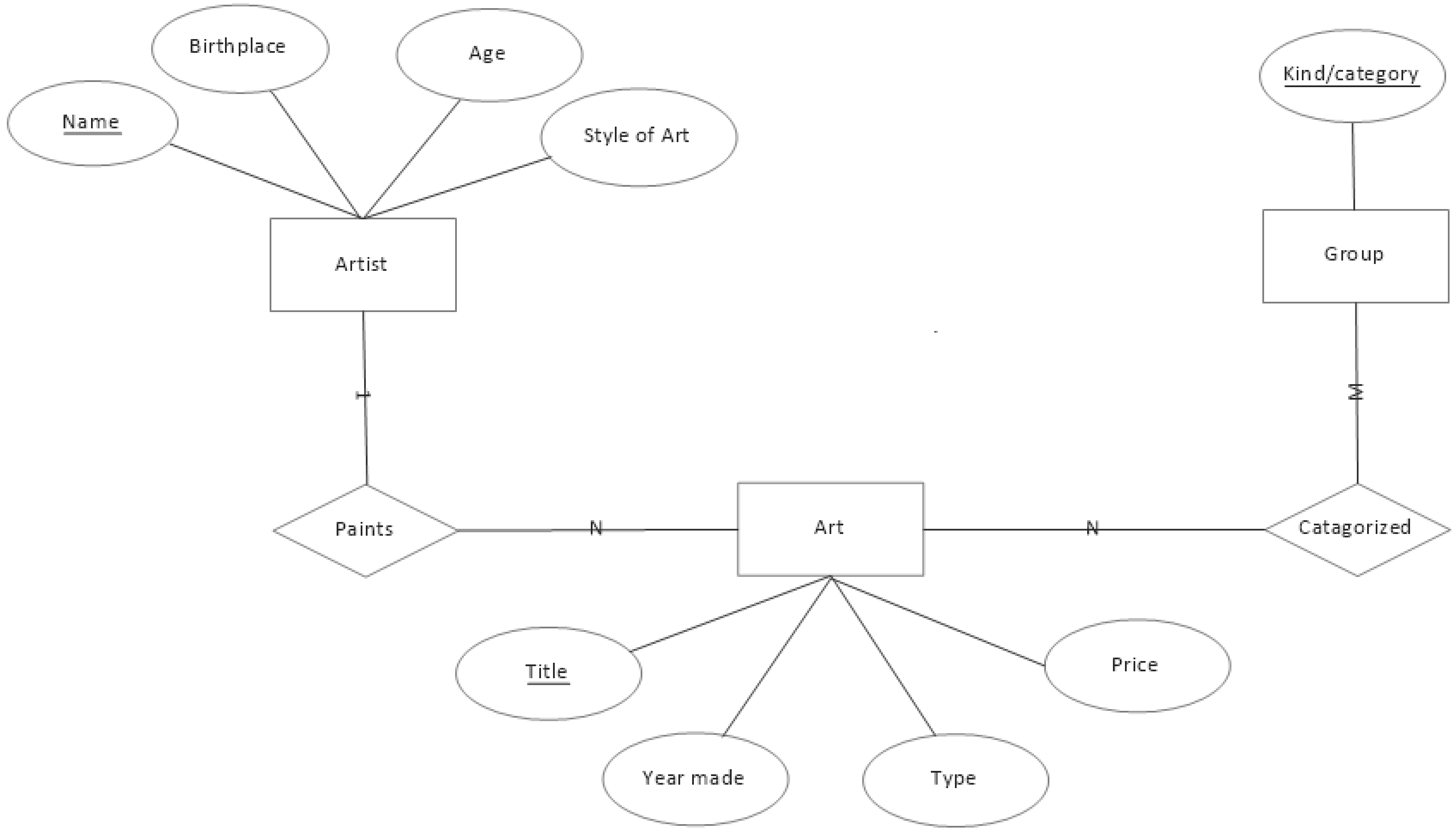
Introduction to Databases

Assignment 1

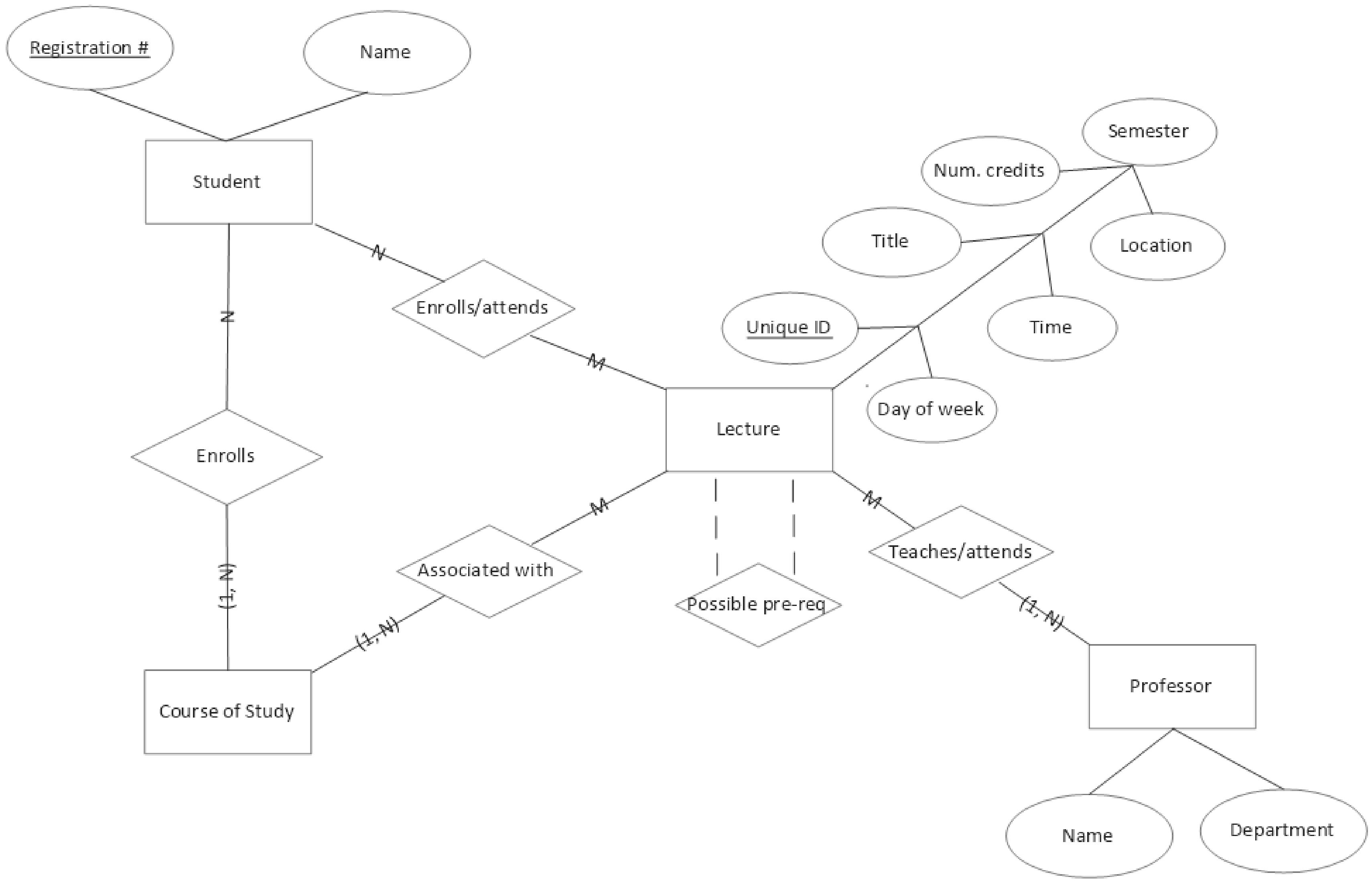
Nicholas Hopewell

Part 1 – ER Diagrams:

1.



2.



Registration #
Name
Student
N
Enrolls/attends
N
M
Enrolls
(1, N)
Course of Study
Associated with
(1, N)
M
Lecture
Semester
Num. credits
Title
Location
Unique ID
Time
Day of week
Possible pre-req
M
Teaches/attends
(1, N)
Professor
Name
Department

Part 2 – Normalization:

<u>Database Scheme:</u>

EMPLOYEE(<u>eid</u>, first name, middle name, last name, date_of_birth, home_address, national_insurance_number, first_day_of_employment).

Candidate keys: Patient#, Prescription#, Doctor#

- Primary Key = <u>single underline</u>
- Foreign Key = <u>double underline</u>

*** In my opinion, this has two viable 3NF schemes, one of which I do not approve of. ***

- **1NF Scheme:**

EMPLOYEE-1(<u>eid</u>, first name, middle name, last name, date_of_birth, home_address, national_insurance_number, first_day_of_employment).

