**Evan Rixom**

**3/18/25**

**SAT5424**

**Part 6 - Aggregation and Visualization of Data for Disease Outbreak Surveillance**

**Question: What are two interesting insights you found within your results from using the script? ( 5 points)**

The first was noticing how it broke out between the suspected COVID cases and then confirmed with the same patient. I based my data on suspected COVID cases vs confirmed. Additionally, it was interesting to see how the data structures are broke out, specifically how the care plan links to the individual encounter and patient using an ID. Seeing the status vs completed on the careplan compared to the activity status as well.

**Link to a copy of publicly shared Google Looker Studio COVID Disease Outbreak  
Surveillance Page with authorization to view how you configured the data visualizations.**

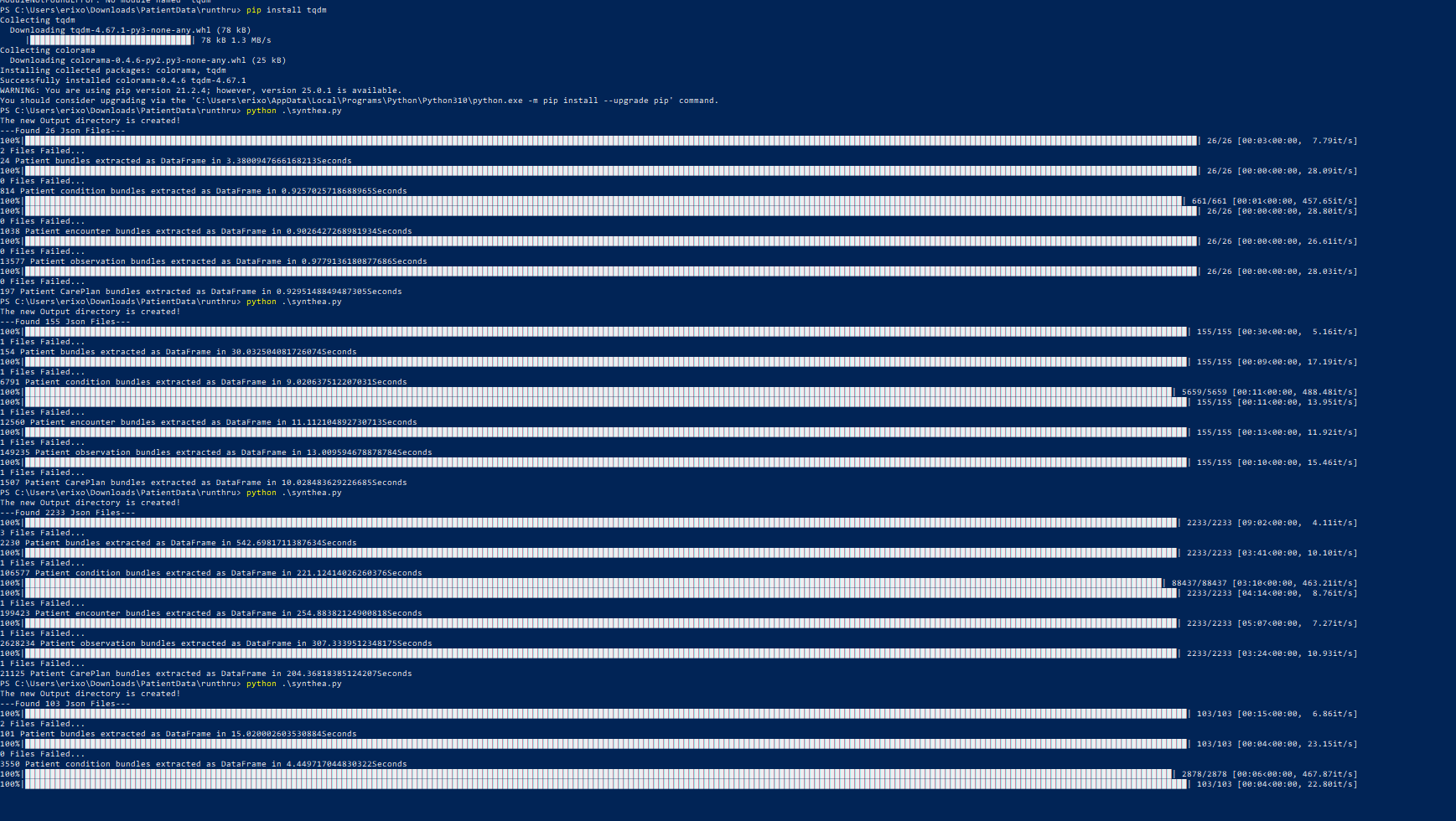
This has been shared with [hchangar@mtu.edu](mailto:hchangar@mtu.edu) and [ooodera@mtu.edu](mailto:ooodera@mtu.edu) – please let me know if it needs to be shared with anyone else. Link: <https://lookerstudio.google.com/reporting/e15b8a24-3be3-4d87-8da4-393668975729>

