Importing Data to a Firestore Database

Task 1. Set up Firestore in Google Cloud

Patrick's task is to upload Pet Theory's existing data to a Cloud Firestore database. He will work closely with Ruby to accomplish this goal. Ruby receives a message from Patrick in IT...

1. On the Cloud Console Navigation menu (☰), click **View All Products** and under **Databases** select **Firestore**.
2. Click **Create a Firestore database**.
3. Select **Standard Edition**.
4. Under Configuration options, select **Firestore Native**.
5. For Security rules, choose **Open**.
6. In Location type, click **Region**, and then select the lab region ____ from the list.
7. Leave the other settings as their defaults, and click **Create Database**.

Task 2. Write database import code

The new Cloud Firestore database is in place, but it's empty. The customer data for Pet Theory still only exists in the old database.

1. In Cloud Shell, run the following command to clone the Pet Theory repository:

git clone https://github.com/rosera/pet-theory

```
student_04_bf6b34691762@cloudshell:~ (qwiklabs-gcp-02-2a7cffadde1a)$ git clone https://github.com/rosera/pet-theory
fatal: destination path 'pet-theory' already exists and is not an empty directory.
student_04_bf6b34691762@cloudshell:~ (qwiklabs-gcp-02-2a7cffadde1a)$
```

2. Use the Cloud Shell Code Editor (or your preferred editor) to edit your files. From the top ribbon of your Cloud Shell session, click **Open Editor**, it will open a new tab. If prompted, click **Open in a new window** to launch the code editor:

```
{
        "name": "lab01",
        "version": "1.0.0",
        "description": "This is lab01 of the Pet Theory labs",
        "main": "index.js",
        "scripts": {
                "test": "echo \"Error: no test specified\" && exit 1"
        },
        "keywords": [],
        "author": "Patrick - IT",
        "license": "MIT",
        "dependencies": {
                "csv-parse": "^5.5.3"
        }
}
```

4. Run the following command to do so:

```
npm install @google-cloud/firestore
```

5. To enable the app to write logs to Cloud Logging, install an additional module:

```
npm install @google-cloud/logging
```

6. Open the file pet-theory/lab01/importTestData.js.
7. Add the following Firestore dependency on line 3 of the file:

```
const { Firestore } = require("@google-cloud/firestore");
```

8. Add the following code underneath line 34, or after the if (process.argv.length < 3) conditional:

```
async function writeToFirestore(records) {
 const db = new Firestore({
  // projectId: projectId
 });
 const batch = db.batch()
```

```
  records.forEach((record)=>{
    console.log(`Write: ${record}`)
    const docRef = db.collection("customers").doc(record.email);
    batch.set(docRef, record, { merge: true })
  })

  batch.commit()
    .then(() => {
      console.log('Batch executed')
    })
    .catch(err => {
      console.log(`Batch error: ${err}`)
    })
  return
}
```

9. Update the importCsv function to add the function call to **writeToFirestore** and remove the call to **writeToDatabase**. It should look like this:

```
async function importCsv(csvFilename) {
  const parser = csv.parse({ columns: true, delimiter: ',' }, async function (err, records) {
    if (err) {
      console.error('Error parsing CSV:', err);
      return;
    }
    try {
      console.log(`Call write to Firestore`);
      await writeToFirestore(records);
```

```
    // await writeToDatabase(records);
    console.log(`Wrote ${records.length} records`);
  } catch (e) {
    console.error(e);
    process.exit(1);
  }
});

 await fs.createReadStream(csvFilename).pipe(parser);
}
```

Task 3. Create test data

Time to import some data! Patrick contacts Ruby about a concern he has about running a test with real customer data...

1. First, install the "faker" library, which will be used by the script that generates the fake customer data. Run the following command to update the dependency in package.json:

npm install faker@5.5.3

3. Add Logging for the codebase. On line 3, add the following reference for the Logging API module from the application code:

const { Logging } = require("@google-cloud/logging");

```
const fs = require("fs");
const faker = require("faker");
const { Logging } = require("@google-cloud/logging"); //add this
```

4. Now, add a few constant variables and initialize the Logging client. Add those just below the const statements:

const logName = "pet-theory-logs-createTestData";

// Creates a Logging client

const logging = new Logging();

const log = logging.log(logName);

const resource = {

// This example targets the "global" resource for simplicity

type: "global",

};

