Noé Flores

Introduction:
Our data consists of a pickled data file (enron_email_df_complete.pkl) containing emails from the collapsed and extinct Enron Corporation. Enron infamously filed for bankruptcy on Sunday, December 2, 2001, under the leadership of Kenneth Lay. The source of the collapse of Enron centered on the revelation that its reported financial condition was sustained by creatively planned accounting fraud since known merely as the Enron scandal. We will try to examine some of the email activity of Kenneth Lay and determine how that activity changed with the announcement of bankruptcy.

The data sets were initially housed in a pickled pandas file which we un-pickled and transferred to a pandas data frame.

(1). Create a 'file_list' from the directory and verify that our json files are located in the new 'file_list'.

```
###Un-pickle file and transfer to Pandas dataframe####
enron_email_df=pd.read_pickle("enron_email_df_complete.pkl")

###Check shape of Pandas Dataframe.
enron_email_df.shape
Out[3]: (501513, 12)
```

We can see that our dataframe contains 501,503 total email records with 12 distinct columns.

The email file contains basic framework that we would expect, such as the body, To, From, Date, Subject, Message-ID, etc.

```
enron_email_df.head(1)
Out[4]:
                                                body  \
0  the scrimmage is still up in the air...\n\n\nw...

                                    Date                       From  \
0  Tue, 14 Nov 2000 08:22:00 -0800 (PST)  michael.simmons@enron.com

                                     Message-ID  \
0  <6884142.1075854677416.JavaMail.evans@thyme>

                           Subject                   To           X-From  \
0  Re: Plays and other information  eric.bass@enron.com  Michael Simmons

        X-To X-bcc X-cc mailbox    subFolder
0  Eric Bass              bass-e  notes_inbox

enron_email_df.tail(1)
Out[5]:
                                                     body  \
501512  i think the YMCA has a class that is for peopl...

                                         Date                      From  \
501512  Mon, 26 Nov 2001 10:48:43 -0800 (PST)  john.zufferli@enron.com

                                           Message-ID           Subject  \
501512  <28618979.1075842030037.JavaMail.evans@thyme>  RE: ali's essays

                                To  \
501512  livia_zufferli@monitor.com

                                                   X-From  \
501512  Zufferli, John </O=ENRON/OU=NA/CN=RECIPIENTS/C...

                                      X-To X-bcc X-cc     mailbox   subFolder
501512  'Livia_Zufferli@Monitor.com@ENRON'             zufferli-j  sent_items
```

We need to do some data preparation in order to proceed with the analysis of the emails. The first thing we must do is replace any NaN values with and empty string.

```
###Prepare data by replacing nan values.
enron_email_df.fillna("", inplace = True)
```

Now that we replaced all the NaN values, we can also str.strip to remove all the characters and space from our columns.

```
 ###Strip all characters
enron_email_df.columns = enron_email_df.columns.str.strip()
```

The final data preparation we will convert the date column from a string to DateTime objects. We import and incorporate the datetime package into Python to make the conversion.

```
 ###Import datetime
from datetime import datetime

###Convert date
enron_email_df["Date"] = pd.to_datetime(enron_email_df.Date,errors = 'ignore')
```

We do a quick check to ensure the conversion, and all of the other date preparation steps have gone through without issue.

```
enron_email_df.head(1)
Out[10]:
                                                body  \
0  the scrimmage is still up in the air...\n\n\nw...

                                    Date                       From  \
0  Tue, 14 Nov 2000 08:22:00 -0800 (PST)  michael.simmons@enron.com

                                     Message-ID  \
0  <6884142.1075854677416.JavaMail.evans@thyme>

                           Subject                   To           X-From  \
0  Re: Plays and other information  eric.bass@enron.com  Michael Simmons

        X-To X-bcc X-cc mailbox    subFolder
0  Eric Bass             bass-e  notes_inbox

enron_email_df.tail(1)
Out[11]:
                                                     body  \
501512  i think the YMCA has a class that is for peopl...

                                          Date                       From  \
501512  Mon, 26 Nov 2001 10:48:43 -0800 (PST)  john.zufferli@enron.com

                                              Message-ID           Subject  \
501512  <28618979.1075842030037.JavaMail.evans@thyme>  RE: ali's essays

                             To  \
501512  livia_zufferli@monitor.com

                                                 X-From  \
501512  Zufferli, John </O=ENRON/OU=NA/CN=RECIPIENTS/C...

                                     X-To X-bcc X-cc     mailbox   subFolder
501512  'Livia_Zufferli@Monitor.com@ENRON'             zufferli-j  sent_items
```

