**Conference Call Project Documentation (20211216)**

**Summary of Main Processing Pipeline (For First Time Runs)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Task | Codes | Input | Output |
| **1** | **Download Raw Data** | | | |
| 1.1 | Download Thomson One’s Conference Calls [L] | mouse\_key\_recorder.py  automatic\_download  .py | - | O1: [yyyymmdd-yyyymmdd].pdf  O2: [yyyymmdd-yyyymmdd].xls |
| **2** | **PDF Processing** | | | |
| 2.1 | Convert Conference Calls from .pdf to .txt [M] | pdftransfer.sh | [O1](#conf_call_pdf) | O3: [yyyymmdd-yyyymmdd].txt |
| 2.2 | Split Conference Call .txt files to separate out individual conference calls, and combine with report information from .xls files [M, L] | ParseCCpdf.jl | [O3](#conf_call_txt), [O2](#conf_call_xls) | O4: [yyyymmdd-yyyymmdd].csv |
| **3** | **Keyword Identification** | | | |
| 3.1 | Download Compustat datasets [L] | - | - | O5: ciqcompany.sas7bdat  O6: ciqcountrygeo. sas7bdat  O7: wrds\_gvkey.sas7bdat |
| 3.2 | Download Hassan dataset [L] | - | - | O8: Hassanfile\_raw\_  updated20219030.csv |
| 3.3 | Process Compustat and Hassan datasets into usable and truncated .csv files. [L] | convert\_sas7bdattocsv.py  join\_compustatcsvfiles.py  hassanfilecsv\_viewable\_  truncate.py | [O5](#ciqcompany_sas7bdat), [O6](#ciqcountrygeo_sas7bdat), [O7](#wrds_gvkey_sas7bdat), [O8](#hassanfile_raw) | O9: ciqcompany.csv  O10: ciqcountrygeo.csv  O11: wrds\_gvkey.csv  O12: ciqcompany\_merged  withgvkeyandcountry.csv  O13: Hassanfile\_raw\_  updated20219030 \_truncated.csv |
| 3.4 | Make a folder structure with x groups and move .csv files into the folders (default: x = 50) [M, L] | mkdir.py  dividefilesequallyinto  folders.py | [O4](#conf_call_csv) | O14: a set of folders {group[i]} where csv files are divided between the groups |
| 3.5 | Make a list of keywords and template entry file [L] | - | - | O15: keyterms.txt  O16: Entry mask.xlsx |
| 3.6 | Identify keywords for the whole CC data. | CC\_identify\_keywords.py | [O14](#set_of_folders_with_csv), [O15](#keyterms) | O17: a set of folders Full\_Identified\_  Keywords/group[i]/ Full\_Identified\_[i].parquet.gzip |
| 3.7 | Concatenating all these files into a single dataset. | concatenateOutputs.py | [O17](#set_of_folders_with_keywords) | O18: Full\_Master\_Keywords.csv |
| 3.8 | Filter based on a more exact keyword identification algorithm (rather than just checking in, doing a holistic check by looking at the spaces around the keyword) | getCorrect.py | [O18](#full_master_keywords) | O19: Amended\_Correct\_No\_IR.csv |
| 3.9 | Filter based on the presence of a percentage (the words percent, per cent, percentage, %) and then order based on the sorting rule provided. | ordering\_and\_filtering.py | [O19](#amended_correct_no_ir) | O20: Filtered\_Ordered\_Amended\_  Correct\_No\_IR.csv |
| 3.10 | Convert current paragraphs and conference call information into entry files format | convertFilteredOrderedAmendedCorrectNoIR\_to\_TotalCircNew.py  convertTotalCircNew\_to\_Cric1newtotal.py  convertCric1newtotal\_to\_entryfilescombined.py | [O20](#filtered_ordered_amended_correct_no_ir) | O21: entryfilescombined\_withoutgvkey.xlsx |
| **4** | **Firm Identification (Firm Name and Gvkey Matching)** | | | |
| 4.1 | Perform the fuzzy matching between the Hassan/Compustat and the CC datasets. | cc\_fuzzy\_match\_part1.py  cc\_fuzzy\_match\_part2.py | [O21](#entryfilescombined), [O13](#hassanfile_truncated), [O12](#ciqcompany_mergedwithgvkeyandcountry_csv) | O22: updated\_matched\_conf\_calls\_match.xlsx  O23: updated\_unmatched\_conf\_calls\_match.xlsx |
| 4.2 | Do manual matching for unconfirmed cases | - | [O23](#updated_unmatched_conf_calls_match) | O24: Filled\_Updated\_CC\_Compustat\_FuzzyMatchCandidates.xlsx  O25: Filled\_Updated\_CC\_Hassan\_FuzzyMatchCandidates.xlsx |
| 4.3 | Combine manually matched cases with results from fuzzy matching | cc\_fuzzy\_match\_part2.py | [O22](#updated_matched_conf_calls_match), [O24](#filled_updated_cc_compustat_fuzzy), [O25](#filled_updated_cc_hassan_fuzzy) | O26: manual\_full\_updated\_  conf\_calls.xlsx |
| 4.4 | Make a paragraph record file that splits the number of entries into groups of 500 [L] | make\_paragraphrecord.py | [O26](#manual_full_updated_conf_calls) | O27: paragraphrecord.xlsx |
| 4.5 | Bold the keywords and separate file into “entryfiles”, each containing 500 entries. [M, L] | makeentryfiles.py  makeentryfiles.sh | [O27](#paragraphrecord), [O26](#manual_full_updated_conf_calls), [O15](#keyterms), [O16](#entrymask) | O28: A set of entryfiles, [i].xlsx |
| 4.6 | Combine entry files [L] | combine\_entryfilesjason.py  combine\_entryfilessixun.py  combine\_sixunand  jasonentryfiles.py | [O28](#set_of_entryfiles) | O29: entryfiles\_  combined.xlsx |
| **5** | **Get Front Page Descriptions** | | | |
| 5.1 | Extract front page descriptions from conference calls [M, L] | extractdescriptioninfront  page.py  extractdescriptioninfront  page.sh  copyfiles.py  copyfiles.sh | [O29](#entryfilescombined), [O3](#conf_call_txt), [O2](#conf_call_xls) | O30: [yyyymmdd-yyyymmdd]\_withfront  pagedesc.xlsx |
| 5.2 | Manually check through error cases and correct accordingly [L] | - | [O30](#conf_call_csv_withfrontpagedesc_nomanual) | O31: [yyyymmdd-yyyymmdd]\_withfront  pagedesc.xlsx |
| 5.3 | Combine xls files [L] | combine\_xlsfiles\_with  description.py | [O31](#conf_call_csv_withfrontpagedesc_manual) | O32: xlscombined\_with  frontpagedescription  .xlsx |
| 5.4 | Match and add front page descriptions to combined entry files [L] | - | [O32](#xlscombined_with_frontpagedesc), [O29](#entryfilescombined) | O33: entryfiles\_combined  \_withfrontpagedesc.xlsx (updated) |

\* M = Mercury, L = Local. [M] / [L] means this stage can be run on Mercury / locally (on your Booth Windows laptop) respectively. [M, L] means this stage can be run on both Mercury and your local laptop, where Mercury is preferred for large datasets and local is preferred for initial testing, debugging and small datasets.

## **1. Download Raw Data**

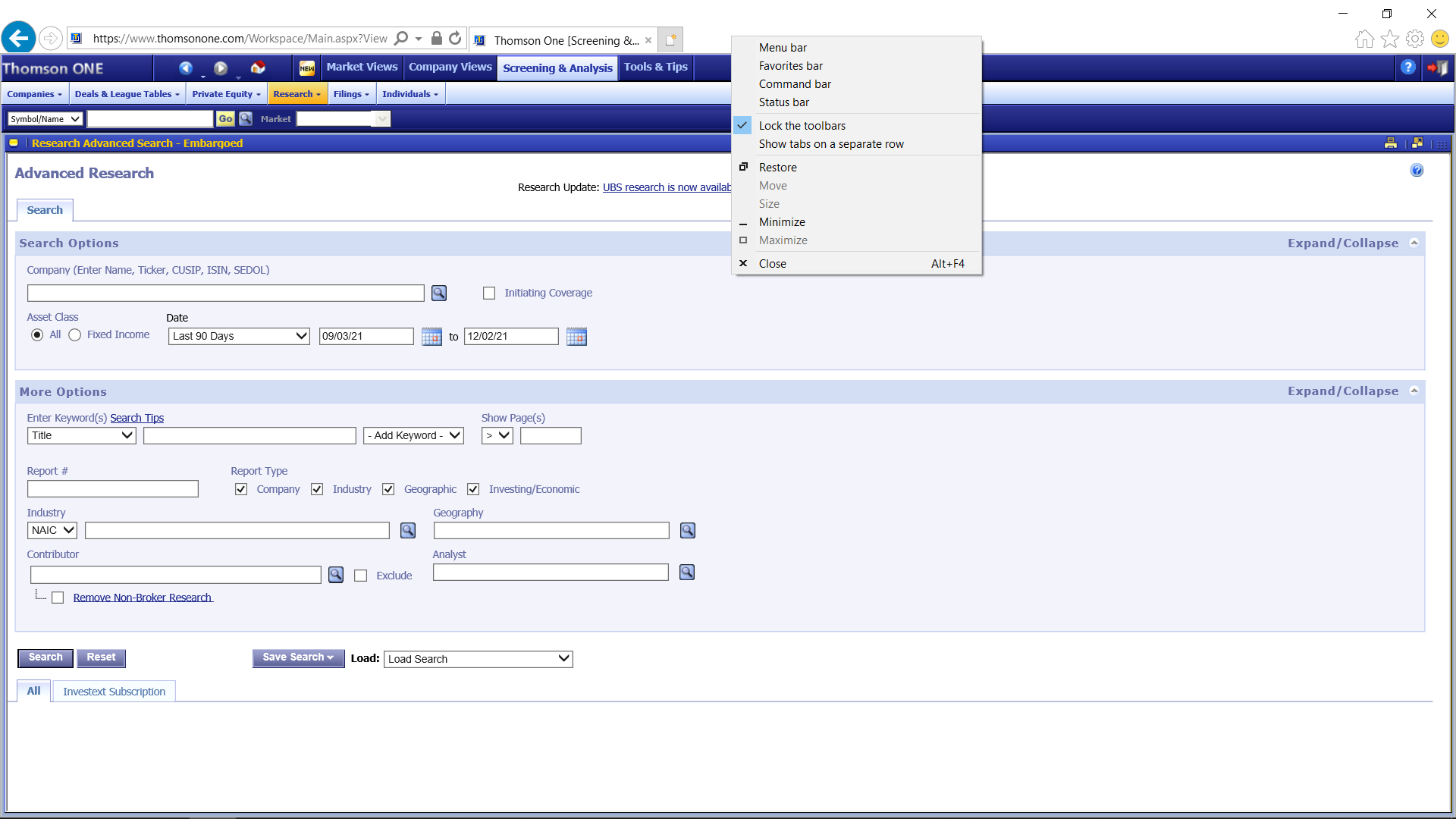
**1.1 Download Thomson One’s Conference Calls [L]**

The goal is to download conference calls from Thomson One. This includes both the pdf files containing the actual calls, and xls files containing identifiers.

