# [https://avatars2.githubusercontent.com/u/4156894?v=3&s=100](http://www.calstatela.edu/centers/hipic)

**CIS5200 Term Project Tutorial**

#### Authors: [Suneel Srikanta](https://www.linkedin.com/in/suneel-s-2185a0110/), Avinash Vallabhaneni, [Shahadat Molla](https://www.linkedin.com/in/shahadat-molla-7519896a?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=ios_app)

#### Instructor: [Jongwook Woo](https://www.linkedin.com/in/jongwook-woo-7081a85)

#### Date: 05/08/2024

**Lab Tutorial**

Suneel Srikanta ([ssrikan3@calstatela.edu](mailto:dvaishn2@calstatela.edu))

Avinash Vallabhaneni ([avallab2@calstatela.edu](mailto:mneethi@calstatela.edu))

Shahadat Molla ([smolla@calstatela.edu](mailto:akhaire3@calstatela.edu))

05/08/2024

**Amazon Product Reviews ( Spam and Non-Spam )**

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'**

**WITH SERDEPROPERTIES (**

**'separatorChar' = ',',**

**'quoteChar' = '"',**

**'escapeChar' = '\\' )**

**STORED AS TEXTFILE LOCATION '/user/ssrikan3/Group4\_Term\_Project/'**

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

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;**