DBMS AND VISUAL BASIC

PRACTICAL FILE

NAME : prateek yadav

ROLL NO: 24236761051074

SEMESTER: 2nd

SECTION :- C

DEPARTMENT OF OPERATIONAL RESEARCH

SOUTH CAMPUS

SUBMISSION DATE :- 06-MAY-2025

SUBMITTED TO : – Dr. Jagvinder Singh

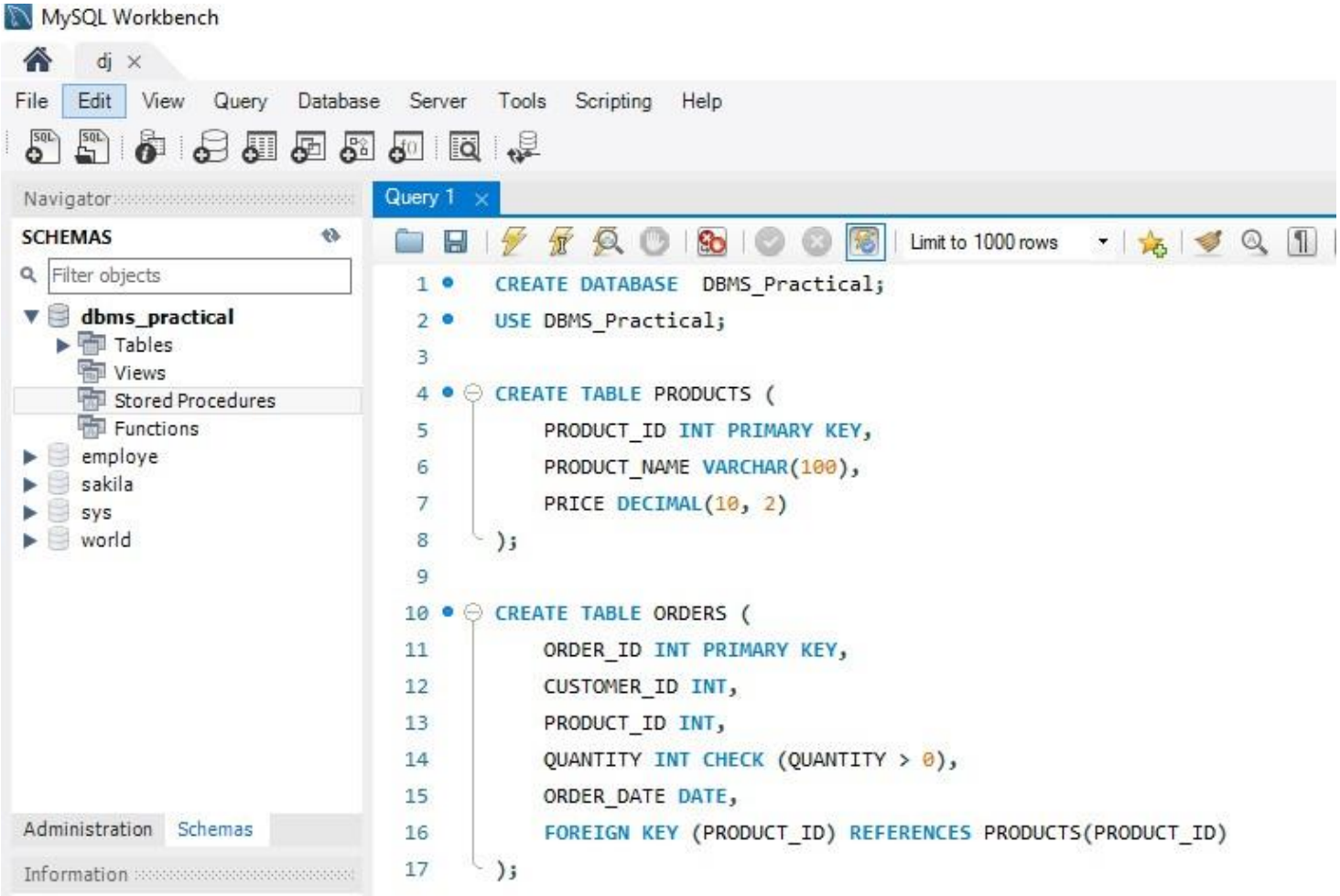
CONTENTS

Q1. Create a table ORDERS and PRODUCTS with the following fields.

ORDERS: {ORDER_ID, CUSTOMER_ID, PRODUCT_ID, QUANTITY, ORDER_DATE} PRODUCTS: {PRODUCT_ID, PRODUCT_NAME, PRICE}

[first create a DATABASE and Table.]

i. Insert at least 10 records into the ORDERS and PRODUCTS tables.



```sql
INSERT INTO PRODUCTS (PRODUCT_ID, PRODUCT_NAME, PRICE) VALUES
(1, 'Laptop', 1200.00),
(2, 'Smartphone', 800.00),
(3, 'Tablet', 450.00),
(4, 'Headphones', 150.00),
(5, 'Monitor', 300.00),
(6, 'Keyboard', 70.00),
(7, 'Mouse', 50.00),
(8, 'Printer', 200.00),
(9, 'Webcam', 90.00),
(10, 'Charger', 25.00);

INSERT INTO ORDERS (ORDER_ID, CUSTOMER_ID, PRODUCT_ID, QUANTITY, ORDER_DATE) VALUES
(1, 101, 1, 1, '2024-05-01'),
(2, 102, 2, 2, '2024-05-02'),
(3, 101, 3, 1, '2024-05-06'),
(4, 103, 4, 3, '2024-05-07'),
(5, 104, 5, 2, '2024-05-05'),
(6, 102, 6, 1, '2024-05-08'),
(7, 105, 7, 4, '2024-05-09'),
(8, 101, 8, 1, '2024-05-03'),
(9, 106, 9, 2, '2024-05-10'),
(10, 104, 10, 3, '2024-05-11');
```

ii. Count the number of orders placed by each customer.

```sql
SELECT CUSTOMER_ID, COUNT(*) AS ORDER_COUNT
FROM ORDERS
GROUP BY CUSTOMER_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CUSTOMER_ID	ORDER_COUNT
101	3
102	2
103	1
104	2
105	1
106	1

Result 1

duct sold.

```
SELECT PRODUCT_ID, SUM(QUANTITY) AS TOTAL_QUANTITY_SOLD
FROM ORDERS
GROUP BY PRODUCT_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PRODUCT_ID	TOTAL_QUANTITY_SOLD
1	1
2	2
3	1
4	3
5	2
6	1
7	4
8	1
9	2
10	3

iv. Find the top 3 products that generated the highest revenue.

```
SELECT PRODUCT_ID, SUM(QUANTITY) AS TOTAL_QUANTITY_SOLD
FROM ORDERS
GROUP BY PRODUCT_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PRODUCT_ID	TOTAL_QUANTITY_SOLD
1	1
2	2
3	1
4	3
5	2
6	1
7	4
8	1
9	2
10	3

v. Display the details of orders placed on or after '2024-05-05' sorted by ORDER_DATE in descending order.

```sql
SELECT *
FROM ORDERS
WHERE ORDER_DATE >= '2024-05-05'
ORDER BY ORDER_DATE DESC;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ORDER_ID	CUSTOMER_ID	PRODUCT_ID	QUANTITY	ORDER_DATE
10	104	10	3	2024-05-11
9	106	9	2	2024-05-10
7	105	7	4	2024-05-09
6	102	6	1	2024-05-08
4	103	4	3	2024-05-07
3	101	3	1	2024-05-06
5	104	5	2	2024-05-05
NULL	NULL	NULL	NULL	NULL

ORDERS 4

vi. Add a data constraint to ensure that the QUANTITY in the ORDERS table is always greater than zero.

```sql
ALTER TABLE ORDERS
ADD CONSTRAINT chk_quantity_positive CHECK (QUANTITY > 0);
```

vii. Calculate the total revenue generated by each customer.

```sql
SELECT CUSTOMER_ID, SUM(O.QUANTITY * P.PRICE) AS TOTAL_REVENUE
FROM ORDERS O
JOIN PRODUCTS P ON O.PRODUCT_ID = P.PRODUCT_ID
GROUP BY CUSTOMER_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CUSTOMER_ID	TOTAL_REVENUE
101	1850.00
102	1670.00
103	450.00
104	675.00
105	200.00
106	180.00

Result 5

Q2. Create a table STUDENT and EXAM with the following fields:

STUDENT TABLE: {STUDENT_ID, Name, Gender, Age, Class ('10th Grade', '11th Grade', '12th Grade')}

EXAM: {STUDENT_ID, MATH_SCORE, SCIENCE_SCORE, ENGLISH_SCORE}

i. *Insert at least 10 records in the tables.*

```sql
CREATE TABLE STUDENT (
    STUDENT_ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Gender VARCHAR(10),
    Age INT,
    Class VARCHAR(20) CHECK (Class IN ('10th Grade', '11th Grade', '12th Grade'))
);

CREATE TABLE EXAM (
    STUDENT_ID INT PRIMARY KEY,
    MATH_SCORE INT,
    SCIENCE_SCORE INT,
    ENGLISH_SCORE INT,
    FOREIGN KEY (STUDENT_ID) REFERENCES STUDENT(STUDENT_ID)
);
```

