***HIVE:***

**Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.**

***SCRIPT:***

**Hive scripts are used to execute a set of Hive commands collectively. This is used for reducing time and effort invested in writing and executing each command manually.**

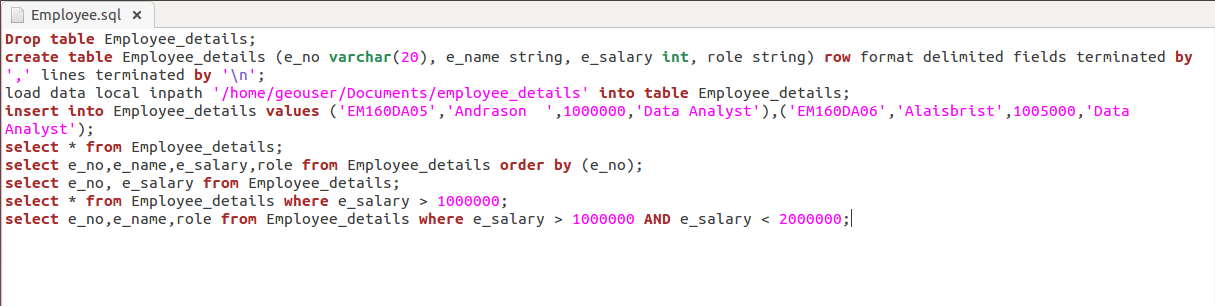
***Creating a Hive Script and run multiple commands***

***Step-1***

**First Open the notepad or any text editor**

***Step-2***

**Writing the script (Multiple commands) and save it as .sql file**

**Save the following script as named Employee.sql  
**

***Step-3***

**Open a terminal**

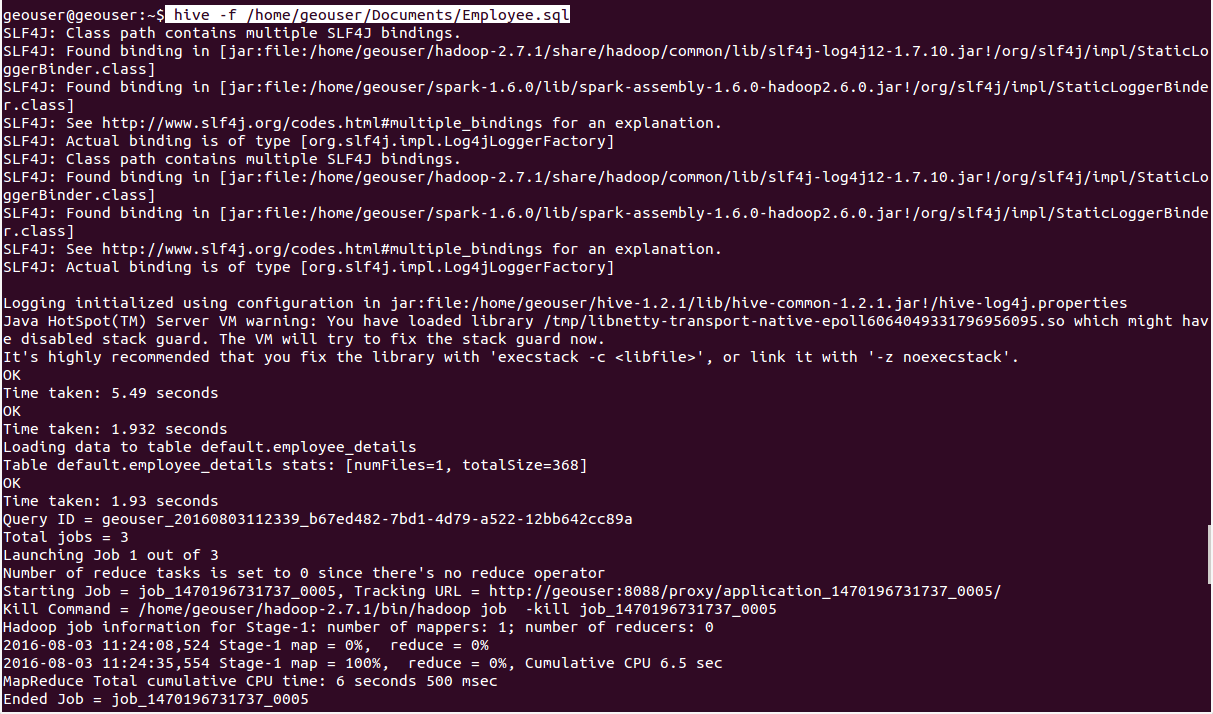
**ctrl+Alt+t**

***Step-4***

**Run the script**

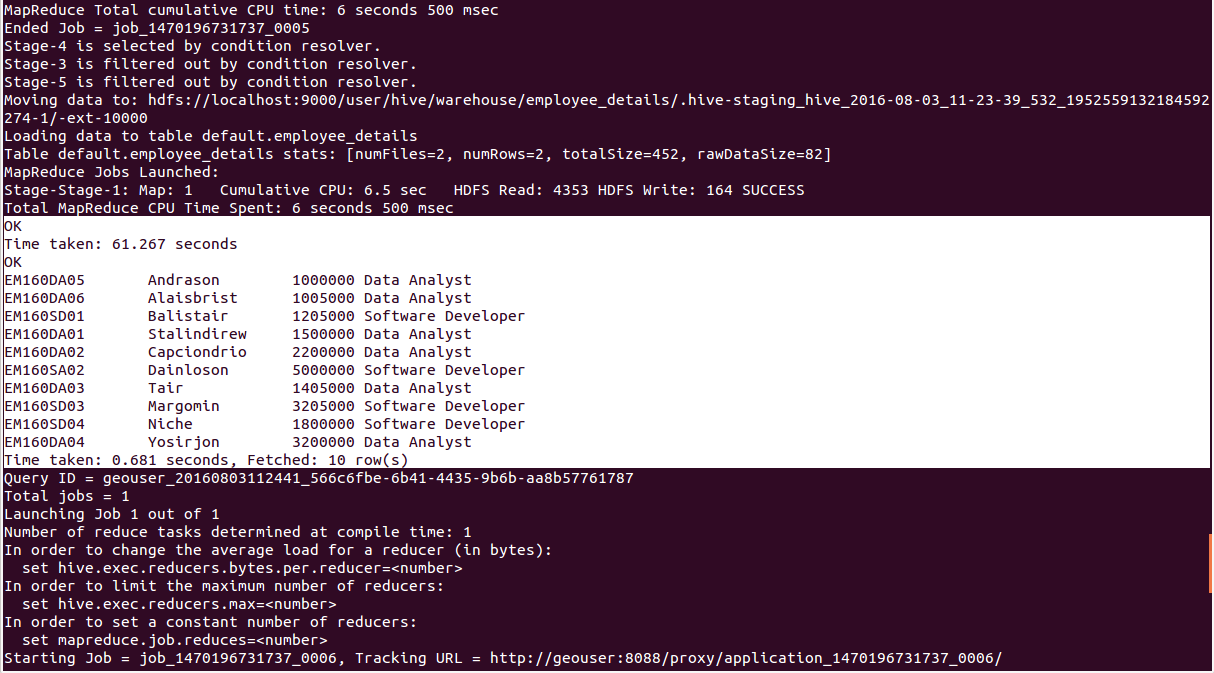
**hive -f <path of the script file>**

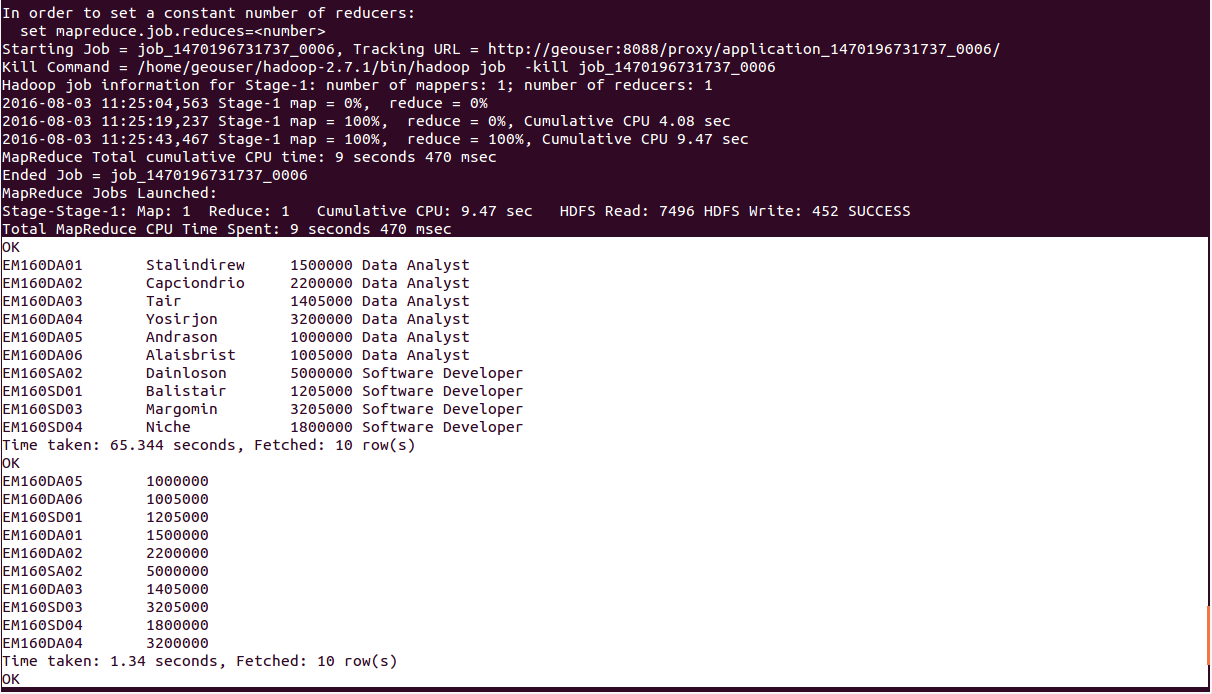
**hive -f /home/geouser/Documents/Employee.sql**

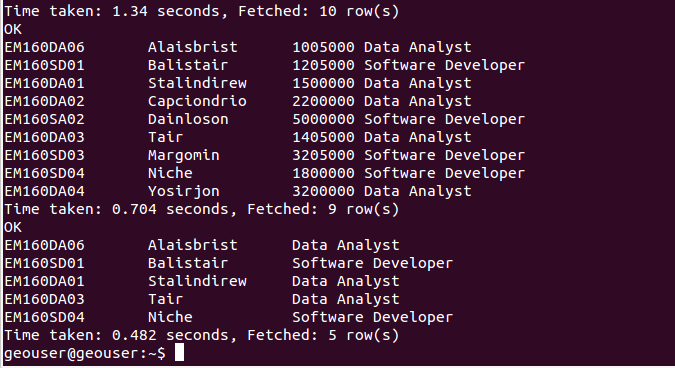


***Step-5***

**Output**







***Description about commands:***

**Drop table Employee\_details;**

**-> Drop the table if already exist,**

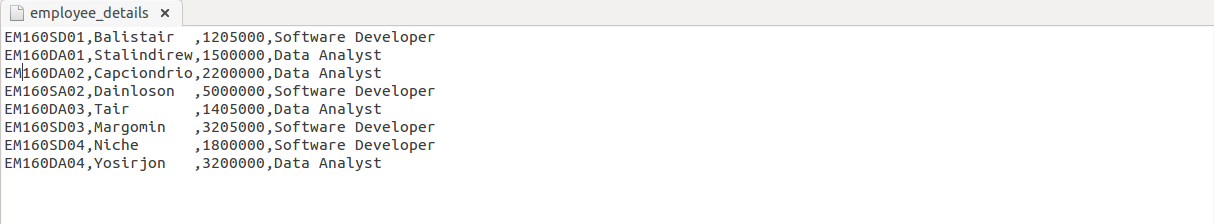
**create table Employee\_details (e\_no varchar(20), e\_name string, e\_salary int, role string) row format delimited fields terminated by ‘,’ lines terminated by ‘\n’;**

**-> Create a table named as Employee\_details with 4 columns (e\_no,e\_name,e\_salary,role)**

**load data local inpath ‘/home/geouser/Documents/employee\_details’ into table Employee\_details;**

**-> Load the datas from local file (employee\_details) to Hive table (Employee\_details)**

**employee\_details (data stored in local)**

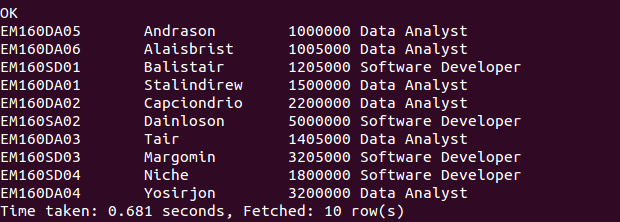


**insert into Employee\_details values (‘EM160DA05’,’Andrason ‘,1000000,’Data Analyst’),(‘EM160DA06′,’Alaisbrist’,1005000,’Data Analyst’);**

**-> Insert the value to the Employee\_details table using Insert command**

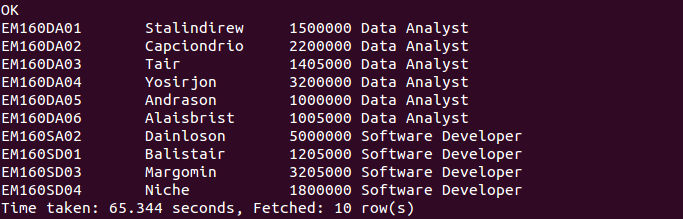
**select \* from Employee\_details;**

**-> Select the all data from the Employee\_details table**



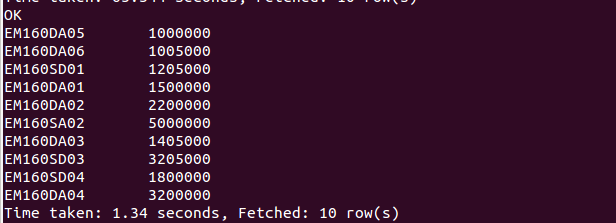
**select e\_no,e\_name,e\_salary,role from Employee\_details order by (e\_no);**

**-> Select all the data from the Employee\_details table with order**



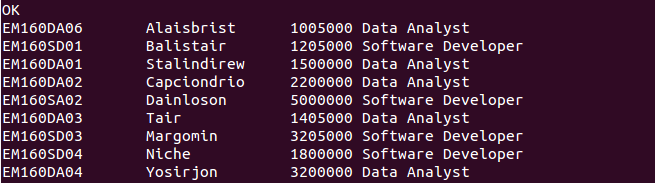
**select e\_no, e\_salary from Employee\_details;**

