Assignment 02

Implementation of a system for sales data analytics using Hadoop Eco-System

—

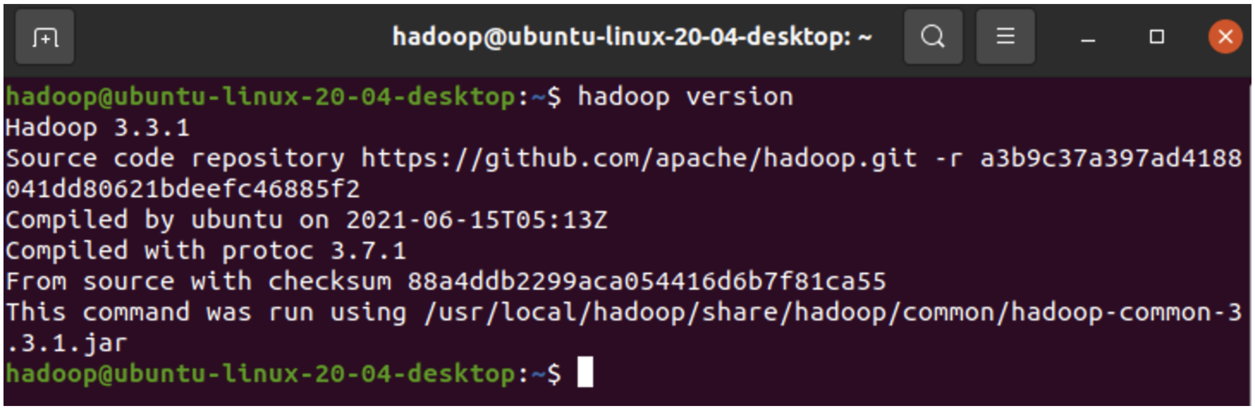
Group 141 Members

Kumar Anurag - 2021FC04016
Das Shubhankar Jagannath Arpita - 2021FC04154
Amit Kumar - 2021FC04433
Siddhant Shivam - 2021FC04090

SETTING UP THE HADOOP & HDFS

1.1 Hadoop Version

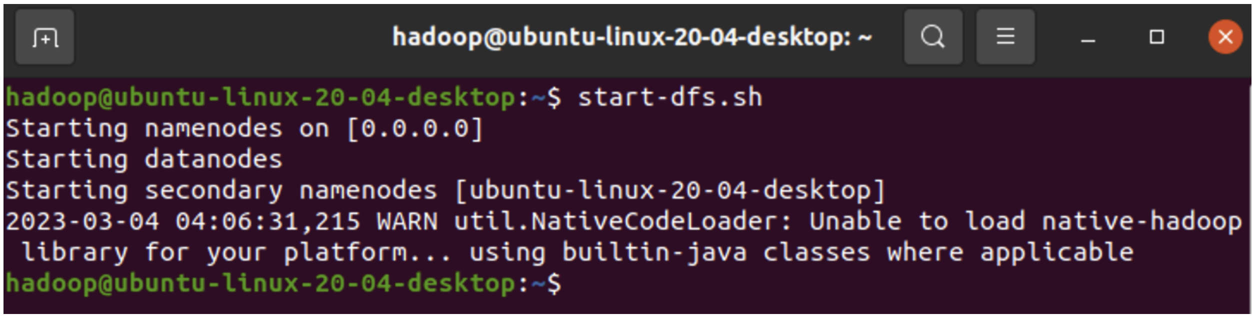
Printing the current Hadoop version.



```
hadoop@ubuntu-linux-20-04-desktop:~$ hadoop version
Hadoop 3.3.1
Source code repository https://github.com/apache/hadoop.git -r a3b9c37a397ad4188
041dd80621bdeefc46885f2
Compiled by ubuntu on 2021-06-15T05:13Z
Compiled with protoc 3.7.1
From source with checksum 88a4ddb2299aca054416d6b7f81ca55
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3
.3.1.jar
hadoop@ubuntu-linux-20-04-desktop:~$ 
```

1.2 Starting the DFS services

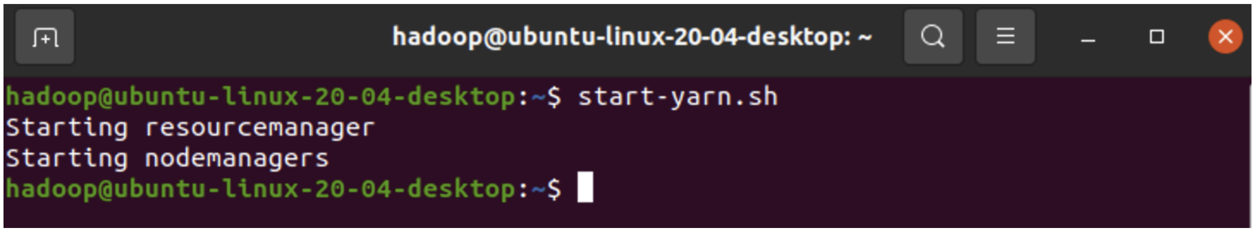
We are starting the DFS services using the **start-dfs.sh** command.



```
hadoop@ubuntu-linux-20-04-desktop:~$ start-dfs.sh
Starting namenodes on [0.0.0.0]
Starting datanodes
Starting secondary namenodes [ubuntu-linux-20-04-desktop]
2023-03-04 04:06:31,215 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
hadoop@ubuntu-linux-20-04-desktop:~$
```

1.3 Starting the Yarn services

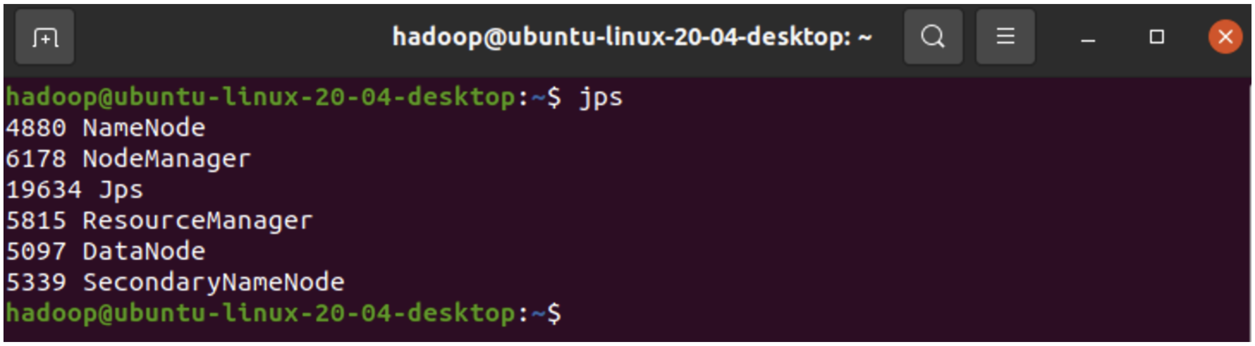
We are starting the yarn services using the **start-yarn.sh** command.



```
hadoop@ubuntu-linux-20-04-desktop:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@ubuntu-linux-20-04-desktop:~$
```

