MED277 Lab: Regular Expressions

Fall 2021

Michael Hogarth, MD

Goals

The purpose of this lab is to learn the basics of regular expressions. Regular expressions, or “regex” is a system agnostic syntax developed to define string-searches in “find” or “find and replace” functions. It was originally formalized by mathematician Stephen Kleene and incorporated in Unix text processing utilities. Regular expression syntax became increasingly popular after being added to Perl, a broadly used scripting language to automate processes on Unix systems, and eventually other operating systems.

The fundamental approach in regular expressions is the ability to use of metacharacter-driven syntax that enables the user to describe multiple variations of string “patterns” to match.

During this lab you will:

1. Use PostgreSQL pgAdmin to connect to a remote database and download 10 admission notes from the MIMIC3 clinical dataset
2. Create a Python program to open/load the 10 admission notes for processing with regular expressions
3. Use regular expressions in your Python program to detect sentences in a paragraph
4. Use regular expressions to redact numbers from a given paragraph
5. Use regular expressions to create a “detector” of history of present illness (HPI) sections in your file
6. Use your sentence detector to decompose the HPI sections further into individual sentences
7. Challenge: Use regular expressions in PostgreSQL to perform HPI section detection.

Expected Time:

You should anticipate spending around 2-3 hours on this lab exercise, depending on your level of experience with Python programming and PostgreSQL querying. You will have 1 week to complete the exercise by uploading your working Python program and 10-note file exported from PostgreSQL.

Getting Started -- Tools for this Lab

To go successfully do this lab exercise, you will need to have a few things available on your computer. These include (1) Python 3.x (at least v3.8), (2) a text editor, a (3) Python integrated development environment (IDE), and (4) pgAdmin4 (PostgreSQL client). An excellent Python IDE is JetBrains PyCharm. A community (free, open source) version of PyCharm is available for WIndows, Linux, and Mac OS, and can be downloaded from:
https://www.jetbrains.com/pycharm/download/

There are several excellent text editors. One of my favorites is Sublime, which is available as a free or for-purchase version for Mac, Windows, or Linux:
https://www.sublimetext.com/download

You will need to install pgAdmin4 (latest stable release) in order to connect to the course database server to retrieve 10 admission notes:

https://www.pgadmin.org/download/

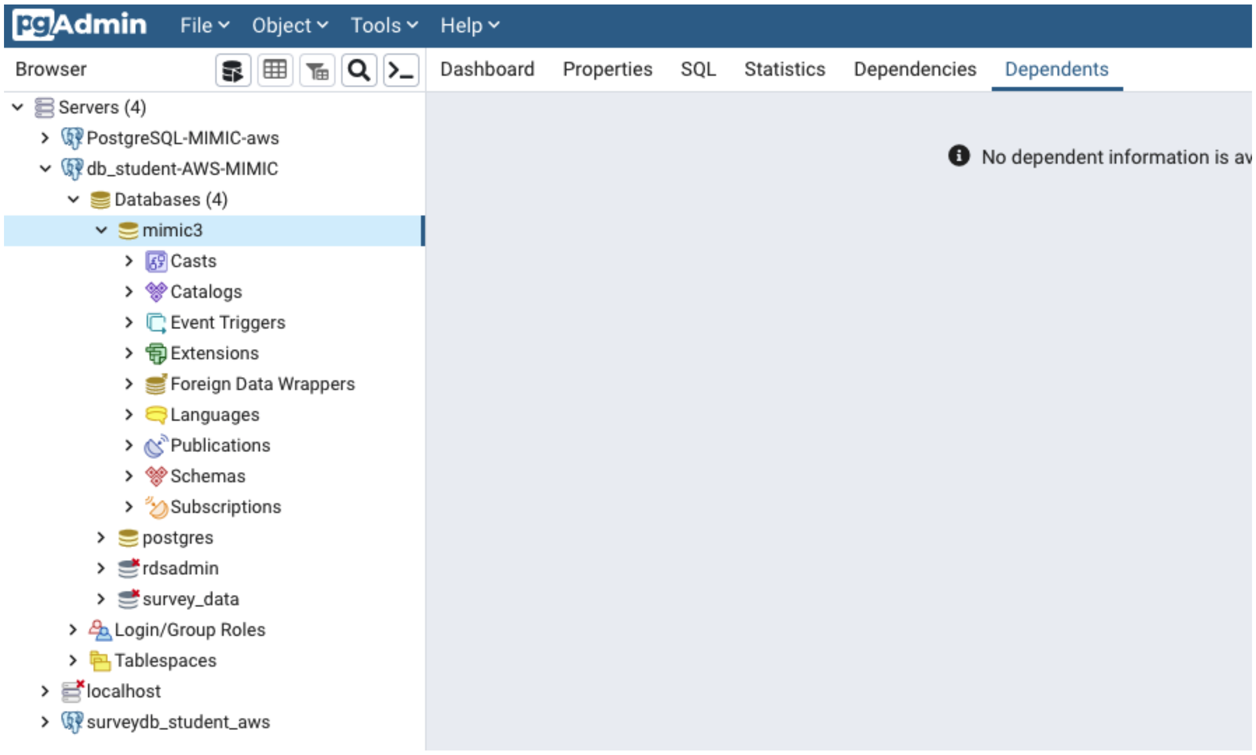
Ok, let's get started.

Part 1: Connecting to the database and retrieving 10 notes

The first thing you will need to do in this lab exercise is to retrieve 10 admission notes from the MIMIC3 clinical data set. For this course, I am providing you access to a PostgreSQL instance running in Amazon Web Services (AWS) so you can query/retrieve the notes. Start by opening pgAdmin. You will need to create a 'server connection' using the connection parameters supplied in the lab assignment.

