SQL & Oracle Databases - Deel 2

# SELECT statement

**SELECT** [**DISTINCT**] select\_list **FROM** table\_name [**WHERE** condition] [**ORDER** **BY** attribuutnaam|volgnr|alias] [**NULLS** **FIRST**/**LAST**]

#### Logische vergelijkingen

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| > | < | = | >= | <= | != | <> |

#### BETWEEN ... AND

identifier [**NOT**] **BETWEEN** ondergrens **AND** bovengrens

#### LIKE

**SELECT** achternaam,voornaam,geb\_datum **FROM** medewerkers **WHERE** **lower**(achternaam) **LIKE** ‘%r%’;  
//zoekt naar medewerkers met de letter r in hun achternaam

**SELECT** achternaam,voornaam,geb\_datum **FROM** medewerkers **WHERE** **lower**(achternaam) **LIKE** ‘\_\_r%’;  
//zoekt naar medewerkers waarvan de 3de letter van hun achternaam een r is

LIKE maakt gebruik van tekstpatronen dewelke naast gewone letters (hoofdlettergevoelig!) ook jokertekens bevatten.

|  |  |
| --- | --- |
| **%** | 0, 1 of meerdere willekeurige tekens |
| \_ | exact 1 willekeurig teken |

#### AND, OR & NOT

**SELECT** achternaam,salaris,afd\_nr **FROM** medewerkers **WHERE** afd\_nr=3 **AND** salaris>30000;

**SELECT** achternaam,salaris,afd\_nr **FROM** medewerkers **WHERE** **NOT**(afd\_nr=3 **AND** salaris>30000);  
//zoekt naar medewerkers die niet deze combinatie hebben

**SELECT** achternaam,salaris,afd\_nr **FROM** medewerkers **WHERE** afd\_nr=3 **OR** salaris>30000;

**** Haakjes zijn bepalend voor hoe de query uitgevoerd zal worden.

**SELECT** achternaam,salaris,afd\_nr **FROM** medewerkers **WHERE** salaris>30000 **AND** afd\_nr=1 **OR** afd\_nr=3;  
//(salaris>30000 + afd\_nr=1) **OR** afd\_nr=3

**SELECT** achternaam,salaris,afd\_nr **FROM** medewerkers **WHERE** salaris>30000 **AND** (afd\_nr=1 **OR** afd\_nr=3);  
//salaris>30000 + afd\_nr=1 **OR** salaris>30000 + afd\_nr=3

#### IN

**SELECT** achternaam,salaris,afd\_nr **FROM** medewerkers **WHERE** salaris=25000 **OR** salaris=30000 **OR** salaris=40000;

//is hetzelfde als

**SELECT** achternaam,salaris,afd\_nr **FROM** medewerkers **WHERE** salaris **IN**(25000,30000,40000);

#### IS NULL & IS NOT NULL

**SELECT** \* **FROM** opdrachten **WHERE** uren **IS** **NULL**;  
//query waarbij geen uren ingevuld zijn

## Functies

|  |  |  |  |
| --- | --- | --- | --- |
| Tekst functies | | | |
| UPPER |  | INSTR |  |
| LOWER |  | CONCAT |  |
| LENGTH |  | LPAD/RPAD |  |
| SUBSTR |  | TRIM |  |
| Numerieke functies | | | |
| ROUND |  | TRUNC |  |
| MOD |  |  |  |
| Datum functies | | | |
| MONTHS\_BETWEEN |  | LAST\_DAY |  |
| ADD\_MONTHS |  | DATE TRUNC |  |
| NEXT\_DAY |  | DATE ROUND |  |
| Onafhankelijke functies | | | |
| NVL |  | REPLACE |  |
| GREATEST/LEAST |  | NVL2 |  |
| DECODE |  | NULLIF |  |
| CASE |  | COALESCE |  |
| Conversiefuncties | | | |
| TO\_CHAR |  |  |  |
| TO\_DATE |  |  |  |
| TO\_NUMBER |  |  |  |

**SELECT** achternaam naam **LENGTH** achternaam lengte\_naam **FROM** medewerkers;

**SELECT** **LENGTH**(‘Martine De Ridder’) naamlengte **FROM** dual;

**SELECT** substr(achternaam,1,4), achternaam **FROM** medewerkers; //eerste 4 letters

**SELECT** substr(achternaam,5), achternaam **FROM** medewerkers; //vanaf 5de karakter tot einde van de String

#### INSTR

**SELECT** **instr**(achternaam,’o’,2,1), achternaam **FROM** medewerkers;  
//op welke positie in de achternaam komt de letter o het eerst voor, beginnend van de tweede positie van de String.

