CSCI5408 – ASSIGNMENT 2

Problem 2

Dhrumil Amish Shah (B00857606)
dh416386@dal.ca

Problem 2 – Task 1

The company **Data5408** has two remote sites vm1 and vm2. The site vm1 hosts customers and geolocations data and the site vm2 hosts remaining data.

Data cleaning and decomposing steps performed on the datasets

olist_customers_dataset.csv

- This dataset contains a list of customers and their location. Each order in olist_orders_dataset.csv is associated with a customer. Moreover, for every order that the customer makes, customer_id uniquely identifies that row in the dataset. The field customer_unique_id tells which customer placed the order. Other information includes customer_zip_code_prefix, customer_city and, customer_state.
- Below steps were performed for cleaning this dataset:
  1. This dataset contains customer_city and customer_state that are transitively dependent on customer_zip_code_prefix. Since these columns are already present in olist_geolocation_dataset.csv, it is not required to create another dataset for the same. Hence they are removed from this dataset.

olist_geolocation_dataset.csv

- This dataset contains a list of zip codes and their coordinates(i.e., latitude and longitude). Information about the geolocation available in the dataset is geolocation_zip_code_prefix, geolocation_lat, geolocation_lng, geolocation_city and, geolocation_state.
- Below steps were performed for cleaning this dataset:
  1. The downloaded dataset was not in UTF-8 encoding format. Because of this, some values in field geolocation_city were not readable (e.g., sÃ£o paulo). This dataset was transformed and loaded again in a new excel workbook (CSV UTF-8 format) using From Text/CSV excel feature. It corrected the values. (E.g., sÃ£o paulo to são paulo).
  2. This dataset is already clean and in normalized form.

olist_order_items_dataset.csv

- This dataset contains details about order items associated with every order made by the customers. The fields detailing the order are order_id, order_item_id, product_id, seller_id, shipping_limit_date, price and, freight_value. The combination of order_id and order_item_id identifies each row uniquely in the dataset. This dataset is already clean and in normalized form.

olist_order_payments_dataset.csv

- This dataset contains details about payments made for every order in the olist_orders_dataset.csv. The fields in this dataset are order_id, payment_sequential, payment_type, payment_installments and, payment_value. The order_id and payment_sequential in this dataset identify each transaction uniquely. This dataset is already clean and in normalized form.

olist_order_reviews_dataset.csv

- This dataset contains details about the reviews made by the customers. The combination of fields review_id and order_id identify each row uniquely. Other fields related to reviews are review_score, review_comment_title, review_comment_message, review_creation_date and, review_answer_timestamp.
- Below steps were performed for cleaning this dataset:

1. The downloaded dataset was not in UTF-8 encoding format. Because of this, filled values in fields review_comment_title and review_comment_message were not readable. This dataset was transformed and loaded again in a new excel workbook (CSV UTF-8 format) using From Text/CSV excel feature. It corrected the unreadable values.
2. There were 88285 and 58245 blank cells in columns review_comment_title and review_comment_message respectively. These blank cells were replaced with NULL values.

olist_orders_dataset.csv

- This dataset contains a list of orders and all the related information. The field order_id identifies each order uniquely. Other information includes order_status, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, and, order_estimated_delivery_date. Each order is associated with a unique customer_id. The field customer_id identifies every order made by the customer in olist_order_customer_dataset.csv uniquely.
- Below steps were performed for cleaning this dataset:
  1. There were 160, 1783 and, 2965 blank cells in columns order_approved_at, order_delivered_carrier_date and, order_delivered_customer_date respectively. These blank cells were replaced with NULL values.

olist_products_dataset.csv

- This dataset contains information about the products sold. The fields in the dataset are product_id, product_category_name, product_name_lenght, product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm and product_width_cm. The product_id field uniquely identifies rows in the dataset.
- Below steps were performed for cleaning this dataset:
  1. There were 610 blank cells in columns product_category_name product_name_lenght, product_description_lenght, product_photos_qty. These blank cells were replaced with NULL values.
  2. There were 2 blank cells in columns product_weight_g product_length_cm, product_height_cm, product_width_cm. These blank cells were replaced with NULL values.

olist_sellers_dataset.csv

- This dataset contains details about the sellers that completed the orders. The fields in the dataset are seller_id, seller_zip_code_prefix, seller_city, and seller_state. The field seller_id identifies each seller uniquely.
- Below steps were performed for cleaning this dataset:
  1. This dataset contains seller_city and seller_state that are transitively dependent on seller_zip_code_prefix. Since these columns are already present in olist_geolocation_dataset.csv, it is not required to create another dataset for the same. Hence they are removed from this dataset.

product_category_name_translation.csv

- This dataset contains columns product_category_name and product_category_name_english. It provides a mapping between category name in Portuguese and English. This dataset is already clean and in normalized form.

Data Distribution Logic

In order to implement Distributed Database, the given datasets can be fragmented and hosted on two sites (virtual machines in our case) as two different databases on Google Cloud Platform (GCP).

The datasets will be fragmented based on the users of the data. One virtual machine will host a database that will focus on customer-related data such as customer location for order delivery and so on. Site 1(i.e., VM1) on the Google Cloud Platform (GCP) will host the customer-related data. Similarly, another database will focus on product, order, and payment-related data. Site 2(i.e., VM2) on the Google Cloud Platform(GCP) will host the remaining datasets. However, both the databases are accessed by the users as they are accessing
a single site.

Implementing Fragmentation Transparency ensures that the site or fragment of the queried database stays hidden from the user. Fragmentation transparency will distribute the database to ease database maintenance and support load-balancing across multiple sites. It will, thus, enhance the efficiency of the database for optimal performance.

