CSCI 5408: Assignment 2

Problem 2:

Task 1:

1. **olist_customers_dataset.csv**
   a. There are no Null, Blank or NaN values in this file.
   b. So, considering the top 500 rows for further operations.

2. **olist_geolocation_dataset.csv**
   a. There are no Null, Blank or NaN values in this file.
   b. So, considering the top 500 rows for further operations.

3. **olist_order_items_dataset.csv**
   a. There are no Null, Blank or NaN values in this file.
   b. So, considering the top 500 rows for further operations.

4. **olist_order_payments_dataset.csv**
   a. There are no Null, Blank or NaN values in this file.
   b. So, considering the top 500 rows for further operations.

5. **olist_order_reviews_dataset.csv**
   a. Considering the top 500 rows for further operations.
   b. Replacing all the blank values with NA under column **review_comment_title** and **review_comment_message**
   c.

6. **olist_orders_dataset.csv**
   a. Replacing all the blank values with NaN under column **order_delivered_carrier_date** and **order_delivered_customer_date**
   b. Considering the top 500 rows for further operations.

7. **olist_products_dataset.csv**
   a. Considering the top 500 rows for further operations.

b. Rectified the name of columns **product_name_lenght** and **product_description_lenght** to **product_description_length** and **product_description_length**
c. Replacing blank values in **product_description_length**, **product_name_length** and **product_photos_qty** with 0.
d. Replacing blank values in **product_category_name** with NA.

8. **olist_sellers_dataset.csv**
    a. There are no Null, Blank or NaN values in this file.
    b. So, considering the top 500 rows for further operations.

9. **product_category_name_translation.csv**
    a. There are no Null, Blank or NaN values in this file.

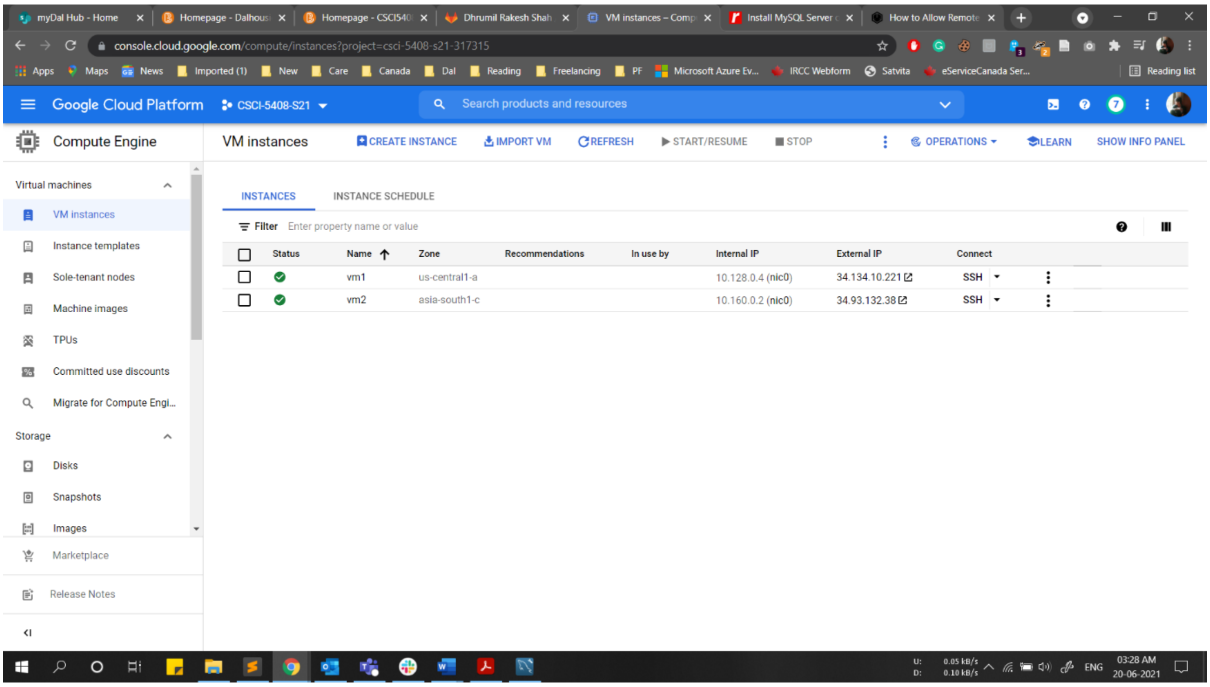
If the datasets are converted to database tables, and database(s), how will it be placed, state the reasons? (E.g. why did you consider specific Fragmentation, transparency etc.)

