Engr. Shahid Ali Bhutta

university of engineering & technology taxila

DBMS LAB # 5

SQL Wildcards & Operators

**LAB # 04**

**SQL Wildcards & Operators**

# Lab Objective:

To understand SQL wild cards and the operators used in SQL.

# SQL Wildcards

SQL wildcards can be used when searching for data in a database.

SQL wildcards can substitute for one or more characters when searching for data in a database.

SQL wildcards must be used with the LIKE operator.

With SQL, the following wildcards can be used:

|  |  |
| --- | --- |
| **Wildcard** | **Description** |
| % | A substitute for zero or more characters |
| \_ | A substitute for exactly one character |
| [charlist] | Any single character in charlist |
| [^charlist] or  [!charlist] | Any single character not in charlist |

# SQL Wildcard Examples

We have the following "Persons" table:



# Using the % Wildcard

Now we want to select the persons living in a city that starts with "sa" from the "Persons" table.

We use the following SELECT statement:

SELECT \* FROM Persons

WHERE City LIKE 'sa%'

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |

Next, we want to select the persons living in a city that contains the pattern "nes" from the "Persons" table.

We use the following SELECT statement:

SELECT \* FROM Persons

WHERE City LIKE '%dne%'

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |

# Using the \_ Wildcard

Now we want to select the persons with a first name that starts with any character, followed by "la" from the "Persons" table.

We use the following SELECT statement:

SELECT \* FROM Persons

WHERE FirstName LIKE '\_ove'

The result-set will look like this:



# Using the [charlist] Wildcard

Now we want to select the persons with a last name that starts with "b" or "s" or "p" from the "Persons" table.

We use the following SELECT statement:

SELECT \* FROM Persons

WHERE LastName LIKE '[bsp]%’

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

Next, we want to select the persons with a last name that do not start with "b" or "s" or "p" from the "Persons" table.

We use the following SELECT statement:

SELECT \* FROM Persons

WHERE LastName NOT LIKE '[bsp]%'

The result-set will look like this:



# SQL IN Operator

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

# SQL IN Syntax

SELECT column\_name(s)

FROM table\_name

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

# IN Operator Example

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

Now we want to select the persons with a last name equal to "Hansen" or "Pettersen" from the table above.

We use the following SELECT statement:

SELECT \* FROM Persons

WHERE LastName IN ('Hansen','Pettersen')

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

# SQL BETWEENOperator

The BETWEEN operator is used in a WHERE clause to select a range of data between two values.

The BETWEEN operator selects a range of data between two values. The values can be numbers, text, or dates.

# SQL BETWEEN Syntax

SELECT column\_name(s)

FROM table\_name

WHERE column\_name

BETWEEN value1 AND value2

# BETWEEN Operator Example

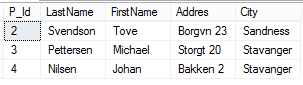
Now we want to select the persons with P\_ID betweek 2 and 4" from the persons table.

We use the following SELECT statement:

SELECT \* FROM Persons

WHERE P\_ID BETWEEN 2 AND 4;

The result-set will look like this:



**Note:** The BETWEEN operator is treated differently in different databases.

In some databases, persons with the P\_ID of 2 and 4 will not be listed, because the BETWEEN operator only selects fields that are between and excluding the test values).

In other databases, persons with id 2 and 4 will be listed, because the BETWEEN operator selects fields that are between and including the test values).

And in other databases, persons with id 2 will be listed, but id=4 will not be listed (like the example above), because the BETWEEN operator selects fields between the test values, including the first test value and excluding the last test value.

Therefore: Check how your database treats the BETWEEN operator.

# Example 2

To display the persons outside the range in the previous example, use NOT BETWEEN:

SELECT \* FROM Persons

WHERE LastName

NOT BETWEEN 'Hansen' AND 'Pettersen'

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

# 

# SQL UNION Operator

The UNION operator is used to combine the result-set of two or more SELECT statements.

Notice that each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

**SQL UNION Syntax**

SELECT column\_name(s) FROM table\_name1

UNION

SELECT column\_name(s) FROM table\_name2

**Note:** The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL.

**SQL UNION ALL Syntax**

SELECT column\_name(s) FROM table\_name1

UNION ALL

SELECT column\_name(s) FROM table\_name2

**PS:** The column names in the result-set of a UNION are always equal to the column names in the first SELECT statement in the UNION.

**SQL UNION Example**

Look at the following tables:

**"Employees\_Norway"**:

|  |  |
| --- | --- |
| **E\_ID** | **E\_Name** |
| 01 | Hansen, Ola |
| 02 | Svendson, Tove |
| 03 | Svendson, Stephen |
| 04 | Pettersen, Kari |

"Employees\_USA":

|  |  |
| --- | --- |
| **E\_ID** | **E\_Name** |
| 01 | Turner, Sally |
| 02 | Kent, Clark |
| 03 | Svendson, Stephen |
| 04 | Scott, Stephen |

Now we want to list **all the different** employees in Norway and USA.