Logic to randomize the rows in excel

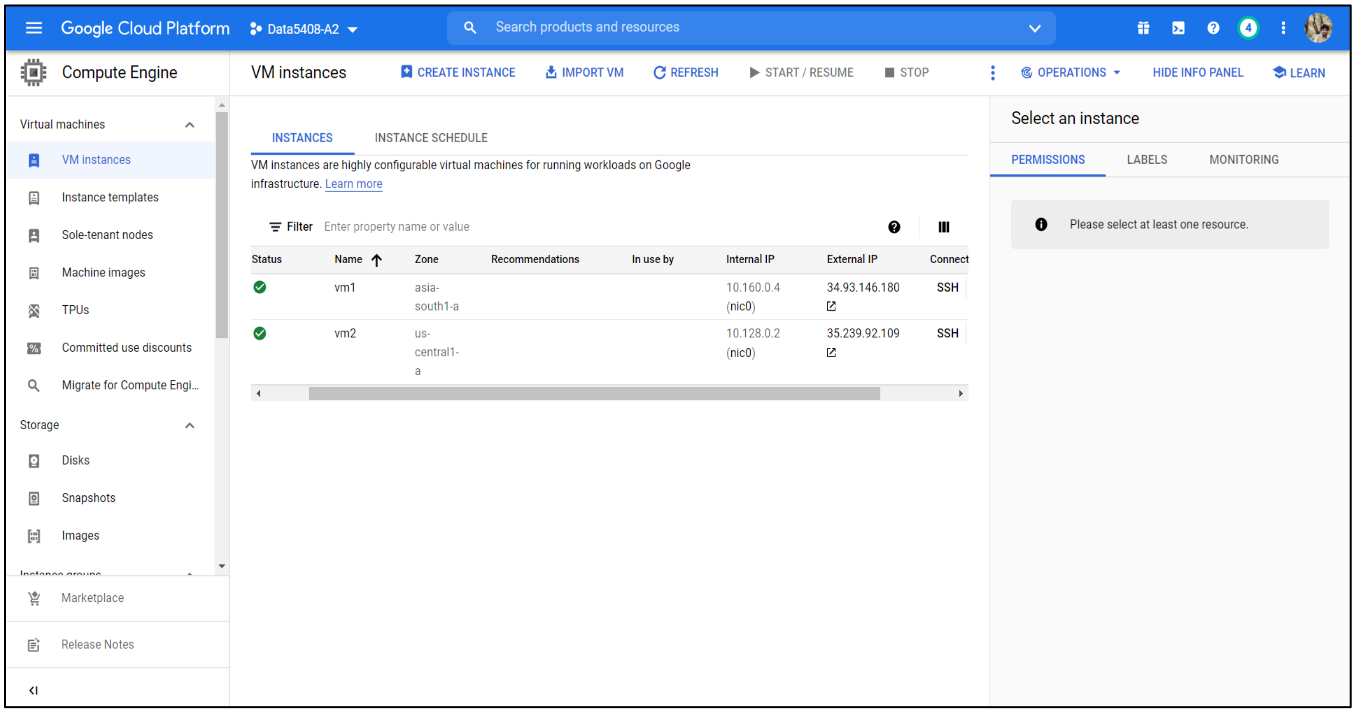
1. Add a new column within the spreadsheet and give it a column name (E.g., **random_numbers**).
2. In the first cell below the entered column name, type "=RAND()" and press "Enter". A random number will appear in that cell.
3. Take the cursor to the bottom right corner of that cell. A "+" icon will appear. Double click on that icon to fill random values in all the cells in that column.
4. Sort all the rows by column **random_numbers**. After sorting, delete column **random_numbers**.
5. Choose the first 1000 entries. Those are the random 1000 entries.

Google Cloud Platform Setup

Step 1 – Create a project named Data5408-A2 on the Google Cloud Platform and link it to a billing account.
Step 2 – Enable the Compute Engine API for the project Data5408-A2. (i.e., compute.googleapis.com)
Step 3 – Create two instances vm1 and vm2 in the project Data5408-A2.

- Instance vm1 is hosted in the asia-south1(Mumbai) region and at zone asia-south1-a.
- Instance vm2 is hosted in the us-central1(Iowa) region and at zone us-central1-a.

⇨ Figure 1 displays the created instances vm1 and vm2 under the project Data5408-A2 on the Google Cloud Platform.



VM instances are highly configurable virtual machines for running workloads on Google infrastructure. Learn more

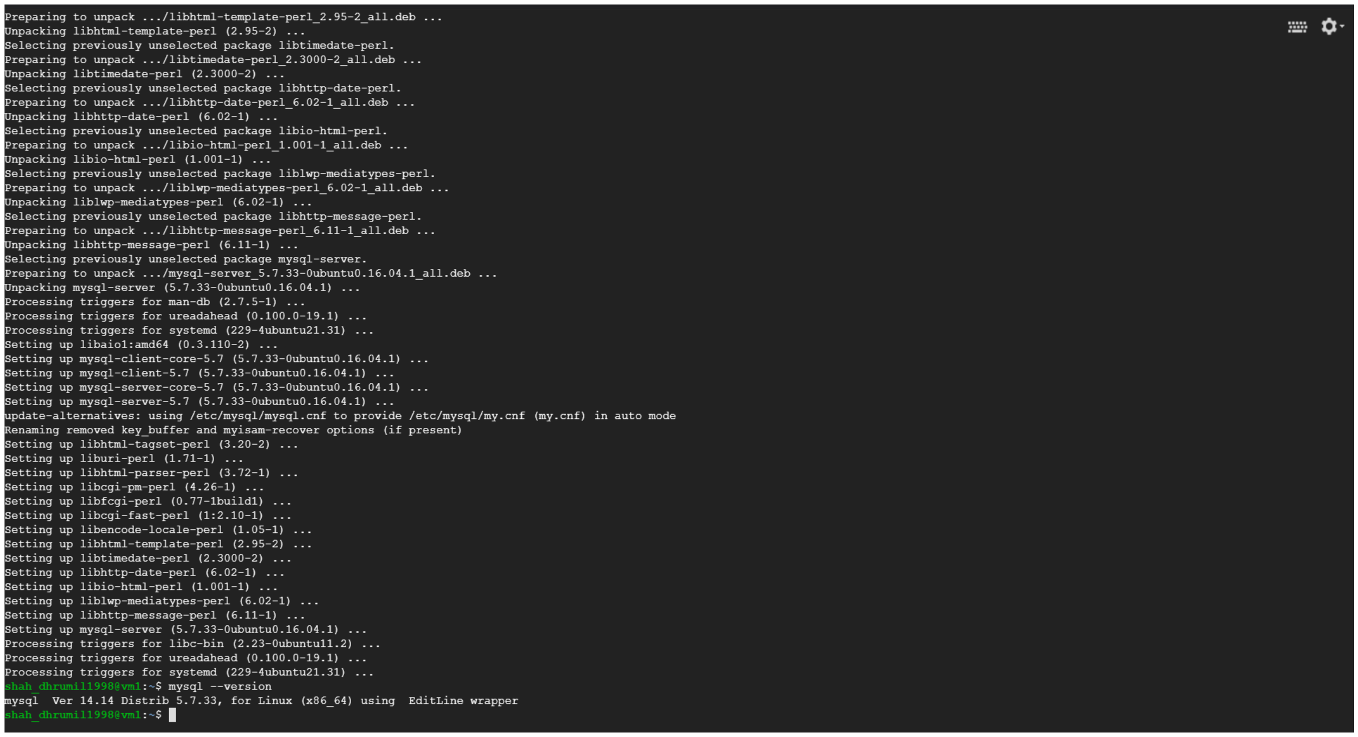
Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	vm1	asia-south1-a			10.160.0.4 (nic0)	34.93.146.180	SSH
✓	vm2	us-central1-a			10.128.0.2 (nic0)	35.239.92.109	SSH

Please select at least one resource.