- Firstly, I have created two MySQL instances on vm1 and vm2 virtual machines on Google Cloud Platform.
- Vm1 is located in USA whereas Vm2 is located in India. Achieving the essence of Distributed Database System.
- The dataset is downloaded from the **Brazilian E-Commerce Public Dataset by Olist website** on Kaggle.
- There are 9 csv files containing data about **customers, geolocation, order_items, order_payments, order_reviews, orders, products, sellers, product_category_name**.
- The fragmentation is done based on multiple factors:
    - From the database tables point of view keeping the Customers data and Geolocation data together at the same location gives more meaning.
        - As there are very high chances of retrieving the customers and geolocation data simultaneously.
    - On the other hand, keeping orders, order items, order payments, order reviews, products, product categories and the respective seller’s data together at the same location.
        - Keeping the order related data with the products and their respective sellers gives more insight upon being fetched.
- Transparency in distributed database is very important, as it gives a clear logical picture about the stored data.

Global Data Dictionary (GDD):

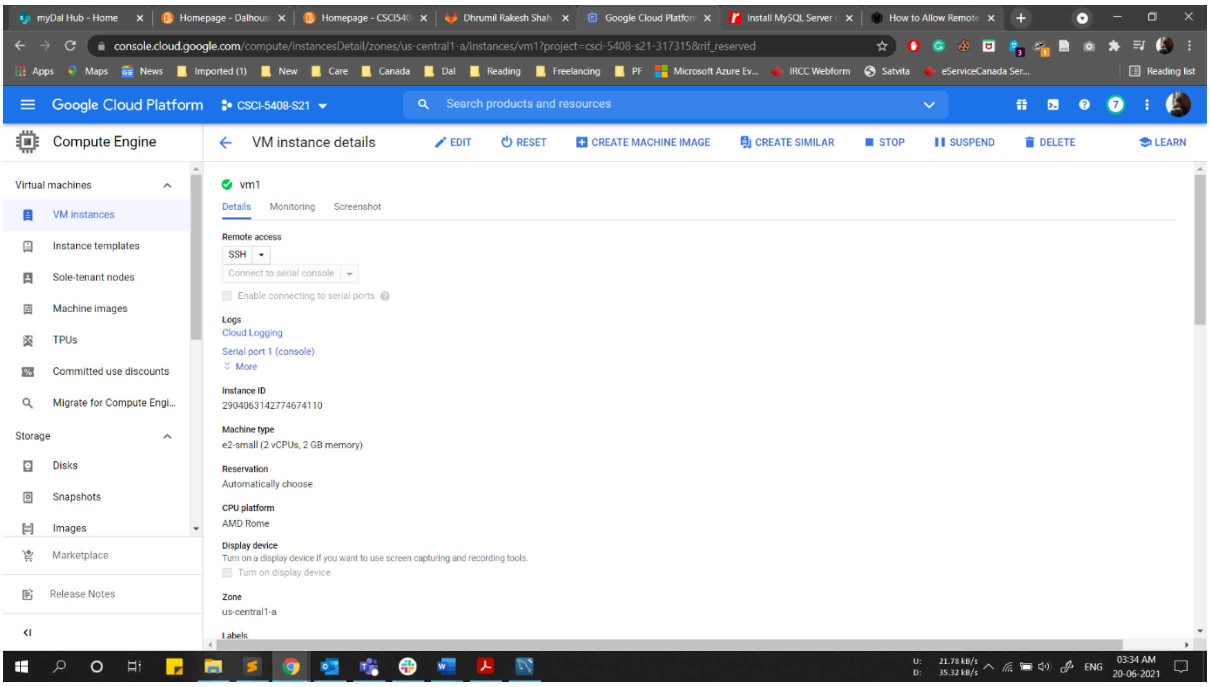
- GDD is an essential element for implementing a fully distributed DB environment.
- This file maps the client's logical DB names to the physical DB names on different servers being used.
- The GDD can be considered to be an address book used by the client.
- Using GDD the client will be able to reference the DB via logical names instead of the physical DB names.
- An GDD.xml file is created storing the details about the two databases and their respective tables stored at different locations.
- The GDD file is stored in both the VM instances.

