

HIVE CASE STUDY

By: Sonali Singh and Prasasti Choudhury

Loading the input data files into S3 bucket

hivecasestudyupgrad

Overview Properties Permissions Management Access points

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions

US East (N. Virginia)

Viewing 1 to 2

Name	Last modified	Size	Storage class
2019-Nov.csv	Oct 4, 2020 8:28:57 PM GMT+0530	520.6 MB	Standard
2019-Oct.csv	Oct 4, 2020 8:28:57 PM GMT+0530	460.2 MB	Standard

Creating EMR cluster

Clone Terminate AWS CLI export

Cluster: Hive-case-study **Starting** Configuring cluster software

Summary Application user interfaces Monitoring Hardware Configurations Events Steps Bootstrap actions

Summary

ID: j-ZX1T6R262DIK
Creation date: 2020-10-11 20:18 (UTC+5:30)
Elapsed time: 3 minutes
After last step completes: Cluster waits
Termination protection: Off [Change](#)
Tags: - [View All](#) / [Edit](#)
Master public DNS: ec2-54-160-191-251.compute-1.amazonaws.com [Copy](#)
[Connect to the Master Node Using SSH](#)

Configuration details

Release label: emr-5.29.0
Hadoop distribution: Amazon 2.8.5
Applications: Hive 2.3.6, Pig 0.17.0, Hue 4.4.0
Log URI: s3://aws-logs-065790028539-us-east-1/elasticmapreduce/ [Copy](#)

Creating a folder in HDFS to load data

```
[hadoop@ip-172-31-37-119 ~]$ hadoop fs -mkdir /user/hive/case-study
[hadoop@ip-172-31-37-119 ~]$ hadoop fs -ls /user/hive/
Found 2 items
drwxr-xr-x - hadoop hadoop 0 2020-10-11 14:56 /user/hive/case-study
drwxrwxrwt - hdfs hadoop 0 2020-10-11 14:54 /user/hive/warehouse
```

Moving the data from the S3 bucket into the HDFS

```
[hadoop@ip-172-31-40-10 ~]$ hadoop distcp 's3://hivecasestudyupgrad/*' '/user/hive/case-study/'
20/10/08 18:43:10 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=false, syncFolder=false, deleteMissing=false,
CRC=false, blocking=true, numListStatusThreads=0, maxMaps=20, mapBandwidth=100, s3lConfigurationFile='null', copyStrategy='un
attrs=false, atomicWorkPath=null, logPath=null, sourceFileListing=null, sourcePaths=[s3://hivecasestudyupgrad/*], targetPath=/
e, filtersFile='null'}
20/10/08 18:43:10 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-40-10.ec2.internal/172.31.40.10:8032
20/10/08 18:43:13 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 2; dirCnt = 0
20/10/08 18:43:13 INFO tools.SimpleCopyListing: Build file listing completed.
20/10/08 18:43:13 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.task.io.sort.mb
20/10/08 18:43:13 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort.factor
20/10/08 18:43:13 INFO tools.DistCp: Number of paths in the copy list: 2
20/10/08 18:43:13 INFO tools.DistCp: Number of paths in the copy list: 2
```

```
[hadoop@ip-172-31-40-10 ~]$ hadoop fs -ls /user/hive/case-study/
Found 2 items
-rw-r--r-- 1 hadoop hadoop 545839412 2020-10-08 18:43 /user/hive/case-study/2019-Nov.csv
-rw-r--r-- 1 hadoop hadoop 482542278 2020-10-08 18:43 /user/hive/case-study/2019-Oct.csv
```

Entering into Hive environment

```
[hadoop@ip-172-31-37-119 ~]$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive>
```

Setting the environment in Hive

```
hive> set hive.resultset.use.unique.column.names=false;
hive> set hive.execution.engine=mr;
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
ases.
hive> set hive.cli.print.header=true;
```

