**Advanced SQL Queries with Joins and Filtering - Documentation**

**Objective**

This project focuses on analyzing relationships between multiple tables using SQL joins and filtering techniques to extract meaningful insights.

**Step 1: Database Setup**

**1.1 Purpose of Queries**

* **Courses Table**: To define the available courses, including their names and descriptions.
* **Enrolments Table**: To record the enrolment data linking students to courses.

**Queries**

* **Create Courses Table**:
* CREATE TABLE Courses (
* course\_id INT AUTO\_INCREMENT PRIMARY KEY,
* course\_name VARCHAR(100),
* course\_description TEXT

);

* **Create Enrolments Table**:
* CREATE TABLE Enrolments (
* enrolment\_id INT AUTO\_INCREMENT PRIMARY KEY,
* student\_id INT,
* course\_id INT,
* enrolment\_date DATE,
* FOREIGN KEY (student\_id) REFERENCES Students(StudentID),
* FOREIGN KEY (course\_id) REFERENCES Courses(course\_id)

);

**Observations**

* These tables establish the relationships between students and courses, enabling complex queries.

**Step 2: Tasks to Perform**

**Task 1: List all students and the courses they are enrolled in**

**Purpose**: To display the names of students and the courses they are enrolled in.

**Query**:

SELECT

s.Name AS StudentName,

c.course\_name AS CourseName

FROM

Students s

INNER JOIN

Enrolments e ON s.StudentID = e.student\_id

INNER JOIN

Courses c ON e.course\_id = c.course\_id;

**Explanation**:

* The query uses INNER JOIN to combine Students, Enrolments, and Courses tables.
* Filters data to show only students enrolled in courses.

**Observations**:

* Provides a complete list of enrolled students with their respective courses.

**Task 2: Find the number of students enrolled in each course**

**Purpose**: To determine the enrolment count for each course, including courses with no enrolments.

**Query**:

SELECT

c.course\_name AS CourseName,

COUNT(e.student\_id) AS EnrolledStudents

FROM

Courses c

LEFT JOIN

Enrolments e ON c.course\_id = e.course\_id

GROUP BY

c.course\_id, c.course\_name;

**Explanation**:

* Uses LEFT JOIN to include all courses, even those without enrolments.
* Groups results by course to calculate enrolment count.

**Observations**:

* Identifies popular courses and those without students.

**Task 3: List students who have enrolled in more than one course**

**Purpose**: To highlight students with multiple enrolments.

**Query**:

SELECT

s.Name AS StudentName,

COUNT(e.course\_id) AS CourseCount

FROM

Students s

INNER JOIN

Enrolments e ON s.StudentID = e.student\_id

GROUP BY

s.StudentID, s.Name

HAVING

COUNT(e.course\_id) > 1;

**Explanation**:

* Groups enrolments by student to count the courses each has enrolled in.
* Uses HAVING to filter students with more than one course.

**Observations**:

* Identifies students actively participating in multiple courses.

**Task 4: Find courses with no enrolled students**

**Purpose**: To list courses without any enrolments.

**Query**:

SELECT

c.course\_name AS CourseName

FROM

Courses c

LEFT JOIN

Enrolments e ON c.course\_id = e.course\_id

WHERE

e.enrolment\_id IS NULL;

**Explanation**:

* Uses LEFT JOIN to include all courses and filters for those with no enrolments.

**Observations**:

* Highlights courses needing promotional efforts or adjustments.

**Deliverables**

**1. SQL Queries**

* All tasks’ queries are provided above.

**2. Observations and Insights**

* **Task 1**: Detailed enrolment data linking students to courses.
* **Task 2**: Enrolment counts reveal course popularity and gaps.
* **Task 3**: Identifies students engaging with multiple courses.
* **Task 4**: Highlights underutilized courses.

**3. Screenshots of Results**

* Execute each query and capture the outputs for reporting.

**Conclusion**

This project demonstrated the use of advanced SQL techniques, such as joins and filtering, to analyze relationships between multiple tables. The insights gained can drive better decision-making in course management and student engagement strategies.