

SQL FUNCTIONS

Assembled by: Feras J Abualrub

Email address: frub84@hotmail.com

• SQL Aggregate Functions.....	3
• SQL Scalar functions	3
• The AVG() Function.....	3
• SQL COUNT(column_name) Syntax.....	5
• SQL COUNT(*) Syntax	5
• SQL COUNT(DISTINCT column_name) Syntax.....	5
• The FIRST() Function.....	6
• SQL FIRST() Syntax	7
• The LAST() Function	7
• SQL LAST() Syntax	7
• The MAX() Function	8
• SQL MAX() Syntax.....	8
• The MIN() Function.....	9
• SQL MIN() Syntax	9
• The SUM() Function.....	10
• SQL SUM() Syntax.....	10
• The GROUP BY Statement	10
• SQL GROUP BY Syntax.....	11
• GROUP BY More Than One Column	12
• The HAVING Clause.....	12
• SQL HAVING Syntax	12
• The UCASE() Function	13
• SQL UCASE() Syntax	13
• The LCASE() Function.....	14
• SQL LCASE() Syntax.....	14
• The MID() Function.....	15
• SQL MID() Syntax	15
• The LEN() Function.....	16
• SQL LEN() Syntax	16
• The ROUND() Function	17
• SQL ROUND() Syntax	17
• The NOW() Function.....	18
• SQL NOW() Syntax.....	18
• The FORMAT() Function.....	18
• SQL FORMAT() Syntax.....	18
• SQL Quick Reference	19
• SQL Hosting	22

SQL has many built-in functions 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

Tip: The aggregate functions and the scalar functions will be explained in details in the next chapters.

• The AVG() Function

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

SQL AVG() Syntax

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

SQL AVG() 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 average value of the "OrderPrice" fields.

We use the following SQL statement:

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

The result-set will look like this:

OrderAverage
950

Now we want to find the customers that have an OrderPrice value higher then the average OrderPrice value.

We use the following SQL statement:

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

The result-set will look like this:

Customer
Hansen
Nilsen
Jensen

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.

SQL COUNT(column_name) 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 count the number of orders from "Customer Nilsen".

We use the following SQL statement:

```
SELECT COUNT(Customer) AS CustomerNilsen FROM Orders
```

WHERE Customer='Nilsen'

The result of the SQL statement above will be 2, because the customer Nilsen has made 2 orders in total:

CustomerNilsen
2

SQL COUNT(*) Example

If we omit the WHERE clause, like this:

```
SELECT COUNT(*) AS NumberOfOrders FROM Orders
```

The result-set will look like this:

NumberOfOrders
6

which is the total number of rows in the table.

SQL COUNT(DISTINCT column_name) Example

Now we want to count the number of unique customers in the "Orders" table.

We use the following SQL statement:

```
SELECT COUNT(DISTINCT Customer) AS NumberOfCustomers FROM Orders
```

The result-set will look like this:

NumberOfCustomers
3

which is the number of unique customers (Hansen, Nilsen, and Jensen) in the "Orders" table.

- **The FIRST() Function**

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

- **SQL FIRST() Syntax**

SELECT FIRST(column_name) FROM table_name

SQL FIRST() 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 first value of the "OrderPrice" column.

We use the following SQL statement:

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

The result-set will look like this:

FirstOrderPrice
1000

- **The LAST() Function**

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

- **SQL LAST() Syntax**

SELECT LAST(column_name) FROM table_name

SQL LAST() 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 last value of the "OrderPrice" column.

We use the following SQL statement:

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

The result-set will look like this:

LastOrderPrice
100

- **The MAX() Function**

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

- **SQL MAX() Syntax**

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

SQL MAX() 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 largest value of the "OrderPrice" column.

We use the following SQL statement:

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

The result-set will look like this:

LargestOrderPrice
2000

- **The MIN() Function**

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

- **SQL MIN() Syntax**

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

SQL MIN() 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 smallest value of the "OrderPrice" column.

