

HIVE ECOMM ASSIGNMENT - Nirakar Padhy

nirakarpadhy001@gmail.com

Launching and connecting to an EMR cluster via SSH:

And copying the 2019-Oct.csv and 2019-Nov.csv files to hadoop

```
(base) nirakar@user-ThinkPad-E15-Gen-2:~$ ssh -i ~/nirakar-key-pair.pem hadoop@ec2-35-175-121-255.compute-1.amazonaws.com
The authenticity of host 'ec2-35-175-121-255.compute-1.amazonaws.com (35.175.121.255)' can't be established.
ECDSA key fingerprint is SHA256:uUVA0xF1zaKHqXxyqUQn5oz4NXv6lRfJ3yah9APsoh0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-35-175-121-255.compute-1.amazonaws.com,35.175.121.255' (ECDSA) to the list of known hosts.

  _ _ _ _ _
 _ | ( _ | /
 _ | \ _ | _
  _ | _ _ | _

Amazon Linux AMI

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
68 package(s) needed for security, out of 106 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMM MMMMMMM RRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M M::::::::M R:::::::::R
EE::::::::EEEEEEEE::::E M::::::::M M::::::::M R:::::::::R
E::::E EEEEE M::::::::M M::::::::M RR::::R R::::R
E::::E M::::::::M M::::::::M R::::R R::::R
E::::EEEEEEEE M::::::::M M::::::::M R:::::::::R
E::::::::::::E M::::::::M M::::::::M R:::::::::R
E::::EEEEEEEE M::::::::M M::::::::M R:::::::::R
E::::E M::::::::M M::::::::M R::::R R::::R
E::::E EEEEE M::::::::M M M M::::::::M R::::R R::::R
EE::::::::EEEEEEEE::::E M::::::::M M::::::::M R::::R R::::R
E::::::::::::E M::::::::M M::::::::M RR::::R R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM MMMMMMM RRRRRRR RRRRRR

[hadoop@ip-172-31-72-185 ~]$ pwd
/home/hadoop
[hadoop@ip-172-31-72-185 ~]$ aws s3 cp s3://ecomhwddata/2019-Oct.csv .
download: s3://ecomhwddata/2019-Oct.csv to ./2019-Oct.csv
[hadoop@ip-172-31-72-185 ~]$ aws s3 cp s3://ecomhwddata/2019-Nov.csv .
download: s3://ecomhwddata/2019-Nov.csv to ./2019-Nov.csv
[hadoop@ip-172-31-72-185 ~]$ hive
```

Run following Commands on Hadoop EMR:

```
aws s3 cp <s3_path_to_dataset_bucket> <instance_path_>
```

```
hive # launching hive engine (can also use beeline connector)
```

Run following Commands on HIVE:

#we could use MR or TEZ exec engine: we use default taz engine

```
create database ecomm_hw; #creating new database
use ecomm_hw; #using the created db
```

#enabling dynamic partition for partitioning table using event_type for query optimisation

```
set hive.exec.dynamic.partition=true;  
set hive.exec.dynamic.partition.mode=nonstrict;
```

#creating two tables for 2019-Oct.csv and 2019-Nov.csv

#can be stored as textfile, parquet or avro format : for simplicity we use textfile

```
CREATE TABLE table_oct (event_time string, event_type string, product_id  
string, category_id string, category_code string, brand string, price float,  
user_id bigint, user_session string) ROW FORMAT DELIMITED FIELDS  
TERMINATED BY ',' LINES TERMINATED BY '\n' stored as textfile;
```

```
CREATE TABLE table_nov (event_time string, event_type string, product_id  
string, category_id string, category_code string, brand string, price float,  
user_id bigint, user_session string) ROW FORMAT DELIMITED FIELDS  
TERMINATED BY ',' LINES TERMINATED BY '\n' stored as textfile;
```

#loading the data from hdfs to hive tables

```
load data local inpath '/home/hadoop/2019-Oct.csv' into table table_oct;  
load data local inpath '/home/hadoop/2019-Nov.csv' into table table_nov;
```

You are required to provide answers to the questions given below.

