Row and Column Functions

Using Functions in SQL

- Now that we've learned to SELECT data from relational tables, let's see how we can manipulate and analyze the data.
- SQL has many of the common functions that are built into other languages.
- It also has some that are common only to SQL. They are specific to the tabular arrangement of the data.

A Column of Numbers…

16
19
28
32
6

????

- What kinds of things can you do with these numbers?
- Think first grade math, think college statistics, think like a business person.
- List as many operations or functions you can think of.

→When we use functions on all the values in the column, it's often called a **column function** or **an aggregate function**.

A Row of Data……

[Smith, John, 8/7/1981, 80000]

- What functions can we do on this data?
  - Think about string functions……
  - Think about date functions…..
  - Assume 80000 is annual salary. How might we want to view that?
- When we use a function to process a single value in a row or even multiple values in the row, we call it a **row function.**

Row v. Column Functions

- **Column functions** involve summarization of column data in 1 or more rows. Examples,
  - **Count, Min, Max, Sum, Avg, Median**
- **Row functions** are the operations you can apply to data values in a single row. Examples,
  - Mathematical Functions
  - Text Functions
  - Date Functions

Starting with Column Functions

Max	Maximum value in column
Min	Minimum value in column
Count(*)	Total number of rows in table
Count(col)	Number of rows where col **IS NOT NULL**
Count (distinct col)	Number of distinct values in col
Sum	Sum of all values in column
Avg	Average of all values in column
Stddev	Standard Deviation
Variance	Variance
Median	Middle Value (Oracle only!)

Examples of Column Functions

```
SELECT MIN(price),MAX(price), AVG(price), MEDIAN(price)
FROM l_foods;
```

MIN(PRICE)	MAX(PRICE)	AVG(PRICE)	MEDIAN(PRICE)
.85	6	2.31	1.75

```
SELECT SUM (quantity)
FROM l_lunch_items
WHERE item_number = 3;
```

SUM(QUANTITY)
21

Use the round function to limit output to 2 decimal places

```
SELECT ROUND(AVG (credit_limit), 2)
FROM l_employees;
```

ROUND(AVG(CREDIT_LIMIT),2)
23.33

MySQL - Tip for functions

- MySQL sometimes requires that there be **NO SPACE** between a function name and the following paren. This is because MySQL allows some function names to also be identifiers and it must have a way to determine the difference. For a complete explanation and list see: https://dev.mysql.com/doc/refman/5.7/en/function-resolution.html
- Here are a few: CAST, COUNT, DATE_ADD, SYSDATE EXTRACT, MAX, MID, MIN, DATE, SUBSTRING, SUM
- **Do this: SELECT COUNT(*)**
- Not This: SELECT COUNT (*)

Basic Statistical Functions

- **Mean** – use ANSI SQL function, **AVG**
- **Median** –Oracle has a **MEDIAN** function. MySQL does not. You must code your own or borrow someone else's solution.
- **Mode** – (most common data value) Neither vendor supports this. It can be simulated with GROUP BY in a subquery, then extract the first or last row of the group. (you'll learn GROUP BY and subqueries soon!)
- **Range** – Easy. Use **BETWEEN** clause.

Count Functions

Very important to remember the difference between 3 types of counts

- **Count(*)** – Count all of the rows in a table. Great for knowing size of table before executing dangerously long query.

```
SELECT COUNT(*) FROM l_employees;
```

- **Count(column)** – count all values in a column excluding nulls

```
SELECT COUNT(manager_id) FROM l_employees;
```

- **Count(distinct column)** – counts the number of different values in a column.

```
SELECT COUNT (DISTINCT manager_id)
FROM l_employees;
```

More on COUNT DISTINCT

- DISTINCT can apply to more than 1 column, but not if we are also using COUNT.
- To apply DISTINCT and COUNT functions to several columns, must first concatenate the columns.
- Example,

```
SELECT COUNT (distinct supplier_id || product_code)
FROM l_lunch_items;              -- result is: 9
```

- Note: It's best to add a column separator character to avoid incorrect result if concatenation of different strings produces the same result.
- Here's the code without COUNT function:

```
SELECT DISTINCT supplier_id,  product_code
FROM l_lunch_items;              -- displays 9 rows of data
```

What's going on here?

```
SELECT
      COUNT (*),
      COUNT(last_name),
      COUNT(hire_date),
      COUNT(manager_id)
FROM I_employees;
```

Count(*)	Count(last_name)	Count(hire_date)	Count(manager_id)
10	10	9	8

Why are these counts different?

