Lab Tutorial

Alan Liu (aliu38@calstatela.edu)

Keith Wang (kwang6@calstatela.edu)

Samuel Park (jpark94@calstatela.edu)

Steven Vallejo (svallej5@calstatela.edu)

December 12, 2018

Analyzing 2017-18 US Spending Data

**Objectives**

* Download data from US Spending government website
* Properly set up table structure using headers in data
* Creating tables in Beeline and managing exceedingly high volumes of data
* Preparing SQL statements for analyzing the data
* Exporting relevant results
* Visualizing exported results using Excel

**Platform Spec**

* Oracle Compute Big Data Cloud Service – Compute Edition
* CPU Speed: 2.2GHz
* # of CPU cores: 10
* # of nodes: 5
* Total Memory Size: 150GB

## Obtaining/Loading the Data

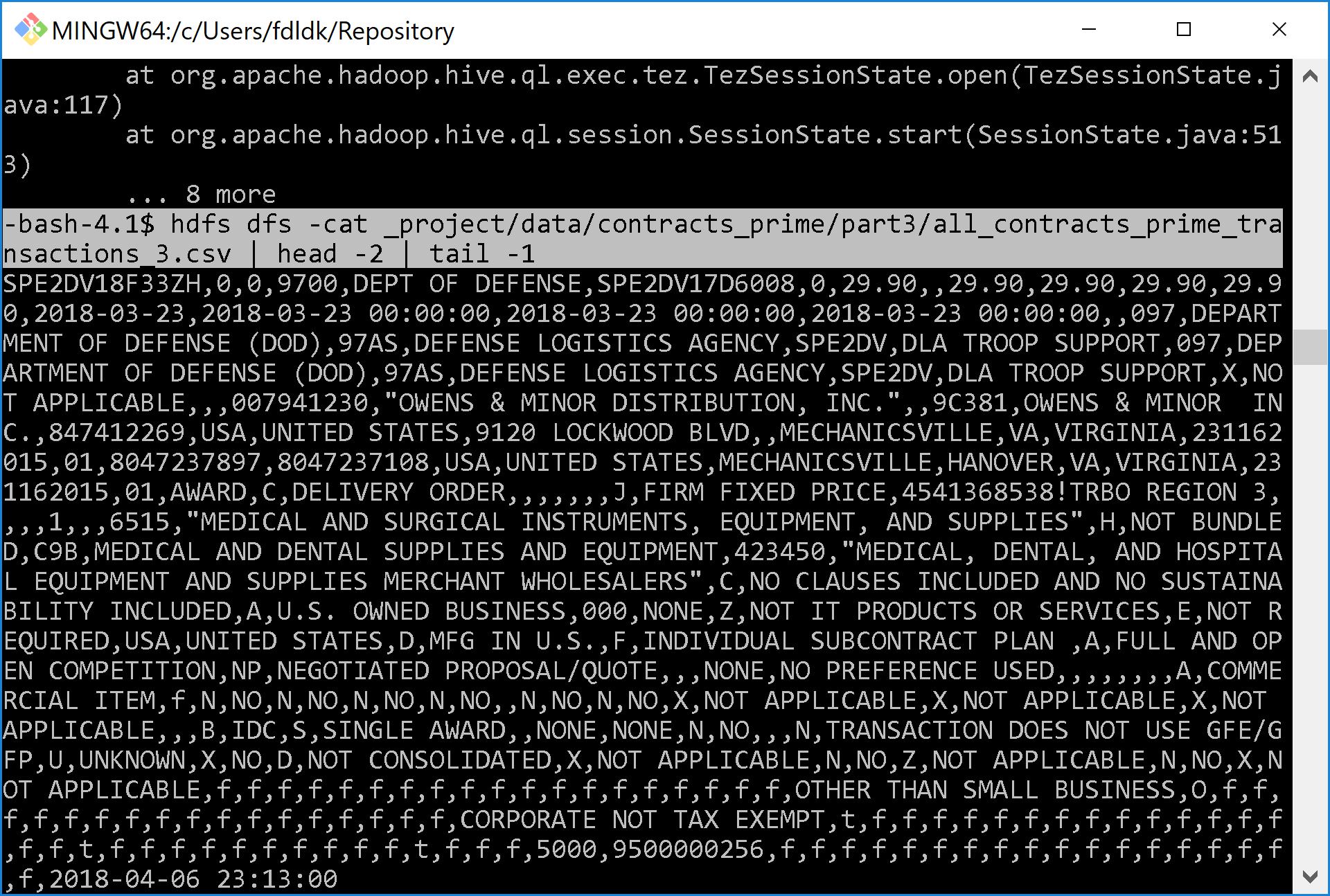
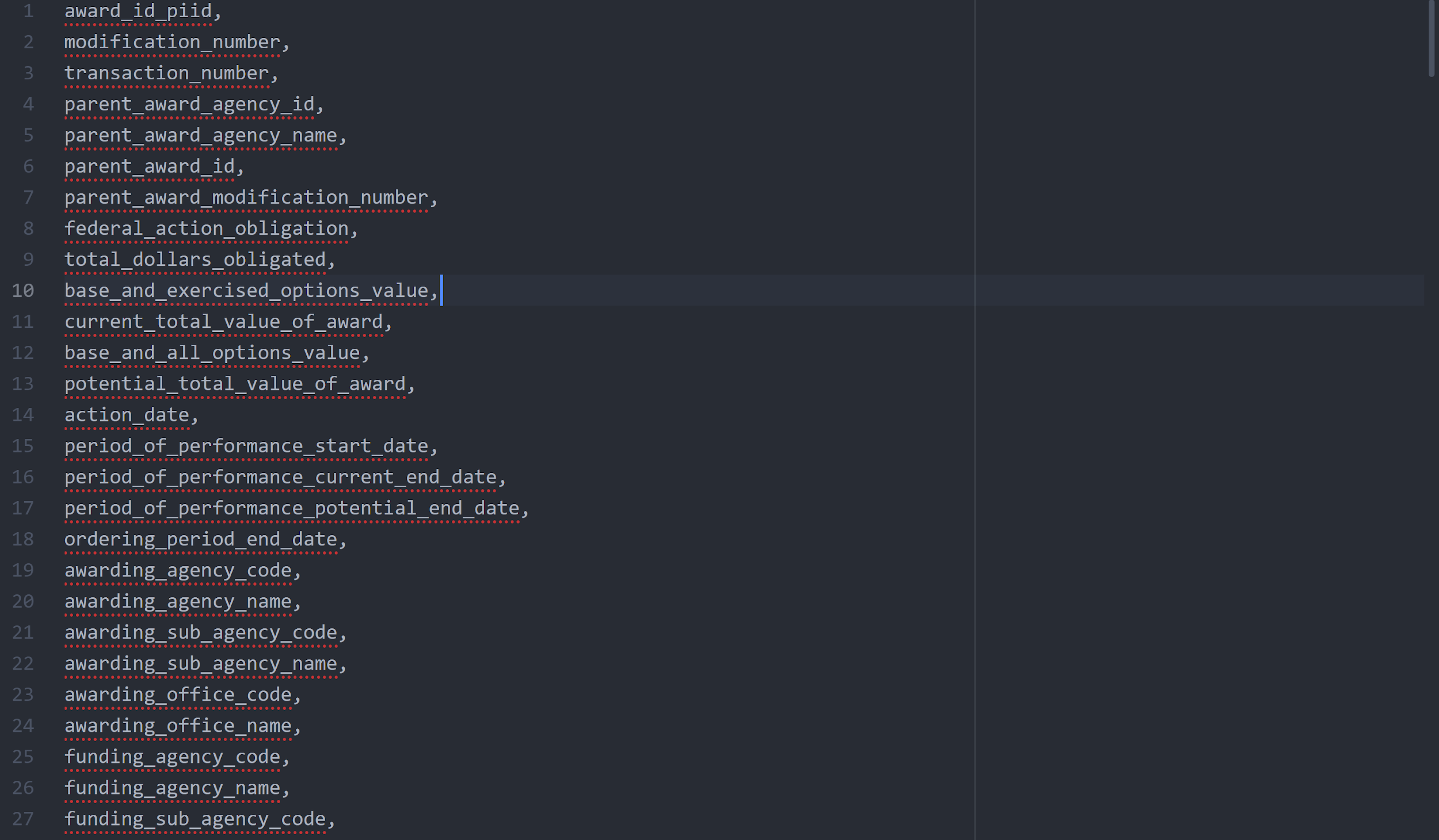
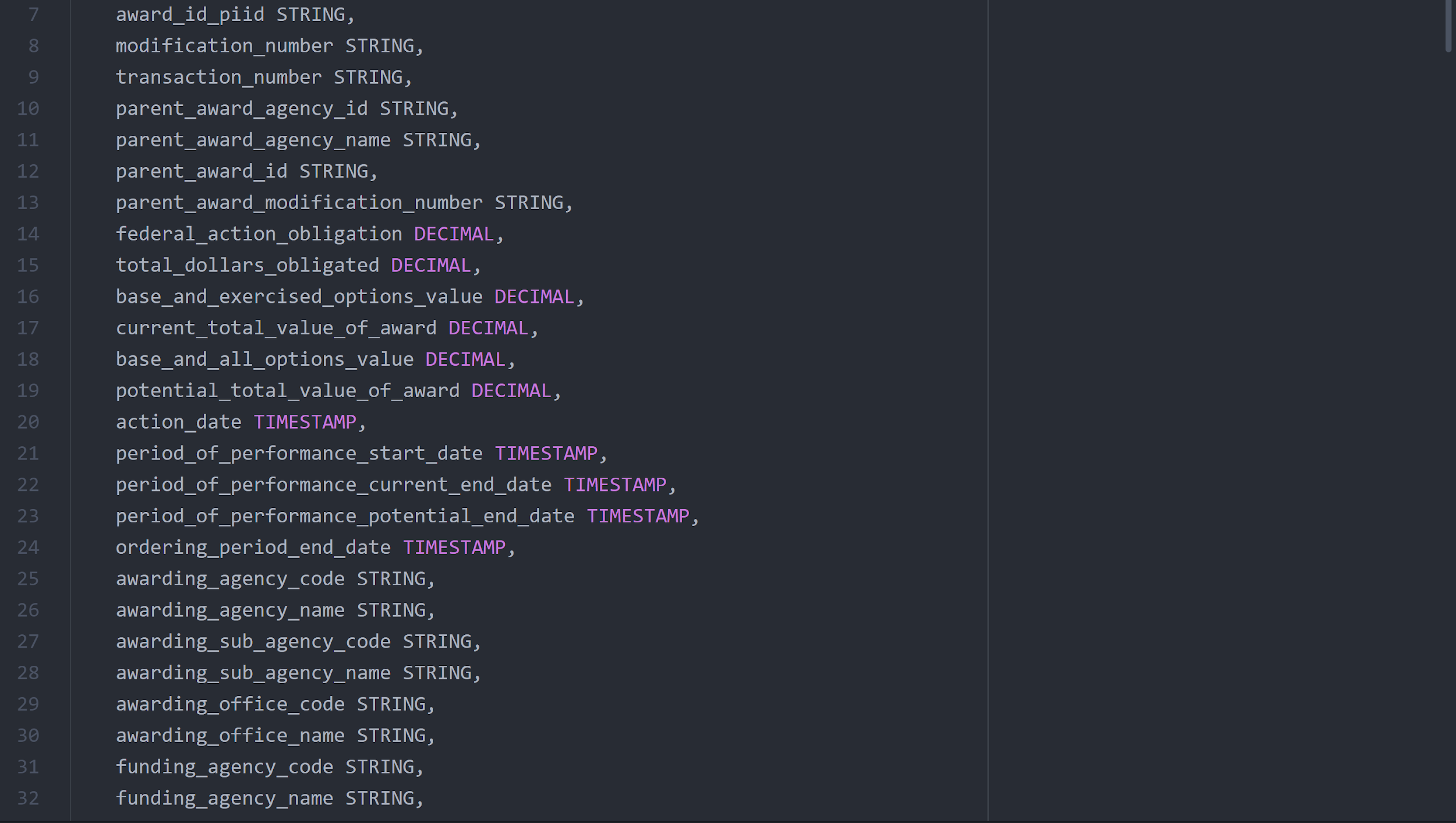
## 

