

Homework 5

▼ Github Link

<https://github.com/ArkashJ/DS561-HW/tree/main/hw5>

▼ Initial Setup

1) Enable the Cloud SQL API - <https://cloud.google.com/sql/docs/mysql/connect-connectors#python>

Before you begin ↻

- Enable the Cloud SQL Admin API.

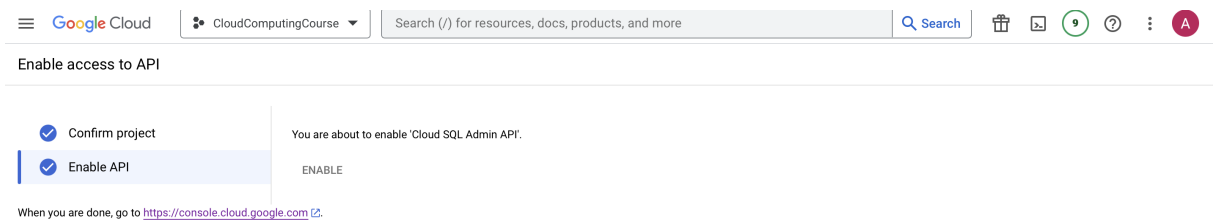
Enable the API

- Create a Cloud SQL instance, including configuring the default user.

For more information about creating instances, see [Create instances](#).

For more information about configuring the default user, see [Set the password for the default user account](#).

- Configure the [roles and permissions](#) required to connect to a Cloud SQL instance.



2) Install necessary dependencies

```
pip3 install pymysql
pip3 install sqlalchemy
pip3 install "cloud-sql-python-connector[pymysql]"
```

Activate credentials locally

```
gcloud auth application-default login
```

You are now authenticated with the gcloud CLI!

[Send feedback](#)

The authentication flow has completed successfully. You may close this window, or check out the resources below.

Information about command-line tools and client libraries



To learn more about Google Cloud CLI commands, see the [gcloud CLI guide](#).

To learn more about the command-line tools for App Engine, Compute Engine, Cloud Storage, BigQuery, Cloud SQL, and Cloud DNS (which are all bundled with the gcloud CLI), see [Accessing services with the gcloud CLI](#).

If you're a client application developer and want to find out more about accessing Google Cloud services with a programming language or framework, see [Client Libraries Explained](#).

3) Make a new service account

- Give it `cloud sql admin` and `cloud sql client` permissions

<input type="checkbox"/>		bu-561-hw5-db-ops@cloudcomputingcourse-398918.iam.gserviceaccount.com	bu-561-hw5-db-ops	Cloud SQL Admin Cloud SQL Client Logs Writer Pub/Sub Publisher Pub/Sub Subscriber Storage Admin	
--------------------------	--	---	-------------------	--	--

4) Create DB Instance

- Enable Service Networking API for Private IP

Google Cloud

CloudComputingCourse

🔍

🛒

📄

🔔


❓

⋮

A

←

Product details



Service Networking API

[Google Enterprise API](#)

Provides automatic management of network configurations

TRY THIS API [↗](#)

OVERVIEW

DOCUMENTATION

RELATED PRODUCTS

Overview

The Service Networking API provides automatic management of network configurations necessary for certain services.

Additional details

Type: [SaaS & APIs](#)

Last product update: 4/30/22

Category: [Networking](#), [Google Enterprise APIs](#)

Service name: servicenetworking.googleapis.com

Tutorials and documentation

[Learn more](#) [↗](#)

Terms of Service

←

Create a MySQL instance

Choose a preset for this edition. Presets can be customized later as needed.

Development

COMPARE EDITION PRESETS

Choose region and zonal availability

For better performance, keep your data close to the services that need it. Region is permanent, while zone can be changed any time.

Region

us-central1 (Iowa)

Zonal availability

☒ Single zone

In case of outage, no failover. Not recommended for production.

☐ Multiple zones (Highly available)

Automatic failover to another zone within your selected region. Recommended for production instances. Increases cost.

▼

SPECIFY ZONES

Customize your instance

You can also customize instance configurations later

Machine configuration

Machine has 4 vCPUs and 16 GB of memory.

