fingerTips

Module-4

Conditionals, Operators, Clauses &

Regular Expressions (RegEx) in SQL

1) Conditionals in SQL (If statement, case statement, where clause)

In SQL, conditional statements are used to specify certain conditions that need to be met in order to execute a particular set of commands or queries. The most commonly used conditional statements in SQL are the IF, CASE, and WHERE clauses.

2) If Statement

The IF statement is used to execute a set of SQL commands based on a specific condition.

Syntax:

```
IF(condition, value_if_true, value_if_false)
```

Example:

```
SELECT *, IF(price>50, "MORE", "LESS")
FROM products;
```

3) Case & when statement

The CASE statement is used to evaluate a set of conditions and return a result based on the first condition that is met.

Syntax:

```
CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  ...
  ELSE default_result
END;
```

Example:

Q. Make a new column attire_type and add values 'cheap' for price<30, 'moderate' for price between 30 and 60 and 'expensive' for price>60

```
SELECT *,
CASE
   WHEN price <30 THEN 'cheap'
   WHEN price between 30 and 60 THEN 'moderate'
   ELSE 'expensive'
END AS attire_type
FROM products;
```

4) Filter (where)

The WHERE clause is used in SQL to filter the rows returned by a query. It is used to specify a condition that must be met for a row to be included in the result set. The condition can be any expression that evaluates to a Boolean value (TRUE or FALSE).

The basic syntax for the WHERE clause is:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example:

Q. find details of 'Michael Chen' from customers table.

```
select * from customers
where name='Michael Chen';
```

5) Comparison Operators (<, >, =, !=, <=, >=)

SQL provides several comparison operators that allows us to compare values in a query. Some of the most commonly used comparison operators are:

- < (less than)
- (greater than)
- = (equal to)
- != or <> (not equal to)
- <= (less than or equal to)
- = (greater than or equal to)

These operators can be used in the WHERE clause of a query to specify a condition that must be true for a row to be included in the result set.

Examples:

Q. Find names of products where price is greater than or equal to 60

select name from products
where price>=60;

Q. Find product details for size 'L'

select *
from products
where size = 'L';

6) Arithmetic Operators (Avg, count, min, max, sum)

SQL provides several arithmetic functions that allows us to perform calculations on the values in a column. Some of the most commonly used arithmetic functions are:

- AVG(): Calculates the average value of a column
- COUNT(): Counts the number of rows or non-null values in a column
- MIN(): Finds the minimum value in a column
- MAX(): Finds the maximum value in a column
- SUM(): Calculates the sum of the values in a column

Examples:

Q. How many products of black color are available

select color, count(*)
from products
where color='Black';

Q. Number of quantity ordered by customer with id 1

select customer_id, sum(quantity)
from orders
where customer_id=1;

Q. What is minimum price of product available

select min(price)
from products;

Q. What is the average price od products ordered

```
select avg(price)
from products;
```

7) Logical Operators (OR, AND)

SQL provides two logical operators that allow you to combine conditions in a query. These operators are:

- OR: Returns true if any of the conditions separated by OR is true
- AND: Returns true only if all of the conditions separated by AND are true

These operators can be used in the WHERE clause of a query to combine multiple conditions.

Examples:

Q. Details of Jeans or Pants

```
select * from products
where name='Jeans' or  name='Pants';
```

Q. Is Yellow color Skirt available in products, if so what's the price?

```
select * from products
where name='Skirt' and color='Yellow';
```

8) Special Operators (Between, Like, Is null, In, Not, In, Distinct)

SQL provides several special operators that allow you to perform more complex queries. Some of the most commonly used special operators include:

- BETWEEN: Returns true if a value is between two other values
- LIKE: Performs pattern matching on a string
- IS NULL: Returns true if a value is null
- IN: Returns true if a value is in a list of values
- Not IN: Returns true if a value is not in a list of values
- DISTINCT: Removes duplicates from the list of values

Examples:

Q. Details of all the products available in price range from 45 to 60
Select * from products
where price between 45 and 60;

