**Name:-Purva Tagde**

**PRN:-23070521111**

**Practical no. 4**

**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**

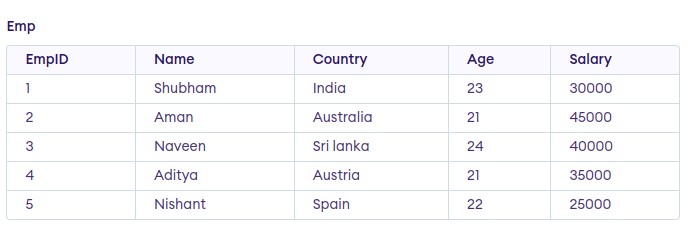
*SELECT column\_name(s)*

*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.



**Query:**

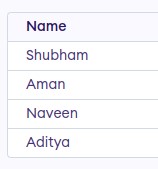
SELECT Name

FROM Emp

WHERE Salary

BETWEEN 30000 AND 45000;

**Output**



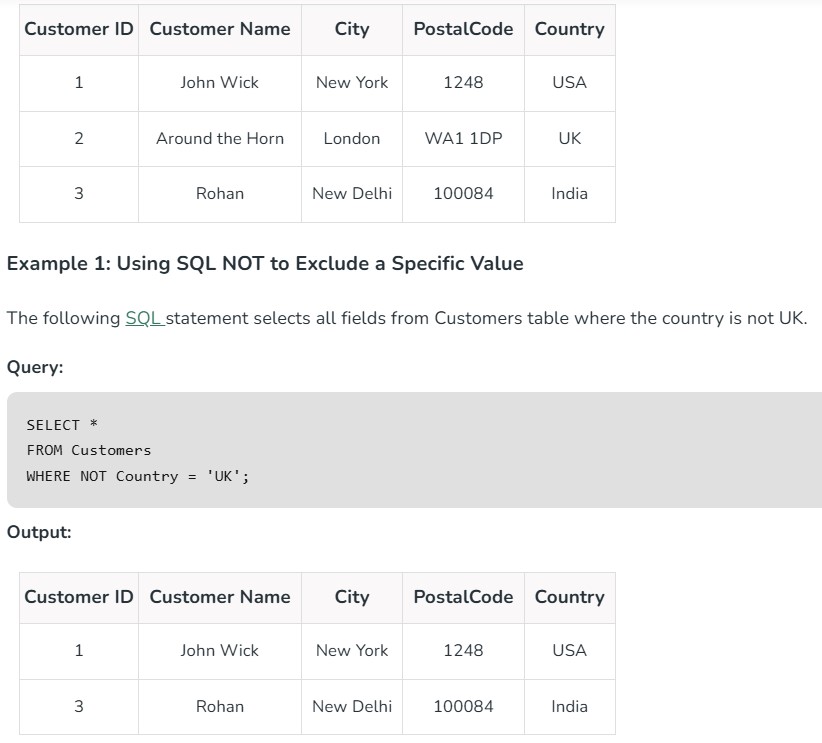
# SQL NOT Operator

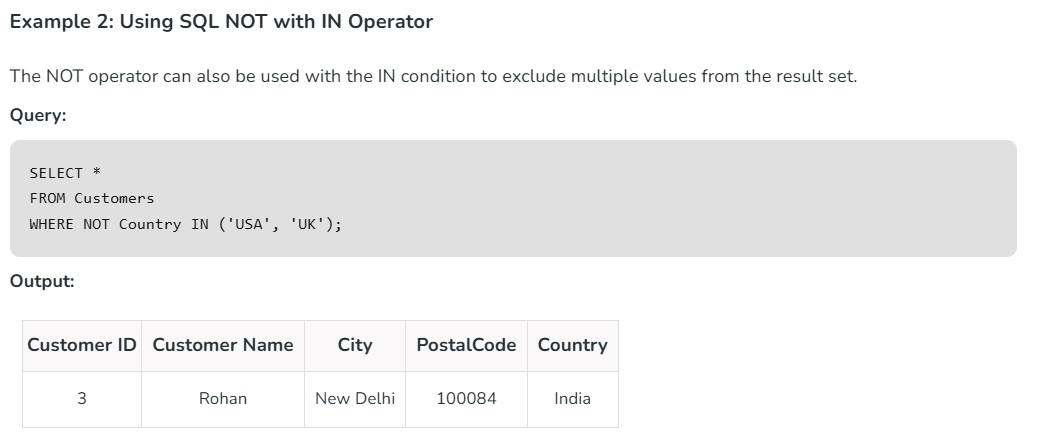
The SQL NOT Operator is a **logical operator** used to **negate** or reverse the result of a condition in SQL queries. It is commonly used with the WHERE clause to filter records that do not meet a specified condition, helping you exclude certain values from your results.

