Grunderna i SQL – del 1

1. SELECT-frågor
2. SELECT
3. WHERE
4. ORDER BY

Kap. 3

5. Inre join
6. Yttre join
7. Andra typer av join
8. Union

Kap. 4

9. Aggregatfunktioner
10. Gruppera och summera

Kap. 5
utom ROLLUP, CUBE, GROUPING SETS

1. SELECT-frågor

- **Med SELECT-*frågor* så *hämtas* information från en databas**
- **SELECT-*delen* anger attributen (kolumnera)**
- **FROM-*delen* anger tabellen (eller tabellerna) som attributen tillhör**
- **WHERE-*delen* anger ett valfritt sökvillkor för att begränsa posterna (= raderna) i resultat**
- **ORDER BY-*delen* sorterar resultatet**

Grundläggande syntax

- ```
  SELECT Attribut            -- Anger attributen som ska hämtas
  FROM Tabell                -- Anger tabellen som attributen tillhör
  [WHERE Sökvillkor]         -- Anger ett sökvillkor för att begränsa resultatet
  [ORDER BY Attribut]        -- Sorterar resultatet
  ```

  (med mera)

Notera

- **WHERE och ORDER BY är valfritt i en SELECT-*fråga!***
  ```

2. SELECT

- **I samband med `SELECT`-*delen* så:**
  - **Måste attribut av intresse anges**
  - **Kan alla attribut av en tabell anges med en *asterisk* (*)**
  - **Kan resultatet begränsas till *alla* (`ALL`) poster eller endast *unika* (`DISTINCT`) poster**
    - **`ALL` är default**
  - **Kan *antalet* poster i resultatet begränsas genom `TOP` (i SQL Server)**
  - **Kan *attributalias* (*kolumnalias*) skapas genom nyckelordet `AS` (även tabellalias)**

Syntax

```
SELECT [ALL|DISTINCT] [TOP n [PERCENT] [WITH TIES]]
    Attribut1 [[AS] Alias1]
    [, Attribut2 [[AS] Alias2]]...
FROM Tabell
```

2.1. Kolumnnamn och stränguttryck

- **Nyckelordet AS används för att ange ett *attributalias* för resulterande attribut**
- **Resulterande attribut kan behandlas som *stränguttryck* (kräver attribut av strängtyp!)**
- ***Stränguttryck* skrivs mellan *enkla apostroftecken* (')**
- ***Stränguttryck* sammanfogas med + -tecken (precis som för `string` -objekt i C#)**
- ***Specialtecken* används genom att inkludera en extra apostrof (') *framför specialtecknet***

Syntax

- **SELECT** 'Stränguttryck' **+** Attribut **AS** Alias...

Exempel

- ***Hämtar fullständigt namn:***

```
SELECT 'Fullnamn: ' + Förnamn + ' ' + Efternamn AS Namntexter
FROM Lärare
```

Namntexter
Fullnamn: Andreas Persson
Fullnamn: Johan Petersson

Lärare

Förnamn	Efternamn	E-post
Andreas	Persson	andreas.persson@oru.se
Johan	Petersson	johan.petersson@oru.se

2.2. Aritmetiska uttryck

- **Attribut bestående av *tal* (heltal eller flyttal) fungerar tillsammans med *aritmetiska operatorer***
- ***Multiplikation*, *division* och *mobulus* utförs före *subtraktion* och *addition***
  - **Ordningen på operationerna kan ändras genom parenteser!**
- **Alias som innehållande *mellanrum* skrivs inom *hakparenteser* ([])!**

Aritmetiska operatorer

- `+` `-- Adderar attribut`
- `-` `-- Subtraktion mellan attribut`
- `*` `-- Multiplicerar attribut`
- `/` `-- Dividerar attribut`
- `%` `-- Modulus mellan attribut`

Lönespecifikation

AnställdID	Lön
5	18700
2	22500
10	21200
7	NULL

Exempel

- ***Lön efter skatt på 30%:***

