Assignment Report

Database Assignment 3

Prepared By:

Devraj Neupane

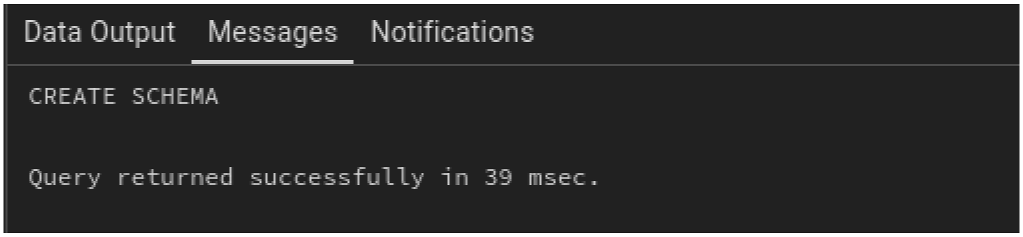
Roll No: 7

Group: D

Create assignment3 schema

```
CREATE SCHEMA IF NOT EXISTS assignment3 AUTHORIZATION postgres;
```

Screenshot:



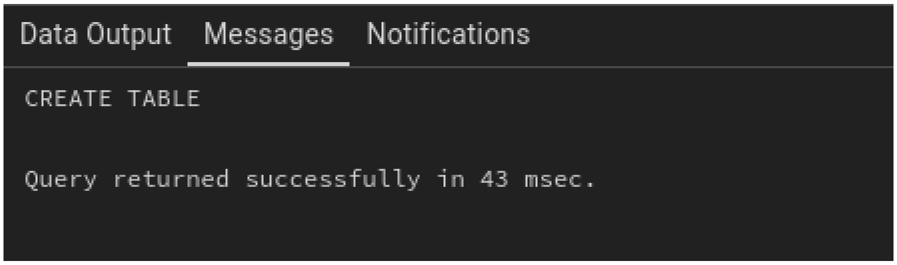
```
CREATE SCHEMA

Query returned successfully in 39 msec.
```

Create Students table

```
CREATE TABLE IF NOT EXISTS assignment3.Students (
  student_id INT PRIMARY KEY,
  student_name VARCHAR(100),
  student_major VARCHAR(100)
);
```

Screenshot:



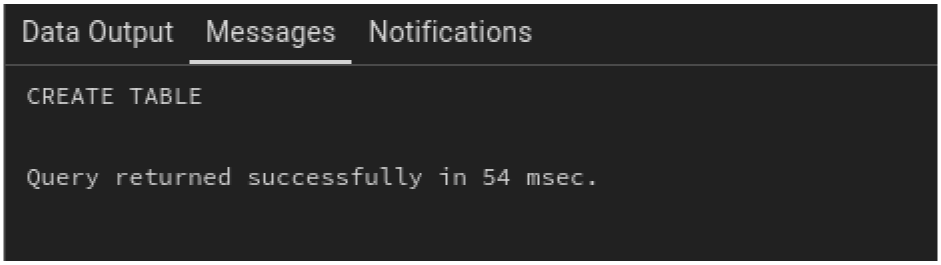
```
CREATE TABLE

Query returned successfully in 43 msec.
```

Create Courses table

```
CREATE TABLE IF NOT EXISTS assignment3.Courses (
  course_id INT PRIMARY KEY,
  course_name VARCHAR(100),
  course_description VARCHAR(255)
);
```

Screenshot:



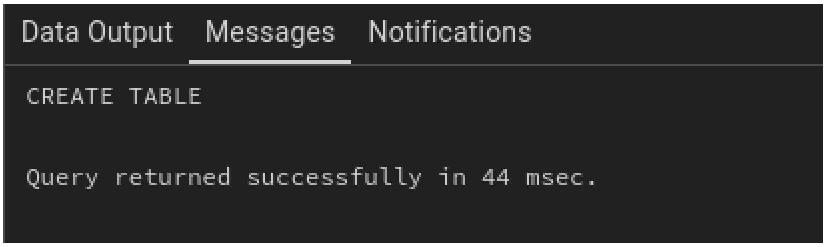
```
CREATE TABLE

Query returned successfully in 54 msec.
```