Figure 1 - vm1 and vm2 virtual machines hosted on GCP

Step 4 – Connect to the vm1 by clicking “SSH” under Connect.
Step 5 – Install MySQL server using command “sudo apt-get install mysql-server” and set the password for the root user.
Step 6 – Run command “mysql --version” to check whether MySQL was installed successfully or not.

⇨ Figure 2 shows the successful installation of the MySQL server on the vm1 instance.



```
Preparing to unpack .../libhtml-template-perl_2.95-2_all.deb ...
Unpacking libhtml-template-perl (2.95-2) ...
Selecting previously unselected package libtimedate-perl.
Preparing to unpack .../libtimedate-perl_2.3000-2_all.deb ...
Unpacking libtimedate-perl (2.3000-2) ...
Selecting previously unselected package libhttp-date-perl.
Preparing to unpack .../libhttp-date-perl_6.02-1_all.deb ...
Unpacking libhttp-date-perl (6.02-1) ...
Selecting previously unselected package libio-html-perl.
Preparing to unpack .../libio-html-perl_1.001-1_all.deb ...
Unpacking libio-html-perl (1.001-1) ...
Selecting previously unselected package liblwp-mediatypes-perl.
Preparing to unpack .../liblwp-mediatypes-perl_6.02-1_all.deb ...
Unpacking liblwp-mediatypes-perl (6.02-1) ...
Selecting previously unselected package libhttp-message-perl.
Preparing to unpack .../libhttp-message-perl_6.11-1_all.deb ...
Unpacking libhttp-message-perl (6.11-1) ...
Selecting previously unselected package mysql-server.
Preparing to unpack .../mysql-server_5.7.33-0ubuntu0.16.04.1_all.deb ...
Unpacking mysql-server (5.7.33-0ubuntu0.16.04.1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for ureadahead (0.100.0-19.1) ...
Processing triggers for systemd (229-4ubuntu21.31) ...
Setting up libaio1:amd64 (0.3.110-2) ...
Setting up mysql-client-core-5.7 (5.7.33-0ubuntu0.16.04.1) ...
Setting up mysql-client-5.7 (5.7.33-0ubuntu0.16.04.1) ...
Setting up mysql-server-core-5.7 (5.7.33-0ubuntu0.16.04.1) ...
Setting up mysql-server-5.7 (5.7.33-0ubuntu0.16.04.1) ...
update-alternatives: using /etc/mysql/mysql.cnf to provide /etc/mysql/my.cnf (my.cnf) in auto mode
Renaming removed key_buffer and myisam-recover options (if present)
Setting up libhtml-tagset-perl (3.20-2) ...
Setting up liburi-perl (1.71-1) ...
Setting up libhtml-parser-perl (3.72-1) ...
Setting up libcgi-pm-perl (4.26-1) ...
Setting up libfcgi-perl (0.77-1build1) ...
Setting up libcgi-fast-perl (1:2.10-1) ...
Setting up libencode-locale-perl (1.05-1) ...
Setting up libhtml-template-perl (2.95-2) ...
Setting up libtimedate-perl (2.3000-2) ...
Setting up libhttp-date-perl (6.02-1) ...
Setting up libio-html-perl (1.001-1) ...
Setting up liblwp-mediatypes-perl (6.02-1) ...
Setting up libhttp-message-perl (6.11-1) ...
Setting up mysql-server (5.7.33-0ubuntu0.16.04.1) ...
Processing triggers for libc-bin (2.23-0ubuntu11.2) ...
Processing triggers for ureadahead (0.100.0-19.1) ...
Processing triggers for systemd (229-4ubuntu21.31) ...
shah_dhrumil1998@vm1:~$ mysql --version
mysql  Ver 14.14 Distrib 5.7.33, for Linux (x86_64) using  EditLine wrapper
shah_dhrumil1998@vm1:~$
```

Figure 2 - Successful MySQL server installation on the vm1 instance

Step 7 – Run command "sudo su" to work as the root user.
Step 8 – Run command "vi /etc/mysql/mysql.conf.d/mysqld.cnf" to open the MySQL configuration file.
Step 9 – Press "i" to edit the file. Change the "bind-address" field from 127.0.0.1 to 0.0.0.0. Press "Esc" and type ":wq<enter>" to save the file. This allows access from all across the internet.
Step 10 – Restart MySQL server using "service mysql restart".
Step 11 – Create a MySQL remote user.

- Run command "mysql -u root -p" and enter the password in the "Enter password:" field.
- Create a MySQL remote user using the below commands:
  - CREATE USER 'username'@'%' IDENTIFIED BY 'password';
  - GRANT ALL PRIVILEGES ON *.* TO 'username'@'%' WITH GRANT OPTION;
  - FLUSH PRIVILEGES;
- Type "\q<enter>" to quit the MySQL command line prompt.

Step 12 – Open 3306 port (MySQL port) on the Google Compute Engine.

- Click the "vm1" instance name from all the VM instances.
- Click the "nic0" network interface name to see the network interface details.
- On the left side, click the "Firewall" tab.
- Click the "CREATE FIREWALL RULE" button to add a new Firewall rule.
  - Add "mysql-3306-open" in the Name field.
  - Add "Firewall rule to allow access to the MySQL server on port 3306 from all across the internet." in the Description field.
  - Add "mysql-3306-open" in the Target tags field.
  - Add "0.0.0.0/0" in the Source IP ranges field.
  - Check the "tcp" field and enter the port number "3306" in it.
  - Click the "Create" button to create a new Firewall rule.