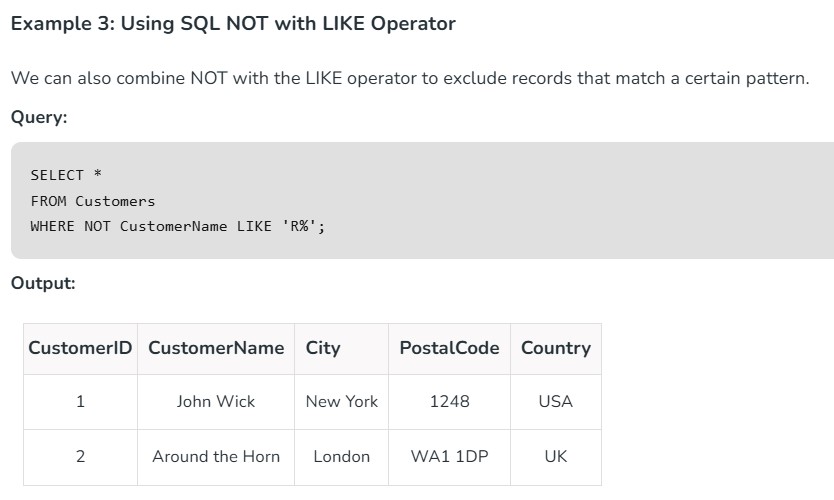
**Syntax:**

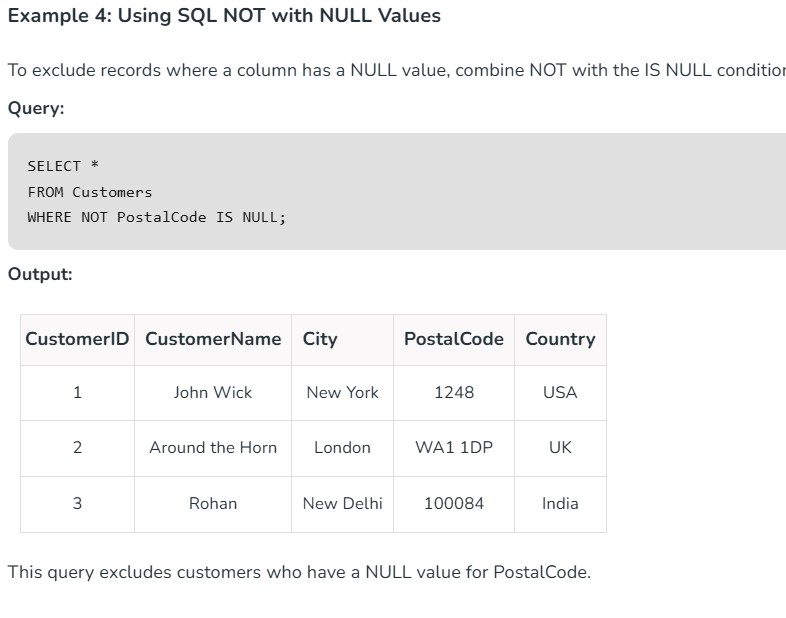
**SELECT** column1, colomn2, …

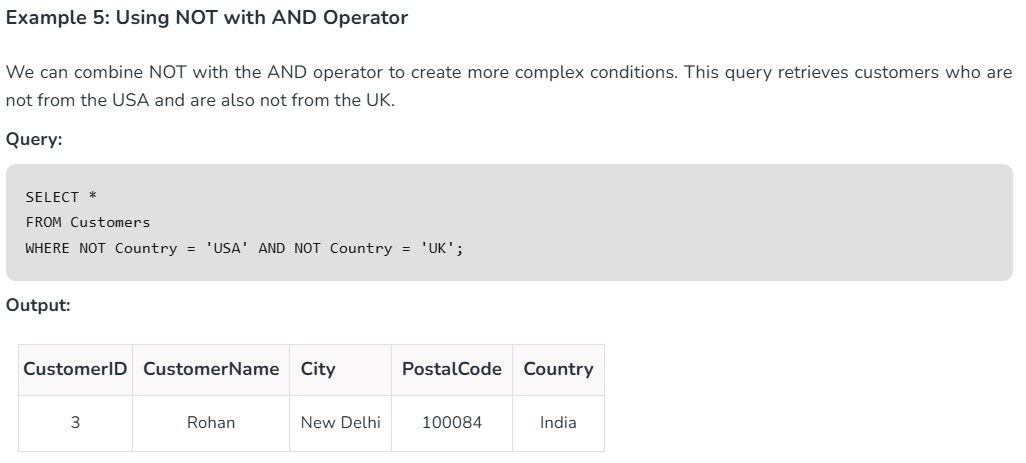
**FROM** table\_name **WHERE NOT** condition;











# 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:**

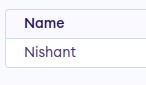
SELECT Name

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**

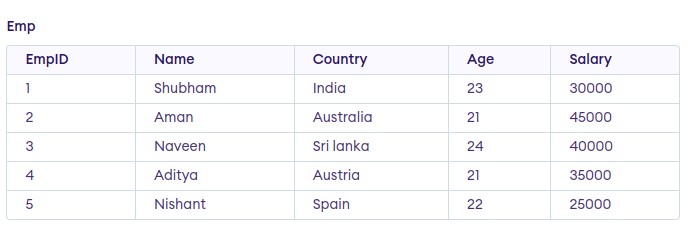
*SELECT column\_name(s)*

*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.



