Zephyr Reames-Zepeda
Corey Zhang

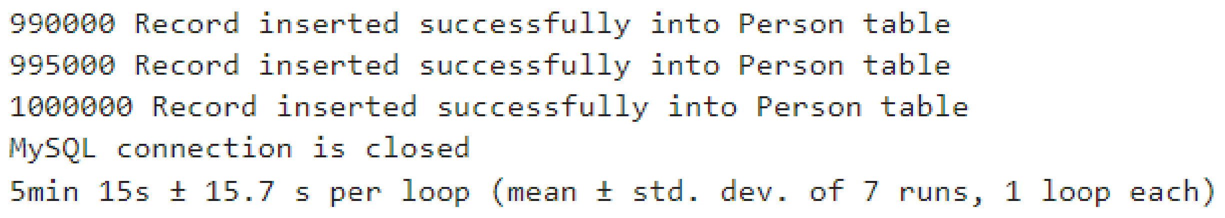
Final Project Report

Section 1: Overview

Our team investigated the write performance of the MySQL database system using Compute Engine instances and Python. We took a baseline of write time by using a python code block that inserts records into a table individually. We then measured the time of different strategies such as batched inserts, system variable tuning and upgrading hardware resources and compared them to our initial baseline.

Section 2: Baseline run (Milestone 1)

The baseline run consisted of a function that would read csvs and return the max size of each column in the csv. Using these max column lengths we create a sql database with one empty table where we use max column sizes to create the size of each cell in the table. Using a mysql connector in python we are able to write into the table by reading each row of the csv and insert it into the table we created with an execute function. We were able to insert all 1,000,000 lines into the table in about 5 minutes and 15 seconds. We noticed that this method took longer than we expected at roughly 40 minutes to complete 8 runs.



```
990000 Record inserted successfully into Person table
995000 Record inserted successfully into Person table
1000000 Record inserted successfully into Person table
MySQL connection is closed
5min 15s ± 15.7 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Section 3: Batch runs (Milestone 2)

Our team implemented batch inserts to our initial python code using the python command executemany(). We ran three different batches sizes, 5,000, 50,000, 500,000.

The times for our different batch sizes were as follows:

Batch Size	Time Results
5,000	37.4 s ± 120 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
50,000	36.3 s ± 141 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
500,000	37.7 s ± 253 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

Our team observed an immense improvement of write time. Batch inserts reduced our loop time from 5 minutes and 15 seconds to 37.4 seconds. Though we also noticed that changing the batch size of our inserts did not alter our loop time.

Zephyr Reames-Zepeda
Corey Zhang

Section 4: Hardware upgrade runs (Milestone 2)

To further explore write times we implemented two changes in our servers hardware. We ran into a technical challenge when trying to upgrade the server's hardware. We did not have a billing dedicated to our project which caused GCP to deny our quota increase for both CPU and RAM. To solve this problem our team created a new project using another account which still had educational coupon credits. The new project that we were able to upgrade our CPU and RAM has a different project id <cocopuffandzmoneyfinalattempt>. We continued to use this newly created project for the rest of our hardware upgrades(high bandwidth). Because we had to split up milestone two into two separated projects, we have submitted both to github named milestone2part1 and milestone2part2.

First, our team upgraded our server to a machine with 30 CPUs and 120GB of RAM. We then reran our batch inserts of size 500,000 using the upgraded server and recorded the time. We found that time to be about 4 seconds faster which we deemed to be an insignificant change. We anticipated a much faster result with such a big increase in CPUs.

```
33.4 s ± 326 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Afterwards we created a virtual machine based off the previous one, however this time we added a higher bandwidth. We then reran our batch inserts with size 500,000 on our high bandwidth machine and got our results. To our surprise, it actually took a longer time to finish running the code than before.

```
40.7 s ± 345 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Section 5: Potential future improvements

One different variable we can change is the zone that our notebook and VM instance is located. Additionally we were also interested in the effects of increasing the ram and cpus of the notebook itself rather than the virtual machine. These changes would not be hard to implement as we would simply need to edit the notebook's properties to test out our questions. We can test both of these potential changes by comparing them to the batch runs we did in the previous steps.