SQL

Aggregerende functies

Luc Vervoort

### Herinner je volgende tips

-- Bij het begin van een nieuwe sessie:

```
USE db_foundations;
```

- -- Kopieer de tekst van de slides in WorkBench
- -- en werk daarin de opdrachten uit.

- -- Bewaar het script dat je maakt als studiemateriaal voor later
- -- (menu File > Save Script As...).

# Aggregerende functie: SUM()

```
SELECT *
FROM employees;

SELECT sum(salary)
FROM employees; -- een aggregerende functie aggregeert gegevens van
-- verschillende rijen tot één resultaat

sum(salary)

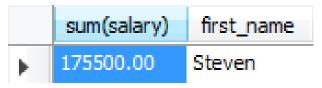
175500.00
```

-- Merk op: er mag geen spatie staan tussen functienaam en openend haakje:

```
SELECT sum (salary)
FROM employees; -- foutmelding
```

## Opgelet: aggregerende functies in combinatie met kolomnamen

```
SELECT sum(salary), first_name -- Is dit het loon van Steven? Neen!
FROM employees; -- MISLEIDENDE RESULTATENLIJST!
```



- -- SQL-standaard verbiedt het gebruik van aggrerende functies IN COMBINATIE MET kolomnamen in de SELECT-clausule (behalve bij groeperen (zie later))
- -- dus: in de SELECT-clausule staan OFWEL enkel kolomnamen, OFWEL enkel aggregerende functies (behalve bij groeperen (zie later))
- -- Opgelet: MySQL geeft hierbij geen foutmelding!

## Wordt vaak gecombineerd met een kolomalias

SELECT sum(salary) AS 'Totale loonkost'

-- kolomalias (enkelvoudig aanhalingstekens)

FROM employees;

SELECT sum(salary) 'Totale loonkost'

-- kolomalias zonder "AS"

FROM employees;

#### -- OPDRACHT:

-- Bereken de totale loonkost van departement 80.

-- antwoord: 301000

## Meer aggregerende functies: SUM() - AVG() - MIN() - MAX() - COUNT()

- -- Pas volgende aggregerende functies toe op departement 80
- -- en formuleer daarbij telkens een passende kolomalias:
- -- SUM(salary) -- zie vorige dia
- -- AVG(salary)
- -- MIN(salary)
- -- MAX(salary)
- -- COUNT(salary)
- -- COUNT(department\_id)
- -- COUNT(DISTINCT department\_id)
- -- SUM(salary), AVG(salary), MIN(salary), MAX(salary) en COUNT(salary) naast elkaar in één SELECT-statement.

## COUNT(\*)

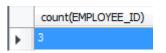
-- Om het aantal rijen van een resultatenlijst te kennen, moet je COUNT() toepassen op een kolom waarvan je zeker weet dat ze geen NULL-waarden bevat (in de resultatenlijst). -- Om het aantal rijen van een resultatenlijst te kennen, kan je ook COUNT(\*) toepassen, zonder je te bekommeren om een kolomkeuze:

-- Vergelijk:

select COUNT(EMPLOYEE\_ID)

from employees

where department\_id = 90;

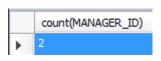


-- met:

select COUNT(MANAGER\_ID)

from employees

where department\_id = 90;



select COUNT(\*)

from employees

where department\_id = 90;



### AVG() with NULL-values

SELECT **AVG**(commission\_pct) FROM employees;

- -- Rijen met null-waarde in deze kolom
- -- worden niet in aanmerking genomen voor de berekening van het gemiddelde.
- -- Stemt dus overeen met:

SELECT **SUM**(commission\_pct)/**COUNT**(commission\_pct) FROM employees;

#### NIET TOEGELATEN in WHERE-clausule

- -- aggrerende functies zijn NIET TOEGELATEN in WHERE-clausule
- -- foutmelding

```
SELECT *
```

FROM employees

```
WHERE salary > avg(salary); -- NIET TOEGELATEN
```