#### **Databases**



# Aggregations and Functions

Lector: Hristo Topuzov

E-mail: Hristo.Topuzov@pragmatic.bg

www.pragmatic.bg

2017

Copyright © Pragmatic LLC

## PRAGMATIC IT Learning & Outsourcing Center

## Agenda...

- Aggregating data
- MySQL built-in functions

## Aggregating data?



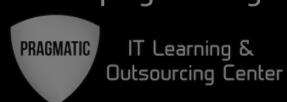




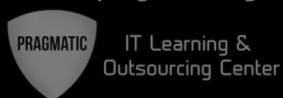
## Aggregating data

- 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





- 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



 You can use AVG() and SUM() for numeric data types

#### Example:

select sum(Salary), avg(Salary)

from Employees where managerId = 5

sum(Salary)	avg(Salary)
17000	1700.0000



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

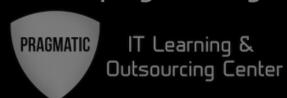
from Employees where managerId = 5

max(Salary)	min(Salary)
2300	800

select min(HireDate), max(HireDate)

from Employees where managerId = 5

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

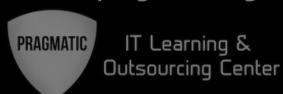


- MIN() and MAX() display the first and last employee's name in alphabetical order
  - select min(Name), max(Name)

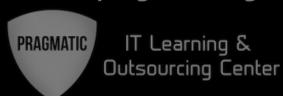
from Employees where managerId = 5

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



- 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

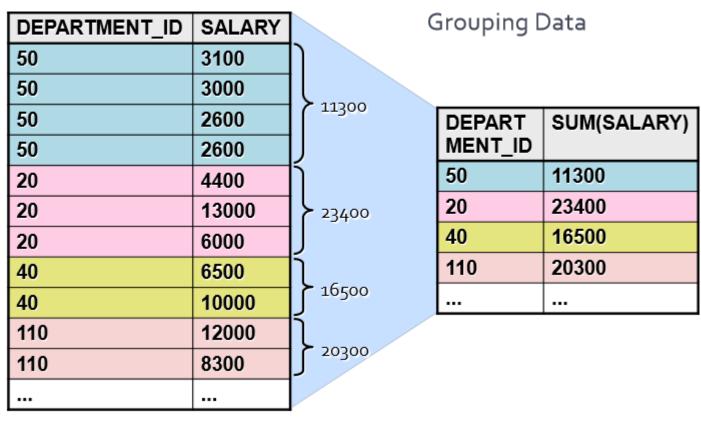


- Group functions ignore all null values in the column
   select avg(Referredd) cpt from employees
  - select avg(ReferrerId) cnt from employees
- If each null value in ReferrerId is considered as o and is included in the calculation, the result would be 1,04 instead of 12,5



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

EMPLOYEES





The syntax:

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

The <group\_by\_expression>is a list of columns



- 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



EMPL(	OYEES
-------	-------

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



- Example of grouping data using several columns
  - select departmentid, titleid, sum(salary)

from employees

group by departmentid, titleid

order by 3

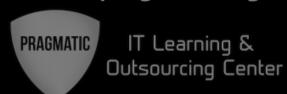
departmentid	titleid	sum(salary)
3	9	900
4	8	1200
3	10	1400
8	16	1600
9	6	1700
2	4	2700
10	5	3200
7	13	3200
5	7	3400



- 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



- 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



#### 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



#### 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

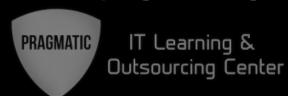
#### **MySQL Functions**



- 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 builtin, system functions
- Built-in functions are provided to help you perform a variety of operations. They cannot be modified



- 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



#### 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)



- 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



#### Examples:

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



- 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

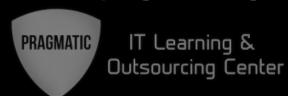


#### Examples:

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



- 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

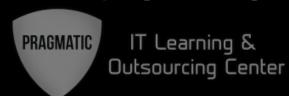


#### Examples:

- 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);



- 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

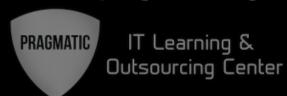


#### Examples:

- 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', '11:1:1.000002');
- SELECT CURDATE();
- SELECTYEAR('2011-01-01');
- SELECT MONTH('2011-05-01');



- 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



#### 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());



- 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);



- 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');



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



- 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