**-> Select specific columns from the Employee\_details table**



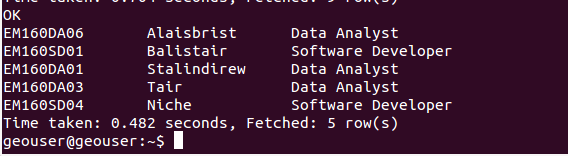
**select \* from Employee\_details where e\_salary > 1000000;**

**-> Select the data based on the condition from the table Employee\_detail**



**select e\_no,e\_name,role from Employee\_details where e\_salary > 1000000 AND e\_salary < 2000000;**

**-> Select the data based on the condition from the Employee\_details table**



***PARTITION AND BUCKETING:***

***HIVE:***

**Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. Hive is good for performing queries on large datasets.**

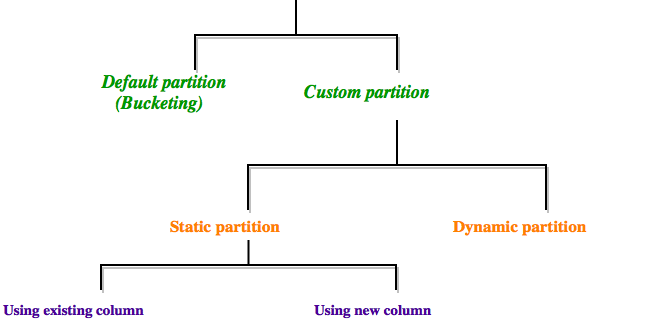
**In Hive one of the main concept is partitioning.**

## **Why we use Partition:**

**A simple query in Hive reads the entire dataset even if we have where clause filter. This becomes a bottleneck for running MapReduce jobs over a large table. We can overcome this issue by implementing partitions in Hive. Hive makes it very easy to implement partitions by using the automatic partition scheme when the table is created.**

***Types of Partition:***

***HIVE Partition***



***Custom Partition:***

### **Static Partition:**

**1.The columns whose values are known at COMPILE TIME (given by user).**

**2.Insert input data files individually into a partition table.**

**3.Usually when loading files (big files) into Hive tables static partitions are preferred.**

**4.Static Partition saves your time in loading data compared to dynamic partition.**

**5.Use where clause to use limit in static partition.**

**6.We can alter the partition in static partition**

**7.Perform Static partition on Hive Manage table or external table.**

**In Static Partition there is a two types.**

***1.Using existing column,***

***2.Using new column.***

***Create a table in Hive:***

###### **Step-1: Create a Hive table**

### create table patient(patient\_id int, patient\_name string, drug string, gender string, total\_amount int) row format delimited fields terminated by ',' stored as textfile;

###### **Step-2: Load value into the Hive table**

# load data local inpath '/home/geouser/Documents/patient' into table patient;

### **Data Set:(Patient)**

**10013201,alaister,para,male,500**

**10013202,briito,metacin,male,800**

**10013203,anifa,crocin,female,600**

**10013204,mohammed,para,male,999**

**10013304,piyush,metacin,male,400**

**10013404,manisha,para,female,200**

**10011204,vijay,crocin,male,670**

**10013224,marsinga,crocin,female,570**

**hive> create table patient(patient\_id int, patient\_name string, drug string, gender string, total\_amount int) row format delimited fields terminated by ‘,’ stored as textfile;  
OK  
Time taken: 0.541 seconds  
hive> load data local inpath ‘/home/geouser/Documents/patient’ into table patient;  
Loading data to table default.patient  
Table default.patient stats: [numFiles=1, totalSize=232]  
OK  
Time taken: 1.055 seconds**

#### **1.Using existing column:**

**In this type of partition, we use the partition column which is already present in the table.**

***Example:***

###### **Step-1: Create a Hive partition table**

### create table p\_patient1(patient\_id int, patient\_name string, gender string, total\_amount int) partitioned by ( drug string);

**hive> create table p\_patient1(patient\_id int, patient\_name string, gender string, total\_amount int) partitioned by ( drug string);  
OK  
Time taken: 0.235 seconds**

###### **Step-2: Insert value into the Partitioned table**

### insert overwrite table p\_patient1 partition(drug='metacin') select patient\_id, patient\_name, gender, total\_amount from patient where drug='metacin';

**hive> insert overwrite table p\_patient1 partition(drug=’metacin’) select patient\_id, patient\_name, gender, total\_amount from patient where drug=’metacin’;  
Query ID = geouser\_20160824111554\_6603e3bf-9b2d-4e57-8ce7-78bb33068a7b  
Total jobs = 3  
Launching Job 1 out of 3  
Number of reduce tasks is set to 0 since there’s no reduce operator  
Starting Job = job\_1472015672745\_0005, Tracking URL = http://geouser:8088/proxy/application\_1472015672745\_0005/  
Kill Command = /home/geouser/hadoop-2.7.1/bin/hadoop job  -kill job\_1472015672745\_0005  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0  
2016-08-24 11:16:19,439 Stage-1 map = 0%,  reduce = 0%  
2016-08-24 11:16:34,999 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.89 sec  
MapReduce Total cumulative CPU time: 5 seconds 890 msec  
Ended Job = job\_1472015672745\_0005  
Stage-4 is selected by condition resolver.  
Stage-3 is filtered out by condition resolver.  
Stage-5 is filtered out by condition resolver.  
Moving data to: hdfs://localhost:9000/user/hive/warehouse/p\_patient1/drug=metacin/.hive-staging\_hive\_2016-08-24\_11-15-54\_037\_2017514360714144410-1/-ext-10000  
Loading data to table default.p\_patient1 partition (drug=metacin)  
Partition default.p\_patient1{drug=metacin} stats: [numFiles=1, numRows=2, totalSize=58, rawDataSize=56]  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1   Cumulative CPU: 5.89 sec   HDFS Read: 4644 HDFS Write: 145 SUCCESS  
Total MapReduce CPU Time Spent: 5 seconds 890 msec  
OK  
Time taken: 45.299 seconds**

###### **Step-3: View the Partitione value**

### select \* from p\_patient1;

**hive> select \* from p\_patient1;  
OK  
10013202    briito      male      800    metacin  
10013304    piyush      male      400    metacin  
Time taken: 0.147 seconds, Fetched: 2 row(s)**

#### **2.Using new column:**

**In this type of partition, we use new partition column**

***Example:***

###### **Step-1: Create a Partitioned Hive table**

### create table p\_patient(patient\_id int, patient\_name string, drug string, gender string, total\_amount int) partitioned by (new string);

**hive>  create table p\_patient(patient\_id int, patient\_name string, drug string, gender string, total\_amount int) partitioned by (new string);  
OK  
Time taken: 0.232 seconds**

###### **Step-2: Insert the value into the table**

### insert overwrite table p\_patient partition(new='metacin') select \* from patient where drug='metacin';

**hive> insert overwrite table p\_patient partition(new=’metacin’) select \* from patient where drug=’metacin’;  
Query ID = geouser\_20160824111010\_a72a7634-5ef5-4e15-a3a1-48046ab5e569  
Total jobs = 3  
Launching Job 1 out of 3  
Number of reduce tasks is set to 0 since there’s no reduce operator  
Starting Job = job\_1472015672745\_0004, Tracking URL = http://geouser:8088/proxy/application\_1472015672745\_0004/  
Kill Command = /home/geouser/hadoop-2.7.1/bin/hadoop job  -kill job\_1472015672745\_0004  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0  
2016-08-24 11:10:32,722 Stage-1 map = 0%,  reduce = 0%  
2016-08-24 11:10:47,186 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.0 sec  
MapReduce Total cumulative CPU time: 6 seconds 0 msec  
Ended Job = job\_1472015672745\_0004  
Stage-4 is selected by condition resolver.  
Stage-3 is filtered out by condition resolver.  
Stage-5 is filtered out by condition resolver.  
Moving data to: hdfs://localhost:9000/user/hive/warehouse/p\_patient/new=metacin/.hive-staging\_hive\_2016-08-24\_11-10-10\_899\_701911370209935060-1/-ext-10000  
Loading data to table default.p\_patient partition (new=metacin)  
Partition default.p\_patient{new=metacin} stats: [numFiles=1, numRows=2, totalSize=74, rawDataSize=72]  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1   Cumulative CPU: 6.0 sec   HDFS Read: 4671 HDFS Write: 159 SUCCESS  
Total MapReduce CPU Time Spent: 6 seconds 0 msec  
OK  
Time taken: 40.037 seconds**

###### **Step-3: View the partitioned value**

**select \* from p\_patient;**

**hive> select \* from p\_patient;  
OK  
10013202    briito        metacin    male      800    metacin  
10013304    piyush      metacin    male      400    metacin  
Time taken: 0.298 seconds, Fetched: 2 row(s)**

### **Dynamic Partition:**

**1.columns whose values are only known at EXECUTION TIME.**

**2.We use dynamic partition while loading from an existing table that is not partitioned.**

**3.We use dynamic partition while unknown values for partition columns.**

**4.Usually dynamic partition load the data from non partitioned table.**

**5.Dynamic Partition takes more time in loading data compared to static partition.**

**6.There is no required where clause to use limit.**

**7.We can’t perform alter on Dynamic partition.**

**8.Perform dynamic partition on hive external table and managed table.**

***In dynamic partition the partitioned column of the partitioned hive table is must present in the last column of the existing hive table.***

***Example:***

###### **Step-1: Create a hive table**

### create table patient1(patient\_id int, patient\_name string, gender string, total\_amount int, drug string) row format delimited fields terminated by ',' stored as textfile;

###### **Step-2: Load data into the hive table**

### load data local inpath '/home/geouser/Documents/patient1' into table patient1;

### **Data Set: (Patient1)**

**10013201,alaister,male,500,para**

**10013202,briito,male,800,metacin**

**10013203,anifa,female,600,crocin**

**10013204,mohammed,male,999,para**

**10013304,piyush,male,400,metacin**

**10013404,manisha,female,200,para**

**10011204,vijay,male,670,crocin**

**10013224,marsinga,female,570,crocin**

**hive> create table patient1(patient\_id int, patient\_name string, gender string, total\_amount int, drug string) row format delimited fields terminated by ‘,’ stored as textfile;  
OK  
Time taken: 0.972 seconds  
hive>  load data local inpath ‘/home/geouser/Documents/patient1’ into table patient1;  
Loading data to table default.patient1  
Table default.patient1 stats: [numFiles=1, totalSize=282]  
OK  
Time taken: 2.338 seconds**

###### **Step-3: Before creating Partitioned table in hive first we set the properties for dynamic partition**

### SET hive.exec.dynamic.partition=true;

### SET hive.exec.dynamic.partition.mode=non-strict;

###### **Step-4: Create a partitioned table in hive**

### create table dynamic\_partition\_patient (patient\_id int,patient\_name string, gender string, total\_amount int) partitioned by (drug string);

**hive> create table dynamic\_partition\_patient (patient\_id int,patient\_name string, gender string, total\_amount int) partitioned by (drug string);  
OK  
Time taken: 0.348 seconds**

###### **Step-5: Insert value into the partitioned table**

### insert into table dynamic\_partition\_patient PARTITION(drug) select \* from patient1;

**hive> insert into table dynamic\_partition\_patient PARTITION(drug) select \* from patient1;  
Query ID = geouser\_20160824121550\_09ad8cb7-121d-4df6-aa07-620fa1a04269  
Total jobs = 3  
Launching Job 1 out of 3  
Number of reduce tasks is set to 0 since there’s no reduce operator  
Starting Job = job\_1472015672745\_0008, Tracking URL = http://geouser:8088/proxy/application\_1472015672745\_0008/  
Kill Command = /home/geouser/hadoop-2.7.1/bin/hadoop job  -kill job\_1472015672745\_0008  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0  
2016-08-24 12:16:32,948 Stage-1 map = 0%,  reduce = 0%  
2016-08-24 12:16:59,856 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.56 sec  
MapReduce Total cumulative CPU time: 4 seconds 560 msec  
Ended Job = job\_1472015672745\_0008  
Stage-4 is selected by condition resolver.  
Stage-3 is filtered out by condition resolver.  
Stage-5 is filtered out by condition resolver.  
Moving data to: hdfs://localhost:9000/user/hive/warehouse/dynamic\_partition\_patient/.hive-staging\_hive\_2016-08-24\_12-15-50\_006\_735085815761966908-1/-ext-10000  
Loading data to table default.dynamic\_partition\_patient partition (drug=null)  
Time taken for load dynamic partitions : 1856  
Loading partition {drug=crocin}  
Loading partition {drug=metacin}  
Loading partition {drug=para}  
Time taken for adding to write entity : 36  
Partition default.dynamic\_partition\_patient{drug=crocin} stats: [numFiles=1, numRows=3, totalSize=85, rawDataSize=82]  
Partition default.dynamic\_partition\_patient{drug=metacin} stats: [numFiles=1, numRows=2, totalSize=58, rawDataSize=56]  
Partition default.dynamic\_partition\_patient{drug=para} stats: [numFiles=1, numRows=3, totalSize=87, rawDataSize=84]  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1   Cumulative CPU: 4.56 sec   HDFS Read: 4405 HDFS Write: 458 SUCCESS  
Total MapReduce CPU Time Spent: 4 seconds 560 msec  
OK  
Time taken: 77.941 seconds**

###### **Step-6: View the value of the Partitioned table using hadoop command**

**(i). List the files in the partitioned table**

**hadoop fs -ls /user/hive/warehouse/dynamic\_partition\_patient**

**geouser@geouser:~$ hadoop fs -ls /user/hive/warehouse/dynamic\_partition\_patient  
16/08/25 12:16:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform… using builtin-java classes where applicable  
Found 3 items  
drwxr-xr-x   – geouser supergroup          0 2016-08-24 12:17 /user/hive/warehouse/dynamic\_partition\_patient/drug=crocin  
drwxr-xr-x   – geouser supergroup          0 2016-08-24 12:17 /user/hive/warehouse/dynamic\_partition\_patient/drug=metacin  
drwxr-xr-x   – geouser supergroup          0 2016-08-24 12:17 /user/hive/warehouse/dynamic\_partition\_patient/drug=para**

