Two tables: `employees` and `departments`. We'll establish a relationship between them using primary and foreign key constraints. Then, we'll explore various SQL join types with explanations for each.

1. Creating Tables with Primary and Foreign Key Constraints:

-- Create the 'departments' table

```
CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50) NOT NULL
);
```

-- Create the 'employees' table with a foreign key referencing 'departments'

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100) NOT NULL,
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

In this setup:

- The `departments` table has a primary key `department_id`.
- The `employees` table includes a `department_id` column that acts as a foreign key, linking each employee to a department.

2. Inserting Sample Data:

-- Insert data into 'departments'

```
INSERT INTO departments (department_id, department_name) VALUES
(1, 'Sales'),
(2, 'Engineering'),
(3, 'Human Resources');
```

-- Insert data into 'employees'

```
INSERT INTO employees (employee_id, employee_name, department_id)
VALUES
(101, 'Alice', 1),
(102, 'Bob', 2),
(103, 'Charlie', 2),
(104, 'Diana', 3),
(105, 'Eve', NULL); -- Eve is not assigned to any department
```

SQL Join Operations

1. INNER JOIN:

```
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees e
INNER JOIN departments d ON e.department_id = d.department_id;
```

Explanation: An `INNER JOIN` returns rows when there is a match in both tables. This query retrieves employees who are assigned to a department.

Output:

employee_id	employee_name	department_name
101	Alice	Sales
102	Bob	Engineering
103	Charlie	Engineering
104	Diana	Human Resources

2. LEFT JOIN (or LEFT OUTER JOIN):

```
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

Explanation: A `LEFT JOIN` returns all rows from the left table (`employees`), along with matching rows from the right table (`departments`). If there's no match, NULL values are returned for columns from the right table. This query includes all employees, even those not assigned to a department.

Output:

employee_id	employee_name	department_name
101	Alice	Sales
102	Bob	Engineering
103	Charlie	Engineering
104	Diana	Human Resources
105	Eve	NULL

3. RIGHT JOIN (or RIGHT OUTER JOIN):

```
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees e
RIGHT JOIN departments d ON e.department_id = d.department_id;
```

Explanation: A `RIGHT JOIN` returns all rows from the right table (`departments`), along with matching rows from the left table (`employees`). If there's no match, NULL values are returned for columns from the left table. This query includes all departments, even those without assigned employees.

Output:

employee_id	employee_name	department_name
101	Alice	Sales
102	Bob	Engineering
103	Charlie	Engineering
104	Diana	Human Resources
NULL	NULL	Marketing

4. FULL JOIN (or FULL OUTER JOIN):

```
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees e
FULL JOIN departments d ON e.department_id = d.department_id;
```

Explanation: A `FULL JOIN` returns rows when there is a match in either the left or right table. It combines the results of both `LEFT` and `RIGHT` joins. This query includes all employees and all departments, with NULLs where there's no match.

Output:

employee_id	employee_name	department_name
101	Alice	Sales
102	Bob	Engineering
103	Charlie	Engineering
104	Diana	Human Resources
105	Eve	NULL
NULL	NULL	Marketing

5. CROSS JOIN:

```
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees e
CROSS JOIN departments d;
```

Explanation: A `CROSS JOIN` returns the Cartesian product of both tables, meaning it combines each row from the first table with all rows from the second table. Use this cautiously, as it can produce a large number of results. This query lists all possible combinations of employees and departments.

Output:

employee_id	employee_name	department_name
101	Alice	Sales
101	Alice	Engineering
101	Alice	Human Resources
101	Alice	Marketing
102	Bob	Sales
102	Bob	Engineering
102	Bob	Human Resources
102	Bob	Marketing
103	Charlie	Sales
103	Charlie	Engineering
103	Charlie	Human Resources
103	Charlie	Marketing
104	Diana	Sales
104	Diana	Engineering
104	Diana	Human Resources
104	Diana	Marketing

105	Eve	Sales
105	Eve	Engineering
105	Eve	Human Resources
105	Eve	Marketing

6. SELF JOIN:

```
SELECT e1.employee_id AS Employee1_ID, e1.employee_name AS
Employee1_Name,
           e2.employee_id AS Employee2_ID, e2.employee_name AS
Employee2_Name
FROM employees e1, employees e2
WHERE e1.department_id = e2.department_id
  AND e1.employee_id < e2.employee_id;
```

Explanation: A self join is a regular join but the table is joined with itself. It's useful for querying hierarchical data or comparing rows within the same table. This query finds pairs of employees within the same department.

Output:

Employee1_ID	Employee1_Name	Employee2_ID	Employee2_Name
101	Alice	102	Bob
101	Alice	103	Charlie
102	Bob	103	Charlie

7. NATURAL JOIN:

```
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees e
NATURAL JOIN departments d;
```

Explanation: A `NATURAL JOIN` automatically joins tables based on columns with the same name and compatible data types. In this case, it joins on `department_id`. It's convenient but should be used carefully to avoid unexpected results if tables have multiple columns with the same name.

Output:

employee_id	employee_name	department_name
101	Alice	Sales
102	Bob	Engineering
103	Charlie	Engineering
104	Diana	Human Resources

8. USING Clause with JOIN:

```
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees e
JOIN departments d USING (department_id);
```

Explanation: The `USING` clause specifies the column(s) to join on when they have the same name in both tables. It's a shorthand for specifying the join condition.

Output:

employee_id	employee_name	department_name
101	Alice	Sales
102	Bob	Engineering