Here's the data…

LAST_NAME	HIRE_DATE	MANAGER_ID
BROWN	01-JUN-98	-
KERN	16-AUG-99	201
WOODS	02-FEB-09	201
OWENS	01-JUL-08	202
PERKINS	01-MAR-06	202
ROSE	-	-
SMITH	01-DEC-08	203
CAMPBELL	01-APR-08	203
JACOBS	17-MAR-99	201
HOFFMAN	16-FEB-07	203

COUNT will skip the NULLs!

Switching Now to Row Functions

- **Row functions** are the operations you can apply to data values in a single row (or a selection of rows).
- Row functions DO NOT operate all selected rows simultaneously. That is the job of the column functions.
- Examples,
  - **Mathematical Functions**
  - **Text Functions**
  - **Date Functions**

Row Functions

Can be applied within the SELECT, WHERE, or ORDER BY clauses

```
SELECT to_char(date_entered, 'mm/dd/yy hh:mm')
FROM l_lunches;                          -- Oracle

SELECT date_format (date_entered, '%m/%d/%y %h:%i');
                                         -- MySQL
SELECT concat(first_name, last_name,)
FROM l_employees;                        -- Oracle & MySQL

SELECT first_name || ' ' || last_name FROM l_employees;
                                         -- Oracle & MySQL*
SELECT last_name FROM l_employees
WHERE length(last_name) = 4;   -- Oracle & MySQL
```

In MySQL, must first allow || for concatenation : set sql_mode=PIPES_AS_CONCAT;

TIP: Searching for Functions

- Every dialect of SQL will have lots of functions available.
- When searching online, be sure to enter the name of your database engine and the general type of function you want:
  - ORACLE: string functions
  - MySQL: Date functions
  - Access: numeric functions

Commonly Used Math Functions

Function	Engine	Description	Example
Power	Oracle & MySQL	Exponents	Power(5,2)=25
Sqrt	Oracle & MySQL	Square root	Sqrt(25) = 5
DIV	MySQL	Integer division	10 div 3 = 3
Mod	BOTH (MySQL can also use %)	Remainder from integer division	Mod (10, 3) = 1 10 % 3 = 1
Sign	BOTH	Sign indicator (1 if positive, -1 if neg)	Sign (-8) = -1
Abs	BOTH	Absolute value	Abs(-8) = 8
Ceil	BOTH	Smallest integer >= n	Ceil (3.5) = 4
Floor	BOTH	Largest integer <= n	Floor (3.5) = 3
Round	BOTH	Round to precision	Round (3.4567,2) = 3.46
Truncate Trunc	BOTH	Truncate to precision (must use precision with MySQL).	Trunc(3.4567,2) = 3.45 Trunc(3.4562) = 3

Basic Math on Columns

(and problems with this!)

- SQL provides normal mathematical operations: + - * /

SELECT price, price + price_increase AS "New Price"
FROM I_foods;

- Why is "new price" sometimes null? What's going on here?

PRICE	NEW PRICE
2	2.25
1.5	-
3.5	3.9
6	6.7
2.5	2.8
1	1.05
1.5	-
1.25	1.5
.85	1
3	3.5

Mathematics with Null Values

- What happens when we add an unknown value to your age? What is the result? Same is true with null values in SQL:
  - colA + null = **null** colA – null = **null**
  - colA * null = **null** colA / null = **null**
- **When you try to do regular math and one column (or value) is null, the result will always be null.**
- How do we get around this problem?

