

# Hive Assignment

Step 1 : Get data files in Master node

wget <https://e-commerce-events-ml.s3.amazonaws.com/2019-Oct.csv>

wget <https://e-commerce-events-ml.s3.amazonaws.com/2019-Nov.csv>

Step 2 : remove UTC from the timestamp column before loading the data

```
sed -i -e 's/ UTC//g' 2019-Oct.csv
```

```
sed -i -e 's/ UTC//g' 2019-Nov.csv
```

Step 3 : Import csv files in HDFS by running following steps

```
hdfs dfs -mkdir /user/hive_assignment
```

```
hdfs dfs -put 2019-Nov.csv /user/hive_assignment/
```

```
hdfs dfs -put 2019-Oct.csv /user/hive_assignment/
```

Step 4 : Verify if the files are imported in HDFS successfully.

```
hdfs dfs -ls /user/hive_assignment/
```

```
hdfs dfs -cat /user/hive_assignment/2019-Nov.csv | head
```

```
hdfs dfs -cat /user/hive_assignment/2019-Oct.csv | head
```

Step 5 : Enter into Hive console by typing 'hive'

Step 6 : Create database and required tables using following commands

```
create database if not exists ECom comment "Created to store events done on website" with  
dbproperties('creator'='Prakash','date'='30-01-2021');
```

```
show databases;
```

```
use ecom;
```

```
create external table if not exists event_nov (event_time timestamp,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  
tblproperties("skip.header.line.count"="1");
```

```
create external table if not exists event_oct (event_time timestamp,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  
tblproperties("skip.header.line.count"="1");
```

```
show tables;
```

Step 7 : Load data to both tables created for October and November

```
load data inpath '/user/hive_assignment/2019-Nov.csv' into table event_nov;
```

```
load data inpath '/user/hive_assignment/2019-Oct.csv' into table event_oct;
```

```
set hive.cli.print.header=true;
```

```
select * from event_nov limit 10;
```

```
select * from event_nov limit 10;
```

Step 8 : Consolidate the data of October and November in a single table

```
create table events
```

```
as
```

```
(
```

```
select 'Nov' as Month,* from event_nov
```

```
union all
```

```
select 'Oct' as Month,* from event_oct
```

```
)
```

Step 9 : Create partitioning(Static partitioning was used) and bucketing. (I have chosen month and event\_type for partitioning and product\_id for bucketing. I chose this because most of the queries were based on event type and month. Both columns were having low cardinality)

```
create table if not exists part_events (event_time timestamp,product_id string,category_id string,category_code string,brand string,price float,user_id bigint,user_session string) partitioned by (month string,event_type string) clustered by (product_id) into 5 buckets row format delimited fields terminated by ',' lines terminated by '\n';
```

```
insert into part_events partition( month='Oct', event_type='cart') select event_time,product_id,category_id,category_code,brand,price,user_id, user_session from events where month='Oct' and event_type='cart';
```

```
insert into part_events partition( month='Oct', event_type='remove_from_cart') select event_time,product_id,category_id,category_code,brand,price,user_id, user_session from events where month='Oct' and event_type='remove_from_cart';
```

```
insert into part_events partition( month='Oct',event_type='view') select event_time,product_id,category_id,category_code,brand,price,user_id, user_session from events where month='Oct' and event_type='view';
```

```
insert into part_events partition( month='Oct',event_type='purchase') select
event_time,product_id,category_id,category_code,brand,price,user_id, user_session from events
where month='Oct' and event_type='purchase';
```

```
insert into part_events partition( month='Nov', event_type='cart') select
event_time,product_id,category_id,category_code,brand,price,user_id, user_session from events
where month='Nov' and event_type='cart';
```

```
insert into part_events partition(month='Nov',event_type='remove_from_cart') select
event_time,product_id,category_id,category_code,brand,price,user_id, user_session from events
where month='Nov' and event_type='remove_from_cart';
```

```
insert into part_events partition(month='Nov',event_type='view') select
event_time,product_id,category_id,category_code,brand,price,user_id, user_session from events
where month='Nov' and event_type='view';
```

```
insert into part_events partition(month='Nov',event_type='purchase') select
event_time,product_id,category_id,category_code,brand,price,user_id, user_session from events
where month='Nov' and event_type='purchase';
```

Step 10 : Verify if partitioning and bucketing is done

```
hdfs dfs -ls /user/hive/warehouse/ecom.db/part_events
```

Step 11 : Run a test if the query will be optimized after partitioning.(I have checked the performance for question #1 )

I have queried the events table directly which is not partitioned and it is taking 21 seconds. However when I used the partitioned table it is giving result in 12 seconds. There is a improvement of 43 percent. Please find the screenshot below.