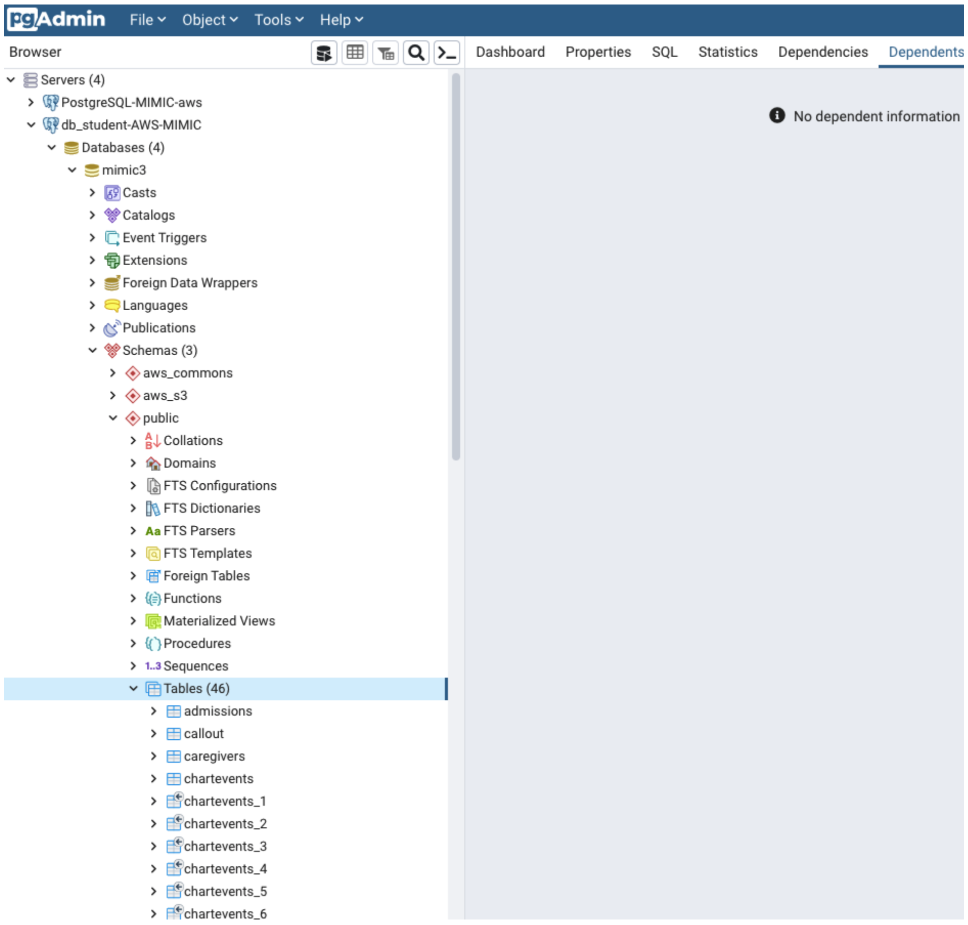
Enter the host address, port, default database, user, and password. PgAdmin will attempt to connect immediately. If you are able to expand the server entry (click the right bracket next to the connection name), you have succeeded!

You should see something like the screen shown below. Click “mimic3” in the database list under your connection and the tree will expand to show various object types in the database.



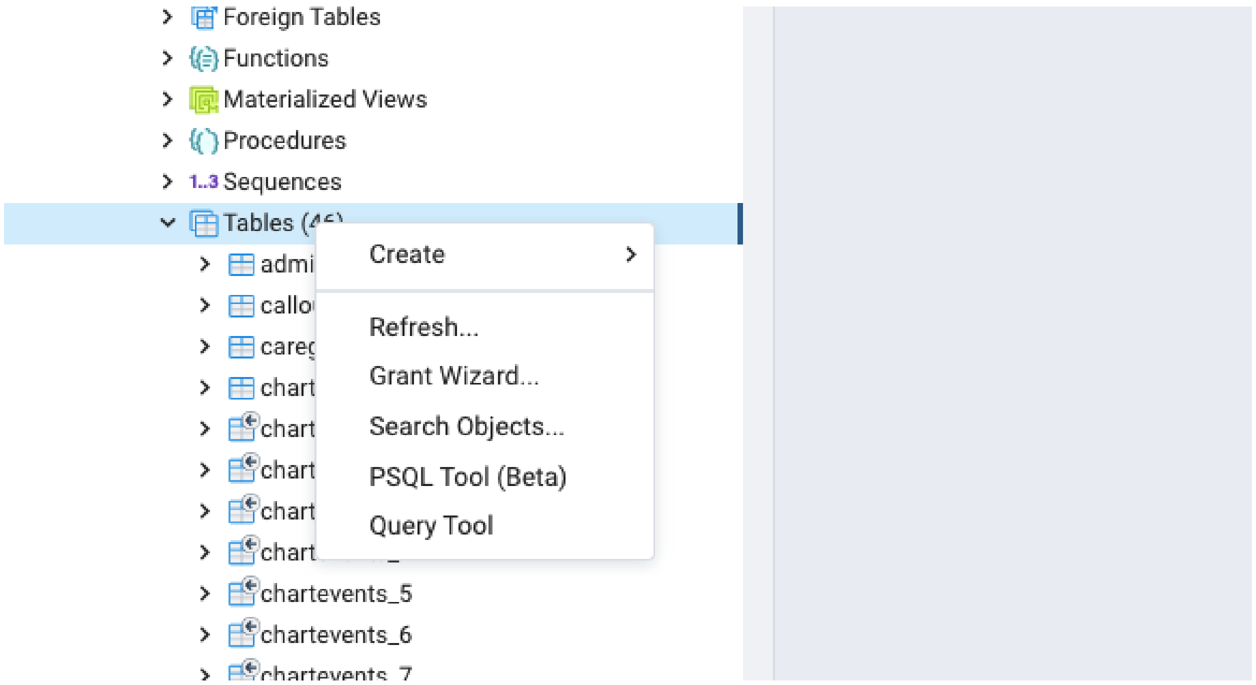
No dependent information is av

To explore the available MIMIC3 database tables in PostgreSQL, select “schemas”, then select “public” as the schema, then select “Tables”. You should see a list of MIMIC3 tables (see below) :