Creating the database and using it

```
hive> create database if not exists case_study;
OK
Time taken: 0.577 seconds
hive> use case_study;
OK
Time taken: 0.027 seconds
```

Creating the table to load both the data files

```
hive> create external table if not exists sales_input (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 serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
> stored as textfile
> location '/user/hive/case-study/'
> tblproperties ('skip.header.line.count' = '1');
OK
Time taken: 0.075 seconds
```

```
hive> select * from sales_input limit 3;
OK
event_time    event_type    product_id    category_id    category_code    brand    price    user_id    user_session
2019-11-01 00:00:02 UTC view    5802432 1487580009286598681
2019-11-01 00:00:09 UTC cart    5844397 1487580006317032337
2019-11-01 00:00:10 UTC view    5837166 1783999064103190764    pnb    22.22    556138645    57ed222e-a54a-4907-9944-5a875c2d7f4f
Time taken: 1.285 seconds, Fetched: 3 row(s)
```

Setting the environment for the dynamic partitioning

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

Creating table with the month 'October' and 'November' as partitions and bucketing on 'event_type' from the main table

```
hive> create table if not exists sales_part_bucket (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 )
> partitioned by (mnth int) clustered by (event_type) into 4 buckets
> row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
> stored as textfile;
OK
Time taken: 0.063 seconds
```

```
hive> insert into table sales_part_bucket partition(mnth)
> select *, month(event_time) as mnth from sales_input
> where month(event_time) in (10,11);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future version
1.X releases.
Query ID = hadoop_20201008193557_e7c3b0a4-21f7-414d-9ff8-23e8eeaa1848
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 5
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1602181695824_0003, Tracking URL = http://ip-172-31-40-10.ec2.internal
```

Checking the partitions created for the "sales_part_bucket" table

```
hive> show partitions sales_part_bucket;
OK
partition
mnth=10
mnth=11
Time taken: 0.058 seconds, Fetched: 2 row(s)
```

Checking for the tables created till now

```
hive> show tables;
OK
tab_name
sales_input
sales_part_bucket
Time taken: 0.115 seconds, Fetched: 2 row(s)
```

Checking the data inserted in 'sales_part_bucket' table

```
hive> select * from sales_part_bucket limit 1;
OK
event_time    event_type    product_id    category_id    category_code    brand    price    user_id    user_session    mnth
2019-10-23 16:30:13 UTC cart    5683350 1487580005671109489    masura    2.84    529605544    474873e7-c44f-4719-8734-869f65ccc824    10
Time taken: 0.124 seconds, Fetched: 1 row(s)
```

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

Using sales_input table

```
hive> select round(sum(price),2) as oct_revenue from sales_input where (month(event_time) = 10) and (event_type == 'purchase');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution
1.X releases.
Query ID = hadoop_20201009152748_b0b085f1-07b3-49ee-b10f-d56d633c0e10
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
```

```
MapReduce Total cumulative CPU time: 1 minutes 52 seconds 160 msec
Ended Job = job_1602255953905_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 4 Reduce: 1 Cumulative CPU: 112.16 sec HDFS Read: 1028867564 HDFS Write: 110 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 52 seconds 160 msec
OK
oct_revenue
1211538.43
Time taken: 92.398 seconds, Fetched: 1 row(s)
```

Using sales_part_bucket for optimisation for optimisation

```
hive> select round(sum(price),2) as oct_revenue from sales_part_bucket where mnth = 10 and event_type == 'purchase';
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
1.X releases.
Query ID = hadoop_20201009153706_84b2f3e0-70a1-4c4c-b900-9b525a480f06
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
```

```
MapReduce Total cumulative CPU time: 46 seconds 990 msec
Ended Job = job_1602255953905_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2 Reduce: 1 Cumulative CPU: 46.99 sec HDFS Read: 556596592 HDFS Write: 110 SUCCESS
Total MapReduce CPU Time Spent: 46 seconds 990 msec
OK
oct_revenue
1211538.43
Time taken: 49.612 seconds, Fetched: 1 row(s)
```

From the above, we can see that the query using the main table takes **92.398** seconds, where as when the same query is made using the partitioned table with buckets we get the output in only **49.612** seconds.

