CMPSC 312
Database Systems
Fall 2020

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/v9lO25y1

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 to COVID-19. Here we will be studying the nucleocapsid protein (N-protein) and spike protein (S-protein), which are encoded by all coronaviruses, including the SARS-CoV-2 that was first detected in Wuhan City, China, in December 2019. A formal description of the N-protein and the S-protein may be found at `https://www.raybiotech.com/covid19-proteins/`.

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 these proteins.

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.

Docker

Figure 1: Docker is often used for deploying and hosting databases in the cloud in industry and academia.

It is assumed at this point that your Docker installation is working properly and that you are able to build containers using a `Dockerfile`. In class, you were given a `Dockerfile` to build a container for SQLite work. Even if you have found an alternative method for running SQLite, it is still strongly encouraged that you use Docker containers to run your database. We note that Docker is an industry standard software that is often used to deploy and host many databases.

Data Sets

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. After your search, you should be taken to an outputs page resembling Figures 2 and 3 for the S-protein and N-protein, respectively.

Data References

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

- `https://www.uniprot.org/uniprot/?query=n-protein&sort=score`
- `https://www.uniprot.org/uniprot/?query=s-protein&sort=score`

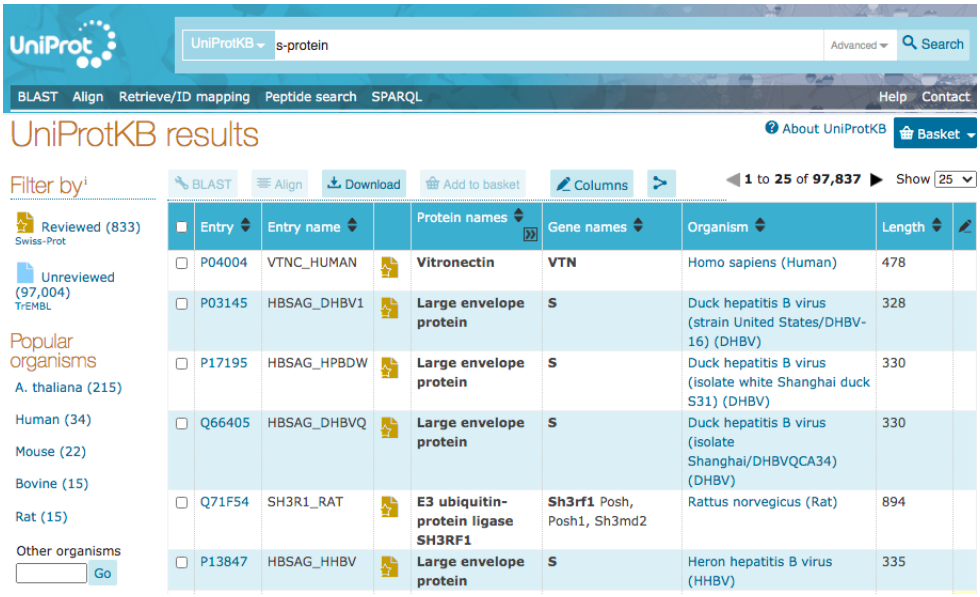

Figure 2: Results from searching *s-protein* in UniProtKB. To create your database tables, you could use the same headers as those used by UniProt (as shown), with all spaces removed.

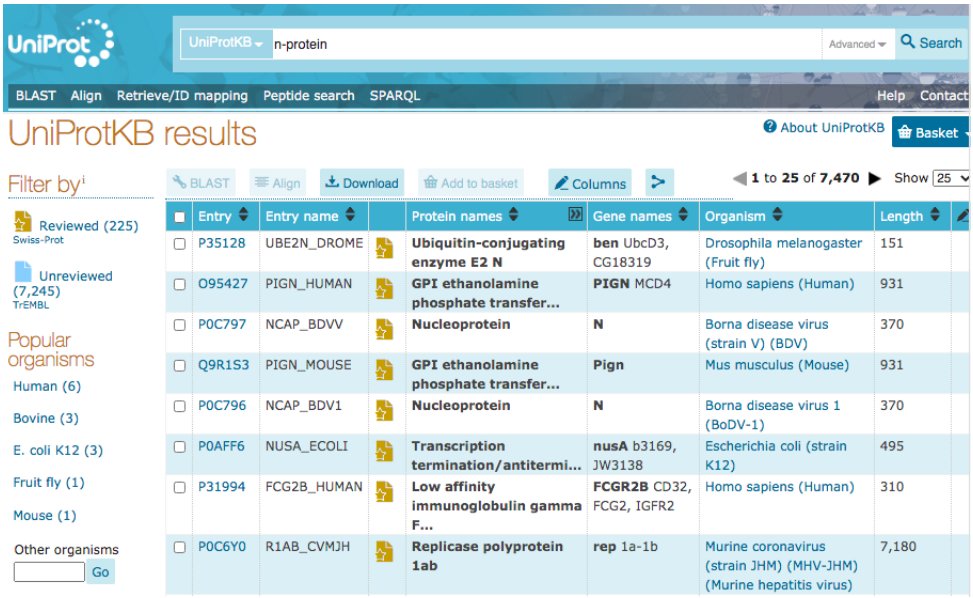

Figure 3: Results from searching for *n-protein* in UniprotKB.

