Vivian Chen, Ryan Liang, Hiren Patel
DS 4300
February 4, 2020

DS 4300: Homework 2
Twitter Performance with Redis

	Strategy 1	Strategy 2
Time to insert followers data from csv to redis (addFollowers)	28.02 s	34.52 s
Time to insert tweet data from csv to redis (postTweet)	100.98 s	6143.46 s
Time to fetch timeline (getTimeline)	139.58 s	1.58 s

Strategy 1 has a faster write time than Strategy 2, because the user is just adding a key and a value rather than inserting the tweet into each follower's timeline. In contrast, Strategy 2's read time is faster than Strategy 1's, because the timeline is already updated with recent tweets rather than having to construct each timeline.

	MySQL	Redis Strategy 1	Redis Strategy 2
Tweets inserted per second	6000	9902.95	162.77
Timelines retrieved per second	5.42	107.47	9493.67

Strategy 1
1000000 tweets / 100.98 s = 9902.95 tweets/s
15000 timelines / 139.58 s = 107.47 timelines/s

Strategy 2
1000000 tweets / 6143.46 s = 162.77 tweets/s
15000 timelines / 1.58 s = 9493.67 timelines/s

Overall, Redis's performance is much faster than MySQL's performance. However, MySQL's write performance is faster than Redis Strategy 2's write performance. Redis is an in memory database which allows it to be very fast, but its memory depends on your RAM. SQL is not in in-memory database which reduces its speed, but allows it to store a large amount of data. Redis is preferable when we have a smaller quantity of data for quick access, and does not require a schema or complex transactions.

Contributions
1. Hiren helped the group develop code to insert data into redis and understand how redis and the code works, spearheaded the implementation of Strategy 2. We used Hiren's HW1 code as the foundation of our HW2.
2. Ryan wrote the ReadMe text, contributed to group discussions, and clarified how to go about implementing the different strategies in Redis.
3. Vivian converted tweet data into Redis using Redis and corresponding Python commands, wrote the report, analyzed the times between the 2 strategies.