SQL aggregations and functions

Agenda

- Aggregating data
- Aggregate functions
- MySQL functions

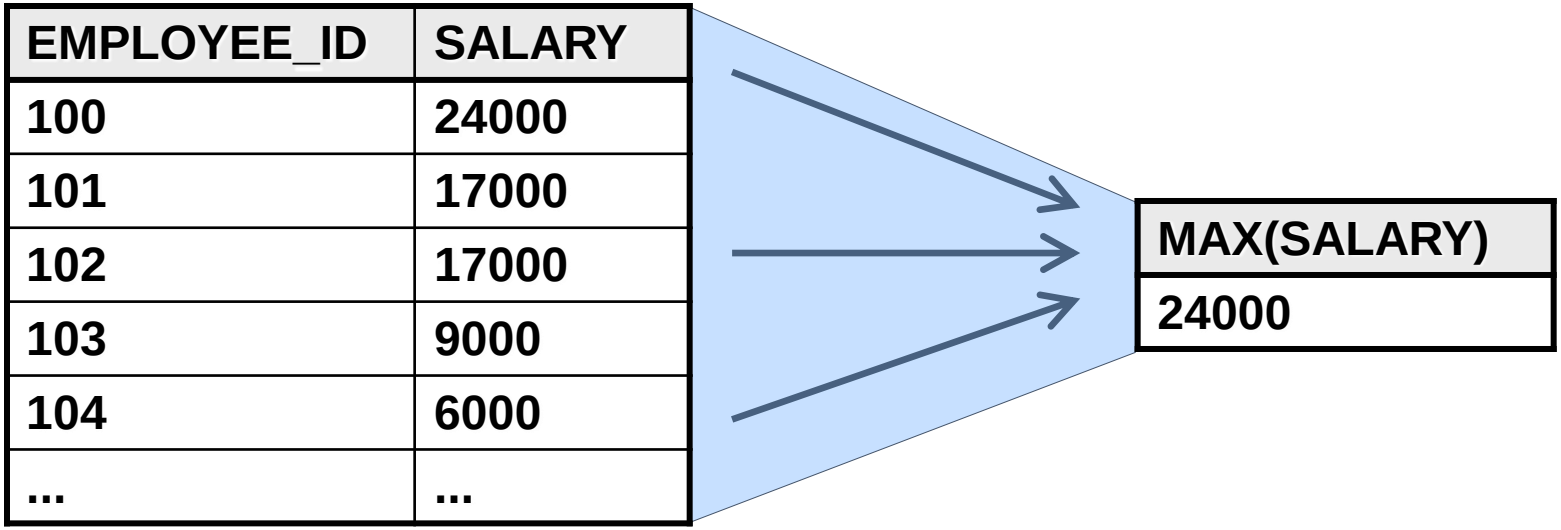
Aggregating Data

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

Aggregating data

- Group functions operate on sets of rows to give one result per group



EMPLOYEE_ID	SALARY
100	24000
101	17000
102	17000
103	9000
104	6000
...	...

MAX(SALARY)
24000

Aggregate Functions

Aggregate functions

- `COUNT(*)` – 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

```
SELECT AVG(SALARY), MAX(SALARY),
  MIN(SALARY), SUM(SALARY)
FROM EMPLOYEES
WHERE JOB_ID LIKE '%REP%'
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8272.72	11500	6000	273000

Aggregate functions

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

```
SELECT MIN(HIRE_DATE), MAX(HIRE_DATE)
FROM EMPLOYEES
```

MIN(HIRE_DATE)	MAX(HIRE_DATE)
17-JUN-1987	29-JAN-00

- Displaying the first and last employee's name in alphabetical order:

```
SELECT MIN(LAST_NAME), MAX(LAST_NAME)
FROM EMPLOYEES
```

Aggregate functions

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

```
SELECT COUNT(*) FROM EMPLOYEES
WHERE DEPARTMENT_ID = 50
```

COUNT(*)
5

- `COUNT(`*`expr`*`)` returns the number of rows with non-null values for the *expr*

```
SELECT COUNT(COMMISSION_PCT)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 80
```

COUNT(COMMISION_PCT)
3

- `COUNT(DISTINCT` *`expr`*`)` returns the number of distinct non-null values

Aggregate functions

- Group functions ignore all `null` values in the column

```
SELECT AVG(COMMISSION_PCT) FROM EMPLOYEES
```

AVG(COMMISSION_PCT)
.2229

- If each null value in `COMMISSION_PCT` is considered as `0` and is included in the calculation, the result would be `0.0425`

Aggregate functions

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
...	...

11300

23400

16500

20300

Grouping Data

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

Aggregate functions

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

- Example of grouping data:

```
SELECT DEPARTMENT_ID, SUM(SALARY)
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID
```

DEPARTMENT_ID	SUM(SALARY)
100	51600
30	24900
(null)	7000
...	...

- The `GROUP BY` column is not required to be in the `SELECT` list

Aggregate functions

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
20	AD_ASST	4400
20	MK_MAN	13000
20	MK_MAN	12000
30	PU_CLERK	2500
30	PU_CLERK	2500
30	PU_CLERK	2500
30	PU_MAN	11000
30	PU_MAN	11500
30	PU_MAN	10000
30	PU_MAN	11000
...	...	...

4400

25000

7500

43500

Grouping Data Using Several Columns

EMPLOYEES

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

Aggregate functions

- Example of grouping data by several columns:

```
SELECT DEPARTMENT_ID, JOB_ID,
  COUNT(EMPLOYEE_ID), SUM(SALARY)
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID, JOB_ID
ORDER BY SUM(SALARY) DESC
```

DEPARTMENT_ID	JOB_ID	COUNT(EMPLOYEE_ID)	SUM(SALARY)
80	SA_REP	29	243500
50	SH_CLERK	20	64300
80	SA_MAN	5	61000
...	...	...	...

Aggregate functions

- This `SELECT` statement is illegal

```
SELECT DEPARTMENT_ID, COUNT(LAST_NAME)
FROM EMPLOYEES
```

  - Can not combine columns with groups functions unless when using `GROUP BY`

- This `SELECT` statement is also illegal

```
SELECT DEPARTMENT_ID, AVG(SALARY)
FROM EMPLOYEES
WHERE AVG(SALARY) > 8000
GROUP BY DEPARTMENT_ID;
```

  - Can not use `WHERE` for group functions

Aggregate functions

- `HAVING` works exactly like `WHERE` but is used for the grouping functions

```
SELECT DEPARTMENT_ID,
  COUNT(EMPLOYEE_ID), AVG(SALARY)
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID
HAVING COUNT(EMPLOYEE_ID) BETWEEN 3 AND 6
```

DEPARTMENT_ID	COUNT(EMPLOYEE_ID)	AVG(SALARY)
100	6	8600
30	6	4150
90	3	19333.33
60	5	5760

Aggregate functions

- We can apply grouping function from joined tables

```
SELECT COUNT(*) AS EMPS, DEPARTMENT_NAME
FROM EMPLOYEES E JOIN DEPARTMENTS D
  ON E.DEPARTMENT_ID=D.DEPARTMENT_ID
WHERE
  HIRE_DATE BETWEEN '1991-1-1' AND '1997-12-31'
GROUP BY DEPARTMENT_NAME
HAVING COUNT(*) > 5
ORDER BY EMPS DESC
```

EMPS	DEPARTMENT_NAME
19	Shipping
15	Sales

Aggregate functions

- All selected columns should stay in the `GROUP BY` statement or in a group function

```
SELECT D.DEPARTMENT_ID, D.DEPARTMENT_NAME, L.CITY,
  MAX(E.SALARY) AS MAX_SALARY
FROM EMPLOYEES E JOIN DEPARTMENTS D
  ON E.DEPARTMENT_ID=D.DEPARTMENT_ID
  JOIN LOCATIONS L
  ON L.LOCATION_ID = D.LOCATION_ID
GROUP BY D.DEPARTMENT_ID, D.DEPARTMENT_NAME,
  L.LOCATION_ID, L.CITY
ORDER BY MAX_SALARY DESC
```

DEPARTMENT_ID	DEPARTMEN_NAME	CITY	MAX_SALARY
90	Executive	Seattle	24000
80	Sales	Oxford	14000

MySQL Functions

MySQL Functions

- MySQL functions can be used in SQL queries and stored programs

- MySQL functions provide additional means for manipulating strings, numbers, dates, control flow, comparison and conversion between the different types of data

MySQL Functions

- String manipulation 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 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 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
  - ORD() - returns character code for leftmost character of the argument
  - QUOTE() - escape the argument for use in an SQL statement

MySQL 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);
SELECT ORD('a');
SELECT ORD('bbc');
SELECT QUOTE('Value');
```

MySQL 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
  - STRCMP() - compares two strings
  - REVERSE() - reverses a string

MySQL Functions

- Examples:

```
SELECT SPACE(7);
SELECT SUBSTR('Value', 2,3);
SELECT SUBSTR('Value', 2);
SELECT SUBSTR('Value', -2);
SELECT REPLACE('Value', 'Val', 'h');
SELECT STRCMP('b', 'a');
SELECT STRCMP('a', 'b');
SELECT STRCMP('a', 'a');
SELECT REVERSE('Value');
```

