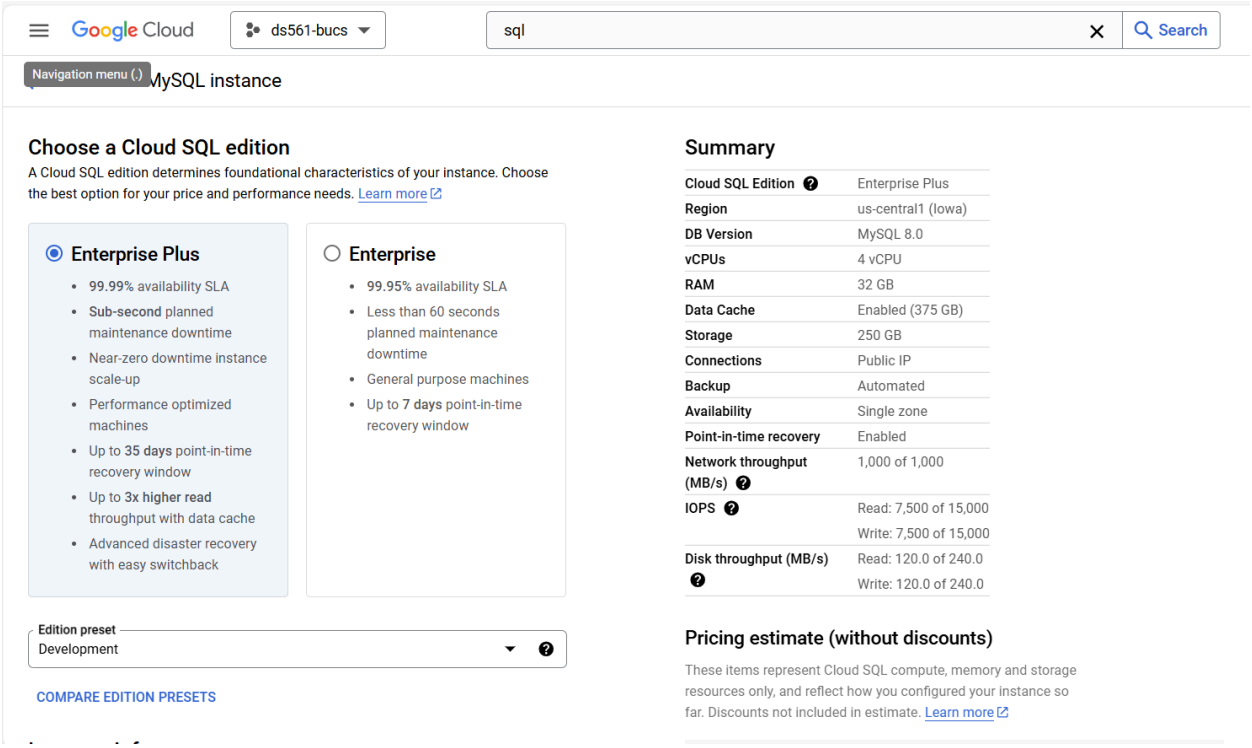
Name: Jeremy Bui
Course: DS 561

# Homework 5

## Setup

- Use existing VMs from HW4
- Create mySQL database instance:




- Connect to mySQL instance

```
jbui@LAPTOP-AA0HIAS7:/mnt/c/Users/jerem/dev/github/ds561/05-cloudsql-bui-jeremy$ mysql -u root -p -h 34.42.100.68
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 60
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2024, 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>
```

- Create DB:

```
mysql> create DATABASE hw5;
Query OK, 1 row affected (0.06 sec)

mysql> USE hw5;
Database changed
```