Q. Details of warm clothes available
select * from products
where description like '%warm%';

Q. Details of all thsoe customers who haven't provided their phone numbers
select * from customers
where phone is null;

Q. Details of all the products available in size 'M', 'L' or 'XL'
select * from products
where size In ('M', 'L', 'XL');

Q. Deatils of all the products except Black color
select * from products
where color Not In ('Black', 'Brown');

Q. What all size products are available?
select distinct(size) from products;

9) Group by clause, having clause and Order by clause

The **GROUP BY** clause is used in SQL to group rows that have the same values into summary rows, like "find the total number of sales by product type."

Example:

Q. What is the count and average price of all size products available
select size, count(*), avg(price) from products
group by size;

The **HAVING** clause is used to filter groups based on a specified condition, similar to the WHERE clause, but it operates on groups rather than individual rows.

Example:

Q. What is the count and average price of all size products having average price > 60

```
select size, count(*), avg(price) from products
group by size
having avg(price)>60;
```

The **ORDER BY** clause is used to sort the result set by one or more columns, either in ascending or descending order.

Examples:

Q. What is the count and average price of all size products and arrange in ascending order of price

```
select size, count(*), avg(price) from products
group by size
order by avg(price);
```

Q. What is the count and average price of all size products and arrange in descending order of price

```
select size, count(*), avg(price) from products
group by size
order by avg(price) desc;
```

10) Aliases, limit and offset

Aliases are used in SQL to give a table or column a temporary name, which can be useful for making the query more readable or for handling self-joins.

Example:

Q. What is the count of all size products, use total as name of resulting column

```
select size, count(*) as total
from products
group by size
order by count(*);
```

The **LIMIT** clause is used to restrict the number of rows returned by a SQL query, which can be useful for paging through large result sets.

Example:

Q. Details of highest price product available
select * from products
order by price desc
limit 1;

The **OFFSET** clause is used in combination with LIMIT to specify how many rows to skip before returning the result set.

Example:

Q. Details of 2nd to 5th highest price product details
select * from products
order by price desc
limit 1,3;
(means, skip 1 row & return 3 rows.)

OR

select * from products
order by price desc
limit 3 offset 1;
(means, skip 1 row & return 3 rows.)

11) RegEx (Regular Expressions)

Regular expressions, or regex, are a powerful tool for searching and manipulating text. They allow you to define a pattern of characters and then search for that pattern within a larger string. For example, we can use a regex to find all email addresses in a text file, or to extract specific data from a file.

Pattern	**What the Pattern matches**
*	Zero or more instances of string preceding

Pattern	**What the Pattern matches**
	it
+	One or more instances of strings preceding it
.	Any single character
?	Match zero or one instances of the strings preceding it.
^	caret(^) matches Beginning of string
$	End of string
[abc]	Any character listed between the square brackets
[^abc]	Any character not listed between the square brackets
[A-Z]	match any upper case letter.
[a-z]	match any lower case letter
[0-9]	match any digit from 0 through to 9.
[[:<:]]	matches the beginning of words.
[[:>:]]	matches the end of words.

Pattern	What the Pattern matches
[:class:]	matches a character class i.e. [:alpha:] to match letters, [:space:] to match white space, [:punct:] is match punctuations and [:upper:] for upper class letters.
p1|p2|p3	Alternation; matches any of the patterns p1, p2, or p3
{n}	n instances of preceding element
{m,n}	m through n instances of preceding element

Examples:

Q. Give names od customers whose name starts with 'J'.
SELECT name FROM customers WHERE name regexp '^J';

Q. Give names of customers having email id with extension '@gmail.com'.
SELECT email FROM customers WHERE email regexp '@gmail.com$';

Q. Give all the names containing 'Jo' or 'ee' or 'lli'.
SELECT name FROM customers WHERE name regexp 'Jo|ee|lli';

Q. Give names of colors containing vowels [aeiou]
SELECT color FROM products WHERE color regexp '[aeiou]';