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. If you have a Windows machine and you have figured out how to, do this please post on slack!

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 I showed you how leaving out the semicolon ; at the end will give you line after line of unexecuted commands until you finally put in the semicolon. Note that when the command prompt has an = you are at a fresh line (the first line of your query):

mondial=#

When the prompt has a - you are extending the query over more than one line:

mondial-#

```
mondial2=# SELECT * FROM mountainonisland
mondial2-#
mondial2-#
mondial2-#
mondial2-#
mondial2-#
mondial2-#
mondial2-# \
Invalid command \. Try \? for help.
mondial2-# SELECT *
mondial2-# FROM mountainonisland
mondial2-# ;
ERROR:  syntax error at or near "SELECT"
LINE 2: SELECT *
        ^
mondial2=# SELECT *
mondial2-# FROM mountainonisland
mondial2-# ;
```

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
 Pico de las Nieves     | Gran Canaria
 Pico de Malpaso        | Hierro
 Garajonay              | Gomera
 Roque de los Muchachos | La Palma
 Pico Ruivo             | Madeira
 Pico                   | Pico
```

```
 Pico da Vara           | Sao Miguel
 Pico da EsperanÃ§a      | Sao Jorge
 CabeÃ§o Gordo           | Faial
 Serra de Santa BÃ¡rbara | Terceira
 Fogo                   | Fogo
 Pico da AntÃ³nia        | Santiago
 Pico Basile            | Bioko
 Pico de Sao Tome       | Sao Tome
 Queen Marys Peak       | Tristan Da Cunha
 Jabal Hajhir           | Sokotra
 Tsaratanana            | Madagaskar
 Tsiafajavona           | Madagaskar
 Andringitra            | Madagaskar
 Piton des Neiges       | Reunion
 Piton de la Fournaise  | Reunion
 Karthala               | Grand Comoro
 Ntringui               | Anjouan
 Pik Sedova             | Novaya Zemlya Severny Island
 Gora Pervousmotrennaya | Novaya Zemlya Yuzhny Island
 Pidurutalagala         | Sri Lanka
 Chikurachki            | Paramuschir
 Krenizyn               | Krenizyn
 Lopatin                | Sachalin
 Zhima                  | Olkhon
 Asahi-Dake             | Hokkaido
 Iwate                  | Honshu
 Hotaka-Dake            | Honshu
 Kita-Dake              | Honshu
 Haku-San               | Honshu
 Fuji-San               | Honshu
 Daisen                 | Honshu
 Ishizuchi-San          | Shikoku
 Sobo-San               | Kyushu
```

Next we used WHERE to find only the mountains on the island of Honshu. Note, just because we see six islands in this list of rows, that doesn't mean another mountain on Honshu isn't hiding somewhere else because the list is not ordered.

```
mondial2=# SELECT * FROM mountainonisland WHERE island = 'Honshu';
  mountain   | island
-------------+--------
 Iwate       | Honshu
 Hotaka-Dake | Honshu
 Kita-Dake   | Honshu
 Haku-San    | Honshu
 Fuji-San    | Honshu
 Daisen      | Honshu
(6 rows)
```

Then we added ORDER BY to get our mountains in alphabetical order (that's the kind of order computers like to do with text).

```
mondial2=# SELECT * FROM mountainonisland WHERE island = 'Honshu' ORDER BY mountain;
  mountain   | island
-------------+--------
 Daisen      | Honshu
 Fuji-San    | Honshu
 Haku-San    | Honshu
 Hotaka-Dake | Honshu
 Iwate       | Honshu
 Kita-Dake   | Honshu
(6 rows)
```

Finally we added LIMIT to our query, to show only the first two lines of the resulting table. Not very useful here, but it does keep things very simple.

```
mondial2=# SELECT * FROM mountainonisland WHERE island = 'Honshu' ORDER BY mountain LIMIT 2;
 mountain | island
----------+--------
 Daisen   | Honshu
 Fuji-San | Honshu
(2 rows)
```

Then we jumped into a much more complex and messy (especially visually) table: city.

```
mondial2=# 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
 Patra                               | GR      | Dytikis Elladas                       |     213984 |    38.25
 Kozani                              | GR      | Dytikis Makedonias                    |            |  :
```

Here we learned the limits of using SELECT * in the command line. If there are a lot of columns it quickly becomes unreadable. So we took a look at the columns using \d , so we could decide what columns we wanted to select.

