HW5 - David Euijoon Kim - Cloud SQL

https://github.com/dk-davidekim/Google-Cloud-Computing.git

1. Create Cloud SQL Instance

Instance info

Instance ID *
hw5
Use lowercase letters, numbers, and hyphens. Start with a letter.

Password *
dk98 GENERATE
Set a password for the root user. Learn more

No password

PASSWORD POLICY

Database version *
MySQL 8.0

SHOW MINOR VERSIONS

Pricing estimate

$0.30 per hour (estimated, without discounts)

That's about $7.21 per day.

Feature usage and traffic costs aren't included in estimate

SHOW COST BREAKDOWN

Summary

Cloud SQL Edition	Enterprise
Region	us-east1 (South Carolina)
DB Version	MySQL 8.0
vCPUs	4 vCPU
Memory	16 GB
Data Cache	Disabled
Storage	100 GB
Connections	Public IP
Backup	Automated
Availability	Single zone
Point-in-time recovery	Enabled
Network throughput (MB/s)	1,000 of 1,000
Disk throughput (MB/s)	Read: 48.0 of 240.0 Write: 48.0 of 240.0
IOPS	Read: 3,000 of 15,000 Write: 3,000 of 15,000

Choose a Cloud SQL edition

A Cloud SQL edition determines foundational characteristics of your instance and cannot be changed later. Choose based on your price and performance needs. Learn more

Enterprise Plus
- 99.99% availability SLA for eligible instances
- High-performance machines, up to 128 vCPUs
- Up to 35 days point-in-time recovery
- Data cache (optional)

Enterprise
- 99.95% availability SLA for eligible instances
- General purpose machines, up to 96 vCPUs
- Up to 7 days point-in-time recovery

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-east1 (South Carolina)

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.

2. Google Cloud Shell

- sudo apt-get update
- sudo apt-get install mysql-client -y

3. Cloud Shell

- gcloud sql connect hw5 --user=root

```
dk98@cloudshell:~ (ds-561)$ gcloud sql connect hw5 --user=root
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 76
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>
```

4. Create a new database

- CREATE DATABASE hw5db;
- USE hw5db;

```
mysql> CREATE DATABASE hw5db;
Query OK, 1 row affected (0.01 sec)

mysql> USE hw5db;
Database changed
```

5. Create a new table for requests (2NF)

```
CREATE TABLE Clients (
    client_id INT AUTO_INCREMENT PRIMARY KEY,
    client_ip VARCHAR(255) UNIQUE NOT NULL,
    country VARCHAR(255),
    gender VARCHAR(255),
    age VARCHAR(255),
    income VARCHAR(255),
    is_banned BOOLEAN
);

CREATE TABLE Files (
    file_id INT AUTO_INCREMENT PRIMARY KEY,
    file_name VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE Requests (
    id INT AUTO_INCREMENT PRIMARY KEY,
    client_id INT,
    file_id INT,
    time_of_day TIME,
    FOREIGN KEY (client_id) REFERENCES Clients(client_id),
    FOREIGN KEY (file_id) REFERENCES Files(file_id)
);
```

```
mysql> CREATE TABLE Clients (
    ->     client_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     client_ip VARCHAR(255) UNIQUE NOT NULL,
    ->     country VARCHAR(255),
    ->     gender VARCHAR(255),
    ->     age VARCHAR(255),
    ->     income VARCHAR(255),
    ->     is_banned BOOLEAN
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Files (
    ->     file_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     file_name VARCHAR(255) UNIQUE NOT NULL
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Requests (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     client_id INT,
    ->     file_id INT,
    ->     time_of_day TIME,
    ->     FOREIGN KEY (client_id) REFERENCES Clients(client_id),
    ->     FOREIGN KEY (file_id) REFERENCES Files(file_id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

6. Create a new table for failed requests

```
CREATE TABLE ErrorCodes (
    error_code_id INT AUTO_INCREMENT PRIMARY KEY,
    error_code INT UNIQUE NOT NULL,
    description TEXT
);

CREATE TABLE Failed_Requests (
    id INT AUTO_INCREMENT PRIMARY KEY,
    time_of_request TIME,
    file_id INT,
    error_code_id INT,
    FOREIGN KEY (file_id) REFERENCES Files(file_id),
    FOREIGN KEY (error_code_id) REFERENCES ErrorCodes(error_code_id)
);
```

