Assignment 1

Consider the given database schema: student (student-name, street, city)

Enrolls (student-name, college-name, scholarship)

College (college- name , city)

Mentors (student-name, mentor name)

To run the SQL queries, we first need to:

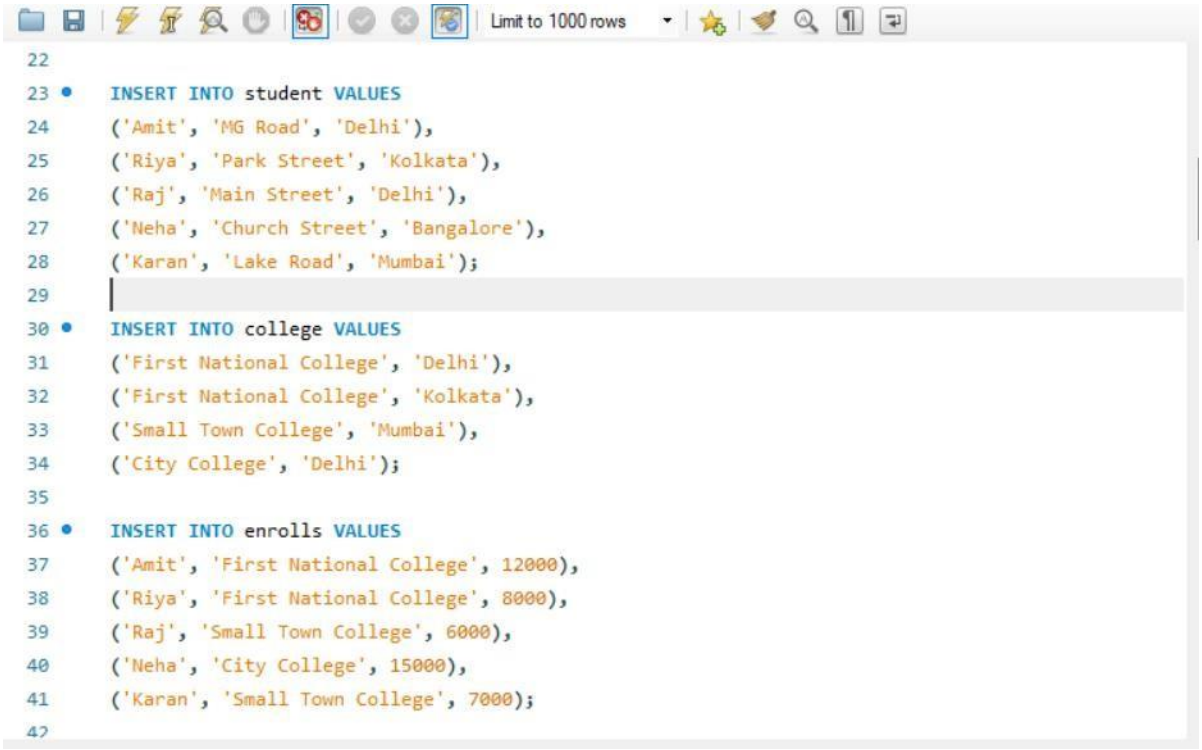
1. **Create a database → so all our tables and data stay in one place.**
2. **Create the required tables → student, enrolls, college, mentors.**
3. **Insert sample data → to test the queries properly.**
4. **Run the queries → to get the answers for the given questions.**

Step 1: Create Database

Step 2: Create Tables

```
1  CREATE TABLE student (
2      student_name VARCHAR(50),
3      street VARCHAR(50),
4      city VARCHAR(50)
5  );
6
7  CREATE TABLE college (
8      college_name VARCHAR(50),
9      city VARCHAR(50)
10 );
11
12 CREATE TABLE enrolls (
13     student_name VARCHAR(50),
14     college_name VARCHAR(50),
15     scholarship INT
16 );
17
18 CREATE TABLE mentors (
19     student_name VARCHAR(50),
20     mentor_name VARCHAR(50)
21 );
```

Step 3: Insert Sample Data

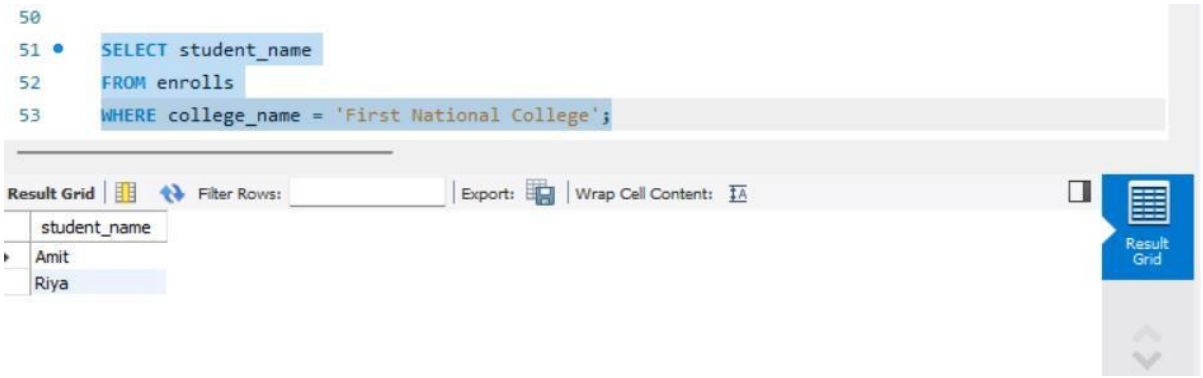


```
INSERT INTO student VALUES
('Amit', 'MG Road', 'Delhi'),
('Riya', 'Park Street', 'Kolkata'),
('Raj', 'Main Street', 'Delhi'),
('Neha', 'Church Street', 'Bangalore'),
('Karan', 'Lake Road', 'Mumbai');

INSERT INTO college VALUES
('First National College', 'Delhi'),
('First National College', 'Kolkata'),
('Small Town College', 'Mumbai'),
('City College', 'Delhi');

INSERT INTO enrolls VALUES
('Amit', 'First National College', 12000),
('Riya', 'First National College', 8000),
('Raj', 'Small Town College', 6000),
('Neha', 'City College', 15000),
('Karan', 'Small Town College', 7000);
```

```
INSERT INTO mentors VALUES
('Amit', 'Raj'),
('Riya', 'Neha'),
('Karan', 'Amit');
```