```
SELECT AnställdID, (Lön * 0.7) AS [Lön efter skatt]
FROM Lönespecifikation
```

Resultat

AnställdID	Lön efter skatt
5	13090
2	15750
10	14840

2.3. ALL eller DISTINCT resultat

- **Default hämtas *alla* (ALL) poster vid en SELECT -frågan (även dubbletter!)**
- **För endast *unika* poster i resultatet så används nyckelordet DISTINCT**

Syntax

- **SELECT** [**ALL**|**DISTINCT**] Attribut...

Exempel

- ***Anställda som har fått en utbetalning under året:***

```
SELECT DISTINCT AnställdID
FROM Utbetalningar
```

- ***Månader med utbetalningar:***

```
SELECT DISTINCT Månad
FROM Utbetalningar

-- Månad:
-- Juli
-- Augusti
```

Utbetalningar

AnställdID	Belopp	Månad
5	18700	Juli
5	1600	Juli
10	21200	Augusti
7	NULL	Juli
2	22500	Juli
2	22500	Augusti

2.4. Begränsat resultat

- **SQL Server (olika i olika system): Genom nyckelordet TOP begränsas resultatet**
- **`TOP` kan begränsas till n st poster eller n `PERCENT` av de resulterande posterna**
- **`TOP` kan kombineras med `WITH TIES` för att inkludera poster med *lika värden***
- **`TOP` bör kombineras med `ORDER BY` *-uttrycket*!**

Syntax

- `SELECT [TOP n [PERCENT] [WITH TIES]] Attribut...`

Exempel

- ***Hämta den högsta utbetalningen:***

```
SELECT TOP 1 Belopp, Månad
FROM Utbetalningar
```

- ***Hämtar högsta utbetalningar (även samma belopp för olika månader):***

```
SELECT TOP 1 WITH TIES Belopp, Månad
FROM Utbetalningar
```

Utbetalningar

AnställdID	Belopp	Månad
5	18700	Juli
5	1600	Juli
10	21200	Augusti
7	NULL	Juli
2	22500	Juli
2	22500	Augusti

3. WHERE

- **Genom `WHERE` -*uttrycket* så anges *sökvillkor* för att begränsa resultatet**
- ***Jämförelseoperatorer* används för att jämföra *attribut* mot *sökuttryck***
  - **Endast poster som uppfyller *sökvillkoret* hämtas!**

Syntax

- `WHERE Attribut operator Sökuttryck`

Jämförelseoperatorer

- `=` -- Lika med
- `>` -- Större än
- `>=` -- Större än eller lika med
- `<` -- Mindre än
- `<=` -- Mindre än eller lika med
- `<>` -- Inte lika med

3.1. Logiska operatorer

- Genom de logiska operatorerna **AND**, **OR** och **NOT** så kan flera *sökvillkor* anges
- Prioritetsordningen för de logiska operatorerna: 1.) **NOT**, 2.) **AND** och 3.) **OR**
  - Ordningen kan ändras genom parenteser!

Exempel

- ***Alla utbetalda belopp över 20000:***

```
SELECT *
FROM Utbetalningar
WHERE Belopp > 20000
```

Utbetalningar

AnställdID	Belopp	Månad
5	18700	Juli
5	1600	Juli
10	21200	Augusti
7	NULL	Juli
2	22500	Juli
2	22500	Augusti

- ***Alt.1.) Alla utbetalda belopp i intervallet 15000-20000:***

```
SELECT *
FROM Utbetalningar
WHERE Belopp >= 15000 AND Belopp <= 20000
```

- ***Alt.2.) Alla utbetalda belopp i intervallet 15000-20000:***

