STSCI 5065 HW 3 Solutions*

(Assigned: 3/22/2019; Due: 3/29/2019 at 11:59PM)

What and how to turn in:

Submit electronically to the course website: a PDF file named STSCI5065-HW3-LastName-FirstName, containing all the steps you performed, including the code, and your answers to the questions, listed in the order of the questions.

In this homework, there are two problem sets. You will practice how to use Hive to establish a database and process the data in HDFS according to the requirements specified below in the problem sets. Unless otherwise directed, list all your code in a separate paragraph with blue font, for example:

```
select flight_delay
from delays
where carrier = “NW”;
```

Problem Set 1

You are given a set of two text files of New York Stock Exchange (NYSE) data, NYSE_daily_prices.csv and NYSE_dividends.csv. You are required to create a Hive database containing two tables based on the text files, and then do some analysis of the data using Hive queries. (70 points total)

A. (6 points) Download the data file, stock.zip, from the course website and unzip it in a local OS directory. Open the files in a text editor to see how the data fields are separated. Create an HDFS directory, **HW3**, right below the CentOS root. In HW3, create a directory called **stockdata**. Use the File Browser in Hue to upload the two data files into stockdata. Display a screenshot (in Ambari) showing all these files in the /HW3/stockdata directory (your screenshot should show this directory).

The data fields are separated by commas in both of the files.

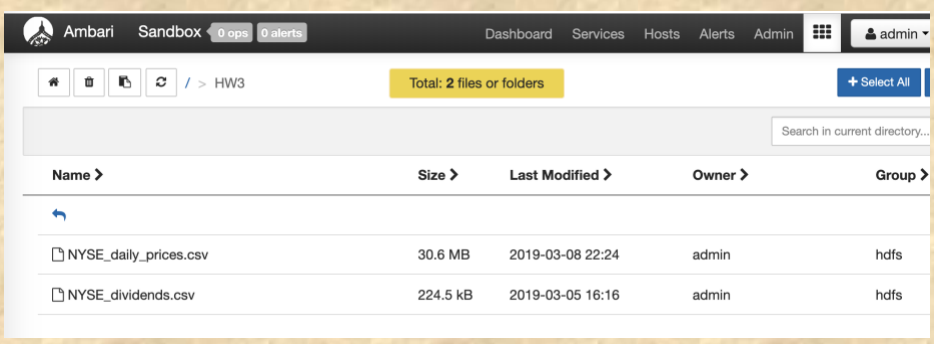
Create an HDFS directory, HW3, right below the CentOS root:

```
hadoop fs –mkdir /HW3
```

In HW3, create a directory called stockdata:

```
hadoop fs –mkdir /HW3/stockdata
```

The screenshot:



B. (13 points) Create a Hive database called **stocksdb** located in HW3. In the stocksdb database, use Hive command line interface to create two tables **stock_prices** and **stock_dividends**. The stock_prices table contains columns in the following order: exchng, symbol, ymd, price_open, price_high, price_low, price_close, price_volume, and price_adj_close. The data types of exchng, symbol and ymd are string (same in the stock_dividends table) and other columns are of double data type. The stock_dividends contains the columns in the following order: exchng, symbol, ymd, and dividend. The data type of dividend is double. Describe these two tables in Hive command line interface, and attach one single screenshot, showing the commands and the results.

Create a Hive database called stocksdb located in HW3: (2 points)

```
create database stocksdb
location '/HW3';
```

Create two tables stock_prices and stock_dividends:

```
create table if not exists stock_prices (          (7 points)
  exchng string,
  symbol string,
  ymd string,
  price_open double,
  price_low double,
  price_high double,
  price_close double,
  price_volume double,
  price_adj_close double
  )
  row format delimited
  fields terminated by ','
  lines terminated by '\n'
  stored as textfile;

create table if not exists stock_dividends (       (5 points)
  exchng string,
  symbol string,
  ymd string,
  dividend double
  )
  row format delimited
  fields terminated by ','
  lines terminated by '\n'
  stored as textfile;
```

(6 points)

```
hive> use stocksdb;
OK
Time taken: 0.227 seconds
hive> describe stock_prices;
OK
exchng                  string
symbol                  string
ymd                     string
price_open              double
price_low               double
price_high              double
price_close             double
price_volume            double
price_adj_close         double
Time taken: 0.525 seconds, Fetched: 9 row(s)
hive> describe stock_dividends;
OK
exchng                  string
symbol                  string
ymd                     string
dividend                double
Time taken: 0.508 seconds, Fetched: 4 row(s)
hive>
```

C. (5 points) Load the data files into the respective tables created in step B.

load data inpath '/HW3/stockdata/NYSE_daily_prices.csv'
overwrite into table stock_prices;

load data inpath '/HW3/stockdata/NYSE_dividends.csv'
overwrite into table stock_dividends;

D. (6 points) Query the stock_prices table to find out the number of entries of each stock symbol. You are only required to show two screenshots of the first 10 rows and the last 10 rows of your result, including the line starting with "Time taken" in your output.