1.4 Verifying the running components

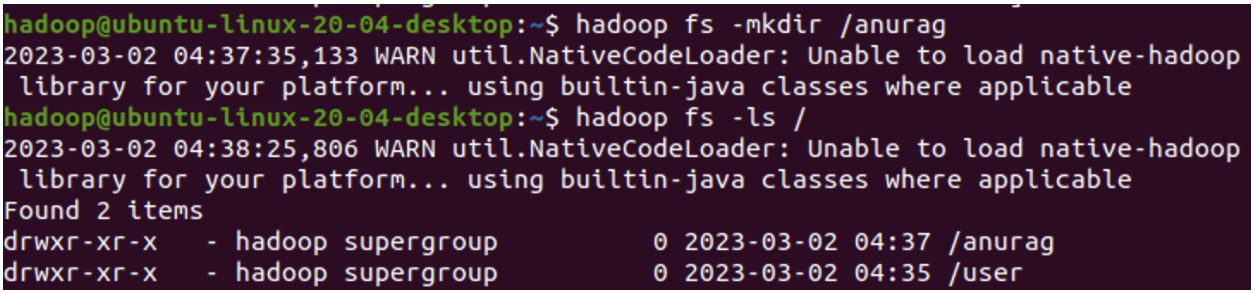
We are verifying the running components using the **jps** command.



```
hadoop@ubuntu-linux-20-04-desktop:~$ jps
4880 NameNode
6178 NodeManager
19634 Jps
5815 ResourceManager
5097 DataNode
5339 SecondaryNameNode
hadoop@ubuntu-linux-20-04-desktop:~$
```

1.5 Creating a directory in HDFS for storing the dataset

We used the **-mkdir** command to create a new directory named **anurag** and **-ls** to show the created folder.

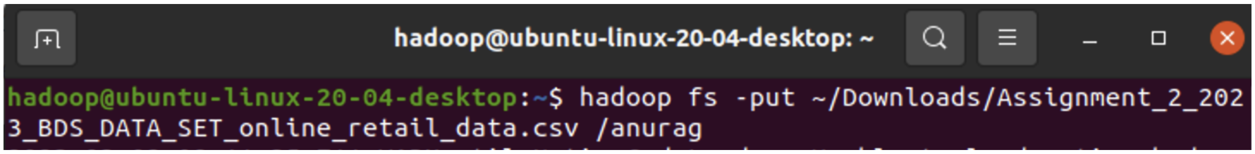


```
hadoop@ubuntu-linux-20-04-desktop:~$ hadoop fs -mkdir /anurag
2023-03-02 04:37:35,133 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
hadoop@ubuntu-linux-20-04-desktop:~$ hadoop fs -ls /
2023-03-02 04:38:25,806 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2023-03-02 04:37 /anurag
drwxr-xr-x   - hadoop supergroup          0 2023-03-02 04:35 /user
```

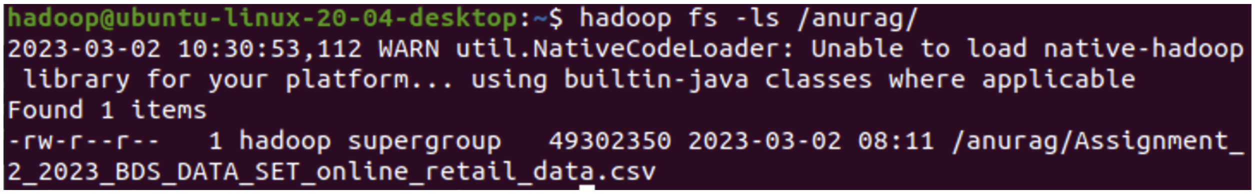
Note: We will always get a warning of **NativeCodeLoader**, the reason is that the native Hadoop Library "**$HADOOP_HOME/lib/native/libhadoop.so.1.0.0**" was actually complied to 32 bit and the OS we are using is Ubuntu 64 bit so that's why it will show this warning at step. However, it won't impact any Hadoop functionality, so kindly ignore it.

1.6 Uploading dataset into the Hadoop Distributed File System

Using **-put** command, we have uploaded the given dataset into the created hadoop directory.



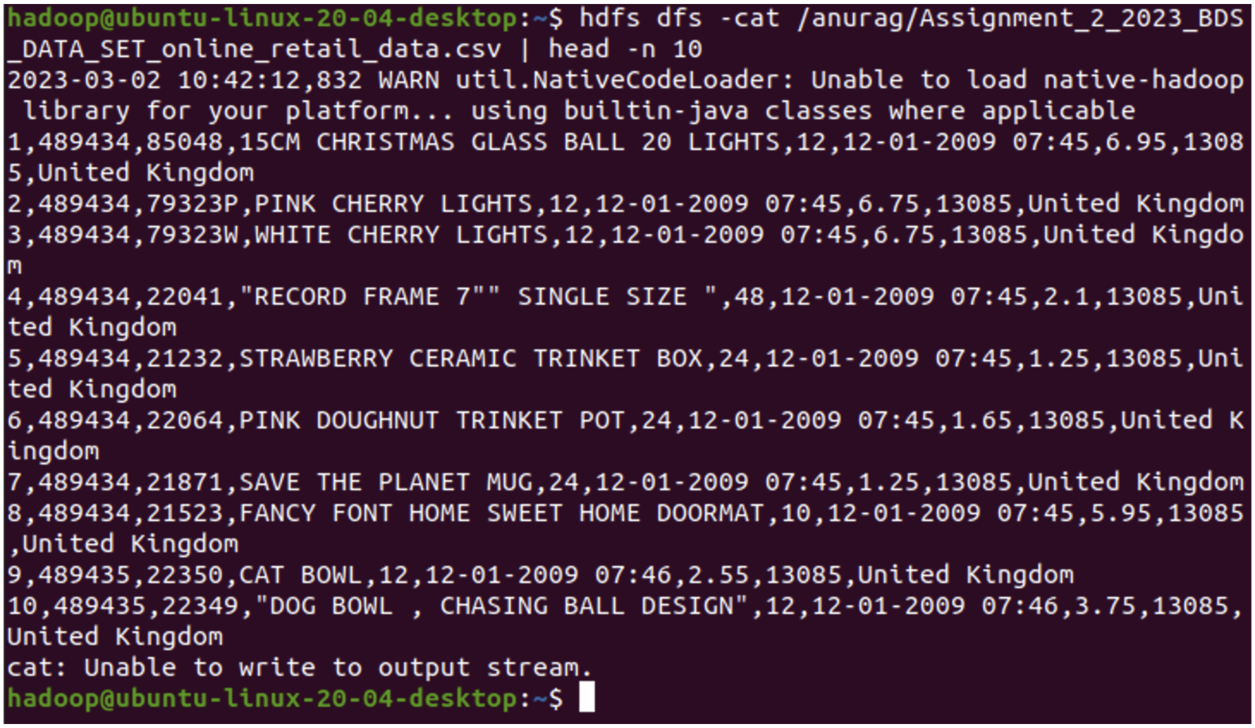
```
hadoop@ubuntu-linux-20-04-desktop:~$ hadoop fs -put ~/Downloads/Assignment_2_202
3_BDS_DATA_SET_online_retail_data.csv /anurag
```



```
hadoop@ubuntu-linux-20-04-desktop:~$ hadoop fs -ls /anurag/
2023-03-02 10:30:53,112 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   1 hadoop supergroup   49302350 2023-03-02 08:11 /anurag/Assignment_
2_2023_BDS_DATA_SET_online_retail_data.csv
```

