Assignment 3:

```
-- Create Students table
CREATE TABLE Students (
  student id INT PRIMARY
KEY.
  student name
VARCHAR(100),
  student major
VARCHAR(100)
);
-- Create Courses table
CREATE TABLE Courses (
  course_id INT PRIMARY KEY,
  course_name
VARCHAR(100),
  course description
VARCHAR(255)
);
-- Create Enrollments table
CREATE TABLE Enrollments (
  enrollment id INT PRIMARY
KEY,
  student id INT,
  course id INT,
```

```
enrollment_date DATE,
FOREIGN KEY (student_id)
REFERENCES
Students(student_id),
FOREIGN KEY (course_id)
REFERENCES
Courses(course_id)
);
```

- -- Insert data into Students table INSERT INTO Students (student_id, student_name, student_major) VALUES (1, 'Alice', 'Computer Science'), (2, 'Bob', 'Biology'), (3, 'Charlie', 'History'), (4, 'Diana', 'Mathematics');
- -- Insert data into Courses table INSERT INTO Courses (course_id, course_name, course_description) VALUES (101, 'Introduction to CS', 'Basics of Computer Science'),

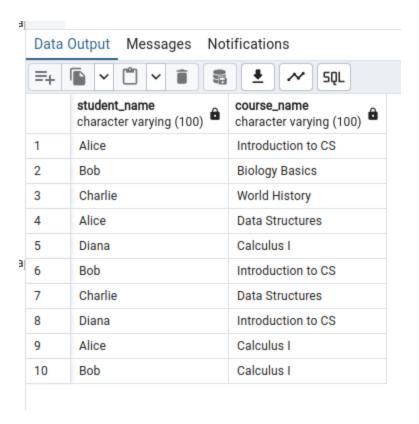
(102, 'Biology Basics',
'Fundamentals of Biology'),
(103, 'World History', 'Historical
events and cultures'),
(104, 'Calculus I', 'Introduction to
Calculus'),
(105, 'Data Structures',
'Advanced topics in CS');

-- Insert data into Enrollments table **INSERT INTO Enrollments** (enrollment id, student id, course_id, enrollment_date) VALUES (1, 1, 101, '2023-01-15'), (2, 2, 102, '2023-01-20'), (3, 3, 103, '2023-02-01'), (4, 1, 105, '2023-02-05'), (5, 4, 104, '2023-02-10'), (6, 2, 101, '2023-02-12'), (7, 3, 105, '2023-02-15'), (8, 4, 101, '2023-02-20'), (9, 1, 104, '2023-03-01'), (10, 2, 104, '2023-03-05');

1. Inner Join:

Question: Retrieve the list of students and their enrolled courses.

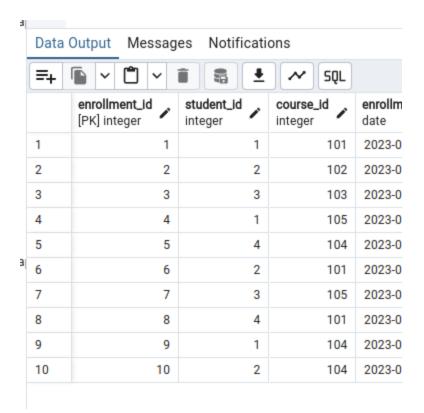
```
SELECT
    student_name,
    course_name
FROM
    students S
    INNER JOIN enrollments E ON E.student_id = S.student_id
    INNER JOIN courses C ON C.course_id = E.course_id;
```



2. Left Join:

Question: List all students and their enrolled courses, including those who haven't enrolled in any course.

```
SELECT
    student_name,
    course_name
FROM
    students S
    LEFT JOIN enrollments E ON E.student_id = S.student_id
    LEFT JOIN courses C ON C.course_id = E.course_id;
```

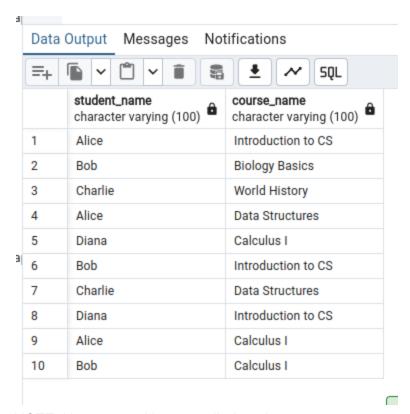


NOTE: No student without enrollment

3. Right Join:

Question: Display all courses and the students enrolled in each course, including courses with no enrolled students.