```sql
INSERT INTO STUDENT (STUDENT_ID, Name, Gender, Age, Class) VALUES
(1, 'Aryan Mehta', 'Male', 16, '10th Grade'),
(2, 'Riya Sharma', 'Female', 15, '10th Grade'),
(3, 'Aditya Verma', 'Male', 17, '11th Grade'),
(4, 'Sneha Kapoor', 'Female', 17, '11th Grade'),
(5, 'Rahul Joshi', 'Male', 18, '12th Grade'),
(6, 'Priya Desai', 'Female', 18, '12th Grade'),
(7, 'Karan Malhotra', 'Male', 17, '11th Grade'),
(8, 'Neha Sinha', 'Female', 16, '10th Grade'),
(9, 'Varun Singh', 'Male', 18, '12th Grade'),
(10, 'Ananya Iyer', 'Female', 17, '11th Grade');

INSERT INTO EXAM (STUDENT_ID, MATH_SCORE, SCIENCE_SCORE, ENGLISH_SCORE) VALUES
(1, 78, 82, 75),
(2, 85, 88, 80),
(3, 90, 91, 87),
(4, 76, 79, 82),
(5, 88, 84, 89),
(6, 92, 90, 94),
(7, 89, 85, 86),
(8, 70, 72, 68),
(9, 95, 93, 96),
(10, 88, 90, 85);
```

ii. *Display the details of male students who are in '12th Grade'.*

```
SELECT * FROM STUDENT
WHERE Gender = 'Male' AND Class = '12th Grade';
```

Result Grid | Filter Rows: | Edit:

STUDENT_ID	Name	Gender	Age	Class
5	Rahul Joshi	Male	18	12th Grade
9	Varun Singh	Male	18	12th Grade

iii. *Display the details of the student who secured the highest total score.*

```
SELECT S.STUDENT_ID, S.Name, (E.MATH_SCORE + E.SCIENCE_SCORE + E.ENGLISH_SCORE) AS TOTAL_SCORE
FROM STUDENT S
JOIN EXAM E ON S.STUDENT_ID = E.STUDENT_ID
ORDER BY TOTAL_SCORE DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

STUDENT_ID	Name	TOTAL_SCORE
9	Varun Singh	284

iv. *Add a new Column HISTORY_SCORE in EXAM table and modify the table by inserting values to HISTORY_SCORE for the records.*

```
ALTER TABLE EXAM ADD COLUMN HISTORY_SCORE INT;

UPDATE EXAM SET HISTORY_SCORE = 80 WHERE STUDENT_ID = 1;
UPDATE EXAM SET HISTORY_SCORE = 85 WHERE STUDENT_ID = 2;
UPDATE EXAM SET HISTORY_SCORE = 88 WHERE STUDENT_ID = 3;
UPDATE EXAM SET HISTORY_SCORE = 79 WHERE STUDENT_ID = 4;
UPDATE EXAM SET HISTORY_SCORE = 90 WHERE STUDENT_ID = 5;
UPDATE EXAM SET HISTORY_SCORE = 92 WHERE STUDENT_ID = 6;
UPDATE EXAM SET HISTORY_SCORE = 84 WHERE STUDENT_ID = 7;
UPDATE EXAM SET HISTORY_SCORE = 70 WHERE STUDENT_ID = 8;
UPDATE EXAM SET HISTORY_SCORE = 95 WHERE STUDENT_ID = 9;
UPDATE EXAM SET HISTORY_SCORE = 87 WHERE STUDENT_ID = 10;
```

v. *List Top 3 students of '11th Grade' based on total score.*

```
SELECT S.STUDENT_ID, S.Name, (E.MATH_SCORE + E.SCIENCE_SCORE + E.ENGLISH_SCORE + E.HISTORY_SCORE) AS TOTAL_SCORE
FROM STUDENT S
JOIN EXAM E ON S.STUDENT_ID = E.STUDENT_ID
WHERE S.Class = '11th Grade'
ORDER BY TOTAL_SCORE DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

STUDENT_ID	Name	TOTAL_SCORE
3	Aditya Verma	356
10	Ananya Iyer	350
7	Karan Malhotra	344

vi. *Display the Average Age of students in '10th Grade'.*

```
SELECT AVG(Age) AS AVERAGE_AGE
FROM STUDENT
WHERE Class = '10th Grade';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVERAGE_AGE
15.6667

vii. *Display the list of students who scored more than the average marks in MATH_SCORE.*

```
SELECT S.STUDENT_ID, S.Name, E.MATH_SCORE
FROM STUDENT S
JOIN EXAM E ON S.STUDENT_ID = E.STUDENT_ID
WHERE E.MATH_SCORE > (
    SELECT AVG(MATH_SCORE) FROM EXAM
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

STUDENT_ID	Name	MATH_SCORE
3	Aditya Verma	90
5	Rahul Joshi	88
6	Priya Desai	92
7	Karan Malhotra	89
9	Varun Singh	95
10	Ananya Iyer	88

Result 14

Q3. Create a table named SALES and ITEMS with the following field.

SALES: {SALE_ID, CUSTOMER_ID, ITEM_ID, UNITS_SOLD, SALE_DATE}

ITEMS: {ITEM_ID, ITEM_NAME, UNIT_PRICE}

Perform the following queries in SQL.

First we create a TABLE.

```sql
CREATE TABLE ITEMS (
    ITEM_ID INT PRIMARY KEY,
    ITEM_NAME VARCHAR(100),
    UNIT_PRICE DECIMAL(10, 2)
);

CREATE TABLE SALES (
    SALE_ID INT PRIMARY KEY,
    CUSTOMER_ID INT,
    ITEM_ID INT,
    UNITS_SOLD INT CHECK (UNITS_SOLD > 0),
    SALE_DATE DATE,
    FOREIGN KEY (ITEM_ID) REFERENCES ITEMS(ITEM_ID)
);
```

i. *Insert at least 10 records into the SALES and ITEMS tables.*

```sql
INSERT INTO ITEMS (ITEM_ID, ITEM_NAME, UNIT_PRICE) VALUES
(1, 'Wireless Mouse', 25.99),
(2, 'Mechanical Keyboard', 79.99),
(3, 'USB-C Charger', 19.99),
(4, 'Laptop Stand', 34.50),
(5, 'Bluetooth Speaker', 45.00),
(6, 'Webcam', 59.99),
(7, 'External Hard Drive', 89.00),
(8, 'LED Desk Lamp', 29.99),
(9, 'Noise-Canceling Headphones', 120.00),
(10, 'Smartwatch', 150.00);

INSERT INTO SALES (SALE_ID, CUSTOMER_ID, ITEM_ID, UNITS_SOLD, SALE_DATE) VALUES
(1, 201, 1, 2, '2024-05-20'),
(2, 202, 3, 1, '2024-06-03'),
(3, 203, 5, 3, '2024-06-01'),
(4, 204, 2, 1, '2024-06-04'),
(5, 201, 4, 2, '2024-06-05'),
(6, 205, 6, 1, '2024-05-25'),
(7, 202, 7, 2, '2024-06-06'),
(8, 206, 9, 1, '2024-06-02'),
(9, 203, 10, 1, '2024-06-07'),
(10, 204, 8, 3, '2024-05-30');
```

ii. *Count the number of sales made by each customer.*

```
SELECT CUSTOMER_ID, COUNT(*) AS TOTAL_SALES
FROM SALES
GROUP BY CUSTOMER_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CUSTOMER_ID	TOTAL_SALES
201	2
202	2
203	2
204	2
205	1
206	1

Result 15

iii. *Calculate the total units sold for each item.*

```
SELECT ITEM_ID, SUM(UNITS_SOLD) AS TOTAL_UNITS_SOLD
FROM SALES
GROUP BY ITEM_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ITEM_ID	TOTAL_UNITS_SOLD
1	2
2	1
3	1
4	2
5	3
6	1
7	2
8	3
9	1
10	1

Result 16

iv. *Find the top 3 items that generated the highest revenue.*

```sql
SELECT I.ITEM_ID, I.ITEM_NAME, SUM(S.UNITS_SOLD * I.UNIT_PRICE) AS TOTAL_REVENUE
FROM SALES S
JOIN ITEMS I ON S.ITEM_ID = I.ITEM_ID
GROUP BY I.ITEM_ID, I.ITEM_NAME
ORDER BY TOTAL_REVENUE DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

ITEM_ID	ITEM_NAME	TOTAL_REVENUE
7	External Hard Drive	178.00
10	Smartwatch	150.00
5	Bluetooth Speaker	135.00

v. *Display the details of sales made on or after '2024-06-01' sorted by SALE_DATE in descending order.*

```sql
SELECT *FROM SALES
WHERE SALE_DATE >= '2024-06-01'
ORDER BY SALE_DATE DESC;
```

Result Grid | Filter Rows: | Edit:

SALE_ID	CUSTOMER_ID	ITEM_ID	UNITS_SOLD	SALE_DATE
9	203	10	1	2024-06-07
7	202	7	2	2024-06-06
5	201	4	2	2024-06-05
4	204	2	1	2024-06-04
2	202	3	1	2024-06-03
8	206	9	1	2024-06-02
3	203	5	3	2024-06-01
NULL	NULL	NULL	NULL	NULL

SALES 18

vi. *Add a data constraint to ensure that the UNITS_SOLD in the SALES table is always greater than zero.*

```sql
ALTER TABLE SALES
ADD CONSTRAINT chk_units_sold_positive CHECK (UNITS_SOLD > 0);
```

vii. *Calculate the total revenue generated by each customer.*

```
SELECT CUSTOMER_ID, SUM(S.UNITS_SOLD * I.UNIT_PRICE) AS TOTAL_REVENUE
FROM SALES S
JOIN ITEMS I ON S.ITEM_ID = I.ITEM_ID
GROUP BY CUSTOMER_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CUSTOMER_ID	TOTAL_REVENUE
201	120.98
202	197.99
203	285.00
204	169.96
205	59.99
206	120.00

Result 19

Q4. Create a table EMPLOYEES and DEPARTMENTS with the following fields. EMPLOYEES: {EMPLOYEE_ID, NAME, DEPARTMENT_ID, SALARY, JOIN_DATE} DEPARTMENTS: {DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID}

Firstly we create a TABLE.