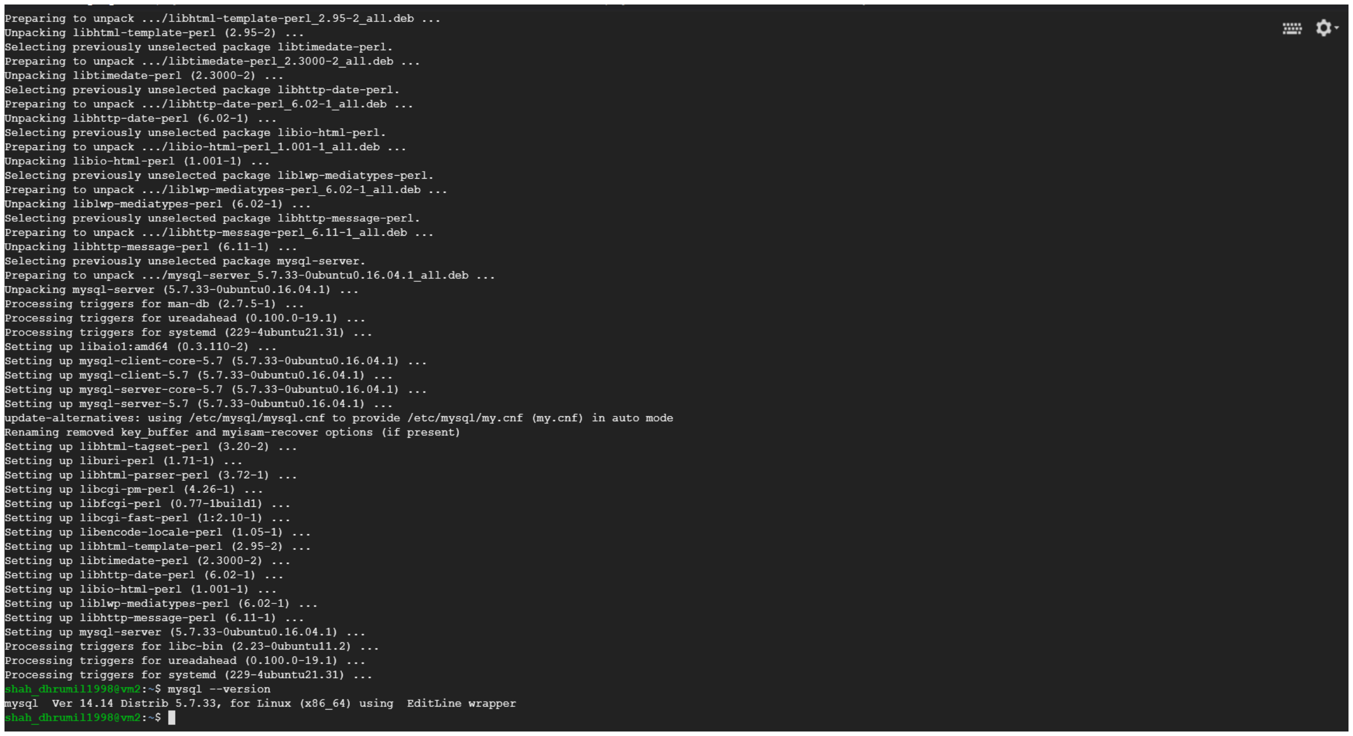
- Click the “Edit” button to edit the vm1 instance details.
- In the “Network tags” field, add the “mysql-3306-open” tag.
- Click “Save” to save the changes.
- Port 3306 is now open on vm1. It can be accessed using the External IP associated with the vm1 instance.

Step 13 – Open MySQL Workbench and create a new connection to the MySQL server installed on the vm1 instance.

- To create a connection, enter the below details:
  - Connection Name – Data5408-A2-VM1
  - Hostname – External IP associated with the vm1
  - Port – 3306
  - Username – Username entered while creating a new MySQL remote user in step 11.
  - Password – Password entered while creating a new MySQL remote user in step 11.
- Click the “Test Connection” button to test the connection.
- This successfully connects the MySQL Workbench client installed on the local machine to the MySQL server installed on the vm1 instance hosted on GCP.

Step 14 – Connect to the vm2 by clicking “SSH” under Connect.
Step 15 – Install MySQL server using command “sudo apt-get install mysql-server” and set the password for the root user.
Step 16 – Run command “mysql --version” to check whether MySQL was installed successfully or not.

⇨ Figure 3 shows the successful installation of the MySQL server on the vm2 instance.



```
Preparing to unpack .../libhtml-template-perl_2.95-2_all.deb ...
Unpacking libhtml-template-perl (2.95-2) ...
Selecting previously unselected package libtimedate-perl.
Preparing to unpack .../libtimedate-perl_2.3000-2_all.deb ...
Unpacking libtimedate-perl (2.3000-2) ...
Selecting previously unselected package libhttp-date-perl.
Preparing to unpack .../libhttp-date-perl_6.02-1_all.deb ...
Unpacking libhttp-date-perl (6.02-1) ...
Selecting previously unselected package libio-html-perl.
Preparing to unpack .../libio-html-perl_1.001-1_all.deb ...
Unpacking libio-html-perl (1.001-1) ...
Selecting previously unselected package liblwp-mediatypes-perl.
Preparing to unpack .../liblwp-mediatypes-perl_6.02-1_all.deb ...
Unpacking liblwp-mediatypes-perl (6.02-1) ...
Selecting previously unselected package libhttp-message-perl.
Preparing to unpack .../libhttp-message-perl_6.11-1_all.deb ...
Unpacking libhttp-message-perl (6.11-1) ...
Selecting previously unselected package mysql-server.
Preparing to unpack .../mysql-server_5.7.33-0ubuntu0.16.04.1_all.deb ...
Unpacking mysql-server (5.7.33-0ubuntu0.16.04.1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for ureadahead (0.100.0-19.1) ...
Processing triggers for systemd (229-4ubuntu21.31) ...
Setting up libaio1:amd64 (0.3.110-2) ...
Setting up mysql-client-core-5.7 (5.7.33-0ubuntu0.16.04.1) ...
Setting up mysql-client-5.7 (5.7.33-0ubuntu0.16.04.1) ...
Setting up mysql-server-core-5.7 (5.7.33-0ubuntu0.16.04.1) ...
Setting up mysql-server-5.7 (5.7.33-0ubuntu0.16.04.1) ...
update-alternatives: using /etc/mysql/mysql.cnf to provide /etc/mysql/my.cnf (my.cnf) in auto mode
Renaming removed key_buffer and myisam-recover options (if present)
Setting up libhtml-tagset-perl (3.20-2) ...
Setting up liburi-perl (1.71-1) ...
Setting up libhtml-parser-perl (3.72-1) ...
Setting up libcgi-pm-perl (4.26-1) ...
Setting up libfcgi-perl (0.77-1build1) ...
Setting up libcgi-fast-perl (1:2.10-1) ...
Setting up libencode-locale-perl (1.05-1) ...
Setting up libhtml-template-perl (2.95-2) ...
Setting up libtimedate-perl (2.3000-2) ...
Setting up libhttp-date-perl (6.02-1) ...
Setting up libio-html-perl (1.001-1) ...
Setting up liblwp-mediatypes-perl (6.02-1) ...
Setting up libhttp-message-perl (6.11-1) ...
Setting up mysql-server (5.7.33-0ubuntu0.16.04.1) ...
Processing triggers for libc-bin (2.23-0ubuntu11.2) ...
Processing triggers for ureadahead (0.100.0-19.1) ...
Processing triggers for systemd (229-4ubuntu21.31) ...
shah_dhrumil1998@vm2:~$ mysql --version
mysql  Ver 14.14 Distrib 5.7.33, for Linux (x86_64) using  EditLine wrapper
shah_dhrumil1998@vm2:~$ 
```

Figure 3 - Successful MySQL server installation on the vm2 instance

Step 17 – Run command “sudo su” to work as the root user.

Step 18 – Run command “vi /etc/mysql/mysql.conf.d/mysqld.cnf” to open the MySQL configuration file.
Step 19 – Pressed “i” to edit the file. Change the “bind-address” field from 127.0.0.1 to 0.0.0.0. Press “Esc” and type “:wq<enter>” to save the file. This allows access from all across the internet.
Step 20 – Restart MySQL server using “service mysql restart”.
Step 21 – Create a MySQL remote user.