VM Instances:



□	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
□	✓	vm1	us-central1-a			10.128.0.4 (nic0)	34.134.10.221	SSH
□	✓	vm2	asia-south1-c			10.160.0.2 (nic0)	34.93.132.38	SSH

VM1 Instance:



VM instance details

vm1

Details Monitoring Screenshot

Remote access

SSH

Connect to serial console

□ Enable connecting to serial ports

Logs

Cloud Logging

Serial port 1 (console)

More

Instance ID

2904063142774674110

Machine type

e2-small (2 vCPUs, 2 GB memory)

Reservation

Automatically choose

CPU platform

AMD Rome

Display device

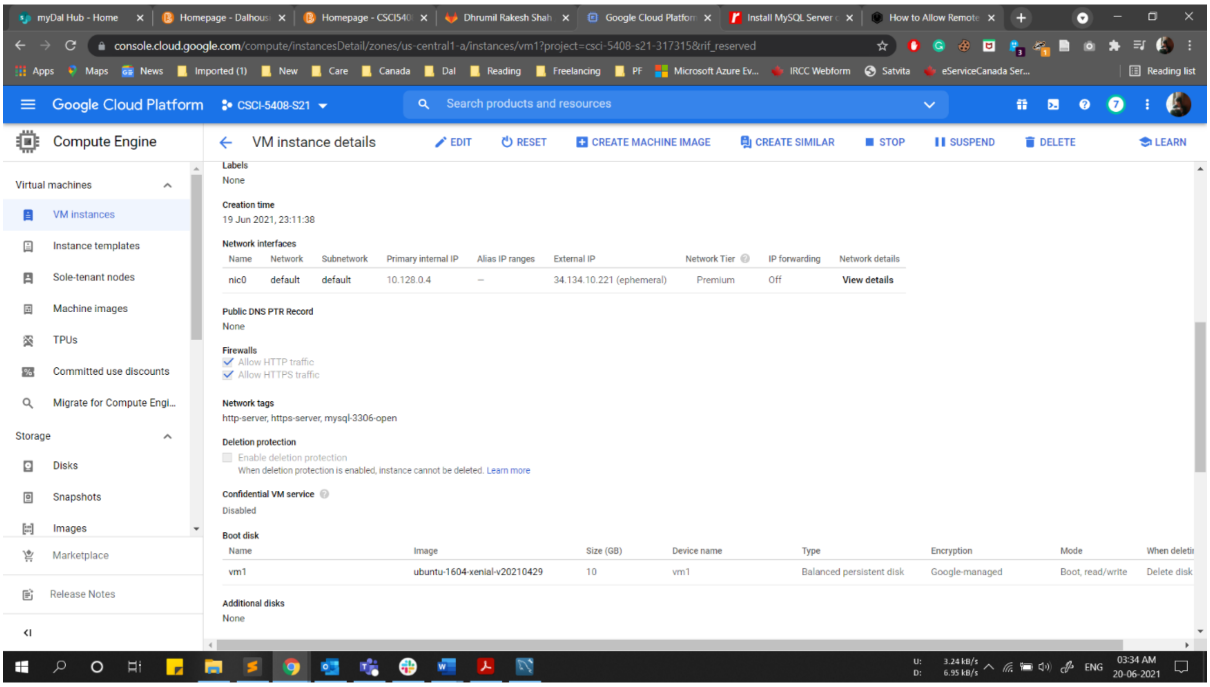
Turn on a display device if you want to use screen capturing and recording tools.

□ Turn on display device

Zone

us-central1-a

Labels



VM instance details

Labels

None

Creation time

19 Jun 2021, 23:11:38

Network interfaces

Name	Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	Network Tier	IP forwarding	Network details
nic0	default	default	10.128.0.4	–	34.134.10.221 (ephemeral)	Premium	Off	View details

