**SQL TOP Clause**

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

The TOP clause can be very useful on large tables with thousands of records. Returning a large number of records can impact on performance.

SELECT \*  
FROM Persons  
WHERE ROWNUM <=5

SELECT TOP 2 \* FROM Persons

Now we want to select only 50% of the records in the table above.

We use the following SELECT statement:

SELECT TOP 50 PERCENT \* FROM Persons

## The LIKE Operator

SELECT \* FROM Persons  
WHERE City LIKE 's%'

SELECT \* FROM Persons  
WHERE City LIKE '%s'

SELECT \* FROM Persons  
WHERE City LIKE '%tav%

SELECT \* FROM Persons  
WHERE City NOT LIKE '%tav%'

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

## SQL Wildcards

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

SQL wildcards must be used with the SQL 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:

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

## 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 '%nes%'

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 '\_la'

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |

Next, we want to select the persons with a last name that starts with "S", followed by any character, followed by "end", followed by any character, followed by "on" from the "Persons" table.

We use the following SELECT statement:

SELECT \* FROM Persons  
WHERE LastName LIKE 'S\_end\_on'

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |

## 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 LIKE '[!bsp]%'

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |

## The 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 BETWEEN Operator

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

## The BETWEEN Operator

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

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 alphabetically between "Hansen" and "Pettersen" from the table above.

We use the following SELECT statement:

SELECT \* FROM Persons  
WHERE LastName  
BETWEEN 'Hansen' AND 'Pettersen'

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |

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

In some databases, persons with the LastName of "Hansen" or "Pettersen" will not be listed, because the BETWEEN operator only selects fields that are between and excluding the test values.

In other databases, persons with the LastName of "Hansen" or "Pettersen" will be listed, because the BETWEEN operator selects fields that are between and including the test values.

And in other databases, persons with the LastName of "Hansen" will be listed, but "Pettersen" 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 Alias

## SQL Alias

You can give a table or a column another name by using an alias. This can be a good thing to do if you have very long or complex table names or column names.

An alias name could be anything, but usually it is short.

### SQL Alias Syntax for Tables

SELECT column\_name(s)  
FROM table\_name  
AS alias\_name

### SQL Alias Syntax for Columns

SELECT column\_name AS alias\_name  
FROM table\_name

## Alias Example

Assume we have a table called "Persons" and another table called "Product\_Orders". We will give the table aliases of "p" and "po" respectively.

Now we want to list all the orders that "Ola Hansen" is responsible for.

We use the following SELECT statement:

SELECT po.OrderID, p.LastName, p.FirstName  
FROM Persons AS p,  
Product\_Orders AS po  
WHERE p.LastName='Hansen' AND p.FirstName='Ola'

The same SELECT statement without aliases:

SELECT Product\_Orders.OrderID, Persons.LastName, Persons.FirstName  
FROM Persons,  
Product\_Orders  
WHERE Persons.LastName='Hansen' AND Persons.FirstName='Ola'

As you'll see from the two SELECT statements above; aliases can make queries easier both to write and to read.

## The GROUP BY Statement

# The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.

## SQL GROUP BY Example

We have the following "Orders" table:

|  |  |  |  |
| --- | --- | --- | --- |
| **O\_Id** | **OrderDate** | **OrderPrice** | **Customer** |
| 1 | 2008/11/12 | 1000 | Hansen |
| 2 | 2008/10/23 | 1600 | Nilsen |
| 3 | 2008/09/02 | 700 | Hansen |
| 4 | 2008/09/03 | 300 | Hansen |
| 5 | 2008/08/30 | 2000 | Jensen |
| 6 | 2008/10/04 | 100 | Nilsen |

Now we want to find the total sum (total order) of each customer.

We will have to use the GROUP BY statement to group the customers.

We use the following SQL statement:

SELECT Customer,SUM(OrderPrice) FROM Orders  
GROUP BY Customer

The result-set will look like this:

|  |  |
| --- | --- |
| **Customer** | **SUM(OrderPrice)** |
| Hansen | 2000 |
| Nilsen | 1700 |
| Jensen | 2000 |

SELECT Customer,SUM(OrderPrice) FROM Orders

The result-set will look like this:

|  |  |
| --- | --- |
| **Customer** | **SUM(OrderPrice)** |
| Hansen | 5700 |
| Nilsen | 5700 |
| Hansen | 5700 |
| Hansen | 5700 |
| Jensen | 5700 |
| Nilsen | 5700 |

The result-set above is not what we wanted.

## GROUP BY More Than One Column

# SELECT Customer,OrderDate,SUM(OrderPrice) FROM Orders GROUP BY Customer,OrderDate

# SQL HAVING Clause

## The HAVING Clause

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

# SELECT Customer,SUM(OrderPrice) FROM Orders

# GROUP BY Customer HAVING SUM(OrderPrice)<2000

# SELECT Customer,SUM(OrderPrice) FROM Orders WHERE Customer='Hansen' OR Customer='Jensen' GROUP BY Customer HAVING SUM(OrderPrice)>1500

# Functions:

# 1.avg()

# 2.sum()

# 3.min()

# 4.max()

# 5.count()

# 6.len()

# 7.ucase

# 8.lcase

# 9.mid

# The MID() function is used to extract characters from a text field.

# SELECT MID(City,1,4) as SmallCity FROM Persons

# 10.now()

## The NOW() Function

The NOW() function returns the current system date and time.

SELECT ProductName, UnitPrice, Now() as PerDate FROM Products

11.