Storage

Storage type is SSD. Storage size is 100 GB, and will automatically scale as needed.

Google-managed key enabled (most common)

Pricing estimate

\$0.30 per hour (estimated, without discounts)

That's about \$7.21 per day.

Basic resource costs

These items represent Cloud SQL compute, memory and storage resources only, and reflect how you configured your instance so far. Discounts not included in estimate. [Learn more](#)

Item	Hourly cost (estimate)
4 vCPU (\$0.041 per vCPU/hour)	\$0.17
16 GiB RAM (\$0.007 per GiB/hour)	\$0.11
100 GiB SSD (\$0.17 per GiB/month)	\$0.02
Total	\$0.30

Usage and traffic costs

These costs vary based on your feature usage, traffic, or data location, and aren't included in the cost estimate above. [Learn more](#)

Backups (\$0.08 per GiB)

Network egress (variable)

▲

HIDE COST BREAKDOWN

Summary

Cloud SQL Edition

Enterprise

Region

us-central1 (Iowa)

Let's update our `getconn` function to connect to our Cloud SQL instance with Private IP.

```
] : from google.cloud.sql.connector import Connector, IPTypes
import sqlalchemy

# initialize connector
connector = Connector()

# getconn now set to private IP
def getconn():
    conn = connector.connect(
        INSTANCE_CONNECTION_NAME, # ::
        "pymysql",
        user=DB_USER,
        password=DB_PASS,
        db=DB_NAME,
        ip_type=IPTypes.PRIVATE
    )
    return conn

# create connection pool
pool = sqlalchemy.create_engine(
    "mysql+pymysql://",
    creator=getconn,
)

# connect to connection pool
with pool.connect() as db_conn:
    # query database and fetch results
    results = db_conn.execute(sqlalchemy.text("SELECT * FROM ratings")).fetchall()

    # show results
    for row in results:
        print(row)

# cleanup connector
connector.close()
```

```
connector = Connector(
    ip_type=IPTypes.PRIVATE,
)

def get_connection() -> pymysql.connections.Connection:
    conn: pymysql.connections.Connection = connector.connect(
        os.environ["INSTANCE_NAME"],
        "pymysql",
        os.environ["DB_USER"],
        os.environ["DB_PASSWORD"],
        os.environ["DB_NAME"],

    )
    return conn
```

5) Once the instance is made, make a user and db

Google Cloud CloudComputingCourse Search (/) for resources, docs, products, and more

SQL Overview EDIT IMPORT EXPORT RESTART STOP DELETE CLONE

PRIMARY INSTANCE

- Overview
- System insights
- Query insights
- Connections
- Users
- Databases
- Backups
- Replicas
- Operations

Release Notes

All instances > ds561-hw5-db

ds561-hw5-db

MySQL 8.0

Creating backup. This may take a few minutes. While this operation is running, you may continue to view information about the instance.

Chart CPU utilization

1 hour 6 hours 1 day 7 days 30 days Custom

No data is available for the selected time frame.

Go to Query insights for more in-depth info on queries and performance

Connect to this instance

Public IP address

35.232.22.48

Connection name

cloudcomputingcourse-398918:us-central1:ds561-hw5-db

Configuration

vCPUs 4 Memory 16 GB SSD storage 100 GB

Uploads and CloudComputingCourse operations

Creating ds561-hw5-db 5 min 15 sec

Implementation

- Make a new bucket

```
gsutil -m cp -r .env gs://hw5-ds561/hw5
```

Google Cloud

Cloud Storage

Buckets

Monitoring

Settings

Marketplace

Bucket details

hw5-ds561

Location us-central1 (Iowa) Storage class Standard Public access Not public Protection None

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE OBSERVABILITY INVENTORY REPORTS

Buckets > hw5-ds561 > hw5

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER TRANSFER DATA MANAGE HOLDS DOWNLOAD DELETE

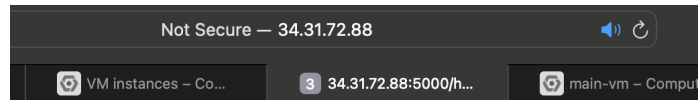
Filter by name prefix only Filter objects and folders Show deleted data

