CMPSC 312
Database Systems
Spring 2019

Lab 2 Assignment:
Relational Data Modeling for Protein Data (Part 1)
Submit deliverables through your assignment GitHib repository.
Place report document `writing/` directory

Objectives

To learn how to create a database in SQlite3 from downloaded data. You will also learn some important skills for writing queries.

GitHub Starter Link

https://classroom.github.com/a/Ysrlnb1Q

To use this link, please follow the steps below.

- Click on the link and accept the assignment
- Once the importing task has completed, click on the created assignment link which will take you to your newly created GitHub repository for this lab,
- Clone this repository (bearing your name) and work locally
- As you are working on your lab, you are to commit and push regularly. The commands are the following.
  - `git add -A`
  - `git commit -m ‘‘Your notes about commit here’’`
  - `git push`

Introduction

In bioinformatics research much of an investigator's time is spent simply managing and organizing data for study. In this lab, you are the investigator in bioinformatics! Your task is to create a simple relational database from data concerning proteins which have been studied by their association of types of disorders. Once their databases have been completed, researchers are able begin their tasks of data analysis.

In this lab, your data will concern the proteins which have been associated with Apoptosis (*Programmed cell death*) and Parkinson's disease. All of your data will come from `http://www.uniprot.org` (one of the world's leading resources of protein knowledge) and will be used to create a local database for information from proteins associated with Parkinson's disease and Apoptosis.

We will use relational data modeling to create this database and apply SQL programming to create tables, populate them and to perform our queries. This assignment focuses on making a database from scratch, designing the tables using the `CREATE TABLE` keywords, and then writing queries using the `SELECT` and `WHERE` keywords in SQL.

