**0) Prerequisites before reading this document:**

* Before doing this project, I assume that you are familiar with some basic Python concepts and regular expressions. If not, please check the link [here](https://docs.python.org/3/tutorial/) to get started.
* Some concepts I used in this project are [the control flow](https://docs.python.org/3/tutorial/controlflow.html), [list](https://docs.python.org/3/tutorial/datastructures.html), [re.split()](https://docs.python.org/3/library/re.html), and [pandas DataFrame](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html).

**1) Problem or Challenge:**

* Convert the [Super Bowl Challenge Complaints](https://drive.google.com/file/d/1fJXMjs9S31mYS3LyPRrH8LQL9GAjoJyx/view?usp=sharing) to .csv or .xlsx file (Generally the excel files)

**2) Setting up the environment:**

I use Python to approach to this solution:

* Where to write Python code? ( Section 1)
* I use [Google Colab](https://colab.research.google.com/notebooks/intro.ipynb#) as my Integrated Development Environment (ide). Basically, it is easier to write and read python on this program. Also, it is online so everyone had access to the link can see and modify the code.
* How to import the pdf files to the code? ( Section 1)
* I use the built-in Google Colab function to extract the path containing the pdf files.

from google.colab import drive

drive.mount('/content/gdrive/')

#The location of the PDF file.

path = 'gdrive/My Drive/ITAP/Summer Intern/Super\_bowl\_challenge/'

* Afterward, I use the PyPDF4 to import and read the PDF file in the Python code.
* What is PyPDF4? ( Section 1)
* PyPDF4 is a built-in Python package that can import, read, and write on the PDF file.
* Firstly, since PyPDF4 is not in Google. Colab, I install it on Google. Colab by using the code below.

!pip install PyPDF4

* Then, I import and use the PyPDF4 to read the PDF file in my Drive (from the path I mentioned above).

import PyPDF4

#Import the PDF file to the code.

Obj\_pdf = open(path+'super\_bowl\_halftime\_complaints.pdf','rb')

pdf = PyPDF4.PdfFileReader(Obj\_pdf)

* What is re in my code? ( Section 1)
* I will explain this built-in package further in another section. For now, just implement it into the code

