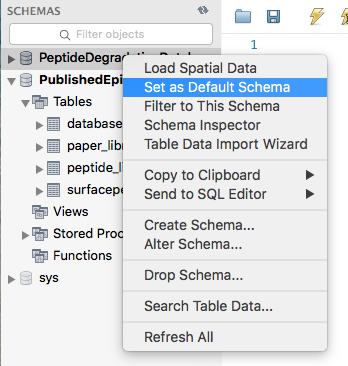
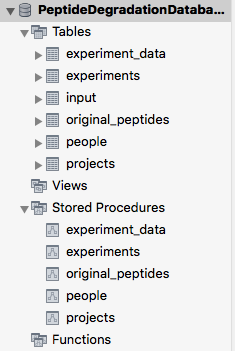
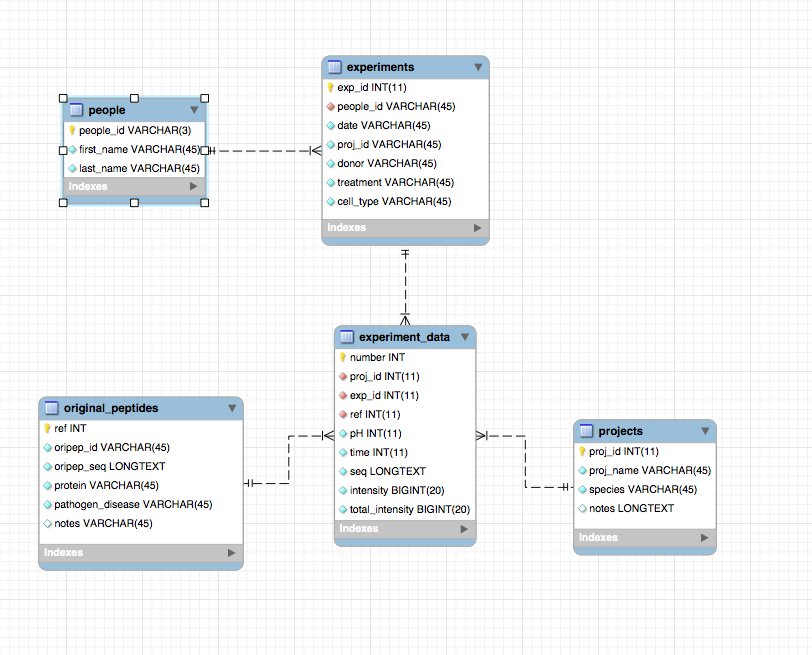
To begin, make sure that you right click on the schema name and select “Set as Default Schema.” This is to ensure that when you edit the database, the settings recognize that you are in that specific database.



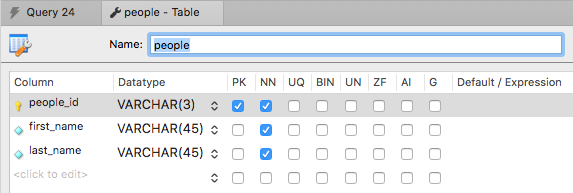
This is an overview of what the tables and stored procedures looks like on the schema drop-down menu.



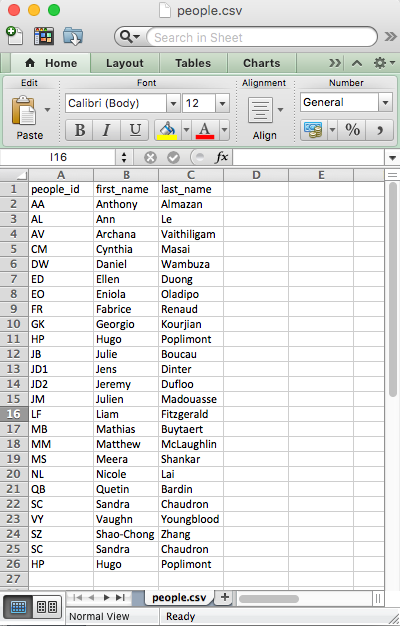


# **‘people’ Table**

This table stores the information for the person’s unique 2-letter name ID (*people\_id*), their first name (*first\_name*), and their last name (*last\_name*) from the local people.csv file.