### **Example 1: Using IN Operator**

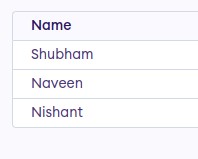
**Query:**

SELECT Name

FROM Emp

WHERE Salary IN (30000, 40000, 25000);

**Output**



### **Example 2: Using the NOT Operator with IN**

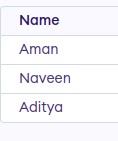
**Query:**

SELECT Name

FROM Emp

WHERE Salary NOT IN (25000, 30000);

**Output**



## 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:

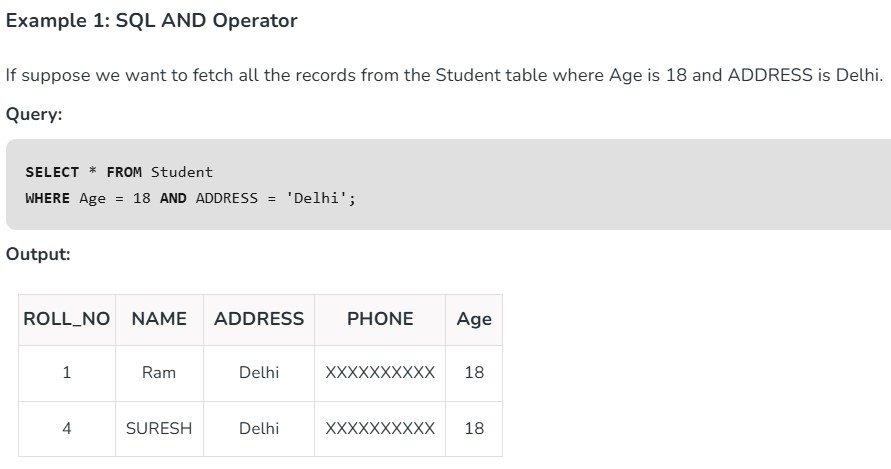
*SELECT \* FROM table\_name WHERE condition1 OR condition2 OR…*

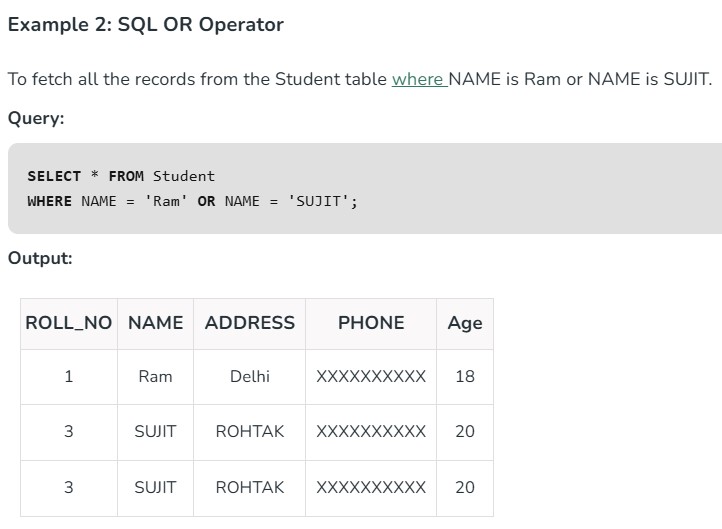
*conditionN;*

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

# SQL AND and OR Operator Examples







## Combining AND and OR Operators in SQL

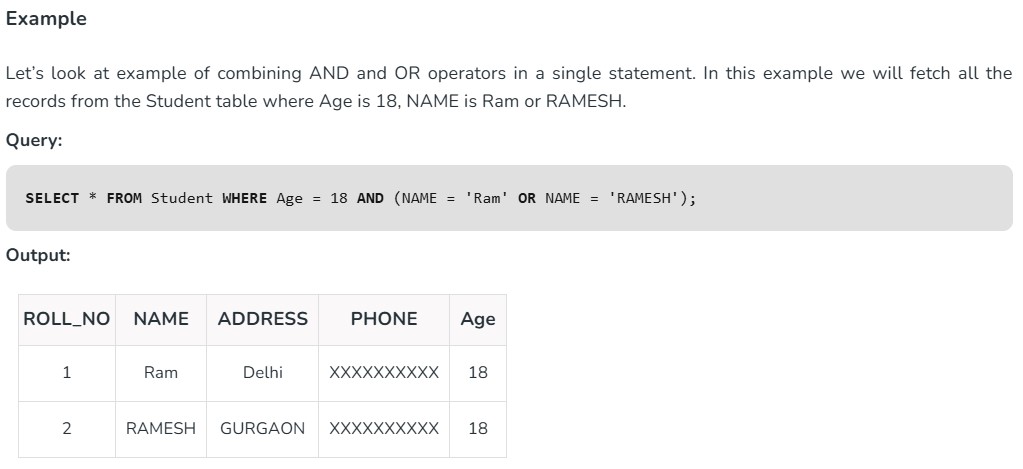
Combining AND and OR Operators in **SQL** allows the creation of complex conditions in queries. This helps in filtering data on multiple conditions.