One of the tables is called “noteevents” and it contains all the narrative clinical notes and reports for 46,520 patients who were admitted at Beth Israel Deaconess hospital between 2001-2012.

To query the database, right click on “Tables” and a dialog box will appear with “Query Tool” at the bottom. Click “Query Tool” and a query window will open where you can begin querying the database.

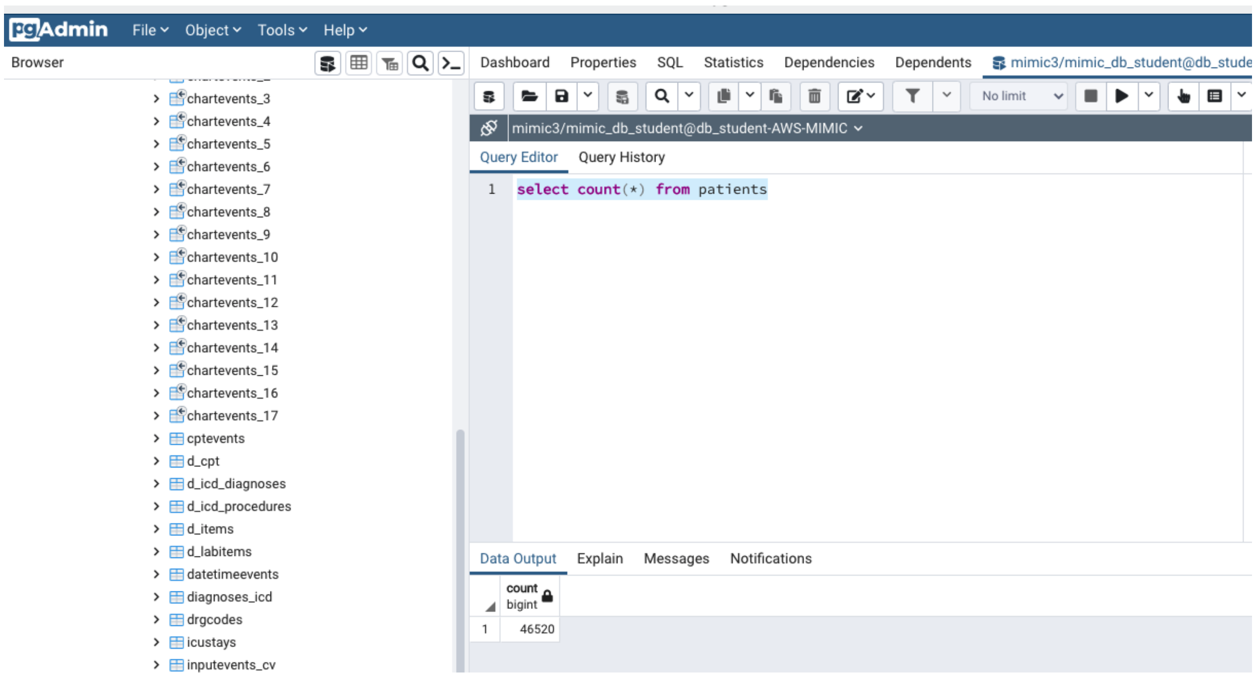


Finding the Query Tool

Once you open a query editor using the query tool, perform the following SQL query (type it exactly), then click the right sided triangle in the menu bar just above the query editor window itself. That right sided triangle will execute the following query:

```
select count(*) from patients
```

The results will be returned in the data output window.



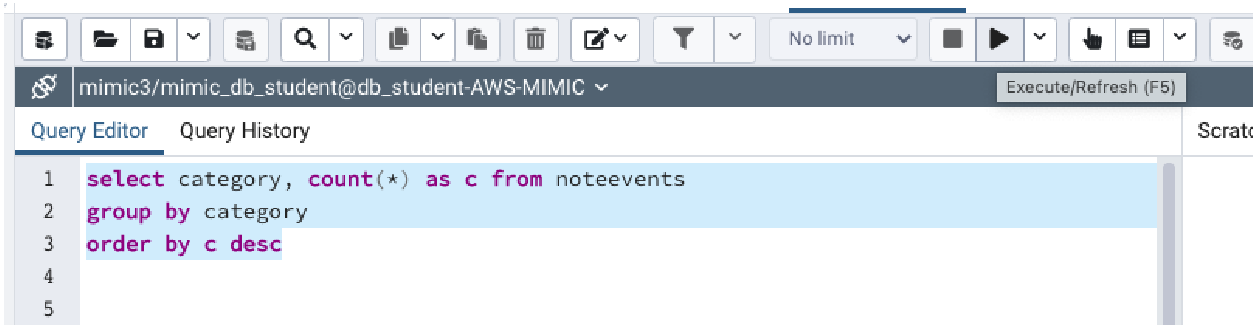
```
select count(*) from patients
```

```
  count
  bigint
1 46520
```

