Processing Medical Records in Real Time

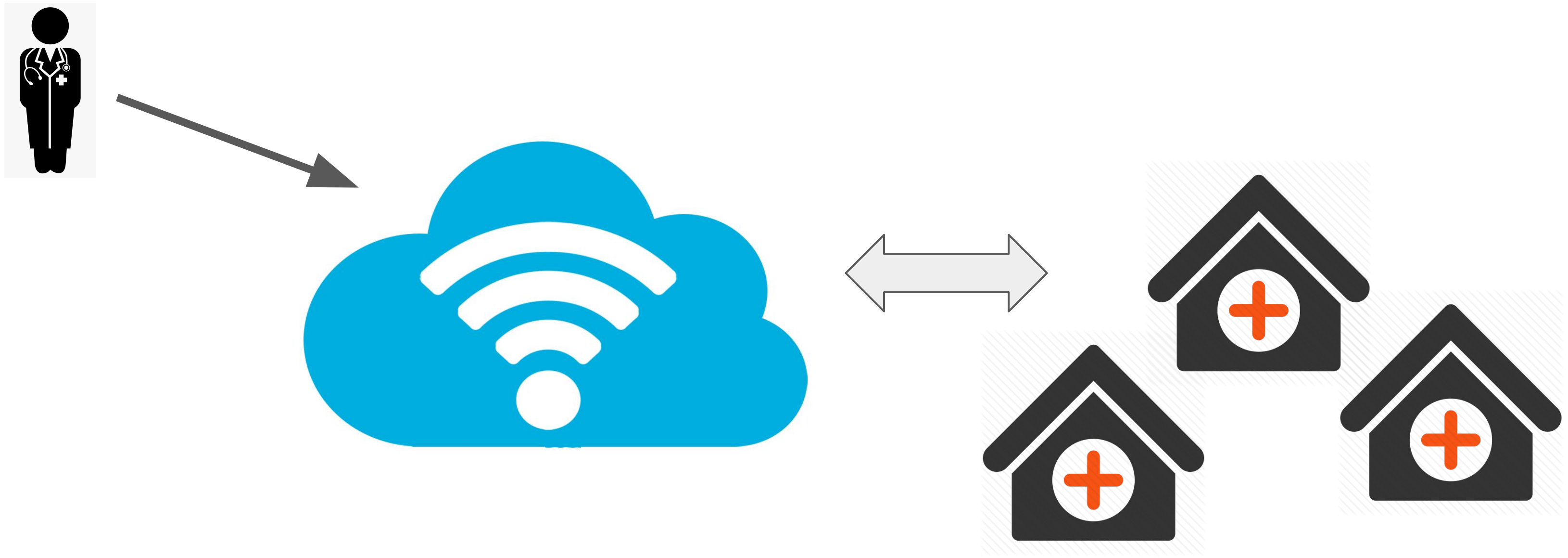
Presenters: Sandip Panesar, Piotr Parkitny

Problem Statement

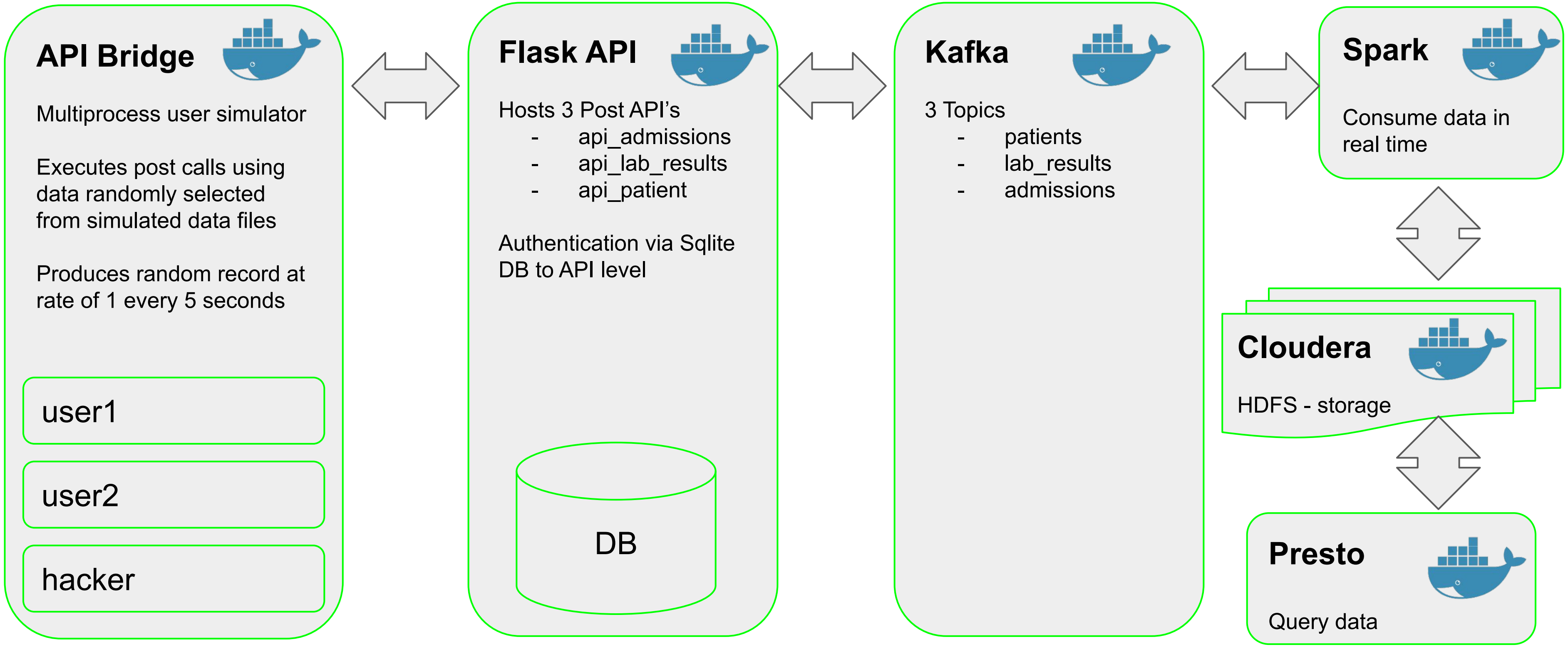
- Compiling large, up-to-date, secure and centralized repositories of health records (for future ML application) is a major challenge.

Proposed Solution

A system capable of pulling health records from various clinical sources (hospitals, clinics, labs etc.), in real time, capable of processing and storing them in a queryable database.



Pipeline Overview



API Bridge

Multiprocess user simulator

Executes post calls using data randomly selected from simulated data files

Produces random record at rate of 1 every 5 seconds

user1

user2

hacker

Flask API

Hosts 3 Post API's
- api_admissions
- api_lab_results
- api_patient

Authentication via Sqlite DB to API level

DB

Kafka

3 Topics
- patients
- lab_results
- admissions

Spark

Consume data in real time

Cloudera

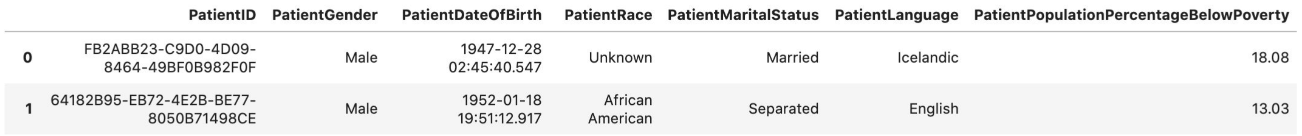
HDFS - storage

Presto

Query data

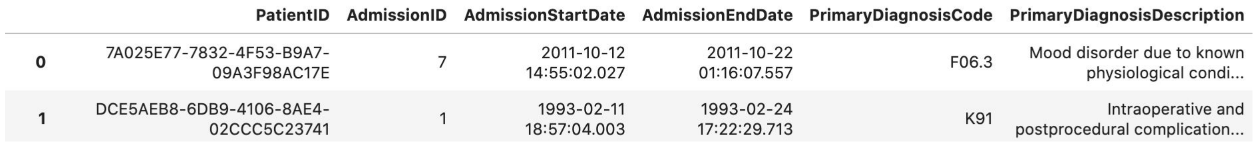
API Bridge: Simulating Submissions

- Randomly Read Records from files and uses POST request to submit:
  - Patient File (10k+ Records)



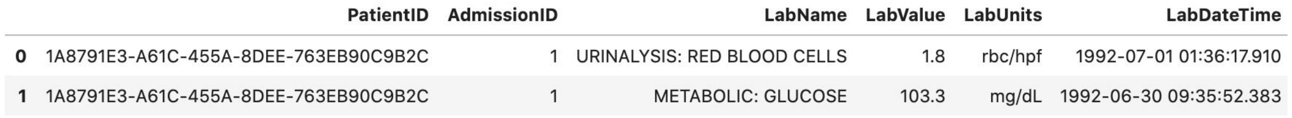
	PatientID	PatientGender	PatientDateOfBirth	PatientRace	PatientMaritalStatus	PatientLanguage	PatientPopulationPercentageBelowPoverty
0	FB2ABB23-C9D0-4D09-8464-49BF0B982F0F	Male	1947-12-28 02:45:40.547	Unknown	Married	Icelandic	18.08
1	64182B95-EB72-4E2B-BE77-8050B71498CE	Male	1952-01-18 19:51:12.917	African American	Separated	English	13.03

  - Admission File (36k+ Records)



	PatientID	AdmissionID	AdmissionStartDate	AdmissionEndDate	PrimaryDiagnosisCode	PrimaryDiagnosisDescription
0	7A025E77-7832-4F53-B9A7-09A3F98AC17E	7	2011-10-12 14:55:02.027	2011-10-22 01:16:07.557	F06.3	Mood disorder due to known physiological condi...
1	DCE5AEB8-6DB9-4106-8AE4-02CCC5C23741	1	1993-02-11 18:57:04.003	1993-02-24 17:22:29.713	K91	Intraoperative and postprocedural complication...

  - Lab Results File (11k+ Records)



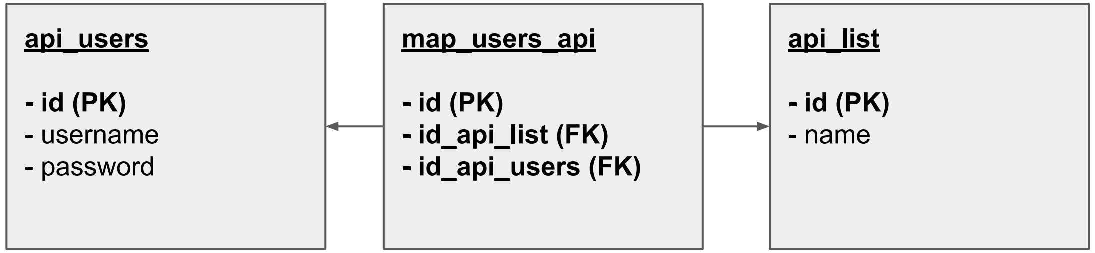
	PatientID	AdmissionID	LabName	LabValue	LabUnits	LabDateTime
0	1A8791E3-A61C-455A-8DEE-763EB90C9B2C	1	URINALYSIS: RED BLOOD CELLS	1.8	rbc/hpf	1992-07-01 01:36:17.910
1	1A8791E3-A61C-455A-8DEE-763EB90C9B2C	1	METABOLIC: GLUCOSE	103.3	mg/dL	1992-06-30 09:35:52.383

- Executes : *r = requests.post(url,auth,headers,timeout,json)*

Flask API

- JSON load via POST
- API
    - Add Patient (user1, user2)
    - Add Admissions (user1, hacker)
    - Add Lab Results (user1, user2)
- API Authentication: Credentials stored in SQLite DB

API Authentication: SQLite DB



Kafka

- Create 3 unique topics:
  - Patient
  - Admissions
  - Lab results
- Get Messages from Flask API

Spark Streaming Jobs

- Executes every 10 seconds
- Stream data from the 3 Kafka topics:
    - Patients, admissions, lab results
    - Data received in .json format.
- Read raw message and clean it.
- Force schema (custom for each type).
- ‘Patients’ category: Undefined PatientRace or PatientLanguage - reject record.
- Write to parquet

Hive

- Hive is used to create tables:
  - admissions
  - lab_results
  - patients
- *Execute: docker-compose exec cloudera hive -f /w205/table_name.sql*

create_admissions_table.sql

```
create external table if not exists default.admissions (
raw_event string,
timestamp string,
patientid string,
AdmissionID string,
AdmissionStartDate string,
AdmissionEndDate string,
PrimaryDiagnosisCode string,
PrimaryDiagnosisDescription string
) stored as parquet location '/tmp/admissions_rcd'
tblproperties ("parquet.compress"="SNAPPY");
```

create_lab_results_table.sql

```
create external table if not exists default.lab_results (
raw_event string,
timestamp string,
patientid string,
AdmissionID string,
LabName string,
LabValue string,
LabUnits string,
LabDateTime string
) stored as parquet location '/tmp/lab_results_rcd'
tblproperties ("parquet.compress"="SNAPPY");
```

create_patients_table.sql

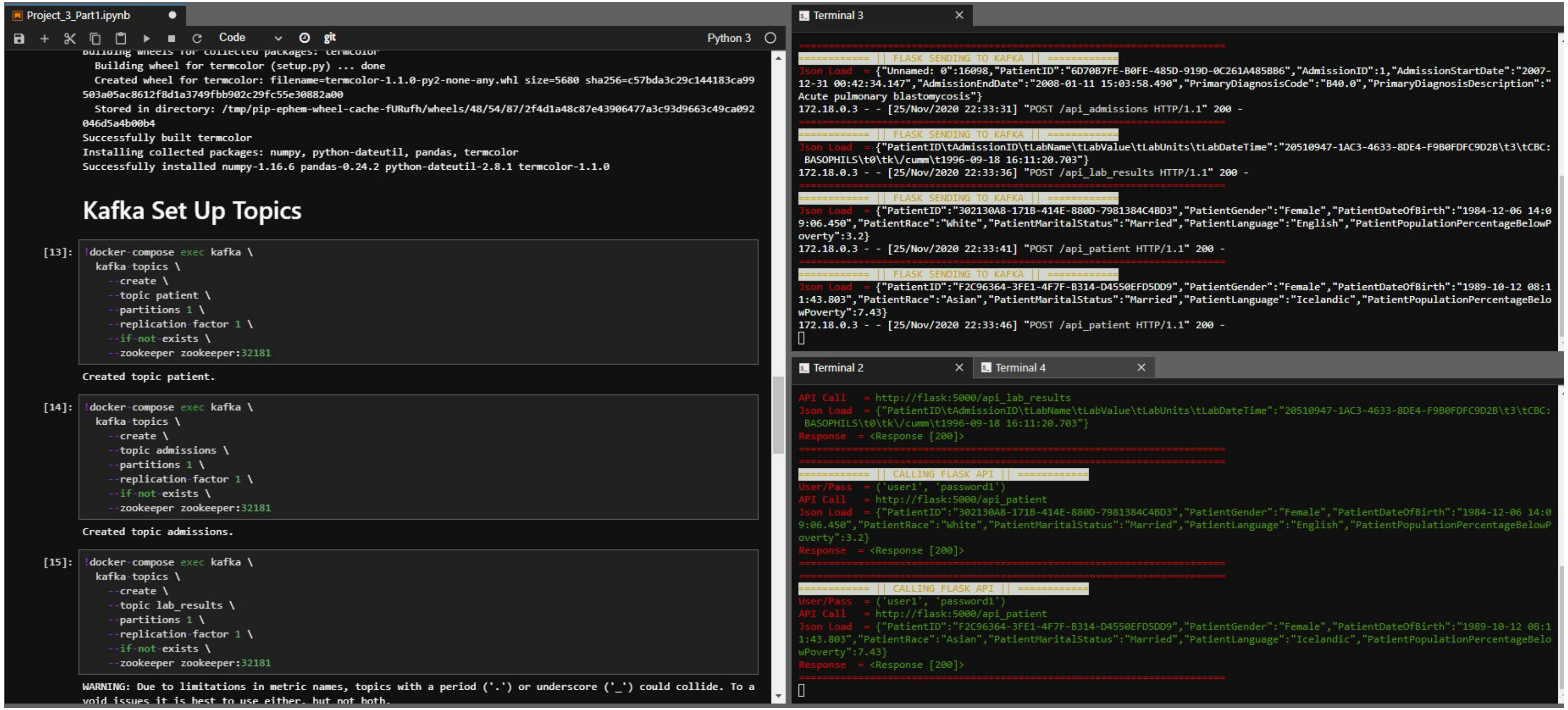
```
create external table if not exists default.patients (
raw_event string,
timestamp string,
patientid string,
patientgender string,
patientdateofbirth string,
patientrace string,
patientmaritalstatus string,
patientlanguage string,
patientpopulationpercentagebelowpoverty string
) stored as parquet location '/tmp/patients_rcd'
tblproperties ("parquet.compress"="SNAPPY");
```

- Once tables are created they can be queried using Presto

Questions

- How many male and female patients are there?
  - M: 500, F: 536
- What is the most common language spoken by patients?
  - English
- What is the most frequent reason for hospital visits?
  - Kaschin-Beck disease
- How many different types of test are there?
  - 35
- What is the most common test type?
  - URINALYSIS: PH

Live Demo



Project_3_Part1.ipynb

```
Building wheels for collected packages: termcolor
  Building wheel for termcolor (setup.py) ... done
  Created wheel for termcolor: filename=termcolor-1.1.0-py2-none-any.whl size=5680 sha256=c57bda3c29c144183ca99
503a05ac8612f8d1a3749fbb902c29fc55e30882a00
  Stored in directory: /tmp/pip-ephem-wheel-cache-fURufh/wheels/48/54/87/2f4d1a48c87e43906477a3c93d9663c49ca092
046d5a4b00b4
Successfully built termcolor
Installing collected packages: numpy, python-dateutil, pandas, termcolor
Successfully installed numpy-1.16.6 pandas-0.24.2 python-dateutil-2.8.1 termcolor-1.1.0
```

Kafka Set Up Topics

```
[13]: !docker-compose exec kafka \
        kafka-topics \
          --create \
          --topic patient \
          --partitions 1 \
          --replication-factor 1 \
          --if-not-exists \
          --zookeeper zookeeper:32181

Created topic patient.

[14]: !docker-compose exec kafka \
        kafka-topics \
          --create \
          --topic admissions \
          --partitions 1 \
          --replication-factor 1 \
          --if-not-exists \
          --zookeeper zookeeper:32181

Created topic admissions.

[15]: !docker-compose exec kafka \
        kafka-topics \
          --create \
          --topic lab_results \
          --partitions 1 \
          --replication-factor 1 \
          --if-not-exists \
          --zookeeper zookeeper:32181

WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To a
void issues it is best to use either, but not both.
```

Terminal 3

```
========================================================================
============ || FLASK SENDING TO KAFKA || ============
Json Load  = {"Unnamed: 0":16098,"PatientID":"6D70B7FE-B0FE-485D-919D-0C261A485BB6","AdmissionID":1,"AdmissionStartDate":"2007-
12-31 00:42:34.147","AdmissionEndDate":"2008-01-11 15:03:58.490","PrimaryDiagnosisCode":"B40.0","PrimaryDiagnosisDescription":"
Acute pulmonary blastomycosis"}
172.18.0.3 - - [25/Nov/2020 22:33:31] "POST /api_admissions HTTP/1.1" 200 -
========================================================================
============ || FLASK SENDING TO KAFKA || ============
Json Load  = {"PatientID\tAdmissionID\tLabName\tLabValue\tLabUnits\tLabDateTime":"20510947-1AC3-4633-8DE4-F9B0FDFC9D2B\t3\tCBC:
 BASOPHILS\t0\tk\/cumm\t1996-09-18 16:11:20.703"}
172.18.0.3 - - [25/Nov/2020 22:33:36] "POST /api_lab_results HTTP/1.1" 200 -
========================================================================
============ || FLASK SENDING TO KAFKA || ============
Json Load  = {"PatientID":"302130A8-171B-414E-880D-7981384C4BD3","PatientGender":"Female","PatientDateOfBirth":"1984-12-06 14:0
9:06.450","PatientRace":"White","PatientMaritalStatus":"Married","PatientLanguage":"English","PatientPopulationPercentageBelowP
overty":3.2}
172.18.0.3 - - [25/Nov/2020 22:33:41] "POST /api_patient HTTP/1.1" 200 -
========================================================================
============ || FLASK SENDING TO KAFKA || ============
Json Load  = {"PatientID":"F2C96364-3FE1-4F7F-B314-D4550EFD5DD9","PatientGender":"Female","PatientDateOfBirth":"1989-10-12 08:1
1:43.803","PatientRace":"Asian","PatientMaritalStatus":"Married","PatientLanguage":"Icelandic","PatientPopulationPercentageBelo
wPoverty":7.43}
172.18.0.3 - - [25/Nov/2020 22:33:46] "POST /api_patient HTTP/1.1" 200 -
▯
```

Terminal 2

Terminal 4

```
API Call   = http://flask:5000/api_lab_results
Json Load  = {"PatientID\tAdmissionID\tLabName\tLabValue\tLabUnits\tLabDateTime":"20510947-1AC3-4633-8DE4-F9B0FDFC9D2B\t3\tCBC:
 BASOPHILS\t0\tk\/cumm\t1996-09-18 16:11:20.703"}
Response  = <Response [200]>
========================================================================
========================================================================
============ || CALLING FLASK API || ============
User/Pass  = ('user1', 'password1')
API Call   = http://flask:5000/api_patient
Json Load  = {"PatientID":"302130A8-171B-414E-880D-7981384C4BD3","PatientGender":"Female","PatientDateOfBirth":"1984-12-06 14:0
9:06.450","PatientRace":"White","PatientMaritalStatus":"Married","PatientLanguage":"English","PatientPopulationPercentageBelowP
overty":3.2}
Response  = <Response [200]>
========================================================================
========================================================================
============ || CALLING FLASK API || ============
User/Pass  = ('user1', 'password1')
API Call   = http://flask:5000/api_patient
Json Load  = {"PatientID":"F2C96364-3FE1-4F7F-B314-D4550EFD5DD9","PatientGender":"Female","PatientDateOfBirth":"1989-10-12 08:1
1:43.803","PatientRace":"Asian","PatientMaritalStatus":"Married","PatientLanguage":"Icelandic","PatientPopulationPercentageBelo
wPoverty":7.43}
Response  = <Response [200]>
========================================================================
▯
```

Thank You

Any Questions ?