**Syntax:**

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

*SELECT \* FROM table\_name*

*WHERE condition1 AND (condition2 OR condition3);*



### ***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.

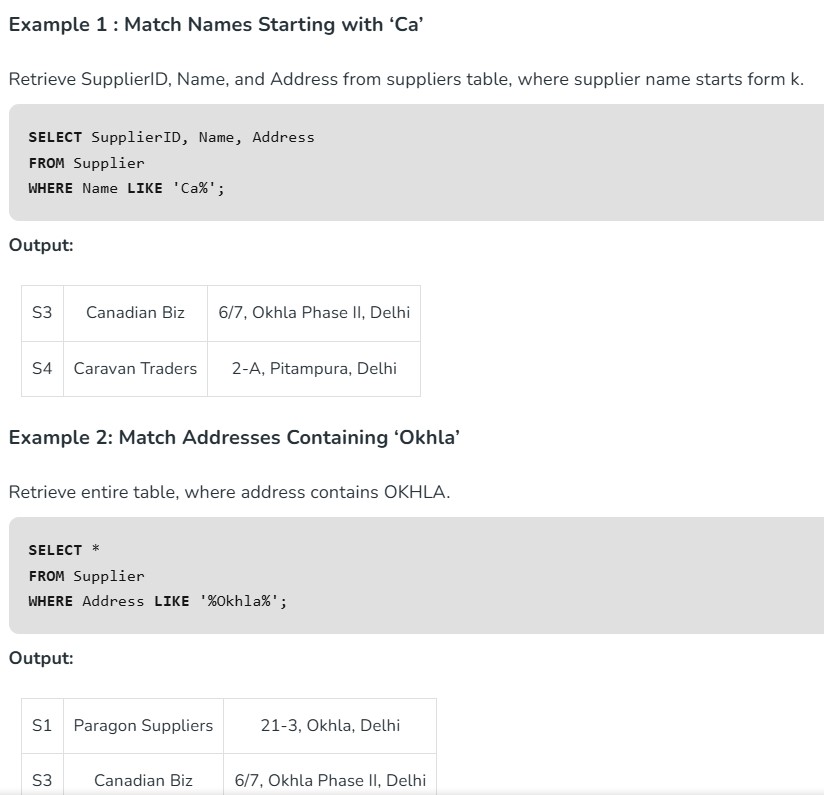
For making the LIKE operator case-sensitive, you can use the “**BINARY” keyword in MySQL or the “COLLATE” keyword in other database systems**.

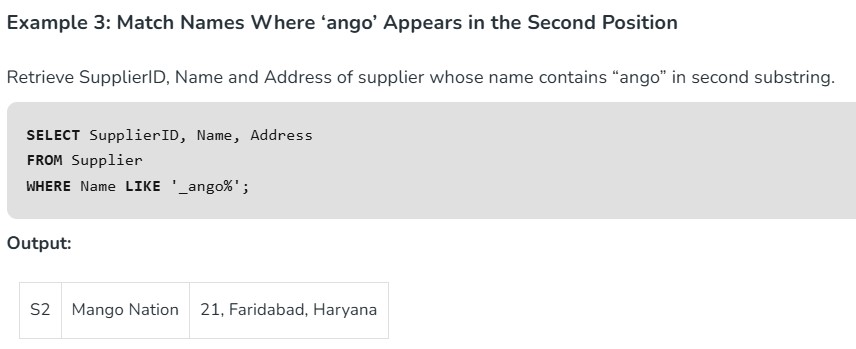
**For example**:

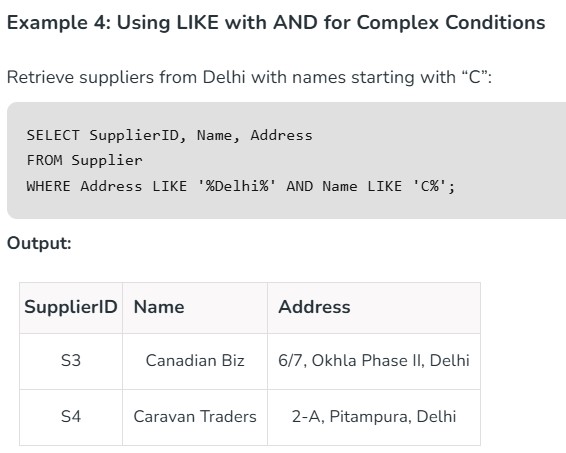
**SELECT** \* **FROM** products **WHERE** name **LIKE BINARY** 'apple%'