You have now verified that you can connect with the course database and perform queries. You are now ready to export the 10 clinical notes. To do this, we want to only retrieve physician notes. However, you will need to explore what kind of notes there are. The table has a column called "category" which looks promising. To list all the categories and the number of rows (notes) in the table with those categories, run the following query:

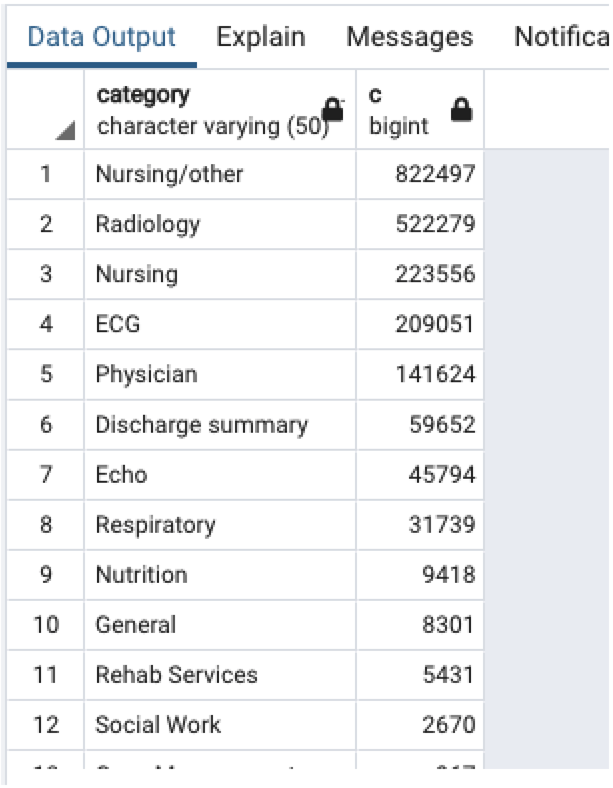
```
select category, count(*) as c from noteevents group by category order by c desc
```

You can cut/paste from here into the pgAdmin tool and it will work. SQL commands/queries do not use carriage returns so you can break the query up into multiple lines, then highlight all the lines and hit the 'execute/run' button.



```
select category, count(*) as c from noteevents
group by category
order by c desc
```

You should see the following as output.

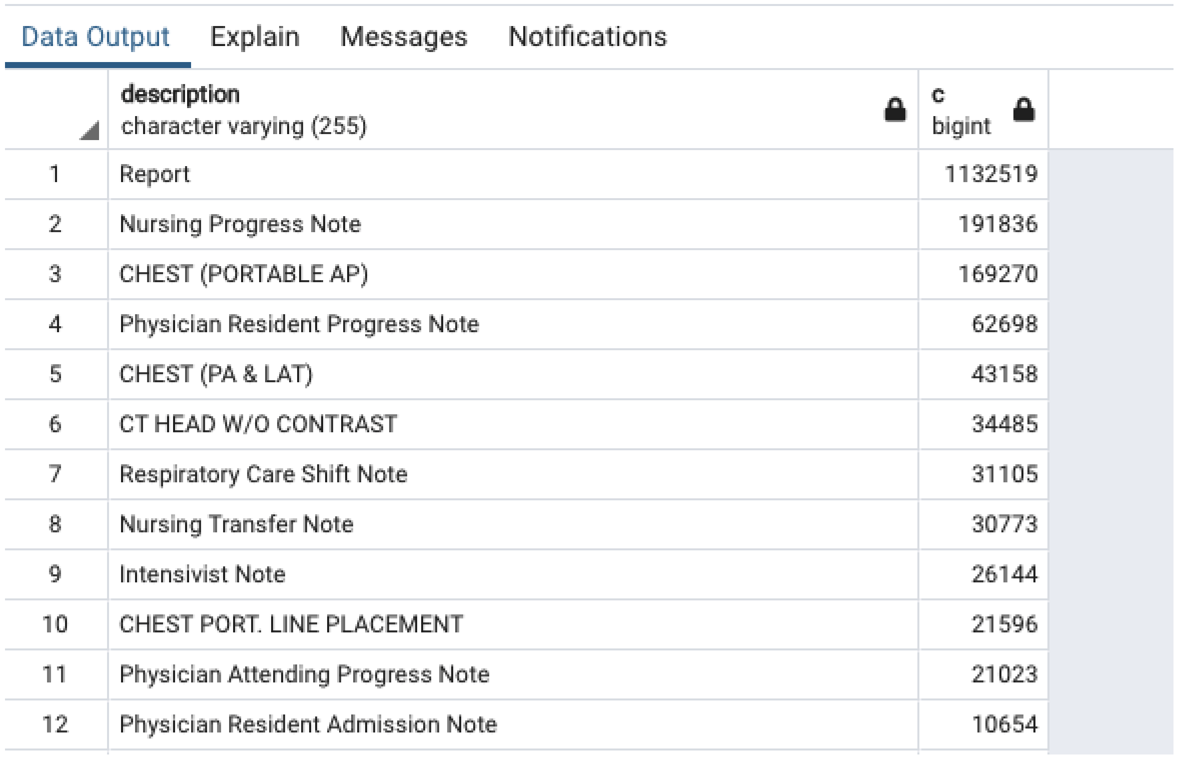


	category character varying (50)	c bigint
1	Nursing/other	822497
2	Radiology	522279
3	Nursing	223556
4	ECG	209051
5	Physician	141624
6	Discharge summary	59652
7	Echo	45794
8	Respiratory	31739
9	Nutrition	9418
10	General	8301
11	Rehab Services	5431
12	Social Work	2670