```
SELECT *
FROM Utbetalningar
WHERE NOT( Belopp < 15000 OR Belopp > 20000)
```

3.2. IN- och BETWEEN-operatorerna

- **`IN`-operatorn jämför ett attribut mot *en lista av uttryck***
- **`BETWEEN`-operatorn jämför ett attribut mot *ett intervall***

Syntax

- `WHERE Attribut [NOT] IN ( Uttryck1, Uttryck2, ...)`
- `WHERE Attribut [NOT] BETWEEN Startuttryck AND Stopputtryck`

Exempel

- ***Alt.3.) Alla utbetalda belopp i intervallet 15000-20000:***

```
SELECT *
FROM Utbetalningar
WHERE Belopp BETWEEN 15000 AND 20000
```

Utbetalningar

AnställdID	Belopp	Månad
5	18700	Juli
5	1600	Juli
10	21200	Augusti
7	NULL	Juli
2	22500	Juli
2	22500	Augusti

Notera

- **`NOT` kan användas framför både `IN` och `BETWEEN` för att få det *motsatta resultatet***

3.3. LIKE-operatorn

- **`LIKE`-operatorn jämför *delvis* ett attribut mot ett uttryck**
- **Genom att jämföra ett *strängmönster* mot ett attribut så *maskas poster ut***

Syntax

- **`WHERE`** Attribut [**`NOT`**] **`LIKE`** Strängmönster

Strängmönster

- `%` -- Alla strängar med noll eller flera tecken
- `_` -- Ett tecken
- **`[abc]`** -- Ett av tecknen inom hakparenteserna
- **`[a-z]`** -- Alla tecken i det angivna intervallet
- **`[^a-z]`** -- Utesluter alla tecken i det angivna intervall

Notera

- ***Strängmönster* som anges i samband med `LIKE`-operatorna är inte *case sensitive!!***
- **`NOT` kan användas framför `LIKE`-operatorn för att i stället utesluta poster.**

3.4. Exempel

- ***Alla telefonnummer till Örebro universitet:***

```
SELECT Telefon
FROM Lärare
WHERE Telefon LIKE '019-30%'
```

- ***Alla efternamn som börjar på ”Per” eller ”Pet”:***

```
SELECT Efternamn
FROM Lärare
WHERE Efternamn LIKE 'Pe[rt]%'
```

Lärare

Förnamn	Efternamn	E-post	Telefon	Rumsnummer
Andreas	Persson	andreas.persson@oru.se	019-303191	L2123
Johan	Petersson	johan.petersson@oru.se	019-303811	L2138

3.5. IS NULL-operatorn

- **Då vissa attribut tillåter `NULL`-värden finns `IS NULL`**
  - **Hämta poster som har just ett `NULL`-värde!**
  - **Behövs eftersom värdet NULL beter sig så konstigt: inget, inte ens NULL, är lika med NULL!**

Syntax

- ***Alla som inte har ett telefonnummer:***

```
SELECT Förnamn, Efternamn
FROM Lärare
WHERE Telefon IS NULL
```

Lärare

Förnamn	Efternamn	E-post	Telefon
Andreas	Persson	andreas.persson@oru.se	019-303191
Johan	Petersson	johan.petersson@oru.se	019-303811
Kalle	Räisänen	kalle.raisanen@oru.se	*NULL*

- ***Alla som har ett telefonnummer:***

```
SELECT Förnamn, Efternamn
FROM Lärare
WHERE Telefon IS NOT NULL
```

Notera

- **`NOT` används i samband med `IS NULL`-operatorn för att *utesluta* poster med `NULL`-värden.**

4. ORDER BY

- Genom **ORDER BY**-*uttrycket* så sorteras resultatet
- Sorteringen kan ske i *stigande ordning* (**ASC**) eller *sjunkande ordning* (**DESC**)
  - Default används *stigande ordning* (**ASC**)

Syntax

- **ORDER BY** Attribut [**ASC**|**DESC**]...

Exempel

- ***Alla utbetalda belopp över 15000 i stigande ordning:***

