HW: Week 12

36-350 – Statistical Computing

Week 12 – Spring 2021

Name: Julian Adler

Andrew ID: jadler

You must submit **your own** lab as a knitted PDF file on Gradescope.

This week's homework is a little different. Here you will be working with `SQL`, specifically with the variant `postgres`. You will do your work "remotely" in a `postgres` terminal and cut-and-paste your answers into plain code blocks below:

```
This is a plain code block. Note the lack of a {r} above.
Try to avoid issues of text going off the page by utilizing
line breaks.
```

Cut-and-paste both your input command and the output. *If the output has many rows, it is sufficient to cut-and-paste the first five, unless otherwise instructed.*

Question 1

(10 points)

Notes 12A (7-11) + Notes 12B (3)

Create a table dubbed `rdata` that has five columns: `id` (type `serial primary key`), `a` and `b` (consisting of strings that should be no more than five characters), `moment` (which is a date), and `x` (which is a number that should have no more than five digits total and two to the right of the decimal point).

```
Input:
create table rdata (id serial primary key, a varchar(5), b varchar(5), moment date, x numeric(5,2) );

Output: CREATE TABLE
```

Question 2

(10 points)

Notes 12B (4,8)

Delete the table and create it again, with certain constraints: `a` and `b` cannot be null and must be unique; `moment` should have a default value of `2020-01-01` (); and `x` should be larger than zero.

```
Input: drop table rdata;

Output: DROP TABLE

Input: create table rdata (id serial primary key, a varchar(5) unique not null
,b varchar(5) unique not null, moment date default '2020-01-01', x numeric(5,2) check (x > 0));

Output: CREATE TABLE
```

Question 3

(10 points)

Notes 12A (4)

Use `\d` in combination with the name of your table to display its properties. Copy and paste all output here.

```
Input: \d rdata;

Output:                                    Table "public.rdata"
 Column |         Type         | Collation | Nullable |              Default
--------+----------------------+-----------+----------+-----------------------------------
 id     | integer              |           | not null | nextval('rdata_id_seq'::regclass)
 a      | character varying(5) |           | not null |
 b      | character varying(5) |           | not null |
 moment | date                 |           |          | '2020-01-01'::date
 x      | numeric(5,2)         |           |          |
Indexes:
    "rdata_pkey" PRIMARY KEY, btree (id)
    "rdata_a_key" UNIQUE CONSTRAINT, btree (a)
    "rdata_b_key" UNIQUE CONSTRAINT, btree (b)
Check constraints:
    "rdata_x_check" CHECK (x > 0::numeric)
```

Question 4

(10 points)

Notes 12B (5)

Insert three rows of data into your table. Do this without explicity referring to the column `id` or `moment`. Display your table using the command `select * from rdata`. You should see a default date in your `moment` column!

```
Input: insert into rdata (a,b, x) values
  ('fun', 'time', 5.22),
  ('eat', 'beef', 62.00),
  ('very', 'cool', 6.12);
Ouput: INSERT 0 3
Input: select * from rdata;
Output:
id |  a   |  b   |   moment   |   x
---+------+------+------------+-------
  1 | fun  | time | 2020-01-01 |  5.22
```

```
  2 | eat  | beef | 2020-01-01 | 62.00
  3 | very | cool | 2020-01-01 |  6.12
(3 rows)
```

Question 5

(10 points)

Notes 12B (5)

Attempt to add another row of data with a negative value for `x`. Show what happens. (If the row is added... that's bad. It means you don't have the constraint `x` > 0 defined.) Afterwords, add a valid row of data, and show the table. Is there anything weird?

```
Input: insert into rdata (a,b, x) values
  ('fun', 'time', -2.00);
Ouput: ERROR:  new row for relation "rdata" violates check constraint "rdata_x_check"
DETAIL:  Failing row contains (4, fun, time, 2020-01-01, -2.00).

Input: insert into rdata (a,b, x) values
  ('ew', 'new', 2.00);
Output: INSERT 0 1
Input: select* from rdata
Output:
 id |  a   |  b   |   moment   |   x
----+------+------+------------+-------
  1 | fun  | time | 2020-01-01 |  5.22
  2 | eat  | beef | 2020-01-01 | 62.00
  3 | very | cool | 2020-01-01 |  6.12
  5 | ew   | new  | 2020-01-01 |  2.00
(4 rows)

The weird thing about this table is that the new id should be four but is instead 5.
```

Question 6

(10 points)

Notes 12B (6)

Change the table `rdata` so as to change the data in `moment` so that two of the dates are in March 2020, while the other two are not in March 2020. Use `where` as shown in the notes, and utilize a logical "or" to update two of the dates to the same date in March at the same time. Show your updated table. (Note that the rows may be rearranged. This is OK.)

```
Input: update rdata
  set moment = '2020-03-01'
  where x = 2.00 or x = 5.22;
Output: UPDATE 2
Input: select* from rdata;
Output:
 id |  a   |  b   |   moment   |   x
----+------+------+------------+-------
  2 | eat  | beef | 2020-01-01 | 62.00
```

```
  3 | very | cool | 2020-01-01 |  6.12
  1 | fun  | time | 2020-03-01 |  5.22
  5 | ew   | new  | 2020-03-01 |  2.00
(4 rows)
```