There are 141,624 physician notes in the database. To find out what kind of notes, run the following query, which enumerates the different description strings and lists them by frequency/count.

```
select description, count(*) as c from noteevents
group by description
order by c desc
```

You should see something like this:



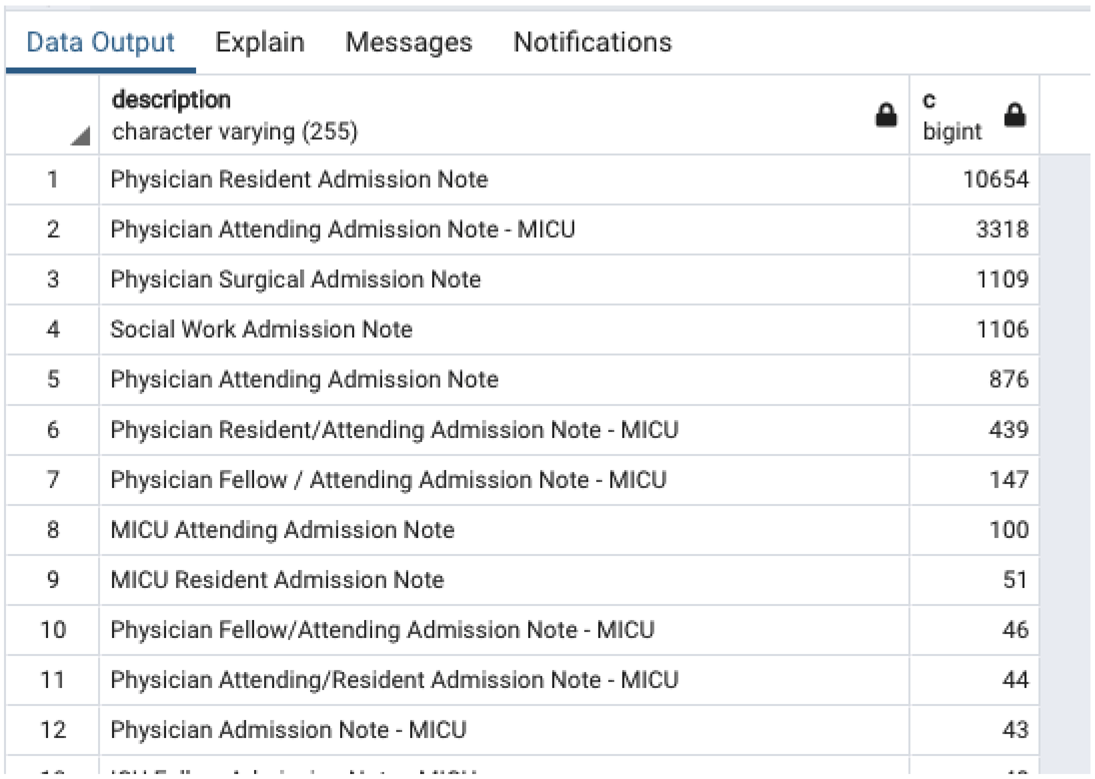
Data Output | Explain | Messages | Notifications

	description character varying (255)	c bigint
1	Report	1132519
2	Nursing Progress Note	191836
3	CHEST (PORTABLE AP)	169270
4	Physician Resident Progress Note	62698
5	CHEST (PA & LAT)	43158
6	CT HEAD W/O CONTRAST	34485
7	Respiratory Care Shift Note	31105
8	Nursing Transfer Note	30773
9	Intensivist Note	26144
10	CHEST PORT. LINE PLACEMENT	21596
11	Physician Attending Progress Note	21023
12	Physician Resident Admission Note	10654

We still do not know how to identify admission notes, so we will now add a condition that retrieves note descriptions that have the word 'admission' in the description.

```
select description, count(*) as c from noteevents
where trim(lower(description)) like '%admission%'
group by description
order by c desc
```

This query includes two string manipulation functions (trim, lower). Trim() removes any leading or following spaces from the data in description before comparing it with the word 'admissions'. lower() will lower case the string. The query also uses a string pattern wildcard (%) to detect the word 'admission' if it occurs anywhere in the description. This just makes it more likely that find all of the relevant descriptions regardless of extra spaces, case, or where the word 'admissions' appears in the string.



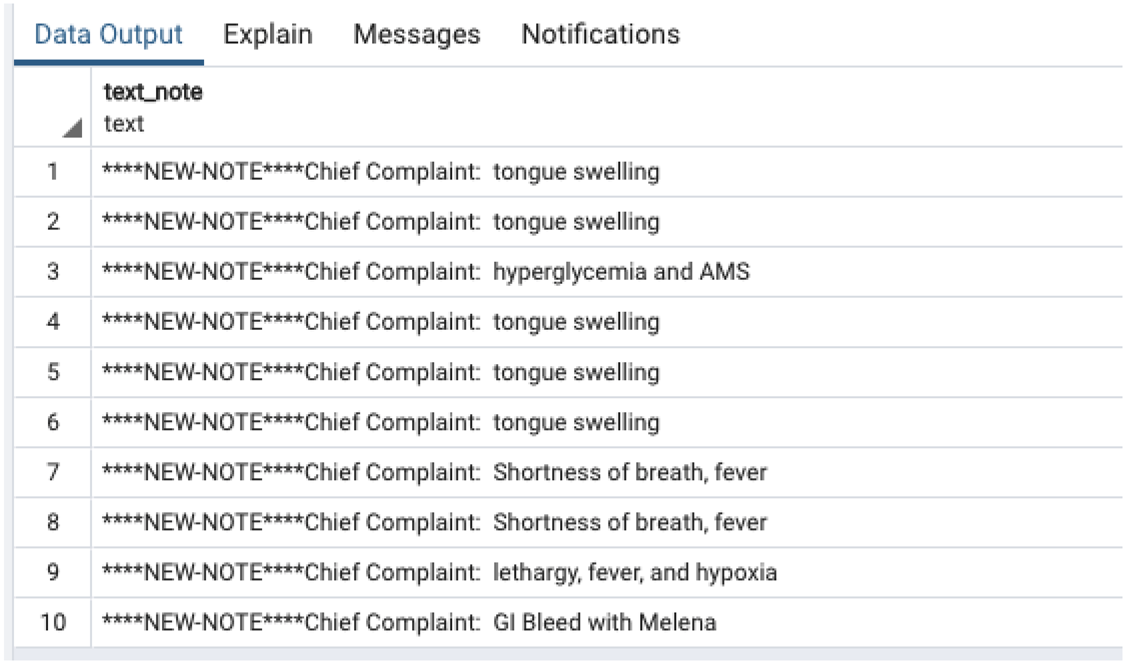
Data Output | Explain | Messages | Notifications

