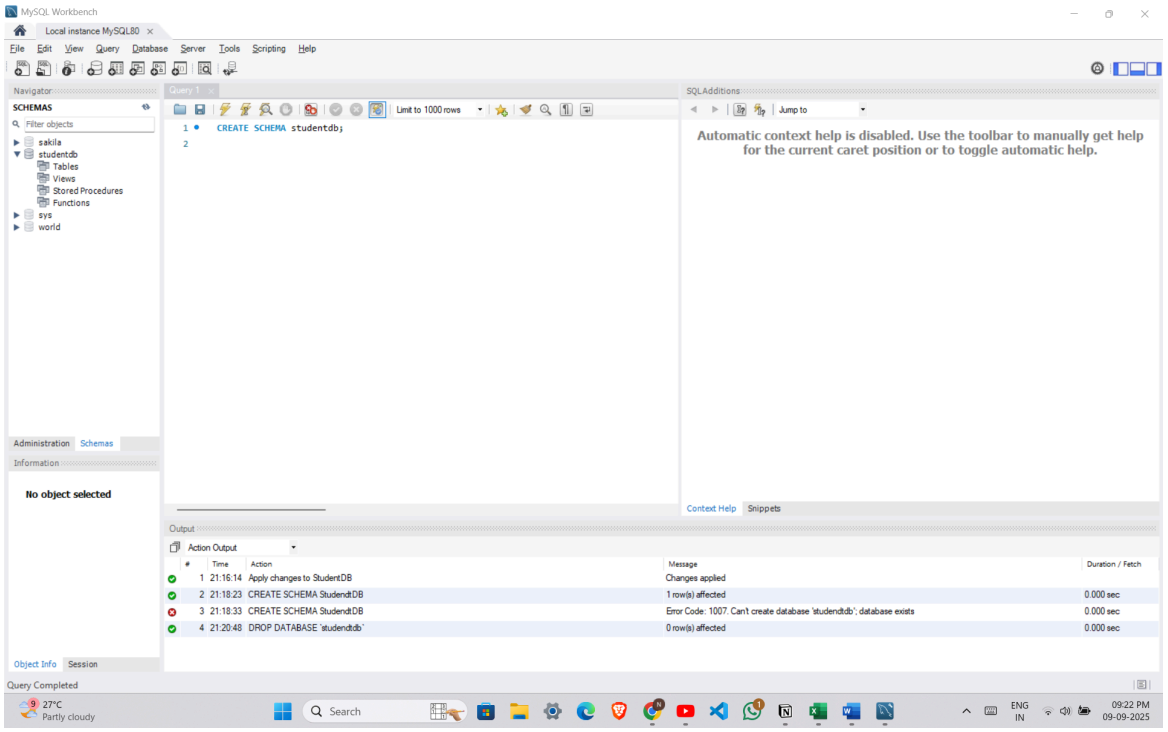
WEEK-1 ASSIGNMENTS

▼ Assignment 1

```
CREATE SCHEMA studentdb;
```



assignment1.sql

▼ Assignment 2

- Select the Database

```
USE studentdb;
```

- Create Students Table

```
CREATE TABLE students(
student_id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100),
age INT,
gender ENUM('Male', 'Female'),
course_id INT
);
```

- `Create Courses Table`

```
CREATE TABLE courses (
course_id INT PRIMARY KEY AUTO_INCREMENT,
course_name VARCHAR(100),
duration VARCHAR(50)
);
```

- `Create Marks Table`

```
CREATE TABLE marks (
mark_id INT PRIMARY KEY AUTO_INCREMENT,
student_id INT,
subject VARCHAR(100),
score DECIMAL(5,2)
);
```

- `Modify Students table to add a new column email`

```
ALTER TABLE Students ADD COLUMN email VARCHAR(100);
```

- `Drop the Marks table and recreate it with the same structure.`

```
DROP TABLE IF EXISTS Marks;
```

```
CREATE TABLE marks (
mark_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
student_id INT,
subject VARCHAR(100),
score DECIMAL(5,2)
);
```