Question 7

(10 points)

Notes 12B (7)

Add a new column to `rdata` dubbed `y`, and let it be of `boolean` type with default value `false`. Display your updated table.

```
Input: alter table rdata
  add column y boolean default false;
Output: ALTER TABLE
Input: select* from rdata;
Output:
id |  a   |  b   |   moment   |   x   | y
----+------+------+------------+-------+---
  2 | eat  | beef | 2020-01-01 | 62.00 | f
  3 | very | cool | 2020-01-01 |  6.12 | f
  1 | fun  | time | 2020-03-01 |  5.22 | f
  5 | ew   | new  | 2020-03-01 |  2.00 | f
(4 rows)
```

Question 8

(10 points)

Notes 12B (8)

Remove the row of your table with `id` value 2. Display your updated table.

```
Input: delete from rdata
        where id = 2;
Ouput: DELETE 1
Input: select* from rdata;
Output:
 id |  a   |  b   |   moment   |  x   | y
----+------+------+------------+------+---
  3 | very | cool | 2020-01-01 | 6.12 | f
  1 | fun  | time | 2020-03-01 | 5.22 | f
  5 | ew   | new  | 2020-03-01 | 2.00 | f
(3 rows)
```

Question 9

(10 points)

Notes 12B (7)

Rename the column `moment` to have the name `date`. Display your updated table.

```
Input: alter table rdata
        rename column moment to date;
Output: ALTER TABLE
Input: select* from rdata;
Output:
 id |  a   |  b   |    date    |  x   | y
----+------+------+------------+------+---
  3 | very | cool | 2020-01-01 | 6.12 | f
  1 | fun  | time | 2020-03-01 | 5.22 | f
  5 | ew   | new  | 2020-03-01 | 2.00 | f
(3 rows)
```

Question 10

(10 points)

Notes 12C (2-4)

Download the file `GalaxyStatistics.txt` from the `DATA` directory on `Canvas`. This file contains three columns: the sky field name, the Gini coefficient value, and the concentration statistic value for each of 8,358 observed galaxies. (Feel free to call the concentration statistic column `conc` for short.) Copy it into a `postgres` session to populate a table named `galaxies`. You should add explicit checks that ensure that `gini` and `conc` have values greater than zero. Hint: you'll have to explicitly provide a `delimiter` value here.

```
Input: create table galaxies (sky_field_name text, gini numeric check (gini > 0), conc numeric check (c
Ouput: CREATE TABLE
Input: \copy galaxies from 'GalaxyStatistics.txt' with (format csv,header, delimiter " ");
Output: COPY 8358
Input: select* from galaxies;
Output:
 sky_field_name |        gini        |       conc
----------------+--------------------+-------------------
 COSMOS         |  0.504693664799751 |  3.57616535107618
 COSMOS         |  0.433492285980024 |  3.10393208720358
 COSMOS         |  0.287995253794197 |  2.27855628892391
 COSMOS         |  0.517034044130523 |  2.81661082728353
 COSMOS         |  0.303455775671215 |  2.45671726779084
```

Question 11

(10 points)

Notes 12B (8)

Delete all rows of the table for which the value of `gini` is less than 0.5 or the value of `conc` is greater than 3. You should find that 7,883 rows are deleted. (Show this by showing the output of your `SQL` command.)

```
Input: delete from galaxies where gini < .5 or conc > 3;
Ouput: DELETE 7883
```

Question 12

(10 points)

Notes 12B (6-7)

Alter the table to create a new variable `cg_rat` that is the ratio of `conc` to `gini`. Then display all the rows where the value of `cg_rat` is greater than 5.9. (This last part is accomplished by combining `select` with `where` in a way that should be hopefully becoming somewhat intuitive.) You should have nine rows of output overall.

```
Input; alter table galaxies
          add column cg_rat numeric;
Ouput: ALTER TABLE
Input: update galaxies
          set cg_rat = conc / gini;
Output: UPDATE 475
Input: select* from galaxies where cg_rat > 5.9;
Output:
 sky_field_name |       gini        |       conc       |       cg_rat
----------------+-------------------+------------------+--------------------
 COSMOS         | 0.505315816622831 |  2.9826252928467 | 5.9024973981230812
 COSMOS         | 0.500590685140458 | 2.97597832910617 | 5.9449334904646828
 COSMOS         |  0.50361598800351 |   2.987334888289 | 5.9317713485064725
 EGS            |  0.50042450368941 | 2.98645170853111 | 5.9678366796855743
 COSMOS         | 0.500671122036005 | 2.97465381069856 | 5.9413329025288627
 COSMOS         | 0.503719930512369 | 2.97337141688731 | 5.9028266240386491
 GOODSS         | 0.500338113641828 | 2.98312251852409 | 5.9622132257939235
 UDS            | 0.503803338671752 | 2.98048530133007 | 5.9159697297519801
 UDS            | 0.500305684629919 | 2.97898274914929 | 5.9543252069040004
(9 rows)
```