LEDE PROGRAM: DATA AND DATABASES DAY 2

Here's my summary yesterday's SQL fun in class.

First we listed the tables using the \d command.

```
mondial=# \d
               List of relations
 Schema |       Name        | Type  |  Owner
--------+-------------------+-------+----------
 public | airport           | table | postgres
 public | borders           | table | postgres
 public | city              | table | postgres
 public | citylocalname     | table | postgres
 public | cityothername     | table | postgres
 public | citypops          | table | postgres
 public | continent         | table | postgres
 public | country           | table | postgres
 public | countrylocalname  | table | postgres
 public | countryothername  | table | postgres
 public | countrypops       | table | postgres
 public | desert            | table | postgres
 public | economy           | table | postgres
 public | encompasses       | table | postgres
 public | ethnicgroup       | table | postgres
 public | geo_desert        | table | postgres
 public | geo_estuary       | table | postgres
 public | geo_island        | table | postgres
 public | geo_lake          | table | postgres
 public | geo_mountain      | table | postgres
 public | geo_river         | table | postgres
 public | geo_sea           | table | postgres
 public | geo_source        | table | postgres
 public | island            | table | postgres
 public | islandin          | table | postgres
 public | ismember          | table | postgres
 public | lake              | table | postgres
 public | lakeonisland      | table | postgres
 public | language          | table | postgres
 public | located           | table | postgres
 public | locatedon         | table | postgres
 public | mergeswith        | table | postgres
 public | mountain          | table | postgres
 public | mountainonisland  | table | postgres
 public | organization      | table | postgres
 public | politics          | table | postgres
 public | population        | table | postgres
 public | province          | table | postgres
 public | provincelocalname | table | postgres
 public | provinceothername | table | postgres
 public | provpops          | table | postgres
 public | religion          | table | postgres
 public | river             | table | postgres
 public | riveronisland     | table | postgres
 public | riverthrough      | table | postgres
 public | sea               | table | postgres
(46 rows)
```

Note that on a Mac control + f and control + b lets you page down or page up respectively-instead of pressing return or the down arrow to get one line at a time.
Also note that you can exit the list by pressing q. This will get you back to the prompt without having to view every single result.

Next we checked the columns of the table "mountainonisland" using \d tablename

```
mondial=# \d mountainonisland
                 Table "public.mountainonisland"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 mountain | character varying(50) |           | not null |
 island   | character varying(50) |           | not null |
```

```
Indexes:
    "mountainislkey" PRIMARY KEY, btree (mountain, island)
```

Then we selected all the columns and rows in that table.

```
mondial=# SELECT * FROM mountainonisland;
        mountain        |            island
------------------------+------------------------------
 GunnbjÃ¸rn Fjeld        | Greenland
 Newtontoppen           | Svalbard
 Hvannadalshnukur       | Iceland
 Snaefell               | Iceland
 Hekla                  | Iceland
 Katla                  | Iceland
 Higravstinden          | AustvÃ¥gÃ¸y
 Ben Nevis              | Great Britain
 Snowdon                | Great Britain
 SgÃ¹rr Alasdair        | Skye
 Carrauntoohil          | Ireland
 Slieve Donard          | Ireland
 Puig Major             | Mallorca
 Monte Cinto            | Corse
 Punta La Marmora       | Sardegna
 Monte Capanne          | Elba
 Etna                   | Sicilia
 Pizzo Carbonara        | Sicilia
 Aenos                  | Kefallinia
 Elati                  | Lefkas
 Dirfi                  | Euboea
 Fengari                | Samothraki
 Kerkis                 | Samos
 Pilineo                | Chios
 Pramnos                | Ikaria
 Attavyros              | Rhodos
 Psiloritis             | Crete
 Olympos                | Cyprus
 Pico de Teide          | Teneriffa
```

We decided to order the output by island to see if anything interesting appeared.

```
mondial=# SELECT * FROM mountainonisland ORDER BY island;
        mountain        |            island
------------------------+------------------------------
 Salahutu               | Ambon
 Ntringui               | Anjouan
 Higravstinden          | AustvÃ¥gÃ¸y
 Mt. Odin               | Baffin Island
 Agung                  | Bali
 La Soufriere           | Basse-Terre
 Pico Basile            | Bioko
 Besar                  | Borneo
 Bukit Raya             | Borneo
 Bukit Batubrok         | Borneo
 Murud                  | Borneo
 Siho                   | Borneo
 Mantam                 | Borneo
 Kinabalu               | Borneo
 Mt. Balbi              | Bougainville
 Kapalatmada            | Buru
 OsmeÃ±a Peak           | Cebu
 Binaiya                | Ceram
 Pilineo                | Chios
 Monte Cinto            | Corse
 Psiloritis             | Crete
 Pico Turquino          | Cuba
 Olympos                | Cyprus
 Treuter Mt.            | Devon Island
 Morne Diablotins       | Dominica
 Monte Capanne          | Elba
 Barbeau Peak           | Ellesmere Island
 Tabwemasana            | Espiritu Santo
 Dirfi                  | Euboea
```

From the look of Borneo, there are some islands that have multiple mountains on them. So we rushed into aggregate queries and decided

to use COUNT and GROUP BY to count how many mountains each island had. **Note** that we counted the column 'island' to see how many rows each island appeared in.

```
mondial=# SELECT island, COUNT(island) FROM mountainonisland GROUP BY island;
            island             | count
-------------------------------+-------
 Basse-Terre                   |     1
 Cyprus                        |     1
 Mallorca                      |     1
 Samosir                       |     1
 Unimak                        |     1
 Guadalcanal                   |     1
 Jamaica                       |     1
 Crete                         |     1
 Luzon                         |     6
 Lanai                         |     1
 Samos                         |     1
 Svalbard                      |     1
 Sao Jorge                     |     1
 Bougainville                  |     1
 Kolombangara                  |     1
 Grande Terre                  |     1
 Halmahera                     |     1
 Krenizyn                      |     1
 Chios                         |     1
 Isla da Ometepe               |     1
 Sumatra                       |    11
 Cuba                          |     1
 Fogo                          |     1
 Hispaniola                    |     2
 Upolu                         |     1
 Krakatau                      |     1
 Panay                         |     1
 Gomera                        |     1
 Teresa Island                 |     1
```

Next we tried to add the 'mountain' column to this query. But we got an error:

```
mondial=#
ERROR:  column "mountainonisland.mountain" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: SELECT island, COUNT(island), mountain FROM mountainonisland...
```

The error tells us that you can't just put a normal column into an aggregate query.