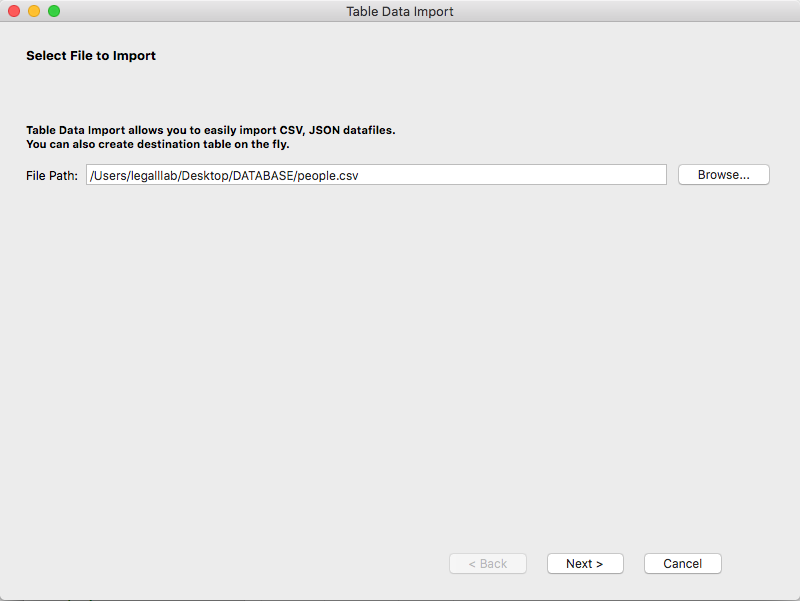
This is an example of what the .csv file can look like to be loaded onto the database.



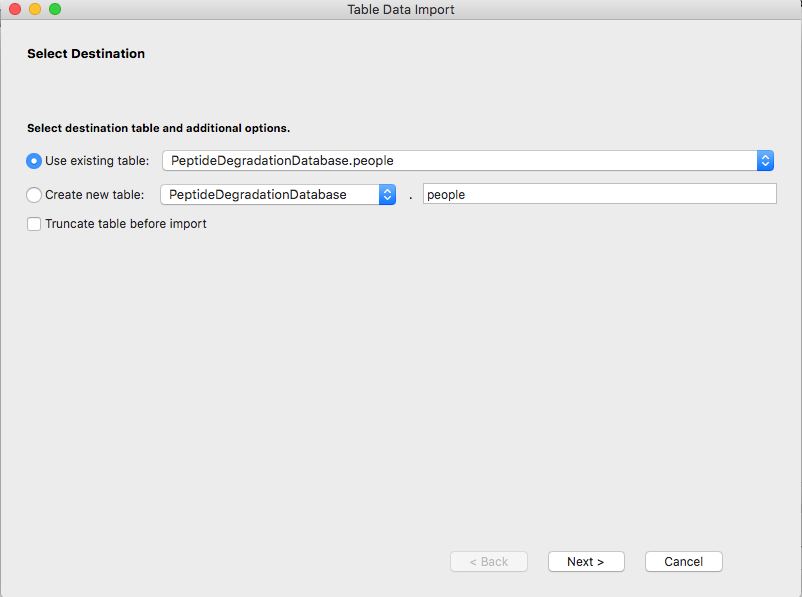
Before loading the names onto the database, make sure there are no duplicates in the people\_id column in the .csv file. If there are duplicates, keep the initials but add a number to it starting with 2.

For example, the people\_id for Allen Addison and Anthony Arnold would be AA and AA2.

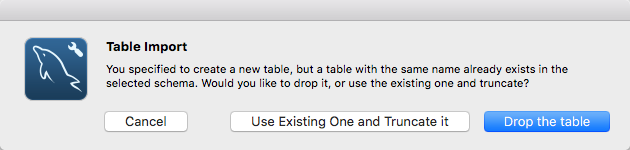
To load the .csv file, click on the “Import” button and select the people.csv file.