- Run command “mysql -u root -p” and enter the password in the “Enter password:” field.
- Create a MySQL remote user using the below commands:
  - CREATE USER ‘username’@’%’ IDENTIFIED BY ‘password’;
  - GRANT ALL PRIVILEGES ON *.* TO ‘username’@’%’ WITH GRANT OPTION;
  - FLUSH PRIVILEGES;
- Type “\q<enter>” to quit the MySQL command line prompt.

Step 22 – Click the “vm2” instance name from all the VM instances.
Step 23 – Click the “Edit” button to edit the vm2 instance details.

- In the “Network tags” field, add the “mysql-3306-open” tag created in step 12 as a new Firewall rule.
- Click “Save” to save the changes.
- Port 3306 is now open on vm2. It can be accessed using the External IP associated with the vm2 instance.

Step 24 – Open MySQL Workbench and create a new connection to the MySQL server installed on the vm2 instance.

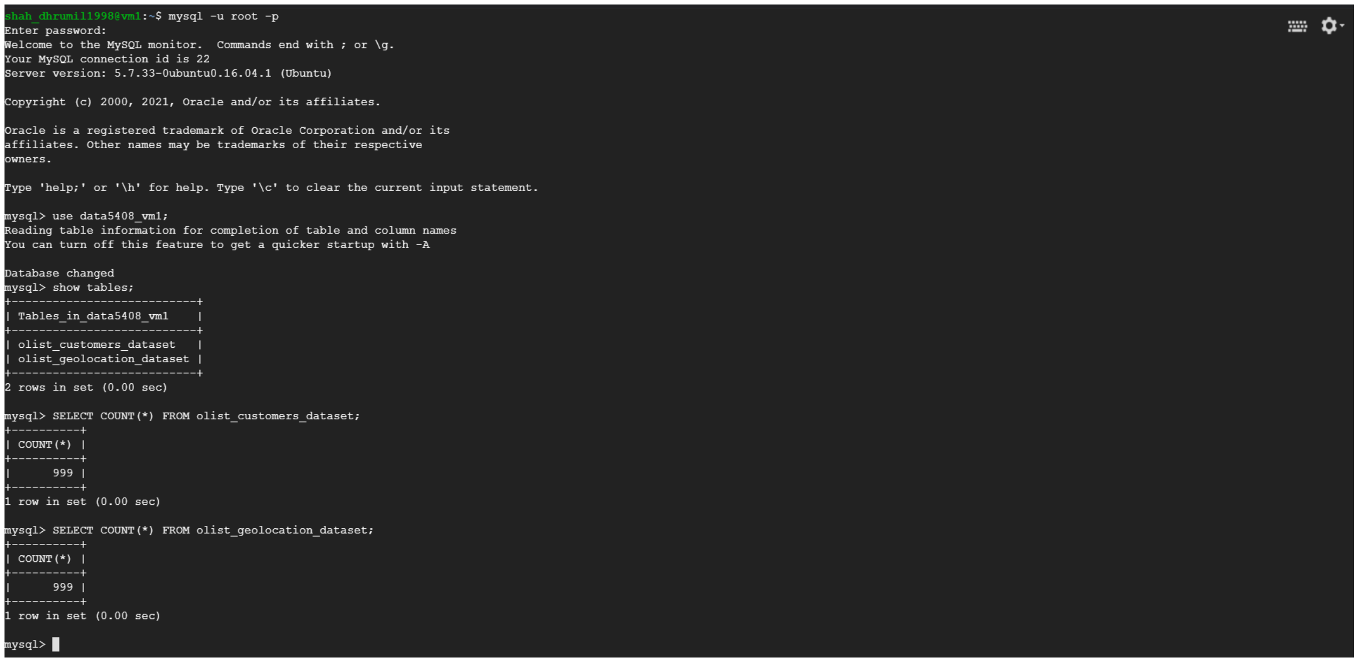
- To create a connection, enter the below details:
  - Connection Name – Data5408-A2-VM2
  - Hostname – External IP associated with the vm2
  - Port – 3306
  - Username – Username entered while creating a new MySQL remote user in step 21.
  - Password – Password entered while creating a new MySQL remote user in step 21.
- Click the “Test Connection” button to test the connection.
- This successfully connects the MySQL Workbench client installed on the local machine to the MySQL server installed on the vm2 instance hosted on GCP.

⇨ GCP is now configured with two virtual machines (i.e., vm1 in Mumbai region and vm2 in the Iowa region). MySQL server is installed on both virtual machines. Also, MySQL Workbench is connected to both the MySQL servers installed on vm1 and vm2.

Screenshots of vm1 and vm2 MySQL instances

The screenshots of virtual machines vm1 and vm2 hosted on GCP, the MySQL instances hosted on both the virtual machines and query results from the databases are given below.

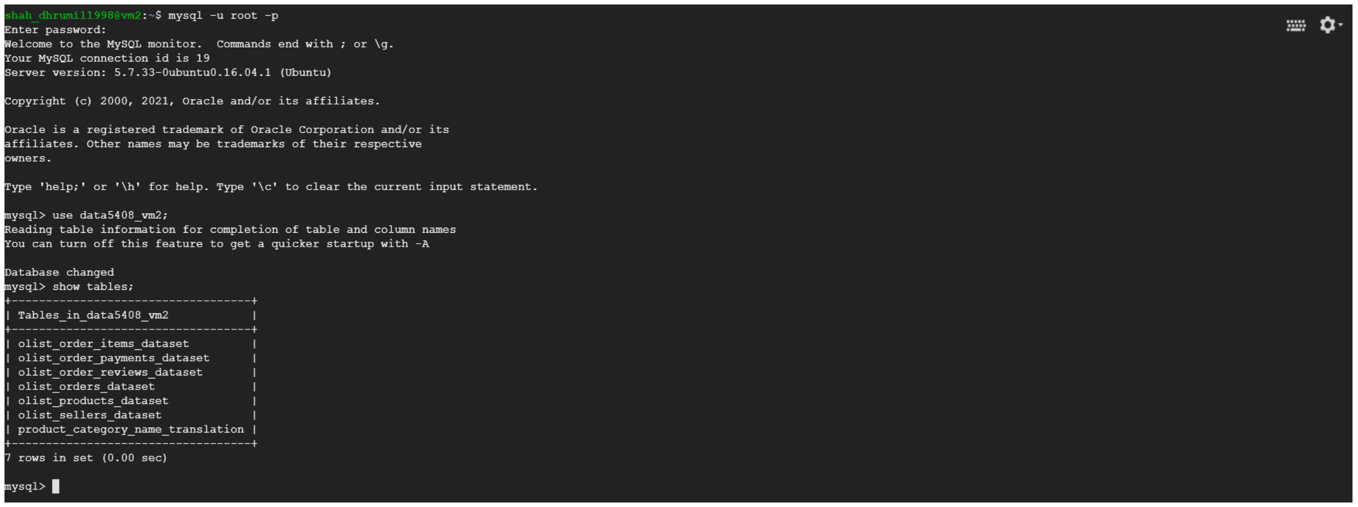
⇨ Figure 4 displays the vm1 instance and the database "data5408_vm1" hosted on the vm1 instance.



```
shah_dhrumil1998@vm1:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.7.33-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use data5408_vm1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------------+
| Tables_in_data5408_vm1    |
+---------------------------+
| olist_customers_dataset   |
| olist_geolocation_dataset |
+---------------------------+
2 rows in set (0.00 sec)

mysql> SELECT COUNT(*) FROM olist_customers_dataset;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM olist_geolocation_dataset;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)

mysql> █
```

