Q.1 Retrieve the list of students and their enrolled courses.

```
select s.student_name, c.course_name from students s
join enrollments e on s.student_id = e.student_id
join courses c on c.course_id = e.course_id order by s.student_name;
```

*<DB_Assignment1> Script-11 × students enrollments courses

```
--Q.1
select s.student_name, c.course_name from students s
join enrollments e on s.student_id = e.student_id
join courses c on c.course_id = e.course_id order by s.student_name;
```

students(+) 1 ×

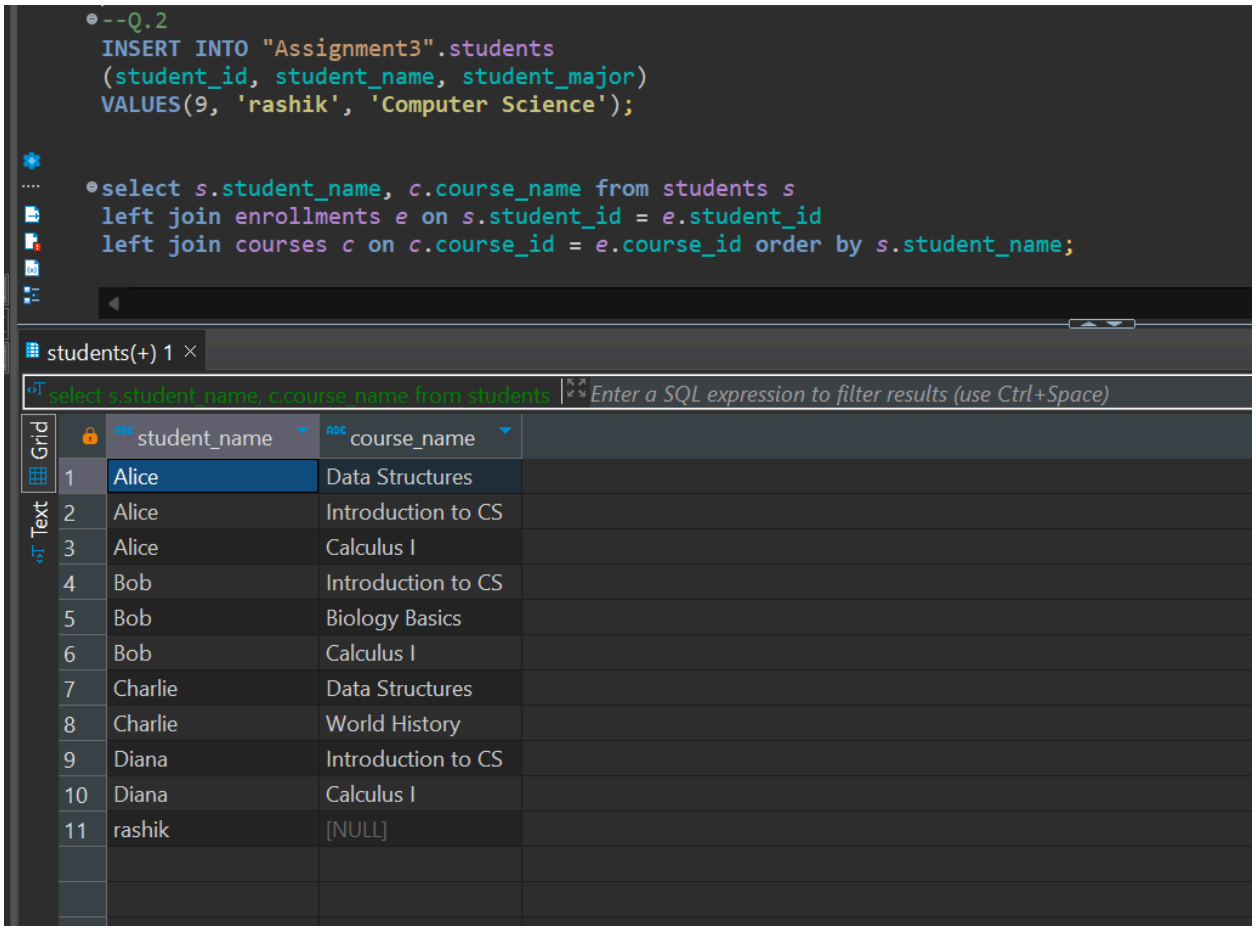
select s.student_name, c.course_name from students | Enter a SQL expression to filter results (use Ctrl+Space)

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

Q.2 List all students and their enrolled courses, including those who haven't enrolled in any course.

```
INSERT INTO "Assignment3".students
(student_id, student_name, student_major)
VALUES(9, 'rashik', 'Computer Science');


select s.student_name, c.course_name from students s
left join enrollments e on s.student_id = e.student_id
left join courses c on c.course_id = e.course_id order by s.student_name;
```