Assignment Queries

Q1. Names of students enrolled in First National College



```
SELECT student_name
FROM enrolls
WHERE college_name = 'First National College';
```

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

student_name
Amit
Riya

Q2. Names and cities of student enrolled in first national college

```
63 •  SELECT s.student_name, s.city
64    FROM student s
65    JOIN enrolls e ON s.student_name = e.student_name
66    WHERE e.college_name = 'First National College';
67
68
```

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

student_name	city
Amit	Delhi
Riya	Kolkata

Result 15 | Read Only

Q3. Students in first national college with scholarship>10000

Limit to 1000 rows

```
67
68
69
70 •  SELECT s.student_name, s.street, s.city
71    FROM student s
72    JOIN enrolls e ON s.student_name = e.student_name
73    WHERE e.college_name = 'First National College'
74      AND e.scholarship > 10000;
75
76
77
78
79
```

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

student_name	street	city
Amit	MG Road	Delhi

Result 16 | Read Only

Q4 students who live in the same city as their college

```
85
86 •  SELECT DISTINCT s.student_name
87    FROM student s
88    JOIN enrolls e ON s.student_name = e.student_name
89    JOIN college c ON e.college_name = c.college_name
90    WHERE s.city = c.city;
```

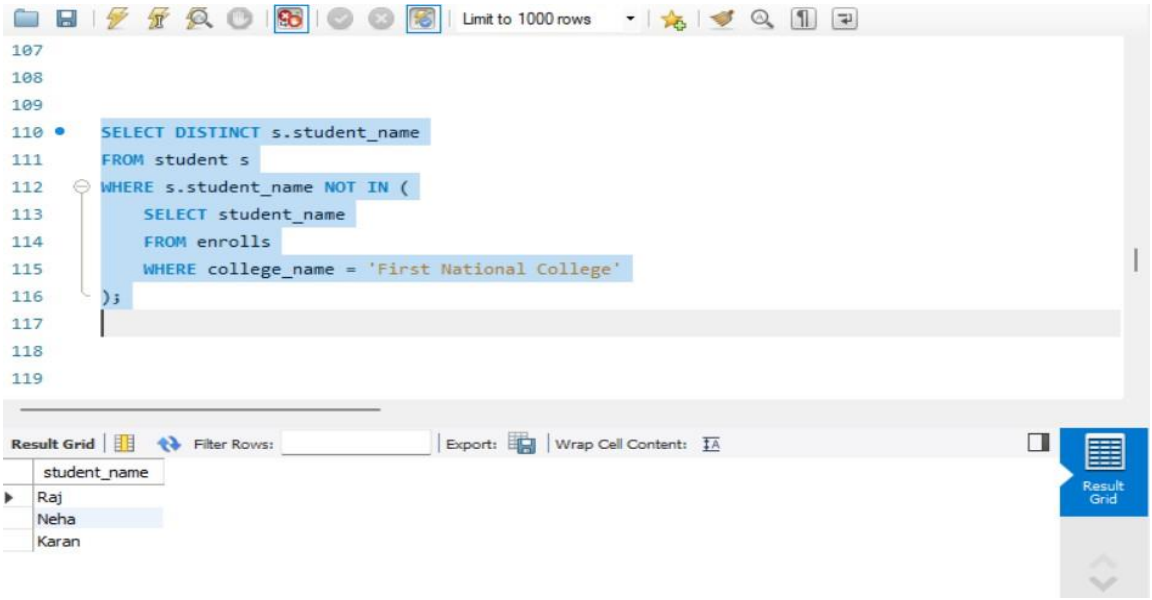
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_name
arpita

Q5. Students who live in the same city and street as their mentors

```
97
98
99 •  SELECT s.student_name
00    FROM student s
01    JOIN mentors m ON s.student_name = m.student_name
02    JOIN student mentor ON m.mentor_name = mentor.student_name
03    WHERE s.city = mentor.city
04      AND s.street = mentor.street;
05
```

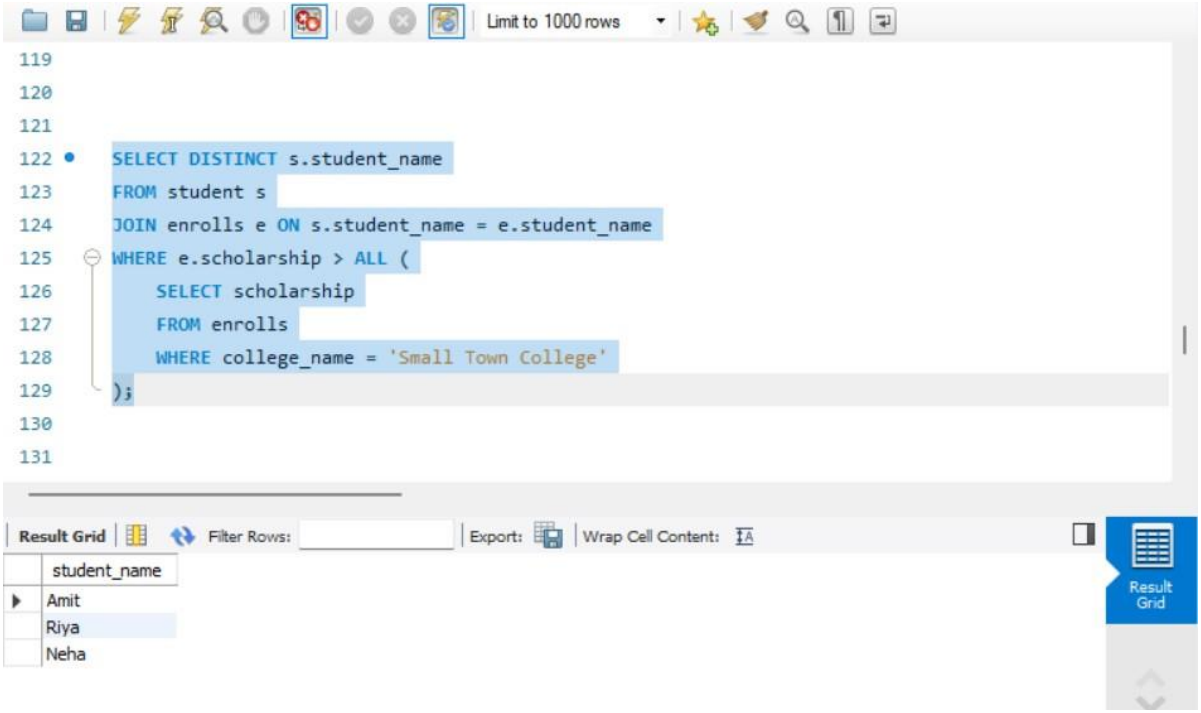
Q6. Students not enrolled in first national college



```
SELECT DISTINCT s.student_name
FROM student s
WHERE s.student_name NOT IN (
    SELECT student_name
    FROM enrolls
    WHERE college_name = 'First National College'
);
```