(2). We now begin our exploration of the emails to find all the emails associated with the former CEO, Kenneth Lay. This will take a bit of digging, so we begin with a quick query to check all the emails in our dataset "To" and "From" a recipient/sender with "lay" in their address. The process resulted in 11,723 instances of the name "lay" addressed in an email.

```
emails_to_lay = (enron_email_df[(enron_email_df["To"].str.contains(".*lay@enron.com*."))])

emails_to_lay.shape
Out[5]: (11723, 12)

emails_from_lay = (enron_email_df[(enron_email_df["From"].str.contains(".*lay@enron.com*."))])

emails_from_lay.shape
Out[6]: (1207, 12)
```

(3). The information above clearly shows that there are a lot of email addresses to sort through and quite obviously not everyone with "lay" in their email address is Kenneth Lay the CEO. We refined our search and found several email addresses associated with Ken Lay, and we broke down how many emails were sent "To" and "From" each of these email addresses. The answers to each are highlighted in the code in **BLUE**.

1.(klay@enron.com)

```
klay_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*klay@enron.com*."))])
print("Emails 'to' klay@enron:",klay_emails_to)
Emails 'TO' klay@enron: 1955

klay_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*klay@enron.com*."))])
print("Emails 'from' klay@enron:", klay_emails_from)
Emails 'FROM' klay@enron: 0
```

2.(kenlay@enron.com)

```
kenlay_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*kenlay@enron.com*."))])
print("Emails 'TO' kenlay@enron:",kenlay_emails_to)
Emails 'TO' kenlay@enron: 1

kenlay_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*kenlay@enron.com*."))])
print("Emails 'FROM' kenlay@enron:", kenlay_emails_from)
Emails 'FROM' kenlay@enron: 0
```

3.(k_lay@enron.com)

```
k_lay_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*k_lay@enron.com*."))])
print("Emails 'TO' k_lay@enron:",k_lay_emails_to)
Emails 'TO' k_lay@enron: 5

k_lay_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*k_lay@enron.com*."))])
print("Emails 'FROM' k_lay@enron:", k_lay_emails_from)
Emails 'FROM' k_lay@enron: 0
```

4.(ken_lay@enron.com)

```
ken_lay_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*ken_lay@enron.com*."))])
print("Emails 'TO' ken_lay@enron:",ken_lay_emails_to)
Emails 'TO' ken_lay@enron: 3

ken_lay_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*ken_lay@enron.com*."))])
print("Emails 'FROM' ken_lay@enron:", ken_lay_emails_from)
Emails 'FROM' ken_lay@enron: 0
```

5.(kenneth.lay@enron.com)

```
kenneth_lay_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*kenneth.lay@enron.com*."))])
print("Emails 'TO' kenneth.lay@enron:",kenneth_lay_emails_to)
Emails 'TO' kenneth.lay@enron: 4296

kenneth_lay_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*kenneth.lay@enron.com*."))])
print("Emails 'FROM' kenneth.lay@enron:", kenneth_lay_emails_from)
Emails 'FROM' kenneth.lay@enron: 36
```

6.(chairman.ken@enron.com)

```
chairmanken_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*chairman.ken@enron.com*."))])
print("Emails 'TO' chairman.ken@enron:",chairmanken_emails_to)
Emails 'TO' chairman.ken@enron: 5

chairmanken_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*chairman.ken@enron.com*."))])
print("Emails 'FROM' chairman.ken@enron:", chairmanken_emails_from)
Emails 'FROM' chairman.ken@enron: 328
```

7.(kennethlay@enron.com)

```
kennethlay_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*kennethlay@enron.com*."))])
print("Emails 'TO' kennethlay@enron:",kennethlay_emails_to)
Emails 'TO' kennethlay@enron: 1

kennethlay_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*kennethlay@enron.com*."))])
print("Emails 'FROM' kennethlay@enron:", kennethlay_emails_from)
Emails 'FROM' kennethlay@enron: 0
```