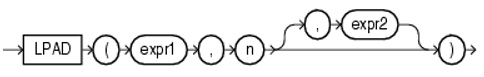
#### CONCAT

**SELECT** **concat**(‘vandaag is ‘,**SYSDATE**) “voorbeeld2 CONCAT” **FROM** dual;

**SELECT** **concat**(‘term1 ‘,**concat**(‘term2 ‘,’term3 ‘)) “voorbeeld3 CONCAT” **FROM** dual; //meerdere termen samenvoegen => CONCAT nesten

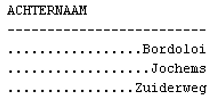
Wanneer de parameters binnen een CONCAT numeriek of datum constanten zijn worden ze eerst geëvalueerd en daarna pas geconverteerd naar een String.

#### LPAD/RPAD

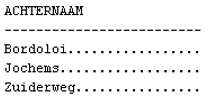


|  |  |
| --- | --- |
| expr1 | *bronstring* |
| n | *lengte van de String dewelke teruggegeven wordt* |
| expr2 | *karakterstring die gebruikt wordt voor het opvullen* |

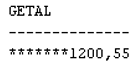
**SELECT** **lpad**(achternaam,25,’.’) achternaam **FROM** medewerkers;



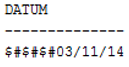
**SELECT** **rpad**(achternaam,25,’.’) achternaam **FROM** medewerkers;



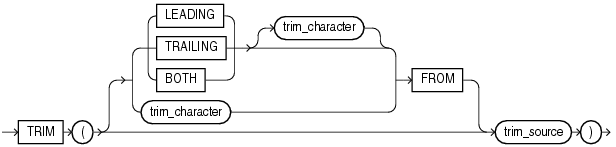
**SELECT** **lpad**(1000+200.55,14,’\*’) getal **FROM** dual;

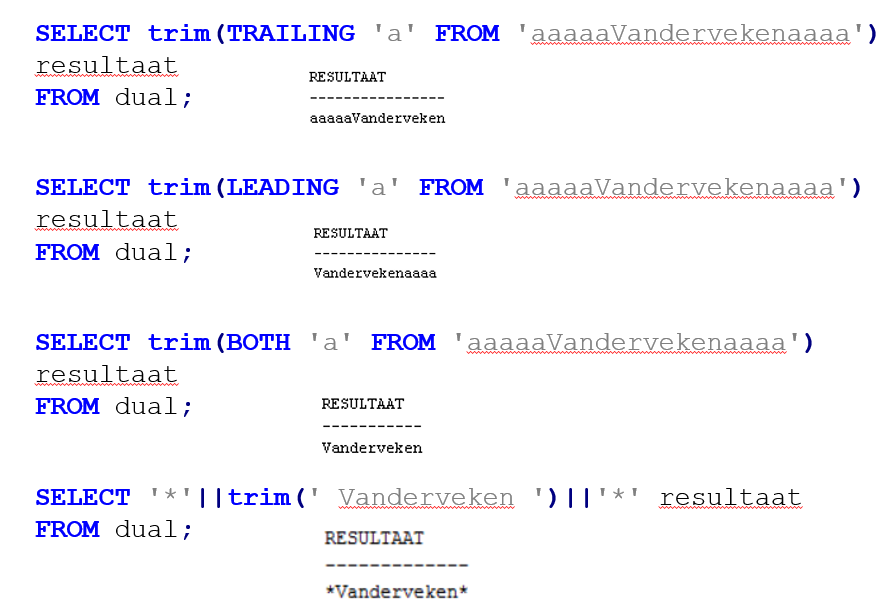


**SELECT** **lpad**(**SYSDATE**,14,’$#’) datum **FROM** dual;



#### TRIM





#### ROUND

**SELECT** **round(**15251.675**)** **FROM** dual**;** //op geheel afgerond 15252

**SELECT** **round(**15251.675**,**1**)** **FROM** dual**;** //op 1 decimaal afgerond 15251,7

**SELECT** **round(**15256.675**,**-1**)** **FROM** dual**;** //afgerond op tiental 15260

#### TRUNC

**SELECT trunc**(15251.675) **FROM** dual**;** //afgekapt op 1 geheel 15251

**SELECT trunc**(15251.675,1) **FROM** dual**;** //afgekapt op 1 decimaal 15251,6

**SELECT trunc**(15251.675,-1) **FROM** dual**;** //afgekapt op tiental 15250

#### MOD

**SELECT mod**(6,3)rest **FROM** dual**;** //0

**SELECT mod**(5,3)rest **FROM** dual**;** //2

## Datumfuncties

SYSDATE of CURRENT\_DATE is systeemdatum, bewerkingen hierop gebeuren met aantal dagen.