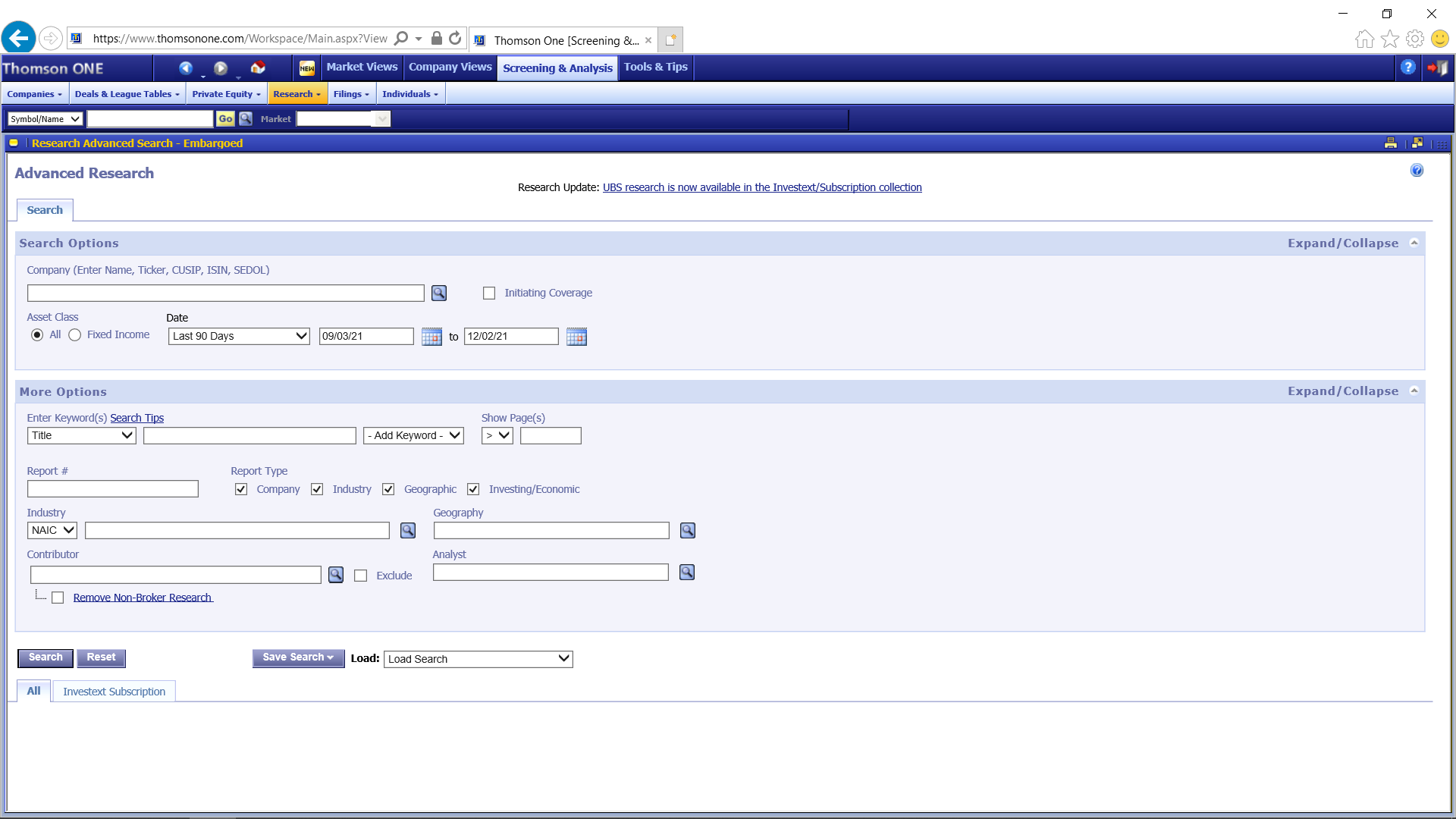
The main obstacles are that: (1) each page will only show 50 conference calls, (2) a maximum of 2,000 conference calls will be presented for every search, (3) web drivers are prohibited / web scraping doesn’t work. The current solution is thus to write a python file that does auto-clicking, and this is automatic\_download.py.

**Before running the code:**

* Open Internet Explorer (no other browers are allowed) to access Thomson One (proxy.uchicago.edu/login/thomsonone).
* Ensure that your browser settings are configured to enable the code to work. The goal is to hide away extraneous elements on the screen, so that no scrolling is needed to be able to click on all co-ordinates.
* The current code works for the Booth laptop, Lenono Thinkpad X1 Extreme (Windows 10) that is not connected to a HDMI screen. If it is connected to another screen, the current saved co-ordinates will likely be off. The screen resolution details (https://whatsmyscreensize.com/):
  + Screen Resolution
  + Width: 1920
  + Height: 1080
  + Device Pixel Ratio: 1.25
  + Display Dimensions (width x height): 16.0" x 9.0"
  + Screen Diagonal: 18.4" Screen
* The settings that worked for this set-up are:
  + Windows Taskbar: “Automatically hide the taskbar in desktop mode” is turned on
  + Internet Explorer settings: 125% zoom + the following



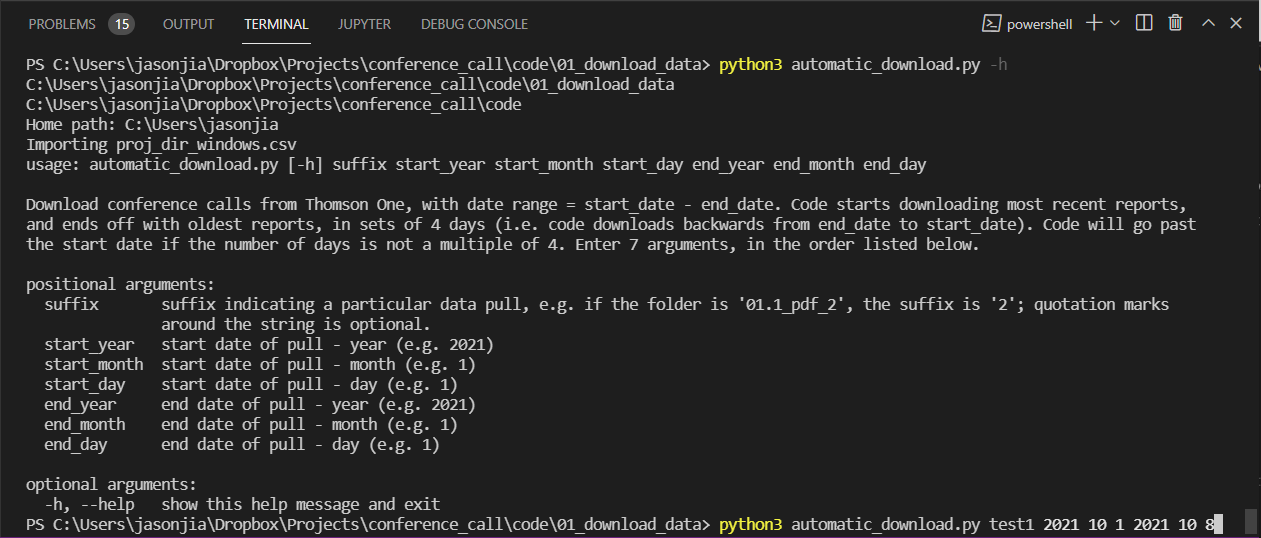
* + So that the screen looks like this:



* Create the necessary folders to store your output
  + Store the folders in: conference\_call\output\01\_download\_cc
  + The root names are 01.1\_pdf and 01.1\_xls.
  + The suffix is used to separate different data pulls. Decide on a suffix.
  + Then create the two folders: “01.1\_pdf\_[suffix]” and “01.1\_xls\_[suffix]”.

**To run the code:**

* Open a terminal.
* Cd to “conference\_call\code\01\_download\_data”.
* Run automatic\_download.py, specifying the suffix, the start date (year, month, day) and the end date year, month, day):
  + E.g. python3 automatic\_download.py test1 2021 10 1 2021 10 8

