

Databases



IT Learning &
Outsourcing Center

Aggregations and Functions

www.pragmatic.bg

Lector: Hristo Topuzov
E-mail: Hristo.Topuzov@pragmatic.bg

2017

Copyright © Pragmatic LLC



Agenda...

- Aggregating data
- MySQL built-in functions

Aggregating data ?

PRAGMATIC

IT Learning &
Outsourcing Center

Support Dashboard


 Sonatica
Turn it up.

EXECUTIVE

SALES

SUPPORT

Specify a date range: January, 2009 15 to June, 2010 15



Hover Over

Support Volumes by Type

Support Volume

Empty Category

- Average Support Volume per Month
- Monthly Support Volume High
- Monthly Support Volume Low
- Ranges**
- Excessive Support Volume



E-Mail



Phone



Web Chat

Employees Tickets Opened

Jan 09	
Ryan Woods	12
Alan Ng	13
Alex Salenko	13
Jay Singh	13
Dave Carkner	12
Elena Orlova	12
Feb 09	
Ryan Woods	8
Alan Ng	9
Alex Salenko	9
Jay Singh	8

Ticket Volume vs. Targets (click to view RS report)





Aggregating data

- Aggregating data allows us to group rows based on certain criteria
- SQL provides a mechanism for aggregating data using the GROUP BY clause in a SELECT statement
- SQL provides a standard set of functions that can be used over aggregated data



Aggregate functions

- COUNT(*) or COUNT(1) - count of the selected rows
- SUM(column) - sum of the values in given column from the selected rows
- AVG(column) - average of the values in given column
- MAX(column) - the maximal value in given column
- MIN(column) - the minimal value in given column



Aggregate functions

- You can use AVG() and SUM() for numeric data types
- Example:
 - `select sum(Salary), avg(Salary)`
`from Employees`
`where managerId = 5`

<code>sum(Salary)</code>	<code>avg(Salary)</code>
17000	1700.0000



Aggregate functions

- You can use MIN() and MAX() for any data type (number, date, varchar, ...)
- Examples:

- select min(Salary), max(Salary)
from Employees
where managerId = 5

min(Salary)	max(Salary)
800	2300

- select min(HireDate), max(HireDate)
from Employees
where managerId = 5

min(HireDate)	max(HireDate)
2008-05-01	2013-10-01



Aggregate functions

- MIN() and MAX() display the first and last employee's name in alphabetical order

- select min(Name),max(Name)

- from Employees

- where managerId = 5

min(Name)	max(Name)
Bogomil Borisov	Vili Nikolova

- COUNT(*) and COUNT(1) returns the number of rows in the result table

- select count(1) cnt from employees where
DepartmentId = 5



Aggregate functions

- COUNT(expr) returns the number of rows with non-null values for the expr
 - select count(enddate) cnt from employees
- COUNT(DISTINCT expr) returns the number of distinct non-null values
 - select count(distinct enddate) cnt from employees



Aggregate functions

- Group functions ignore all null values in the column
 - `select avg(ReferrerId) cnt from employees`
- If each null value in ReferrerId is considered as 0 and is included in the calculation, the result would be 1,04 instead of 12,5



Aggregate functions

- We can divide rows in a table into smaller groups by using the GROUP BY clause

EMPLOYEES

DEPARTMENT_ID	SALARY
50	3100
50	3000
50	2600
50	2600
20	4400
20	13000
20	6000
40	6500
40	10000
110	12000
110	8300
...	...

Grouping Data

DEPART MENT_ID	SUM(SALARY)
50	11300
20	23400
40	16500
110	20300
...	...



Aggregate functions

- The syntax:


```
SELECT <columns>, <group_function(column)>  
FROM <table>  
[WHERE <condition>]  
[GROUP BY<group_by_expression>]  
[ORDER BY<columns>]
```

- The <group_by_expression> is a list of columns



Aggregate functions

- All selected columns should stay in the GROUP BY statement or in a group function
- Example of grouping data:
 - select departmentid, sum(salary)
from employees
group by departmentid

 departmentid	sum(salary)
2	2700
3	2300
4	1200
5	3400
6	41000
7	43400
8	17000
9	1700
10	3200



Aggregate functions

EMPLOYEES

DEPART MENT_ID	JOB_ID	SALARY	
20	AD_ASST	4400	4400
20	MK_MAN	13000	
20	MK_MAN	12000	25000
30	PU_CLERK	2500	
30	PU_CLERK	2500	7500
30	PU_CLERK	2500	
30	PU_CLERK	2500	
30	PU_MAN	11000	43500
30	PU_MAN	11500	
30	PU_MAN	10000	
30	PU_MAN	11000	
...	