Output: So the revenue generated due to purchases in October is Rs. **1211538.43**.

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

Using sales_input table

```
hive> (select round(sum(price),2) as total_purchases from sales_input where (event_type == 'purchase' and month(event_time) = 10)) union (select round(sum(price),2) as
total_purchases from sales_input where (event_type == 'purchase' and month(event_time) = 11));
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
Query ID = hadoop_20201009155714_4af81b9a-a265-4b7f-aa13-d8953f2352bf
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 4 Reduce: 1 Cumulative CPU: 94.81 sec HDFS Read: 1028866855 HDFS Write: 121 SUCCESS
Stage-Stage-3: Map: 4 Reduce: 1 Cumulative CPU: 96.4 sec HDFS Read: 1028866867 HDFS Write: 121 SUCCESS
Stage-Stage-2: Map: 2 Reduce: 1 Cumulative CPU: 7.03 sec HDFS Read: 9912 HDFS Write: 132 SUCCESS
Total MapReduce CPU Time Spent: 3 minutes 18 seconds 240 msec
OK
total_purchases
1211538.43
1531016.9
Time taken: 198.648 seconds, Fetched: 2 row(s)
```

Using sales_part_bucket table for optimisation


```

hive> select mnth as month, round(sum(price),2) as total_purchases from sales_part_bucket where event_type == 'purchase' group by mnth;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
1.X releases.
Query ID = hadoop_20201009155101_0bdde6dd-1c92-4ad3-b06a-baac128d7be8
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 5
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>

MapReduce Jobs Launched:
Stage-Stage-1: Map: 4 Reduce: 5 Cumulative CPU: 111.39 sec HDFS Read: 1186114003 HDFS Write: 486 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 51 seconds 390 msec
OK
month    total_purchases
10       1211538.43
11       1531016.9
Time taken: 108.387 seconds, Fetched: 2 row(s)

```

From the above, we can see that the query using the main table takes **198.648** seconds, where as when the same query is made using the partitioned table with buckets we get the output in only **108.387** seconds.

Output: The total sum of the purchases per month is :- **October: 1211538.43** and **November: 1531016.90**.

Setting the environment to do the cartesian join

```

hive> set hive.strict.checks.cartesian.product=false;
hive> set hive.mapred.mode=nonstrict;

```

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

Using sales_input table

```

hive> select round((nov.Revenue - oct.Revenue),2) as diff_revenue
> from (select sum(price) as Revenue from sales_input where (month(event_time) = 10 and event_type == 'purchase')) as oct
> left outer join (select sum(price) as Revenue from sales_input where (month(event_time) = 11 and event_type == 'purchase')) as nov;
Warning: Map Join MAPJOIN[25][bigTable=?] in task 'Stage-4:MAPRED' is a cross product
Warning: Shuffle Join JOIN[16][tables = [$hdt$0, $hdt$1]] in Stage 'Stage-2:MAPRED' is a cross product
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (1.X releases).
Query ID = hadoop_20201009163231_a4df815c-f0ee-4e99-b01c-b3e65c7b64eb

MapReduce Jobs Launched:
Stage-Stage-1: Map: 4 Reduce: 1 Cumulative CPU: 113.36 sec HDFS Read: 1028866586 HDFS Write: 121 SUCCESS
Stage-Stage-3: Map: 4 Reduce: 1 Cumulative CPU: 113.86 sec HDFS Read: 1028866597 HDFS Write: 121 SUCCESS
Stage-Stage-4: Map: 1 Cumulative CPU: 3.13 sec HDFS Read: 5326 HDFS Write: 109 SUCCESS
Total MapReduce CPU Time Spent: 3 minutes 50 seconds 350 msec
OK
diff_revenue
319478.47
Time taken: 216.362 seconds, Fetched: 1 row(s)