**(ii). View the value of the files using cat command**

**hadoop fs -cat /user/hive/warehouse/dynamic\_partition\_patient/drug=crocin/000000\_0**

**hadoop fs -cat /user/hive/warehouse/dynamic\_partition\_patient/drug=para/000000\_0**

**hadoop fs -cat /user/hive/warehouse/dynamic\_partition\_patient/drug=metacin/000000\_0**

**geouser@geouser:~$ hadoop fs -cat /user/hive/warehouse/dynamic\_partition\_patient/drug=crocin/000000\_0  
16/08/25 12:22:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform… using builtin-java classes where applicable  
10013203anifa   female600  
10011204vijay   male670  
10013224marsingafemale570  
geouser@geouser:~$ hadoop fs -cat /user/hive/warehouse/dynamic\_partition\_patient/drug=para/000000\_0  
16/08/25 12:23:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform… using builtin-java classes where applicable  
10013201alaistermale  500  
10013204mohammedmale  999  
10013404manisha female200  
geouser@geouser:~$ hadoop fs -cat /user/hive/warehouse/dynamic\_partition\_patient/drug=metacin/000000\_0  
16/08/25 12:23:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform… using builtin-java classes where applicable  
10013202briito  male  800  
10013304piyush  male  400**

***Default Partition:***

**Default Partition in Hive is called as Bucketing.**

**Usually Partitioning in Hive offers a way of segregating hive table data into multiple files/directories. But partitioning gives effective results when,**

**1.There are limited number of partitions,**

**2.Comparatively equal sized partitions.**

**But this may not possible in all scenarios, like when are partitioning our tables based geographic locations like country, some bigger countries will have large partitions where as small countries data will create small partitions . So, In these cases Partitioning will not be ideal.**

**To overcome the problem of over partitioning, Hive provides Bucketing concept, another technique for decomposing table data sets into more manageable parts.**

### ***Bucketing:***

**1.Bucketing concept is based on (hashing function on the bucketed column)mod(by total number of buckets). The hash\_function depends on the type of the bucketing column.**

**2.Records with the same bucketed column will always be stored in the same bucket.**

**3.We use CLUSTERED BY clause to divide the table into buckets.**

**4.Physically, each bucket is just a file in the table directory, and Bucket numbering is 1-based.**

**5.Bucketing can be done along with Partitioning on Hive tables and even without partitioning.**

**6.Bucketed tables will create almost equally distributed data file parts.**

***Example:***

###### **Step-1: Create a Hive table**

### create table patient(patient\_id int, patient\_name string, drug string, gender string, total\_amount int) row format delimited fields terminated by ',' stored as textfile;

###### **Step-2: Load value into the Hive table**

**load data local inpath '/home/geouser/Documents/patient' into table patient;**

###### **Step-3: First set the property before create bucketing table in hive**

**set hive.enforce.bucketing =true;**

###### **Step-4: Create a bucketing table in Hive**

### create table bucket\_patient(patient\_id int, patient\_name string, drug string,gender string, total\_amount int) clustered by (drug) into 4 buckets;

**hive> create table bucket\_patient(patient\_id int, patient\_name string, drug string, gender string, total\_amount int) clustered by (drug) into 4 buckets;  
OK  
Time taken: 1.283 seconds**

###### **Step-5: Insert the value into the bucketing table**

### insert overwrite table bucket\_patient select \* from patient;

**hive> insert overwrite table bucket\_patient select \* from patient;  
Query ID = geouser\_20160824112021\_33ee4b7d-d424-4662-9061-bcefeebd05d2  
Total jobs = 1  
Launching Job 1 out of 1  
Number of reduce tasks determined at compile time: 4  
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\_1472015672745\_0006, Tracking URL = http://geouser:8088/proxy/application\_1472015672745\_0006/  
Kill Command = /home/geouser/hadoop-2.7.1/bin/hadoop job  -kill job\_1472015672745\_0006  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 4  
2016-08-24 11:20:55,084 Stage-1 map = 0%,  reduce = 0%  
2016-08-24 11:21:08,616 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.66 sec  
2016-08-24 11:21:51,562 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 9.2 sec  
2016-08-24 11:22:24,830 Stage-1 map = 100%,  reduce = 89%, Cumulative CPU 18.69 sec  
2016-08-24 11:22:29,080 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 26.08 sec  
MapReduce Total cumulative CPU time: 26 seconds 80 msec  
Ended Job = job\_1472015672745\_0006  
Loading data to table default.bucket\_patient  
Table default.bucket\_patient stats: [numFiles=4, numRows=8, totalSize=294, rawDataSize=286]  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1  Reduce: 4   Cumulative CPU: 29.67 sec   HDFS Read: 18822 HDFS Write: 547 SUCCESS  
Total MapReduce CPU Time Spent: 29 seconds 670 msec  
OK  
Time taken: 147.019 seconds**

###### **Step-6: View the value of first bucket in the bucketing table**

### select \* from bucket\_patient TABLESAMPLE(BUCKET 1 OUT OF 4 ON drug);

**hive> select \* from bucket\_patient TABLESAMPLE(BUCKET 1 OUT OF 4 ON drug);  
OK  
10013224    marsinga    crocin     female    570  
10011204    vijay       crocin     male    670  
10013404    manisha     para       female    200  
10013204    mohammed    para       male      999  
10013203    anifa       crocin     female    600  
10013201    alaister    para       male      500  
Time taken: 0.196 seconds, Fetched: 6 row(s)**

###### **Step-7: View the 10% of value in the bucketing table**

### select \* from bucket\_patient TABLESAMPLE(10 percent);

**hive> select \* from bucket\_patient TABLESAMPLE(10 percent);  
OK  
10013224    marsinga    crocin     female    570  
Time taken: 0.208 seconds, Fetched: 1 row(s)**

###### **Step-8: View the value in limit 5**

### select \* from bucket\_patient LIMIT 5;

**hive> select \* from bucket\_patient LIMIT 5;  
OK  
10013224    marsinga    crocin     female    570  
10011204    vijay       crocin     male    670  
10013404    manisha     para       female    200  
10013204    mohammed    para       male      999  
10013203    anifa       crocin     female    600  
Time taken: 0.155 seconds, Fetched: 5 row(s)**

***Apache Hive Dynamic Partition  table***

**Difference between Static and Dynamic partition :**

**Static Partition  columns: in DML/DDL involving multiple partitioning columns, the columns whose values are known at COMPILE TIME (given by user).**

**Dynamic Partition columns: columns whose values are only known at EXECUTION TIME.**

**To enable dynamic partitioning in hive table , set these following hive properties in hive prompt**

**SET hive.exec.dynamic.partition=true;  
SET hive.exec.dynamic.partition.mode=non-strict;**

**By default , hive partition is static**

**Table creation :**

**#Creating hive table using single dynamic column . e.g  year**

**create table hive\_partition(id int, name string, salary float, location string) partitioned by (year int) row format delimited fields terminated by ‘,’ lines terminated by ‘\n’;**

**#Creating hive table using multiple dynamic partition column.  e.g  year,location**

**create table hive\_partition(id int, name string, salary float) partitioned by (year int , location string) row format delimited fields terminated by ‘,’ lines terminated by ‘\n’;**

**#Creating hive table using  static partition and Dynamic partition column. e.g year  as static partition column ,location as dynamic column**

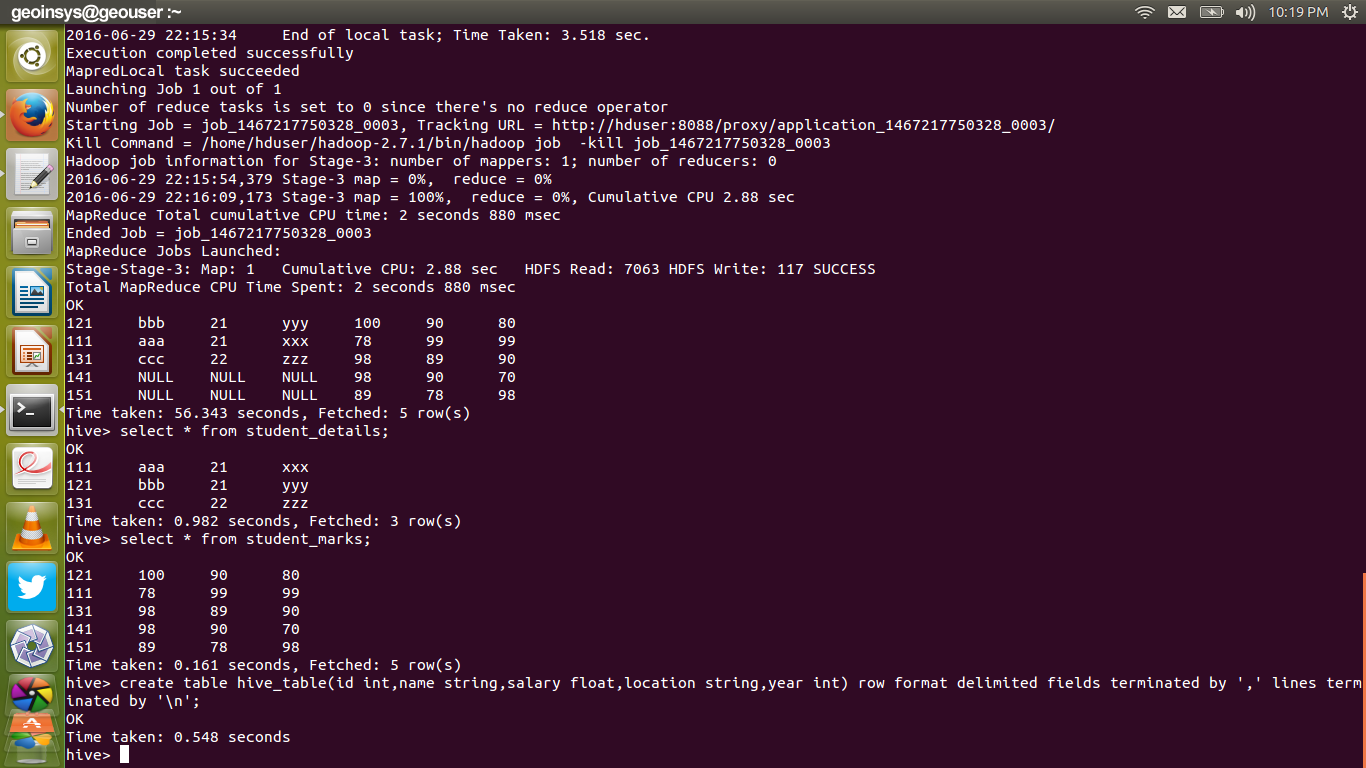
**create table hive\_partition(id int, name string, salary float) partitioned by (year int , location string) row format delimited fields terminated by ‘,’ lines terminated by ‘\n’;**

**#Note : Dont use dynamic partition column before static partition Column .**



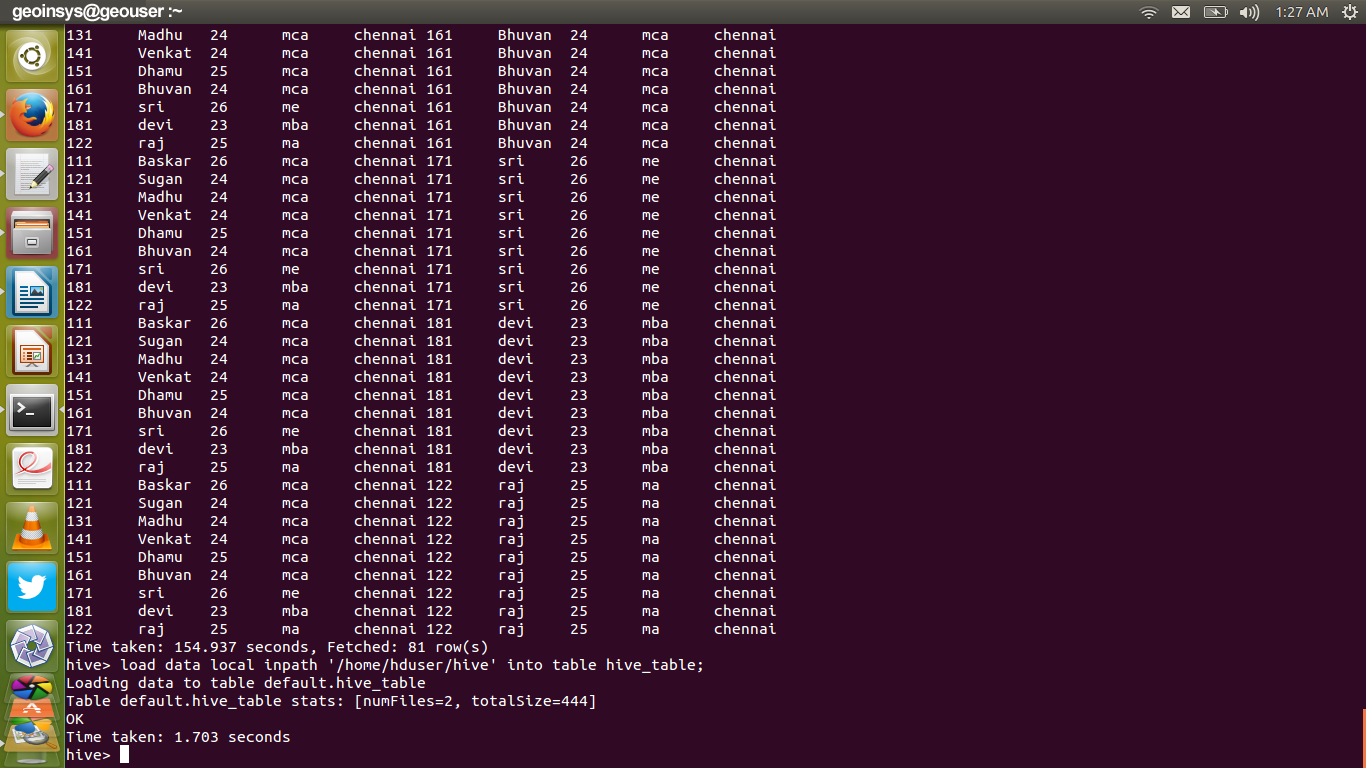
**# Creating Source table , non-partitioned table**

**create table hive\_table(id int,name string,salary float,location string,year int) row format delimited fields terminated by ‘,’ lines terminated by ‘\n’;**