Downloading and Formatting

Please download each of the returned protein listings from UniProt using the un-compressed, `tab-separated` options as shown in Figure 4. **PLEASE DO NOT PLACE THESE FILES IN YOUR SUBMISSION REPOSITORY; they are not necessary to evaluate your grade.** Please keep all your database building files in your GitHub classroom repository and not in the course *ClassDocs* repository.

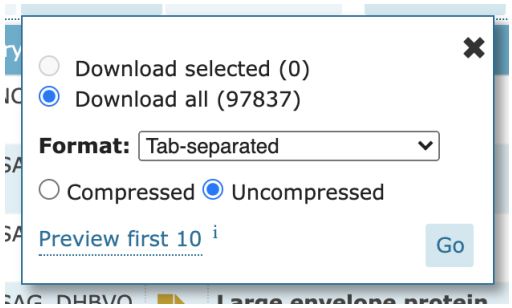

Figure 4: Download the data using the options for un-compressed and the tab-separated format. Please do not include these files in your submission repository as they are not necessary to evaluate your grade.

Trimming the File-Headers

Your downloaded files have the column headers given on the first line which must be removed before you can use them to build your database. To do this, load each file in a text editor such as `Atom` and remove the top line of the file. This line will contain the terms; *Entry*, *Entry name*, *Protein names*, and etc., and were added in the file by Uniprot as column headers. Save your work as a text file.

Preparing a Build-file to Create a Database

In this part of your lab, we will be preparing a *build* file to create your protein database from file imports. This file allows you to conveniently rebuild your database for any reason. Please edit the file `src/proteinBase – build.txt` of your repository to complete this task. You are to edit the code of the file (shown in Figure 5) to build tables. The other code in the file will not require editing, unless you change the file names of your data.

Compiling and Populating Your Local Database

Make a working directory in your repository. Be sure that your build file is in this working directory. Next, create a directory called `data/` (see Figure 6)

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

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

Figure 5: **This code will not execute as it is!!** Please complete the code to successfully create both of your tables. 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`.

in this directory where you will store your data files from uniprot; `uniprot-n-protein.tab` and `uniprot-s-protein.tab`.

```
.separator "\t"

/* find the file in the data/ path to be loaded into sqlite3.*/
.import data/uniprot-n-protein.tab nprot
.import data/uniprot-s-protein.tab sprot


/* To build your database in Docker (implying a Linux environment),  */
/* use the following command. This command also works as it is in MacOS */
/* cat proteinBase_build.txt | sqlite3 proteinDB.sqlite3 */
```

Figure 6: Be sure to place your data files in the directory `data/` since this directory has been specified by `proteinBase-build.txt`. This code contains the import functions to import your datafiles. The command to compile your database is also shown. Note: if your files are placed in another directory, then new directory must be reflected in this code.

Querying the Database Tables

Now that you have built a successful database, please answer the following questions-in-blue. When grading, the instructor will often be concentrating on the structure of your query, since the results may change depending on the age of the data. Note, to access your database with SQLite3, you will use the command,

```
sqlite3 proteinDB.sqlite3
```

1. What is the command to list the tables of the database?
2. What is the command that shows the attributes of all the tables of your database?
3. How many protID entities are contained in each of the tables in your database? Hint: use the `count()` function in your query.
4. How many unique organisms are in each table of your database? Hint: use `count(distinct())` in your query.
5. How many *reviewed* and *nonreviewed* status-entities are there in each of the tables of your database? Hint: you are to write a conditional query.
6. What is the combined sum of all protein *lengths* each of the tables of your database? Hint, use the `sum()` function in your query.
7. For each table, what are the queries to show that there are more *geneNames* than *ProteinNames* in each table of your database. There will be four queries for this response.
8. How many proteins exist in each of your table where the lengths of the proteins is equal to each value in the set {100,300, 600, 900, 1000}? Hint, this will be a different query for each of these set values. Each table will have similar queries.
9. What is the average length of the proteins in each table of your database? Hint, use the `avg()` function in your query.
10. What is the average length of all proteins in each table of your database for when the "`Organism` == "`Homosapiens(Human)`?
11. Is the average length for "Homo sapiens (Human)" larger or smaller than the average length of all proteins in each table of your database?
12. How many unique GeneNames are in each table of your database? Hint: use `count(distinct())` in your query.

Summary of the Required Deliverables

Please submit your work by pushing it to your GitHub Classroom repository.

1. **Query and Results document**: You will modify the file `writing/queries.md` to reflect your query code and results for each of the questions-in-blue, above.
2. **Database-building file**: You will submit your edited build file (`src/proteinBase-build.txt`) to be used to build your database from your data files.

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.