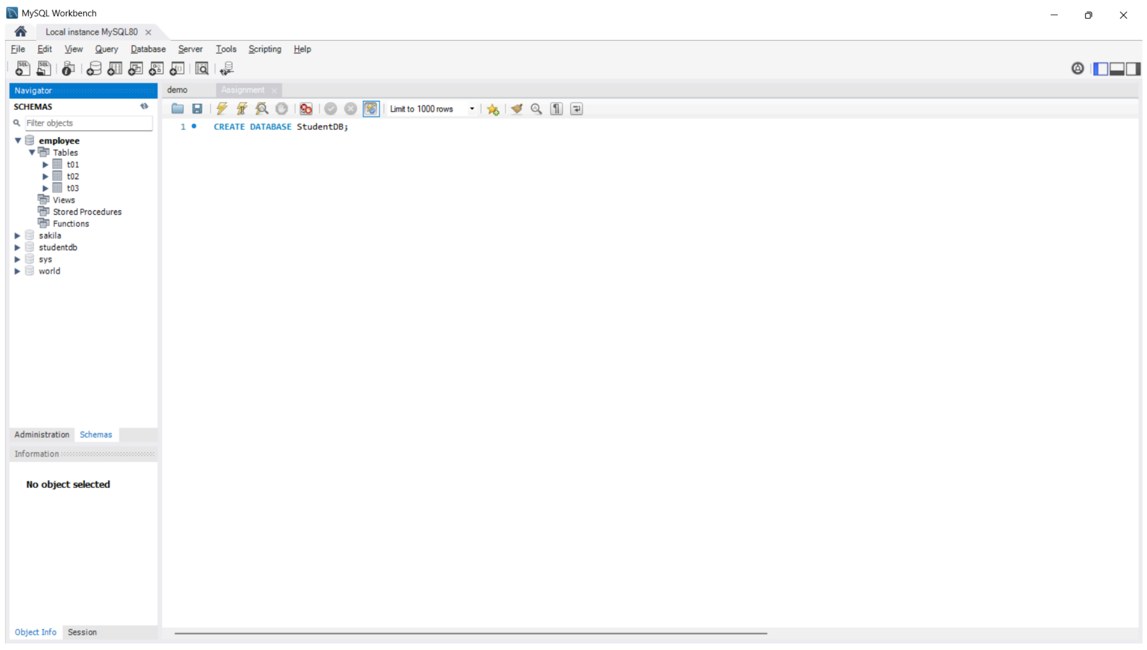
Week 1 - Assignments

Created by	Dakshil Gorasiya

Assignment 1

```
CREATE DATABASE StudentDB;
```



```
CREATE DATABASE StudentDB;
```

Assignment 2

```
-- Courses
CREATE TABLE T02
(
        T02F01 INT PRIMARY KEY,  -- course_id
    T02F02 VARCHAR(50) NOT NULL, -- course_nae
    T02F03 INT NOT NULL -- duration
);

-- Students
CREATE TABLE T01
(
    T01F01 INT PRIMARY KEY,  -- student_id
  T01F02 VARCHAR(50) NOT NULL, -- name
  T01F03 INT NOT NULL, -- age
  T01F04 ENUM("MALE", "FEMALE", "OTHER"), -- gender
  T01F05 INT, -- course_id
  CONSTRAINT T01F05_FK FOREIGN KEY (T01F05) REFERENCES T02(T02F01) ON DELETE RE
STRICT
);

-- Marks
CREATE TABLE T03
(
    T03F01 INT PRIMARY KEY, -- mark_id
  T03F02 INT, -- student_id
  T03F03 VARCHAR(50) NOT NULL, -- subject
  T03F04 DECIMAL(5, 2) NOT NULL,
  CONSTRAINT T03F02_FK FOREIGN KEY (T03F02) REFERENCES T01(T01F01) ON DELETE CA
SCADE
);

ALTER TABLE T01 ADD COLUMN T01F06 VARCHAR(100) NOT NULL UNIQUE;  -- email

DROP TABLE T03;
```