**Dashboard set to refresh every 15 minutes. (2 points)**

NOTE: Due to the data source type (CSV upload) it is impossible to set a refresh of 15 minutes – Looker Studio does not allow it. The dashboard will process the latest data as soon as it is uploaded.  **------------------------------------------------------------------------------  
Bonus: What was the most challenging portion of this lab? Please explain in detail. (2 points)**

The whole lab was tough. To begin, had issues getting the script to run, which took some time to resolve in my environment. Once I finally got the script functioning, ran into issues with Looker Studio. It was also very new so I had issues getting everything imported. I hit processing errors a few times, but finally got everything running. For the location, I was able to write a basic script to calculate the location.

**Output Screenshot:**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**Completed Python script illustrating output of commands based on student’s work of  
aggregating data using sample script to generate insights based on combined data.  
(12 points)**

# -\*- coding: utf-8 -\*-

"""Synthea\_Patient\_Data\_Aggregation\_Util

Original file is located at

https://colab.research.google.com/drive/1fqJf4Xva\_eXkw4TvpCGLHOXW4-UcjpOj

"""

##Necessary modules. If running locally - I used a pip install.

**import** pandas **as** pd

**import** numpy **as** np

**import** json

**import** glob

**from** tqdm **import** tqdm

**import** time

**import** os

**import** gc

## Configuring OS support

OPERATING\_SYS **=** 'Win'

# OPERATING\_SYS = 'Linux'

delim **=** '\\'

**if** OPERATING\_SYS **!=** 'Win'**:**

delim **=** '/'

#input and output

input\_root\_folder\_path **=** '.\\fhir'

output\_folder\_path **=** '.\\output'

##Create output directory if needed

**if** **not** os**.**path**.**exists**(**output\_folder\_path**):**

os**.**makedirs**(**output\_folder\_path**)**

**print(**"The new Output directory is created!"**)**

##Import files and filter based on resource

files **=** glob**.**glob**(**input\_root\_folder\_path**+**delim**+**'\*\*'**+**delim**+**'\*.json'**,**recursive**=True)**

**print(**'---Found '**+str(len(**files**))+**' Json Files---'**)**

**def** filter\_resource**(**data**,** resource\_type**):**

**return** **list(filter(lambda** x**:** x**[**'resource'**][**'resourceType'**]** **==** resource\_type**.**strip**(),** data**[**'entry'**]))**

##Below converts each resource to a CSV

"""## Patient"""

#configure necessary parameters

cols **=** **[**'id'**,**'gender'**,**'birthDate'**,**'maritalStatus'**,**'city'**,**'state'**,**'postalCode'**,**'country'**,**'deceased'**,**'deceasedDateTime'**]**

arr **=** **[]**

start **=** time**.**time**()**

f\_count **=** 0 #amount of files failed

#extract data from json file

**for** file **in** tqdm**(**files**):**

**try:**

#load File

f **=** **open(**file**)**

data **=** json**.**load**(**f**)**

f**.**close**()**

ar **=** **[]**

#extract data

patient **=** filter\_resource**(**data**,** 'Patient'**)[**0**]**

ar**.**append**(**patient**[**'resource'**][**'id'**])**

ar**.**append**(**patient**[**'resource'**][**'gender'**])**

ar**.**append**(**patient**[**'resource'**][**'birthDate'**])**

ar**.**append**(**patient**[**'resource'**][**'maritalStatus'**][**'text'**])**

ar**.**append**(**patient**[**'resource'**][**'address'**][**0**][**'city'**])**

ar**.**append**(**patient**[**'resource'**][**'address'**][**0**][**'state'**])**

ar**.**append**(**patient**[**'resource'**][**'address'**][**0**][**'postalCode'**])**

ar**.**append**(**patient**[**'resource'**][**'address'**][**0**][**'country'**])**

#is patient decesased?