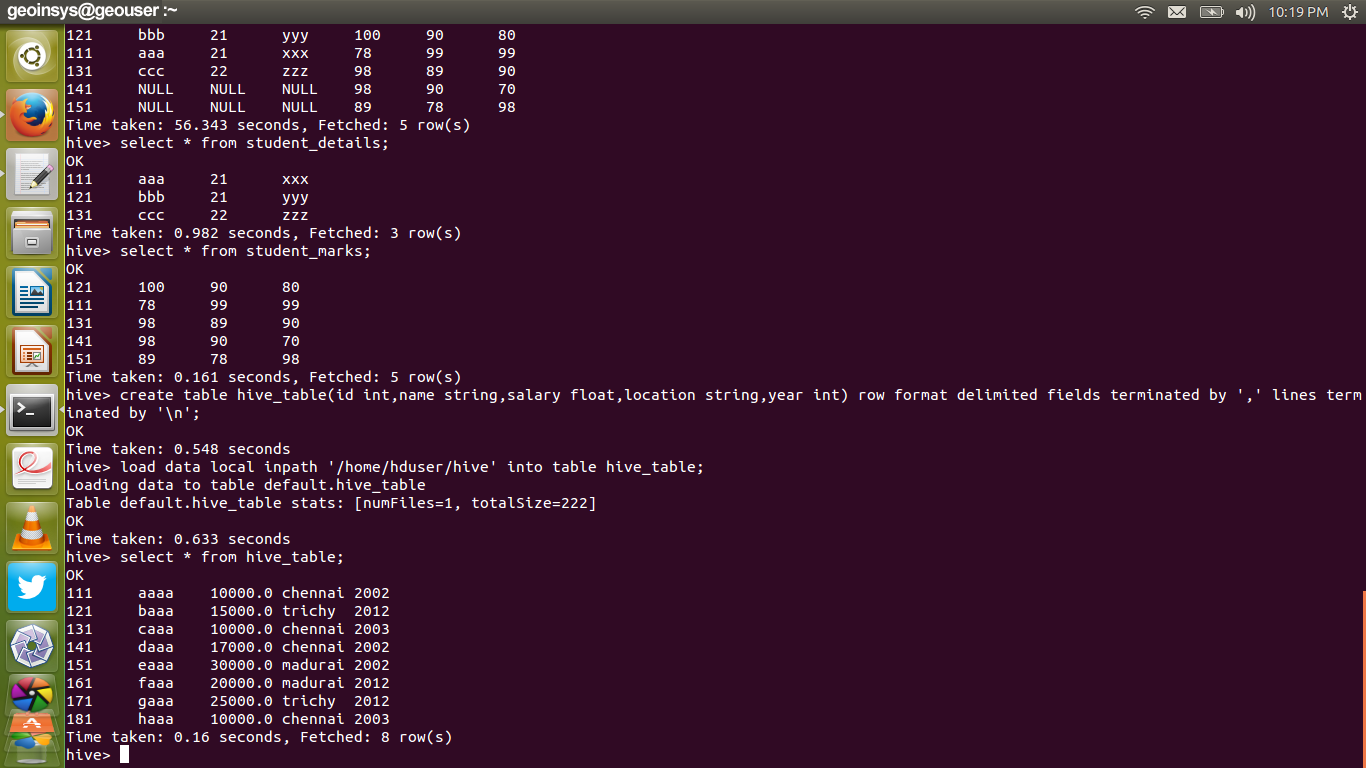


**#Loading data from local file into the non-partitioned table**

**load data local inpath ‘/home/geouser/student.txt’ into table hive\_table;**

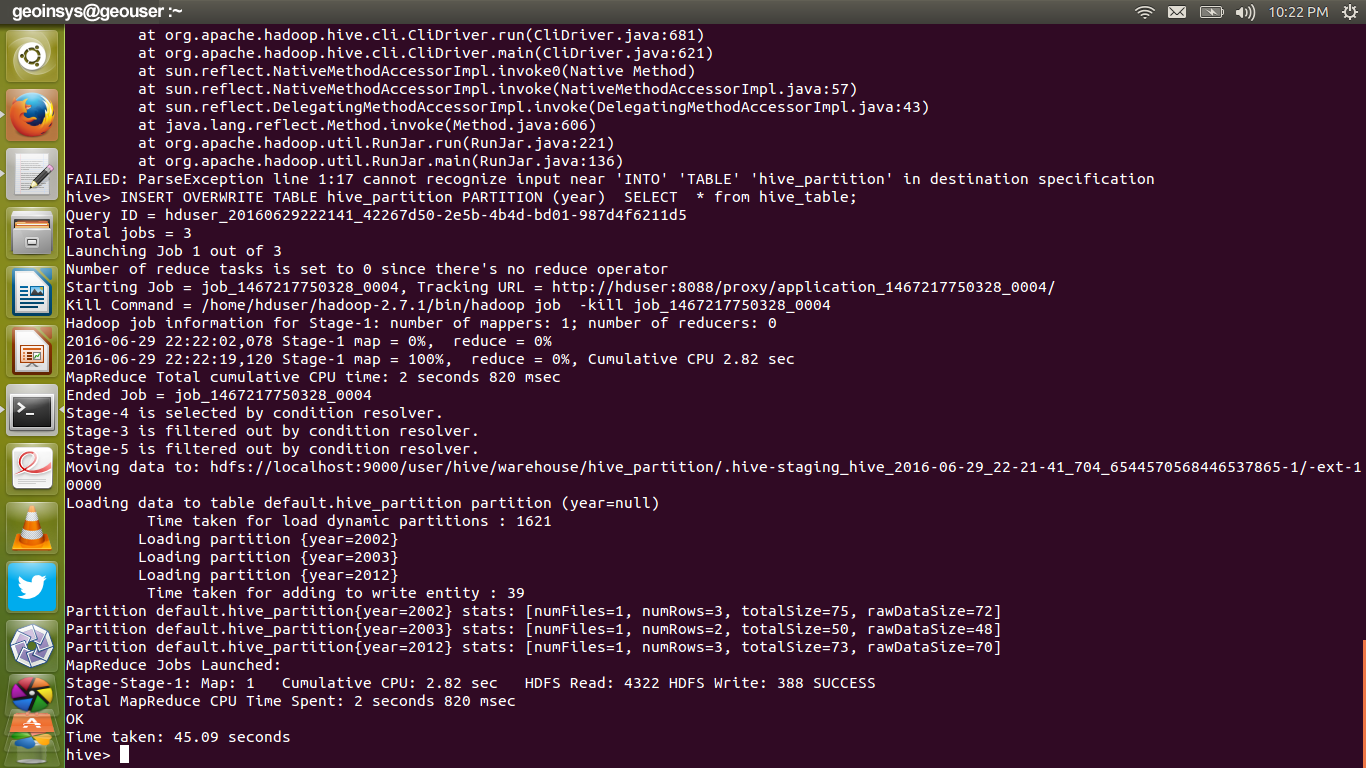


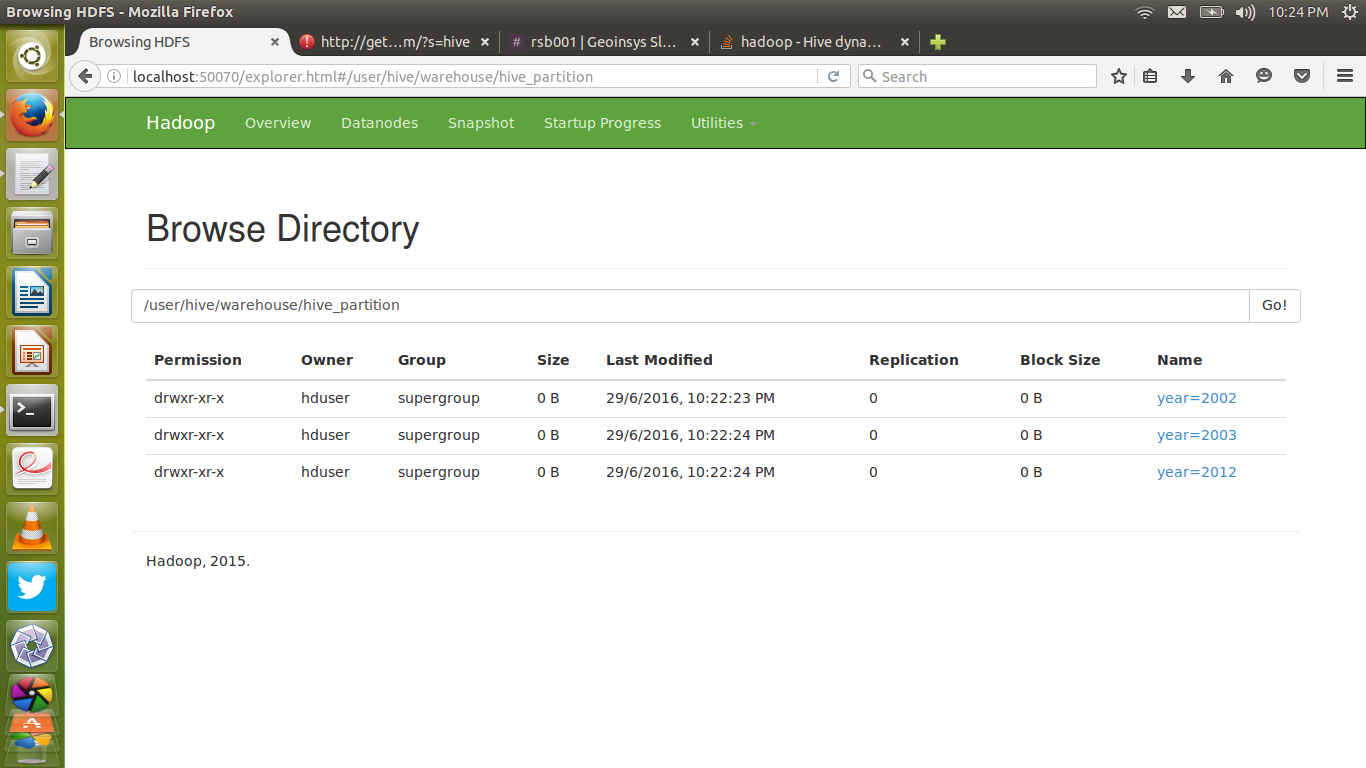
**select \* from hive\_table;**

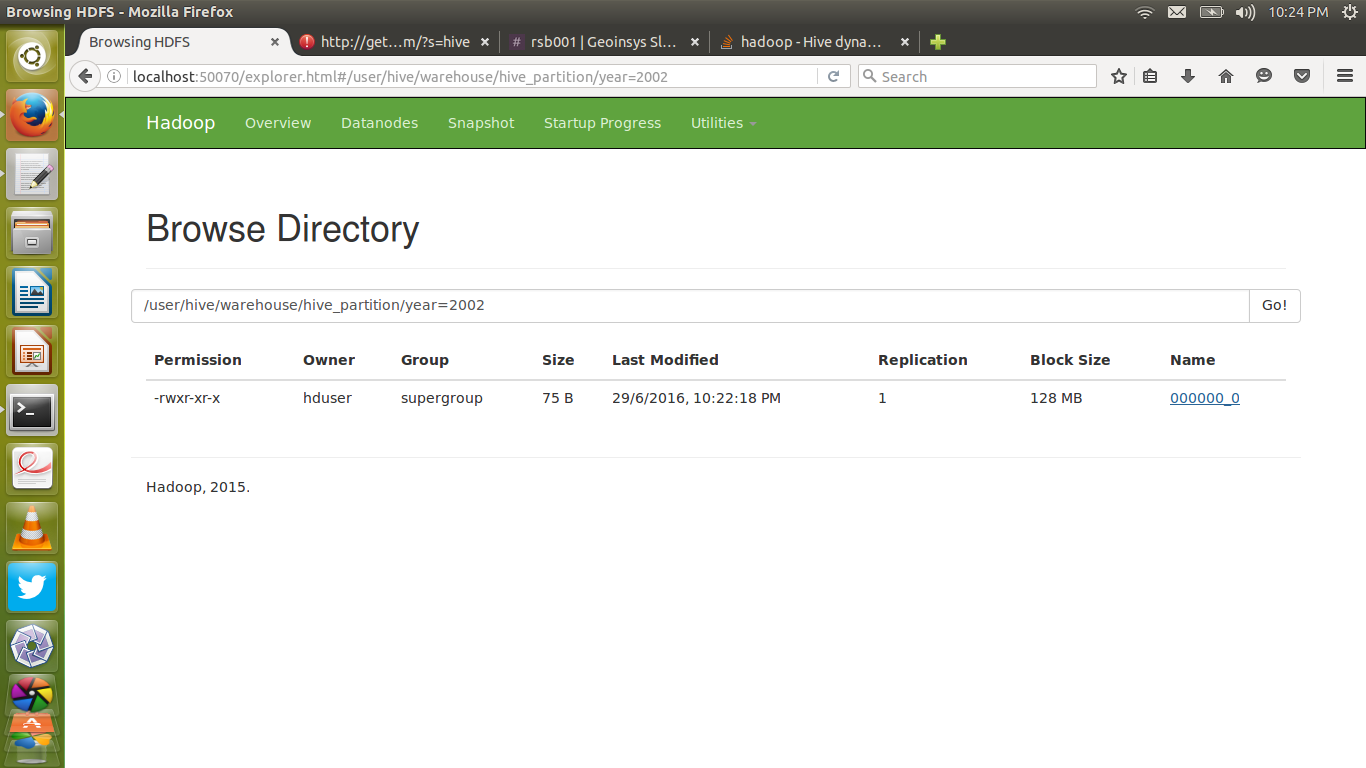


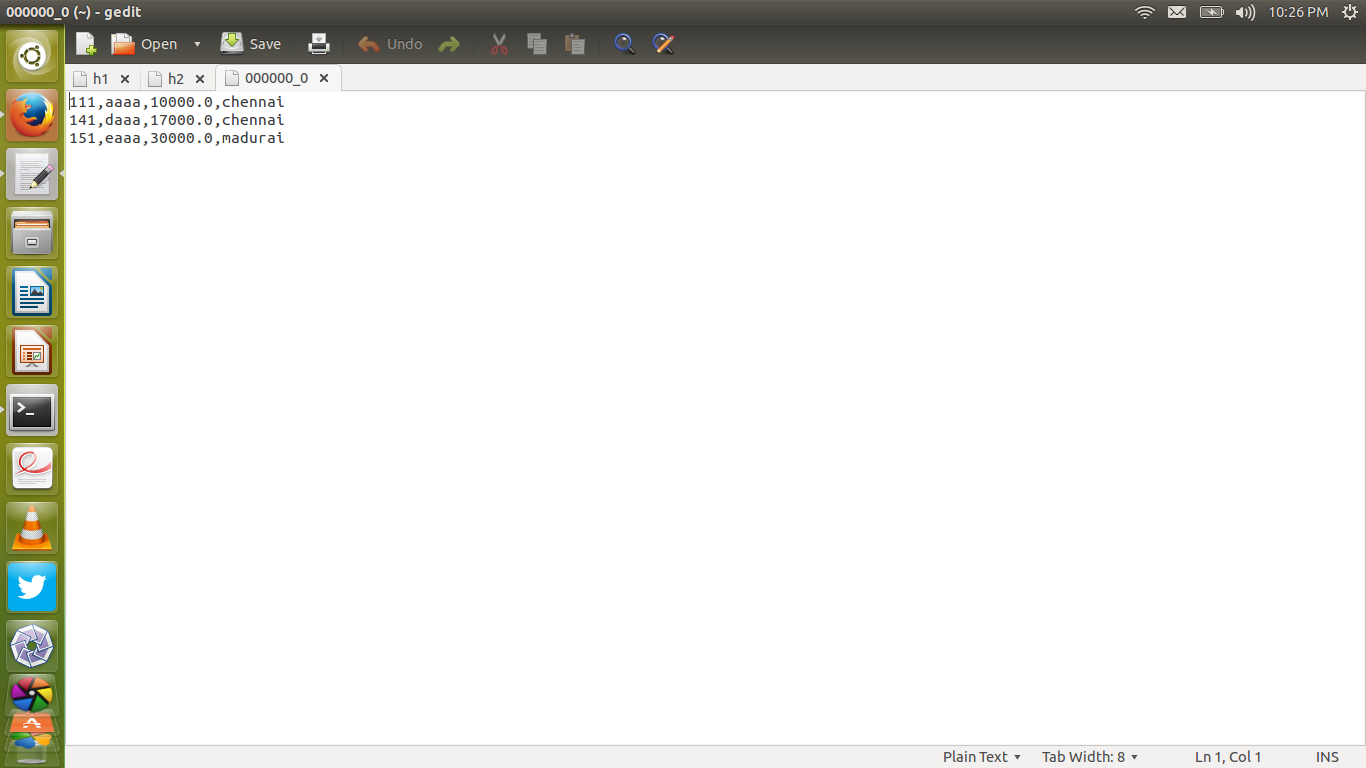


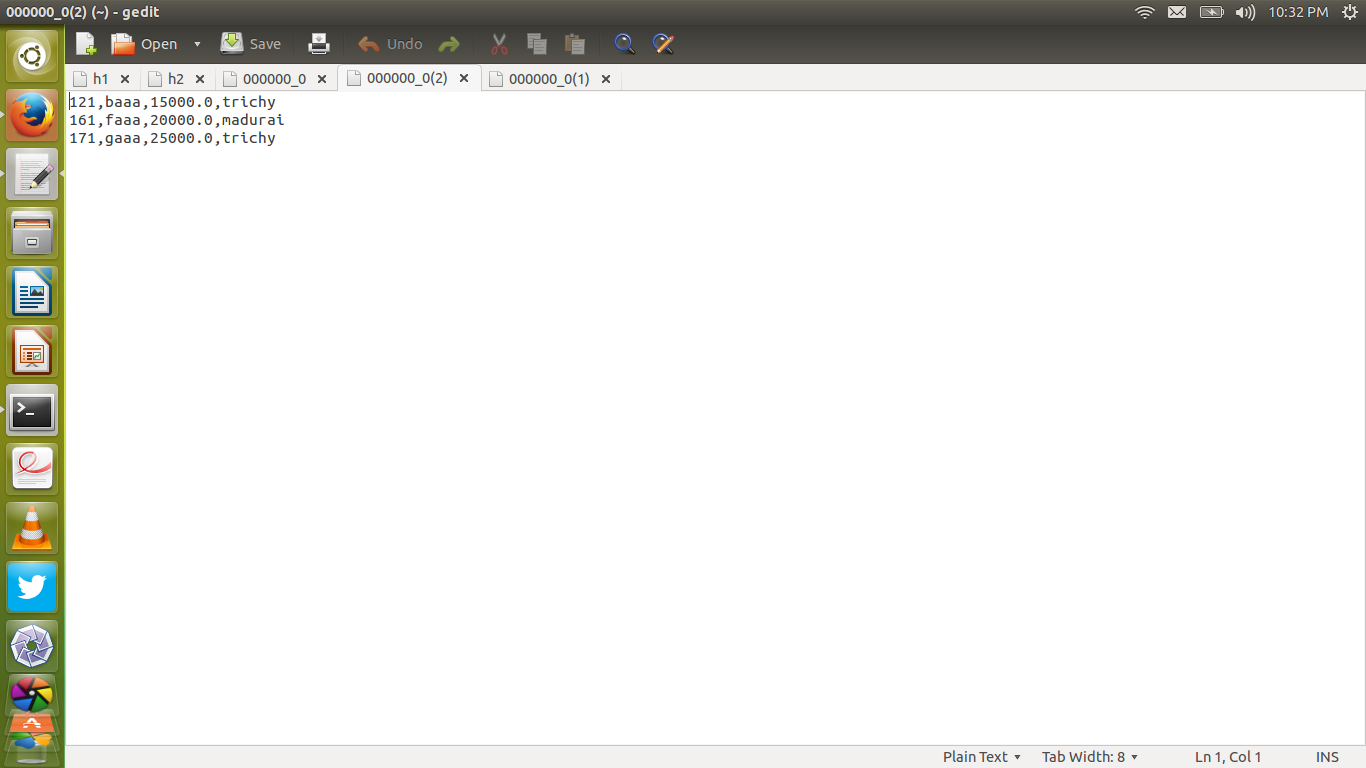
**INSERT OVERWRITE INTO TABLE hive\_partition PARTITION (year) SELECT \* from hive\_table;**

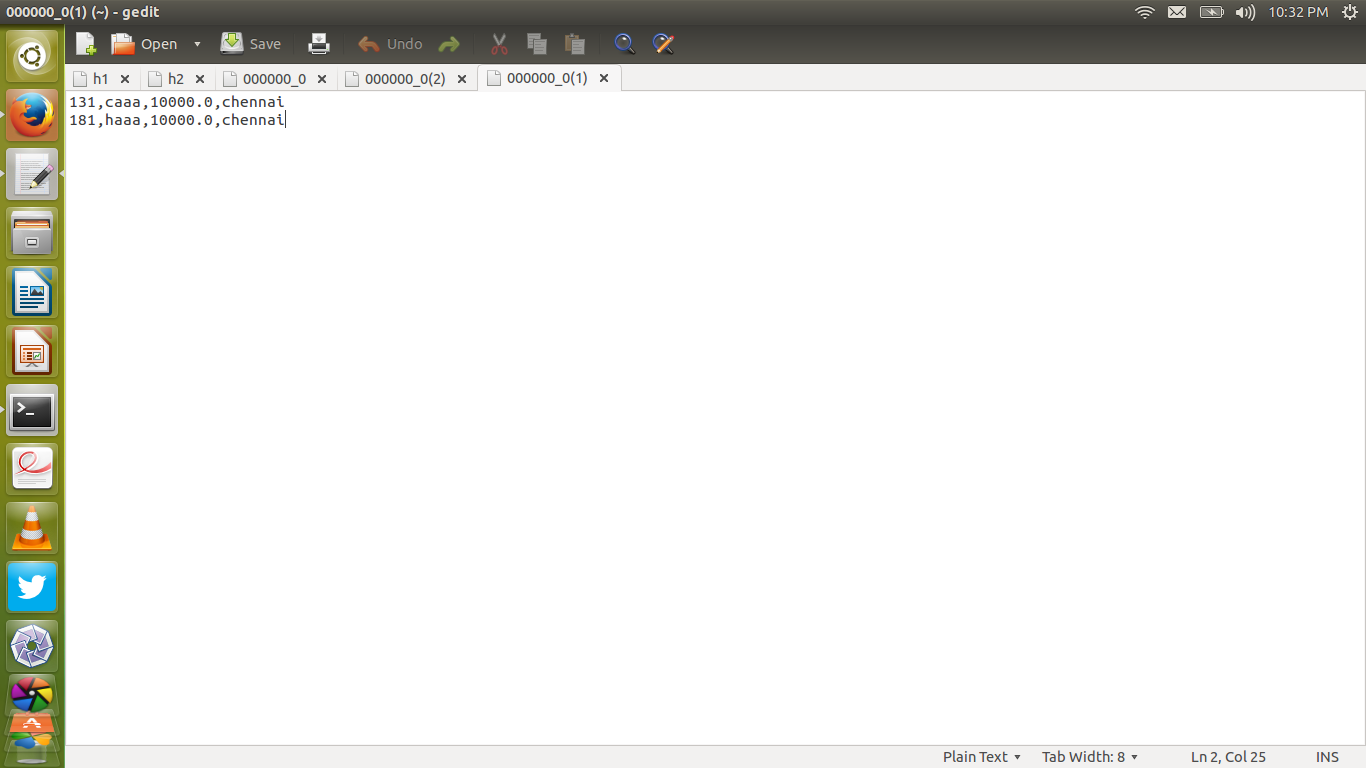












#### **Hive interview questions and answers:**

#### **[1. What is hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "1)**

**Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy. It’s an open source project under the Apache Software Foundation, it’s a data warehouse software ecosystem in Hadoop. Which manage vast amount of structured data sets, by using HQl language; it’s similar to SQL.**

#### **[2. Why do we need Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "2)**

**Hive is a tool in Hadoop ecosystem which provides an interface to organize and query data in a database like fashion and write SQL like queries. It is suitable for accessing and analyzing data in Hadoop using SQL syntax.**

#### **[3. Who developed hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "3)**

**Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.**

#### **[4.Which companies use hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "4)**

**Hive is commonly used at Facebook for analytical purposes,**

**Amazon uses it in Amazon Elastic MapReduce and etc.**

#### **[5. Does Hive support 100% SQL Quries like Insert, Delete and Updates?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "5)**

**Hive doesn’t support Updates in record level. To update, It integrate with Hbase. The versions above Hive 0.13 will support update and delete in but it has some limitations.**

**They are,**

**1.It can only be performed on tables that support ACID.  
2.If a table is to be used in ACID writes (insert, update, delete) then the table property “transactional” must be set on that table.  
3.Only ORC file format is supported in this.  
4.Tables must be bucketed to make use of these features.**

#### **[6. Where hive is the best suitable?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "6)**

**When you are doing data warehouse applications, where you are getting static data instead of dynamic data, when the application on high latency (response time high), where a large data set is maintained and mined for insights, reports, when we are using queries instead of scripting we have to use Hive.**

#### **[7. When hive is not suitable?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "7)**

**It doesn’t provide OLTP transactions supports only OLAP transactions. If application required OLTP, switch to NoSQL databases. HQL queries have higher latency, due to the mapreduce.**

#### **[8. Why I choose Hive instead of MapReduce?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "8)**

**There are Partitions to simplify the data process, Bucketing for sampling the data, sort the data quickly, and simplify the mapreduce process. Partitions and Buckets can segmenting large data sets to improve Query performance in Hive. So It is highly recommendable for structure data.**

#### **[9. Can I access Hive without Hadoop?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "9)**

**Hive store and process the data on the top of Hadoop, but it’s possible to run in Other data storage systems like Amazon S3, GPFS (IBM) and MapR file systems.**

#### **[10. Hive Support Acid Transactions?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "10)**

**By default it doesn’t support record-level update, insert and delete, but recent Hive 1.4 later versions supporting insert, update and delete operations. So hive support ACID transactions.**

#### **[11. What is the relationship between MapReduce and Hive? or How Mapreduce jobs submits on the cluster?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "11)**

**Hive provides no additional capabilities to MapReduce. The programs are executed as MapReduce jobs via the interpreter. The Interpreter runs on a client machine which rurns HiveQL queries into MapReduce jobs. Framework submits those jobs onto the cluster.**

#### **[12. What are the three different modes in which hive can be run?