```
mondial2=# \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 our columns, and also filtered (WHERE) so that we would get populations above 1 million. Getting a much more readable table.

```
mondial2=# SELECT name, country, province, population FROM city WHERE population > 1000000;
          name           | country |       province        | population
-------------------------+---------+-----------------------+------------
 Beograd                 | SRB     | Serbia                |    1639121
 Paris                   | F       | Ãžle-de-France        |    2249975
 Barcelona               | E       | Catalunya             |    1611013
 Madrid                  | E       | Madrid                |    3198645
 Wien                    | A       | Wien                  |    1761738
 Praha                   | CZ      | Praha                 |    1289556
 MÃ¼nchen                | D       | Bayern                |    1348335
 Berlin                  | D       | Berlin                |    3292365
 Hamburg                 | D       | Hamburg               |    1706696
 KÃ¶ln                   | D       | Nordrhein-Westfalen   |    1005775
 Budapest                | H       | Budapest              |    1729040
 Milano                  | I       | Lombardia             |    1242123
 Roma                    | I       | Lazio                 |    2617175
 Minsk                   | BY      | Minsk City            |    1836808
 Warszawa                | PL      | Mazowieckie           |    1711324
 Kharkiv                 | UA      | Kharkivska            |    1441362
 Odesa                   | UA      | Odeska                |    1008162
 KyÃ¯v                   | UA      | KyÃ¯v                |    2814258
 Sankt Peterburg         | R       | Sankt-Peterburg       |    5028000
 Moskva                  | R       | Moscow                |   11979529
 Nizhnii Novgorod        | R       | Nizhnii Novgorodskaya |    1259921
 Voronezh                | R       | Voronezhskaya         |    1003638
 Kazan                   | R       | Tatarstan             |    1176187
 Volgograd               | R       | Volgogradskaya        |    1018790
 Samara                  | R       | Samarskaya            |    1171598
 Rostov-na-Donu          | R       | Rostovskaya           |    1103733
 Ufa                     | R       | Bashkortostan         |    1077719
 Perm                    | R       | Permskij              |    1013887
 Yekaterinburg           | R       | Sverdlovskaya         |    1396074
```

Next we explored how ORDER BY works. Ordering by the numerical column 'population'. Note that it defaults to ascending ASC order.

```
mondial2=# SELECT name, country, province, population FROM city WHERE population > 1000000
mondial2-# ORDER BY population;
          name           | country |       province        | population
-------------------------+---------+-----------------------+------------
 Quetta                  | PK      | Balochistan           |    1001205
 Kota                    | IND     | Rajasthan             |    1001694
 Kathmandu               | NEP     | Nepal                 |    1003285
 Voronezh                | R       | Voronezhskaya         |    1003638
 Kayseri                 | TR      | Kayseri               |    1004276
 KÃ¶ln                   | D       | Nordrhein-Westfalen   |    1005775
 Odesa                   | UA      | Odeska                |    1008162
 Islamabad               | PK      | FCT Islamabad         |    1009832
 Raipur                  | IND     | Chhattisgarh          |    1010433
 Monrovia                | LB      | Liberia               |    1010970
 Khartoum North          | SUD     | Al-KharÃ¹Ã«m          |    1012211
 Perm                    | R       | Permskij              |    1013887
 Krasnoyarsk             | R       | Krasnoyarsk           |    1016385
 Shubra al Khimah        | ET      | Egypt                 |    1016722
 Madurai                 | IND     | Tamil Nadu            |    1017865
 Volgograd               | R       | Volgogradskaya        |    1018790
 Sendai                  | J       | Miyagi                |    1020241
 Niamey                  | RN      | Niamey                |    1026848
 Managua                 | NIC     | Nicaragua             |    1028808
 Jodhpur                 | IND     | Rajasthan             |    1033756
 Vijayawada              | IND     | Andhra Pradesh        |    1034358
 Kisangani               | ZRE     | Tshopo                |    1040000
 Aba                     | WAN     | Abia                  |    1040000
 Coimbatore              | IND     | Tamil Nadu            |    1050721
 Changwon                | ROK     | South Korea           |    1053551
 Gwalior                 | IND     | Madhya Pradesh        |    1054420
 Jabalpur                | IND     | Madhya Pradesh        |    1055525
 Campinas                | BR      | SÃ£o Paulo           |    1061540
 Maiduguri               | WAN     | Borno                 |    1065000
```

So we used DESC to show the highest populations first.

```
mondial2=# SELECT name, country, province, population FROM city WHERE population > 1000000 ORDER BY population DES(
          name           | country |       province        | population
-------------------------+---------+-----------------------+------------
 Shanghai                | CN      | Shanghai              |   22315474
 Karachi                 | PK      | Sindh                 |   14916456
 Lagos                   | WAN     | Lagos                 |   13745000
 Istanbul                | TR      | Ä°stanbul             |   13710512
 Mumbai                  | IND     | Maharashtra           |   12442373
 Moskva                  | R       | Moscow                |   11979529
 Beijing                 | CN      | Beijing               |   11716620
 Kinshasa                | ZRE     | Kinshasa              |   11575000
 SÃ£o Paulo              | BR      | SÃ£o Paulo            |   11152344
 Lahore                  | PK      | Punjab                |   11126285
 Tianjin                 | CN      | Tianjin               |   11090314
 Guangzhou               | CN      | Guangdong             |   11071424
 Delhi                   | IND     | Delhi                 |   11034555
 Shenzhen                | CN      | Guangdong             |   10358381
 Seoul                   | ROK     | South Korea           |    9805506
 Wuhan                   | CN      | Hubei                 |    9785388
 Jakarta                 | RI      | DKI Jakarta           |    9607787
 Tehran                  | IR      | Tehran                |    8693706
 Tokyo                   | J       | Tokyo                 |    8591695
 Ciudad de MÃ©xico       | MEX     | Distrito Federal      |    8555272
 Al Qahirah              | ET      | Egypt                 |    8471859
 Bangalore               | IND     | Karnataka             |    8443675
 New York                | USA     | New York              |    8405837
 Bangkok                 | THA     | Thailand              |    8305218
 London                  | GB      | London                |    8250205
 Dongguan                | CN      | Guangdong             |    8220207
 BogotÃ¡                 | CO      | Santa Fe de BogotÃ¡    |    7776845
 Lima                    | PE      | Lima City             |    7605742
 Chongqing               | CN      | Chongqing             |    7457600
```