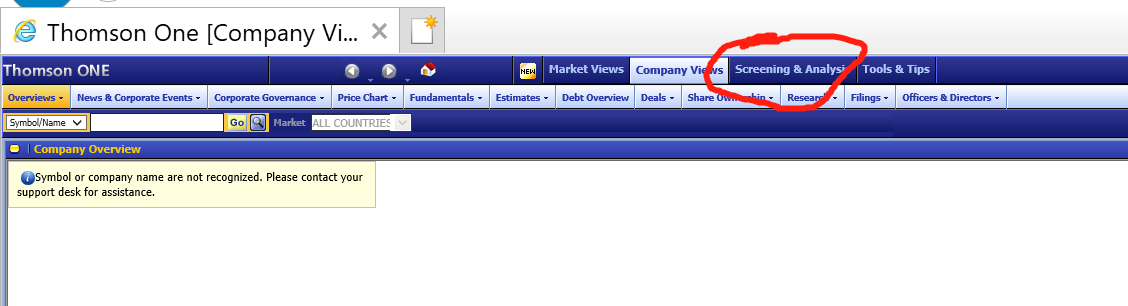


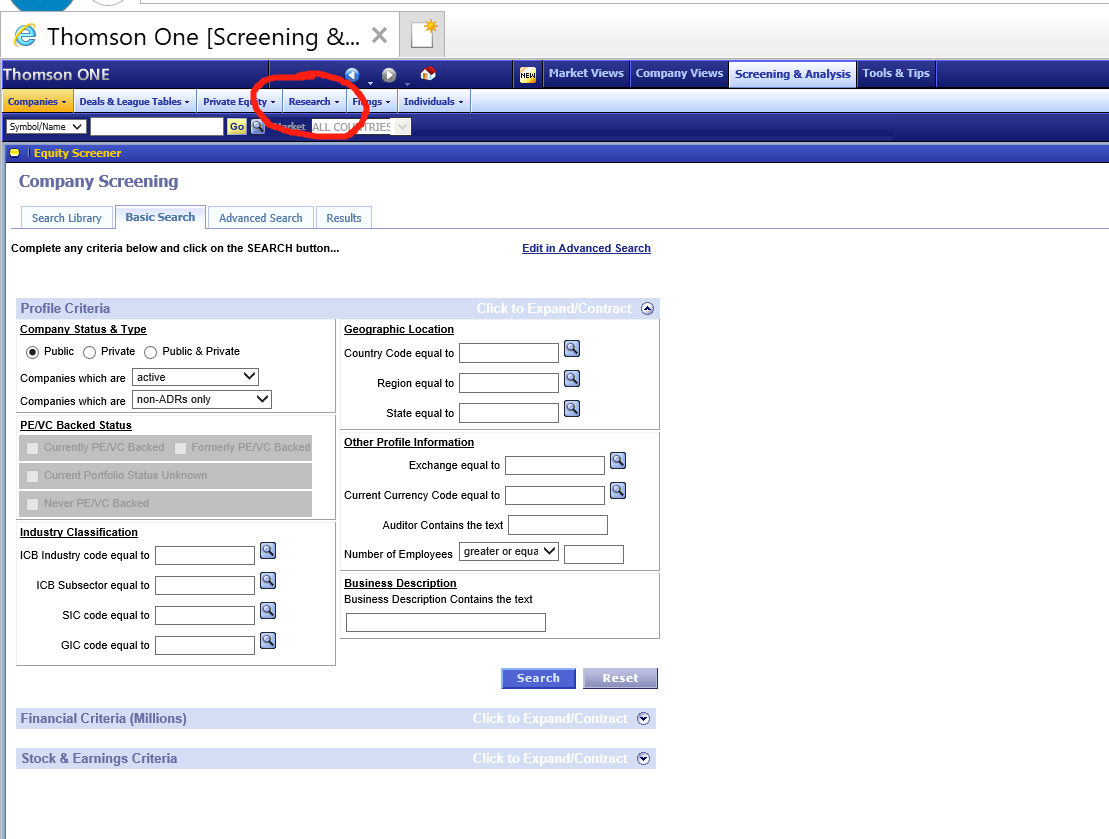
**When running the code:**

The main loop of the code tries to do the following:

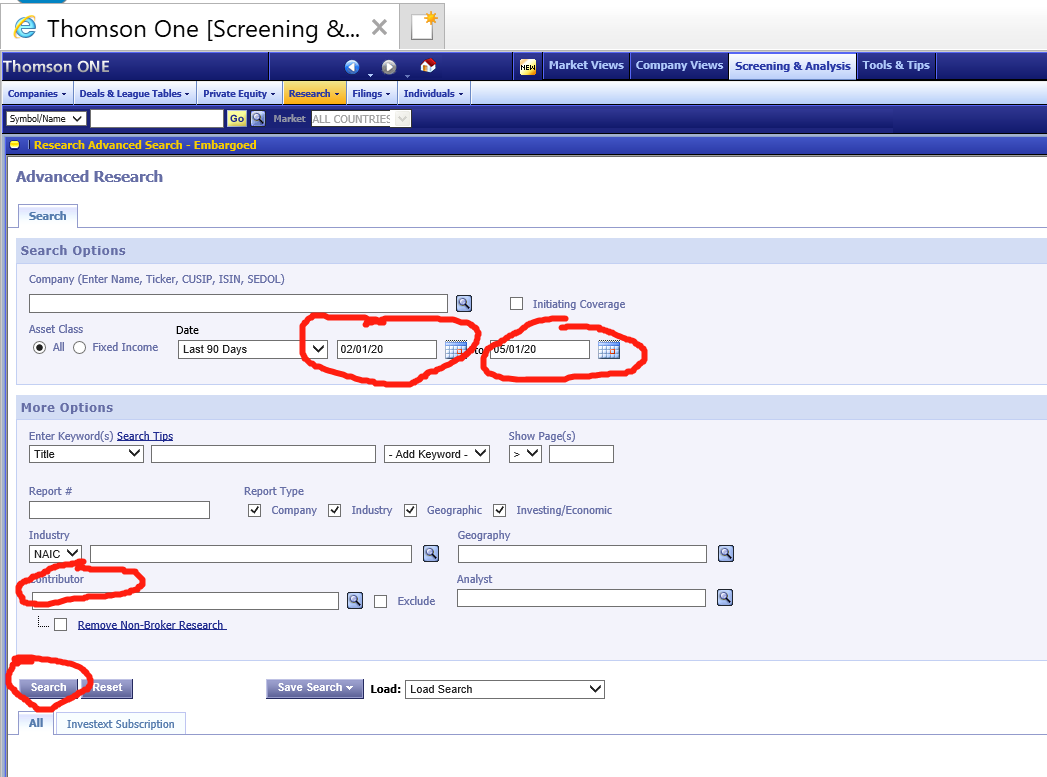
(a) Enter search details

* Click on Screening & Analysis -> Research



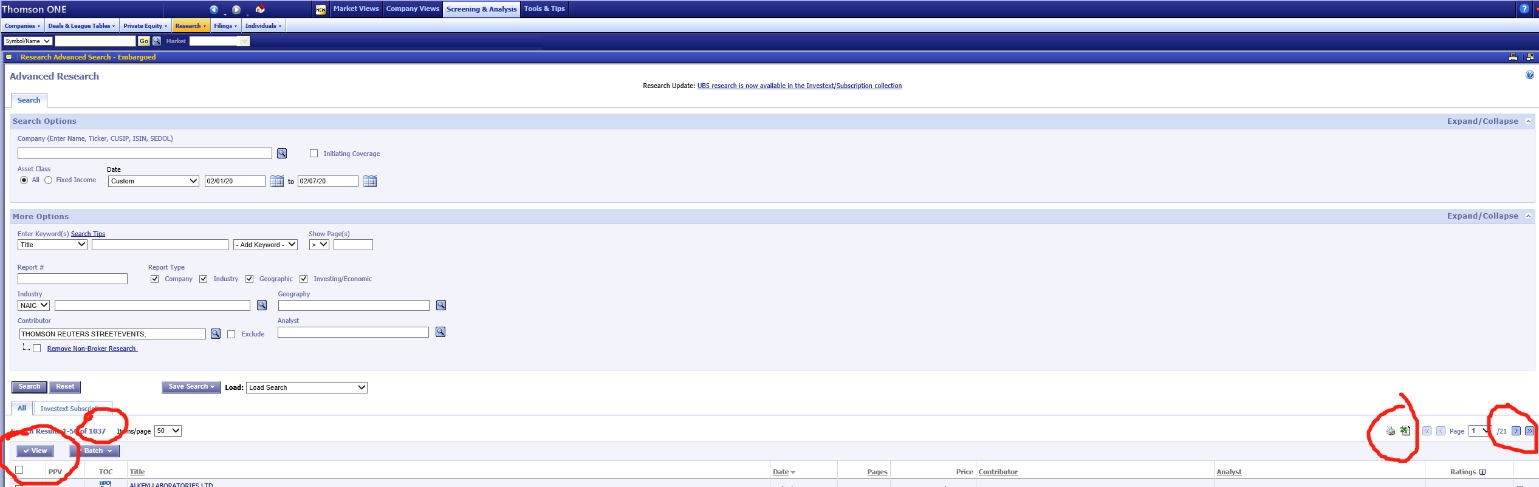


* Click on the Contributor field
* Type Streetevents in the Contributor field
* Click “Refinitiv Streetevents” in the drop-down list
  + Used to be “Thomson Reuter Streetevents” and may be something else in the future. More importantly, there should only be one option containing Streetevents, and it should be the correct option.
* Click on the start date field
* Type the start date
* Click on the end date field
* Type the end date
  + A 4-day time interval is used to ensure that each search gives no more than 2,000 calls.
* Press enter, which is equivalent to clicking the search button.



(b) Download the pdfs and xls files

* Copy the number of calls given by the search, save the number into a separate data set, and calculate the number of pages.
* Select all calls in one page
* Download the pdf file with all calls in that page into 01.1/pdf\_[suffix]
* Download the xls file which contains information of each call into 01.1/xls\_[suffix].
* To ensure that pdf and xls files are in pairs, check the existence of files in the folders.
* If there are some errors, the code will restart the process (by typing in the login websites, and automatically finishing the access steps.
* Uncheck all calls, click next page, repeat (b). If this is the last page of the date period, go back to (a) with date moving backwards.



Errors Handling

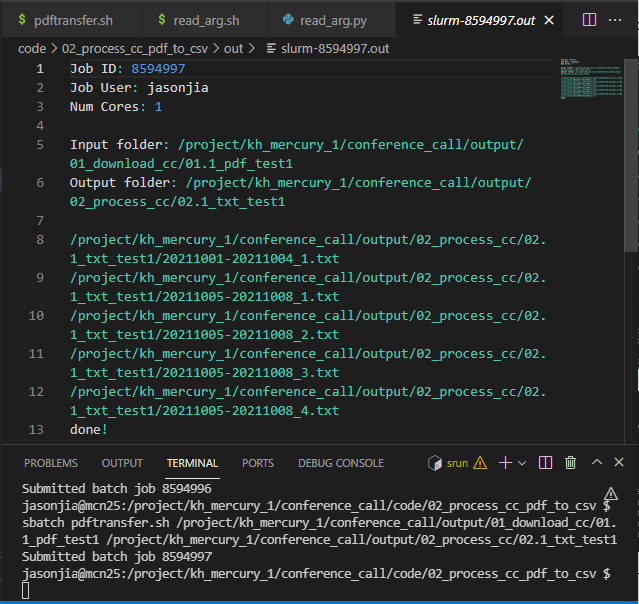
Errors can happen for many reasons, e.g. (1) automatic log off, (2) sudden network error, (3) system authentication error (log in failure), (4) change of file orders in the subsequent login, and (5) broken or corrupted files, and (6) unsuccessful download of files. The point is that some “manual coaxing” is necessary to help the code run smoothly from start to end.

