Practical no. 4

Sushmit Partakke
23070521156
Sem – 4
Sec – B

Aim: Write and execute SQL queries- Operators (and, or, not, like, between, in)

SQL Logical Operators are essential tools used to test the truth of conditions in SQL queries. They return boolean values such as TRUE, FALSE, or NULL, making them invaluable for filtering, retrieving, or manipulating data. These operators allow developers to build complex queries by combining, negating, or comparing conditions effectively.

SQL Between Operator

The **SQL BETWEEN** operator is used to test whether a value falls within a given range of values (inclusive). The values can be **text**, **date**, or **numbers**. It can be used in a SELECT, INSERT, UPDATE or DELETE statement. The **SQL BETWEEN Condition** will return the records where the expression is within the range of **value1** and **value2.**

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Key Features:

- Inclusive of both boundary values (`value1` and `value2`).
- Simplifies queries when working with continuous ranges.

Emp

EmpID	Name	Country	Age	Salary
1	Shubham	India	23	30000
2	Aman	Australia	21	45000
3	Naveen	Sri lanka	24	40000
4	Aditya	Austria	21	35000
5	Nishant	Spain	22	25000

Query:

```
SELECT Name
FROM Emp
WHERE Salary
BETWEEN 30000 AND 45000;
```

Output

Name
Shubham
Aman
Naveen
Aditya

SQL NOT Operator

The SQL NOT Operator is a **logical operator** used to **negate** or reverse the result of a condition in SQL queries. It is commonly used with the WHERE clause to filter records that do not meet a specified condition, helping you exclude certain values from your results.

Syntax:

```
SELECT column1, colomn2, …
FROM table_name WHERE NOT condition;
```

Customer ID	Customer Name	City	PostalCode	Country
1	John Wick	New York	1248	USA
2	Around the Horn	London	WA1 1DP	UK
3	Rohan	New Delhi	100084	India

Example 1: Using SQL NOT to Exclude a Specific Value

The following SQL statement selects all fields from Customers table where the country is not UK.

Query:

```
SELECT *
FROM Customers
WHERE NOT Country = 'UK';
```

Output:

Customer ID	Customer Name	City	PostalCode	Country
1	John Wick	New York	1248	USA
3	Rohan	New Delhi	100084	India

Example 2: Using SQL NOT with IN Operator

The NOT operator can also be used with the IN condition to exclude multiple values from the result set.

Query:

```
SELECT *
FROM Customers
WHERE NOT Country IN ('USA', 'UK');
```

Output:

Customer ID	Customer Name	City	PostalCode	Country
3	Rohan	New Delhi	100084	India

Example 3: Using SQL NOT with LIKE Operator

We can also combine NOT with the LIKE operator to exclude records that match a certain pattern.

Query:

```
SELECT *
FROM Customers
WHERE NOT CustomerName LIKE 'R%';
```

Output:

CustomerID	CustomerName	City	PostalCode	Country
1	John Wick	New York	1248	USA
2	Around the Horn	London	WA1 1DP	UK

Example 4: Using SQL NOT with NULL Values

To exclude records where a column has a NULL value, combine NOT with the IS NULL conditio

Query:

```
SELECT *
FROM Customers
WHERE NOT PostalCode IS NULL;
```

Output:

CustomerID	CustomerName	City	PostalCode	Country
1	John Wick	New York	1248	USA
2	Around the Horn	London	WA1 1DP	UK
3	Rohan	New Delhi	100084	India

This query excludes customers who have a NULL value for PostalCode.

Example 5: Using NOT with AND Operator

We can combine NOT with the AND operator to create more complex conditions. This query retrieves customers who are not from the USA and are also not from the UK.