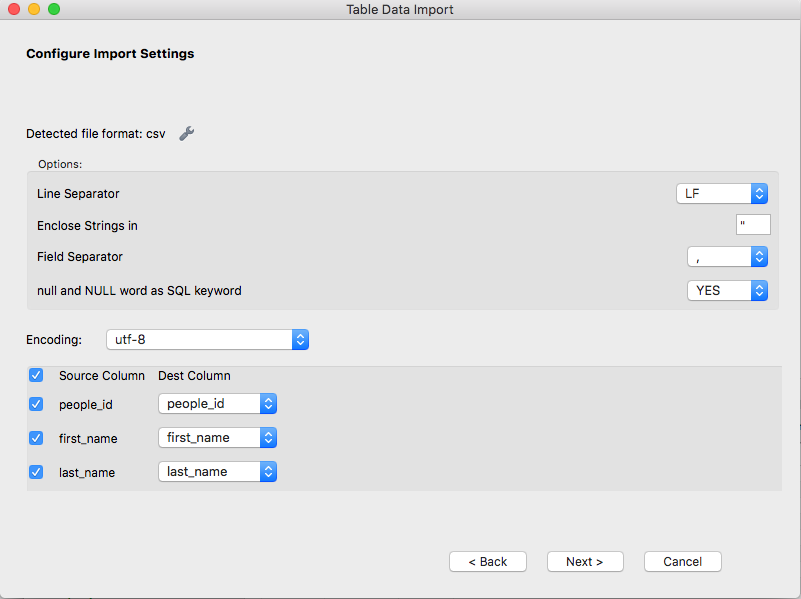
Select “Use existing table:” and select “PeptideDegradationDatabase.people”.

Macintosh HD:Users:legalllab:Desktop:Screen Shot 2019-06-24 at 2.33.39 PM.png

This will show up and choose “Use Existing One and Truncate it”.



Configure the import settings by clicking on the tool symbol. Change the “Field Separator” to the “,”.



To view the loaded table, click the lightning bolt icon next to the stored procedure titled ‘people’ that has the following SQL script.

-- select all from the people table

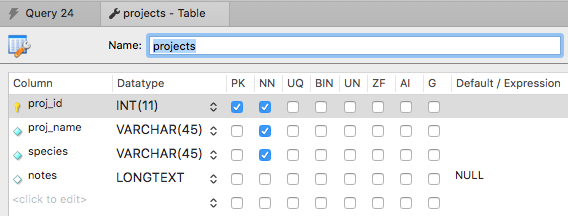
SELECT \* FROM PeptideDegradationDatabase.people;

The table should look something like the following.

