SQL

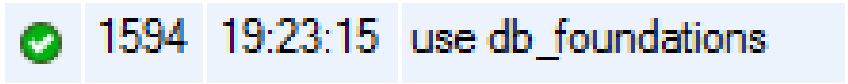
SELECT-instructie

Inleiding

Damien Decorte

Instructies uitvoeren in MySQL WorkBench

-- typ
use db_foundations;
-- plaats de cursor in/achter “ use db_foundations ; ” en druk CTRL+ENTER.
-- bevestiging van correcte uitvoering (groen icon) onderaan in WorkBench:



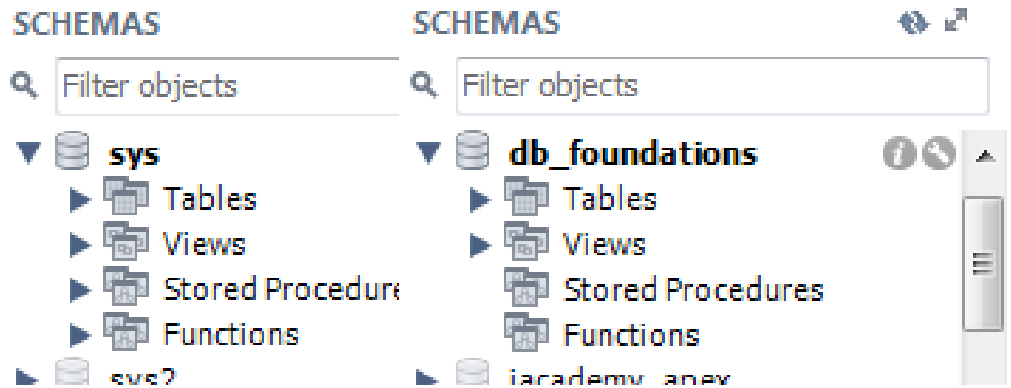
-- typ (met een typfout)
uuuuse db_foundations;
-- plaats de cursor in/achter “ uuuuse db_foundations ; ” en druk CTRL+ENTER.
-- melding van foute uitvoering (rood icon) onderaan in WorkBench:

-- Plaats de cursor achtereenvolgens in/achter de eerste en in/achter de tweede instructie
-- en druk telkens CTRL+ENTER.
-- Zo kan je in een script naar wens instructies afzonderlijk uitvoeren.

Activeer het schema (de databank) waarin je instructies wil uitvoeren

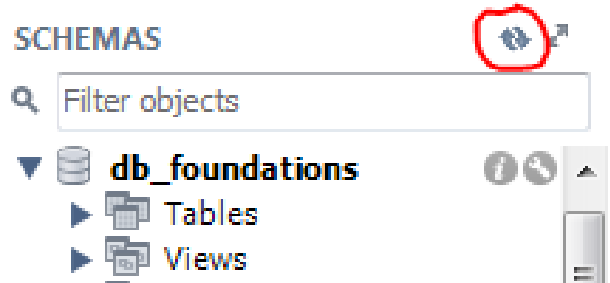
-- Voer volgende instructies één per één uit:

```
use sys ;
use db_foundations ;
use sys ;
USE DB_Foundations ;  --  hoofdlettergevoelig?
```



-- Merk op dat elke SQL-instructie wordt afgesloten met een puntkomma.

-- De schema-lijst in WorkBench wordt niet automatisch geüpdatet.
-- Handmatig updaten kan met een klik op de refresh-knop:



Commentaar invoegen in SQL-code commentaarregel

-- Dit is een commentaarregel.
-- Bij de uitvoering van de SQL-code word commentaar niet uitgevoerd.
-- Een commentaarregel wordt bekomen door te beginnen met **twee koppeltekens en een spatie.**

-- Commentaar kan ook halverwege een regel beginnen, zoals bijvoorbeeld in:

```
USE DB_Foundations ;        --  hoofdlettergevoelig?
```

-- OPDRACHT: maak zelf een commentaarregel

-- OPDRACHT: test eens wat er gebeurt als je de spatie na de twee koppeltekens vergeet.