**if** 'deceasedDateTime' **in** patient**[**'resource'**]:**

ar**.**append**(True)**

ar**.**append**(**patient**[**'resource'**][**'deceasedDateTime'**])**

**else:**

ar**.**append**(False)**

ar**.**append**(**np**.**nan**)**

arr**.**append**(**ar**)**

#file failed

**except** **Exception** **as** e**:**

f\_count **+=** 1

**continue**

end **=** time**.**time**()**

**print(str(**f\_count**)+**' Files Failed...'**)**

**print(str(len(**arr**))+**' Patient bundles extracted as DataFrame in '**+str(**end**-**start**)+** 'Seconds'**)**

#Place data into a data frame

df\_patient **=** pd**.**DataFrame**(**arr**,** columns **=** cols**)**

df\_patient**[**'city'**].**value\_counts**()**

df\_patient**[**'deceased'**].**value\_counts**()/len(**df\_patient**)**

#Dropping Duplicates If Any

df\_patient **=** df\_patient**.**drop\_duplicates**(**'id'**,**

inplace**=False,**

ignore\_index**=True)**

df\_patient**.**head**()**

#save to CSV

df\_patient**.**to\_csv**(**output\_folder\_path**+**delim**+**'Patient.csv'**)**

**del** df\_patient

gc**.**collect**()**v# memory cleanup.

#The same steps as outlined above follow below per resource. A progress bar processes, and memory cleanup occurs.

"""## Conditions"""

cols **=** **[**'code'**,**'codeText'**,**'patientId'**,**'encounterId'**,**'onsetDateTime'**,**'recordedDate'**,**'clinicalStatusCode'**]**

arr **=** **[]**

start **=** time**.**time**()**

f\_count **=** 0

**for** file **in** tqdm**(**files**):**

**try:**

#load File

f **=** **open(**file**)**

data **=** json**.**load**(**f**)**

f**.**close**()**

conditions **=** filter\_resource**(**data**,** 'Condition'**)**

**for** cond **in** conditions**:**

ar **=** **[]**

ar**.**append**(**cond**[**'resource'**][**'code'**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**cond**[**'resource'**][**'code'**][**'text'**])**

ar**.**append**(**cond**[**'resource'**][**'subject'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**cond**[**'resource'**][**'encounter'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**cond**[**'resource'**][**'onsetDateTime'**])**

ar**.**append**(**cond**[**'resource'**][**'recordedDate'**])**

ar**.**append**(**cond**[**'resource'**][**'clinicalStatus'**][**'coding'**][**0**][**'code'**])**

arr**.**append**(**ar**)**

**except** **Exception** **as** e**:**

# print(e)

f\_count **+=** 1

**continue**

end **=** time**.**time**()**

**print(str(**f\_count**)+**' Files Failed...'**)**

**print(str(len(**arr**))+**' Patient condition bundles extracted as DataFrame in '**+str(**end**-**start**)+** 'Seconds'**)**

df\_condition **=** pd**.**DataFrame**(**arr**,** columns **=** cols**)**

df\_condition**.**head**()**

df\_condition**[**'onsetDateTime'**]** **=** pd**.**to\_datetime**(**df\_condition**[**'onsetDateTime'**],** **format=**"%Y-%m-%dT%H:%M:%S%z"**,** utc**=True)**

df\_condition**[**'recordedDate'**]** **=** pd**.**to\_datetime**(**df\_condition**[**'recordedDate'**],** **format=**"%Y-%m-%dT%H:%M:%S%z"**,** utc**=True)**