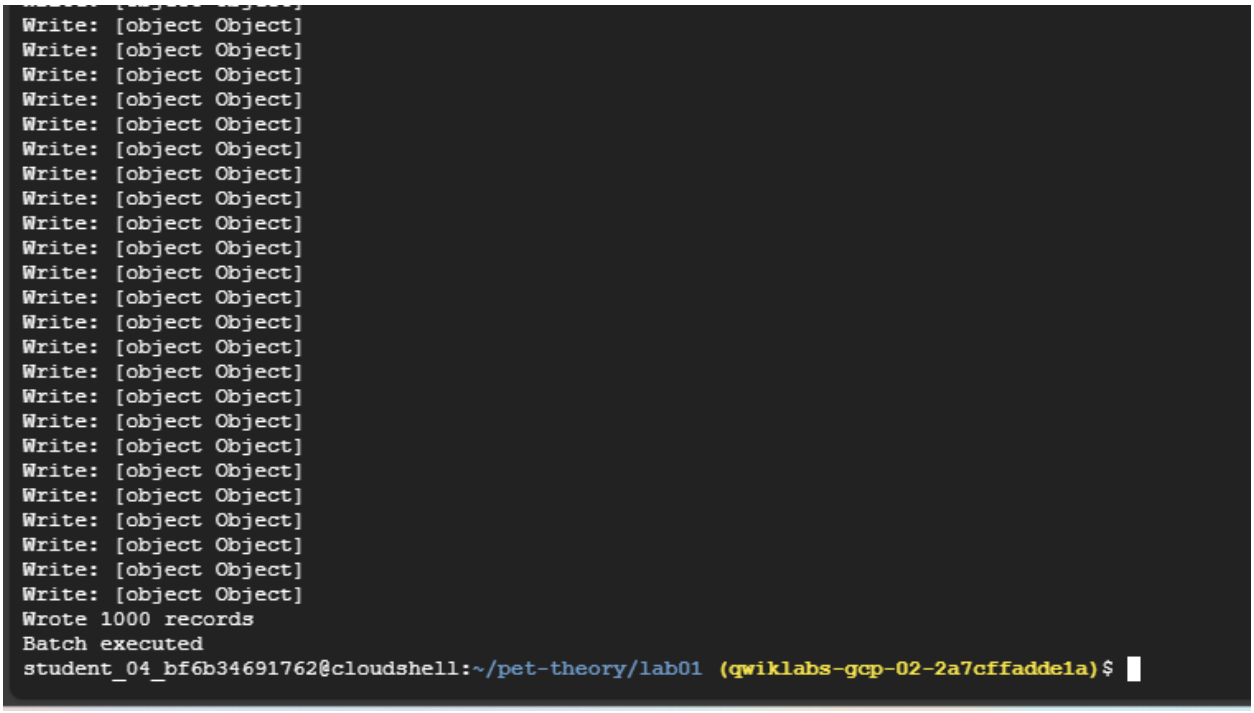
5. Add code to write the logs in the **createTestData** function just below the line "console.log(Created file ${fileName} containing ${recordCount} records.);" which will look like this:

```
// A text log entry
const success_message = `Success: createTestData - Created file
${fileName} containing ${recordCount} records.`;
const entry = log.entry(
        { resource: resource },
        {
                name: `${fileName}`,
                recordCount: `${recordCount}`,
                message: `${success_message}`,
        }
);
log.write([entry]);
```

