SQL—Structured Query Language

- Repetisjon av select spørringer
- Nestede select spørringer
- Mengdeoperasjoner
- Views
- Flere operatorer

Generelt utseende av SQL-spørsmål

```
  select    [distinct] <resultatattributter>
  from      <tabeller/join uttrykk>
[ where     <utvalgbetingelse> ]
[ group by  <grupperingsattributter>
[ having    <resultatbetingelse> ] ]
[ order by  <ordningsattributter> ]
```

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
   - Eventuelt utfør eksplisite (f.eks. outer) joins

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
   - Eventuelt utfør eksplisite (f.eks. outer) joins
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
   - Eventuelt utfør eksplisite (f.eks. outer) joins
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
   - Eventuelt utfør eksplisite (f.eks. outer) joins
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe
5. Fjern de gruppene som ikke oppfyller resultatbetingelsen i **having**-klausulen

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
   - Eventuelt utfør eksplisite (f.eks. outer) joins
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe
5. Fjern de gruppene som ikke oppfyller resultatbetingelsen i **having**-klausulen
6. Behold bare attributtene i **select** (projeksjon)

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
   - Eventuelt utfør eksplisite (f.eks. outer) joins
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe
5. Fjern de gruppene som ikke oppfyller resultatbetingelsen i **having**-klausulen
6. Behold bare attributtene i **select** (projeksjon)
7. Fjern flerforekomster hvis **select distinct**

Hvordan SQL-spørsmål med GROUP BY evalueres

1. Selekter ifølge seleksjonsbetingelsene i **where**
2. Join relasjonene i **from** i henhold til joinbetingelsene i **where**
   - Eventuelt utfør eksplisite (f.eks. outer) joins
3. Grupper resultattuplene i henhold til like verdier i grupperingsattributtene angitt i **group by**-klausulen
4. Beregn aggregeringer (count, sum,. . .) per gruppe
5. Fjern de gruppene som ikke oppfyller resultatbetingelsen i **having**-klausulen
6. Behold bare attributtene i **select** (projeksjon)
7. Fjern flerforekomster hvis **select distinct**
8. Sorter i henhold til **order by**

Nested select

Nestede spørsmål

- Gitt tabellen Ansatt(anr, navn, lonn, avd)
  Finn antall selgere som tjener *mer enn det dobbelte* av markedsførernes gjennomsnittslønn

```
select count *
from   Ansatt
where avd = 'salg' and
      lonn > ( select 2 * avg(lonn)
               from Ansatt
               where avd = 'marketing');
```

- Merk: En **select** inne i **where**-betingelsen må være omsluttet av parenteser

For hvert prosjekt, list opp medvirkende avdelinger som har minst 10 ansatte og sorter dem etter innsats (Altså: ta bare med avdelinger som har minst 10 ansatte):

Ansatt(anr, navn, lonn, avd)
Avdeling(avdnr, a-navn, leder)
Prosjektplan(pnr, anr, timer)

```
select    pnr as prosjekt, avdnr as avdnummer,
          a-navn as avdeling, sum(timer) as innsats
from      Ansatt A, Avdeling, Prosjektplan P
where     avd = avdnr and A.anr = P.anr
group by  pnr, avdnr, a-navn
having    sum(timer) > 99 and
          9 < (select count(*)
               from  Ansatt A1
               where A1.avd = avdnr)
order by  prosjekt, innsats desc;
```

Merk bruken av avdnr i den indre select-setningen! Den gjør at den indre select-setningen må beregnes én gang for hver verdi av avdnr beregnet i den ytre select-setningen

Online Tutorial

Flott interaktiv tutorial:

