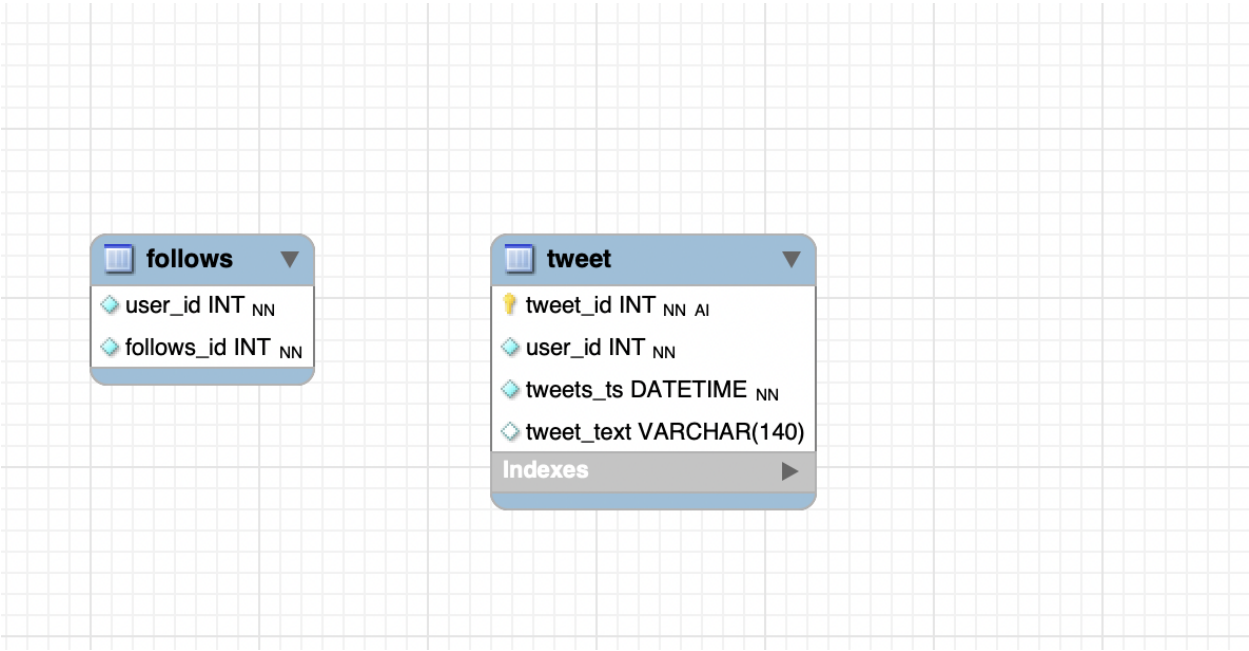
Felix Yang & Ayushi Shirke
Twitter HW1

API Method	API Calls/Sec
postTweet	3431.0
getHomeTimeline	81.3

Methodology:

First, we created an ER diagram in MySQL Workbench, which we used to Forward Engineer two tables: Follows and Tweets. We decided there should be no primary key constraints, since we realized we would just use an inner join on the shared user_id variable between both tables. We implemented this code in MySQL and python, using the docpat.py files from class to guide our organization. In our implementation of the first program, we wrote the code to insert one tweet at a time into the tweet table, and after testing the time it took to insert the entire file of tweets into the database, we tried modifying parts of the implementation in the main file, which is where the csv is read and iterated through, but we didn't find any significant changes to the total time. As a result, we left it as is since improving the write time significantly would mean changing the entire design of the implementation. Additionally, for the home timeline, we ran it and got about 2 API calls/sec, so in order to improve this run time, we decided to remove the print statement of the each tweet from inside the for loop, which calls the gethometimeline() function for randomly generated users, to outside the loop and calls it once. This ended up significantly improving the number of API calls/sec to about 81 calls per second.

ER Diagram:



Profiling Code:

From the docpat.py code provided in class, we used the DBUtils.py file that was included as a utility file to provide the functionality that initializes the mysql connection, closes it, executes a query, and inserts one or many rows in a given table.

We created a tweet_object.py file to host the Tweet class, which contains the variables that correspond to the variables defined in our Tweet Table from the MySQL database.

Additionally, we created a TwitterAPI within our tweet_api.py, which contains the functions called post_tweet, get_users, get_users_timeline, and generate_follows_from_dataframe (to programmatically insert into the follows table).

Lastly, in the driver.py file, we have a main function that authenticates the API and runs the two programs to post all the tweets from the file into the database and retrieve the hometimeline for the randomly generated user_id, respectively. This is also where the total runtime is printed for each program and the tweets are all printed for the home timeline of the randomly generated user.