root, text="Name:").pack()

entry\_name = tk.Entry(root)

entry\_name.pack()

tk.Label(root, text="Marks for Subject 1:").pack()

entry\_subject1 = tk.Entry(root)

entry\_subject1.pack()

tk.Label(root, text="Marks for Subject 2:").pack()

entry\_subject2 = tk.Entry(root)

entry\_subject2.pack()

# Step 4: Buttons to trigger DB Operations

tk.Button(root, text="Insert", command=insert\_data).pack()

tk.Button(root, text="Delete", command=delete\_data).pack()

listbox = tk.Listbox(root, width=50)

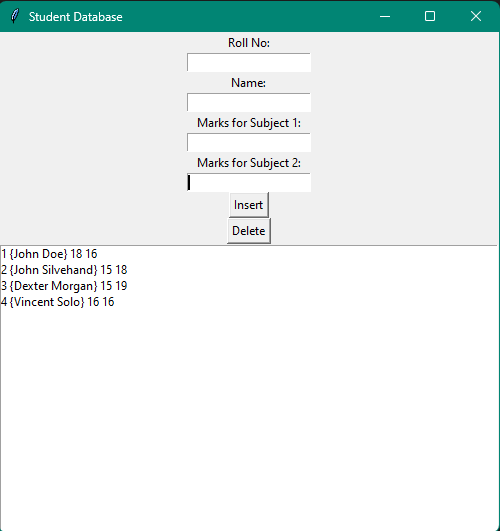
listbox.pack(fill=tk.BOTH, expand=True)

create\_table()

fetch\_data()

root.mainloop()

OUTPUT-



**Code-**

#exp-12

import pandas as pd

print("Pandas Series Example:")

data\_series = pd.Series([10, 20, 30, 40, 50], index=['a', 'b', 'c', 'd', 'e'])

print("Series:\n", data\_series)

print("\nAccessing Element with Label 'c':", data\_series['c'])

print("Accessing Element by Position 3:", data\_series[3])

print("\nSeries multiplied by 2:\n", data\_series \* 2)

print("Sum of the series:", data\_series.sum())

print("\nPandas DataFrame Example:")

data = {'Name': ['John', 'Anna', 'Peter', 'Linda'],

        'Age': [28, 24, 35, 32],

        'Department': ['HR', 'Finance', 'Marketing', 'IT']}

data\_frame = pd.DataFrame(data)

print("DataFrame:\n", data\_frame)

print("\nAccessing 'Name' column:\n", data\_frame['Name'])

print("Accessing first row using .loc[]:\n", data\_frame.loc[0])

print("\nSorting DataFrame by Age:\n", data\_frame.sort\_values(by='Age'))

print("Mean Age of employees:", data\_frame['Age'].mean())

data\_frame['Salary'] = [50000, 60000, 55000, 62000]

print("\nDataFrame with new 'Salary' column:\n", data\_frame)

print("\nFinal DataFrame after modifications:\n", data\_frame)

OUTPUT-