`http://sqlzoo.net/wiki/SELECT_within_SELECT_Tutorial`

Mengdeoperatorer

SQLs operatorer for union, snitt og differanse av tabeller

- I SQL utgjør ikke tuplene i svaret fra en select-setning en mengde

- En multimengde (multiset) eller *bag*

```
fdb=> select C.country
from Filmcountry C
where C.filmid < 50
order by C.country;
   country
-------------
 Belgium
 Colombia
 Denmark
 Denmark
 Netherlands
 Spain
 Sweden
 Sweden
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 USA
(16 rows)
```

```
fdb=> select C.country
from Filmcountry C
where C.filmid < 30
order by C.country;
 country
---------
 Belgium
 Denmark
 Spain
 Sweden
 Sweden
 USA
 USA
(7 rows)
```

```
( )
union all
( )
```

```
( )
union all
( )


  ( select C.country
    from Filmcountry C
    where C.filmid < 50
    order by C.country )

 union all

  ( select C.country
    from Filmcountry C
    where C.filmid < 30
    order by C.country );
```

```
 ( select C.country
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

union all

 ( select C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
```

```
 country
-------------
 Belgium
 Colombia
 Denmark
 Denmark
 Netherlands
 Spain
 Sweden
 Sweden
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 Belgium
 Denmark
 Spain
 Sweden
 Sweden
 USA
 USA
(23 rows)
```

```
 ( select C.country
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

union all

 ( select C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
```

unionen

```
 country
-------------
 Belgium
 Colombia
 Denmark
 Denmark
 Netherlands
 Spain
 Sweden
 Sweden
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 Belgium
 Denmark
 Spain
 Sweden
 Sweden
 USA
 USA
(23 rows)
```

```
 country
-------------
 Belgium
 Colombia
 Denmark
 Denmark
 Netherlands
 Spain
 Sweden
 Sweden
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 USA
(16 rows)
```

```
 country
---------
 Belgium
 Denmark
 Spain
 Sweden
 Sweden
 USA
 USA
(7 rows)
```

```
 ( select C.country
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

intersect all

 ( select C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
```

```
 country
-------------
 Belgium
 Colombia
 Denmark
 Denmark
 Netherlands
 Spain
 Sweden
 Sweden
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 USA
(16 rows)
```

```
 country
---------
 Belgium
 Denmark
 Spain
 Sweden
 Sweden
 USA
 USA
(7 rows)
```

```
 ( select C.country
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

intersect all

 ( select C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
```

```
 country
---------
 Denmark
 Sweden
 Sweden
 Spain
 Belgium
 USA
 USA
(7 rows)
```

```
 country
-------------
 Belgium
 Colombia
 Denmark
 Denmark
 Netherlands
 Spain
 Sweden
 Sweden
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 USA
(16 rows)
```

snittet

```
 country
---------
 Belgium
 Denmark
 Spain
 Sweden
 Sweden
 USA
 USA
(7 rows)
```

```
 ( select C.country
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

except all

 ( select C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
```

```
 country
-------------
 Belgium
 Colombia
 Denmark
 Denmark
 Netherlands
 Spain
 Sweden
 Sweden
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 USA
(16 rows)
```

```
 country
---------
 Belgium
 Denmark
 Spain
 Sweden
 Sweden
 USA
 USA
(7 rows)
```

```
 ( select C.country
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

except all

 ( select C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
```

```
 country
-------------
 Colombia
 Netherlands
 Denmark
 USA
 USA
 USA
 USA
 USA
 USA
(9 rows)
```

```
 country
-------------
 Belgium
 Colombia
 Denmark
 Denmark
 Netherlands
 Spain
 Sweden
 Sweden
 USA
 USA
 USA
 USA
 USA
 USA
 USA
 USA
(16 rows)
```

differansen

```
 country
---------
 Belgium
 Denmark
 Spain
 Sweden
 Sweden
 USA
 USA
(7 rows)
```

```
 ( select C.country, C.filmid
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

union all

 ( select  C.filmid, C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
```

```
   country   | filmid
-------------+--------
 Belgium     |     21
 Colombia    |     34
 Denmark     |     20
 Denmark     |     36
 Netherlands |     40
 Spain       |     18
 Sweden      |     20
 Sweden      |     24
 USA         |     35
 USA         |     23
 USA         |     37
 USA         |     42
 USA         |     43
 USA         |     26
 USA         |     49
 USA         |     33
(16 rows)
```

```
 filmid | country
--------+---------
     21 | Belgium
     20 | Denmark
     18 | Spain
     20 | Sweden
     24 | Sweden
     23 | USA
     26 | USA
(7 rows)
```

```
 ( select C.country, C.filmid
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

union all

 ( select  C.filmid, C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
```

```
   country   | filmid
-------------+--------
 Belgium     |     21
 Colombia    |     34
 Denmark     |     20
 Denmark     |     36
 Netherlands |     40
 Spain       |     18
 Sweden      |     20
 Sweden      |     24
 USA         |     35
 USA         |     23
 USA         |     37
 USA         |     42
 USA         |     43
 USA         |     26
 USA         |     49
 USA         |     33
(16 rows)
```

```
 ( select C.country, C.filmid
   from Filmcountry C
   where C.filmid < 50
   order by C.country )
 union all
 ( select C.filmid, C.country
   from Filmcountry C
   where C.filmid < 30
   order by C.country );
ERROR:  UNION types text and integer cannot be matched
LINE 6:  ( select C.filmid, C.country
```

```
 filmid | country
--------+---------
     21 | Belgium
     20 | Denmark
     18 | Spain
     20 | Sweden
     24 | Sweden
     23 | USA
     26 | USA
(7 rows)
```

```
  ( select C.country, C.filmid
    from Filmcountry C
    where C.filmid < 50
    order by C.country )

union all

  ( select  C.filmid, C.country
    from Filmcountry C
    where C.filmid < 30
    order by C.country );
```

```
   country   | filmid
-------------+--------
 Belgium     |     21
 Colombia    |     34
 Denmark     |     20
 Denmark     |     36
 Netherlands |     40
 Spain       |     18
 Sweden      |     20
 Sweden      |     24
 USA         |     35
 USA         |     23
 USA         |     37
 USA         |     42
 USA         |     43
 USA         |     26
 USA         |     49
 USA         |     33
(16 rows)
```

