1. INNER JOIN

When to use: When you need to retrieve records that have matching values in both tables.

Example: Finding customers who have placed orders.

```
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.order_date
FROM
    customers c
INNER JOIN
    orders o ON c.customer_id = o.customer_id;
```

2. LEFT JOIN

When to use: When you need all records from the left table and the matched records from the right table. Unmatched records from the right table will be NULL.

Example: Find all customers, including those who haven't placed any orders.

```sql
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.order_date
FROM
    customers c
LEFT JOIN
    orders o ON c.customer_id = o.customer_id;
```

3. RIGHT JOIN

When to use: When you need all records from the right table and the matched records from the left table. Unmatched records from the left table will be NULL.

Example: Listing all orders and the customers who placed them, if any.

```
SELECT
    o.order_id,
    o.order_date,
    c.customer_id,
    c.customer_name
FROM
    customers c
RIGHT JOIN
    orders o ON c.customer_id = o.customer_id;
```

4. FULL JOIN

When to use: When you need all records when there is a match in either the left or right table. Records with no match will have NULLs

Example: Finding all customers and all orders, including those without matches.

```sql
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.order_date
FROM
    customers c
FULL OUTER JOIN
    orders o ON c.customer_id = o.customer_id;
```

5. CROSS JOIN

When to use: When you need to return the Cartesian product of the two tables. Be cautious as this can result in a large number of rows.

Example: Pairing each customer with every product.

```
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.order_date
FROM
    customers c
CROSS JOIN
    orders o;
```

6. SELF JOIN

When to use: When you need to join a table with itself to compare rows within the same table.

Example: Finding pairs of employees who have the same manager.

```sql
SELECT
    e1.employee_name AS employee_1,
    e2.employee_name AS employee_2,
    e1.manager_id
FROM
    employees e1
JOIN
    employees e2 ON e1.manager_id = e2.manager_id
WHERE
    e1.employee_id <> e2.employee_id;
```

Choosing the right join depends on your specific use case. Understanding these joins and when to use which JOIN will help you write more efficient and effective SQL queries.

This is also one of the most asked interview questions!