CIS5200 Term Project Tutorial

Authors: Suneel Srikanta, Avinash Vallabhaneni, Shahadat Molla

Instructor: **Jongwook Woo**

Date: 05/08/2024

Lab Tutorial

Suneel Srikanta (ssrikan3@calstatela.edu)

Avinash Vallabhaneni (avallab2@calstatela.edu)

Shahadat Molla (smolla@calstatela.edu)

05/08/2024

Amazon Product Reviews (Spam and Non-Spam)

Step 1: All query snippets which have been executed

Use the commands listed below to obtain the cluster details:

```
$ -bash-4.2$ hdfs version

$ -bash-4.2$ lscpu

$ -bash-4.2$ nproc

$ -bash-4.2$ hdfs dfs -df -h

$ -bash-4.2$ beeline

$ -bash-4.2$ yarn node -list -all
```

1. To copy data from a local PC to a remote Linux system, we must use the **scp** command

```
$ scp C:/5200_Term_Project/Toys_and_Games/Toys_and_Games.csv
ssrikan3@129.153.214.22:/tmp
```

2. Open a new bash CLI to **ssh** into remote Linux system using your credentials and check whether file has been uploaded under /tmp directory.

```
$ scp C:/5200_Term_Project/Toys_and_Games/Toys_and_Games.csv
ssrikan3@129.153.214.22:/home/ssrikan3/tmp.
```

3. Created a new directory in HDFS and we named as “Group 4_Term_Project”, then use “**ls**” command to verify that the whether directory is created or not

```
$ -bash-4.2$ hdfs dfs -mkdir Group 4_Term_Project
$ -bash-4.2$ hdfs dfs -ls
```

4. Use the following command to upload all the 7 datasets from Linux system into newly created directory (**Group4_Term_Project**) in the HDFS.

```
$ -bash-4.2$ hdfs dfs -put Clothing_Shoes_and_Jewelry.csv Group4_Term_Project/
$ -bash-4.2$ hdfs dfs -ls Group4_Term_Project/
```

5. To check the header of the CSV file, use the below command

```
$ head -n 1 Clothing_Shoes_and_Jewelry.csv
```

6. We need to grant access permission for the teammates, you can modify the permissions using the chmod command. Here's is the command how can change the permission

```
$ bash-4.2$ hdfs dfs -chmod  -R og+rwx /user/ssrikan3/Group4_Term_Project
```

7. Use the SSH command to establish a connection with your Oracle Big Data Server instance, enter "**beeline**" to start the HiveQL environment.

```
$  ssh ssrikan3@129.153.214.22
$ -bash-4.2$ beeline
```

8. Now you need to create your database using **Create** query and use your database using Use **query**, if you have an existing database can remove using **Drop** query.

```
CREATE DATABASE if not exists ssrikan3;
show databases;
use ssrikan3;
```

9. Create an external table named product_review in HiveQL. The header of the csv can be found in above step.

```
CREATE EXTERNAL TABLE IF NOT EXISTS product_review(`id` string, `reviewerID` string, `asin` string,  `reviewerName` string, `stateCode` string, `reviewText` string, `overall` string, `summary` string, `unixReviewTime` string, `reviewDate` string, `category` string, `class` string)
 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
```

9. Check whether table has been created successfully and verify data being inserted to the columns, to validate use below commands.

```
0: jdbc:hive2://bigdaiun0.sub03291929060.trai> show tables;
0: jdbc:hive2://bigdaiun0.sub03291929060.trai> select * from product_review LIMIT 10;
```

10. Created another table as "product_review_1" with the desired column data types.

```
CREATE TABLE product_review_1(`id` string, `reviewerID` string, `asin` string, `reviewerName` string, `stateCode` string, `reviewText` string, `overall` string, `summary` string, `unixReviewTime` bigint, `reviewDate` date, `category` string, `class` int);
```

11. Now, load the data from product_review into the new table **product_review_1.** Perform the following actions: Cast the reviewDate column to date format; Cast the unixReviewTime column to bigint and class column to int data type.

```
INSERT INTO product_review_1
SELECT `id`, `reviewerID`, `asin`, `reviewerName`, `stateCode`, `reviewText`, `overall`, `summary`,
CAST(`unixReviewTime` AS bigint) AS `unixReviewTime`,
CAST(`reviewDate` AS date) AS `reviewDate`,
`category`,
CAST(`class` AS int) AS `int`
FROM product_review;
```

12. Now, verify the new table whether datatypes are changed.

```
0: jdbc:hive2://bigdaiun0.sub03291929060.trai> describe product_review_1;
```

13. Finally, now you can drop the initial external table “product_review” and rename newly create table from “product_review_1” to “product_review”. Later, make sure only one table exists.

```
DROP table product_review;
ALTER TABLE product_review_1 RENAME TO product_review;
show tables;
```