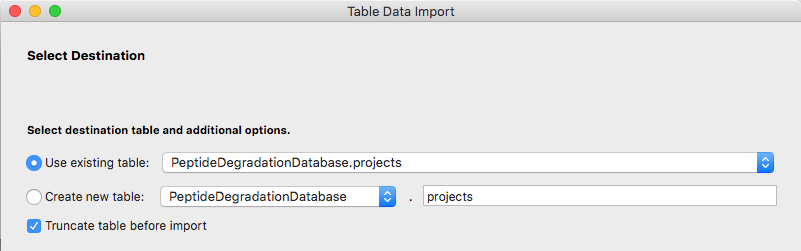


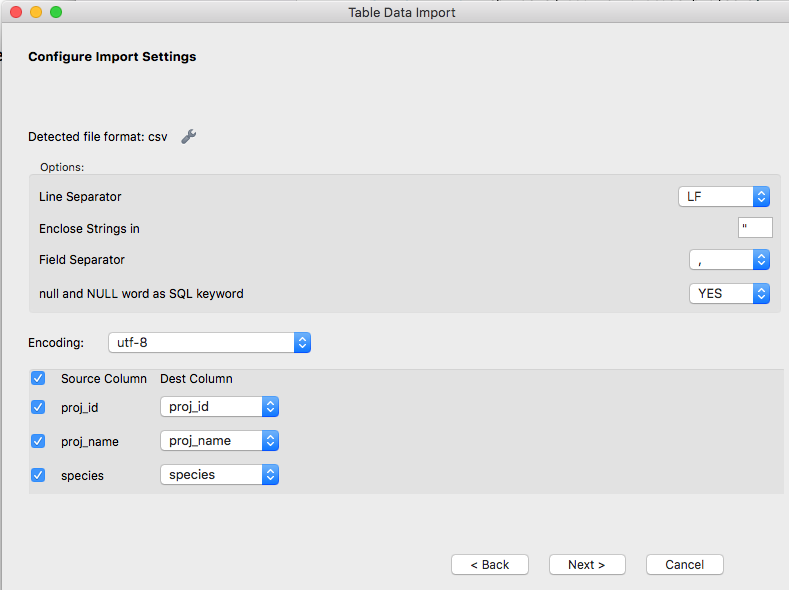
# **‘projects’ Table**

This table stores the information for the name of the project (*proj\_name*), the species of the project subject (*species*), and any additional information (*notes*).



Import the projects.csv file with the following criteria.

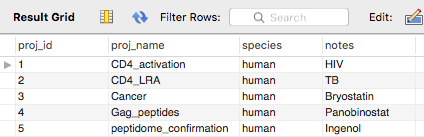


****

To view the loaded table, click the lightning bolt icon next to the stored procedure titled ‘projects’ that has the following SQL script.

-- select all from the projects table

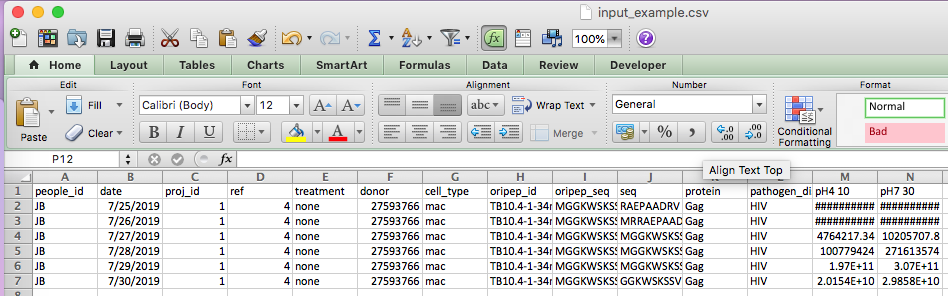
SELECT \* FROM PeptideDegradationDatabase.projects;

The table should look something like the following. 