**ALTER** **SESSION** **SET** NLS\_DATE\_FORMAT=’dd-mm-yyyy’; //bepaalt voor sessie hoe datum geformatteerd dient te worden, default DD-MON-YYYY

**SELECT** value **FROM** nls\_session\_parameters **WHERE** parameter=’NLS\_DATE\_FORMAT’;

Met NLS\_LANGUAGE kan ook de taal ingesteld worden.

**ALTER** **SESSION** **SET** NLS\_LANGUAGE=’FRENCH’;

**SELECT** **trunc**((**SYSDATE**-geb\_datum)/365) leeftijd **FROM** medewerkers;

#### MONTHS\_BETWEEN

**SELECT trunc(months\_between(SYSDATE,** geb\_datum**)/**12**)** leeftijd **FROM** medewerkers; //maanden afgerond op decimalen

#### ADD\_MONTHS

**SELECT** mgr\_start\_datum **add\_months(**mgr\_start\_datum**),**-4**)** “4 maanden eerder” **FROM** afdelingen;

#### NEXT\_DATE

**SELECT next\_date(SYSDATE**,’zo’**)** voorbeeld **FROM** dual;  
//zoekt eerstvolgende zondag na sysdate

#### LAST\_DATE

**SELECT last\_date(**’11-NOV-2014’**)** laatste\_dag **FROM** dual; //30/11/2014  
//neemt de maand en berekent de laatste dag van die maand

#### ROUND



|  |  |
| --- | --- |
| date | *datum* |
| fmt | *format (bepaalt mate van afronding): CC, YYYY, Q, MM, W, DD, HH, ...* |

**SELECT round(SYSDATE**,’CC’**)** **FROM** dual; //01-01-01 eerste dag van deze eeuw

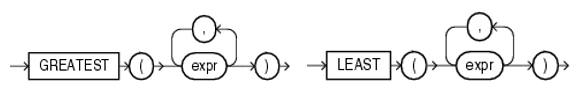
**SELECT round(SYSDATE**,’Q’**)** **FROM** dual; //eerste dag van kwartaal

**** Indien de maand tussen 1 en 6 afronding naar beneden, erboven naar volgend jaar.

#### TRUNC

**SELECT trunc(SYSDATE**,’YYYY’**)** **FROM** dual; //01/01/17  
//datum wordt afgekapt tot de 1ste dag van de 1ste maand van het jaar

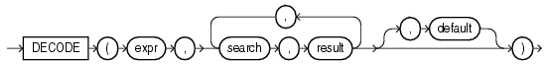
#### GREATEST/LEAST



Geef de grootste/kleinste waarde binnen een reeks we

**SELECT** proj\_nr,uren **greatest(**proj\_nr,uren**)** **FROM** opdrachten;

#### DECODE



Met de DECODE functie kan je een waarde kiezen uit een lijst, als de *expr* overeenkomt met een *search* term dan wordt het *result* weergegeven. Indien dit niet overeenkomst wordt de *default* teruggegeven. Indien geen default wordt er NULL ingevuld.

**SELECT** sofi\_nr, **decode**(provincie,’NB’,’Noord Brabant’,’LI’,’Limburg’) provincie **FROM** medewerkers;

//als de provincie noch NB noch LI is wordt er NULL ingevuld

**SELECT** **decode(**provincie**,**'NB'**,**'Noord Brabant'**,** 'LI'**,**'Limburg',provincie**)** provincie **FROM** medewerkers**;**

//als de provincie noch NB noch LI is wordt de default waarde ingevuld, in dit geval de inhoud van provincie

#### ANSI-SQL alternatief voor DECODE: CASE

**SELECT** sofi\_nr**,** **CASE** **WHEN** provincie**=**'NB' **THEN** 'Noord Brabant' **WHEN** provincie**=**'LI' **THEN** 'Limburg' [**ELSE** provincie] **END** provincie **FROM** medewerkers**;**

//zonder ELSE default NULL, met ELSE default waarde provincie

#### NVL (Null-VaLue substition)

Wordt gebruikt om een NULL waarde tijdelijk te vervangen door een andere waarde (voor het uitzicht van de resultatentabel of voor berekeningen).