```
SELECT Belopp, Månad
FROM Utbetalningar
WHERE Belopp > 15000
ORDER BY Belopp
```

Utbetalningar

AnställdID	Belopp	Månad
5	18700	Juli
5	1600	Juli
10	21200	Augusti
7	NULL	Juli
2	22500	Juli
2	22500	Augusti

Notera

- Sorteringsordningen för **ORDER BY**-*uttrycket* är: 1.) **NULL** -*värden*, 2.) *specialtecken*, 3.) *siffror* och 4.) *bokstäver*.

4.1. Sortering på alias och attributnummer

- **Sortering med `ORDER BY` -uttrycket kan även ske på *attributalias* eller direkt på *attributnumret***

Exempel

- ***Sorterat i sjunkande ordning på ett alias för fullständigt namn:***

```
SELECT Förnamn + ' ' + Efternamn AS Fullnamn
FROM Lärare
ORDER BY Fullnamn DESC
```

- ***Sorterat i sjunkande ordning på andra attributet (Efternamn):***

```
SELECT Förnamn, Efternamn
FROM Lärare
ORDER BY 2 DESC

-- Efternamn:
-- Räisänen
-- Petersson
-- Persson
```

Lärare

Förnamn	Efternamn	E-post	Telefon
Andreas	Persson	andreas.persson@oru.se	019-303191
Johan	Petersson	johan.petersson@oru.se	019-303811
Kalle	Räisänen	kalle.raisanen@oru.se	*NULL*

5. Inre join (”vanlig join”)

- **Med *inre join* (inner join) kombineras attribut från två eller flera tabeller**
- **Med *join* jämförs attribut från flera tabeller mot varandra**
  - **Resultatet är de poster som uppfyller ett *joinvillkor***
- **En *join* skapas ofta på förhållandet mellan en *primärnyckel och en främmande nyckel***

Syntax

- ```
  SELECT Attribut
  FROM Tabell1
      [INNER] JOIN Tabell2 ON Joinvillkor
      [[INNER] JOIN Tabell3 ON...
  ```

Notera

- **Ifall attribut har samma namn i två tabeller så måste attributen skiljas genom att använda ”hela” namnet, ex:**

  ```
  FROM Tabell1 JOIN Tabell2 ON Tabell1.Attribut = Tabell2.Attribut
  ```

- **Alla *jämförelseoperatorn* fungerar för *joinvillkor* (dock är =-*operatorn* vanligast).**
- **Nyckelorder `INNER` är valfritt och används sällan då en `JOIN` är underförstått en `INNER JOIN`.**
  ```

5.1. Exempel

- ***En inre join som kombinerar fullständigt namn och tillhörande akademi:***

```
SELECT Förnamn + ' ' + Efternamn AS Fullnamn, Akademier.Namn AS Akademi
FROM Lärare JOIN Akademier
    ON Lärare.Akademi = Akademier.AkademiID
ORDER BY Fullnamn
```

Lärare

Förnamn	Efternamn	E-post	Telefon	Akademi
Andreas	Persson	andreas.persson@oru.se	019-303191	3
Johan	Petersson	johan.petersson@oru.se	019-303811	3
Kalle	Räisänen	kalle.raisanen@oru.se	*NULL*	3

Akademier

AkademiID	Namn
3	Handelshögskolan
4	Institutionen för naturvetenskap och teknik

Resultat

Fullnamn	Akademi
Andreas Persson	Handelshögskolan
Johan Petersson	Handelshögskolan
Kalle Räisänen	Handelshögskolan

5.2. Implicit inre join

- **Den syntax som har använts hittills har varit *explicit join-syntax***
- **Innan *SQL-92-standarden* fanns bara *implicit join-syntaxen*:**
  - **Alla tabeller skrivs i `FROM`-*uttrycket* separerade med komman**
  - ***Joinvillkoret* skrivs i `WHERE`-*uttrycket***

Syntax

- ```
  SELECT Attribut
  FROM Tabell1, Tabell2, ...
  WHERE Tabell1.Attribut operator Tabell2.Attribut [AND|OR] ...
  ```