1. Find the total revenue generated due to the purchases made in October.

```
select sum(price) from table_oct where event_type = 'purchase';
```

```

hive> select * from table_oct limit 3;
OK
event_time      event_type      product_id      category_id      category_code      brand
2019-10-01 00:00:00 UTC cart      5773203 1487580005134238553      runail 2.62
2019-10-01 00:00:03 UTC cart      5773353 1487580005134238553      runail 2.62
Time taken: 0.133 seconds, Fetched: 3 row(s)
hive> select sum(price) from table_oct where event_type = 'purchase';
Query ID = hadoop_20220118110905_ff1c691a-4d43-4db0-9cb2-d86e13babab1
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1642495356771_0006)

```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KI
Map 1	container	SUCCEEDED	10	10	0	0	0		
Reducer 2	container	SUCCEEDED	1	1	0	0	0		

VERTICES: 02/02 [=====>>] 100% ELAPSED TIME: 21.71 s

```

OK
1211538.4295325726
Time taken: 33.311 seconds, Fetched: 1 row(s)
hive> Select "October", sum(price) FROM table_oct where event_type = "purchase"
> UNION ALL
> Select "November", sum(price) FROM table_nov where event_type = "purchase"
> ;
Query ID = hadoop_20220118110954_f2eb49bd-36f7-4f41-bead-9c78303e7876
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1642495356771_0006)

```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KI
Map 1	container	SUCCEEDED	10	10	0	0	0		
Map 4	container	SUCCEEDED	11	11	0	0	0		
Reducer 2	container	SUCCEEDED	1	1	0	0	0		
Reducer 5	container	SUCCEEDED	1	1	0	0	0		

VERTICES: 04/04 [=====>>] 100% ELAPSED TIME: 28.00 s

```

OK
October 1211538.4295325726
November 1531016.8991247676
Time taken: 28.911 seconds, Fetched: 2 row(s)
hive> select distinct(coc, category_code) from

```

2. Write a query to yield the total sum of purchases per month in a single output.

```

Select "October", sum(price) FROM table_oct where event_type = "purchase"
UNION ALL
Select "November", sum(price) FROM table_nov where event_type = "purchase";

```

- Write a query to find the change in the revenue generated due to purchases made from October to November.

```
SELECT sum(nov_rev-oct_rev) as Difference  
FROM (  
select sum(price) as oct_rev, 0 as nov_rev FROM table_oct where event_type =  
"purchase" UNION ALL  
select 0 as oct_rev, sum(price) as nov_rev FROM table_nov where event_type =  
"purchase" ) unioned;
```

```
hive> SELECT sum(nov_rev-oct_rev) as Difference  
  > FROM (  
  > select sum(price) as oct_rev, 0 as nov_rev FROM table_oct where event_type = "purchase" UNION ALL  
  > select 0 as oct_rev, sum(price) as nov_rev FROM table_nov where event_type = "purchase" ) unioned;  
Query ID = hadoop_20220119140445_6e023ec6-e2b2-45db-a8d4-6f7a94bd389b  
Total jobs = 1  
Launching Job 1 out of 1  
Status: Running (Executing on YARN cluster with App id application_1642599511867_0004)  
  
-----  
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  
-----  
Map 1 ..... container  SUCCEEDED    10         10         0         0         0         0  
Map 5 ..... container  SUCCEEDED    10         10         0         0         0         0  
Reducer 2 ..... container  SUCCEEDED     1          1         0         0         0         0  
Reducer 4 ..... container  SUCCEEDED     1          1         0         0         0         0  
Reducer 6 ..... container  SUCCEEDED     1          1         0         0         0         0  
-----  
VERTICES: 05/05 [=====>>] 100% ELAPSED TIME: 22.28 s  
-----  
OK  
319478.469592195  
Time taken: 24.078 seconds, Fetched: 1 row(s)  
hive> 
```