Public DNS PTR Record

None

Firewalls

☑ Allow HTTP traffic

☑ Allow HTTPS traffic

Network tags

http-server, https-server, mysql-3306-open

Deletion protection

□ Enable deletion protection

When deletion protection is enabled, instance cannot be deleted. Learn more

Confidential VM service

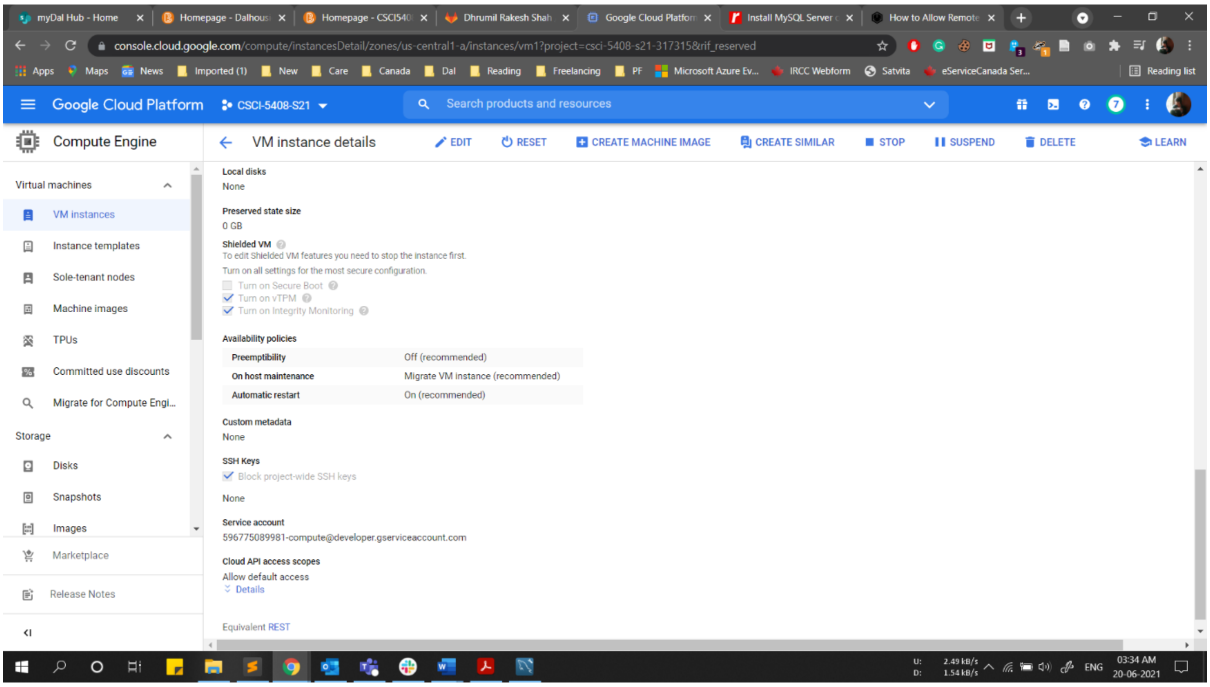
Disabled

Boot disk

Name	Image	Size (GB)	Device name	Type	Encryption	Mode	When deleti
vm1	ubuntu-1604-xenial-v20210429	10	vm1	Balanced persistent disk	Google-managed	Boot, read/write	Delete disk

Additional disks

None



Local disks
None

Preserved state size
0 GB

Shielded VM
To edit Shielded VM features you need to stop the instance first.
Turn on all settings for the most secure configuration.
☐ Turn on Secure Boot
☑ Turn on vTPM
☑ Turn on Integrity Monitoring

Availability policies

Preemptibility	Off (recommended)
On host maintenance	Migrate VM instance (recommended)
Automatic restart	On (recommended)