Exempel

- ***Implicit join som listar fullständigt namn och tillhörande akademi:***

```
SELECT Förnamn + ' ' + Efternamn AS Fullnamn, Akademier.Namn AS Akademi
FROM Lärare, Akademier
WHERE Lärare.Akademi = Akademier.AkademiID
```

Notera

- **Vid join mellan fler än två tabeller så sker join från *vänster till höger* och mellan varje join så sparas resultatet i en *provisorisk tabell.* (Nej, det gör det inte, men man kan tänka sig det.)**
```

5.3. Tabellalias

- ***Tabellalias* kan användas för ett *temporärt tabellnamn* inom `FROM` *-uttrycket***
- ***Tabellalias* skapas (liksom för attribut) genom nyckelordet `AS` framför ett alias**

Syntax

- ```
  SELECT Attribut
  FROM Tabell1 AS Alias1
      [INNER] JOIN Tabell2 AS Alias2 ON...
  ```

Exempel

- ***Föregående exempel fast med tabellalias:***

```
SELECT Förnamn + ' ' + Efternamn AS Fullnamn, Akademier.Namn AS Akademi
FROM Lärare AS L JOIN Akademier AS A ON L.Akademi = A.AkademiID
ORDER BY Fullnamn
```

Notera

- ***Tabellalias* kan ge en tydligare syntax ifall långa tabell- och attributnamn används!**
```

5.4. Självjoin

- **Med *självjoin* så skapas en join mellan attribut i en och samma tabell**
- **Vid *självjoin* så måste *tabellalias* användas**

Exempel

- ***Skapar en tabell över ”samarbeten” mellan akademier:***

```
SELECT DISTINCT A1.Namn AS [Första akademin], A2.Namn AS [Andra akademin]
FROM Akademier AS A1 JOIN Akademier AS A2 ON A1.Namn <> A2.Namn
ORDER BY 1 DESC
```

Resultat

Första akademin	Andra akademin
Handelshögskolan	Akademin för Naturvetenskap och teknik
Akademin för Naturvetenskap och teknik	Handelshögskolan

Notera

- **`DISTINCT` används ofta tillsamman med *självjoin* för att eliminera dubbletter!**
- **Flera *joinvillkor* kan användas vid samma join genom de logiska operanderna `AND` och `OR`.**

6. Yttre join

- **En *yttre join* (outer join) inkluderar även poster som inte uppfyller *joinvillkoret***
- **En *yttre join* kan ske som: `LEFT`, `RIGHT` eller `FULL`**
- **En yttre join hämtar poster som uppfyller *joinvillkoret* + poster som inte uppfyller villkoret men som tillhör `LEFT` eller `RIGHT` tabell, eller båda tabellerna (`FULL`)**

Syntax

- ```
  SELECT Attribut
  FROM Tabell1
      {LEFT|RIGHT|FULL} [OUTER] JOIN Tabell2 ON Joinvillkor
      [{LEFT|RIGHT|FULL} [OUTER] JOIN Tabell3 ON...
  ```

Notera

- **Nyckelordet `OUTER` är valfritt och används sällan då nyckelorden `LEFT`, `RIGHT` eller `FULL` (framför `JOIN`) identifierar att det är en *yttre join*.**
- **Även vid *yttre join* så skapas en *provisorisk tabell* ifall fler än två tabeller kombineras. (Nej, det gör det inte, men man kan tänka sig det.)**
  ```

6.1. Exempel

- ***En yttre* LEFT *join som kombinerar fullständigt namn och tillhörande akademi:***