```
CREATE TABLE DEPARTMENTS (
    DEPARTMENT_ID INT PRIMARY KEY,
    DEPARTMENT_NAME VARCHAR(100),
    MANAGER_ID INT
);

CREATE TABLE EMPLOYEES (
    EMPLOYEE_ID INT PRIMARY KEY,
    NAME VARCHAR(100),
    DEPARTMENT_ID INT,
    SALARY DECIMAL(10, 2) CHECK (SALARY >= 0),
    JOIN_DATE DATE,
    FOREIGN KEY (DEPARTMENT_ID) REFERENCES DEPARTMENTS(DEPARTMENT_ID)
);
```

Perform the following queries in SQL:

i. *Insert at least 5 records into the EMPLOYEES and DEPARTMENTS tables.*

```sql
-- DEPARTMENTS Table
INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID) VALUES
(1, 'Human Resources', 101),
(2, 'Engineering', 102),
(3, 'Marketing', 103),
(4, 'Finance', 104),
(5, 'Legal', 105);

-- EMPLOYEES Table
INSERT INTO EMPLOYEES (EMPLOYEE_ID, NAME, DEPARTMENT_ID, SALARY, JOIN_DATE) VALUES
(1, 'Anjali Rao', 1, 55000.00, '2024-01-10'),
(2, 'Rohit Sharma', 2, 75000.00, '2023-09-15'),
(3, 'Simran Kaur', 3, 62000.00, '2024-03-22'),
(4, 'Karan Patel', 2, 80000.00, '2022-06-05'),
(5, 'Neha Verma', 4, 50000.00, '2024-02-28');
```

ii. *Display the names of all employees.*

```sql
SELECT NAME FROM EMPLOYEES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

NAME
Anjali Rao
Rohit Sharma
Simran Kaur
Karan Patel
Neha Verma

EMPLOYEES 21

iii. *Display the department names and their corresponding manager IDs.*

```sql
SELECT DEPARTMENT_NAME, MANAGER_ID FROM DEPARTMENTS;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DEPARTMENT_NAME	MANAGER_ID
Human Resources	101
Engineering	102
Marketing	103
Finance	104
Legal	105

DEPARTMENTS 22

iv. *Find the employee with the highest salary.*

```
SELECT * FROM EMPLOYEES
ORDER BY SALARY DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Edit: | Export/

EMPLOYEE_ID	NAME	DEPARTMENT_ID	SALARY	JOIN_DATE
4	Karan Patel	2	80000.00	2022-06-05
NULL	NULL	NULL	NULL	NULL

EMPLOYEES 23

v. *Display the details of employees who joined in the year 2024.*

```
SELECT * FROM EMPLOYEES
WHERE YEAR(JOIN_DATE) = 2024;
```

Result Grid | Filter Rows: | Edit: | Export/Import

EMPLOYEE_ID	NAME	DEPARTMENT_ID	SALARY	JOIN_DATE
1	Anjali Rao	1	55000.00	2024-01-10
3	Simran Kaur	3	62000.00	2024-03-22
5	Neha Verma	4	50000.00	2024-02-28
NULL	NULL	NULL	NULL	NULL

EMPLOYEES 24

vi. *Ensure that the SALARY column in the EMPLOYEES table does not accept negative values.*

```
ALTER TABLE EMPLOYEES
ADD CONSTRAINT chk_salary_positive CHECK (SALARY >= 0);
```

Output

Action Output

#	Time	Action	Message
11	15:01:13	SELECT * FROM EMPLOYEES WHERE YEAR(JOIN_DATE) = 2024 LIMIT 0, 1000	3 row(s) returned
12	15:02:45	ALTER TABLE EMPLOYEES ADD CONSTRAINT chk_salary_positive CHECK (SALARY >=...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0

vii. *Calculate the total number of employees.*

```
SELECT COUNT(*) AS TOTAL_EMPLOYEES FROM EMPLOYEES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TOTAL_EMPLOYEES
5

viii. *List the names of employees along with their department names.*

```
SELECT E.NAME AS EMPLOYEE_NAME, D.DEPARTMENT_NAME
FROM EMPLOYEES E
JOIN DEPARTMENTS D ON E.DEPARTMENT_ID = D.DEPARTMENT_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMPLOYEE_NAME	DEPARTMENT_NAME
Anjali Rao	Human Resources
Rohit Sharma	Engineering
Simran Kaur	Marketing
Karan Patel	Engineering
Neha Verma	Finance

Result 26

ix. *Display the details of departments that have no employees.*

```
SELECT D.*FROM DEPARTMENTS D
LEFT JOIN EMPLOYEES E ON D.DEPARTMENT_ID = E.DEPARTMENT_ID
WHERE E.EMPLOYEE_ID IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID
5	Legal	105

x. *Update the salary of an employee with EMPLOYEE_ID 1 to 60000.*

```
UPDATE EMPLOYEES
SET SALARY = 60000
WHERE EMPLOYEE_ID = 1;
```

Output

Action Output

#	Time	Action	Message
16	15:07:42	SELECT D.*FROM DEPARTMENTS D LEFT JOIN EMPLOYEES E ON D.DEPARTMENT I...	1 row(s) returned

Q5. Create a table PATIENTS and APPOINTMENTS with the following fields:

PATIENTS: {PATIENT_ID, NAME, AGE, GENDER}

APPOINTMENTS: {APPOINTMENT_ID, PATIENT_ID, DOCTOR, APPOINTMENT_DATE, FEES}

Firstly we create a TABLE.

```
CREATE TABLE PATIENTS (
    PATIENT_ID INT PRIMARY KEY,
    NAME VARCHAR(100),
    AGE INT,
    GENDER VARCHAR(10)
);

CREATE TABLE APPOINTMENTS (
    APPOINTMENT_ID INT PRIMARY KEY,
    PATIENT_ID INT,
    DOCTOR VARCHAR(100),
    APPOINTMENT_DATE DATE,
    FEES DECIMAL(8, 2) CHECK (FEES >= 0),
    FOREIGN KEY (PATIENT_ID) REFERENCES PATIENTS(PATIENT_ID)
);
```

Perform the following queries in SQL:

i. *Insert at least 5 records into the PATIENTS and APPOINTMENTS tables.*

```sql
-- PATIENTS Table
INSERT INTO PATIENTS (PATIENT_ID, NAME, AGE, GENDER) VALUES
(1, 'Aarav Mehta', 32, 'Male'),
(2, 'Ishita Roy', 28, 'Female'),
(3, 'Kabir Khan', 45, 'Male'),
(4, 'Rhea Sen', 35, 'Female'),
(5, 'Yash Gupta', 52, 'Male');

-- APPOINTMENTS Table
INSERT INTO APPOINTMENTS (APPOINTMENT_ID, PATIENT_ID, DOCTOR, APPOINTMENT_DATE, FEES) VALUES
(101, 1, 'Dr. Smith', '2024-04-12', 500),
(102, 2, 'Dr. Kumar', '2024-03-25', 600),
(103, 3, 'Dr. Smith', '2024-04-28', 700),
(104, 1, 'Dr. Smith', '2024-05-02', 500),
(105, 4, 'Dr. Mehra', '2024-04-15', 550);
```

ii. *Display the names of all patients.*

```sql
SELECT NAME FROM PATIENTS;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

NAME
Aarav Mehta
Ishita Roy
Kabir Khan
Rhea Sen
Yash Gupta

PATIENTS 28

iii. *Display the details of appointments for a particular doctor (e.g., 'Dr. Smith').*

```sql
SELECT * FROM APPOINTMENTS
WHERE DOCTOR = 'Dr. Smith';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

APPOINTMENT_ID	PATIENT_ID	DOCTOR	APPOINTMENT_DATE	FEES
101	1	Dr. Smith	2024-04-12	500.00
103	3	Dr. Smith	2024-04-28	700.00
104	1	Dr. Smith	2024-05-02	500.00
NULL	NULL	NULL	NULL	NULL

APPOINTMENTS 29

iv. *Find the patient with the highest number of appointments.*