1.7 Printing the first 10 rows of the given dataset

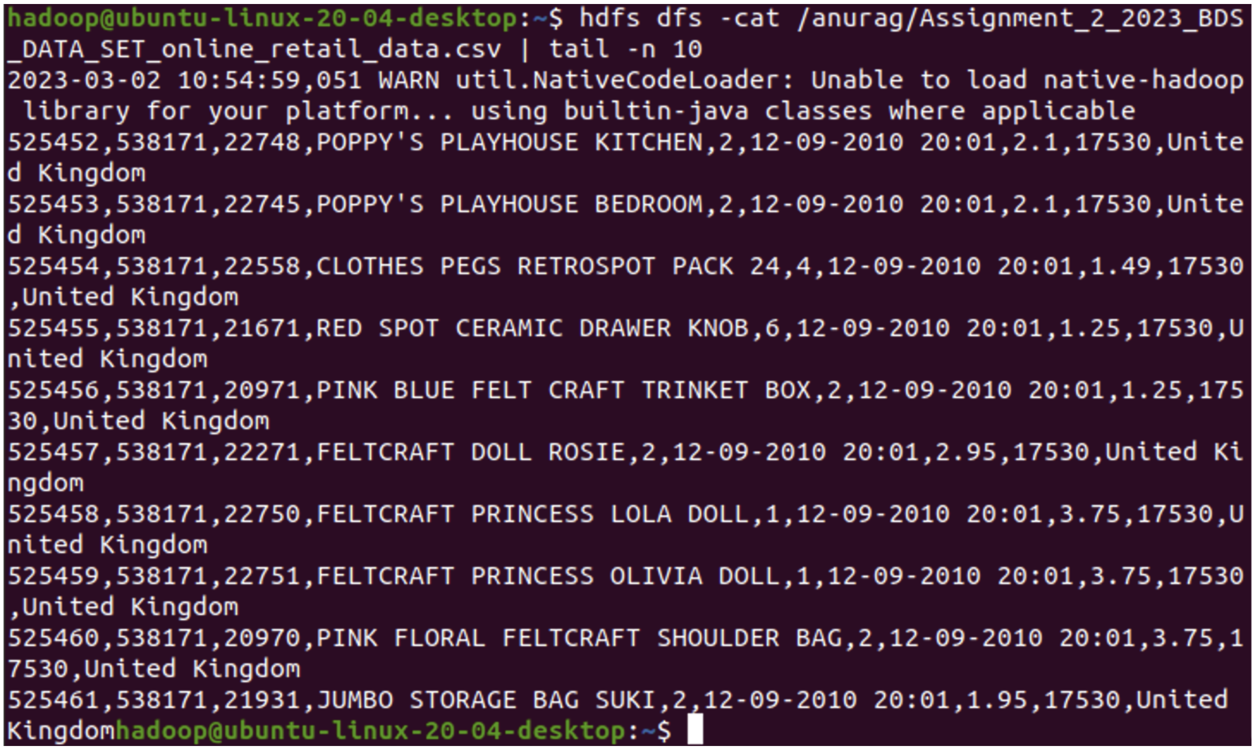
Using the **-cat** and **head** commands, we have printed the first 10 rows of the given dataset.



```
hadoop@ubuntu-linux-20-04-desktop:~$ hdfs dfs -cat /anurag/Assignment_2_2023_BDS
_DATA_SET_online_retail_data.csv | head -n 10
2023-03-02 10:42:12,832 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
1,489434,85048,15CM CHRISTMAS GLASS BALL 20 LIGHTS,12,12-01-2009 07:45,6.95,1308
5,United Kingdom
2,489434,79323P,PINK CHERRY LIGHTS,12,12-01-2009 07:45,6.75,13085,United Kingdom
3,489434,79323W,WHITE CHERRY LIGHTS,12,12-01-2009 07:45,6.75,13085,United Kingdo
m
4,489434,22041,"RECORD FRAME 7"" SINGLE SIZE ",48,12-01-2009 07:45,2.1,13085,Uni
ted Kingdom
5,489434,21232,STRAWBERRY CERAMIC TRINKET BOX,24,12-01-2009 07:45,1.25,13085,Uni
ted Kingdom
6,489434,22064,PINK DOUGHNUT TRINKET POT,24,12-01-2009 07:45,1.65,13085,United K
ingdom
7,489434,21871,SAVE THE PLANET MUG,24,12-01-2009 07:45,1.25,13085,United Kingdom
8,489434,21523,FANCY FONT HOME SWEET HOME DOORMAT,10,12-01-2009 07:45,5.95,13085
,United Kingdom
9,489435,22350,CAT BOWL,12,12-01-2009 07:46,2.55,13085,United Kingdom
10,489435,22349,"DOG BOWL , CHASING BALL DESIGN",12,12-01-2009 07:46,3.75,13085,
United Kingdom
cat: Unable to write to output stream.
hadoop@ubuntu-linux-20-04-desktop:~$ 
```

1.8 Printing the last 10 rows of the given dataset

Using the **-cat** and **tail** commands, we have printed the last 10 rows of the dataset.