Custom metadata
None

SSH Keys
☑ Block project-wide SSH keys

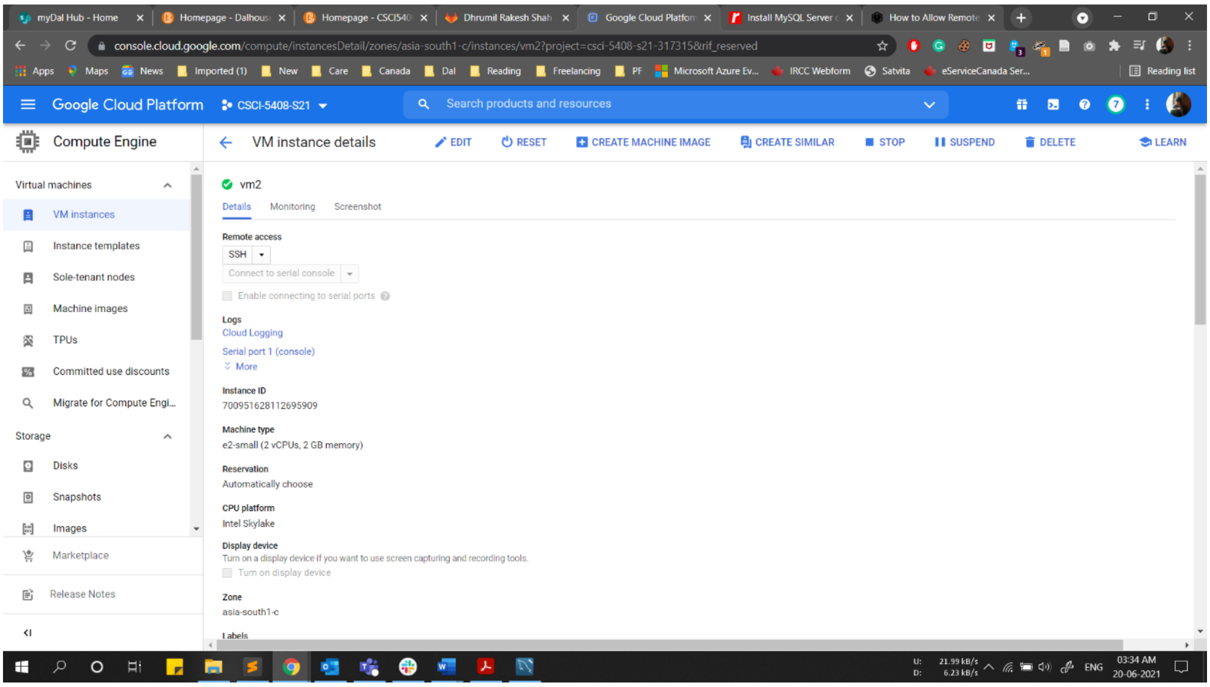
None

Service account
596775089981-compute@developer.gserviceaccount.com

Cloud API access scopes
Allow default access
Details

Equivalent REST

VM2 Instance:



vm2

Remote access
SSH
Connect to serial console
☐ Enable connecting to serial ports

Logs
Cloud Logging
Serial port 1 (console)
More

Instance ID
700951628112695909

Machine type
e2-small (2 vCPUs, 2 GB memory)

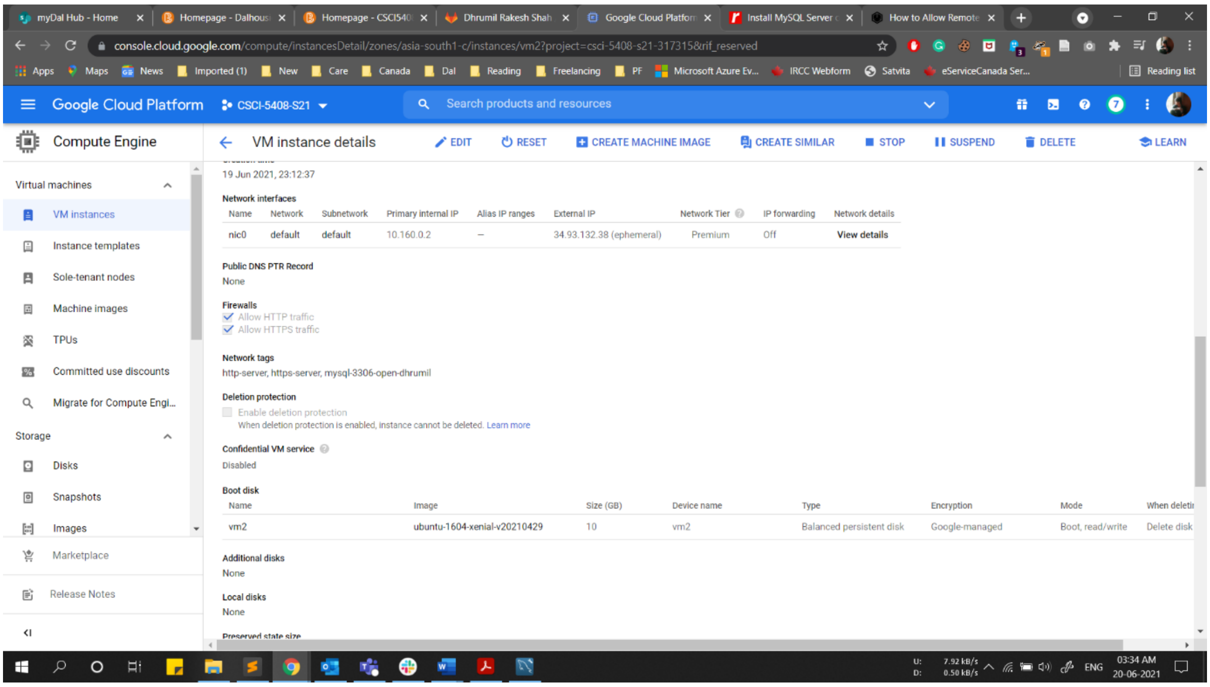
Reservation
Automatically choose

CPU platform
Intel Skylake

Display device
Turn on a display device if you want to use screen capturing and recording tools.
☐ Turn on display device