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

Q.3 Display all courses and the students enrolled in each course, including courses with no enrolled students.

```sql
INSERT INTO "Assignment3".courses
(course_id, course_name, course_description)
VALUES(5, 'English', 'Advanced English Grammar');


select s.student_name, c.course_name from students s
right join enrollments e on s.student_id = e.student_id
right join courses c on c.course_id = e.course_id order by s.student_name;
```

```sql
--Q.3
INSERT INTO "Assignment3".courses
(course_id, course_name, course_description)
VALUES(5, 'English', 'Advanced English Grammar');


select s.student_name, c.course_name from students s
right join enrollments e on s.student_id = e.student_id
right join courses c on c.course_id = e.course_id order by s.student_name;
```

students(+) 1

select s.student_name, c.course_name from students

Enter a SQL expression to filter results (use Ctrl+Space)

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

Q.4 Find pairs of students who are enrolled in at least one common course

```sql
SELECT
  (select student_name from students where student_id = e1.student_id),
  (select student_name from students where student_id = e2.student_id),
  c.course_name
FROM
  Enrollments e1
  JOIN Enrollments e2 ON e1.course_id = e2.course_id AND e1.student_id < e2.student_id
  JOIN Courses c ON e1.course_id = c.course_id
ORDER BY
        c.course_name;
```

```sql
--Q.4
SELECT
    (select student_name from students where student_id = e1.student_id),
    (select student_name from students where student_id = e2.student_id),
    c.course_name
FROM
    Enrollments e1
    JOIN Enrollments e2 ON e1.course_id = e2.course_id AND e1.student_id < e2.student_id
    JOIN Courses c ON e1.course_id = c.course_id
ORDER BY
    c.course_name;
```

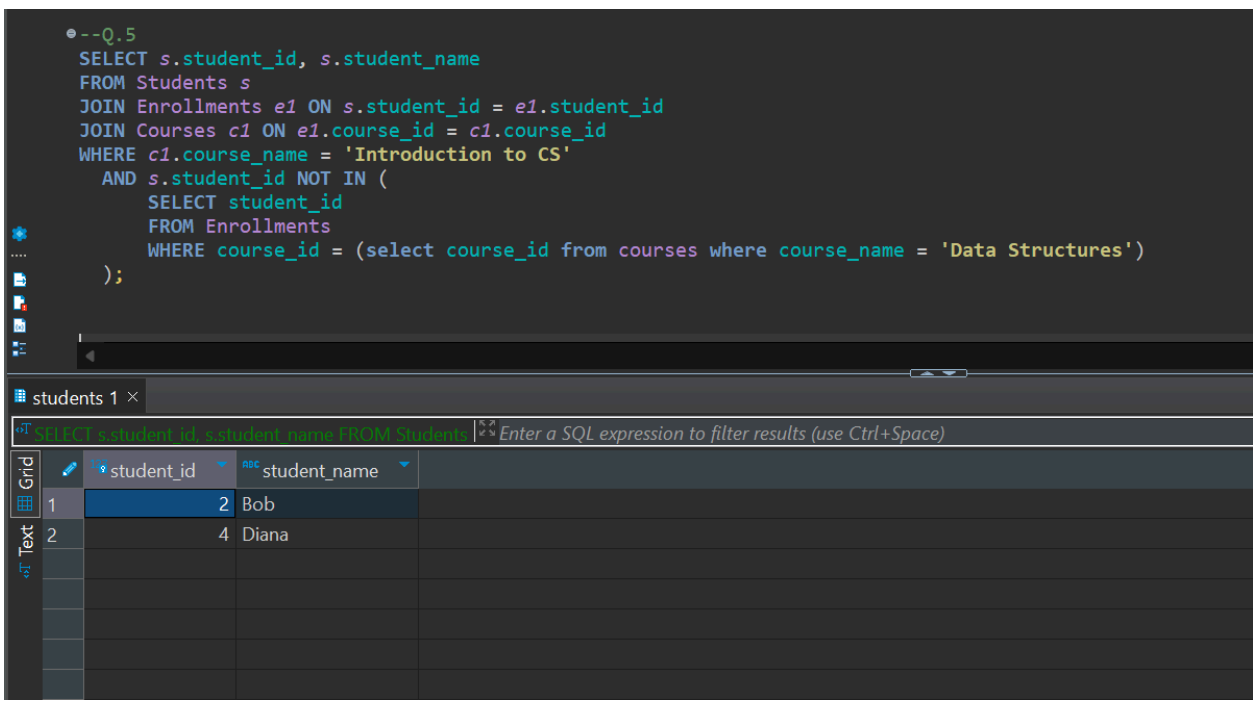
courses 1

SELECT (select student_name from students where stu | Enter a SQL expression to filter results (use Ctrl+Space)

	student_name	student_name	course_name
1	Alice	Diana	Calculus I
2	Alice	Bob	Calculus I
3	Bob	Diana	Calculus I
4	Alice	Charlie	Data Structures
5	Alice	Bob	Introduction to CS
6	Alice	Diana	Introduction to CS
7	Bob	Diana	Introduction to CS