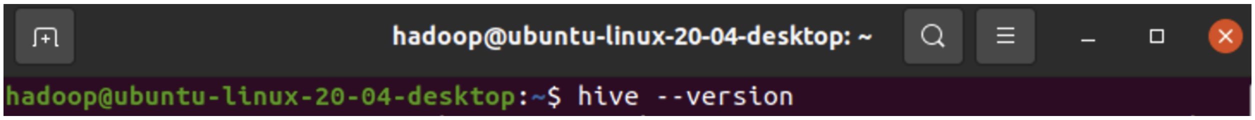


```
hadoop@ubuntu-linux-20-04-desktop:~$ hdfs dfs -cat /anurag/Assignment_2_2023_BDS
_DATA_SET_online_retail_data.csv | tail -n 10
2023-03-02 10:54:59,051 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
525452,538171,22748,POPPY'S PLAYHOUSE KITCHEN,2,12-09-2010 20:01,2.1,17530,Unite
d Kingdom
525453,538171,22745,POPPY'S PLAYHOUSE BEDROOM,2,12-09-2010 20:01,2.1,17530,Unite
d Kingdom
525454,538171,22558,CLOTHES PEGS RETROSPOT PACK 24,4,12-09-2010 20:01,1.49,17530
,United Kingdom
525455,538171,21671,RED SPOT CERAMIC DRAWER KNOB,6,12-09-2010 20:01,1.25,17530,U
nited Kingdom
525456,538171,20971,PINK BLUE FELT CRAFT TRINKET BOX,2,12-09-2010 20:01,1.25,175
30,United Kingdom
525457,538171,22271,FELTCRAFT DOLL ROSIE,2,12-09-2010 20:01,2.95,17530,United Ki
ngdom
525458,538171,22750,FELTCRAFT PRINCESS LOLA DOLL,1,12-09-2010 20:01,3.75,17530,U
nited Kingdom
525459,538171,22751,FELTCRAFT PRINCESS OLIVIA DOLL,1,12-09-2010 20:01,3.75,17530
,United Kingdom
525460,538171,20970,PINK FLORAL FELTCRAFT SHOULDER BAG,2,12-09-2010 20:01,3.75,1
7530,United Kingdom
525461,538171,21931,JUMBO STORAGE BAG SUKI,2,12-09-2010 20:01,1.95,17530,United 
Kingdomhadoop@ubuntu-linux-20-04-desktop:~$ 
```

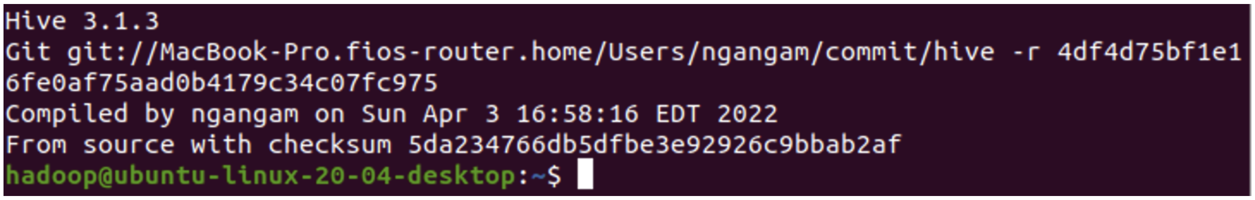
SETTING UP THE HIVE

2.1 Hive Version

Printing the hive version.



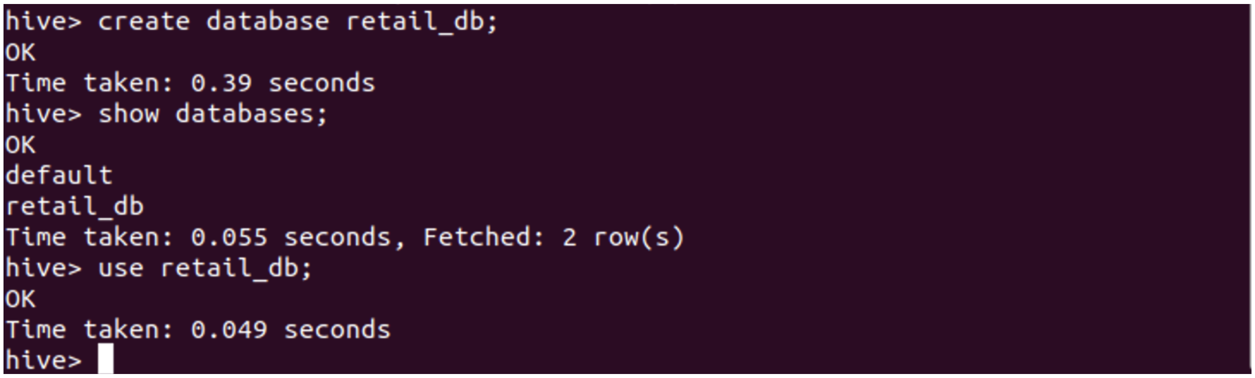
```
hadoop@ubuntu-linux-20-04-desktop:~$ hive --version
```



```
Hive 3.1.3
Git git://MacBook-Pro.fios-router.home/Users/ngangam/commit/hive -r 4df4d75bf1e1
6fe0af75aad0b4179c34c07fc975
Compiled by ngangam on Sun Apr 3 16:58:16 EDT 2022
From source with checksum 5da234766db5dfbe3e92926c9bbab2af
hadoop@ubuntu-linux-20-04-desktop:~$ 
```

2.2 Creating database in Hive

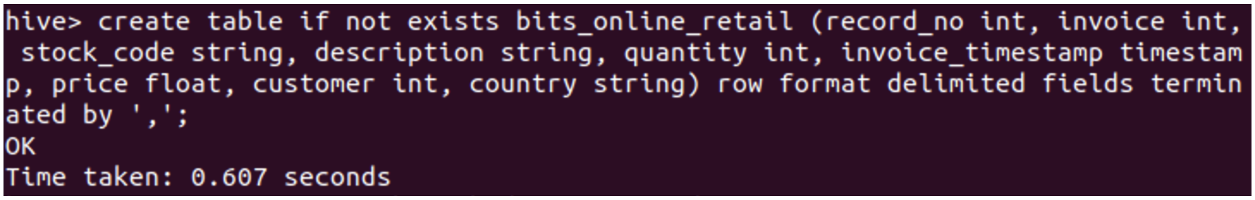
Using the create database query, we have created a new database named **retail_db**.



```
hive> create database retail_db;
OK
Time taken: 0.39 seconds
hive> show databases;
OK
default
retail_db
Time taken: 0.055 seconds, Fetched: 2 row(s)
hive> use retail_db;
OK
Time taken: 0.049 seconds
hive> 
```