`select sum(price) as Total_Revenue_from_purchases from events where month='Oct' and event_type='purchase';`

The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo, a refresh button, and fields for "Add a name..." and "Add a description...". On the right, there are icons for a chart, a document, and a menu. Below the header, the query is displayed: `1 select sum(price) as Total_Revenue_from_purchases from part_events where month='Oct' and event_type='purchase';`. The status bar indicates "20.97s Database ecom Type text". Below the query, the execution log shows: `INFO : Map 1: 4/4 Reducer 2: 1/1`, `INFO : Completed executing command(queryId=hive_20210206090924_4a8211fa-8f25-4626-900d-0844402ffc2f); Time taken: 20.818 seconds`, and `INFO : OK`. A link to the application ID is visible. At the bottom, there are tabs for "Query History", "Saved Queries", "Query Builder", and "Results (1)". The "Results (1)" tab is active, showing a table with one row: 

| total_revenue_from_purchases |                    |
|------------------------------|--------------------|
| 1                            | 1211538.4295325726 |

`select sum(price) as Total_Revenue_from_purchases from part_events where month='Oct' and event_type='purchase';`

The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo, a refresh button, and fields for "Add a name..." and "Add a description...". On the right, there are icons for a chart, a document, and a menu. Below the header, the query is displayed: `1 select sum(price) as Total_Revenue_from_purchases from part_events where month='Oct' and event_type='purchase';`. The status bar indicates "12.20s Database ecom Type text". Below the query, the execution log shows: `INFO : Map 1: 3/3 Reducer 2: 1/1`, `INFO : Completed executing command(queryId=hive_20210206091007_2bdca2d2-238c-4499-945e-f7855561a058); Time taken: 11.764 seconds`, and `INFO : OK`. A link to the application ID is visible. At the bottom, there are tabs for "Query History", "Saved Queries", "Query Builder", and "Results (1)". The "Results (1)" tab is active, showing a table with one row: 

| total_revenue_from_purchases |                    |
|------------------------------|--------------------|
| 1                            | 1211538.4295325726 |

Step 12 : Answer the Questions asked. Please find below the Answers of all queries/questions.

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

```
select sum(price) as Total_Revenue_from_purchases from part_events where month='Oct' and event_type='purchase';
```

The screenshot shows the Hive web interface. At the top, there's a header with the Hive logo, a refresh button, and fields for 'Add a name...' and 'Add a description...'. On the right, there are icons for a chart, a document, and a menu. Below the header, the query is displayed in a text area: `1 select sum(price) as Total_Revenue_from_purchases from part_events where month='Oct' and event_type='purchase';`. The status bar indicates '12.20s Database ecom Type text'. Below the query, there's a log section showing execution progress: 'INFO : Map 1: 3/3 Reducer 2: 1/1', 'INFO : Completed executing command(queryId=hive\_20210206091007\_2bdca2d2-238c-4499-945e-f78555e1a068); Time taken: 11.764 seconds', and 'INFO : OK'. A link to 'application\_1612593054396\_0010' is also visible. At the bottom, there are tabs for 'Query History', 'Saved Queries', 'Query Builder', and 'Results (1)'. The 'Results (1)' tab is active, showing a table with one row: 

| total_revenue_from_purchases |                    |
|------------------------------|--------------------|
| 1                            | 1211538.4295325726 |

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

```
select month, sum(price) as Sum_of_purchases from part_events where event_type='purchase' group by month ;
```

Hive
★
🔄
Add a name...
Add a description...

4.26s Database ecom Type text ⚙️ ?

```
1 select month, sum(price) as Sum_of_purchases from part_events where event_type='purchase' group by month ;
```

```
INFO : Map 1: 5/5 Reducer 2: 1/1/2
INFO : Completed executing command(queryId=hive_20210206092000_16864aae-e338-4f80-8720-6609c3eb7433); Time taken: 3.326 seconds
INFO : OK
```


Query History
Saved Queries
Query Builder
Results (2)

|   | month | sum_of_purchases   |
|---|-------|--------------------|
| 1 | Nov   | 1531016.8991247676 |
| 2 | Oct   | 1211538.4295325726 |

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

with MonthlyRevenue as  
(select month, sum(price) as Revenue\_purchases  
from part\_events  
where event\_type='purchase'  
group by month)


select (nov.Revenue\_purchases-oct.Revenue\_purchases) as change\_in\_revenue  
from MonthlyRevenue nov cross join MonthlyRevenue oct  
where nov.month<>oct.month and nov.month='Nov' and oct.month='Oct';


 Hive


↺

Add a name...

Add a description...







7.79s Database ecom ▾ Type text ▾ ⚙ ?

```
1 with MonthlyRevenue as
2   (select month, sum(price) as Revenue_purchases
3    from part_events
4    where event_type='purchase'
5    group by month)
6
7 select (nov.Revenue_purchases-oct.Revenue_purchases) as change_in_revenue
8 from MonthlyRevenue nov cross join MonthlyRevenue oct
9 where nov.month<>oct.month and nov.month='Nov' and oct.month='Oct';
```

INFO : Map 1: 5/5 Map 3: 3/3 Reducer 2: 2/2 Reducer 4: 2/2

INFO : Completed executing command(queryId=hive\_20210206121110\_08cb23cd-cb9a-4d53-b7a1-27247b66a742), Time taken: 7.792 seconds

INFO : OK

application\_1612593054396\_0016

Query History

Saved Queries

Query Builder

Results (1)



change\_in\_revenue

1 319478.469592195

Note: use set hive.strict.checks.cartesian.product = False for cross product.