Figure 4 - vm1 and data5408_vm1(hosted on vm1 instance)

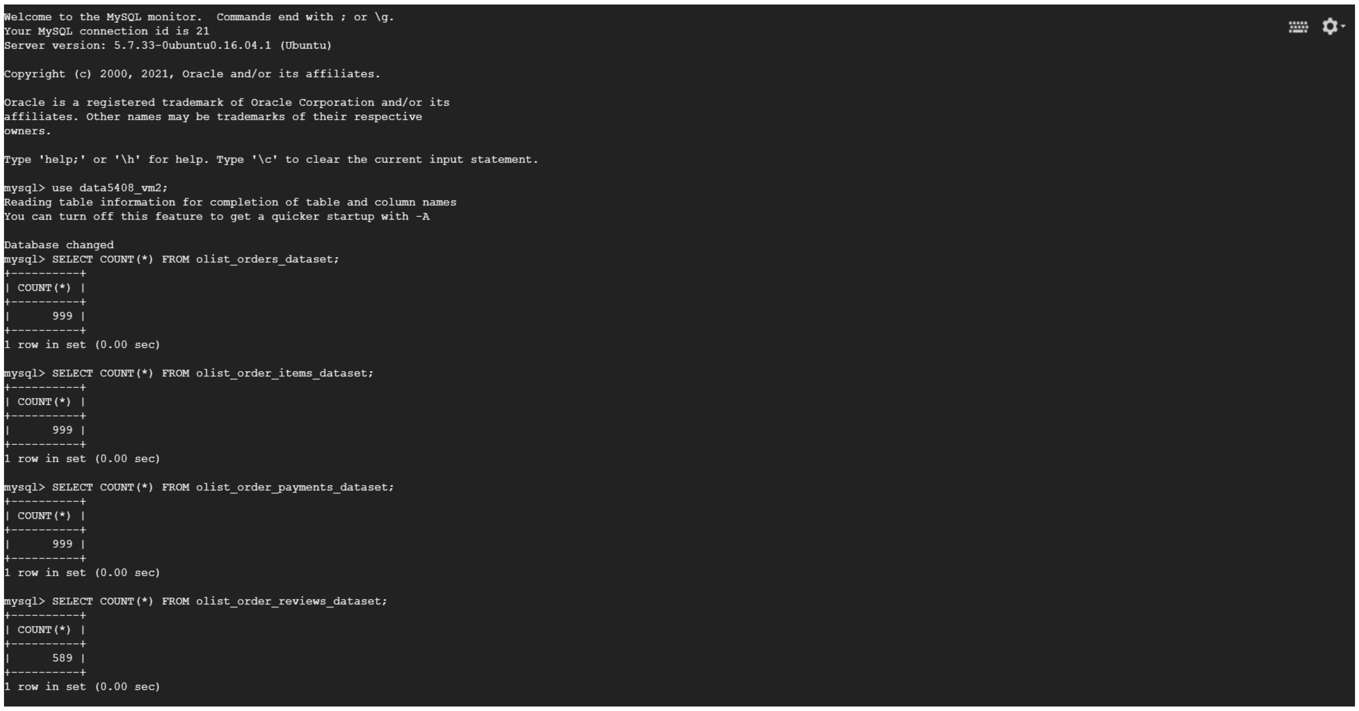
⇨ Figure 5 displays the vm2 instance and the database "data5408_vm2" hosted on the vm2 instance.



```
shah_dhrumil1998@vm2:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 5.7.33-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use data5408_vm2;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------------------------+
| Tables_in_data5408_vm2            |
+-----------------------------------+
| olist_order_items_dataset         |
| olist_order_payments_dataset      |
| olist_order_reviews_dataset       |
| olist_orders_dataset              |
| olist_products_dataset            |
| olist_sellers_dataset             |
| product_category_name_translation |
+-----------------------------------+
7 rows in set (0.00 sec)

mysql> █
```

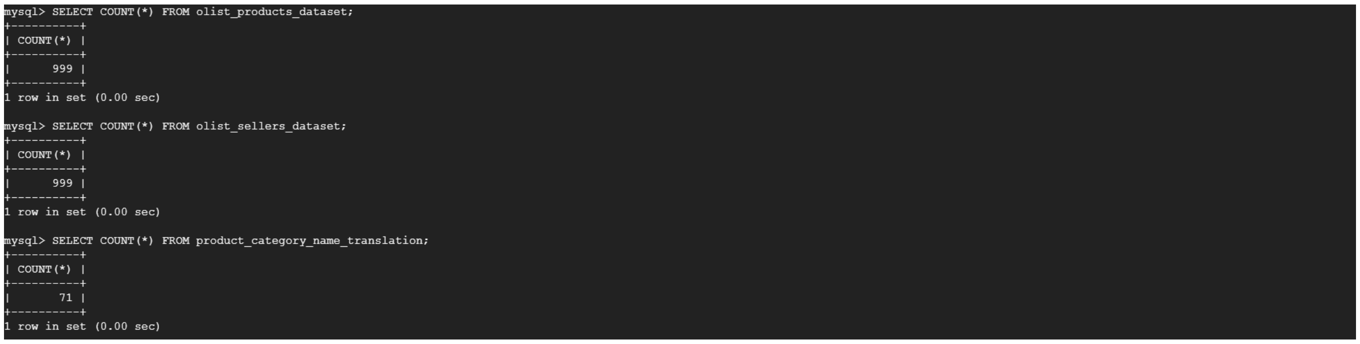
Figure 5 - vm2 and data5408_vm2(hosted on vm2 instance)

⇨ Figure 6 and 7 displays the query results fetched from tables of data5408_vm2.



```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 21
Server version: 5.7.33-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use data5408_vm2;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT COUNT(*) FROM olist_orders_dataset;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM olist_order_items_dataset;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM olist_order_payments_dataset;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM olist_order_reviews_dataset;
+----------+
| COUNT(*) |
+----------+
|      589 |
+----------+
1 row in set (0.00 sec)
```

Figure 6 - Query results fetched from tables of data5408_vm2



```
mysql> SELECT COUNT(*) FROM olist_products_dataset;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM olist_sellers_dataset;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM product_category_name_translation;
+----------+
| COUNT(*) |
+----------+
|       71 |
+----------+
1 row in set (0.00 sec)
```

Figure 7 - Query results fetched from tables of data5408_vm2

Global Data Dictionary

Global Data Dictionary (GDD) is a representation of a distributed database system. It stores the information related to the database schema like table names, column names, column attributes and, information related to databases location. The GDD created for databases **data5408_vm1** and **data5408_vm2** is **data5408_global_data_dictionary.xml**. This GDD is placed on both virtual machines so that it can be used in case either of the two virtual machines fails.