```
mysql> CREATE TABLE ErrorCodes (
    ->     error_code_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     error_code INT UNIQUE NOT NULL,
    ->     description TEXT
    -> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> CREATE TABLE Failed_Requests (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     time_of_request TIME,
    ->     file_id INT,
    ->     error_code_id INT,
    ->     FOREIGN KEY (file_id) REFERENCES Files(file_id),
    ->     FOREIGN KEY (error_code_id) REFERENCES ErrorCodes(error_code_id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

7. Check Tables (Optional)

```
mysql> SHOW TABLES
    -> ;
+-----------------+
| Tables_in_hw5db |
+-----------------+
| Clients         |
| ErrorCodes      |
| Failed_Requests |
| Files           |
| Requests        |
+-----------------+
5 rows in set (0.00 sec)
```

8. Edit Code - Implemented Object-Oriented Programming

appone.py - DatabaseConnector Class

```python
from flask import Flask, request
from google.cloud import storage, logging, pubsub_v1
from google.cloud.sql.connector import Connector
import os
from dotenv import load_dotenv
from datetime import datetime
import pymysql
import sqlalchemy

app = Flask(__name__)

load_dotenv()

class DatabaseConnector:
    def __init__(self):
        db_connection_string = os.getenv('DB_CONNECTION_STRING')
        db_user = os.getenv('DB_USER')
        db_password = os.getenv('DB_PASSWORD')
        db_database = os.getenv('DB_DATABASE')

        self.connector = Connector()

        def getconn():
            return self.connector.connect(
                db_connection_string,
                "pymysql",
                user=db_user,
                password=db_password,
                db=db_database
            )

        self.pool = sqlalchemy.create_engine(
            "mysql+pymysql://",
            creator=getconn
        )
```

appone.py - Logger, PubSub

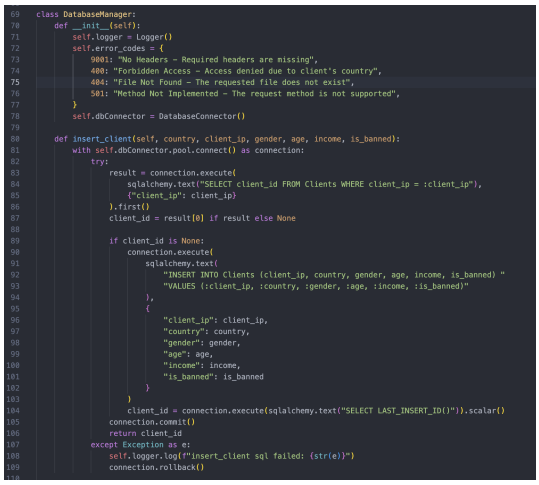
```python
class Logger:
    def __init__(self):
        project_id = 'ds-561'
        logger_name = 'hw5'

        self.logging_client = logging.Client(project=project_id)
        self.logger = self.logging_client.logger(logger_name)

    def log(self, message):
        self.logger.log_text(message)


class PubSub:
    def __init__(self):
        project_id = 'ds-561'
        topic_name = 'hw3'

        self.pub_client = pubsub_v1.PublisherClient()
        self.topic_path = self.pub_client.topic_path(project_id, topic_name)

        self.logger = Logger()

    def publish(self, message):
        try:
            data = message.encode('utf-8')
            future = self.pub_client.publish(self.topic_path, data)
            message_id = future.result()
            self.logger.log(f"Message published with ID: {message_id}")
        except Exception as e:
            self.logger.log(f"PubSub Notification Failed: {str(e)}")
```

