

HCS Challenge

Daniel Becerra – Software Engineer

Populating the data

The info is filled out using a file reader plus a csv parser called csv-parser

```
1 const mongodb = require('mongodb');
2 const MongoClient = mongodb.MongoClient;
3 const url = 'mongodb+srv://dajobez:eiB3bZqI2bclH1Pd@hcs.tcgbp.mongodb.net/myFirstDatabase?retryWrites=true&w=majority';
4 const csv = require('csv-parser');
5 const fs = require('fs');
6
7 let patientsData = [];
8 let emails = [];
9
10 fs.createReadStream('patients.csv')
11   .pipe(csv({ separator: '|' }))
12   .on('data', (data) => patientsData.push(data))
13   .on('end', () => {
14     let curDate = new Date();
15     patientsData.filter(patient => { return patient["CONSENT"] === "Y"}).forEach(filteredPatient => {
16       emails.push({
17         'Name': "Day 1",
18         'scheduled_date': curDate.getFullYear() + '-' + curDate.getMonth() + '-' + (curDate.getDate() + 1),
19         'email': filteredPatient['Email Address']
20       }),
21       {
22         'Name': "Day 2",
23         'scheduled_date': curDate.getFullYear() + '-' + curDate.getMonth() + '-' + (curDate.getDate() + 2),
24         'email': filteredPatient['Email Address']
25       },
26       {
27         'Name': "Day 3",
28         'scheduled_date': curDate.getFullYear() + '-' + curDate.getMonth() + '-' + (curDate.getDate() + 3),
29         'email': filteredPatient['Email Address']
30       },
31       {
32         'Name': "Day 4",
33         'scheduled_date': curDate.getFullYear() + '-' + curDate.getMonth() + '-' + (curDate.getDate() + 4),
34         'email': filteredPatient['Email Address']
35       }
36     });
37   });
38 //loading patient data
```

```

//loading patient data
MongoClient.connect(url).then(result => {
  var dbo = result.db("challenge");
  dbo.collection("Patients").insertMany(patientsData, (err, res) => {
    if (err) throw err;
    console.log(res.insertedCount);
    result.close();
  });
}).catch(err => {
  console.log(err);
});

//loading email data
MongoClient.connect(url).then(result => {
  var dbObject = result.db("challenge");
  dbObject.collection("Emails").insertMany(emails, (err, res) => {
    if (err) throw err;
    console.log(res.insertedCount);
    result.close();
  });
}).catch(err => {
  console.log(err);
});

```

Here's the mongodb view

The screenshot displays the MongoDB Atlas web interface. On the left is a sidebar with navigation options: DEPLOYMENT, Database (selected), Data Lake, DATA SERVICES (with Triggers and Data API sub-items), and SECURITY (with Database Access, Network Access, and Advanced sub-items). The Data API sub-item is highlighted with a 'PREVIEW' badge. The main panel shows the 'challenge' database with two collections: 'Emails' and 'Patients'. The 'Emails' collection is selected, showing its details: STORAGE SIZE: 20KB, TOTAL DOCUMENTS: 32, and INDEXES TOTAL SIZE: 20KB. Below this, there are tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A filter bar is present with the text '{ field: 'value' }'. The query results section shows 'QUERY RESULTS 1-20 OF MANY' and displays two document snippets. The first document has fields: _id: ObjectId("6269a19f1fc7002f482e7026"), Name: "Day 1", scheduled_date: "2022-3-28", and email: "test0@humancaresystems.com". The second document has fields: _id: ObjectId("6269a19f1fc7002f482e7027"), Name: "Day 2", and scheduled_date: "2022-3-29". At the bottom, there are 'PREVIOUS' and 'NEXT' navigation buttons, and a status '1-20 of many results'. A chat icon is visible in the bottom right corner.

DEPLOYMENT

Database

Data Lake

DATA SERVICES

Triggers

Data API PREVIEW

SECURITY

Database Access

Network Access

Advanced

DATABASES: 1 COLLECTIONS: 2

+ Create Database

NAMESPACES

challenge

Emails

Patients

challenge.Patients

STORAGE SIZE: 40KB TOTAL DOCUMENTS: 90 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER

{ field: 'value' }

OPTIONS

Apply

Reset

QUERY RESULTS 1-20 OF MANY

```

_id: ObjectId("62699100323b50733a1282e7")
Program Identifier: "50777445"
Data Source: "WEB 3RD PARTY"
Card Number: "342121211"
Member ID: "43233"
First Name: "LOAD"
Last Name: "TEST 0"
Date of Birth: "04/29/2000"
Address 1: "3100 S Ashley Drive"
Address 2: ""
City: "Chandler"
State: "AZ"

```

PREVIOUS

1-20 of many results

NEXT

And here's the automation process

```

danielbecerra@MacBook-Pro hcs_challenge % npm run automation

> hcs_challenge@1.0.0 automation
> node printing.js && npm run test

Printing ID with no email but consent is Y: 62699100323b50733a1282f2, 62699151fcbf3f3f9c0b6f3e, 626991af15b70935affd1f6f, 626992e596b514b5873cc1ce, 6269933c5c570536e89b9251
Print ID were first name is empty 62699100323b50733a1282ee, 62699100323b50733a1282f1, 62699151fcbf3f3f9c0b6f3a, 62699151fcbf3f3f9c0b6f3d, 626991af15b70935affd1f6b, 626991af15b70935affd1f6e, 626992e596b514b5873cc1ca, 626992e596b514b5873cc1cd, 6269933c5c570536e89b924d, 6269933c5c570536e89b9250

> hcs_challenge@1.0.0 test
> jest

PASS ./test.spec.js
Automation and Test Suite
  ✓ Verify the data in flat file matches the data in Patients collection. (2 ms)
  ✓ Verify Emails were created in Emails Collection for patients who have CONSENT as Y. (10 ms)
  ✓ Verify emails for each patient are scheduled correctly. (3 ms)

Test Suites: 1 passed, 1 total
Tests: 3 passed, 3 total
Snapshots: 0 total
Time: 0.673 s, estimated 1 s
Ran all test suites.

```