Query:

```
SELECT *
FROM Customers
WHERE NOT Country = 'USA' AND NOT Country = 'UK';
```

Output:

CustomerID	CustomerName	City	PostalCode	Country
3	Rohan	New Delhi	100084	India

Key TakeAways About NOT Operator

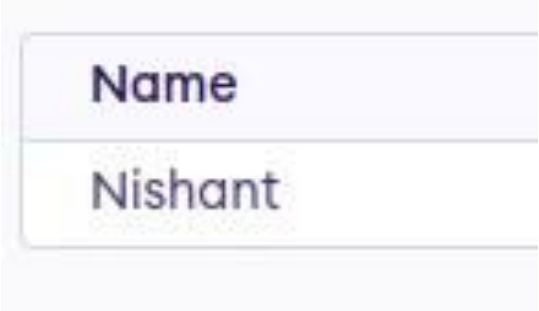
- *NOT operator returns opposite results or negative results. It negates the boolean condition in the WHERE clause.*
- *It is used to exclude specific data from the result set.*

Using the NOT Operator with BETWEEN

Query:

```
SELECT Name
FROM Emp
WHERE Salary
NOT BETWEEN 30000 AND 45000;
```

Output



SQL IN Operator

IN operator allows us to easily test **if the expression matches any value** in the list of values. It is used to **remove** the need for **multiple OR conditions** in **SELECT**, INSERT, **UPDATE**, or DELETE. We can also use **NOT IN** to exclude the rows in our list. We should note that any kind of duplicate entry will be retained.

Syntax

SELECT column_name(s)

FROM table_name

WHERE column_name IN (list_of_values);

Key Features:

- Ideal for filtering non-sequential values.
- Handles duplicates in the list of values.

Emp

EmpID	Name	Country	Age	Salary
1	Shubham	India	23	30000
2	Aman	Australia	21	45000
3	Naveen	Sri lanka	24	40000
4	Aditya	Austria	21	35000
5	Nishant	Spain	22	25000

Example 1: Using IN Operator

Query:

```
SELECT Name
FROM Emp
WHERE Salary IN (30000, 40000, 25000);
```

Output

Name
Shubham
Naveen
Nishant

Example 2: Using the NOT Operator with IN

Query:

```
SELECT Name
FROM Emp
WHERE Salary NOT IN (25000, 30000);
```

Output

Name
Aman
Naveen
Aditya

SQL AND Operator

The AND operator allows you to **filter data** based on multiple conditions, all of which must be true for the record to be included in the result set.

Syntax:

The syntax to use the AND operator in SQL is:

*SELECT * FROM table_name WHERE condition1 AND condition2 AND …conditionN;*

Here,

- **table_name**: name of the table
- **condition1,2,..N**: first condition, second condition, and so on.

SQL OR Operator

The OR Operator in **SQL** displays the records where any one condition is true, i.e. either condition1 or condition2 is True.

Syntax:

The syntax to use the OR operator in SQL is:

*SELECT * FROM table_name WHERE condition1 OR condition2 OR… conditionN;*

- **table_name**: name of the table
- **condition1,2,..N**: first condition, second condition, and so on

SQL AND and OR Operator Examples

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	xxxxxxxxxx	18
2	RAMESH	GURGAON	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
4	SURESH	Delhi	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
2	RAMESH	GURGAON	xxxxxxxxxx	18

Student Table

Example 1: SQL AND Operator

If suppose we want to fetch all the records from the Student table where Age is 18 and ADDRESS is Delhi.

Query:

```
SELECT * FROM Student
WHERE Age = 18 AND ADDRESS = 'Delhi';
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
4	SURESH	Delhi	XXXXXXXXXX	18

Example 2: SQL OR Operator

To fetch all the records from the Student table where NAME is Ram or NAME is SUJIT.

Query:

```
SELECT * FROM Student
WHERE NAME = 'Ram' OR NAME = 'SUJIT';
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
3	SUJIT	ROHTAK	XXXXXXXXXX	20
3	SUJIT	ROHTAK	XXXXXXXXXX	20

Combining AND and OR Operators in SQL

Combining AND and OR Operators in **SQL** allows the creation of complex conditions in queries. This helps in filtering data on multiple conditions.

Syntax:

Syntax to use AND and OR operator in one statement in SQL is:

*SELECT * FROM table_name*

WHERE condition1 AND (condition2 OR condition3);

Example

Let's look at example of combining AND and OR operators in a single statement. In this example we will fetch all the records from the Student table where Age is 18, NAME is Ram or RAMESH.