But look at what happens when we take out that's very convenient WHERE > 1000000 filter from the previous query.

```
mondial2=# SELECT name, country, province, population FROM city ORDER BY population DESC;
               name                | country |                province                | population
-----------------------------------+---------+-----------------------------------------+------------
 Ocotepeque                        | HCA     | Ocotepeque                              |
```

```
 Jamestown                          | HELX    | Saint Helena                          |
 Mamoutzou                          | MAYO    | Mayotte                               |
 Port Louis                         | MS      | Mauritius                             |
 SorÃ¸                               | DK      | SjÃ¦lland                              |
 Otsu                               | J       | Shiga                                 |
 Nara                               | J       | Nara                                  |
 Labuan                             | MAL     | Labuan                                |
 Anau                               | TM      | Akhal                                 |
 Guelmim                            | MA      | Guelmim Es Semara                     |
 Zouerate                           | RIM     | Tiris Zemmour                         |
 Tidjikja                           | RIM     | Tagant                                |
 Wakayama                           | J       | Wakayama                              |
 Akjoujt                            | RIM     | Inchiri                               |
 Aioun                              | RIM     | Hodh El Gharbi                        |
 Saint Peter Port                   | GBG     | Guernsey                              |
 Nema                               | RIM     | Hodh Chargui                          |
 Gibraltar                          | GBZ     | Gibraltar                             |
 Selibaby                           | RIM     | Guidimagha                            |
 Aleg                               | RIM     | Brakna                                |
```

A big difference! This is because in the previous query WHERE was not just filtering out low values, but also null values (empty fields in the columns). In psql if you order by descending, it defaults to putting the nulls first. So conveniently there is an additional command NULLS LAST .

```
mondial2=# SELECT name, country, province, population FROM city ORDER BY population DESC NULLS LAST;
                name                 | country |                province                | population
-------------------------------------+---------+-----------------------------------------+------------
 Shanghai                            | CN      | Shanghai                                |   22315474
 Karachi                             | PK      | Sindh                                   |   14916456
 Lagos                               | WAN     | Lagos                                   |   13745000
 Istanbul                            | TR      | Ä°stanbul                               |   13710512
 Mumbai                              | IND     | Maharashtra                             |   12442373
 Moskva                              | R       | Moscow                                  |   11979529
 Beijing                             | CN      | Beijing                                 |   11716620
 Kinshasa                            | ZRE     | Kinshasa                                |   11575000
 SÃ£o Paulo                          | BR      | SÃ£o Paulo                              |   11152344
 Lahore                              | PK      | Punjab                                  |   11126285
 Tianjin                             | CN      | Tianjin                                 |   11090314
 Guangzhou                           | CN      | Guangdong                               |   11071424
 Delhi                               | IND     | Delhi                                   |   11034555
 Shenzhen                            | CN      | Guangdong                               |   10358381
 Seoul                               | ROK     | South Korea                             |    9805506
 Wuhan                               | CN      | Hubei                                   |    9785388
```

Next we used LIMIT again to just show the top 10.

```
mondial2=# SELECT name, country, province, population FROM city ORDER BY population DESC NULLS LAST
mondial2-# LIMIT 10;
   name    | country |  province   | population
-----------+---------+-------------+------------
 Shanghai  | CN      | Shanghai    |   22315474
 Karachi   | PK      | Sindh       |   14916456
 Lagos     | WAN     | Lagos       |   13745000
 Istanbul  | TR      | Ä°stanbul    |   13710512
 Mumbai    | IND     | Maharashtra |   12442373
 Moskva    | R       | Moscow      |   11979529
 Beijing   | CN      | Beijing     |   11716620
 Kinshasa  | ZRE     | Kinshasa    |   11575000
 SÃ£o Paulo | BR      | SÃ£o Paulo   |   11152344
 Lahore    | PK      | Punjab      |   11126285
(10 rows)
```

