SQL Course

Aggregate Functions

- Introduce the functions available for use in a SQL query.
- Illustrate the use of the many functions using a few examples.

- SQL lets you summarise data from the database through a set of column functions.
- A SQL column function takes an entire column of data as its argument and produces a single data item that summarises the column.

SQL offers six different column functions, as shown below :
Sum()
Computes the total of a column. The result of the **SUM()** function has the same basic data type as the data in the column, but the result may have a higher precision. The syntax is as follows :

SUM ([**DISTINCT**|**ALL**] *column*)**;**

Avg()
Computes the average value in a column. The result of the **AVG()** function may be of a different data type then the values in the column. The syntax is as follows:

AVG([**DISTINCT** | **ALL**] *column*)**;**

Min()
Finds the smallest value in a column. The syntax is as follows :

MIN (expr)**;**

Max()
Finds the largest value in a column. The syntax is as follows :

MAX (expr)**;**

Count()
Counts the number of values in a column. The syntax is as follows :

COUNT([**DISTINCT**|**ALL**] expr)**;**

Count(*)
Counts rows of query results. The syntax is as follows :

COUNT (*)**;**

Both **MIN()** and **MAX()** can operate on numeric, string or date/time information. For numeric numbers, SQL compares the numbers in algebraic order. Dates are compared sequentially. Durations are compared based on their length. The comparison of two strings depend on the character set being used.

There are also a few other SQL functions :
Stddev()

Finds the Standard Deviation in a column. The syntax is as follows :

STDDEV ([**DISTINCT**|**ALL**] *column*)**;**

Variance()
Finds the Variance in a column. The syntax is as follows :

VARIENCE ([**DISTINCT**|**ALL**] *column*)**;**

You can ask SQL to eliminate duplicate values from a column before applying a column function to it. To eliminate duplicate values, the keyword **DISTINCT** is included before the column function argument, immediately after the opening parenthesis.

One of the best ways to think about column functions is to imagine the query processing broken down into two steps :

- First, imagine how the query would work without the column functions, producing many rows of query results.
- Then imagine the DBMS applying the column functions to the detailed query results, producing a single summary row.

It's illegal to nest column functions.It's also illegal to mix column functions and ordinary column names in a select list, because the resulting query doesn't make sense.
As stated, an aggregate function returns one single data item that summarises the chosen column, but if you want to categorise that column by a value in another column you must use the **GROUP BY** clause.

The ANSI/ISO standard specifies that **NULL** values are ignored by column functions. This can lead to inconsistencies when counting the number of rows in a query result. The ANSI/ISO standard specifies these precise rules for handling **NULL** values in column functions :

- **NULL** values are ignored by column functions.
- If every data item in a column is **NULL** , then **SUM()**, **AVG()**, **MIN()** and **MAX()** return a **NULL** result. **COUNT()** returns a value of zero.
- If there are no data items in the column, then **SUM()**, **AVG()**, **MIN()** and **MAX()** return a **NULL** value and **COUNT()** returns a value of zero.
- **COUNT(*)** does not depend on the presence or absence of **NULL** so will work as usual.

These rules may vary from specific implementation to implementation.

The SQL1 standard specifies that when **DISTINCT** is used, the argument to the column function must be a simple column name, it cannot be an expression. SQL2 relaxes these restrictions and allows the **DISTINCT** keyword to be applied for any of the column functions, and permitting expressions as arguments for any of the functions as well.
The **DISTINCT** keyword can only be specified once in a query. If it appears in one column function, then it cannot appear in any other. If it is specified before the column list then it cannot appear in any column functions. The only exception is that it may appear within a sub-query.

SQL1 did not have support for built in functions, even though most implementations incorporated their own. In general, a built in function can be specified in a SQL expression anywhere that a constant of the same data type can be specified. The SQL2 standard incorporated the most useful built in functions from commercial implementations, in many cases

with slightly different syntax. The functions are summarised in the table below :

Built-In functions in SQL2

Function	Returns
Bit_Length(string)	numbers of bits in a string
Cast(value AS data_type)	the value, converted to the specified data type
Char_Length(string)	Numbers of bits in a string
Conver(string **Using** conv)	string converted as specified by a named conversion function
Current_Date	current date
Current_Time(precision)	current time with the specified precision
Current_Timestamp(precision)	current date and time with the specified precision
Extract(part **From** source)	specified part (**Day**, **Hour**, etc.) from a **Datetime** value
Lower(string)	string converted to all lower case letters
Octet_Length(string)	number of 8 bit bytes in a character string
Position(target **In** source)	position where the target string appears within the source string
Substring(source **From** n **For** len)	a portion of the source string, beginning at the nth character, for a length of len
Translate(string **Using** trans)	string translated as specified by a named translation function
Trim(**Both** char **From** string)	string with both leading and trailing occurrences of char trimmed off
Trim(**Leading** char **From** string)	string with any leading occurrences of char trimmed off
Trim(**Trailing** char **From** string)	string with any trailing occurrences of char trimmed off
Upper(string)	string converted to all upper case letters

There are many types of function which include :
Arithmetic Functions

Arithmetic Functions are used in expressions involving column values:

Abs (n)	absolute value of n
Ceil (n)	smallest integer greater than or equal to n
Floor (n)	largest integer greater than or equal to n
Mod (m,n)	remainder of m divided by n
Power (m,n)	m raised to nth power. If n is not an integer it will be truncated.

Other arithmetic functions include **Round**, **Sign**, **Sqrt** and **Trunc**.

Functions on Character Strings
There are many Character string functions including:

Ascii (c)	**Ascii** value of first character of c
Chr (n)	The character with **Ascii** value n (integer)
Initcap (c)	Capitalises the first letter of each word contained in c

Other functions include **Upper**, **Substr**, **Lower**, **Length** and **Soundex** .

Date functions
Several functions which operate on **Date** types (columns holding values defined by type date) include:

Add_Months (d,n)	Adds a number of months n to a date d
Last_Day (d)	Get last day of month containing date d
Months_Between (d,e)	Number of months between dates d and e.

Other date functions include **New_Time**, **Next_Day** and **Trunc**.

Miscellaneous and conversion functions
There are several MISCELLANEOUS functions:
Decode, **Dump**, **Greatest**, **Least** , **Nvl** and **Vsize**.

Another set of functions will perform CONVERSIONS of columns or values into different datatypes e.g. **To_Number**, **To_Char** and **To_Date**.

The following example will count the number of flights in the database from the table aircraft_flight :

SELECT COUNT(*) FROM *aircraft_flight*;

Call Sign	Aircraft Name	Model	Club Seats	Econ Seats
N410C	Eagle Flyer	ATR42	22	40
9J-AEB	NULL	Boeing 707-320C	50	102
N7255U	NULL	Boeing 727-200	34	100
DQ-FDM	Finians Dream	Boeing 727-200	22	96
N301SW	NULL	Boeing 737-200	8	120

The answer returned is 5

The following example calculates the minimum fare paid on any flight :

SELECT MIN(*fare***) FROM** *fare***;**

Flight No	Departure Date	Seat Class	Fare
EI082	01/08/99	Econ	49
BD303	11/06/99	Econ	69
EI124	08/07/99	First Class	95
SK537	21/05/99	Econ	99
EI989	12/06/99	First Class	89

The answer returned is 4

SELECT*country*, **COUNT(***id***)FROM** *airport* **GROUP BY** *country*;

ID	Location	Country	Time Difference
a1	Dublin	Ireland	0
a2	Shannon	Ireland	0
a3	Heathrow	England	0

The answer returned is:

Ireland 2 England 1