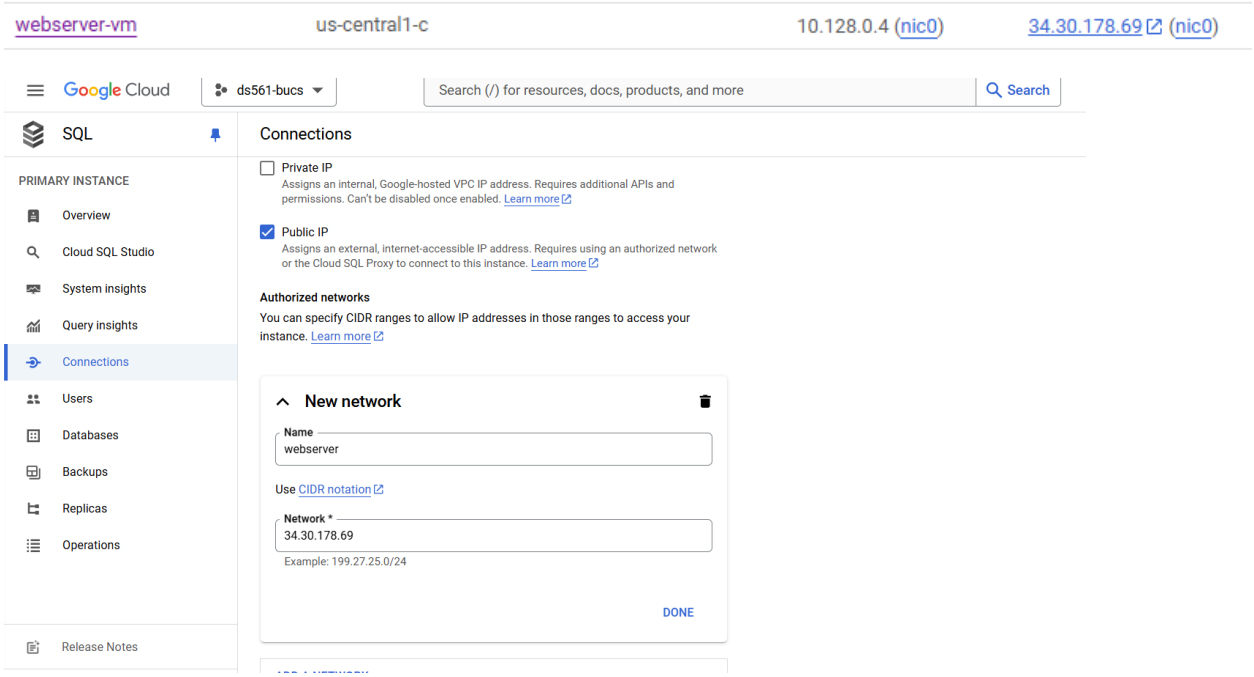
- Create table to store requests and table for failed requests:

```
mysql> CREATE TABLE requests (id INT AUTO_INCREMENT PRIMARY KEY, country VARCHAR(255), client_ip VARCHAR(255), gender VARCHAR(255), age VARCHAR(10), inc
ome VARCHAR(255), is_banned BOOLEAN, time_of_day VARCHAR(25), requested_file VARCHAR(255));
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> CREATE TABLE failed_requests (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> time_req TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    -> req_file VARCHAR(255),
    -> err_code INT
    -> );
Query OK, 0 rows affected (0.09 sec)
```

- Both tables satisfy 2NF because
  - It satisfies 1NF :
    - Each attribute in both tables, requests and failed_requests, hold only a single atomic value of country, client_ip, gender, requested file, etc
    - No attribute in both tables has multiple values
  - None of the non-key attribute depends on a subset of a key and depends on the primary key
    - For the request table, the primary key is ‘id’ and is unique for each request. All of the non-key attributes depend on the ‘id’
    - In the failed_requests table, the ‘id’ is the primary key and the non-key attributes depend on it
  - All tables that are 1NF and have a key that is a single attribute is always 2NF:
    - Both table have a single primary key ‘id’ making it 2NF
- From trying out part 6, my VM’s seemed to be too slow so I had to upgrade the webserver-vm and http-client vm to a higher CPU one
  - I ended up using an n2-standard with 4 CPUs for both and made sure they were on same region
- Provide credentials to service account to access storage bucket, sql, etc (I just added sql as an extra permission from the last HW)

    - gcloud compute instances set-service-account webserver-vm --zone=us-east1-c --service-account=844994497823-compute@developer.gserviceaccount.com --scopes=https://www.googleapis.com/auth/cloud-platform
    - gcloud compute instances set-service-account httpclient-vm --zone=us-east1-c --service-account=844994497823-compute@developer.gserviceaccount.com --scopes=https://www.googleapis.com/auth/cloud-platform
- Install dependencies on vms:
    - sudo apt update
    - sudo apt install -y python3 python3-pip
    - python3 -m venv myenv
    - source myenv/bin/activate
    - pip install google-cloud-storage google-cloud-logging google-cloud-pubsub mysql-connector-python
- Add webserver-vm's external IP to authorized networks