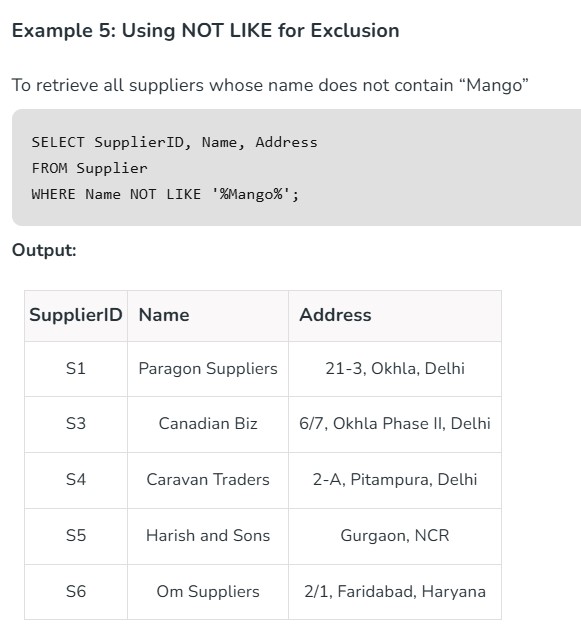
This following query will only return products whose name starts with “apple” and is spelled exactly like that, without capital letters.











## 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

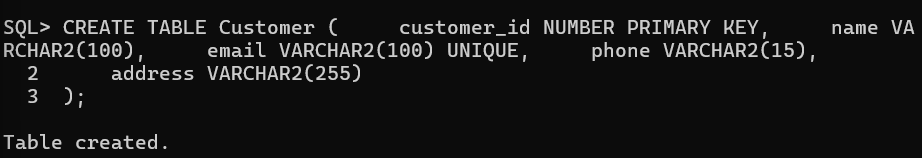
* *LIKE operator is used to search for specific patterns in a column.*
* *It is mostly used with WHERE clause for finding or filtering specific data.*
* *Like Operator is case-insensitive by default, to make it case sensitive, we can use BINARY keyword.*
* *LIKE operator has 4 wild cards, which we can 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),

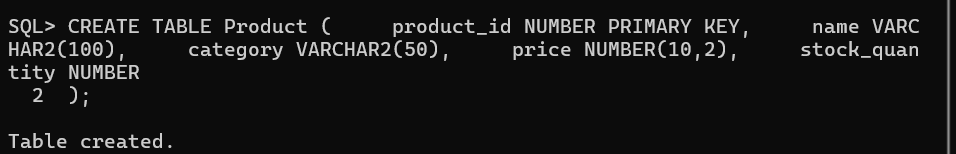
address VARCHAR2(255)

);



-- Product Table CREATE TABLE Product ( product\_id NUMBER PRIMARY KEY, name VARCHAR2(100), category VARCHAR2(50), price NUMBER(10,2), stock\_quantity NUMBER

);



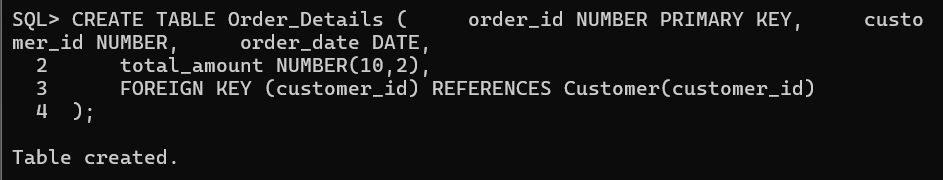
### -- Orders Table

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)

);



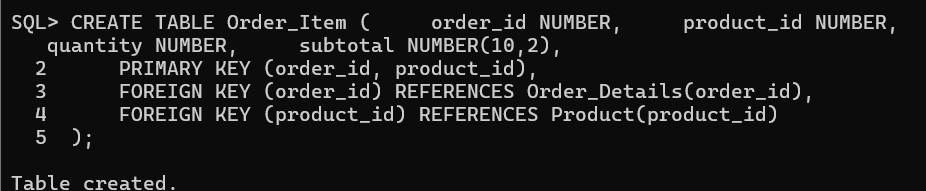
-- Order Items Table CREATE TABLE Order\_Item ( order\_id NUMBER, product\_id NUMBER, quantity NUMBER, subtotal NUMBER(10,2),

PRIMARY KEY (order\_id, product\_id),

FOREIGN KEY (order\_id) REFERENCES Order\_Details(order\_id),

FOREIGN KEY (product\_id) REFERENCES Product(product\_id)

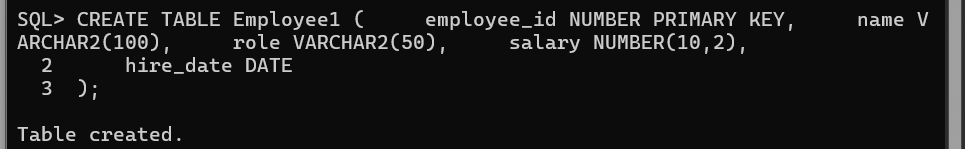
);