We use the following SELECT statement:

SELECT E\_Name FROM Employees\_Norway

UNION

SELECT E\_Name FROM Employees\_USA

The result-set will look like this:

|  |
| --- |
| **E\_Name** |
| Hansen, Ola |
| Svendson, Tove |
| Svendson, Stephen |
| Pettersen, Kari |
| Turner, Sally |
| Kent, Clark |
| Scott, Stephen |

**Note:** This command cannot be used to list all employees in Norway and USA. In the example above we have two employees with equal names, and only one of them will be listed. The UNION command selects only distinct values.

**SQL UNION ALL Example**

Now we want to list **all** employees in Norway and USA:

SELECT E\_Name FROM Employees\_Norway

UNION ALL

SELECT E\_Name FROM Employees\_USA

Result

|  |
| --- |
| **E\_Name** |
| Hansen, Ola |
| Svendson, Tove |
| Svendson, Stephen |
| Pettersen, Kari |
| Turner, Sally |
| Kent, Clark |
| Svendson, Stephen |
| Scott, Stephen |

**Lab Tasks:**

## 1: Consider the following table “Customers”:

**Create the above table and do the following.**



* Write an SQL statement that selects all Customers with a Country starting with the letter

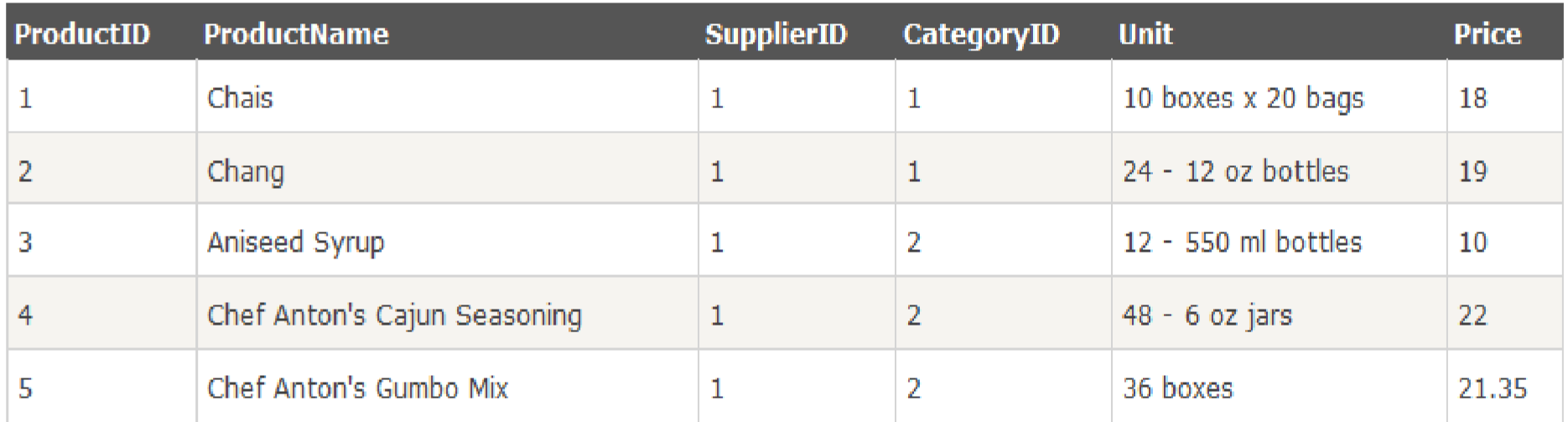
“s”.

* Write an SQL statement that selects all Customers with a Contact Name ending with the letter “s”.
* Write an SQL statement that selects all Customers with a City containing the pattern “ndo”.
* Write an SQL statement that selects all Customers with a City not containing the pattern “ndo”.
* Write an SQL statement that selects the two first Customers from table who belong to

“Germany” or “Sweden”.

* Write an SQL statement that selects all Customers with a City of "Paris" or "London" without using ‘OR’ operator.

## 2. Consider the following table “Products”



## Create the above table and do the following.

* Write an SQL statement that selects all products with a price from 10 to 20.
* Write an SQL statement that selects all products with a price from 20 to 30.
* Write an SQL statement that selects all products with a price from 10 to 22 but products with a **CategoryID**of 1,2, or 3 should not be displayed.
* Write an SQL statement that selects all products with a **ProductName** beginning with any of the letter not between 'C' and 'M'.