The query for the first 10 rows:

```
select symbol, count(*)
   from stock_prices
   group by symbol
   limit 10;
```

```
OK
BA      12109
BAC     5977
BAF     1830
BAK     2785
BAM     6495
BAP     3539
BAS     1047
BAX     7115
BBD     1891
BBF     2142
Time taken: 11.482 seconds, Fetched: 10 row(s)
```

To get the last 10 rows, you can run the above query by taking out the limit clause and then crop out the portion you want to report:

```
select symbol, count(*)
   from stock_prices
   group by symbol;
```

```
BXG     5562
BXP     3175
BXS     6132
BYD     4103
BYI     6354
BYM     1830
BZ      658
BZA     1953
BZH     4018
BZMD    18
Time taken: 6.333 seconds, Fetched: 168 row(s)
```

E. (6 points) Calculate the average stock opening price of each stock symbol rounded to 4 decimal points (same below). Use an alias, ap, for the average column. Attach a screenshot of the last ten rows of output.

The query:

```
select symbol, round(avg(price_open), 4) ap
  from stock_prices
  group by symbol;
```

The screenshot:

```
BXG     5.3577
BXP     55.4753
BXS     23.523
BYD     17.449
BYI     11.3204
BYM     13.8966
BZ      4.6387
BZA     14.7472
BZH     37.9346
BZMD    24.5267
Time taken: 6.99 seconds, Fetched: 168 row(s)
```

F. (3 points) Create a Hive view, **average_price_v**, based on step E.

```
create view average_price_v as
    select symbol, round(avg(price_open), 4) ap
      from stock_prices
      group by symbol;
```

G. (12 points) Find out the stock symbols that have the highest and lowest average opening prices, respectively. You are required to use the view created in step F and to complete this query in Ambari using the Hive View. Attach the screenshots of the query results.

The query to find the highest average:

```
select a.symbol, round(a.ap, 4) highest_average_open_price from
 (select max(ap) ma from max_avg_v) b, max_avg_v a
 where a.ap= b.ma;
```

The query result:

	a.symbol	highest_average_open_price
0	BLK	97.0208

The query to find the highest average:

```
select a.symbol, round(a.ap, 4) lowest_average_open_price from
 (select min(ap) mi from max_avg_v) b, max_avg_v a
 where a.ap= b.mi;
```

The query result:

	a.symbol	lowest_average_open_price
0	BZ	4.6387

H. (12 points) Find out the stock symbol that has the highest dividends. What are the date and the opening price of that stock when that happened? Complete this query also in Ambari's Hive View. Attach the screenshots of the query results.

The query:

```
select a.symbol, a.ymd, a.price_open, c.md highest_dividend
  from  stock_prices a, stock_dividends b,
  (select max(dividend) md from stock_dividends) c
  where b.dividend = c.md and a.ymd=b.ymd and a.symbol=b.symbol;
```

The screenshot:

	a.symbol	a.ymd	a.price_open	highest_dividend
0	BCE	2000-05-09	26.62	87.057999

So the stock symbol that has the highest, 87.057999, is BCE. That happened on May 5, 2000. The opening price of the stock of that day was 26.62.

Problem Set 2

You are given a text file (flight12.csv) of flight delays and are required to create a Hive table using regular expressions, and then do some queries on the table. (30 points total)

A. (5 points) Download the data file and use the Files View in Ambari to load the file into HDFS in the HW3 directory. Create a Hive database called **flightsdb** located in HW3, and then in flightsdb create a Hive table, **flight_delays_hw3**. This table has the following columns: ymd, flight_num, carrier_delay, weather_delay, nas_delay, security_delay, and late_aircraft_delay. The first two columns are of string data type and the other columns are of double data type.