import re

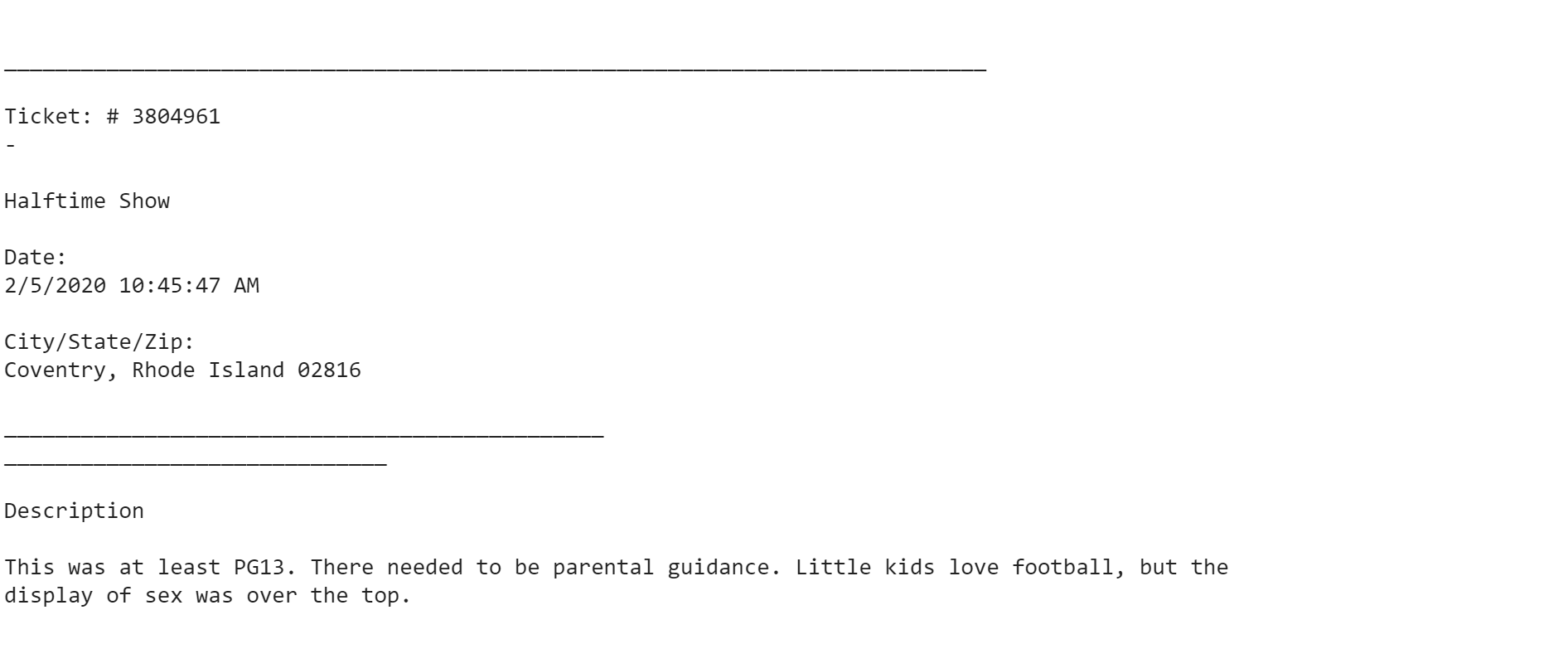
**3) Handling the PDF files:**

* How to get the content and the words from the PDF files? ( Section 2)
* My method is to assign a variable to a PDF page by using PyPDF4 built-in function. Afterward, I extract its text as a string to read the files.
* For example, the code below is to check Page 2 in the files. Notice here, PyPDF4 starts the PDF files from page 0.

#Checking the content in each page

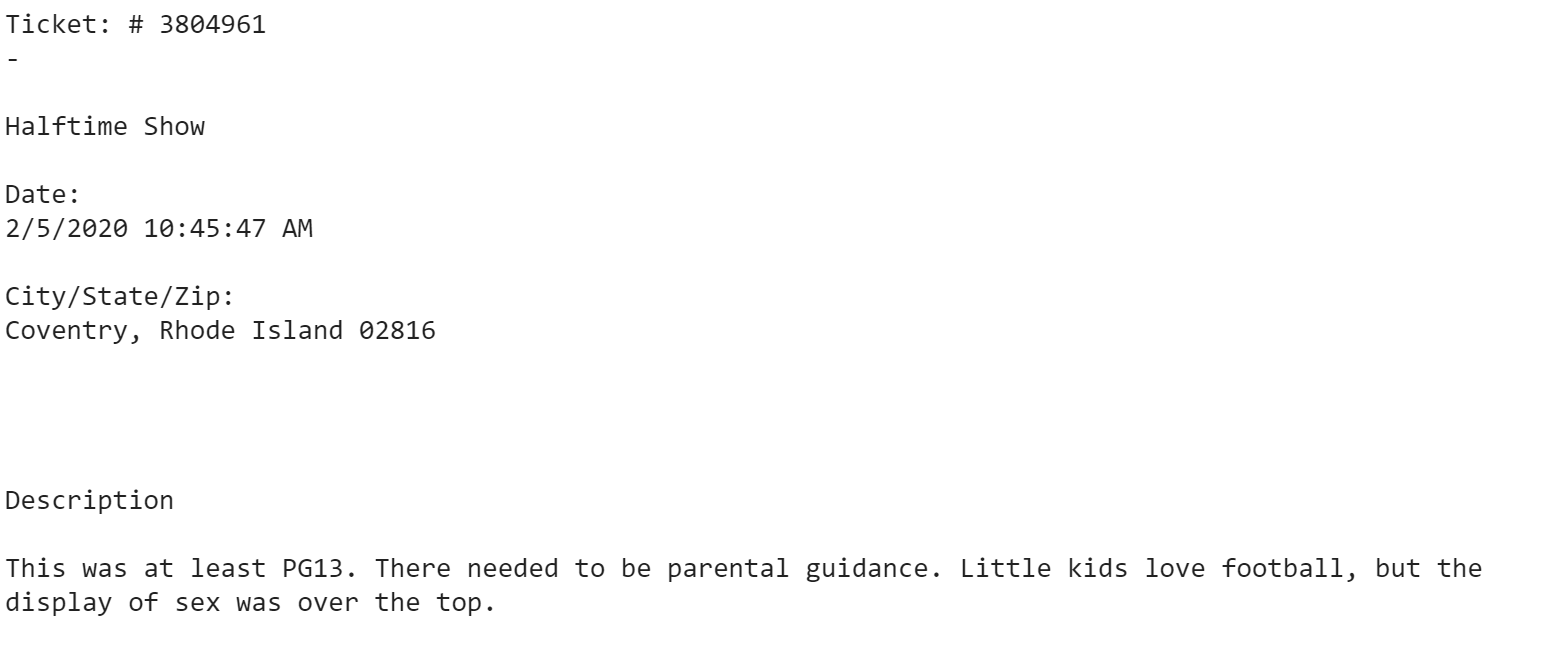
page = pdf.getPage(1)

print(page.extractText())



* How do I remove unnecessary components? ( Section 2)
* As you may notice here, we don’t need the ‘\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_’ in the picture above. Surprisingly, it is just a string that consists of multiple ‘\_’. Therefore, we will replace it with a simple empty string and later remove the empty string.

print(page.extractText().replace('\_', ''))



* How do we extract the string as the exact component and read them? ( Section 2)
* As you can see, right now, our code has multiple complex strings and we want to classify those complex string into multiple categories
* These categories include: ‘Ticket’, ‘Date’, ‘City/State/Zip’, ‘Company Complaining About’, ‘Description’.
* The solution is to separate the string into the list if whitespace character or newline is between these string

m = re.split(r'[\s\n]\s\*\n\*',page.extractText().replace('\_', ''))

* r'[\s\n]\s\*\n\*' represents the whitespace character (\s) and the newline (\n). Below is an example of the list (m) of the string after we have split.