# Code:

tracker.py:

```
from google.cloud import pubsub_v1

def callback(message):
    print("Received message from banned country: " +
message.data.decode('utf-8'))
    message.ack()

def listenToBannedCountryMessages():
    subscriber = pubsub_v1.SubscriberClient()
    subscriptionPath = subscriber.subscription_path('ds561-bucs',
'banned-country-subscription')
    print("Listening for messages on " + subscriptionPath + "...")
    streamingPullFuture = subscriber.subscribe(subscriptionPath,
callback=callback)

    try:
        streamingPullFuture.result()
    except KeyboardInterrupt:
        streamingPullFuture.cancel()

if __name__ == "__main__":
    listenToBannedCountryMessages()
```

- Python app using pubsub_v1 to listen for messages from Pub/Sub
- Callback function prints messages from banned countries
- Subscribes to the banned-country-subscription in project ds561-bucs
- Listens for messages from the Pub/Sub topic with subscriber.subscribe
- Run in the background, waiting for messages until stopped manually
- Handle interrupts to stop listening with streamingPullFuture.cancel
- When a message is received, it shows the banned country and tracks the request
- Use pub/sub to connect and share info between this app and the web server

web-server.py:

```
from http.server import BaseHTTPRequestHandler, HTTPServer
from socketserver import ThreadingMixIn
from google.cloud import storage, logging as cloudLogging, pubsub_v1
import json
import mysql.connector
```

```
cloudLoggingClient = cloudLogging.Client()
cloudLogger = cloudLoggingClient.logger("httpServerLog")
bannedCountries = ['North Korea', 'Iran', 'Cuba', 'Myanmar', 'Iraq',
'Libya', 'Sudan', 'Zimbabwe', 'Syria']
projectId = 'ds561-bucs'
topicId = 'banned-country-topic'

def notifyTrackerApp(country):
    publisher = pubsub_v1.PublisherClient()
    topicPath = publisher.topic_path(projectId, topicId)
    messageData = json.dumps({'country': country}).encode('utf-8')
    publisher.publish(topicPath, messageData)

def fetchFileFromBucket(bucketPath):
    bucketName, filePath = bucketPath.split("/", 1)
    bucket = storage.Client().get_bucket(bucketName)
    blob = bucket.blob(filePath)
    if blob.exists():
        return blob.download_as_text()
    return None

def log_request_to_db(country, client_ip, gender, age, income, is_banned,
time_of_day, requested_file):
    conn = mysql.connector.connect(host="34.42.100.68", user="root",
password="", database="hw5")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO requests (country, client_ip, gender, age,
income, is_banned, time_of_day, requested_file) VALUES (%s, %s, %s, %s,
%s, %s, %s, %s)",
                   (country, client_ip, gender, age, income, is_banned,
time_of_day, requested_file))
    conn.commit()
    cursor.close()
    conn.close()

def log_error_to_db(requested_file, error_code):
    conn = mysql.connector.connect(host="34.42.100.68", user="root",
password="", database="hw5")
    cursor = conn.cursor()
```

```
    cursor.execute("INSERT INTO failed_requests (req_file, err_code)
VALUES (%s, %s)",
                   (requested_file, error_code))
    conn.commit()
    cursor.close()
    conn.close()

class RequestHandler(BaseHTTPRequestHandler):
    def log_message(self, format, *args):
        pass

    def do_GET(self):
        country = self.headers.get('X-Country', 'unknown')
        client_ip = self.headers.get('X-client-IP', 'unknown')
        gender = self.headers.get('X-gender', 'unknown')
        age = self.headers.get('X-age', 'unknown')
        income = self.headers.get('X-income', 'unknown')
        time_of_day = self.headers.get('X-time', 'unknown')

        is_banned = country in bannedCountries
        if is_banned:
            cloudLogger.log_text("Access denied from banned country: " +
country, severity="ERROR")
            notifyTrackerApp(country)
            self.response_code = 400
            self.send_response(400)
            self.end_headers()
            self.wfile.write(b"Access denied: Request from banned
country")
            log_request_to_db(country, client_ip, gender, age, income,
True, time_of_day, self.path)
            log_error_to_db(self.path, 400)
            return

        try:
            fileContent = fetchFileFromBucket(self.path.strip("/"))
            if fileContent is None:
                cloudLogger.log_text("File " + self.path + " not found",
severity="ERROR")
                self.response_code = 404
```

```
                self.send_response(404)
                self.end_headers()
                self.wfile.write(b"File not found")
                log_request_to_db(country, client_ip, gender, age, income,
False, time_of_day, self.path)
                log_error_to_db(self.path, 404)
            else:
                self.response_code = 200
                self.send_response(200)
                self.end_headers()
                self.wfile.write(fileContent.encode("utf-8"))
                log_request_to_db(country, client_ip, gender, age, income,
False, time_of_day, self.path)
        except Exception as e:
            cloudLogger.log_text("Error occurred: " + str(e),
severity="ERROR")
            self.response_code = 500
            self.send_response(500)
            self.end_headers()
            log_request_to_db(country, client_ip, gender, age, income,
False, time_of_day, self.path)
            log_error_to_db(self.path, 500)

    def handle_unsupported_methods(self):
        cloudLogger.log_text("Unsupported method " + self.command + " was
used for " + self.path, severity="ERROR")
        self.response_code = 501
        self.send_response(501)
        self.end_headers()
        self.wfile.write((self.command + " not
implemented").encode("utf-8"))
        log_request_to_db("unknown", "unknown", "unknown", "unknown",
"unknown", False, "unknown", self.path)
        log_error_to_db(self.path, 501)

    def do_POST(self):
        self.handle_unsupported_methods()

    def do_PUT(self):
        self.handle_unsupported_methods()
```

```
    def do_DELETE(self):
        self.handle_unsupported_methods()

    def do_HEAD(self):
        self.handle_unsupported_methods()

    def do_CONNECT(self):
        self.handle_unsupported_methods()

    def do_OPTIONS(self):
        self.handle_unsupported_methods()

    def do_TRACE(self):
        self.handle_unsupported_methods()

    def do_PATCH(self):
        self.handle_unsupported_methods()

class ThreadingHTTPServer(ThreadingMixIn, HTTPServer):
    pass

def runServer(serverAddress, port):
    server = ThreadingHTTPServer((serverAddress, port), RequestHandler)
    server.serve_forever()

if __name__ == "__main__":
    runServer("0.0.0.0", 8080)
```

