

**COURSE:** CSE-311L Database Management System

**LAB:** 2,3

**INSTRUCTOR:** Azmary Zannat Aurin

Below is a selection from the "Customers" table in the Northwind sample database:



# Activity – 01: Create and populate the ‘Customers’ table.

CREATE TABLE customers(

CustomerID int not null AUTO\_INCREMENT, CustomerName VARCHAR(50),

ContactName VARCHAR(50), Address VARCHAR(50),

City VARCHAR(50),

PostalCode VARCHAR(50), Country VARCHAR(50), PRIMARY key(CustomerID));

INSERT INTO CUSTOMERS(CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES('Ana Trujillo Emparedados y helados', 'Ana Trujillo', 'Avda. de la Constitución 2222', 'México D.F.', '05021', 'Mexico'),

('Antonio Moreno Taquería', 'Antonio Moreno', 'Mataderos 2312', 'México D.F.', '05023', 'Mexico'),

('Around the Horn', 'Thomas Hardy', '120 Hanover Sq.', 'London', 'WA1 1DP', 'UK'),

('Berglunds snabbköp', 'Christina Berglund', 'Berguvsvägen 8', 'Luleå', 'S-958 22', 'Sweden');

**WHERE Clause**

The WHERE clause is used to filter records. It is used to extract only those records that fulfill a specified condition.

# 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.!

* You can use one or more tables separated by a comma to include various conditions using a WHERE clause, but the WHERE clause is an optional part of the SELECT command.
* You can specify any condition using the WHERE clause.
* You can specify more than one condition using the **AND** or the **OR** operators.
* A WHERE clause can be used along with DELETE or UPDATE SQL command also to specify a condition.

The **WHERE** clause works like an **if condition** in any programming language. This clause is used to compare the given value with the field value available in a MySQL table. If the given value from outside is equal to the available field value in the MySQL table, then it returns that row. If the given condition does not match any record in the table, then the query would not return any row.

# Operators in The WHERE Clause:

The following operators can be used in the WHERE clause:

= *equal*

< *less than*

* *greater than*

<= *less than or equal*

>= *greater than or equal*

<> *not equal.* ***Note:*** *In some versions of SQL this operator may be written as !=*

BETWEEN *between a certain range*

LIKE *search for a pattern*

IN *to specify multiple possible values for a column*

# Activity – 02: Write a query to show all the customers from "Mexico".

SELECT \*

FROM Customers

WHERE Country = 'Mexico';

# Activity – 03: Select all records where the City column has the value "Berlin".

SELECT \*

FROM Customers WHERE city = 'berlin';

# Activity – 04: Select all records where "Berlin" is not the city.

SELECT \*

FROM Customers WHERE city <> 'berlin';

**ADD, OR, NOT Statement**

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

* + The AND operator displays a record if all the conditions separated by AND are TRUE.
  + The OR operator displays a record if any of the conditions separated by OR is TRUE.
  + The NOT operator displays a record if the condition(s) is NOT TRUE.

# AND Syntax:

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE *condition1* AND *condition2* AND *condition3 ...*;

# OR Syntax:

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE *condition1* OR *condition2* OR *condition3 ...*;

# NOT Syntax:

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE NOT *condition*;

# Activity – 05: Write an SQL query to show the details of the customers from Germany and Berlin.

SELECT \*

FROM Customers

WHERE Country = 'Germany' AND City = 'Berlin';

# Activity - 06: Select all records where "Berlin" is not the city.

SELECT \*

FROM Customers WHERE not city = 'berlin';

# Activity – 07: Show the customer IDs and customer names whose country is Germany AND city is either Berlin OR Stuttgart.

SELECT CustomerID, CustomerName FROM Customers

WHERE Country = 'Germany' AND (City = 'Berlin' OR City = 'Stuttgart');

# Activity – 08: Write an SQL query to show the cities and postal codes where country is NOT "Germany" and NOT "USA".

SELECT City, PostalCode FROM Customers

WHERE NOT Country = 'Germany' AND NOT Country = 'USA';

# Activity – 09: Select the customer IDs from city 'Berlin' *and* the PostalCode 12209.

SELECT CustomerID FROM Customers

WHERE city = ‘Berlin’ and PostalCode = ‘12209’;

**IN, 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.

# IN Syntax

SELECT *column\_name(s)*

FROM *table\_name*

WHERE *column\_name* IN (*value1*, *value2*, ...);

# NOT IN Syntax

SELECT *column\_name(s)*

FROM *table\_name*

WHERE *column\_name* NOT IN (*value1*, *value2*, ...);

# Activity – 10: Show the records of all the customers who are located in "Germany", "France" or "UK".

SELECT \* FROM Customers

WHERE Country IN ('Germany', 'France', 'UK');

# Activity – 11: Show the records of all the customers who are not located in "Germany", "France" or "UK".

SELECT \* FROM Customers

WHERE Country NOT IN ('Germany', 'France', 'UK');

**ORDER BY**

The ORDER BY keyword is used to sort the result-set in ascending or descending order. The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

# Syntax:

SELECT *column1*, *column2, ...*

FROM *table\_name*

ORDER BY *column1, column2, ...* ASC|DESC;

# Activity – 12: Show the country names from the ‘Customers’ table, sorted by the "Country" column in ascending order.

SELECT CustomerName, country FROM Customers

ORDER BY COUNTRY;

# Activity – 13: Show the customer IDs and the country names sorted by the customer ID in ascending order:

SELECT CustomerID, CustomerName FROM Customers

ORDER BY CustomerID desc;

**DISTINCT Clause**

SQL DISTINCT clause is used to remove the duplicate columns from the result set. The unique values are fetched when we use the distinct keyword. DISTINCT can be used with aggregates: COUNT, AVG, MAX, etc. DISTINCT operates on a single column. Multiple columns are not supported for DISTINCT.