*** Nothing needs to be changed here. The only possible field with multiple values is 'middle_name' but it is not appropriate to split this field into different fields as multiple middle names are considered as one complete middle name. Splitting is not appropriate. ***

- **2NF Scheme:**

EMPLOYEE-2(<u>eid</u>, first name, middle name, last name, date_of_birth, home_address, national_insurance_number, first_day_of_employment).

- OR -

EMPLOYEE-2(<u>eid</u>, <u>national_insurance_number,</u> first name, middle name, last name, first_day_of_employment).

CITIZEN-2(<u>national_insurance_number</u>, date_of_birth, home_address)

*** Here is where I see two possible options. Firstly, one could argue that the original scheme is already in 2NF because all non-key fields can be argued to be dependant on the entire primary key. One could say, once you have the id, you have all the information together. There is one clear possible exception I see depending on the database design, and that is home_address (and possibly date of birth which would be quite irrelevant to an employee table). Why I say this is because a table which includes an employees address would likely be a table which also has billing and payment information of all kinds. This table does not seem like that type of table (although it could easily be related to such a table via employment id and so I am only speculating). Home_address may not be functionally dependant on employee_id, but if we also used national_insurance_number as a partial key, home_address would definitively be* *functionally dependant on that number (as would date_of_birth). I illustrate this in the second possible solution. I don't really like this solution because the individual's names should also be in this new table, thus, the citizen table on its own in this form seems very incomplete. It doesn't seem like a satisfactory solution over a single table. If there were more attributes such as job title, job code and so on, this database would to much more interesting. ***

- **3NF Scheme:**

EMPLOYEE-3(<u>eid</u>, first name, middle name, last name, date_of_birth, home_address, national_insurance_number, first_day_of_employment).

- OR -

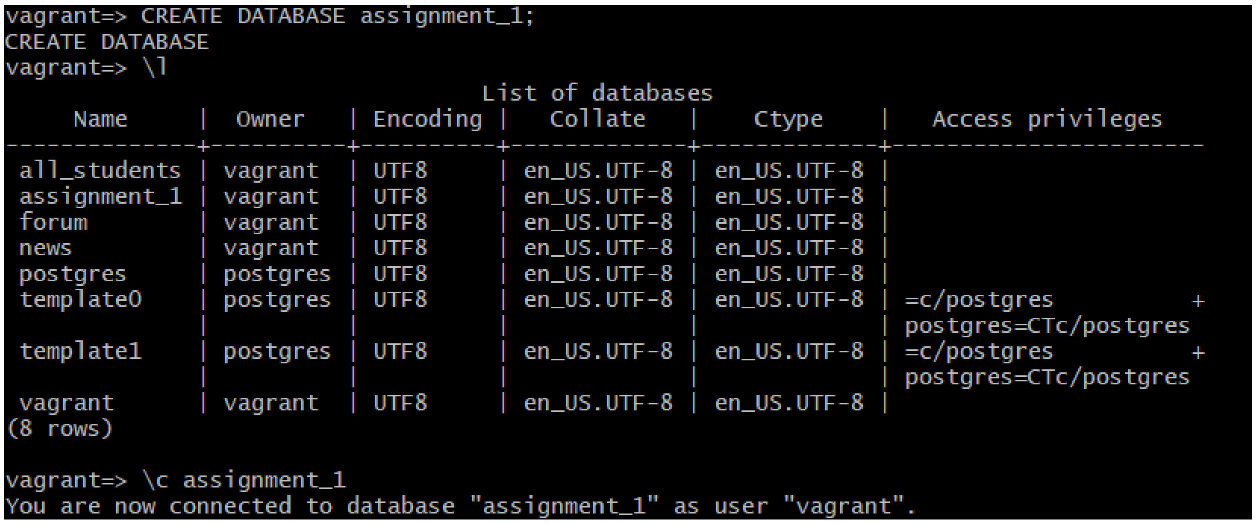
EMPLOYEE-3(<u>eid</u>, <u>national_insurance_number,</u> first name, middle name, last name, first_day_of_employment).