Assignment 3

```
INSERT INTO T02 (T02F01, T02F02, T02F03) VALUES
(101, 'Computer Science', 4),
(102, 'Business Administration', 3),
(103, 'Data Analytics', 1),
(104, 'MACHINE LEARNING', 2),
(105, 'WEB DEVELOPMENT', 2);

INSERT INTO T01 (T01F01, T01F02, T01F03, T01F04, T01F05, T01F06) VALUES
(1, 'Alice Johnson', 21, 'Female', 101, 'alice@gmail.com'),
(2, 'Bob Smith', 22, 'Male', 101, 'bob@gmail.com'),
(3, 'Charlie Brown', 20, 'Male', 102, 'charlie@gmail.com'),
(4, 'Diana Prince', 23, 'Female', 103, 'diana@gmail.om'),
(5, 'Eve Adams', 21, 'Female', 102, 'eve@gmail.com');

INSERT INTO T03 (T03F01, T03F02, T03F03, T03F04) VALUES
(1, 1, 'OS', 85),
(2, 2, 'ACCOUNTS', 91),
(3, 3, 'PYTHON', 76),
(4, 4, 'TENSORFLOW', 95),
(5, 5, 'COA', 88);

UPDATE T01 SET T01F05 = 104 WHERE T01F01 = 1;

DELETE FROM T01 WHERE T01F01 = 1;
```

Assignment 4

```
-- 1. Fetch all students above age 20
SELECT
    T01F02,
    T01F03
FROM
    T01
WHERE
    T01F03 > 20;

-- 2. Fetch all students ordered by name alphabetically.
SELECT
    T01F02
FROM
    T01
ORDER BY
    T01F02;

-- 3. Show total number of students enrolled in each course using GROUP BY.
SELECT
    T02F01,
  T02F02,
  COUNT(T01F05)
FROM
    T01
  INNER JOIN T02 ON T02F01 = T01F05
GROUP BY
    T01F05;

-- 4. Show courses that have more than 2 students using HAVING.
SELECT
    T02F01,
  T02F02,
  COUNT(T01F05)
FROM
    T01
  INNER JOIN T02 ON T02F01 = T01F05
GROUP BY
```

```
    T01F05
HAVING
    COUNT(T01F05) > 2;
```

Assignment 5

```
-- 1. Display students with their enrolled course names using INNER JOIN.
SELECT
    T01F01,
    T01F02,
    T02F02
FROM
    T01
    INNER JOIN T02 ON T01F05 = T02F01;

-- 2. Display all students even if they are not enrolled in any course (LEFT JOIN).
SELECT
    T01F01,
    T01F02,
    COALESCE(T02F02, 'Not Enrolled')
FROM
    T01
    LEFT JOIN T02 ON T01F05 = T02F01;

-- 3. Display all courses and their students (RIGHT JOIN).
SELECT
    T02F01,
    T02F02,
    COUNT(T01F05)
FROM
    T01
    RIGHT JOIN T02 ON T01F05 = T02F01
GROUP BY
    T02F01;

-- 4. Find highest, lowest, and average marks per subject.
SELECT
    T03F03,
    MAX(T03F04),
    MIN(T03F04),
    AVG(T03F04)
FROM
    T03
GROUP BY
    T03F03;
```

```
-- 5. Count how many male and female students exist.
SELECT
    T01F04,
    COUNT(T01F04)
FROM
    T01
GROUP BY
    T01F04;
```