```

Using sales_part_bucket table for optimisation

```

hive> select round((nov.Revenue - oct.Revenue),2) as diff_revenue
> from (select sum(price) as Revenue from sales_part_bucket where mnth = 10 and event_type == 'purchase') as oct
> left outer join (select sum(price) as Revenue from sales_part_bucket where mnth = 11 and event_type == 'purchase') as nov;
Warning: Map Join MAPJOIN[25][bigTable=?] in task 'Stage-4:MAPRED' is a cross product
Warning: Shuffle Join JOIN[16][tables = [$hdt$0, $hdt$1]] in Stage 'Stage-2:MAPRED' is a cross product
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (1.X releases).
Query ID = hadoop_20201009163231_a4df815c-f0ee-4e99-b01c-b3e65c7b64eb

Total MapReduce CPU Time Spent: 1 minutes 40 seconds 350 msec
OK
diff_revenue
319478.47
Time taken: 136.943 seconds, Fetched: 1 row(s)

```

From the above, we can see that the query using the main table takes **216.362** seconds, where as when the same query is made using the partitioned table with buckets we get the output in only **136.943** seconds.

Output: The change in the revenue generated due to purchases from October to November is **319478.47**

Creating table with 'category_code' as partitions from the main table

```
hive> create table if not exists sales_part_category(event_time string, event_type string,
> product_id string, category_id string, brand string, price float, user_id bigint, user_session string )
> partitioned by (category_code string)
> row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
> stored as textfile;
OK
Time taken: 0.265 seconds
```

Checking if the table is created properly

```
hive> show tables;
OK
tab name
sales_input
sales_part_bucket
sales_part_category
Time taken: 0.04 seconds, Fetched: 3 row(s)
```

Inserting the data from main table into 'sales_part_category' table

```
hive> insert into table sales_part_category partition (category_code)
> select event_time, event_type, product_id, category_id, brand, price, user_id, user_session, category_code
> from sales_input;
```

Checking for the partitions created in the category_code table

```
hive> show partitions sales_part_category;
OK
partition
category_code= __HIVE_DEFAULT_PARTITION__
category_code=accessories.bag
category_code=accessories.cosmetic_bag
category_code=apparel.glove
category_code=appliances.environment.air_conditioner
category_code=appliances.environment.vacuum
category_code=appliances.personal.hair_cutter
category_code=furniture.bathroom.bath
category_code=furniture.living_room.cabinet
category_code=furniture.living_room.chair
category_code=sport.diving
category_code=stationery.cartridge
Time taken: 0.087 seconds, Fetched: 12 row(s)
```

From the above we can see that an unnecessary partition '__HIVE_DEFAULT_PARTITION__' has been created where the category_code is null

Deleting the '__HIVE_DEFAULT_PARTITION__' partition from the table

```
hive> alter table sales_part_category drop if exists partition(category_code=' __HIVE_DEFAULT_PARTITION__ ');
Dropped the partition category_code= __HIVE_DEFAULT_PARTITION__
OK
Time taken: 0.497 seconds
```

4. Find distinct categories of products. Categories with null category code can be ignored.

Using sales_input table

```
hive> select category_code from sales_input group by category_code;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in
1.X releases.
Query ID = hadoop_20201009182732_b3d57e4f-965d-48c7-b144-91a1dfdbe5a9
Total jobs = 1
```

```
Total MapReduce CPU Time Spent: 1 minutes 35 seconds 30 msec
OK
category_code
appliances.personal.hair_cutter
accessories.cosmetic_bag
furniture.living_room.cabinet
stationery.cartridge
apparel.glove
appliances.environment.vacuum
accessories.bag
appliances.environment.air_conditioner
furniture.bathroom.bath
furniture.living_room.chair
sport.diving
Time taken: 106.027 seconds, Fetched: 12 row(s)
```

