UNIT II – Relational Model

Relational Data Model - keys, referential integrity and foreign keys, Relational Algebra - SQL fundamentals- Introduction, data definition in SQL, table, key and foreign key definitions, update behaviors-Views, Triggers, Joins, Constraints, Stored Procedure-Intermediate SQL-Advanced SQL features -Embedded SQL- Dynamic SQL

<u>Intermediate SQL</u>

1. **SQL ORDER BY**

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

Syntax

SELECT *column1*, *column2*, ...
FROM *table_name*
ORDER BY *column1, column2, ...* ASC|DESC;

Example

SELECT * FROM Products ORDER BY Price DESC;

SELECT * FROM Customers ORDER BY Country, CustomerName; ***// Several Columns***

SELECT * FROM Customers ORDER BY Country ASC, CustomerName DESC;

2. **SQL AND Operator**

The WHERE clause can contain one or many AND operators.

Syntax

SELECT *column1*, *column2*, ...
FROM *table_name*
WHERE *condition1* AND *condition2* AND *condition3 ...*;

Example SELECT * FROM Customers WHERE Country = 'Germany' AND Country = 'Spain';

3. **SQL AND Operator**

Example SELECT * FROM Customers WHERE Country = 'Germany' OR Country = 'Spain';

4. **SQL NOT Operator**

Syntax : SELECT *column1*, *column2, ...*FROM *table_name* WHERE NOT *condition*;

Example: SELECT * FROM Customers WHERE NOT Country = 'Spain';

5. **The SQL LIKE Operator**

The LIKE operator is used in a WHERE clause to search for a **specified pattern in a column.**

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign % represents zero, one, or multiple characters
- The underscore sign _ represents one, single character

Syntax: SELECT *column1, column2, ...* FROM *table_name* WHERE *columnN* LIKE *pattern*;

Example:

Select all customers that starts with the letter "a"

SELECT * FROM Customers WHERE CustomerName LIKE 'a%';

Return all customers from a city that starts with 'L' followed by one wildcard character, then 'nd' and then two wildcard characters

SELECT * FROM Customers WHERE City LIKE 'L_nd__';

Return all customers from a city that *contains* the letter 'L'

SELECT * FROM Customers WHERE city LIKE '%L%';

Return all customers that ends with 'a'

SELECT * FROM Customers WHERE CustomerName LIKE '%a'

Return all customers that starts with 'a' or starts with 'b'

SELECT * FROM Customers

WHERE CustomerName LIKE 'a%' OR CustomerName LIKE 'b%';

6. **The SQL IN Operator and NOT IN Operator**

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

Syntax :

SELECT *column_name(s)* FROM *table_name* WHERE *column_name* IN (*value1*, *value2*, ...);

Example

Return all customers from 'Germany', 'France', or 'UK'

SELECT * FROM Customers WHERE Country IN ('Germany', 'France', 'UK');

Return all customers that are NOT from 'Germany', 'France', or 'UK'

SELECT * FROM Customers WHERE Country NOT IN ('Germany', 'France', 'UK');

7. **The SQL BETWEEN Operator**

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

Syntax: SELECT *column_name(s)* FROM *table_name*
WHERE *column_name* BETWEEN *value1* AND *value2;*

Example

Selects all products with a price between 10 and 20:

SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;

8. **SQL Aliases**

SQL aliases are used to give a table, or a column in a table, a temporary name.

Syntax: SELECT *column_name* AS *alias_name* FROM *table_name;*

Example : SELECT CustomerID AS ID FROM Customers;

SELECT CustomerID AS ID, CustomerName AS Customer FROM Customers;

9. **The SQL GROUP BY Statement**

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

Syntax:

SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)*
ORDER BY *column_name(s);*

Example:

```
SELECT COUNT(CustomerID), Country FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

10. **The SQL HAVING Clause**

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

Syntax:

SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*

Example:

```
SELECT COUNT(CustomerID), Country FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```