Zone
asia-south1-c

Labels



console.cloud.google.com/compute/instancesDetail/zones/asia-south1-c/instances/vm2?project=csci-5408-s21-317315&rif_reserved

19 Jun 2021, 23:12:37

Network interfaces

Name	Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	Network Tier	IP forwarding	Network details
nic0	default	default	10.160.0.2	–	34.93.132.38 (ephemeral)	Premium	Off	View details

Public DNS PTR Record
None

Firewalls
☑ Allow HTTP traffic
☑ Allow HTTPS traffic

Network tags
http-server, https-server, mysql-3306-open-dhrumil

Deletion protection
□ Enable deletion protection
When deletion protection is enabled, instance cannot be deleted. Learn more

Confidential VM service
Disabled

Boot disk

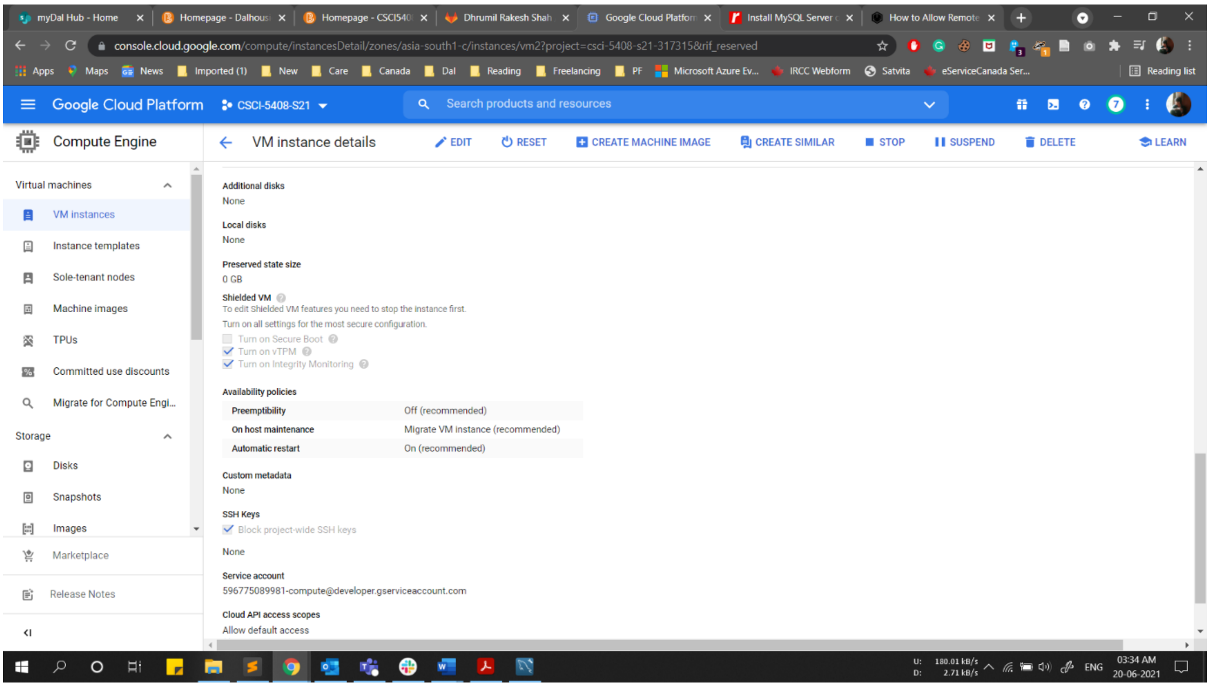
Name	Image	Size (GB)	Device name	Type	Encryption	Mode	When deleti
vm2	ubuntu-1604-xenial-v20210429	10	vm2	Balanced persistent disk	Google-managed	Boot, read/write	Delete disk

Additional disks
None

Local disks
None

Preserved state size

03:34 AM 20-06-2021



Additional disks
None

Local disks
None

Preserved state size
0 GB

Shielded VM
To edit Shielded VM features you need to stop the instance first.
Turn on all settings for the most secure configuration.
□ Turn on Secure Boot
☑ Turn on vTPM
☑ Turn on Integrity Monitoring

Availability policies

Preemptibility	Off (recommended)
On host maintenance	Migrate VM instance (recommended)
Automatic restart	On (recommended)