Data Sets

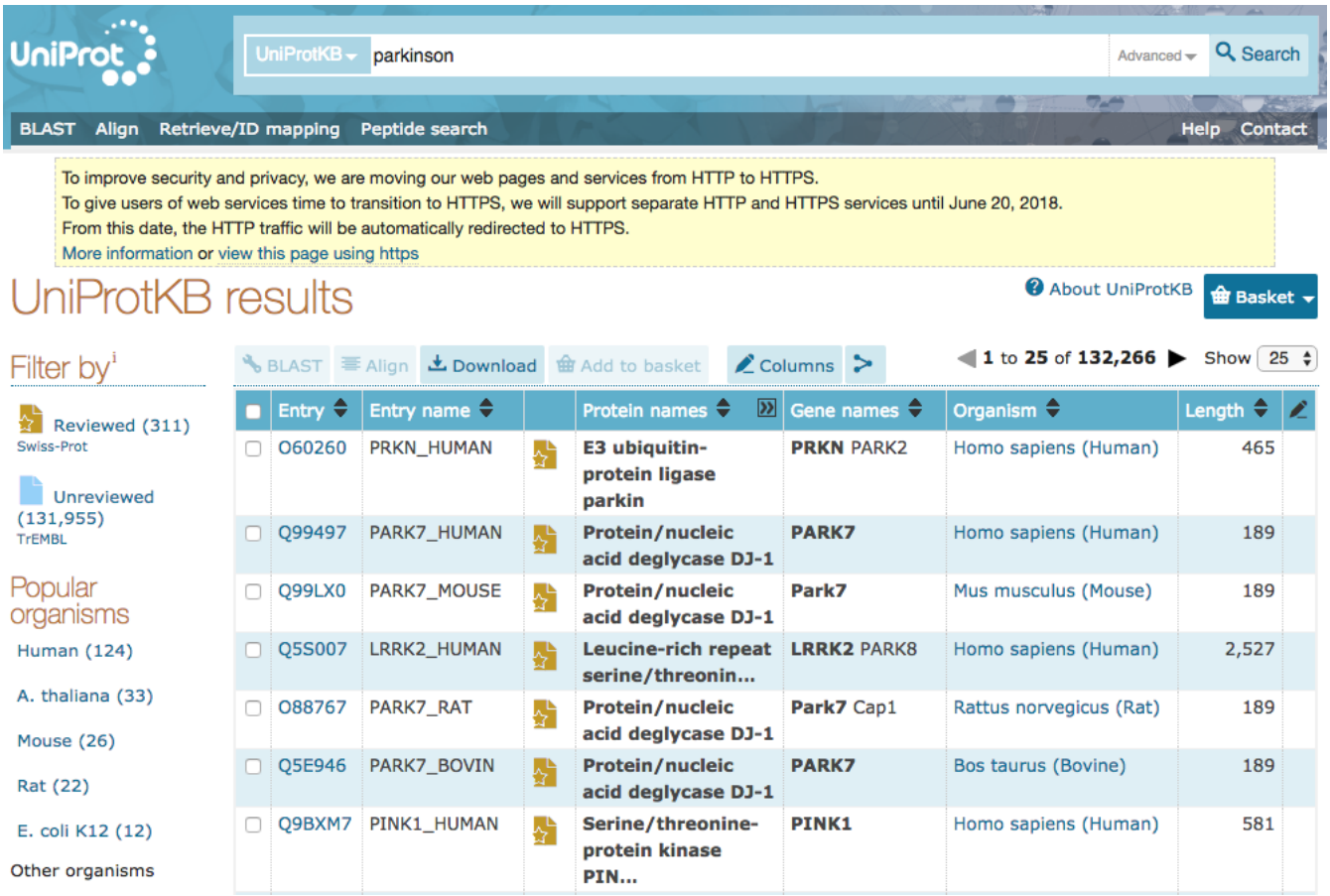

Figure 1: Results from searching *Parkinson* in UniProtKB. To create your database tables, you could use the same headers as those used by UniProt (as shown), with all spaces removed.

Your database must be designed to hold protein data from UniProt at `http://www.uniprot.org`. To obtain this data, please open your browser and find the UniProt website and perform a search of your two protein-oriented data sets; `Parkinson` and `Apoptosis`. After your search, you should be taken to an outputs page resembling Figures 1 and 2 for Parkinsons disease and Apoptosis-related disorders, respectively.

