Scenario

Your big data consulting company has been hired by a small law firm to help them make sense of a document dump they have received for a big trial.

The firm believes that the outcome of their trial depends on finding certain information in the emails from the opposition's clients.

They have secured an initial dump of employee's emails at the company in question, but in order to get additional data they need to prove that there is value in the sample. In order for their document analysts to do that in a timely manner, they will need some metadata extracted from each email so they can process it using their document review tools.

If they are able to find what they need by the deadline, your company will get an ongoing contract to build a pipeline to process incoming document dumps (YAY!)

Assignment

Using the sample data archive file while consists of a series of emails, write a Spark application to extract the Message ID, Date, From and To fields from the header of each message, and output those fields along with the email contents into a CSV file.

Input:
Sample: https://bigdatatechnologiesstor.blob.core.windows.net/data/enron_2015_sample.tgz
Full: https://bigdatatechnologiesstor.blob.core.windows.net/data/enron_mail_20150507.tar.gz
The full data set is a tar gz file. Once expanded, the emails are in a dir structure of the following:

maildir/user/outlook-folder/message Where each message is a numbered text file (1., 2., etc..)

A sample of the full data set is provided as a convenience to help you get started

Output:
The output file should have the following format: Message-ID,Date,From,To,Message

- Note that the email will most likely contain commas, so you will need to make sure the fields in the CSV file are properly delimited (" is the standard for this)
- All data can be included "as-is"; you don't need to do any further cleaning, i.e. Parse the date into a timestamp
- Also because of the newlines in the email, the last column will be very ugly, that is expected.
- Writing an application that does not rely on Spark primitives to allow for scaling out will result in the assignment being rejected
- You will need to increase the Spark memory up from the default to run the full data set

Example

Given the following email text:

```
Message-ID: <16159836.1075855377439.JavaMail.evans@thyme>
Date: Fri, 7 Dec 2001 10:06:42 -0800 (PST)
From: heather.dunton@enron.com
To: k..allen@enron.com
Subject: RE: West Position
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Dunton, Heather </O=ENRON/OU=NA/CN=RECIPIENTS/CN=HDUNTON>
X-To: Allen, Phillip K. </O=ENRON/OU=NA/CN=RECIPIENTS/CN=Pallen>
X-cc:
X-bcc:
X-Folder: \Phillip_Allen_Jan2002_1\Allen, Phillip K.\Inbox
X-Origin: Allen-P
X-FileName: pallen (Non-Privileged).pst


Please let me know if you still need Curve Shift.

Thanks,
Heather
```

The corresponding line in the output CSV file would look like: (colored and split onto multiple lines for clarity)

```
<16159836.1075855377439.JavaMail.evans@thyme>,
"Fri, 7 Dec 2001 10:06:42 -0800 (PST)",
heather.dunton@enron.com,
k..allen@enron.com,
"Message-ID: <16159836.1075855377439.JavaMail.evans@thyme>
Date: Fri, 7 Dec 2001 10:06:42 -0800 (PST)
From: heather.dunton@enron.com
To: k..allen@enron.com
Subject: RE: West Position
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Dunton, Heather </O=ENRON/OU=NA/CN=RECIPIENTS/CN=HDUNTON>
X-To: Allen, Phillip K. </O=ENRON/OU=NA/CN=RECIPIENTS/CN=Pallen>
X-cc:
X-bcc:
X-Folder: \Phillip_Allen_Jan2002_1\Allen, Phillip K.\Inbox
X-Origin: Allen-P
X-FileName: pallen (Non-Privileged).pst

Please let me know if you still need Curve Shift.

Thanks,
Heather"
```

Bonus Exercise

We also would like to provide the client the option of directly querying the data through Spark using SQL. We can demonstrate that functionality to see if they would like to add-on to the contract.

As an additional optional exercise, write the output to a Parquet file with the data stored in Azure Data Lake. The output file should look the same, but as you learned in the first class, we should have much better performance for interactive querying using Parquet.

Once the data has been stored in Parquet, run some sample queries against both the CSV file and the Parquet and take note of any performance differences between the two.

Connecting to Databricks from a BI tool is out of scope for this exercise, but we will discuss this later when we discuss landing/serving the results of our data pipeline.