**Daily Assignment**

**Sprint 1 Day 3**

1. Create a employee table with required details.

**Explanation:**  
Create a table for employees with basic fields like ID, name, email, etc.  
  
**Solution:**

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100),

Department VARCHAR(100)

);

1. Create a Student table with required details.

**Explanation:**

Define a student table with details like ID, name, DOB, etc.

**Solution:**

CREATE TABLE Student (

StudID INT PRIMARY KEY,

FIRST NAME VARCHAR(100),

LAST NAME VARCHAR(100),

DOB DATE,

EMAIL VARCHAR(100)

);

1. Create a employee and salary details tables with by having a foreing key.

**Explanation:**

Salary table should reference Employee table with all the required details.

**Solution:**

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100),

Department VARCHAR(100)

);

CREATE TABLE SALARY(

SALARYID INT PRIMARY KEY,

SALARYAMOUNT DECIMAL(10,2),

FOREIGN KEY (EmployeeID)REFERENCES Employee(EmployeeID)

);

1. Create a Student table with 2 values is identified as primay key.

**Explanation:**

Composite primary key using StudentID and CourseID.

**Solution:**

CREATE TABLE T1(

STUDENTID INT,

COURSEID INT,

PRIMARY KEY(STUDENTID,COURSEID)

);

1. Update the employee table to add new designation column.

**Explanation:**

We will use ALTER keyword to add a new designation column in the employee table.

**Solution:**

ALTER TABLE Employee ADD AGE INT;

1. Update the Employee table to increase the size of the column email.

**Explanation:**

In this question lets say the VARCHAR(100) is defined we will use ALTER to make changes and after that we will use MODIFY keyword to do VARCHAR(150).

**Solution:**

ALTER TABLE Employee MODIFY Email VARCHAR(150);

1. Rename the Student table to studentDetails

**Explanation:**

We will use RENAME keyword to rename the student table to studentDetails.

**Solution:**

RENAME TABLE Student TO studentDetails;

1. Drop the studentDetails table if the table is exists.

**Explanation:**

We will use Drop keyword to drop a table.

**Solution:**

DROP TABLE IF EXISTS studentDetails;

1. Create studentDetails table with (courseId as mandatory, email as unique, any 2 fields as primary key)

**Explanation**:

We will use NOT NULL to make the column mandatory, Unique to make it unique and composite key to make more than one primary key.  
**Solution:**

CREATE TABLE studentDetails (

StudentID INT,

CourseID INT NOT NULL,

Email VARCHAR(100) UNIQUE,

PRIMARY KEY (StudentID, CourseID)

);

1. Demonstrate the difference between drop and truncate.

**Explanation:**

-- DROP removes the table completely.

-- TRUNCATE removes all rows but keeps the structure.  
**Solution:**  
-- DROP removes the table completely.

-- TRUNCATE removes all rows but keeps the structure.

1. Remove the comments column from the Student table.

**Explanation:**

We will first ALTER and then DROP the columns .

**Solution:**

ALTER TABLE studentDetails DROP COLUMN Comments

1. Insert 40-50 records to each table studentDetails, employee, salary, course.

**Explanation:**

We will use INSERT to call for the columns and Values to the respective columns.

**Solution:**

INSERT INTO Employee (EmployeeID, Name, Email, Department, Designation) VALUES

(1, 'Asutosh Dash', 'dash@example.com', 'HR', 'Manager'),

(2, 'Aman', 'aman@example.com', 'IT', 'Developer');

-- (Repeat or use a script to generate 40-50 entries)

1. Demonstrate the update command by updating the data in the tables studentDetails, employee, salary, course (Single record update, Multiple records update )

**Explanation:**

We will use UPDATE command here , first we will access the table and update the data

**Solution:**

UPDATE Employee SET Department = 'Finance' WHERE EmployeeID = 1;

UPDATE Employee SET Department = 'General' WHERE Department = 'HR';

1. Demonstrate the delete command by deleting the data in the tables studentDetails, employee, salary, course (Single record update, Multiple records update )

**Solution:**

DELETE FROM Employee WHERE EmployeeID = 2;

DELETE FROM Employee WHERE Department = 'General';

1. Demonstrate the difference between delete, truncate and drop commands with examples and explain its business usages.

**Solution:**

Difference: DELETE (removes rows with condition), TRUNCATE (all rows, resets identity), DROP (removes table).

-- DELETE FROM Employee WHERE Department = 'IT';

-- TRUNCATE TABLE Employee;

-- DROP TABLE Employee;

1. Demonstrate the REPLACE / UPSERT functionality with examples

**Solution:**

INSERT INTO Employee (EmployeeID, Name, Email, Department, Designation)

VALUES (1, 'Asutosh Dash', 'dash@newmail.com', 'Developer', 'Lead')

ON DUPLICATE KEY UPDATE Name = 'Dash Updated', Email = 'dash@newmail.com';

1. Demonstrate the keys(primary, foreign and composite) by creating the tables Students, Courses and departments

**Solution:**

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

CREATE TABLE Courses (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100),

Credits INT

);

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

FirstName VARCHAR(50),

DepartmentID INT,

FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

);

-- Composite Key

CREATE TABLE StudentCourses (

StudentID INT,

CourseID INT,

PRIMARY KEY (StudentID, CourseID)

);

1. Demonstrate SELECT query for the below conditions

select single column

select multiple columns

select all columns

select using WHERE conditions

select using ORDER BY

select using LIMIT and fetch

select using BETWEEN both numeric and non numeric

select using IN

select using LIKE

**Solution:**

SELECT FirstName FROM Students;

SELECT FirstName, DepartmentID FROM Students;

SELECT \* FROM Students;

SELECT \* FROM Students WHERE DepartmentID = 1;

SELECT \* FROM Students ORDER BY FirstName;

SELECT \* FROM Students LIMIT 5;

SELECT \* FROM Courses WHERE Credits BETWEEN 3 AND 5;

SELECT \* FROM Students WHERE FirstName IN ('John', 'Jane');

SELECT \* FROM Students WHERE FirstName LIKE 'J%';

1. 19) Demonstrate SELECT using all the aggregate functions & aggregate function with GROUP BY and HAVING

**Solution:**

SELECT COUNT(\*) FROM STUDENTS;

SELECT AVG(CREDITS) FROM COURSE;

SELECT DEPARTID ,COUNT(\*) FROM GRUOP BY DEPARTID HAVING COUNT(\*)>1;

1. 20) Demonstrate SELECT using DISTINCT keyword

**Solution:**  
SELECT DISTINCT DepartmentID FROM Students;

1. 21) Demonstrate the SELECT using with subqueries.

**Solution:**

SELECT \* FROM STUDENTS WHERE DEPARTMENTID= (SELECT DEPARTMENTID FROM DEPARTMENT WHERE DEPARTMENT="IT");

1. 22) Demonstrate Different type of JOINs with examples.

**Explanation:**

There are two types of joins one is outer join and another is inner join . With in inner join we have left join , right join and full join .

**Solution:**

SELECT \* FROM STUDENTS s JOIN DEPARTMENT d ON s.DEPARTMENTID = d.DEPARTMENTID;

SELECT \* FROM STUDENTS s LEFT JOIN DEPARTMENT d ON s.DEPARTMENTID=d.DEPARTMENTID;

SELECT \* FROM STUDENTS s RIGHT JOIN DEPARTMENT d ON s.DEPARTMENTID=d.DEPARTMENTID;

SELECT \* FROM STUDENTS s FULL OUTER JOIN DEPARTMENT d ON s.DEPARTMENTID=d.DEPARTMENTID;

1. 23) Create the below tables and insert 20-30 records into the table.

Students(StudentID, FirstName, LastName, DateOfBirth, Email)

Courses(CourseID, CourseName, Credits)

Enrollments(EnrollmentID, StudentID, CourseID, EnrollmentDate)

**Solution:**

CREATE TABLE STUDENTS(

STUDENTSID INT PRIMARY KEY,

FIRSTNAME VARCHAR(100);

LASTNAME VARCHAR(100);

DATEOFBIRTH DATE,

EMAIL VARCHAR(100)

);

CREATE TABLE COURSE(

COURSEID INT PRIMARY KEY,

COURSENAME VARCHAR(100);

CREDITS VARCHAR(100);

);

CREATE TABLE ENROLLMENTS(

ENROLLMENTID INT PRIMARY KEY,

STUDENTID INT,

COURSEID INT,

ENROLLMETDATE DATE,

FOREIGN KEY (STUDENTID)REFERNCES STUDENTS(STUDENTID),

FOREIGN KEY (COURSEID)REFERNCES COURSE(COURSEID)

);

INSERT INTO STUDENTS(STUDENTID,FIRSTNAME,LASTNAME,DATEOFBIRTH,EMAIL)VALUES(120,'ASUTOSH','DASH','15/11/2002','EMAIL@GMAIL.COM');

INSERT INTO COURSE(COURSEID,COURSENAME,CREDITS)VALUES(11,'SQL','2');

INSERT INTO ENROLLMENTS(ENROLLMENTSID,ENROLLMENTDATE)VALUES(132,15-11-2025);

1. Write the queries for below questions.
2. Retrieve all students from the Students table.

**Solution:** SELECT \* FROM STUDENTS;

1. display only the FirstName and Email of all students.