```
SELECT P.NAME, COUNT(A.APPOINTMENT_ID) AS APPOINTMENT_COUNT FROM PATIENTS P
JOIN APPOINTMENTS A ON P.PATIENT_ID = A.PATIENT_ID
GROUP BY P.NAME
ORDER BY APPOINTMENT_COUNT DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

NAME	APPOINTMENT_COUNT
Aarav Mehta	2

v. *Display the details of appointments that occurred in the last month.*

```
-- Assuming CURRENT_DATE is '2025-05-04', last month is April 2025
SELECT * FROM APPOINTMENTS
WHERE APPOINTMENT_DATE >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

APPOINTMENT_ID	PATIENT_ID	DOCTOR	APPOINTMENT_DATE	FEES
NULL	NULL	NULL	NULL	NULL

vi. *Ensure that the FEES column in the APPOINTMENTS table does not accept negative values.*

```
ALTER TABLE APPOINTMENTS
ADD CONSTRAINT chk_fees_positive CHECK (FEES >= 0);
```

vii. *Calculate the total number of appointments.*

```
SELECT COUNT(*) AS TOTAL_APPOINTMENTS FROM APPOINTMENTS;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TOTAL_APPOINTMENTS
5

viii. *List the names of patients along with their appointment details.*

```
SELECT
    P.NAME AS PATIENT_NAME,A.APPOINTMENT_ID,A.DOCTOR,A.APPOINTMENT_DATE,A.FEES FROM PATIENTS P
JOIN   APPOINTMENTS A ON P.PATIENT_ID = A.PATIENT_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PATIENT_NAME	APPOINTMENT_ID	DOCTOR	APPOINTMENT_DATE	FEES
Aarav Mehta	101	Dr. Smith	2024-04-12	500.00
Ishita Roy	102	Dr. Kumar	2024-03-25	600.00
Kabir Khan	103	Dr. Smith	2024-04-28	700.00
Aarav Mehta	104	Dr. Smith	2024-05-02	500.00
Rhea Sen	105	Dr. Mehra	2024-04-15	550.00

Result 34

ix. *Display the details of patients who have not had any appointments.*

```
SELECT *FROM PATIENTS P
LEFT JOIN APPOINTMENTS A ON P.PATIENT_ID = A.PATIENT_ID
WHERE A.APPOINTMENT_ID IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PATIENT_ID	NAME	AGE	GENDER	APPOINTMENT_ID	PATIENT_ID	DOCTOR	APPOINTMENT_DATE	FEES
5	Yash Gupta	52	Male	NULL	NULL	NULL	NULL	NULL

x. *Update the appointment date for a specific appointment (e.g., APPOINTMENT_ID 1) to a new Date.*

```
UPDATE APPOINTMENTS
SET APPOINTMENT_DATE = '2025-05-10'
WHERE APPOINTMENT_ID = 1;
```

Output

Action Output

#	Time	Action	Message
32	15:33:15	SELECT *FROM PATIENTS P LEFT JOIN APPOINTMENTS A ON P.PATIENT_ID = A.PAT...	1 row(s) returned

Q6. Create a table EMPLOYEES and SALARIES with the following fields: EMPLOYEES TABLE: {EMPLOYEE_ID, NAME, DEPARTMENT, GENDER, AGE} SALARIES TABLE: {EMPLOYEE_ID, BASE_SALARY, BONUS, DEDUCTIONS}

Firstly we create a TABLE.

```sql
CREATE DATABASE D_file;
use D_file;
CREATE TABLE EMPLOYEES (
    EMPLOYEE_ID INT PRIMARY KEY,
    NAME VARCHAR(100),
    DEPARTMENT VARCHAR(50),
    GENDER VARCHAR(10),
    AGE INT
);

CREATE TABLE SALARIES (
    EMPLOYEE_ID INT PRIMARY KEY,
    BASE_SALARY DECIMAL(10, 2),
    BONUS DECIMAL(10, 2),
    DEDUCTIONS DECIMAL(10, 2),
    FOREIGN KEY (EMPLOYEE_ID) REFERENCES EMPLOYEES(EMPLOYEE_ID)
);
```

Perform the following queries in SQL:

i. *Insert at least 10 records in the tables.*

```sql
INSERT INTO EMPLOYEES (EMPLOYEE_ID, NAME, DEPARTMENT, GENDER, AGE) VALUES
(1, 'Aarav Sharma', 'Finance', 'Male', 34),
(2, 'Neha Verma', 'HR', 'Female', 29),
(3, 'Karan Mehta', 'IT', 'Male', 31),
(4, 'Isha Singh', 'Marketing', 'Female', 27),
(5, 'Rahul Jain', 'Finance', 'Male', 38),
(6, 'Pooja Rao', 'HR', 'Female', 33),
(7, 'Siddharth Das', 'IT', 'Male', 36),
(8, 'Tanya Kapoor', 'Marketing', 'Female', 30),
(9, 'Vikram Batra', 'Finance', 'Male', 42),
(10, 'Ananya Joshi', 'HR', 'Female', 26);
-- SALARIES Table
INSERT INTO SALARIES (EMPLOYEE_ID, BASE_SALARY, BONUS, DEDUCTIONS) VALUES
(1, 60000, 5000, 2000),
(2, 50000, 4000, 1500),
(3, 70000, 6000, 2500),
(4, 55000, 3500, 1000),
(5, 65000, 4500, 2200),
(6, 48000, 3000, 1000),
(7, 73000, 7000, 3000),
(8, 56000, 4000, 1800),
(9, 80000, 8000, 3500),
(10, 47000, 2500, 1200);
```

ii. *Display the details of employees in the 'Finance' department.*

```
SELECT * FROM EMPLOYEES
WHERE DEPARTMENT = 'Finance';
```

Result Grid | Filter Rows: | Edit: | Export/

EMPLOYEE_ID	NAME	DEPARTMENT	GENDER	AGE
1	Aarav Sharma	Finance	Male	34
5	Rahul Jain	Finance	Male	38
9	Vikram Batra	Finance	Male	42
NULL	NULL	NULL	NULL	NULL

iii. *Display the details of the employee with the highest net salary.*

```
SELECT E.*, S.BASE_SALARY + S.BONUS - S.DEDUCTIONS AS NET_SALARY FROM EMPLOYEES E
JOIN SALARIES S ON E.EMPLOYEE_ID = S.EMPLOYEE_ID
ORDER BY NET_SALARY DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

EMPLOYEE_ID	NAME	DEPARTMENT	GENDER	AGE	NET_SALARY
9	Vikram Batra	Finance	Male	42	84500.00

iv. *Add a new column ALLOWANCES in the SALARIES table and modify the table by inserting values to ALLOWANCES for the records.*

```
-- Add column
ALTER TABLE SALARIES
ADD COLUMN ALLOWANCES DECIMAL(10, 2);

-- Update with values
UPDATE SALARIES SET ALLOWANCES = 3000 WHERE EMPLOYEE_ID IN (1, 2, 3);
UPDATE SALARIES SET ALLOWANCES = 2500 WHERE EMPLOYEE_ID IN (4, 5, 6);
UPDATE SALARIES SET ALLOWANCES = 3500 WHERE EMPLOYEE_ID IN (7, 8, 9, 10);
```

Output

Action Output

#	Time	Action	Message
45	15:42:35	UPDATE SALARIES SET ALLOWANCES = 2500 WHERE EMPLOYEE_ID IN (4, 5, 6)	3 row(s) affected Rows matched: 3 Changed: 3 Warnings: 0

v. *List the top 3 employees with the highest total compensation (BASE_SALARY + BONUS + ALLOWANCES - DEDUCTIONS).*

```
-- Update with values
UPDATE SALARIES SET ALLOWANCES = 3000 WHERE EMPLOYEE_ID IN (1, 2, 3);
UPDATE SALARIES SET ALLOWANCES = 2500 WHERE EMPLOYEE_ID IN (4, 5, 6);
UPDATE SALARIES
SET
    ALLOWANCES = 3500
WHERE
    EMPLOYEE_ID IN (7 , 8, 9, 10);
```

Output

Action Output

#	Time	Action	Message
48	15:43:51	UPDATE SALARIES SET ALLOWANCES = 2500 WHERE EMPLOYEE_ID IN (4, 5, 6)	0 row(s) affected Rows matched: 3 Changed: 0 Warnings: 0

vi. *Display the average age of employees in the 'HR' department.*

```
SELECT AVG(AGE) AS AVERAGE_AGE
FROM EMPLOYEES
WHERE DEPARTMENT = 'HR';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVERAGE_AGE
29.3333

Q7. Create a table PROFESSOR and PUBLICATION with the following fields:

PROFESSOR TABLE: {PROFESSOR_ID, NAME, DEPARTMENT, AGE, SALARY}

PUBLICATION TABLE: {PUBLICATION_ID, PROFESSOR_ID, TITLE, JOURNAL, YEAR}

Firstly we create a TABLE.

```sql
CREATE TABLE PROFESSOR (
    PROFESSOR_ID INT PRIMARY KEY,
    NAME VARCHAR(100),
    DEPARTMENT VARCHAR(100),
    AGE INT,
    SALARY DECIMAL(10, 2)
);

CREATE TABLE PUBLICATION (
    PUBLICATION_ID INT PRIMARY KEY,
    PROFESSOR_ID INT,
    TITLE VARCHAR(200),
    JOURNAL VARCHAR(150),
    YEAR INT,
    FOREIGN KEY (PROFESSOR_ID) REFERENCES PROFESSOR(PROFESSOR_ID)
);
```

