**Due: Feb. 15**

**Econ 5253 - Spring 2022** **Problem Set 3** **beginning of class**

This problem set will provide an opportunity for you to practice working with the command line, writing a Linux shell script, and working with data in SQL.

As with the previous problem sets, you will submit this problem set by pushing the document to *your* (private) fork of the class repository. You will put this and all other problem sets in the path `/DScourseS22/ProblemSets/PS3/` and name the file `PS3_LastName.*`. Your directory should contain four files:

- `PS3_LastName.sh`
- `PS3_LastName.sql`
- `PS3_LastName.tex`
- `PS3_LastName.pdf`

1. Log in to OSCER, change to the directory where you cloned your forked GitHub repository (probably `˜/DScourseS22`), and make sure the OSCER version of your repository is synchronized with what is listed on GitHub by issuing a pull. That is, type `git pull origin master` from your OSCER `DScourseS22` folder.

2. Create the PS3 subdirectory by issuing `cd ˜/DScourseS22/ProblemSets` followed by `mkdir PS3` on OSCER. Change into this directory by typing `cd PS3`.

3. Next you will get some practice with some new Linux shell commands, as well as how to create and execute a Linux shell script. (Note: you can also do these commands on your Macbook, or at the `git` shell on a Windows machine.)

   (a) The `wget` command allows you to download a file from a URL right from the command line. Try it out by typing `wget http://spatialkeydocs.s3.amazonaws.com/FL_insurance_sample.csv.zip` at the command line. This will download the file contained at the preceding URL, which is a sample data set on Florida property insurance provided by a company called SpatialKey$^{TM}$.

   (b) Type `ls` to see the contents of the current directory. You should see a file called `FL_insurance_sample.csv.zip`, which is a compressed (ZIP) file.

   (c) The way to uncompress a file with a `.zip` extension is with `unzip`. Type `unzip FL_insurance_sample.csv.zip`

   (d) You will now notice that the system created a folder called `__MACOSX/` in your current. This is because the company that provided the data used Mac OS to zip the file. Let's delete that, along with the .zip file, by issuing the `rm` command: `rm -rf __MACOSX` followed by `rm -f FL_insurance_sample.csv.zip`.

(e) Now let's check how big the file size is. This is done by typing `ls -al --block-size=MB FL_insurance_sample.csv`

(f) When downloading a file it is sometimes also useful to examine the first $N$ lines of the file. The `head` command provides this option: `head -5 FL_insurance_sample.csv`

(g) Something is weird here! The output from the immediately preceding command wasn't just five lines. Let's check how many lines the file has: `wc -l FL_insurance_sample.csv`

(h) It turns out that the file we are dealing with has a problem with its "end of line" (EOL) interpreter. Because the file was created on Mac OS, it isn't formatted appropriately to be read on a Linux system. Knowing how to fix EOL conversion issues is a key skill for a data scientist to have! We can convert the file to what we need by typing `dos2unix -c mac FL_insurance_sample.csv`

(i) Now run `head` again and you should be able to see something that resembles a CSV file.

(j) Run `wc` again and there should be more than one line in the file!

The above tasks may seem mundane, but it's important to be able to: $(i)$ download a data set from somewhere on the internet using the command line; and $(ii)$ know how to check that the file you downloaded looks like what you expect it to, without having to open a program like Excel, etc. If the Florida insurance sample file were 100x as large, it would be much more difficult to do this outside the command line!

4. Now that you've downloaded and cleaned up the example file, let's create a shell script that contains all of the commands you issued in the previous question. This will make sure that your work is reproducible! The following steps will help you create a shell script that you can execute from your terminal:

   (a) First, let's create the shell script file. Call it PS3.sh. The way to do this at the command line is by typing `touch PS3.sh`. "Touch" means to create a file (if it doesn't exist), or to update the modified time stamp of a file (if it does exist).

   (b) Now edit `PS3.sh` using a text editor of your choice. If you don't want to have to transfer the file, I would recommend typing `nano PS3.sh` at the command line prompt. This will open the built-in `nano` text editor, which has very few features, but which has enough for you to do write the following script.

   (c) Once in your text editor, start the script on line 1 by invoking the Linux "shebang" term, which tells the operating system that this is a script that should be executed. The "shebang" term is: `#!/bin/sh`.

(d) Now copy the commands you issued in points (a) through (j) of the previous question.

(e) You should now have a shell script that contains 11 lines of code.

(f) Now close out of the shell script and save it. If you're using `nano`, hit `Ctrl+X` followed by Y. **NOTE: If you are a Mac OS user, "Ctrl" means "Control" *not* Command**

The final step in getting your shell script operational is to make it "executable." This tells the operating system that the file, even though it is a text file, should be interpreted as a sequence of commands.

The way to make it executable is via the `chmod` command: `chmod 774 PS3.sh`.

Now to execute the command, type `./PS3.sh` and watch it run. You now have a reproducible way to download and clean a data file from an internet source.

5. Now write a SQL script that does the following:[1]

(a) Read in the Florida insurance data CSV file

(b) Print out the first 10 rows of the data set

(c) List which counties are in the sample (i.e. list unique values of the `county` variable)

(d) Compute the average property appreciation from 2011 to 2012 (i.e. compute the mean of `tiv_2012 - tiv_2011`)

(e) Create a frequency table of the `construction` variable to see what fraction of buildings are made out of wood or some other material

6. Go to `www.overleaf.com` and create another .tex document, this time naming it `PS3_LastName.tex`. In it, write down your answers to the following questions, obtained from the shell commands you practiced above.

1. How big was the insurance .csv file (once uncompressed)? (i.e. what was the output in 3(e)?)
2. How many lines did the CSV file have before doing the end-of-line (EOL) conversion? (see 3(g))
3. How many lines did the CSV file have after doing the EOL conversion? (see 3(j))

[1] Recall that two dashes (`--`) are how to comment a line in SQL. You should comment your code well!

7. Compile your .tex file, download the PDF and .tex file, and transfer it to your cloned repository on OSCER using your SFTP client of choice. (More detailed directions on how to do this are contained in PS2.)

8. You should turn in the following files: .tex, .pdf, .sql, and .sh. Make sure that these files each have the correct naming convention (see top of this problem set for directions) and are located in the correct directory (i.e. `/DScourseS22/ProblemSets/PS3`).

9. Update your local git repository (in your OSCER home directory) by using the commands in Problem Set 2. Once you have done this, issue a `git pull` from the location of your other local git repository (e.g. on your personal computer). Verify that the PS3 files appear in the appropriate place in your other local repository.

10. Synchronize your fork with the class repository by doing a `git fetch upstream` and then merging the resulting branch. More simply, you may also just go to your fork on GitHub and click the button that says "Fetch upstream." Then make sure to pull any changes to your local copy of the fork.

    - If you decide to sychronize your fork via the command line, make sure that, before doing so, you have set your default git text editor to Nano (and not Vim) by typing the following at the command line: `git config --global core.editor "nano"`