**** Hierbij verander je de waarde van het attribuut in de DB niet!

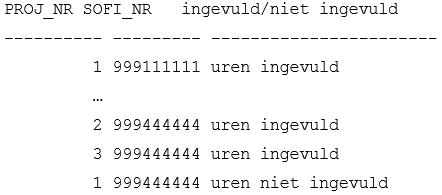
**SELECT** sofi\_nr,proj\_nr,nvl(uren,0)**FROM** opdrachten; //NULL wordt 0

**SELECT** sofi\_nr,proj\_nr,uren + 4 **FROM** opdrachten; //NULL blijft null  
**SELECT** sofi\_nr,proj\_nr,nvl(uren,0) + 4 **FROM** opdrachten; //NULL wordt 4

#### NVL2

Neemt 3 parameters, één als resultaat bij niet NULL en één voor NULL.

**SELECT** opd\_proj\_nr**,** opd\_mdw\_sofi\_nr**, NVL2(**opd\_uren**,**'uren ingevuld'**,**'uren niet ingevuld'**)** "ingevuld/niet ingevuld" **FROM** opdracht**;**



#### COALESCE

**SELECT** **COALESCE**(NULL,NULL,'aaa', NULL) **FROM** dual; //aaa

//geeft de eerste niet NULL waarde uit een lijst

vb.: je wenst contact te nemen met iemand a.d.h.v. deze volgorde:

1. zakelijk telefoonnummer
2. geen zakelijk nummer, maar wel mobiel nummer
3. geen zakelijk en geen mobiel nummer, maar wel huistelefoonnummer

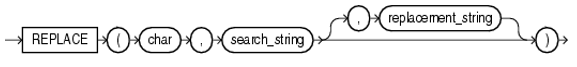
**SELECT** name **COALESCE**(businessphone,cellphone,homephone) contact\_phone **FROM** contact\_info;

#### NULLIF

**SELECT** **nullif(**1234**,**1234**) FROM** dual**;** //NULL omdat gelijk  
**SELECT** **nullif(**1234**,**123**+**4**)** **FROM** dual**;** //1234

//indien niet gelijk eerste parameter

#### REPLACE

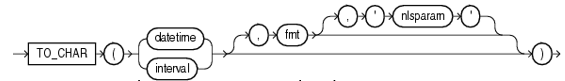


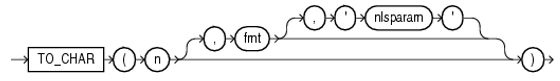
|  |  |
| --- | --- |
| char | *te onderzoeken karakterstring* |
| search\_string | *de karakterstring dewelke men zoekt in char* |
| replacement\_string | *de karakterstring waarmee search\_string vervangen zal worden (if found)* |

**SELECT** **REPLACE(** 'ab\*cd\*ef\*'**,**'\*'**,**'°'**)** "voorbeeld1 replace" **FROM** dual**;**   
//ab°cd°ef°

**SELECT** **REPLACE(** 'ab\*cd\*ef\*'**,**'\*'**)** "voorbeeld2 replace" **FROM** dual**;**   
//abcdef (geen replacement\_string gedefinieerd dus vervangen met niets)

#### TO\_CHAR



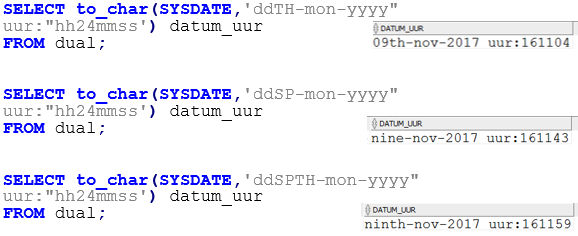


**SELECT** **to\_char(SYSDATE,**'Month'**)||**' is een speciale maand' Voorbeeld **FROM** dual**;** //November is een speciale maand

**SELECT** **to\_char(SYSDATE,**'fmMonth'**) ||**' is een speciale maand' Voorbeeld **FROM** dual**;** //November is een speciale maand (trimt spaties)