2.3 Creating a table in Hive

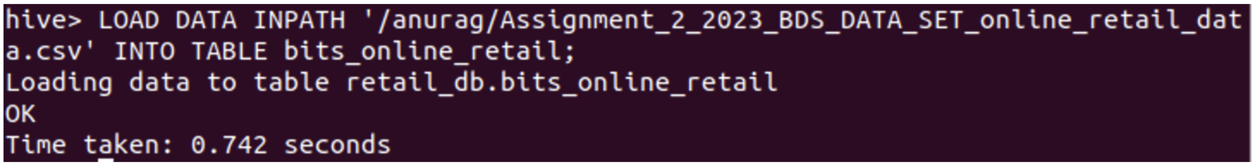
Using create table query, we have created a new table named **bits_online_retail**.



```
hive> create table if not exists bits_online_retail (record_no int, invoice int,
 stock_code string, description string, quantity int, invoice_timestamp timestam
p, price float, customer int, country string) row format delimited fields termin
ated by ',';
OK
Time taken: 0.607 seconds
```

2.4 Importing the values in the Hive table from Hadoop stored database

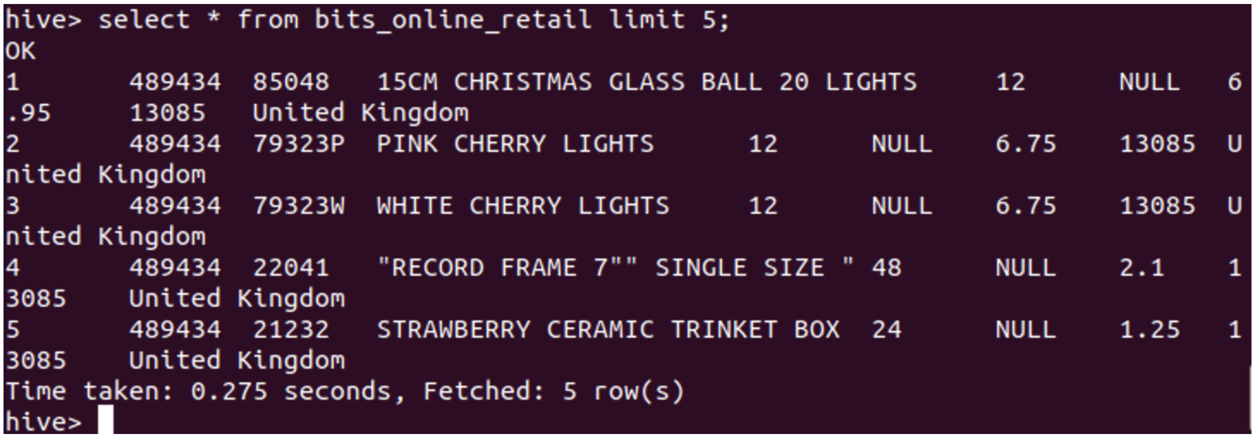
Using the **LOAD DATA** query, we have imported the csv file values from the Hadoop database that we created earlier.



```
hive> LOAD DATA INPATH '/anurag/Assignment_2_2023_BDS_DATA_SET_online_retail_dat
a.csv' INTO TABLE bits_online_retail;
Loading data to table retail_db.bits_online_retail
OK
Time taken: 0.742 seconds
```

2.5 Printing first 5 rows of Hive table

Using **select** and **limit** query, we have printed the first 5 rows of the table **bits_online_retail**.



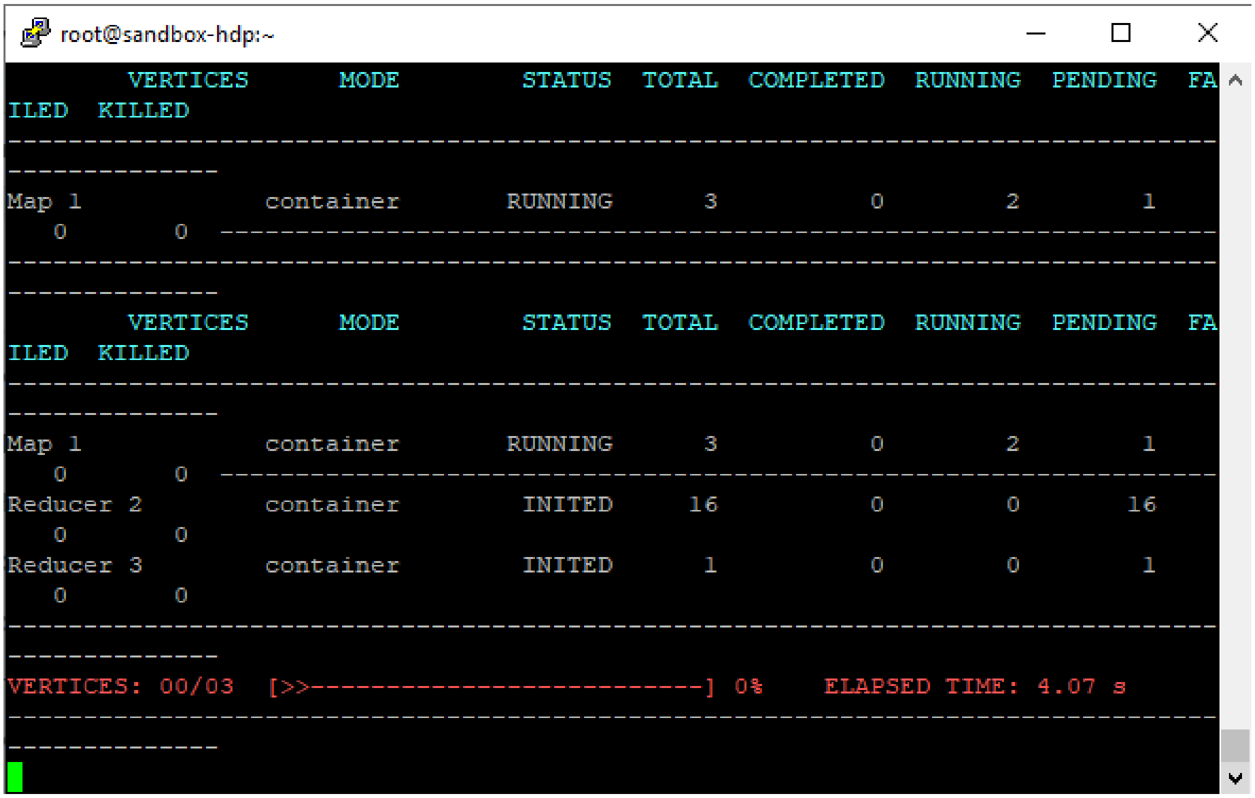
```
hive> select * from bits_online_retail limit 5;
OK
1       489434  85048   15CM CHRISTMAS GLASS BALL 20 LIGHTS     12      NULL    6
.95     13085   United Kingdom
2       489434  79323P  PINK CHERRY LIGHTS      12      NULL    6.75    13085   U
nited Kingdom
3       489434  79323W  WHITE CHERRY LIGHTS     12      NULL    6.75    13085   U
nited Kingdom
4       489434  22041   "RECORD FRAME 7"" SINGLE SIZE " 48      NULL    2.1     1
3085    United Kingdom
5       489434  21232   STRAWBERRY CERAMIC TRINKET BOX  24      NULL    1.25    1
3085    United Kingdom
Time taken: 0.275 seconds, Fetched: 5 row(s)
hive>
```