Using sales_part_category table for optimisation

```
hive> select category_code from sales_part_category group by category_code;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
1.X releases.
Query ID = hadoop_20201010165409_d997f4f5-bb09-4f6a-86fc-a62932fd1d54
Total jobs = 1
```

```
Total MapReduce CPU Time Spent: 5 seconds 770 msec
OK
category_code
accessories.bag
accessories.cosmetic_bag
apparel.glove
appliances.environment.air_conditioner
appliances.environment.vacuum
appliances.personal.hair_cutter
furniture.bathroom.bath
furniture.living_room.cabinet
furniture.living_room.chair
sport.diving
stationery.cartridge
Time taken: 27.146 seconds, Fetched: 11 row(s)
```

From the above, we can see that the query using the main table takes **106.027** seconds, where as when the same query is made using the partitioned table we get the output in only **27.146** seconds.

Output: The distinct categories of products are: accessories.bag, accessories.cosmetic_bag, apparel.glove, appliances.environment.air_conditioner, appliances.environment.vacuum, appliances.personal.hair_cutter, furniture.bathroom.bath, furniture.living_room.cabinet, furniture.living_room.chair, sport.diving and stationery.cartridge

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

Using sales_input table

```
hive> select category_code, count(product_id) as product_count from sales_input group by category_code;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider us
1.X releases.
Query ID = hadoop_20201010173351_8964593a-52f9-4e05-9b77-8c5a1b0aa584
Total jobs = 1
```

```
Total MapReduce CPU Time Spent: 1 minutes 37 seconds 980 msec
OK
category_code    product_count
8594895
appliances.personal.hair_cutter 1643
accessories.cosmetic_bag      1248
furniture.living_room.cabinet  13439
stationery.cartridge          26722
apparel.glove                18232
appliances.environment.vacuum  59761
accessories.bag              11681
appliances.environment.air_conditioner 332
furniture.bathroom.bath      9857
furniture.living_room.chair   308
sport.diving                 2
Time taken: 103.994 seconds, Fetched: 12 row(s)
```

Using sales_part_category table for optimisation

```
hive> select category_code, count(product_id) as product_count from sales_part_category group by category_code;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a di
1.X releases.
Query ID = hadoop_20201010174055_00ee4fee-3a39-4c06-9aac-be22c01eac2b
Total jobs = 1
```

```
Total MapReduce CPU Time Spent: 6 seconds 740 msec
OK
category_code    product_count
accessories.bag  11681
accessories.cosmetic_bag      1248
apparel.glove    18232
appliances.environment.air_conditioner 332
appliances.environment.vacuum  59761
appliances.personal.hair_cutter 1643
furniture.bathroom.bath      9857
furniture.living_room.cabinet 13439
furniture.living_room.chair   308
sport.diving        2
stationery.cartridge 26722
Time taken: 30.155 seconds, Fetched: 11 row(s)
```

From the above, we can see that the query using the main table takes **103.994** seconds, whereas when the same query is made using the partitioned table we get the output in only **30.155** seconds.

Creating table with month and brand as partitions and bucketing using 'event_type' from the main table

```
hive> create table if not exists sales_part_brand (event_time timestamp, event_type string,
> product_id string, category_id string, category_code string, price float, user_id bigint, user_session string)
> partitioned by (mnth int, brand string) clustered by (event_type) into 4 buckets
> row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
> stored as textfile;
OK
Time taken: 0.07 seconds
```

Checking if the table is created properly

```
hive> show tables;
OK
tab_name
sales_input
sales_part_brand
sales_part_bucket
sales_part_category
Time taken: 0.023 seconds, Fetched: 4 row(s)
```