appone.py - main function - from hw4

```python
# Main Application Service
class AppService:
    BANNED_COUNTRIES = ["North Korea", "Iran", "Cuba", "Myanmar",
                        "Iraq", "Libya", "Sudan", "Zimbabwe", "Syria"]

    def __init__(self):
        self.logger = Logger()
        self.pubsub = PubSub()
        self.db_manager = DatabaseManager()

    def handle_request(self, filename, request_method, headers):
        country = headers.get("X-country")
        client_ip = headers.get("X-client-IP")
        gender = headers.get("X-gender")
        age = headers.get("X-age")
        income = headers.get("X-income")

        if not (country and client_ip and gender and age and income):
            error_code = 9001 # Custom Error Code
            self.logger.log(f"Error Code 9001: No Header")
            return 'No Headers', error_code

        is_banned = country in AppService.BANNED_COUNTRIES

        time_of_day = datetime.now().strftime('%H:%M:%S')

        requested_file = filename.replace('bu-ds561-dk98-bucket/', '')

        error_code = None
```

```python
    if request_method == 'GET':
        if is_banned:
            data = str({'400 Forbidden from country': country})
            self.pubsub.publish(data)
            self.logger.log(f"Error Code 400: Forbidden: {str(country)}")
            error_code = 400
            self.db_manager.handle_database(country, client_ip, gender, age, income, is_banned, time_of_day, requested_file, error_code)
            return "Permission Denied", error_code

        try:
            storage_client = storage.Client()
            bucket = storage_client.bucket('bu-ds561-dk98-bucket')
            blob = bucket.blob(requested_file)
            file_content = blob.download_as_text()
            self.logger.log(f"200: {requested_file}")
            self.db_manager.handle_database(country, client_ip, gender, age, income, is_banned, time_of_day, requested_file, error_code)
            return file_content, 200
        except Exception as e:
            self.logger.log(f"Error Code 404: {requested_file}: {str(e)}")
            error_code = 404
            self.db_manager.handle_database(country, client_ip, gender, age, income, is_banned, time_of_day, requested_file, error_code)
            return 'File not found', error_code

    else:
        error_code = 501
        self.db_manager.handle_database(country, client_ip, gender, age, income, is_banned, time_of_day, requested_file, error_code)
        self.logger.log(f"Error Code 501: {request_method}")
        return 'Not implemented', error_code

# Flask route handling
service = AppService()

@app.route('/', defaults={'path': ''}, methods=['GET', 'POST', 'PUT', 'DELETE', 'HEAD', 'CONNECT', 'OPTIONS', 'TRACE', 'PATCH'])
@app.route('/<path:filename>', methods=['GET', 'POST', 'PUT', 'DELETE', 'HEAD', 'CONNECT', 'OPTIONS', 'TRACE', 'PATCH'])
def app_one(filename):
    return service.handle_request(filename, request.method, request.headers)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080)
```