assignment2.sql

▼ Assignment 3

1. Insert 5 Rows per Table

```
-- Insert into courses
INSERT INTO courses (course_name, duration) VALUES
('DBMS', '6 months'),
('OS', '1 year'),
('Python', '1 year'),
('Java', '6 months'),
('C++', '6 months');

-- Insert into students
INSERT INTO students (name, age, gender, course_id, email) VALUES
('Nisharg Soni', 21, 'Male', 1, 'nisharg@gmail.com'),
('Dakshil Gorasiya', 19, 'Male', 3, 'dakshil@gmail.com'),
('Diya Mehta', 22, 'Female', 2, 'diya@gmail.com'),
('Manish Patel', 20, 'Male', NULL, 'manish@gmail.com'),
('Krisha Shah', 23, 'Female', 1, 'krisha@gmail.com');

-- Insert into marks
INSERT INTO marks (student_id, subject, score) VALUES
(1, 'DBMS', 88.5),
(2, 'Python', 92.0),
(3, 'OS', 85.0),
(4, 'DLD', 78.0),
(5, 'DBMS', 91.5);
```

2. Update one student’s course.

```
UPDATE students SET course_id = 4 WHERE student_id = 2;
```

3. Delete a student record.

```
DELETE FROM students WHERE student_id = 4;
```

assignment3.sql

▼ Assignment 4

1. Students Above Age 20

```
SELECT * FROM students WHERE age > 20;
```

2. Students Ordered Alphabetically

```
SELECT * FROM students ORDER BY name ASC;
```

3. Total Students per Course

```
SELECT course_id, COUNT(*) AS total_students
FROM students
GROUP BY course_id;
```

4. Courses with More Than 2 Students

```
SELECT course_id, COUNT(*) AS student_count
FROM students
GROUP BY course_id
HAVING student_count > 2;
```

assignment4.sql

▼ Assignment 5

1. Display students with their enrolled course names using INNER JOIN.

```
SELECT s.student_id, s.name, c.course_id, c.course_name
FROM students s
INNER JOIN courses c ON s.course_id = c.course_id;
```

2. Display all students even if they are not enrolled in any course (LEFT JOIN).

```
SELECT s.student_id, s.name, c.course_id, c.course_name
FROM students s
LEFT JOIN courses c ON s.course_id = c.course_id;
```

3. Display all courses and their students (RIGHT JOIN).

```
SELECT s.student_id, s.name, c.course_id, c.course_name
FROM students s
RIGHT JOIN courses c ON s.course_id = c.course_id;
```

4. Find highest, lowest, and average marks per subject.

```
SELECT subject,
    MAX(score) AS max_score,
    MIN(score) AS min_score,
    AVG(score) AS avg_score
FROM marks
GROUP BY subject;
```

5. Count how many male and female students exist.

```
SELECT gender, COUNT(*) AS total_count
FROM students
GROUP BY gender;
```

assignment5.sql

▼ LibraryDB

```
CREATE SCHEMA librarydb ;
```

```
USE librarydb;
```

```
CREATE TABLE authors (
author_id INT PRIMARY KEY AUTO_INCREMENT,
author_name VARCHAR(150),
country VARCHAR(100)
);
```

```
CREATE TABLE books (
book_id INT PRIMARY KEY AUTO_INCREMENT,
title VARCHAR(200),
author_id INT,
publish_year INT,
price DECIMAL(6,2),
FOREIGN KEY (author_id) REFERENCES Authors(author_id)
);
```

```
CREATE TABLE borrowers (
borrower_id INT PRIMARY KEY AUTO_INCREMENT,
book_id INT,
borrower_name VARCHAR(150),
borrow_date DATE,
```

```
return_date DATE,
FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

```
INSERT INTO Authors (author_name, country) VALUES
('Chetan Bhagat', 'India'),
('Ruskin Bond', 'India'),
('Amish Tripathi', 'India'),
('Arundhati Roy', 'India'),
('R.K. Narayan', 'India');
```

```
INSERT INTO Books (title, author_id, publish_year, price) VALUES
('Five Point Someone', 1, 2004, 299.00),
('The Room on the Roof', 5, 1956, 199.00),
('The Shiva Trilogy', 3, 2010, 499.00),
('The God of Small Things', 4, 1997, 399.00),
('Our Trees Still Grow in Dehra', 2, 1991, 249.00);
```

```
INSERT INTO Borrowers (book_id, borrower_name, borrow_date, return_
date) VALUES
(1, 'Rajesh Kumar', '2025-08-01', '2025-08-15'),
(3, 'Sneha Sharma', '2025-08-05', NULL),
(4, 'Anil Verma', '2025-08-10', '2025-08-20'),
(2, 'Priya Singh', '2025-08-12', NULL),
(5, 'Vikas Mehta', '2025-08-15', '2025-08-25');
```