```
SELECT Förnamn + ' ' + Efternamn AS Fullnamn, Akademier.Namn AS Akademi
FROM Lärare LEFT JOIN Akademier
    ON Lärare.Akademi = Akademier.AkademiID
```

Lärare

Förnamn	Efternamn	E-post	Telefon	Akademi
Andreas	Persson	andreas.persson@oru.se	019-303191	*NULL*
Johan	Petersson	johan.petersson@oru.se	019-303811	3
Kalle	Räisänen	kalle.raisanen@oru.se	*NULL*	3

Akademier

AkademiID	Namn
3	Handelshögskolan
4	Akademin för Naturvetenskap och teknik
5	Musikhögskolan

***Resultat* (LEFT)**

Fullnamn	*Akademi*
Andreas Persson	*NULL*
Johan Petersson	Handelshögskolan
Kalle Räisänen	Handelshögskolan

6.2. Exempel2

- ***En yttre* RIGHT *join som kombinerar fullständigt namn och tillhörande akademi:***

```
SELECT Förnamn + ' ' Efternamn AS Fullnamn, Akademier.Namn AS Akademi
FROM Lärare RIGHT JOIN Akademier
    ON Lärare.Akademi = Akademier.AkademiID
```

- ***En yttre* FULL *join som kombinerar fullständigt namn och tillhörande akademi:***

```
SELECT Förnamn + ' ' Efternamn AS Fullnamn, Akademier.Namn AS Akademi
FROM Lärare FULL JOIN Akademier
    ON Lärare.Akademi = Akademier.AkademiID
```

***Resultat* (RIGHT)**

Fullnamn	Akademi
NULL	Akademin för Naturvetenskap och teknik
Johan Petersson	Handelshögskolan
Kalle Räisänen	Handelshögskolan
NULL	Musikhögskolan

***Resultat* (FULL)**

Fullnamn	Akademi
NULL	Akademin för Naturvetenskap och teknik
Andreas Persson	*NULL*
Johan Petersson	Handelshögskolan
Kalle Räisänen	Handelshögskolan
NULL	Musikhögskolan

6.3. Implicit yttre join (Gammalt och deprecated. Använd inte detta.)

- ***Joinvillkoret* i en *implicit yttre join* skrivs i `WHERE` -uttrycket, men är begränsad till två operatorer:**
  - **`*=` för `LEFT` *yttre join***
  - **`=*` för `RIGHT` *yttre join***

Syntax

- ```
  SELECT Attribut
  FROM Tabell1, Tabell2, ...
  WHERE Tabell1.Attribut {*=|=*} Tabell2.Attribut [AND|OR] ...
  ```

Exempel

- ***En implicit yttre* `LEFT` *join som kombinerar fullständigt namn och tillhörande akademi:***

```
SELECT Förnamn + ' ' + Efternamn AS Fullnamn, Akademier.Namn AS Akademi
FROM Lärare, Akademier
WHERE Lärare.Akademi *= Akademier.AkademiID
```

Notera

- **Det finns ingen motsvarande `FULL JOIN` med *implicit yttre join*!**
```

7. Andra typer av join

- **Kombinationer av *inre* och *yttre join* går bra med *explicita join* (dock inte med *implicita join*)**
- **Genom kartesisk produkt *(”korsad join”)* kombineras alla poster i två tabeller**
- **Nyckelordet `CROSS` används för att skapa en *korsad join***

Syntax

- ```
  SELECT Attribut
  FROM Tabell1 CROSS JOIN Tabell2    -- Korsad join
  ```

Exempel

- ***En korsad join som kombinerar fullständigt namn och akademier:***

```
SELECT Förnamn + ' ' + Efternamn AS Fullnamn, Akademier.Namn AS Akademi
FROM Lärare CROSS JOIN Akademier   -- Resultatet består av 9 poster
```

Notera

- **Utesluts *joinvillkoren* i `WHERE`-uttrycket så skapas en *implicit korsad join!***
- **Resultatet av en *korsad join* kallas även för den *kartesiska produkten* av två tabeller.**
```

8. Union

- **Med `UNION` så kombineras resultaten från flera `SELECT`-frågor**
- **Resultaten från respektive `SELECT`-fråga måste innehålla lika många attribut**
  - **Datatyperna för attributen måste även stämma överens!**
- **Attributen i det kombinerade resultatet får namnen från den första `SELECT`-frågan**
- **Används `ORDER BY`-uttrycket för att sortera resultatet av en UNION så måste sorteringen ske på attributen i första `SELECT`-frågan**

Syntax