Setting the environment to increase the number of partitions

```
hive> set hive.exec.max.dynamic.partitions=100000;
hive> set hive.exec.max.dynamic.partitions.pernode=100000;
```

Inserting the data from main table into 'sales_part_brand' table

```
hive> insert into table sales_part_brand partition(mnth, brand) select event_time, event_type, product_id, category_id, category_code, price, user_id, user_session, month(event_time) as mnth, brand from sales_input where month(event_time) in (10,11);
```

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

Using sales_input table

```
hive> select brand, round(sum(price),2) as total_sales from sales_input
> where event_type = 'purchase' and brand is not null
> group by brand order by total_sales desc limit 5;
```

```
brand    total_sales
runail   1094188.3
runail   148297.94
grattol  106918.25
irisk    92538.0
uno      86341.78
Time taken: 140.638 seconds, Fetched: 5 row(s)
```

Using sales_part_brand table for optimisation

```
hive> select brand, round(sum(price),2) as total_sales from sales_part_brand
> where brand is not null and event_type = 'purchase'
> group by brand order by total_sales desc limit 5;
```

```
brand    total_sales
runail   148297.94
grattol  106918.25
irisk    92538.0
uno      86341.78
strong   67867.9
Time taken: 116.123 seconds, Fetched: 5 row(s)
```

From the above, we can see that the query using the main table takes **140.638** seconds, whereas when the same query is made using the partitioned table we get the output in only **116.123** seconds.

Output: The brand with the maximum sales in October and November combined is **Runail**.

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

Using sales_input table


```
hive> with oct as
> (select brand, sum(price) as total_sales from sales_input where (month(event_time)=10 and event_type = 'purchase' and brand is not null) group by brand),
> nov as
> (select brand, sum(price) as total_sales from sales_input where (month(event_time)=11 and event_type = 'purchase' and brand is not null) group by brand)
> select nov.brand, round((nov.total_sales - oct.total_sales)/2) as diff_sales from nov inner join oct on nov.brand = oct.brand
> where (nov.total_sales - oct.total_sales) >0 order by diff_sales desc;
```

brand	diff_sales	markell	1065.68	skinlite	238.51	beautyblender	30.67
grattol	36027.17	sanoto	1052.54	provoc	235.83	biore	29.66
uno	15737.72	nagaraku	957.94	fedua	211.43	orly	28.71
lianail	10501.4	ecolab	951.45	ecocraft	200.79	estelare	27.06
ingarden	10404.82	art-visage	905.09	keen	199.27	profepil	24.66
strong	9474.64	levissime	857.81	mane	193.47	blixz	24.45
lessnail	7057.39	missha	856.45	freshbubbli	183.64	godefroy	23.9
cosmoprofi	6214.18	solomeya	786.1	chi	179.67	glysolid	21.86
polarus	5358.21	rosi	764.52	cristalinas	157.32	veraclar	21.1
runail	5219.38	refectocil	759.4	farmona	150.97	kamill	18.48
freedecor	4250.02	kaaral	673.64	latinolil	135.07	treaclemoon	18.12
staleks	3355.88	kosmekka	631.93	misikin	135.03	supertan	16.14
bpw.style	3265.29	kinetics	611.01	elizavecca	133.77	deoproce	12.33
lovely	3234.68	browkenna	585.36	nefertiti	133.12		
marathon	2992.35	airnails	572.62	finish	132.0		
haruyama	2962.22	uskuai	548.04	agrobeauty	131.41		
yoko	2950.97	coffin	525.49	dizao	126.38		
italwax	2859.13	s.care	500.39	osmo	116.73		
benovoy	2850.35	limoni	487.7	batiste	101.77		
kaypro	2387.36	matrix	483.49	carmex	98.28		
estel	2385.92	gehwol	468.61	eos	98.27		
concept	2348.26	greymy	460.28	depilflax	96.71		
kapous	2165.92	bioaqua	455.23	enjoy	95.22		
f.o.w	1959.05	farmavita	454.6	kerasys	94.29		
masura	1792.39	sophin	447.66	aura	93.56		
milv	1737.07	yu-r	402.3	plazan	92.64		
beautix	1729.0	kiss	395.78	koelf	84.56		
artex	1596.61	lador	387.92	nirvel	71.29		
domix	1537.12	ellips	360.19	konad	70.84		
shik	1498.52	jas	338.47	egomania	68.57		
smart	1444.88	lowence	324.91	cutrin	68.25		
roubloff	1422.41	nitriple	315.4	laboratorium	66.02		
levrana	1420.54	shary	304.53	inn	63.19		
onik	1416.24	kiss	302.0	marutaka-foot	60.11		
irisk	1354.08	happyfons	289.67	profhenna	57.62		
severina	1344.6	kocostar	284.08	koelcia	57.25		
joico	1309.58	insight	278.26	balbcare	57.05		
zeitun	1300.97	candy	264.42	elskin	56.56		
beauty-free	1228.69	bluesky	258.29	foamie	45.45		
swarovski	1155.23	beaugreen	256.84	ladykin	44.92		
de.lux	1115.81	protokeratin	255.84	likato	44.91		
metzger	1083.71	trind	244.89	mavala	37.28		
markell	1065.68	entity	239.55	vilenta	33.61		
		skinlite	238.51	beautyblender	30.67		