- Find distinct categories of products.

```
select distinct(res.category_code) FROM ( select category_code from table_oct  
UNION ALL select category_code from table_nov) res;
```

```

NOVEMBER 15 10:03:12:7070
Time taken: 28.911 seconds, Fetched: 2 row(s)
hive> select distinct(res.category_code) from
> (
> select category_code from table_oct
> UNION ALL
> select category_code from table_nov
> ) res;
Query ID = hadoop_20220118111027_975806b5-0ec1-43d7-9fbb-37a4eec3f8a1
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1642495356771_0006)

-----
VERTICES      MODE           STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container    SUCCEEDED   10        10         0         0         0         0
Map 4 ..... container    SUCCEEDED   11        11         0         0         0         0
Reducer 3 ..... container  SUCCEEDED    5         5         0         0         0         0
-----
VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 13.22 s
-----
OK

accessories.cosmetic_bag
stationery.cartridge
accessories.bag
appliances.environment.vacuum
category_code
furniture.living_room.chair
sport.diving
appliances.personal.hair_cutter
appliances.environment.air_conditioner
apparel.glove
furniture.bathroom.bath
furniture.living_room.cabinet
Time taken: 13.915 seconds, Fetched: 13 row(s)
hive> SELECT unioned.category_code, count(unioned.product_id)

```

5. Find the total number of products available under each category.

```

SELECT unioned.category_code, count(unioned.product_id)
FROM (
  SELECT o.category_code, o.product_id FROM table_oct o
  UNION ALL
  SELECT n.category_code, n.product_id FROM table_nov n
) unioned
GROUP BY unioned.category_code;

```

```

Time taken: 13.915 seconds, Fetched: 13 row(s)
hive> SELECT unioned.category_code, count(unioned.product_id)
> FROM (
>   SELECT o.category_code, o.product_id FROM table_oct o
>   UNION ALL
>   SELECT n.category_code, n.product_id FROM table_nov n
> ) unioned
> GROUP BY unioned.category_code;
Query ID = hadoop_20220118111203_404a570e-a701-494f-b588-2ea0c856af26
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1642495356771_0006)

```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	SUCCEEDED	10	10	0	0	0	0	0
Map 4	container	SUCCEEDED	11	11	0	0	0	0	0
Reducer 3	container	SUCCEEDED	5	5	0	0	0	0	0

VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 31.43 s

OK

```

8594895
accessories.cosmetic_bag      1248
stationery.cartridge          26722
accessories.bag               11681
appliances.environment.vacuum  59761
category_code                 2
furniture.living_room.chair    308
sport.diving                   2
appliances.personal.hair_cutter 1643
appliances.environment.air_conditioner 332
apparel.glove                 18232
furniture.bathroom.bath       9857
furniture.living_room.cabinet  13439
Time taken: 32.22 seconds, Fetched: 13 row(s)
hive>

```

6. Which brand had the maximum sales in October and November combined?

```

SELECT unioned.brand, sum(unioned.price) as sales
FROM (
  SELECT o.price,o.brand FROM table_oct o where event_type = 'purchase'
  UNION ALL
  SELECT n.price,n.brand FROM table_nov n where event_type = 'purchase'
) unioned
GROUP BY unioned.brand
ORDER BY sales DESC
LIMIT 1;

```

```

> SELECT n.price,n.brand FROM table_nov n where event_type = 'purchase'
> ) unioned
> GROUP BY unioned.brand
> ORDER BY sales DESC
> LIMIT 1;
Query ID = hadoop_20220118111441_a74582ad-6a4c-41b5-b220-d99a7fc6f946
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1642495356771_0006)