Create a Hive database called flightsdb located in HW3:

```
create database flightsdb
location '/HW3';
```

Create the flight_delays_hw3 table in flightsdb:

```
create table flight_delays_hw3 (
```

```
ymd string,
flight_num string,
carrier_delay double,
weather_delay double,
nas_delay double,
security_delay double,
late_aircraft_delay double);
```

B. (3 points) Create a temporary one-column Hive table, **temp_flight**, to read in the data from the text file so that a line of text becomes a row in the temp_flight table, and then load the text data file into temp_flight.

Create the temp_flight table:

```
create table temp_flight(aline string);
```

Load the text data file into temp_flight:

```
load data inpath '/HW3/flight12.csv'
overwrite into table temp_flight;
```

C. (15 points) Insert data into flight_delays_hw3 by extracting it from temp_flight with regular expression (regexp_extract) calls. The data fields are separated by commas. You are required to extract fields 2, 5,19, 20, 21, 22, and 23 to populate your table.

```
INSERT OVERWRITE TABLE flight_delays_hw3
SELECT
 regexp_extract(aline, '^(?:([^,]*)\,?){2}', 1) ymd,
 regexp_extract(aline, '^(?:([^,]*)\,?){5}', 1) flight_num,
 regexp_extract(aline, '^(?:([^,]*)\,?){19}', 1) carrier_delay,
 regexp_extract(aline, '^(?:([^,]*)\,?){20}', 1) weather_delay,
 regexp_extract(aline, '^(?:([^,]*)\,?){21}', 1) nas_delay,
 regexp_extract(aline, '^(?:([^,]*)\,?){22}', 1) security_delay,
 regexp_extract(aline, '^(?:([^,]*)\,?){23}', 1) late_aircraft_delay
FROM temp_flight;
```

D. (2 points) Query the flight_delays_hw3 table with Hive command line and by only display the first 10 rows. Attach a screenshot of Hive commandline and its result.

```
hive> select * from flight_delays_hw3 limit 10;
OK
2013-12-01      "2900"  NULL    NULL    NULL    NULL    NULL
2013-12-02      "2900"  0.0     0.0     0.0     0.0     36.0
2013-12-03      "2900"  NULL    NULL    NULL    NULL    NULL
2013-12-04      "2900"  NULL    NULL    NULL    NULL    NULL
2013-12-05      "2900"  NULL    NULL    NULL    NULL    NULL
2013-12-06      "2900"  10.0    0.0     0.0     0.0     11.0
2013-12-07      "2900"  NULL    NULL    NULL    NULL    NULL
2013-12-08      "2900"  NULL    NULL    NULL    NULL    NULL
2013-12-09      "2900"  NULL    NULL    NULL    NULL    NULL
2013-12-10      "2900"  0.0     0.0     4.0     0.0     83.0
Time taken: 0.19 seconds, Fetched: 10 row(s)
```

E. (5 points) Query the table and find out the longest delay of each category of delay. Use a meaningful alias for each column. Do this in Ambari Hive View and attach a screenshot of the results.

The query:

```
select max(carrier_delay) max_carrier_delay,
    max(weather_delay) max_weather_delay,
    max(nas_delay) max_nas_delay,
    max(security_delay) max_security_delay,
    max(late_aircraft_delay) max_late_aircraft_delay
    from flight_delays_hw3;
```

The result:

max_carrier_delay	max_weather_delay	max_nas_delay	max_security_delay	max_late_aircraft_delay
1975.0	1451.0	1174.0	175.0	892.0

If you do it in Hive CLI, you will get the following (**not required** in this HW):

```
hive> select max(carrier_delay) max_carrier_delay,
max(weather_delay) max_weather_delay,  max(nas_delay
) max_nas_delay, max(security_delay) max_security_de
lay,  max(late_aircraft_delay) max_late_aircraft_del
ay from flight_delays_hw3;
Query ID = root_20170412141417_d111d2c2-485a-4981-b9
78-b718015cedac
Total jobs = 1
Launching Job 1 out of 1


Status: Running (Executing on YARN cluster with App
id application_1491828968370_0005)

Map 1: 0/3      Reducer 2: 0/1
Map 1: 0(+2)/3  Reducer 2: 0/1
Map 1: 0(+3)/3  Reducer 2: 0/1
Map 1: 1(+2)/3  Reducer 2: 0/1
Map 1: 2(+1)/3  Reducer 2: 0(+1)/1
Map 1: 3/3      Reducer 2: 0(+1)/1
Map 1: 3/3      Reducer 2: 1/1
OK
1975.0  1451.0  1174.0  175.0   892.0
Time taken: 7.192 seconds, Fetched: 1 row(s)
```