Functions to convert NULLs

Database Engine	Functions
Oracle	Coalesce (), Case, **NVL()**
DB2	Coalesce(), Case, NULLIF()
MySQL	Coalesce(), Case, **IFNULL()**
MS SQL	Coalesce(), Case, IFNULL(), NULLIF()
MS Access	NZ()

- We use a special function that will convert all NULL values to a legal value for performing mathematical calculations.

```
SELECT COALESCE(credit_limit, 0)
FROM  l_employees;
```

Or

```
SELECT nvl(credit_limit, 0)
FROM  l_employees;
```

For these queries, credit_limit will be set to zero if it is NULL

Using NVL or IFNULL

MYSQL:

```
SELECT  IFNULL(credit_limit, 0)
FROM l_employees;
```

IFNULL(credit_limit, 0)
30.00
25.00
25.00
15.00
25.00
0.00
25.00
25.00
15.00
25.00

Oracle:

```
SELECT  NVL(credit_limit, 0)
FROM l_employees;
```

NVL(CREDIT_LIMIT,0)
30
25
25
15
25
0
25
25
15
25

NULL VALUE Functions

- In Oracle, most common to use the **NVL()** Function.
- In MySQL, **IFNULL()** is most common.
- Standard SQL recommends **COALESCE**
- If most systems support “coalesce” why doesn’t everybody use it?
- **?? Perhaps it’s just too hard to spell??**
- **Or perhaps is just a legacy thing.**

How to use Null Value Functions

- When changing nulls to a new value, **must match the data type** of the column.

```
SELECT last_name,
   NVL(dept_code, 'undefined'),    -- varchar
   NVL(hire_date, '01-jan-1900'),  -- date
   NVL(credit_limit, 0)            -- integer
FROM l_employees;
```

Use NVL to fix the price query

```
SELECT price,
price + nvl(price_increase, 0) AS “New Price”
FROM I_foods;
```

- Now we don’t have any NULLs in the NEW PRICE column!
- The rows colored yellow indicate where nulls were converted to zero.

PRICE	NEW PRICE
2	2.25
1.5	1.5
3.5	3.9
6	6.7
2.5	2.8
1	1.05
1.5	1.5
1.25	1.5
.85	1
3	3.5

Using NVL properly

- Be careful with NVL. If you must apply multiple functions to a column, **always apply NVL first.**
- Look at this query & notice how NVL is used:

```
SELECT description,  price, price_increase,
price_increase + price as new_price,
nvl(price + price_increase, 0) as new_price_NVL_outside,
price + nvl(price_increase, 0) as new_price_NVL_inside
FROM l_foods;
```

What's the difference? Does it matter?

Results:

Notice that the rows highlighted in yellow Have incorrect calculations for new_price when Nvl() was not applied FIRST

Desc	price	price_ increase	New_price	New_price_ NVL_outside	New_price_ NVL_inside
FRESH SALAD	2	.25	2.25	2.25	2.25
SOUP OF THE DAY	1.5	-	-	0	1.5
SANDWICH	3.5	.4	3.9	3.9	3.9
GRILLED STEAK	6	.7	6.7	6.7	6.7
HAMBURGER	2.5	.3	2.8	2.8	2.8
BROCCOLI	1	.05	1.05	1.05	1.05
FRENCH FRIES	1.5	-	-	0	1.5
SODA	1.25	.25	1.5	1.5	1.5
COFFEE	.85	.15	1	1	1
DESSERT	3	.5	3.5	3.5	3.5

Integer Division

- As programmers, we often find integer division to be useful. Why?
  - determine if something is evenly divisible by another number
  - break numbers apart
- See next slide for comparison of methods to achieve integer division.

The Case of Integer Division

SQL dialects differ in how they handle Integer Division:

- MySQL: **SELECT DIV (7,3)**
- Oracle: **SELECT FLOOR (7/3)**
- DB2 & MS: **SELECT a/b.**

(If a and b are integers, result will be integer)

- Access: **SELECT A\B**
- SQL standard does not specify due to differences in platforms.

The Problem of Division by Zero

- ORACLE, MySQL, Access, others: All *can* produce an error message. By default Oracle will generate error.
- **MySQL:** Can configure the behavior. By default,your PHPMyAdmin will return NULL when you divide by zero (no error!). MySQL will only generate an error on INSERTS if you use the last option in this table.

sql_mode Value	Result
(Default)	**No warning, no error; i is set to NULL.**
STRICT_ALL_TABLES	**No warning, no error; i is set to NULL.**
ERROR_FOR_DIVISION_BY_ZERO	**Warning, no error; i is set to NULL.**
STRICT_ALL_TABLES, ERROR_FOR_DIVISION_BY_ZERO	Error condition; no row is inserted.