Commentaar invoegen in SQL-code commentaarblok

```
/*
Dit is een commentaarblok.
Het begint met slash + ster
en eindig met ster + slash.
*/

 -- OPDRACHT: maak zelf een commentaarblok
```

Eerste SELECT-instructie (of SELECT-statement)

-- voorbeeld: een resultatenlijst met de inhoud van alle kolommen en alle rijen (de volledige inhoud, dus)
-- uit de tabel "employees" :

```
SELECT *
FROM employees ;
```

-- Merk op, onderaan in WorkBench verschijnt het aantal rijen in de resultatenlijst: “**20 rows returned**”

1608	19:53:22	SELECT *	FROM employees	20 row(s) returned

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBE	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	BONUS
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	24000.00	NULL	NULL	90	NULL
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21	AD_VP	17000.00	NULL	100	90	NULL
102	Lex	De Haan	LDEHAAN	515.123.4569	1993-01-13	AD_VP	17000.00	NULL	100	90	NULL
103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03	IT_PROG	9000.00	NULL	102	60	NULL
104	Bruce	Ernst	BERNST	590.423.4568	1991-05-21	IT_PROG	6000.00	NULL	103	60	NULL
107	Diana	Lorentz	DLORENTZ	590.423.5567	1999-02-07	IT_PROG	4200.00	NULL	103	60	NULL
124	Kevin	Mourgos	KMOURGOS	650.123.5234	1999-11-16	ST_MAN	5800.00	NULL	100	50	NULL
141	Trenna	Rajs	TRAJS	650.121.8009	1995-10-17	ST_CLERK	3500.00	NULL	124	50	NULL
142	Curtis	Davies	CDAVIES	650.121.2994	1997-01-29	ST_CLERK	3100.00	NULL	124	50	NULL
143	Randall	Matos	RMATOS	650.121.2874	1998-03-15	ST_CLERK	2600.00	NULL	124	50	NULL
144	Peter	Vargas	PVARGAS	650.121.2004	1998-07-09	ST_CLERK	2500.00	NULL	124	50	NULL

Eerste SELECT-instructie (of SELECT-statement)

-- voorbeeld: een resultatenlijst met de inhoud van alle kolommen en alle rijen (de volledige inhoud, dus)
-- uit de tabel "employees" :

```
SELECT *
FROM employees ;
```

-- merk op: we beginnen elke clausule (SELECT, FROM, ...) op een afzonderlijke regel
-- onderaan in WorkBench lees je af hoeveel rijen er voorkomen in de resultatenlijst: “20 rows returned”

-- OPDRACHT: Ga na: zijn de instructies HOOFDLETTERGEVOELIG ?

-- OPDRACHT: Ga na: zijn de instructies SPATIE- of END-OF-LINE-gevoelig ?

Vooruitblik
SELECT-instructie: algemene structuur

-- De verschillende clausules van de SELECT-instructie zullen worden geïntroduceerd in wat volgt.

-- Deze clausules zijn:

```
SELECT …
FROM …
WHERE …
GROUP BY …
   HAVING …
ORDER BY …
```

-- Om de leesbaarheid te bevorderen, noteren we elke clausule op een nieuwe regel.

Denk aan SELECT *
FROM employees;

/*OPDRACHT:

maak in achtereenvolgende instructies telkens een resultatenlijst

met de volledige inhoud van een van de volgende tabellen:

employees

departments

locations

countries

regions

jobs

job_history

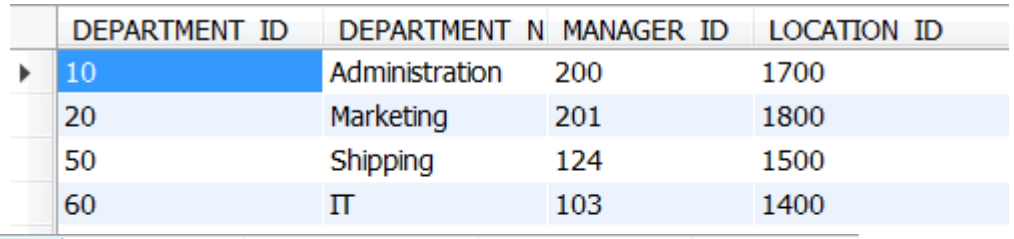
job_grades

*/

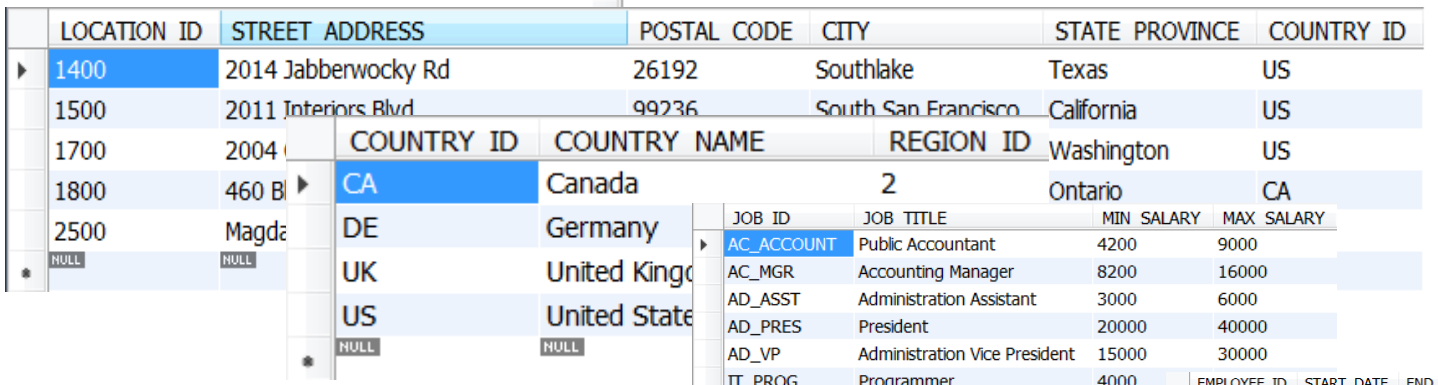
-- Bestudeer de kolommen en de gegevens in elk van de tabellen.

-- We zullen er nog vaak gebruik van maken. Onze oefeningen zullen gebaseerd zijn op deze tabellen.

-- Plak in een word-document een screenshot van deze 7 resultatenlijsten. Dit overzicht zal jou nog handig van pas komen.



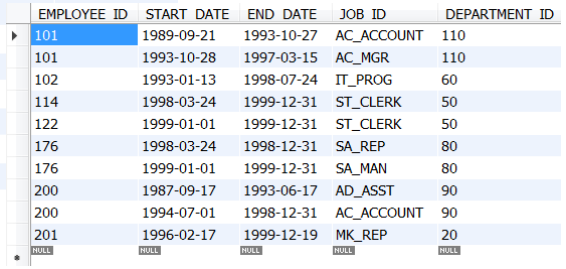
DEPARTMENT_ID	DEPARTMENT_N	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400



LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1700	2004			Washington	US
1800	460 B			Ontario	CA
2500	Magda				
NULL	NULL				

COUNTRY_ID	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	
UK	United Kingd	
US	United State	
NULL	NULL	

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
AC_ACCOUNT	Public Accountant	4200	9000
AC_MGR	Accounting Manager	8200	16000
AD_ASST	Administration Assistant	3000	6000
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
IT_PROG	Programmer	4000	
MK_MAN	Marketing Manager	9000	
MK_REP	Marketing Representative	4000	
SA_MAN	Sales Manager	10000	
SA_REP	Sales Representative	6000	
ST_CLERK	Stock Clerk	2000	
ST_MAN	Stock Manager	5500	
NULL	NULL	NULL	



EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
101	1989-09-21	1993-10-27	AC_ACCOUNT	110
101	1993-10-28	1997-03-15	AC_MGR	110
102	1993-01-13	1998-07-24	IT_PROG	60
114	1998-03-24	1999-12-31	ST_CLERK	50
122	1999-01-01	1999-12-31	ST_CLERK	50
176	1998-03-24	1998-12-31	SA_REP	80
176	1999-01-01	1999-12-31	SA_MAN	80
200	1987-09-17	1993-06-17	AD_ASST	90
200	1994-07-01	1998-12-31	AC_ACCOUNT	90
201	1996-02-17	1999-12-19	MK_REP	20
NULL	NULL	NULL	NULL	NULL

Tip 1: Kopieer de tekst van de slides in WorkBench en werk daarin de opdrachten uit.

```
/*OPDRACHT:
maak in achtereenvolgende instructies telkens een resultatenlijst
met de volledige inhoud van een van de volgende tabellen:
  employees
  departments
  locations
  countries
  regions
  jobs
  job_history
  job_grades
*/

-- Bestudeer de kolommen en de gegevens in elk van de tabellen.
-- We zullen er nog vaak gebruik van maken.
```

Tip 2: Bewaar het script dat je maakt als studiemateriaal voor later

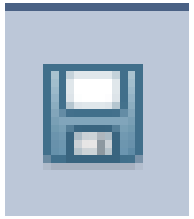
-- Maak een folder voor alle SQL-scripts van het vak databanken,

-- zodat je alles gemakkelijk terugvindt.