appone.py - new function - sql insertion



```python
class DatabaseManager:
    def __init__(self):
        self.logger = Logger()
        self.error_codes = {
            9001: "No Headers - Required headers are missing",
            400: "Forbidden Access - Access denied due to client's country",
            404: "File Not Found - The requested file does not exist",
            501: "Method Not Implemented - The request method is not supported",
        }
        self.dbConnector = DatabaseConnector()

    def insert_client(self, country, client_ip, gender, age, income, is_banned):
        with self.dbConnector.pool.connect() as connection:
            try:
                result = connection.execute(
                    sqlalchemy.text("SELECT client_id FROM Clients WHERE client_ip = :client_ip"),
                    {"client_ip": client_ip}
                ).first()
                client_id = result[0] if result else None

                if client_id is None:
                    connection.execute(
                        sqlalchemy.text(
                            "INSERT INTO Clients (client_ip, country, gender, age, income, is_banned) "
                            "VALUES (:client_ip, :country, :gender, :age, :income, :is_banned)"
                        ),
                        {
                            "client_ip": client_ip,
                            "country": country,
                            "gender": gender,
                            "age": age,
                            "income": income,
                            "is_banned": is_banned
                        }
                    )
                    client_id = connection.execute(sqlalchemy.text("SELECT LAST_INSERT_ID()")).scalar()
                connection.commit()
                return client_id
            except Exception as e:
                self.logger.log(f"insert_client sql failed: {str(e)}")
                connection.rollback()
```

```python
    def insert_file(self, requested_file):
        with self.dbConnector.pool.connect() as connection:
            try:
                result = connection.execute(
                    sqlalchemy.text("SELECT file_id FROM Files WHERE file_name = :requested_file"),
                    {"requested_file": requested_file}
                ).first()
                file_id = result[0] if result else None

                if file_id is None:
                    connection.execute(
                        sqlalchemy.text("INSERT INTO Files (file_name) VALUES (:requested_file)"),
                        {"requested_file": requested_file}
                    )
                    file_id = connection.execute(sqlalchemy.text("SELECT LAST_INSERT_ID()")).scalar()
                connection.commit()
                return file_id
            except Exception as e:
                self.logger.log(f"insert_file sql failed: {str(e)}")
                connection.rollback()

    def insert_error_code(self, error_code, description=""):
        with self.dbConnector.pool.connect() as connection:
            try:
                result = connection.execute(
                    sqlalchemy.text("SELECT error_code_id FROM ErrorCodes WHERE error_code = :error_code"),
                    {"error_code": error_code}
                ).first()
                error_code_id = result[0] if result else None

                if error_code_id is None:
                    connection.execute(
                        sqlalchemy.text(
                            "INSERT INTO ErrorCodes (error_code, description) VALUES (:error_code, :description)"
                        ),
                        {
                            "error_code": error_code,
                            "description": description
                        }
                    )
                    error_code_id = connection.execute(sqlalchemy.text("SELECT LAST_INSERT_ID()")).scalar()
                connection.commit()
                return error_code_id
            except Exception as e:
                self.logger.log(f"insert_error_code sql failed: {str(e)}")
                connection.rollback()
```

```python
    def insert_request_or_failure(self, time_of_day, file_id, client_id, error_code_id):
        with self.dbConnector.pool.connect() as connection:
            try:
                if error_code_id:
                    # Insert into Failed_Requests
                    connection.execute(
                        sqlalchemy.text(
                            "INSERT INTO Failed_Requests (time_of_request, file_id, error_code_id) "
                            "VALUES (:time_of_day, :file_id, :error_code_id)"
                        ),
                        {
                            "time_of_day": time_of_day,
                            "file_id": file_id,
                            "error_code_id": error_code_id
                        }
                    )
                else:
                    # Insert into Requests
                    connection.execute(
                        sqlalchemy.text(
                            "INSERT INTO Requests (client_id, file_id, time_of_day) "
                            "VALUES (:client_id, :file_id, :time_of_day)"
                        ),
                        {
                            "client_id": client_id,
                            "file_id": file_id,
                            "time_of_day": time_of_day
                        }
                    )
                connection.commit()
            except Exception as e:
                self.logger.log(f"insert_request_or_failure sql failed: {str(e)}")
                connection.rollback()
```

```python
    def handle_database(self, country, client_ip, gender, age, income, is_banned, time_of_day, requested_file, error_code):
        try:
            error_code_id = None
            file_id = self.insert_file(requested_file)
            client_id = self.insert_client(country, client_ip, gender, age, income, is_banned)

            if error_code:
                description = self.error_codes.get(error_code, "Unknown error")
                error_code_id = self.insert_error_code(error_code, description)

            self.insert_request_or_failure(time_of_day, file_id, client_id, error_code_id)

            self.logger.log('Handling database completed successfully')

        except Exception as e:
            self.logger.log(f'Error in handle_database: {e}')
```

.env

```
DB_CONNECTION_STRING=ds-561:us-east1:hw5
DB_USER=root
DB_PASSWORD=# HIDDEN FOR SCREENSHOT
DB_DATABASE=hw5db
```

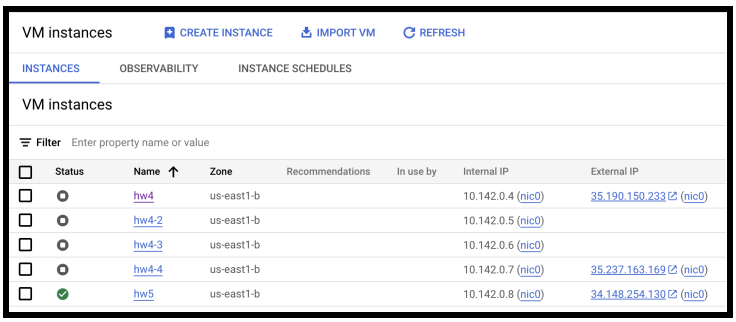
9. Create VM (Exactly Like First VM in HW4) - 34.75.244.245
   - (Changed to E2 Standard)

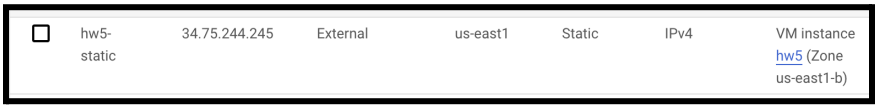
Automation

Startup script

```
sudo systemctl enable hw5webserver
sudo systemctl start hw5webserver
```

You can choose to specify a startup script that will run when your instance boots up or restarts. Startup scripts can be used to install software and updates, and to ensure that services are running within the virtual machine. Learn more