MySQL: Error for division by 0

- Here is the exact syntax to use if you would like MySQL to behave more strictly.
- Note that even when this is used SELECT statements will still return NULL for division, by zero. This setting affects INSERTS:

```
SET sql_mode =
    'STRICT_ALL_TABLES,ERROR_FOR_DIVISION_BY_ZERO';
INSERT INTO l_employees VALUES (500, 'Spasyk', 'Joan', 'SAL',
    '2015/01/16', 1/0, 2222, 201);
```

MySQL said:

#1365 - Division by 0

Be careful! It's tricky to do some of this in PHPMyAdmin because each command is a separate connection. The mode setting is not persistent.

Testing Functions

- Oracle provides a general purpose table that helps with testing or calculations.
- The table is called “DUAL”
- Think of it as a temporary variable.
- Example:

SELECT 3 * 4 from dual;

- In MySQL, you do not need to specify FROM Dual, but it is accepted. You can merely enter, **SELECT 3 * 4;**

String Functions

All these functions are available in Oracle and MySQL.

concat or ||	Concatenation	concat (‘sun’, ‘flower’) in MySQL ‘sun’ || ‘flower’ in Oracle
substr	Substring	substr (‘sunflower’, 4,3) = ‘flo’ substr(str, start, numchars)
soundex	Sounds similar	where soundex(col)= soundex (‘Jon’)
upper	Uppercase	upper(‘sun’) = ‘SUN’
lower	Lowercase	lower (‘SUN’) = ‘sun’
Initcap*	Capitalize 1st letter	Initcap(‘sun’) = ‘Sun’ Oracle only
ltrim	Left trim	ltrim (' hi ') = 'hi '
rtrim	Right trim	rtrim (' hi ') = ' hi'
trim	Trim both	trim (' hi ') = 'hi'
length	String length	length ('sun') = 3
Instr	Substring position	Instr ('sunflower', 'low') = 5

Many more string functions exist – always check vendor’s web pages

* Oracle only. In MySQL use substring and upper()/lower() functions.

Comments

- Most of the string functions operate much like other languages. Just look up the syntax.
- Oracle ‘concat’ is a bit limited. Cannot concatenate more than two strings. Better to use the ‘||’ operator.

```
'MRS.'  ||  LAST_NAME
MRS. BROWN
```

```
SELECT 'MRS. ' || last_name
FROM l_employees where last_name LIKE 'BR%';
```

- MySQL lets you use concat on > 2 strings.
- MySQL only lets you use || if you first use:

```
set sql_mode = 'PIPES_AS_CONCAT';
```

Soundex

- Soundex is an algorithm that predates computers. It was developed in 1918 for use by the census bureau.
- It can be used to determine whether two strings sound like each other.
- The algorithm will return the first letter of the string followed by a 3 digit number that represents sound of the remainder of the string.
- Oracle and MySQL soundex functions are a bit different.

A little fun with soundex

Oracle Example:

SELECT soundex('Matthew') FROM dual;

- M300

MySQL Example:

SELECT substr(soundex ('David'), 1,4)

- D130 -- From Dual not needed in MySQL

Note: The algorithms for Soundex in MySQL and Oracle a are a little bit different. Soundex in Oracle will always return 4 characters. Soundex in MySQL returns a string of arbitrary length. I suggest that in MySQL you just look at the first 4 characters by using a substring function.

Exercises: Write SQL for -

- Combine first and last names into a single column (easy)
- Output full phone number given a standard area code and prefix

Date Functions - Advice

- Both Oracle and MySQL support many date functions, but you will find the solutions to queries requiring manipulation of dates to be quite different.
- It is best to use the vendor's web pages and test things out.
- Next slide shows one example…..

Date Calculations

Note: Easier to calculate number of days first and then calculate weeks, months, or years.