Name	Size	Type	Created	Storage class	Last modified	Public access
.env	137 B	application/octet-stream	Oct 28, 2023, 10:28:24 AM	Standard	Oct 28, 2023, 10:28:24 AM	Not public
main.py	7.4 KB	text/x-python-script	Oct 28, 2023, 10:25:33 AM	Standard	Oct 28, 2023, 10:25:33 AM	Not public
requirements.txt	1 KB	text/plain	Oct 28, 2023, 10:25:33 AM	Standard	Oct 28, 2023, 10:25:33 AM	Not public

- Make the cheapest VM as we did in HW4

- Add the same startup script as last time

- Go on the link to see the files:



acididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro

acididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro

acididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro

acididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro

acididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro

acididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro

acididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro

acididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro

http://34.31.72.88:5000/hw2-arkjain-mini-internet/mini_internet_test/110.html

Connect to your database and make 2 tables

```

DS561-HW/hw5 on 🐧 main [!+] via 🐍 v3.11.4 on ☁ arkjain@bu.edu(us-east4)
> gcloud sql connect ds561-hw5-db --user=root --quiet
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2023, 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> █

```

```

mysql> CREATE TABLE request(
->      country VARCHAR(255),
->      gender ENUM('Male', 'Female'),
->      age ENUM('0-16', '17-25', '26-35', '36-45', '46-55', '56-65', '66-75', '76+'),
->      income ENUM('0-10k', '10k-20k', '20k-40k', '40k-60k', '60k-100k', '100k-150k', '150k-250k', '250k+'),
->      is_banned BOOLEAN,
->      client_ip VARCHAR(15),
->      time_of_request TIMESTAMP
-> );
Query OK, 0 rows affected (0.32 sec)

mysql> CREATE TABLE request_time(
->      request_time TIMESTAMP,
->      requested_file VARCHAR(255),
->      error_code INT
-> );
Query OK, 0 rows affected (0.11 sec)

mysql> █

```

Initially empty

```

mysql> SELECT * FROM request;
Empty set (0.06 sec)

mysql> SELECT * FROM request_time;
Empty set (0.05 sec)

mysql> █

```

- 200 - Make a curl request


```
DS561-HW/hw5 on 主 main [!:] via 🐙 v3.11.4 (env) on ☁ arkjain@bu.edu(us-east4) took 5s
> curl -X GET -G "http://34.31.72.88:5000/hw2-arkjain-mini-internet/mini_internet_test/110.html"
```

```
Running on http://127.0.0.1:5000
Press CTRL+C to quit
hw2-arkjain-mini-internet mini_internet_test 7229.html
headers: {'Host': '127.0.0.1:8000', 'Accept-Encoding': 'identity', 'X-Country': 'South Africa', 'X-Client-IP': '240.177.216.132', 'X-Gender': 'Female', 'X-Age': '46-55', 'X-Income': '40k-60k', 'X-Time': '2023-10-28 09:00:00'}
here in if
Here trying to make a connection
here-----
Prefix mini_internet_test/7229.html blobs <google.api_core.page_iterator.HTTPIterator object at 0x106a28990>
blob name: mini_internet_test/7229.html
127.0.0.1 -- [28/Oct/2023 16:32:04] "GET hw2-arkjain-mini-internet/mini_internet_test/7229.html HTTP/1.1" 200 -
```

```
mysql> SELECT * FROM request;
+-----+-----+-----+-----+-----+-----+-----+
| country | gender | age | income | is_banned | client_ip | time_of_request |
+-----+-----+-----+-----+-----+-----+-----+
| South Africa | Female | 46-55 | 40k-60k | NULL | 240.177.216.132 | 2023-10-28 09:00:00 |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.05 sec)
```