De to multimengdene er ikke *unionkompatible* !

```
 filmid | country
--------+---------
     21 | Belgium
     20 | Denmark
     18 | Spain
     20 | Sweden
     24 | Sweden
     23 | USA
     26 | USA
(7 rows)
```

```
  ( select C.country, C.filmid
    from Filmcountry C
    where C.filmid < 50
    order by C.country )

union all

  ( select  C.country, C.filmid
    from Filmcountry C
    where C.filmid < 30
    order by C.country );
```

```
   country   | filmid
-------------+--------
 Belgium     |     21
 Colombia    |     34
 Denmark     |     20
 Denmark     |     36
 Netherlands |     40
 Spain       |     18
 Sweden      |     20
 Sweden      |     24
 USA         |     35
 USA         |     23
 USA         |     37
 USA         |     42
 USA         |     43
 USA         |     26
 USA         |     49
 USA         |     33
(16 rows)
```

```
 country | filmid
---------+--------
 Belgium |     21
 Denmark |     20
 Spain   |     18
 Sweden  |     20
 Sweden  |     24
 USA     |     23
 USA     |     26
(7 rows)
```

De to multimengdene er *unionkompatible* !

```
  ( select C.country, C.filmid
    from Filmcountry C
    where C.filmid < 50
    order by C.country )

union all

  ( select  C.country, C.filmid
    from Filmcountry C
    where C.filmid < 30
    order by C.country );
```

```
 country    | filmid
------------+--------
 Belgium    |     21
 Colombia   |     34
 Denmark    |     20
 Denmark    |     36
 Netherlands|     40
 Spain      |     18
 Sweden     |     20
 Sweden     |     24
 USA        |     35
 USA        |     23
 USA        |     37
 USA        |     42
 USA        |     43
 USA        |     26
 USA        |     49
 USA        |     33
 Belgium    |     21
 Denmark    |     20
 Spain      |     18
 Sweden     |     20
 Sweden     |     24
 USA        |     23
 USA        |     26
(23 rows)
```

De to multimengdene er *unionkompatible* !

```
 ( select C.country, C.filmid
   from Filmcountry C
   where C.filmid < 50
   order by C.country )

union all

 ( select  F.title, F.filmid
   from Film F
   where F.filmid < 30
   order by F.title );
```

```
  ( select C.country, C.filmid
    from Filmcountry C
    where C.filmid < 50
    order by C.country )

union all

  ( select  F.title, F.filmid
    from Film F
    where F.filmid < 30
    order by F.title );
```

