MYSQL LABORATION 2

Även i den här laborationen arbetar vi från Shell/Terminalen.

- Börja med att ladda ner filerna countries.sql och cities.sql från min Github WSP_ViBo.
- Skapa en ny databas och välj den sedan med kommandot **USE**.
- Skriv **SOURCE** följt av sökvägen till de två ovanstående filerna som du laddat hem.

Exempel:

1. **CREATE DATABASE *lab2*;**
2. **USE *lab2*;**
3. **SOURCE ex: C:/Users/elev/Desktop/cities.sql**
4. **SOURCE ex: C:/Users/elev/countries.sql**

Testa nu att undersöka vad som finns i de olika tabellerna och hur de är uppbyggda. (**EXPLAIN** och **SELECT ***)

Bra!

NU SKA VI LEKA LITE MED DE INBYGGDA KOMMANDONA I MYSQL.

De kommandon vi kommer jobba med är följande:

- **SUM**(kolumn) - Räknar ut summan av en hel kolumn från ett resultat.
- **COUNT**(kolumn) - Räknar antalet rader i en kolumn från ett resultat.
- **CONCAT**(string1,string2...) - Slår ihop två eller flera strängar till en sträng.
- **CAST**(kolumn AS datatyp).
- **MIN**(kolumn) - Returnerar det minsta värdet i en kolumn från ett resultat.
- **MAX**(kolumn) - Returnerar det högsta värdet i en kolumn från ett resultat.
- **ABS**(värde) - Returnerar absolutbeloppet av ett tal
- Sedan har vi alla räknesätt så klart:
    - **/** - Division
    - ***** - Multiplikation
    - **+** - Addition
    - **-** - Subtratkion

Testa nu att summera kolumnen population i tabellen cities:
SELECT SUM(population) FROM cities;

> Observera att kolumnens namn i resultatet nu heter: SUM(population) vilket inte är ett så snyggt namn på en kolumn. Då kan vi ändra namn på den med kommandot AS.

Då skriver vi så här:
SELECT SUM(population) AS Världspopulation FROM cities;

Alltså efter kolumnens namn skriver vi AS följt av vad vi vill döpa om den till. Observera att om vi vill döpa om den till något med mellanslag i måste vi sätta det inom citattecken!

Exempel: SELECT SUM(population) AS "Total population" FROM cities;

Vill vi räkna antalet städer i tabellen cities kan vi använda COUNT().

SELECT COUNT(city) AS "Antal städer" FROM cities;

Observera att detta är ju samma siffra som det hade stått längst ned ifall vi hämtat all data ifrån cities. (Det brukar stå: XXX rows in set (0.00 sec))

COUNT() och SUM() är exempelvis bra när vi vill räkna ut medel av någon data. För om vi här delar totalen med antal rader så får vi ju medelpopulationen för alla städer:

SELECT SUM(population)/COUNT(city) AS Medelpopulation FROM cities;

Vill vi ta bort decimalerna får vi använda kommandot CAST().

CAST() är lite konstigt i det aspektet att man omvandlar ju data mellan olika datatyper men de datatyperna heter konstigt nog inte samma som när vi skapar tabeller… För att göra om till en sträng använder man CHAR och för att göra om till siffror använder man DECIMAL.

Hur som helst, om vi innesluter SUM(*population*)/COUNT(*city*) innanför CAST() och gör om det till datatypen DECIMAL så kan vi även specificera hur många decimaler som ska vara med.

SELECT CAST(SUM(population)/COUNT(city) AS DECIMAL(0)) AS Medelpopulation FROM cities;

DECIMAL(0) betyder att vi gör om ett tal eller sträng till ett heltal. Vill man ha till exempel exakt 3 siffror kan man byta ut 0:an mot en 3:a. Vill man ha X antal decimaler så lägger man till ett argument som motsvarar hur många decimaler man vill ha.

Exempel: CAST(123.456 AS DECIMAL(5,2)) blir då 123.46

5 STÅR FÖR HUR MÅNGA SIFFROR TOTALT OCH 2 ÄR ALLTSÅ HUR MÅNGA AV DEM SOM ÄR DECIMALER.

Alltså måste alltid den första siffran vara större än den sista. (Du kan ju inte ha fler decimaler än siffror eftersom en decimaltal är en siffra)

MIN() och **MAX()** returnerar, som man kan gissa, det minsta respektive det högsta värdet i en kolumn. Testa att hämta endast population för den stad med minst respektive högst population!

