CIS 4560 Term Project Tutorial

Authors: Alexander Castellanos, Armando Perez, Eric Peralta Erick Robles, Steve Hwang, Vincent Cheung

Instructor: Jongwook Woo

Date: 05/15/2023

Lab Tutorial

Load Pain Pills Data to Hadoop

Objectives

List what your objectives are. In this hands-on lab, you will learn how to:

- Connect to Hadoop Cluster remotely
- Load Pain Pills Data into Hadoop Clusters
- Create a staging table in Beeline
- Check and verify the data
- Load the clean data into PainPills table
- Check and verify the data again

Platform Spec

- Oracle Linux Server
- CPU Speed: 1995 MHz

- # of CPU cores: 8
- # of nodes: 3
- Total Memory Size: 58 GB

Step 1: Connect to Hadoop Cluster remotely

You need to remote to your Hadoop Clusters using the ***ssh*** command from the Git Bash terminal as follows:

```
$ ssh username@ipaddress
```

Step 2: Load Pain Pills Data into Hadoop Clusters

You can download the data files using the ***wget*** command from the terminal as follows:

```
$ wget https://github.com/vcheung621/cis4560/raw/main/arcos-southern-ca-
itemized.zip
```

Once you download the data file, please proceed with the commands below to create a temporary directory (arcos) and unzip the zip file into the directory.

```
$ mkdir arcos

$ mv arcos-southern-ca-itemized.zip arcos

$ cd arcos/

$ unzip arcos-southern-ca-itemized.zip
```

After you unzip all the CSV files, the below commands will create an HDFS directory (PainPillsFiles) and put all the CVS files into it.

```
$ hdfs dfs -mkdir PainPillsFiles

$ hdfs dfs -put *.csv PainPillsFiles

$ hdfs dfs -ls PainPillsFiles
```

Step 3: Create a staging table in Beeline

The following Hive statement creates an external staging table (painpills_stage). External tables preserve the data in the original file format while allowing Hive to perform queries against the data within the file.

NOTE: You have to replace the user name **<username>** to your username.

```
USE your_databasename;

--drop the table painpills_stage
DROP TABLE IF EXISTS painpills_stage;

--create the painpills staging table on comma-separated data
CREATE EXTERNAL TABLE IF NOT EXISTS painpills_stage(
REPORTER_DEA_NO STRING,
REPORTER_BUS_ACT STRING,
REPORTER_NAME STRING,
REPORTER_ADDL_CO_INFO STRING,
REPORTER_ADDRESS1 STRING,
REPORTER_ADDRESS2 STRING,
REPORTER_CITY STRING,
REPORTER_STATE STRING,
REPORTER_ZIP BIGINT,
REPORTER_COUNTY STRING,
BUYER_DEA_NO STRING,
BUYER_BUS_ACT STRING,
BUYER_NAME STRING,
BUYER_ADDL_CO_INFO STRING,
BUYER_ADDRESS1 STRING,
BUYER_ADDRESS2 STRING,
BUYER_CITY STRING,
BUYER_STATE STRING,
BUYER_ZIP BIGINT,
BUYER_COUNTY STRING,
TRANSACTION_CODE STRING,
DRUG_CODE BIGINT,
NDC_NO STRING,
DRUG_NAME STRING,
QUANTITY BIGINT,
UNIT STRING,
ACTION_INDICATOR STRING,
ORDER_FORM_NO STRING,
CORRECTION_NO STRING,
STRENGTH STRING,
TRANSACTION_DATE STRING,
CALC_BASE_WT_IN_GM FLOAT,
DOSAGE_UNIT BIGINT,
TRANSACTION_ID BIGINT,
Product_Name STRING,
Ingredient_Name STRING,
Measure STRING,
MME_Conversion_Factor FLOAT,
Combined_Labeler_Name STRING,
Revised_Company_Name STRING,
Reporter_family STRING,
dos_str STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = "\,",
   "quoteChar"     = "\"",
   "escapeChar"    = "\\"
)
STORED AS TEXTFILE LOCATION '/user/<username>/PainPillsFiles'
TBLPROPERTIES ('skip.header.line.count'='1');
```