**Solution:** SELECT FIRSTNAME,EMAIL FROM STUDENTS;

1. List all courses with exactly 4 credits.

**Solution:** select \* from course where credits=4;

1. Show all students born after January 1, 2000.

**Solution:**select \* from students where dateofbirth > '2000-01-01';

1. List students whose first name starts with "J".

**Solution:** select \* from students where firstname like'J%';

1. Find the number of students enrolled in each course.

**Solution:** select count(\*),Courseid from enrollments group by courseid;

1. Display students and their enrollment dates.

**Solution:** SELECT Students.FirstName, Enrollments.EnrollmentDate FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;

1. List unique course IDs from the Enrollments table.

**Solution:**

select distinct courseid from enrollment;

1. Find all students who are enrolled in CourseID 102.

**Solution:**

select \* from students s join enrollment e on s.studentid=e.studentid where e.courseid='102';

1. Display students ordered by LastName alphabetically.

**Solution:**

select lastname from students order by lastname desc;

1. List all students who have not enrolled in any course.

**Solution:**

select \* from students s join enrollment e on s.studentid=e.studentid where courseid is NULL;

1. Show the average number of credits across all courses.

**Solution:**

select avg(credits)from course;

1. List the top 3 most recently enrolled students.

**Solution:**

select firstname from students s join enrollment e on s.studntid=e.studnetid order by e.enrollmentdate desc limit 3;

1. Find all courses with names containing the word 'Data'.

**Solution:**

select \* from course where coursename like '%Data%';

1. Display the youngest student in the database.

**Solution:**

select \* from studnets order by dateofbirth asc limit 1;

1. Count the number of students with the same last name.

**Solution:**

select lastname,count(\*) from students group by lastname having count(\*)>1;

1. List courses along with the total number of enrolled students, sorted by count descending.

**Solution:**

select c.course , count(e.studentid) as enrolledCOunt from course c left join enerollment e on c.courseid=e.courseid group by c.courseName order by enrolledCount desc;

1. Find students who share the same first name.

**Solution:**

select firstname,count(\*) from students group by firstname having count(\*)>1;

1. Display enrollments made in the current year.

**Solution:**

SELECT \* FROM Enrollments WHERE YEAR(EnrollmentDate) = YEAR(CURDATE());

1. List students along with the names of the courses they are enrolled in.

**Solution:**

SELECT s.FirstName, c.CourseName FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

JOIN Courses c ON e.CourseID = c.CourseID;

1. Using joins
2. List all students along with the courses they are enrolled in.
3. List all courses and the students enrolled in them (include courses with no enrollments).
4. Show all students who have not enrolled in any course.
5. List course names along with the count of enrolled students using JOIN.
6. Show each student's full name and all course names they are enrolled in, ordered by student nAnswer:
7. List students and enrollment details using INNER JOIN.
8. List all courses with student names using LEFT JOIN.
9. List student names and their course names using RIGHT JOIN (if supported by your DBMS).
10. Show students and courses using FULL OUTER JOIN (if supported).
11. Find students who are enrolled in both 'Database Systems' and 'Data Structures'.

**Solution For 25:**

25. Using joins

-- a)

SELECT s.\*, c.CourseName FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

JOIN Courses c ON e.CourseID = c.CourseID;

-- b)

SELECT c.\*, s.FirstName FROM Courses c

LEFT JOIN Enrollments e ON c.CourseID = e.CourseID

LEFT JOIN Students s ON e.StudentID = s.StudentID;

-- c)

SELECT \* FROM Students WHERE StudentID NOT IN (SELECT StudentID FROM Enrollments);

-- d)

SELECT c.CourseName, COUNT(e.StudentID) FROM Courses c

JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY c.CourseName;

-- e)

SELECT CONCAT(s.FirstName, ' ', s.LastName) AS FullName, c.CourseName FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

JOIN Courses c ON e.CourseID = c.CourseID

ORDER BY FullName;

-- f)

SELECT s.\*, e.\* FROM Students s

INNER JOIN Enrollments e ON s.StudentID = e.StudentID;

-- g)

SELECT c.CourseName, s.FirstName FROM Courses c

LEFT JOIN Enrollments e ON c.CourseID = e.CourseID

LEFT JOIN Students s ON e.StudentID = s.StudentID;

-- h)

-- Only valid if RIGHT JOIN is supported:

SELECT s.FirstName, c.CourseName FROM Courses c

RIGHT JOIN Enrollments e ON c.CourseID = e.CourseID

RIGHT JOIN Students s ON s.StudentID = e.StudentID;

-- i)

-- FULL OUTER JOIN

-- SELECT \* FROM Students s FULL OUTER JOIN Enrollments e ON s.StudentID = e.StudentID;

-- j)

SELECT StudentID FROM Enrollments WHERE CourseID IN (

SELECT CourseID FROM Courses WHERE CourseName = 'Database Systems'

) INTERSECT

SELECT StudentID FROM Enrollments WHERE CourseID IN (

SELECT CourseID FROM Courses WHERE CourseName = 'Data Structures'

);