We use the following SQL statement:

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

The result-set will look like this:

SmallestOrderPrice
100

- **The SUM() Function**

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

- **SQL SUM() Syntax**

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

SQL SUM() 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 sum of all "OrderPrice" fields".

We use the following SQL statement:

```
SELECT SUM(OrderPrice) AS OrderTotal FROM Orders
```

The result-set will look like this:

OrderTotal
5700

Aggregate functions often need an added 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
```

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

Nice! Isn't it? :)

Let's see what happens if we omit the GROUP BY statement:

```
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.

Explanation of why the above SELECT statement cannot be used: The SELECT statement above has two columns specified (Customer and SUM(OrderPrice)). The "SUM(OrderPrice)" returns a single value (that is the total sum of the "OrderPrice" column), while "Customer" returns 6 values (one value for each row in the "Orders" table). This will therefore not give us the correct result. However, you have seen that the GROUP BY statement solves this problem.

• GROUP BY More Than One Column

We can also use the GROUP BY statement on more than one column, like this:

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

• 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
```

SQL HAVING 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 if any of the customers have a total order of less than 2000.

We use the following SQL statement:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
HAVING SUM(OrderPrice)<2000
```

The result-set will look like this:

Customer	SUM(OrderPrice)
Nilsen	1700

Now we want to find if the customers "Hansen" or "Jensen" have a total order of more than 1500.

We add an ordinary WHERE clause to the SQL statement:

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

The result-set will look like this:

Customer	SUM(OrderPrice)
Hansen	2000
Jensen	2000

• The UCASE() Function

The UCASE() function converts the value of a field to uppercase.

• SQL UCASE() Syntax

```
SELECT UCASE(column_name) FROM table_name
```

SQL UCASE() Example

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

Now we want to select the content of the "LastName" and "FirstName" columns above, and convert the "LastName" column to uppercase.

We use the following SELECT statement:

```
SELECT UCASE(LastName) as LastName,FirstName FROM Persons
```

The result-set will look like this:

LastName	FirstName
HANSEN	Ola
SVENDSON	Tove
PETTERSEN	Kari

• The LCASE() Function

The LCASE() function converts the value of a field to lowercase.

• SQL LCASE() Syntax

```
SELECT LCASE(column_name) FROM table_name
```

SQL LCASE() Example

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

Now we want to select the content of the "LastName" and "FirstName" columns above, and convert the "LastName" column to lowercase.

We use the following SELECT statement:

```
SELECT LCASE(LastName) as LastName,FirstName FROM Persons
```

The result-set will look like this:

LastName	FirstName
hansen	Ola
svendson	Tove
pettersen	Kari

• The MID() Function

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

• SQL MID() Syntax

```
SELECT MID(column_name,start[,length]) FROM table_name
```

Parameter	Description
column_name	Required. The field to extract characters from
start	Required. Specifies the starting position (starts at 1)
length	Optional. The number of characters to return. If omitted, the MID() function returns the rest of the text

SQL MID() Example

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

Now we want to extract the first four characters of the "City" column above.

We use the following SELECT statement:

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

The result-set will look like this:

SmallCity
Sand
Sand
Stav

- **The LEN() Function**

The LEN() function returns the length of the value in a text field.

- **SQL LEN() Syntax**

```
SELECT LEN(column_name) FROM table_name
```

SQL LEN() Example

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

Now we want to select the length of the values in the "Address" column above.

We use the following SELECT statement:

```
SELECT LEN(Address) as LengthOfAddress FROM Persons
```

The result-set will look like this:

LengthOfAddress
12

9
9

• The ROUND() Function

The ROUND() function is used to round a numeric field to the number of decimals specified.

• SQL ROUND() Syntax

SELECT ROUND(column_name,decimals) FROM table_name

Parameter	Description
column_name	Required. The field to round.
decimals	Required. Specifies the number of decimals to be returned.