SQL Dump

The folder **problem2_task1_sql_dump_files** contains two folders:

1. data5408_vm1_datadump – It contains data dump of customers and geolocations datasets
2. data5408_vm2_datadump – It contains data dump of other remaining datasets.

Problem 2 – Task 2

Final Code is available at:

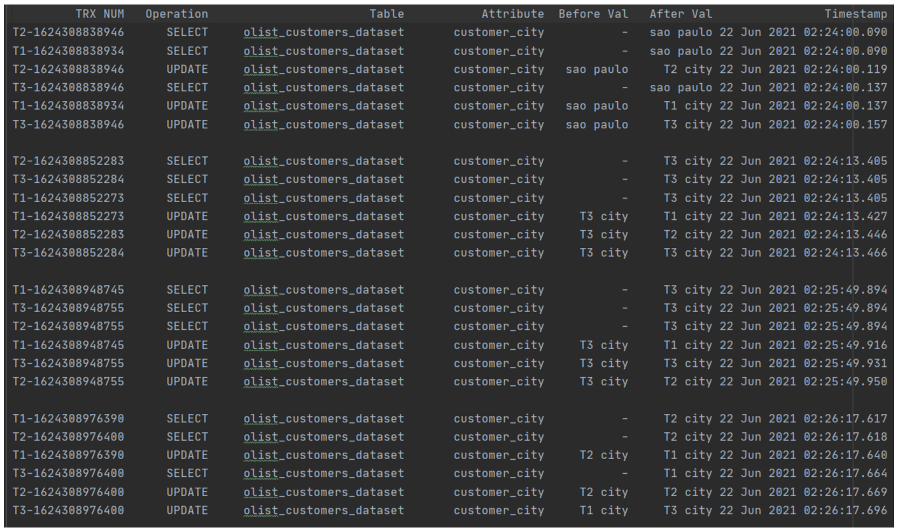
⇨ https://git.cs.dal.ca/dashah/csci-5408-s2021-b00857606-dhrumil-amish-shah/-/tree/master/A2

Before locking mechanism

Code before locking mechanism is available at:

⇨ https://git.cs.dal.ca/dashah/csci-5408-s2021-b00857606-dhrumil-amish-shah/-/tree/7b1f7dbac032283acef5e293e55e6bc4ed39d76c/A2

⇨ Figure 8 displays the logs of transactions before the exclusive locking mechanism was applied.



```
          TRX NUM  Operation                    Table       Attribute  Before Val  After Val                      Timestamp
T2-1624308838946     SELECT    olist_customers_dataset   customer_city           -  sao paulo 22 Jun 2021 02:24:00.090
T1-1624308838934     SELECT    olist_customers_dataset   customer_city           -  sao paulo 22 Jun 2021 02:24:00.090
T2-1624308838946     UPDATE    olist_customers_dataset   customer_city   sao paulo    T2 city 22 Jun 2021 02:24:00.119
T3-1624308838946     SELECT    olist_customers_dataset   customer_city           -  sao paulo 22 Jun 2021 02:24:00.137
T1-1624308838934     UPDATE    olist_customers_dataset   customer_city   sao paulo    T1 city 22 Jun 2021 02:24:00.137
T3-1624308838946     UPDATE    olist_customers_dataset   customer_city   sao paulo    T3 city 22 Jun 2021 02:24:00.157

T2-1624308852283     SELECT    olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 02:24:13.405
T3-1624308852284     SELECT    olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 02:24:13.405
T1-1624308852273     SELECT    olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 02:24:13.405
T1-1624308852273     UPDATE    olist_customers_dataset   customer_city     T3 city    T1 city 22 Jun 2021 02:24:13.427
T2-1624308852283     UPDATE    olist_customers_dataset   customer_city     T3 city    T2 city 22 Jun 2021 02:24:13.446
T3-1624308852284     UPDATE    olist_customers_dataset   customer_city     T3 city    T3 city 22 Jun 2021 02:24:13.466

T1-1624308948745     SELECT    olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 02:25:49.894
T3-1624308948755     SELECT    olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 02:25:49.894
T2-1624308948755     SELECT    olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 02:25:49.894
T1-1624308948745     UPDATE    olist_customers_dataset   customer_city     T3 city    T1 city 22 Jun 2021 02:25:49.916
T3-1624308948755     UPDATE    olist_customers_dataset   customer_city     T3 city    T3 city 22 Jun 2021 02:25:49.931
T2-1624308948755     UPDATE    olist_customers_dataset   customer_city     T3 city    T2 city 22 Jun 2021 02:25:49.950

T1-1624308976390     SELECT    olist_customers_dataset   customer_city           -    T2 city 22 Jun 2021 02:26:17.617
T2-1624308976400     SELECT    olist_customers_dataset   customer_city           -    T2 city 22 Jun 2021 02:26:17.618
T1-1624308976390     UPDATE    olist_customers_dataset   customer_city     T2 city    T1 city 22 Jun 2021 02:26:17.640
T3-1624308976400     SELECT    olist_customers_dataset   customer_city           -    T1 city 22 Jun 2021 02:26:17.664
T2-1624308976400     UPDATE    olist_customers_dataset   customer_city     T2 city    T2 city 22 Jun 2021 02:26:17.669
T3-1624308976400     UPDATE    olist_customers_dataset   customer_city     T1 city    T3 city 22 Jun 2021 02:26:17.696
```

Figure 8 - Logs of transactions executed concurrently before locking mechanism

The above figure shows the execution of the Java program four times which resulted in four transaction logs.

Observations

Before applying exclusive locks on transactions, there is no control over the execution sequence of transactions. It leaves the table under process in an inconsistent state because of interleaving transactions. For example, figure 8 shows that when the program was executed for the first time, T2 and T1 transactions read the data (i.e., customer_city field), and the value read was "sao paulo". After that, the T2 transaction updated the customer_city field from "sao paulo" to "T2 city". Even though the field customer_city was updated by T2, the value read by T3 was still "sao paulo". This is a classic example of a dirty read. This was followed by transactions T1 and T3 updating the field customer_city. Since the last update was made by the T3 transaction and the last commit was made by the T3 transaction, the final value in the table is "T3 city".