Perform the following queries in SQL:

i. *Insert at least 10 records into the PROFESSOR and PUBLICATION tables.*

```sql
-- PROFESSOR Table
INSERT INTO PROFESSOR (PROFESSOR_ID, NAME, DEPARTMENT, AGE, SALARY) VALUES
(1, 'Dr. Raghav Sharma', 'Operations Research', 45, 95000),
(2, 'Dr. Meera Nair', 'Mathematics', 50, 98000),
(3, 'Dr. Aarav Joshi', 'Physics', 39, 91000),
(4, 'Dr. Priya Malhotra', 'Mathematics', 42, 97000),
(5, 'Dr. Kabir Anand', 'Computer Science', 37, 99000),
(6, 'Dr. Sneha Kapoor', 'Operations Research', 48, 93000),
(7, 'Dr. Arjun Verma', 'Mathematics', 51, 96000),
(8, 'Dr. Tanya Singh', 'Operations Research', 46, 94000),
(9, 'Dr. Nikhil Sinha', 'Physics', 40, 92000),
(10, 'Dr. Isha Rao', 'Computer Science', 38, 95500);
```

```sql
-- PUBLICATION Table
INSERT INTO PUBLICATION (PUBLICATION_ID, PROFESSOR_ID, TITLE, JOURNAL, YEAR) VALUES
(101, 1, 'Stochastic Models in Supply Chains', 'OR Journal', 2022),
(102, 1, 'Inventory Optimization Techniques', 'Applied OR', 2021),
(103, 2, 'Advanced Algebra Structures', 'Math World', 2020),
(104, 4, 'Number Theory Applications', 'Math World', 2023),
(105, 5, 'AI in Decision Systems', 'Tech Frontier', 2023),
(106, 6, 'Linear Programming Advances', 'OR Journal', 2021),
(107, 7, 'Probability in Education', 'Math Insight', 2022),
(108, 1, 'Game Theory in Markets', 'OR Journal', 2023),
(109, 8, 'Queueing Theory Models', 'Applied OR', 2022),
(110, 2, 'Matrix Theories', 'Math World', 2022);
```

ii. *Display the details of professors from the 'OPERATIONS RESEARCH' department.*

```
SELECT * FROM PROFESSOR
WHERE DEPARTMENT = 'Operations Research';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

PROFESSOR_ID	NAME	DEPARTMENT	AGE	SALARY
1	Dr. Raghav Sharma	Operations Research	45	95000.00
6	Dr. Sneha Kapoor	Operations Research	48	93000.00
8	Dr. Tanya Singh	Operations Research	46	94000.00

iii. *Display the details of the professor with the highest number of publications.*

```
SELECT P.NAME, COUNT(PUB.PUBLICATION_ID) AS TOTAL_PUBLICATIONS
FROM PROFESSOR P
JOIN PUBLICATION PUB ON P.PROFESSOR_ID = PUB.PROFESSOR_ID
GROUP BY P.NAME
ORDER BY TOTAL_PUBLICATIONS DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

NAME	TOTAL_PUBLICATIONS
Dr. Raghav Sharma	3

iv. *Add a new column EMAIL in the PROFESSOR table and update the table by inserting values for EMAIL for the records.*

```sql
-- Add new column
ALTER TABLE PROFESSOR
ADD EMAIL VARCHAR(100);

-- Update email values
UPDATE PROFESSOR SET EMAIL = 'raghav.sharma@univ.edu' WHERE PROFESSOR_ID = 1;
UPDATE PROFESSOR SET EMAIL = 'meera.nair@univ.edu' WHERE PROFESSOR_ID = 2;
UPDATE PROFESSOR SET EMAIL = 'aarav.joshi@univ.edu' WHERE PROFESSOR_ID = 3;
UPDATE PROFESSOR SET EMAIL = 'priya.malhotra@univ.edu' WHERE PROFESSOR_ID = 4;
UPDATE PROFESSOR SET EMAIL = 'kabir.anand@univ.edu' WHERE PROFESSOR_ID = 5;
UPDATE PROFESSOR SET EMAIL = 'sneha.kapoor@univ.edu' WHERE PROFESSOR_ID = 6;
UPDATE PROFESSOR SET EMAIL = 'arjun.verma@univ.edu' WHERE PROFESSOR_ID = 7;
UPDATE PROFESSOR SET EMAIL = 'tanya.singh@univ.edu' WHERE PROFESSOR_ID = 8;
UPDATE PROFESSOR SET EMAIL = 'nikhil.sinha@univ.edu' WHERE PROFESSOR_ID = 9;
UPDATE PROFESSOR SET EMAIL = 'isha.rao@univ.edu' WHERE PROFESSOR_ID = 10;
```

Output

Action Output

#	Time	Action	Message
67	15:52:59	UPDATE PROFESSOR SET EMAIL = 'nikhil.sinha@univ.edu' WHERE PROFESSOR_ID = 9	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

v. *List the top 3 journals with the highest number of publications.*

```sql
SELECT JOURNAL, COUNT(*) AS NUM_PUBLICATIONS
FROM PUBLICATION
GROUP BY JOURNAL
ORDER BY NUM_PUBLICATIONS DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch r

JOURNAL	NUM_PUBLICATIONS
OR Journal	3
Math World	3
Applied OR	2

vi. *Display the average salary of professors in the 'MATHEMATICS' department.*

```sql
SELECT AVG(SALARY) AS AVERAGE_SALARY
FROM PROFESSOR
WHERE DEPARTMENT = 'Mathematics';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVERAGE_SALARY
97000.000000

8. *Create a table CUSTOMER and TRANSACTION with the following fields:*

CUSTOMER TABLE: {CUSTOMER_ID, NAME, AGE, CITY, PHONE_NUMBER}

TRANSACTION TABLE: {TRANSACTION_ID, CUSTOMER_ID, AMOUNT, TRANSACTION_DATE, TRANSACTION_TYPE}

Firstly we create a TABLE.

```sql
CREATE TABLE CUSTOMER (
    CUSTOMER_ID INT PRIMARY KEY,
    NAME VARCHAR(100),
    AGE INT,
    CITY VARCHAR(50),
    PHONE_NUMBER VARCHAR(15)
);

CREATE TABLE TRANSACTION (
    TRANSACTION_ID INT PRIMARY KEY,
    CUSTOMER_ID INT,
    AMOUNT DECIMAL(10, 2),
    TRANSACTION_DATE DATE,
    TRANSACTION_TYPE VARCHAR(20), -- 'Deposit' or 'Withdrawal'
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID)
);
```

Output

Action Output

#	Time	Action	Message
72	16:05:10	CREATE TABLE CUSTOMER (CUSTOMER_ID INT PRIMARY KEY, NAME VARCHAR(10...	0 row(s) affected

Perform the following queries in SQL:

i. *Insert at least 10 records into the CUSTOMER and TRANSACTION tables.*

```sql
-- CUSTOMER Table
INSERT INTO CUSTOMER (CUSTOMER_ID, NAME, AGE, CITY, PHONE_NUMBER) VALUES
(1, 'John Carter', 35, 'New York', '123-456-7890'),
(2, 'Emma Watson', 29, 'Los Angeles', '234-567-8901'),
(3, 'Michael Brown', 40, 'Chicago', '345-678-9012'),
(4, 'Sophia Davis', 31, 'New York', '456-789-0123'),
(5, 'Daniel Lee', 45, 'Houston', '567-890-1234'),
(6, 'Olivia Wilson', 27, 'San Francisco', '678-901-2345'),
(7, 'Liam Martinez', 33, 'New York', '789-012-3456'),
(8, 'Ava Thompson', 38, 'Miami', '890-123-4567'),
(9, 'Ethan White', 41, 'Seattle', '901-234-5678'),
(10, 'Isabella Harris', 36, 'Boston', '012-345-6789');
```

```sql
-- TRANSACTION Table
INSERT INTO TRANSACTION (TRANSACTION_ID, CUSTOMER_ID, AMOUNT, TRANSACTION_DATE, TRANSACTION_TYPE) VALUES
(101, 1, 1500.00, '2024-06-01', 'Deposit'),
(102, 2, 700.00, '2024-06-03', 'Withdrawal'),
(103, 3, 2100.00, '2024-06-05', 'Deposit'),
(104, 4, 900.00, '2024-06-06', 'Deposit'),
(105, 5, 450.00, '2024-06-07', 'Withdrawal'),
(106, 1, 2000.00, '2024-06-08', 'Deposit'),
(107, 7, 1200.00, '2024-06-09', 'Deposit'),
(108, 8, 800.00, '2024-06-10', 'Withdrawal'),
(109, 1, 1800.00, '2024-06-11', 'Deposit'),
(110, 4, 300.00, '2024-06-12', 'Deposit');
```

ii. *Display the details of customers living in 'New York'.*

```sql
SELECT * FROM CUSTOMER
WHERE CITY = 'New York';
```