Q.5 Retrieve students who are enrolled in 'Introduction to CS' but not in 'Data Structures'.

```
SELECT s.student_id, s.student_name
FROM Students s
JOIN Enrollments e1 ON s.student_id = e1.student_id
JOIN Courses c1 ON e1.course_id = c1.course_id
WHERE c1.course_name = 'Introduction to CS'
 AND s.student_id NOT IN (
    SELECT student_id
    FROM Enrollments
    WHERE course_id = (select course_id from courses where course_name = 'Data Structures')
);
```



Window function:

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

```sql
SELECT
  s.student_name,
  e.enrollment_date,
  ROW_NUMBER() OVER (ORDER BY e.enrollment_date ASC) AS row_num
FROM
  Enrollments e
  JOIN Students s ON e.student_id = s.student_id
ORDER BY
  e.enrollment_date ASC;
```

```sql
--Q.6
SELECT
    s.student_name,
    e.enrollment_date,
    ROW_NUMBER() OVER (ORDER BY e.enrollment_date ASC) AS row_num
FROM
    Enrollments e
    JOIN Students s ON e.student_id = s.student_id
ORDER BY
    e.enrollment_date ASC;
```

students(+) 1

SELECT s.student_name, e.enrollment_date, ROW_NU | Enter a SQL expression to filter results (use Ctrl+Space)

	student_name	enrollment_date	row_num
1	Alice	2023-01-15	1
2	Bob	2023-01-20	2
3	Charlie	2023-02-01	3
4	Alice	2023-02-05	4
5	Diana	2023-02-10	5
6	Bob	2023-02-12	6
7	Charlie	2023-02-15	7
8	Diana	2023-02-20	8
9	Alice	2023-03-01	9
10	Bob	2023-03-05	10

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

```sql
SELECT
  s.student_id,
  s.student_name,
  COUNT(e.course_id) AS course_count,
  RANK() OVER (ORDER BY COUNT(e.course_id) DESC) AS rank
FROM
  Students s
  JOIN Enrollments e ON s.student_id = e.student_id
GROUP BY
  s.student_id, s.student_name
ORDER BY
  s.student_id;
```

```sql
--Q.7
SELECT
    s.student_id,
    s.student_name,
    COUNT(e.course_id) AS course_count,
    RANK() OVER (ORDER BY COUNT(e.course_id) DESC) AS rank
FROM
    Students s
    JOIN Enrollments e ON s.student_id = e.student_id
GROUP BY
    s.student_id, s.student_name
ORDER BY
    s.student_id;
```

students 1

SELECT s.student_id, s.student_name, COUNT(e.course | Enter a SQL expression to filter results (use Ctrl+Space)

	student_id	student_name	course_count	rank
1	1	Alice	3	1
2	2	Bob	3	1
3	3	Charlie	2	3
4	4	Diana	2	3

3. Dense rank: Determine the dense rank of courses based on their enrollment count across all students

```sql
SELECT
  c.course_id,
  c.course_name,
  COUNT(e.student_id) AS enrollment_count,
  DENSE_RANK() OVER (ORDER BY COUNT(e.student_id)) AS dense_rank
FROM
  Courses c
  JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY
  c.course_id, c.course_name
ORDER BY
  dense_rank, c.course_id;
```

```sql
--Q.8
SELECT
    c.course_id,
    c.course_name,
    COUNT(e.student_id) AS enrollment_count,
    DENSE_RANK() OVER (ORDER BY COUNT(e.student_id)) AS dense_rank
FROM
    Courses c
    JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY
    c.course_id, c.course_name
ORDER BY
    dense_rank, c.course_id;
```

courses 1

SELECT c.course_id, c.course_name, COUNT(e.student | Enter a SQL expression to filter results (use Ctrl+Space)

	course_id	course_name	enrollment_count	dense_rank
1	102	Biology Basics	1	1
2	103	World History	1	1
3	105	Data Structures	2	2
4	101	Introduction to CS	3	3
5	104	Calculus I	3	3