- Python multithreaded web server using http.server and ThreadingHTTPServer to handle concurrent requests
    - ThreadingMixIn allows each request to be handled in own thread
- Use do_GET to check the X-Country header against a list of banned countries
    - If matched, logs the request, sends a Pub/Sub message to tracker app, and returns a 400 error
    - If the requested file is not from a banned country, do_GET() we want to get the file from google cloud storage
    - If the file exists, get its content and return a 200
    - If not, it logs a 404 error and responds with "file not found"

- Connect to the database using mysql.connector.connect with host IP, username, password, and database name
- Log each request's details to MySQL using function log_request_to_db, recording country, client_ip, gender, age, income, is_banned, time_of_day, and requested_file
- Unsupported HTTP methods (POST, DELETE, etc.) are logged and responded to with a 501 error using handle_unsupported_methods
- Executes an INSERT SQL command to add request data to the requests table and commits after each entry to save it
- Log all incoming requests (both successful and error cases) to google cloud logging and requests tables
- Log all failed requests (requests that are not 200) to the failed_requests table

## 4) Demonstrate the functionality of your app by using curl to issue one successful and 2 erroneous requests and show the contents of your tables before and after each request.

- Successful request (200)

```
jbui@LAPTOP-AA0HIAS7:/mnt/c/Users/jerem/dev/github/ds561/05-cloudsql-bui-jeremy$ curl -X GET http://34.30.178.69:8080/jeremybui_ps2/files/100.html -H "X-Cou
ntry: USA" -H "X-client-IP: 123.456.78.9" -H "X-gender: Male" -H "X-age: 22" -H "X-income: 110k" -H "X-time: 2024-10-24 12:00:00"
<!DOCTYPE html>
<html>
<body>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit e
sse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
<p>
```

(Before)

```
mysql> SELECT * FROM requests;
Empty set (0.03 sec)

mysql> SELECT * FROM failed_requests;
Empty set (0.03 sec)
```

(After)

```
mysql> SELECT * FROM requests;
+----+---------+-------------+--------+------+--------+-----------+---------------------+--------------------------
----+
| id | country | client_ip   | gender | age  | income | is_banned | time_of_day         | requested_file
    |
+----+---------+-------------+--------+------+--------+-----------+---------------------+--------------------------
----+
|  1 | USA     | 123.456.78.9 | Male   | 22   | 110k   |         0 | 2024-10-24 12:00:00 | /jeremybui_ps2/files/100.h
tml |
+----+---------+-------------+--------+------+--------+-----------+---------------------+--------------------------
----+
1 row in set (0.04 sec)

mysql> SELECT * FROM failed_requests;
Empty set (0.03 sec)
```