Result Grid | Filter Rows: | Edit:

CUSTOMER_ID	NAME	AGE	CITY	PHONE_NUMBER
1	John Carter	35	New York	123-456-7890
4	Sophia Davis	31	New York	456-789-0123
7	Liam Martinez	33	New York	789-012-3456
NULL	NULL	NULL	NULL	NULL

iii. *Display the details of the customer who made the highest total transaction amount.*

```
SELECT C.NAME, SUM(T.AMOUNT) AS TOTAL_AMOUNT FROM CUSTOMER C
JOIN TRANSACTION T ON C.CUSTOMER_ID = T.CUSTOMER_ID
GROUP BY C.NAME
ORDER BY TOTAL_AMOUNT DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

NAME	TOTAL_AMOUNT
John Carter	5300.00

iv. *Add a new column EMAIL in the CUSTOMER table and update the table by inserting values for EMAIL for the records.*

```
-- Add column
ALTER TABLE CUSTOMER
ADD EMAIL VARCHAR(100);

-- Update values
UPDATE CUSTOMER SET EMAIL = 'john.carter@email.com' WHERE CUSTOMER_ID = 1;
UPDATE CUSTOMER SET EMAIL = 'emma.watson@email.com' WHERE CUSTOMER_ID = 2;
UPDATE CUSTOMER SET EMAIL = 'michael.brown@email.com' WHERE CUSTOMER_ID = 3;
UPDATE CUSTOMER SET EMAIL = 'sophia.davis@email.com' WHERE CUSTOMER_ID = 4;
UPDATE CUSTOMER SET EMAIL = 'daniel.lee@email.com' WHERE CUSTOMER_ID = 5;
UPDATE CUSTOMER SET EMAIL = 'olivia.wilson@email.com' WHERE CUSTOMER_ID = 6;
UPDATE CUSTOMER SET EMAIL = 'liam.martinez@email.com' WHERE CUSTOMER_ID = 7;
UPDATE CUSTOMER SET EMAIL = 'ava.thompson@email.com' WHERE CUSTOMER_ID = 8;
UPDATE CUSTOMER SET EMAIL = 'ethan.white@email.com' WHERE CUSTOMER_ID = 9;
UPDATE CUSTOMER SET EMAIL = 'isabella.harris@email.com' WHERE CUSTOMER_ID = 10;
```

Output

Action Output

#	Time	Action	Message
87	16:12:37	UPDATE CUSTOMER SET EMAIL = 'ethan.white@email.com' WHERE CUSTOMER_ID = 9	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

v. *List the top 3 customers based on the number of transactions they have made.*

```
SELECT C.NAME, COUNT(T.TRANSACTION_ID) AS NUM_TRANSACTIONS FROM CUSTOMER C
JOIN TRANSACTION T ON C.CUSTOMER_ID = T.CUSTOMER_ID
GROUP BY C.NAME
ORDER BY NUM_TRANSACTIONS DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

NAME	NUM_TRANSACTIONS
John Carter	3
Sophia Davis	2
Emma Watson	1

vi. *Display the average transaction amount of 'Deposit' transactions.*

```
SELECT AVG(AMOUNT) AS AVERAGE_DEPOSIT
FROM TRANSACTION
WHERE TRANSACTION_TYPE = 'Deposit';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVERAGE_DEPOSIT
1400.000000

vii. *Display the list of customers who have made transactions totalling more than $5000.*

```
SELECT C.NAME, SUM(T.AMOUNT) AS TOTAL_AMOUNT FROM CUSTOMER C
JOIN TRANSACTION T ON C.CUSTOMER_ID = T.CUSTOMER_ID
GROUP BY C.NAME
HAVING TOTAL_AMOUNT > 5000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

NAME	TOTAL_AMOUNT
John Carter	5300.00

Q9. Create a table BOOK and BORROW with the following fields:

BOOK TABLE: {BOOK_ID, TITLE, AUTHOR, GENRE,

PUBLISHED_YEAR}

BORROW TABLE: {BORROW_ID, BOOK_ID, MEMBER_ID, BORROW_DATE, RETURN_DATE}

First we create a TABLE.

```sql
CREATE TABLE BOOK (
    BOOK_ID INT PRIMARY KEY,
    TITLE VARCHAR(150),
    AUTHOR VARCHAR(100),
    GENRE VARCHAR(50),
    PUBLISHED_YEAR INT
);

CREATE TABLE BORROW (
    BORROW_ID INT PRIMARY KEY,
    BOOK_ID INT,
    MEMBER_ID INT,
    BORROW_DATE DATE,
    RETURN_DATE DATE,
    FOREIGN KEY (BOOK_ID) REFERENCES BOOK(BOOK_ID)
);
```

Output

Action Output

#	Time	Action	Message
92	16:18:30	CREATE TABLE BOOK (BOOK_ID INT PRIMARY KEY, TITLE VARCHAR(150), AUTH...	0 row(s) affected

Perform the following queries in SQL:

i. *Insert at least 10 records into the BOOK and BORROW tables.*

```sql
-- BOOK table
INSERT INTO BOOK (BOOK_ID, TITLE, AUTHOR, GENRE, PUBLISHED_YEAR) VALUES
(1, 'Harry Potter and the Sorcerer\'s Stone', 'J.K. Rowling', 'Fantasy', 1997),
(2, 'Harry Potter and the Chamber of Secrets', 'J.K. Rowling', 'Fantasy', 1998),
(3, 'The Hobbit', 'J.R.R. Tolkien', 'Fantasy', 1937),
(4, 'To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960),
(5, '1984', 'George Orwell', 'Dystopian', 1949),
(6, 'The Da Vinci Code', 'Dan Brown', 'Thriller', 2003),
(7, 'Angels & Demons', 'Dan Brown', 'Thriller', 2000),
(8, 'Pride and Prejudice', 'Jane Austen', 'Romance', 1813),
(9, 'The Alchemist', 'Paulo Coelho', 'Fiction', 1988),
(10, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Classic', 1925);
```

```sql
-- BORROW table
INSERT INTO BORROW (BORROW_ID, BOOK_ID, MEMBER_ID, BORROW_DATE, RETURN_DATE) VALUES
(101, 1, 201, '2024-06-01', '2024-06-10'),
(102, 2, 202, '2024-06-02', '2024-06-12'),
(103, 3, 203, '2024-06-03', '2024-06-08'),
(104, 1, 201, '2024-06-15', '2024-06-20'),
(105, 4, 204, '2024-06-05', '2024-06-15'),
(106, 2, 202, '2024-06-18', '2024-06-25'),
(107, 5, 205, '2024-06-10', '2024-06-17'),
(108, 6, 201, '2024-06-12', '2024-06-22'),
(109, 1, 206, '2024-06-13', '2024-06-18'),
(110, 7, 207, '2024-06-20', '2024-06-27');
```

ii. *Display the details of books authored by 'J.K. Rowling'.*

```sql
SELECT * FROM BOOK WHERE AUTHOR = 'J.K. Rowling';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

BOOK_ID	TITLE	AUTHOR	GENRE	PUBLISHED_YEAR
1	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	1997
2	Harry Potter and the Chamber of Secrets	J.K. Rowling	Fantasy	1998
NULL	NULL	NULL	NULL	NULL

iii. *Display the details of the book that has been borrowed the most times.*

```sql
SELECT B.TITLE, COUNT(*) AS TIMES_BORROWED FROM BOOK B
JOIN BORROW BR ON B.BOOK_ID = BR.BOOK_ID
GROUP BY B.TITLE
ORDER BY TIMES_BORROWED DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

TITLE	TIMES_BORROWED
Harry Potter and the Sorcerer's Stone	3

iv. *Add a new column ISBN in the BOOK table and update the table by inserting values for ISBN for the records.*

```
-- Add new column
ALTER TABLE BOOK
ADD ISBN VARCHAR(20);

-- Update values
UPDATE BOOK SET ISBN = '9780747532699' WHERE BOOK_ID = 1;
UPDATE BOOK SET ISBN = '9780747538493' WHERE BOOK_ID = 2;
UPDATE BOOK SET ISBN = '9780547928227' WHERE BOOK_ID = 3;
UPDATE BOOK SET ISBN = '9780061120084' WHERE BOOK_ID = 4;
UPDATE BOOK SET ISBN = '9780451524935' WHERE BOOK_ID = 5;
UPDATE BOOK SET ISBN = '9780307474278' WHERE BOOK_ID = 6;
UPDATE BOOK SET ISBN = '9780743493468' WHERE BOOK_ID = 7;
UPDATE BOOK SET ISBN = '9780141439518' WHERE BOOK_ID = 8;
UPDATE BOOK SET ISBN = '9780061122415' WHERE BOOK_ID = 9;
UPDATE BOOK SET ISBN = '9780743273565' WHERE BOOK_ID = 10;
```