#Extracting resolvedDateTime form Conditions DataFrame

cols **=** **[**'patientId'**,**'code'**,**'encounterId'**,**'onsetDateTime'**,**'resolvedDateTime'**,**'codeText'**]**

arr **=** **[]**

**for** name**,**group **in** tqdm**(**df\_condition**.**groupby**([**'patientId'**,**'encounterId'**,**'onsetDateTime'**])):**

#Groupby Condition Code Again

**for** name2**,** group2 **in** group**.**groupby**([**'code'**,**'codeText'**]):**

ar **=** **[]**

# Add patientId

ar**.**append**(**name**[**0**])**

# Add code

ar**.**append**(**name2**[**0**])**

# Add encounterId

ar**.**append**(**name**[**1**])**

# Add onsetDateTime

ar**.**append**(**name**[**2**])**

#Get Records with clinicalStatusCode as Resolved

resolved **=** group2**.**query**(**'clinicalStatusCode == "resolved"'**)**

#Add Resolved Date to Array if Resolved Record exists

**if** **len(**resolved**)** **>** 0 **:**

ar**.**append**(**resolved**[**'recordedDate'**].max())**

**else:**

ar**.**append**(**group2**[**'recordedDate'**].max())**

# Add codeText

ar**.**append**(**name2**[**1**])**

arr**.**append**(**ar**)**

df\_condition\_new **=** pd**.**DataFrame**(**arr**,** columns **=** cols**)**

df\_condition\_new**[**'onsetDateTime'**]** **=** pd**.**to\_datetime**(**df\_condition\_new**[**'onsetDateTime'**],**

**format=**"%Y-%m-%dT%H:%M:%S%z"**,** utc**=True)**

df\_condition\_new**[**'resolvedDateTime'**]** **=** pd**.**to\_datetime**(**df\_condition\_new**[**'resolvedDateTime'**],**

**format=**"%Y-%m-%dT%H:%M:%S%z"**,** utc**=True)**

df\_condition\_new**.**head**()**

df\_condition\_new**.**query**(**'code == "840539006"'**)**

"""840539006 Is the Code for COVID 19"""

df\_condition\_new**.**to\_csv**(**output\_folder\_path**+**delim**+**'Condition.csv'**)**

**del** df\_condition\_new

gc**.**collect**()**

"""## Encounters"""

cols **=** **[**'id'**,**'status'**,**'code'**,**'codeText'**,**'start'**,**'end'**,**'patientId'**,**'location'**,**'serviceProvider'**,**'encounterClass'**]**

arr **=** **[]**

start **=** time**.**time**()**

f\_count **=** 0

**for** file **in** tqdm**(**files**):**

**try:**

#load File

f **=** **open(**file**)**

data **=** json**.**load**(**f**)**

f**.**close**()**

encounters **=** filter\_resource**(**data**,** 'Encounter'**)**

**for** encounter **in** encounters**:**

ar **=** **[]**

ar**.**append**(**encounter**[**'resource'**][**'id'**])**

ar**.**append**(**encounter**[**'resource'**][**'status'**])**

ar**.**append**(**encounter**[**'resource'**][**'type'**][**0**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**encounter**[**'resource'**][**'type'**][**0**][**'text'**])**

ar**.**append**(**encounter**[**'resource'**][**'period'**][**'start'**])**

ar**.**append**(**encounter**[**'resource'**][**'period'**][**'end'**])**

ar**.**append**(**encounter**[**'resource'**][**'subject'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**encounter**[**'resource'**][**'location'**][**0**][**'location'**][**'display'**])**

ar**.**append**(**encounter**[**'resource'**][**'serviceProvider'**][**'display'**])**

ar**.**append**(**encounter**[**'resource'**][**'class'**][**'code'**])**

arr**.**append**(**ar**)**

**except** **Exception** **as** e**:**

# print(e)

f\_count **+=** 1

**continue**

end **=** time**.**time**()**

**print(str(**f\_count**)+**' Files Failed...'**)**

**print(str(len(**arr**))+**' Patient encounter bundles extracted as DataFrame in '**+str(**end**-**start**)+** 'Seconds'**)**

df\_encounter **=** pd**.**DataFrame**(**arr**,** columns **=** cols**)**

df\_encounter**.**head**()**

df\_encounter**[**'encounterClass'**].**value\_counts**()**

df\_encounter**.**to\_csv**(**output\_folder\_path**+**delim**+**'Encounter.csv'**)**