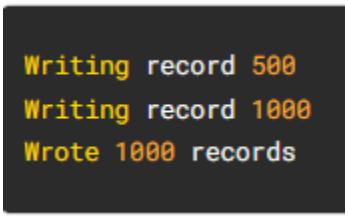
Task 4. Import the test customer data

1. To test the import capability, use both the import script and the test data created earlier:

node importTestData customers_1000.csv



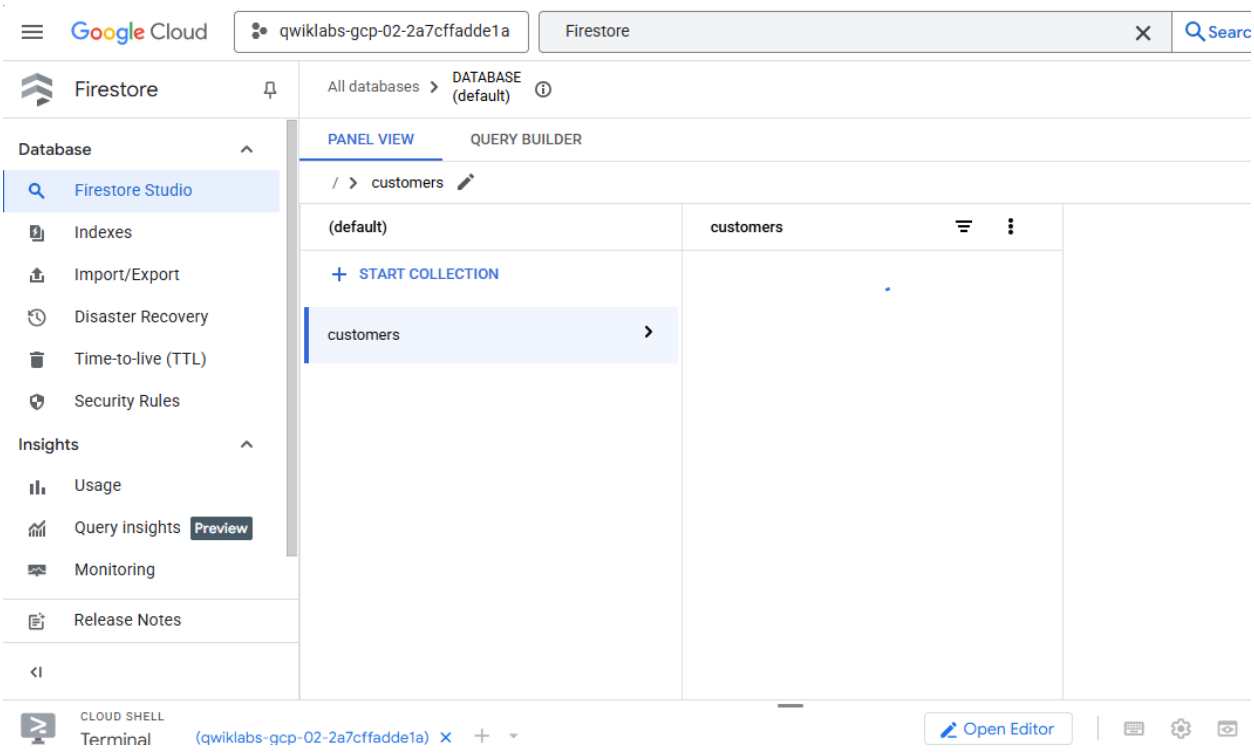
npm install csv-parse

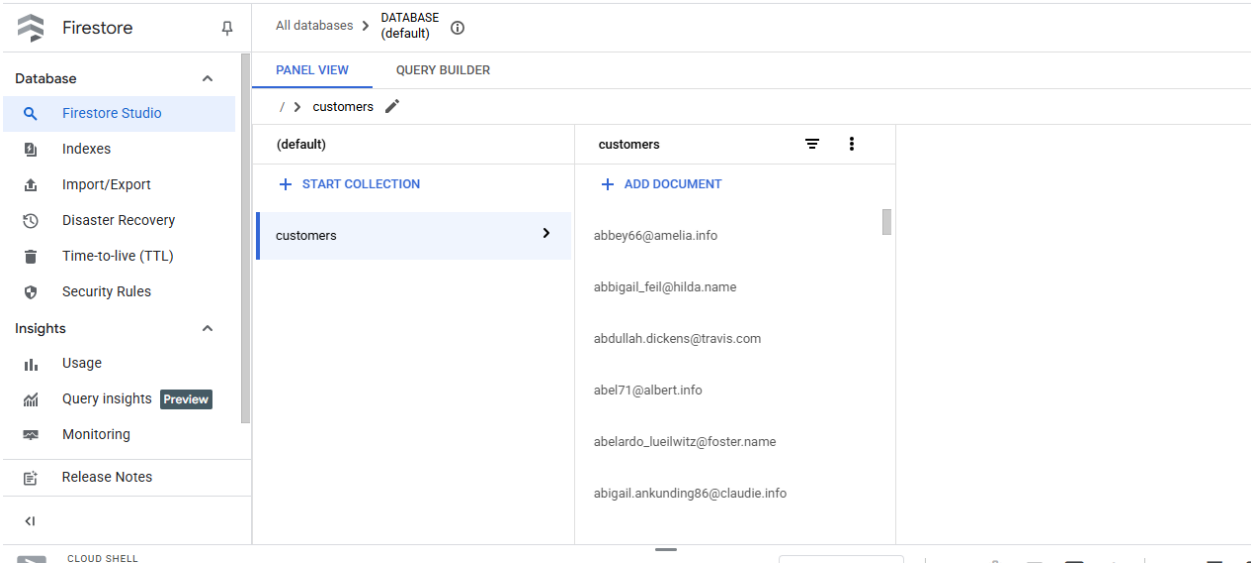


Task 5. Inspect the data in Firestore

With a little help from you and Ruby, Patrick has now successfully migrated the test data to the Firestore database. Open up Firestore and see the results!

1. Return to your Cloud Console tab. In the **Navigation menu** (☰), click **View All Products** and under **Databases** select **Firestore** then click on **default** database, Once there, click on the pencil icon.





1. Type in /customers and press **Enter**.
2. Refresh your browser tab and you should see the following list of customers successfully migrated