**Partitioning:**

Partitioning is used to slice the data horizontally over the entire range or on a smaller range of values using one or more column . Partition concept is well known in RDBMS as well, so if you know the database then you should be aware of the concept and if not then also its not an issue as you will be well aware of the concept after reading this article. e.g if you are having hundred records in a table with student data in an university and you want to divide the entire record into Male and Female student so here it is nothing but partitioning and GENDER is my column which is used to split the records. Again, as I told, there can be more than one column used to split the records.

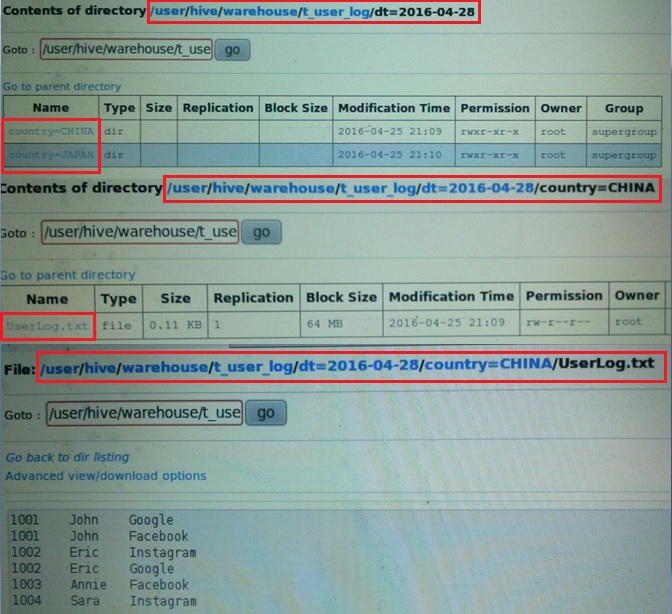
In real word scenario, if you want to analyze the log files of user activity on internet, it would be great if we can keep the data as per date and geographies. **There are two types of partitioning in HIVE:                                     1.Static Partitioning        2.Dynamic Partitioning**The table DDL statement will be same in case of both the partitioning as shown below:I have created a table T\_USER\_LOG with DT and COUNTRY column as my partitioning column. I have use hive script mode where "HivePartition.hql' is my script file. You can use hive shell command as well or whichever is feasible for you. As highlighted in the image, the partition columns appears in table schema like normal table column.