- Erroneous request - Not Found (404)

```
cloudsql-bui-jeremy$ curl -X GET "http://34.150.145.172:8080/jeremybui_ps2/files/99999.html" -H "X-Cou
ntry: USA" -H "X-client-IP: 123.456.78.9" -H "X-gender: Male" -H "X-age: 22" -H "X-income: 110k-120k"
-H "X-time: 2024-10-25 12:00:00"
File not foundjbui@LAPTOP-AA0HIAS7:/mnt/c/Users/jerem/dev/github/ds561/05-cloudsql-bui-jeremy$
```

(Before)

```
mysql> SELECT * FROM requests;
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
| id | country     | client_ip   | gender | age  | income    | is_banned | time_of_day         | requested_file              |
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
|  1 | North Korea | 123.456.78.9 | Male   | 22   | 110k-120k |         1 | 2024-10-25 12:00:00 | /jeremybui_ps2/files/100.html |
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
1 row in set (0.04 sec)

mysql> SELECT * FROM failed_requests;
+----+---------------------+-----------------------------+----------+
| id | time_req            | req_file                    | err_code |
+----+---------------------+-----------------------------+----------+
|  1 | 2024-10-25 21:14:48 | /jeremybui_ps2/files/100.html |      400 |
+----+---------------------+-----------------------------+----------+
1 row in set (0.04 sec)
```

(After)

```
mysql> SELECT * FROM requests;
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
| id | country     | client_ip   | gender | age  | income    | is_banned | time_of_day         | requested_file              |
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
|  1 | North Korea | 123.456.78.9 | Male   | 22   | 110k-120k |         1 | 2024-10-25 12:00:00 | /jeremybui_ps2/files/100.html   |
|  2 | USA         | 123.456.78.9 | Male   | 22   | 110k-120k |         0 | 2024-10-25 12:00:00 | /jeremybui_ps2/files/99999.html |
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
2 rows in set (0.03 sec)

mysql> SELECT * FROM failed_requests;
+----+---------------------+-----------------------------+----------+
| id | time_req            | req_file                    | err_code |
+----+---------------------+-----------------------------+----------+
|  1 | 2024-10-25 21:14:48 | /jeremybui_ps2/files/100.html   |      400 |
|  2 | 2024-10-25 21:15:55 | /jeremybui_ps2/files/99999.html |      404 |
+----+---------------------+-----------------------------+----------+
2 rows in set (0.04 sec)
```

```
> 2024-10-24 22:02:25.212  webserver-vm  File /jeremybui_ps2/files/non_existent_file.html not found
```

- Erroneous request - Banned (400)

```
jbui@LAPTOP-AA0HIAS7:/mnt/c/Users/jerem/dev/github/ds561/05-cloudsql-bui-jeremy$ curl -X GET "http://3
4.150.145.172:8080/jeremybui_ps2/files/100.html" -H "X-Country: North Korea" -H "X-client-IP: 123.456.
78.9" -H "X-gender: Male" -H "X-age: 22" -H "X-income: 110k-120k" -H "X-time: 2024-10-25 12:00:00"
Access denied: Request from banned countryjbui@LAPTOP-AA0HIAS7:/mnt/c/Users/jerem/dev/github/ds561/05-
cloudsql-bui-jeremy$
```

(Before)

```
mysql> SELECT * FROM requests;
Empty set (0.03 sec)

mysql> SELECT * FROM failed_requests;
Empty set (0.03 sec)
```

(After)

```
mysql> SELECT * FROM requests;
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
| id | country     | client_ip   | gender | age  | income    | is_banned | time_of_day         | requested_file              |
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
|  1 | North Korea | 123.456.78.9 | Male   | 22   | 110k-120k |         1 | 2024-10-25 12:00:00 | /jeremybui_ps2/files/100.html |
+----+-------------+-------------+--------+------+-----------+-----------+---------------------+-----------------------------+
1 row in set (0.04 sec)

mysql> SELECT * FROM failed_requests;
+----+---------------------+-----------------------------+----------+
| id | time_req            | req_file                    | err_code |
+----+---------------------+-----------------------------+----------+
|  1 | 2024-10-25 21:14:48 | /jeremybui_ps2/files/100.html |      400 |
+----+---------------------+-----------------------------+----------+
1 row in set (0.04 sec)
```