student_name
Raj
Neha
Karan

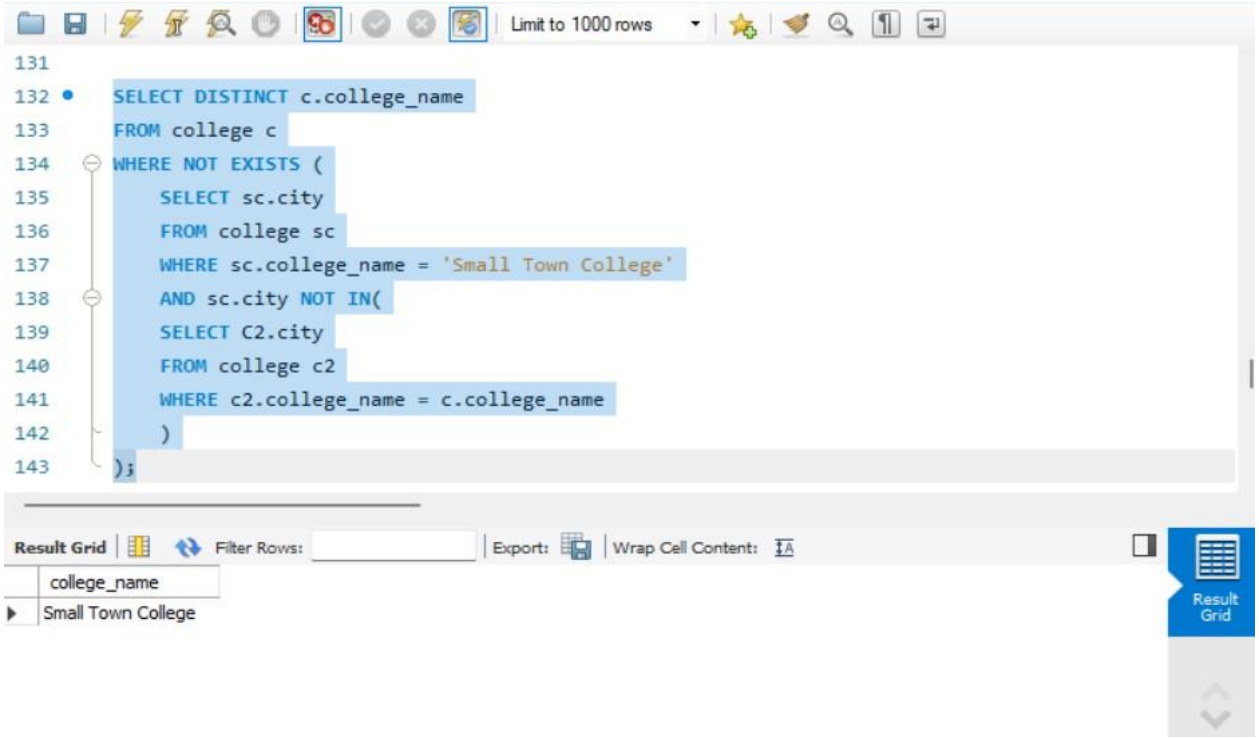
Q7students with scholarship greater than every student in small town college



```
SELECT DISTINCT s.student_name
FROM student s
JOIN enrolls e ON s.student_name = e.student_name
WHERE e.scholarship > ALL (
    SELECT scholarship
    FROM enrolls
    WHERE college_name = 'Small Town College'
);
```