- 501 make a bad request

```
headers: {'Host': '127.0.0.1:8000', 'User-Agent': 'curl/7.81.1', 'Accept': '*/'}
here-----
Prefix mini_internet_test/110.html blobs <google.api_core.page_iterator.HTTPIterator object at 0x106a0fad0>
blob name: mini_internet_test/110.html
127.0.0.1 -- [28/Oct/2023 16:35:25] "GET /hw2-arkjain-mini-internet/mini_internet_test/110.html HTTP/1.1" 200 -
hw2-arkjain-mini-internet mini_internet_test 110.html
Error, wrong HTTP Request Type
127.0.0.1 -- [28/Oct/2023 16:35:35] "POST /hw2-arkjain-mini-internet/mini_internet_test/110.html HTTP/1.1" 501 -
```

```
mysql> SELECT * FROM request_time
-> ;
+-----+-----+-----+
| request_time | requested_file | error_code |
+-----+-----+-----+
| 2023-10-28 09:00:00 | 7229.html | 200 |
| NULL | 110.html | 501 |
+-----+-----+-----+
2 rows in set (0.05 sec)

mysql>
```

- SIDE NOTE:
- Make a firewall rule and reserve a static IP

```

DS561-HW/hw5 on P main [!?:] via 🐉 v3.11.4 on ☁ arkjain@bu.edu(us-east4)
> gcloud compute firewall-rules create hw5-server --allow tcp:5000 --source-tags=hw5-server --source-ranges=0.0.0/0
Creating firewall...Created [https://www.googleapis.com/compute/v1/projects/cloudcomputingcourse-398918/global/firewalls/hw5-server].
Creating firewall...done.
NAME          NETWORK  DIRECTION  PRIORITY  ALLOW    DENY  DISABLED
hw5-server    default  INGRESS    1000      tcp:5000      False
DS561-HW/hw5 on P main [!?:] via 🐉 v3.11.4 on ☁ arkjain@bu.edu(us-east4) took 5s
> █

```

▼ Stress Test

- Make a new **N1 4GB** micro shared VM and upload the bash script to it
 - Its starts 2 concurrent clients and makes 50k requests to the server

```

#!/bin/bash

# Set the number of clients to run
NUM_CLIENTS=2

# Start the clients
for i in $(seq 1 $NUM_CLIENTS); do
    python3 http-client.py -d 35.208.125.55 -p 5000 -n 50000 \
    -i 9999 -b /hw2-arkjain-mini-internet -w mini_internet_test -r 137 &
done

# Wait for all the clients to finish
wait

# Print the results
echo "All clients finished running."

```

- In the VM upload the http-client as well
 - Give the bash script permissions and run it
 - `chmod u+x blow_up_client.sh`

```
mysql> select count(*) from request;
```

```
+-----+
```

```
| count(*) |
```

```
+-----+
```

```
|    68196 |
```

```
+-----+
```

```
1 row in set (0.05 sec)
```

```
mysql> select count(*) from request;
```

```
+-----+
```

```
| count(*) |
```

```
+-----+
```

```
|    72525 |
```

```
+-----+
```

```
1 row in set (0.10 sec)
```

```
mysql> select count(*) from request;
```

```
+-----+
```

```
| count(*) |
```

```
+-----+
```

```
|    72538 |
```

```
+-----+
```

```
1 row in set (0.06 sec)
```

```
mysql> select count(*) from request;
+-----+
| count(*) |
+-----+
|   102288 |
+-----+
1 row in set (0.05 sec)

mysql> select count(*) from request;
+-----+
| count(*) |
+-----+
|   102292 |
+-----+
1 row in set (0.06 sec)

mysql> select count(*) from request;
+-----+
| count(*) |
+-----+
|   102304 |
+-----+
1 row in set (0.18 sec)

mysql> █
```

- I ran the stress test a few times because of errors. It took 4 hours to run

Streaming logs...			
> i	2023-10-29 21:59:07.671	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 21:59:08.107	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 21:59:12.121	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 21:59:25.624	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 21:59:39.663	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 21:59:48.011	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 21:59:54.811	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 21:59:55.362	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:00:03.171	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:00:08.899	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:00:19.231	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:00:21.379	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:00:21.379	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 2
> i	2023-10-29 22:00:25.786	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:00:27.083	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:00:35.937	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:00:53.679	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:01:04.628	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:01:04.629	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 1
> i	2023-10-29 22:01:05.518	hw5-server	startup-script: WARNING:waitress.queue:Task queue depth is 2



- How many requests were you able to process successfully vs unsuccessfully?
- How many requests came from banned countries?
- How many requests were made by Male vs Female users?
- What were the top 5 countries sending requests to your server?
- What age group issued the most requests to your server?
- What income group issued the most requests to your server?

