Module 2: Structured Query Language and Transaction Management

Unit 1: SQL Functions

Page

1.0 Introduction

A *function* is a special type of command word in the SQL command set. In effect, functions are one-word commands that return a single value. The value of a function can be determined by input parameters, as with a function that averages a list of database values. But many functions do not use any type of input parameter, such as the function that returns the current system time, *CURRENT_TIME*.

The SQL supports a number of useful functions. This unit covers those functions, providing detailed descriptions and examples.

2.0 Objectives

By the end of this unit, you should be able to:

o. Perform arithmetic operations such as: finding average of column, finding sum of a column, finding the number of records in a table; finding the minimum and maximum values in a column.
p. Convert a field to upper or lower case
q. Extract characters from a text field
r. Format how a column or field should be displayed.

3.1 SQL Aggregate Functions

SQL aggregate functions return a single value, calculated from values in a column. In this section, we discuss the following SQL aggregate commands. By the end of this section, you will learn the basics of retrieving data from the database using SQL.

Useful aggregate functions are:

i. AVG() - Returns the average value
ii. COUNT() - Returns the number of rows
iii. FIRST() - Returns the first value iv. LAST() - Returns the last value
v. MAX() - Returns the largest value
vi. MIN() - Returns the smallest value
vii. SUM() - Returns the sum

3.1.1 The AVG Function

The AVG function returns the average value of a numeric column. AVG Syntax is:

SELECT AVG(column_name) FROM table_name

Example: Let us consider the following "Orders" table:

OrderId	OrderDate	Price	Customername
11	2008/11/12	1000	Henry Bank
21	2008/10/23	1600	Niyi Alade
31	2008/09/02	700	Henry Bank
41	2008/09/03	300	Henry Bank
51	2008/08/30	2000	James Adeola
61	2008/10/04	100	Niyi Alade

Question: Let us find the average value of the Price column.

Answer: We use the following SQL statement:

SELECT AVG(Price) AS AveragePrice FROM Orders

The result-set will look like this:

AveragePrice
950

We may decide to find the customers that have order Price value higher than the average Price value.

We use the following SQL statement:

SELECT Customername FROM Orders
WHERE Price>(SELECT AVG(Price) FROM Orders)

The result-set will look like this:

Customername
Henry Bank
Niyi Alade
James Adeola

3.1.2 The COUNT function

The COUNT function returns the number of rows that matches specified criteria. Note that null values will not be counted. In the section, we shall consider the following:

- SQL COUNT(column_name) Syntax
- SQL COUNT(*) Syntax
- SQL COUNT(DISTINCT column_name) Syntax

a. 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

Example: Let us consider our order table in section 3.11 again.

Now we want to count the number of orders from "Customer Niyi Alade".

We use the following SQL statement:

```
SELECT COUNT(Customer) AS NiyiAlade FROM Orders
WHERE Customer='Niyi Alade'
```

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

NiyiAlade
2

b. SQL COUNT(*) Syntax

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

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

Example: Let us consider our order table again. Now we want to find the number of records in the order table.

We use the following SQL statement:

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

The result-set will look like this

NumberOfOrders
6

This is the total number of rows in the order table.