**del** df\_encounter

gc**.**collect**()**

"""## Observations"""

cols **=** **[**'id'**,**'patientId'**,**'issuedDate'**,**'effectiveDateTime'**,**'category'**,**'encounter'**,**'code'**,**'codeText'**,**'value'**,**'units'**,**'snomedCode'**,**'observationType'**]**

arr **=** **[]**

start **=** time**.**time**()**

f\_count **=** 0

**for** file **in** tqdm**(**files**):**

**try:**

#load File

f **=** **open(**file**)**

data **=** json**.**load**(**f**)**

f**.**close**()**

observations **=** filter\_resource**(**data**,** 'Observation'**)**

**for** observation **in** observations**:**

**if** 'component' **in** observation**[**'resource'**].**keys**():**

**for** comp **in** observation**[**'resource'**][**'component'**]:**

ar **=** **[]**

ar**.**append**(**observation**[**'resource'**][**'id'**])**

ar**.**append**(**observation**[**'resource'**][**'subject'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**observation**[**'resource'**][**'issued'**])**

ar**.**append**(**observation**[**'resource'**][**'effectiveDateTime'**])**

ar**.**append**(**observation**[**'resource'**][**'category'**][**0**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**observation**[**'resource'**][**'encounter'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**comp**[**'code'**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**comp**[**'code'**][**'coding'**][**0**][**'display'**])**

**if** 'valueCodeableConcept' **in** comp**.**keys**():**

ar**.**append**(**comp**[**'valueCodeableConcept'**][**'coding'**][**0**][**'display'**])**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**comp**[**'valueCodeableConcept'**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**'text'**)**

**elif** 'valueQuantity' **in** comp**.**keys**():**

ar**.**append**(**comp**[**'valueQuantity'**][**'value'**])**

ar**.**append**(**comp**[**'valueQuantity'**][**'unit'**])**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**'numeric'**)**

**else:**

ar**.**append**(**comp**[**'valueString'**])**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**'text'**)**

arr**.**append**(**ar**)**

**else:**

ar **=** **[]**

ar**.**append**(**observation**[**'resource'**][**'id'**])**

ar**.**append**(**observation**[**'resource'**][**'subject'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**observation**[**'resource'**][**'issued'**])**

ar**.**append**(**observation**[**'resource'**][**'effectiveDateTime'**])**

ar**.**append**(**observation**[**'resource'**][**'category'**][**0**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**observation**[**'resource'**][**'encounter'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**observation**[**'resource'**][**'code'**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**observation**[**'resource'**][**'code'**][**'coding'**][**0**][**'display'**])**

**if** 'valueCodeableConcept' **in** observation**[**'resource'**].**keys**():**

ar**.**append**(**observation**[**'resource'**][**'valueCodeableConcept'**][**'coding'**][**0**][**'display'**])**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**observation**[**'resource'**][**'valueCodeableConcept'**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**'text'**)**

**elif** 'valueString' **in** observation**[**'resource'**].**keys**():**

ar**.**append**(**observation**[**'resource'**][**'valueString'**])**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**'text'**)**

**else:**

ar**.**append**(**observation**[**'resource'**][**'valueQuantity'**][**'value'**])**

ar**.**append**(**observation**[**'resource'**][**'valueQuantity'**][**'unit'**])**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**'numeric'**)**

arr**.**append**(**ar**)**

**except** **Exception** **as** e**:**

# print(e)

# print(observation['resource'])

f\_count **+=** 1

**continue**

end **=** time**.**time**()**

**print(str(**f\_count**)+**' Files Failed...'**)**

**print(str(len(**arr**))+**' Patient observation bundles extracted as DataFrame in '**+str(**end**-**start**)+** 'Seconds'**)**

df\_observation **=** pd**.**DataFrame**(**arr**,** columns **=** cols**)**

df\_observation

df\_observation**.**to\_csv**(**output\_folder\_path**+**delim**+**'Observation.csv'**)**