14. Query to check total number of reviews received each year and along with that percentage change of reviews compared to previous year.

```
SELECT cur.year,
    cur.total_reviews_received,
    Round(( cur.total_reviews_received - prev.total_reviews_received ) /
            NULLIF(prev.total_reviews_received, 0) * 100, 2) AS
    percentage_change
FROM   (SELECT Year(reviewdate) AS Year,
           Count(*)       AS total_reviews_received
     FROM   product_review
     GROUP  BY Year(reviewdate)
     ORDER  BY year) cur
    LEFT JOIN (SELECT Year(reviewdate) AS Year,
                  Count(*)       AS total_reviews_received
              FROM   product_review
              GROUP  BY Year(reviewdate)
              ORDER  BY year) prev
          ON cur.year = prev.year + 1
WHERE  cur.year IS NOT NULL;
```

15. Query to check top three states which has highest number of spam_reviews received during the period from 2012 to 2020.

```
SELECT statecode, YEAR(reviewdate) AS review_year, COUNT(*) AS spam_reviews
FROM product_review
WHERE class = 1
  AND YEAR(reviewdate) IN (2012,2013,2014,2015,2016,2017,2018,2019,2020)
GROUP BY statecode, YEAR(reviewdate)
ORDER BY review_year, spam_reviews DESC;
WITH ranked_states AS (
  SELECT
    statecode,
    YEAR(reviewdate) AS review_year,
    COUNT(*) AS spam_reviews,
    ROW_NUMBER() OVER (PARTITION BY YEAR(reviewdate) ORDER BY COUNT(*) DESC) AS rank
  FROM product_review
  WHERE class = 1
    AND YEAR(reviewdate) IN (2012,2013,2014,2015,2016,2017,2018,2019,2020)
  GROUP BY statecode, YEAR(reviewdate)
)
SELECT statecode, review_year, spam_reviews
FROM ranked_states
WHERE rank <= 3
ORDER BY review_year, rank;
```

16. Query to check top three states which has highest number of non_spam_reviews received during the period from 2012 to 2020.

```
SELECT statecode, YEAR(reviewdate) AS review_year, COUNT(*) AS non_spam_reviews
FROM product_review
WHERE class = 0
  AND YEAR(reviewdate) IN (2012,2013,2014,2015,2016,2017,2018,2019,2020)
GROUP BY statecode, YEAR(reviewdate)
ORDER BY review_year, non_spam_reviews DESC;
WITH ranked_states AS (
  SELECT
    statecode,
    YEAR(reviewdate) AS review_year,
    COUNT(*) AS non_spam_reviews,
    ROW_NUMBER() OVER (PARTITION BY YEAR(reviewdate) ORDER BY COUNT(*) DESC) AS rank
  FROM product_review
  WHERE class = 0
    AND YEAR(reviewdate) IN (2012,2013,2014,2015,2016,2017,2018,2019,2020)
  GROUP BY statecode, YEAR(reviewdate)
)
SELECT statecode, review_year, non_spam_reviews
FROM ranked_states
WHERE rank <= 3
ORDER BY review_year, rank;
```

17. Query to check total number of reviews received by each state

```
SELECT statecode, COUNT(*) AS Total_Reviews
FROM product_review
GROUP BY statecode
HAVING COUNT(*) > 200
ORDER BY Total_Reviews DESC;
```

18. Query to check top five states which has received most spam reviews.

```
SELECT statecode,
     Count(*) AS spam_reviews
FROM   product_review
WHERE  class = 1
GROUP  BY statecode
ORDER  BY spam_reviews DESC
LIMIT  5;
```

19. Query to check number one state which has received most spam reviews.

The below response shows that amazon has received more from spam reviews peoples from “FLORIDA (FL)” have made less spam reviews.

```
SELECT statecode,
    Count(*) AS spam_reviews
FROM   product_review
WHERE  class = 1
GROUP  BY statecode
ORDER  BY spam_reviews DESC
LIMIT  1;
```

20. Query to check which state has received least spam reviews.

The below response shows that peoples from “MISSOURI (MO)” have made less spam reviews.

```
SELECT statecode,
    Count(*) AS spam_reviews
FROM   product_review
WHERE  class = 1
GROUP  BY statecode
ORDER  BY spam_reviews ASC
LIMIT  1;
```

21. Query to check top five states which has received most non-spam reviews.

```
SELECT statecode, COUNT(*) AS non_spam_reviews FROM product_review WHERE class = 0
GROUP BY statecode ORDER BY non_spam_reviews DESC LIMIT 5;
```

22. Query to check which state has received least non-spam reviews.

The below response shows that peoples from “AMERICAN SAMOA (AS)” has made less non-spam reviews.

```
SELECT statecode, COUNT(*) AS non_spam_reviews FROM product_review WHERE class = 0
GROUP BY statecode ORDER BY non_spam_reviews DESC LIMIT 5;
```

23, Query to check which state has received least non-spam reviews.