```
    SELECT Attribut första frågan...
UNION [ALL]
    SELECT Attribut andra frågan...
[UNION [ALL]
    SELECT Attribut tredje frågan...]
[ORDER BY Attribut från första frågan]
```

Notera

- **En `UNION` tar automatiskt bort alla *dubblettposter*.**
  - **För att inkludera alla poster används nyckelordet `ALL`.**

8.1. Exempel

- ***En union som kombinerar resultatet från två* SELECT*-frågor:***

```
    SELECT 'Människa' AS Typ, Förnamn + ' ' + Efternamn AS Namn
    FROM Lärare
UNION
    SELECT 'Plats' AS Typ, Namn
    FROM Akademier
ORDER BY Namn
```

Lärare

Förnamn	Efternamn	E-post	Telefon	Akademi
Andreas	Persson	andreas.persson@oru.se	019-303191	*NULL*
Johan	Petersson	johan.petersson@oru.se	019-303811	3
Kalle	Räisänen	kalle.raisanen@oru.se	*NULL*	3

Akademier

AkademiID	Namn
3	Handelshögskolan
4	Akademin för Naturvetenskap och teknik

Resultat

Typ	Namn
Plats	Akademin för Naturvetenskap och teknik
Människa	Andreas Persson
Plats	Handelshögskolan
Människa	Johan Petersson
Människa	Kalle Räisänen

8.2. EXCEPT (differens) och INTERSECT (snitt)

- **Med EXCEPT så *utesluts* poster som finns med som resultat för två SELECT-frågor**
- **Med INTERSECT så *inkluderas endast* poster som finns med som resultat för två SELECT-frågor**
- **Attributen i resultaten från SELECT-frågorna måste även här matcha till *antal* och *datatyper***
- **Attributen i resultatet får samma namn som attributen i den första SELECT-frågan**

Syntax

```
SELECT Attribut första frågan...
{EXCEPT|INTERSECT}
    SELECT Attribut andra frågan...
[ORDER BY Attribut från första frågan]
```

Exempel

- ***Utesluter alla lärare utan ett telefonnummer (den långa vägen):***

```
    SELECT * FROM Lärare
EXCEPT
    SELECT *
    FROM Lärare
    WHERE Telefon IS NULL
```

9. Aggregatfunktioner

- **Med *aggregatfunktioner* (eller *kolumnfunktioner*) så kan beräkningar utföras på poster för ett attribut**
- **En `SELECT`-fråga som innehåller *aggregatfunktioner* kallas även för en *summeringsfråga***
- **Genom `DISTINCT` istället för `ALL` (vilket är default) så utförs beräkningarna endast på *unika poster***

Funktioner

```
AVG([ALL|DISTINCT] Attribut)    -- Medel av posterna för attributet
SUM([ALL|DISTINCT] Attribut)    -- Summan av posterna för attributet
MIN([ALL|DISTINCT] Attribut)    -- Den minsta posten för attributet
MAX([ALL|DISTINCT] Attribut)    -- Den största posten för attributet
COUNT([ALL|DISTINCT] Attribut)  -- Antalet poster för ett attribut
COUNT(*)                        -- Totala antalet poster för hela tabellen
```

Notera

- **Alla aggregatfunktioner med angivet attribut ignorerar poster med `NULL`-värden.**
- **`COUNT(*)` räknar det totala antalet poster inklusive poster med `NULL`-värden!**

9.1. Exempel

- ***Medellönen för alla anställda:***

```
SELECT AVG(Lön) AS Medellön
FROM Lönespecifikation          -- Medellön: 20800
```

Lönespecifikation

AnställdID	Lön
5	18700
2	22500
10	21200
7	NULL

- ***Sammanlagd utbetalning:***