- Show all employees, their hire_date, and months worked as of 1/1/2015.

```
SELECT first_name, last_name, hire_date,
      FLOOR ((to_date('1/1/2015') - hire_date) / 30)
       AS MONTHS_WORKED
FROM l_employees;                    -- Oracle


SELECT first_name, last_name, hire_date,
     FLOOR (datediff(str_to_date('1/1/2015', '%m/%d/%Y'),
      hire_date) / 30)
AS MONTHS_WORKED
FROM l_employees;                    -- MySQL
```

More Useful Functions

Oracle	MySQL	Purpose
user	user()	Name of userid for current session
sysdate	sysdate()	Current Date & Time
to_char()	cast (col-name as char)	Converts a number or date to a string
to_date()	str_to_date()	Converts a number or string to a date
to_number()	cast (‘number’ as signed)	Converts a string to a number. See manual pages.
greatest()	greatest()	Chooses greatest member from a list of numbers, text, dates
least()	least()	Chooses the least member from a list of numbers, text, dates

IMPORTANT: Always check web pages for arguments and details.

Nesting Functions

- SQL allows you to nest functions.
- Examples:

```
SELECT TO_DATE(RTRIM(SUBSTR('The date is
    1/16/2012', 13)), 'mm/dd/yy') AS Trimmed_Date
FROM dual;
```

TRIMMED_DATE
1/16/2012

```
SELECT ROUND(AVG (credit_limit), 2)
FROM l_employees;
```

ROUND(AVG(CREDIT_LIMIT),2)
23.33

Data Type Conversions

- When are data types automatically converted and when must we explicitly use functions?
- Can we add 6 + 5 if they are declared as character data types?
- Oracle and MySQL will both allow this. At first glance they seem quite liberal with data type conversions!
- However, there are many times when more complex operations on mixed data types will fail or produce incorrect results. BE CAREFUL.

Discussion

- What's the result of a column function?
- What's the result of a row function?
- What happens when the column contains null values?
- What happens if there is also a WHERE clause, e.g.

```
SELECT max (price) FROM l_foods
WHERE supplier_id = 'ASP';
```

Review: Problems with Addition

- Some background: SQL “math” works differently on rows and columns
- The difference is in how the nulls are treated!
- In row addition, nulls are an unknown value. Adding a null produces a null result –

  3 + null = null

- In column addition, e.g when using the SUM function, nulls are ignored :

```
SELECT SUM (price_increase)
FROM I_foods;
```

SUM(PRICE_INCREASE)
2.6

Very important to remember this!

Problems with Addition – Example

Pk_1	Col_2	Col_3
A	1	4
B	Null	5
C	2	6
D	3	Null

```
SELECT SUM (col_2) + SUM (col_3) AS columns_first,
       SUM (col_2 + col_3) AS rows_first
FROM sec1111;
```

COLUMNS_FIRST	ROWS_FIRST
21	13

Best Solution

- Always remember to apply columns functons first! This will produce a “correct” result
- BUT if you must add across the row, be aware of numeric columns that allow nulls.
- Use the nvl function to change the values to 0 before performing calculations.

NVL function

- Using nvl, coalesce, etc. will also ensure correct results!

```
SELECT
SUM (col_2) + sum(col_3) AS col_first,
SUM(nvl (col_2,0) + nvl (col_3,0)) AS row_first
FROM sec1111;
```

- *(note: this is Oracle syntax; For MySQL substitute ifnull for nvl)*

Counting Nulls in a Column

- How do we do this when NULLS are normally ignored in column functions?
- Not too tricky….

```
SELECT COUNT(*) as number_of_nulls
FROM l_employees
WHERE manager_id IS NULL;
```

Remember to use count(*) here and not count(col)!!

RECAP:
Row v. Column Functions

- **Row functions** are the operations you can apply for each row. You will get one result per row. Examples,
  - Mathematical Functions
  - Text Functions
  - Date Functions
- **Column functions** produce one result per column for ALL selected rows. Examples,
  - Count, Min, Max, Sum, Avg

Another Way to Look at it

- Column functions are called Aggregate Functions. They include min, max, average, sum, count, etc.
- Row functions are all the other functions - string, date, and other math functions.