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)
);
```

Screenshot:



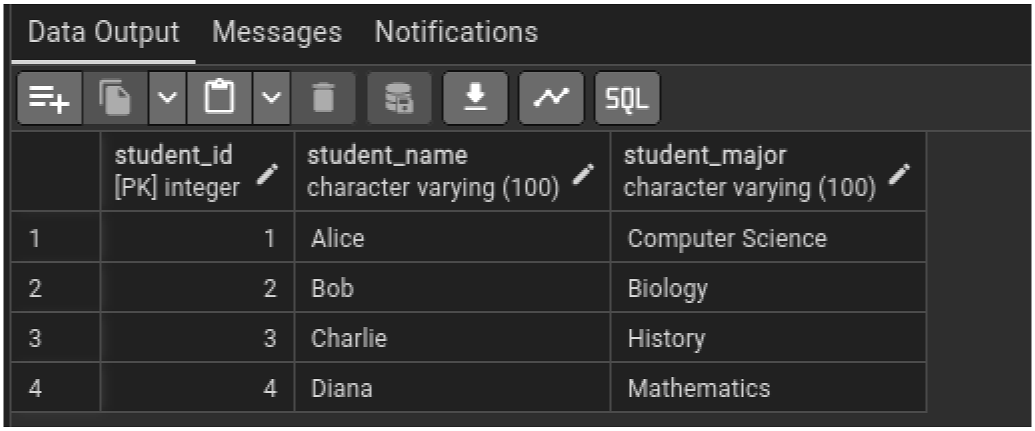
```
CREATE TABLE

Query returned successfully in 44 msec.
```

Insert data into Students table

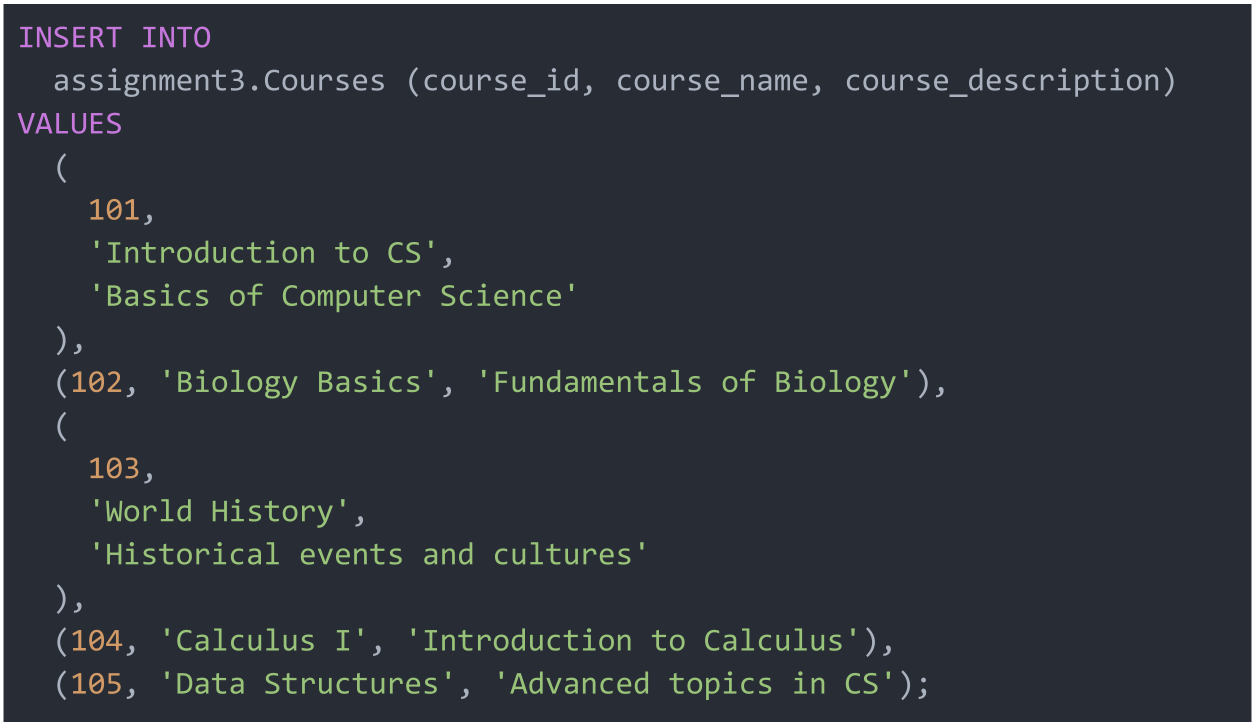
```
INSERT INTO
  assignment3.Students (student_id, student_name, student_major)
VALUES
  (1, 'Alice', 'Computer Science'),
  (2, 'Bob', 'Biology'),
  (3, 'Charlie', 'History'),
  (4, 'Diana', 'Mathematics');
```

Screenshot:



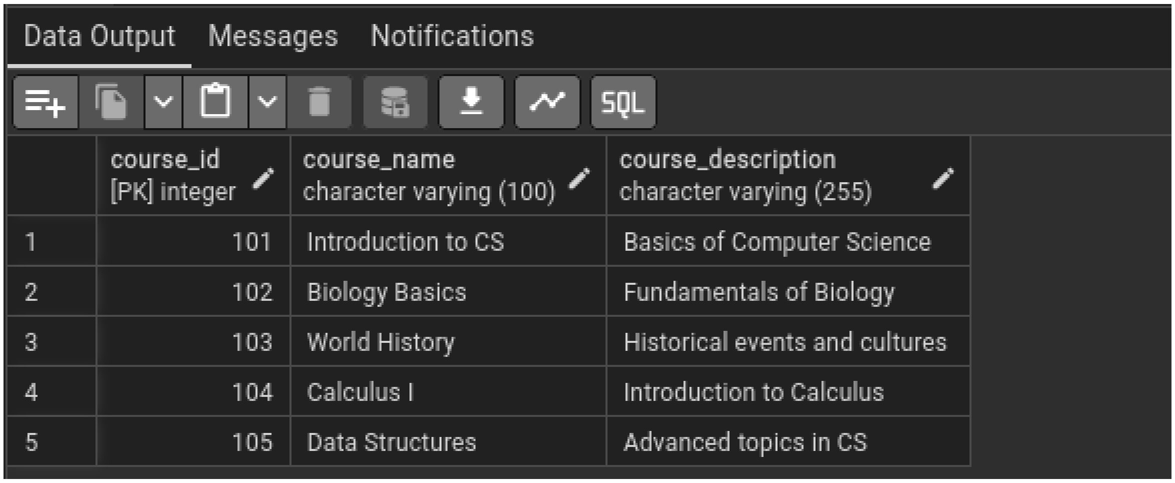
	student_id [PK] integer	student_name character varying (100)	student_major character varying (100)
1	1	Alice	Computer Science
2	2	Bob	Biology
3	3	Charlie	History
4	4	Diana	Mathematics

Insert data into Courses table