```
-- 1. Retrieve the list of all books along with their author's name.
SELECT b.title, a.author_name
FROM Books b
INNER JOIN Authors a ON b.author_id = a.author_id;
```

```
-- 2. Find all authors from the India who published books after 2009.
SELECT a.author_name,b.publish_year
FROM Authors a
```

```
INNER JOIN Books b ON a.author_id = b.author_id
WHERE a.country = 'India' AND b.publish_year >2009;
```

```
-- 3. List all borrowers who haven’t returned the book yet.
SELECT borrower_name, borrow_date
FROM borrowers
WHERE return_date IS NULL;
```

```
-- 4. Number of books per author
SELECT a.author_name, COUNT(b.book_id) AS total_books
FROM authors a
LEFT JOIN books b ON a.author_id = b.author_id
GROUP BY a.author_id;
```

```
-- 5. Average price of books per author
SELECT a.author_name, AVG(b.price) AS avg_price
FROM authors a
INNER JOIN books b ON a.author_id = b.author_id
GROUP BY a.author_id;
```

```
ALTER TABLE books
ADD COLUMN genre VARCHAR(50) NOT NULL DEFAULT 'Uncategorize
d';
```

```
SELECT * FROM books;
```

```
ALTER TABLE books
ADD CONSTRAINT chk_price CHECK (price > 0);
```

```
UPDATE books SET genre = 'Contemporary' WHERE book_id = 1;
UPDATE books SET genre = 'Classic' WHERE book_id = 2;
UPDATE books SET genre = 'Mythology' WHERE book_id = 3;
```

```
UPDATE books SET genre = 'Fiction' WHERE book_id = 4;
UPDATE books SET genre = 'Classic' WHERE book_id = 5;
```

```
ALTER TABLE borrowers
ADD COLUMN borrower_email VARCHAR(255),
ADD COLUMN due_date DATE;
```

```
ALTER TABLE borrowers
ADD CONSTRAINT chk_return_date CHECK (return_date IS NULL OR ret
urn_date >= borrow_date);
```

```
ALTER TABLE borrowers
ADD CONSTRAINT chk_email CHECK (borrower_email LIKE '%@%.%');
```

```
SELECT * FROM borrowers;
```

```
UPDATE borrowers SET borrower_email = 'rajesh@example.com', due_d
ate = '2025-08-22' WHERE borrower_id = 1;
UPDATE borrowers SET borrower_email = 'sneha@example.com', due_d
ate = '2025-08-25' WHERE borrower_id = 2;
UPDATE borrowers SET borrower_email = 'anil@example.com', due_dat
e = '2025-08-30' WHERE borrower_id = 3;
UPDATE borrowers SET borrower_email = 'priya@example.com', due_da
te = '2025-09-01' WHERE borrower_id = 4;
UPDATE borrowers SET borrower_email = 'vikas@example.com', due_d
ate = '2025-09-05' WHERE borrower_id = 5;
```

```
-- 6. Find all books that are currently overdue.
SELECT
br.borrower_name,
br.borrower_email,
b.title,
br.due_date
FROM borrowers AS br
```

```
JOIN books AS b ON br.book_id = b.book_id
WHERE br.return_date IS NULL AND br.due_date < CURDATE();
```

```
-- 7. List all books that have never been borrowed.
SELECT b.title
FROM books AS b
LEFT JOIN borrowers AS br ON b.book_id = br.book_id
WHERE br.borrower_id IS NULL;
```

```
-- 8. Categorize books by their publication era using a CASE statement.
SELECT
title,
publish_year,
CASE
WHEN publish_year < 1990 THEN 'Classic Era'
WHEN publish_year >= 1990 AND publish_year < 2010 THEN 'Modern Er
a'
ELSE 'Contemporary Era'
END AS era
FROM books;
```

```
-- 9. Find the average number of days a book is borrowed for.
SELECT
AVG(DATEDIFF(return_date, borrow_date)) AS avg_borrow_duration_day
s
FROM borrowers
WHERE return_date IS NOT NULL;
```

```
-- 10. Create a summary report of how many books were borrowed in ea
ch month of 2025.
SELECT
COUNT(CASE WHEN MONTH(borrow_date) = 8 THEN 1 END) AS 'Augus
t_Borrows',
COUNT(CASE WHEN MONTH(borrow_date) = 9 THEN 1 END) AS 'Septe
mber_Borrows'
```

```
FROM borrowers
WHERE YEAR(borrow_date) = 2025;
```