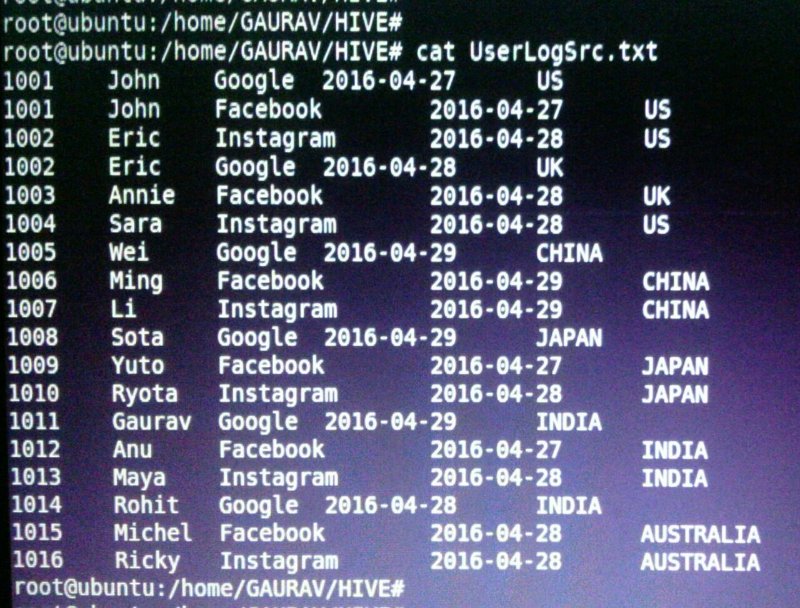
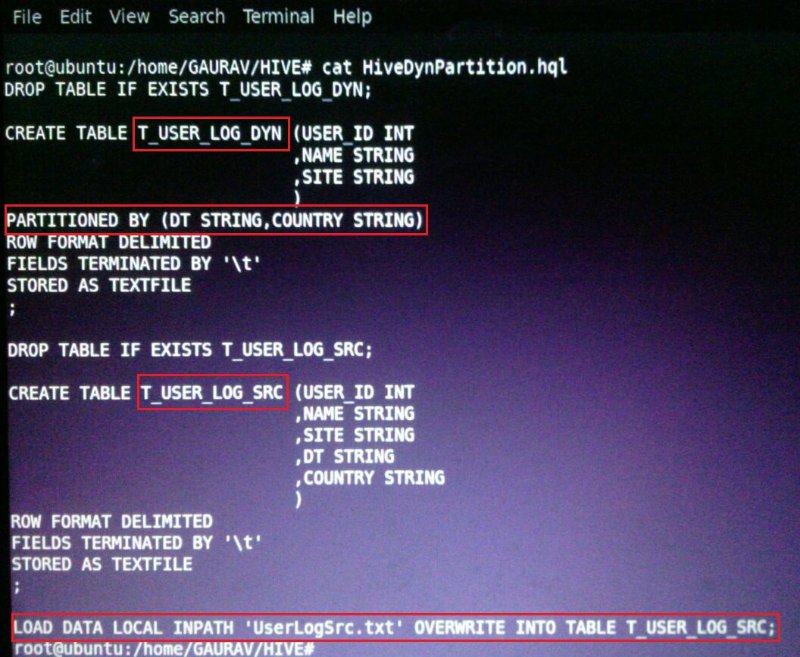
**Static Partitioning:**In Static Partition, we know the partition column before itself. So far so good, now when we load data there it makes the difference.

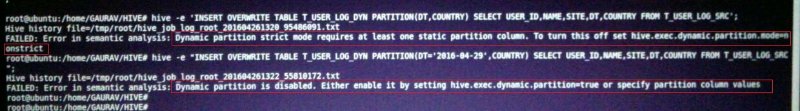
LOAD DATA LOCAL INPATH [path\_name] OVERWRITE INTO TABLE [table\_name] PARTITION(partition\_column='value'....). Here we have to give the partition column value explicitly whenever we want to create new partition as shown below:

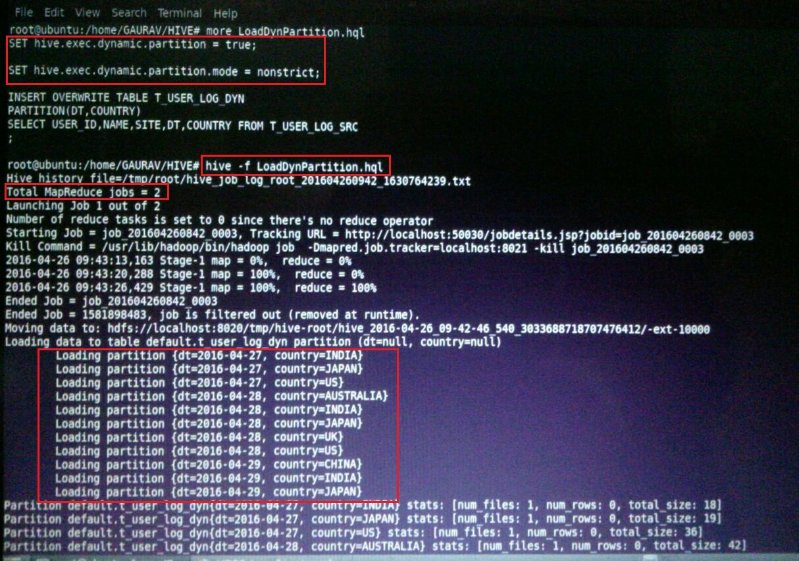
**LOAD SCRIPTS.**

As you can see, we have mentioned the partition column value in each load.