student_name
Amit
Riya
Neha

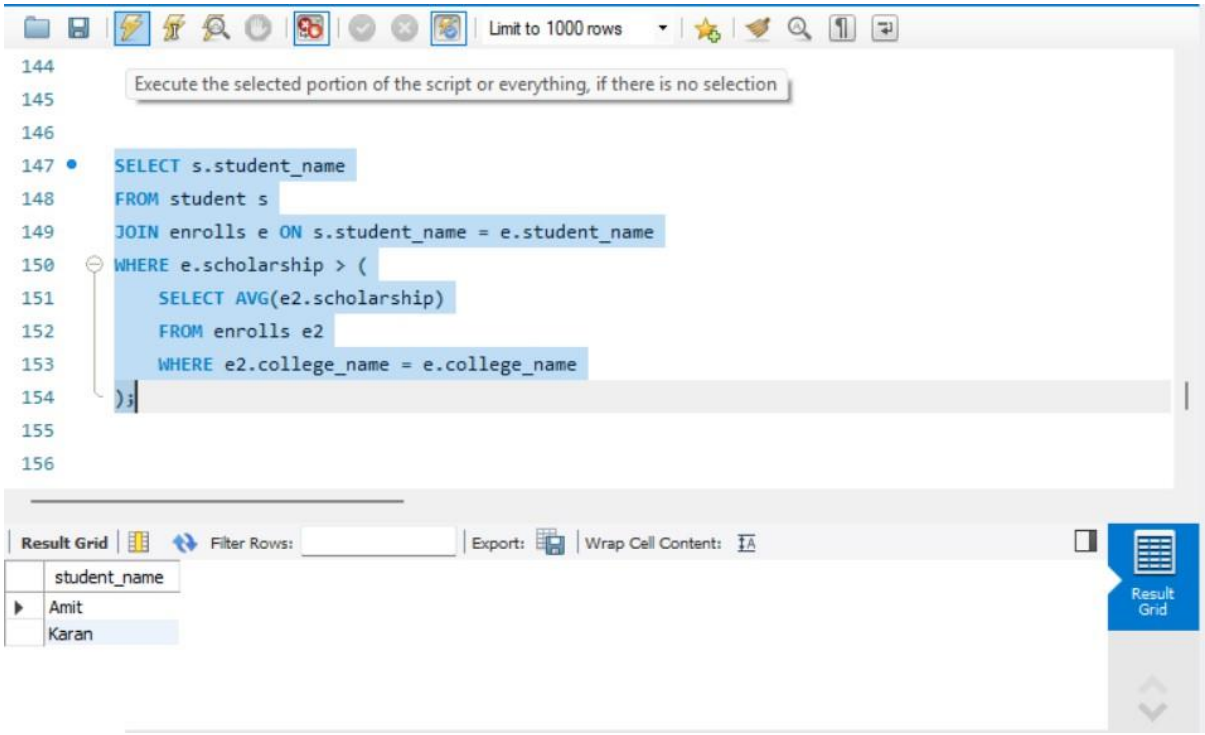
Q8. College located in every city where small town college is located



```
SELECT DISTINCT c.college_name
FROM college c
WHERE NOT EXISTS (
    SELECT sc.city
    FROM college sc
    WHERE sc.college_name = 'Small Town College'
    AND sc.city NOT IN(
    SELECT C2.city
    FROM college c2
    WHERE c2.college_name = c.college_name
    )
);
```

college_name
Small Town College

Q9. Students with scholarship greater than average at their college



```
SELECT s.student_name
FROM student s
JOIN enrolls e ON s.student_name = e.student_name
WHERE e.scholarship > (
    SELECT AVG(e2.scholarship)
    FROM enrolls e2
    WHERE e2.college_name = e.college_name
);
```

student_name
Amit
Karan

Q10. College with most student enrolled

```
158
159 •  SELECT e.college_name
160    FROM enrolls e
161    GROUP BY e.college_name
162    ORDER BY COUNT(e.student_name) DESC
163    LIMIT 1;
164
165
```

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

college_name
First National College

Result Grid

Q11. College with smallest total scholar ship payout

```
171 •  SELECT e.college_name
172    FROM enrolls e
173    GROUP BY e.college_name
174    ORDER BY SUM(e.scholarship) ASC
175    LIMIT 1;
176
177
178
179
```

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

college_name
Small Town College

Result Grid

Q12.college with avg scholarship >avg at first national college

```
182 •  SELECT e.college_name
183    FROM enrolls e
184    GROUP BY e.college_name
185    HAVING AVG(e.scholarship) > (
186        SELECT AVG(scholarship)
187        FROM enrolls
188        WHERE college_name = 'First National College'
189    );
```

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

college_name
City College

Result Grid

Assignment 2

Q) consider the given rational table

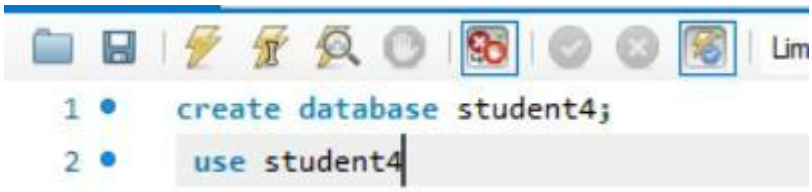
student(std no, std name, cource, city, scholarship,zipcode, district)

write SQL quries for the following

To run the SQL queries, we first need to:

1. **Create a database → so all our tables and data stay in one place.**
2. **Create the required tables → student(std no, std name, cource, city, scholarship,zipcode, district)**
3. **Insert sample data → to test the queries properly.**
4. **Run the queries → to get the answers for the given questions.**

Step 1: Create Database



Step 2: Create Tables

```
CREATE TABLE Student (
    stdno INT AUTO_INCREMENT PRIMARY KEY,
    stdname VARCHAR(50) NOT NULL,
    course VARCHAR(50),
    city VARCHAR(50),
    scholarship int,
    zipcode VARCHAR(10),
    district VARCHAR(50)
) AUTO_INCREMENT=101;
```

Step 3: Insert Sample Data

```
14 •  INSERT INTO Student (stdname, course, city, scholarship, zipcode, district)
15    VALUES
16    ('Amit Sharma', 'B.Tech', 'Mumbai', 45000, '071', 'Mumbai Suburban'),
17    ('Priya Singh', 'MBA', 'Delhi', 60000, '110', 'South Delhi'),
18    ('Rohan Patil', 'B.Sc', 'Mumbai', 30000, '071', 'Mumbai Central'),
19    ('Neha Verma', 'B.Com', 'Pune', 52000, '411', 'Pune City'),
20    ('Sneha Kulkarni', 'MBA', 'Mumbai', 48000, '071', 'Mumbai Suburban');
```

Assignment Queries

1. **create a sequence used to generate student numbers for std no column of the student table.**

```
3 •  CREATE TABLE Student (
4        stdno INT AUTO_INCREMENT PRIMARY KEY,
5        stdname VARCHAR(50) NOT NULL,
6        course VARCHAR(50),
7        city VARCHAR(50),
8        scholarship int,
9        zipcode VARCHAR(10),
10       district VARCHAR(50)
11   ) AUTO_INCREMENT=101;
12
13 •  select *from student;
```

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

stdno	stdname	course	city	scholarship	zipcode	district
101	Amit Sharma	B.Tech	Mumbai	45000	071	Mumbai Suburban
102	Priya Singh	MBA	Delhi	60000	110	South Delhi
103	Rohan Patil	B.Sc	Mumbai	30000	071	Mumbai Central
104	Neha Verma	B.Com	Pune	52000	411	Pune City
105	Sneha Kulkarni	MBA	Mumbai	48000	071	Mumbai Suburban

tudent 10 | Apply | Revert

2. **create an index on the district**

```
CREATE INDEX idx_district ON Student(district);
```

