

BIOINFORMATICA

INFORMATIESYSTE MEN MET PYTHON_

SUBQUERY'S EN
GROEPFUNCTIES

STUDIEWIJZER

Les	Onderwerp	Theorie
1	Introductie tot Informatie Systemen	H1 Introduction to Database Development H2 Entity Relationships H3 Complex Relationships H4 Logical Database Design
2	Database ontwerp Constraints en normaliseren	H5 Normalisation H6 Introduction to Oracle SQL H7 Foreign Keys
3	SQL, operatoren en functies	H8 Selecting data from a Table
4	Rijen ophalen uit meerdere tabellen	H9 Selecting data from multiple tables
5	Subquery's en groepfuncties	H10 Subqueries and Group Functions
6	SQL Embedden in Python Set operatoren	
7	Python en SQL: mogelijkheden en risico's	

AGENDA

Subquery's

Subquery's met groepsfuncties

Gecorreleerde subquery's

Groepen met groepsfuncties

Having by clause

SUBQUERY

Welke studenten zitten in de klas met iemand die in Nijmegen woont?

SUBQUERY

Een subquery is een query in een query

Subquery's maken het mogelijk om tijdens het uitvoeren van de hoofdquery een extra query uit te voeren

Ook wel inner - en outer query

SUBQUERY

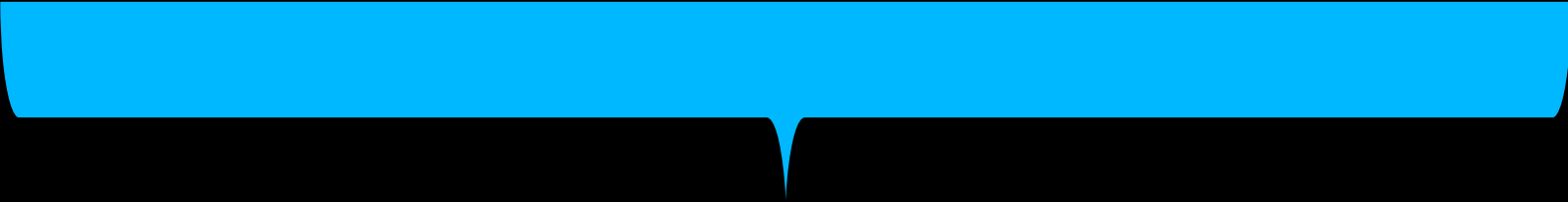
Stel je wilt weten wie zit in de klas met iemand die in Nijmegen woont?

Je zult twee vragen stellen

1. Wie woont er in Nijmegen?
2. Wie zit bij die student in de klas?

1) WIE WOONT ER IN NIJMEGEN?

```
select *  
from student  
where woonplaats = 'Nijmegen';
```



voornaam	tussenvoegsels	achternaam	woonplaats	klas_id
Ernst	<null>	Kuipers	Nijmegen	3

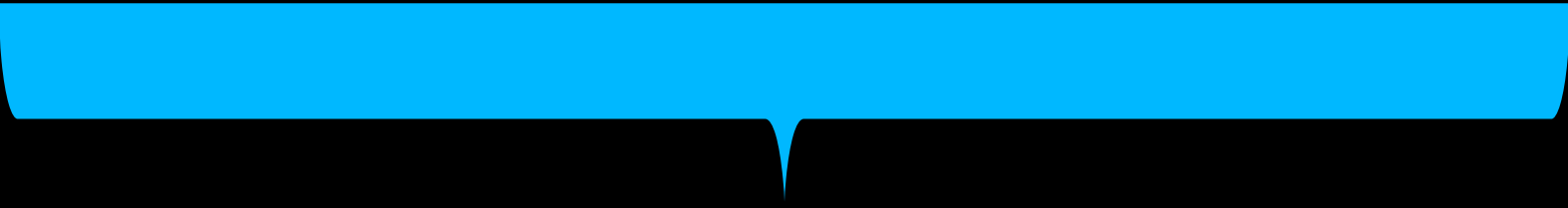
OPHALEN VAN KLAS_ID

Van de rij gegevens wil je alleen het klas_id weten

In dit geval dus `klas_id = 3`

2)WIE ZIT ER BIJ ... IN DE KLAS?

```
select *  
from studenten  
where klas_id = 3
```