ANALYTICS

As we know Hive queries make use of MapReduce function so in order to make sure, the system doesn't go in hung state, so we have executed the queries inside the Sandbox.

MapReduce:

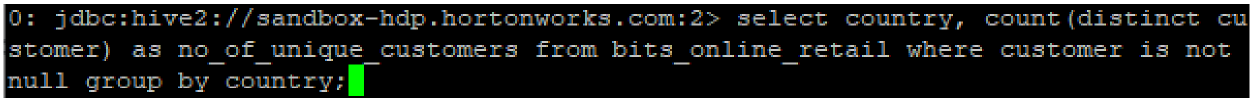
For answering the below questions, we have made use of MapReduce() function of Hive and Hadoop. Below is the snippet for reference.



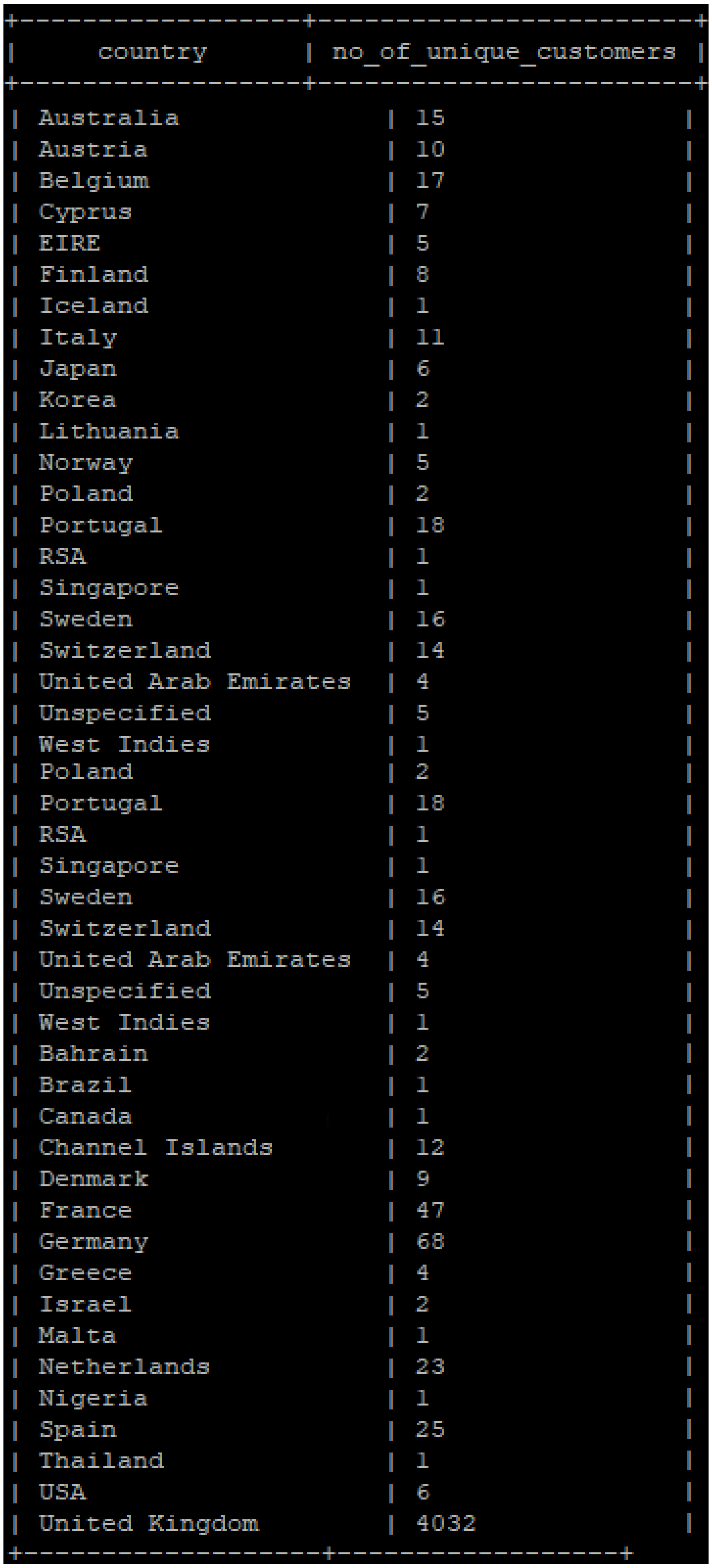
```
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FA
ILED  KILLED
--------------------------------------------------------------------------------
--------------
Map 1            container       RUNNING      3          0        2        1
   0       0  ------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FA
ILED  KILLED
--------------------------------------------------------------------------------
--------------
Map 1            container       RUNNING      3          0        2        1
   0       0  ------------------------------------------------------------------
Reducer 2        container        INITED     16          0        0       16
   0       0
Reducer 3        container        INITED      1          0        0        1
   0       0
--------------------------------------------------------------------------------
--------------
VERTICES: 00/03  [>>--------------------------] 0%    ELAPSED TIME: 4.07 s
--------------------------------------------------------------------------------
--------------
```

3.1 Total number of unique customers in the "given country".

For counting the unique customers, we are making use of the **DISTINCT** keyword and also we are making sure that we don’t count any **NULL** values so we are putting an extra where clause.