3. find the district whose zip code = 071 and check whether the query uses the index and write your observation

```
25 •  SELECT *
26    FROM Student
27    WHERE zipcode = '071';
```

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

stdno	stdname	course	city	scholarship	zipcode	district
101	Amit Sharma	B.Tech	Mumbai	45000	071	Mumbai Suburban
103	Rohan Patil	B.Sc	Mumbai	30000	071	Mumbai Central
105	Sneha Kulkarni	MBA	Mumbai	48000	071	Mumbai Suburban
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid

tudent 2 × Apply Revert

4. create a view for students having scholarship <50000 and residing in 'mumbai'.

```
29
30 •  CREATE VIEW Mumbai_Scholarship_View AS
31    SELECT *
32    FROM Student
33    WHERE scholarship < 50000 AND city = 'Mumbai';
```

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

stdno	stdname	course	city	scholarship	zipcode	district
101	Amit Sharma	B.Tech	Mumbai	45000	071	Mumbai Suburban
103	Rohan Patil	B.Sc	Mumbai	30000	071	Mumbai Central
105	Sneha Kulkarni	MBA	Mumbai	48000	071	Mumbai Suburban
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid

Student 4 × Apply Revert

5. Display a count of students who reside in 'mumbai'.

```
35
36 •  SELECT COUNT(*) AS Mumbai_Students
37    FROM Student
38    WHERE city = 'Mumbai';
```

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

Mumbai_Students
3

Result Grid

Result 5 × Read Only

6. **Find the average scholarship of students from the created view.**

```
27
28 •  SELECT AVG(scholarship) AS Avg_Scholarship
29    FROM Mumbai_Scholarship_View;
```

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

Avg_Scholarship
41000.0000

Result Grid

Result 8 × Read Only

7. **Display student names who reside on the same street as the students in the view**

```
32 •  SELECT stdname
33    FROM Student
34    WHERE district IN (SELECT district FROM Mumbai_Scholarship_View);
```

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

stdname
Amit Sharma
Rohan Patil
Sneha Kulkarni

Result Grid

Student 9 × Read Only

Assignment 3

Q) Consider the given database schema :

Patient (patient id, patient name ,doctor id, patient city)

Doctor (doctor id, doctor name, doctor city , specialization)

Use all types of joins and write SQL queries for the following

To run the SQL queries, we first need to:

1. **Create a database → so all our tables and data stay in one place.**
2. **Create the required tables → patient(patient id, patient name,doctor id,patientcity)**

   → Doctor(doctorid,doctorname,doctorcity,specillazation)
3. **Insert sample data → to test the queries properly.**
4. **Run the queries → to get the answers for the given questions.**

Step 1: Create Database

```
1
2 •  CREATE DATABASE HospitalDB;
3 •  USE HospitalDB;
4
```

Step 2: Create Tables

```
6 •  CREATE TABLE Doctor (
7        doctorid INT PRIMARY KEY,
8        doctorname VARCHAR(50),
9        doctorcity VARCHAR(50),
10       specialization VARCHAR(50)
11   );
12
13 •  CREATE TABLE Patient (
14       patientid INT PRIMARY KEY,
15       patientname VARCHAR(50),
16       doctorid INT NULL,
17       patientcity VARCHAR(50),
18       FOREIGN KEY (doctorid) REFERENCES Doctor(doctorid)
19   );
```

Step 3: Insert Sample Data

```
22 •  INSERT INTO Doctor VALUES
23    (1, 'Dr. Mehta', 'Mumbai', 'Cardiology'),
24    (2, 'Dr. Sharma', 'Delhi', 'Orthopedic'),
25    (3, 'Dr. Iyer', 'Mumbai', 'Neurology'),
26    (4, 'Dr. Khan', 'Pune', 'Cardiology');
27
28 •  INSERT INTO Patient VALUES
29    (101, 'Amit', 1, 'Mumbai'),
30    (102, 'Ravi', 2, 'Delhi'),
31    (103, 'Sneha', NULL, 'Pune'),
32    (104, 'Priya', 1, 'Mumbai'),
33    (105, 'Arjun', 4, 'Pune'),
34    (106, 'Manoj', NULL, 'Delhi');
```

Assignment Queries

1. Find the doctor of each patient.

```
38
39 •  SELECT p.patientid, p.patientname, d.doctorname, d.specialization
40    FROM Patient p
41    INNER JOIN Doctor d ON p.doctorid = d.doctorid;
```

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

patientid	patientname	doctorname	specialization
101	Amit	Dr. Mehta	Cardiology
104	Priya	Dr. Mehta	Cardiology
102	Ravi	Dr. Sharma	Orthopedic
105	Arjun	Dr. Khan	Cardiology

Result Grid

2. Find the patients who are not assigned to any doctor.

```
SELECT p.patientid, p.patientname
FROM Patient p
LEFT JOIN Doctor d ON p.doctorid = d.doctorid
WHERE d.doctorid IS NULL;


SELECT p.patientid, p.patientname, d.doctorid, d.doctorname
```

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

patientid	patientname
103	Sneha
106	Manoj

3. Find the patients who are not assigned to any doctor as well as doctors who do not have any patients.

```
SELECT p.patientid, p.patientname, d.doctorid, d.doctorname
FROM Patient p
LEFT JOIN Doctor d ON p.doctorid=
d.doctorid
where d.doctorid is null
union
select p.patientid,p.patientname,
d.doctorid,d.doctorname
from patient p
right join Doctor d on p.doctorid=
d.doctorid
where p.doctorid is null;
```

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

patientid	patientname	doctorid	doctorname
103	Sneha	NULL	NULL
106	Manoj	NULL	NULL
NULL	NULL	3	Dr. Iyer

4. Find the patients whose doctor's specialization is Cardiology.

```sql
SELECT p.patientid, p.patientname, d.doctorname, d.specialization
FROM Patient p
INNER JOIN Doctor d ON p.doctorid = d.doctorid
WHERE d.specialization = 'Cardiology';
```