8.(layk@enron.com)

```
layk_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*layk@enron.com*."))])
print("Emails 'to' layk@enron:",layk_emails_to)
Emails 'to' layk@enron: 1

layk_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*layk@enron.com*."))])
print("Emails 'to' layk@enron:", layk_emails_from)
Emails 'to' layk@enron: 0
```

9.(kenneth_lay@enron.com)

```
kenneth_lay_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*kenneth_lay@enron.com*."))])
print("Emails 'TO' kenneth_lay@enron:",kenneth_lay_emails_to)
Emails 'TO' kenneth_lay@enron: 40

kenneth_lay_emails_from = len(enron_email_df[(enron_email_df["From"].str.contains(".*kenneth_lay@enron.com*."))])
print("Emails 'FROM' kenneth_lay@enron:", kenneth_lay_emails_from)
Emails 'FROM' kenneth_lay@enron: 0
```

10.(kenneth_lay@enron.net)

```
kenneth_lay_net_emails_to = len(enron_email_df[(enron_email_df["To"].str.contains(".*kenneth_lay@enron.net*."))])
print("Emails 'TO' kenneth_lay@enron.net:",kenneth_lay_net_emails_to)
Emails 'TO' kenneth_lay@enron.net: 3

kenneth_lay_net_emails_from =
len(enron_email_df[(enron_email_df["From"].str.contains(".*kenneth_lay@enron.net*."))])
print("Emails 'FROM' kenneth_lay@enron.net:", kenneth_lay_net_emails_from)
Emails 'FROM' kenneth_lay@enron.net: 0
```

We had four email addresses from the list above produce a significant amount of emails involving Kenneth Lay.

(klay@enron.com)

```
Emails 'TO' klay@enron: 1955
Emails 'FROM' klay@enron: 0
```

(kenneth.lay@enron.com)

```
Emails 'TO' kenneth.lay@enron: 4296
Emails 'FROM' kenneth.lay@enron: 36
```

(chairman.ken@enron.com)

```
Emails 'TO' chairman.ken@enron: 5
Emails 'FROM' chairman.ken@enron: 328
```

(kenneth_lay@enron.com)

```
Emails 'TO' kenneth.lay@enron: 40
Emails 'FROM' kenneth.lay@enron: 0
```

(4). We can now proceed to determine who Ken Lay received the most emails "From" using **(klay@enron.com)** and how many the person sent.

```
###Count number of emails to klay@enron
klay_to_count = enron_email_df
klay_to_count["Count"] = 1
klay_to_count = klay_to_count[(enron_email_df["To"].str.contains(".*klay@enron.com*."))]
emails_to_klay = klay_to_count.groupby(["From"])["Count"].sum()
emails_to_klay = emails_to_klay.sort_values(ascending = False)
print("Person who sent the most emails to Ken Lay:", emails_to_klay.index[0])
print("Total emails sent", emails_to_klay.index[0], "to Ken Lay:", emails_to_klay[0])

Person who sent the most emails to Ken Lay: savont@email.msn.com
Total emails sent savont@email.msn.com to Ken Lay: 54
```

Ken Lay received the most emails from (**savont@email.msn.com**). Total equaling 54.

We know from our original analysis of searching for the emails that **(klay@enron.com)** did not send any emails from this address particular.

Moving on to determine who Ken Lay received the most emails "From" using **(kenneth.lay@enron.com)** and how many the person sent.

```
###Count number of emails to kenneth.lay@enron
kennethlay_to_count = enron_email_df
kennethlay_to_count["Count"] = 1
kennethlay_to_count = kennethlay_to_count[(enron_email_df["To"].str.contains(".*kenneth.lay@enron.com*."))]
emails_to_kennethlay = kennethlay_to_count.groupby(["From"])["Count"].sum()
emails_to_kennethlay = emails_to_kennethlay.sort_values(ascending = False)
print("Person who sent the most emails to Ken Lay:", emails_to_kennethlay.index[0])
print("Total emails sent", emails_to_kennethlay.index[0], "to Ken Lay:", emails_to_kennethlay[0])

Person who sent the most emails to Ken Lay: leonardo.pacheco@enron.com
Total emails sent leonardo.pacheco@enron.com to Ken Lay: 386
```

Out of the total 4,296 emails Ken Lay received in this particular account, the most emails were from (**leonardo.pacheco@enron.com**), total equaling 386 emails.