So we went ahead and decided to get an ordered list of the number of mountains on each island, using ORDER BY

```
mondial=# SELECT island, COUNT(island) FROM mountainonisland GROUP BY island ORDER BY COUNT(island);
            island             | count
-------------------------------+-------
 Faial                         |     1
 Cyprus                        |     1
 Mallorca                      |     1
 Samosir                       |     1
 Unimak                        |     1
 Guadalcanal                   |     1
 Jamaica                       |     1
 Crete                         |     1
 Kauai                         |     1
 Leyte                         |     1
 Oahu                          |     1
 Sardegna                      |     1
 Novaya Zemlya Yuzhny Island   |     1
 Negros                        |     1
 Goodenough Island             |     1
 Anjouan                       |     1
 Rhodos                        |     1
 Baffin Island                 |     1
```

Because ORDER BY defaults to ascending values ASC we saw the beginning of a list that starts with all the mountains that just have one island. So we used DESC to change the order--and get the highest values first.

```
mondial=# SELECT island, COUNT(island) FROM mountainonisland GROUP BY island ORDER BY COUNT(island) DESC;
```

```
            island             | count
-------------------------------+-------
 Sumatra                       |    11
 New Guinea                    |     9
 Java                          |     7
 Borneo                        |     7
 Sulawesi                      |     7
 Luzon                         |     6
 Honshu                        |     6
 Iceland                       |     4
 Mindanao                      |     4
 Madagaskar                    |     3
 Te Waka-a-Maui (South Island) |     3
 Hispaniola                    |     2
 Ireland                       |     2
 Flores                        |     2
 Te Ika-a-Maui (North Island)  |     2
 Great Britain                 |     2
 Hawaii                        |     2
 Sicilia                       |     2
 Jeju                          |     2
 Reunion                       |     2
 Timor                         |     2
 Upolu                         |     1
 Krakatau                      |     1
 Panay                         |     1
 Gomera                        |     1
 Teresa Island                 |     1
 Terceira                      |     1
```

Sumatra wins! At least according to the database. Next I introduced AS which allows you to rename columns (give them aliases). Below is the exact same query as above except that I renamed the column.

```
mondial=# SELECT island, COUNT(island) AS mostmn FROM mountainonisland GROUP BY island ORDER BY mostmn DESC;
            island             | mostmn
-------------------------------+--------
 Sumatra                       |     11
 New Guinea                    |      9
 Java                          |      7
 Borneo                        |      7
 Sulawesi                      |      7
 Luzon                         |      6
 Honshu                        |      6
 Iceland                       |      4
 Mindanao                      |      4
 Madagaskar                    |      3
 Te Waka-a-Maui (South Island) |      3
 Hispaniola                    |      2
 Ireland                       |      2
 Flores                        |      2
 Te Ika-a-Maui (North Island)  |      2
 Great Britain                 |      2
 Hawaii                        |      2
 Sicilia                       |      2
 Jeju                          |      2
 Reunion                       |      2
 Timor                         |      2
 Upolu                         |      1
 Krakatau                      |      1
 Panay                         |      1
 Gomera                        |      1
 Teresa Island                 |      1
 Terceira                      |      1
```

Next, we jumped out of aggregate queries, and into the city table

```
mondial=# \d city
                        Table "public.city"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 name       | character varying(50) |           | not null |
 country    | character varying(4)  |           | not null |
 province   | character varying(50) |           | not null |
 population | numeric               |           |          |
 latitude   | numeric               |           |          |
```

```
 longitude  | numeric               |           |          |
 elevation  | numeric               |           |          |
Indexes:
    "citykey" PRIMARY KEY, btree (name, country, province)
Check constraints:
    "citylat" CHECK (latitude >= '-90'::integer::numeric AND latitude <= 90::numeric)
    "citylon" CHECK (longitude >= '-180'::integer::numeric AND longitude <= 180::numeric)
    "citypop" CHECK (population >= 0::numeric)
```

We selected all of the columns to see what's in there.

```
mondial=# SELECT * FROM city;
               name                | country |                province                | population | latitude
-----------------------------------+---------+---------------------------------------+------------+----------
 Tirana                            | AL      | Albania                               |     418495 |    41.33
 ShkodÃ«r                          | AL      | Albania                               |      77075 |    42.07
 DurrÃ«s                           | AL      | Albania                               |     113249 |    41.32
 VlorÃ«                            | AL      | Albania                               |      79513 |    40.47
 Elbasan                           | AL      | Albania                               |      78703 |     41.1
 KorÃ§Ã«                           | AL      | Albania                               |      51152 |    40.62
 Komotini                          | GR      | Anatolikis Makedonias kai Thrakis     |            |     41.1
 Kavala                            | GR      | Anatolikis Makedonias kai Thrakis     |      58790 |    40.93
 Athina                            | GR      | Attikis                               |     664046 |    37.97
 Peiraias                          | GR      | Attikis                               |     163688 |    37.95
 Peristeri                         | GR      | Attikis                               |     139981 |    38.02
 Acharnes                          | GR      | Attikis                               |     106943 |    38.08
```

A lot of columns, not so easy to read. So we specified our columns:

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

name	country	population	elevation
Tirana	AL	418495	110
ShkodÃ«r	AL	77075	13
DurrÃ«s	AL	113249	0
VlorÃ«	AL	79513	25
Elbasan	AL	78703	150
KorÃ§Ã«	AL	51152	850
Komotini	GR		45
Kavala	GR	58790	0
Athina	GR	664046	70
Peiraias	GR	163688	0
Peristeri	GR	139981	50
Acharnes	GR	106943	186
Patra	GR	213984	0
Kozani	GR		710
Kerkyra	GR		0
Ioannina	GR	112486	480
Thessaloniki	GR	325182	0
Iraklio	GR	173993	150
Chania	GR	108642	0
Ermoupoli	GR		0
Rhodes	GR	115490	26
Tripoli	GR		655
Lamia	GR	75315	50
Chalkida	GR	102223	0
Larissa	GR	162591	67
Volos	GR	144449	0
Mytilini	GR		8