**del** df\_observation

**del** ar

**del** arr

gc**.**collect**()**

"""## Care Plan"""

cols **=** **[**'id'**,**'status'**,**'patientId'**,**'start'**,**'end'**,**'category'**,**'code'**,**'codeText'**,**'intent'**,**'encounter'**,**'careTeam'**,**'activityCode'**,**'activityCodeText'**,**'activityStatus'**,**'activityLocation'**]**

arr **=** **[]**

start **=** time**.**time**()**

f\_count **=** 0

**for** file **in** tqdm**(**files**):**

**try:**

#load File

f **=** **open(**file**)**

data **=** json**.**load**(**f**)**

f**.**close**()**

cps **=** filter\_resource**(**data**,** 'CarePlan'**)**

**for** cp **in** cps**:**

**if** 'activity' **in** cp**[**'resource'**].**keys**():**

**for** activity **in** cp**[**'resource'**][**'activity'**]:**

ar **=** **[]**

ar**.**append**(**cp**[**'resource'**][**'id'**])**

ar**.**append**(**cp**[**'resource'**][**'status'**])**

ar**.**append**(**cp**[**'resource'**][**'subject'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**cp**[**'resource'**][**'period'**][**'start'**])**

**if** 'end' **in** cp**[**'resource'**][**'period'**].**keys**():**

ar**.**append**(**cp**[**'resource'**][**'period'**][**'end'**])**

**else:**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**cp**[**'resource'**][**'category'**][**0**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**cp**[**'resource'**][**'category'**][**1**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**cp**[**'resource'**][**'category'**][**1**][**'coding'**][**0**][**'display'**])**

ar**.**append**(**cp**[**'resource'**][**'intent'**])**

ar**.**append**(**cp**[**'resource'**][**'encounter'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**cp**[**'resource'**][**'careTeam'**][**0**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**activity**[**'detail'**][**'code'**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**activity**[**'detail'**][**'code'**][**'coding'**][**0**][**'display'**])**

ar**.**append**(**activity**[**'detail'**][**'status'**])**

ar**.**append**(**activity**[**'detail'**][**'location'**][**'display'**])**

arr**.**append**(**ar**)**

**else:**

ar **=** **[]**

ar**.**append**(**cp**[**'resource'**][**'id'**])**

ar**.**append**(**cp**[**'resource'**][**'status'**])**

ar**.**append**(**cp**[**'resource'**][**'subject'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**cp**[**'resource'**][**'period'**][**'start'**])**

**if** 'end' **in** cp**[**'resource'**][**'period'**].**keys**():**

ar**.**append**(**cp**[**'resource'**][**'period'**][**'end'**])**

**else:**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**cp**[**'resource'**][**'category'**][**0**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**cp**[**'resource'**][**'category'**][**1**][**'coding'**][**0**][**'code'**])**

ar**.**append**(**cp**[**'resource'**][**'category'**][**1**][**'coding'**][**0**][**'display'**])**

ar**.**append**(**cp**[**'resource'**][**'intent'**])**

ar**.**append**(**cp**[**'resource'**][**'encounter'**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**cp**[**'resource'**][**'careTeam'**][**0**][**'reference'**].**strip**().**split**(**'urn:uuid:'**)[**1**])**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**np**.**nan**)**

ar**.**append**(**np**.**nan**)**

arr**.**append**(**ar**)**

**except** **Exception** **as** e**:**

# print(e)

f\_count **+=** 1

**continue**

end **=** time**.**time**()**

**print(str(**f\_count**)+**' Files Failed...'**)**

**print(str(len(**arr**))+**' Patient CarePlan bundles extracted as DataFrame in '**+str(**end**-**start**)+** 'Seconds'**)**

df\_cp **=** pd**.**DataFrame**(**arr**,** columns **=** cols**)**

df\_cp

df\_cp**.**to\_csv**(**output\_folder\_path**+**delim**+**'CarePlan.csv'**)**

**del** df\_cp

gc**.**collect**()**