1. Navigate to USAspending website:  
   https://www.usaspending.gov/#/download\_center/award\_data\_archive
2. Use website form to download desired scope of data, or submit a custom data request. Copy this download link.
   1. If submitting a custom data request, configure parameters for desired data on the download form and click download.
   2. Custom data requests take anywhere between 15 minutes to hours, and staying on the page is required to obtain the permanent download link once it is available.
3. Log into remote server using ssh:
   1. ssh user@xxx.xxx.xxx.xxx
4. Download file to remote server using download link:
   1. wget -o <link> .
5. Make a new directory and decompress file contents into it:
   1. mkdir data  
      unzip filename.zip data
6. Resulting text files may have varying columns depending on the name of the files: contracts\_prime/contracts\_subaward/assistance etc.
7. Create the directories on the Hadoop for storing these files, and make a directory for each type of data (i.e. contracts, assistance, etc):
   1. hdfs dfs -mkdir data  
      hdfs dfs -mkdir data/contracts  
      hdfs dfs -mkdir data/assist
8. Because we are going to be making external tables which load all files in a directory, and our total data is too big to fit on a single table, create a subfolder for each part of the data so that we can later make external tables for each part.
   1. hdfs dfs -mkdir data/contracts/part1  
      hdfs dfs -mkdir data/contracts/part2  
      ...  
      hdfs dfs -mkdir data/assist/part1  
      ...
9. Move files from local server to corresponding folders in HDFS:
   1. hdfs dfs -put data/contract1.csv data/contracts/part1  
      hdfs dfs -put data/contract2.csv data/contracts/part2  
      ...  
      hdfs dfs -put data/assistance1.csv data/assist/part1  
      ...

