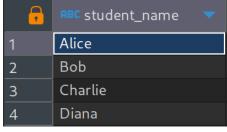
```
CREATE SCHEMA IF NOT EXISTS assignment3;
CREATE TABLE
IF NOT EXISTS assignment3.Students (
  student id INT PRIMARY KEY,
  student_name VARCHAR(100),
 student_major VARCHAR(100)
 );
-- Create Courses table
CREATE TABLE
IF NOT EXISTS assignment3.Courses (
 course_id INT PRIMARY KEY,
 course name VARCHAR(100),
 course description VARCHAR(255)
 );
-- Create Enrollments table
CREATE TABLE
IF NOT EXISTS assignment3.Enrollments (
  enrollment_id INT PRIMARY KEY,
  student id INT,
 course id INT,
  enrollment_date DATE,
  FOREIGN KEY (student_id) REFERENCES assignment3.Students (student_id),
  FOREIGN KEY (course id) REFERENCES assignment3.Courses (course id)
 );
INSERT INTO
 assignment3.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
assignment3. Course (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
assignment3.Enrollments (
  enrollment 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');
```

```
-- Retrieve the list of students and their enrolled courses.
SELECT student_name
FROM assignment3.students s
JOIN assignment3.enrollments e
ON e.student_id = s.student_id
GROUP BY s.student_name;
```



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

SELECT student_name, c.course_name

FROM assignment3.students s

LEFT JOIN assignment3.enrollments e

ON s.student_id = e.student_id

LEFT JOIN assignment3.courses c

ON e.course_id = c.course_id

ORDER BY s.student_name;
```

•	ABC student_name T	RBC course_name T
1	Alice	Data Structures
2	Alice	Introduction to CS
3	Alice	Calculus I
4	Bhuban Bam	[NULL]
5	Bob	Calculus I
6	Bob	Biology Basics
7	Bob	Introduction to CS
8	Charlie	Data Structures
9	Charlie	World History
10	Diana	Introduction to CS
11	Diana	Calculus I

```
-- Display all courses and the students enrolled in each course,
-- including courses with no enrolled students.

SELECT c.course_name, student_name

FROM assignment3.students s

RIGHT JOIN assignment3.enrollments e

ON s.student_id = e.student_id

RIGHT JOIN assignment3.courses c

ON e.course_id = c.course_id

ORDER BY c.course_name;
```

•	RBC course_name	RBC student_name ▼
1	Biology Basics	Bob
2	Calculus I	Bob
3	Calculus I	Alice
4	Calculus I	Diana
5	Data Structures	Charlie
6	Data Structures	Alice
7	Introduction to CS	Bob
8	Introduction to CS	Alice
9	Introduction to CS	Diana
10	World History	Charlie

```
-- Find pairs of students who are enrolled in at least one common course.
SELECT DISTINCT s.student_name AS student_1 , s2.student_name AS student_2
FROM assignment3.enrollments e
JOIN assignment3.enrollments e2 ON e.course_id = e2.course_id
JOIN assignment3.students s
ON e.student_id = s.student_id
JOIN assignment3.students s2
ON e2.student id = s2.student id
WHERE e.student_id < e2.student_id</pre>
   RBC student_1
                       RBC student_2
      Alice
                       Diana
      Bob
                       Diana
      Alice
                       Bob
      Alice
                       Charlie
```

```
'Data Structures'.
SELECT s.student_name
FROM assignment3.students s
JOIN assignment3.enrollments e
ON s.student id = e.student id
JOIN assignment3.courses c
ON e.course id = c.course id
WHERE c.course_name = 'Introduction to CS'
AND s.student id NOT IN (
  SELECT e2.student_id
  FROM assignment3.enrollments e2
  JOIN assignment3.courses c2
  ON e2.course id = c2.course id
  WHERE c2.course_name = 'Data Structures'
);
        RBC student_name
        Bob
        Diana
```

	MBC student_name	HBC course_name	123 row_number_count	
1	Alice	Introduction to CS		1
2	Bob	Biology Basics		2
3	Charlie	World History		3
4	Alice	Data Structures		4
5	Diana	Calculus I		5
6	Bob	Introduction to CS		6
7	Charlie	Data Structures		7
8	Diana	Introduction to CS		8
9	Alice	Calculus I		9
10	Bob	Calculus I	1	0

```
- Rank students based on the number of courses they are enrolled in,
-- handling ties by assigning the same rank. Using RANK()
SELECT s.student_name,
COUNT(e.course_id) AS course_count,
RANK () OVER(
      ORDER BY COUNT(e.course id) DESC
FROM assignment3.enrollments e
JOIN assignment3.students s
ON s.student id = e.student id
GROUP BY s.student_i
     RBC student_name
                        123 course_count
                                           123 rank
      Bob
      Charlie
      Diana
```

```
-- Rank students based on the number of courses they are enrolled in,
SELECT c.course_name,
COUNT(e.student_id) AS student_count,
DENSE_RANK () OVER(
      ORDER BY COUNT(e.student_id) DESC
FROM assignment3.enrollments e
JOIN assignment3.courses c
ON e.course_id = c.course_id
GROUP BY c.course_name
                   ▼ 123 student_count
                                     ▼ 123 dense_rank
  Gourse_name
     Calculus I
     Introduction to CS
     Data Structures
     World History
     Biology Basics
```