SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

DBMS Introduction to SQL

Course Name : 23CAT603 - DATA BASE MANAGEMENT SYSTEM

Class : I Year / II Semester

Unit I – Introduction to SQL

Introduction to SQL

- SQL is a standard language for accessing and manipulating databases.

What is SQL?

  - SQL stands for Structured Query Language
  - SQL lets you access and manipulate databases
  - SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

Introduction to SQL

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Introduction to SQL

SQL is a Standard - BUT....

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

Using SQL in Your Web Site

- To build a web site that shows data from a database, you will need:
- An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
- To use a server-side scripting language, like PHP or ASP
- To use SQL to get the data you want
- To use HTML / CSS to style the page

Introduction to SQL

RDBMS

RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

Look at the "Customers" table:

```
SELECT * FROM Customers;
```

Every table is broken up into smaller entities called fields. The fields in the Customers table consist of CustomerID, CustomerName, ContactName, Address, City, PostalCode and Country. A field is a column in a table that is designed to maintain specific information about every record in the table.

A record, also called a row, is each individual entry that exists in a table. For example, there are 91 records in the above Customers table. A record is a horizontal entity in a table.

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

SQL Syntax

SQL Statements

Most of the actions you need to perform on a database are done with SQL statements.

SQL statements consists of keywords that are easy to understand.

The following SQL statement returns all records from a table named "Customers":

Example

Select all records from the Customers table:

SELECT * FROM Customers;

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

SQL Syntax

Database Tables

A database most often contains one or more tables. Each table is identified by a name (**e.g. "Customers" or "Orders"**), and contain records (rows) with data.

In this tutorial we will use the well-known Northwind sample database (included in MS Access and MS SQL Server).

Below is a selection from the **Customers** table used in the examples:

The table above contains five records (one for each customer) and seven columns (CustomerID, CustomerName, ContactName, Address, City, PostalCode, and Country).

•SQL keywords are NOT case sensitive: `select` is the same as `SELECT`

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

SQL

Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

Some of The Most Important SQL Commands

- `SELECT` - extracts data from a database
- `UPDATE` - updates data in a database
- `DELETE` - deletes data from a database
- `INSERT INTO` - inserts new data into a database
- `CREATE DATABASE` - creates a new database
- `ALTER DATABASE` - modifies a database
- `CREATE TABLE` - creates a new table
- `ALTER TABLE` - modifies a table
- `DROP TABLE` - deletes a table
- `CREATE INDEX` - creates an index (search key)
- `DROP INDEX` - deletes an index

SQL SELECT Statement

- The SQL SELECT Statement
- The SELECT statement is used to select data from a database.

Example

- Return data from the Customers table:
- SELECT CustomerName, City FROM Customers;

CustomerName	City
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y helados	México D.F.
Antonio Moreno Taquería	México D.F.

SQL SELECT Statement

- The SQL SELECT Statement
- The SELECT statement is used to select data from a database.

Example

- Return data from the Customers table:
- SELECT CustomerName, City FROM Customers;

CustomerName	City
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y helados	México D.F.
Antonio Moreno Taquería	México D.F.

SQL

Syntax

SELECT column1, column2, ...

FROM table_name;

Here, column1, column2, ... are the field names of the table you want to select data from.

The table_name represents the name of the table you want to select data from.

Demo Database

Below is a selection from the **Customers** table used in the examples:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico

SQL

Select ALL columns

If you want to return all columns, without specifying every column name, you can use the SELECT * syntax:

Example

Return all the columns from the Customers table:

```
SELECT * FROM Customers;
```

The SQL SELECT DISTINCT Statement

The `SELECT DISTINCT` statement is used to return only distinct (different) values.

Example

Select all the different countries from the "Customers" table:

```
SELECT DISTINCT Country FROM Customers;
```

Country
Argentina
Austria
Belgium

SQL

Select ALL columns

If you want to return all columns, without specifying every column name, you can use the SELECT * syntax:

Example

Return all the columns from the Customers table:

```
SELECT * FROM Customers;
```

The SQL SELECT DISTINCT Statement

The `SELECT DISTINCT` statement is used to return only distinct (different) values.

Example

Select all the different countries from the "Customers" table:

```
SELECT DISTINCT Country FROM Customers;
```

Country
Argentina
Austria
Belgium

SQL WHERE Clause

The SQL WHERE Clause

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

Example

Select all customers from Mexico:

```
SELECT * FROM Customers WHERE Country='Mexico';
```

SQL WHERE Clause

Example

Select all customers from Mexico:

```
SELECT * FROM Customers WHERE Country='Mexico';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
58	Pericles Comidas clásicas	Guillermo Fernández	Calle Dr. Jorge Cash 321	México D.F.	05033	Mexico
80	Tortuga Restaurante	Miguel Angel Paolino	Avda. Azteca 123	México D.F.	05033	Mexico

SQL

Syntax

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Note: The `WHERE` clause is not only used in `SELECT` statements, it is also used in `UPDATE`, `DELETE`, etc.!

Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes:

Example

```
SELECT * FROM Customers WHERE CustomerID=1;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

SQL

Operators in The WHERE Clause

You can use other operators than the = operator to filter the search.