-- Sla je script op in die folder: menu File > Save Script As…

-- Bewaar daarna regelmatig de tussentijdse toestand van je script

-- (menu File > Save Script of druk CTRL+S of klik op knop “Opslaan”).



De SELECT-clausule: selecteer kolommen

-- voorbeeld: een resultatenlijst met de inhoud van
-- BEPAALDE kolommen en alle rijen uit de tabel "employees"

```
SELECT  FIRST_NAME, LAST_NAME
FROM    employees ;
```

FIRST_NAME	LAST_NAME
Ellen	Abel
Curtis	Davies
Lex	De Haan
Bruce	Ernst
Pat	Fay
William	Gietz
Kimberely	Grant
Michael	Hartstein
Shelley	Higgins
Alexander	Hunold
Steven	King
Neena	Kochhar
Diana	Lorentz
Randall	Matos

-- OPDRACHT: maak een resultatenlijst met (van links naar rechts)
-- telefoonnummer, naam en voornaam van alle medewerkers in tabel "employees"

De SELECT-clausule: rekenen met kolommen

-- voorbeelden:

```
SELECT last_name, salary, 12*salary+1000
FROM employees;

SELECT last_name, salary, 12*(salary+1000)
FROM employees;
```

-- OPDRACHT:
-- Hoe genereer je volgende resultatenlijst ?

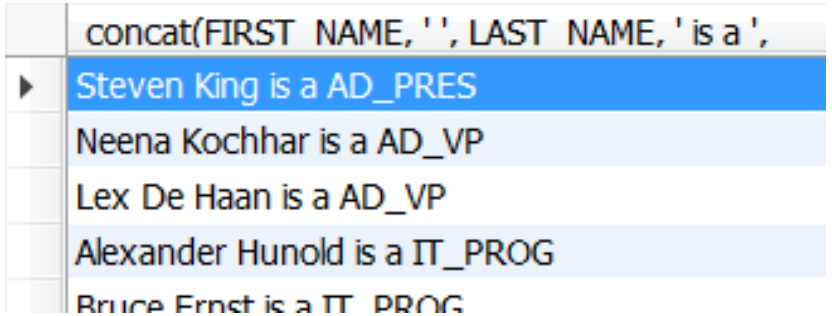
last_name	salary	12*salary+1000	12*(salary+1000)
King	24000.00	289000.00	300000.00
Kochhar	17000.00	205000.00	216000.00
De Haan	17000.00	205000.00	216000.00
Hunold	9000.00	109000.00	120000.00
Ernst	6000.00	73000.00	84000.00
Lorentz	4200.00	51400.00	62400.00

De SELECT-clausule: rekenen met kolommen

-- voorbeeld:

```
SELECT concat(FIRST_NAME, ' ', LAST_NAME, ' is a ', job_id)
FROM employees;
```

-- geeft een vreemd ogend kolom-label:



concat(FIRST_NAME, ' ', LAST_NAME, ' is a ',
Steven King is a AD_PRES
Neena Kochhar is a AD_VP
Lex De Haan is a AD_VP
Alexander Hunold is a IT_PROG
Bruce Ernst is a IT_PROG

De SELECT-clausule: kolom-aliassen

-- voorbeelden: aangepast kolomlabel bekom je als volgt

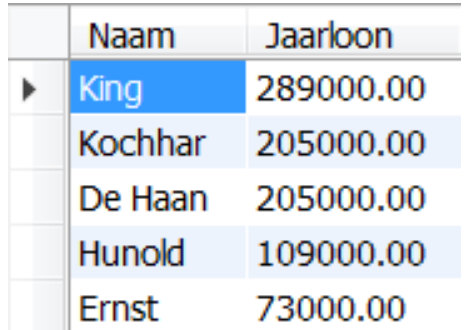
```
SELECT concat(FIRST_NAME, ' ', LAST_NAME, ' is a ', job_id) AS "Who is who ?"    -- met AS
FROM employees;

SELECT concat(FIRST_NAME, ' ', LAST_NAME, ' is a ', job_id) "Who is who ?"      -- zonder AS
FROM employees;
```

-- Met of zonder AS: je mag dus kiezen.