Output

Action Output

#	Time	Action	Message
107	16:22:21	UPDATE BOOK SET ISBN = '9780061122415' WHERE BOOK_ID = 9	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

v. *List the top 3 members based on the number of books they have borrowed.*

```
SELECT MEMBER_ID, COUNT(*) AS TOTAL_BORROWS FROM BORROW
GROUP BY MEMBER_ID
ORDER BY TOTAL_BORROWS DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

MEMBER_ID	TOTAL_BORROWS
201	3
202	2
203	1

vi. *Display the average number of days books are borrowed for.*

```
SELECT AVG(DATEDIFF(RETURN_DATE, BORROW_DATE)) AS AVG_BORROW_DAYS FROM BORROW;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVG_BORROW_DAYS
7.5000

vii. *Display the list of books borrowed by members who have borrowed more than 5 books.*

```
SELECT DISTINCT B.* FROM BOOK B
JOIN BORROW BR ON B.BOOK_ID = BR.BOOK_ID
WHERE BR.MEMBER_ID IN (
    SELECT MEMBER_ID
    FROM BORROW
    GROUP BY MEMBER_ID
    HAVING COUNT(*) > 5
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

BOOK_ID	TITLE	AUTHOR	GENRE	PUBLISHED_YEAR	ISBN

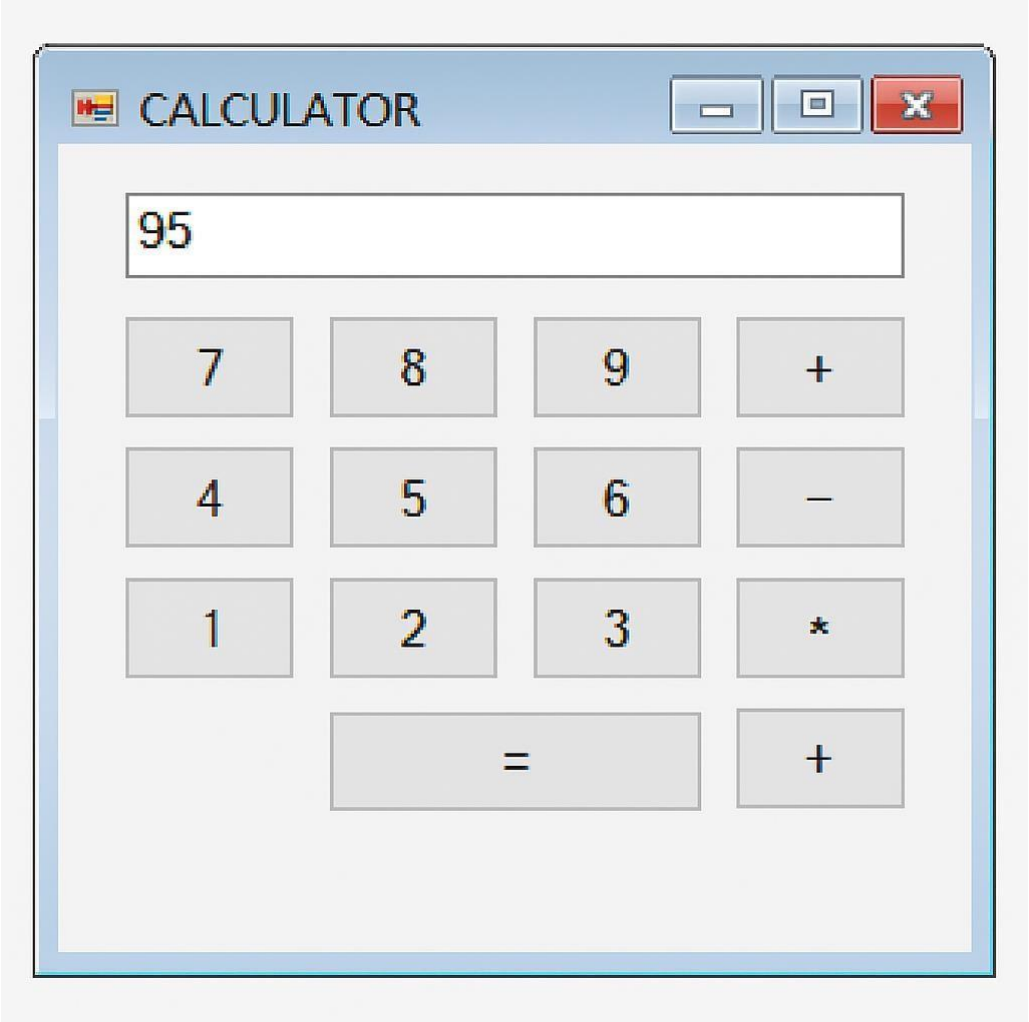
VISUAL BASIC PROGRAM

Write a program to make CALCULATOR by using VISUAL BASIC .

```
Public Class Form1
    Dim firstNum As Double
    Dim secondNum As Double
    Dim operation As String

    Private Sub btn_Click(sender As Object, e As EventArgs) Handles btnAdd.Click, btnSub.Click, btnMul.Click, btnDiv.Click
        firstNum = Val(txtFirst.Text)
        secondNum = Val(txtSecond.Text)

        Select Case CType(sender, Button).Text
            Case "+"
                lblResult.Text = "Result: " & (firstNum + secondNum)
            Case "-"
                lblResult.Text = "Result: " & (firstNum - secondNum)
            Case "*"
                lblResult.Text = "Result: " & (firstNum * secondNum)
            Case "/"
                If secondNum = 0 Then
                    lblResult.Text = "Cannot divide by zero."
                Else
                    lblResult.Text = "Result: " & (firstNum / secondNum)
                End If
        End Select
    End Sub
End Class
```

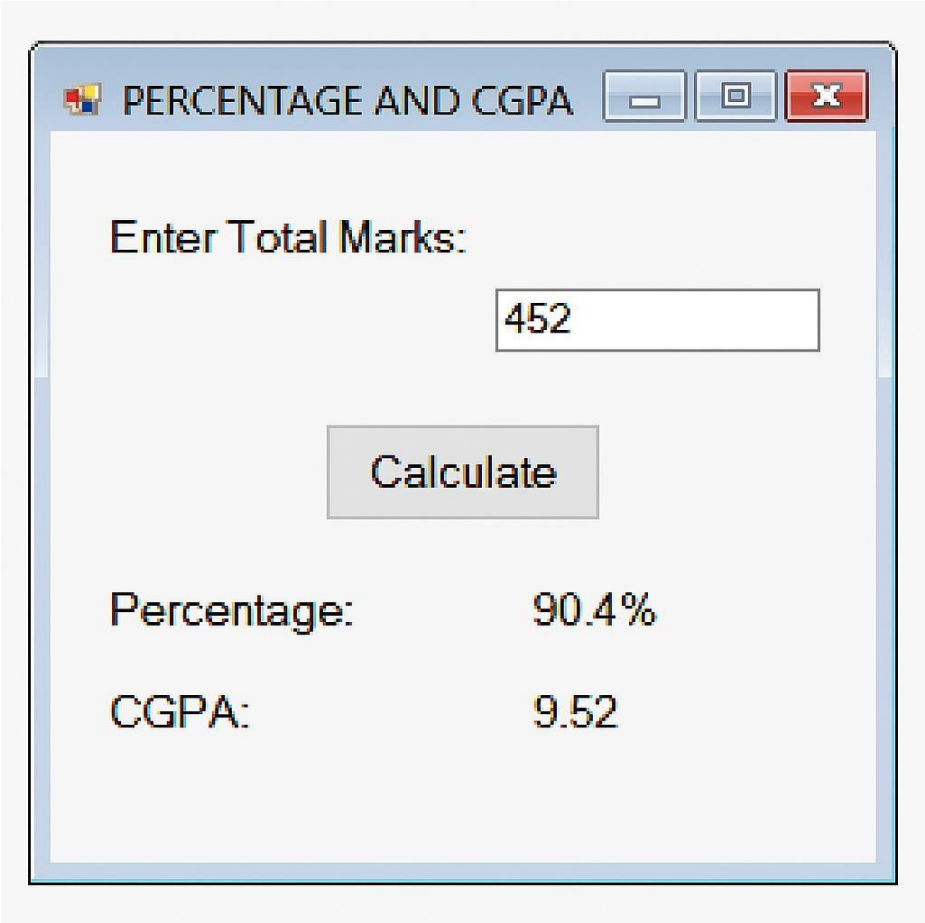


10. Write a program to calculate the PERCENTAGE and CGPA by using VISUAL BASIC.