De to multimengdene *er* unionkompatible !

```
                country                | filmid
---------------------------------------+--------
 Belgium                               |     21
 Colombia                              |     34
 Denmark                               |     20
 Denmark                               |     36
 Netherlands                           |     40
 Spain                                 |     18
 Sweden                                |     20
 Sweden                                |     24
 USA                                   |     35
 USA                                   |     23
 USA                                   |     37
 USA                                   |     42
 USA                                   |     43
 USA                                   |     26
 USA                                   |     49
 USA                                   |     33
 !Huff                                 |     26
 Brooklyn Connection, The              |     23
 E.R. Sluts                            |     17
 Huff                                  |     28
 Jack l'éventreur                      |     21
 Jag en kvinna, II - äktenskapet       |     20
 Kronvittnet                           |     24
 Privilege                             |     19
 Tarantos, Los                         |     18
(25 rows)
```

```
 ( select C.country
   from Filmcountry C
 )

intersect all

 ( select  F.title
   from Film F
 );
```

```
 ( select C.country
   from Filmcountry C
 )

intersect all

 ( select  F.title
   from Film F
 );
```

```
....

 China
 China
 Kenya
 Mauritius
 Mauritius
 Armenia
 Armenia
 Greece
 Yemen
 Andorra
 Andorra
 Andorra
 Andorra
 Switzerland
 Switzerland
 Switzerland
 Indonesia
 Belgium
 Belgium
 Bolivia
 Mali
(152 rows)
```

```
 ( select C.country
   from Filmcountry C
 )

intersect

 ( select  F.title
   from Film F
 );
```

```
 ( select C.country
   from Filmcountry C
 )

intersect

 ( select  F.title
   from Film F
 );
```

```
....

 Thailand
 Venezuela
 Aruba
 Hong Kong
 Portugal
 Romania
 Canada
 China
 Kenya
 Mauritius
 Armenia
 Greece
 Yemen
 Andorra
 Switzerland
 Indonesia
 Belgium
 Bolivia
 Mali
(71 rows)
```

```
 ( select C.country, C.filmid
   from Filmcountry C
 )

intersect all

 ( select  F.title, F.filmid
   from Film F
 );
```

```
 ( select C.country, C.filmid
   from Filmcountry C
 )

intersect all

 ( select  F.title, F.filmid
   from Film F
 );
```

```
 country   | filmid
-------------+---------
 Australia   |  113619
 France      |  503461
 Hong Kong   |  851611
 Mexico      | 1116904
 Portugal    | 2207717
 Puerto Rico | 5563914
(6 rows)
```

Views

Views

- Et view er en tenkt relasjon som vi bruker som mellomresultat i kompliserte SQL-beregninger
- Det er to måter å lage view på:
  - **create view** navn(attributtliste) **as select** ...
  - **create view** navn **as select** ...
- I det første tilfellet må det være like mange navn i attributtlisten som det er attributter i **select**-setningen
- I det andre tilfellet arver viewet attributtnavnene fra **select**-setningen
- Når man har laget et view, kan det brukes som en vanlig tabell i senere **select**-setninger

Eksempel på view og union

Prosjektplan(pnr, anr, timer)

```
create view Innsats as
            select   anr, sum(timer) as timer
            from     Prosjektplan
            group by anr;


create view Bonus(anr, bonusiNOK) as
            (select  anr, 3000
            from     Innsats
            where    timer >= 500 )
       union                              -- en ny relasjonsoperator!
            (select  anr, 1500
             from    Innsats
             where   timer >= 300 and timer < 500 );
```

```
/* Lage et view med oversikt over hvor
 * mange funksjoner filmarbeiderne
 * med flere enn 1 funksjon har hatt */

-- først lager vi selectsetningen:

select p.personid, count(distinct parttype)
from  person p, filmparticipation x
where x.personid = p.personid
group by p.personid
having count(distinct parttype) > 1;
```

```
-- så viewet

create view funksjoner (personid, antroller) as
select p.personid, count(distinct parttype)
from  person p, filmparticipation x
where x.personid = p.personid
group by p.personid
having count(distinct parttype) > 1;
```

```
select max(firstname) || ' ' || max(lastname) as navn,
       parttype as deltakerfunksjon,
       count(*) as antall_filmer
from   person p,
       filmparticipation x,
       funksjoner f
where  x.personid = p.personid
       and f.personid = p.personid
       and f.antroller > 5
group by p.personid, parttype;
```

Flere Operasjoner

Relasjonssammenligninger

SQL har fem operatorer som sammenligner med innholdet i en hel relasjon:

i SQL-2	*betyr*
exists R	at R har minst én forekomst
not exists R	at R ikke har noen forekomster
in R	$\in$ R
not in R	$\notin$ R
any R	en vilkårlig verdi i R
all R	alle verdier i R