-- OPDRACHT:
-- resultatenlijst met achternaam (label “Naam”) en jaarloon (12 x salaris + 1000, label “Jaarloon”) van elke medewerker.
-- Vertrek daarvoor van:

```
SELECT last_name, 12*salary+1000
FROM employees;
```



Naam	Jaarloon
King	289000.00
Kochhar	205000.00
De Haan	205000.00
Hunold	109000.00
Ernst	73000.00

De SELECT-clausule: SELECT DISTINCT

-- voorbeeld: met of zonder duplicate rijen in de resultatenlijst ?

```
SELECT JOB_ID, DEPARTMENT_ID
FROM employees ;                -- 20 rows returned
```

JOB_ID	DEPARTMENT_ID
AD_PRES	90
AD_VP	90
AD_VP	90
IT_PROG	60
IT_PROG	60
IT_PROG	60
ST_MAN	50

```
SELECT DISTINCT JOB_ID, DEPARTMENT_ID
FROM employees ;                -- 13 rows returned
```

JOB_ID	DEPARTMENT_ID
AD_PRES	90
AD_VP	90
IT_PROG	60
ST_MAN	50

-- In de praktijk zijn duplicate rijen in de output vaak overbodig of storend.
-- Stel je dus steeds de vraag of het gebruik van DISTINCT niet aangewezen is.

-- OPDRACHT: maak een lijst van alle manager-id's die voorkomen in tabel "employees“ (9 rows returned)

De WHERE-clausule

-- we selecteren (alle kolommen van) niet ALLE rijen, maar BEPAALDE RIJEN

```
SELECT      *
FROM        employees
WHERE       Department_ID = 60 ;
```

-- Hoe selecteren we bepaalde KOLOMMEN van bepaalde RIJEN?

-- OPDRACHT: maak een resultatenlijst met
-- (departements)naam en (departements)manager_id van alle departementen met location_id 1700
-- 4 rows returned

department_NAME	Manager_id
Administration	200
Executive	100
Accounting	205
Contracting	NULL

-- merk op dat we hierbij de Location_ID (1700) niet per se hoeven te laten verschijnen in de resultatenlijst.

De WHERE-clausule
de vergelijkings-operatoren

/*

De vergelijkings-operatoren zijn (in deze volgorde van voorrang):

=, <> ,<, <=, >, >=

IS (NOT) NULL, (NOT) LIKE, (NOT) IN

(NOT) BETWEEN...AND...

Ze worden een voor een behandeld in de volgende slides.

*/

De WHERE-clausule operator =

-- Toegepast op een numeriek veld:

```
SELECT      *
FROM        employees
WHERE       Department_ID = 60 ;
```

-- Quid met TEKST-veld?

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers

-- waarvan de job ‘IT_PROG’ is. (IT_PROG moet tussen enkelvoudige aanhalingstekens) -- 3 rows returned

-- Quid met TIJD-veld?

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers

-- die aangeworven zijn op 16 november 1999. (datum (in MySQL-standaardnotatie) moet tussen enkelvoudige aanhalingstekens) -- 1 row returned

De WHERE-clausule
operator <> (of !=) en operator IS NULL

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers -- 20 rows returned

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers

-- die in departement 60 zitten -- 3 rows returned

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers

-- die NIET in departement 60 zitten (gebruik <> of !=) -- 16 rows returned

-- De rekening klopt niet. Waar is de twintigste rij?

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers

-- waarvan we niet weten in welk departement ze zitten (gebruik IS NULL) -- 1 row returned -- Wat betekent een NULL-value dus? “We weten het niet.”

De WHERE-clausule operatoren < , >

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- die in departement 60 zitten -- 3 rows returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- met een department_id groter dan 60 -- 8 rows returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- met een department_id kleiner dan 60 -- 8 rows returned

-- 3 + 8 + 8 = 19 rijen. Waar is de 20ste rij?
-- NULL is niet gelijk, noch kleiner, noch groter dan 60 !!
-- Een voorwaarde waarin met NULL vergeleken wordt, zal nooit voldaan zijn.