```sql
INSERT INTO
  assignment3.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');
```

Screenshot:

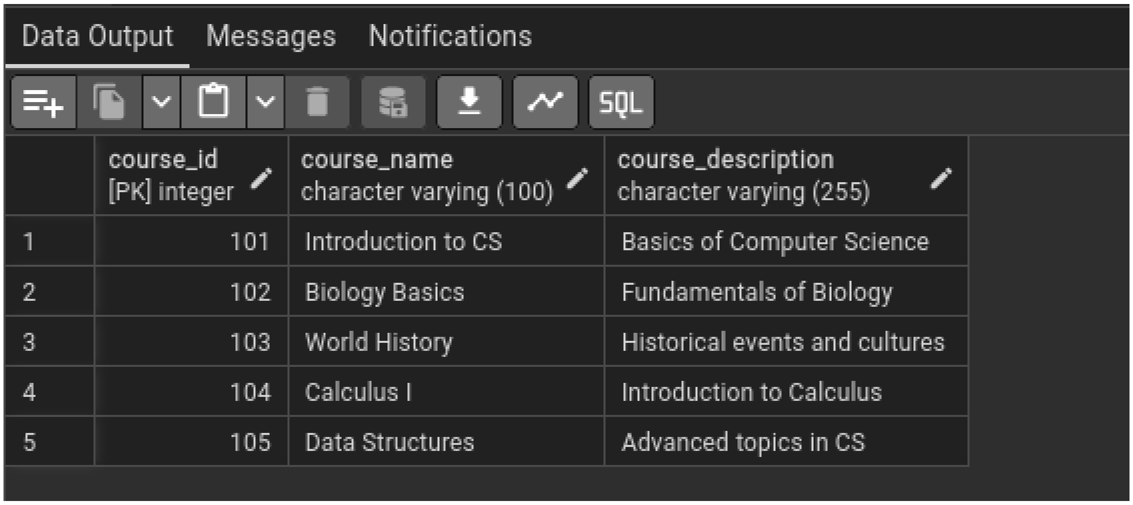


	course_id [PK] integer	course_name character varying (100)	course_description character varying (255)
1	101	Introduction to CS	Basics of Computer Science
2	102	Biology Basics	Fundamentals of Biology
3	103	World History	Historical events and cultures
4	104	Calculus I	Introduction to Calculus
5	105	Data Structures	Advanced topics in CS

Insert data into Enrollments table