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

patientid	patientname	doctorname	specialization
101	Amit	Dr. Mehta	Cardiology
104	Priya	Dr. Mehta	Cardiology
105	Arjun	Dr. Khan	Cardiology

5. Create a view containing the total number of patients whose doctor belongs to Mumbai.

```sql
CREATE VIEW vw_patients_mumbai AS
SELECT COUNT(p.patientid) AS total_patients_mumbai
FROM Patient p
INNER JOIN Doctor d ON p.doctorid = d.doctorid
WHERE d.doctorcity = 'Mumbai';


SELECT * FROM vw_patients_mumbai;
```

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

total_patients_mumbai
2

Assignment no 4

ER Modelling and Normalization:

Design using ER features using tools like ERD plus, ER Win etc. (Identifying entities, relationships between entities, attributes, keys, cardinalities,generalization, specialization etc.) Convert the ER diagram into relational tables and normalize the Relational data model.

ER Diagram Design

Entities and Attributes:

Person (PersonID PK, Name, Email)

Student (StudentID PK, DateOfBirth) — Student is a subtype of Person

Professor (ProfessorID PK, Department) — Professor is a subtype of

Person Course (CourseID PK, CourseName, Credits, ProfessorID FK)

Enrollment (EnrollmentID PK, StudentID FK, CourseID FK, Grade)

Relationships:

Person is a general entity, with Student and Professor as specializations (Generalization/Specialization)

Enrollment relates Student with Course (many-to-many)

Professor teaches Course (one-to-many)

Cardinalities:

A Student can enroll in multiple Courses; a Course can have multiple Students (many-to-many)

A Professor can teach multiple Courses; each Course taught by exactly one Professor (one-to-many)

Conversion to Relational Tables

Table	Attributes	Key
Person	**PersonID (PK), Name, Address, Phone**	**PersonID (PK)**
Student	**PersonID (PK, FK to Person), Major, Year**	**PersonID (PK, FK)**
Professor	**PersonID (PK, FK to Person), Dept, Title**	**PersonID (PK, FK)**
Course	**CourseID (PK), CourseName, Credits**	**CourseID (PK)**
Enrollment	**EnrollmentID (PK), PersonID (FK to Student), CourseID (FK), Grade**	**EnrollmentID (PK), PersonID (FK), CourseID (FK)**

Normalization

All tables are in 1NF (atomic columns).

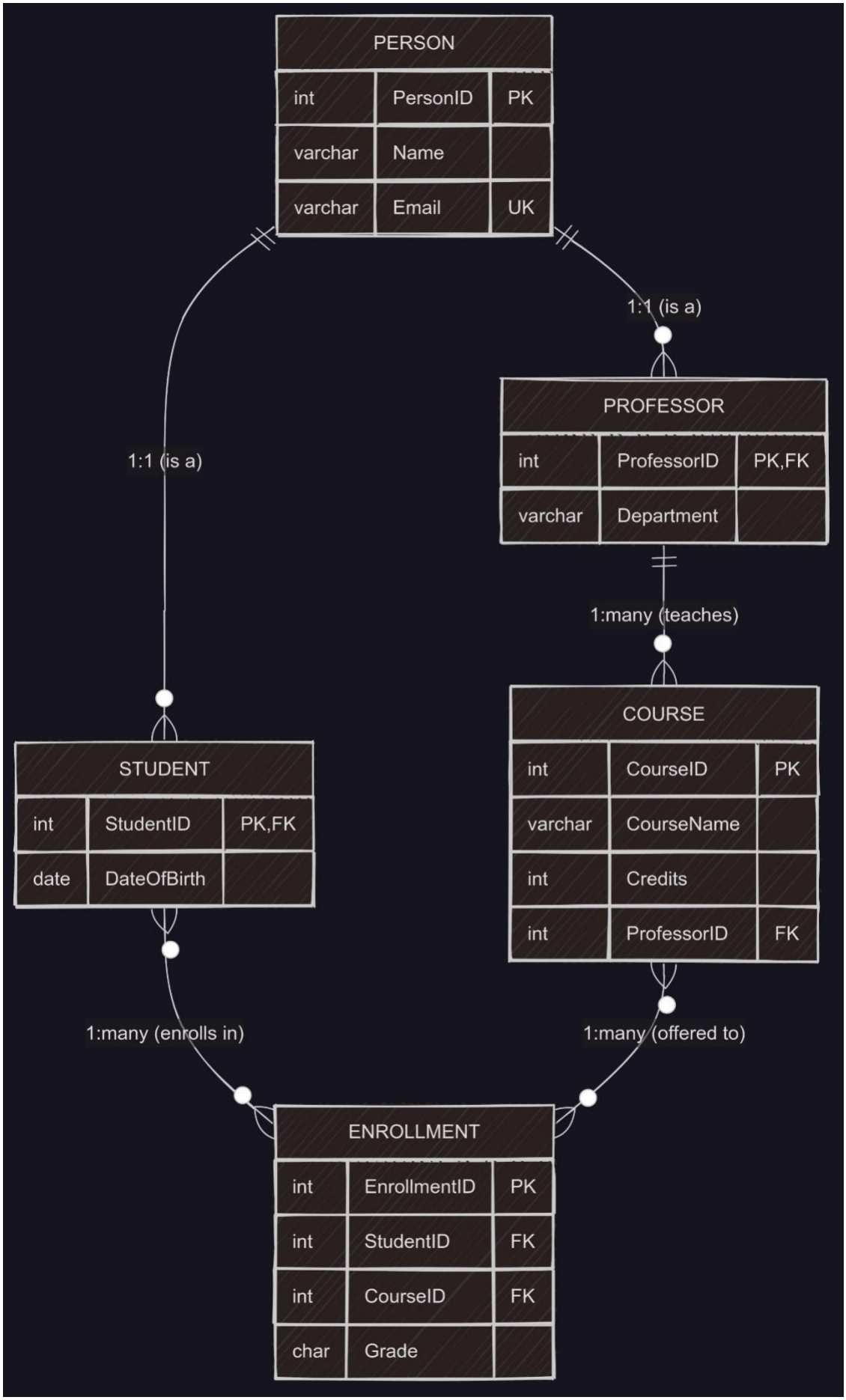
Student and Professor have PersonID as PK and FK, no partial dependencies (2NF). No transitive dependencies exist in any table (3NF).