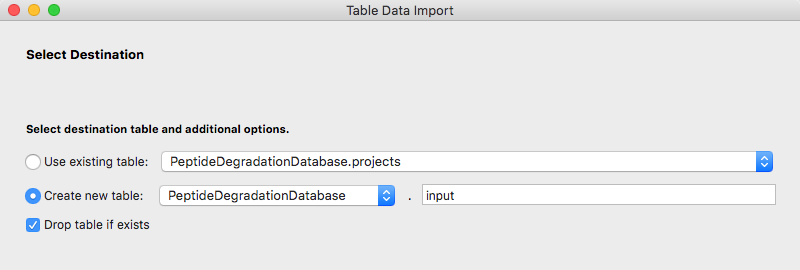
# **‘input’ Table**

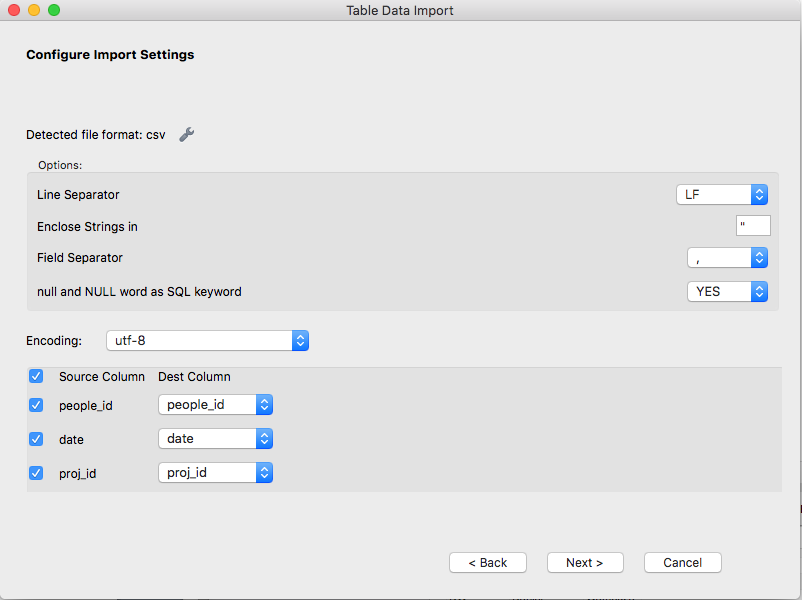
This table stores the information from the local input.csv file and provides information for the ‘experiments’, ‘experiment\_data’, and ‘original\_peptides’ tables.

Fill in the input.csv with your peptide degradation information. The *people\_id* should come from the people.csv file. The *proj\_id* and *ref* should come from projects.csv file.

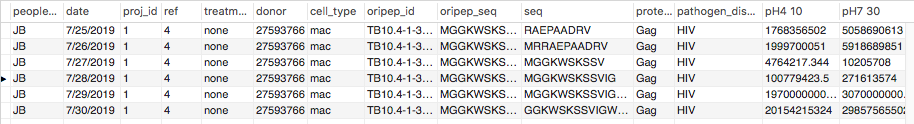


In another table, import the input.csv using the same method as above, making sure to match the following settings. Refresh the schema to view the table.

****

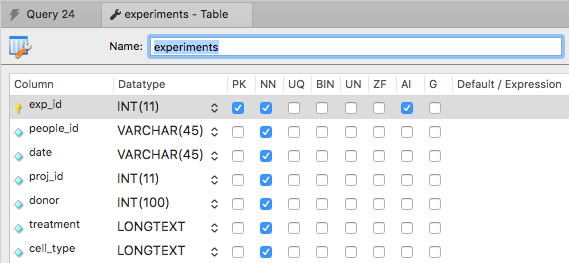
****

This is an example of an upload.

****

# **‘experiments’ Table**

This table stores the information for the person that performed the type of project it is associated with (*proj\_id*) experiment (*people\_id*), the date of the experiment (*date*), the donor identity (*donor*), the type of treatment (*treatment*), and the cell type (*cell\_type*).



First, load the input.csv to the ‘input’ table.

To view the loaded table, click the lightning bolt icon next to the stored procedure titled ‘experiments’ that has the following SQL script.

-- increment the exp\_id by 1 each time a new entry is added

ALTER TABLE

PeptideDegradationDatabase.experiments

AUTO\_INCREMENT = 1;

-- insert people\_id, date, and proj\_id from input table into the experiments table

INSERT

INTO

PeptideDegradationDatabase.experiments (people\_id, date, proj\_id, donor, treatment, cell\_type)

SELECT DISTINCT

people\_id, date, proj\_id, donor, treatment, cell\_type

FROM

PeptideDegradationDatabase.input

WHERE

people\_id IS NOT NULL AND date IS NOT NULL;

-- delete any duplicate entries if the people, date, and proj\_id, donor, treatment, and cell\_type rows match with another one

DELETE a

FROM

PeptideDegradationDatabase.experiments a

INNER JOIN PeptideDegradationDatabase.experiments a2

WHERE

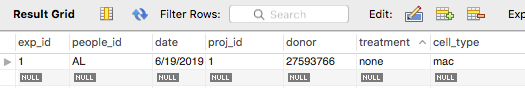
a.exp\_id>a2.exp\_id AND a.people\_id=a2.people\_id AND a.date=a2.date AND a.proj\_id=a2.proj\_id

AND a.donor=a2.donor AND a.treatment=a2.treatment AND a.cell\_type=a2.cell\_type;