```
Public Class Form1
    Private Sub btnCalculate_Click(sender As Object, e As EventArgs) Handles btnCalculate.Click
        Dim totalMarks As Double
        Dim obtainedMarks As Double
        Dim percentage As Double
        Dim cgpa As Double

        If Double.TryParse(txtTotal.Text, totalMarks) AndAlso Double.TryParse(txtObtained.Text, obtainedMarks)
            percentage = (obtainedMarks / totalMarks) * 100
            cgpa = percentage / 9.5 ' Common CGPA conversion scale

            lblPercentage.Text = "Percentage: " & Math.Round(percentage, 2) & "%"
            lblCGPA.Text = "CGPA: " & Math.Round(cgpa, 2)
        Else
            MessageBox.Show("Please enter valid numeric values.")
        End If
    End Sub
End Class
```



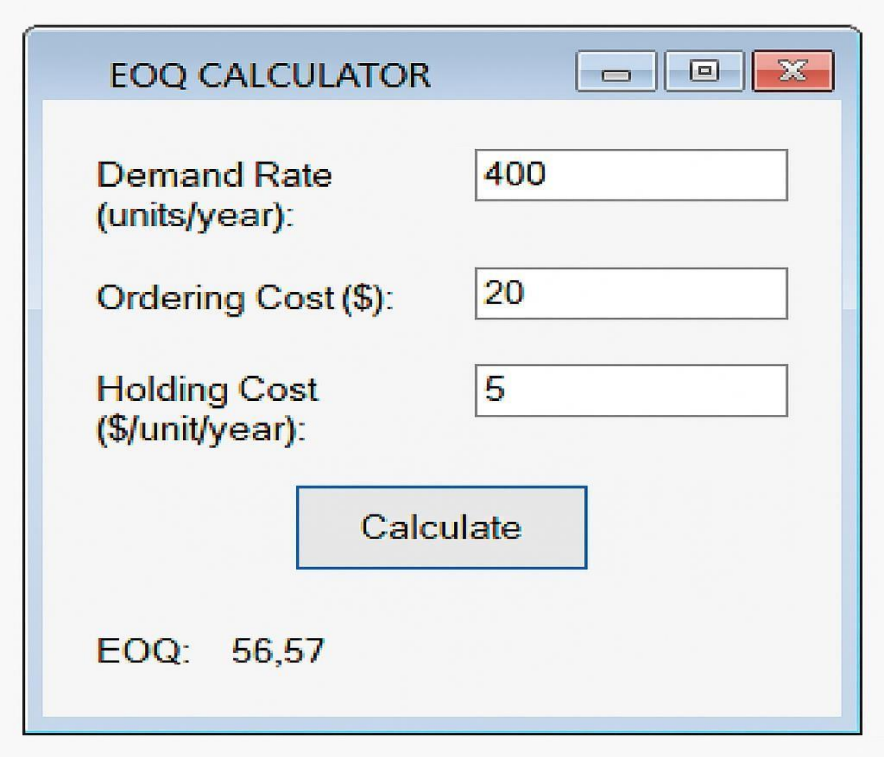
11. Write a program to calculate first three model of EOQ by using VISUAL BASIC.

```
Public Class Form1
    Private Sub btnCalculate_Click(sender As Object, e As EventArgs) Handles btnCalculate.Click
        Dim D As Double, S As Double, H As Double
        If Double.TryParse(txtDemand.Text, D) AndAlso Double.TryParse(txtOrdering.Text, S) AndAlso Double.TryParse(txtHolding.Text, H) Then
            ' 1. Basic EOQ Model
            Dim EOQ_Basic As Double = Math.Sqrt((2 * D * S) / H)

            ' 2. EOQ with Discount - Assume lower holding cost
            Dim EOQ_Discount As Double = Math.Sqrt((2 * D * S) / (H * 0.8))

            ' 3. EOQ with Shortages - Simplified (50% of holding cost)
            Dim EOQ_Shortages As Double = Math.Sqrt((2 * D * S) / (H * 0.5))

            lblEOQ1.Text = "Basic EOQ: " & Math.Round(EOQ_Basic, 2)
            lblEOQ2.Text = "EOQ with Discount: " & Math.Round(EOQ_Discount, 2)
            lblEOQ3.Text = "EOQ with Shortages: " & Math.Round(EOQ_Shortages, 2)
        Else
            MessageBox.Show("Please enter valid numeric values.")
        End If
    End Sub
End Class
```



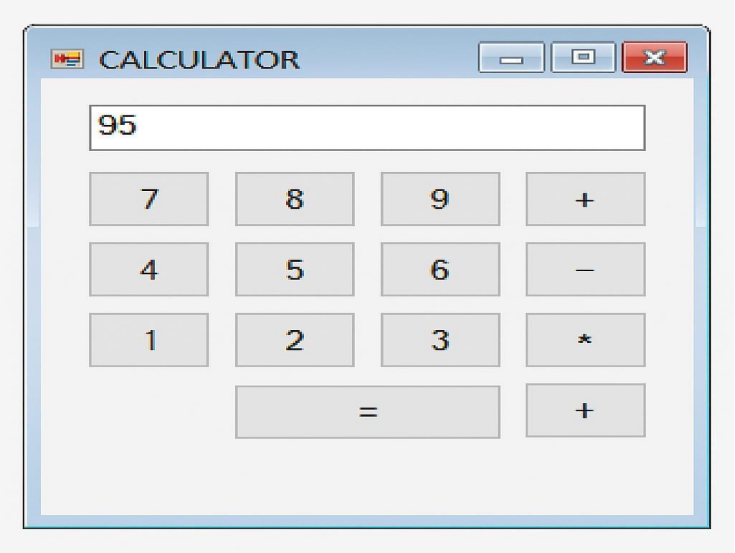
12. *Write a program to perform ARITHMETIC OPERATION on two numbers by using VISUAL BASIC.*

```
Public Class Form1
  Private Sub btnCalculate_Click(sender As Object, e As EventArgs) Handles btnCalculate.Click
      Dim num1 As Double
      Dim num2 As Double
      Dim result As Double

      ' Error Handling
      If Not Double.TryParse(txtNum1.Text, num1) Or Not Double.TryParse(txtNum2.Text, num2) Then
          MessageBox.Show("Please enter valid numbers.")
          Exit Sub
      End If

      Select Case cmbOperation.Text
          Case "Addition"
              result = num1 + num2
          Case "Subtraction"
              result = num1 - num2
          Case "Multiplication"
              result = num1 * num2
          Case "Division"
              If num2 = 0 Then
                  MessageBox.Show("Cannot divide by zero.")
                  Exit Sub
              End If
              result = num1 / num2
          Case Else
              MessageBox.Show("Please select an operation.")
              Exit Sub
      End Select

      lblResult.Text = "Result: " & result.ToString()
  End Sub
End Class
```



13. Write a program to find the SIMPLE and COMPOUND INTEREST by using VISUAL BASIC.

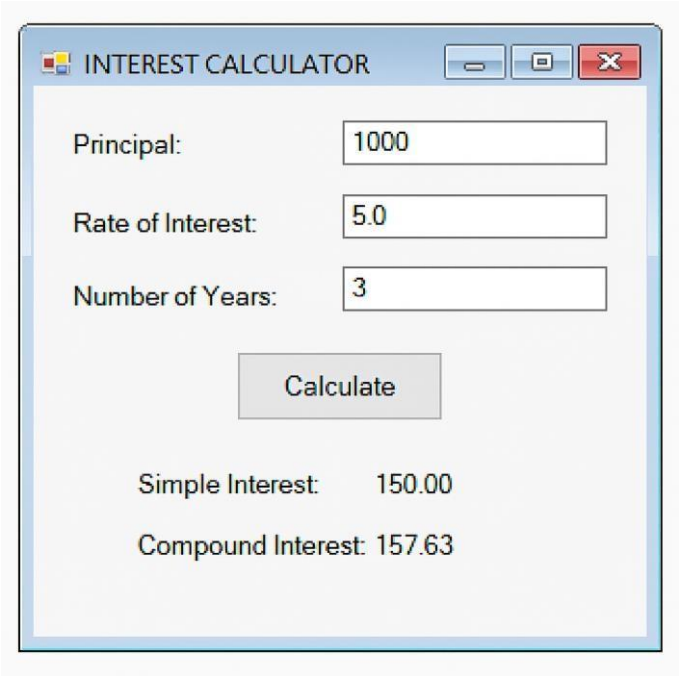
```
Public Class Form1
    Private Sub btnCalculate_Click(sender As Object, e As EventArgs) Handles btnCalculate.Click
        Dim principal As Double
        Dim rate As Double
        Dim time As Double
        Dim simpleInterest As Double
        Dim compoundInterest As Double

        ' Input validation
        If Not Double.TryParse(txtPrincipal.Text, principal) OrElse
            Not Double.TryParse(txtRate.Text, rate) OrElse
            Not Double.TryParse(txtTime.Text, time) Then
            MessageBox.Show("Please enter valid numeric values.")
            Exit Sub
        End If

        ' Simple Interest
        simpleInterest = (principal * rate * time) / 100

        ' Compound Interest
        compoundInterest = principal * (Math.Pow((1 + rate / 100), time)) - principal

        ' Output
        lblSimple.Text = "Simple Interest: " & Math.Round(simpleInterest, 2)
        lblCompound.Text = "Compound Interest: " & Math.Round(compoundInterest, 2)
    End Sub
End Class
```



14. Write a program to calculate AGE by using VISUAL BASIC.

```
Public Class Form1
    Private Sub btnCalculate_Click(sender As Object, e As EventArgs) Handles btnCalculate.Click
        Dim dob As Date
        Dim today As Date = Date.Today
        Dim age As Integer

        If Date.TryParse(txtDOB.Text, dob) Then
            age = today.Year - dob.Year
            If today < dob.AddYears(age) Then
                age -= 1
            End If
            lblAge.Text = "Age: " & age.ToString()
        Else
            MessageBox.Show("Please enter a valid date (e.g. 01/01/2000).")
        End If
    End Sub
End Class
```