(**kenneth.lay@enron.com**) sent a total of 36 emails from this account. In our query, we found that he sent the exact same number of emails to a massive list of recipients from this account. A total of 6 emails to each.

```
kennethlaycount = enron_email_df
kennethlaycount["Count"] = 1
kennethlaycount = kennethlaycount[(enron_email_df["From"].str.contains(".*kenneth.lay@enron*."))]
emails_kennethlay = kennethlaycount.groupby(["To"])["Count"].sum()
emails_kennethlay = emails_kennethlay.sort_values(ascending = False)
print("Person received the most emails from Kenneth Lay:", emails_kennethlay.index[0])
print("Total emails sent", emails_kennethlay.index[0], "from Ken Lay:", emails_kennethlay[0])
Person received the most emails from Kenneth Lay: erica.adams@enron.com, john.addison@enron.com,
matthew.almy@enron.com,hector.alviar@enron.com,
Total emails sent erica.adams@enron.com, john.addison@enron.com, matthew.almy@enron.com,
hector.alviar@enron.com,from Ken Lay: 6
```

The final email account we will examine is (**chairman.ken@enron.com**)**.** We can recall that this particular email account saw the most messages sent **(328),** by Kenneth Lay in our datafile.

```
cken_to_count = enron_email_df
cken_to_count["Count"] = 1
cken_to_count = cken_to_count[(enron_email_df["To"].str.contains(".*chairman.ken@enron*."))]
emails_to_cken = cken_to_count.groupby(["From"])["Count"].sum()
emails_to_cken = emails_to_cken.sort_values(ascending = False)
print("Person who sent the most emails to Chairman Ken:", emails_to_klay.index[0])
print("Total emails sent", emails_to_cken.index[0], "to CKen Lay:", emails_to_cken[0])
Person who sent the most emails to Chairman Ken: savont@email.msn.com
Total emails sent a..shankman@enron.com to Chairman Ken: 2
```

Of the 5 emails sent to this account, a total of 2 were sent by (**a..shankman@enron.com**).

This particular account sent out a total of 328 total messages. Of those 328 messages, almost a third were sent to (**dl-ga-all_enron_worldwide2@enron.com**). This doesn't look like an official employee Enron account at first glance.

```
cken_to_count = enron_email_df
cken_to_count["Count"] = 1
cken_to_count = cken_to_count[(enron_email_df["From"].str.contains(".*chairman.ken@enron*."))]
emails_to_cken = cken_to_count.groupby(["To"])["Count"].sum()
emails_to_cken = emails_to_cken.sort_values(ascending = False)
print("Person received the most emails from Chairman Ken:", emails_to_klay.index[0])
print("Total emails sent", emails_to_cken.index[0], "to CKen Lay:", emails_to_cken[0])
Person received the most emails from Chairman Ken: savont@email.msn.com
Total emails sent dl-ga-all_enron_worldwide2@enron.com from Chairman Ken: 101
```

Our final account is (**kenneth_lay@enron.com**)**. This account utilizes an underscore as opposed to a (.) to separate the first and last name.**

```
kennethlay_to_count = enron_email_df
kennethlay_to_count["Count"] = 1
kennethlay_to_count = kennethlay_to_count[(enron_email_df["To"].str.contains(".*kenneth_lay@enron.com*."))]
emails_to_kennethlay = kennethlay_to_count.groupby(["From"])["Count"].sum()
emails_to_kennethlay = emails_to_kennethlay.sort_values(ascending = False)
print("Person who sent the most emails to Kenneth Lay:", emails_to_kennethlay.index[0])
print("Total emails sent", emails_to_kennethlay.index[0], "to Ken Lay:", emails_to_kennethlay[0])
Person who sent the most emails to Kenneth Lay: mrslinda@lplpi.com
Total emails sent mrslinda@lplpi.com to Ken Lay: 15
```

Ken Lay received the most emails form (**mrslinda@lplpi.com**). Total equaling 15.