-- select all from the experiments table

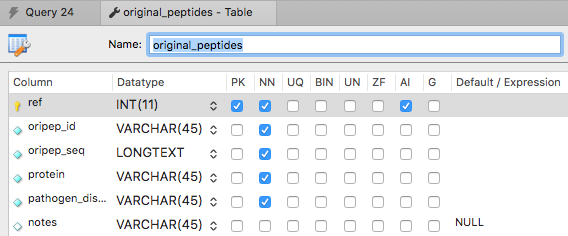
SELECT \* FROM PeptideDegradationDatabase.experiments;

The table should look something like the following.



# **‘original\_peptides’ Table**

This table stores the information for a reference to the peptide (*ref*), the peptide name (*oripep\_id*), the type of protein (*protein*), the pathogenic disease the peptide is linked to (*pathogen\_disease*), and any additional information (*notes*).



First, load the input.csv to the ‘input’ table.

To view the loaded table, click the lightning bolt icon next to the stored procedure titled ‘original\_peptides’ that has the following SQL script.

-- increment the exp\_id by 1 each time a new entry is added

ALTER TABLE PeptideDegradationDatabase.original\_peptides AUTO\_INCREMENT = 1;

-- insert peptide sequence from input table into the original\_peptides table if not already in original\_peptides table

INSERT

INTO

PeptideDegradationDatabase.original\_peptides (oripep\_id, oripep\_seq, protein, pathogen\_disease)

SELECT DISTINCT

oripep\_id, oripep\_seq, protein, pathogen\_disease

FROM

PeptideDegradationDatabase.input

WHERE

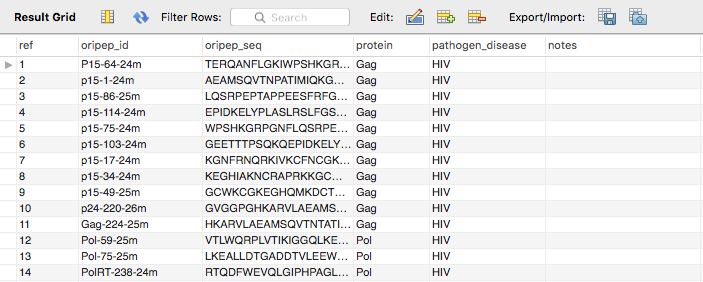
(PeptideDegradationDatabase.input.oripep\_id NOT IN (SELECT oripep\_id FROM PeptideDegradationDatabase.original\_peptides) AND

PeptideDegradationDatabase.input.oripep\_seq NOT IN (SELECT oripep\_seq FROM PeptideDegradationDatabase.original\_peptides));

-- select all from the original\_peptides table

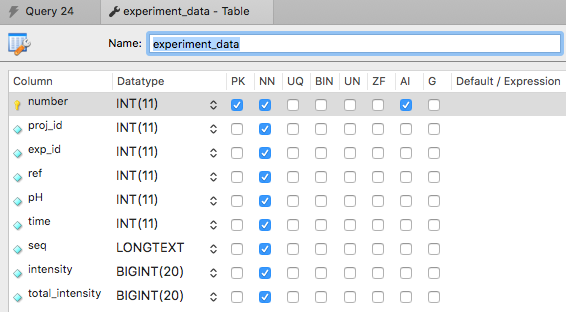
SELECT \* FROM PeptideDegradationDatabase.original\_peptides;

The table should look something like the following.



# **‘experiment\_data’ Table**

This table stores the information for the type of project from the projects.csv file (*proj\_id*), person that performed the experiment, the experiment identification from the experiments.csv file (*exp\_id*) from the pH (*pH*), time point (*time*), peptide sequence (*seq*), the reference peptide information (*ref*), the intensity (*intensity*), the and total intensity of the peptides (*total\_intensity*).



First, load the input.csv to the ‘input’ table.

Then, based on the amount of intensity columns you have in your input.csv file: copy, paste, and re-edit the script with the corresponding information for pH, time, intensity column name, and the experiment ID as well as the variable names. I have highlighted in the code what would be changed in blue.