This design covers generalization (Person -> Student, Professor) and relationships with appropriate keys and cardinalities.

Step 1: Create Tables

```sql
CREATE DATABASE startersql;
use startersql;

CREATE TABLE Person (
    PersonID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    DateOfBirth DATE,
    FOREIGN KEY (StudentID) REFERENCES Person(PersonID)
);

CREATE TABLE Professor (
    ProfessorID INT PRIMARY KEY,
    Department VARCHAR(50) NOT NULL,
    FOREIGN KEY (ProfessorID) REFERENCES Person(PersonID)
);

CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100) NOT NULL,
    Credits INT NOT NULL,
    ProfessorID INT,
    FOREIGN KEY (ProfessorID) REFERENCES Professor(ProfessorID)
);

CREATE TABLE Enrollment (
    EnrollmentID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    Grade CHAR(2),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)
);
```

This is the ER diagram for the following Data.


PERSON
int
PersonID
PK
varchar
Name
varchar
Email
UK
1:1 (is a)
1:1 (is a)
PROFESSOR
int
ProfessorID
PK,FK
varchar
Department
1:many (teaches)
STUDENT
int
StudentID
PK,FK
date
DateOfBirth
COURSE
int
CourseID
PK
varchar
CourseName
int
Credits
int
ProfessorID
FK
1:many (enrolls in)
1:many (offered to)
ENROLLMENT
int
EnrollmentID
PK
int
StudentID
FK
int
CourseID
FK
char
Grade

Assignment – 5

Create a datebase with following schemas

Borrower (roll in, name, date of issue, name of book, status) & fine (roll no , date , Amt)

Step 1 – create database

```
3 •  CREATE DATABASE LibraryDB;
4 •  USE LibraryDB;
```

Step 2 – create tables

```
7 •  CREATE TABLE Borrower (
8        RollIn INT PRIMARY KEY,
9        Name VARCHAR(50),
10       Date_of_Issue DATE,
11       Name_of_Book VARCHAR(100),
12       Status VARCHAR(20)
13   );
14
15 •  CREATE TABLE Fine (
16       FineID INT AUTO_INCREMENT PRIMARY KEY,
17       Roll_no INT,
18       Fine_Date DATE,
19       Amt DECIMAL(10,2),
20       FOREIGN KEY (Roll_no) REFERENCES Borrower(RollIn)
21   );
```

Step 3 – Insert data

```
24 •  INSERT INTO Borrower (RollIn, Name, Date_of_Issue, Name_of_Book, Status) VALUES
25    (1, 'Amit', '2025-09-01', 'Database Systems', 'Issued'),
26    (2, 'Ravi', '2025-08-15', 'Operating Systems', 'Returned'),
27    (3, 'Sneha', '2025-09-05', 'Computer Networks', 'Issued'),
28    (4, 'Priya', '2025-08-20', 'Data Structures', 'Returned');
29
30 •  INSERT INTO Fine (Roll_no, Fine_Date, Amt) VALUES
31    (2, '2025-08-25', 50.00),
32    (4, '2025-08-28', 30.00);
```

Result of table Borrower –

```
35 •  SELECT * FROM Borrower;
```

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

RollIn	Name	Date_of_Issue	Name_of_Book	Status
1	Amit	2025-09-01	Database Systems	Issued
2	Ravi	2025-08-15	Operating Systems	Returned
3	Sneha	2025-09-05	Computer Networks	Issued
4	Priya	2025-08-20	Data Structures	Returned
NULL	NULL	NULL	NULL	NULL

Borrower 5 ×

Result of table Fine-

```
38 •  SELECT * FROM Fine;
39
40
41
```

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

FineID	Roll_no	Fine_Date	Amt
1	2	2025-08-25	50.00
2	4	2025-08-28	30.00
NULL	NULL	NULL	NULL

Assignment No:6 Cursors

- **Consider a table employee with columns:**

  (emp_id, emp_name, hire_date, salary, incentive)

- **Use an explicit cursor to fetch employees whose hire month = current month.**
- **Calculate incentive based on years of experience.**
- **Display results using DBMS_OUTPUT.**

1. Drop old objects

Removes any existing employee table or generate_incentive_report procedure.

```
4
5
6 •   DROP PROCEDURE IF EXISTS generate_incentive_report;
7 •   DROP TABLE IF EXISTS employee;
8
```

2. Create table + sample data

Makes an employee table (emp_id, emp_name, hire_date, salary, incentive) and inserts some rows.

```
9  • CREATE TABLE employee (
10      emp_id INT PRIMARY KEY,
11      emp_name VARCHAR(100),
12      hire_date DATE,
13      salary DECIMAL(12,2),
14      incentive DECIMAL(12,2) DEFAULT NULL
15  );
```

```
17 • INSERT INTO employee (emp_id, emp_name, hire_date, salary) VALUES
18    (101, 'Rahul',  '2021-10-05', 50000.00),
19    (102, 'Priya',  '2020-10-12', 60000.00),
20    (103, 'Amit',   '2019-09-22', 55000.00),
21    (104, 'Sneha',  '2018-10-02', 70000.00),
22    (105, 'Rita',   '2024-01-15', 45000.00);
23
24 • COMMIT;
```

3. Change delimiter ($$)

```
25
26      DELIMITER $$
27
```

4. Create stored procedure

```
CREATE PROCEDURE generate_incentive_report()
BEGIN
```

```
31      DECLARE done INT DEFAULT 0;
32
33      DECLARE v_emp_id INT;
34      DECLARE v_emp_name VARCHAR(100);
35      DECLARE v_hire_date DATE;
36      DECLARE v_salary DECIMAL(12,2);
37      DECLARE v_incentive DECIMAL(12,2);
38      DECLARE v_experience INT;
39
40
41      DECLARE emp_cur CURSOR FOR
42        SELECT emp_id, emp_name, hire_date, salary
43        FROM employee
44        WHERE MONTH(hire_date) = MONTH(CURDATE());
45
46
47      DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
48
```