c. 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
```

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 TotalCustomers FROM Orders

The result-set will look like this:

TotalCustomers
3

3.1.3 The FIRST Function

The FIRST function returns the first value of the selected column. The SQL Syntax is:

SELECT FIRST(column_name) FROM table_name

Example: We will still make use of our orders table in section 3.11

Now we want to find the first value of the "Price" column.

We use the following SQL statement:

SELECT FIRST(Price) AS FirstPrice FROM Orders

The result-set will look like this:

FirstOrderPrice
1000

3.1.4 The LAST Function

The LAST function returns the last value of the selected column. The syntax is:

SELECT LAST(column_name) FROM table_name

Example: We have the "Orders table in section 3.11

Now we want to find the last value of the Price column.

We will make use of the following SQL statement:

SELECT LAST(Price) AS LastOrderPrice FROM Orders

The result-set will look like this:

LastOrderPrice
100

3.1.5 The MAX Function

The MAX function returns the largest value of the selected column. The SQL MAX Syntax is:

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

Example: Let us consider Orders table again:

This time around we want to find the largest value of the Price column.

We shall make use of the following SQL statement:

SELECT MAX(Price) AS LargestPrice FROM Orders

The result-set will look like this:

LargestPrice
2000

3.1.6 The MIN Function

The MIN function returns the smallest value of the selected column. The SQL MIN Syntax is as follows:

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

Example from our Orders table: let us find the smallest value of the Price column.

We use the following SQL statement:

SELECT MIN(Price) AS SmallestPrice FROM Orders

The result-set will look like this:

SmallestPrice
100

3.1.7 SUM Function

The SUM function is used to calculate the total for a column. The syntax is,

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

Example from Orders table: we want to find the sum of all Price field.

We use the following SQL statement:

SELECT SUM(Price) AS OrderTotal FROM Orders

The result-set will look like this:

OrderTotal
5700

3.1.8 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. The syntax is:

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

Example: let us consider the Orders table again:

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 Customername, SUM(Price) FROM Orders
GROUP BY Customername

The result-set will look like this:

Customername	SUM(Price)
Henry Bank	2000
Niyi Alade	1700
James Adeola	2000

Let us see what will happen if we omit the GROUP BY statement:

SELECT Customername, SUM(Price) FROM Orders

The result-set will look like this:

Customername	SUM(Price)
Henry Bank	5700
Niyi Alade	5700
Henry Bank	5700
Henry Bank	5700
James Adeola	5700
Niyi Alade	5700

The result-set above is not what we wanted.

3.1.9 The HAVING Clause

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

```
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
```

Example: 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 Customername, SUM(Price) FROM Orders
GROUP BY Customer

HAVING SUM(Price)<2000

The result-set will look like this:

Customername	SUM(Price)
Niyi Alade	1700

Now we want to find if the customers "Henry Bank" or "James Adeola" have a total order of more than 1500.

We add an ordinary WHERE clause to the SQL statement:

```
SELECT Customername, SUM(Price) FROM Orders
WHERE Customername='Henry Bank' OR Customername='James Adeola'
GROUP BY Customername
HAVING SUM(Price)>1500
```

The result-set will look like this:

Customername	SUM(Price)
Henry Bank	2000
James Adeola	2000

Activity A

1. Write out the SQL syntax for the following functions
   i. A
   VG()
   ii. C
   OUNT()
   iii.
   FIRST()
   iv.
   LAST()
   v.
   MAX()
   vi.
   MIN()
vii. SUM()

3.2 SQL Scalar functions

SQL scalar functions return a single value, based on the input value. In this section, we discuss the following SQL scalar commands. By the end of this section, you will learn the basics of manipulating data from the database using SQL.

Some useful scalar functions are:

- 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

We shall make use of the following Persons table throughout this section

PersonId	Surname	Firstname	Address	City	
1	Henry	Bank	15 Allen Avenue	Lagos 2	
Ebuka	Tunji	23 Wuse Zone 4	Abuja		
3	Peter	Kasim	78 Baba street	Kaduna	

3.2.1 The UCASE Function

The UCASE function is used to convert the value of a column to uppercase. The syntax is

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

Example: We have a Persons table in section 3.2, now we want to select the content of the Surname and FirstName columns, and convert the Surname column to uppercase.

We use the following SELECT statement:

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

The result-set will look like this:

Surname	FirstName
HENRY	Bank
EBUKA	Tunji
PETER	Kasim

3.2.2 The LCASE Function

The LCASE() function converts the value of a column to lowercase. The syntax is:

SELECT LCASE(column_name) FROM table_name

Example: Let us select the content of the Surname and FirstName columns from our Persons table, and convert the Surname column to lowercase.

We use the following SELECT statement:

SELECT LCASE(Surname) as Surname, FirstName FROM Persons

The result-set will look like this:

Surname	FirstName
henry	Bank
ebuka	Tunji
peter	Kasim

3.2.3 The MID Function

The MID function is used to extract characters from a text column. The syntax is:

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

Parameters:	Description
column_name:	Required. The column 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

Example: Let us extract the first four characters of the "City" column from Persons table.

We use the following SELECT statement:

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

The result-set will look like this:

City
Lago
Abuj
Kadu

3.2.4 The LEN Function

The LEN function returns the length of the value in a text column. The syntax is:

SELECT LEN(column_name) FROM table_name

Example: Let us select the length of the values in the Address column of Persons table.

We use the following SELECT statement:

SELECT LEN(Address) as LengthOfAddress FROM Persons

The result-set will look like this:

LengthOfAddress
15
14
14

3.2.5 The ROUND() Function

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

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.

Example: Let us consider the Products table below:

ProductID	ProductName	Unit	UnitPrice
11	Sugar	1000 g	10.45
12	Salt	1000 g	32.56
13	Palm Oil	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
Sugar	10
Salt	33
Palm Oil	16

3.2.6 The NOW Function

The NOW function returns the current system date and time. The syntax is:

SELECT NOW() FROM table_name

Example: Let us consider the product table again. 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	
Sugar	10.45	8/18/2009 10:35:02 AM	
Salt	32.56	8/18/2009 10:35:02 AM	
Palm Oil	15.67	8/18/2009 10:35:02 AM	

3.2.7 The FORMAT Function

The FORMAT function is used to format how a field is to be displayed. The syntax is:

SELECT FORMAT(column_name,format) FROM table_name

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

Example: Let us make use of the products table here. 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
Sugar	10.45	2009/8/18
Salt	32.56	2009/8/18
Palm Oil	15.67	2009/8/18

Activity B

1. Write out the SQL syntax for the following functions:

i. U
CASE
ii. L
EN iii.
MID
iv. LCA
SE
v. RO
UN
D vi.
NO
W
vii. FORMAT

4.0 Conclusion

SQL has many built-in functions for performing calculations on data. These functions were categorized into: SQL Aggregate functions and SQL Scalar functions. The aggregate functions operate against a collection of values, but return a single, summarizing value. Scalar functions Operate against a single value, and return a single value based on the input value. Some scalar functions, *CURRENT_TIME* for example, do not require any arguments.

5.0 Summary

In this unit, we have learnt:

xl. The basics of retrieving data from the database using SQL. xli. AVG function is to return the average value of a column in a database table. xlii. COUNT function returns the number of rows in a database table. xliii. FIRST function returns the first value in a database table. xliv. LAST function returns the last value in a database table. xlv. MAX function returns the largest value

xlvi. MIN function returns the smallest value

xlvii. SUM function returns the sum

xlviii. UCASE function converts a field to upper case xlix. LCASE converts a field to lower case

l. MID function extract characters from a text field

li. LEN function returns the length of a text field

lii. ROUND function rounds a numeric field to the number of decimals specified

liii. NOW function returns the current system date and time

liv. FORMAT function formats how a field is to be displayed

6.0 Tutor Marked Assignment

PFNO	NAMES	STATUS	HIREDATESALARY	SALARY	COMM	DEPTNO
1	AJAYI	CLERK	17-Dec-80	800		10
2	CHIM	SALESMAN	20-Feb-81	1600	300	40
3	JOHN	MANAGER	2-Apr-81	1250		40
4	WILL	SALESMAN	28-Sep-81	1250	300	30
5	KUDI	MANAGER	1-May-81	2975		30
6	TOLA	MANAGER	9-Jun-81	2850		20
7	ABDUL	ANALYST	27-Jun-90	3000		20
8	JAKE	PRESIDENT	3-Dec-81	5000		10
9	CLERK	SALESMAN	31-Jul-90	1234	500	40
10	SHEU	CLERK	3-Dec-81	1100		40
11	CHIDI	CLERK	3-Dec-81	950		20
12	HENRY	ANALYST	23-Jan-82	3000		20

13	IDIA	CLERK	23-Jan-82	1200		30
14	KUTI	SALESMAN	23-Jan-82	1600	600	20
15	BELLO	CLERK	23-Jan-82	1250		10

Employees Table

From the above tables, write the SQL statement that:

xi. Calculate the employees salary xii. Find the number of employees
xiii. Find the highest salary
xiv. Find the total sum of salary paid xv. Find the total salary for each status
xvi. Which positions are paid higher than average salary?

7.0 Further Reading and other Resources

David M. Kroenke, David J. Auer (2008). Database Concepts. New Jersey . Prentice Hall
Elmasri Navathe (2003). Fundamentals of Database Systems. England. Addison Wesley.
Fred R. McFadden, Jeffrey A. Hoffer (1994). Modern Database management. England. Addison Wesley Longman
Graeme C. Simsion, Graham C. Witt (2004). Data Modeling Essentials. San Francisco. Morgan Kaufmann
Pratt Adamski, Philip J. Pratt (2007). Concepts of Database Management. United States. Course Technology.