**Verwendete Befehle für Suche, 24.5.2018**

**mondial=#**

\d – zum Öffnen

Q – zum zurück gehen

\d Suchbegriff und um etwas anzuschauen – etwa: \d mountain

SELECT \* FROM – selektioniert etwas von einem bestehenden Table (etwa mountain).

SELECT Suchbegriff (etwa island) kreiert einen eigenen Table, den wir zur Anschauung schaffen wollen.

SELECT island, COUNT(island) FROM mountainonisland GROUP BY island;

Warum Klammer nach COUNT: Weil COUNT ein Kommando ist. Es ist im Kommando SELECT drin. Die Klammer ist nötig, damit der Computer weiss, dass (island) zum COUNT gehört.

Das Komma ist nötig, genauso wie das Strichpunkt am Schluss.

**Generell**: Nicht die Wörter sind zentral, sondern wo diese Wörter platziert sind. Beispiel: Country kann manchmal die Spalte bezeichnen, oder aber den Table oder die Reihe. country.code etwa ist eine Spalte, country.name etwas anderes.

GROUP BY ist die Bezeichnung einer Spalte, sucht nach einer Spalte

SELECT island, COUNT(island) FROM mountainonisland GROUP BY island ORDER BY COUNT(island) DESC;

SELECT island, COUNT(island) AS mostmn FROM mountainonisland GROUP BY island ORDER BY mostmn DESC;

mondial=# \d city

mondial=# SELECT \* FROM city;

mondial=# SELECT name, country, population, elevation FROM city;

**Zum Ordnen nach Meereshöhe:**

mondial=# SELECT name, country, population, elevation FROM city ORDER BY elevation;

**Zum Ordnen nach der höchsten Stadt zuerst:**

SELECT name, country, population, elevation FROM city ORDER BY elevation DESC NULLS LAST;

**Stadtbevölkerungen suchen und ordnen:**

SELECT SUM(population) FROM city;

**Länder nach Country-Codes und Stadtbevölkerung:**

SELECT country, SUM(population) FROM city GROUP BY country;

**Bessere Darstellung mit Ländernamen anstatt Country-Codes:**

SELECT SUM(city.population), country.name FROM city JOIN country ON country.code = city.country GROUP BY country.name;

**Um die Länder dann nach Bevölkerungszahl der Städte zu ordnen, der Grösse nach absteigend:**

SELECT SUM(city.population), country.name FROM city JOIN country ON country.code = city.country GROUP BY country.name ORDER BY SUM(city.population) DESC NULLS LAST;

**Berge:**

\d mountain

**danach**: SELECT \* FROM mountain;

**Mit Namen nach Höhe geordnet:**

SELECT name, mountains, elevation FROM mountain ORDER BY mountains, elevation;

SELECT name, mountains, elevation FROM mountain WHERE elevation > 4000 ORDER BY mountains, elevation;

**Der Höchste zuerst:**

SELECT name, mountains, elevation FROM mountain WHERE elevation > 4000 ORDER BY elevation DESC;

SELECT name, mountains, elevation FROM mountain WHERE mountains IS NULL ORDER BY elevation DESC LIMIT 33;

SELECT name, mountains, elevation FROM mountain WHERE mountains IS NULL AND elevation > 4000 ORDER BY elevation DESC LIMIT 10;

**Um herauszufinden, welche Bergkette die meisten Berge hat:**

SELECT mountains, COUNT(mountains) from mountain GROUP BY mountains ORDER BY COUNT(mountains) DESC;

Achtung!! Die Database sagt uns, es gebe 19 Berge in den Anden. In Wirklichkeit hat die Database einfach bei den Anden nur 19 Zeilen, die der Computer zählt.

**Um Höhe der Berge zusammenzuzählen:**

SELECT mountains, COUNT(mountains), SUM(elevation) from mountain GROUP BY mountains ORDER BY COUNT(mountains) DESC;

**Für durchschnittliche Höhe der Berge:**

SELECT mountains, COUNT(mountains), AVG(elevation) from mountain GROUP BY mountains ORDER BY COUNT(mountains) DESC;

**Für grenzüberschreitende Berge:**

SELECT \* FROM geo\_mountain ORDER BY mountain;

**Höchster Berg von Griechenland:**

SELECT geo\_mountain.country, mountain.name, mountain.elevation, mountain.type FROM geo\_mountain JOIN mountain ON mountain.name = geo\_mountain.mountain WHERE geo\_mountain.country = 'GR';

**Nach Höchstem abwärts geordnet:**

SELECT geo\_mountain.country, mountain.name, mountain.elevation, mountain.type FROM geo\_mountain JOIN mountain ON mountain.name = geo\_mountain.mountain WHERE geo\_mountain.country = 'GR'ORDER BY mountain.elevation DESC;

**Das ist die letzte und am schwersten zu verstehende Suche: Bergketten, nach Anzahl Bergen sortiert, mit einer Durchschnittshöhe angegeben.**

SELECT mountains, COUNT(mountains), AVG(elevation) from mountain WHERE mountains LIKE('A%') GROUP BY mountains HAVING AVG(elevation) > 3000 ORDER BY COUNT(mountains) DESC;