voornaam	tussenvoegsels	achternaam	woonplaats	klas_id
Sigrid	<null>	Kaag	Amsterdam	3
Ernst	<null>	Kuipers	Nijmegen	3
Wopke	<null>	Hoekstra	Millingen	3
Eric	van der	Burg	Elst	3
Rob	<null>	Jetten	Utrecht	3
Alexandra	van	Huffelen	Groesbeek	3
Hans	<null>	Vijlbrief	<null>	3

EEN SUBQUERY GEBRUIKEN

Een subquery is een query in een query

Hierdoor kun je in een keer de vraag beantwoorden

2)WIE ZIT ER BIJ ... IN DE KLAS?

```
select *  
from studenten  
where klas_id = (select klas_id  
                 from studenten  
                 where woonplaats='Nijmegen')
```



voornaam	tussenvoegsels	achternaam	woonplaats	klas_id
Sigrid	<null>	Kaag	Amsterdam	3
Ernst	<null>	Kuipers	Nijmegen	3
Wopke	<null>	Hoekstra	Millingen	3
Eric	van der	Burg	Elst	3
Rob	<null>	Jetten	Utrecht	3
Alexandra	van	Huffelen	Groesbeek	3
Hans	<null>	Vijlbrief	<null>	3

WAT IS EEN SUBQUERY?

Een subquery is een query met dezelfde opbouw als een query

```
( SELECT [DISTINCT]
  subquery_select_list
FROM {table_name}
 [WHERE search_conditions]
 [GROUP BY aggregate_expression [,
  aggregate_expression] ...]
 [HAVING search_conditions] )
```

SUBQUERY

Een subquery is een gewone query met alle clauses

Een clause is echter niet mogelijk in een subquery, welke zal dat zijn?

Order by

Subquery kan resultaat niet manipuleren

Order by zal resultaat hoofdquery sorteren

REGELS

Een subquery staat rechts van de **operator** in de where clause

De resultante van de subquery moet **compatibel** zijn met de
expressie links van de subquery

REGELS

Wie zit er bij iemand in de klas die in Utrecht woont?

SUBQUERY'S EN OPERATOREN

Als uit een subquery meerdere resultaten kunnen voortvloeien dan moet de operator dat ondersteunen

Gebruik in dat soort gevallen dus een *in* operator

Net zoals bij gewone query's

```
select *  
from studenten  
where klas_id in (1, 1, 2)
```

WIE ZITTEN ER BIJ IEMAND UIT UTRECHT IN DE KLAS?

```
select *  
from student  
where klas_id = (select klas_id  
                 from student  
                 where woonplaats = 'Utrecht')
```

[21000][1242] Subquery returns more than 1 row

MEERDERE WAARDES SUBQUERY

```
select *  
from student  
where klas_id = (select klas_id  
                 from studenten  
                 where woonplaats='Utrecht')
```

1,3,1,4,1

MEERDERE WAARDES SUBQUERY

```
select *  
from student  
where klas_id in (1,3,1,4,1)
```

De in operator is nodig omdat meerder waardes terug kunnen komen uit de subquery

NESTING

Subquery's mogen zelf ook weer subquery's bevatten die ook weer subquery's mogen bevatten enzovoorts

Dit principe noemen we **nesting**, net zoals dat bij loops en if statements nesting heet

AGENDA

Subquery's

Subquery's met groepsfuncties

Gecorreleerde subquery's

Groepen met groepsfuncties

Having by clause

GROEPSFUNCTIES

In SQL kunnen we groepsfuncties gebruiken

Groepsfuncties zijn functies die over een hele groep een uitspraak doen

GROEPSFUNCTIES

Naam	
sum()	sommering van waardes
count()	tellen van aantal resultaatrijen
min()	minimale waarde in groep
max()	maximale waarde in groep
avg()	gemiddelde waarde van een groep getallen