(5). Now, we want to shift our focus now to the activity of the emails before, and after Enron filed for bankruptcy on December 2, 2001. The file as a whole shows a marked increase in email traffic after the filling for bankruptcy. Total of 188,770 before vs. 312,743 on or after December 2nd 2001 .

```
min = enron_email_df["Date"].min()
max = enron_email_df["Date"].max()

bankruptcy = enron_email_df
before = bankruptcy ['Date'] <= 'Sun, 02 Dec 2001'
print("emails before bankruptcy:", before.sum())
after = bankruptcy ['Date'] >= 'Sun, 02 Dec 2001'
print("emails after bankruptcy:", after.sum())
Number of emails before bankruptcy: 188770
Number of emails after bankruptcy: 312743
```

Now, we can observe those same behavior within the specific Kenneth Lay email addresses below.

(klay@enron.com)

```
Emails 'TO' klay@enron: 1955
bankruptcy = klay_emails_to
before = bankruptcy['Date'] <= 'Sun, 02 Dec 2001'
print("emails before bankruptcy:", before.sum())
after = bankruptcy['Date'] >= 'Sun, 02 Dec 2001'
print("emails after bankruptcy:", after.sum())

Number of emails before bankruptcy: 286
Number of emails after bankruptcy: 1669
```

(kenneth.lay@enron.com)

```
Emails 'TO' kenneth.lay@enron: 4296
bankruptcy = kenneth_lay_emails_to
before = bankruptcy['Date'] <= 'Sun, 02 Dec 2001'
print("emails before bankruptcy:", before.sum())
after = bankruptcy['Date'] >= 'Sun, 02 Dec 2001'
print("emails after bankruptcy:", after.sum())

Number of emails before bankruptcy: 1601
Number of emails after bankruptcy: 2695

Emails 'FROM' kenneth.lay@enron: 36
bankruptcy = kenneth_lay_emails_From
before = bankruptcy['Date'] <= 'Sun, 02 Dec 2001'
print("emails before bankruptcy:", before.sum())
after = bankruptcy['Date'] >= 'Sun, 02 Dec 2001'
print("emails after bankruptcy:", after.sum())

Number of emails before bankruptcy: 13
Number of emails after bankruptcy: 23
```

(chairman.ken@enron.com)

```
Emails 'FROM' chairman.ken@enron: 328
bankruptcy = chairmanken_emails_from
before = bankruptcy['Date'] <= 'Sun, 02 Dec 2001'
print("emails before bankruptcy:", before.sum())
after = bankruptcy['Date'] >= 'Sun, 02 Dec 2001'
print("emails after bankruptcy:", after.sum())

Number of emails before bankruptcy: 130
Number of emails after bankruptcy: 198
```

(kenneth_lay@enron.com)

```
Emails 'TO' kenneth.lay@enron: 40
bankruptcy = kenneth_lay_emails_to
before = bankruptcy['Date'] <= 'Sun, 02 Dec 2001'
print("emails before bankruptcy:", before.sum())
after = bankruptcy['Date'] >= 'Sun, 02 Dec 2001'
print("emails after bankruptcy:", after.sum())

Number of emails before bankruptcy: 12
Number of emails after bankruptcy: 28
```

(6).Finally, we want to see how many of these emails mention Arthur Anderson. We use the same code to extract the results for each of the following accounts but abbreviate the results after number (1).

(klay@enron.com)

```
import re
aa_email = klay_emails_to[["Subject", "body"]]
arthur_count = aa_email.applymap(lambda x: bool(re.search(".*Arthur Andersen*.", x))).any(axis=1)
print("emails that mention Athur Andersen:", arthur_count.sum())
emails that mention Athur Andersen: 3
```

(kenneth.lay@enron.com)

```
aa_email = kenneth_lay_emails_to[["Subject", "body"]]
emails that mention Athur Andersen: 5

aa email = kenneth lay emails from[["Subject", "body"]]
emails that mention Athur Andersen: 0
```

(chairman.ken@enron.com)

```
aa_email = chairmanken_emails_to[["Subject", "body"]]
emails that mention Athur Andersen: 0

aa_email = chairmanken_emails_from[["Subject", "body"]]
emails that mention Athur Andersen: 0
```

(kenneth_lay@enron.com)

```
aa_email = kenneth_lay_emails[["Subject", "body"]]
emails that mention Athur Andersen: 0
```