De WHERE-clausule operatoren <= , >=

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers

-- waarvan het salaris minstens 9000 bedraagt -- 8 rows returned

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers

-- waarvan het salaris hoogstens 9000 bedraagt -- 13 rows returned

De WHERE-clausule
operatoren < , > , <= , >= in tekst- en tijdsvelden

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan de first_name alfabetisch na "Peter" komt. -- 5 rows returned

-- OPDRACHT: test wat je krijgt bij

```
select *
from employees
where FIRST_NAME > 'p';
```

-- 7 rows returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- die aangeworven zijn sedert 1999-11-16 -- 2 rows returned

De WHERE-clausule
operatoren IS NULL en IS NOT NULL

-- OPDRACHT: maak een resultatenlijst met
-- voornaam en naam van alle medewerkers
-- waarvan we niet weten of ze een manager hebben -- 1 row returned

-- OPDRACHT: maak een resultatenlijst met
-- voornaam en naam van alle medewerkers
-- die WEL een manager hebben -- 19 rows returned

De WHERE-clausule
operatoren LIKE en NOT LIKE

```
-- twee WILDCARD characters:
--    UNDERSCORE "_" :  precies één willekeurig character
--    PERCENT    "%" :      een willekeurig aantal willekeurige characters
--                          ("willekeurig aantal" kan ook nul of één zijn)


-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan voornaam begint met "k"              -- 2 rows returned
                                                --  hoofdlettergevoelig?
-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan voornaam NIET begint met "k"      -- 18 rows returned
```

De WHERE-clausule
operatoren LIKE en NOT LIKE

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan voornaam eindigt op "r“ -- 3 rows returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan voornaam een "r" bevat -- 8 rows returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan voornaam een "o" heeft op de tweede positie -- 1 row returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan voornaam een "e" heeft op de voorlaatste positie -- 8 rows returned

De WHERE-clausule
operatoren LIKE en NOT LIKE

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- die aangeworven zijn in 1997 -- LIKE werkt dus (enigszins verrassend) ook op een TIJDSVELD
-- datumnotatie “jjjj-mm-dd" is standaardnotatie in MySQL
-- 2 rows returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan manager_id een "2" bevat -- LIKE werkt dus (enigszins verrassend) ook op een NUMERIEK VELD
-- 7 rows returned

De WHERE-clausule
operatoren LIKE en NOT LIKE

-- OPDRACHT: maak een resultatenlijst met
-- Id, naam en job_id van alle medewerkers
-- waarvan job_id "A_" bevat

```
SELECT employee_id, last_name, job_id
FROM employees
WHERE job_id LIKE '%A_%' ;   -- is NIET GOED, want UNDERSCORE wordt hier als een wildcard geïnterpreteerd
```

-- om UNDERSCORE of PERCENT op te nemen als character (niet als wildcard) bij (NOT) LIKE:
-- voorafgaan door BACKSLASH (= ESCAPE CHARACTER): “_” en “\%”
-- ook om BACKSLASH op te nemen als character (niet als escape character) bij (NOT) LIKE:
-- voorafgaan door BACKSLASH (= ESCAPE CHARACTER): “\\ ”

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- waarvan last_name een BACKSLASH bevat -- 0 rows returned

De WHERE-clausule operatoren IN en NOT IN

```
-- VOORBEELD: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- in de departementen 110, 10 en 80.

select *
from employees
where department_id IN (110,10,80);      -- 6 rows returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- die IT_PROG of AD_VP zijn.                -- 5 rows returned

-- OPDRACHT: maak een resultatenlijst met
-- alle gegevens van alle medewerkers
-- die GEEN IT_PROG en GEEN AD_VP zijn.      -- 15 rows returned
```

De WHERE-clausule operatoren (NOT) BETWEEN...AND…

-- VOORBEELD: maak een resultatenlijst met

-- alle gegevens van alle medewerkers

-- die aangeworven zijn tussen 1998-03-15 en 1998-04-01 . -- 2 rows returned

-- BELANGRIJKE VASTSTELLING:

-- '1998-03-15' voldoet ook aan BETWEEN '1998-03-15' AND '1998-04-01‘

-- INCLUSIEF de grenzen, dus !!

-- Operator BETWEEN…AND mag je dus niet interpreteren als "TUSSEN“ (ecxlusief de genzen) !!

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers met

-- een department_id tussen 20 en 90 -- 11 rows returned

-- ("tussen" betekent: grenzen NIET inbegrepen).

-- OPDRACHT: maak een resultatenlijst met

-- alle gegevens van alle medewerkers met

-- een department_id groter dan 90 of kleiner dan 20 -- 3 rows returned