Much easier to read! So we decided to look for cities at the lowest elevation using ORDER BY

```
mondial=# SELECT name, country, population, elevation FROM city ORDER BY elevation;
               name                | country | population | elevation
-----------------------------------+---------+------------+-----------
 Baku                              | AZ      |    2150800 |       -28
 Astrakhan                         | R       |     527345 |       -28
 Atyrau                            | KAZ     |     196494 |       -20
 Aktau                             | KAZ     |     181526 |        -8
 David                             | PA      |     144858 |        -6
 Almere                            | NL      |     196244 |        -3
 Lelystad                          | NL      |      76252 |        -3
```

```
 Georgetown                        | GUY     |     118363 |        -2
 New Orleans                       | USA     |     343829 |        -2
 Babol                             | IR      |     250217 |        -2
 Bergen                            | N       |     267950 |         0
 Malabo                            | GQ      |      92900 |         0
 Kerkyra                           | GR      |            |         0
 Stockholm                         | S       |     881235 |         0
 Thessaloniki                      | GR      |     325182 |         0
 Chania                            | GR      |     108642 |         0
 Ermoupoli                         | GR      |            |         0
 Delhi                             | IND     |   11034555 |         0
 Chalkida                          | GR      |     102223 |         0
 Callao                            | PE      |     438326 |         0
 Volos                             | GR      |     144449 |         0
 Saint-Paul                        | REUN    |     103916 |         0
 Brest                             | F       |     140547 |         0
 Ajaccio                           | F       |      66245 |         0
 Le Havre                          | F       |     174156 |         0
 Praia                             | CV      |     131719 |         0
 Marseille                         | F       |     850636 |         0
```

Then we decided to look for the cities that the highest elevation by adding DESC to reverse the order.

```
mondial=# SELECT name, country, population, elevation FROM city ORDER BY elevation DESC;
               name                | country | population | elevation
-----------------------------------+---------+------------+-----------
 Shashi                            | CN      |     281352 |
 Victoria                          | SY      |      24970 |
 Jamestown                         | HELX    |            |
 Saint-Denis                       | REUN    |     145347 |
 Tekirdag                          | TR      |     150112 |
 Korla                             | CN      |     159344 |
 KarabÃ¼k                          | TR      |     110537 |
 Aksu                              | CN      |     164092 |
 Yalova                            | TR      |     102874 |
 Duzce                             | TR      |     135557 |
 Osmaniye                          | TR      |     209255 |
 Yining                            | CN      |     177193 |
 Shihezi                           | CN      |     299676 |
 Wulumuqi                          | CN      |    3029372 |
 Shizuishan                        | CN      |     257862 |
 Manzhouli                         | CN      |     120023 |
 Linhe                             | CN      |     133183 |
 Ulanhot                           | CN      |     159538 |
 Jining                            | CN      |     163552 |
 SorÃ¸                              | DK      |            |
 Wuzhou                            | CN      |     210452 |
 TÃ³rshavn                          | FARX    |      13130 |
 Mariehamn                         | SF      |      10851 |
 HÃ¤meenlinna                       | SF      |      67803 |
 Lahti                             | SF      |     103396 |
 Tampere                           | SF      |     220678 |
 Kuopio                            | SF      |     106475 |
 Lappeenranta                      | SF      |      72617 |
```

But something weird happened! We got all of these rows with no values for the elevation. This is because there were null values (missing values), and you have to tell the database what to do with them or they will show up somewhere. So we added NULLS LAST to the ORDER BY command, to make sure that they get pushed to the end of the list.

```
mondial=# SELECT name, country, population, elevation FROM city ORDER BY elevation DESC NULLS LAST;
               name                | country | population | elevation
-----------------------------------+---------+------------+-----------
 Cerro de Pasco                    | PE      |            |      4330
 Lhasa                             | CN      |     106885 |      4200
 El Alto                           | BOL     |     848840 |      4150
 PotosÃ                            | BOL     |     189652 |      4067
 Puno                              | PE      |     120229 |      3830
 Juliaca                           | PE      |     216716 |      3825
 Oruro                             | BOL     |     264683 |      3735
 Huancavelica                      | PE      |            |      3676
 La Paz                            | BOL     |     764617 |      3640
 Cusco                             | PE      |     348935 |      3399
 Huancayo                          | PE      |     323054 |      3259
```

```
 Huaraz                              | PE      |     100931 |      3052
 Quito                               | EC      |    1619146 |      2850
 Tunja                               | CO      |     184864 |      2820
 Sucre                               | BOL     |     259388 |      2810
 Ayacucho                            | PE      |     151019 |      2761
 Cajamarca                           | PE      |     162326 |      2750
 Sacaba                              | BOL     |     169494 |      2719
 Toluca                              | MEX     |     489333 |      2660
 BogotÃ¡                              | CO      |    7776845 |      2640
 Soacha                              | CO      |     500097 |      2566
 Cuenca                              | EC      |     331888 |      2560
 Cochabamba                          | BOL     |     630587 |      2558
 Pasto                               | CO      |     434486 |      2527
 Dessie                              | ETH     |     147592 |      2470
 Zacatecas                           | MEX     |     129011 |      2440
 Pachuca                             | MEX     |     256584 |      2432
```

Then we got into aggregates again, and compared the two different aggregate commands COUNT() and SUM()

```
mondial=# SELECT COUNT(population) FROM city;
 count
-------
  3051
(1 row)

mondial=# SELECT SUM(population) FROM city;
    sum
------------
 1762114278
(1 row)
```

There are a few interesting things here. COUNT() is adding up unique values in the column, not numbers. Most cities have different populations, but there are some that have to same number and a bunch that have missing entries. If you do a count of all the cities (SELECT COUNT(name) FROM city;) you will find that there are 3390 unique cities, as opposed to 3051 unique population numbers. SUM() is actually adding up all the numbers.
Also note that you don't need GROUP BY if you just want aggregate all of the values in a single column.

Then we added up the city populations for each country. Here we need GROUP BY.

```
mondial=# SELECT country, SUM(population) FROM city GROUP BY country;
 country |    sum
---------+-----------
 NEP     |   1699953
 RA      |  15351698
 CH      |   1381946
 L       |     99852
 AMSA    |
 WD      |     14725
 LB      |   1010970
 BF      |   1965190
 EW      |    502624
 MV      |    133019
 BEN     |   1378292
 CAYM    |
 SY      |     24970
 IRQ     |  12950153
 GROX    |
 NOK     |   9087275
 MNG     |    760077
 MW      |   1335704
 AFG     |   3371100
 NZ      |   1222122
 BI      |    497166
 WAFU    |
 WG      |     37000
 J       |  48415803
 I       |  14398216
 SK      |   1106091
 ARU     |
```

Now we want to know what the actual countries are, because these country codes are quite cryptic. So we need to look for the country table to get the full name.

```
mondial=# \d country
                      Table "public.country"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 name       | character varying(50) |           | not null |
 code       | character varying(4)  |           | not null |
 capital    | character varying(50) |           |          |
 province   | character varying(50) |           |          |
 area       | numeric               |           |          |
 population | numeric               |           |          |
Indexes:
    "countrykey" PRIMARY KEY, btree (code)
    "country_name_key" UNIQUE CONSTRAINT, btree (name)
Check constraints:
    "countryarea" CHECK (area >= 0::numeric)
    "countrypop" CHECK (population >= 0::numeric)

mondial=# SELECT * FROM country;
               name                | code |       capital        |             province              |   area   | pop
-----------------------------------+------+----------------------+-----------------------------------+----------+----
 Albania                           | AL   | Tirana               | Albania                           |    28750 |
 Greece                            | GR   | Athina               | Attikis                           |   131940 |   1
 Macedonia                         | MK   | Skopje               | Macedonia                         |    25333 |
 Serbia                            | SRB  | Beograd              | Serbia                            |    77474 |
 Montenegro                        | MNE  | Podgorica            | Montenegro                        |    14026 |
 Kosovo                            | KOS  | Prishtine            | Kosovo                            |    10887 |
 Andorra                           | AND  | Andorra la Vella     | Andorra                           |      450 |
 France                            | F    | Paris                | Ãžle-de-France                     |   547030 |
 Spain                             | E    | Madrid               | Madrid                            |   504750 |   4
 Austria                           | A    | Wien                 | Wien                              |    83850 |
 Czech Republic                    | CZ   | Praha                | Praha                             |    78703 |   1
 Germany                           | D    | Berlin               | Berlin                            |   356910 |   8
 Hungary                           | H    | Budapest             | Budapest                          |    93030 |
```

After a slight tangent investigating what the province column meant, we went ahead and did a JOIN between the two tables.

```
mondial=# SELECT SUM(city.population), country.name
mondial-# FROM city JOIN country ON country.code = city.country GROUP BY country.name;
    sum    |                name
-----------+-----------------------------------
   1145367 | Costa Rica
    703963 | Cambodia
    775000 | Eritrea
  46045206 | Turkey
    951418 | Chad
    207482 | Cyprus
     36735 | Samoa
    394368 | Slovenia
           | Cayman Islands
     50182 | Kiribati
  14446981 | Vietnam
    637411 | Kuwait
    662426 | Jamaica
     22219 | Antigua and Barbuda
   5002435 | Cameroon
  10367222 | Saudi Arabia
    131719 | Cape Verde
    622712 | Macedonia
   1047048 | Bosnia and Herzegovina
           | Guinea-Bissau
    125000 | Curacao
   5454065 | Netherlands
           | Svalbard
    357238 | Gambia
  22867466 | Zaire
  49410900 | Nigeria
   1100000 | Lebanon
```

Some things to note about JOIN: first you need to use DOT notation (city.population) to specify the columns you are selecting. The first part is name of the table (city.) The second part is the column from that table (.population).
JOIN is part of the FROM command. You put the first table after FROM then the second table after JOIN and then the criteria for joining (country.code = city.country) comes after ON. To put this in order that makes sense:
FROM firsttable JOIN secondtable ON condition (country.code = city.country)

Next we ordered the resulting joined table. Note that all the nulls came up first. But then the countries with large populations showed up after them.

```
 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;
    sum    |               name
-----------+----------------------------------
           | Turks and Caicos Islands
           | French Polynesia
           | Nauru
           | Jersey
           | Cook Islands
           | Macao
           | Norfolk Island
           | Wallis and Futuna
           | British Virgin Islands
           | Guernsey
           | Somalia
           | Northern Mariana Islands
           | Cayman Islands
           | Anguilla
           | Isle of Man
           | Greenland
           | Guam
           | American Samoa
           | Falkland Islands
           | Bermuda
           | Cocos Islands
           | Mayotte
           | Gibraltar
           | Guinea-Bissau
           | Saint Helena
           | Guadeloupe
           | Svalbard
           | Aruba
           | Christmas Island
           | Mauritius
           | Niue
 326058186 | China
 129752758 | India
  93687185 | Brazil
  81882315 | United States
  72000673 | Russia
  50920843 | Pakistan
  49410900 | Nigeria
  48415803 | Japan
  47137170 | Mexico
  46627466 | Indonesia
  46045206 | Turkey
  34845642 | Iran
  33570031 | South Korea
  26579496 | Colombia
  25333235 | Germany
```

Again, we pushed down the null values.

```
 FROM city JOIN country ON country.code = city.country GROUP BY country.name ORDER BY SUM(city.population) DESC NUI
    sum    |               name
-----------+----------------------------------
 326058186 | China
 129752758 | India
  93687185 | Brazil
  81882315 | United States
  72000673 | Russia
  50920843 | Pakistan
  49410900 | Nigeria
  48415803 | Japan
  47137170 | Mexico
  46627466 | Indonesia
  46045206 | Turkey
  34845642 | Iran
  33570031 | South Korea
  26579496 | Colombia
  25333235 | Germany
  25252422 | United Kingdom
```

```
  23470701 | South Africa
  22867466 | Zaire
  22364857 | Egypt
  19594294 | Philippines
  18553641 | Spain
  17198533 | Ukraine
  16915048 | Australia
  16481172 | Taiwan
  15670355 | Canada
  15351698 | Argentina
  14607665 | Bangladesh
```

Then we moved into 'mountain'

```
 mondial=# \d mountain
                      Table "public.mountain"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 name        | character varying(50) |           | not null |
 mountains   | character varying(50) |           |          |
 elevation   | numeric               |           |          |
 type        | character varying(10) |           |          |
 coordinates | geocoord              |           |          |
Indexes:
    "mountainkey" PRIMARY KEY, btree (name)
Check constraints:
    "mountaincoord" CHECK ((coordinates).latitude >= '-90'::integer::numeric AND (coordinates).latitude <= 90::nume
```

```
mondial=# SELECT * FROM mountain;
              name               |          mountains           | elevation |   type   |   coordinates
---------------------------------+------------------------------+-----------+----------+------------------
 GunnbjÃ¸rn Fjeld                 |                              |      3694 |          | (68.92,-29.9)
 Newtontoppen                    |                              |      1713 |          | (79.0,17.3)
 Hvannadalshnukur                |                              |      2110 | volcanic | (64.1,-16.7)
 Snaefell                        |                              |      1833 | volcanic | (64.5,-15.2)
 Hekla                           |                              |      1491 | volcano  | (64,-19.7)
 Katla                           |                              |      1450 | volcano  | (63.6,-19.0)
 Haltiatunturi                   | Scandinavian Mountains       |      1365 |          | (69.3,21.27)
 Kebnekaise                      | Scandinavian Mountains       |      2099 |          | (67.9,18.5)
 Sarektjokko                     | Scandinavian Mountains       |      2089 |          | (67.43,17.71)
 Higravstinden                   |                              |      1146 |          | (68.36,14.79)
 Galdhoeppig                     | Scandinavian Mountains       |      2469 |          | (61.64,8.31)
 Glittertind                     | Scandinavian Mountains       |      2465 |          | (61.65,8.55)
 Snoehetta                       | Scandinavian Mountains       |      2286 |          | (62.32,9.27)
 Ben Nevis                       | Grampians                    |      1344 |          | (56.8,-5.0)
 Snowdon                         |                              |      1015 |          | (53.08,-4.08)
 SgÃ¹rr Alasdair                 |                              |       993 |          | (57.21,-6.22)
 Carrauntoohil                   | Armorican Highlands          |      1041 |          | (52.00,-9.74)
 Slieve Donard                   |                              |       849 |          | (54.18,-5.92)
 Feldberg                        | Black Forest                 |      1493 |          | (47.87,8.00)
 Brocken                         | Harz                         |      1141 |          | (51.8,10.6)
 Grosser Arber                   | Bayrischer Wald              |      1456 |          | (49.1,13)
 Zugspitze                       | Alps                         |      2963 |          | (47.42,10.99)
 Barre des Ecrins                | Alps                         |      4101 |          | (44.92,6.36)
 Mont Ventoux                    | Alps                         |      1912 |          | (44.18,5.28)
 Gran Paradiso                   | Alps                         |      4061 |          | (45.5,7.25)
 Mont Blanc                      | Alps                         |      4808 |          | (45.83,6.86)
 Grand Combin                    | Alps                         |      4314 |          | (45.93,7.30)
```

We ordered it alphabetically, to see if there were ranges with multiple mountains. And we narrowed down our columns. Note that we ordered first alphabetically, and then by elevation.

```
 mondial=# SELECT name, mountains, elevation FROM mountain ORDER BY mountains, elevation;
              name               |          mountains           | elevation
---------------------------------+------------------------------+-----------
 Mt. Marcy                       | Adirondacks                  |      1629
 Tahat                           | Ahaggar                      |      3003
 Fansipan                        | Ailao Shan                   |      3143
 Jebel Shams                     | Al Hajar Mountains           |      3009
 Ayrybaba                        | Alai                         |      3138
 Tschimtarga                     | Alai                         |      5489
 Pik Tandykul                    | Alai                         |      5544
 Mt. Ratz                        | Alaska Boundary Range        |      3090
 Mt. McKinley (Denali)           | Alaska Range                 |      6193
 Mt. Redoubt                     | Aleutian Range               |      3108
 Mont Ventoux                    | Alps                         |      1912
```

```
 Hochgolling                  | Alps                         |      2862
 Triglav                      | Alps                         |      2864
 Zugspitze                    | Alps                         |      2963
 Marmolata                    | Alps                         |      3343
 Grossglockner                | Alps                         |      3797
 Piz Bernina                  | Alps                         |      4048
 Gran Paradiso                | Alps                         |      4061
 Barre des Ecrins             | Alps                         |      4101
 Finsteraarhorn               | Alps                         |      4274
 Grand Combin                 | Alps                         |      4314
 Matterhorn                   | Alps                         |      4478
 Monte Rosa                   | Alps                         |      4634
 Mont Blanc                   | Alps                         |      4808
 KhÃ¼iten Peak                | Altai                        |      4374
 Bjelucha                     | Altai                        |      4506
 Altun Shan Peak              | Altyn-Tagh                   |      5830
```

Next we used WHERE to only get mountains with elevations above 4000.

```
 mondial=# SELECT name, mountains, elevation FROM mountain WHERE elevation > 4000 ORDER BY mountains, elevation;
              name               |          mountains           | elevation
---------------------------------+------------------------------+-----------
 Tschimtarga                     | Alai                         |      5489
 Pik Tandykul                    | Alai                         |      5544
 Mt. McKinley (Denali)           | Alaska Range                 |      6193
 Piz Bernina                     | Alps                         |      4048
 Gran Paradiso                   | Alps                         |      4061
 Barre des Ecrins                | Alps                         |      4101
 Finsteraarhorn                  | Alps                         |      4274
 Grand Combin                    | Alps                         |      4314
 Matterhorn                      | Alps                         |      4478
 Monte Rosa                      | Alps                         |      4634
 Mont Blanc                      | Alps                         |      4808
 KhÃ¼iten Peak                   | Altai                        |      4374
 Bjelucha                        | Altai                        |      4506
 Altun Shan Peak                 | Altyn-Tagh                   |      5830
 Sulamutag Feng                  | Altyn-Tagh                   |      6245
 Monte San Valentin              | Andes                        |      4058
 Maipo                           | Andes                        |      5264
 Licancabur                      | Andes                        |      5920
 Nevado El Plomo                 | Andes                        |      6070
 Marmolejo                       | Andes                        |      6108
 Tupungato                       | Andes                        |      6550
 Llullaillaco                    | Andes                        |      6739
 Monte Pissis                    | Andes                        |      6795
 Ojos del Salado                 | Andes                        |      6893
 Aconcagua                       | Andes                        |      6962
 SÃ¼phan DaÄŸÄ±                   | Armenian Highlands           |      4058
 Ararat                          | Armenian Highlands           |      5165
```

Then we stopped ordering alphabetically, and just by elevation.

```
 mondial=# SELECT name, mountains, elevation FROM mountain WHERE elevation > 4000 ORDER BY elevation;
              name               |          mountains           | elevation
---------------------------------+------------------------------+-----------
 Boundary Peak                   | Sierra Nevada California     |      4006
 Otgon Tenger                    | Khangai Range                |      4008
 Wheeler Peak                    | Rocky Mountains              |      4011
 Mt. Waddington                  | Pacific Ranges               |      4019
 Mt. Victoria                    | Owen Stanley Range           |      4038
 Piz Bernina                     | Alps                         |      4048
 Monte San Valentin              | Andes                        |      4058
 SÃ¼phan DaÄŸÄ±                   | Armenian Highlands           |      4058
 Gran Paradiso                   | Alps                         |      4061
 Aragaz                          | Lesser Kaukasus              |      4090
 Fako                            | Cameroon Line                |      4095
 Kinabalu                        | Crocker Range                |      4096
 Barre des Ecrins                | Alps                         |      4101
 Kings Peak                      | Rocky Mountains              |      4123
 Pacha Mama                      |                              |      4138
 Mt. Boising                     | Finisterre Range             |      4150
 Tubkhal                         | Atlas                        |      4167
 Mauna Loa                       | Hawaii                       |      4170
 Gannett Peak                    | Rocky Mountains              |      4207
 Mauna Kea                       | Hawaii                       |      4214
```

```
 Tajumulco                     | Sierra Madre de Chiapas      |      4220
 Finsteraarhorn                | Alps                         |      4274
 Pikes Peak                    | Rocky Mountains              |      4302
 Grand Combin                  | Alps                         |      4314
 Elgon                         | East African Rift            |      4321
 Nevado de Colima              | Sierra Volcanica Transversal |      4330
 Mt. Giluwe                    | Bismarck Range               |      4368
```

Then we changed the order to get the highest elevation first. Note that we don't need to deal with null values because the WHERE command is making sure we only get values above 4000.

```
 mondial=# SELECT name, mountains, elevation FROM mountain WHERE elevation > 4000 ORDER BY elevation DESC;
             name              |          mountains           | elevation
-------------------------------+------------------------------+-----------
 Mt. Everest                   | Himalaya                     |      8848
 K2                            | Karakorum                    |      8611
 Kangchendzonga                | Himalaya                     |      8586
 Lhotse                        | Himalaya                     |      8516
 Makalu                        | Himalaya                     |      8485
 Cho Oyu                       | Himalaya                     |      8188
 Dhaulagiri                    | Himalaya                     |      8167
 Manaslu                       | Himalaya                     |      8163
 Nanga Parbat                  | Himalaya                     |      8125
 Annapurna                     | Himalaya                     |      8091
 Gasherbrum I                  | Karakorum                    |      8080
 Broad Peak                    | Karakorum                    |      8051
 Gasherbrum II                 | Karakorum                    |      8034
 Shishapangma                  | Himalaya                     |      8027
 Nanda Devi                    | Himalaya                     |      7816
 Batura Sar                    | Karakorum                    |      7795
 Namcha Barwa                  | Himalaya                     |      7782
 Saltoro Kangri                | Karakorum                    |      7742
 Tirich Mir                    | Hindukush                    |      7708
 Gurla Mandhata                | Himalaya                     |      7694
 Saser Kangri                  | Karakorum                    |      7672
 Kongur                        | Pamir                        |      7649
 Gangkhar Puensum              | Himalaya                     |      7570
 Gongga Shan                   | Hengduan Shan                |      7556
 Muztagh Ata                   | Pamir                        |      7509
 Pik Ismoil Somoni             | Pamir                        |      7495
 Noshaq                        | Hindukush                    |      7492
 Pik Pobeda                    | Tian Shan                    |      7439
```

Here I introduced a different condition in the WHERE command BETWEEN

```
mondial=# SELECT name, mountains, elevation FROM mountain WHERE elevation  4000 AND 6000 ORDER BY elevation DESC;
             name              |          mountains           | elevation
-------------------------------+------------------------------+-----------
 Alto Toroni                   | Cordillera Volcanica         |      5982
 Mt. Logan                     | Elias Range                  |      5959
 Alpamayo                      | Cordillera Blanca            |      5947
 Licancabur                    | Andes                        |      5920
 Cotopaxi                      | Cordillera Occidental        |      5897
 Kilimanjaro                   | East African Rift            |      5895
 Hkakabo Razi                  | Hengduan Shan                |      5881
 OllagÃ¼e                       | Cordillera Volcanica         |      5870
 Altun Shan Peak               | Altyn-Tagh                   |      5830
 Kangze'gyai                   | Qiliang Shan                 |      5808
 Cayambe                       | Cordillera Occidental        |      5796
 Pico Cristobal Colon          | Sierra Nevada de Santa Marta |      5775
 Nevado del Huila              | Cordillera Occidental        |      5750
 Zapaleri                      | Cordillera de LÃpez          |      5653
 Elbrus                        | Kaukasus                     |      5642
 Citlaltepetl (Pico de Orizaba) | Sierra Volcanica Transversal |      5636
```

Here we used the LIMIT command to only show a certain number of rows. This shows the first 10:

```
mondial=# SELECT name, mountains, elevation FROM mountain WHERE elevation BETWEEN 4000 AND 6000 ORDER BY elevation
      name       |       mountains       | elevation
-----------------+-----------------------+-----------
 Alto Toroni     | Cordillera Volcanica  |      5982
 Mt. Logan       | Elias Range           |      5959
 Alpamayo        | Cordillera Blanca     |      5947
```

```
 Licancabur      | Andes                 |      5920
 Cotopaxi        | Cordillera Occidental |      5897
 Kilimanjaro     | East African Rift     |      5895
 Hkakabo Razi    | Hengduan Shan         |      5881
 OllagÃ¼e         | Cordillera Volcanica  |      5870
 Altun Shan Peak | Altyn-Tagh            |      5830
 Kangze'gyai     | Qiliang Shan          |      5808
(10 rows)
```

This also includes OFFSET to show the next 10 rows:

```
mondial=# SELECT name, mountains, elevation FROM mountain WHERE elevation BETWEEN 4000 AND 6000 ORDER BY elevation
              name              |          mountains           | elevation
--------------------------------+------------------------------+-----------
 Cayambe                        | Cordillera Occidental        |      5796
 Pico Cristobal Colon           | Sierra Nevada de Santa Marta |      5775
 Nevado del Huila               | Cordillera Occidental        |      5750
 Zapaleri                       | Cordillera de LÃpez          |      5653
 Elbrus                         | Kaukasus                     |      5642
 Citlaltepetl (Pico de Orizaba) | Sierra Volcanica Transversal |      5636
 Damavand                       | Elburs                       |      5610
 Shanzidou                      | Hengduan Shan                |      5596
 Qilian Shan Peak               | Qiliang Shan                 |      5547
 Pik Tandykul                   | Alai                         |      5544
(10 rows)
```

Here we used the IS NULL in the WHERE command to look for mountains with missing ranges.

```
mondial=# SELECT name, mountains, elevation FROM mountain WHERE mountains IS NULL ORDER BY elevation DESC LIMIT 10
    name    | mountains | elevation
------------+-----------+-----------
 Semeru     |           |      3676
 Rantemario |           |      3478
 Baru       |           |      3475
 Koryaksky  |           |      3456
 Slamet     |           |      3428
 Raung      |           |      3332
 Etna       |           |      3323
 Shiveluch  |           |      3307
 Hasan Dagi |           |      3268
 Kita-Dake  |           |      3193
(10 rows)
```

Here we used AND in the WHERE command to look for mountains with missing ranges AND that had elevations higher than 4000.

```
SELECT name, mountains, elevation FROM mountain WHERE mountains IS NULL AND elevation > 4000 ORDER BY elevation DES
         name          | mountains | elevation
-----------------------+-----------+-----------
 Kljutschewskaja Sopka |           |      4750
 Hazaran               |           |      4500
 Pacha Mama            |           |      4138
```

Here is another aggregate query counting mountains in ranges.

```
mondial=# SELECT mountains, COUNT(mountains) from mountain GROUP BY mountains ORDER BY COUNT(mountains) DESC;
          mountains           | count
------------------------------+-------
 Andes                        |    19
 Himalaya                     |    15
 Alps                         |    14
 Rocky Mountains              |    13
 Barisan Mountains            |    12
 East African Rift            |    11
 Hawaii                       |     7
 Karakorum                    |     7
 Pamir                        |     7
 Dinaric Alps                 |     6
 Kaukasus                     |     6
 Scandinavian Mountains       |     6
 Pyrenees                     |     5
 Cameroon Line                |     5
```

```
 Canary Islands               |     5
 Apennin                      |     5
 Cordillera Occidental        |     5
 Azores                       |     5
 Sierra Volcanica Transversal |     5
 Tian Shan                    |     5
 Atlas                        |     5
 Ural                         |     5
 Cordillera Volcanica         |     5
 Annamite Range               |     4
 Elias Range                  |     4
 Hengduan Shan                |     4
 Cascade Range                |     4
```

Here we added a new column to the aggregate query to get total elevations

```
SELECT mountains, COUNT(mountains), SUM(elevation) from mountain GROUP BY mountains ORDER BY COUNT(mountains) DESC,
          mountains           | count |  sum
------------------------------+-------+--------
 Andes                        |    19 |  90721
 Himalaya                     |    15 | 121193
 Alps                         |    14 |  52459
 Rocky Mountains              |    13 |  48021
 Barisan Mountains            |    12 |  31423
 East African Rift            |    11 |  45958
 Hawaii                       |     7 |  16798
 Karakorum                    |     7 |  55985
 Pamir                        |     7 |  50238
 Dinaric Alps                 |     6 |  13466
 Kaukasus                     |     6 |  28890
 Scandinavian Mountains       |     6 |  12773
 Pyrenees                     |     5 |  15213
 Cameroon Line                |     5 |  14560
 Canary Islands               |     5 |  11081
 Apennin                      |     5 |  12092
 Cordillera Occidental        |     5 |  29119
 Azores                       |     5 |   6573
 Sierra Volcanica Transversal |     5 |  25348
 Tian Shan                    |     5 |  28807
 Atlas                        |     5 |  12583
 Ural                         |     5 |   7736
 Cordillera Volcanica         |     5 |  31107
 Annamite Range               |     4 |  10542
 Elias Range                  |     4 |  21116
 Hengduan Shan                |     4 |  25773
 Cascade Range                |     4 |  14046
```

Here we used AVG() to get average elevations

```
SELECT mountains, COUNT(mountains), AVG(elevation) from mountain GROUP BY mountains ORDER BY COUNT(mountains) DESC,
          mountains           | count |          avg
------------------------------+-------+-----------------------
 Andes                        |    19 | 4774.7894736842105263
 Himalaya                     |    15 | 8079.5333333333333333
 Alps                         |    14 | 3747.0714285714285714
 Rocky Mountains              |    13 | 3693.9230769230769231
 Barisan Mountains            |    12 | 2618.5833333333333333
 East African Rift            |    11 | 4178.0000000000000000
 Hawaii                       |     7 | 2399.7142857142857143
 Karakorum                    |     7 | 7997.8571428571428571
 Pamir                        |     7 | 7176.8571428571428571
 Dinaric Alps                 |     6 | 2244.3333333333333333
 Kaukasus                     |     6 | 4815.0000000000000000
 Scandinavian Mountains       |     6 | 2128.8333333333333333
 Pyrenees                     |     5 | 3042.6000000000000000
 Cameroon Line                |     5 | 2912.0000000000000000
 Canary Islands               |     5 | 2216.2000000000000000
 Apennin                      |     5 | 2418.4000000000000000
 Cordillera Occidental        |     5 | 5823.8000000000000000
 Azores                       |     5 | 1314.6000000000000000
 Sierra Volcanica Transversal |     5 | 5069.6000000000000000
 Tian Shan                    |     5 | 5761.4000000000000000
 Atlas                        |     5 | 2516.6000000000000000
 Ural                         |     5 | 1547.2000000000000000
 Cordillera Volcanica         |     5 | 6221.4000000000000000
```

```
 Annamite Range               |     4 | 2635.5000000000000000
 Elias Range                  |     4 | 5279.0000000000000000
 Hengduan Shan                |     4 | 6443.2500000000000000
 Cascade Range                |     4 | 3511.5000000000000000
```

We got a little sidetracked with geo_mountain, see below.
But first, to stick with this query, we went on to filter out averages below 3001 elevation (that is, selecting averages > 3000). We used HAVING which is like WHERE but for aggregates. HAVING must come after GROUP BY because it is a test on the aggregates.

```
mondial=# SELECT mountains, COUNT(mountains), AVG(elevation) from mountain GROUP BY mountains
mondial-# HAVING AVG(elevation) > 3000
mondial-# ORDER BY COUNT(mountains) DESC;
          mountains           | count |          avg
------------------------------+-------+-----------------------
 Andes                        |    19 | 4774.7894736842105263
 Himalaya                     |    15 | 8079.5333333333333333
 Alps                         |    14 | 3747.0714285714285714
 Rocky Mountains              |    13 | 3693.9230769230769231
 East African Rift            |    11 | 4178.0000000000000000
 Pamir                        |     7 | 7176.8571428571428571
 Karakorum                    |     7 | 7997.8571428571428571
 Kaukasus                     |     6 | 4815.0000000000000000
 Tian Shan                    |     5 | 5761.4000000000000000
 Cordillera Occidental        |     5 | 5823.8000000000000000
 Pyrenees                     |     5 | 3042.6000000000000000
 Cordillera Volcanica         |     5 | 6221.4000000000000000
 Sierra Volcanica Transversal |     5 | 5069.6000000000000000
 Cascade Range                |     4 | 3511.5000000000000000
 Hengduan Shan                |     4 | 6443.2500000000000000
 Elias Range                  |     4 | 5279.0000000000000000
 Alai                         |     3 | 4723.6666666666666667
 Asir Mountains               |     3 | 3054.3333333333333333
 Cordillera Blanca            |     3 | 6450.0000000000000000
 Maoke Mountains              |     3 | 4798.0000000000000000
 Armenian Highlands           |     3 | 4057.0000000000000000
 Kunlun                       |     3 | 7000.0000000000000000
 Sierra Madre de Chiapas      |     3 | 3574.0000000000000000
 Altai                        |     2 | 4440.0000000000000000
 Cordillera de LÃpez          |     2 | 5830.5000000000000000
 Lesser Kaukasus              |     2 | 3907.0000000000000000
 Sredinny Range               |     2 | 3111.5000000000000000
```

You can have WHERE and HAVING in the same query. WHERE always comes first, and filters out the original table, before aggregation takes place. Also note that we used a different conditional LIKE() which looks for patterns in language.

```
mondial=# SELECT mountains, COUNT(mountains), AVG(elevation) from mountain
mondial-# WHERE mountains LIKE('A%')
mondial-# GROUP BY mountains
mondial-# HAVING AVG(elevation) > 3000
mondial-# ORDER BY COUNT(mountains) DESC;
       mountains        | count |          avg
------------------------+-------+-----------------------
 Andes                  |    19 | 4774.7894736842105263
 Alps                   |    14 | 3747.0714285714285714
 Asir Mountains         |     3 | 3054.3333333333333333
 Alai                   |     3 | 4723.6666666666666667
 Armenian Highlands     |     3 | 4057.0000000000000000
 Altai                  |     2 | 4440.0000000000000000
 Altyn-Tagh             |     2 | 6037.5000000000000000
 Ahaggar                |     1 | 3003.0000000000000000
 Ailao Shan             |     1 | 3143.0000000000000000
 Arakan Mountains       |     1 | 3070.0000000000000000
 Aleutian Range         |     1 | 3108.0000000000000000
 Al Hajar Mountains     |     1 | 3009.0000000000000000
 Alaska Boundary Range  |     1 | 3090.0000000000000000
 Alaska Range           |     1 | 6193.0000000000000000
(14 rows)
```

Finally, here's the search we did for mountains in Greece! This joins the geo_mountain and mountains tables, searches for the country code, and orders by elevation.

```
mondial=# SELECT geo_mountain.country, mountain.name, mountain.elevation, mountain.type
mondial-# FROM geo_mountain JOIN mountain ON mountain.name = geo_mountain.mountain
```

```
mondial-# WHERE geo_mountain.country = 'GR'
mondial-# ORDER BY mountain.elevation DESC;
 country |      name       | elevation | type
---------+-----------------+-----------+------
 GR      | Olymp           |      2917 |
 GR      | Olymp           |      2917 |
 GR      | Smolikas        |      2637 |
 GR      | Profitis Ilias  |      2497 |
 GR      | Psiloritis      |      2456 |
 GR      | Kyllini         |      2376 |
 GR      | Athos           |      2033 |
 GR      | Dirfi           |      1743 |
 GR      | Aenos           |      1628 |
 GR      | Fengari         |      1611 |
 GR      | Kerkis          |      1433 |
 GR      | Pilineo         |      1297 |
 GR      | Attavyros       |      1215 |
 GR      | Elati           |      1158 |
 GR      | Pramnos         |      1037 |
(15 rows)
```