ABS() returnerar absolutbeloppet av varje tal i en kolumn. D.v.s. alla negativa tal blir positiva.

CONCAT() slår ihop strängar till en gemensam sträng. Det kan exempelvis användas om man har varsin kolumn för för- och efternamn och vill ha det tillsammans i en kolumn.

NU SKA VI BÖRJA JOBBA MED ALIAS OCH JOIN!

Alias används, som vi gjorde ovan, för att döpa om en tabell eller kolumn så den blir lättare att hantera eller förstå. Alias MÅSTE även användas ifall du vill ställa en fråga inuti en fråga *(Inception).*

*Exempelvis: SELECT First_name, Mail FROM (SELECT * FROM Customers) AS result;*

(VILKET FÖR ÖVRIGT ÄR EXAKT SAMMA SOM ATT HÄMTA FIRST_NAME OCH MAIL FRÅN CUSTOMERS…)

Alias skrivs alltså **AS** i SQL. Med andra ord: Man döper om ett resultat eller en kolumn.

Kommandot **JOIN** däremot används för att slå ihop två tabeller med varandra. Och är extremt användbart när man jobbar med relationsdatabaser! Syntaxen för **JOIN** lyder som följande:

Tabell1 JOIN Tabell2 ON Tabell1.kolumn=Tabell2.kolumn;

Den kommer alltså slå ihop tabell1 med tabell2 på(***ON***) alla värden där efterföljande villkor stämmer. Och man kan självklart ha fler villkor genom att lägga till **OR** eller **AND**.

> Observera att detta bara genererar ett resultat som är en tabell vilken man självklart måste hämta data ifrån. Så SELECT ska stå innan JOIN. Fullständigt exempel:

SELECT * FROM cities JOIN countries ON cities.country = countries.country;

D.v.s. Vi hämtar alla rader ifrån resultatet av de två tabellerna vi slagit ihop. Om vi skulle vilja specificera vilka kolumner vi vill titta på så måste vi specificera vilken tabell också.

Exempel: SELECT od.Order_number,p.Product_name,p.Price FROM Order_details AS od JOIN Products AS p ON p.Product_number=od.Product_number;

Observera att jag har döpt om Order_details till od och Products till p vilket gör det kortare att skriva!

GROUP / ORDER:

Alla resultat vi får kan vi sortera och gruppera utefter en, eller flera kolumner.

Kommandot **ORDER** säger nästan sig självt, att man ordnar sitt resultat efter en viss kolumn. T.ex. att man ordnar sitt resultat efter en kolumn med namn i bokstavsordning.

ORDER BY *kolumnnamn* (DESC/ASC); - Sorterar resultatet efter en kolumn med stigande/fallande ordning. Funkar på både bokstäver och siffror.

GROUP BY *kolumnnamn*; - Returnerar informationen så att varje unikt resultat i en kolumn bara förekommer EN gång. Detta är extremt användbart när man använder t.ex. **SUM()** och **COUNT()** som i sig själva bara returnerar en rad. Men tillsammans med **GROUP BY** så kan du exempelvis summera alla med ett visst unikt värde.

LIMIT:

Det sista kommandot vi ska kolla på är **LIMIT x**. Det är något man kan lägga till sist i sin fråga så kommer man bara få de x översta raderna från sitt resultat.

UPPGIFTER:

Vad ska man skriva för att:

1. Hämta de 10 största städerna från cities, sett till population.
2. Slå ihop cities med countries där country är samma för båda.
3. Hämta en tabell med enbart fullständigt namn på land och stad.
4. Hämta den stad som ligger närmast ekvatorn. (Latitude)
5. Hämta en tabell med totala populationen i varje land.
6. Hämta en tabell med medelstads-populationen för varje land.
7. Hämta det landsnamn som har den minsta staden i vår tabell.
8. Hämta alla länder som börjar på bokstaven S.
9. Hämta den största staden som börjar på A.
10. Hämta den stad som ligger närmast samma breddgrad som Stockholm. (Latitude)
11. Hämta den stad som ligger längst norrut.
12. Hämta den stad som har närmast samma population som Stockholm.

Svårare uppgifter:

A. Hämta den stad med störst population i varje unikt land.
B. Hämta den stad som ligger närmast Berlin fågelvägen.

```
SELECT ABS((SELECT Latitude FROM cities WHERE city = "Stockholm")-Latitude) AS skillnad FROM cities ORDER BY skillnad;
```