**Objective**

This document describes the design and solution for the enron email corpse parsing as part of the coding exercise

**Assumptions**

Logs are complete and have the same format. Hence, no extra effort was placed to further clean up or validate the data

**Design and Solution**

Upon reviewing data sample, leveraged regex patterns to identify each field distinctly. After each regex expression is evaluated, cleaned white spaces and stripped extra columns to arrive at the data for final analysis.

Following are the regex patterns –

﻿

*dt\_pattern = re.compile('\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2}')*

*mid\_pattern = re.compile("message\_id=<.\*>")*

*from\_pattern = re.compile("from=\"(.\*?)\"")*

*to\_pattern = re.compile("to=\"(.\*?)\"")*

*cc\_pattern = re.compile("cc=\"(.\*?)\"")*

*bcc\_pattern = re.compile("bcc=\"(.\*?)\"")*

*rto\_pattern = re.compile("reply\_to=\"(.\*?)\"")*

*sender\_pattern = re.compile("sender=\"(.\*?)\"")*

*subj\_pattern = re.compile("subject=\"(.\*?)\",")*

*fname\_pattern = re.compile("filename=\"(.\*?)\"")*

*mt\_pattern = re.compile("mime\_type=\"(.\*?)\"")*

*disp\_pattern = re.compile("disposition\_type=\"(.\*?)\"")*

*fw\_except = open('enron\_exception.log', "w+")*

**Data Granularity**

General email information like date, message\_id, from, subject etc form the parent record which is on higher grain than the to, cc and bcc fields as any of these fields can contains multiple recipients which means more than 1 record in database table. In order to accommodate this grain differences, data is parsed in four different functions and output is stored in different files, and four different tables in database.

**Data Parsing**

**﻿process\_enron\_data (line, fw)**

- Function takes each line in email corpse and file object as input, applies regex matching to the line and extracts individual fields. It concatenates data into tab separated string and writes data into file, fw.

Exception block will log any exceptions into a file that store data in json format. This file can be stored in hdfs/s3 partition that can be queries from hive/Athena for diagnostics.

File format -

﻿Email\_date message\_id from\_addr to\_addr reply\_to sender subject file\_name media\_type disposition\_type line\_insert\_datetime

**process\_enron\_to (line, fw)**

**-** While process\_enron\_data() parses email at header grain, this function parses data and particularly splits multiple email addresses into rows for the convenience in future analysis.

Message\_id acts as reference to the header table, in database, the same field acts as foreign key joining all tables.

File format -

﻿Email\_date message\_id to\_addr line\_insert\_datetime

**process\_enron\_cc (line, fw)**

**-** processes cc data in each line

File format -

﻿Email\_date message\_id cc\_addr line\_insert\_datetime

**process\_enron\_bcc (line, fw)**

**-** processes bcc data in each line

File format -

﻿Email\_date message\_id bcc\_addr line\_insert\_datetime

Data from above functions store a record for each message id even in the absence of corresponding field(like cc/bcc/to). This gives complete transparency for downstream analysis of data and a way to look at this raw data in future to identify data lineage

**Database Layer**

TSV files from above functions are loaded into local mysql database using “Load Data” utility.

DB Schema - enron

DB Tables

- enron.enron\_mail

Captures email header data

Primary Key – Mesaage\_ID

- enron.enron\_mail\_to

Captures recipients in to field as separate records

Primary Key – [Mesaage\_ID, to\_email\_addr]

- enron.enron\_mail\_cc

Captures recipients in cc field as separate records

Primary Key – [Mesaage\_ID, cc\_email\_addr]

- enron.enron\_mail\_bcc :

Captures recipients in bcc field as separate records

Primary Key – [Mesaage\_ID, bcc\_email\_addr]

DDLs and SQL Queries are uploaded in the zip folder under src folder