10. Give SQL Access Service Account

```
gcloud projects add-iam-policy-binding ds-561 \
   --member serviceAccount:bu-ds561-dk98-sa@ds-561.iam.gserviceaccount.com \
   --role roles/cloudsql.admin
```

11. Connect Service Account to VM

```
gcloud compute instances set-service-account hw5 \
   --service-account bu-ds561-dk98-sa@ds-561.iam.gserviceaccount.com \
   --scopes cloud-platform
```

12. Send Local Python File to VM (Local)

```
gcloud compute scp
    /Users/davidekim/Desktop/DataScience/BU/DS561/ds561-davidekim-U66545284/hw5/
    app_one.py hw5:~/app_one.py --zone us-east1-b

gcloud compute scp
    /Users/davidekim/Desktop/DataScience/BU/DS561/ds561-davidekim-U66545284/hw5/
    .env hw5:~/.env --zone us-east1-b
```

13. Install necessary libraries (SSH)

```
sudo apt-get update
sudo apt-get install python3 python3-pip

pip3 install cloud-sql-python-connector python-dotenv sqlalchemy pymysql
pip3 install Flask google-cloud-storage google-cloud-pubsub google-cloud-logging
```

14. Create Webserver (SSH)

```
    sudo nano /etc/systemd/system/hw5webserver.service

    [Unit]
    Description=HW5 Web Server
```

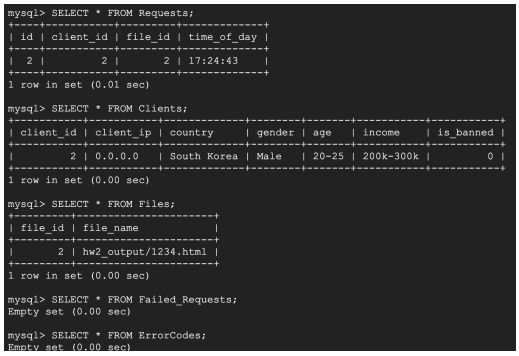
```
[Service]
ExecStart=/usr/bin/python3 app_one.py
Restart=always
User=dk98
WorkingDirectory=/home/davidekim

[Install]
WantedBy=multi-user.target
```

15. Run Webserver (Restart VM or SSH)

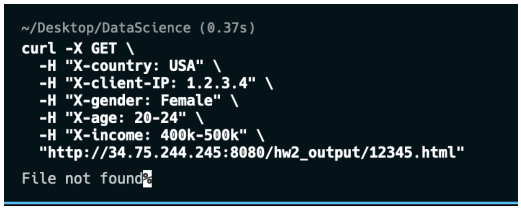
```
sudo systemctl enable hw5webserver
sudo systemctl start hw5webserver
sudo systemctl status hw5webserver
```

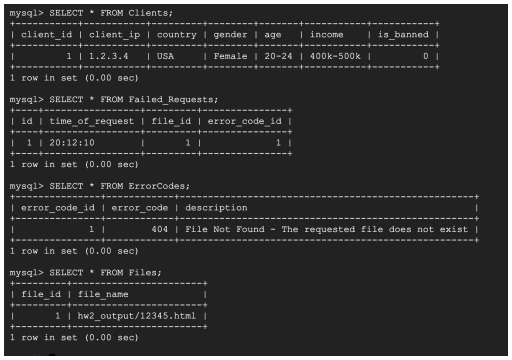
16. Curl Commands

```
curl -X GET \
  -H "X-country: South Korea" \
  -H "X-client-IP: 0.0.0.0" \
  -H "X-gender: Male" \
  -H "X-age: 20-25" \
  -H "X-income: 200k-300k" \
  "http://34.75.244.245:8080/hw2_output/1234.html"
```