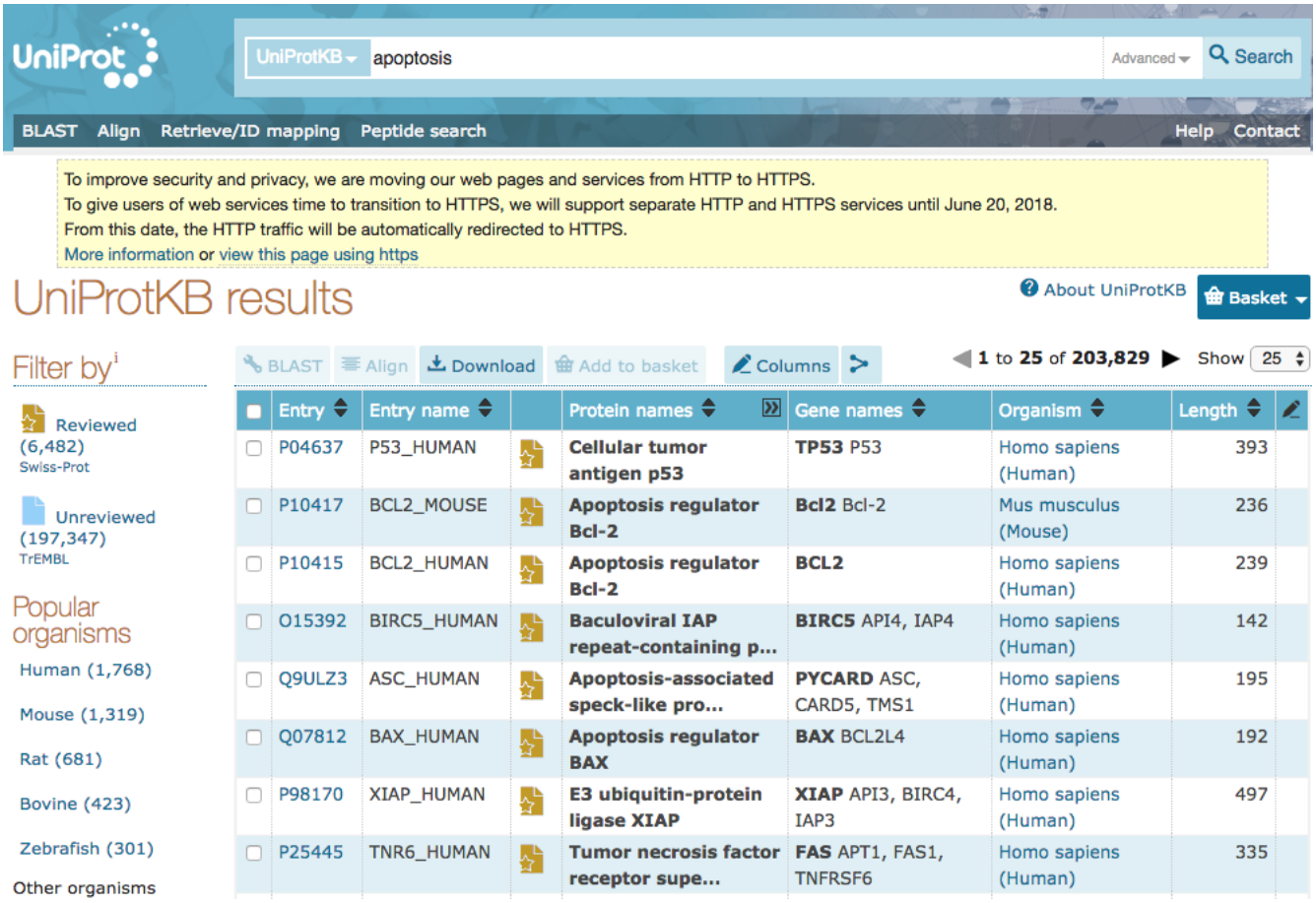

Figure 2: Results from searching for *Apoptosis* in UniprotKB.

Data file references

Your database is to contain the data from the following sources.

- `http://www.uniprot.org/uniprot/?query=apoptosis&sort=score`
- `http://www.uniprot.org/uniprot/?query=parkinson&sort=score`

Downloading and Formatting

Please download each of the returned protein listings from UniProt using the compressed, `tab-separated` options as shown in Figure 3. **PLEASE DO NOT PLACE THESE FILES IN YOUR SUBMISSION REPOSITORY; I do not want these files and they are not necessary to evaluate your grade.** You will have to set-up a working directory on your local machine (not the assignment repository) in which you will create your database from the data files.

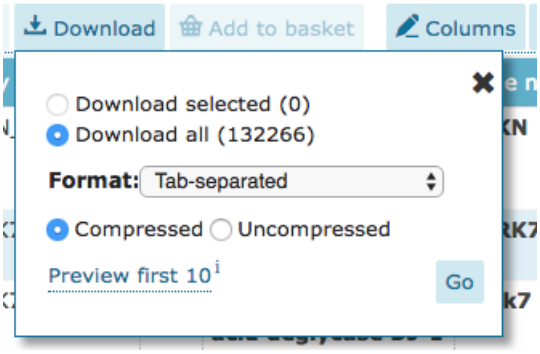

Figure 3: Download the data using the options for compressed and the tab-separated format. Please do not include these files in your submission repository which will come back to me. I do not need these files to evaluate your grade. Instead, place these files in a working directory outside of your submission repository. Use `mkdir` to create your directory for this work.

Trimming the File-Headers

Your downloaded files have the column headers given on the first line which must be removed before you can can them to your database. To do this, load each file in a text editor such as `Atom` where you can remove the first line at the top of the file. Save your work as a text file.

Prepare to Build a Database

In this part of your lab, we will be preparing a *build* file to create your protein database from file imports. If you would like to re-create your database from scratch, you can use your build-script to perform this task.

Prepare to Create your Tables

Use the code given in your class repositories (the "sandbox" directories), as well as, the template code shown in Figure 4 to begin your work on writing your `CREATE TABLE` code to build your tables. Note: Once your tables have been built, you will use the `.separator` and `.import` keywords in SQL to import the data from the downloaded files to populate your database.

```
drop table Park;
create table Park (
        protID VARCHAR NOT NULL PRIMARY KEY,
        entryName ... ,
        Status ... ,
        ProteinName ... ,
        GeneName ... ,
        Organism ... ,
        Length ...
);

drop table Apop;
create table Park (
        protID VARCHAR NOT NULL PRIMARY KEY,
        entryName ... ,
        Status ... ,
        ProteinName ... ,
        GeneName ... ,
        Organism ... ,
        Length ...
);
```

Figure 4: **This code will not execute as it is!!** Complete the code to create both of your tables: `Park and Apop`. Remember, you will need a line of code to create the attributes of your tables which contain each member of the row's tuple from the data. It is recommended that you use the following names for attributes to prepare your tables: `protID, entryName, status, proteinNames, geneNames, Organism`, and `Length`.

Compiling and Populating Your Local Database

Using the `mkdir` command, create a directory outside of your repository directories (such as on your desktop) called `myDBproject/`, and then create another directory inside it called `data/`. You will copy your data files from UniProt to `myDBproject/data` to populate your database. This is done so that your files and database do not find their way back to the class repositories (your own or the shared repository).