```
SELECT SUM(Lön) AS Totalt
FROM Lönespecifikation          -- Totalt: 62400
```

- ***Skillnaden mellan* `COUNT(Attribut)` *och* `COUNT(*)`*:***

```
SELECT COUNT(Lön) AS Utbetalningar:
FROM Lönespecifikation          -- Utbetalningar: 3

SELECT COUNT(*) AS AntalAnställda
FROM Lönespecifikation          -- AntalAnställda: 4 (även dom utan lön)
```

- ***Vem har lägst lön?***

```
SELECT AnställdID AS Anställd
FROM Lönespecifikation
WHERE Lön = (
    SELECT MIN(LÖN) FROM Lönespecifikation)   -- Anställd: 5
```

10. Gruppera och summera

- **Med `GROUP BY`-*uttrycket* så grupperas resulterande posterna efter angivet attribut**
- **Genom `HAVING`-*uttrycket* så anges ett *sökvillkor för grupperingen***
- **Genom att ange gruppering för flera attribut så skapas en *sorterad hierarki***

Syntax

- ```
  SELECT Attribut
  FROM Tabell
  [WHERE Sökvillkor]
  [GROUP BY Attribut]       -- Grupperar på angivet attribut
  [HAVING Sökvillkor]       -- Anger ett sökvillkor för grupperingen
  [ORDER BY Attribut]
  ```

Notera

- **`HAVING`-*uttrycket* används tillsammans med aggregatfunktioner!**
- **Självklart kan flera sökvillkor anges med hjälp av de logiska operatorerna, *ex.:***

```
HAVING NOT(Sökvillkor1 AND Sökvillkor2)
```
```

10.1. Exempel

- ***Grupperar det totala beloppet och listar alla anställda som får mer än 30000 utbetalat:***

```
SELECT AnställdID, SUM(Belopp) AS Totalt
FROM Utbetalningar
GROUP BY AnställdID
HAVING SUM(Belopp) > 30000

-- AnställdID: 2 | Belopp: 45000
```

Utbetalningar

AnställdID	Belopp	Månad
5	18700	Juli
5	1600	Juli
10	21200	Augusti
7	NULL	Juli
2	22500	Juli
2	22500	Augusti

- ***Grupperar medelutbetalningen för varje månad:***

```
SELECT Månad, AVG(Belopp) AS Medel
FROM Utbetalningar
GROUP BY Månad
ORDER BY Månad
```

Notera

- **När `GROUP BY` -*uttrycket* används så kan `SELECT` -uttrycket innehålla: *aggregatfunktioner*, attribut som används i grupperingen, eller *konstanta värden*.**

10.2. Skillnad mellan WHERE och HAVING

- **Med ett `WHERE`-*sökvillkor* så sker jämförelsen *direkt mot posterna* för attributet**
- **Med ett `HAVING`-*sökvillkor* så sker jämförelsen mot posterna *efter gruppering***
- **Ett `HAVING`-*uttryck* kan endast referera till attribut i `SELECT`-*uttrycket***
- **En *aggregatfunktion* kan endast användas som argument i ett `HAVING`-*uttryck***

Exempel

- ***Anställda med total utbetalning över 20000 (med `HAVING`):***

```
SELECT AnställdID, SUM(Belopp) AS Totalt
FROM Utbetalningar
GROUP BY AnställdID
HAVING SUM(Belopp) > 20000
```

- ***Anställda med total utbetalning över 20000 (med `WHERE`):***

```
SELECT AnställdID, SUM(Belopp) AS Totalt
FROM Utbetalningar
WHERE Belopp > 20000
GROUP BY AnställdID
```

Utbetalningar

AnställdID	Belopp	Månad
5	18700	Juli
5	1600	Juli
10	21200	Augusti
7	NULL	Juli
2	22500	Juli
2	22500	Augusti