SQL ROUND() Example

We have the following "Products" table:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56
3	Gorgonzola	1000 g	15.67

Now we want to display the product name and the price rounded to the nearest integer.

We use the following SELECT statement:

SELECT ProductName, ROUND(UnitPrice,0) as UnitPrice FROM Products

The result-set will look like this:

ProductName	UnitPrice
Jarlsberg	10
Mascarpone	33
Gorgonzola	16

- **The NOW() Function**

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

- **SQL NOW() Syntax**

SELECT NOW() FROM table_name

SQL NOW() Example

We have the following "Products" table:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56
3	Gorgonzola	1000 g	15.67

Now we want to display the products and prices per today's date.

We use the following SELECT statement:

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

The result-set will look like this:

ProductName	UnitPrice	PerDate
Jarlsberg	10.45	10/7/2008 11:25:02 AM
Mascarpone	32.56	10/7/2008 11:25:02 AM
Gorgonzola	15.67	10/7/2008 11:25:02 AM

- **The 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.

SQL 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
3	Gorgonzola	1000 g	15.67

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
Gorgonzola	15.67	2008-10-07

- **SQL Quick Reference**

SQL Statement	Syntax
AND / OR	SELECT column_name(s) FROM table_name WHERE condition AND OR condition
ALTER TABLE	ALTER TABLE table_name ADD column_name datatype or

	ALTER TABLE table_name DROP COLUMN column_name
AS (alias)	SELECT column_name AS column_alias FROM table_name or SELECT column_name FROM table_name AS table_alias
BETWEEN	SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2
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
GROUP BY	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name
HAVING	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
IN	SELECT column_name(s) FROM table_name WHERE column_name IN (value1,value2,...)
INSERT INTO	INSERT INTO table_name VALUES (value1, value2, value3,...) <i>or</i> INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)
INNER JOIN	SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name
LEFT JOIN	SELECT column_name(s) FROM table_name1 LEFT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
RIGHT JOIN	SELECT column_name(s) FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
FULL JOIN	SELECT column_name(s) FROM table_name1 FULL JOIN table_name2 ON table_name1.column_name=table_name2.column_name
LIKE	SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern
ORDER BY	SELECT column_name(s)

	FROM table_name ORDER BY column_name [ASC DESC]
SELECT	SELECT column_name(s) FROM table_name
SELECT *	SELECT * FROM table_name
SELECT DISTINCT	SELECT DISTINCT column_name(s) FROM table_name
SELECT INTO	SELECT * INTO new_table_name [IN externaldatabase] FROM old_table_name <i>or</i> SELECT column_name(s) INTO new_table_name [IN externaldatabase] FROM old_table_name
SELECT TOP	SELECT TOP number percent column_name(s) FROM table_name
TRUNCATE TABLE	TRUNCATE TABLE table_name
UNION	SELECT column_name(s) FROM table_name1 UNION SELECT column_name(s) FROM table_name2
UNION ALL	SELECT column_name(s) FROM table_name1 UNION ALL SELECT column_name(s) FROM table_name2
UPDATE	UPDATE table_name SET column1=value, column2=value,... WHERE some_column=some_value
WHERE	SELECT column_name(s) FROM table_name WHERE column_name operator value

- SQL Hosting

If you want your web site to be able to store and display data from a database, your web server should have access to a database system that uses the SQL language.

If your web server will be hosted by an Internet Service Provider (ISP), you will have to look for SQL hosting plans.

The most common SQL hosting databases are MySQL, MS SQL Server, and MS Access.

You can have SQL databases on both Windows and Linux/UNIX operating systems.

Below is an overview of which database system that runs on which OS.

MS SQL Server

Runs only on Windows OS.

MySQL

Runs on both Windows and Linux/UNIX operating systems.

MS Access (recommended only for small websites)

Runs only on Windows OS.