Query:

```
SELECT * FROM Student WHERE Age = 18 AND (NAME = 'Ram' OR NAME = 'RAMESH');
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
2	RAMESH	GURGAON	XXXXXXXXXX	18

Important Points About SQL AND and OR Operators

- *The SQL AND operator is used to combine multiple conditions, where all the conditions must be true for the row to be included in the result set.*
- *The OR operator is used to combine multiple conditions, where at least one of the conditions must be true for the row to be included in the result set.*
- *Any kind of condition, including equality, inequality, comparison, and logical operators, can be utilized with the AND and OR operators.*
- *The AND operator is more important than the OR operator. In other words, when both are used in the same SQL statement, the AND operator will be executed first. To change the order of evaluation, parentheses can be used.*
- *You can employ the AND and OR operators inside of other conditions because they can both be nested.*

SQL LIKE Operator

The SQL LIKE operator is used for performing **pattern-based** searches in a database. It is used in combination with the **WHERE clause** to filter records based on specified patterns, making it essential for any database-driven application that requires flexible search functionality. LIKE operator is **case-insensitive** by default in most database systems. This means that if you search for "apple" using the LIKE operator, it will return results that include "Apple", "APPLE", "aPpLe", and so on.

Syntax:

SELECT column1, column2, …
FROM table_name
WHERE column_name LIKE pattern;

- column_name: The column to be searched.
- pattern: The pattern to search for, which can include wildcard characters.

For making the LIKE operator case-sensitive, you can use the "**BINARY" keyword in MySQL or the "COLLATE" keyword in other database systems**.

For example:

```
SELECT * FROM products WHERE name LIKE BINARY 'apple%'
```

This following query will only return products whose name starts with “apple” and is spelled exactly like that, without capital letters.

SupplierID	Name	Address
S1	Paragon Suppliers	21-3, Okhla, Delhi
S2	Mango Nation	21, Faridabad, Haryana
S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi
S5	Harish and Sons	Gurgaon, NCR
S6	Om Suppliers	2/1, Faridabad, Haryana

Example 1 : Match Names Starting with 'Ca'

Retrieve SupplierID, Name, and Address from suppliers table, where supplier name starts form k.

```
SELECT SupplierID, Name, Address
FROM Supplier
WHERE Name LIKE 'Ca%';
```

Output:

S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi

Example 2: Match Addresses Containing 'Okhla'

Retrieve entire table, where address contains OKHLA.

```
SELECT *
FROM Supplier
WHERE Address LIKE '%Okhla%';
```

Output:

S1	Paragon Suppliers	21-3, Okhla, Delhi
S3	Canadian Biz	6/7, Okhla Phase II, Delhi

Example 3: Match Names Where 'ango' Appears in the Second Position

Retrieve SupplierID, Name and Address of supplier whose name contains "ango" in second substring.

```
SELECT SupplierID, Name, Address
FROM Supplier
WHERE Name LIKE '_ango%';
```

Output:

S2	Mango Nation	21, Faridabad, Haryana

Example 4: Using LIKE with AND for Complex Conditions

Retrieve suppliers from Delhi with names starting with “C”:

```
SELECT SupplierID, Name, Address
FROM Supplier
WHERE Address LIKE '%Delhi%' AND Name LIKE 'C%';
```

Output:

SupplierID	Name	Address
S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi

Example 5: Using NOT LIKE for Exclusion

To retrieve all suppliers whose name does not contain “Mango”

```
SELECT SupplierID, Name, Address
FROM Supplier
WHERE Name NOT LIKE '%Mango%';
```

Output:

SupplierID	Name	Address
S1	Paragon Suppliers	21-3, Okhla, Delhi
S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi
S5	Harish and Sons	Gurgaon, NCR
S6	Om Suppliers	2/1, Faridabad, Haryana

SQL LIKE Application

The LIKE operator is extremely resourceful in situations such as address filtering wherein we know only a segment or a portion of the entire address (such as locality or city) and would like to retrieve results based on that. The wildcards can be resourcefully exploited to yield even better and more filtered tuples based on the requirement.

Key Takeaways About LIKE Operator

- *LIKE operator is used to search for specific patterns in a column.*
- *It is mostly used with WHERE clause for finding or filtering specific data.*
- *Like Operator is case-insensitive by default, to make it case sensitive, we can use BINARY keyword.*
- *LIKE operator has 4 wild cards, which we can use with LIKE operator to specify the filter. The wild cards are: %,_,[] and -.*

Queries for Practice