Time taken: 291.087 seconds, Fetched: 153 row(s)

Using sales_part_brand table for optimisation

```
hive> with oct as
> (select brand, sum(price) as total_sales from sales_part_brand where (mnth=10 and event_type = 'purchase' and brand is not null) group by brand),
> nov as
> (select brand, sum(price) as total_sales from sales_part_brand where (mnth=11 and event_type = 'purchase' and brand is not null) group by brand)
> select nov.brand, round((nov.total_sales - oct.total_sales)/2) as diff_sales from nov inner join oct on nov.brand = oct.brand
> where (nov.total_sales - oct.total_sales) >0 order by diff_sales desc;
```

brand	diff_sales	sanoto	1052.54	provoc	235.83	biore	29.66
grattol	36027.17	nagaraku	957.94	fedua	211.43	orly	28.71
uno	15737.72	ecolab	951.45	ecocraft	200.79	estelare	27.06
lianail	10501.4	art-visage	905.09	keen	199.27	profepil	24.66
ingarden	10404.82	levissime	857.81	mane	193.47	blixz	24.45
strong	9474.64	missha	856.45	freshbubble	183.64	godefroy	23.9
lessnail	7057.39	solomeya	786.1	chi	179.67	glysolid	21.86
cosmoprofi	6214.18	rosi	764.52	cristalinas	157.32	veraclar	21.1
polarus	5358.21	refectocil	759.4	farmona	150.97	kamill	18.48
runail	5219.38	kaaral	673.64	latinolil	135.07	treaclemoon	18.12
freedecor	4250.02	kosmekka	631.93	misikin	135.03	supertan	16.14
staleks	3355.88	kinetics	611.01	elizavecca	133.77	deoproce	12.33
bpw.style	3265.29	browkenna	585.36	nefertiti	133.12		
lovely	3234.68	airnails	572.62	finish	132.0		
marathon	2992.35	uskuai	548.04	agrobeauty	131.41		
haruyama	2962.22	coffin	525.49	dizao	126.38		
yoko	2950.97	s.care	500.39	osmo	116.73		
italwax	2859.13	limoni	487.7	batiste	101.77		
benovoy	2850.35	matrix	483.49	carmex	98.28		
kaypro	2387.36	gehwol	468.61	eos	98.27		
estel	2385.92	greymy	460.28	depilflax	96.71		
concept	2348.26	bioaqua	455.23	enjoy	95.22		
kapous	2165.92	farmavita	454.6	kerasys	94.29		
f.o.w	1959.05	sophin	447.66	aura	93.56		
masura	1792.39	yu-r	402.3	plazan	92.64		
milv	1737.07	kiss	395.78	koelf	84.56		
beautix	1729.0	lador	387.92	nirvel	71.29		
artex	1596.61	ellips	360.19	konad	70.84		
domix	1537.12	jas	338.47	egomania	68.57		
shik	1498.52	lowence	324.91	cutrin	68.25		
smart	1444.88	nitriple	315.4	laboratorium	66.02		
roubloff	1422.41	shary	304.53	inn	63.19		
levrana	1420.54	kiss	302.0	marutaka-foot	60.11		
onik	1416.24	happyfons	289.67	profhenna	57.62		
irisk	1354.08	kocostar	284.08	koelcia	57.25		
severina	1344.6	insight	278.26	balbcare	57.05		
joico	1309.58	candy	264.42	elskin	56.56		
zeitun	1300.97	bluesky	258.29	foamie	45.45		
beauty-free	1228.69	beaugreen	256.84	ladykin	44.92		
swarovski	1155.23	protokeratin	255.84	likato	44.91		
de.lux	1115.81	trind	244.89	mavala	37.28		
metzger	1083.71	entity	239.55	vilenta	33.61		
markell	1065.68	skinlite	238.51	beautyblender	30.67		

