**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-

```
Enter the first number: 10
Enter the second number: 0
Error: Cannot divide by zero!
Program execution completed.
```

```
Enter the first number: 10
Enter the second number: 2
The result is: 5.0
Program execution completed.
```

**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-

```
Content from input.txt has been written to output.txt
Updated content of output.txt:
hi worldAppended content

Lines: 1, Words: 3, Characters: 25
```

**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-

```
Lines: 1, Words: 3, Characters: 25
Files in the current directory:
EXP_1.odt
exp_1.py
exp_2.py
exp_3.py
exp_4.py
exp_5.py
exp_6.py
exp_7.py
exp_8.py
exp_9.py
input.txt
mannual.txt
output.txt
python_exp-1,6.pdf
```

**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-

```
Table created successfully!
Values inserted successfully!
Values inserted successfully!
(1, 'John Doe', 20, 'A')
(2, 'Jane Smith', 22, 'B')
Values updated successfully!
(1, 'John Doe', 20, 'A+')
(2, 'Jane Smith', 22, 'B')
Connection closed successfully!
```

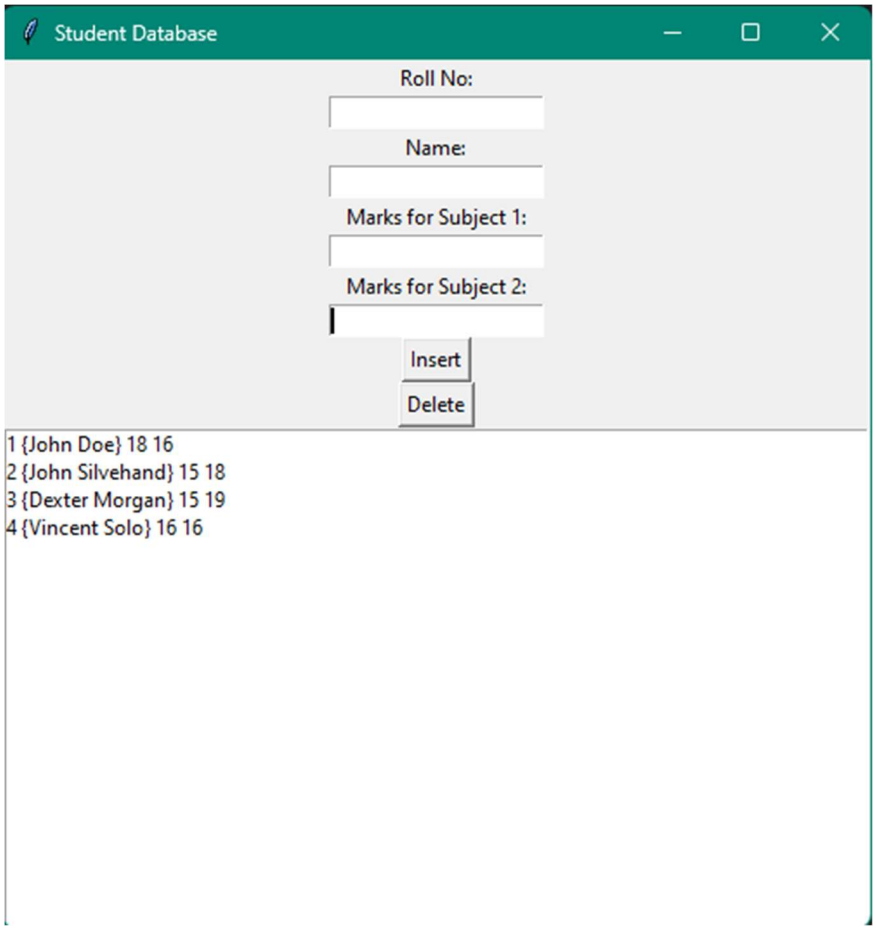
**Code-**

```
#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-


Student Database
Roll No:
Name:
Marks for Subject 1:
Marks for Subject 2:
Insert
Delete
1 {John Doe} 18 16
2 {John Silvehand} 15 18
3 {Dexter Morgan} 15 19
4 {Vincent Solo} 16 16

**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-

```
Pandas DataFrame Example:
DataFrame:
     Name  Age Department
0   John   28         HR
1   Anna   24    Finance
2  Peter   35  Marketing
3  Linda   32         IT

Accessing 'Name' column:
 0     John
1     Anna
2    Peter
3    Linda
Name: Name, dtype: object
Accessing first row using .loc[]:
 Name          John
Age             28
Department      HR
Name: 0, dtype: object

Sorting DataFrame by Age:
     Name  Age Department
1   Anna   24    Finance
0   John   28         HR
3  Linda   32         IT
2  Peter   35  Marketing
Mean Age of employees: 29.75
```

```
DataFrame with new 'Salary' column:
     Name  Age Department  Salary
0   John   28         HR   50000
1   Anna   24    Finance   60000
2  Peter   35  Marketing   55000
3  Linda   32         IT   62000
```