	description character varying (255)	c bigint
1	Physician Resident Admission Note	10654
2	Physician Attending Admission Note - MICU	3318
3	Physician Surgical Admission Note	1109
4	Social Work Admission Note	1106
5	Physician Attending Admission Note	876
6	Physician Resident/Attending Admission Note - MICU	439
7	Physician Fellow / Attending Admission Note - MICU	147
8	MICU Attending Admission Note	100
9	MICU Resident Admission Note	51
10	Physician Fellow/Attending Admission Note - MICU	46
11	Physician Attending/Resident Admission Note - MICU	44
12	Physician Admission Note - MICU	43

Ah, we now have a way of retrieving admission notes! Although there are many different descriptions, we will use the most frequent ("Physician Resident Admission Note"). To retrieve 10 of those notes, and include a "marker" added as a string to the output for each note, run the following query (you can cut/paste it into pgAdmin):

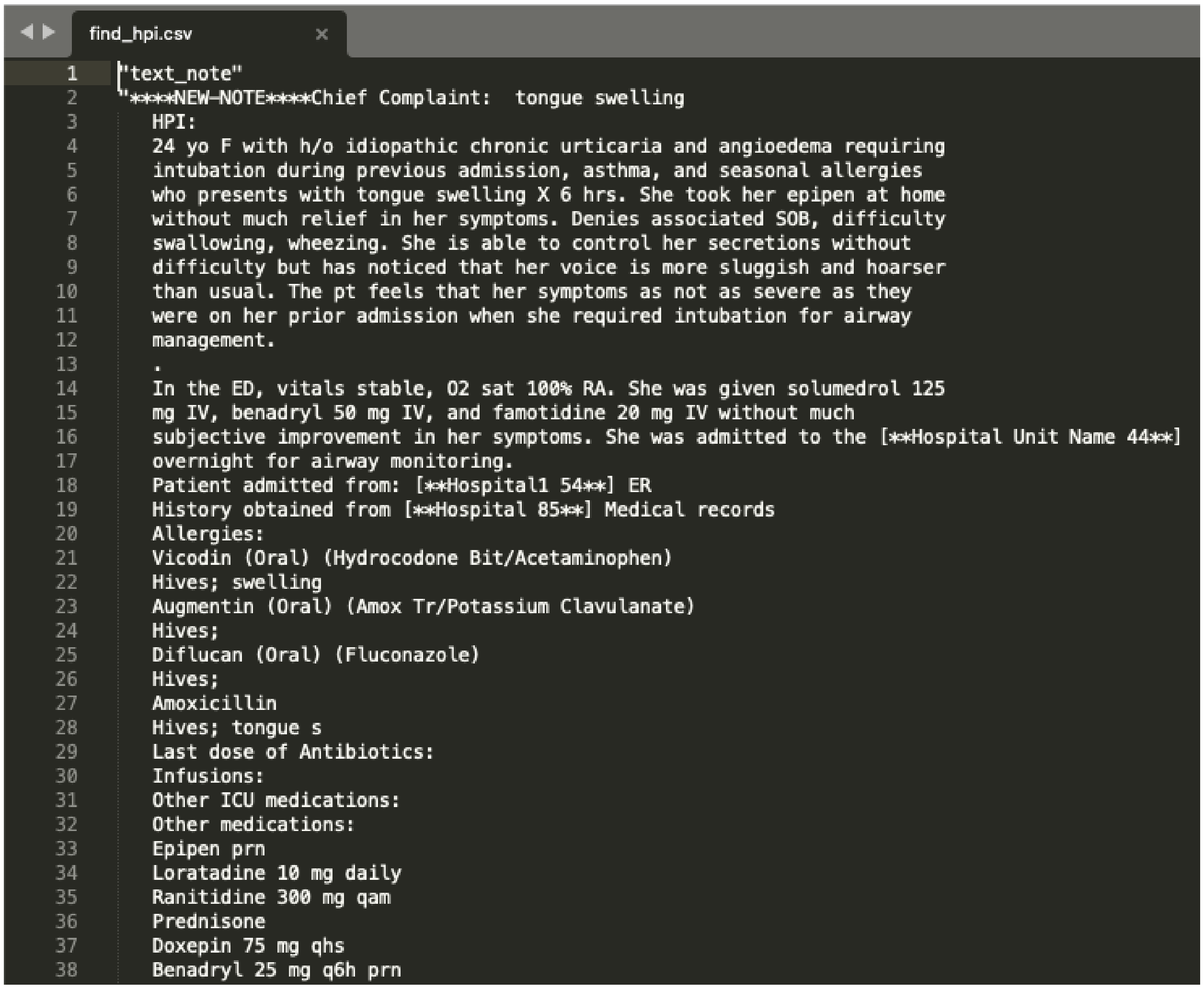
```
select '****NEW-NOTE****' || text AS text_note from noteevents
where trim(lower(category)) = 'physician'
and trim(lower(description)) = 'physician resident admission note'
limit 10
```