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "12)**

**1.Local mode,**

**2.Distributed mode,**

**3.Pseudodistributed mode.**

#### **[13. What is Hive MetaStore?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "13)**

**It is a relational database storing the metadata of hive tables, partitions, Hive databases etc. MetaStore is a central repository of Hive, that allows to store meta data in external database. By default Hive store meta data in Derby database, but you can store in MySql, Oracle depends on project.**

#### **[14. What is HiveQL or HQL?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "14)**

**The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore.**

#### **[15. Wherever you run hive query, first it creates new metastore\_db, why? What is the importance of Metastore\_db?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "15)**

**When we run the hive query, first it creates a local metastore, before creates the metastore first Hive checks whether metastore is already exists or not? If presents shows error, else the process goes on. This configuration is set in hive-site.xml like this.**

**<property>**

**<name>javax.jdo.option.ConnectionURL</name>**

**<value>jdbc:derby:;databaseName=metastore\_db;create=true</value>**

**<description>JDBC connect string for a JDBC metastore</description>**

**</property>**

#### **[16. Different Hive metastore configuration.](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "16)**

**There are three types of metastores configuration called as,**

**1) Embedded metastore**

**If Hive run any query first it enter into embedded mode, It’s default mode. In Command line all operations done in embedded mode only, it can access Hive libraries locally. In the embedded metastore configuration, hive driver, metastore interface and databases use same JVM. It’s good for development and testing.**

**2) Local metastore**

**In local metastore the metastore store data in external databases like MYSQL. Here Hive driver and metastore run in the same JVM, but remotely communicate with external Database. For better protection required credentials in Local metastore.**

**3) Remote metastore.**

**Where as in Remote server, use remote mode to run the queries over Thift server. In Remote metastore, Hive driver and metastore interface would be running in a different JVM. So for better protection, required credentials such are isolated from Hive users.**

#### **[17. Hive can process any type of data formats?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "17)**

**Yes, Hive uses the SerDe interface for IO operations. Different SerDe interfaces can read and write any type of data. If normal directly process the data where as different type of data is in the Hadoop, Hive use different SerDe interface to process such data.**

**Example:**

**MetadataTypedColumnsetSerDe : used to read/write CSV format data.**

**JsonSerDe : process Json data.**

**RejexSerDe : process weblog data.**

**AvroSerde : Avro format data.**

#### **[18. What is SerDe in Apache Hive ?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "18)**

**A SerDe is a short name for a Serializer Deserializer. Hive uses SerDe (and FileFormat) to read and write data from tables. An important concept behind Hive is that it DOES NOT own the Hadoop File System (HDFS) format that data is stored in.**

**Users are able to write files to HDFS with whatever tools/mechanism takes their fancy(“CREATE EXTERNAL TABLE” or “LOAD DATA INPATH,” ) and use Hive to correctly “parse” that file format in a way that can be used by Hive.**

**A SerDe is a powerful (and customizable) mechanism that Hive uses to “parse” data stored in HDFS to be used by Hive.**

#### **[19. Can you explain different type of SerDe?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "19)**

**By default Hive used Lazy Serde also allows Jeson Serde and most often used RegexSerde to be Serialized and DeSerialized Data.**

#### **[20. How Hive Serialize and DeSerialize the data?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "20)**

**In Hive language, SerDe also called Serialization and DeSerialization. Usually when read/write the data, user first communicate with inputformat, then it connect with Record reader to read/write record. The data is stored in Serialized (binary) format in Record.**

**To serialize the data dat goes to row, here deserialized custem serde use object inspector to deserialize the data in fields. now user see the data in the fields, that deliver to the end user.**