ANY og ALL

- any og all brukes i praksis bare på relasjoner med ett attributt

- Eksempel:
Finn antall selgere som tjener mer enn samtlige markedsførere

Ansatt(anr, navn, lonn, avd)
Avdeling(avdnr, avdelingsnavn, leder)
Prosjektplan(pnr, anr, timer)

```
select count(*)
from   Ansatt
where  avd = 'salg'  and
       lonn > all (select lonn
                   from   Ansatt
                   where  avd = 'marketing');
```

IN og NOT IN

- [not] in kan brukes på ett attributt eller på en liste av attributter

- Eksempel:

Finn navn på ansatte som ikke har ført noen prosjekttimer

Ansatt(anr, navn, lonn, avd)
Avdeling(avdnr, avdelingsnavn, leder)
Prosjektplan(pnr, anr, timer)

```
select navn
from   Ansatt
where  anr not in (select anr
                   from   Prosjektplan);
```

IN og NOT IN

- [not] in kan brukes på ett attributt eller på en liste av attributter

- Eksempel fra filmdatabasen:

Finn navn på personer som (har en personid som) ikke finnes i filmdeltakelsetabellen:

Person (personid, lastname, firstname, gender)
Filmparticipation (partid, personid, filmid, parttype)

```
select lastname || ', ' || firstname as navn
from   Person
where  personid not in (select distinct personid
                        from   Filmparticipation);
```

IN og NOT IN

- [not] in kan brukes på ett attributt eller på en liste av attributter

- Eksempel fra filmdatabasen:

Finn navn på personer som har deltatt i en film med en tittel som matcher en liste av titler:

Person (personid, lastname, firstname, gender))
Filmparticipation (partid, personid, filmid, parttype)
Film (filmid, title, prodyear)

```
select lastname || ', ' || firstname as navn,
title deltok_i
from   Person
       natural join Filmparticipation
       natural join Film
where  title in ('A', 'France', 'Tintin', 'B');
```

EXISTS og NOT EXISTS

- **exists** R
  er sann hvis tabellen inneholder tupler (ett eller flere)
- **not exists** R
  er sann hvis tabellen ikke inneholder noen tupler
- Merk at SQL ikke har noen egen all-kvantor ($\forall$)
- Skulle vi trenge en all-kvantor, må vi uttrykke den ved hjelp av andre SQL-konstruksjoner

Noen nyttige formler fra logikken

- $F \Rightarrow G \equiv \text{not } F \text{ or } G$
- $\text{not } (F \text{ and } G) \equiv \text{not } F \text{ or not } G$
- $\text{not } (F \text{ or } G) \equiv \text{not } F \text{ and not } G$
- $\forall u.F \equiv \text{not } (\exists u.\text{not } F)$
- $\exists u.F \equiv \text{not } (\forall u.\text{not } F)$

Eksempel

Finn navn på ansatte som skal arbeide mer enn 10 timer på samtlige av sine prosjekter

Ansatt(anr, navn, lonn, avd)
Avdeling(avdnr, avdelingsnavn, leder)
Prosjektplan(pnr, anr, timer)

```
select A.navn
from   Ansatt A
where  not exists ( select *
                    from   Prosjektplan P
                    where  P.anr = A.anr
                    and P.timer <= 10 );
```

Finn titler på filmer som bare har hatt kvinner som deltakere.

```
Person (personid, lastname, firstname, gender)
Filmparticipation (partid, personid, filmid, parttype)
Film (filmid, title, prodyear)
```

```
select gender, count(*) from Person
group by gender;
```

```
 gender | count
--------+--------
        | 415520
 M      | 810006
 F      | 483858
```

```
-- Filmer med minst en ikke-kvinne:

                    select distinct filmid
                    from Filmparticipation fp
                    natural join Person p
                    where (p.gender in ('␣', 'M')
                    or p.gender is null)
```

Filmer med fler enn 25 deltakere hvor samtlige av deltakerne er kvinner:

```
select f.title, count(distinct x.personid)
from Filmparticipation x
     natural join Film f
where not exists (
                  select filmid
                  from Filmparticipation fp
                       natural join Person p
                  where ( p.gender in ('␣', 'M')
                          or p.gender is null )
                        and f.filmid = fp.filmid
                 )
      and exists (
                  select filmid
                  from Filmparticipation fp
                       natural join Person p
                  where p.gender = 'F'
                        and f.filmid = fp.filmid
                 )
group by f.title
having count(distinct x.personid) > 25;
```