-- 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);

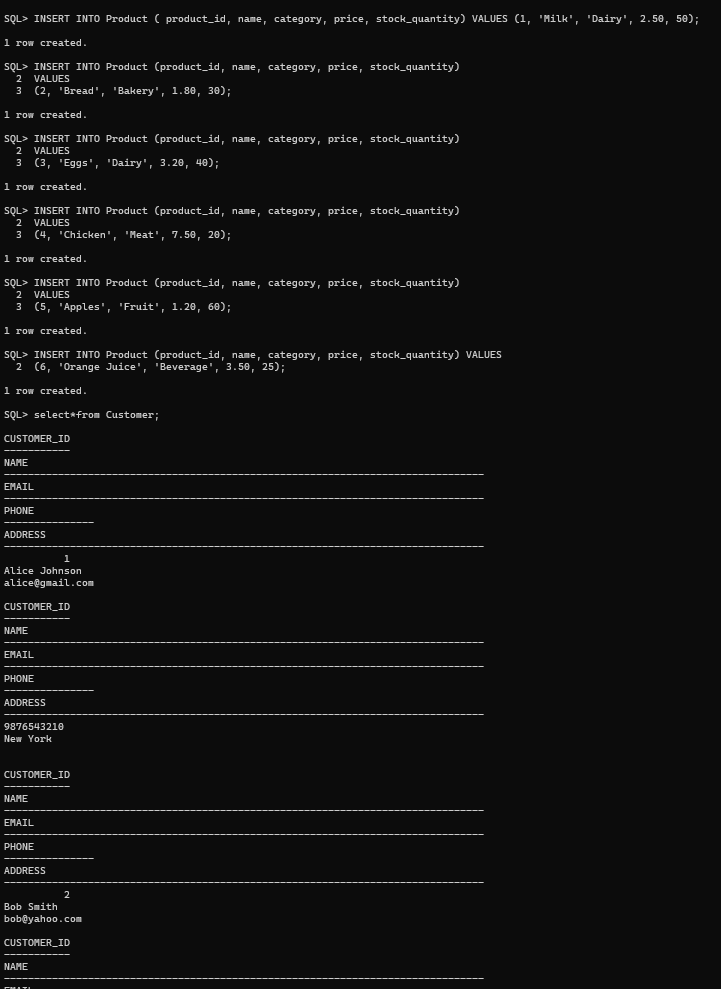
INSERT INTO Product (product\_id, name, category, price, stock\_quantity)

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'),

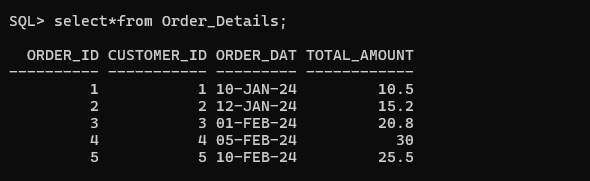
20.80);

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'));

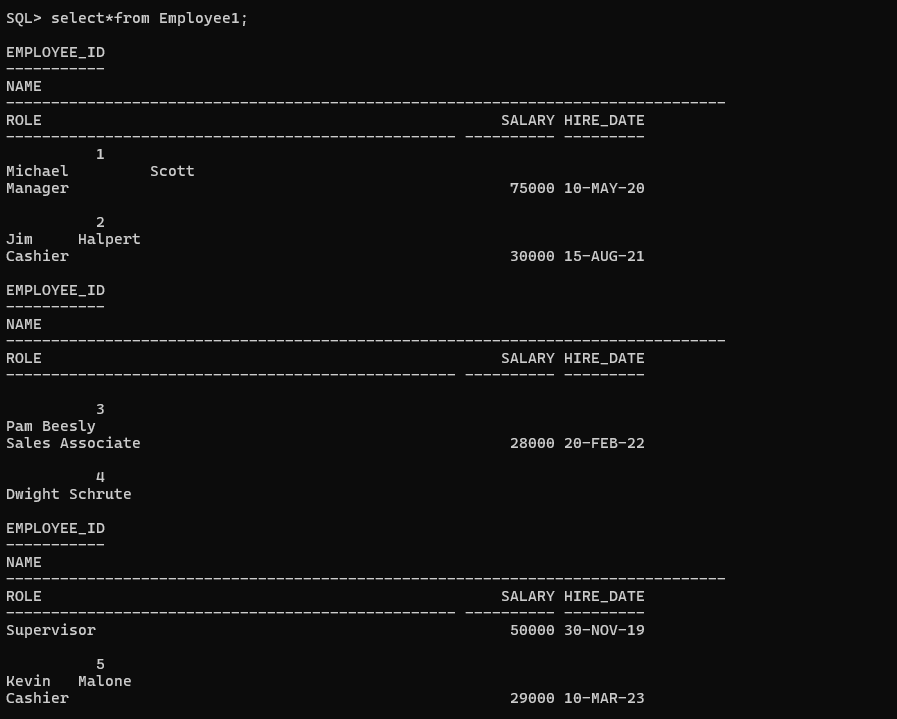
INSERT INTO Employee1 ( employee\_id, name, role, salary, hire\_date) VALUES