- 

**5/6) Provision a VM that can handle the request load from 2 concurrent clients (use the client that has been provided to you) and run each client to issue 50,000 requests against your VM. Make sure to start the 2 clients with the same random seed so the clients are consistent about their request generation. This may take a little while to finish (multiple minutes). Once the clients have completed, compute the following statistics on your requests:**

python3 http-client.py --domain 34.136.9.127  --bucket jeremybui_ps2 --webdir files --num_requests 1000 --port 8080 --random 561 &
python3 http-client.py --domain 34.136.9.127 --bucket jeremybui_ps2 --webdir files --num_requests 1000  --port 8080 --random 561 &
wait

- Total Time: 3 hours and 3 minutes to run 50,000 requests across 2 clients
- Note: My code and DB runs as expected, but there are sometimes outside connections to my webserver, which I experienced through logs.

```
mysql> SELECT COUNT(*) AS outside_source_requests
    -> FROM requests
    -> WHERE country = 'unknown'
    ->    OR client_ip = 'unknown'
    ->    OR gender = 'unknown'
    ->    OR age = 'unknown'
    ->    OR income = 'unknown';
+--------------------------+
| outside_source_requests |
+--------------------------+
|                       19 |
+--------------------------+
1 row in set (0.08 sec)
```

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

```
mysql> SELECT
    ->     COUNT(*) AS total_requests,
    ->     COUNT(*) - (SELECT COUNT(*) FROM failed_requests) AS successful_requests,
    ->     (SELECT COUNT(*) FROM failed_requests) AS unsuccessful_requests
    -> FROM requests;
+----------------+--------------------+----------------------+
| total_requests | successful_requests | unsuccessful_requests |
+----------------+--------------------+----------------------+
|         100019 |               9552 |                90467 |
+----------------+--------------------+----------------------+
1 row in set (0.06 sec)
```

- Count rows in the requests table
- Calculate number of successful requests by subtracting number of failed_requests from total number of requests
- Count all the rows in failed_requests table

*Ignoring the outside requests, the total number of requests should be 19, where all of these 19 failed. Therefore, the total should be:*

- *total_requests: 100000*
- *successful_requests: 9952*
- *unsuccessful_requests: 90448*

- How many requests came from banned countries?

```
mysql> SELECT COUNT(*) AS banned_requests
    -> FROM requests
    -> WHERE is_banned = TRUE;
+-----------------+
| banned_requests |
+-----------------+
|            4528 |
+-----------------+
1 row in set (0.05 sec)
```

- Count all the rows where the is_banned boolean is = TRue, meaning that it is a banned country

- How many requests were made by Male vs Female users?

```
Database changed
mysql> SELECT gender, COUNT(*) AS count
    -> FROM requests
    -> GROUP BY gender;
+---------+-------+
| gender  | count |
+---------+-------+
| Female  | 50246 |
| Male    | 49754 |
| unknown |    19 |
+---------+-------+
3 rows in set (0.09 sec)
```

- Group by gender field
- Count the total number requests by each gender group

- What were the top 5 countries sending requests to your server?

```
mysql> SELECT country, COUNT(*) AS count
    -> FROM requests
    -> GROUP BY country
    -> ORDER BY count DESC
    -> LIMIT 5;
+------------+-------+
| country    | count |
+------------+-------+
| Czechia    |   580 |
| Latvia     |   580 |
| Tajikistan |   578 |
| Liberia    |   578 |
| Lithuania  |   576 |
+------------+-------+
5 rows in set (0.09 sec)
```

- Group the data by the country
- Count the total number of requests from each country
- Sort the results in descending order by the count
- Use limit to show only the top 5

- What age group issued the most requests to your server?

```
mysql> SELECT age, COUNT(*) AS count
    -> FROM requests
    -> GROUP BY age
    -> ORDER BY count DESC
    -> LIMIT 1;
+------+-------+
| age  | count |
+------+-------+
| 0-16 | 12874 |
+------+-------+
1 row in set (0.08 sec)
```

- Group the data by the age
- Count the total number of requests by the age
- Order by the result
- Limit to 1 to show the most active group

- What income group issued the most requests to your server?

```
mysql> SELECT income, COUNT(*) AS count
    -> FROM requests
    -> GROUP BY income
    -> ORDER BY count DESC
    -> LIMIT 1;
+---------+-------+
| income  | count |
+---------+-------+
| 10k-20k | 12698 |
+---------+-------+
1 row in set (0.09 sec)
```

- Group by the income
- Count the total number of requests by income
- Order by the income in descending order so the highest income group is at the top
- Limit to 1 to show only the top group

## Billing Report