We added WHERE to just get cities in China.

```
mondial2=# SELECT name, country, province, population FROM city WHERE country = 'CN' ORDER BY population DESC NULLS
   name    | country | province  | population
-----------+---------+-----------+------------
 Shanghai  | CN      | Shanghai  |   22315474
 Beijing   | CN      | Beijing   |   11716620
 Tianjin   | CN      | Tianjin   |   11090314
 Guangzhou | CN      | Guangdong |   11071424
 Shenzhen  | CN      | Guangdong |   10358381
 Wuhan     | CN      | Hubei     |    9785388
 Dongguan  | CN      | Guangdong |    8220207
 Chongqing | CN      | Chongqing |    7457600
```

```
 Chengdu   | CN      | Sichuan   |    7415590
 Foshan    | CN      | Guangdong |    7194311
(10 rows)
```

Next we add AND to WHERE have two parameters to filter our search for cities in China. We change the order to the default (ASC) so our resulting table begins with the least populous city that has more than 1 million people.

```
mondial2=# SELECT name, country, province, population FROM city
mondial2-# WHERE country = 'CN' AND population > 1000000 ORDER BY population;
     name     | country |   province   | population
--------------+---------+--------------+------------
 Datong       | CN      | Shanxi       |    1110000
 Handan       | CN      | Hebei        |    1110000
 Luoyang      | CN      | Henan        |    1190000
 Huainan      | CN      | Anhui        |    1200000
 Jilin        | CN      | Jilin        |    1270000
 Fushun       | CN      | Liaoning     |    1350000
 Qiqihar      | CN      | Heilongjiang |    1380000
 Anshan       | CN      | Liaoning     |    1390000
 Weifang      | CN      | Shandong     |    2044028
 Baotou       | CN      | Nei Mongol   |    2096851
 Zaozhuang    | CN      | Shandong     |    2125481
 Xiangyang    | CN      | Hubei        |    2199690
 Yantai       | CN      | Shandong     |    2227733
 Nantong      | CN      | Jiangsu      |    2274113
 Linyi        | CN      | Shandong     |    2303648
 Huizhou      | CN      | Guangdong    |    2344634
 Nanchang     | CN      | Jiangxi      |    2357839
 Lanzhou      | CN      | Gansu        |    2628426
 Huaian       | CN      | Jiangsu      |    2635406
 Shijiazhuang | CN      | Hebei        |    2834942
 Fuzhou       | CN      | Fujian       |    2921762
 Wulumuqi     | CN      | Xinjiang     |    3029372
```

I am skipping the very confusing example where I use OR instead of AND because it gives us very complicated results. The thing to understand with all conditional statements in all programming languages: with AND, the test has to be true for both cases, with OR the test only needs to be true for one of the cases, so you will get more results.

Here I introduced a few more of the possible conditional clauses starting with BETWEEN

```
mondial2=# SELECT name, country, province, population FROM city
mondial2-# WHERE  population BETWEEN 100000 AND 1000000 ORDER BY population;
                name                  | country |                  province                  | population
--------------------------------------+---------+--------------------------------------------+------------
 Fort-de-France                       | MART    | Martinique                                 |     100000
 Andria                               | I       | Puglia                                     |     100052
 Barakaldo                            | E       | Euskadi                                    |     100064
 Noginsk                              | R       | Moskovskaya                                |     100072
 Francistown                          | RB      | Botswana                                   |     100079
 Kaspijsk                             | R       | Dagestan                                   |     100129
 Worcester                            | GB      | West Midlands                              |     100153
 Lincoln                              | GB      | East Midlands                              |     100160
 Piacenza                             | I       | Emilia-Romagna                             |     100311
 Haining                              | CN      | Zhejiang                                   |     100478
 Masjed Soleyman                      | IR      | Khuzestan                                  |     100497
 Ancona                               | I       | Marche                                     |     100497
 SzÃ©kesfehÃ©rvÃ¡r                        | H       | FejÃ©r                                      |     100570
 Giresun                              | TR      | Giresun                                    |     100712
 Huaraz                               | PE      | Ancash                                     |     100931
```

IN() searches a list of values for specific matches. The search below is the same thing as "WHERE population=100000 OR population = 1000000" it is just much more efficient, especially if you wanted to put in a much longer list of values.

```
mondial2=# SELECT name, country, province, population FROM city
mondial2-# WHERE  population IN (100000, 1000000) ORDER BY population;
      name      | country |  province  | population
----------------+---------+------------+------------
 Fort-de-France | MART    | Martinique |     100000
(1 row)
```

Finally we used LIKE() which uses regular expressions to search for word patterns to find country codes that begin with the letter F.

```
mondial2=# SELECT name, country, province, population FROM city
mondial2-# WHERE  country LIKE ('F%') ORDER BY population;
         name         | country |          province          | population
----------------------+---------+----------------------------+------------
 Vaduz                | FL      | Liechtenstein              |       5207
 Kolonia              | FSM     | Micronesia                 |       5681
 Palikir              | FSM     | Micronesia                 |       6444
 TÃ³rshavn             | FARX    | Faroe Islands              |      13130
 Cayenne              | FGU     | French Guiana              |      57229
 Ajaccio              | F       | Corse                      |      66245
 Suva                 | FJI     | Fiji                       |      75225
 Calais               | F       | Hauts-de-France            |      75293
 Montreuil            | F       | ÃŽle-de-France             |     103068
 Argenteuil           | F       | ÃŽle-de-France             |     104282
 Nancy                | F       | Grand Est                  |     105382
 Caen                 | F       | Normandie                  |     108793
```