You should see the following output from that query:



	text_note text
1	****NEW-NOTE****Chief Complaint: tongue swelling
2	****NEW-NOTE****Chief Complaint: tongue swelling
3	****NEW-NOTE****Chief Complaint: hyperglycemia and AMS
4	****NEW-NOTE****Chief Complaint: tongue swelling
5	****NEW-NOTE****Chief Complaint: tongue swelling
6	****NEW-NOTE****Chief Complaint: tongue swelling
7	****NEW-NOTE****Chief Complaint: Shortness of breath, fever
8	****NEW-NOTE****Chief Complaint: Shortness of breath, fever
9	****NEW-NOTE****Chief Complaint: lethargy, fever, and hypoxia
10	****NEW-NOTE****Chief Complaint: GI Bleed with Melena

Don't worry, pgAdmin only shows the first part of the text fields. Download the results by clicking on the download button in the query editor menu just above the query box (⤓) and save the file. Now, open the file using your text editor.


find_hpi.csv

```
"text_note"
"****NEW-NOTE****Chief Complaint:  tongue swelling
   HPI:
   24 yo F with h/o idiopathic chronic urticaria and angioedema requiring
   intubation during previous admission, asthma, and seasonal allergies
   who presents with tongue swelling X 6 hrs. She took her epipen at home
   without much relief in her symptoms. Denies associated SOB, difficulty
   swallowing, wheezing. She is able to control her secretions without
   difficulty but has noticed that her voice is more sluggish and hoarser
   than usual. The pt feels that her symptoms as not as severe as they
   were on her prior admission when she required intubation for airway
   management.
   .
   In the ED, vitals stable, O2 sat 100% RA. She was given solumedrol 125
   mg IV, benadryl 50 mg IV, and famotidine 20 mg IV without much
   subjective improvement in her symptoms. She was admitted to the [**Hospital Unit Name 44**]
   overnight for airway monitoring.
   Patient admitted from: [**Hospital1 54**] ER
   History obtained from [**Hospital 85**] Medical records
   Allergies:
   Vicodin (Oral) (Hydrocodone Bit/Acetaminophen)
   Hives; swelling
   Augmentin (Oral) (Amox Tr/Potassium Clavulanate)
   Hives;
   Diflucan (Oral) (Fluconazole)
   Hives;
   Amoxicillin
   Hives; tongue s
   Last dose of Antibiotics:
   Infusions:
   Other ICU medications:
   Other medications:
   Epipen prn
   Loratadine 10 mg daily
   Ranitidine 300 mg qam
   Prednisone
   Doxepin 75 mg qhs
   Benadryl 25 mg q6h prn
```

You now have 10 admission notes and are ready to start exploring the use of regular expressions with this file.