It is possible to try to account for all the errors, but from practical experience, the benefit of fewer errors, relative to the cost of more complex and harder-to-maintain code, diminishes quickly. We thus choose the following approach:

* If files fail to download, there is a time interval where the code will pause, for you to manually click to the correct state. Then, the code will try to save the file again.
* For all other errors, stop the code, get back into a workable state, and then rerun the code with a now truncated time frame.

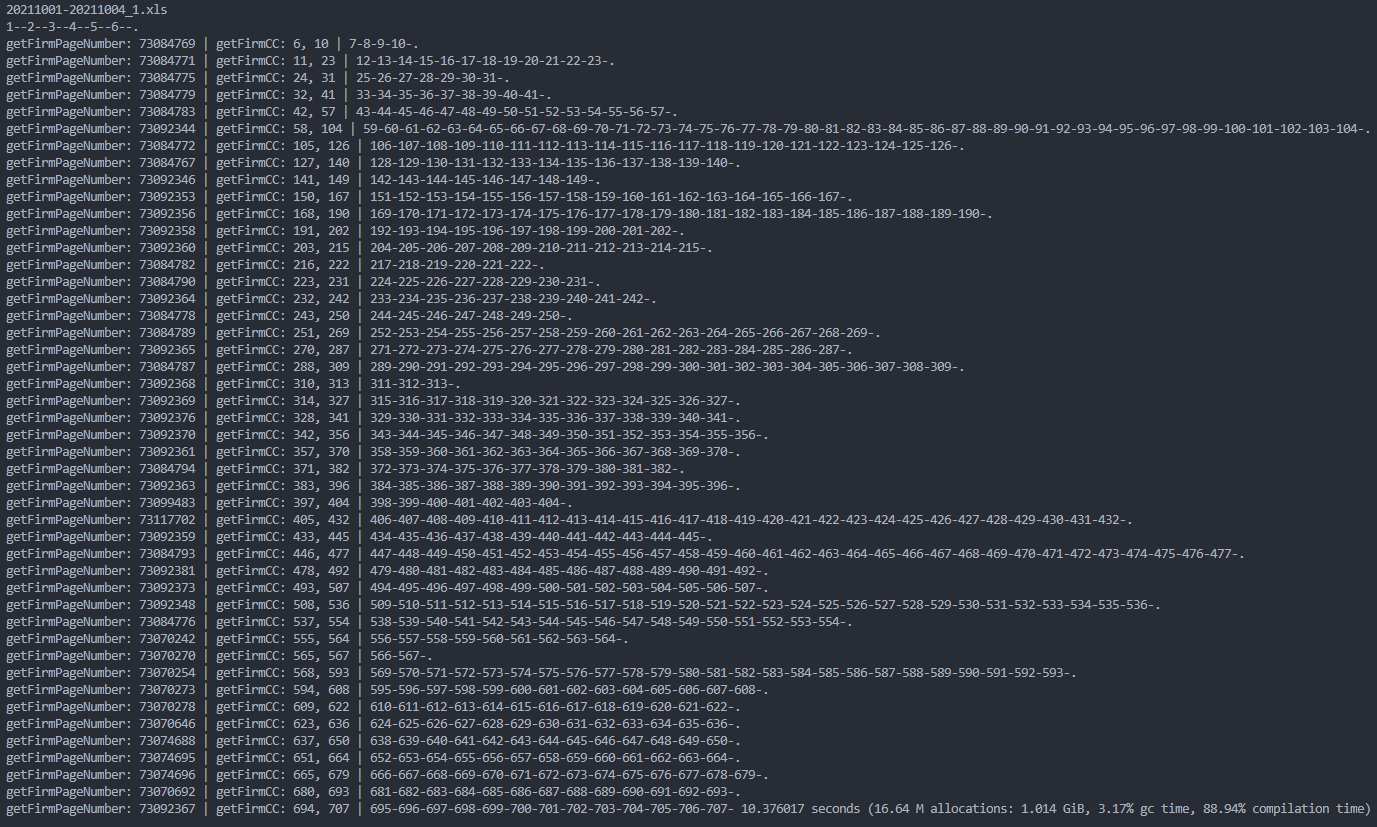
## **2. PDF Processing**

**2.1 Convert Conference Calls from .pdf to .txt [M]**

* Copy over the pdf and xls files to Mercury
* Run pdftransfer.sh on Mercury, specifying the full location of the input folder and the output folder
  + cd “conference\_call/code/02\_process\_cc\_pdf\_to\_csv”
  + sbatch pdftransfer.sh [input folder] [output folder]
* The important command is pdftotext, a Linux command that converts pdf to txt files. It also adds page and paragraph delimiters, which helps split the txt files by conference call later on.
* The txt files will be in the output folder

**2.2 Split Conference Call .txt files to separate out individual conference calls, and combine with report information from .xls files [M, L]**

* Copy over the txt files to Dropbox
* Run ParseCCpdf.jl
* Print messages have been added so you can see how the pdf files are processed. An example is given below:



* The output will be csv files that can be thought of as .xls files combined with the conference call text. This serves as the “primary database” containing the following variables: Title (firm name), Subtitle (firm name, date, and whether final/primary transcripts), Date, Pages (the number of pages of the call), Analyst (analysts who collect these transcripts, different analysts have slightly different forms of transcripts.), Report (Unique report number), Call (Raw call transcripts).
* The code:
  + Start from the delimiters contained in the .txt file to identify pages
  + Use title and pages in the information file to locate the beginning and end of each conference call.
  + Generate a new variable in the information file to store the raw call scripts.

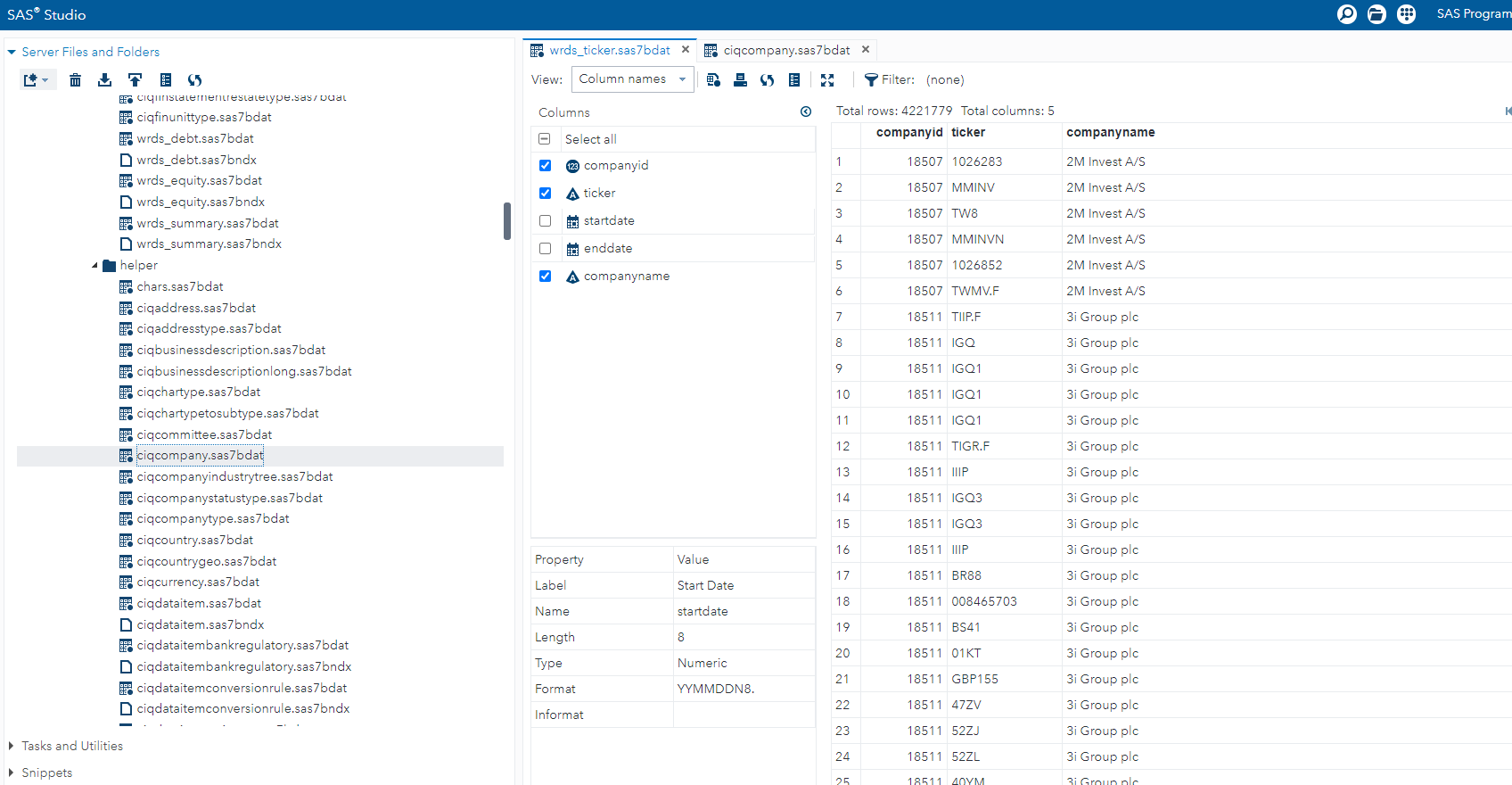
## **3 Firm Identification (Firm Name and Gvkey Matching)**