LibraryDB

```
CREATE DATABASE LibraryDB;

USE LibraryDB;

-- Authors
CREATE TABLE T01
(
    T01F01 INT PRIMARY KEY AUTO_INCREMENT, -- author_id
    T01F02 VARCHAR(50) NOT NULL, -- first_name
    T01F03 VARCHAR(50) NOT NULL -- last_name
);

-- Books
CREATE TABLE T02
(
    T02F01 INT PRIMARY KEY AUTO_INCREMENT, -- book_id
    T02F02 VARCHAR(100) NOT NULL, -- title
    T02F03 INT, -- author_id
    T02F04 VARCHAR(50) NOT NULL, -- genre
    T02F05 INT NOT NULL, -- publication_year
    T02F06 VARCHAR(13) UNIQUE, -- isbn
    FOREIGN KEY (T02F03) REFERENCES T01(T01F01)
);

-- Borrowers
CREATE TABLE T03
(
    T03F01 INT PRIMARY KEY AUTO_INCREMENT, -- borrower_id
    T03F02 VARCHAR(50) NOT NULL, -- first_name
    T03F03 VARCHAR(50) NOT NULL, -- last_name
    T03F04 VARCHAR(100) UNIQUE NOT NULL, -- email
    T03F05 DATE -- registration_date
);

-- Borrowing_Records
CREATE TABLE T04
(
    T04F01 INT PRIMARY KEY AUTO_INCREMENT, -- record_id
    T04F02 INT, --   book_id
    T04F03 INT, -- borrower_id
    T04F04 DATE NOT NULL, -- borrow_date
    T04F05 DATE NOT NULL, -- due_date
    T04F06 DATE, -- return date
    FOREIGN KEY (T04F02) REFERENCES T02(T02F01),
    FOREIGN KEY (T04F03) REFERENCES T03(T03F01)
);

INSERT INTO T01 (T01F02, T01F03) VALUES
('George', 'Orwell'),
('J.K.', 'Rowling'),
('Haruki', 'Murakami'),
('Gabriel', 'Garcia Marque'),
('Isaac', 'Asimov');

INSERT INTO T03 (T03F02, T03F03, T03F04, T03F05) VALUES
('John', 'Smith', 'john.smith@email.com', '2023-01-15'),
('Jane', 'Doe', 'jane.doe@email.com', '2023-03-22'),
('Peter', 'Jones', 'peter.jones@email.com', '2024-05-30'),
('Mary', 'Williams', 'mary.w@email.com', '2024-08-01');

INSERT INTO T02 (T02F02, T02F03, T02F04, T02F05, T02F06) VALUES
('1984', 1, 'Dystopian', 1949, '9780451524935'),
('Animal Farm', 1, 'Political Satire', 1945, '9780451526342'),
('Harry Potter and the Sorcerer''s Stone', 2, 'Fantasy', 1997, '9780590353427'),
('Norwegian Wood', 3, 'Fiction', 1987, '9780375704024'),
('Kafka on the Shore', 3, 'Magical Realism', 2002, '9781400079278'),
('One Hundred Years of Solitude', 4, 'Magical Realism', 1967, '9780060883287'),
('I, Robot', 5, 'Science Fiction', 1950, '9780553382563');

INSERT INTO T04 (T04F02, T04F03, T04F04, T04F05, T04F06) VALUES
(1, 1, '2025-07-01', '2025-07-15', '2025-07-14'),
(4, 2, '2025-08-15', '2025-08-29', '2025-08-28'),
(3, 1, '2025-08-10', '2025-08-24', NULL),
(7, 3, '2025-08-20', '2025-09-03', NULL),
(1, 2, '2025-08-25', '2025-09-08', NULL),
(6, 4, '2025-09-01', '2025-09-15', NULL);

-- List books which are currently borrowed and who has it and when its due date
SELECT
    T02F01,
  T02F02,
  T03F02,
  T03F03
FROM
    T02
  INNER JOIN T04 ON T02F01 = T04F02 AND T04F06 IS NULL
  INNER JOIN T03 ON T04F03 = T03F01;

-- List author with more than one book in library
SELECT
    T01F01,
  T01F02,
  T01F03,
  COUNT(T02F03)
FROM
    T01
  INNER JOIN T02 ON T01F01 = T02F03
GROUP BY
    T02F03
HAVING
    COUNT(T02F03) > 1;

-- Identify Overdue Books and Borrower Contact Info
SELECT
    T03F01,
  T03F02,
  T03F03,
  T03F04,
  T04F05
FROM
    T03
  INNER JOIN T04 ON T03F01 = T04F03
WHERE
    T04F06 IS NULL AND
    T04F05 < (CURRENT_DATE());

-- Find the Most Actively Borrowed Book
SELECT
    T02F01,
  T02F02,
  T02F06,
  COUNT(T02F04)
FROM
    T02
  INNER JOIN T04 ON T02F01 = T04F02
GROUP BY
    T02F01,
  T02F02,
  T02F06,
    T02F04
ORDER BY
    COUNT(T02F04) DESC
LIMIT 1;

-- List All Books That Have Never Been Borrowed
SELECT
    T02F01,
  T02F02
FROM
    T02
  LEFT JOIN T04 ON T02F01 = T04F02
WHERE
    T04F02 IS NULL;

-- Find Borrowers Who Read Across Multiple Genres
WITH BORROWERGENERECOUNT AS
(
    SELECT
        T03F01,
    T03F02,
    T03F03,
    COUNT(DISTINCT T02F04) AS DISTINCT_GENERE
    FROM
        T03
    INNER JOIN T04 ON T03F01 = T04F03
    INNER JOIN T02 ON T02F01 = T04F02
    GROUP BY
        T03F01,
    T03F02,
    T03F03
)
SELECT
    T03F01,
  T03F02,
  T03F03,
  DISTINCT_GENERE
FROM
    BORROWERGENERECOUNT
WHERE
    DISTINCT_GENERE > 1;
```