Then we got simple to introduce the concept of **aggregate queries**. Here we are just looking at the number of cities in each country by ordering our results by country.

```
mondial2=# SELECT country, name FROM city ORDER BY country;
 country |                name
---------+-------------------------------------
 A       | Salzburg
 A       | Graz
 A       | Lienz
 A       | Innsbruck
 A       | Wels
 A       | Linz
 A       | Wien
 A       | Dornbirn
 A       | Bregenz
 A       | Villach
 A       | Klagenfurt
 A       | Eisenstadt
 A       | St. PÃ¶lten
 AFG     | Herat
 AFG     | Kandahar
 AFG     | Mazar-i-Sharif
 AFG     | Kabul
 AG      | St. John's
 AL      | KorÃ§Ã«
 AL      | Tirana
 AL      | ShkodÃ«r
 AL      | DurrÃ«s
 AL      | VlorÃ«
 AL      | Elbasan
 AMSA    | Pago Pago
 AND     | Andorra la Vella
 ANG     | Cabinda
 ANG     | Ngiva
```

Then we used GROUP BY and COUNT() to make an aggregate query that counts up each unique country in the 'city' table. Aggregate queries, by definition, use the GROUP BY parameter. You can't aggregate without deciding what you are aggregating on (what you are grouping by). We are aggregating countries. In the table above A appeared 13 times in 13 rows. In this query, because it is an aggregate query, there is one row per country along with a number of times that country appears in the 'city' table. COUNT() is one method of aggregating: it counts up unique values.

```
mondial2=# SELECT country, COUNT(country) FROM city GROUP BY country ORDER BY country;
 country | count
---------+-------
 A       |    13
 AFG     |     4
 AG      |     1
 AL      |     6
 AMSA    |     1
 AND     |     1
 ANG     |    18
 ARM     |     1
 ARU     |     1
 AUS     |    15
 AXA     |     1
 AZ      |     3
 B       |    16
```

```
 BD      |    13
 BDS     |     1
 BEN     |     4
 BERM    |     1
 BF      |     2
 BG      |     6
 BHT     |     1
 BI      |     2
 BIH     |     7
 BOL     |    13
 BR      |   210
 BRN     |     1
 BRU     |     1
 BS      |     1
 BVIR    |     1
```

Noting that 'BR' (Brazil) had 210 cities I did the following query to show you how COUNT() works to aggregate.

```
mondial2=# SELECT country FROM city WHERE country = 'BR';
 country
---------
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 BR
 ...
BR
(210 rows)
```

If we scroll to the end of this list we would see that there are 210 rows. So technically in this count, we are really just getting the number of rows in which 'BR' appears in the 'country' column.

Next, we ordered our aggregate count() so we could see which countries appear the most.

```
mondial2=# SELECT country, COUNT(country) FROM city GROUP BY country ORDER BY count(country) DESC;
 country | count
---------+-------
 CN      |   302
 USA     |   251
 BR      |   210
 R       |   171
 IND     |    99
 TR      |    88
 D       |    85
 GB      |    84
 MEX     |    83
 J       |    72
 E       |    65
 WAN     |    58
 RI      |    58
 I       |    56
 CO      |    52
 IR      |    50
 RO      |    42
 PL      |    41
 F       |    41
 RA      |    39
 UA      |    38
 CDN     |    37
 RP      |    36
 PK      |    35
 YV      |    34
 ZRE     |    33
 PE      |    31
 S       |    30
```

This prompted a discussion of the order of processes in psql queries. ORDER BY sorts the final table resulting from the query, after the grouping in aggregation is done.

But!!!!! And this is where things can get confusing, we added WHERE to our aggregate query. And things got very different. Why? Because WHERE is not part of the aggregation. It filters the original table before the grouping and aggregation happens. So when we take out cities that have populations below 1000000, those rows disappear from the table that will be grouped and aggregated.

```
mondial2=# SELECT country, COUNT(country) FROM city WHERE population > 1000000
mondial2-# GROUP BY country ORDER BY count(country) DESC;
 country | count
---------+-------
 CN      |    60
 IND     |    44
 R       |    15
 BR      |    14
 J       |    12
 RI      |    11
 PK      |    10
 TR      |    10
 WAN     |    10
 USA     |     9
 MEX     |     9
 ROK     |     9
 IR      |     8
 RSA     |     6
 RC      |     5
 ZRE     |     5
 AUS     |     5
 RP      |     4
 CO      |     4
 ET      |     4
 D       |     4
 UA      |     3
 CDN     |     3
 MYA     |     3
 RA      |     3
 IRQ     |     3
 YV      |     3
 SA      |     3(6 rows)
```

Importantly, I then introduced HAVING which is a conditional parameter--it tests and filters values. It uses the same conditional operations (AND, OR, IN(), etc) as WHERE. But instead of WHERE, which tests and filters values from the original table, HAVING filters the resulting table after it has been aggregated.

```
mondial2=# SELECT country, COUNT(country) FROM city WHERE population > 1000000
mondial2-# GROUP BY country HAVING count(country) > 9 ORDER BY count(country) DESC;
 country | count
---------+-------
 CN      |    60
 IND     |    44
 R       |    15
 BR      |    14
 J       |    12
 RI      |    11
 TR      |    10
 PK      |    10
 WAN     |    10
(9 rows)
```

You cannot have HAVING without GROUP BY . HAVING tests/filters that group.