#### **[21. How Hive use Java in SerDe?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "21)**

**To insert data into table, Hive create an object by using Java. To transfer java objects over network, the data should be serialized. Each field serialized by using Object inspector and finally serialized data stored in Hive table.**

#### **[22. How do you write your own custom SerDe ?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "22)**

**In most cases, users want to write a Deserializer instead of a SerDe, because users just want to read their own data format instead of writing to it.**

**For example,**

**The RegexDeserializer will deserialize the data using the configuration parameter ‘regex’, and possibly a list of column names If your SerDe supports DDL (basically, SerDe with parameterized columns and column types), you probably want to implement a Protocol based on DynamicSerDe, instead of writing a SerDe from scratch.**

**The reason is that the framework passes DDL to SerDe through “thrift DDL” format, and it’s non – trivial to write a “thrift DDL” parser.**

#### **[23. What is the need for custom Serde?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "23)**

**Depending on the nature of data the user has, the inbuilt SerDe may not satisfy the format of the data. SO users need to write their own java code to satisfy their data format requirements.**

#### **[24.What Is the HWI?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "24)**

**The Hive Web Interface is an alternative to the command line interface. HWI is a simple graphical interface, It’s hive web interface. The HWI allows start at database level directly. you can get all SerDe, column names and types and simplifies the hive steps. It’s seccession based interface, so you can run multiple hive queries simultaneously. There is no local metastore mode in HWI.**

#### **[25. Which classes are used by the Hive to Read and Write HDFS Files?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "25)**

**Following classes are used by Hive to read and write HDFS files**

**-TextInputFormat/HiveIgnoreKeyTextOutputFormat : These 2 classes read/write data in plain text file format.**

**-SequenceFileInputFormat/SequenceFileOutputFormat : These 2 classes read/write data in hadoop SequenceFile format**

#### **[26. What is the functionality of Query Processor in Apached Hive ?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "26)**

**This component implements the processing framework for converting SQL to a graph of map/reduce jobs and the execution time framework to run those jobs in the order of dependencies.**

#### **[27. How many ways you can run Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "27)**

**1. In CLI mode (By using command line inerface).**

**[2. By using JDBC or ODBC.](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "27)**

**[3. By Called Hive Thift client. It allows java, PHP, Python, Ruby and C++ to write commands to run in Hive.](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "27)**

#### **[28. What is different between database and data-warehouse?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "2)**

#### **[29. What are the different types of tables available in HIve?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "29)**

**There are two types. Managed table and external table. In managed table both the data an schema in under control of hive but in external table only the schema is under control of Hive.**

#### **[30. Difference between Internal and External Table?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "30)**

**External table: Schema is stored in Database. Actual data stored in Hive tables. If data lost in External table, it lost only metastore, but not actual data.**

**Internal table: MetaStore and actual data both stored in local system. If any situation, data lost, both actual data and meta store will be lost.**

#### [**31. How to load data in hive table?**](http://www.geoinsyssoft.com/hive-interview-questions-answers/31)

#### **[32. What is the difference between Describe and describe extended?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "32)**

**To see table definition in Hive, use the command describe <table name>;**

**To see more detailed information about the table, We use the command describe extended <tablename>;**

**Another important command it’s also describe all details in a clean manner describe formatted <tablename>;**

#### **[33. What are the functions for alter table keyword in hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "33)**

**Alter the attributes of a table such as changing its table name, changing column names, adding columns, and deleting or replacing columns.**

#### **[34. How to drop a table in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "34)**

**Using drop table command.**

**DROP TABLE <TABLE\_NAME>;**

**When you drop a table from Hive Metastore, it removes the table/column data and their metadata. It can be a normal table (stored in Metastore) or an external table (stored in local file system); Hive treats both in the same manner, irrespective of their types.**

#### **[35. Define ORDER BY in Hive:](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "35)**

#### **[36. Difference between order by and sort by in hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "36)**

**ORDER BY – use single reducer. If data is too large, it’s take a long time to sort the data.**

**SORT BY – use number of reducers, so it can process quickly.**

**In the first case (order by) maps sends each value to the single reducer and count them all. In the second case (sort by) maps splits up the values to many reducers and each reduce generates its list and finds the count. So it can sort quickly.**

**Example:**

**SELECT name, id, cell FROM user\_table ORDER BY id, name;**

**SELECT name, id, cell FROM ur\_table DISTRIBUTE BY id SORT BY name;**

#### **[37. Define GROUP BY.](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "37)**

#### **[38. What is the difference between Like and Rlike operators in HIVE?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "38)**

**Like : Used to find the substrings within a main string with regular expression %.**

**Rlike : It’s a special function which also finds the sub strings within a main string, but return true or false without using regular expression.**

**Example:**

**Tablename is table, column is name.**

**name=VenuKatragadda, venkatesh, venkateswarlu**

**Select \* from table where name like “venu%. //VenuKatragadda.**

**select \* from table where name rlike “venk%”. // false, true, true.**

#### **[39. How Hive Organize the data?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "39)**

**Hive organize in three ways such as Tables, Partitions and Buckets.**

**-Tables organize based on Arrays, Maps, primitive column types.**

**-Partitions has one or more partition keys based on project requirements.**

**-Buckets used for analyze the data for sampling purpose. It’s good approach to process a pinch of data in the form of buckets instead of process all data.**

#### **[40. What is partitioning in hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "40)**

**Hive organizes tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department. Using partition, it is easy to query a portion of the data.**

#### **[41. What is the use of partition in hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "41)**

**To analyze a particular set of data, not required to load entire data, desired data partition is a good approach. To achieve this goal, Hive allows to partition the data based on particular column.**

#### **[42. Types of partitions in Hive.](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "42)**

**Static partition and Dynamic partition, both can optimize the Hive performance.**

**For Instant, required a particular year information, partition based on year.**

#### **[43. What is difference between static and dynamic partition of a table?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "43)**

**To prune data during query, partition can minimize the query time. The partition is created when the data is inserted into table.**

**Static partition can insert individual rows where as Dynamic partition can process entire table based on a particular column. At least one static partition is must to create any (static, dynamic) partition.**

**If you are partitioning a large datasets, doing sort of a ETL flow Dynamic partition partition recommendable.**

#### **[44. What is bucketing?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "44)**

**Tables or partitions are sub-divided into buckets, to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table**

#### **[45. Why we are using buckets in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "45)**

**To process many chunks of files, to analyze vast amount of data, sometime burst the process and time. Bucketing is a sampling concept to analyze the data, by using hashing algorithm.**

**set hive.enforce.bucketing=true; can enable the process**

#### **[46. What is the difference between partition and bucketing?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "46)**

**The main aim of both Partitioning and Bucketing is execute the query more efficiently. When you are creating a table the slices are fixed in the partitioning the table.**

**Bucketing follows Hash algorithm. Based on number of buckets, randomly the data inserted into the bucket to sampling of the data.**

#### **[47. What are the built in operators in hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "47)**

**1. Relational Operators – These operators are used to compare two operands.**

**Operators:**

**=, !=, <=, <, >=, >, IS NULL, IS NOT NULL, LIKE, RLIKE, REGEXP**

**2. Arithmetic Operators – These operators support various common arithmetic operations on the operands. All of them return number types**

**Operators:**

**+, -, \*, /, %, &, |, ^, ~**

**3. Logical Operators – Operators are logical expressions.All of them return either TRUE or FALSE.**

**Operators:**

**AND,&&,OR,||,NOT,!**

**4. Complex Operators – These operators provide an expression to access the elements of Complex Types.**

**Operators:**

**A[n],M[key],S.x**

#### **[48. What is view in hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "48)**

**Views are generated based on user requirements. You can save any result set data as a view. The usage of view in Hive is same as that of the view in SQL. It is a standard RDBMS concept. We can execute all DML operations on a view.**

#### **[49. What is index?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "49)**

**An Index is nothing but a pointer on a particular column of a table. Creating an index means creating a pointer on a particular column of a table.**

#### **[50. How Hive can improve performance with ORC format tables?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "50)**

**Hive can store the data in highly efficient manner in the Optimized Row Columnar (ORC) file format. It can ease many Hive file format limitations. Using ORC files can improves the performance when reading, writing, and processing data.**

**Enable this format by run this command and create table like this.**

**set hive.compute.query.using.stats=true;**

**set hive.stats.dbclass=fs;**

**CREATE TABLE orc\_table (**

**id int,**

**name string )**

**ROW FORMAT DELIMITED**

**FIELDS TERMINATED BY ‘\;’**

**LINES TERMINATED BY ‘\n’**

**STORED AS ORC;**

#### **[51. What is the importance of Vectorization in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "51)**

**It’s a query optimization technique. Instead of processing multiple rows, Vectorization allows to process process a batch of rows as a unit. Consequently it can optimize query performance.**

**The file must be stored in ORC format to enable this Vectorization. It’s disabled by default, but enable this property by run this command.**

**set hive.vectorized.execution.enabled=true;**

#### **[52. What is User Interface (UI)?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "52)**

**UI: This interface is interpreter between users and Driver, which accept queries from User and execute on the Driver.**

**Now two types of interfaces available in Hive such as command line interface and GUI interface. Hadoop provides Thrift interface and JDBC/ODBC for integrating other applications.**

#### **[53. What is importance of Driver in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "53)**

**Driver: It manages life cycle of HiveQL queries. Driver receives the queries from User Interface and fetch on the ODBC/JDBC interfaces to process the query.**

**Driver create separate independent section to handle each query.**

**Compiler: Compiler accept plans from Drivers and gets the required metadata from MetaStore, to execute Plan.**

**MetaStore: Hive Store meta data in the table. It means information about data is stored in MetaStore in the form of table, it may be internal or external table. Hive compiler get the meta data information from metastore table.**

**Execute Engine: Hive Driver execute the output in the execution Engine.**

**Here, execute engine executes the queries in the MapReduce**

**JobTracker: Based on Required information, Hive queries run in the MapReduce to process the data.**

#### **[54. When we are use explode in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "54)**

**Sometime Hadoop developer takes array as input and convert into a separate table row. To achieve this goal, Hive use explode, it acts as interpreter to convert complex data-types into desired table formats.**

**Syntax:**

**SELECT explode (arrayName) AS newCol FROM TableName;**

**SELECT explode(map) AS newCol1, NewCol2 From TableName;**

#### **[55. What is ObjectInspector functionality in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "55)**

**Hive uses ObjectInspector to analyze the internal structure of the rows, columns and complex objects. Additionally gives us ways to access the internal fields inside the object. It not only process common data-types like int, bigint, STRING, but also process complex data-types like arrays, maps, structs and union.**

#### **[56. Is a job split into map?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "56)**

#### **[57. How to display the present database name in the terminal?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "57)**

**There are two ways to know the current database. One temporary in cli and second one is persistently.**

**1) in CLI just enter this command: set hive.cli.print.current.db=true;**

**2) In hive-site.xml paste this code:**

**<property>**

**<name>hive.cli.print.current.db</name>**

**<value>true</value>**

**</property>**

**In second scenario, you can automatically display the Hive database name when you open terminal.**

#### **[58. Can you overwrite Hadoop Mapreduce configuration in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "58)**

**Yes, You can overwrite Hive map, reduce steps in hive conf settings. Hive allows to overwrite Hadoop configuration files.**

#### **[59. What is Thrift server & client, JDBC and ODBC driver importance in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "59)**

**Thrift is a cross language RPC framework which generate code and combines a software stack finally execute the Thrift code in remote server. Thrift compiler acts as interpreter between server and client. Thrift server allows a remove client to submit request to Hive, using different programming languages like Python, Ruby and scala.**

**JDBC driver: A JDBC driver is a software component enabling a Java application to interact with a database.**

**ODBC driver: ODBC accomplishes DBMS independence by using an ODBC driver as a translation layer between the application and the DBMS.**

#### **[60. How to write single and multiple line commands in Hive?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "60)**

**To write single line commands we use –followed by commands.**

**Eg: –It is too important step.**

**Hive doesn’t supports multiple comments now.**

#### **[61. What is the difference between Hive and Hbase?](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "61)**

**-Hive allows most of the SQL queries, but Hbase not allows SQL queries directly.**

**-Hive doesn’t support record level update, insert, and deletion operations on table, but Hbase can do it.**

**-Hive is a Data warehouse framework where as Hbase is a NoSQL database.**

**-Hive run on the top of Mapreduce, Hbase run on the top of HDFS.**

#### **[AVRO FILE:](http://www.geoinsyssoft.com/hive-interview-questions-answers/" \l "62)**

**Avro stores both the data definition and the data together in one message or file making it easy for programs to dynamically understand the information stored in an Avro file or message.**

**Avro stores the data definition in JSON format making it easy to read and interpret, the data itself is stored in binary format making it compact and efficient. Avro files include markers that can be used to splitting large data sets into subsets suitable for MapReduce processing.**

#### **SQOOP:**

**Sqoop is a tool designed to transfer data between Hadoop and relational database servers. It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases.**

#### **HIVE:**

**Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.  It’s an open source project under the Apache Software Foundation, it’s a data warehouse software ecosystem in Hadoop. Which manage vast amount of structured data sets, by using HQL language; it’s similar to SQL.**

***Step-1***

**Create database in MySQL**

***create database sqoopdb;***

***Step-2***

**Create database in MySQL**

***use sqoopdb;***

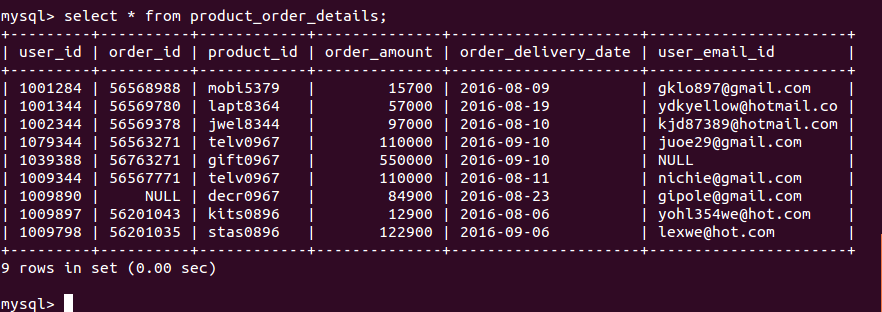
***Step-3***

**Create table in sqoopdb;**

***Create table product\_order\_details (user\_id int, order\_id int, product\_id varchar(20), order\_amount bigint, order\_delivery\_date varchar(20), user\_email\_id varchar(20));***