(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',

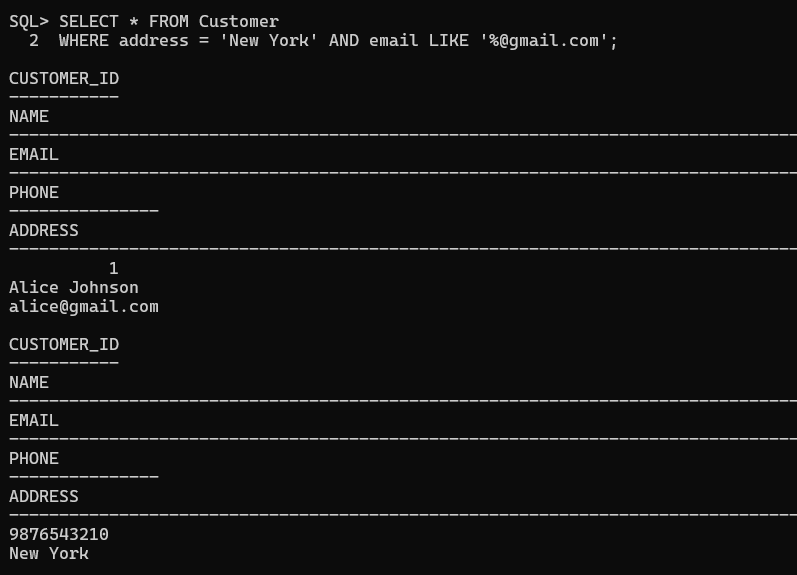
'YYYY-MM-DD'));



1. ⃣ **AND Operator**

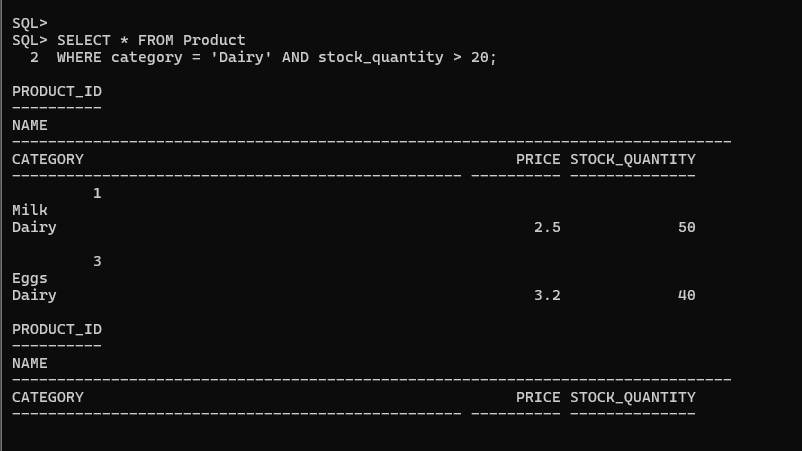
SELECT \* FROM Customer

WHERE address = 'New York' AND email LIKE '%@gmail.com';



SELECT \* FROM Product

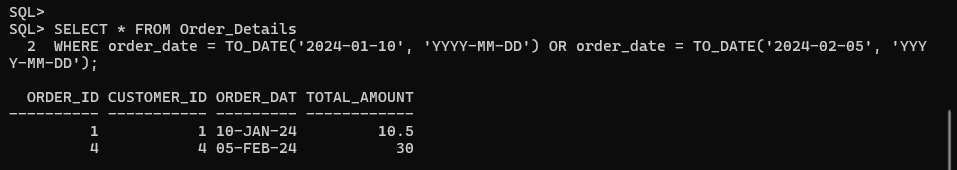
WHERE category = 'Dairy' AND stock\_quantity > 20;



