SQL

SQL Functions

SQL has many built-in functions[i] for performing calculations on data.

SQL Aggregate Functions

SQL aggregate functions return a single value, calculated from values in a column.

Useful aggregate functions:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

SQL Scalar functions

SQL scalar functions return a single value, based on the input value.

Useful scalar functions:

- UCASE() - Converts a field to upper case
- LCASE() - Converts a field to lower case
- MID() - Extract characters from a text field
- LEN() - Returns the length of a text field
- ROUND() - Rounds a numeric field to the number of decimals specified
- NOW() - Returns the current system date and time
- FORMAT() - Formats how a field is to be displayed

SQL Aggregate Functions

The AVG() Function

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

SQL AVG() Syntax

```
SELECT AVG(column_name) FROM table_name
```

Es:

SELECT AVG(OrderPrice) **AS** OrderAverage FROM Orders

SELECT Customer FROM Orders
WHERE OrderPrice>(SELECT AVG(OrderPrice) FROM Orders)

SQL

SQL COUNT() Function

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

SQL COUNT(column_name) Syntax

The COUNT(column_name) function returns the number of values (NULL values will not be counted) of the specified column:

```
SELECT COUNT(column_name) FROM table_name
```

SQL COUNT(*) Syntax

The COUNT(*) function returns the number of records in a table:

```
SELECT COUNT(*) FROM table_name
```

SQL COUNT(DISTINCT column_name) Syntax

The COUNT(DISTINCT column_name) function returns the number of distinct values of the specified column:

```
SELECT COUNT(DISTINCT column_name) FROM table_name
```

Note: COUNT(DISTINCT) works with ORACLE and Microsoft SQL Server, but not with Microsoft Access.

Es:

SELECT COUNT(Customer) AS CustomerNilsen FROM Orders
WHERE Customer='Nilsen'

SELECT COUNT(*) AS NumberOfOrders FROM Orders

SQL FIRST() Function

The FIRST() Function

The FIRST() function returns the first value of the selected column.

SQL FIRST() Syntax

```
SELECT FIRST(column_name) FROM table_name
```

Es:

```
SELECT FIRST(OrderPrice) AS FirstOrderPrice FROM Orders
```

Tip: Workaround if FIRST() function is not supported:

```
SELECT OrderPrice FROM Orders ORDER BY O_Id LIMIT 1
```

SQL

SQL LAST() Function

The LAST() Function

The LAST() function returns the last value of the selected column.

SQL LAST() Syntax

```
SELECT LAST(column_name) FROM table_name
```

Es:

```
SELECT LAST(OrderPrice) AS LastOrderPrice FROM Orders
```

Tip: Workaround if LAST() function is not supported:

```
SELECT OrderPrice FROM Orders ORDER BY O_Id DESC LIMIT 1
```

SQL MAX() Function

The MAX() Function

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

SQL MAX() Syntax

```
SELECT MAX(column_name) FROM table_name
```

Es:

```
SELECT MAX(OrderPrice) AS LargestOrderPrice FROM Orders
```

SQL MIN() Function

The MIN() Function

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

SQL MIN() Syntax

```
SELECT MIN(column_name) FROM table_name
```

Es:

```
SELECT MIN(OrderPrice) AS SmallestOrderPrice FROM Orders
```

SQL

SQL SUM() Function

The SUM() Function

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

SQL SUM() Syntax

```
SELECT SUM(column_name) FROM table_name
```

Es:
SELECT SUM(OrderPrice) AS OrderTotal FROM Orders

SQL GROUP BY Statement

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 Syntax

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
```

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

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

SQL

SQL HAVING Clause

The HAVING Clause

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

SQL HAVING Syntax

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```

Es:
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

SQL Scalar functions

SQL FORMAT() Function

The FORMAT() function is used to format how a field is to be displayed.

SQL FORMAT() Syntax

```
SELECT FORMAT(column_name,format) FROM table_name
```

Parameter	Description
column_name	Required. The field to be formatted.
format	Required. Specifies the format.

Example

We have the following "Products" table:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56

Now we want to display the products and prices per today's date (with today's date displayed in the following format "YYYY-MM-DD"). We use the following SELECT statement:

```
SELECT ProductName, UnitPrice, FORMAT(Now(),'YYYY-MM-DD') as PerDate
FROM Products
```

The result-set will look like this:

ProductName	UnitPrice	PerDate
Jarlsberg	10.45	2008-10-07
Mascarpone	32.56	2008-10-07

SQL

SQL NOW() Function

The NOW() Function

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

SQL NOW() Syntax

```
SELECT NOW() FROM table_name
```

Es:

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

ProductName	UnitPrice	PerDate
Jarlsberg	10.45	10/7/2008 11:25:02 AM
Mascarpone	32.56	10/7/2008 11:25:02 AM

SQL DML and DDL

SQL can be divided into two parts: The Data Manipulation Language (DML) and the Data Definition Language (DDL).

The query and update commands form the **DML** part of SQL:

- **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

The DDL part of SQL permits database tables to be created or deleted. It also defines indexes (keys), specifies links between tables, and imposes constraints between tables. The most important **DDL** statements in SQL are:

- **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

CREATE DATABASE	CREATE DATABASE database_name
CREATE TABLE	CREATE TABLE table_name (column_name1 data_type, column_name2 data_type, column_name2 data_type, ...)
CREATE INDEX	CREATE INDEX index_name ON table_name (column_name) or CREATE UNIQUE INDEX index_name ON table_name (column_name)
CREATE VIEW	CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition
DELETE	DELETE FROM table_name WHERE some_column=some_value or DELETE FROM table_name (**Note:** Deletes the entire table!!) DELETE * FROM table_name (**Note:** Deletes the entire table!!)
DROP DATABASE	DROP DATABASE database_name
DROP INDEX	DROP INDEX table_name.index_name (SQL Server) DROP INDEX index_name ON table_name (MS Access) DROP INDEX index_name (DB2/Oracle) ALTER TABLE table_name DROP INDEX index_name (MySQL)
DROP TABLE	DROP TABLE table_name

[i] Nell'uso di phpMuAdmin si dispone di interfaccia per gestire DBMS di tipo **MariaDB**
(esempi d'uso da URL https://mariadb.com/kb/en/library/built-in-functions/)