**** Namen van dagen en maanden worden automatisch aangevuld met spaties.

Gegeven dat de SYSDATE 9 november 2017 is:



**SELECT** **to\_char(**000001**)||**' is een speciaal getal' voorbeeld **FROM** dual**;**   
//voorloopnullen worden weggelaten => 1 is een speciaal getal

**SELECT** **to\_char(**1234**,**'L9999'**)** **FROM** dual**;** //€1234

#### TO\_DATE

**SELECT** **to\_date(**'03-11'**)** datum **FROM** dual**;** //ERROR

**SELECT** **to\_date(**'11-03-2017'**,**'mm-dd-yyyy'**)** datum **FROM** dual**;** //03/11/17

**SELECT** **to\_date(**'03-11'*,* 'dd-mm'**)** datum **FROM** dual**;** //03/11/17

#### TO\_NUMBER

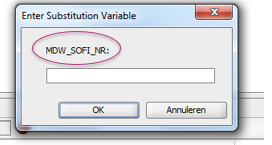
**SELECT** **to\_number(**'$25,000.50'**) FROM** dual**;** //ERROR

**SELECT** **to\_number(**'$25,000.50'*,*'$99,999.99'**)** getal **FROM** dual**;** //25000,5

## Substitutievariabele

Een substitutievariabele maakt deel uit van een script dat op regelmatige tijdstippen gerund zal worden. Wordt gebruikt om bij uitvoering van de instructie een waarde in te geven waarmee de instructie dan zal uitgevoerd worden.

**SELECT** voornaam**,**achternaam**,**afd\_nr**,** geb\_datum **FROM** medewerkers **WHERE** sofi\_nr**=**'&sofi\_nr'**;**



vb.: je moet een medewerker zoeken op basis van sofi\_nr of de achternaam. Indien men het sofi\_nr niet kent kan men bij sofi\_nr bv. een 0 ingeven.

**SELECT** voornaam**,**achternaam**,**afd\_nr**,** geb\_datum **FROM** medewerkers **WHERE** sofi\_nr**=**'&sofi\_nr' **OR** **upper(**achternaam**)=upper(**'&achternaam'**);**

vb.: we willen medewerkers ophalen die zowel in hun voor- als achternaam een ‘e’ hebben.

**SELECT** voornaam**,**achternaam **,**afd\_nr**,**geb\_datum **FROM** medewerkers **WHERE** l**ower(**voornaam**)** **LIKE** **lower(**'%&variabele%'**)** **AND** **lower(**achternaam**)** **LIKE** **lower(**'%&variabele%'**);**

 Men dient twee keer hetzelfde karakter in te geven.

Wanneer men binnen een query meerdere keren naar dezelfde substitutievariabele wil refereren kan men gebruik maken van &&. Deze waarde kan ook later in de sessie gebruikt worden.

**SELECT** voornaam**,**achternaam **,**afd\_nr**,**geb\_datum **FROM** medewerkers **WHERE** **lower(**voornaam**)** **LIKE** **lower(**'%&&variabele%'**)** **AND** **lower(**achternaam**)** **LIKE** **lower(**'%&variabele%'**);**

**UNDEFINE** variabele; //waarde van de variabele ongedaan maken

**DEFINE** variabele = waarde; //waarde toekennen aan een variabele

# GROUP BY clause

## Regular expressions

Een reguliere expressie is een manier om patronen te beschrijven waarmee een PC tekst kan herkennen. Dit is een standaard die door vele tools en programmeertalen ondersteund worden.

|  |  |  |
| --- | --- | --- |
| REGEXP\_LIKE | REGEXP\_INSTR |  |
| REGEXP\_SUBSTR | REGEXP\_REPLACE |  |
| |  |  | | --- | --- | | jokers | | | . | om het even welk teken | | [...] | één van de tekens uit de array | | [m-p] | bereik van waarden van m t.e.m. p | | [^...] | werkt als niet operator, zoekt naar een teken dat niet in de lijst voorkomt | | + | voorgaand teken 1 of meer keer | | \* | voorgaand teken 0 of meer keer | | ? | voorgaand teken 0 of 1 keer | | {n,m} | voorgaand teken n tot m keer | | {m,} | voorgaand teken komt minimum m keer voor | | {m} | voorgaand teken komt exact m keer voor | | abc|def | werkt op Strings, abc of def | | () | bakent de expressie af | | ^ | komt overeen met het begin van een lijn *(^def zoekt naar def in het begin van tekst)* | | $ | komt overeen met het einde van een lijn *(def$ zoekt naar def aan het einde van tekst)* | | \ | zorgt ervoor dat het volgende karakter niet als metakarakter geïnterpreteerd wordt | | |  |