```
-- Customer Table CREATE TABLE Customer (
customer_id NUMBER PRIMARY KEY,
name VARCHAR2(100),     email
VARCHAR2(100) UNIQUE,     phone
VARCHAR2(15),
   address VARCHAR2(255)
);

-- Product Table CREATE TABLE Product (
product_id NUMBER PRIMARY KEY,
name VARCHAR2(100),     category
VARCHAR2(50),     price NUMBER(10,2),
stock_quantity NUMBER
);

-- Orders Table
CREATE TABLE Order_Details (
order_id NUMBER PRIMARY KEY,
customer_id NUMBER,     order_date
DATE,
```

```
    total_amount NUMBER(10,2),
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

-- Order Items Table CREATE
TABLE Order_Item (     order_id
NUMBER,     product_id
NUMBER,     quantity NUMBER,
subtotal NUMBER(10,2),
    PRIMARY KEY (order_id, product_id),
    FOREIGN KEY (order_id) REFERENCES Order_Details(order_id),
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
);

-- Employee Table CREATE TABLE Employee1
(     employee_id NUMBER PRIMARY KEY,
name VARCHAR2(100),     role
VARCHAR2(50),     salary NUMBER(10,2),
    hire_date DATE
);

-- Insert Customers
INSERT INTO Customer (customer_id,name, email, phone, address) VALUES
(1,'Alice Johnson', 'alice@gmail.com', '9876543210', 'New York');
INSERT INTO Customer (customer_id,name, email, phone, address) VALUES
(2, 'Bob Smith', 'bob@yahoo.com', '9123456789', 'Los Angeles');
INSERT INTO Customer (customer_id,name, email, phone, address) VALUES
(3, 'Charlie Brown', 'charlie@outlook.com', '9998887776', 'Chicago');
INSERT INTO Customer (customer_id,name, email, phone, address) VALUES
(4, 'David Miller', 'david@gmail.com', '8765432109', 'Miami');
INSERT INTO Customer (customer_id,name, email, phone, address) VALUES (5, 'Emily
Davis', 'emily@hotmail.com', '7654321098', 'New York');
```

```
-- Insert Products
INSERT INTO Product ( product_id, name, category, price, stock_quantity) VALUES (1, 'Milk', 'Dairy', 2.50, 50);
INSERT INTO Product (product_id, name, category, price, stock_quantity)
VALUES
(2, 'Bread', 'Bakery', 1.80, 30);
INSERT INTO Product (product_id, name, category, price, stock_quantity)
VALUES
(3, 'Eggs', 'Dairy', 3.20, 40);
INSERT INTO Product (product_id, name, category, price, stock_quantity)
VALUES
(4, 'Chicken', 'Meat', 7.50, 20);
INSERT INTO Product (product_id, name, category, price, stock_quantity)
VALUES
(5, 'Apples', 'Fruit', 1.20, 60);
INSERT INTO Product (product_id, name, category, price, stock_quantity) VALUES
(6, 'Orange Juice', 'Beverage', 3.50, 25);

-- Insert Orders
INSERT INTO Order_Details (order_id, customer_id, order_date, total_amount) VALUES
(1, 1, TO_DATE('2024-01-10', 'YYYY-MM-DD'),
10.50);
INSERT INTO Order_Details (order_id, customer_id, order_date, total_amount) VALUES
(2, 2, TO_DATE('2024-01-12', 'YYYY-MM-DD'),
15.20);
INSERT INTO Order_Details (order_id, customer_id, order_date, total_amount) VALUES
(3, 3, TO_DATE('2024-02-01', 'YYYY-MM-DD'),
20.80);
INSERT INTO Order_Details (order_id, customer_id, order_date, total_amount) VALUES
(4, 4, TO_DATE('2024-02-05', 'YYYY-MM-DD'),
30.00);
```

```
INSERT INTO Order_Details (order_id, customer_id, order_date, total_amount) VALUES
(5, 5, TO_DATE('2024-02-10', 'YYYY-MM-DD'),
25.50);