There was a little digression on how ORDER BY works. Note the two queries. It shifts the final table. You can pass multiple columns, and it will sort starting with the first column and then the next.

```
mondial2=# SELECT country, COUNT(country) FROM city WHERE population > 1000000                          GROUP BY
 country | count
---------+-------
 WAN     |    10
 TR      |    10
 RI      |    11
 R       |    15
 PK      |    10
 J       |    12
 IND     |    44
 CN      |    60
 BR      |    14
(9 rows)

mondial2=# SELECT country, COUNT(country) FROM city WHERE population > 1000000                          GROUP BY
 country | count
---------+-------
 CN      |    60
 IND     |    44
 R       |    15
 BR      |    14
 J       |    12
 RI      |    11
 PK      |    10
 TR      |    10
 WAN     |    10
(9 rows)
```

Here I did another aggregate query using SUM() on the population column -- because it is a numerical column, SUM() works. It is adding up all of the populations by country. (You could try using AVG() to get the average city populations.)

Note, in the first query we get a lot of empty values because the sums were null.

```
mondial2=# SELECT country, SUM(population) FROM city
mondial2-# GROUP BY country ORDER BY SUM(population) DESC;
 country |    sum
---------+-----------
 GROX    |
 NAU     |
 GNB     |
 FPOL    |
 NORF    |
 COOK    |
 XMAS    |
 CAYM    |
 GBM     |
 MACX    |
 MS      |
 GBZ     |
 BVIR    |
```

```
 AMSA    |
 AXA     |
 MAYO    |
 SVAX    |
 ARU     |
 SP      |
 GBG     |
 GUAD    |
 GUAM    |
 HELX    |
 FALK    |
 GBJ     |
 TUCA    |
 NIUE    |
 BERM    |
mondial2=# SELECT country, SUM(population) FROM city
GROUP BY country ORDER BY SUM(population) DESC NULLS LAST;
 country |    sum
---------+-----------
 CN      | 326058186
 IND     | 129752758
 BR      |  93687185
 USA     |  81882315
 R       |  72000673
 PK      |  50920843
 WAN     |  49410900
 J       |  48590545
 MEX     |  47137170
 RI      |  46893102
 TR      |  46045206
 IR      |  34845642
 ROK     |  33570031
 CO      |  26579496
 D       |  25333235
 GB      |  25252422
 RSA     |  23470701
 ZRE     |  22867466
 ET      |  22364857
 RP      |  19594294
 E       |  18553641
 UA      |  17198533
 AUS     |  16915048
 RC      |  16481172
 CDN     |  15670355
 RA      |  15351698
 BD      |  14607665
 VN      |  14446981
```

Another small digression/clarification on SELECT led me to this query, which just shows the sums and does not show the related countries because be left out the country column.

```
mondial2=# SELECT SUM(population) FROM city
mondial2-# GROUP BY country ORDER BY SUM(population) DESC NULLS LAST;
    sum
-----------
 326058186
 129752758
  93687185
  81882315
  72000673
  50920843
  49410900
  48590545
  47137170
  46893102
  46045206
  34845642
  33570031
  26579496
  25333235
  25252422
  23470701
  22867466
  22364857
  19594294
  18553641
  17198533
```

```
  16915048
  16481172
  15670355
  15351698
  14607665
  14446981
```

Finally we are on our way to using JOIN so that we can see the actual names of the countries. But first we want to look at the 'country' table, because that is the only table that contains the full names of the countries.

```
mondial2=# \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)
```

The key column, the main column we would use to join is country.code (in all the other tables that key is called 'country' (like city.country)). Once we are doing a JOIN it is important to use dot notation (table.column) to specify which table each column is coming from.

Here we take our previous aggregate query, but we use JOIN to get the full name of the country: country.name -- NOTE: that JOIN is inside the FROM parameter. Until now FROM has been very simple because we just want one table. But once you start joining you are making hybrid tables using multiple pre-existing tables. These tables are brought together using ON which matchs values from key columns that are shared across the different tables.