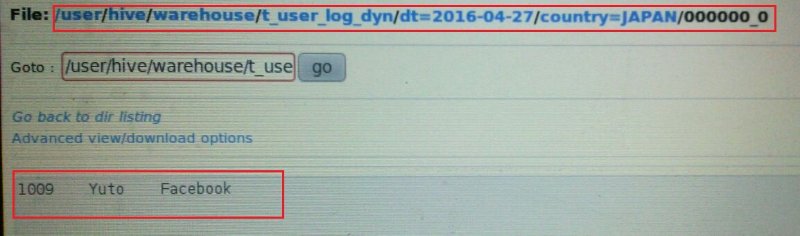
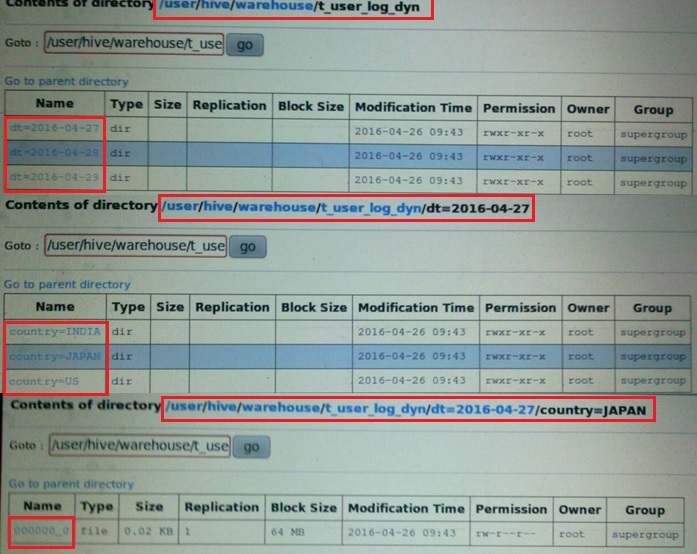
**OUTPUT:**Since our table T\_USER\_LOG is managed table so the data is loaded in hive warehouse path i.e **/user/hive/warehouse/t\_user\_log. **Hereyou can check all other partition as well, it will have the file UserLog.txt. Total two level of partitioning is there in our example, one as **DT** and another as **COUNTRY** , then the final data will be stored inside. All partitions in hive is there as directories. **Loading in hive is instantaneous process and it won't trigger a Map/Reduce job.** That's why our file is stored as **UserLog.txt**instead of 00000\_o file. Please follow the article as I will show in dynamic partition where we will LOAD table using another table where Map/reduce job is triggered.

**DYNAMIC PARTITIONING:**Let us see now the load script of Dynamic Partitioning. We will create new table T\_USER\_LOG\_DYN for dynamic partition and also as we told earlier that we will load this table using a new table, let's create another table T\_USER\_LOG\_SRC. Below is the data of sourse table:Let's us check the hive script for table DDL**.**We will see first the negative scenario and then the final load scripts. We have to set two hive properties as below:

**1.SET hive.exec.dynamic.partition= true;                                                          2.SET hive.exec.dynamic.partition.mode= nonstrict**Sorry for the small screenshot but it show the error if you try to load the table and achieve partitioning dynamically without setting above two properties. Here while loading the table, the partition will be created dynamically on all partition columns if **hive.exec.dynamic.partition.mode= nonstrict**is set. If it is strict [which is by default], it will need at least one partition column to be defined in load script. Now let see the final load script as below:

**LOAD SCRIPT:**Below is the screenshot for loading the table T\_USER\_LOG\_DYN using the data from T\_USER\_LOG\_SRC and creating dynamic partitions. **LoadDynPartition.hql** is my script to load the table dynamically as shown below:

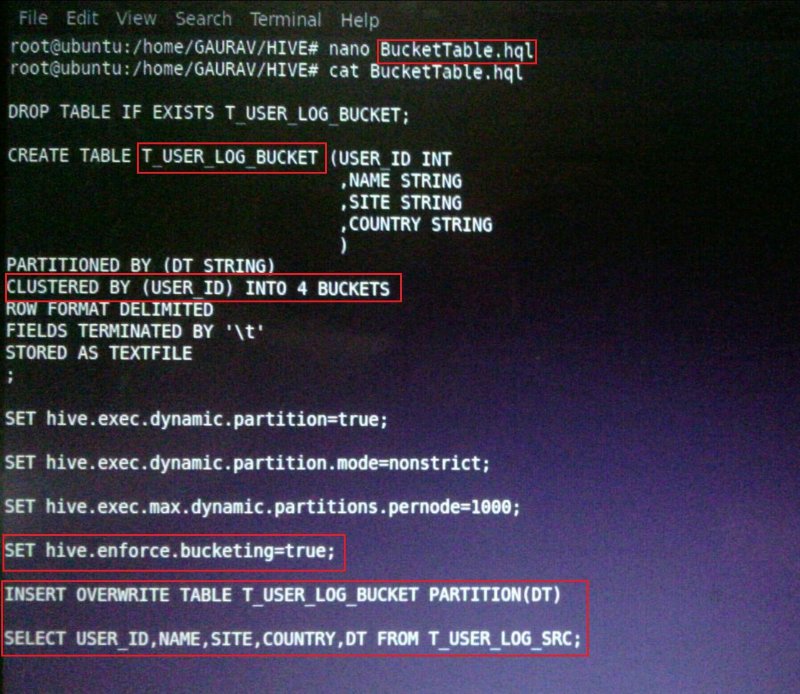
**hive -f LoadDynPartition.hql** is used to execute the hive script.

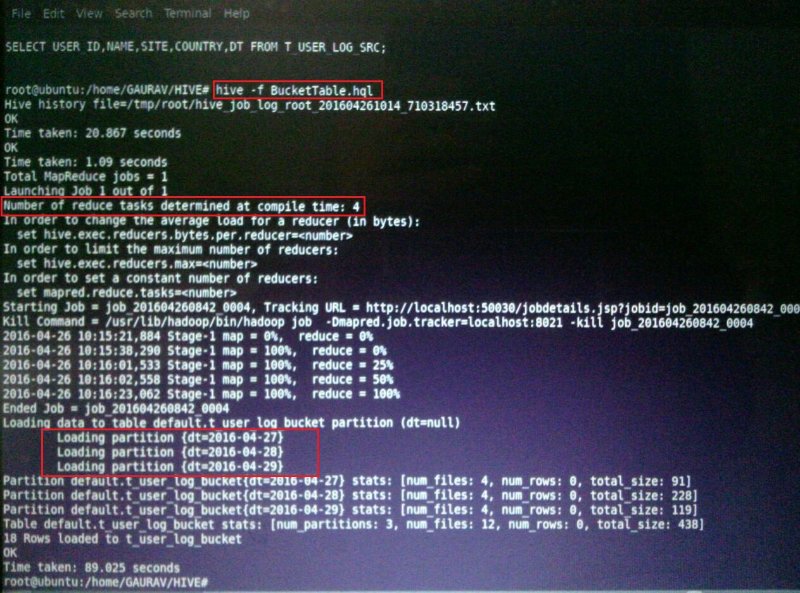
**OUTPUT:** Once the script is run, the file will be created as below:As you can see, here we need not to load the table multiple time to create the multiple partition as it was the case in static partitioning. Here all the partitions are created in T\_USER\_LOG\_DYN based on number of distinct DT and COUNTRY column values in T\_USER\_LOG\_SRC table dynamically at run time. Also we told earlier, since the data is loaded using another table and map/reduce job is executed by framework, output file is as 000000\_0 but not by any name.