***Step-4***

**Insert values into product\_order\_details**



***Step-5***

**Import table from MySQL to HIVE in AVRO File Format**

***sqoop import \***

**–*connect jdbc:mysql://localhost/sqoopdb \***

**–*username root \***

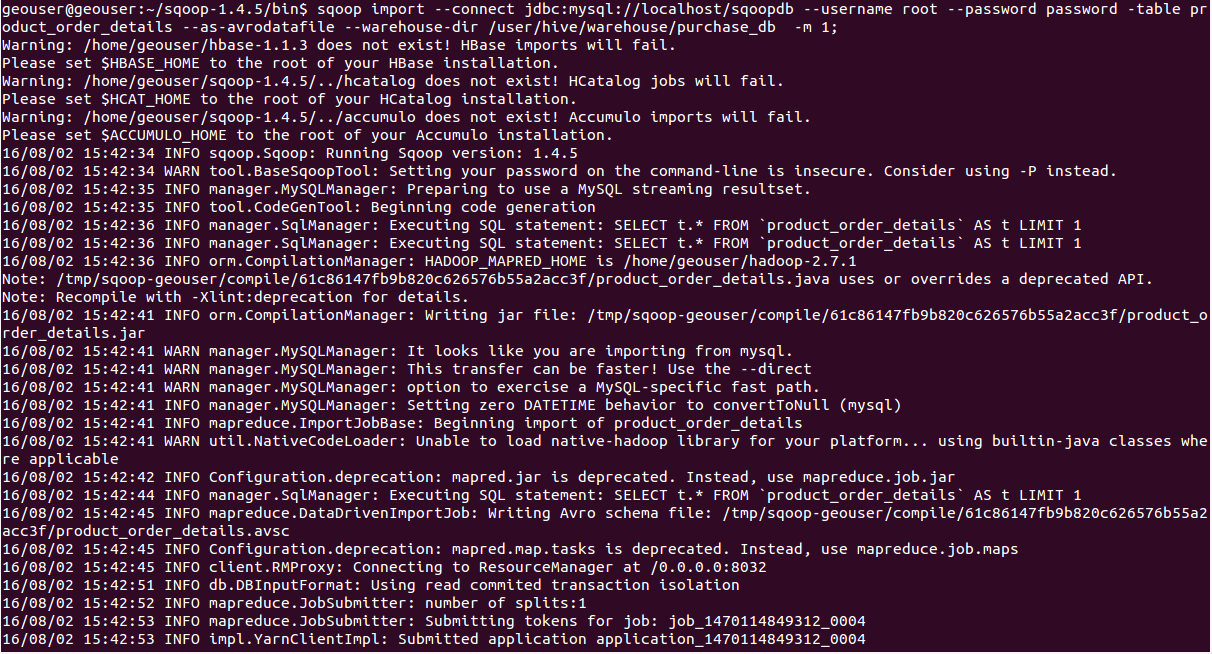
**–*password password \***

**–*table product\_order\_details \***

**–*as-avrodatafile \***

**–*warehouse-dir /user/hive/warehouse/purchase\_db\***

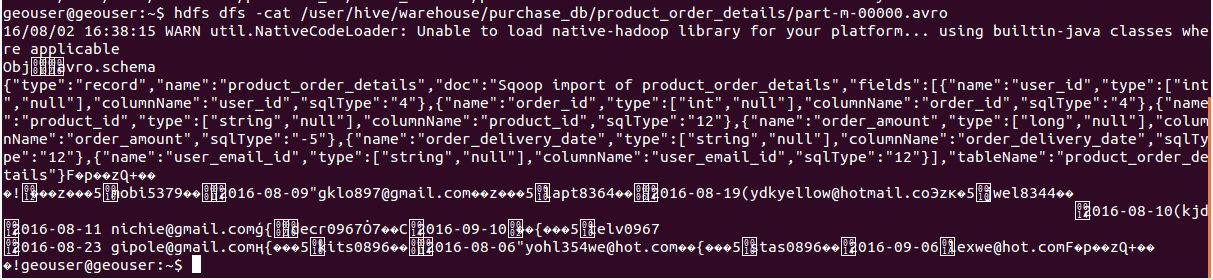
***-m 1;***



***Step-6***

**View the content of the AVRO files**

***hdfs dfs -cat /user/hive/warehouse/purchase\_db/product\_order\_details/part-m-00000.avro***



#### **Convert AVRO file into AVSC:**

***AVSC:***

**It contain Schema of the AVRO file. We can create a new table using the avsc schema file.**

***Step-7***

**Get the avro file from HDFS to Local**

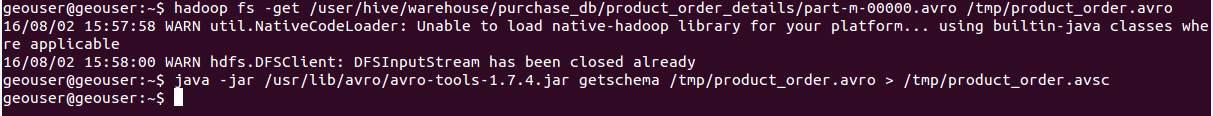
***hadoop fs -get /user/hive/warehouse/purchase\_db/product\_order\_details/part-***

***m-00000.avro /tmp/purchase.avro***

***Step-8***

**Convert Avro into Avsc**

***java -jar /usr/lib/avro/avro-tools-1.7.4.jar getschema /tmp/product\_order.avro > /tmp/product\_order.avsc***

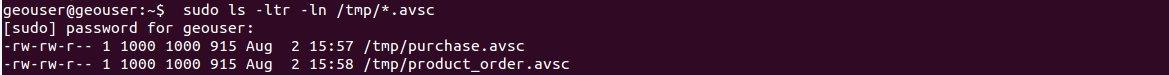


**Create a new table using AVSC schema file in Hive:**

***Step-9***

**List the avsc files in local**

**sudo ls -ltr -ln /tmp/\*.avsc**



***Step-10:***

**Put the avsc file from local to Hive**

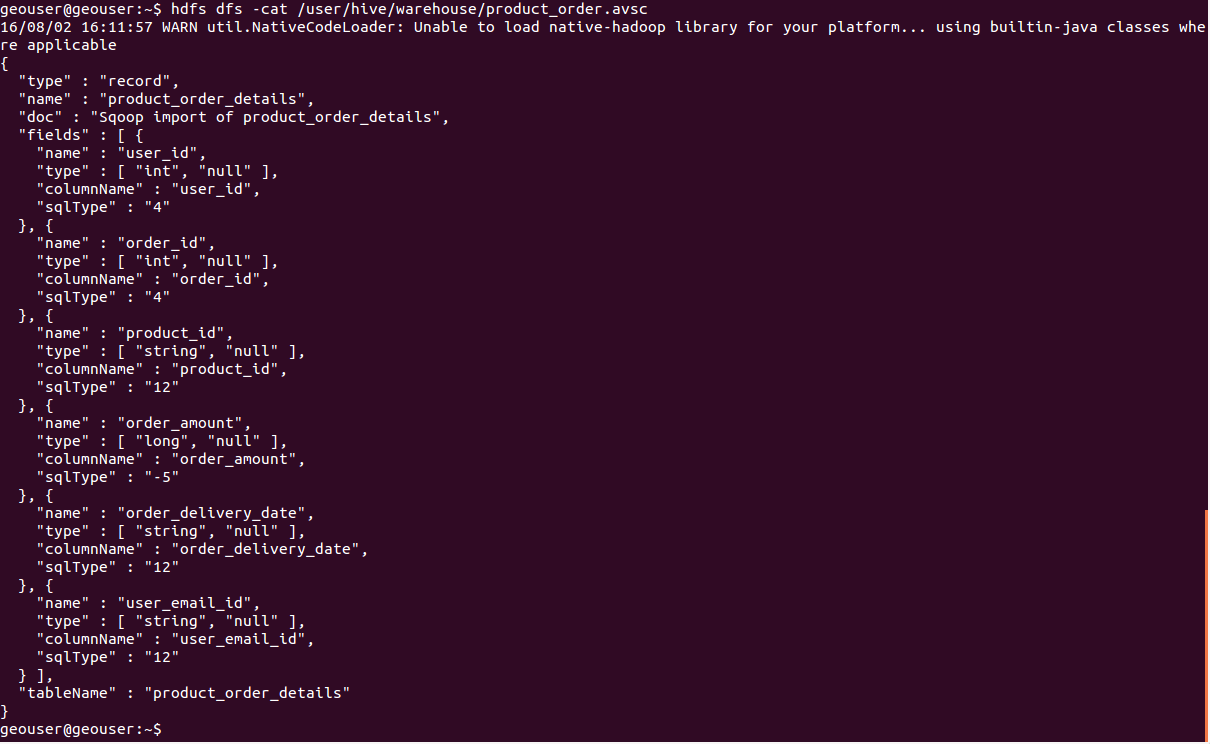
***hdfs dfs -put /tmp/product\_order.avsc /user/hive/warehouse/***



***Step\_11***

**View the content of the AVSC file**

***hdfs dfs -cat /user/hive/warehouse/product\_order.avsc***



***Step-12:***

**Create a new hive table using avsc schema file**

***create external table purchase\_details\_table stored as avro tblproperties(‘avro.schema.url’=’/user/hive/warehouse/product\_order.avsc’);***

***Step-14***

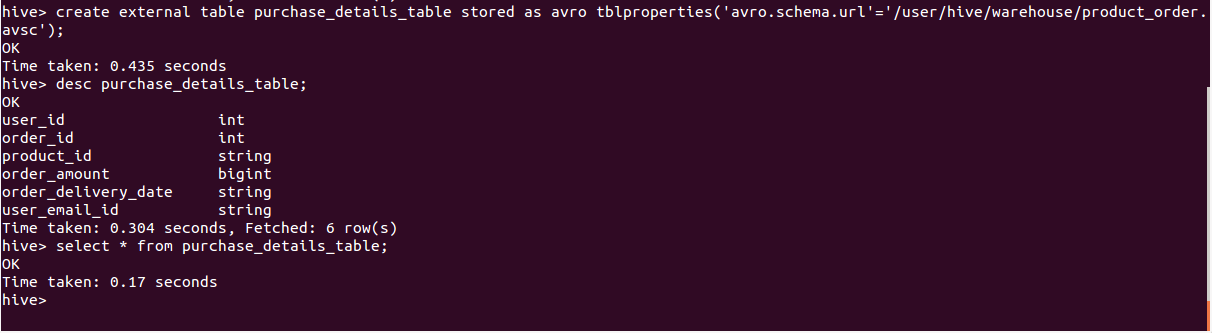
**Describe and Select the newly created table**

**(i) Describe**

***desc purchase\_details\_table;***

**(ii). Select**

***select \* from purchase\_details\_table;***



***Step-15***

**Create a new hive table using avsc schema file and insert value for the table using another file**

***create external table purchase\_details stored as avro location’/user/hive/warehouse/purchase\_db/product\_order\_details’ tblproperties(‘avro.schema.url’=’/user/hive/warehouse/product\_order.avsc’);***

***Step-16:***

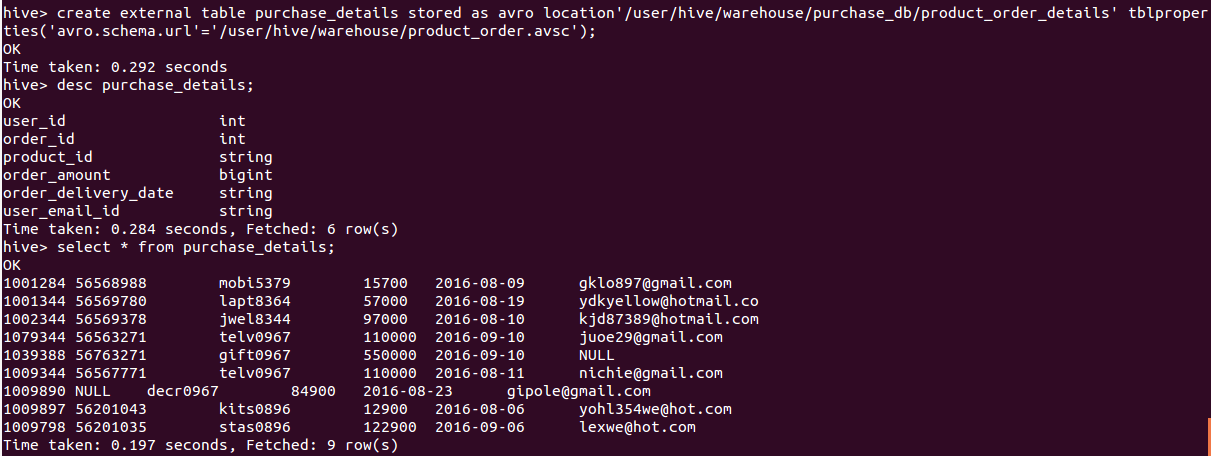
**Describe and Select the newly created table**

**(i) Describe**

***desc purchase\_details;***

**(ii). Select**

***select \* from purchase\_details;***



#### **Description about command:**

**sqoop import -> keyword for importing**

**connect -> keyword**

**<path> -> path of the mysql**

**username -> keyword**

**root -> name of the user, which is stored in the mysql**

**password -> keyword**

**password -> password for the user**

**table -> keyword**

**product\_order\_details -> importing table**

**as-avrodatafile -> keyword for avro format file**

**warehouse-dir -> keyword for target directory**

**/user/hive/warehouse/purchase\_db/product\_order\_details -> path for where our table should be stored.**

**m 1 -> number of partitioned**

***AVRO JSON CONVERSIONS:***

### **What is AVRO File Format:**

**Avro stores both the data definition and the data together in one message or file making it easy for programs to dynamically understand the information stored in an Avro file or message.**

**Avro stores the data definition in JSON format making it easy to read and interpret, the data itself is stored in binary format making it compact and efficient. Avro files include markers that can be used to splitting large data sets into subsets suitable for MapReduce processing.**

### **What is JSON File Format:**

**JSON(JavaScript Object Notation) is a minimal, readable format for structuring data. It is used primarily to transmit data between a server and web application,  as an alternative to XML.  Square space uses JSON to store and organize site content created with the CMS.**

### **AVSC:**

**AVSC is a Schema File.**

### **AVRO to JSON Conversion:**

#### **Step-1: First Download the avro-tools-1.7.4.jar**

#### **Step-2: Example Avro File**

**Objavro.schema\8E{“type”:”record”,”name”:”Avro\_File”,”doc”:”Sqoop import of Avro\_File”,”fields”:[{“name”:”User\_\_Name”,”type”:[“string”,”null”],”columnName”:”User\_\_Name”,”sqlType”:”12″},{“name”:”Product\_Id”,”type”:[“int”,”null”],”columnName”:”Product\_Id”,”sqlType”:”4″},{“name”:”Order\_Id”,”type”:[“string”,”null”],”columnName”:”Order\_Id”,”sqlType”:”12″},{“name”:”Delivery\_Date”,”type”:[“string”,”null”],”columnName”:”Delivery\_Date”,”sqlType”:”12″}],”tableName”:”Avro\_File”}\00\D0!\B2\00e\B1\AC\CBI6\80aɳ\D6\00alaister briito\00ʺ\DA/\00mo862041\0026-09-2016\00anifa mohammed\00̺\DA/\00la862041\0014-09-2016\00piyush manish\00κ\DA/\00mo862032\0016-09-2016\00vijay karthik\00к\DA/\00wa862098\0029-09-2016\D0!\B2\00e\B1\AC\CBI6\80aɳ**