**3.1 Download Compustat datasets [L]**

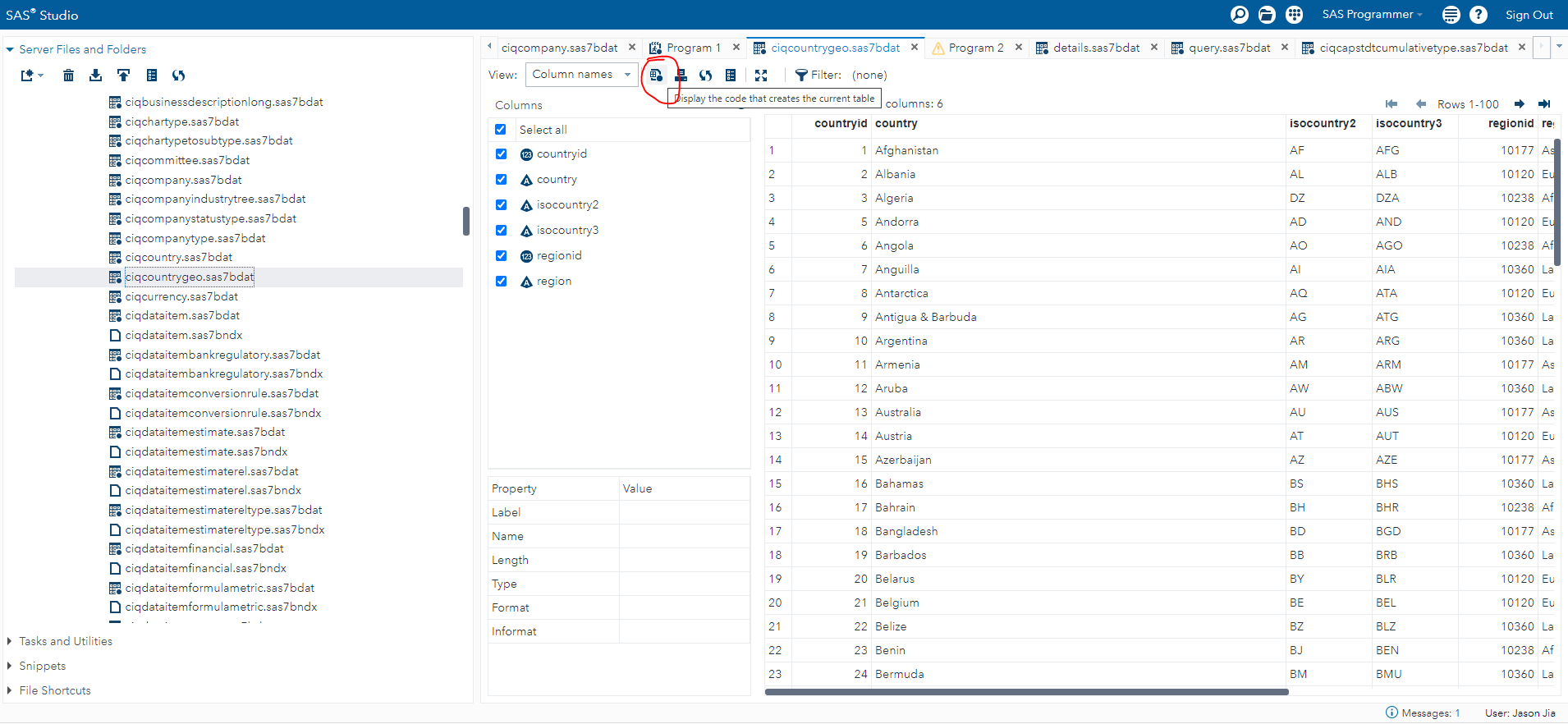
* This step aims to match firms in conference calls to gvkeys, a unique firm identifier, as well as country information. In the xls (and csv) files, the ‘title’ variable gives the firm name associated with a particular conference call.
* Gvkeys are found in the Compustat – Capital IQ datasets. Access the database using Wharton Research Data Services (WRDS).

Steps:

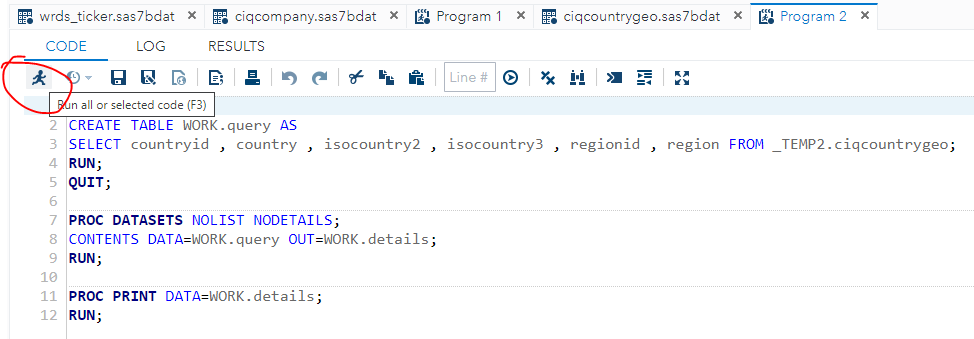
* Register for an account and wait for approval by the IT team: <https://wrds-www.wharton.upenn.edu/register/>
* Sign into the SAS-studio web application: <https://wrds-www.wharton.upenn.edu/pages/data/sasstudio-wrds/>
* On the left, there is a folder directory, titled “Server Files and Folder”.
* Capital-IQ auxiliary files are located at Files -> wrds/capitaliq/sasdata/helper.



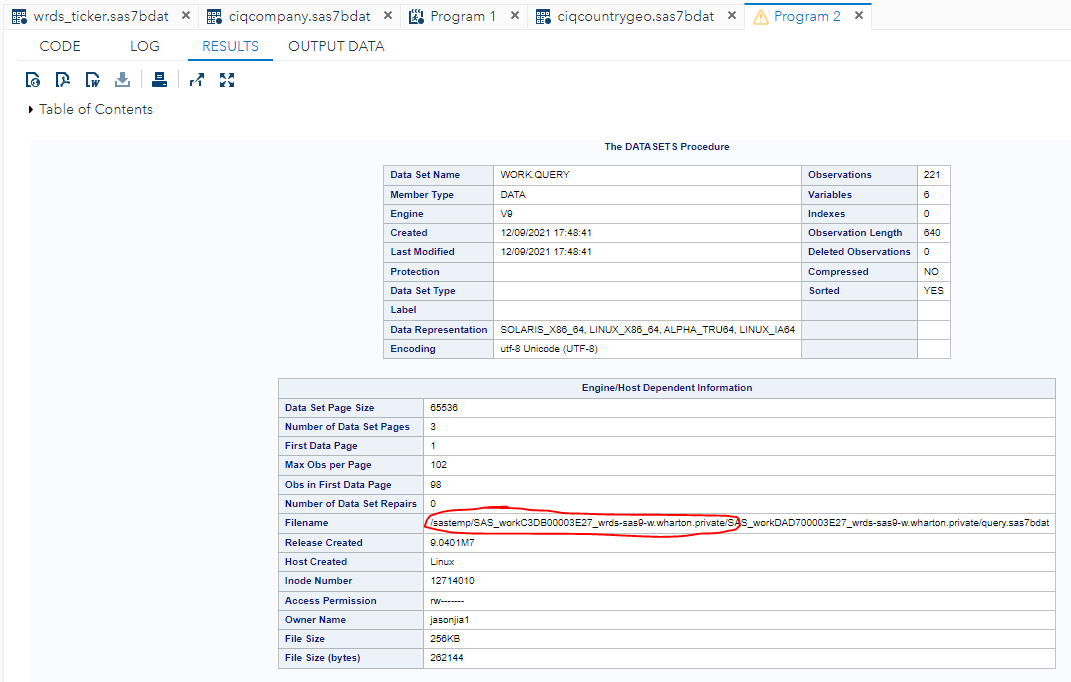
* Open the desired table.
* Select the desired columns.
* The best way to download data is to create Query (right mouse button on a table ⇒ new ⇒ Query).
* Downloading Query’s result is a little bit tricky, since you can only print the result. The result is located in user’s temporary folder.
* Click the button to “display the code that creates the current table”.