It might be helpful to imagine this entire statement: 'FROM city JOIN country ON city.country = country.code' to be the same thing as a single new customized table that you have invented.

```
mondial2=# SELECT country.name, SUM(city.population) FROM city
mondial2-# JOIN country ON city.country = country.code
mondial2-# GROUP BY country.name ORDER BY SUM(city.population) DESC NULLS LAST;
               name               |    sum
----------------------------------+-----------
 China                            | 326058186
 India                            | 129752758
 Brazil                           |  93687185
 United States                    |  81882315
 Russia                           |  72000673
 Pakistan                         |  50920843
 Nigeria                          |  49410900
 Japan                            |  48590545
 Mexico                           |  47137170
 Indonesia                        |  46893102
 Turkey                           |  46045206
 Iran                             |  34845642
 South Korea                      |  33570031
 Colombia                         |  26579496
 Germany                          |  25333235
 United Kingdom                   |  25252422
 South Africa                     |  23470701
 Zaire                            |  22867466
 Egypt                            |  22364857
 Philippines                      |  19594294
 Spain                            |  18553641
 Ukraine                          |  17198533
 Australia                        |  16915048
 Taiwan                           |  16481172
 Canada                           |  15670355
 Argentina                        |  15351698
 Bangladesh                       |  14607665
 Vietnam                          |  14446981
m
```

Below I breakdown how JOIN works to create a brand-new table. Below shows a single table 'city', then that table with the new column that was brought over from 'country'. This is what happens before the aggregation.

```
mondial2=# SELECT country, population FROM city ORDER BY country;
 country | population
---------+------------
 A       |     146676
 A       |     269211
 A       |      12046
 A       |     124386
 A       |      59239
 A       |     193511
 A       |    1761738
 A       |      45922
 A       |      27831
 A       |      59942
 A       |      96531
 A       |      13485
 A       |      52100
 AFG     |     335200
 AFG     |     311800
 AFG     |     288700
 AFG     |    2435400
 AG      |      22219
 AL      |      51152
 AL      |     418495
 AL      |      77075
 AL      |     113249
 AL      |      79513
 AL      |      78703
 AMSA    |
 AND     |      22256
 ANG     |
 ANG     |
mondial2=# SELECT city.country, city.population, country.name
mondial2-# FROM city JOIN country ON city.country = country.code
mondial2-# ORDER BY city.country;
 country | population |               name
---------+------------+----------------------------------
 A       |     146676 | Austria
 A       |     269211 | Austria
 A       |      12046 | Austria
 A       |     124386 | Austria
 A       |      59239 | Austria
 A       |     193511 | Austria
 A       |    1761738 | Austria
 A       |      45922 | Austria
 A       |      27831 | Austria
 A       |      59942 | Austria
 A       |      96531 | Austria
 A       |      13485 | Austria
 A       |      52100 | Austria
 AFG     |     335200 | Afghanistan
 AFG     |     311800 | Afghanistan
 AFG     |     288700 | Afghanistan
 AFG     |    2435400 | Afghanistan
 AG      |      22219 | Antigua and Barbuda
 AL      |      51152 | Albania
 AL      |     418495 | Albania
 AL      |      77075 | Albania
 AL      |     113249 | Albania
 AL      |      79513 | Albania
 AL      |      78703 | Albania
 AMSA    |            | American Samoa
 AND     |      22256 | Andorra
 ANG     |            | Angola
 ANG     |            | Angola
```

Here I just change the order of appearance of the columns the exact same table above.

```
mondial2=# SELECT city.country, country.name, city.population
mondial2-# FROM city JOIN country ON city.country = country.code
mondial2-# ORDER BY country;
 country |               name               | population
---------+----------------------------------+------------
 A       | Austria                          |     146676
```

```
 A       | Austria                        |     269211
 A       | Austria                        |      12046
 A       | Austria                        |     124386
 A       | Austria                        |      59239
 A       | Austria                        |     193511
 A       | Austria                        |    1761738
 A       | Austria                        |      45922
 A       | Austria                        |      27831
 A       | Austria                        |      59942
 A       | Austria                        |      96531
 A       | Austria                        |      13485
 A       | Austria                        |      52100
 AFG     | Afghanistan                    |     335200
 AFG     | Afghanistan                    |     311800
 AFG     | Afghanistan                    |     288700
 AFG     | Afghanistan                    |    2435400
 AG      | Antigua and Barbuda            |      22219
 AL      | Albania                        |      51152
 AL      | Albania                        |     418495
 AL      | Albania                        |      77075
 AL      | Albania                        |     113249
 AL      | Albania                        |      79513
 AL      | Albania                        |      78703
 AMSA    | American Samoa                 |
 AND     | Andorra                        |      22256
 ANG     | Angola                         |
 ANG     | Angola                         |
```

Same aggregate query as above, but this time I put the country names in alphabetical order--note this will give you a different order than alphabetizing by country codes.

```
mondial2=# SELECT country.name, SUM(city.population) FROM city
JOIN country ON city.country = country.code
GROUP BY country.name ORDER BY country.name;
               name               |    sum
----------------------------------+-----------
 Afghanistan                      |   3371100
 Albania                          |    818187
 Algeria                          |   7235824
 American Samoa                   |
 Andorra                          |     22256
 Angola                           |   7273439
 Anguilla                         |
 Antigua and Barbuda              |     22219
 Argentina                        |  15351698
 Armenia                          |   1066264
 Aruba                            |
 Australia                        |  16915048
 Austria                          |   2862618
 Azerbaijan                       |   2763900
 Bahamas                          |    248948
 Bahrain                          |    143035
 Bangladesh                       |  14607665
 Barbados                         |     88529
 Belarus                          |   3878063
 Belgium                          |   2256798
 Belize                           |     67186
 Benin                            |   1378292
 Bermuda                          |
 Bhutan                           |     42465
 Bolivia                          |   5139125
 Bosnia and Herzegovina           |   1047048
 Botswana                         |    327412
 Brazil                           |  93687185
```