```
INSERT INTO
  assignment3.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');
```

Screenshot:

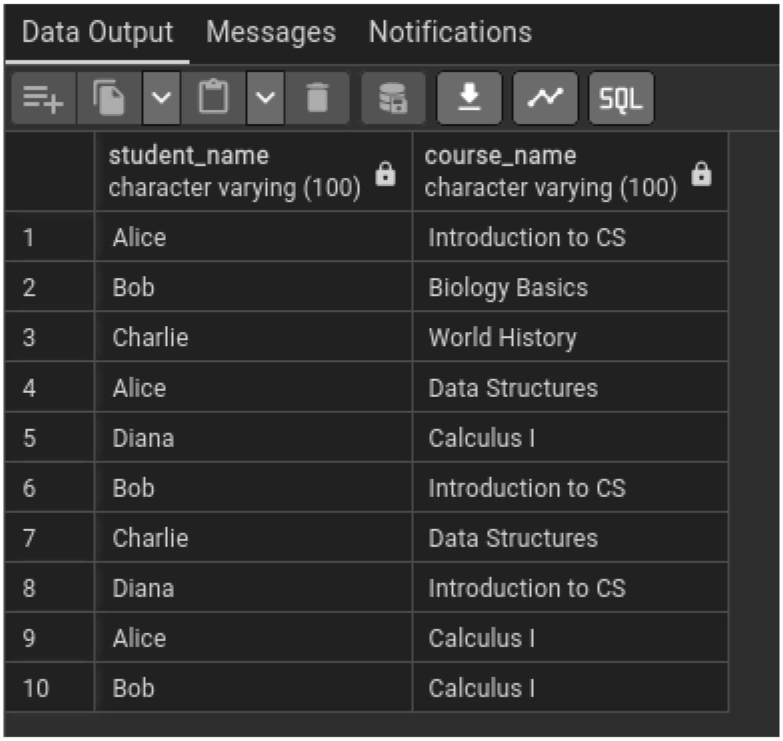


	course_id [PK] integer	course_name character varying (100)	course_description character varying (255)
1	101	Introduction to CS	Basics of Computer Science
2	102	Biology Basics	Fundamentals of Biology
3	103	World History	Historical events and cultures
4	104	Calculus I	Introduction to Calculus
5	105	Data Structures	Advanced topics in CS

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

```
SELECT
  s.student_name,
  c.course_name
FROM
  assignment3.students s
  JOIN assignment3.enrollments e ON e.student_id = s.student_id
  JOIN assignment3.courses c ON c.course_id = e.course_id;
```

Screenshot:

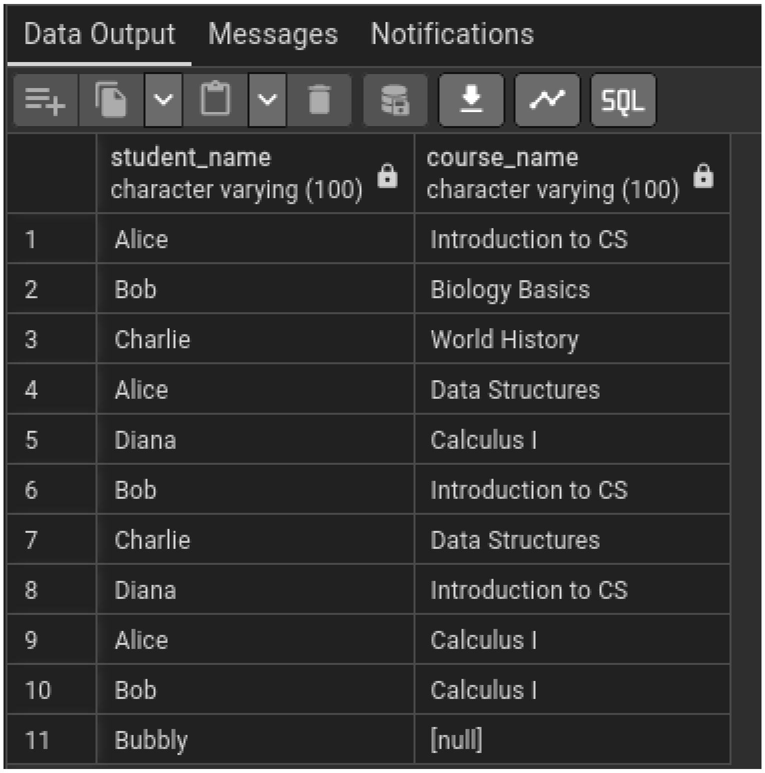


	student_name character varying (100)	course_name character varying (100)
1	Alice	Introduction to CS
2	Bob	Biology Basics
3	Charlie	World History
4	Alice	Data Structures
5	Diana	Calculus I
6	Bob	Introduction to CS
7	Charlie	Data Structures
8	Diana	Introduction to CS
9	Alice	Calculus I
10	Bob	Calculus I

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