-- increment the number by 1 each time a new entry is added

ALTER TABLE PeptideDegradationDatabase.experiment\_data AUTO\_INCREMENT = 1;

-- \*\*\*\*\*\*\*\*[COLUMN 1: USER INPUT NEEDED]\*\*\*\*\*\*\*\*\*\*\*\*

-- fill out pH, time, intensity, and exp\_id column from input table into experiment\_data table

SELECT @pH1 := '4';

SELECT @time1 := '10';

SELECT @totalint1 := SUM(`pH4 10`) FROM PeptideDegradationDatabase.input;

SELECT @exp1 := '1';

INSERT INTO

PeptideDegradationDatabase.experiment\_data (proj\_id, exp\_id, ref, pH, time, seq, intensity, total\_intensity)

SELECT

proj\_id, @exp1, ref, @pH1, @time1, seq, `pH4 10`, @totalint1

FROM

PeptideDegradationDatabase.input

WHERE

(PeptideDegradationDatabase.input.proj\_id NOT IN (SELECT proj\_id FROM PeptideDegradationDatabase.experiment\_data)) AND

(PeptideDegradationDatabase.input.ref NOT IN (SELECT ref FROM PeptideDegradationDatabase.experiment\_data)) AND

(PeptideDegradationDatabase.input.`pH4 10` NOT IN (SELECT intensity FROM PeptideDegradationDatabase.experiment\_data)) AND

(PeptideDegradationDatabase.input.seq NOT IN (SELECT seq FROM PeptideDegradationDatabase.experiment\_data))

GROUP BY

proj\_id, ref, `pH4 10`, seq;

This is a following example if the second entry were to be for an intensity column with an intensity column name of ‘pH7 30’ and the experiment ID were to be 3:

-- \*\*\*\*\*\*\*\*[COLUMN 2: USER INPUT NEEDED]\*\*\*\*\*\*\*\*\*\*\*\*

-- fill out pH, time, intensity, and exp\_id column from input table into experiment\_data table

SELECT @pH2 := '7';

SELECT @time2 := 30;

SELECT @totalint2 := SUM(`pH7 30`) FROM PeptideDegradationDatabase.input;

SELECT @exp2 := '3';

INSERT INTO

PeptideDegradationDatabase.experiment\_data (proj\_id, exp\_id, ref, pH, time, seq, intensity, total\_intensity)

SELECT

proj\_id, @exp2, ref, @pH2, @time2, seq, `pH7 30`, @totalint2

FROM

PeptideDegradationDatabase.input

WHERE

(PeptideDegradationDatabase.input.proj\_id NOT IN (SELECT proj\_id FROM PeptideDegradationDatabase.experiment\_data)) AND

(PeptideDegradationDatabase.input.ref NOT IN (SELECT ref FROM PeptideDegradationDatabase.experiment\_data)) AND

(PeptideDegradationDatabase.input.`pH7 30` NOT IN (SELECT intensity FROM PeptideDegradationDatabase.experiment\_data)) AND

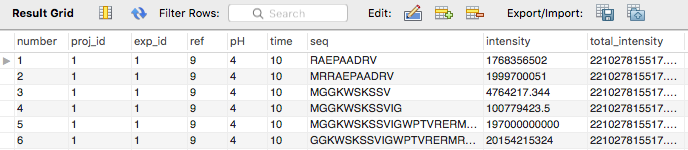
(PeptideDegradationDatabase.input.seq NOT IN (SELECT seq FROM PeptideDegradationDatabase.experiment\_data))

GROUP BY

proj\_id, ref, `pH7 30`, seq;

To increment the information into the table, copy and paste the edited SQL query into the console and run by pressing ‘command’ + ‘shift’ + ‘return’.

The table should look something like the following.



To view the loaded table, click the lightning bolt icon next to the stored procedure titled ‘experiment\_data’ that has the following SQL script.

-- select all from the experiment\_data table

SELECT \* FROM PeptideDegradationDatabase.experiment\_data;