Klute Analysis Report for Twitter Project

Software/hardware used:

- Server: MySQL (and MySQL Workbench)
- Storage Engine:
- Language: Python (in Pycharm)
- Database: Terminal
- OS: Mac Pro
- Processor: M1
- RAM: 16GB
- Storage Engine: InnoDB
- Prepared Statements: Yes
- Secondary Indexing: No

Set Up:

- Used environmental variables (inputted placeholders for TA)
- Set up serve connection with MySQL Workbench, specified in advanced
- OPT_LOCAL_INFILE=1 to load follows.csv into FOLLOWS table

Approach:
I set up my FOLLOWS table by reading in the csv file in my main function and using the api function I created called load_data_from_csv (which is in the tweets_mysql.py file).

I then created a dataframe of df_TWEET and looped through each row in the df and using the Tweet object and use the api function of post_tweet to insert_one through dbutils into the TWEET table in the database. This loads the 1,000,000 tweets into the table. I used a timer function to calculate the average tweets per second which came to 3779.46.

I then created api functions that get a random user, calculate the timelines per second through calling on the home_timeline function for the random user which I pulled by creating a get_random_user_ids function (in tweets_mysql.py file).

I tested it with differing amounts of users, and landed on 500 as it took a really long time to run more than that. My outputted number of timelines retrieved per second: 2.47.

I also created get followers, get_followees, get tweets to use for future implementation.

Final outputs:

```
/Users/sarahklute/anaconda3/envs/ds/bin/python
/Users/sarahklute/Documents/northeastern/year2/ds4300/hw1/tweets_tester.py
root
Data inserted successfully.
Number of tweets processed: 1000000
```

```
Time taken to process tweets: 264.59 seconds
Average Tweets per Second: 3779.46

Sample Timeline for User 4857:
  tweet_id  user_id                              tweet_text  \
0   2009927     8542  y g erem knvcgjrqnyyz v ojeyhb kpqhwex pvuw o ...
1   2008972     8542  hfty   bljb edfafj qp  n  ld   uob v euzne xoy...
2   2004054     2604  h wm axdhjnyixaslel wa  upfabe hqow qxe o rtvj...
3   2004951     6142                            yvp   iqyfb jas
4   1999029     8542  c  fbmby ase oi lcuiyybdfu h n hgeeyebujbg m e...
5   1995189     6142          yy kwr  vfhgol pfryundnprugjrs etcn  v k
6   1992183     8542  qvwhnwak e  uelsagqly qqsjwux saanymfg dyaxz  ...
7   1993150     8542                k wnbhuybsnqvvqreija xramnrhqgtfy
8   1992413     7520        n lr txwpoie cvwg  edwr bhurmvska mat jhmw
9   1990870     8542  n spskyr tbvq wy   nee cx j ar  cxcyirp  mrsli...

            tweet_ts
0 2024-01-21 18:43:53
1 2024-01-21 18:43:53
2 2024-01-21 18:43:52
3 2024-01-21 18:43:52
4 2024-01-21 18:43:51
5 2024-01-21 18:43:50
6 2024-01-21 18:43:49
7 2024-01-21 18:43:49
8 2024-01-21 18:43:49
9 2024-01-21 18:43:48

Number of timelines retrieved per second: 2.47
Time taken to retrieve timelines for 500 users: 202.66 seconds

Process finished with exit code 0
```

Possible Limitations:

- 16GB RAM on M1 might not be the most efficient processor.
- Mac OS, some libraries are optimized for Windows, Linux
- Random user calls within other function might slow down the timeline for fetching the home timeline.
- Absence of secondary indexing might be impacting query performance.
- Dependency on anaconda environment
    - I set up environmental variables.