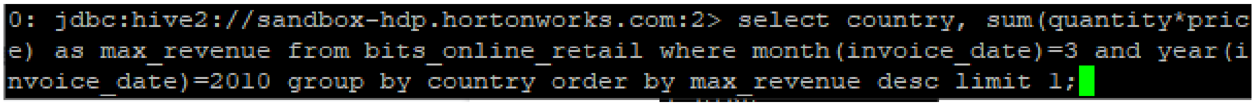
```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select country, count(distinct cu
stomer) as no_of_unique_customers from bits_online_retail where customer is not
null group by country;
```



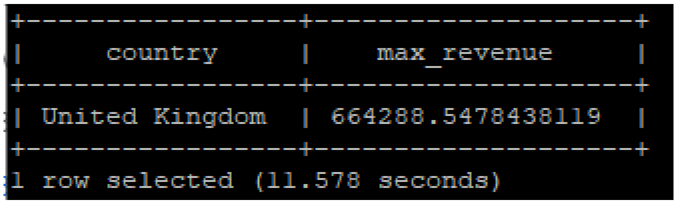
```
+-------------------+------------------------+
|      country      | no_of_unique_customers |
+-------------------+------------------------+
| Australia              | 15                 |
| Austria                | 10                 |
| Belgium                | 17                 |
| Cyprus                 | 7                  |
| EIRE                   | 5                  |
| Finland                | 8                  |
| Iceland                | 1                  |
| Italy                  | 11                 |
| Japan                  | 6                  |
| Korea                  | 2                  |
| Lithuania              | 1                  |
| Norway                 | 5                  |
| Poland                 | 2                  |
| Portugal               | 18                 |
| RSA                    | 1                  |
| Singapore              | 1                  |
| Sweden                 | 16                 |
| Switzerland            | 14                 |
| United Arab Emirates   | 4                  |
| Unspecified            | 5                  |
| West Indies            | 1                  |
| Poland                 | 2                  |
| Portugal               | 18                 |
| RSA                    | 1                  |
| Singapore              | 1                  |
| Sweden                 | 16                 |
| Switzerland            | 14                 |
| United Arab Emirates   | 4                  |
| Unspecified            | 5                  |
| West Indies            | 1                  |
| Bahrain                | 2                  |
| Brazil                 | 1                  |
| Canada                 | 1                  |
| Channel Islands        | 12                 |
| Denmark                | 9                  |
| France                 | 47                 |
| Germany                | 68                 |
| Greece                 | 4                  |
| Israel                 | 2                  |
| Malta                  | 1                  |
| Netherlands            | 23                 |
| Nigeria                | 1                  |
| Spain                  | 25                 |
| Thailand               | 1                  |
| USA                    | 6                  |
| United Kingdom         | 4032               |
+-------------------+-------------------+
```

3.2 Country from which the maximum revenue was collected from sales in the month of March 2010.

For getting this output, first we have used **GROUP BY** country, so that we can get country-wise data, then we have used **SUM** function to calculate to total revenue, after that we set the condition of March 2010 with the help of **WHERE** clause, and finally we have sorted the results in **DESC** order using **ORDER BY** clause.



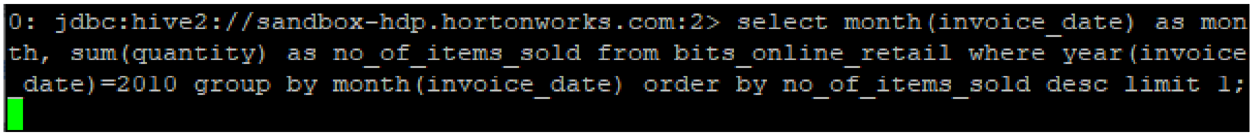
```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select country, sum(quantity*pric
e) as max_revenue from bits_online_retail where month(invoice_date)=3 and year(i
nvoice_date)=2010 group by country order by max_revenue desc limit 1;
```



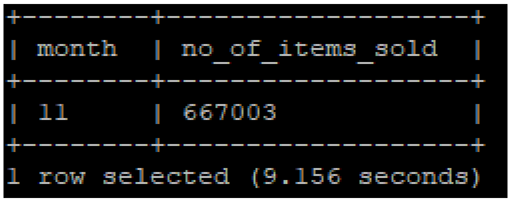
```
+-----------------+---------------------+
|     country     |     max_revenue     |
+-----------------+---------------------+
| United Kingdom  | 664288.5478438119   |
+-----------------+---------------------+
1 row selected (11.578 seconds)
```

3.3 Month of 2010 in which maximum number of items were sold.

Using the **month()** function, we have extracted the month from invoice column, then by applying the **sum()** function on quantity column, we have calculated the total no of items sold, and finally we have arranged the data in **DESC** order using the **ORDER BY** clause.



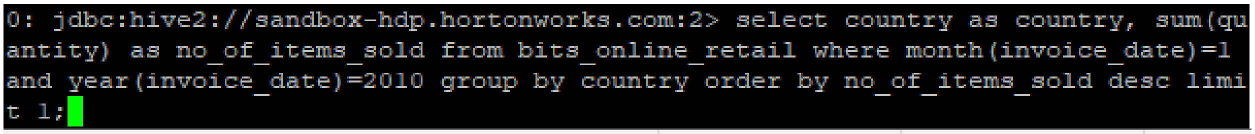
```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select month(invoice_date) as mon
th, sum(quantity) as no_of_items_sold from bits_online_retail where year(invoice
_date)=2010 group by month(invoice_date) order by no_of_items_sold desc limit 1;
```



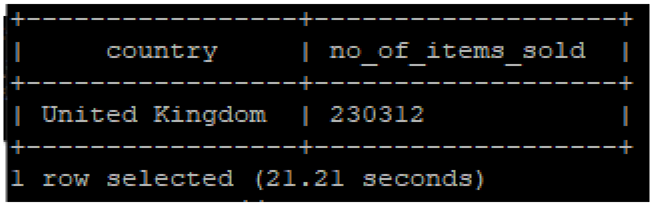
```
+--------+-------------------+
| month  | no_of_items_sold  |
+--------+-------------------+
| 11     | 667003            |
+--------+-------------------+
1 row selected (9.156 seconds)
```

3.4 In the month of January 2010, find the country in which the maximum number of items were sold.

Using the **sum()** function, we have calculated the total number of items sold and with the help of **month()** function we have set the **WHERE** clause to January month and with **year()** function, we set the year to 2010. Finally, we arranged the data in **DESC** using **ORDER BY** clause and since we need only the country with maximum no of items sold so we have used **LIMIT** clause to 1 so that it can only print the highest top record.



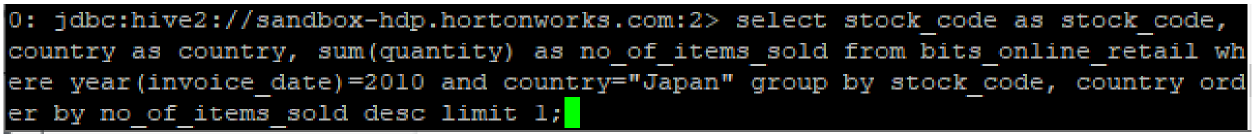
```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select country as country, sum(qu
antity) as no_of_items_sold from bits_online_retail where month(invoice_date)=1
and year(invoice_date)=2010 group by country order by no_of_items_sold desc limi
t 1;
```



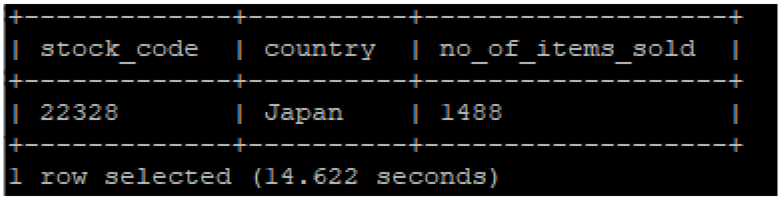
```
+-----------------+-------------------+
|     country     | no_of_items_sold  |
+-----------------+-------------------+
| United Kingdom  | 230312            |
+-----------------+-------------------+
1 row selected (21.21 seconds)
```

3.5 The StockCode of the item with the highest number of sales in the "given country" in the year 2010.

As we have a column named **stock_name** in our table **bits_online_retail**, so we have selected the same using **SELECT**, then have used **GROUP BY** clause on **stock_code** and **country**. Finally, we arranged the series in **DESC** order using the **ORDER BY** clause and printed the required top row using the **LIMIT** keyword.



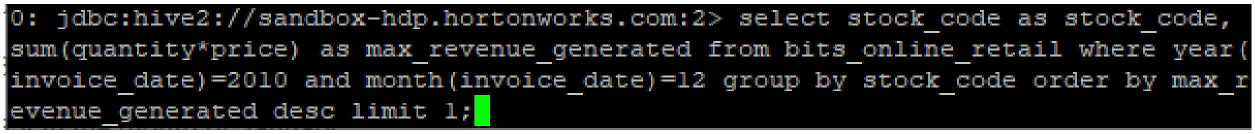
```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select stock_code as stock_code,
country as country, sum(quantity) as no_of_items_sold from bits_online_retail wh
ere year(invoice_date)=2010 and country="Japan" group by stock_code, country ord
er by no_of_items_sold desc limit 1;
```



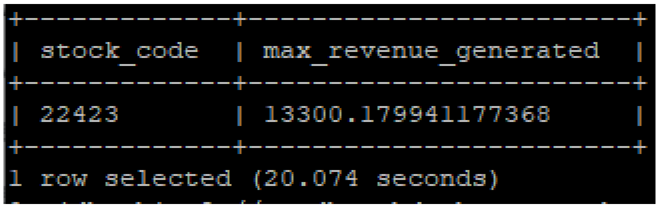
```
+-------------+----------+-------------------+
| stock_code  | country  | no_of_items_sold  |
+-------------+----------+-------------------+
| 22328       | Japan    | 1488              |
+-------------+----------+-------------------+
1 row selected (14.622 seconds)
```

3.6 StockCode of the item for which the maximum revenue was received by sales in the month of December 2010.

With the help of the **sum()** function, we have calculated the total revenue generated and using **WHERE** clause we have set the month to 12 i.e. December and the year to 2010. We have set the **GROUP BY** clause to **stock_code** so that the data is grouped as per the stock codes. Finally, we have arranged the rows in **DESC** order using the **ORDER BY** clause and we have printed the required top most row using the **LIMIT** keyword.



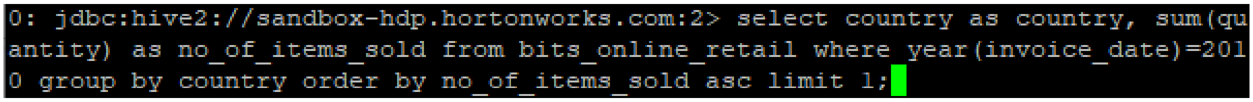
```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select stock_code as stock_code,
sum(quantity*price) as max_revenue_generated from bits_online_retail where year(
invoice_date)=2010 and month(invoice_date)=12 group by stock_code order by max_r
evenue_generated desc limit 1;
```



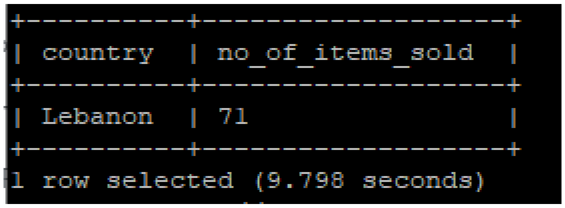
```
+-------------+------------------------+
| stock_code  | max_revenue_generated  |
+-------------+------------------------+
| 22423       | 13300.179941177368     |
+-------------+------------------------+
1 row selected (20.074 seconds)
```

3.7 The country in which the minimum number of sales happened in 2010.

Using the **sum()** function, we have calculated the total no of sales for each country. Also, with the help of the **WHERE** clause, we have set the year condition to 2010. Finally, we arranged the rows in **ASC** order of their no of sales by making use of **ORDER BY** clause and with **LIMIT** keyword, we have just printed the desired top most row.



```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select country as country, sum(qu
antity) as no_of_items_sold from bits_online_retail where year(invoice_date)=201
0 group by country order by no_of_items_sold asc limit 1;
```



```
+-----------+--------------------+
| country   | no_of_items_sold   |
+-----------+--------------------+
| Lebanon   | 71                 |
+-----------+--------------------+
1 row selected (9.798 seconds)
```

EVALUATOR'S SETUP

Instructions:

1. Prior Hadoop and Hive setup is required for executing the queries.
2. Load the CSV dataset in the Hadoop distributed file system from the local system.
3. Import the CSV from HDFS to Hive Metastore.
4. In the Hive Shell, now we can execute the above mentioned queries one by one.

-x-x- Thank You -x-x-