# Mini SQL kursus

Vi skal udvide vores kendskab til sproget SQL v.h.a. små opgaver som findes på websitet:
<https://sqlzoo.net/>

## Simple SQL kommandoer

*LIKE*, bruges til søgninger hvor man ikke nødvendigvis har et præcist match.

Select \* From Personer Where Navn LIKE 'B%'; (Alle personer der starter med B)

*BETWEEN*, bruges til at finde tal der ligger i et interval

Select \* From Konto Where Saldo BETWEEN 0 AND 5000;

*IN*, bruges når der findes en liste af valgmuligheder

Select \* From Verden Where Landenavn IN ('Brasilien', 'Rusland', 'Indien');

*SUM*, bruges til at opsummere tal

Select SUM(alder) From Personer; (Giver den samlede alder af alle personer)

*Count*, bruges til at bestemme hvor mange datarækker der opfylder SQL (Where) betingelserne

Select Count(Landenavn) From Verden;

MAX, bruges til at bestemme højeste værdi

Select MAX(alder) From Personer; (Giver alder på ældste person)

DISTINCT\*, bruges til at fjerne dubletter fra ens data

Select Navn From Personer; (Giver samme navn flere gange)

Select DISTINCT Navn From Personer; (Giver samme navn kun en gang)

\*, DISTINCT bør bruges med meget omtanke da der ofte er en bedre løsning end DISTINCT. Særlig når arbejder med data fra flere tabeller skal man overveje ens løsning i det man let kommer til at stå med et optimeringsproblem (som i, det skal laves om) hvis man bruger DISTINCT.

ORDER BY med DESC og ASC, kender I nok allerede godt. Det bruges til at sortere data enten i stigende (ASC) orden eller faldende (DESC) orden

Select Navn From Personer Order By Navn Asc; (sorteret fra A til Å)

Select Navn From Personer Order By Navn Desc; (sorteret fra Å til A)

### Opgaver til ovenstående simple SQL kommandoer

Gå på websitet <https://sqlzoo.net/> og gennemgå opgaverne 0 til 3 samt 5 (4 kommer længere nede).

## Mere komplekse SQL kommandoer

*Select i Select*\*, bruges f.eks. hvis man har brug for at lave en betingelse baseret på et dataopslag

Select Landenavn From Verden Where Befolkningstal > (Select Befolkningstal From Verden Where Landenavn = 'Danmark');

\*, kan også bruges mellem forskellige tabeller således at man henter data fra flere tabeller på samme tid. Det er dog bedre at bruge JOINs
 **Opgaver til select i select**: Gå på websitet <https://sqlzoo.net/> og gennemgå opgaven 4.

*Forskellige JOINs*, bruges til at specificere hvorledes data fra to tabeller skal kombineres. Fra W3school (https://www.w3schools.com/sql/sql\_join.asp) er følgende illustration taget:



*Eksempel*:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table1 *( ”As T1” betyder alias er T1 for table1)*

|  |  |
| --- | --- |
| ID | FremmedNøgle\_ID |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | Null |

 | Table2 *( ”As T2” betyder alias er T2 for table2)*

|  |
| --- |
| ID |
| 1 |
| 2 |
| 3 |

 |

SQL: Select T1.ID, T2.ID From table1 As T1 ? Join table2 As T2 On t1.FremmedNøgle\_ID = t2.ID;

**Inner Join** (Join\*) er fællesmængden dvs. kun der hvor der er match i begge tabeller returneres data. Fra eksemplet (hvor ? er Inner) returneres:

|  |  |
| --- | --- |
| T1.ID | T2.ID |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |

\*, man kan vælge kun at skrive Join men det gør læsbarheden mindre så skriv altid Inner Join.

**Left Join** er alt fra table1 uanset match i table2

|  |  |
| --- | --- |
| T1.ID | T2.ID |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | Null |

 **Right Join** er alt fra table2 uanset match i table1

|  |  |
| --- | --- |
| T1.ID | T2.ID |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| Null | 3 |

 **Full Outer Join** er alt fra begge tabeller uanset om der er match

|  |  |
| --- | --- |
| T1.ID | T2.ID |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | Null |
| Null | 3 |

*NULL værdien*, er noget man ofte støder ind i når man arbejder med data fra databaser. Den er præcis som null fra div. programmeringssprog og derfor en kilde til hovedpine hvis man ikke res-pekterer den og håndterer den på en fornuftig måde. Fra ovenstående ”Full Outer Join” eksempel kan vi se at vi får to rækker med null tilbage fra database hvis vi lavede denne SQL efterspørgsel. Begynder man dernæst ukritisk at løbe datasættet igennem og tror at alle værdier er tal, vil ens program når det når til en null værdi, i bedste fald smide en fejl (som det håndterer så det ikke går ned), i værste fald går programmet ned. Derfor kan man lige så godt sikre sig mod null værdier allerede i databasen (med mindre særlige behov dikterer andet). Vi ændrer derfor vores SQL til:

Select Isnull(T1.ID,-1) , Isnull(T2.ID,-1) From table1 As T1 Full Outer Join table2 As T2 On t1.FremmedNøgle\_ID = t2.ID;

Nu bliver vores returdata (som er lettere at håndtere):

|  |  |
| --- | --- |
| T1.ID | T2.ID |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | -1 |
| -1 | 3 |

Isnull er en indbygget function i MySQL som returnerer den første parameter (T1.ID el. T2.ID) hvis den ikke er null, ellers returneres den anden parameter (-1). Man vælger således selv hvad Isnull funktionen skal returnere f.eks. Isnull(T1.ID, null) *(tænk lige over hvad denne funktion gør 😊)*.

### Opgaver til ovenstående komplekse SQL kommandoer

Gå på websitet <https://sqlzoo.net/> og gennemgå opgaverne 6 til 8.