```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	SUCCEEDED	10	10	0	0	0	0
Map 5	container	SUCCEEDED	11	11	0	0	0	0
Reducer 3	container	SUCCEEDED	3	3	0	0	0	0
Reducer 4	container	SUCCEEDED	1	1	0	0	0	0

VERTICES: 04/04 [=====>>] 100% ELAPSED TIME: 29.65 s

OK

```

1094188.2993474863
Time taken: 30.351 seconds, Fetched: 1 row(s)
hive> SELECT unioned.brand, sum(unioned.price) as sales
> FROM (
> SELECT o.price,o.brand FROM table_oct o where event_type = 'purchase'
> UNION ALL
> SELECT n.price,n.brand FROM table_nov n where event_type = 'purchase'
> ) unioned
> GROUP BY unioned.brand
> ORDER BY sales DESC
> LIMIT 2;

```

```

Query ID = hadoop_20220118111614_a5bb0f73-00d6-4594-a352-9ad5d9e5b2d9
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1642495356771_0006)

```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	SUCCEEDED	10	10	0	0	0	0
Map 5	container	SUCCEEDED	11	11	0	0	0	0
Reducer 3	container	SUCCEEDED	3	3	0	0	0	0
Reducer 4	container	SUCCEEDED	1	1	0	0	0	0

VERTICES: 04/04 [=====>>] 100% ELAPSED TIME: 32.06 s

OK

```

1094188.2993474863
runail 148297.93996394053
Time taken: 32.727 seconds, Fetched: 2 row(s)
hive>

```

7. Which brands increased their sales from October to November?

SELECT oct.brand FROM

(SELECT brand, sum(price) as sales FROM table_oct where event_type = 'purchase'
GROUP BY brand) oct

JOIN

(SELECT brand, sum(price) as sales FROM table_nov where event_type = 'purchase'
GROUP BY brand) nov

ON oct.brand=nov.brand

WHERE (nov.sales - oct.sales)>0;

```
Time taken: 32.727 seconds, Fetched: 2 row(s)
hive> SELECT oct.brand FROM
>
> (SELECT brand, sum(price) as sales FROM table_oct where event_type = 'purchase' GROUP BY brand ) oct
>
> JOIN
>
> (SELECT brand, sum(price) as sales FROM table_nov where event_type = 'purchase' GROUP BY brand) nov
>
> ON oct.brand=nov.brand
>
> WHERE (nov.sales - oct.sales)>0;
Query ID = hadoop_20220118111736_f6771379-348f-4463-95c6-4e71d81192c6
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1642495356771_0006)
```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	SUCCEEDED	10	10	0	0	0	0	0
Map 4	container	SUCCEEDED	11	11	0	0	0	0	0
Reducer 2	container	SUCCEEDED	1	1	0	0	0	0	0
Reducer 3	container	SUCCEEDED	2	2	0	0	0	0	0
Reducer 5	container	SUCCEEDED	2	2	0	0	0	0	0

```
VERTICES: 05/05 [=====>>] 100% ELAPSED TIME: 31.69 s
OK
art-visage
artex
batiste
beautix
beautyblender
bioaqua
biore
blixz
browxenna
carmex
concept
cutrin
```