Example

Select all customers with a CustomerID greater than 80:

SELECT * FROM Customers WHERE CustomerID > 80;

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
81	Tradição Hipermercados	Anabela Domingues	Av. Inês de Castro, 414	São Paulo	05634-030	Brazil
82	Trail's Head Gourmet Provisioners	Helvetius Nagy	722 DaVinci Blvd.	Kirkland	98034	USA
83	Vaffeljernet	Palle Ibsen	Smagsløget 45	Århus	8200	Denmark
84	Victuailles en stock	Mary Saveley	2, rue du Commerce	Lyon	69004	France
85	Vins et alcools Chevalier	Paul Henriot	59 rue de l'Abbaye	Reims	51100	France

SQL

SELECT * FROM Products WHERE Price = 18;

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
35	Steeleye Stout	16	1	24 - 12 oz bottles	18
39	Chartreuse verte	18	1	750 cc per bottle	18
76	Lakkalikööri	23	1	500 ml	18

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97
10	Ikura	4	8	12 - 200 ml jars	31
12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38

SQL

SELECT * FROM Products WHERE Price < 30;

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10

SELECT * FROM Products WHERE Price >= 30;

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97
10	Ikura	4	8	12 - 200 ml jars	31

SQL

SELECT * FROM Products WHERE Price <= 30;

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22

S

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

SQL

SELECT * FROM Products WHERE Price BETWEEN 50 AND 60;

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
51	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53
59	Raclette Courdavault	28	4	5 kg pkg.	55

SELECT * FROM Customers WHERE City LIKE 's%';

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil

SQL

SELECT * FROM Customers WHERE City IN ('Paris','London');

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK

SQL ORDER BY Keyword

The SQL ORDER BY

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

Example

Sort the products by price:

SELECT * FROM Products ORDER BY Price;

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
33	Geitost	15	4	500 g	2.5
24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.5
13	Konbu	6	8	2 kg box	6

SQL ORDER BY Keyword

Syntax

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

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
33	Geitost	15	4	500 g	2.5
24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.5
13	Konbu	6	8	2 kg box	6
52	Filo Mix	24	5	16 - 2 kg boxes	7
54	Tourtière	25	6	16 pies	7.45
75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7.75
23	Tunnbröd	9	5	12 - 250 g pkgs.	9
19	Teatime Chocolate Biscuits	8	3	10 boxes x 12 pieces	9.2

SQL AND Operator

SQL AND Operator
The SQL AND Operator
The WHERE clause can contain one or many AND operators.

The AND operator is used to filter records based on more than one condition, like if you want to return all customers from Spain that starts with the letter 'G':

Example

Select all customers from Spain that starts with the letter 'G':

```
SELECT * FROM Customers
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
29	Galería del gastrónomo	Eduardo Saavedra	Rambla de Cataluña, 23	Barcelona	08022	Spain
30	Godos Cocina Típica	José Pedro Freyre	C/ Romero, 33	Sevilla	41101	Spain

SQL AND Operator

Syntax

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

AND vs OR

The `AND` operator displays a record if *all* the conditions are TRUE.
The `OR` operator displays a record if *any* of the conditions are TRUE.

SQL OR Operator

The SQL OR Operator

The WHERE clause can contain one or more OR operators.

The OR operator is used to filter records based on more than one condition, like if you want to return all customers from Germany but also those from Spain:

Example

Select all customers from Germany or Spain:

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

SQL OR Operator

SQL OR Operator

The SQL OR Operator

The WHERE clause can contain one or more OR operators.

The OR operator is used to filter records based on more than one condition, like if you want to return all customers from Germany but also those from Spain:

Example

Select all customers from Germany or Spain:

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

Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

OR vs AND

The `OR` operator displays a record if *any* of the conditions are TRUE.
The `AND` operator displays a record if *all* the conditions are TRUE.

SQL NOT Operator

The NOT Operator

The NOT operator is used in combination with other operators to give the opposite result, also called the negative result.

In the select statement below we want to return all customers that are NOT from Spain:

Example

Select only the customers that are NOT from Spain:

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

SQL INSERT INTO Statement

The SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

SQL NULL Values

What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation.

How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

<u>IS NULL Syntax</u>

SELECT *column_names*
FROM *table_name*
WHERE *column_name* IS NULL;

<u>IS NOT NULL Syntax</u>

SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;

SQL UPDATE Statement

The SQL UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

UPDATE Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Note: Be careful when updating records in a table! Notice the `WHERE` clause in the `UPDATE` statement.
The `WHERE` clause specifies which record(s) that should be updated. If you omit the `WHERE` clause, all records in the table will be updated!

SQL DELETE Statement

The SQL DELETE Statement

The DELETE statement is used to delete existing records in a table.

DELETE Syntax

DELETE FROM table_name WHERE condition;

Note: Be careful when deleting records in a table! Notice the `WHERE` clause in the `DELETE` statement.
The `WHERE` clause specifies which record(s) should be deleted.
If you omit the `WHERE` clause, all records in the table will be deleted!

SQL TOP, LIMIT, FETCH FIRST or ROWNUM Clause

Note: Not all database systems support the `SELECT TOP` clause.
MySQL supports the `LIMIT` clause to select a limited number of records, while Oracle uses `FETCH FIRST n ROWS ONLY` and `ROWNUM`.

SQL Server / MS Access Syntax:

```
SELECT TOP number|percent column_name(s)
FROM table_name WHERE condition;
```

MySQL Syntax:

```
SELECT column_name(s) FROM table_name WHERE condition LIMIT number;
```

Oracle 12 Syntax:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s)
FETCH FIRST number ROWS ONLY;
```

Older Oracle Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE ROWNUM <= number;
```

Older Oracle Syntax (with ORDER BY):

```
SELECT *
FROM (SELECT column_name(s) FROM table_name ORDER BY column_name(s))
WHERE ROWNUM <= number;
```

SQL Aggregate Functions

SQL Aggregate Functions

SQL Aggregate Functions
An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

Aggregate functions are often used with the GROUP BY clause of the SELECT statement. The GROUP BY clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.

The most commonly used SQL aggregate functions are:

MIN() - returns the smallest value within the selected column
MAX() - returns the largest value within the selected column
COUNT() - returns the number of rows in a set
SUM() - returns the total sum of a numerical column
AVG() - returns the average value of a numerical column
Aggregate functions ignore null values (except for COUNT()).

SQL MIN() and MAX() Functions

The SQL MIN() and MAX() Functions

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

MIN Example

Find the lowest price in the Price column:

```
SELECT MIN(Price) FROM Products;
```

Expr1000
2.5

MAX Example

Find the highest price in the Price column:

```
SELECT MAX(Price)
FROM Products;
```

Expr1000
263.5

SQL MIN() and MAX() Functions

Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;

SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

SQL COUNT() Function

SQL COUNT() Function

```
SELECT COUNT(*) FROM Products;
```

Expr1000
77

Syntax

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

SQL SUM() Function

The SQL SUM() Function

The SUM() function returns the total sum of a numeric column.

Example

Return the sum of all Quantity fields in the OrderDetails table:

```
SELECT SUM(Quantity) FROM OrderDetails;
```

Expr1000
12743

Syntax

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

SQL AVG() Function

The SQL AVG() Function

The AVG() function returns the average value of a numeric column.

Example

Find the average price of all products:

SELECT AVG(Price) FROM Products;

Expr1000
28.8664

Note: NULL values are ignored.

Syntax

SELECT AVG(column_name)
FROM table_name
WHERE condition;

SQL LIKE Operator

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

Example

Select all customers that starts with the letter "a":

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

SQL LIKE Operator

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

Example

Select all customers that starts with the letter "a":

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

Syntax

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

SQL Wildcards

<u>SQL Wildcard Characters</u>

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the LIKE operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

<u>Example</u>

Return all customers that starts with the letter 'a':

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

SQL Wildcards

SQL Wildcard Characters

Wildcard Characters

* Not supported in PostgreSQL and MySQL databases.
** Supported only in Oracle databases.

Symbol	Description
%	Represents zero or more characters
_	Represents a single character
[]	Represents any single character within the brackets *
^	Represents any character not in the brackets *
-	Represents any single character within the specified range *
{}	Represents any escaped character **

SQL IN Operator

The SQL 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.

Example

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

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany

SQL IN Operator

The SQL 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.

Example

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

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

Syntax

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany

SQL BETWEEN Operator

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.

Example

Selects all products with a price between 10 and 20:

SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;

Syntax

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

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
15	Genen Shouyu	6	2	24 - 250 ml bottles	15.5

SQL Aliases

<u>SQL Aliases</u>

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

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

<u>Example</u>

SELECT CustomerID AS ID FROM Customers;

ID
1
2
3
4

SQL Aliases

AS is Optional

Actually, in most database languages, you can skip the AS keyword and get the same result:

Example

SELECT CustomerID ID FROM Customers;

ID
1
2
3
4

Syntax

When alias is used on column:

```
SELECT column_name AS alias_name
FROM table_name;
```

When alias is used on table:

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

References

https://www.geeksforgeeks.org/introduction-of-er-model/?ref=lbp

https://www.javatpoint.com/dbms-er-model-concept

https://www.ntrgdc.ac.in/userfiles/file/Course%20Materials/MPCS_DBMS_Notes_Unit-II_NTRGDC.pdf

https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms