Heart Failure & Stroke Database & API

ETL Project - Group 7

Team Members- Gabe, Noah, Bailey, Renuka

Overview

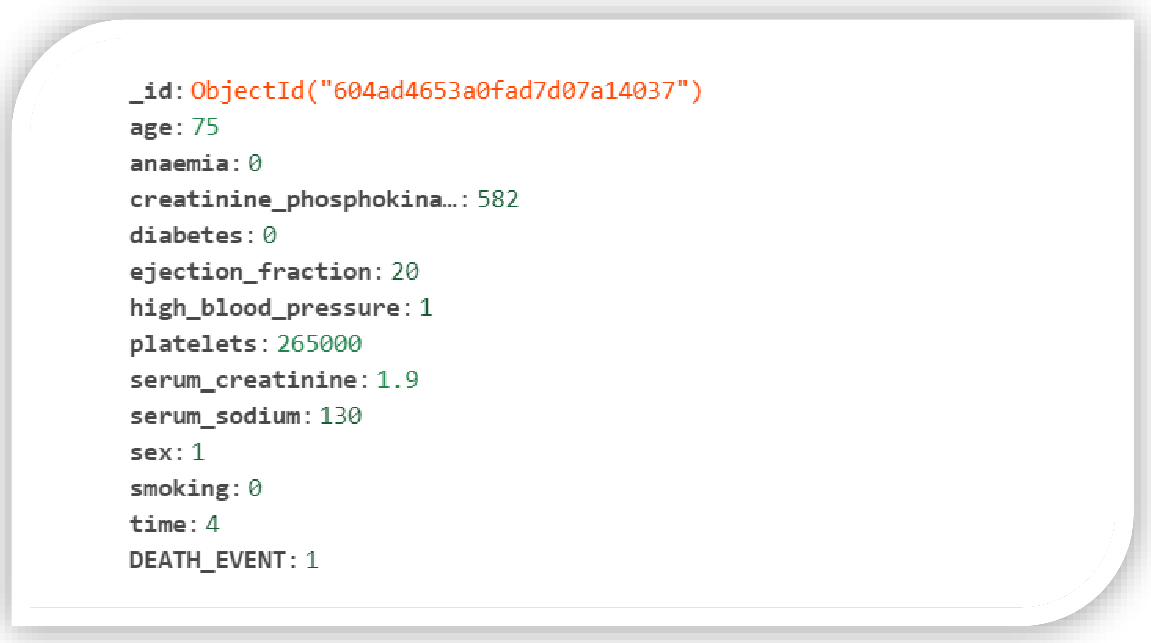
According to the Centers for Disease Control and Preventions' National Center for Health Statistics, heart disease and stroke are the first and fifth most common cause of death in the United States. The aim of this project is to help inform the public of common heart and brain health metrics to be aware of, by generating a user-friendly API that can be queried by both medical professionals and laymen alike. Due to the disparate nature of the data collected between each dataset, a Non-Relational (NoSQL) database has been selected to store the data for ease of use and portability.

Data Sources:

Stroke Prediction Dataset | Kaggle - Datasets containing 11 clinical features for predicting stroke events.

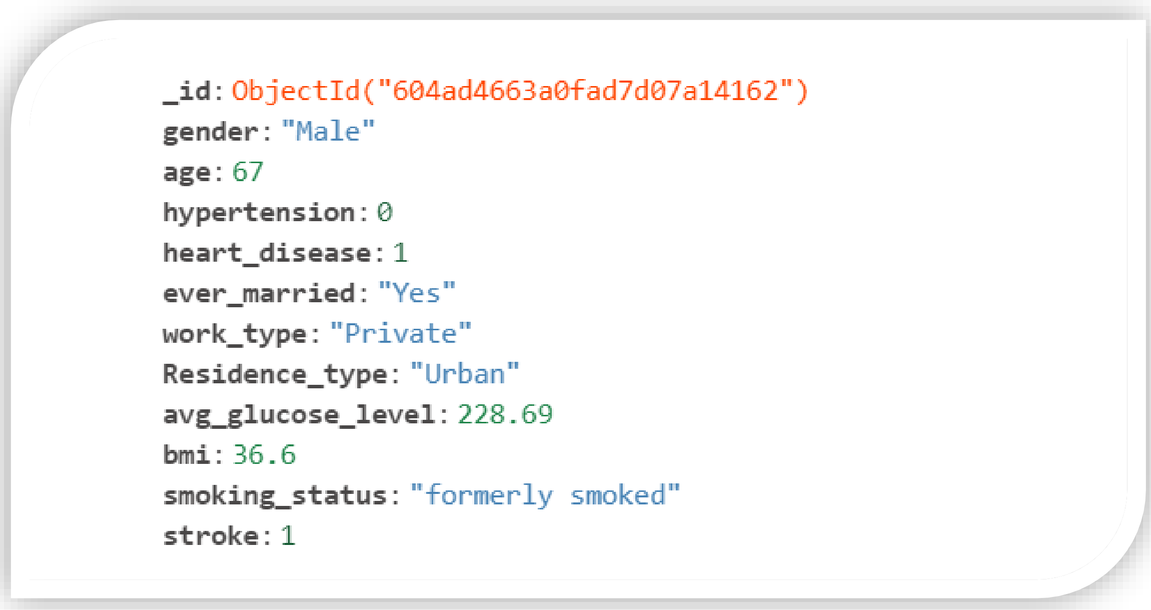
Heart Failure Dataset - Data set for Heart Failure

Data Attributes:



```
_id: ObjectId("604ad4653a0fad7d07a14037")
age: 75
anaemia: 0
creatinine_phosphokina…: 582
diabetes: 0
ejection_fraction: 20
high_blood_pressure: 1
platelets: 265000
serum_creatinine: 1.9
serum_sodium: 130
sex: 1
smoking: 0
time: 4
DEATH_EVENT: 1
```

Figure 1: Screenshot from MongoDB Heart_Failure_Record Collection

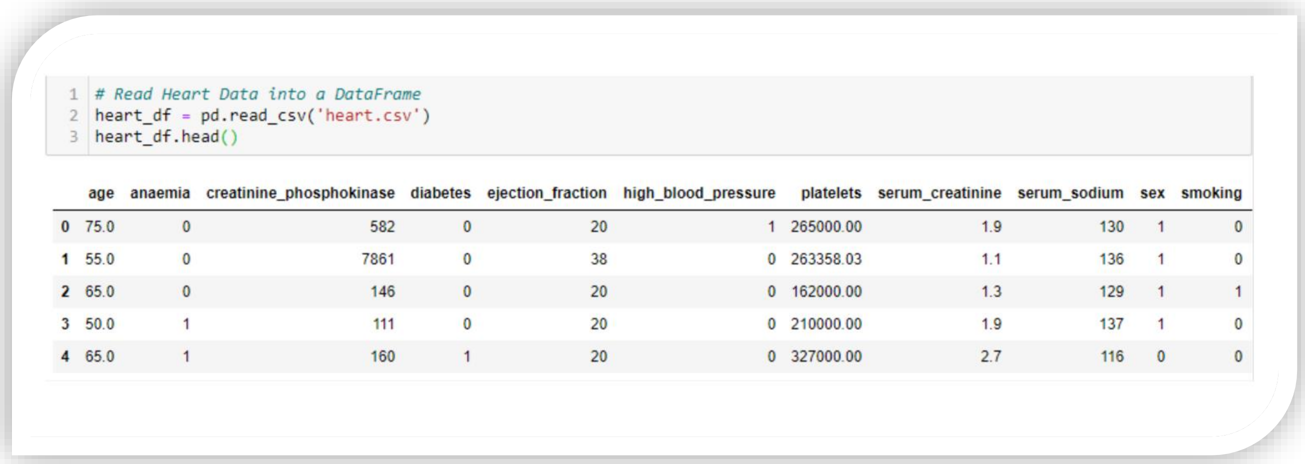


```
_id: ObjectId("604ad4663a0fad7d07a14162")
gender: "Male"
age: 67
hypertension: 0
heart_disease: 1
ever_married: "Yes"
work_type: "Private"
Residence_type: "Urban"
avg_glucose_level: 228.69
bmi: 36.6
smoking_status: "formerly smoked"
stroke: 1
```

Figure 2: Screenshot from MongoDB Stroke Collection

ETL Process Summary

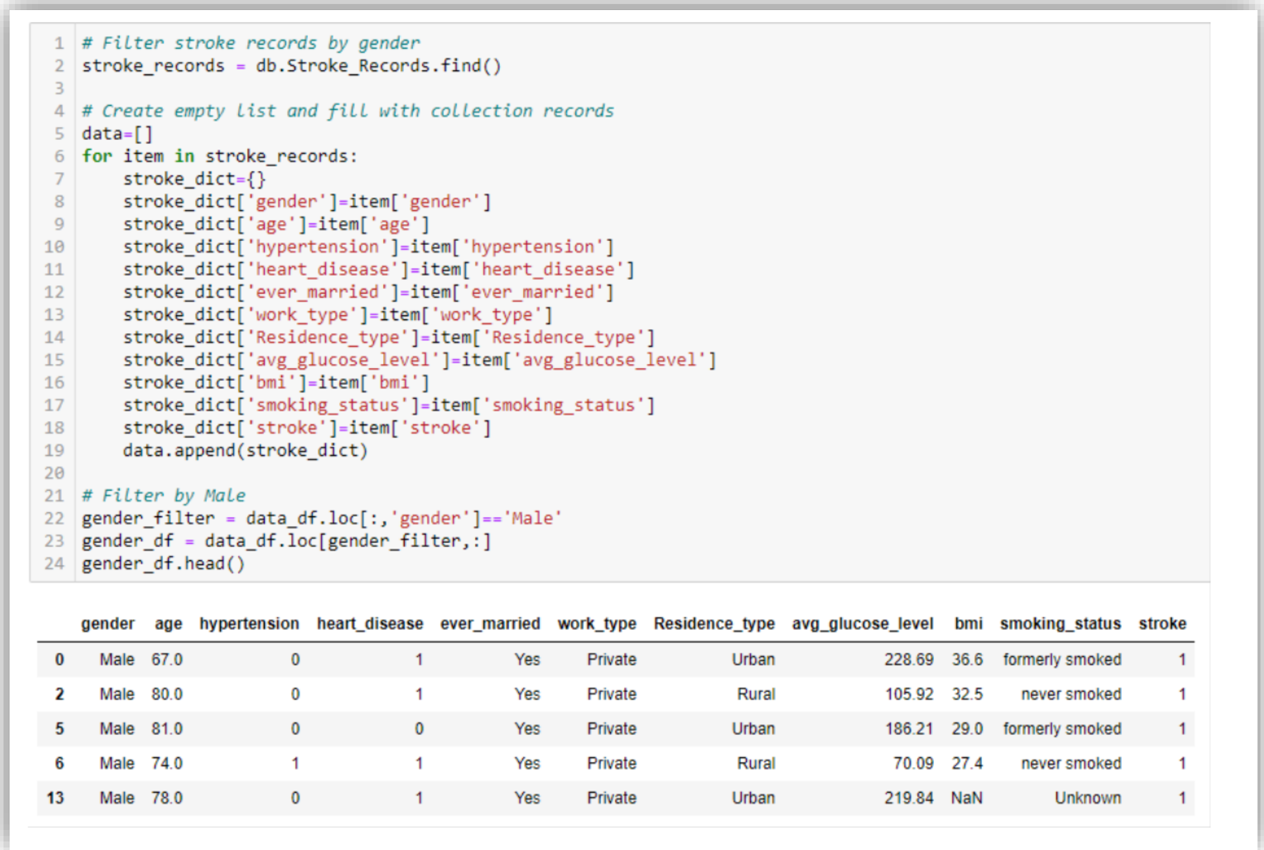
EXTRACT: Each dataset was downloaded from Kaggle.com in CSV format and subsequently loaded into a Jupyter Notebook for data munging and cleaning as required.



```
# Read Heart Data into a DataFrame
heart_df = pd.read_csv('heart.csv')
heart_df.head()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0

<u>Transform:</u> Data has been grouped by key categorical variables to produce common aggregated queries across each dataset.



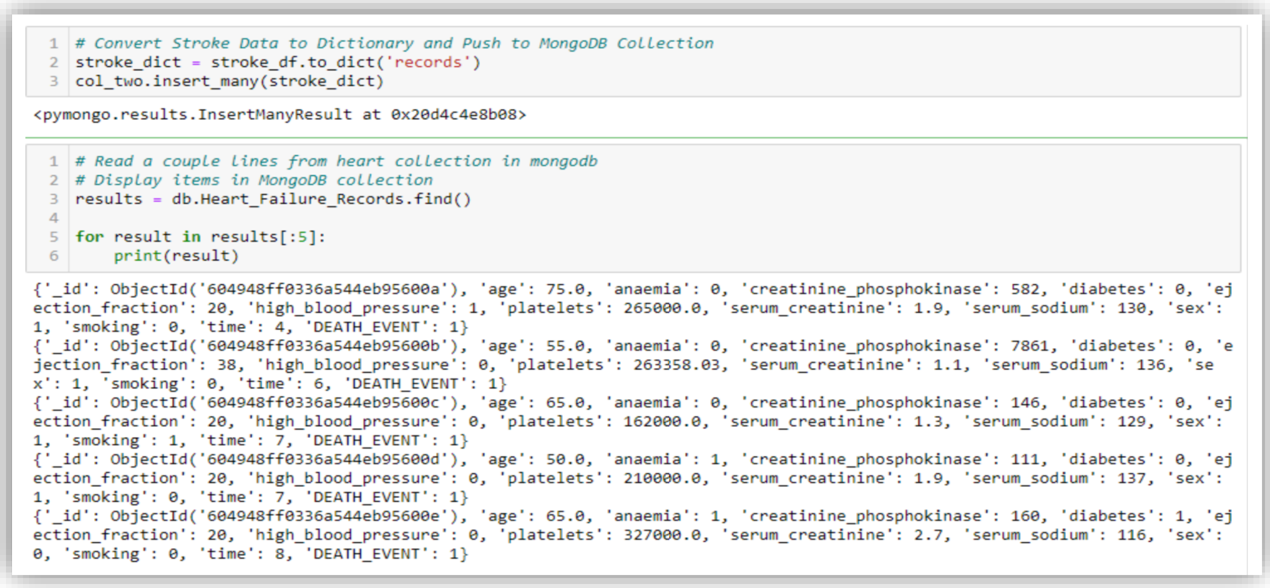
```python
# Filter stroke records by gender
stroke_records = db.Stroke_Records.find()

# Create empty list and fill with collection records
data=[]
for item in stroke_records:
    stroke_dict={}
    stroke_dict['gender']=item['gender']
    stroke_dict['age']=item['age']
    stroke_dict['hypertension']=item['hypertension']
    stroke_dict['heart_disease']=item['heart_disease']
    stroke_dict['ever_married']=item['ever_married']
    stroke_dict['work_type']=item['work_type']
    stroke_dict['Residence_type']=item['Residence_type']
    stroke_dict['avg_glucose_level']=item['avg_glucose_level']
    stroke_dict['bmi']=item['bmi']
    stroke_dict['smoking_status']=item['smoking_status']
    stroke_dict['stroke']=item['stroke']
    data.append(stroke_dict)

# Filter by Male
gender_filter = data_df.loc[:,'gender']=='Male'
gender_df = data_df.loc[gender_filter,:]
gender_df.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
5	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
6	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
13	Male	78.0	0	1	Yes	Private	Urban	219.84	NaN	Unknown	1

LOAD: Using the pymongo module available for Python, insert raw and transformed tables into MongoDB to serve as a backend for a Flask API.



```
# Convert Stroke Data to Dictionary and Push to MongoDB Collection
stroke_dict = stroke_df.to_dict('records')
col_two.insert_many(stroke_dict)
```

```
<pymongo.results.InsertManyResult at 0x20d4c4e8b08>
```

```
# Read a couple lines from heart collection in mongodb
# Display items in MongoDB collection
results = db.Heart_Failure_Records.find()

for result in results[:5]:
    print(result)
```

```
{'_id': ObjectId('604948ff0336a544eb95600a'), 'age': 75.0, 'anaemia': 0, 'creatinine_phosphokinase': 582, 'diabetes': 0, 'ej
ection_fraction': 20, 'high_blood_pressure': 1, 'platelets': 265000.0, 'serum_creatinine': 1.9, 'serum_sodium': 130, 'sex':
1, 'smoking': 0, 'time': 4, 'DEATH_EVENT': 1}
{'_id': ObjectId('604948ff0336a544eb95600b'), 'age': 55.0, 'anaemia': 0, 'creatinine_phosphokinase': 7861, 'diabetes': 0, 'e
jection_fraction': 38, 'high_blood_pressure': 0, 'platelets': 263358.03, 'serum_creatinine': 1.1, 'serum_sodium': 136, 'se
x': 1, 'smoking': 0, 'time': 6, 'DEATH_EVENT': 1}
{'_id': ObjectId('604948ff0336a544eb95600c'), 'age': 65.0, 'anaemia': 0, 'creatinine_phosphokinase': 146, 'diabetes': 0, 'ej
ection_fraction': 20, 'high_blood_pressure': 0, 'platelets': 162000.0, 'serum_creatinine': 1.3, 'serum_sodium': 129, 'sex':
1, 'smoking': 1, 'time': 7, 'DEATH_EVENT': 1}
{'_id': ObjectId('604948ff0336a544eb95600d'), 'age': 50.0, 'anaemia': 1, 'creatinine_phosphokinase': 111, 'diabetes': 0, 'ej
ection_fraction': 20, 'high_blood_pressure': 0, 'platelets': 210000.0, 'serum_creatinine': 1.9, 'serum_sodium': 137, 'sex':
1, 'smoking': 0, 'time': 7, 'DEATH_EVENT': 1}
{'_id': ObjectId('604948ff0336a544eb95600e'), 'age': 65.0, 'anaemia': 1, 'creatinine_phosphokinase': 160, 'diabetes': 1, 'ej
ection_fraction': 20, 'high_blood_pressure': 0, 'platelets': 327000.0, 'serum_creatinine': 2.7, 'serum_sodium': 116, 'sex':
0, 'smoking': 0, 'time': 8, 'DEATH_EVENT': 1}
```

API Documentation

An **application programming interface** (**API**) is a computing interface that defines interactions between multiple software or mixed hardware-software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. It can also provide extension mechanisms so that users can extend existing functionality in various ways and to varying degrees.[1]

The Heart Failure and Stroke Database API contains six easily accessible routes that can handle requests without the need for an access key or credentials, it is fully open sourced. The following is a list of the available API routes along with utilization parameters and output examples.

- *Below route returns all heart records from database:*
    - /api/v1.0/Heart_Failure_Records

- *Below route returns all stroke records from database:*
    - /api/v1.0/Stroke_Records

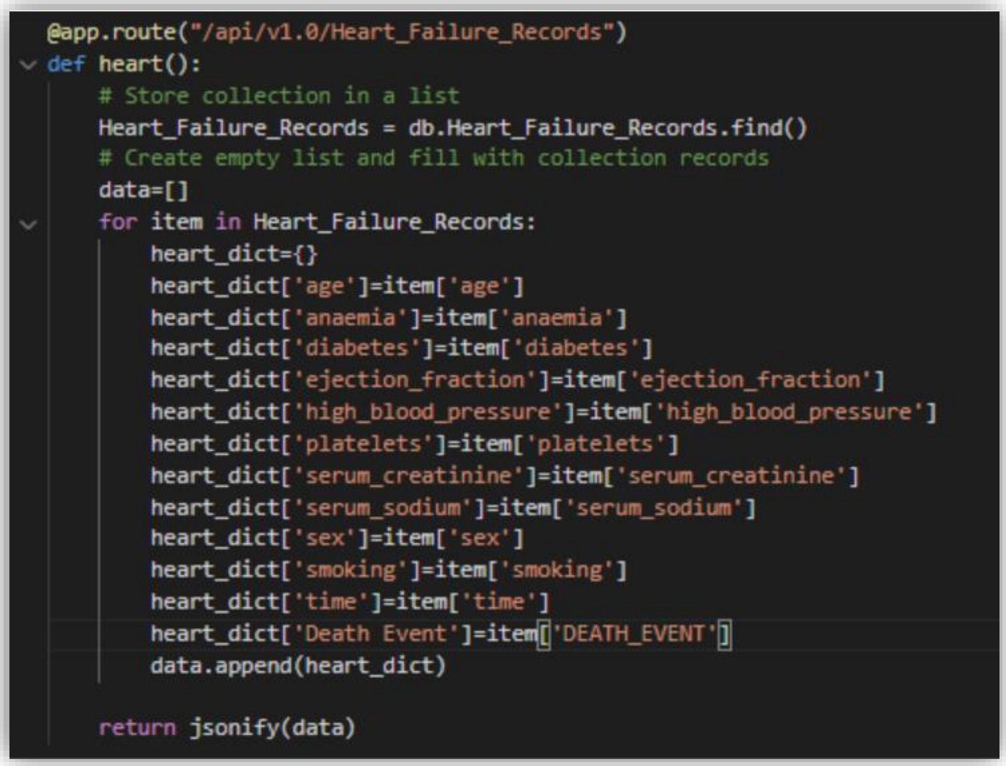
- *Below route returns heart failure death by mean age | Death = 1, Non-Death = 0:*
    - /api/v1.0/Heart_Failure_by_Age

- *Below route returns strokes by mean age | Stroke = 1, no stroke = 0:*
    - /api/v1.0/Stroke_by_Age

- *Below route returns heart failure data by smoking or not | Input 1 for Smoker, 0 for Non-Smoker:*
    - /api/v1.0/Heart_Failure_by_Smoking/

- *Below route returns stroke data by gender | Input Male or Female:*
    - /api/v1.0/Stroke_by_Gender/

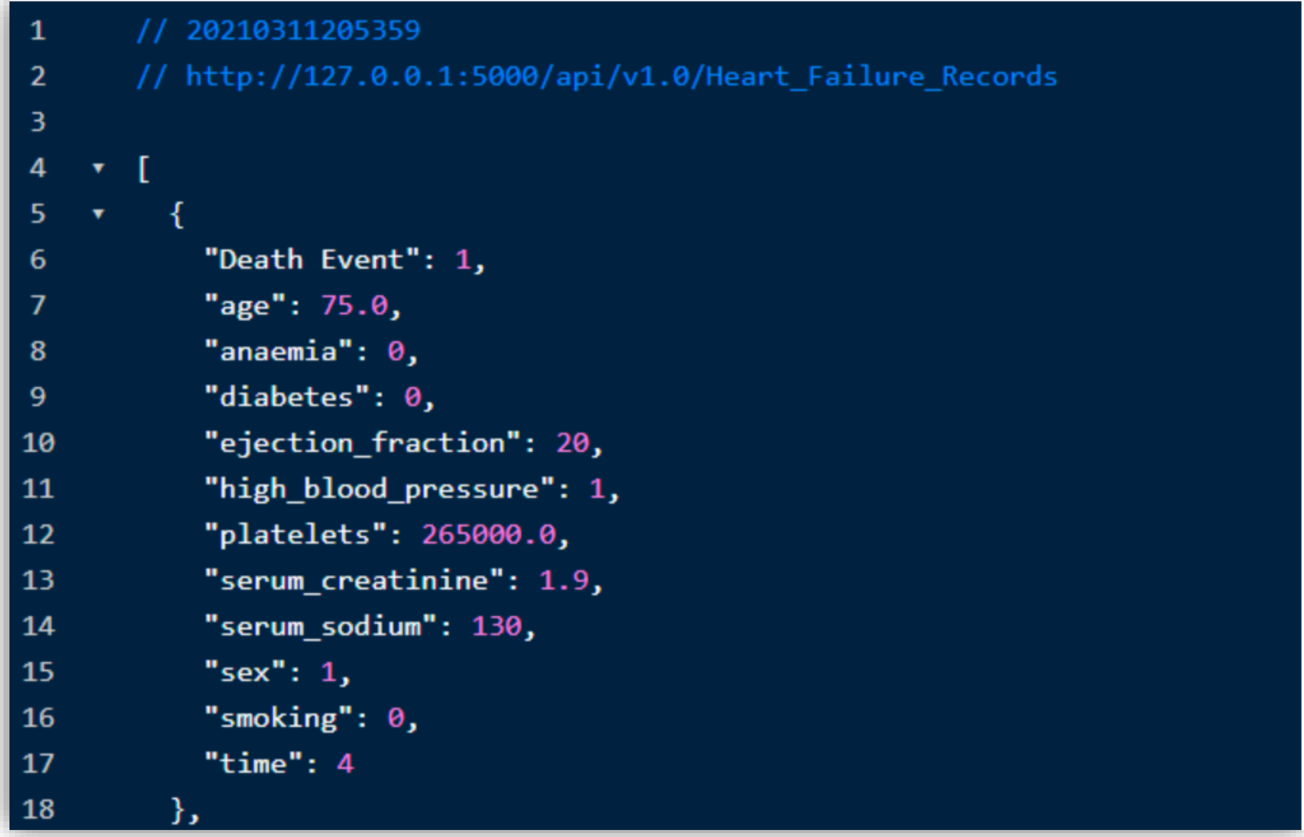
Example Flask API Route



```
@app.route("/api/v1.0/Heart_Failure_Records")
def heart():
    # Store collection in a list
    Heart_Failure_Records = db.Heart_Failure_Records.find()
    # Create empty list and fill with collection records
    data=[]
    for item in Heart_Failure_Records:
        heart_dict={}
        heart_dict['age']=item['age']
        heart_dict['anaemia']=item['anaemia']
        heart_dict['diabetes']=item['diabetes']
        heart_dict['ejection_fraction']=item['ejection_fraction']
        heart_dict['high_blood_pressure']=item['high_blood_pressure']
        heart_dict['platelets']=item['platelets']
        heart_dict['serum_creatinine']=item['serum_creatinine']
        heart_dict['serum_sodium']=item['serum_sodium']
        heart_dict['sex']=item['sex']
        heart_dict['smoking']=item['smoking']
        heart_dict['time']=item['time']
        heart_dict['Death Event']=item['DEATH_EVENT']
        data.append(heart_dict)

    return jsonify(data)
```

Example API Result Screen



```
// 20210311205359
// http://127.0.0.1:5000/api/v1.0/Heart_Failure_Records

[
  {
    "Death Event": 1,
    "age": 75.0,
    "anaemia": 0,
    "diabetes": 0,
    "ejection_fraction": 20,
    "high_blood_pressure": 1,
    "platelets": 265000.0,
    "serum_creatinine": 1.9,
    "serum_sodium": 130,
    "sex": 1,
    "smoking": 0,
    "time": 4
  },
```