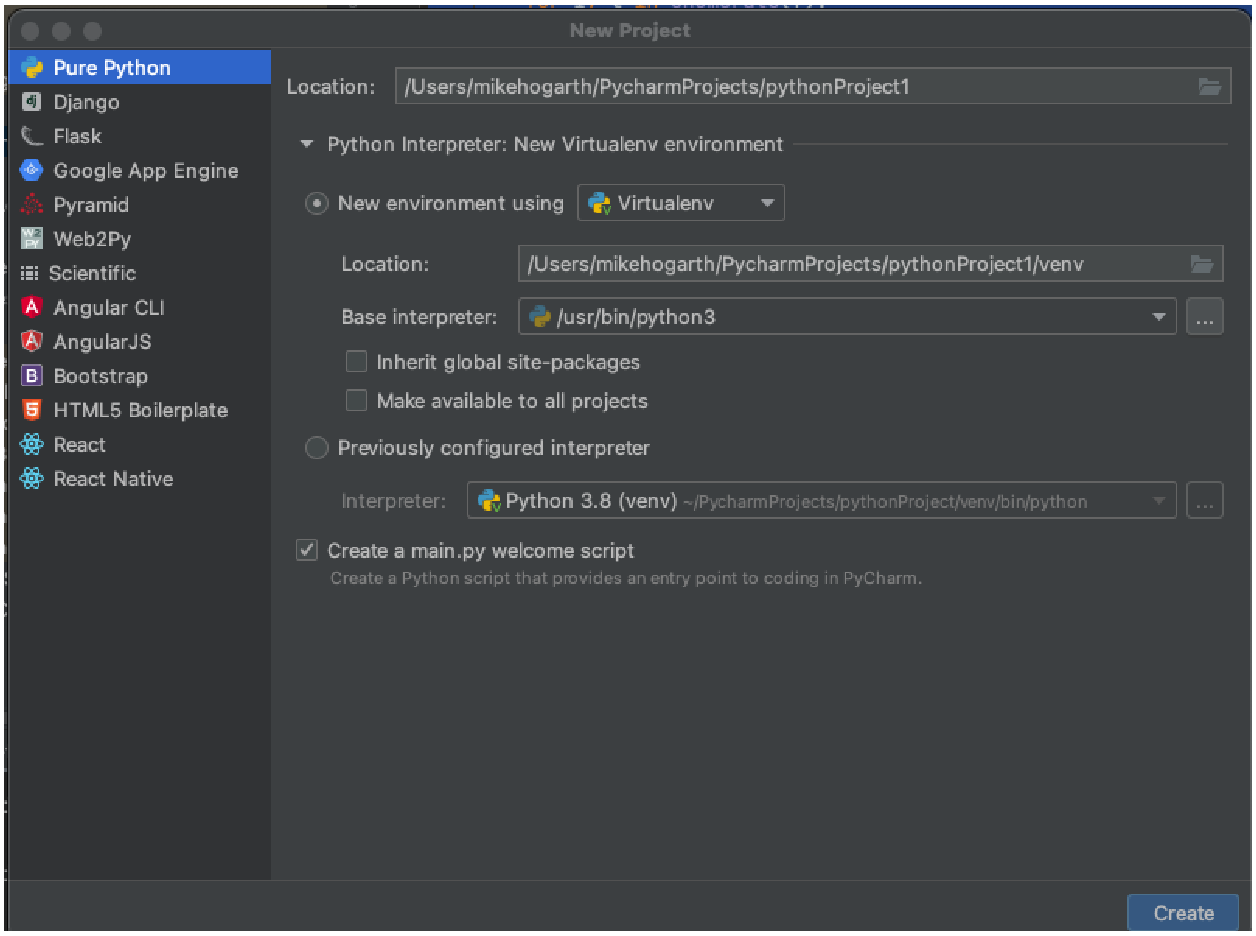
Part 2: Regular Expression Basics

To start with a basic understanding of regular expressions in Python, please read the “*Simple Patterns*” and “*Performing Matches*” sections in this Python web-documentation (will take you ~30min):

https://docs.python.org/3/howto/regex.html

Part 3: Using your Python IDE to Count Lines in the File

Open your Python IDE and create a new Python file (.py). Python is a versatile yet simple programming language. For this lab, I recommend creating a single python program file and use Python 'functions' for uses of regular expressions on the notes. Start with something simple like a function that simply opens the file and counts the number of lines. This will test that you can open the file from your program running in the IDE. To do this with PyCharm, start a project (give it any name) and it will create a 'virtual environment' (venv). You can put it anywhere on your computer, including leaving it in the default. Please note the directory you choose as the project location.



Location: /Users/mikehogarth/PycharmProjects/pythonProject1

Python Interpreter: New Virtualenv environment

(●) New environment using Virtualenv

Location: /Users/mikehogarth/PycharmProjects/pythonProject1/venv

Base interpreter: /usr/bin/python3

☐ Inherit global site-packages

☐ Make available to all projects

( ) Previously configured interpreter

Interpreter: Python 3.8 (venv) ~/PycharmProjects/pythonProject/venv/bin/python

☑ Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

Make sure to verify the new environment is “Virtualenv”. Now copy the 10-note file into the venv directory for your project (in the example above: `/Users/mikehogarth/PycharmProjects/pythonProject1`)

Having the file in the project root directory will let you open it with python programs in the IDE without having to use lengthy paths.

Now create your lab1 program by starting with a small function that opens the file and counts the number of lines.

```
import re
def file_line_count(fname):
    with open(fname) as f:

       for i, l in enumerate(f):

           pass

       return i+1



file = 'find_hpi.csv'

print ("file:"+file+"\nline count:", file_line_count(file))
```

If you cut/paste this into your IDE, make sure to modify the file name to the name you used for the file!

If this is successful, you have demonstrated that your Python IDE is working and that you have the 10-note file in the right place.

Part 4: Using Python and Regular Expressions to extract the HPI sections

The rest of the lab is pretty much up to you now! For the rest of the lab exercise, you will add to your python program by writing functions that do the following:

1. Identifies the history of present illness (HPI) section for the 10 notes in the file and counts the number of sections.
2. A sentence "splitter" that will return the sentences in a given HPI section (hint:
3. A function that redacts all numbers from a given HPI section

I will provide hints only. There are multiple different coding approaches to these functions. You will only be graded on your program's ability to perform the functions above.

Please write the following functions:

find_hpi_sections(fname) -- takes a filename as a parameter and detects the HPI section (hint: use the two strings "HPI:" and "Allergies:" to detect the beginning and end of the HPI sections. Use the "starts with" metacharacter).

find_sentences(input_text) -- identifies the sentences in a section of text (hint: use the regular expression function `re.split()` and a pattern for a period at the end of a sentence)

redact_numbers(input_text) -- uses reg expressions to replace numbers in the input_text with the phrase [num].

Challenge:

For those who would like to challenge themselves and to learn a convenient way of invoking regular expressions in SQL, write a PostgreSQL query that will identify the HPI sections. Here is documentation for regexp_matches():

https://www.postgresqltutorial.com/postgresql-regexp_matches/

Part 5: Assignment

Your are required to have a program that will output the following in this order:

1. line count in the file
2. the start line and end line for the HPI sections
3. the number of HPI sections your program has detected
4. print each HPI section while inserting a dashed line between HPI sections.

5. Take the last HPI section you have detected and decompose only that one into sentences, and print the sentence and the sentence count number and length
6. Take the last HPI section only and redact all the numeric digits and replace them with [num]

Please turn in the following 3 items for the assignment:

1. **Your python program**
2. **The output of the python program**
3. **Your 10-note file**
4. **If you take the challenge -- provide a file with the SQL that attempts to match HPI sections using the regexp_matches() function in PostgreSQL.**

I have provided an example of an output file as part of the Lab assignment in Canvas.

Good luck!!