CS 327E Final Project: Milestone 1, due Thursday 05/05

1. Choose either MySQL or Postgres for your project and follow the appropriate setup guide (linked below) to deploy your environment on GCP:

   MySQL Server 8
   Postgres Server 13

2. Download the load testing dataset from Google Cloud Storage:

   ```
   gsutil cp gs://cs327e-open-access/load_testing/us-500.zip .
   gsutil cp gs://cs327e-open-access/load_testing/us-1000000.zip .
   ```

3. Unzip both files. The file `us-500.csv` contains 500 rows and is used for sampling the data and testing your code. The `us-1000000.csv` contains 1,000,000 rows of data and is what we ultimately want to load into the database. The first line in both files contains the column names. The column names are:

   - First Name
   - Last Name
   - Company
   - Address
   - City
   - County (where applicable)
   - State/Province (where applicable)
   - ZIP/Postal Code
   - Phone 1
   - Phone 2
   - Email
   - Web

4. In a new Jupyter notebook named `milestone1.ipynb`, write some Python code to sample the character lengths of the fields in the sample file (`us-500.csv`). Your code should scan the contents of the file and keep track of the max length of each column.

5. Based on the results from the previous step, write the DDL to create the table.

If you are using MySQL, include the logic to drop the database if it exists before creating it. Hint: review Project 1 if you don't remember how to do this step.

If you are using Postgres, include the logic to drop the schema and its constraints if they exist before creating them. Hint: review Project 2 if you don't remember how to do this step.

The table should use VARCHAR types for all variable-length fields and CHAR types for regular length fields (e.g. zip code, phone number, etc.).

For each VARCHAR field in the table, add 10 extra characters for padding. For example, if the email field in the sample file has 34 characters, define the email field in the table as varchar(44).

Define a Primary Key for the table using an auto-incremented field. Refer to the documentation for instructions on how to create the auto-incremented field (MySQL doc) (Postgres doc).

Follow these naming conventions when writing the DDL script:

Database (mysql)	load_testing
Schema (postgres)	load_testing
Table	Person
Columns	first_name, last_name, etc.
Primary Key	id
DDL File	create_table.sql

6. In your Jupyter notebook, install the appropriate Python connector for your chosen system.

   `pip install mysql-connector-python` (documentation)
   `pip install psycopg[binary]` (documentation)

7. In your Jupyter notebook, write the Python code that loads the contents of `us-1000000.csv` into the Person table.

   Skip the first line of the file as it contains only the column headings.

If you are using MySQL, use the connector's execute() method to insert each row. See code sample for more details. **Do not use the connector's executemany() method as this will be one of the optimizations we try in Milestone 2**.

If you are using Postgres, use the connector's execute() method to insert each row. See code sample for more details. **Do not use the connector's execute_batch() method as this will be one of the optimizations we try in Milestone 2.**

Notice the `try`, `except`, `finally` blocks in the code sample, make sure to include those blocks in your solution.

Issue a commit every 5000 rows and print the number of records successfully inserted into your table.

Use the `%%timeit` magic function to measure the total execution time of your code. This function executes the code block a total of 8 times and reports the average time across 7 runs.

Your code should produce the following output:

```
 5000 records inserted successfully into Person table
10000 records inserted successfully into Person table
15000 records inserted successfully into Person table
20000 records inserted successfully into Person table
25000 records inserted successfully into Person table
30000 records inserted successfully into Person table
35000 records inserted successfully into Person table
...
1000000 records inserted successfully into Person table
MySQL/Postgres connection is closed
4min 45s ± 19.6 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Note that your `%%timeit` results may differ from those reported here.

8. Query the Person table to get the record count. You should end up with exactly 8,000,000 records in the table after loading the `us-1000000.csv` file under the `%%timeit` function. If you have only 7,999,992 rows, you may be missing the last line of the file.

CS 327E Milestone 1 Rubric
Due Date: 05/05/22

Deployed fully functional database environment on Compute Engine instance, as per Final Project setup guide (MySQL 8 or Postgres 13)	15
Installed python database connector for MySQL 8 (`mysql-connector-python`) or Postgres 13 (`psycopg`) on Jupyter instance **-5** no connector found on Jupyter instance **-3** incorrect connector found on Jupyter instance	5
Code block that scans sample file (`us-500.csv`) and outputs max length of each field **-5** used wrong file for sampling **-5** incorrect calculation of max field length -incorrect output produced by code block: **-3** one or more fields missing from output **-3** incorrect length for one or more fields	15
Write a DDL script that creates the database/schema and table with the Primary Key. **-2** for each missing column or incorrect column name **-2** for each incorrect data type used (e.g. varchar in place of char, etc.) **-2** for each incorrect field length used (e.g. varchar(12) in place of varchar(22)) **-2** incorrect table name **-3** incorrect primary key specification (sequence generator missing or incorrect) **-2** missing or incorrect database/schema creation and drop	15
Code block that loads full file (`us-1000000.csv`) into table and measures average load time **-10** code has syntax errors or doesn't run **-10** incorrect CSV file parsing logic used **-10** incorrect insert logic (e.g. not using execute() method) **-10** incorrect or incomplete output produced (e.g. missing `%%timeit` function) **-10** incorrect commit interval **-5** missing or incorrect try/except/finally blocks	50
`milestone1.ipynb` and `create_table.sql` pushed to your group's private repo on GitHub. Your project **will not** be graded without this submission.	**Required**

`submission.json` submitted into Canvas. Your project **will not** be graded without this submission. The file should have the following schema:
 	
`{`
`  "commit-id": "your most recent commit ID from GitHub",`
`  "project-id": "your project ID from GCP"`
`}`
 	
Example:
 	
`{`
`  "commit-id": "dab96492ac7d906368ac9c7a17cb0dbd670923d9",`
	Required

`"project-id": "some-project-id"` `}`	
Total Credit:	**100**