**Business Case Scenario**

**Scenario**: A company wants to manage its employees and the assets assigned to them. Each employee can have multiple assets, but not all employees may have assets assigned. Additionally, some assets may not be assigned to any employee.

**DDL for employees Table**

sql

Copy code

CREATE TABLE employees (

employee\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

department VARCHAR(50) NOT NULL,

position VARCHAR(50) NOT NULL

);

-- Insert sample data into employees

INSERT INTO employees (name, department, position) VALUES

('Alice Johnson', 'HR', 'Manager'),

('Bob Smith', 'Finance', 'Analyst'),

('Charlie Brown', 'Engineering', 'Developer'),

('David Wilson', 'Engineering', 'Senior Developer'),

('Eve Davis', 'Marketing', 'Executive');

**DDL for assets Table**

sql

Copy code

CREATE TABLE assets (

asset\_id SERIAL PRIMARY KEY,

asset\_name VARCHAR(100) NOT NULL,

assigned\_to INT,

FOREIGN KEY (assigned\_to) REFERENCES employees(employee\_id)

);

-- Insert sample data into assets

INSERT INTO assets (asset\_name, assigned\_to) VALUES

('Laptop', 1), -- Assigned to Alice

('Monitor', 2), -- Assigned to Bob

('Phone', NULL), -- Not assigned

('Tablet', 4), -- Assigned to David

('Keyboard', 1); -- Assigned to Alice

**1. INNER JOIN**

**Query**: Retrieve all employees and their assigned assets.

sql

Copy code

SELECT

e.name AS employee\_name,

a.asset\_name

FROM

employees e

INNER JOIN

assets a ON e.employee\_id = a.assigned\_to;

**Explanation**: This query retrieves only those employees who have been assigned assets. If an employee has no assets assigned, they will not appear in the result.

**2. LEFT JOIN**

**Query**: Retrieve all employees and their assigned assets, including those who have no assets.

sql

Copy code

SELECT

e.name AS employee\_name,

a.asset\_name

FROM

employees e

LEFT JOIN

assets a ON e.employee\_id = a.assigned\_to;

**Explanation**: This query retrieves all employees regardless of whether they have assets assigned. Employees without assets will show NULL for the asset\_name.

**3. RIGHT JOIN**

**Query**: Retrieve all assets and their assigned employees, including assets that are not assigned to anyone.

sql

Copy code

SELECT

e.name AS employee\_name,

a.asset\_name

FROM

employees e

RIGHT JOIN

assets a ON e.employee\_id = a.assigned\_to;

**Explanation**: This query retrieves all assets, including those that are not assigned to any employee. For assets not assigned, the employee\_name will be NULL.

**4. SELF JOIN**

**Query**: Retrieve employees in the same department.

sql

Copy code

SELECT

e1.name AS employee\_1,

e2.name AS employee\_2,

e1.department

FROM

employees e1

JOIN

employees e2 ON e1.department = e2.department AND e1.employee\_id <> e2.employee\_id;

**Explanation**: This query finds pairs of employees working in the same department. It uses a self join to join the employees table with itself, ensuring that it does not pair an employee with themselves.

**Summary of Joins**

* **INNER JOIN**: Returns records that have matching values in both tables. In our case, only employees with assigned assets are returned.
* **LEFT JOIN**: Returns all records from the left table (employees), and the matched records from the right table (assets). If there’s no match, NULLs are returned for the right table's columns.
* **RIGHT JOIN**: Returns all records from the right table (assets) and the matched records from the left table (employees). If there’s no match, NULLs are returned for the left table's columns.
* **SELF JOIN**: Joins the table to itself to find related data within the same table. Here, it helps in finding employees working in the same department.

This DDL and the accompanying queries illustrate how to use different types of joins effectively in a business context involving employees and assets.