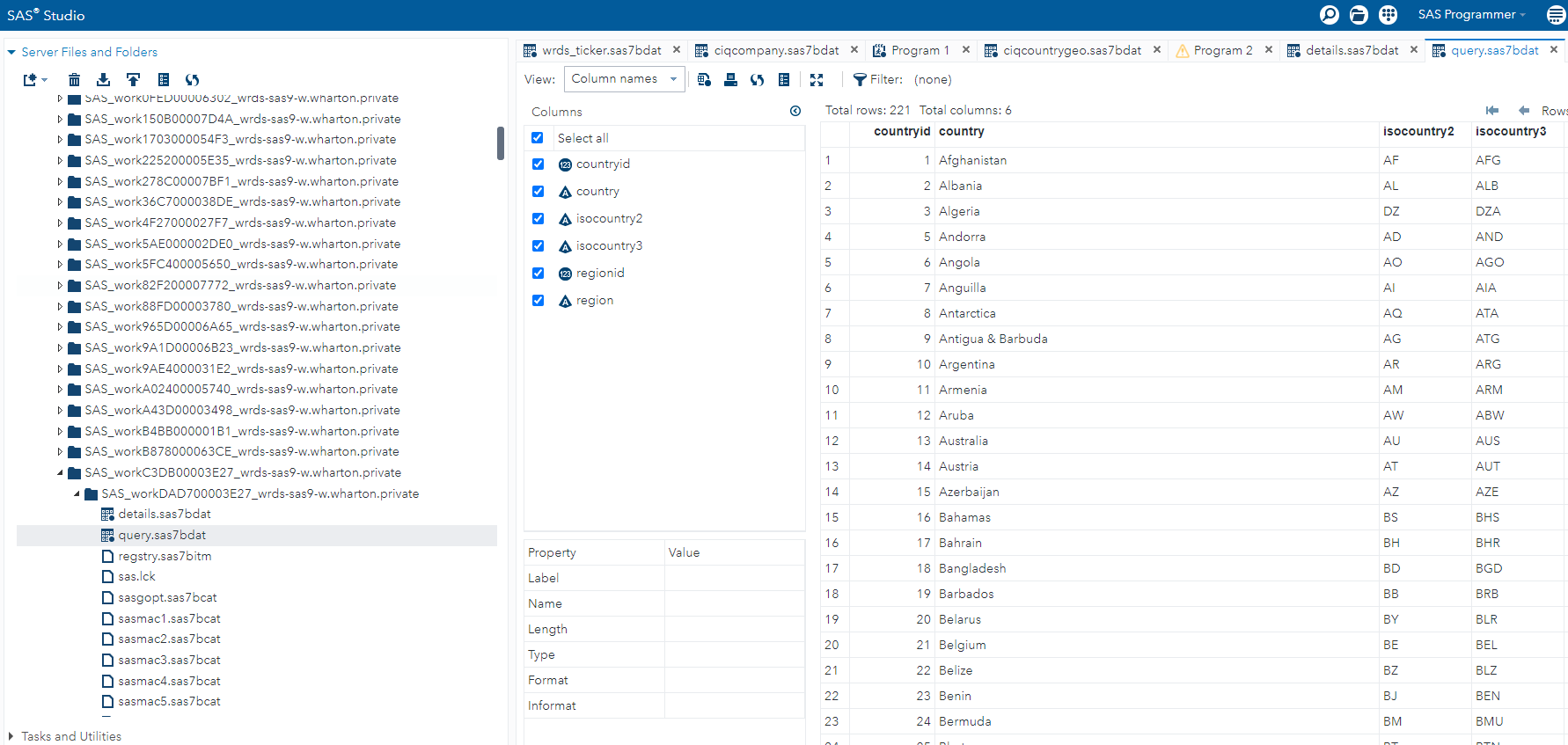
* 3) run the code



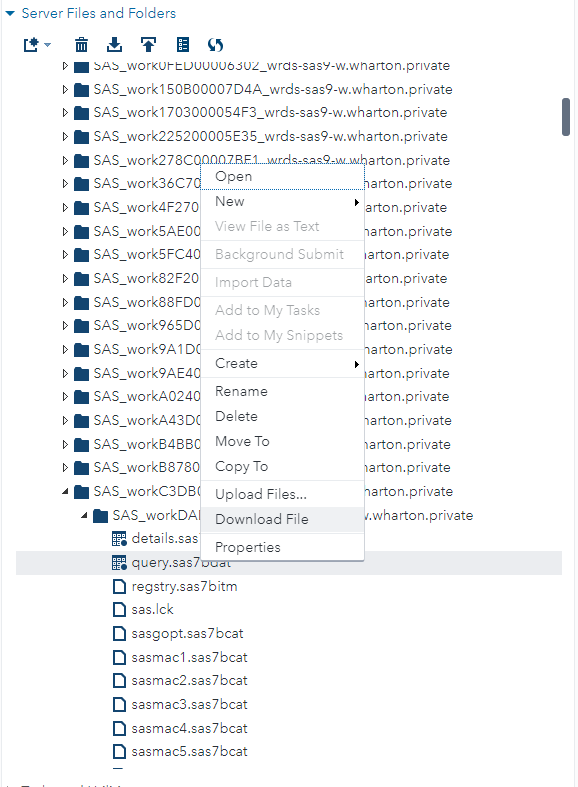
* 4) Result window → Engine/Host Dependent Information → filename shows your temporary folder.



* Go to the temporary folder and open query.sas7bdat to confirm this is the dataset you want to download.



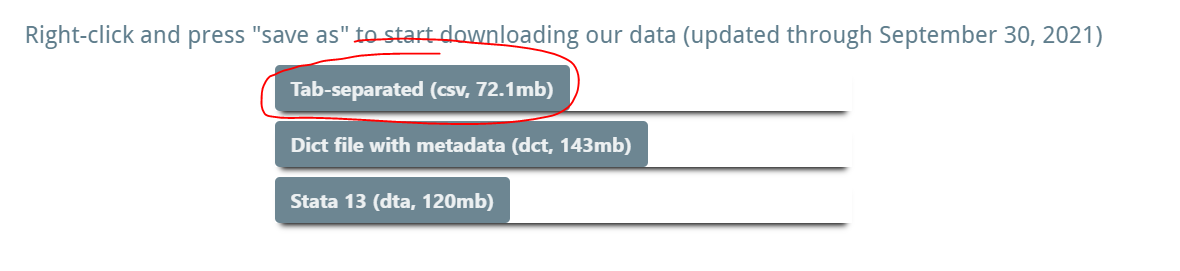
* Right click on query.sas7bdat and click download file.



* We used the following tables with specific columns from Compustat *[to update]*:
  + 1. CIQCOMPANY: companyid, companyname, tickersymbol, countryid and other columns.
  + Total: 24,511,757 obs. Public Companies: 66,256. Private Companies: 16,544,322. Public Investment Firms: 1,987 and Private Investment Firms: 203,407
  + 2. WRDS GVKEY: companyid, gvkey (115,357 observations).
  + 3. CIQCOUNTRY: countryid, countryname: countryid, countryname (221 countries).

**3.2 Download Hassan dataset [L]**

* Gvkeys are also found in Hassan’s Firm-Level Political Risk dataset.
* Go to <https://www.firmlevelrisk.com/download> and download the tab-separated file in csv.
* This data set was also used because it also uses conference calls, but have already matched firm names to gvkeys, which would help in our firm name matching. Note that the dataset is updated over time.
* This data set is updated every so often – the most recent version as of writing is 2021-09-30.

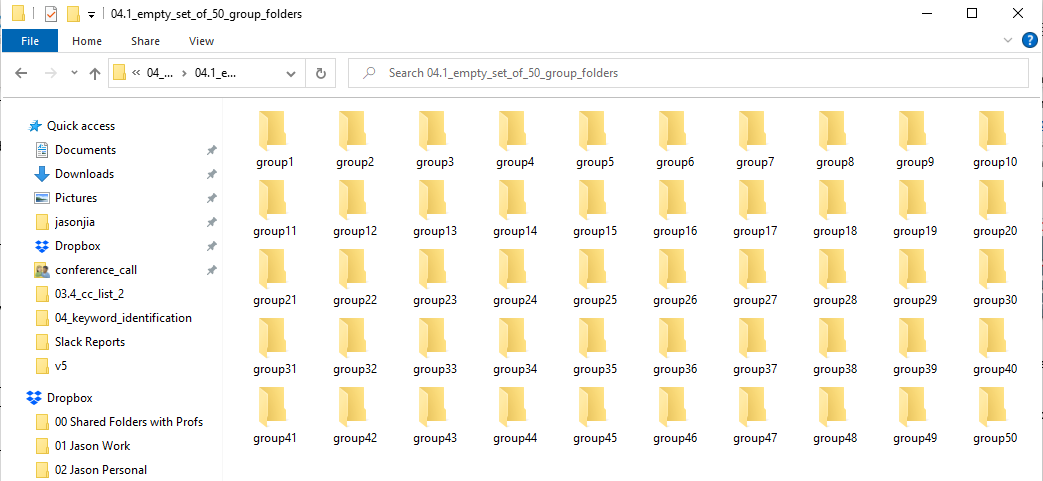


**3.3 Process Compustat and Hassan datasets into usable and truncated .csv files. [L]**

* Go to conference\_call\code\03\_firm\_identification
* Run the codes convert\_compustat.py on terminal and process\_hassan.py on terminal.

**3.4 Make a folder structure with x groups and move .csv files into the folders (default: x = 50)**

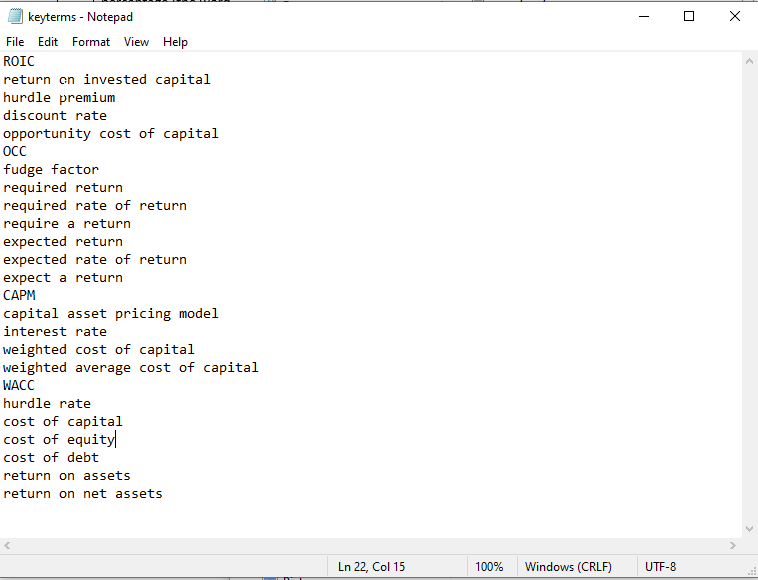
* Run mkdir.py
* Output will be a folder containing 50 empty sub-folders, named group[i]:



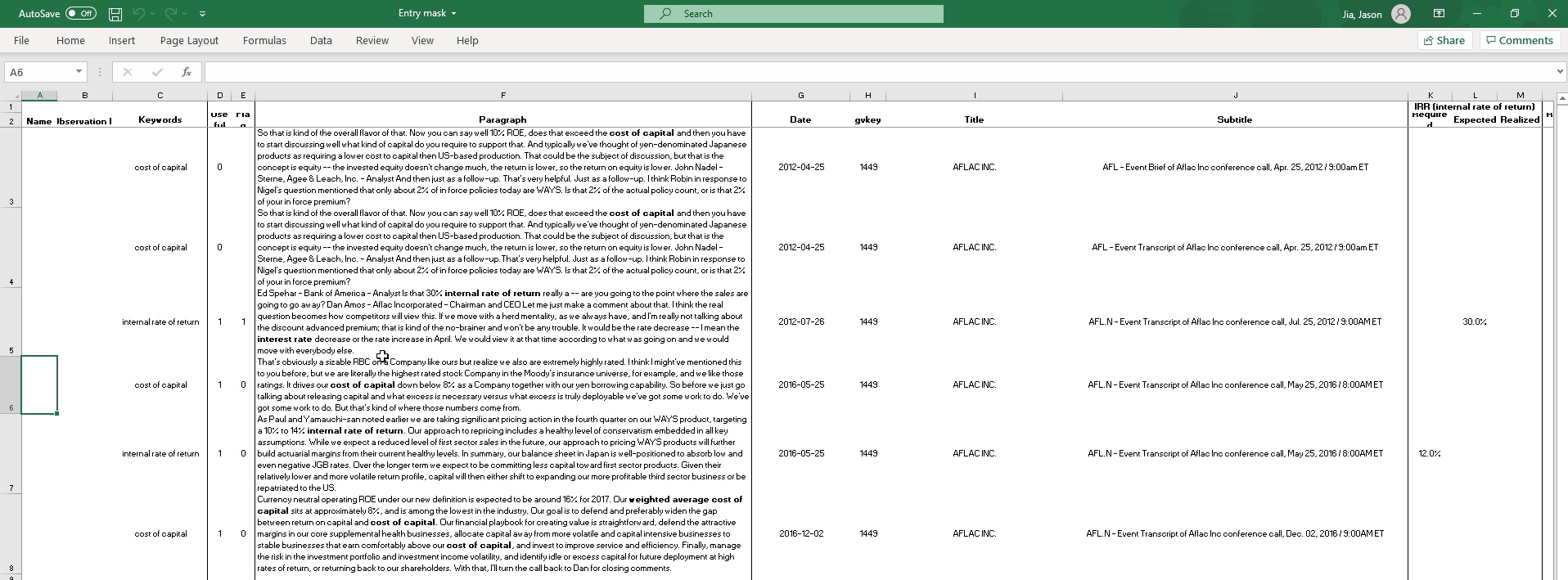
* Run dividefilesequallyinto folders.py
* This will copy the csv files from your previous folder containing csv files, divide it equally between the 50 sub-folders, and paste them into the allocated sub-folder.