Grouping Data Using
Several Columns

EMPLOYEES

DPT_ID	JOB_ID	SUM(S ALARY)
20	AD_ASST	4400
20	MK_MAN	25000
30	PU_CLERK	7500
30	PU_MAN	43500
...



Aggregate functions

- Example of grouping data using several columns
 - `select departmentid, titleid, sum(salary)`
`from employees`
`group by departmentid, titleid`
`order by 3`

departmentid	titleid	sum(salary)
3	9	900
4	8	1200
3	10	1400
8	16	1600
9	6	1700
2	4	2700
10	5	3200
7	13	3200
5	7	3400



Aggregate functions

- This SELECT statement is illegal
 - `select titleid, sum(salary)`
`from employees`
- This SELECT statement is also illegal
 - `select departmentid, titleid, sum(salary)`
`from employees`
`where sum(salary) > 10000`
`group by departmentid, titleid`
`order by 3`



Aggregate functions

- Can not use WHERE for group functions
- The WHERE clause places conditions on the selected columns, where as the HAVING clause places conditions on groups created by the GROUP BY clause
- The HAVING clause enables you to specify conditions that filter which group results appear in the final results
- HAVING works exactly like WHERE but is used for the grouping functions



Aggregate functions

- Example:
 - select departmentid, titleid, sum(salary)
from employees
where departmentid in (5,6,7)
group by departmentid, titleid
having sum(salary) > 10000
order by 3

departmentid	titleid	sum(salary)
7	11	17000
6	3	21000
7	12	23200



Aggregate functions

- Exercises:
 - select the average salary of all employees
 - select the maximum salary of all employees
 - select the minimum salary of employees from department 5
 - count number of employees
 - count number of employees with a manager (2)
 - count number of managers
 - count number of employees in departments with average salary higher than 2000



MySQL Functions

- MySQL functions are routines that accept parameters, perform an action, such as a complex calculation, and return the result of that action as a value
- MySQL supports user-defined functions and built-in, system functions
- Built-in functions are provided to help you perform a variety of operations. They cannot be modified

MySQL Built-in Functions



- String manipulation built-in functions include:
 - INSERT()-insert a substring at a specified position
 - INSTR() -returns the index of first occurrence of a substring
 - LCASE()/LOWER()-returns a string in lowercase letters
 - UCASE()/UPPER()-returns a string in uppercase letters
 - LENGTH()-returns the length of a string
 - LOCATE()/POSITION()-returns the position of the first occurrence of substring
 - REPEAT()-repeats a string a number of times



MySQL Built-in Functions

■ Examples:

- `SELECT INSERT('Name', 5, 4, 'First');`
- `SELECT INSTR('Name', 'am');`
- `SELECT LOWER('Name');`
- `SELECT UPPER('Name');`
- `SELECT LENGTH('Name');`
- `SELECT POSITION('am' in 'Name');`
- `SELECT LOCATE('am', 'Name');`
- `SELECT REPEAT('Name', 3)`

MySQL Built-in Functions



- String manipulation functions include:
 - LPAD()/RPAD() -returns a string padded with a given string on the left/right
 - LTRIM()-trims spaces on the left of a string
 - RTRIM()-trims spaces on the right of a string
 - TRIM()-trims characters on the left, right or both from a string
 - MID() - returns a substring starting from a given position
 - QUOTE()-escape the argument for use in an SQL statement



MySQL Built-in Functions

■ Examples:

- `SELECT LPAD('Value', 10, '#');`
- `SELECT RPAD('Value', 10, '#');`
- `SELECT LTRIM(' Value ');`
- `SELECT RTRIM(' Value');`
- `SELECT TRIM(' Value ');`
- `SELECT TRIM(BOTH '#' from '##VALUE###');`
- `SELECT TRIM(LEADING '#' from '##VALUE###');`
- `SELECT TRIM(TRAILING '#' from '##VALUE###');`
- `SELECT MID('Value', 3, 2);`
- `SELECT QUOTE('Value');`