Use code as shown in class to create the tables and to load this data from a text file. Place all database creation and data import commands neatly in the provided text file called `src/proteinBase-build.txt` which you will used to create and reproduce your database as needed.

```
.separator "\t"

/* find the file and load it into sqlite3 which will create the database.*/
.import data/uniprot-apoptosis.tab Apop
.import data/uniprot-parkinson.tab Park


/* cat proteinBase_build.txt | sqlite3 proteinDB.sqlite3 */
```

Figure 5: Be sure to place your data files in the directory `data/`. If your files are found in another directory, the new directory must be reflected in the code above.

Querying the Database Tables

During this part of the lab you will design and run several queries to answer the following questions. You may have to look up some of these commands to complete your queries.

1. For each table (`Park` and `Apop`), write a query to determine the count of the `protID` elements. Hint: use `count(protID)`
2. For each table (`Park` and `Apop`), write a query to determine the *distinct* count of the `protID` elements. Hint: use `count(distinct(protID))`
3. For each table (`Park` and `Apop`), write a query to determine the count of the `GeneName` elements.
4. For each table (`Park` and `Apop`), write a query to determine the *distinct* count of the `GeneName` elements.
5. In the above two questions, you were asked to find the difference between `count()` and `count(distinct())` when applied to the `ProtID` and `GeneName` attributes of each table. In your opinion, which attribute (i.e., `protID` or `GeneName`) would make a better acting primary key that would ensure that each row is completely unique in the table? Inform your argument by using the differences between the results of *counts* and *distinct counts* from the above questions. Explain your reasoning.
6. For each table (Park and Apop), write a query that will return the number of distinct organisms which may be found in each table.
7. For each table (Park and Apop), determine the number of distinct `protID` entries which are associated with the organism, *Vaccinia virus*. Hint: You will need to add the `WHERE` clause to your query code.
8. Combining tables: Write a query that returns the distinct count of `ProtID` entries which are found in *both* the `Apop` and `Park` tables. Hint: Your query will be written using the below pseudocode form:

   $SELECT\ count(distinct(attribute_1))$
   $FROM\ Table_i, Table_j$
   $WHERE\ table_i.attribute_1\ ==\ table_j.attribute_1;$

9. Quality of Data: Write a query that returns the distinct listing of `GeneName` entries which are found in *both* the `Apop` and `Park` tables (i.e., simultaneous entries of the attribute.) You do not have to wait for long to note that, in the results of your query, there are blank entries listed. Explain what these blanks are and speculate how they got into your data.
10. For each table, write a query to display the `distinct(GeneName)` along with its associated `Organism`. How many entries did you find in each?
11. Write a query to return the value of the `count(distinct(proteinNames))` of all `proteinNames` which are found in both tables at the same time. This query may take some time to complete... Can you speculate why this particular query takes so long?

12. Write a query that will return the number of `Apop.protID`'s (of both tables) which are also associated with the, *Vaccinia virus* `organism`. How many are there in each table? Hint: your queries for both tables will contain code like the following.

    - `count(distinct(apop.protID))`
    - `count(distinct(Park.protID))`

```
The pseudocode will take the following form:\newline
SELECT DISTINCT(Apop.protID)
  FROM Park, Apop
  WHERE Park.attribute_1  == Apop.attribute_1
  AND Apop.attribute_1 == "Vaccinia virus";
```

Summary of the Required Deliverables

This assignment invites you to submit an electronic version of the following deliverables through Bitbucket:

1. **Query and Results document**: You will modify the file `writing/queries.md` to reflect your query code and results for each of the ten questions in red, above.
2. **Database-building file**: You will edit the file `src/proteinBase-build.txt` to be used to build your database from data files.
3. Please consider **not** pushing your data files to GitHub as they will take up a lot of space. The instructor will not use them to grade your work. Please note, that this implies that you will have to make a build directory outside of your repository.
4. Please do not forget to push the above two files to submit your work. The instructions for this are above.

In adherence to the Honor Code, students should complete this assignment on an individual basis. While it is appropriate for students in this class to have high-level conversations about the assignment, it is necessary to distinguish carefully between the student who discusses the principles underlying a problem with others and the student who produces assignments that are identical to, or merely variations on, someone else's work. Deliverables that are nearly identical to the work of others will be taken as evidence of violating Allegheny College's Honor Code.