WAT IS HET GEMIDDELDDE STUDENTENNUMMER?

```
select avg(student_nr)  
from student
```

VOORBEELD GROEPSFUNCTIE

Bepaal wat de geboortedatum is van de jongste student

```
select max(geb_datum)  
from student
```

1997-12-13

WIE HOORT BIJ DEZE DATUM?

```
select max(geb_datum)
from student
```

1999-01-31

```
select *
from student
where geb_datum = '1999-01-31'
```

	student_nr	voornaam	tussenvoegsels	achternaam	geb_datum
1	12	Kim	van	Pleggen	1999-01-31

IN EEN QUERY

```
select max(geb_datum)
from student
```

1999-01-31

```
select *
from student
where geb_datum = (select max(geb_datum)
from student)
```

	student_nr	voornaam	tussenvoegsels	achternaam	geb_datum
1	12	Kim	van	Pleggen	1999-01-31

GROEPSFUNCTIES

Groepsfuncties als **sum**, **count**, **max** en **min** retourneren altijd slechts **1** waarde

Wanneer je deze functies gebruikt kun je dus de **=** operator gebruiken

GROEPSFUNCTIES

De groepsfuncties **avg** en **sum** zijn alleen van toepassing op getallen

count, **max** en **min** kun je ook toepassen op datums en teksten

GROEPSFUNCTIE COUNT

Functie	Resultaat
count(*)	Telt het aantal resultaatrijen
count(geb_datum)	Telt het aantal geboortedatums die niet null zijn
count (distinct woonplaats)	Telt het aantal verschillende woonplaatsen

GROEPSFUNCTIES

Groepsfuncties kun je opnemen in de **select** clausule maar **niet** in de **where** clausule

Als je een vergelijking wilt maken in de **where** clausule met een groepsfunctie dan moet je die opnemen in een subquery

QUERY

Schrijf de query die de studenten toont met een studentnummer lager dan het gemiddelde studentnummer

AGENDA

Subquery's

Subquery's met groepsfuncties

Gecorreleerde subquery's

Groepen met groepsfuncties

Having by clause

GECORRELEERDE SUBQUERY

Een subquery met een relatie naar de hoofdquery noemen we een gecorreleerde subquery

In de vraagstelling is meestal een verwijzing naar een groep waar een rij deel van uitmaakt

VRAAG

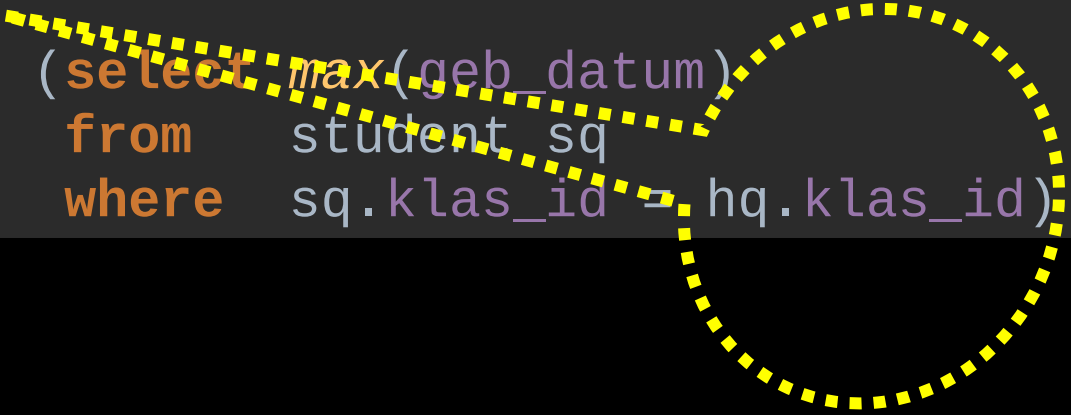
Welke student is van zijn of haar klas de jongste student?