# Syntax:

SELECT DISTINCT expressions FROM table\_name

WHERE conditions;

**LIMIT Clause**

The LIMIT clause is used to specify the number of records to return.

The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

# Syntax:

SELECT *column\_name(s)* FROM *table\_name* WHERE *condition* LIMIT *number*;

# Activity 14: Select the first three records from the "Customers" table, where the country is Germany.

SELECT \*

FROM Customers

WHERE Country='Germany' LIMIT 3;

**MySQL Aliases**

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. Single or double quotation marks are required if the alias name contains spaces.

# Alias Column Syntax:

SELECT *column\_name* AS *alias\_name*

FROM *table\_name;*

# Alias Table Syntax:

SELECT *column\_name(s)*

FROM *table\_name* AS *alias\_name;*

# Activity 15: Create two aliases, one for the CustomerName column and one for the ContactName column.

SELECT CustomerName AS Customer,ContactName AS "Contact\_Person" FROM Customers;

**UPDATE Statement**

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

# Syntax

UPDATE *table\_name*

SET *column1* = *value1*, *column2* = *value2*… WHERE *condition*;

# Activity – 16: Update the first customer with a new contact person Alfred Schmidt *and* a new city Frankfurt.

UPDATE Customers

SET ContactName= 'Alfred',City= 'Frankfurt' WHERE CustomerID = 1;

**ALTER Statement**

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

# ADD Column:

ALTER TABLE *table\_name*

ADD *column\_name datatype*;

# Activity - 17: Add an "Email" column to the "Customers" table.

ALTER TABLE Customers ADD Email varchar(255);

# DROP Column:

ALTER TABLE *table\_name*

DROP COLUMN *column\_name*;

# Activity – 18: Delete the "Email" column from the "Customers" table.

ALTER TABLE Customers DROP COLUMN Email;

# CHANGE column:

ALTER TABLE table\_name

Change OldColumnName NewColumnName datatype

**DELETE Statement**

The DELETE statement is used to delete existing records in a table. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted.

# DELETE Syntax

DELETE

FROM *table\_name*

WHERE *condition*;

# Activity – 19: Delete the customer "Alfreds Futterkiste" from the "Customers" table.

DELETE

FROM Customers

WHERE CustomerName='Alfreds Futterkiste';

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact.

# Syntax:

DELETE

FROM *table\_name*;

# Activity – 20: Delete all rows in the "Customers" table, without deleting the table.

DELETE

FROM Customers;

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

There are different wildcards used for different purposes.

* + 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*;

# Activity–21: Select all customers with a CustomerName starting with "a".

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

# Activity-22: Select all customers with a CustomerName that have "or" in any position.

SELECT \* FROM Customers

WHERE CustomerName LIKE '%or%';

# Activity-23: Select all customers with a CustomerName that have "r" in the second position.

SELECT \* FROM Customers WHERE CustomerName LIKE '\_r%';

# Activity-24: Selects all customers with a CustomerName that starts with "a" and are at least 3 characters in length.

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

# Activity-25: Select all customers with a ContactName that starts with "a" and ends with "o".

SELECT \* FROM Customers WHERE ContactName LIKE 'a%o';

# Activity-26: Select all customers with a CustomerName that does NOT start with "a".

SELECT \* FROM Customers

WHERE CustomerName NOT LIKE 'a%';

**FUNCTIONS**

Log() Upper() Lower() Concat()

Pow(): SELECT POW(2,2)

Greatest(): SELECT greatest(10,2,-3) Least(): SELECT least(-2,3.8,100)

Truncate(): select TRUNCATE(10.87655, 3)

**AGGREGATE FUNCTIONS**

## *Count():*

The COUNT() function returns the number of rows that matches a specified criterion.

Syntax:

SELECT COUNT(*column\_name*) FROM *table\_name*

WHERE *condition*;

Example:

SELECT count(distinct city)

FROM customers

where CustomerName LIKE 'a%';

## *MIN(), MAX():*

The MIN() function returns the smallest value of the selected column. The MAX() function returns the largest value of the selected column. MIN() Syntax:

SELECT MIN(*column\_name*) FROM *table\_name*

WHERE *condition*

MAX() Syntax:

SELECT MAX(*column\_name*) FROM *table\_name*

WHERE *condition*

## *SUM():*

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

SELECT SUM(*column\_name*) FROM *table\_name*

WHERE *condition*;

## *AVG():*

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

SELECT AVG(*column\_name*) FROM *table\_name*

WHERE *condition*;

**BETWEEN, NOT BETWEEN**

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. BETWEEN SYNTAX:

SELECT *column\_name(s)*

FROM *table\_name*

WHERE *column\_name* BETWEEN *value1* AND *value2;*

# Activity 27: Selects all customers with a age between 20 and 30.

SELECT \*

FROM customers

where age BETWEEN 20 and 30

To display the products outside the range we use NOT BETWEEN. NOT BETWEEN SYNTAX:

SELECT *column1, column2…*

FROM *table\_name*

WHERE *column\_name* NOT BETWEEN value1 AND value2;