

## Title: Using Set Operations

**Author: S. Santhosh Kumar**

---

### Objectives

After completing this exercise, students should be able to:

- Describe set operators.
- Use a set operator to combine multiple queries into a single query.
- Control the order of rows returned.

The set operators combine the results of two or more component queries into one result. Queries containing set operators are called compound queries.

### Tables Used in This Lesson

- **EMPLOYEES:** Provides details regarding all current employees.
  - **JOB\_HISTORY:** Records the details of the start date and end date of the former job, job identification number, and department when an employee switches jobs.
- 

## UNION Operator

### Guidelines

- The number of columns and the data types of the columns being selected must be identical in all the `SELECT` statements used in the query. The names of the columns need not be identical.
- `UNION` operates over all of the columns being selected.
- `NULL` values are not ignored during duplicate checking.
- The `IN` operator has a higher precedence than the `UNION` operator.

**Example 1: Display the current and previous job details of all employees. Display each employee only once.**

### Query:

```
SELECT employee_id, job_id
FROM employees
UNION
SELECT employee_id, job_id
FROM job_history;
```

**Sample Input:**

- **EMPLOYEES Table:**

employee_id	job_id
100	SA_REP
101	SA_REP
102	IT_PROG
103	IT_PROG

- **JOB\_HISTORY Table:**

employee_id	job_id
100	SA_REP
104	HR_REP
105	ST_CLERK

**Sample Output:**

employee_id	job_id
100	SA_REP
101	SA_REP
102	IT_PROG
103	IT_PROG
104	HR_REP
105	ST_CLERK

## UNION ALL Operator

### Guidelines

- The guidelines for `UNION` and `UNION ALL` are the same, with the following two exceptions:
  - Unlike `UNION`, duplicate rows are not eliminated, and the output is not sorted by default.
  - The `DISTINCT` keyword cannot be used.

### Example 2: Display the current and previous departments of all employees.

#### Query:

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

#### Sample Input:

- **EMPLOYEES Table:**

employee_id	job_id	department_id
100	SA_REP	80
101	SA_REP	80
102	IT_PROG	60

- **JOB\_HISTORY Table:**

employee_id	job_id	department_id
100	SA_REP	80
103	IT_PROG	60

**Sample Output:**

employee_id	job_id	department_id
100	SA_REP	80
100	SA_REP	80
101	SA_REP	80
102	IT_PROG	60
103	IT_PROG	60

**INTERSECT Operator****Guidelines**

- The number of columns and the data types of the columns being selected by the `SELECT` statements in the queries must be identical in all the `SELECT` statements used in the query. The names of the columns need not be identical.
- Reversing the order of the intersected tables does not alter the result.
- `INTERSECT` does not ignore `NULL` values.

**Example 3: Display the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired.**

**Query:**

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

**Sample Input:**

- **EMPLOYEES Table:**

employee_id	job_id
100	SA_REP
101	SA_REP
102	IT_PROG

- **JOB\_HISTORY Table:**

employee_id	job_id
100	SA_REP
103	HR_REP

### Sample Output:

employee_id	job_id
100	SA_REP

## MINUS Operator

### Guidelines

- The number of columns and the data types of the columns being selected by the `SELECT` statements in the queries must be identical in all the `SELECT` statements used in the query. The names of the columns need not be identical.
- All of the columns in the `WHERE` clause must be in the `SELECT` clause for the `MINUS` operator to work.

**Example 4: Display the employee IDs of those employees who have not changed their jobs even once.**

### Query:

```
SELECT employee_id, job_id
FROM employees
MINUS
SELECT employee_id, job_id
FROM job_history;
```

### Sample Input:

- **EMPLOYEES Table:**

employee_id	job_id
100	SA_REP
101	SA_REP
102	IT_PROG

- **JOB\_HISTORY Table:**

employee_id	job_id
100	SA_REP
103	HR_REP

**Sample Output:**

employee_id	job_id
101	SA_REP
102	IT_PROG

---

## Exercises

1. **HR Department Report:** List department IDs for departments that do not contain the job ID ST\_CLERK. Use set operators to create this report.

### Solution:

```
SELECT department_id
FROM departments
MINUS
SELECT department_id
FROM employees
WHERE job_id = 'ST_CLERK';
```

### Sample Input:

- **DEPARTMENTS Table:**

department_id	department_name
10	Admin
20	HR
50	Sales

- **EMPLOYEES Table:**

employee_id	job_id	department_id
100	SA_REP	50
101	ST_CLERK	20

### Sample Output:

department_id
10
50

2. **Countries Without Departments:** List countries that have no departments located in them. Display the country ID and the name of the countries. Use set operators to create this report.

**Solution:**

```
SELECT country_id, country_name
FROM countries
MINUS
SELECT country_id, country_name
FROM countries c, departments d
WHERE c.country_id = d.country_id;
```

**Sample Input:**

- **COUNTRIES Table:**

country_id	country_name
US	United States
UK	United Kingdom
IN	India

- **DEPARTMENTS Table:**

department_id	department_name	country_id
10	Admin	US
20	HR	IN

**Sample Output:**

country_id	country_name
UK	United Kingdom

3. **Jobs for Specific Departments:** Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using set operators.

**Solution:**

```
SELECT job_id, department_id
FROM employees
WHERE department_id IN (10, 50, 20)
ORDER BY department_id;
```



**Sample Input:**

- **EMPLOYEES Table:**

employee_id	job_id	department_id
100	SA_REP	50
101	ST_CLERK	20
102	IT_PROG	10

**Sample Output:**

job_id	department_id
IT_PROG	10
ST_CLERK	20
SA_REP	50

4. **Employees with Original Jobs:** Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company.

**Solution:**

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

**Sample Input:**

- **EMPLOYEES Table:**

employee_id	job_id
100	SA_REP
101	SA_REP
102	IT_PROG

- **JOB\_HISTORY Table:**

employee_id	job_id
100	SA_REP
103	HR_REP

**Sample Output:**

employee_id	job_id
100	SA_REP

5. **Comprehensive HR Report:** The HR department needs a report with the following specifications:
- Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department.
  - Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them.

**Solution:**

```
SELECT last_name, department_id
FROM employees
UNION
SELECT department_id, department_name
FROM departments;
```

**Sample Output:**

last_name	department_id
King	90
Kochhar	80
De Haan	60
Executive	90
Sales	80
IT	50