CITIZEN-3(<u>national_insurance_number</u>, date_of_birth, home_address)

*** There are no transitive dependencies between the attributes and the primary key. This is already in 3NF in my opinion. The reason why I have a second solution I do not really like comes down to the fact that I think this database is unfinished and not very helpful. If we had more information, it would be actually easier to organize in my opinion. ***

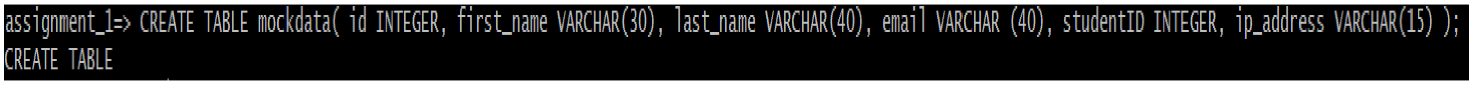
Part 3 – PSQL:

Create and connect to the db



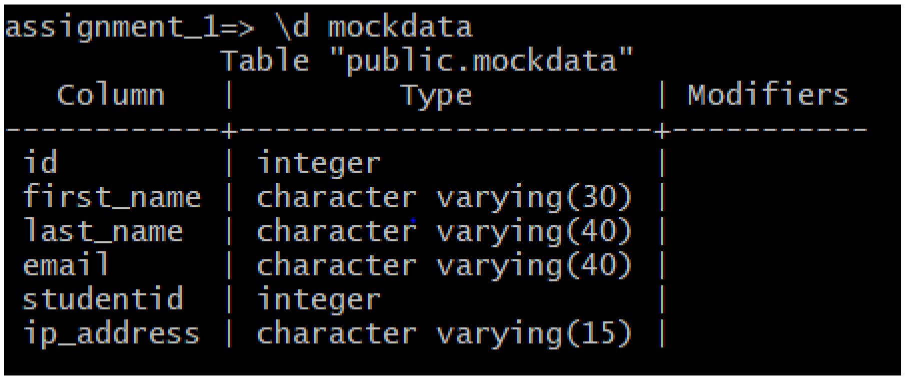
```
vagrant=> CREATE DATABASE assignment_1;
CREATE DATABASE
vagrant=> \l
                                  List of databases
     Name     |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
--------------+----------+----------+-------------+-------------+-----------------------
 all_students | vagrant  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 assignment_1 | vagrant  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 forum        | vagrant  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 news         | vagrant  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres     | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
              |          |          |             |             | postgres=CTc/postgres
 template1    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
              |          |          |             |             | postgres=CTc/postgres
 vagrant      | vagrant  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(8 rows)

vagrant=> \c assignment_1
You are now connected to database "assignment_1" as user "vagrant".
```

Create table:



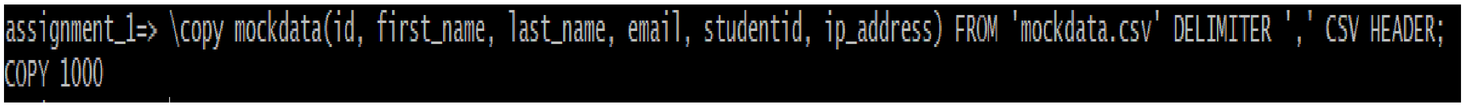
```
assignment_1=> CREATE TABLE mockdata( id INTEGER, first_name VARCHAR(30), last_name VARCHAR(40), email VARCHAR (40), studentID INTEGER, ip_address VARCHAR(15) );
CREATE TABLE
```

Describe table:



```
assignment_1=> \d mockdata
            Table "public.mockdata"
   Column    |         Type          | Modifiers
-------------+-----------------------+-----------
 id          | integer               |
 first_name  | character varying(30) |
 last_name   | character varying(40) |
 email       | character varying(40) |
 studentid   | integer               |
 ip_address  | character varying(15) |
```

