**SQL Set Operations and Comparison Operators**

Here are examples of how to use various SQL set operations like UNION, UNION ALL, INTERSECT, and comparison operators like >, <, >ALL, <ANY, etc. These examples will be based on two simple tables of employees from different departments.

**Step 1: Create Sample Tables**

We will use two employee tables employees\_A and employees\_B for this demonstration.

sql

Copy code

CREATE TABLE employees\_A (

employee\_id SERIAL PRIMARY KEY,

name VARCHAR(100),

salary NUMERIC(10, 2)

);

CREATE TABLE employees\_B (

employee\_id SERIAL PRIMARY KEY,

name VARCHAR(100),

salary NUMERIC(10, 2)

);

**Step 2: Insert Sample Data**

**Insert data into employees\_A:**

sql

Copy code

INSERT INTO employees\_A (name, salary)

VALUES

('John Doe', 60000),

('Alice Johnson', 70000),

('Sara Lee', 80000);

**Insert data into employees\_B:**

sql

Copy code

INSERT INTO employees\_B (name, salary)

VALUES

('Bob Williams', 70000),

('Jane Smith', 60000),

('Sara Lee', 80000);

**1. UNION**

The UNION operator combines the result sets of two or more SELECT queries and removes duplicate records. The columns in both queries must have the same data types.

sql

Copy code

SELECT name, salary FROM employees\_A

UNION

SELECT name, salary FROM employees\_B;

**Output:**

| **name** | **salary** |
| --- | --- |
| John Doe | 60000.00 |
| Alice Johnson | 70000.00 |
| Sara Lee | 80000.00 |
| Bob Williams | 70000.00 |
| Jane Smith | 60000.00 |

**2. UNION ALL**

The UNION ALL operator is similar to UNION, but it does not remove duplicate records. It keeps all the rows from both queries.

sql

Copy code

SELECT name, salary FROM employees\_A

UNION ALL

SELECT name, salary FROM employees\_B;

**Output:**

| **name** | **salary** |
| --- | --- |
| John Doe | 60000.00 |
| Alice Johnson | 70000.00 |
| Sara Lee | 80000.00 |
| Bob Williams | 70000.00 |
| Jane Smith | 60000.00 |
| Sara Lee | 80000.00 |

**3. INTERSECT**

The INTERSECT operator returns only the rows that are common between the two SELECT queries (i.e., the intersection of the two result sets).

sql

Copy code

SELECT name, salary FROM employees\_A

INTERSECT

SELECT name, salary FROM employees\_B;

**Output:**

| **name** | **salary** |
| --- | --- |
| Sara Lee | 80000.00 |

**4. EXCEPT**

The EXCEPT operator returns the rows from the first SELECT query that are not present in the second SELECT query (i.e., rows in employees\_A but not in employees\_B).

sql

Copy code

SELECT name, salary FROM employees\_A

EXCEPT

SELECT name, salary FROM employees\_B;

**Output:**

| **name** | **salary** |
| --- | --- |
| John Doe | 60000.00 |
| Alice Johnson | 70000.00 |

**5. Comparison Operators: >, <, > ALL, < ANY, > ANY**

**Example 1: >**

Find employees with a salary greater than 60000 in employees\_A.

sql

Copy code

SELECT name, salary

FROM employees\_A

WHERE salary > 60000;

**Output:**

| **name** | **salary** |
| --- | --- |
| Alice Johnson | 70000.00 |
| Sara Lee | 80000.00 |

**Example 2: <**

Find employees with a salary less than 70000 in employees\_B.

sql

Copy code

SELECT name, salary

FROM employees\_B

WHERE salary < 70000;

**Output:**

| **name** | **salary** |
| --- | --- |
| Jane Smith | 60000.00 |

**Example 3: > ALL**

Find employees in employees\_A with a salary greater than **all** the employees in employees\_B.

sql

Copy code

SELECT name, salary

FROM employees\_A

WHERE salary > ALL (SELECT salary FROM employees\_B);

**Output:**

| **name** | **salary** |
| --- | --- |
| Sara Lee | 80000.00 |

**Example 4: < ANY**

Find employees in employees\_B with a salary less than **any** of the salaries in employees\_A.

sql

Copy code

SELECT name, salary

FROM employees\_B

WHERE salary < ANY (SELECT salary FROM employees\_A);

**Output:**

| **name** | **salary** |
| --- | --- |
| Jane Smith | 60000.00 |

**Example 5: > ANY**

Find employees in employees\_A with a salary greater than **any** of the employees in employees\_B.

sql

Copy code

SELECT name, salary

FROM employees\_A

WHERE salary > ANY (SELECT salary FROM employees\_B);

**Output:**

| **name** | **salary** |
| --- | --- |
| Alice Johnson | 70000.00 |
| Sara Lee | 80000.00 |

**Conclusion**

* **Set operations** (UNION, UNION ALL, INTERSECT, EXCEPT) allow you to combine and compare result sets.
* **Comparison operators** (>, <, > ALL, < ANY, > ANY) help you compare values across rows in the same or different tables.