Zijn of haar verwijst naar de klas, dus per student (rij) kijk je naar zijn of haar hele klas

QUERY

Welke student is van zijn of haar klas de jongste student?

```
select voornaam, klas_id
from student hq
where geb_datum = (select max(geb_datum)
                   from student sq
                   where sq.klas_id = hq.klas_id)
```



AGENDA

Subquery's

Subquery's met groepsfuncties

Gecorreleerde subquery's

Groepen met groepsfuncties

Having by clause

GROUP BY

De group by clause maakt het mogelijk om rijen in te delen in groepen

HET SELECT STATEMENT MET GROUP BY

```
select <kolomnamen | *>  
from <tabel>  
group by <kolomnaam>
```



Group by clause maakt een indeling in groepen

GROUP BY

Wat is het aantal studenten per klas?

```
select count(*)  
from studenten  
group by klas_id
```

GROUP BY

Wat is het aantal studenten per klas?

```
select klas_id, count(*)  
from studenten  
group by klas_id
```

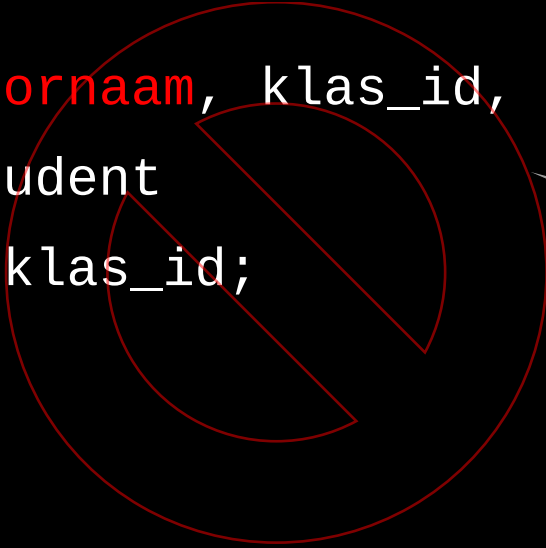
GROUP BY

Wat is het aantal studenten per klas?

Maar met de studentnamen...

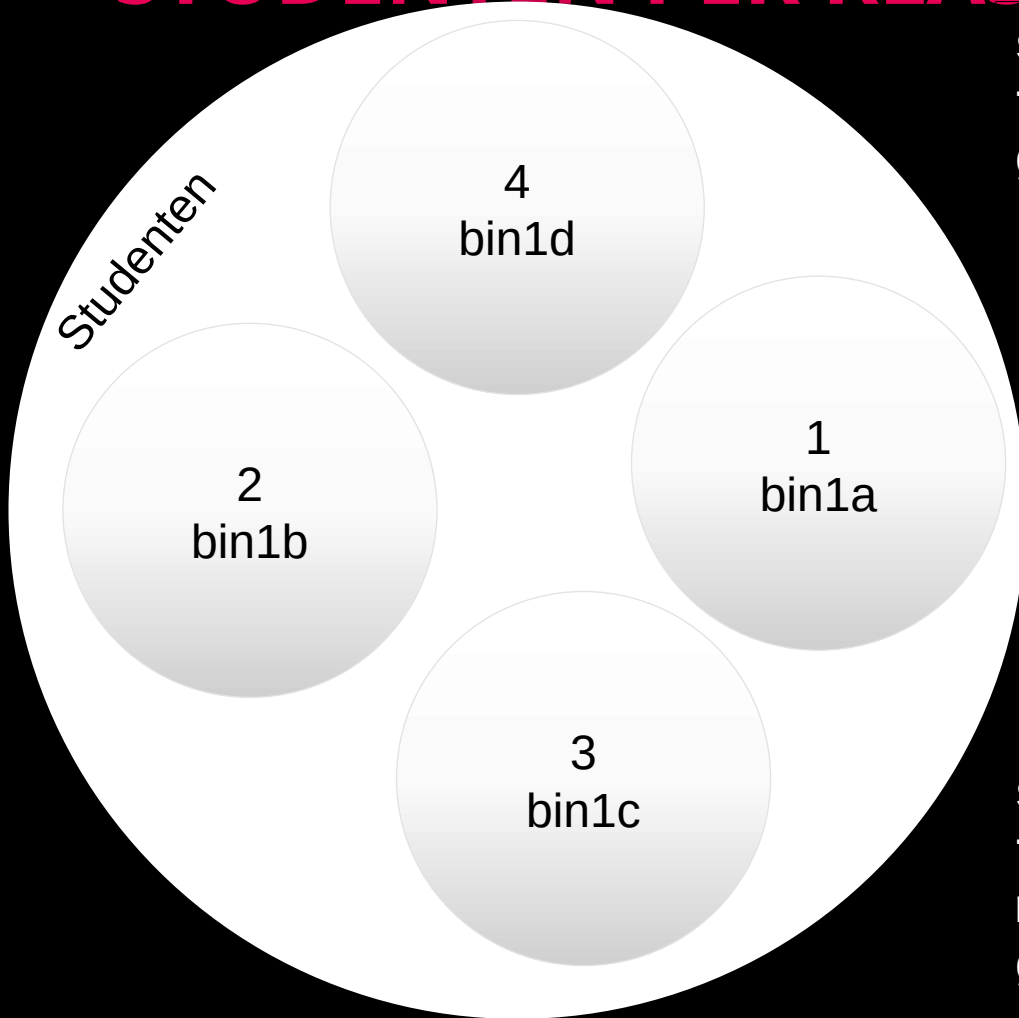
Wat gaat hier fout?

```
select voornaam, klas_id, count(*)  
from student  
group by klas_id;
```