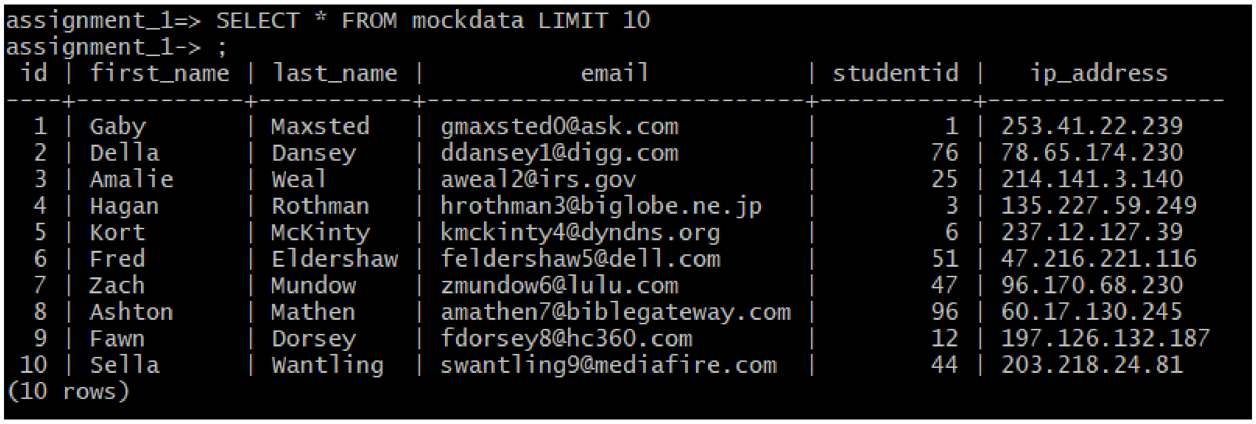
Copy data into table:



```
assignment_1=> \copy mockdata(id, first_name, last_name, email, studentid, ip_address) FROM 'mockdata.csv' DELIMITER ',' CSV HEADER;
COPY 1000
```

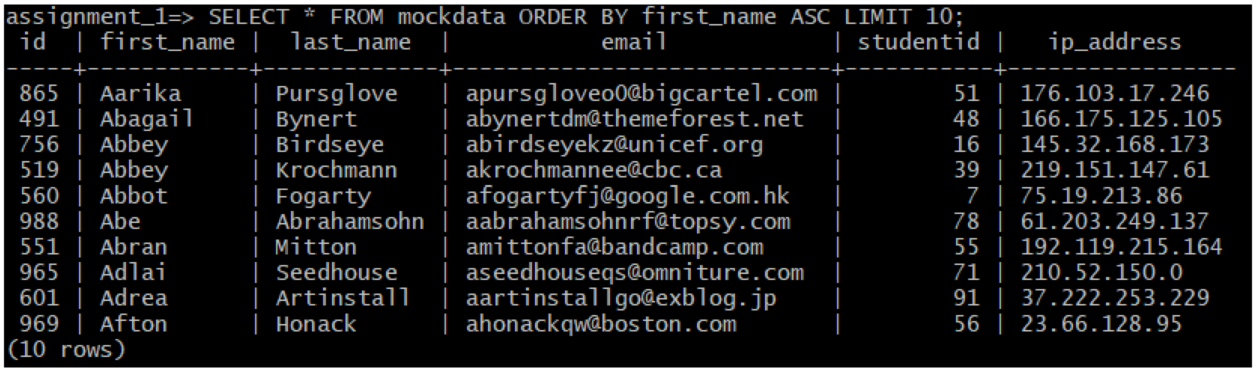
Copied 1000 tupples of data.

Show 10 values of the table:



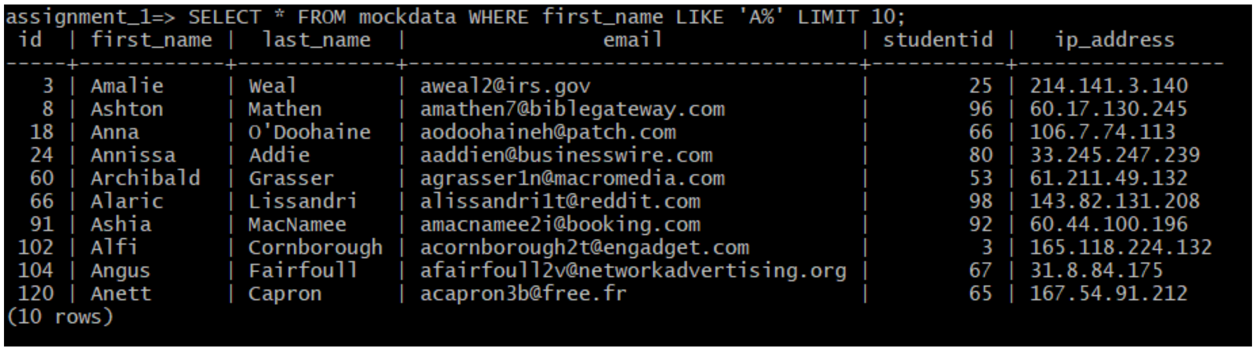
```
assignment_1=> SELECT * FROM mockdata LIMIT 10
assignment_1-> ;
 id | first_name | last_name |           email            | studentid |    ip_address
----+------------+-----------+----------------------------+-----------+-----------------
  1 | Gaby       | Maxsted   | gmaxsted0@ask.com          |         1 | 253.41.22.239
  2 | Della      | Dansey    | ddansey1@digg.com          |        76 | 78.65.174.230
  3 | Amalie     | Weal      | aweal2@irs.gov             |        25 | 214.141.3.140
  4 | Hagan      | Rothman   | hrothman3@biglobe.ne.jp    |         3 | 135.227.59.249
  5 | Kort       | McKinty   | kmckinty4@dyndns.org       |         6 | 237.12.127.39
  6 | Fred       | Eldershaw | feldershaw5@dell.com       |        51 | 47.216.221.116
  7 | Zach       | Mundow    | zmundow6@lulu.com          |        47 | 96.170.68.230
  8 | Ashton     | Mathen    | amathen7@biblegateway.com  |        96 | 60.17.130.245
  9 | Fawn       | Dorsey    | fdorsey8@hc360.com         |        12 | 197.126.132.187
 10 | Sella      | Wantling  | swantling9@mediafire.com   |        44 | 203.218.24.81
(10 rows)
```