## Prepare Table Structure

## 

## 

1. Before we can prepare a HiveQL query to create a table, we need to identify all the column and types needed for the data. The first row of each file contains the headers. You can print the first row to see the headers:  
   head -1 data/contract1.csv  
     
   Or the second row to see the first row of records:  
   head -2 data/contract1.csv | tail -1 -  
   
2. To make the headers readable, separate each column (each of which are separated by a comma) to a new line. This can be accomplished using any method. Do this for the assistance data as well.
   1. Easiest method is to copy/paste the header text from the CLI to a text editor, and then separate each line manually or use a find/replace function.  
      
3. After separating each header column onto their own line, you can easily use it to compose the SQL statement for creating the table:  
   

## Load the data into tables in Beeline

## 

## 

1. Prepare SQL query for creating the external tables. Instead of the default loading function for delimited text files, use SERDE to correctly read rows where some columns contain commas within double quotes. Do the corresponding statements for each part of both contracts and assist, and replace 'x' with the part number:

CREATE EXTERNAL TABLE IF NOT EXISTS contracts\_x (

award\_id\_piid STRING,

modification\_number STRING,  
 transaction\_number STRING,  
 ...  
 sba\_certified\_8a\_joint\_venture STRING,

last\_modified TIMESTAMP

)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

"separatorChar" = ",",

"quoteChar" = "\""

)

STORED AS TEXTFILE

LOCATION 'path/data/contracts/partx/'

TBLPROPERTIES ('skip.header.line.count'='1');

1. Open a new terminal and login using ssh so that you have two terminals open. Use one of them for Beeline and the other for bash commands. Connect to Beeline using instructor-provided !connect command:
   1. beeline  
      !connect ...
2. Make new database:
   1. CREATE DATABASE spending
3. Switch to database:
   1. USE spending
4. Create external tables using SQL statements prepared before. Verify that a table exists for each part of contracts and assist:
   1. SHOW TABLES;

## Trim down the data

## 

## 

1. Data can now be accessed from any of the created tables using SQL statements, but the size of the data is too big and are spread across many tables. We should compose an SQL statement to create an internal table and only include columns that are of interest to us. Do the same for assists.
   1. CREATE TABLE IF NOT EXISTS contracts\_trimmed (  
       action\_date TIMESTAMP,  
       current\_total\_value\_of\_award DECIMAL,  
       recipient\_name STRING,  
       recipient\_address\_line\_1 STRING,  
       recipient\_city\_name STRING,  
       recipient\_state\_name STRING,  
       awarding\_agency\_name STRING,  
       award\_type STRING,  
       award\_description STRING  
      )  
      ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'  
      WITH SERDEPROPERTIES (  
       "separatorChar" = ",",  
       "quoteChar" = "\""  
      );
2. And then insert data from each of the external tables into the appropriate trimmed table (replace x with part number). Do the similar statement for assist:
   1. INSERT INTO TABLE contracts\_trimmed  
      SELECT  
       action\_date,  
       current\_total\_value\_of\_award,  
       recipient\_name,  
       recipient\_address\_line\_1,  
       recipient\_city\_name, recipient\_state\_name,  
       awarding\_agency\_name,  
       award\_type, award\_description  
      FROM contracts\_x;

## Query data using SQL

## 

## 

1. Now we can submit queries to these trimmed tables such as:
   1. --top 100 largest contracts awarded  
      SELECT  
       recipient\_name,  
       sum(current\_total\_value\_of\_award) Total\_Contract\_Value,   
       awarding\_agency\_name  
      FROM contracts\_trimmed  
      GROUP BY recipient\_name, awarding\_agency\_name  
      ORDER BY Total\_Contract\_Value DESC LIMIT 100;
   2. --total assistance awarded by departments  
      SELECT  
       awarding\_agency\_name,  
       sum(total\_funding\_amount) Dollars\_Awarded  
      FROM assist\_trimmed  
      GROUP BY awarding\_agency\_name  
      ORDER BY Dollars\_Awarded DESC;

