SQL: LIKE

What if we do not know what we're looking for but if have some partial information like the first letter of a name ?

Well this is where the `LIKE` keyword comes into play:

```
SELECT column1, column2, ... FROM table_name WHERE column1
LIKE pattern;
```

The `pattern` can take several forms:

Use Cases	Meaning
LIKE '%2'	Fields that end with 2
LIKE '%2%'	Fields that have 2 anywhere in the value
LIKE '_00%'	Fields that have 2 zero's as the second and third character and anything after
LIKE '%200%'	Fields that have 200 anywhere in the value
LIKE '2_%_%'	Finds any values that start with 2 and are at least 3 characters in length
LIKE '2___3'	Finds any values in a five-digit number that start with 2 and end with 3

💡 Note: Postgres `LIKE` only does text comparison so we must cast whatever we use to text.

The note is important and specific to Postgres ! So we need to cast everything into the string format (text format)

```
CAST(column1 as TEXT); -- or
column1 :: TEXT;
```

Instead of the LIKE we can use the ILIKE which is the same but case insensitive.

Examples of Using `LIKE`

Let's look at some examples to understand how the `LIKE` operator works.

Example Table: Employees

employee_id	first_name	last_name	department
1	John	Doe	Sales
2	Jane	Smith	Marketing
3	Alice	Johnson	Sales
4	Bob	Brown	IT
5	Carol	White	HR

1. **Finding Values that Start with a Specific Letter**:

   Suppose you want to find all employees whose first name starts with 'J':

   ```
   SELECT employee_id, first_name, last_name
   FROM Employees
   WHERE first_name LIKE 'J%';
   ```

   Result:

employee_id	first_name	last_name
1	John	Doe
2	Jane	Smith

   In this query, the `%` wildcard after 'J' allows for any number of characters following 'J'.

2. **Finding Values that End with a Specific Letter**:

   Suppose you want to find all employees whose last name ends with 'n':

   ```
   SELECT employee_id, first_name, last_name
   FROM Employees
   WHERE last_name LIKE '%n';
   ```

   Result:

employee_id	first_name	last_name

3	Alice	Johnson

Here, the `%` wildcard before 'n' allows for any number of characters preceding 'n'.

3. **Finding Values that Contain a Specific Substring**:

   Suppose you want to find all employees whose department contains the substring 'ar':

   ```
   SELECT employee_id, first_name, last_name, department
   FROM Employees
   WHERE department LIKE '%ar%';
   ```

   Result:

employee_id	first_name	last_name	department
2	Jane	Smith	Marketing
5	Carol	White	HR

   In this example, `%ar%` means that 'ar' can appear anywhere within the department name.

4. **Finding Values with a Specific Character at a Certain Position**:

   Suppose you want to find all employees whose first name has an 'o' as the second letter:

   ```
   SELECT employee_id, first_name, last_name
   FROM Employees
   WHERE first_name LIKE '_o%';
   ```

   Result:

employee_id	first_name	last_name
1	John	Doe
4	Bob	Brown

   Here, `_o%` means the first character can be anything (`_`), 'o' must be the second character, and there can be any number of characters after that (`%`).

Key Points

- **Case Sensitivity**: The `LIKE` operator is often case-insensitive by default in many SQL databases (e.g., MySQL), but it can be case-sensitive in others (e.g., PostgreSQL). Check your database's documentation for specifics.
- **Wildcards**: Use `%` to represent any sequence of characters (including none) and `_` to represent a single character.
- **Pattern Flexibility**: The `LIKE` operator provides flexible pattern matching, allowing you to search for values that meet various criteria, such as starting with, ending with, or containing specific characters.

The `LIKE` operator is powerful for searching text and string data in SQL, providing the ability to perform partial and flexible matches based on patterns.