5.open cursour and process rows (main loop)

```
61      OPEN emp_cur;
62
63      read_loop: LOOP
64          FETCH emp_cur INTO v_emp_id, v_emp_name, v_hire_date, v_salary;
65          IF done = 1 THEN
66             LEAVE read_loop;
67          END IF;
68
69          SET v_experience = FLOOR(TIMESTAMPDIFF(MONTH, v_hire_date, CURDATE()) / 12);
70
71          IF v_experience < 1 THEN
72              SET v_incentive = v_salary * 0.05;
73          ELSEIF v_experience BETWEEN 1 AND 3 THEN
74              SET v_incentive = v_salary * 0.10;
75          ELSE
76              SET v_incentive = v_salary * 0.15;
77          END IF;
```

```
        UPDATE employee
        SET incentive = v_incentive
        WHERE emp_id = v_emp_id;

        INSERT INTO temp_report(emp_id, emp_name, hire_date, experience_years, incentive)
        VALUES (v_emp_id, v_emp_name, v_hire_date, v_experience, v_incentive);
    END LOOP;

    CLOSE emp_cur;
```

6. Return the report and cleanup

```
SELECT emp_id, emp_name, DATE_FORMAT(hire_date, '%d-%b-%Y') AS hire_date,
       experience_years, incentive
FROM temp_report
ORDER BY emp_id;

DROP TEMPORARY TABLE IF EXISTS temp_report;
```

7. End procedure & restore delimiter

```
96    END$$
97
98    DELIMITER ;
```

8. Execute procedure and check table

```
100
101 •  CALL generate_incentive_report();
102 •  SELECT * FROM employee;
103
104
```

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

emp_id	emp_name	hire_date	salary	incentive
101	Rahul	2021-10-05	50000.00	7500.00
102	Priya	2020-10-12	60000.00	9000.00
103	Amit	2019-09-22	55000.00	NULL
104	Sneha	2018-10-02	70000.00	10500.00
105	Rita	2024-01-15	45000.00	NULL
NULL	NULL	NULL	NULL	NULL

Assignment no : 7 Database Trigger

Create a Library database with schema:Books (AccNo, Title, Author, Publisher, Count)

Then:

a) Create Library_Audit table (same fields as Books + Date + Status).

b) Create AFTER triggers to record changes (Insert, Update, Delete) from Books into Library_Audit.

1) **create the main Books table**

```
CREATE TABLE Books (
    AccNo INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100),
    Publisher VARCHAR(100),
    Count INT
);
```

2) **Create the library_Audit table**

```
CREATE TABLE Library_Audit (
    Audit_ID INT AUTO_INCREMENT PRIMARY KEY,
    AccNo INT,
    Title VARCHAR(100),
    Author VARCHAR(100),
    Publisher VARCHAR(100),
    Count INT,
    Action_Date DATETIME,
    Action_Type VARCHAR(20)
);
```

3) Restore Delimiter & Trigger on insert (after_books_insert)

```
25    DELIMITER $$
26 •  CREATE TRIGGER after_books_insert
27    AFTER INSERT ON Books
28    FOR EACH ROW
29    BEGIN
30        INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count, Action_Date, Action_Type)
31        VALUES (NEW.AccNo, NEW.Title, NEW.Author, NEW.Publisher, NEW.Count, NOW(), 'INSERTED');
32    END$$
33    DELIMITER ;
```

4) Restore Delimeter & Trigger on update (after_books_update)

```
36    DELIMITER $$
37 •  CREATE TRIGGER after_books_update
38    AFTER UPDATE ON Books
39    FOR EACH ROW
40    BEGIN
41        INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count, Action_Date, Action_Type)
42        VALUES (NEW.AccNo, NEW.Title, NEW.Author, NEW.Publisher, NEW.Count, NOW(), 'UPDATED');
43    END$$
44    DELIMITER ;
```

5) Restore Delimeter & Trigger on Delete (after_books_delete)

```
47    DELIMITER $$
48 •  CREATE TRIGGER after_books_delete
49    AFTER DELETE ON Books
50    FOR EACH ROW
51    BEGIN
52        INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count, Action_Date, Action_Type)
53        VALUES (OLD.AccNo, OLD.Title, OLD.Author, OLD.Publisher, OLD.Count, NOW(), 'DELETED');
54    END$$
55    DELIMITER ;
```

6) Test The Triggers

```
58 • INSERT INTO Books VALUES (1, 'DBMS Concepts', 'Korth', 'McGraw Hill', 5);
59 • UPDATE Books SET Count = 8 WHERE AccNo = 1;
60 • DELETE FROM Books WHERE AccNo = 1;
61 • SELECT * FROM Library_Audit;
62
63
64
```

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

Audit_ID	AccNo	Title	Author	Publisher	Count	Action_Date	Action_Type
1	1	DBMS Concepts	Korth	McGraw Hill	5	2025-10-13 00:08:08	INSERTED
2	1	DBMS Concepts	Korth	McGraw Hill	8	2025-10-13 00:08:08	UPDATED
3	1	DBMS Concepts	Korth	McGraw Hill	8	2025-10-13 00:08:08	DELETED
4	1	DBMS Concepts	Korth	McGraw Hill	5	2025-10-13 00:27:57	INSERTED
5	1	DBMS Concepts	Korth	McGraw Hill	8	2025-10-13 00:27:57	UPDATED
6	1	DBMS Concepts	Korth	McGraw Hill	8	2025-10-13 00:27:57	DELETED
7	1	DBMS Concepts	Korth	McGraw Hill	5	2025-10-13 00:28:14	INSERTED
8	1	DBMS Concepts	Korth	McGraw Hill	8	2025-10-13 00:28:14	UPDATED
9	1	DBMS Concepts	Korth	McGraw Hill	8	2025-10-13 00:28:14	DELETED
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types

Library_Audit 9 ×

Apply | Revert