The below response shows that peoples from “AMERICAN SAMOA (AS)” has made less non-spam reviews.

```
SELECT statecode, COUNT(*) AS non_spam_reviews FROM product_review WHERE class = 0
GROUP BY statecode ORDER BY non_spam_reviews DESC LIMIT 5;
```

24. Now, the following query gives the result top five states which received spam reviews for the period from 2012 to 2020.

```
SELECT statecode, YEAR(reviewdate) AS review_year, COUNT(*) AS spam_reviews
FROM product_review
WHERE class = 1
AND YEAR(reviewdate) IN (2012,2013,2014,2015,2016,2017,2018,2019,2020)
GROUP BY statecode, YEAR(reviewdate)
ORDER BY spam_reviews DESC
LIMIT 5;
```

25. Now, the following query gives the result top five states which received non-spam reviews for the period from 2012 to 2020.

```
SELECT statecode, YEAR(reviewdate) AS review_year, COUNT(*) AS non_spam_reviews
FROM product_review
WHERE class = 0
AND YEAR(reviewdate) IN (2012,2013,2014,2015,2016,2017,2018,2019,2020)
GROUP BY statecode, YEAR(reviewdate)
ORDER BY non_spam_reviews DESC
LIMIT 10;
```

26. Query to check Top 5 states which received highest reviews.

```
SELECT statecode,
    total_reviews
FROM   (SELECT statecode,
           Count(*) AS Total_Reviews
     FROM   product_review
     GROUP  BY statecode
     ORDER  BY total_reviews DESC
     LIMIT  5) AS top_states
WHERE  total_reviews > 100000;
```

27. Total number of reviews received overall including all states.

```
SELECT COUNT(*) AS total_records from product_review;
```

28. Now, the following query gives the result top 3 categories which received spam reviews for the period from 2013, 2014, 2016 & 2019.

```
SELECT category, COUNT(*) AS spam_reviews
FROM product_review
WHERE class = 1
AND YEAR(reviewdate) IN (2013,2014,2016,2019)
GROUP BY category
ORDER BY spam_reviews DESC
LIMIT 3;
```

29. Now, the following query gives the result top 3 categories which received non-spam reviews for the period from 2015,2019,2020 & 2021.

```
SELECT category, COUNT(*) AS non_spam_reviews
FROM product_review
WHERE class = 0
AND YEAR(reviewdate) IN (2015,2019,2020,2021)
GROUP BY category
ORDER BY non_spam_reviews DESC
LIMIT 3;
```

30. Now, the following query gives the result which one of the month from each year has received highest spam reviews and along with that for category.

```
SELECT
  review_year AS year,
  CASE review_month
    WHEN 1 THEN 'January'
    WHEN 2 THEN 'February'
    WHEN 3 THEN 'March'
    WHEN 4 THEN 'April'
    WHEN 5 THEN 'May'
    WHEN 6 THEN 'June'
    WHEN 7 THEN 'July'
    WHEN 8 THEN 'August'
    WHEN 9 THEN 'September'
    WHEN 10 THEN 'October'
    WHEN 11 THEN 'November'
    WHEN 12 THEN 'December'
  END AS month,
  category,
  reviews_count AS spam_reviews_received
FROM (
  SELECT
    YEAR(reviewdate) AS review_year,
    MONTH(reviewdate) AS review_month,
    category,
    COUNT(*) AS reviews_count,
    ROW_NUMBER() OVER (PARTITION BY YEAR(reviewdate) ORDER BY COUNT(*) DESC) AS rank
  FROM
    product_review
  WHERE
    class = 1
    AND YEAR(reviewdate) BETWEEN 2012 AND 2021
  GROUP BY
    YEAR(reviewdate),
    MONTH(reviewdate),
    category
) AS subquery
WHERE
  rank = 1;
```

31. Now, the following query gives the result which one of the month from each year has received highest spam reviews and along with that for category.

```
SELECT
  review_year AS year,
  CASE review_month
    WHEN 1 THEN 'January'
    WHEN 2 THEN 'February'
    WHEN 3 THEN 'March'
    WHEN 4 THEN 'April'
    WHEN 5 THEN 'May'
    WHEN 6 THEN 'June'
    WHEN 7 THEN 'July'
    WHEN 8 THEN 'August'
    WHEN 9 THEN 'September'
    WHEN 10 THEN 'October'
    WHEN 11 THEN 'November'
    WHEN 12 THEN 'December'
  END AS month,
  category,
  reviews_count AS non_spam_reviews_received
FROM (
  SELECT
    YEAR(reviewdate) AS review_year,
    MONTH(reviewdate) AS review_month,
    category,
    COUNT(*) AS reviews_count,
    ROW_NUMBER() OVER (PARTITION BY YEAR(reviewdate) ORDER BY COUNT(*) DESC) AS rank
  FROM
    product_review
  WHERE
    class = 0
    AND YEAR(reviewdate) BETWEEN 2012 AND 2021
  GROUP BY
    YEAR(reviewdate),
    MONTH(reviewdate),
    category
) AS subquery
WHERE
  rank = 1;
```