**SELECT** achternaam **FROM** medewerkers **WHERE** **REGEXP\_LIKE(**achternaam**,**'B.r'**);**   
//zoekt naar 1ste letter B, derde letter r => Borderloi

**SELECT** achternaam **FROM** medewerkers **WHERE** **REGEXP\_LIKE(**achternaam**,** '[imu]'**);**   
//achternaam waar de letters i, m of u in voorkomen

**SELECT** achternaam **FROM** medewerkers **WHERE** **REGEXP\_LIKE(**achternaam**,**'o+'**);**   
//zoekt waar ‘o’ 1 of meer keer in voorkomt => Amelsvoort, Joosten

**SELECT** achternaam **FROM** medewerkers **WHERE** **REGEXP\_LIKE(**achternaam**,** '([aeiou][^aeiou]){2}'**);**   
//Zu**ider**weg, Mu**id**en, Pr**eger**s (1ste char, 2de char niet) x 2 keer

#### REGEXP\_LIKE

**SELECT** achternaam **FROM** test **WHERE** **REGEXP\_LIKE**(achternaam,'a{2}');  
//Goovaarts

***SELECT*** *achternaam* ***FROM*** *test* ***WHERE******REGEXP\_LIKE(****achternaam,'Goov(aa|ae)rts',****'****i****')****; // ‘i’ = case-insensitive //Goovaerts/Goovaarts/GOOVAERTS*

#### REGEXP\_LIKE

**SELECT** achternaam**, regexp\_substr(**achternaam**,**'[aeiou]'**)** "eerste klinker"   
**FROM** medewerkers**;**//Bordoloi => o, Zuiderweg => u, Amelsvoort => e (case sensitive!)

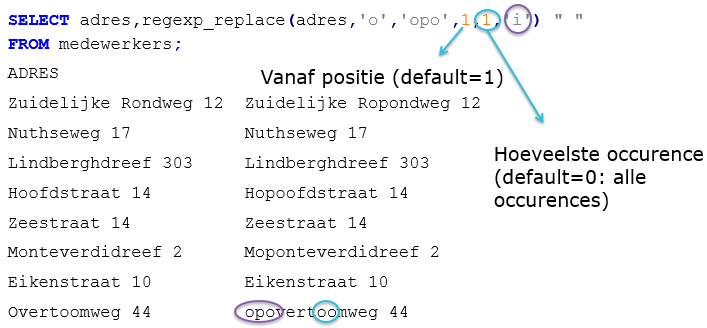
**SELECT**contact\_id,last\_name*,* **REGEXP\_SUBSTR(**last\_name,'[aeiou]',1,2,'i')"tweede klinker" **FROM** CONTACT;  
//vanaf 1ste positie, 2de occurence; Anderson => e, Johnson => o

#### REGEXP\_INSTR

**SELECT** achternaam**, regexp\_instr(**achternaam**,**'en|er'**)** " " **FROM** medewerkers**;**//Bordeloi => 0, Muiden => 5, Joosten => 6

#### REGEXP\_REPLACE

**SELECT** adres,**regexp\_replace(**adres**,** 'o'**,**'opo'**)** " " **FROM** medewerkers**;**



## Analytische functies

|  |
| --- |
| COUNT(\*) |
| COUNT([ALL|DISTINCT] expressie) |
| SUM([ALL|DISTINCT] expressie) |
| AVG([ALL|DISTINCT] expressie) |
| MIN(expressie) |
| MAX(expressie) |
|  |

Expressie kan een kolomnaam, constante, functie, combinatie van ... zijn. Een statische functie geeft per groep telkens 1 rij als resultaat. Deze functies staan telkens in de SELECT list, nooit in de WHERE clausule!