```
SELECT
  s.student_name,
  c.course_name
FROM
  assignment3.students s
  LEFT JOIN assignment3.enrollments e ON e.student_id = s.student_id
  LEFT JOIN assignment3.courses c ON c.course_id = e.course_id;
```

Screenshot:

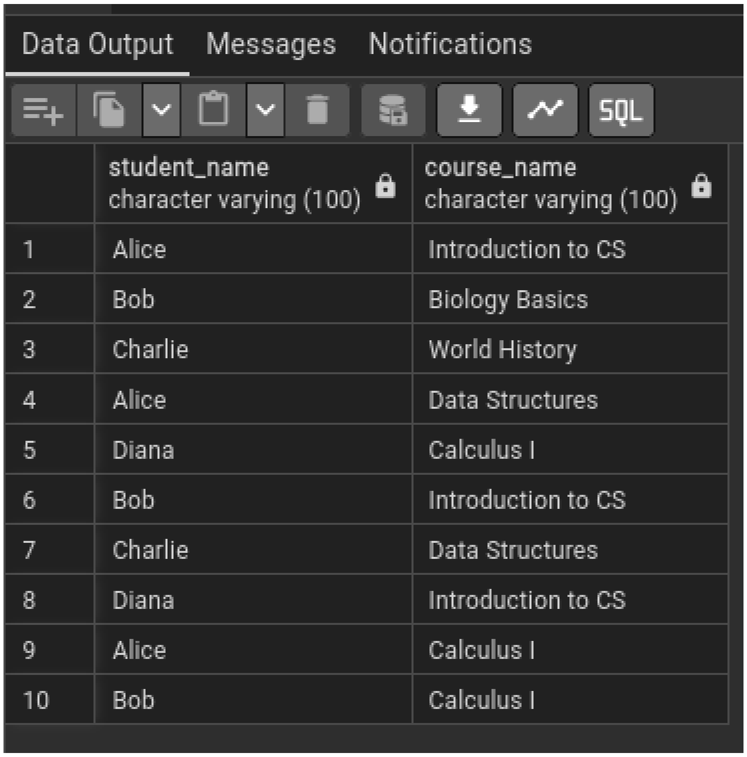


	student_name character varying (100)	course_name character varying (100)
1	Alice	Introduction to CS
2	Bob	Biology Basics
3	Charlie	World History
4	Alice	Data Structures
5	Diana	Calculus I
6	Bob	Introduction to CS
7	Charlie	Data Structures
8	Diana	Introduction to CS
9	Alice	Calculus I
10	Bob	Calculus I
11	Bubbly	[null]

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

```
SELECT
  s.student_name,
  c.course_name
FROM
  assignment3.students s
  RIGHT JOIN assignment3.enrollments e ON e.student_id = s.student_id
  RIGHT JOIN assignment3.courses c ON c.course_id = e.course_id;
```

Screenshot:

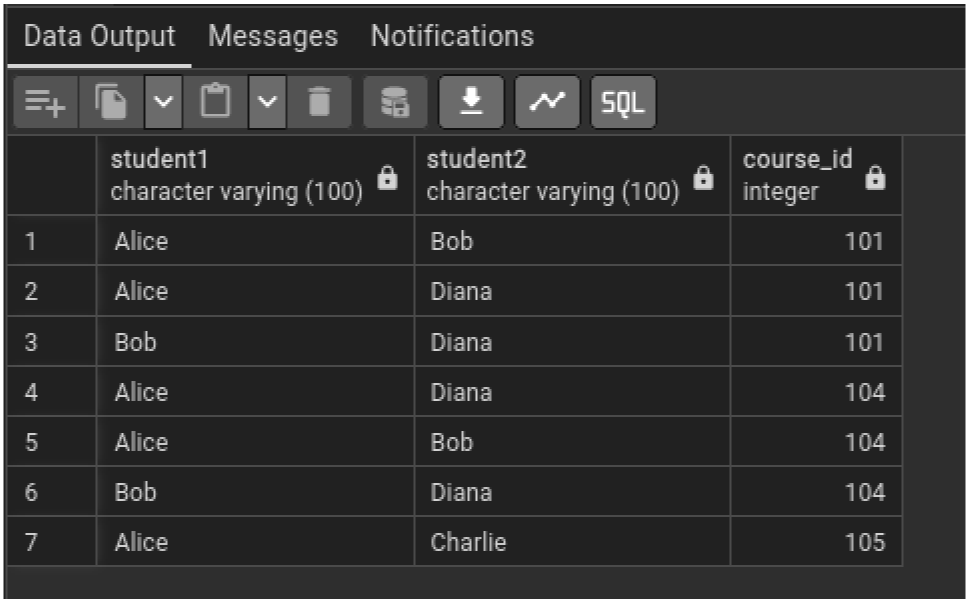


	student_name character varying (100)	course_name character varying (100)
1	Alice	Introduction to CS
2	Bob	Biology Basics
3	Charlie	World History
4	Alice	Data Structures
5	Diana	Calculus I
6	Bob	Introduction to CS
7	Charlie	Data Structures
8	Diana	Introduction to CS
9	Alice	Calculus I
10	Bob	Calculus I

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