Connect to your db and test as follows

Kindly note that I had some additional elements in my database, probably by 1000 or so from old tests which I realized a bit late and it skewed my results by a tiny bit not much

```
gcloud sql connect ds561-hw5-db --user=root --quiet
```

1) Successful vs unsuccessful:

```
mysql> SELECT COUNT(*) AS error_count
-> FROM request_time
-> WHERE error_code <> 200;
```

```
+-----+
| error_count |
+-----+
|         4790 |
+-----+
1 row in set (0.08 sec)
```

```
mysql> █
```

```
mysql> SELECT
-> (SELECT COUNT(*) FROM request_time WHERE error_code = 200) AS successful_count,
-> (SELECT COUNT(*) FROM request_time WHERE error_code <> 200) AS unsuccessful_count,
-> (SELECT COUNT(*) FROM request_time WHERE error_code = 200) - (SELECT COUNT(*) FROM request_time WHERE error_code <> 200) AS difference;
+-----+-----+-----+
| successful_count | unsuccessful_count | difference |
+-----+-----+-----+
|          99985 |             4790 |        95195 |
+-----+-----+-----+
1 row in set (0.31 sec)

mysql> █
```

2) Banned Countries

```
SELECT COUNT(*) FROM request WHERE is_banned=TRUE;
```

```
+-----+
| error_count |
+-----+
|         4790 |
+-----+
```

3) Count Male versus Female requesters

```
mysql> SELECT
->     SUM(CASE WHEN gender = 'Male' THEN 1 ELSE 0 END) AS male_request_count,
->     SUM(CASE WHEN gender = 'Female' THEN 1 ELSE 0 END) AS female_request_count
-> FROM request;
+-----+-----+
| male_request_count | female_request_count |
+-----+-----+
|          52424    |          52403      |
+-----+-----+
1 row in set (0.17 sec)
```

4) Top 5 requesters

```
mysql>
mysql> SELECT country, COUNT(*) AS request_count
-> FROM request
-> GROUP BY country
-> ORDER BY request_count DESC
-> LIMIT 5;
+-----+-----+
| country                | request_count |
+-----+-----+
| Central African Republic |          664 |
| Kiribati                |          640 |
| Georgia                 |          634 |
| Iceland                 |          634 |
| Italy                   |          620 |
+-----+-----+
5 rows in set (0.21 sec)

mysql> █
```

5) Max requests age group

```
mysql> SELECT age, COUNT(*) AS request_count
-> FROM request
-> GROUP BY age
-> ORDER BY request_count DESC
-> LIMIT 1;
```

```
+-----+-----+
| age    | request_count |
+-----+-----+
| 36-45  |          13375 |
+-----+-----+
1 row in set (0.14 sec)
```

```
mysql> █
```

6) Max income requesters group

```
mysql> SELECT income, COUNT(*) AS request_count
-> FROM request
-> GROUP BY income
-> ORDER BY request_count DESC
-> LIMIT 1;
```

```
+-----+-----+
| income | request_count |
+-----+-----+
| 20k-40k |          13648 |
+-----+-----+
1 row in set (0.15 sec)
```

▼ Code Explanation

Detailed Comments can be found in the README with useful links

Setting up the Connection Pool

The following code has been added to my original `main.py` file such that it makes a connector with a **private IP**, as gets the connection from my google cloud account, including the DB information.