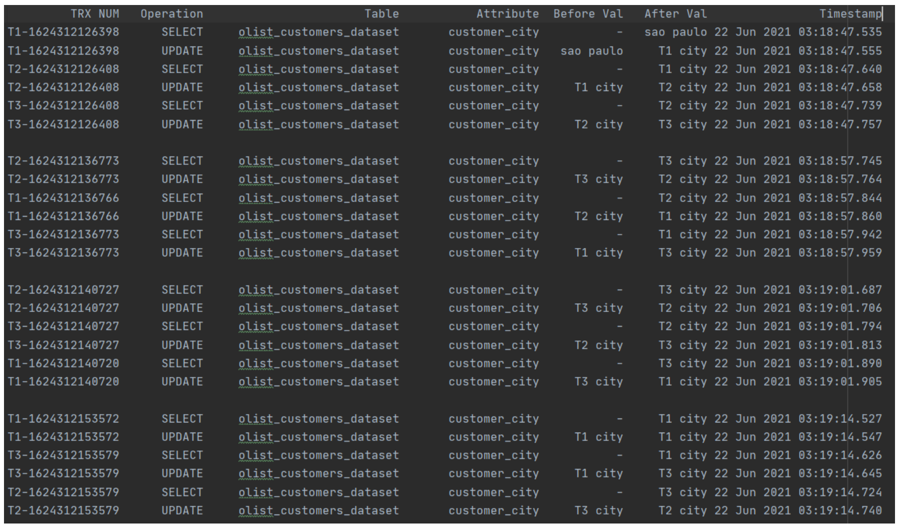
When the program was executed for the second time, transactions T2, T3, and T1 read the data from the table. Since the previous program execution had "T3 city" as the last update, the value read during the second time execution by T2, T3 and T1 is "T3 city". This was followed by transactions T1, T2 and T3 updating the customer_city field to the respective values. Since the last commit was done by T3, the lastest value of the customer_city field is "T3 city".

After locking mechanism

Code after applying exclusive locking mechanism is available at:

- ⇨ https://git.cs.dal.ca/dashah/csci-5408-s2021-b00857606-dhrumil-amish-shah/-/tree/5158e6478847f517db76eccb16578278b51a27e1/A2

- ⇨ Figure 9 displays the logs of transactions after the exclusive locking mechanism was applied.



```
        TRX NUM  Operation                     Table       Attribute  Before Val  After Val                   Timestamp
T1-1624312126398    SELECT   olist_customers_dataset   customer_city           -  sao paulo 22 Jun 2021 03:18:47.535
T1-1624312126398    UPDATE   olist_customers_dataset   customer_city   sao paulo    T1 city 22 Jun 2021 03:18:47.555
T2-1624312126408    SELECT   olist_customers_dataset   customer_city           -    T1 city 22 Jun 2021 03:18:47.640
T2-1624312126408    UPDATE   olist_customers_dataset   customer_city     T1 city    T2 city 22 Jun 2021 03:18:47.658
T3-1624312126408    SELECT   olist_customers_dataset   customer_city           -    T2 city 22 Jun 2021 03:18:47.739
T3-1624312126408    UPDATE   olist_customers_dataset   customer_city     T2 city    T3 city 22 Jun 2021 03:18:47.757

T2-1624312136773    SELECT   olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 03:18:57.745
T2-1624312136773    UPDATE   olist_customers_dataset   customer_city     T3 city    T2 city 22 Jun 2021 03:18:57.764
T1-1624312136766    SELECT   olist_customers_dataset   customer_city           -    T2 city 22 Jun 2021 03:18:57.844
T1-1624312136766    UPDATE   olist_customers_dataset   customer_city     T2 city    T1 city 22 Jun 2021 03:18:57.860
T3-1624312136773    SELECT   olist_customers_dataset   customer_city           -    T1 city 22 Jun 2021 03:18:57.942
T3-1624312136773    UPDATE   olist_customers_dataset   customer_city     T1 city    T3 city 22 Jun 2021 03:18:57.959

T2-1624312140727    SELECT   olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 03:19:01.687
T2-1624312140727    UPDATE   olist_customers_dataset   customer_city     T3 city    T2 city 22 Jun 2021 03:19:01.706
T3-1624312140727    SELECT   olist_customers_dataset   customer_city           -    T2 city 22 Jun 2021 03:19:01.794
T3-1624312140727    UPDATE   olist_customers_dataset   customer_city     T2 city    T3 city 22 Jun 2021 03:19:01.813
T1-1624312140720    SELECT   olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 03:19:01.890
T1-1624312140720    UPDATE   olist_customers_dataset   customer_city     T3 city    T1 city 22 Jun 2021 03:19:01.905

T1-1624312153572    SELECT   olist_customers_dataset   customer_city           -    T1 city 22 Jun 2021 03:19:14.527
T1-1624312153572    UPDATE   olist_customers_dataset   customer_city     T1 city    T1 city 22 Jun 2021 03:19:14.547
T3-1624312153579    SELECT   olist_customers_dataset   customer_city           -    T1 city 22 Jun 2021 03:19:14.626
T3-1624312153579    UPDATE   olist_customers_dataset   customer_city     T1 city    T3 city 22 Jun 2021 03:19:14.645
T2-1624312153579    SELECT   olist_customers_dataset   customer_city           -    T3 city 22 Jun 2021 03:19:14.724
T2-1624312153579    UPDATE   olist_customers_dataset   customer_city     T3 city    T2 city 22 Jun 2021 03:19:14.740
```

Figure 9 - Logs of transactions executed concurrently after applying exclusive locking mechanism

The above figure shows the execution of the Java program four times which resulted in four transaction logs.

Observations

I have followed a strategy of applying exclusive locks on the table end. This strategy ensures that no interleaving transactions occur. All three transactions start concurrently but, the transaction that acquires the lock first will perform all the operations and commit preventing any other transaction from intervening. I have applied a WRITE exclusive lock at the start of the transaction that blocks any other transaction from accessing the table while a SQL operation is under process on the same table. This guarantees that no dirty reads or lost updates take place. An alternative approach would be to use a synchronized keyword, however, this would lock the transaction at the client-side (i.e., it would block the object under process) and not the server at which the database is hosted. This makes the program robust for all situations.

Figure 9 shows that all four program executions follow a defined sequence where every transaction that is started is completed and committed first before the next transaction takes over. All transactions are started concurrently and, they are executed in the same order as they are received at the database end. For example, when the program was executed for the first time, the sequence received was T1, T2 and then T3. Since T1 came first, it acquired the WRITE lock, performed the READ and UPDATE operations and committed the changes and released the WRITE lock. Similarly, this was followed by T2 and T3.

From the figure, it can be observed that all four program executions followed the same strategy.

References

"Brazilian E-Commerce Public Dataset by Olist", Kaggle.com, 2021. [Online]. Available: https://www.kaggle.com/olistbr/brazilian-ecommerce [Accessed: 12-Jun-2021].