1. Sort first name in ascending order: (using limit 10 for space for all queries)



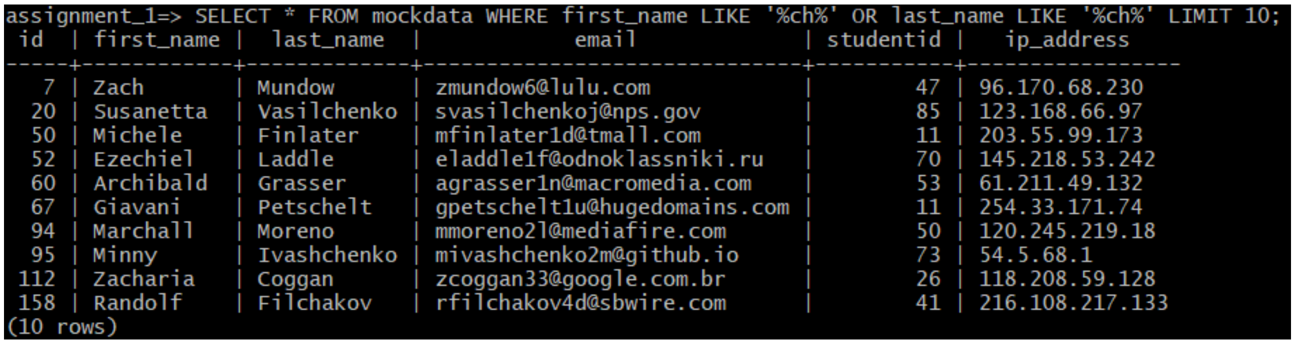
```
assignment_1=> SELECT * FROM mockdata ORDER BY first_name ASC LIMIT 10;
 id  | first_name |  last_name  |            email             | studentid |   ip_address
-----+------------+-------------+------------------------------+-----------+-----------------
 865 | Aarika     | Pursglove   | apursgloveo0@bigcartel.com   |        51 | 176.103.17.246
 491 | Abagail    | Bynert      | abynertdm@themeforest.net    |        48 | 166.175.125.105
 756 | Abbey      | Birdseye    | abirdseyekz@unicef.org       |        16 | 145.32.168.173
 519 | Abbey      | Krochmann   | akrochmannee@cbc.ca          |        39 | 219.151.147.61
 560 | Abbot      | Fogarty     | afogartyfj@google.com.hk     |         7 | 75.19.213.86
 988 | Abe        | Abrahamsohn | aabrahamsohnrf@topsy.com     |        78 | 61.203.249.137
 551 | Abran      | Mitton      | amittonfa@bandcamp.com       |        55 | 192.119.215.164
 965 | Adlai      | Seedhouse   | aseedhouseqs@omniture.com    |        71 | 210.52.150.0
 601 | Adrea      | Artinstall  | aartinstallgo@exblog.jp      |        91 | 37.222.253.229
 969 | Afton      | Honack      | ahonackqw@boston.com         |        56 | 23.66.128.95
(10 rows)
```

