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.

```sql
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;
```

Data Output Messages Notifications

	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
    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;
```

Data Output Messages Notifications

	enrollment_id [PK] integer	student_id integer	course_id integer	enrollm date
1	1	1	101	2023-0
2	2	2	102	2023-0
3	3	3	103	2023-0
4	4	1	105	2023-0
5	5	4	104	2023-0
6	6	2	101	2023-0
7	7	3	105	2023-0
8	8	4	101	2023-0
9	9	1	104	2023-0
10	10	2	104	2023-0

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;
```

Data Output Messages Notifications

	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

NOTE: No course without enrolled students

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

```
SELECT distinct
    S1.student_name AS student1,
    T2.student_name AS student2
from
    students S1
    INNER JOIN enrollments E1 ON S1.student_id = E1.student_id
    INNER JOIN (
        SELECT
            *
        from
            students S2
            INNER JOIN enrollments E2 ON S2.student_id = E2.student_id
    ) AS T2 ON T2.course_id = E1.course_id
    AND T2.student_name < S1.student_name
order by
    student1
```

Data Output Messages Notifications

	student1 character varying (100)	student2 character varying (100)
1	Bob	Alice
2	Charlie	Alice
3	Diana	Alice
4	Diana	Bob

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'
    );
```

Data Output Messages Notifications

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

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
```

Data Output Messages Notifications

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

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;
```

Data Output Messages Notifications

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

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

```
SELECT
    DENSE_RANK() OVER (
        ORDER BY
            COUNT(student_id) DESC
    ) as rank,
    C.course_name,
    COUNT(student_id) as no_of_students
from
    Courses C
    INNER JOIN enrollments E ON C.course_id = E.course_id
GROUP BY
    C.course_id;
```

Data Output Messages Notifications

	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
5	3	World History	1