MySQL Built-in Functions



- String manipulation functions include:
 - SPACE() - returns a string with a number of spaces only
 - CONCAT() - concatenates a string
 - SUBSTR() - returns a substring from the given string
 - REPLACE() - replaces characters in a string
 - REVERSE() - reverses a string



MySQL Built-in Functions

- Examples:
 - `SELECT SPACE(7);`
 - `SELECT CONCAT('Value', '=', '5')`
 - `SELECT SUBSTR('Value', 2,3);`
 - `SELECT REPLACE('Value', 'Val', 'h');`
 - `SELECT REVERSE('Value');`



MySQL Built-in Functions

- Numerical functions include:
 - ABS() - return the absolute value
 - CEIL() - return the smallest integer value not less than the argument
 - EXP()/POW() - raise to the power of
 - DIV() - Integer division
 - MOD() - returns the remainder
 - ROUND() - round the argument
 - TRUNCATE() - truncate to specified number of decimal places
 - SQRT() - return the square root of the argument



MySQL Built-in Functions

■ Examples:

- `SELECT ABS(-123.567);`
- `SELECT CEIL(123.567);`
- `SELECT POW(2, 3);`
- `SELECT 8 DIV 3;`
- `SELECT 8 MOD 3;`
- `SELECT ROUND(123.567);`
- `SELECT ROUND(123.567, 1);`
- `SELECT TRUNCATE(123.567, 1);`
- `SELECT SQRT(9);`



MySQL Built-in Functions

- Date functions include:
 - DATEDIFF() - subtract two dates
 - ADDDATE() - add time values (intervals) to a date value
 - ADDTIME() - add time
 - YEAR() - return the year from the date passed
 - MONTH() - return the month from the date passed
 - CURDATE() - return the current date



MySQL Built-in Functions

■ Examples:

- `SELECT DATEDIFF('2010-12-15','2010-12-10')`
- `SELECT ADDDATE('2014-01-01', INTERVAL 40 DAY);`
- `SELECT ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002');`
- `SELECT CURDATE();`
- `SELECT YEAR('2011-01-01');`
- `SELECT MONTH('2011-05-01');`

MySQL Built-in Functions



- Date functions include:
 - DATE_FORMAT() - format date as specified
 - DAY()/DAYOFMONTH() - return the day of the month
 - EXTRACT() - extract part of a date
 - MINUTE() - return the minute from the argument
 - HOUR() - extract the hour

MySQL Built-in Functions



- Examples:
 - `SELECT DATE_FORMAT(CURDATE(), '%D:%M:%Y');`
 - `SELECT DAY(CURDATE());`
 - `SELECT DAYOFMONTH(CURDATE());`
 - `SELECT EXTRACT(YEAR FROM CURDATE());`
 - `SELECT MINUTE(CURTIME());`
 - `SELECT HOUR(CURTIME());`



MySQL Built-in Functions

- Functions that handle NULL values:
 - COALESCE() - return the first non-NULL argument
 - ISNULL() - test whether the argument is NULL
- Examples:
 - `SELECT COALESCE(NULL, NULL, 'VALUE1', 'VALUE2');`
 - `SELECT ISNULL('VALUE');`
 - `SELECT ISNULL(null);`



MySQL Built-in Functions

- Comparison functions:
 - GREATEST() - return the largest argument
 - LEAST() - return the smallest argument
- Examples:
 - `SELECT GREATEST(2,0);`
 - `SELECT GREATEST('B','A','C');`
 - `SELECT LEAST(2,0);`
 - `SELECT LEAST('B','A','C');`



MySQL Built-in Functions

- Control flow functions:
 - IF() - If/else construct
 - IFNULL() - null if/else construct
 - NULLIF() - return NULL if expr1 = expr2
- Examples:
 - SELECT IF(1=2,2,3);
 - SELECT IFNULL(MANAGERID, '(no manager)') FROM EMPLOYEES;
 - SELECT NULLIF('1', '2');



MySQL Built-in Functions

- Type conversion functions:
 - CAST() - cast a value as a certain type
 - FORMAT() - return a number formatted to specified number of decimal places

- Examples:
 - `SELECT CAST(123 AS CHAR);`
 - `SELECT FORMAT(123.55, 1);`
 - `SELECT FORMAT(123.55, 0);`
 - `SELECT FORMAT(123.55, 5);`

Questions