Two more examples of simple JOINs without aggregating. Noting that with SELECT you're just specifying which columns to actually show.

```
mondial2=# SELECT city.country, country.name, city.population
mondial2-# FROM city JOIN country ON city.country = country.code
mondial2-# ORDER BY country;
 country |               name               | population
---------+----------------------------------+------------
 A       | Austria                          |     146676
 A       | Austria                          |     269211
 A       | Austria                          |      12046
```

```
 A       | Austria                          |     124386
 A       | Austria                          |      59239
 A       | Austria                          |     193511
 A       | Austria                          |    1761738
 A       | Austria                          |      45922
 A       | Austria                          |      27831
 A       | Austria                          |      59942
 A       | Austria                          |      96531
 A       | Austria                          |      13485
 A       | Austria                          |      52100
 AFG     | Afghanistan                      |     335200
 AFG     | Afghanistan                      |     311800
 AFG     | Afghanistan                      |     288700
 AFG     | Afghanistan                      |    2435400
 AG      | Antigua and Barbuda              |      22219
 AL      | Albania                          |      51152
 AL      | Albania                          |     418495
 AL      | Albania                          |      77075
 AL      | Albania                          |     113249
 AL      | Albania                          |      79513
 AL      | Albania                          |      78703
 AMSA    | American Samoa                   |
 AND     | Andorra                          |      22256
 ANG     | Angola                           |
 ANG     | Angola                           |
mondial2=# SELECT city.country, country.code, country.name, city.population
mondial2-# FROM city JOIN country ON city.country = country.code
mondial2-# ORDER BY country;
 country | code |               name               | population
---------+------+----------------------------------+------------
 A       | A    | Austria                          |     146676
 A       | A    | Austria                          |     269211
 A       | A    | Austria                          |      12046
 A       | A    | Austria                          |     124386
 A       | A    | Austria                          |      59239
 A       | A    | Austria                          |     193511
 A       | A    | Austria                          |    1761738
 A       | A    | Austria                          |      45922
 A       | A    | Austria                          |      27831
 A       | A    | Austria                          |      59942
 A       | A    | Austria                          |      96531
 A       | A    | Austria                          |      13485
 A       | A    | Austria                          |      52100
 AFG     | AFG  | Afghanistan                      |     335200
 AFG     | AFG  | Afghanistan                      |     311800
 AFG     | AFG  | Afghanistan                      |     288700
 AFG     | AFG  | Afghanistan                      |    2435400
 AG      | AG   | Antigua and Barbuda              |      22219
 AL      | AL   | Albania                          |      51152
 AL      | AL   | Albania                          |     418495
 AL      | AL   | Albania                          |      77075
 AL      | AL   | Albania                          |     113249
 AL      | AL   | Albania                          |      79513
 AL      | AL   | Albania                          |      78703
 AMSA    | AMSA | American Samoa                   |
 AND     | AND  | Andorra                          |      22256
 ANG     | ANG  | Angola                           |
 ANG     | ANG  | Angola                           |
```

Finally, I introduced AS which allows you to alias columns (and tables as well). You can give them more simple/clear names using AS, and you can use those aliases in specific parts of your query to make things more efficient. Note the column headers in the results are different.

```
mondial2=# SELECT city.country AS countryCode, country.name AS cName, city.population AS pop
mondial2-# FROM city JOIN country ON city.country = country.code
mondial2-# ORDER BY country;
 countrycode |              cname               |   pop
-------------+----------------------------------+----------
 A           | Austria                          |   146676
 A           | Austria                          |   269211
 A           | Austria                          |    12046
 A           | Austria                          |   124386
 A           | Austria                          |    59239
 A           | Austria                          |   193511
 A           | Austria                          |  1761738
 A           | Austria                          |    45922
 A           | Austria                          |    27831
```

```
A           | Austria                        |    59942
A           | Austria                        |    96531
A           | Austria                        |    13485
A           | Austria                        |    52100
AFG         | Afghanistan                    |   335200
AFG         | Afghanistan                    |   311800
AFG         | Afghanistan                    |   288700
AFG         | Afghanistan                    |  2435400
AG          | Antigua and Barbuda            |    22219
AL          | Albania                        |    51152
AL          | Albania                        |   418495
AL          | Albania                        |    77075
AL          | Albania                        |   113249
AL          | Albania                        |    79513
AL          | Albania                        |    78703
AMSA        | American Samoa                 |
AND         | Andorra                        |    22256
ANG         | Angola                         |
ANG         | Angola                         |
```