- Find distinct categories of products.

```
select distinct category_id from part_events;
```

25.10s Database ecom ▾ Type text ▾  

1

```
select distinct category_id from part_events;
```

INFO : Map 1: 6/6 Reducer 2: 1/1

INFO : Completed executing command(queryId=hive\_20210206101510\_961f1d27-c500-4f00-803f-3f5d233d3b0c); Time taken: 24.369 seconds

INFO : OK

Query History Saved Queries Query Builder Results (100+)

|   | category_id         |
|---|---------------------|
| 1 | 1487580004832248652 |
| 2 | 1487580004857414477 |
| 3 | 1487580004882580302 |
| 4 | 1487580004916134735 |
| 5 | 1487580004966466385 |



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

```
with categories as (select distinct category_id, product_id from part_events)
select category_id, count(category_id) from categories group by category_id;
```

The screenshot shows the Hive query interface. At the top, there's a header with the Hive logo, a refresh icon, and fields for 'Add a name...' and 'Add a description...'. On the right, there are icons for a chart, a document, and a menu. Below the header, the query execution time is shown as '13.44s', and the database is 'ecom'. The query type is 'text'. The query itself is displayed in a code editor, and below it, the execution logs are visible. The logs show the progress of the query execution, including the number of maps and reducers, and the time taken. The results are displayed in a table with two columns: 'category\_id' and 'number\_of\_product'.

```
1 with categories as
2 (select distinct category_id, product_id
3  from part_events)
4
5 select category_id, count(product_id) as number_of_product
6  from categories
7  group by category_id;
8
```

INFO : Map 1: 6/6 Reducer 2: 1/1  
INFO : Completed executing command(queryId=hive\_20210206122139\_e3589b27-e4ba-4275-90c1-ad11d574b054); Time taken: 12.124 seconds  
INFO : OK

| category_id           | number_of_product |
|-----------------------|-------------------|
| 1 1487580004832248652 | 283               |
| 2 1487580004857414477 | 543               |
| 3 1487580004882580302 | 148               |
| 4 1487580004916134735 | 585               |
| 5 1487580004966466385 | 9                 |
| 6 1487580004983243602 | 43                |
| 7 1487580005008409427 | 460               |
| 8 1487580005025186644 | 1                 |

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

```
select brand, sum(price) as revenue from part_events where event_type='purchase' and
brand <> ""
group by brand order by revenue desc limit 1;
```

Add a name...
Add a description...

4.10s
Database ecom
Type text

```

1 select brand, sum(price) as revenue from part_events where event_type='purchase' and brand <> ""
2 group by brand order by revenue desc limit 1;

```

```

INFO : Map 1: 5/5    Reducer 2: 2/2    Reducer 3: 1/1
INFO : Completed executing command(queryId=hive_20210206111909_ffa6d3fa-a5dc-4b03-a17d-594cb3306c0d), Time taken: 3.266 seconds
INFO : OK

```

application\_1612593054396\_0015

Query History
Saved Queries
Query Builder
Results (1)

|   | brand  | revenue            |
|---|--------|--------------------|
| 1 | runail | 148297.93996394053 |

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


```


with BrandRevenue as
(select month, brand ,sum(price) as sales_per_brand
from part_events
where event_type='purchase'
and brand <> ""
group by month,brand)




select nov.brand
from BrandRevenue nov
inner join BrandRevenue oct
on nov.brand =oct.brand and nov.month <> oct.month
where nov.sales_per_brand >oct.sales_per_brand

```



 Hive

 Add a name... Add a description...

14.93s Database ecom ▾ Type text ▾ ⚙ ?

1

2

3

```
select user_id,sum(price) as purchase_Amount from part_events where event_type = 'purchase'
group by user_id order by purchase_Amount desc limit 10;
```

INFO : Map 1: 5/5 Reducer 2: 2/2 Reducer 3: 1/1

INFO : Map 1: 5/5 Reducer 2: 2/2 Reducer 3: 1/1

INFO : Completed executing command(queryId=hive\_20210206105917\_1c25d021-6bee-431a-9f59-dd3638b556ad); Time taken: 4.939 seconds





INFO : OK

Query History

Saved Queries

Query Builder

Results (10)



|   | user_id   | purchase_amount    |
|---|-----------|--------------------|
| 1 | 557790271 | 2715.8699957430363 |
| 2 | 150318419 | 1645.970008611679  |
| 3 | 562167663 | 1352.8499938696623 |
| 4 | 531900924 | 1329.4499949514866 |
| 5 | 557850743 | 1295.4800310581923 |
| 6 | 522130011 | 1185.3899966478348 |
| 7 | 561592095 | 1109.700007289648  |
| 8 | 431050134 | 1097.5900000333786 |