2. Select tupples where first name starts with ‘A”.



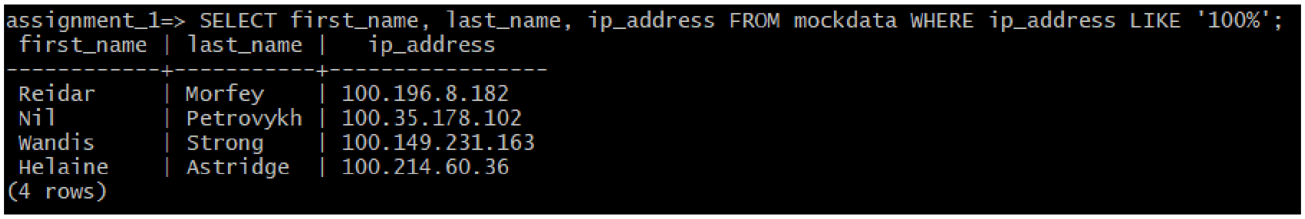
```
assignment_1=> SELECT * FROM mockdata WHERE first_name LIKE 'A%' LIMIT 10;
 id  | first_name |  last_name  |                 email                  | studentid |   ip_address
-----+------------+-------------+----------------------------------------+-----------+-----------------
   3 | Amalie     | Weal        | aweal2@irs.gov                         |        25 | 214.141.3.140
   8 | Ashton     | Mathen      | amathen7@biblegateway.com              |        96 | 60.17.130.245
  18 | Anna       | O'Doohaine  | aodoohaineh@patch.com                  |        66 | 106.7.74.113
  24 | Annissa    | Addie       | aaddien@businesswire.com               |        80 | 33.245.247.239
  60 | Archibald  | Grasser     | agrasser1n@macromedia.com              |        53 | 61.211.49.132
  66 | Alaric     | Lissandri   | alissandri1t@reddit.com                |        98 | 143.82.131.208
  91 | Ashia      | MacNamee    | amacnamee2i@booking.com                |        92 | 60.44.100.196
 102 | Alfi       | Cornborough | acornborough2t@engadget.com            |         3 | 165.118.224.132
 104 | Angus      | Fairfoull   | afairfoull2v@networkadvertising.org    |        67 | 31.8.84.175
 120 | Anett      | Capron      | acapron3b@free.fr                      |        65 | 167.54.91.212
(10 rows)
```

3. Select tupples where first or last name contains the letters ‘ch’.



```
assignment_1=> SELECT * FROM mockdata WHERE first_name LIKE '%ch%' OR last_name LIKE '%ch%' LIMIT 10;
 id  | first_name |  last_name  |            email              | studentid |   ip_address
-----+------------+-------------+-------------------------------+-----------+-----------------
   7 | Zach       | Mundow      | zmundow6@lulu.com             |        47 | 96.170.68.230
  20 | Susanetta  | Vasilchenko | svasilchenkoj@nps.gov         |        85 | 123.168.66.97
  50 | Michele    | Finlater    | mfinlater1d@tmall.com         |        11 | 203.55.99.173
  52 | Ezechiel   | Laddle      | eladdle1f@odnoklassniki.ru    |        70 | 145.218.53.242
  60 | Archibald  | Grasser     | agrasser1n@macromedia.com     |        53 | 61.211.49.132
  67 | Giavani    | Petschelt   | gpetschelt1u@hugedomains.com  |        11 | 254.33.171.74
  94 | Marchall   | Moreno      | mmoreno2l@mediafire.com       |        50 | 120.245.219.18
  95 | Minny      | Ivashchenko | mivashchenko2m@github.io      |        73 | 54.5.68.1
 112 | Zacharia   | Coggan      | zcoggan33@google.com.br       |        26 | 118.208.59.128
 158 | Randolf    | Filchakov   | rfilchakov4d@sbwire.com       |        41 | 216.108.217.133
(10 rows)
```

4. Select first, last, and ip address where ip address begins with ‘100’.



```
assignment_1=> SELECT first_name, last_name, ip_address FROM mockdata WHERE ip_address LIKE '100%';
 first_name | last_name |   ip_address
------------+-----------+-----------------
 Reidar     | Morfey    | 100.196.8.182
 Nil        | Petrovykh | 100.35.178.102
 Wandis     | Strong    | 100.149.231.163
 Helaine    | Astridge  | 100.214.60.36
(4 rows)
```