```
SELECT
  s1.student_name AS student1,
  s2.student_name AS student2,
  e1.course_id
FROM
  assignment3.enrollments e1
  INNER JOIN assignment3.enrollments e2 ON e1.course_id =
e2.course_id
  AND e1.student_id < e2.student_id
  INNER JOIN assignment3.students s1 ON e1.student_id = s1.student_id
  INNER JOIN assignment3.students s2 ON e2.student_id =
s2.student_id;
```

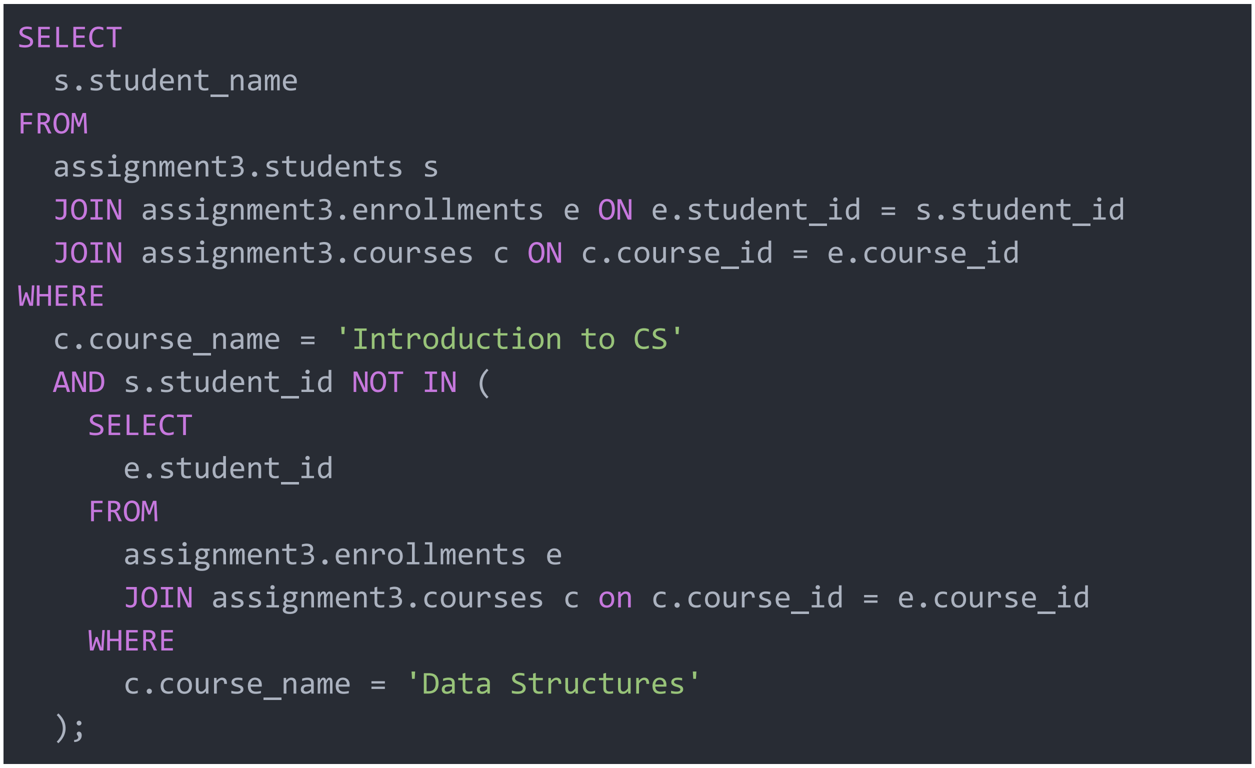
Screenshot:



	student1 character varying (100)	student2 character varying (100)	course_id integer
1	Alice	Bob	101
2	Alice	Diana	101
3	Bob	Diana	101
4	Alice	Diana	104
5	Alice	Bob	104
6	Bob	Diana	104
7	Alice	Charlie	105

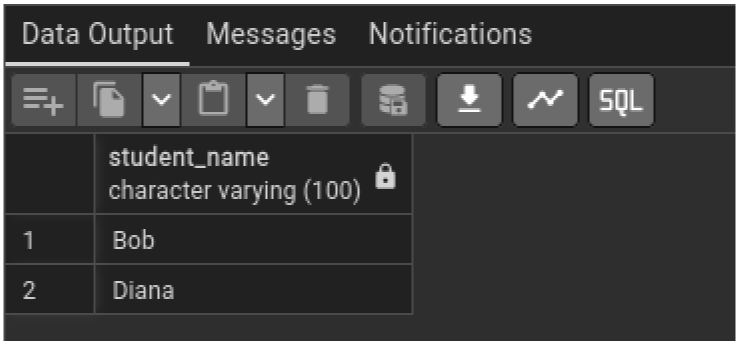
5. Complex Join:

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