Voornaam is geen eigenschap
van de groep

STUDENTEN PER KLAS_ID



```
select klas_id, count(*)  
from student  
group by klas_id
```

```
select klas_id, max(geb_datum)  
from student  
group by klas_id
```

```
select avg(studnr), count(*)  
from student  
group by klas_id
```

```
select klas_naam, count(*)  
from student  
natural join klas  
group by klas_naam
```

GROUP BY

Alles wat in de group by clause staat om te groeperen mag je opnemen in de select clause

Het is een eigenschap van de subgroep

AGENDA

Subquery's

Subquery's met groepsfuncties

Gecorreleerde subquery's

Groepen met groepsfuncties

Having by clause

HAVING CLAUSULE

Een having clause is altijd in combinatie met een group by clause

De having clause kan filteren op eigenschappen van groepen

HAVING CLAUSULE

De **having** clausule kan in tegenstelling tot de **where** clausule **wel** groepsfuncties bevatten!

VRAAG

Schrijf de query om het aantal genen per biotype te bepalen
Toon alleen de biotypes waar meer dan 200 genen voor zijn
gevonden

HAVING

```
select biotype
,      count(*)
from   gene
group by biotype
having count(*) > 200
order by 2 desc
```

VRAAG

Hoe toon je per biotype het gemiddelde GC percentage?

Toon alleen die biotypes waarvan er meer dan 1000 genen aanwezig zijn.

QUERY

```
select biotype
,      floor(avg(value))
from   gene
       natural join gene_attrib
       join attrib_type using (attrib_type_id)
where  code = 'GeneGC'
group by biotype
having count(*) > 1000
```

HET SELECT STATEMENT MET HAVING

```
select <kolomnamen | *>
```

```
from <tabel>
```

```
group by <kolomnaam>
```

```
having <conditie>
```

De having clause kan alleen in combinatie met group by worden opgenomen

SAMENVATTING

Analytische functies

avg

min

max

count

sum

Group by geeft de mogelijkheid uitspraken te doen over een groep

Having geeft de mogelijkheid te filteren op groepeigenschappen

SAMENVATTING

```
select <kolomnamen|*>  
from <tabel>  
where <conditie>  
group by <kolomnaam>  
having <conditie>  
order by <kolom>
```

OPDRACHT

- Schrijf een query om iedereen te tonen die in de klas zit bij iemand uit Apeldoorn
- In welke woonplaatsen is de jongste student geboren voor 1975?

VERANTWOORDING

In deze uitgave is géén auteursrechtelijk beschermd werk opgenomen

Alle teksten © Martijn van der Bruggen/HAN tenzij expliciet externe bronnen zijn aangegeven

Screenshots op basis van eigen werk auteur en/of vernoemde sites

Eventuele images zijn opgenomen met vermelding van bron