-- Insert Employees
INSERT INTO Employee1 ( employee_id, name, role, salary, hire_date) VALUES
(1, 'Michael Scott', 'Manager', 75000.00, TO_DATE('2020-05-10', 'YYYY-MM-DD'));
INSERT INTO Employee1 ( employee_id, name, role, salary, hire_date) VALUES
(2, 'Jim Halpert', 'Cashier', 30000.00, TO_DATE('2021-08-15', 'YYYY-MM-DD'));
INSERT INTO Employee1 ( employee_id, name, role, salary, hire_date) VALUES
(3, 'Pam Beesly', 'Sales Associate', 28000.00, TO_DATE('2022-02-20', 'YYYY-MM-DD'));
INSERT INTO Employee1 ( employee_id, name, role, salary, hire_date) VALUES
(4, 'Dwight Schrute', 'Supervisor', 50000.00, TO_DATE('2019-11-30', 'YYYY-MM-DD'));
INSERT INTO Employee1 ( employee_id, name, role, salary, hire_date) VALUES
(5, 'Kevin Malone', 'Cashier', 29000.00, TO_DATE('2023-03-10', 'YYYY-MM-DD'));
```

1️⃣AND Operator

```
SELECT * FROM Customer
WHERE address = 'New York' AND email LIKE '%@gmail.com';
```

customer_id	name	email	phone	address
1	Alice Johnson	alice@gmail.com	9876543210	New York

```
SELECT * FROM Product
WHERE category = 'Dairy' AND stock_quantity > 20;
```

2️⃣ OR Operator

```
SELECT * FROM Employee
WHERE role = 'Manager' OR role = 'Supervisor';
```

customer_id	name	email	phone	address
1	Alice Johnson	alice@gmail.com	9876543210	New York
5	Emily Davis	emily@hotmail.com	7654321098	New York

```
SELECT * FROM Order_Details
WHERE order_date = TO_DATE('2024-01-10', 'YYYY-MM-DD') OR
order_date = TO_DATE('2024-02-05', 'YYYY-MM-DD');
```

3️⃣ NOT Operator

```
SELECT * FROM Customer
WHERE address NOT LIKE '%New York%';
```

customer_id	name	email	phone	address
2	Bob Smith	bob@yahoo.com	9123456789	Los Angeles
3	Charlie Brown	charlie@outlook.com	9998887776	Chicago
4	David Miller	david@gmail.com	8765432109	Miami

```
SELECT * FROM Employee
WHERE role NOT IN ('Cashier');
```

4️⃣ LIKE Operator

```
SELECT * FROM Customer
WHERE name LIKE 'A%';
```

customer_id	name	email	phone	address
1	Alice Johnson	alice@gmail.com	9876543210	New York

```
SELECT * FROM Customer
WHERE email LIKE '%hotmail%';
```

5️⃣ BETWEEN Operator

```
SELECT * FROM Product
WHERE price BETWEEN 2 AND 5;
SELECT * FROM Customer
WHERE customer_id BETWEEN 2 AND 5;
```

customer_id	name	email	phone	address
2	Bob Smith	bob@yahoo.com	9123456789	Los Angeles
3	Charlie Brown	charlie@outlook.com	9998887776	Chicago
4	David Miller	david@gmail.com	8765432109	Miami
5	Emily Davis	emily@hotmail.com	7654321098	New York

```
SELECT * FROM Employee
WHERE hire_date BETWEEN TO_DATE('2021-01-01',
'YYYY-MM-DD') AND TO_DATE('2023-12-31', 'YYYY-MM-DD');

SELECT * FROM Order_Details
WHERE order_date BETWEEN TO_DATE('2024-02-01',
'YYYY-MM-DD') AND TO_DATE('2024-02-28', 'YYYY-MM-DD');
```

6️⃣ **IN Operator**

```
SELECT * FROM Customer
WHERE address IN ('New York', 'Los Angeles', 'Miami');
```

customer_id	name	email	phone	address
1	Alice Johnson	alice@gmail.com	9876543210	New York
2	Bob Smith	bob@yahoo.com	9123456789	Los Angeles
4	David Miller	david@gmail.com	8765432109	Miami
5	Emily Davis	emily@hotmail.com	7654321098	New York

```
SELECT * FROM Product
WHERE category IN ('Dairy', 'Bakery');

SELECT * FROM Employee
WHERE role IN ('Cashier', 'Sales Associate');
```