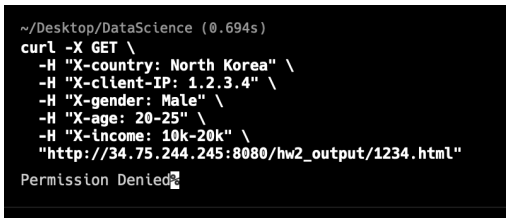


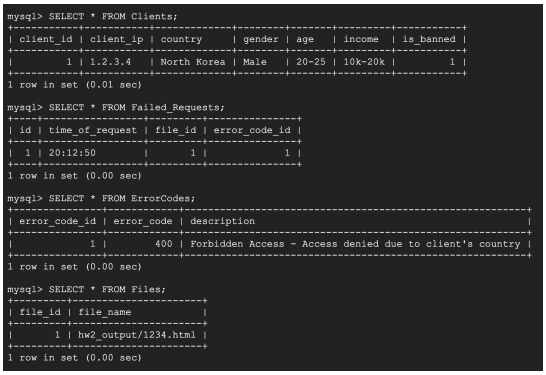

```
curl -X GET \
  -H "X-country: USA" \
  -H "X-client-IP: 1.2.3.4" \
  -H "X-gender: Female" \
  -H "X-age: 20-24" \
  -H "X-income: 400k-500k" \
  "http://34.75.244.245:8080/hw2_output/12345.html"
```





curl -X GET \
-H "X-country: North Korea" \
-H "X-client-IP: 1.2.3.4" \
-H "X-gender: Male" \
-H "X-age: 20-25" \
-H "X-income: 10k-20k" \
"http://34.75.244.245:8080/hw2_output/1234.html"





17. SQL Commands

```
DELETE FROM Requests;
DELETE FROM Clients;
DELETE FROM Failed_Requests;
DELETE FROM ErrorCodes;
DELETE FROM Files;

ALTER TABLE Requests AUTO_INCREMENT = 1;
ALTER TABLE Clients AUTO_INCREMENT = 1;
ALTER TABLE Failed_Requests AUTO_INCREMENT = 1;
ALTER TABLE ErrorCodes AUTO_INCREMENT = 1;
ALTER TABLE Files AUTO_INCREMENT = 1;

SELECT * FROM Requests;
SELECT * FROM Clients;
SELECT * FROM Failed_Requests;
```

SELECT * FROM ErrorCodes;
SELECT * FROM Files;

18. Run http-client on VM. - Two concurrent clients with 50,000 requests each

seq 2 | xargs -I{} -P2 python3 http-client.py -d 34.75.244.245 -b /bu-ds561-dk98-bucket -w hw2_output -n 50000 -i 11000 -p 8080 -f

```
dk98@hw4-2:/home/davidekim$ seq 2 | xargs -I{} -P2 python3 http-client.py -d 34.75.244.245 -b /bu-ds561-dk98-bu
cket -w hw2_output -n 6338 -i 11000 -p 8080 -f
dk98@hw4-2:/home/davidekim$
```

19. Statistics (My data might be different because I had to stop in mid and rerun.)

a. How many requests were you able to process successfully vs unsuccessfully?

i.

```
mysql> SELECT COUNT(*) FROM Requests;
+----------+
| COUNT(*) |
+----------+
|    86657 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM Failed_Requests;
+----------+
| COUNT(*) |
+----------+
|    13343 |
+----------+
1 row in set (0.01 sec)
```

ii. 86,657 Successful vs. 13,343 Unsuccessful

b. How many requests came from banned countries?

i. SELECT COUNT(*)
ii. FROM Failed_Requests fr
iii. INNER JOIN ErrorCodes ec ON fr.error_code_id = ec.error_code_id
iv. WHERE ec.error_code = 400;

v.

```
mysql> SELECT COUNT(*)
    -> FROM Failed_Requests fr
    -> INNER JOIN ErrorCodes ec ON fr.error_code_id = ec.error_code_id
    -> WHERE ec.error_code = 400;
+----------+
| COUNT(*) |
+----------+
|     4609 |
+----------+
1 row in set (0.00 sec)
```

vi. 4,609 Requests

c. How many requests were made by Male vs Female users?

i. SELECT COUNT(*)
ii. FROM Requests r
iii. INNER JOIN Clients c ON r.id = c.client_id
iv. WHERE c.gender = 'Male';

v.

```
mysql> SELECT COUNT(*)
    -> FROM Requests r
    -> INNER JOIN Clients c ON r.id = c.client_id
    -> WHERE c.gender = 'Male';
+----------+
| COUNT(*) |
+----------+
|    43324 |
+----------+
1 row in set (0.08 sec)
```

vi. Male: 43,324 Requests

vii. SELECT COUNT(*)
viii. FROM Requests r
ix. INNER JOIN Clients c ON r.id = c.client_id
x. WHERE c.gender = 'Female';