```
SELECT
  s.student_name
FROM
  assignment3.students s
  JOIN assignment3.enrollments e ON e.student_id = s.student_id
  JOIN assignment3.courses c ON c.course_id = e.course_id
WHERE
  c.course_name = 'Introduction to CS'
  AND s.student_id NOT IN (
    SELECT
      e.student_id
    FROM
      assignment3.enrollments e
      JOIN assignment3.courses c on c.course_id = e.course_id
    WHERE
      c.course_name = 'Data Structures'
  );
```

Screenshot:



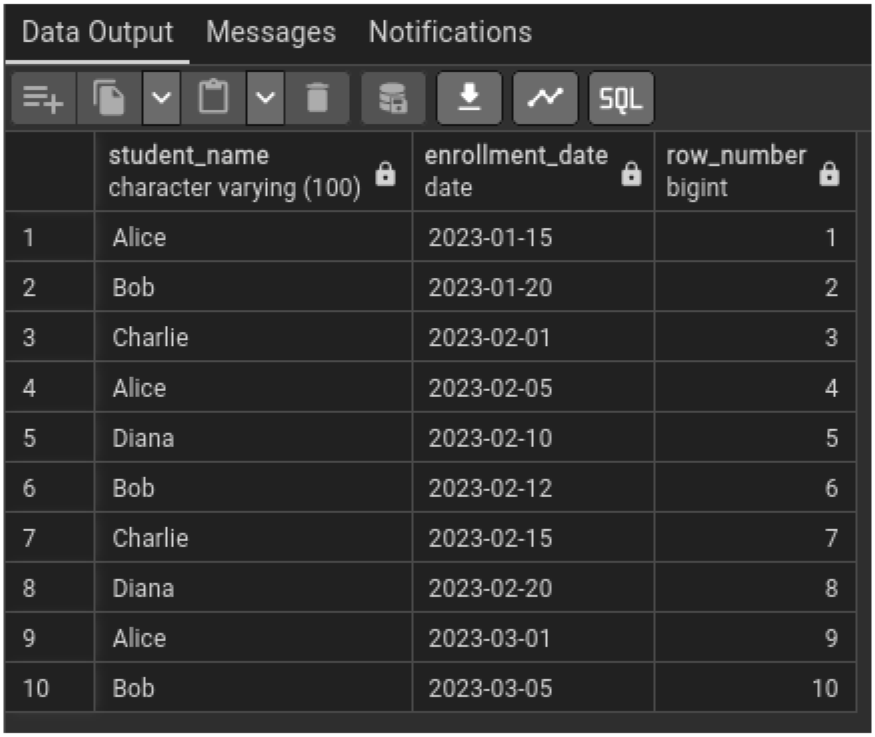
	student_name character varying (100)
1	Bob
2	Diana

Windows function:

1. Using ROW_NUMBER():
Question: List all students along with a row number based on their enrollment date in ascending order.