MySQL Functions

- Numerical functions include:
  - ABS()
  - CEIL()
  - EXP()/POW()
  - DIV()
  - MOD()
  - ROUND()
  - TRUNCATE()
  - SQRT();

MySQL Functions

- Examples:

```
SELECT ABS(-123.567);
SELECT CEIL(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 Functions

- Date functions include:
  - ADDDATE()
  - ADDTIME()
  - YEAR()
  - CURDATE()

MySQL Functions

- Examples:

```
SELECT ADDDATE('2014-01-01', INTERVAL 40 DAY);
SELECT CAST(ADDTIME('2007-12-31
23:59:59.999999', '1 1:1:1.000002') AS CHAR);
SELECT CURDATE();
SELECT YEAR('2011-01-01');
```

MySQL Functions

- Date functions include:
  - DATE_FORMAT()
  - DAY()/DAYOFMONTH()
  - EXTRACT()
  - MONTH()
  - MINUTE()
  - HOUR();

MySQL Functions

- Examples:

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

MySQL Functions

- Functions that handle NULL values:
  - COALESCE()
  - ISNULL();

MySQL Functions

- Examples:

```
SELECT COALESCE(NULL, NULL, 'VALUE1', 'VALUE2');
SELECT ISNULL('VALUE');
```

MySQL Functions

- Comparison functions:
  - GREATEST()
  - LEAST()

MySQL Functions

- Examples:

```
select least('a', 'b', 'c');
select greatest('a', 'b', 'c');
```

MySQL Functions

- Control flow functions:
  - IF()
  - IFNULL()
  - NULLIF()
- CASE operator

MySQL Functions

- Examples:

```
SELECT IF(MANAGERID IS NULL,'(no manager)',
MANAGERID) FROM EMPLOYEES;

SELECT IFNULL(MANAGERID, '(no manager)') FROM
EMPLOYEES;

SELECT NULLIF('1', '2');
```

MySQL Functions

- Examples:

```
SELECT
    CASE
        WHEN NAME LIKE 'I%' THEN 'I-named'
        WHEN NAME LIKE 'M%' THEN 'M-named'
        ELSE NAME
    END
FROM EMPLOYEES;
SELECT
    CASE SALARY
        WHEN 500 THEN 'LOW'
        WHEN 1000 THEN 'MIDDLE'
        WHEN 2000 THEN 'HIGH'
        ELSE SALARY
    END
FROM EMPLOYEES;
```

MySQL Functions

- Type conversion functions:
  - CAST()
  - FORMAT()
  - STR_TO_DATE()

MySQL Functions

- Examples:

```
SELECT CAST(123 AS CHAR);
SELECT FORMAT(123.55, 1);
SELECT FORMAT(123.55, 0);
SELECT STR_TO_DATE('2011-11-11', '%Y-%m-%d');
```

MySQL Functions

- Functions for working with regular expressions:
  - REGEXP/RLIKE

MySQL Functions

- Examples:

```
SELECT 'a123a' REGEXP '^[0-9]{1,}';
SELECT 'a123a' RLIKE '\[[:digit:]]{1,}';
```

Questions ?

Exercises (1)

1. Write an SQL query to find the count of all employees in each department. Display the name and number of employees for each department.
2. *Write an SQL query to find for each department and for each manager the count of all corresponding employees.*
3. Write an SQL query to find all managers that have exactly 3 employees. Display their names and the name and their department.
4. Write an SQL query to find the total number of employees that have more than two certificates.
5. Write an SQL query to find for each department and for each job title the total number of employees.

Exercises (2)

6. Write an SQL query to find the names and salaries of the employees that take the minimal salary in the company. Use nested `SELECT` statement.
7. Write an SQL query to find the names and salaries of the employees that get a salary that is up to 10% higher than the minimal salary for the company.
8. Write an SQL that displays all departments and the highest salary for each department along with the name of the employee that takes it. If multiple employees in the same department have highest salary, display the first of them.
9. Write an SQL query to find the names of all employees whose name is exactly 14 characters long.

.

Exercises (3)

10. Write an SQL query to find the names of all employees whose first name and last name start with the same letter.
11. Display all departments names and their manager's name. For departments without manager display "(No manager)".
12. Display all employees along with their number of directly managed people. For employees not managing anybody display "Just and employee". For employees managing only 1 employee display "Junior manager". Use `DECODE()` function.
13. Write a SQL query to print the current date and time in the format " hour:minutes:seconds day-month-year". Display also the date coming after a week.