**3.5 Make a list of keywords and template entry file [L]**

* Create keyterms.txt, a txt file containing all keywords you want to search for in conference calls.

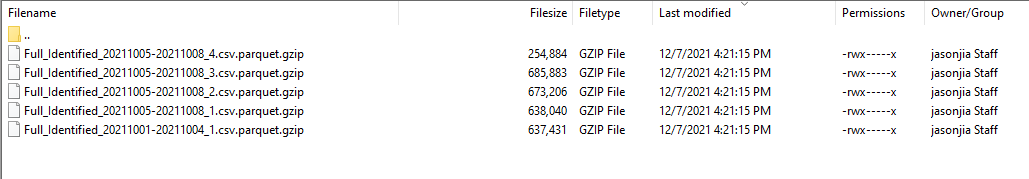


* Create Entry mask.xlsx, a template for entry files



**3.6 Identify keywords for the whole CC data.**

* Run CC\_identify\_keywords.py
* The output will be a set of folders Full\_Identified\_Keywords/group[i]/ Full\_Identified\_[i].parquet.gzip



**3.7 Concatenating all these files into a single dataset.**

* Run concatenateOutputs.py
* Output of focus is Full\_Master\_Keywords.csv

**3.8 Filter based on a more exact keyword identification algorithm (rather than just checking in, doing a holistic check by looking at the spaces around the keyword)**

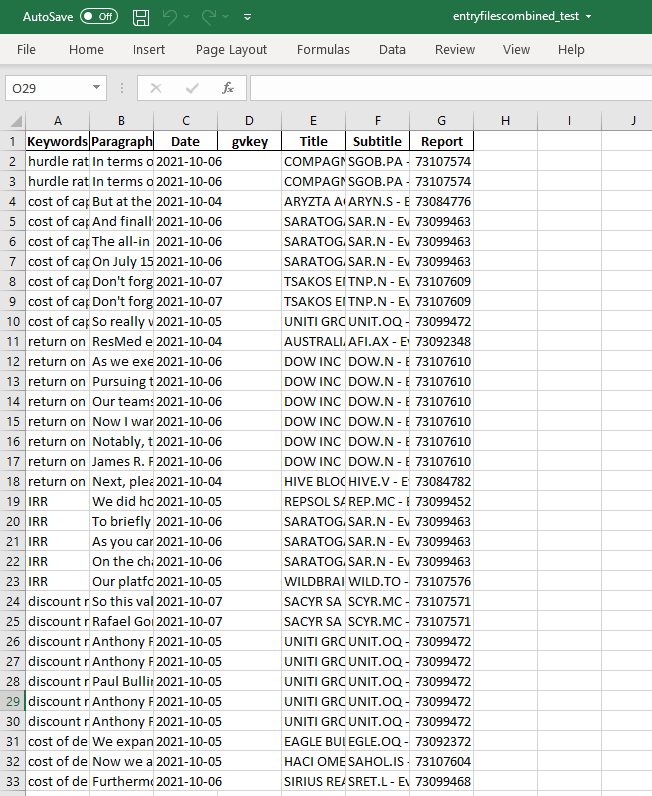
* Run getCorrect.py
* Output is Amended\_Correct\_No\_IR.csv

**3.9 Filter based on the presence of a percentage (the words percent, per cent, percentage, %) and then order based on the sorting rule.**

* Run ordering\_and\_filtering.py
* Output is Filtered\_Ordered\_Amended\_Correct\_No\_IR.csv

**3.10 Convert current paragraphs and conference call information into entry files format**

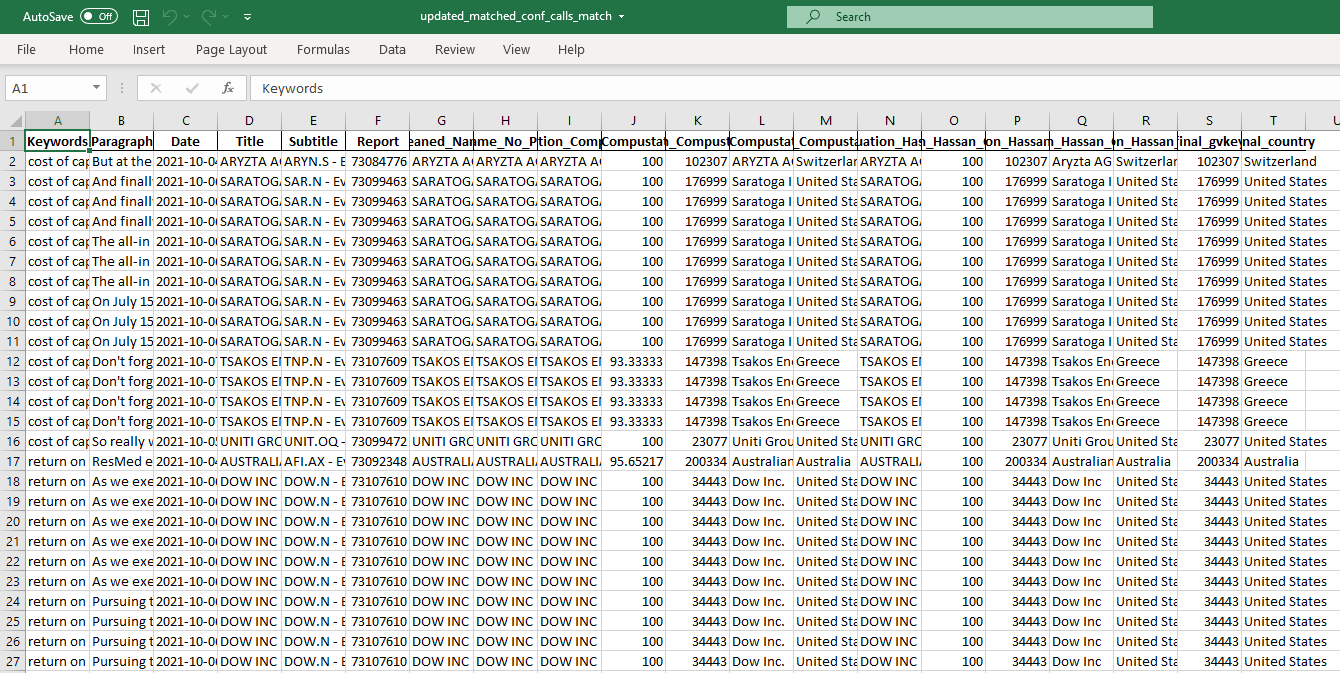
* Run the following codes, in order:
  + convertFilteredOrderedAmendedCorrectNoIR\_to\_TotalCircNew.py
  + convertTotalCircNew\_to\_Cric1newtotal.py
  + convertCric1newtotal\_to\_entryfilescombined.py
* Output: entryfilescombined\_withoutgvkey.xlsx



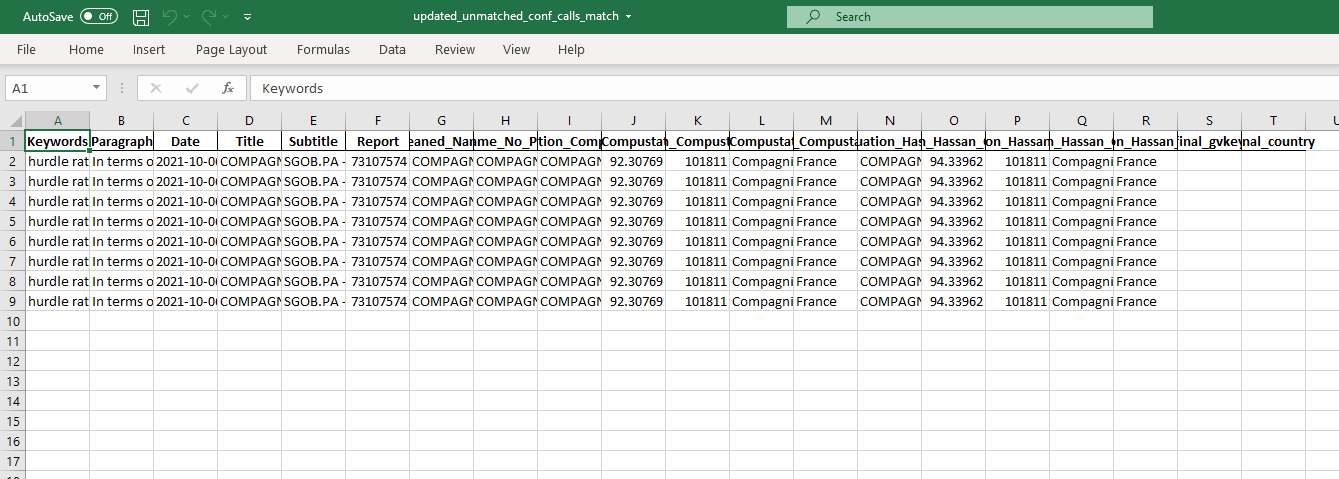
## **4 Firm Identification (Firm Name and Gvkey Matching)**

**4.1 Perform the fuzzy matching between the Hassan/Compustat and the CC datasets.**

* Run the following codes, in order:
  + cc\_fuzzy\_match\_part1.py
  + cc\_fuzzy\_match\_part2.py
  + Use highmem on Mercury for this, as the files are extremely large.
* Output:
  + updated\_matched\_conf\_calls\_match.xlsx



* + updated\_unmatched\_conf\_calls\_match.xlsx



**4.2 Do manual matching for unconfirmed cases**

* 1 for matches, 0 for non-matches
* Output:
  + Filled\_Updated\_CC\_Compustat\_FuzzyMatchCandidates.xlsx
  + Filled\_Updated\_CC\_Hassan\_FuzzyMatchCandidates.xlsx

**4.3 Combine manually matched cases with results from fuzzy matching**

* Run cc\_fuzzy\_match\_part2.py (the commented out section)
* Output: manual\_full\_updated\_conf\_calls.xlsx