The `make_connection_pool()` function makes a sqlalchemy engine and calls the `get_connection()` function to establish a database connection.

```
import pymysql
import sqlalchemy

connector = Connector(
    "cloudcomputingcourse-398918:us-central1:cloudcomputingcourse",
    "pymysql",
    ip_type=IPTypes.PRIVATE,
)
app = Flask(__name__)

def get_connection() -> pymysql.connections.Connection:
    conn: pymysql.connections.Connection = connector.connect(
        os.environ["INSTANCE_NAME"],
        "pymysql",
        os.environ["DB_USER"],
        os.environ["DB_PASSWORD"],
        os.environ["DB_NAME"],
    )
    return conn

def make_connection_pool():
    pool = sqlalchemy.create_engine(
        "mysql+pymysql://",
        creator=lambda: get_connection(),
        pool_size=5,
        max_overflow=2,
        pool_timeout=30,
        pool_recycle=1800,
    )
    return pool
```

Modifying Flask App

I modified the initial flask route I had to store the country name, gender, income, age, time and client ip in a dictionary which I send as an argument to the `make_countries_mysql_table` function - a function to make 2 2-NF supported tables.

The first part simply checks if the tables exists or not and makes them and the second part indexes the `data_from_headers` dictionary element to store the value.

```
def make_countries_mysql_table(data_from_headers: dict,
                              data_from_request: dict):

    pool = make_connection_pool()

    with pool.connect() as conn:
        query = """
            CREATE TABLE IF NOT EXISTS Users (
                id INT AUTO_INCREMENT PRIMARY KEY,
                country VARCHAR(255) NOT NULL,
                client_id INT NOT NULL,
                gender_req ENUM('Male', 'Female'),
                age ENUM('0-16', '17-25', '26-35', '36-45', '46-55', '56-65', '66-75', '76+'),
                income_req ENUM('0-10k', '10k-20k', '20k-40k', '40k-60k', '60k-100k', '100k-150k', '150k-250k', '250k+'),
                is_banned BOOLEAN NOT NULL,
            )
        """
        conn.execute(query)

    with pool.connect() as conn:
        query = """
            CREATE TABLE IF NOT EXISTS Requests (
                id INT AUTO_INCREMENT PRIMARY KEY,
```

```

        time TIMESTAMP NOT NULL,
        file_requested VARCHAR(255) NOT NULL,
    )
    """
    conn.execute(query)
    with pool.connect() as connection:
        query = f"""
            INSERT INTO Users(country, client_id,
gender, income, age, is_banned)
            VALUES (
                '{data_from_headers["country"]}',
                '{data_from_headers["client_id"]}',
                '{data_from_headers["gender"]}',
                '{data_from_headers["income"]}',
                '{data_from_headers["age"]}',
                '{data_from_headers["population"]}',
                '{int(data_from_headers["is_banned"])[1]})'
            )
        """
    connection.execute(query)
    with pool.connect() as connection:
        query = f"""
            INSERT INTO Requests(time, file_requested)
            VALUES (
                '{data_from_request["time"]}',
                '{data_from_request["file_requested"]}'
            )
        """
    connection.execute(query)

```

▼ Resources

Useful links

- https://github.com/GoogleCloudPlatform/cloud-sql-python-connector/blob/main/samples/notebooks/mysql_python_connector.ipynb
- <https://dev.mysql.com/downloads/mysql/>
- https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE_PROTOCOL.html#protocol_overview
- <https://cloud.google.com/sql/docs/mysql/language-connectors>
- https://docs.google.com/presentation/d/1u2EA9-9X_dNn8RBdBivoCuuxwGpdQV7YCcH6SCIZCgY/edit#slide=id.g258da33442f_0_141
- https://docs.google.com/presentation/d/120uEEuhQNR984r-iGnGD5GELDZVFRm3RGC1N0gXTNrA/edit#slide=id.g27446ef6af8_0_70
- Information on cloud connectors - <https://cloud.google.com/sql/docs/mysql/connect-connectors#python>
- Github showing how to make a pool connection - <https://github.com/GoogleCloudPlatform/cloud-sql-python-connector#how-to-use-this-connector>
- Private IP info - https://cloud.google.com/sql/docs/mysql/configure-private-ip?_ga=2.112557828.-1858195586.1698198634#gcloud_1
- Making a DB on a cloud instance - <https://support.google.com/appsheets/answer/10107301?hl=en>

Useful commands

Get the status of your VM

```
sudo journalctl -u google-startup-scripts.service -f
```

Find and Kill a busy server

```
lsof -i :8080  
kill -9 $PID
```