**** Statische functies houden geen rekening met NULL waarden.

#### AVG

**SELECT** **AVG**(salaris) "GEMIDDELD SALARIS" **FROM** medewerkers;

**SELECT** **AVG**(**DISTINCT** salaris) "GEMIDDELD SALARIS" **FROM** medewerkers;  
//het gemiddelde van de verschillende(!) salarissen

#### SUM

**SELECT** **SUM**(salaris) totaal **FROM** medewerkers;

**SELECT** **SUM**(**DISTINCT** salaris) totaal **FROM** medewerkers;  
//de som van de verschillende salarissen

#### MIN/MAX

**SELECT** **MAX**(salaris) "hoogste salaris", **MIN**(salaris) "laagste salaris" **FROM** medewerkers;

**SELECT** **MIN**(achternaam) "eerste", **MAX**(achternaam) "laatste" **FROM** medewerkers; //welke achternaam komt eerst/laatst in het alfabet

#### COUNT

**SELECT** **COUNT**(\*) aantal **FROM** medewerkers; //telt geselecteerde rijen

**SELECT** **COUNT**(mgr\_sofi\_nr) aantal **FROM** medewerkers; //telt enkel ingevulde rijen

**SELECT** **COUNT**(\*) aantal\_afd3 **FROM** medewerkers **WHERE** afd\_nr=3;

## GROUP BY

Rijen groeperen volgens een bepaald criterium en op deze groepjes analytische functies toepassen.

**SELECT** afd\_nr,**COUNT**(\*) aantal **FROM** medewerkers **GROUP** **BY** afd\_nr;

|  |  |
| --- | --- |
|  | *Geeft het aantal medewerkers per afdeling weer.* |

**SELECT** afd\_nr,**COUNT**(\*) aantal (3)  
**FROM** medewerkers (1)  
**GROUP** BY afd\_nr; (2)

|  |  |
| --- | --- |
| (1) | de tabel in het FROM statement wordt aangesproken |
| (2) | de rijen worden gegroepeerd op basis van de groepskenmerken (attributen) |
| (3) | voor elk groepje wordt de SELECT lijst uitgevoerd wat 1 RIJ PER GROEP oplevert |

**** Bij het GROUP BY statement zijn analytische functies, constanten of attributen toegelaten.

**SELECT** afd\_nr,**SUM**(salaris) "jaarlijkse loonkost" (4)  
**FROM** medewerkers (1)  
**WHERE** **LOWER**(geslacht)='v' (2)  
**GROUP** **BY** afd\_nr; (3)

|  |  |
| --- | --- |
| (1) | de tabel in het FROM statement wordt aangesproken |
| (2) | er wordt bepaald welke rijen meespelen |
| (3) | de rijen worden gegroepeerd op basis van de groepskenmerken (attributen) |
| (4) | voor elk groepje wordt de SELECT lijst uitgevoerd wat 1 RIJ PER GROEP oplevert |

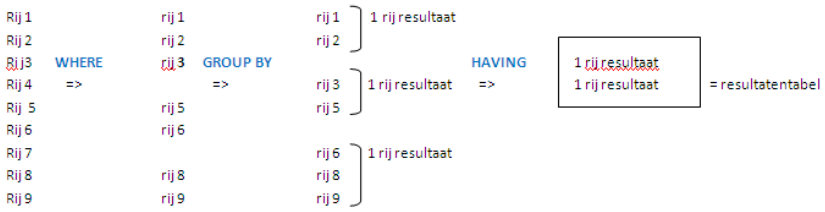
**SELECT** afd\_nr,**SUM**(salaris) TOT\_LOONKOST (4)   
**FROM** medewerkers (1)  
**WHERE** **UPPER**(geslacht)='M' (2)  
**GROUP** **BY** afd\_nr (3)  
**ORDER** **BY** 2 **DESC**; (5)

 Nesten van analytische functies is niet mogelijk!

**~~SELECT~~****~~AVG~~**~~(MAX(salaris))~~ **~~FROM~~** ~~medewerkers;~~ //  syntax fout

## GROUP BY de HAVING clausule

De HAVING clausule heeft het gedrag van de WHERE clausule op rijen uit de tabel maar dan op de bekomen rijen uit het GROUP BY statement.



**SELECT** afd\_nr,**AVG**(salaris) **FROM** medewerkers **GROUP** **BY** afd\_nr **HAVING** **AVG**(salaris)>33000;

**** In tegenstelling tot de WHERE clausule laat de HAVING clausule wel analytische functies toe.

**SELECT** afd\_nr,**AVG**(salaris) gemiddeld\_sal **FROM** medewerkers **GROUP** **BY** afd\_nr **HAVING** **COUNT**(\*)>2;