**4.4 Make a paragraph record file that splits the number of entries into groups of 500 [L]**

* Run make\_paragraphrecord.py and add the number of entries as an argument
* Output: paragraphrecord.xlsx

**4.5 Bold the keywords and separate file into “entryfiles”, each containing 500 entries. [M, L]**

* Run makeentryfiles.py
* Output: A set of entryfiles, [i].xlsx

**4.6 Combine entry files [L]**

* Run combine\_entryfiles.py
* Output: entryfiles\_combined.xlsx

## **5 Get Front Page Descriptions**

**5.1 Extract front page descriptions from conference calls [M, L**]

* Run extractdescriptioninfrontpage.py
* Helpful code: copyfiles.py
* Output: [yyyymmdd-yyyymmdd]\_withfrontpagedesc.xlsx

**5.2 Manually check through error cases and correct accordingly [L]**

* Output: [yyyymmdd-yyyymmdd]\_withfrontpagedesc.xlsx

**5.3 Combine xls files [L]**

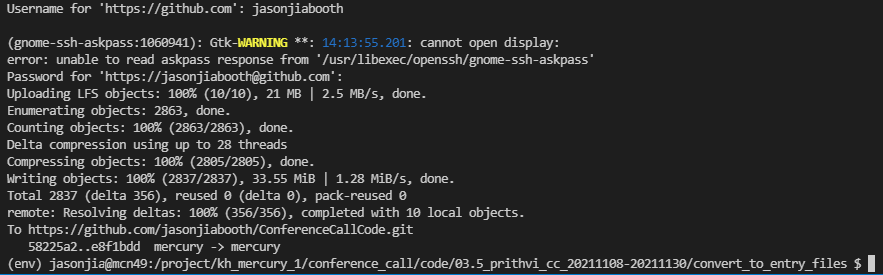
* Run combine\_xlsfiles\_withdescription.py
* Output: xlscombined\_withfrontpagedescription.xlsx

**5.4 Match and add front page descriptions to combined entry files [L]**

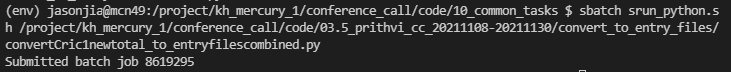
* O33: entryfiles\_combined\_withfrontpagedesc.xlsx (updated)

## **Misc**

* Working with Git Large File Storage (LFS):



* A .sh file to run python files: srun\_python.sh
  + Useful because you won’t have to make a new .sh file for each .py file you want to run, when the only thing that changes is the location and name of the python file.
  + Optional: Copy it into a folder for easier use. You can edit the settings for your purposes too (e.g. highmem)
  + Ensure that your current directory has an out folder to contain the out files



**Summary of Main Processing Pipeline (Previous Version, based on codes by Sixun and Valerii)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Task | Codes | Input | Output |
| **1** | **Download Raw Data** | | | |
| 1.1 | Download Thomson One’s Conference Calls [L] | mouse\_key\_recorder.py  automatic\_download  .py | - | O1: [yyyymmdd-yyyymmdd].pdf  O2: [yyyymmdd-yyyymmdd].xls |
| **2** | **PDF Processing** | | | |
| 2.1 | Convert Conference Calls from .pdf to .txt [M] | pdftransfer.sh | O1 | O3: [yyyymmdd-yyyymmdd].txt |
| 2.2 | Split Conference Call .txt files to separate out individual conference calls, and combine with report information from .xls files [M, L] | ParseCCpdf.jl | O3, O2 | O4: [yyyymmdd-yyyymmdd].csv |
| **3** | **Firm Identification (Firm Name Matching)** | | | |
| 3.1 | Download Compustat datasets [L] | - | - | O5: ciqcompany.sas7bdat  O6: ciqcountrygeo. sas7bdat  O7: wrds\_gvkey.sas7bdat |
| 3.2 | Download Hassan dataset [L] | - | - | O8: Hassanfile\_raw\_  updated20219030.csv |
| 3.3 | Process Compustat and Hassan datasets into usable and truncated .csv files. [L] | convert\_sas7bdattocsv.py  join\_compustatcsvfiles.py  hassanfilecsv\_viewable\_  truncate.py | O5, O6, O7, O8 | O9: ciqcompany.csv  O10: ciqcountrygeo.csv  O11: wrds\_gvkey.csv  *O12\*: ciqcompany\_merged*  *withgvkeyandcountry.csv*  O12: Hassanfile\_raw\_  updated20219030 \_truncated.csv |
| 3.4 | Match titles in conference calls with firm names in Hassan and Compustat datasets, with both exact and fuzzy matching [M, L] | linkCCtoGvkey.jl | O9, O10, O11, O12 | O13: CC\_List[yyyy].csv  O14: CC\_List\_2020-2021.csv |
| **4** | **Keyword Identification** | | | |
| 4.1 | Make a folder structure with x groups (default: x = 50) [M, L] | mkdir.py  dividefilesequallyinto  folders.py | - | - |
| 4.2 | Make a list of keywords and template entry file [L] | - | - | O15: keyterms.txt  O16: Entry mask.xlsx |
| 4.3 | Identify keywords in each conference call [M, L] | keyword\_ident\_1.py  keyword\_ident\_1.sh | O15, O4 | O17: FR5.csv |
| 4.4 | Extract all paragraphs from each conference call that contains a specific keyword [M, L] | keyword\_ident\_2.py  keyword\_ident\_2.sh | O17 | O18: TotalCircnew.xlsx |
| 4.5 | Cleans the identified matches and merges with gvkey dataset [L] | mergeclean.do | O18, O14 | O19: cric1\_newtotal.xlsx |
| 4.6 | Make a paragraph record file that splits the number of entries into groups of 500 [L] | make\_paragraphrecord.py | O19 | O20: paragraphrecord.xlsx |
| 4.7 | Bold the keywords and separate file into “entryfiles”, each containing 500 entries. [M, L] | makeentryfiles.py  makeentryfiles.sh | O19, O20, O15, O16 | O21: [i].xlsx |
| 4.8 | Combine entry files [L] | combine\_entryfilesjason.py  combine\_entryfilessixun.py  combine\_sixunand  jasonentryfiles.py | O21 | O22: entryfiles\_  combined.xlsx |
| **5** | **Get Front Page Descriptions** | | | |
| 5.1 | Extract front page descriptions from conference calls [M, L] | extractdescriptioninfront  page.py  extractdescriptioninfront  page.sh  copyfiles.py  copyfiles.sh | O21, O2, O3 | O23: [yyyymmdd-yyyymmdd]\_withfront  pagedesc.xlsx |
| 5.2 | Manually check through error cases and correct accordingly [L] | - | O23 | O24: [yyyymmdd-yyyymmdd]\_withfront  pagedesc.xlsx |
| 5.3 | Combine xls files [L] | combine\_xlsfiles\_with  description.py | O24 | O25: xlscombined\_with  frontpagedescription  .xlsx |
| 5.4 | Match and add front page descriptions to combined entry files [L] | - | O25, O22 | O26: entryfiles\_  combined.xlsx (updated) |

\* M = Mercury, L = Local. [M] / [L] means this stage can be run on Mercury / locally (on your Booth Windows laptop) respectively. [M, L] means this stage can be run on both Mercury and your local laptop, where Mercury is preferred for large datasets and local is preferred for initial testing, debugging and small datasets.

**Prithvi’s Additional Part**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Identify keywords for the whole CC data. | CC\_identify\_keywords.py | CC Data on the server at “CriCount/group{X}”  where X = 1, 2, 3 ..50 | XXX1: CC Data with relevant keywords at “/project/kh\_mercury\_1/  CriCount/Full\_Identified\_  Keywords/group{X}” where  X = 1, 2, 3 ..50 |
| 2 | Concatenating all these files into a single dataset. | concatenateOutputs.py | XXX1 | XXX2: Full\_Master\_Keywords.csv |
| 3 | Filter only those entries which were not collected during the first run | optimizedGetNewEntries.py | XXX2 | XXX3: Full\_New\_Not\_Done.csv |
| 4 | Filter based on a more exact keyword identification algorithm (rather than just checking in, doing a holistic check by looking at the spaces around the keyword) | getCorrect.py | XXX3 | XXX4: Amended\_Correct\_No\_IR.csv |
| 5 | Filter based on the presence of a percentage (the words percent, per cent, percentage, %) and then order based on the sorting rule provided. | Ordering And Filtering.ipynb | XXX4 | XXX5: Filtered\_Ordered\_Amended\_  Correct\_No\_IR.csv |
| 6 | Perform the fuzzy matching between the Hassan/Compustat and the CC datasets. | CCFuzzyMatch.ipynb | XXX5, Hassanfile\_raw\_  updated2019030\_viewable.csv, O10.5 | XXX6: manual\_full\_updated\_  conf\_calls.xlsx |

**3.4 Match titles in conference calls with firm names in Hassan and Compustat datasets, with both exact and fuzzy matching [M, L]**

[to be added]

**4 Keyword Identification**

**4.1 Make a folder structure with x groups (default: x = 50) [M, L]**

* Run mkdir.py, specifying the output folder that will contain x groups.
* The output will be x empty sub-folders in the output folder, named group1, …, groupx.

**4.2 Make a list of keywords and template entry file [L]**

* This is done manually. The existing version of keywords and template entry file are found in \conference\_call\output\04\_keyword\_identification\04.2\_reference\_files as keterms.txt and Entrymask.xlsx.
* Changes to keyterms are recorded in changelog.txt. You can also consider adding suffixes to record different sets of keywords.

**4.3 Identify keywords in each conference call [M, L]**

* The csvs will now be copied over and divided equally into 50 (or x) groups, to enable parallelization on Mercury.
* Run dividefilesequallyintofolders.py.
* Then, identify keywords in each conference call.
* Run keyword\_ident\_1.py.
* The output will be

**4.4 Extract all paragraphs from each conference call that contains a specific keyword [M, L]**

* -

**4.5 Cleans the identified matches and merges with gvkey dataset [L]**

* -

**4.6 Make a paragraph record file that splits the number of entries into groups of 500 [L]**

* Run make\_paragraphrecord.py and inputting the number of entries.

**4.7 Bold the keywords and separate file into “entryfiles”, each containing 500 entries. [M, L]**

* If doing locally, run makeentryfiles.py; if doing on Mercury, run makeentryfiles.sh

**4.8 Combine entry files [L]**

* Run combine\_entryfiles.py.