Step 4: Check and verify the data

After constructing the table, we will examine and validate the data. The queries provided below will determine the total data count. It is essential to confirm the absence of any corrupt data. One method to validate data involves verifying if the column data aligns with the CSV files, selecting the DRUG_NAME column as an example because we know it contains only two unique values in the source CSV file. Additionally, we can inspect the zip code column (buyer_zip) for any letters and examine the manufacturer column (combined_labeler_name) for null values.

```
select count(*) from painpills_stage;
+-----------+
|   _c0     |
+-----------+
| 9571662   |
+-----------+
```

```
select count(*) from painpills_stage where drug_name not in
('HYDROCODONE','OXYCODONE');
+-------+
| _c0   |
+-------+
| 507   |
+-------+
```

```
select count(*) from painpills_stage where drug_name in
('HYDROCODONE','OXYCODONE');
+-----------+
|   _c0     |
+-----------+
| 9571155   |
+-----------+
SELECT COUNT(*) AS count_letters
FROM painpills_stage
WHERE LENGTH(regexp_extract(buyer_zip, '[a-zA-Z]', 0)) > 0;
+----------------+
| count_letters  |
+----------------+
| 507            |
+----------------+
```

```
SELECT COUNT(*) AS null_count
FROM painpills_stage
WHERE Combined_Labeler_Name = 'null';
+-------------+
| null_count  |
+-------------+
| 11430       |
+-------------+
```

Step 5: Load the clean data into PainPills table

Having identified 11,937 (507+11,430) instances of corrupt data through the previous query, we will proceed to clean this data. Additionally, the original dataset contains numerous columns that are not required for our analysis. We will selectively choose the relevant columns. The following statement will generate a new table called PAINPILLS, consisting of clean data and the essential columns.

```
--use beeline
DROP TABLE IF EXISTS painpills;

--create the painpills table on comma-separated data
CREATE TABLE painpills
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = "\,",
   "quoteChar"     = "\"",
   "escapeChar"    = "\\"
)
STORED AS TEXTFILE LOCATION '/user/vcheung4/PainPills'
AS
SELECT buyer_dea_no AS dea_no,
buyer_name as pharmacy,
buyer_addl_co_info as addl_co_info,
buyer_address1 as address1,
buyer_address2 as address2,
buyer_city as city,
buyer_state as state,
buyer_zip as zip,
buyer_county as county,
drug_name,
quantity,
TO_DATE(from_unixtime(unix_timestamp(transaction_date,'MMddyyyy'),'yyyy-MM-
dd')) AS transaction_date,
calc_base_wt_in_gm,
dosage_unit as number_of_pills,
transaction_id,
product_name,
ingredient_name,
combined_labeler_name as manufacturer,
revised_company_name as distributor,
dos_str
FROM painpills_stage where drug_name in ('OXYCODONE','HYDROCODONE') AND
Combined_Labeler_Name <> 'null';
```

Step 6: Check and verify the data again

Now check and verify the data again to see any dirty data.

```
select count(*) from painpills;
+-----------+
|   _c0    |
+-----------+
| 9559725  |
+-----------+
```

```
select count(*) from painpills where drug_name not in
('HYDROCODONE','OXYCODONE');
+-------+
| _c0  |
+-------+
| 0    |
+-------+
```

```
select count(*) from painpills where drug_name in
('HYDROCODONE','OXYCODONE');
+-----------+
|   _c0    |
+-----------+
| 9559725  |
+-----------+
```

```
SELECT COUNT(*) AS count_letters
FROM painpills
WHERE LENGTH(regexp_extract(zip, '[a-zA-Z]', 0)) > 0;
+-----------------+
| count_letters  |
+-----------------+
| 0               |
+-----------------+
```

```
SELECT COUNT(*) AS null_count
FROM painpills
WHERE manufacturer = 'null';
+--------------+
| null_count  |
+--------------+
| 0            |
+--------------+
```

Here are the expected results. The DRUG_NAME column only contains two distinct values. The zip column contains no letter characters. The manufacturer column contains no null values. The new total record count is 9559725 (9571662 – 11937).

References

1. Data Source: https://www.washingtonpost.com/graphics/2019/investigations/dea-pain-pill-database/
2. Github: https://github.com/vcheung621/cis4560
3. References: https://www.kaggle.com/datasets/paultimothymooney/pain-pills-in-the-usa