Consider a simple scenario in a university setting where we have two tables: students and enrollments.
Students Table:

student_id	student_name	major
1	Daniel	Information Technology
2	Emma	Physics
3	Grace	Chemistry

Enrollments Table:

enrollment_id	student_id	course_name
1	1	Database Systems
2	2	Calculus
3	4	Genetics

Use Cases and SQL Statements for Different Joins:

INNER JOIN:
Use Case: Retrieve a list of students and the courses they are enrolled in. This join will only show records where there is a match in both tables.
SQL Statement:

```
SELECT students.student_id, students.student_name, students.major,
enrollments.course_name
```

```
FROM students

INNER JOIN enrollments ON students.student_id =
enrollments.student_id;
```

Explanation: This query will return only those students who have an entry in the enrollments table, thus showing which courses they are enrolled in.

LEFT JOIN:
Use Case: Get a list of all students and the courses they are enrolled in, including those who are not enrolled in any courses. This join will show all students, even if they are not enrolled in any courses.
SQL Statement:

```
SELECT students.student_id, students.student_name, students.major,
enrollments.course_name

FROM students

LEFT JOIN enrollments ON students.student_id =
enrollments.student_id;
```

Explanation: This query will return all students, including those who are not found in the enrollments table, ensuring that students without courses are still listed.

RIGHT JOIN:
Use Case: Retrieve a list of all courses and the students enrolled in each course, including courses with no students. This join will show all courses, even if there are no students enrolled in them.
SQL Statement:

```
SELECT students.student_id, students.student_name, students.major,
enrollments.course_name

FROM students

RIGHT JOIN enrollments ON students.student_id =
enrollments.student_id;
```

Explanation: This query will return all courses, including those which have no matching student entries, ensuring that all courses are listed.

UNION:
Use Case: Combine the lists of students and enrollments into a single list. This join will show a combined list of students and courses.
SQL Statement:

```
SELECT student_id, student_name AS name, major AS info

FROM students

UNION

SELECT enrollment_id, course_name AS name, 'Enrollment' AS info

FROM enrollments;
```

Explanation: This query will combine the student and enrollment data into a single list, presenting a unified view of students and their enrollments.

CROSS JOIN:
Use Case: Generate all possible combinations of students and courses. This join will show every possible pairing of students and courses.
SQL Statement:

```
SELECT students.student_id, students.student_name, students.major,
enrollments.course_name

FROM students

CROSS JOIN enrollments;
```

Explanation: This query will return a Cartesian product of the students and enrollments tables, showing all possible combinations of students and courses, regardless of actual enrollments.

These SQL statements and use cases illustrate how different types of joins can be used to retrieve and combine data in various ways