



Database Management Systems

Associate Professor Dr. Raed Ibraheem Hamed

University of Human Development, College of Science and Technology
Departments of IT and Computer Science



2015 – 2016

SQL NULL Functions

SQL ISNULL(), IFNULL() and COALESCE() Functions

Look at the following "Products" table:

P_Id	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	
3	Gorgonzola	15.67	9	20

SQL NULL Functions

Suppose that the "UnitsOnOrder" column is optional, and may contain NULL values.

We have the following SELECT statement:

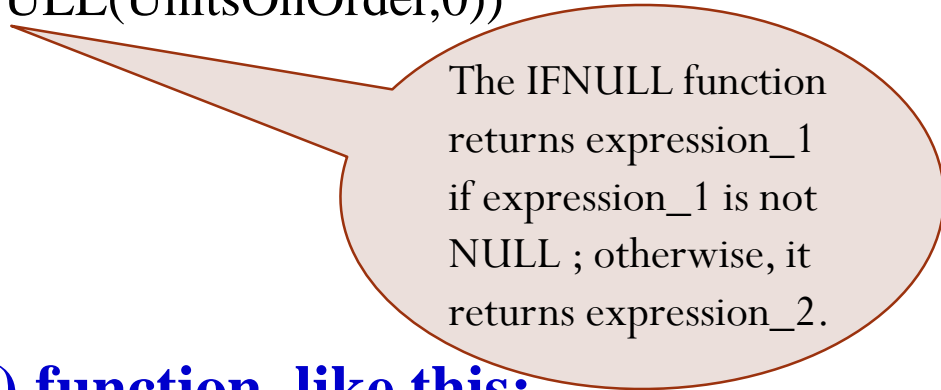
```
SELECT ProductName, UnitPrice*(UnitsInStock+UnitsOnOrder)
FROM Products
```

In the example above, if any of the "UnitsOnOrder" values are NULL, the result is NULL.

The IFNULL(), and COALESCE() functions can also be used to achieve the same result.

In MySQL we can use the IFNULL() function, like this:

```
SELECT ProductName, UnitPrice*(IFNULL(UnitsOnOrder,0))  
FROM Products
```



The IFNULL function returns expression_1 if expression_1 is not NULL ; otherwise, it returns expression_2.

or we can use the COALESCE() function, like this:

```
SELECT ProductName, UnitPrice*(COALESCE(UnitsOnOrder,0))  
FROM Products
```

	contactName	bizphone	homephone
▶	John Doe	(541) 754-3009	NULL
	Cindy Smith	NULL	(541) 754-3110
	Sue Greenspan	(541) 754-3010	(541) 754-3011
	Lily Bush	NULL	(541) 754-3111

The IFNULL function returns the home phone if the business phone is NULL.

SELECT

contactname, phone*(IFNULL(bizphone, homephone))
FROM contacts;

	contactname	phone
▶	John Doe	(541) 754-3009
	Cindy Smith	(541) 754-3110
	Sue Greenspan	(541) 754-3010
	Lily Bush	(541) 754-3111

SQL Functions

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:

- 1) AVG() - Returns the average value
- 2) COUNT() - Returns the number of rows
- 3) FIRST() - Returns the first value
- 4) LAST() - Returns the last value
- 5) MAX() - Returns the largest value
- 6) MIN() - Returns the smallest value
- 7) SUM() - Returns the sum

SQL Scalar functions

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

Useful scalar functions:

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

SQL AVG() Syntax

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	21.35
5	Chef Anton's Gumbo Mix	2	2	36 boxes	25

SQL AVG() Example

The following SQL statement gets the average value of the "Price" column from the "Products" table:

Example

```
SELECT AVG(Price) AS PriceAverage FROM Products;
```

Result:

of Records: 1

PriceAverage

28.866

SQL AVG() Syntax

The following SQL statement selects the "ProductName" and "Price" records that have an above average price:

Example

```
SELECT ProductName, Price FROM Products  
WHERE Price > (SELECT AVG(Price) FROM Products);
```

ProductName	Price
Uncle Bob's Organic Dried Pears	30
Northwoods Cranberry Sauce	40
Mishi Kobe Niku	97
Ikura	31
Queso Manchego La Pastora	38

SQL COUNT() Function

Below is a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10265	7	2	1996-07-25	1
10266	87	3	1996-07-26	3
10267	25	4	1996-07-29	1

SQL COUNT() Function

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;

The following SQL statement counts the number of orders from "CustomerID"=7 from the "Orders" table:

Example

```
SELECT COUNT(CustomerID) AS ITandCS FROM Orders  
WHERE CustomerID=7;
```

Result: ITandCS

4

SQL COUNT(*) Example

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

Example

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

Result: NumberOfOrders
196

SQL COUNT(DISTINCT column_name) Syntax

SQL COUNT(DISTINCT column_name) Example

The following SQL statement counts the number of unique customers in the "Orders" table:

Example

```
SELECT COUNT(DISTINCT CustomerID) AS ITandCS FROM Orders;
```

Result: ITandCS

74

SQL FIRST() Function

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

SQL FIRST() Syntax

```
SELECT column_name FROM table_name  
ORDER BY column_name ASC  
LIMIT 1;
```

Example

```
SELECT CustomerName FROM Customers  
ORDER BY CustomerID ASC  
LIMIT 1;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico



CustomerName
Alfreds Futterkiste

The LAST() Function

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

SQL LAST() Syntax

```
SELECT column_name FROM table_name
ORDER BY column_name DESC
LIMIT 1;
```

"Customers" table:

89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

SQL LAST() Example

```
SELECT CustomerName FROM Customers
ORDER BY CustomerID DESC
LIMIT 1;
```



CustomerName
Wolski

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

```
SELECT MAX(Price) AS HighestPrice FROM Products;
```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10



HighestPrice
19

The MIN() Function

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

SQL MIN() Syntax

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

```
SELECT MIN(Price) AS  
least, MAX(Price) AS max  
FROM Products;
```

SQL MIN() Example

```
SELECT MIN(Price) AS SmallestOrderPrice FROM Products;
```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10



SmallestOrderPrice
10

The SUM() Function

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

SQL SUM() Syntax

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

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

TotalItemsOrdered

76

SQL SUM() Example

```
SELECT SUM(Quantity) AS TotalItemsOrdered FROM OrderDetails;
```

SQL Aliases

SQL aliases are used to give a database **table**, or a **column** in a table, a temporary name.

Basically aliases are created to make column names more readable.

- **SQL Alias Syntax for Columns:-**

```
SELECT column_name AS alias_name  
FROM table_name;
```

- **SQL Alias Syntax for Tables:-**

```
SELECT column_name(s)  
FROM table_name AS alias_name;
```

Aliases Example

"Customers" table:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

Example

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]  
FROM Customers;
```

Customer	Contact Person
Alfreds Futterkiste	Maria Anders
Ana Trujillo Emparedados y helados	Ana Trujillo
Antonio Moreno Taquería	Antonio Moreno
Around the Horn	Thomas Hardy

Alias Example for Tables

```
SELECT c.CustomerName  
FROM Customers AS c;
```

The following SQL statement selects all the customer name from Customer. We use the "Customers" table, and give it the table alias of "c" .

CustomerName
Alfreds Futterkiste
Ana Trujillo Emparedados y helados
Antonio Moreno Taquería
Around the Horn
Berglunds snabbköp

MySQL **CONCAT** function

MySQL **CONCAT** function is used to concatenate two or more strings to form a single string. Try out the following example:

To understand **CONCAT** function in more detail, consider an **employee_tbl** table, which is having the following records:

```
mysql> SELECT * FROM employee_tbl;
```

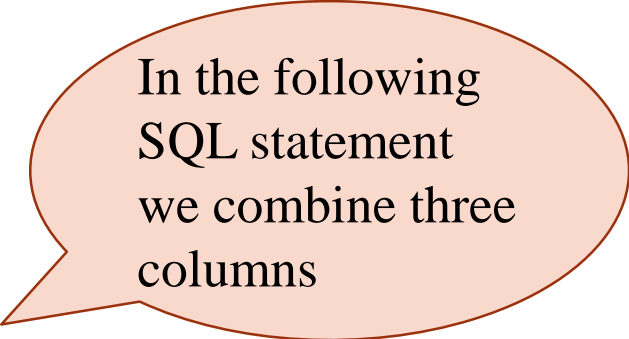
id	name	work_date	daily_typing_pages
1	John	2007-01-24	250
2	Ram	2007-05-27	220
3	Jack	2007-05-06	170
3	Jack	2007-04-06	100
4	Jill	2007-04-06	220
5	Zara	2007-06-06	300
5	Zara	2007-02-06	350

CONCAT Example

Now, suppose based on the above table you want to concatenate all the **names** , **employee ID and work_date**, then you can do it using the following command:

```
SELECT CONCAT(id, name, work_date)
FROM employee_tb1;
```

```
+-----+
| CONCAT(id, name, work_date) |
+-----+
| 1John2007-01-24             |
| 2Ram2007-05-27              |
| 3Jack2007-05-06             |
| 3Jack2007-04-06             |
| 4Jill2007-04-06             |
| 5Zara2007-06-06             |
| 5Zara2007-02-06            |
+-----+
```



In the following SQL statement we combine three columns

The SQL SELECT LIMIT Clause

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

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

MySQL Syntax

```
SELECT column_name(s)  
FROM table_name  
LIMIT number,
```

```
SELECT *  
FROM Persons  
LIMIT 4;
```


The SQL SELECT LIMIT Example

Number of Records: 4

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2
10251	84	3	1996-07-08	1

THANK
YOU