Time taken: 181.507 seconds, Fetched: 152 row(s)

From the above, we can see that the query using the main table takes **291.087** seconds, whereas when the same query is made using the partitioned table we get the output in only **181.507** seconds.

Creating table with 'event_type' as partitions and bucketing using 'user_id' from the main table

```
hive> create table if not exists sales_part_userid (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) clustered by (user_id) into 32 buckets
> row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
> stored as textfile;
OK
Time taken: 0.221 seconds
```

Checking if the table is created properly

```
hive> show tables;
OK
tab_name
sales_input
sales_part_brand
sales_part_bucket
sales_part_category
sales_part_userid
Time taken: 0.022 seconds, Fetched: 5 row(s)
```

Inserting the data from main table into 'sales_part_userid' table

```
hive> insert into table sales_part_userid partition(event_type)
> select event_time, product_id, category_id, category_code, brand, price, user_id, user_session, event_type from sales_input;
```

7. Write a query to generate a list of top 10 users who spend the most.

Using sales_input table

```
hive> select user_id, round(sum(price),2) as total_purchase from sales_input where event_type = 'purchase'
> group by user_id order by total_purchase desc limit 10;
```

```
user_id total_purchase
557790271      2715.87
150318419      1645.97
562167663      1352.85
531900924      1329.45
557850743      1295.48
522130011      1185.39
561592095      1109.7
431950134      1097.59
566576008      1056.36
521347209      1040.91
Time taken: 137.652 seconds, Fetched: 10 row(s)
```

Using sales_part_userid table for optimisation

```
hive> select user_id, round(sum(price),2) as total_purchase from sales_part_userid where event_type = 'purchase'
> group by user_id order by total_purchase desc limit 10;
```

```
user_id total_purchase
557790271      2715.87
150318419      1645.97
562167663      1352.85
531900924      1329.45
557850743      1295.48
522130011      1185.39
561592095      1109.7
431950134      1097.59
566576008      1056.36
521347209      1040.91
Time taken: 58.596 seconds, Fetched: 10 row(s)
```

From the above, we can see that the query using the main table takes **137.652** seconds, whereas when the same query is made using the partitioned table we get the output in only **58.596** seconds.

Output: The top 10 users with the most purchases has been queried in the output.

Dropping the database

```
hive> drop database case_study;
OK
Time taken: 0.438 seconds
```

Terminating the cluster

▶	Hive-case-study	j-ZX1T6R262DIK	Terminated User request	2020-10-11 20:18 (UTC+5:30)
---	---------------------------------	----------------	----------------------------	-----------------------------