```
SELECT
  s.student_name,
  e.enrollment_date,
  ROW_NUMBER() OVER (
    ORDER BY
      e.enrollment_date ASC
  )
FROM
  assignment3.students s
  JOIN assignment3.enrollments e on s.student_id = e.student_id;
```

Screenshot:



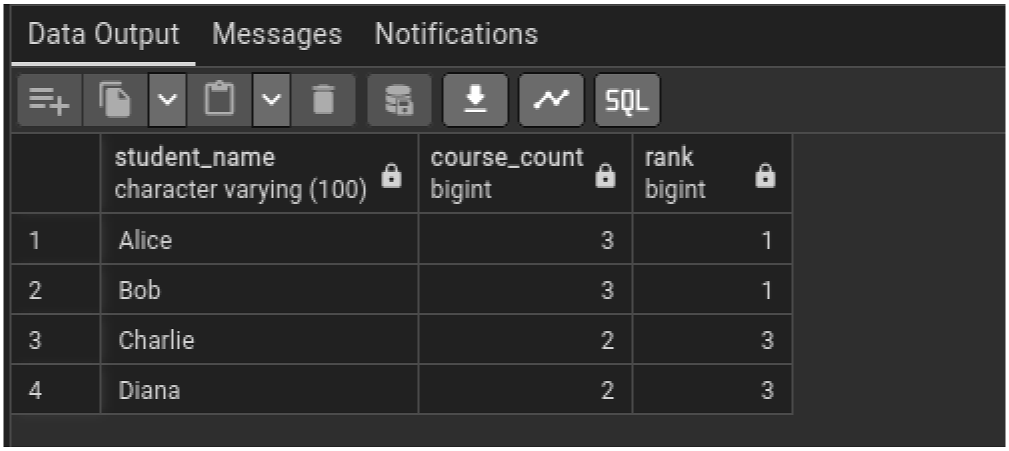
	student_name character varying (100)	enrollment_date date	row_number bigint
1	Alice	2023-01-15	1
2	Bob	2023-01-20	2
3	Charlie	2023-02-01	3
4	Alice	2023-02-05	4
5	Diana	2023-02-10	5
6	Bob	2023-02-12	6
7	Charlie	2023-02-15	7
8	Diana	2023-02-20	8
9	Alice	2023-03-01	9
10	Bob	2023-03-05	10

2. Using RANK():

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

```
SELECT
  student_name,
  course_count,
  RANK() OVER (
    ORDER by
      course_count DESC
  ) AS RANK
FROM
  (
    SELECT
      s.student_name,
      COUNT(e.course_id) AS course_count
    FROM
      assignment3.students s
      JOIN assignment3.enrollments e ON s.student_id = e.student_id
    GROUP BY
      s.student_name
  );
```

Screenshot:



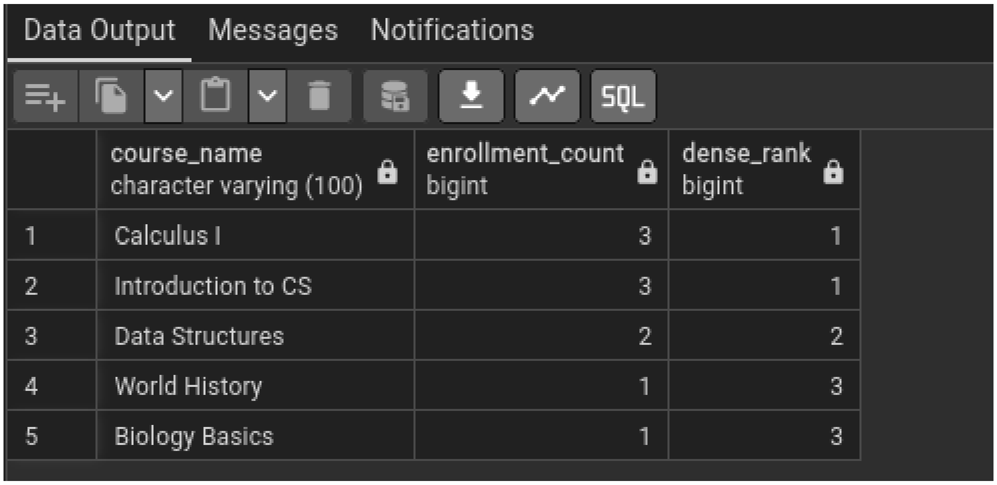
	student_name character varying (100)	course_count bigint	rank bigint
1	Alice	3	1
2	Bob	3	1
3	Charlie	2	3
4	Diana	2	3

3. Using DENSE_RANK():
Question: Determine the dense rank of courses based on their enrollment count across all students

```
SELECT
  course_name,
  enrollment_count,
  DENSE_RANK() OVER (
    ORDER by
      enrollment_count DESC
  ) AS dense_rank
FROM
  (
    SELECT
      c.course_name,
      COUNT(e.student_id) AS enrollment_count
    FROM
      assignment3.courses c
      LEFT JOIN assignment3.enrollments e ON c.course_id =
e.course_id
```

```
  GROUP BY
    c.course_name
);
```

Screenshot:



	course_name character varying (100)	enrollment_count bigint	dense_rank bigint
1	Calculus I	3	1
2	Introduction to CS	3	1
3	Data Structures	2	2
4	World History	1	3
5	Biology Basics	1	3