Custom metadata
None

SSH Keys
☑ Block project-wide SSH keys

None

Service account
596775089981-compute@developer.gserviceaccount.com

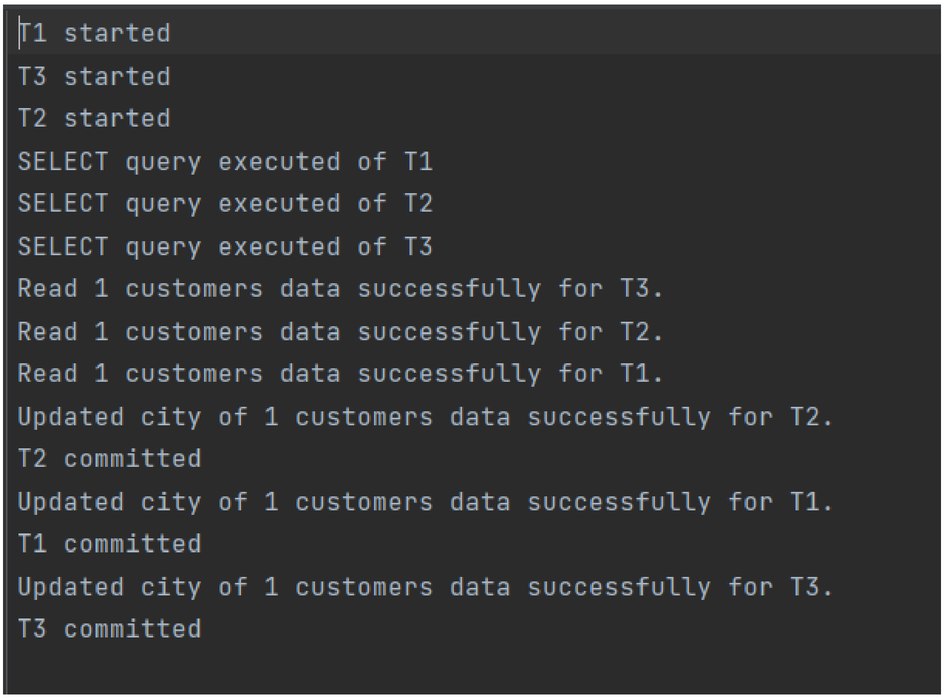
Cloud API access scopes
Allow default access

03:34 AM 20-06-2021

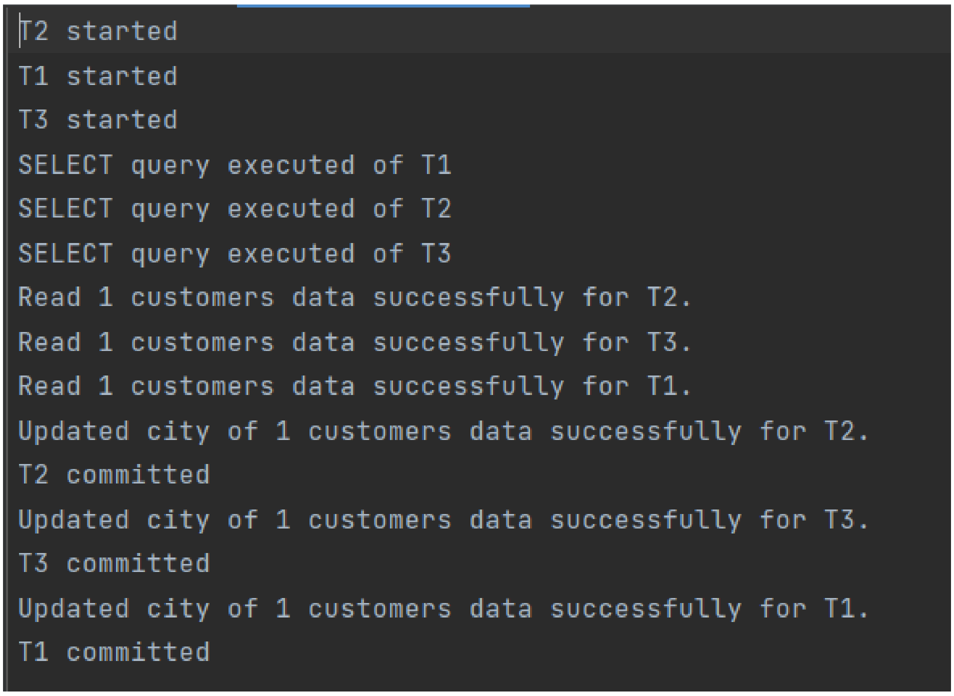
Task 2:

Before Locking:

The first part where before applying the exclusive locks on transactions T1, T2 and T3, the sequence of execution of transactions is random and we don't have any control over it. This randomness causes the table to be in an inconsistent state because of the switching between the transactions. This causes the problems like dirty read and no locking is being done to prevent such issues. Every time the transactions are run, different order is followed.



```
T1 started
T3 started
T2 started
SELECT query executed of T1
SELECT query executed of T2
SELECT query executed of T3
Read 1 customers data successfully for T3.
Read 1 customers data successfully for T2.
Read 1 customers data successfully for T1.
Updated city of 1 customers data successfully for T2.
T2 committed
Updated city of 1 customers data successfully for T1.
T1 committed
Updated city of 1 customers data successfully for T3.
T3 committed
```



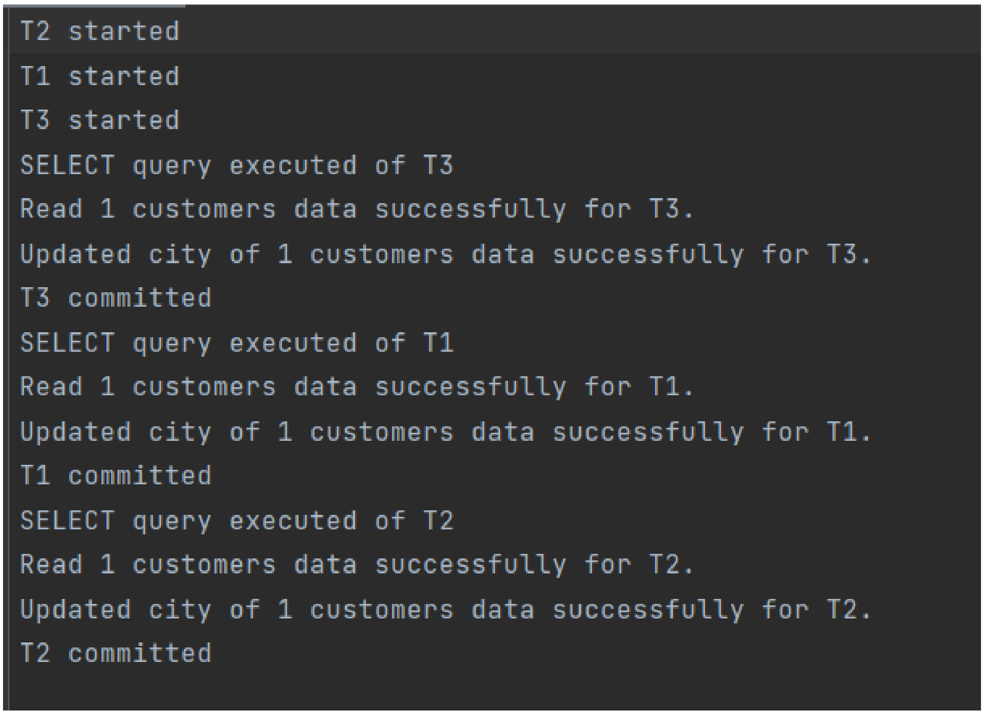
```
T2 started
T1 started
T3 started
SELECT query executed of T1
SELECT query executed of T2
SELECT query executed of T3
Read 1 customers data successfully for T2.
Read 1 customers data successfully for T3.
Read 1 customers data successfully for T1.
Updated city of 1 customers data successfully for T2.
T2 committed
Updated city of 1 customers data successfully for T3.
T3 committed
Updated city of 1 customers data successfully for T1.
T1 committed
```

The point being that, exclusive locks on transactions is very essential to order the transaction execution and avoid problems like dirty read and write causing inconsistency in the database.

<u>After Locking:</u>

After using the exclusive locks on the transactions being executed in the given sequence, we can clearly see the difference in the execution sequence, commit sequence and complete sequence. The use of exclusive lock on transactions basically grants the transactions the permission to perform their operations while preventing other transactions to gain access to the locked resource, which here in this case in the customers table.

The exclusive lock helps avoid dirty read and write problems as it locks the tables while some transaction is performing operations on that table. In the code, exclusive read and write locks have been used for the sequence of executions mentioned in the pdf. So, before using exclusive locks, any of the three transactions would commit, but after using the exclusive locks, the commits are also properly sequenced with either T3 or T1 committing before T2.



```
T2 started
T1 started
T3 started
SELECT query executed of T3
Read 1 customers data successfully for T3.
Updated city of 1 customers data successfully for T3.
T3 committed
SELECT query executed of T1
Read 1 customers data successfully for T1.
Updated city of 1 customers data successfully for T1.
T1 committed
SELECT query executed of T2
Read 1 customers data successfully for T2.
Updated city of 1 customers data successfully for T2.
T2 committed
```