['', 'Ticket:', '#', '3804961', '-', 'Halftime', 'Show', 'Date:', '2/5/2020', '10:45:47', 'AM', 'City/State/Zip:', 'Coventry,', 'Rhode', 'Island', '02816', 'Description', 'This', 'was', 'at', 'least', 'PG13.', 'There', 'needed', 'to', 'be', 'parental', 'guidance.', 'Little', 'kids', 'love', 'football,', 'but', 'the', 'display', 'of', 'sex', 'was', 'over', 'the', 'top.', '']

* As we can see from the example here, ‘Ticket’, ‘Date’, ‘City/State/Zip’, ‘Description’ is still mixed up with other string. However, we can now easily access them by the index of the list or we can loop through the list to categorize the elements.
* In this situation, I choose to loop through the list and begin to clean the empty string as well as categorize the elements.
* How do I clean up the empty string? ( Section 2)
* As you convert the ‘\_’ to an empty string, you still have to remove the empty string in some pages.
* For the empty string:

while(m.count('')):

m.remove('')

* In some situations, the whitespace string characters may appear and this is solution to handle them:

while(m.count(' ')):

m.remove(' ')

* How do we categorize each element with the least overlapping? ( Section 2)
* Overall, I create each list represent as each column. After I find the corresponding keyword of the column, I add them to a temporary list. The temporary list will stop when it meets the corresponding keyword for the new column Then I convert the temporary list to the necessary element and push them to the corresponding list.
* For example, below is how I separate the ‘Ticket’ column.

1. I initialize two different lists: one is the corresponding ‘Ticket’ list

ticket = []

And the other is the temporary list which temporarily holds the value of the ‘Ticket’ element of each page.

n = []

1. I begin to loop through each element in the list. As I find the corresponding keyword, I begin to push the corresponding element to the temporary list. As soon as I see the next keyword element which is ‘Date:’, I stop the loop.

i = 0

if m[i][0] == "T":

i += 1

if m[i] != '#':

i += 1

while(m[i] != "Date:" and m[i] + m[i+1] != "Date:"):

n.append(m[i])

i += 1

1. Finally, I merge the elements in the temporary list (n) to become a string and push the element on the top of the ‘Ticket’ list

n = " ".join(n)

ticket.append(n)

* We do the same process for other lists.
* How do we convert the list to csv file? ( Section 5)
* In python, we have the pandas built-in function that will convert the list to the DataFrame and the column. After I have converted the list to the column, I add all of them to a variable df.

import pandas as pd

data = {'Ticket': ticket, 'Date': date, 'Location': location, 'Company Complaining About': cpm, 'Description': des}

df = pd.DataFrame(data, columns = ['Ticket', 'Date', 'Location', 'Company Complaining About', 'Description'])

* Finally, I change df to the csv file.

df.to\_csv(path+'super\_bowl\_halftime\_complaints\_raw.csv', index=False)

**4) Problems encountered:**

* What if some pages do not contain the corresponding elements? ( Section 2)
* Throughout 1312 pages, not every single page contains ‘Company Complaining About’ or ‘Description’. Therefore, instead of using the null list and simultaneously appending the element, I instantiate a list consisting of 1312 null values. Then, I modify the corresponding element in the list by using the list index.
* What if PyPDF4 misses some elements in the PDF? ( Section 4)
* PyPDF4 is still under maintenance and it still needs to improve in the future. Without a doubt, it still has many mistakes. One of them is to automatically ignore the element in the ‘Description’ column.
* Therefore, I have to check this by hand and add them manually.

**5) Disadvantages and advantages of this method:**

* Disadvantages:
* Require a large amount of knowledge of coding experience, not a beginner project.
* Use multiple different Python libraries.
* Many typos throughout the files.
* Sometimes may accidentally miss a couple of elements in the PDF.
* Takes a lot of time to handle the list and string
* Advantages:
* Separate each of the elements into the corresponding columns.
* Read the PDF file more correctly than other Python libraries.
* Convert more correctly than other methods so far.
* It is free and Python libraries are easily accessible for everyone.

**6) Alternative methods:**

* Is there any other method easier and no coding experience requirement?
* Yes. In fact, I have found 2 different alternative methods that require no coding experience. One is to use the Adobe convert and another method is through this [link](https://smallpdf.com/pdf-to-excel).
* With the Adobe conversion method, you will have to pay the fee for it to convert. However, it only converts the PDF pages to Excel which is not optimal.

<https://drive.google.com/file/d/1bt1lo4-XNa5wFVuDybP4uZAIrOPoa2dN/view?usp=sharing>

* With the conversion online, it is free to convert. However, the file is messed up and has to handle later on.

<https://drive.google.com/file/d/1KxdruOjqyVEZ4ggNZsalF-u038nuPgWo/view?usp=sharing>