***Advantage of Partitioning:****Partitioning has its own benefit when it comes to its usage in HIVE. Its helps to organize the data in logical fashion and when we query the partitioned table using partition column, it allows hive to skip all but relevant sub-directories and files. This can lead to tremendous reduction in data required to read and filter in the initial map stage. This reduces the number of mapper, I/O operation and response time for the query.*

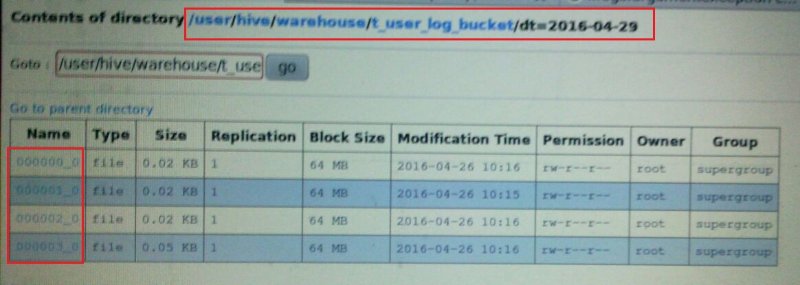
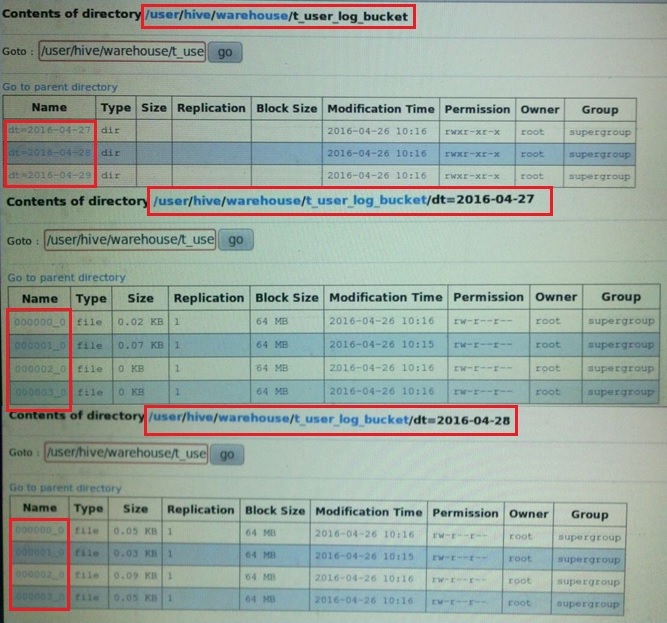
**BUCKETING in HIVE**: When we write data in bucketed table in hive, it places the data in distinct buckets as files. Hive uses some hashing algorithm to generate a number in range of **1 to N** buckets [as mentioned in DDL] and based on the result of hashing, data is placed in a particular buckets as a file. Let's create a hive bucketed table T\_USER\_LOG\_BUCKET with a partition column as **DT and having 4 buckets**. We specify bucketing column in CLUSTERED BY (column\_name) clause in hive table DDL as shown below in hive script files:

**LOAD SCRIPT:**

****Below screenshots shows the hive script execution to create and load Bucketed table:

****As we know that the number of reduce task determine the number of output file so here we have defined table as four buckets so the number of reduce taks is 4 as highlighted in above screenshot.

**OUTPUT**: Here the output is partitioned on **DT** column and each partition will contain the four buckets as files. So all three partitions will have four files each as shown in below screenshots

****

***Advantage of Bucketing:***

***Sampling****: When we want to test a table which has huge amount of data or when we want to draw some patterns or when we want some aggregations [where accuracy is not out top priority] then we need to sample i.e we need to run the query on smaller set of data of the entire table data with evenly distributed sample. There are two types of sampling:*

***1.Bucket Sampling : e.g****SELECT \* FROM T\_USER\_LOG\_BUCKET TABLESAMPLE (BUCKET 1 OUT OF 4 AT USED\_ID).... It will select the data from the first buckets of each partition from T\_USER\_LOG\_BUCKET table otherwise in normal select we can't select data in such distributed and evenly manner.*