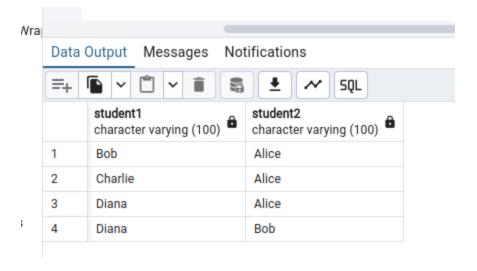
```
SELECT
    student_name,
    course_name
FROM
    students S
    RIGHT JOIN enrollments E ON E.student_id = S.student_id
    RIGHT JOIN courses C ON C.course_id = E.course_id;
```



NOTE: No course without enrolled students

4. Self Join:

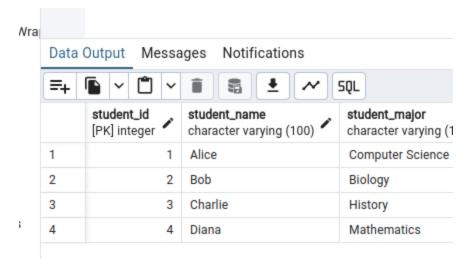
Question: Find pairs of students who are enrolled in at least one common course.



5. Complex Join:

Question: Retrieve students who are enrolled in 'Introduction to CS' but not in 'Data Structures'.

```
SELECT
    student_name
FROM
    students S
    INNER JOIN enrollments E ON s.student_id = E.student_id
    INNER JOIN courses C on C.course_id = E.course_id
WHERE
    course_name = 'Introduction to CS'
    AND S.student_id NOT IN (
        SELECT
            student_id
        FROM
            enrollments E
            INNER JOIN courses C ON C.course_id = E.course_id
        WHERE
            course_name = 'Data Structures'
    );
```



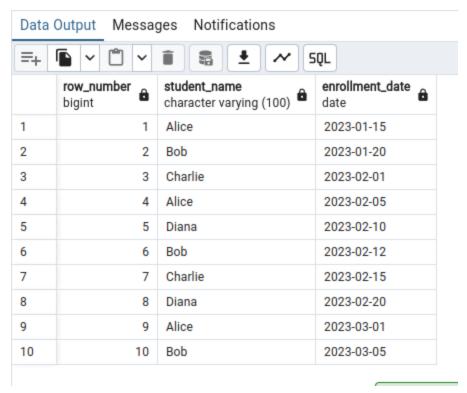
Windows function:

1. Using ROW_NUMBER():

Question: List all students along with a row number based on their enrollment date in ascending order.

```
Select
   ROW_NUMBER() OVER (
        ORDER BY
        enrollment_date
   ) as row_number,
   student_name,
   enrollment_date

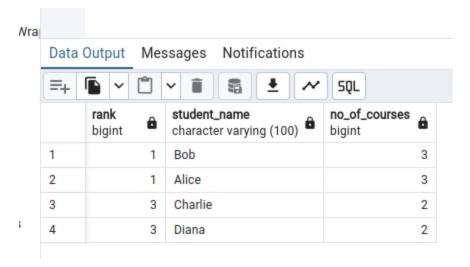
from
   students S
   INNER JOIN enrollments E ON S.student_id = E.student_id
```



2. Using RANK():

Question: Rank students based on the number of courses they are enrolled in, handling ties by assigning the same rank.

```
SELECT
    RANK() OVER (
        ORDER BY
        COUNT(course_id) DESC
    ) AS rank,
    student_name,
    COUNT(course_id) AS no_of_courses
FROM
    students S
    INNER JOIN enrollments E ON S.student_id = E.student_id
GROUP BY
    S.student_id;
```



3. Using DENSE_RANK():

Question: Determine the dense rank of courses based on their enrollment count across all students

ıa	a								
	Data	Output Mes	ssages Notifications						
	=+	• ~ •	v i	SQL					
		rank bigint	course_name character varying (100)	no_of_students bigint					
	1	1	Introduction to CS	3					
	2	1	Calculus I	3					
	3	2	Data Structures	2					
	4	3	Biology Basics	1					
rai	5	3	World History	1					