## Export results to file

## 

## 

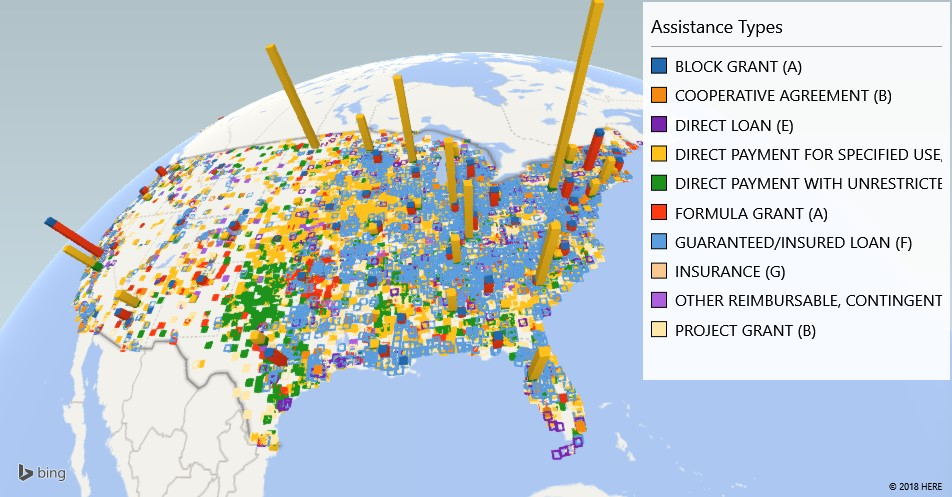
1. You can export theses trimmed tables as a tab-separated-value file. Create directories and export both tables as appropriate:
   1. hdfs dfs -mkdir path/results  
      hdfs dfs -mkdir path/results/assist\_trimmed  
      hdfs dfs -mkdir path/results/contract\_trimmed
   2. INSERT OVERWRITE DIRECTORY 'path/results/assist\_trimmed/'  
      ROW FORMAT DELIMITED  
      FIELDS TERMINATED BY '\t'  
      STORED AS TEXTFILE  
      SELECT \* FROM assist\_trimmed;
2. The resulting files in these folders will be differ in quantity depending on the size of the data and MapReduce settings. The files will be named in the pattern '000000\_0'. Download the entire folders of these outputs to their own folders on the local server:
   1. mkdir contracts\_trimmed  
      mkdir assist\_trimmed  
      hdfs dfs -get 'path/results/contracts\_trimmed' 'contracts\_trimmed'  
      hdfs dfs -get 'path/results/assist\_trimmed' 'assist\_trimmed'
3. Compress the folders into their own zip files:
   1. zip -r contract\_trimmed.zip contract\_trimmed  
      zip -r assist\_trimmed.zip assist\_trimmed
4. And then, either log out or open a new terminal to download the files to your local machine using scp:
   1. scp user@xxx.xxx.xxx.xxx:/home/user/contracts\_trimmed.zip localFolderPath/contracts\_trimmed.zip  
      scp user@xxx.xxx.xxx.xxx:/home/user/assist\_trimmed.zip localFolderPath/assist\_trimmed.zip

## Loading files on local machine into Excel for Visualization

## 

## 

1. Uncompress the files, and open a new workbook in Excel and load the files by:
   1. Data > From Text/CSV
   2. Depending on your computer, you might have to load each of the parts of the assist or contract trimmed data (000000\_0, 000001\_0, ...) one by one.
2. Rename the header columns once the data is loaded.
3. Select all the cells and create a 3D Map by:
   1. Insert > 3D Map > Add Selected Data to 3D Maps
4. Add a Layer and select "Stacked Bar" for the type.
5. Select the address, city, and state fields to the Location sections of the map. Excel will use these fields for displaying where on the map data appears.
6. Select the 'total dollars' (or similar) field to the Height section. The height of the 3D bars on the map will be determined by this field.
7. Select 'award\_description' or 'award\_type" (or similar) to the Category field. Different colored bars will represent what type of spending it is.
8. In the Filter section, add the blank category to filter it out from the map (to make it look less cluttered).
9. For Time, add the 'action\_date' field.
10. Repeat from Step 25 for assist data.  
    The resulting maps will look similar to this:



## References

## 

1. Data Source: https://www.usaspending.gov/#/download\_center/award\_data\_archive
2. Github: https://github.com/shozonu/fall2018-4560-us-fy2018-spending
3. Google any Bash or Hadoop command documentation!