1. ⃣ **OR Operator**

SELECT \* FROM Order\_Details

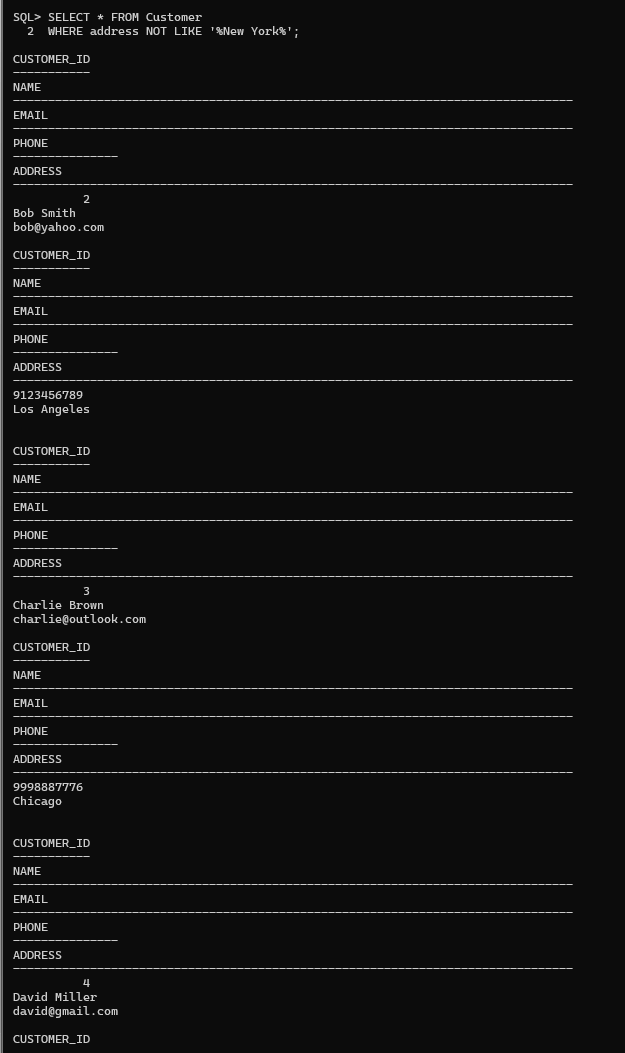
WHERE order\_date = TO\_DATE('2024-01-10', 'YYYY-MM-DD') OR order\_date = TO\_DATE('2024-02-05', 'YYYY-MM-DD');



1. ⃣ **NOT Operator**

SELECT \* FROM Customer

WHERE address NOT LIKE '%New York%';



1. ⃣ **LIKE Operator**

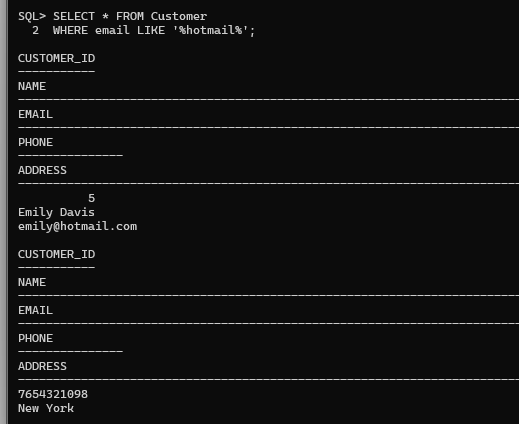
SELECT \* FROM Customer

WHERE name LIKE 'A%';



SELECT \* FROM Customer

WHERE email LIKE '%hotmail%';



1. ⃣ **BETWEEN Operator**

SELECT \* FROM Product

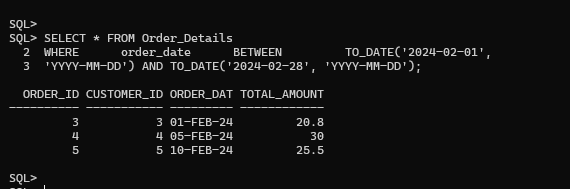
WHERE price BETWEEN 2 AND 5;



SELECT \* FROM Order\_Details

WHERE order\_date BETWEEN TO\_DATE('2024-02-01',

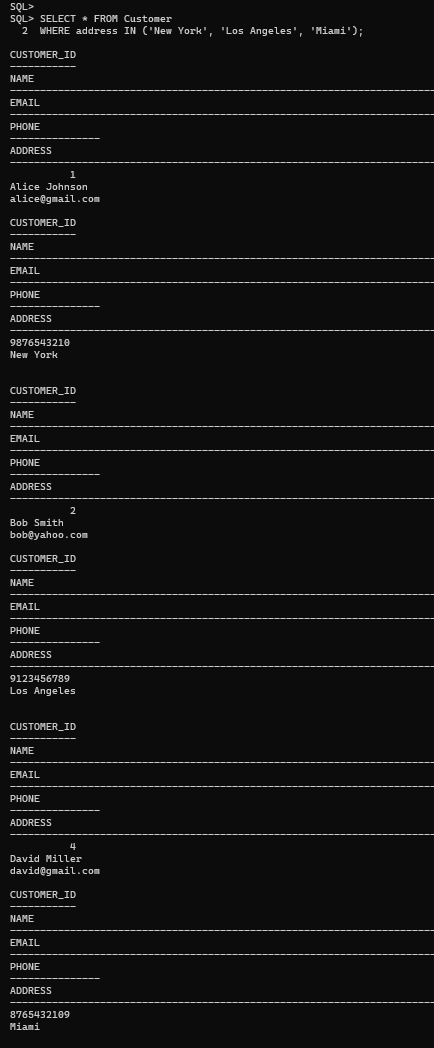
'YYYY-MM-DD') AND TO\_DATE('2024-02-28', 'YYYY-MM-DD');



1. ⃣ **IN Operator**

SELECT \* FROM Customer

WHERE address IN ('New York', 'Los Angeles', 'Miami');



SELECT \* FROM Product

WHERE category IN ('Dairy', 'Bakery');