xi.

```
mysql> SELECT COUNT(*)
    -> FROM Requests r
    -> INNER JOIN Clients c ON r.id = c.client_id
    -> WHERE c.gender = 'Female';
+----------+
| COUNT(*) |
+----------+
|    43333 |
+----------+
1 row in set (0.09 sec)
```

xii. Female: 43,333 Requests

d. What were the top 5 countries sending requests to your server?
   i. SELECT C.country, COUNT(R.id)
   ii. FROM Requests R
   iii. JOIN Clients C ON R.id = C.client_id
   iv. GROUP BY C.country
   v. ORDER BY COUNT(R.id) DESC
   vi. LIMIT 5;

vii.

```
mysql> SELECT C.country, COUNT(R.id)
    -> FROM Requests R
    -> JOIN Clients C ON R.id = C.client_id
    -> GROUP BY C.country
    -> ORDER BY COUNT(R.id) DESC
    -> LIMIT 5;
+-------------+-------------+
| country     | COUNT(R.id) |
+-------------+-------------+
| New Zealand |         504 |
| Ireland     |         499 |
| Finland     |         497 |
| Ghana       |         497 |
| Indonesia   |         496 |
+-------------+-------------+
5 rows in set (0.17 sec)
```

viii. New Zealand - 504 Requests
ix. Ireland - 499 Requests
x. Finland - 497 Requests
xi. Ghana - 497 Requests
xii. Indonesia - 496 Requests

e. What age group issued the most requests to your server?
   i. SELECT C.age, COUNT(R.id)
   ii. FROM Requests R
   iii. JOIN Clients C ON R.id = C.client_id
   iv. GROUP BY C.age
   v. ORDER BY COUNT(R.id) DESC
   vi. LIMIT 1;

vii.

```
mysql> SELECT C.age, COUNT(R.id)
    -> FROM Requests R
    -> JOIN Clients C ON R.id = C.client_id
    -> GROUP BY C.age
    -> ORDER BY COUNT(R.id) DESC
    -> LIMIT 1;
+-------+-------------+
| age   | COUNT(R.id) |
+-------+-------------+
| 36-45 |       10950 |
+-------+-------------+
1 row in set (0.14 sec)
```

viii. Age Group = 36-45 = 10,950 Requests

f. What income group issued the most requests to your server?
   i. SELECT C.income, COUNT(R.id)
   ii. FROM Requests R
   iii. JOIN Clients C ON R.id = C.client_id
   iv. GROUP BY C.income
   v. ORDER BY COUNT(R.id) DESC
   vi. LIMIT 1;

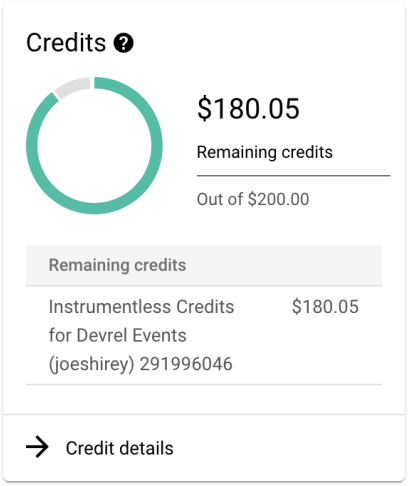
```
mysql> SELECT C.income, COUNT(R.id)
    -> FROM Requests R
    -> JOIN Clients C ON R.id = C.client_id
    -> GROUP BY C.income
    -> ORDER BY COUNT(R.id) DESC
    -> LIMIT 1;
+---------+-------------+
| income  | COUNT(R.id) |
+---------+-------------+
| 20k-40k |       10969 |
+---------+-------------+
1 row in set (0.15 sec)
```

vii.

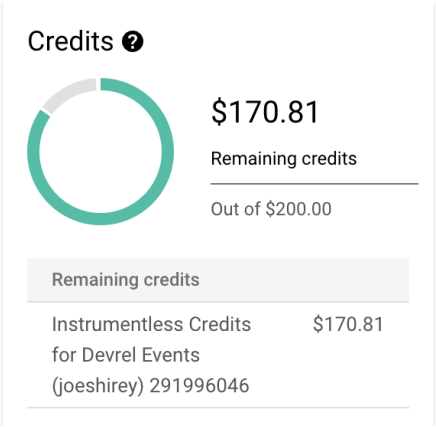
viii. Income Group = 20k-40k = 10,969 Requests

20. Cost

Before



After



END.