- Your company wants to reward the top 10 users of its website with a Golden Customer plan. Write a query to generate a list of top 10 users who spend the most on purchases.

```
SELECT unioned.user_id,sum(unioned.price) as total_purchase
FROM (
SELECT o.user_id,o.price FROM table_oct o where event_type = 'purchase'
UNION ALL
SELECT n.user_id,n.price FROM table_nov n where event_type = 'purchase'
) unioned
GROUP BY unioned.user_id
ORDER BY total_purchase DESC
LIMIT 10;
```



```

veraclara
vilenta
yu-r
zeitun
Time taken: 32.555 seconds, Fetched: 153 row(s)
hive> SELECT unioned.user_id,sum(unioned.price) as total_purchase
  > FROM (
  >   SELECT o.user_id,o.price FROM table_oct o where event_type = 'purchase'
  >   UNION ALL
  >   SELECT n.user_id,n.price FROM table_nov n where event_type = 'purchase'
  > ) unioned
  > GROUP BY unioned.user_id
  > ORDER BY total_purchase DESC
  > LIMIT 10;
Query ID = hadoop_20220118111910_08d7f611-4c1e-4258-91a2-6ea123a1f1c0
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1642495356771_0006)

-----
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED   10         10         0         0         0
Map 5 ..... container  SUCCEEDED   11         11         0         0         0
Reducer 3 ..... container  SUCCEEDED    3          3         0         0         0
Reducer 4 ..... container  SUCCEEDED    1          1         0         0         0
-----
VERTICES: 04/04 [=====>>] 100% ELAPSED TIME: 32.38 s
-----
OK
557790271      2715.8699957430363
150318419      1645.970008611679
562167663      1352.8499938696623
531900924      1329.4499949514866
557850743      1295.4800310581923
522130011      1185.3899966478348
561592095      1109.700007289648
431950134      1097.5900000333786
566576008      1056.3600097894669
521347209      1040.9099964797497
Time taken: 33.036 seconds, Fetched: 10 row(s)
hive> 

```

OPTIMISATION OF QUERIES:

We can optimize queries by choosing appropriate storage format for tables. Additionally we can use methods such as indexing, partitioning, bucketing, etc. to make our queries faster.

Indexing: When the data set is large in size and a faster query execution is required we can use indexing on columns that are used more frequently. It is wise to use indexing for read-heavy applications, where you need to read the data more frequently and not write heavy-applications.

Hive Partition is a way to organize large tables into smaller logical tables based on values of columns; one logical table (partition) for each distinct value. In Hive, tables are created as a directory on HDFS. A table can have one or more partitions that correspond to a sub-directory for each partition inside a table directory. Hive Bucketing a.k.a (Clustering) is a technique to split the data into more manageable files, (By specifying the number of buckets to create). The value of the bucketing column will be hashed by a user-defined number into buckets. Bucketing can be created on just one column, you can also create bucketing on a partitioned table to further split the data which further improves the query performance of the partitioned table. Each bucket is stored as a file within the table's directory or the partitions directories. Note that partition creates a directory and you can have a partition on one or more columns; these are some of the differences between Hive partition and bucket.

As the size of data increases, partitioning and bucketing yields results much quicker than unoptimised queries.

```
create table table_oct_part (event_time string, product_id string, category_id
string,category_code string,brand string, price float, user_id bigint, user_session string)
PARTITIONED BY (event_type string) ROW FORMAT DELIMITED FIELDS TERMINATED BY
',' LINES TERMINATED BY '\n' stored as avro;
```

```
insert into table table_oct_part partition(event_type) select event_time, event_type, product_id,
category_id, category_code, brand, price, user_id, user_session from table_oct;
```

```
create table table_nov_part (event_time string, product_id string, category_id
string,category_code string,brand string, price float, user_id bigint, user_session string)
PARTITIONED BY (event_type string) ROW FORMAT DELIMITED FIELDS TERMINATED BY
',' LINES TERMINATED BY '\n' stored as avro;
```

```
insert into table table_nov_part partition(event_type) select event_time, event_type, product_id,
category_id, category_code, brand, price, user_id, user_session from table_nov;
```

Optimized Query Q8:

```
SELECT unioned.user_id,sum(unioned.price) as total_purchase
FROM (
  SELECT o.user_id,o.price FROM table_oct_part o where event_type = 'purchase'
  UNION ALL
  SELECT n.user_id,n.price FROM table_nov_part n where event_type = 'purchase'
) unioned
GROUP BY unioned.user_id
ORDER BY total_purchase DESC
LIMIT 10;
```

It takes 21.245 secs to run the above query vis-a-vis 33.036 secs in the previous case!