Two-Table Queries with Foreign Keys

Introduction -

In SQL, real-world data rarely exists in a single table. Businesses store customers, orders, employees, products—each in its own table. To extract meaningful insights, we need to combine them. That's where **JOINs** and **foreign keys** come into play. In this document, we'll go from understanding basic table relationships to writing powerful multi-table queries using `INNER JOIN`, `LEFT JOIN`, and more.

Step 1: Understand Relationships Between Tables

Concepts:

- **Primary Key:** Uniquely identifies a row.
- **Foreign Key:** A reference to a primary key in another table.
- **One-to-Many Relationship:** Most common (e.g., one student → many enrollments).

```sql
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    name VARCHAR(100)
);

CREATE TABLE enrollments (
    enrollment_id INT PRIMARY KEY,
    student_id INT,
    course_name VARCHAR(100),
    FOREIGN KEY (student_id) REFERENCES students(student_id)
);

```

Description :

- Here we have two tables [students and enrollments]. Now we need to attach both the tables as student table have student data and enrollments have data about the courses that students have enrolled.
- Now we have a primary key in student table which is unique which can be assigned to enrollments so that student will be mapped to corresponding enrollment using student id.
- Pay close attention on second query student_id is a foreign key which should be added.

Step 2: Insert Sample Data

```sql
INSERT INTO students VALUES (1, 'Alice'), (2, 'Bob');

INSERT INTO enrollments VALUES
(101, 1, 'Math'),
(102, 1, 'Science'),
(103, 2, 'Math');
```

Step 3: INNER JOIN - The Heart of Two-Table Queries

Objective: Learn how to combine related data from two tables.

👉 **Concepts to Cover:**

- Matching rows using `INNER JOIN`
- ON clause (how keys relate)

-
  ```sql
  SELECT s.name, e.course_name
  FROM students s
  JOIN enrollments e ON s.student_id = e.student_id;
  ```

Step 4: LEFT JOIN - Keep All from Left Table

Objective: Understand optional relationships (some students may not have enrollments).

```sql
SELECT s.name, e.course_name
FROM students s
LEFT JOIN enrollments e ON s.student_id = e.student_id;
```

Step 5: Aggregate with Joins

Objective: Combine joins with `GROUP BY`, `COUNT()`, etc.

```sql
SELECT s.name, COUNT(e.enrollment_id) AS total_courses
FROM students s
LEFT JOIN enrollments e ON s.student_id = e.student_id
GROUP BY s.name;
```

Step 8: Filtering with JOINs

Objective: Use `WHERE` in multi-table context.

```sql
SELECT s.name, e.course_name
FROM students s
JOIN enrollments e ON s.student_id = e.student_id
WHERE e.course_name = 'Math';
```

✅ *Practice Questions with Solutions*

🔹 **1. List all students with their enrolled courses.**

```sql
SELECT s.name, e.course_name FROM students s JOIN enrollments e
  ON s.student_id = e.student_id;
```

🔹 **2. Show all students, including those who are not enrolled in any course.**

```sql
SELECT s.name, e.course_name FROM students s LEFT JOIN enrollments e
ON s.student_id = e.student_id;
```

> 🧠 `LEFT JOIN` keeps all students, even if no matching enrollments.

🔹 **3. Show students who are not enrolled in any course.**

```sql
SELECT s.name FROM students s LEFT JOIN enrollments e ON s.student_id =
  e.student_id
WHERE e.enrollment_id IS NULL;
```

🔹 **4. Count the number of courses each student is enrolled in.**

```sql
SELECT s.name, COUNT(e.course_name) AS course_count FROM students s LEFT JOIN
enrollments e ON s.student_id = e.student_id
GROUP BY s.name;
```

🔹 **5. Find all students who are enrolled in "Math".**

```sql
SELECT s.name FROM students s JOIN enrollments e ON s.student_id = e.student_id
WHERE e.course_name = 'Math';
```

🔹 **6. List all courses and the number of students enrolled in each.**

```sql
SELECT e.course_name, COUNT(DISTINCT e.student_id) AS student_count
FROM enrollments e GROUP BY e.course_name;
```

🔹 **7. Find students who are enrolled in more than one course.**

```sql
SELECT s.name FROM students s JOIN enrollments e ON s.student_id = e.student_id
GROUP BY s.name HAVING COUNT(e.course_name) > 1;
```

🔹 **8. List student names along with the first course they enrolled in (based on `enrollment_id`).**

```sql
SELECT s.name, e.course_name FROM students s JOIN enrollments e
ON s.student_id = e.student_id
WHERE e.enrollment_id = (SELECT MIN(enrollment_id)
FROM enrollments e2     WHERE e2.student_id = s.student_id );
```

🔹 **9. Get a list of students and their total number of enrollments, sorted by highest to lowest.**

```sql
SELECT s.name, COUNT(e.enrollment_id) AS total_enrollments
FROM students s LEFT JOIN enrollments e ON s.student_id =
e.student_id GROUP BY s.name ORDER BY total_enrollments DESC;
```

🔹 **10. Show course names that have more than one student enrolled.**

```sql
SELECT course_name FROM enrollments GROUP BY course_name
HAVING COUNT(DISTINCT student_id) > 1;
```

Practice Problems

Objective: Reinforce learning through applied problems.

📝 **Sample Practice Questions:**

1. List all students and their courses.
2. Show students who are not enrolled in any course.
3. Find how many courses each student is enrolled in.
4. Find courses taken by more than one student.
5. Show students enrolled in “Math” or “Science”.

Document by ***Suyash***😇