#### **Step-3: Convert the AVRO file into JSON**

**java -jar <jar file with location> <keyword> <avro file with location>  >  <json file name with json file location where the json file will be stored>**

### java  -jar  /usr/lib/avro/avro-tools-1.7.4.jar  tojson  /tmp/Avro\_Format\_File.avro  >  /tmp/Json\_Format\_File.json

#### **Step-4: View the Json Format File which is converted from Avro file**

### hadoop fs -cat file:/tmp/Json\_Format\_File.json

**geouser@geouser:~$ hadoop fs -cat file:/tmp/Json\_Format\_File.json  
{“User\_\_Name”:{“string”:”alaister briito”},”Product\_Id”:{“int”:50024101},”Order\_Id”:{“string”:”mo862041″},”Delivery\_Date”:{“string”:”26-09-2016″}}  
{“User\_\_Name”:{“string”:”anifa mohammed”},”Product\_Id”:{“int”:50024102},”Order\_Id”:{“string”:”la862041″},”Delivery\_Date”:{“string”:”14-09-2016″}}  
{“User\_\_Name”:{“string”:”piyush manish”},”Product\_Id”:{“int”:50024103},”Order\_Id”:{“string”:”mo862032″},”Delivery\_Date”:{“string”:”16-09-2016″}}  
{“User\_\_Name”:{“string”:”vijay karthik”},”Product\_Id”:{“int”:50024104},”Order\_Id”:{“string”:”wa862098″},”Delivery\_Date”:{“string”:”29-09-2016″}}**



### **AVRO to AVSC Conversion:**

**It is used for get the schema from the AVRO File**

#### **Step-1: Convert the Avro File into Avsc**

### java -jar /usr/lib/avro/avro-tools-1.7.4.jar getschema /tmp/Avro\_Format\_File.avro > /tmp/Avsc\_Format\_File.avsc

#### **Step-2: View the Avsc file which is converted from Avro file**

### hadoop fs -cat file:/tmp/Avsc\_Format\_File.avsc

**geouser@geouser:~$ hadoop fs -cat file:/tmp/Avsc\_Format\_File.avsc  
{  
  “type” : “record”,  
  “name” : “Avro\_File”,  
  “doc” : “Sqoop import of Avro\_File”,  
  “fields” : [ {  
    “name” : “User\_\_Name”,  
    “type” : [ “string”, “null” ],  
    “columnName” : “User\_\_Name”,  
    “sqlType” : “12”  
  }, {  
    “name” : “Product\_Id”,  
    “type” : [ “int”, “null” ],  
    “columnName” : “Product\_Id”,  
    “sqlType” : “4”  
  }, {  
    “name” : “Order\_Id”,  
    “type” : [ “string”, “null” ],  
    “columnName” : “Order\_Id”,  
    “sqlType” : “12”  
  }, {  
    “name” : “Delivery\_Date”,  
    “type” : [ “string”, “null” ],  
    “columnName” : “Delivery\_Date”,  
    “sqlType” : “12”  
  } ],  
  “tableName” : “Avro\_File”  
}**

### **JSON to AVRO Conversion:**

**It is work, when we have the Schema file (AVSC) and the JSON file**

#### **Step-1: Convert the Json file into Avro file**

**java -jar <jar file with location> <keyword> <keyword for schema> <schema file with location> <json file with location> > /tmp/Avro\_File\_Converted\_File.avro**

### java -jar /usr/lib/avro/avro-tools-1.7.4.jar fromjson --schema-file /tmp/Avsc\_Format\_File.avsc /tmp/Json\_Format\_File.json > <avro file name with avro file location where the avro file will be stored>

#### **Step-2: View the Avro file which is converted from Json file**

### hadoop fs -cat file:/tmp/Avro\_Format\_Converted\_File.avro

**geouser@geouser:~$ hadoop fs -cat file:/tmp/Avro\_Format\_Converted\_File.avro  
Objavro.schema{“type”:”record”,”name”:”Avro\_File”,”doc”:”Sqoop import of Avro\_File”,”fields”:[{“name”:”User\_\_Name”,”type”:[“string”,”null”],”columnName”:”User\_\_Name”,”sqlType”:”12″},{“name”:”Product\_Id”,”type”:[“int”,”null”],”columnName”:”Product\_Id”,”sqlType”:”4″},{“name”:”Order\_Id”,”type”:[“string”,”null”],”columnName”:”Order\_Id”,”sqlType”:”12″},{“name”:”Delivery\_Date”,”type”:[“string”,”null”],”columnName”:”Delivery\_Date”,”sqlType”:”12″}],”tableName”:”Avro\_File”}avro.codenull1�얽w(�t\*pV��alaister briitoʺ�/mo86204126-09-2016anifa mohammed̺�/la86204114-09-2016piyush manishκ�/mo86203216-09-2016vijay karthikк�/wa86209**

***Hive json file with example:***

##### ***Hive:***

**Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.  It’s an open source project under the Apache Software Foundation, it’s a data warehouse software ecosystem in Hadoop. Which manage vast amount of structured data sets, by using HQL language; it’s similar to SQL.**



##### ***JSON:***

**JSON(JavaScript Object Notation) is a minimal, readable format for structuring data. It is used primarily to transmit data between a server and web application,  as an alternative to XML.  Square space uses JSON to store and organize site content created with the CMS.**

#### **Simple Json ( without using JsonSerDe ):**

***Step-1: Create simple json table with simple value***

**CREATE TABLE json\_table (json string);**

***Step-2: Load value into the json\_table***

**LOAD DATA LOCAL INPATH ‘/home/geouser/simple.json’ INTO TABLE json\_table;**

***Step-3: View the value in the table json\_table***

**SELECT \* from json\_table;**

**hive> CREATE TABLE json\_table (json string);  
OK  
Time taken: 0.353 seconds  
hive> LOAD DATA LOCAL INPATH ‘/home/geouser/Simple.json’ INTO TABLE  json\_table;  
Loading data to table default.json\_table  
Table default.json\_table stats: [numFiles=1, totalSize=456]  
OK  
Time taken: 0.64 seconds  
hive> SELECT \*  from json\_table;  
OK  
{“Name”:”Alaister briito”,”Id”:”1001265″,”Address”:{“City”:”Amsterdam”,”Country”:”Netherlands”,}}  
{“Name”:”Anifa mohammed “,”Id”:”1001285″,”Address”:{“City”:”Beijing”,”Country”:”China”,}}  
{“Name”:”Piyush manish  “,”Id”:”1001263″,”Address”:{“City”:”Cairo”,”Country”:”Egypt”,}}  
{“Name”:”Vijay karthik  “,”Id”:”1001295″,”Address”:{“City”:”Dublin”,”Country”:”Ireland”,}}  
{“Name”:”Narasingh rao  “,”Id”:”1001267″,”Address”:{“City”:”Havana”,”Country”:”Cuba”,}}  
Time taken: 0.191 seconds, Fetched: 6 row(s)**

### 

**Complex JSON FILE ( Using JsonSerDe ):**

##### **Step-1: First add json-serde-1.3.7- SNAPSHOT-jar-with-dependencies jar into hive**

**ADD JAR /home/geouser/Hive-JSON-Serde/json-serde/target/json-serde-1.3.6-SNAPSHOT-jar-with-dependencies.jar;**

**hive> ADD JAR /home/geouser/Hive-JSON-Serde/json-serde/target/json-serde-1.3.6-SNAPSHOT-jar-with-dependencies.jar; Added [/home/geouser/Hive-JSON-Serde/json-serde/target/json-serde-1.3.6-SNAPSHOT-jar-with-dependencies.jar] to class path Added resources: [/home/geouser/Hive-JSON-Serde/json-serde/target/json-serde-1.3.6-SNAPSHOT-jar-with-dependencies.jar]**

##### **Step-2: Create a json hive table**

**create table complex\_json (E\_id string, User\_details struct<ID:int, Username:string,  Name:string, Office\_Address:struct<Street:string, Area:string, Country:string>,  Orders:array<struct<ItemId:int, OrderDate:string>>> ) row format serde  ‘org.openx.data.jsonserde.JsonSerDe’ stored as textfile;**

**hive> create table complex\_json (  
> E\_id string,  
> User\_details struct<ID:int,  
> Username:string,  
> Name:string,  
> Office\_Address:struct<Street:string,  
> Area:string,  
> Country:string>,  
> Orders:array<struct<ItemId:int,  
> OrderDate:string>>>  
> )  
> row format serde ‘org.openx.data.jsonserde.JsonSerDe’ stored as textfile;  
OK  
Time taken: 0.466 seconds**

##### **Step-3: Complex.json (file)**

**{“E\_id”:GISS101,”User\_details”:{“Id”:101,”Username”:”briito123″,”Name”:”Alaister briito”,”Office\_Address”:{“Street”:”309 KENT STREET”,”Area”:”BENTLEY WA”,”Country”:”AUSTRALIA”},”Orders”:[{“ItemId”:6789,”OrderDate”:”11/07/2016″},{“ItemId”:4352,”OrderDate”:”16/07/2016″}]}}{“E\_id”:GISS102,”User\_details”:{“Id”:102,”Username”:”Anifa435″,”Name”:”Anifa mohammed”,”Office\_Address”:{“Street”:”117 RUSSELL DRIVE”,”Area”:”LONDON  WIP”,”Country”:”ENGLAND”},”Orders”:[{“ItemId”:6791,”OrderDate”:”12/07/2016″},{“ItemId”:4354,”OrderDate”:”15/07/2016″}]}}{“E\_id”:GISS103,”User\_details”:{“Id”:103,”Username”:”Piyu2300″,”Name”:”Piyush manish”,”Office\_Address”:{“Street”:”56 Church Street”,”Area”:”Hamilton HM”,”Country”:”BERMUDA”},”Orders”:[{“ItemId”:6795,”OrderDate”:”11/07/2016″},{“ItemId”:4356,”OrderDate”:”18/07/2016″}]}}{“E\_id”:GISS104,”User\_details”:{“Id”:104,”Username”:”Vithik”,”Name”:”Vijay karthik”,”Office\_Address”:{“Street”:”21-B 125252 Moskva”,”Area”:”MOSCOW”,”Country”:”RUSSIA”},”Orders”:[{“ItemId”:6815,”OrderDate”:”11/07/2016″},{“ItemId”:4358,”OrderDate”:”18/07/2016″}]}}{“E\_id”:GISS105,”User\_details”:{“Id”:105,”Username”:”Narasio”,”Name”:”Narasingh rao”,”Office\_Address”:{“Street”:”13 West Chang An Street”,”Area”:”100804 BEIJING”,”Country”:”CHINA”},”Orders”:[{“ItemId”:6825,”OrderDate”:”11/07/2016″},{“ItemId”:4360,”OrderDate”:”18/07/2016″}]}}**

##### **Step-4: Load the json file into hive json table**

**LOAD DATA LOCAL INPATH ‘/home/geouser/Complex\_json.json’ OVERWRITE INTO TABLE complex\_json;**

##### **Step-5: View the table**

**select \* from complex\_json;**

**hive>select \* from complex\_json;  
OK  
{“E\_id”:GISS101,”User\_details”:{“Id”:101,”Username”:”briito123″,”Name”:”Alaister briito”,”Office\_Address”:{“Street”:”309 KENT STREET”,”Area”:”BENTLEY WA”,”Country”:”AUSTRALIA”},”Orders”:[{“ItemId”:6789,”OrderDate”:”11/07/2016″},{“ItemId”:4352,”OrderDate”:”16/07/2016″}]}}  
{“E\_id”:GISS102,”User\_details”:{“Id”:102,”Username”:”Anifa435″,”Name”:”Anifa mohammed”,”Office\_Address”:{“Street”:”117 RUSSELL DRIVE”,”Area”:”LONDON  WIP”,”Country”:”ENGLAND”},”Orders”:[{“ItemId”:6791,”OrderDate”:”12/07/2016″},{“ItemId”:4354,”OrderDate”:”15/07/2016″}]}}  
{“E\_id”:GISS103,”User\_details”:{“Id”:103,”Username”:”Piyu2300″,”Name”:”Piyush manish”,”Office\_Address”:{“Street”:”56 Church Street”,”Area”:”Hamilton HM”,”Country”:”BERMUDA”},”Orders”:[{“ItemId”:6795,”OrderDate”:”11/07/2016″},{“ItemId”:4356,”OrderDate”:”18/07/2016″}]}}  
{“E\_id”:GISS104,”User\_details”:{“Id”:104,”Username”:”Vithik”,”Name”:”Vijay karthik”,”Office\_Address”:{“Street”:”21-B  
125252 Moskva”,”Area”:”MOSCOW”,”Country”:”RUSSIA”},”Orders”:[{“ItemId”:6815,”OrderDate”:”11/07/2016″},{“ItemId”:4358,”OrderDate”:”18/07/2016″}]}}  
{“E\_id”:GISS105,”User\_details”:{“Id”:105,”Username”:”Narasio”,”Name”:”Narasingh rao”,”Office\_Address”:{“Street”:”13 West Chang An Street”,”Area”:”100804 BEIJING”,”Country”:”CHINA”},”Orders”:[{“ItemId”:6825,”OrderDate”:”11/07/2016″},{“ItemId”:4360,”OrderDate”:”18/07/2016″}]}}  
Time taken: 0.315 seconds, Fetched: 3 row(s)**

##### **Step-6: View particuler Fields**

**SELECT E\_id, User\_details.Id, User\_details.Office\_Address.Country as country, User\_details.Orders[0].ItemId as order0id, User\_details.Orders[1].ItemId as order1id FROM complex\_json;**

**hive>SELECT E\_id,User\_details.Id, User\_details.Office\_Address.Country as country,  
>    User\_details.Orders[0].ItemId as order0id,  
>    User\_details.Orders[1].ItemId as order1id  
>    FROM complex\_json;  
OK  
GISS101 101 AUSTRALIA 6789 4352  
GISS102 102 ENGLAND   6791 4354  
GISS103 103 BERMUDA   6795 4356  
GISS104 104 RUSSIA    6815 4358  
GISS105 105 CHINA     6825 4360  
Time taken: 0.315 seconds, Fetched: 3 row(s)**

**Here are the steps to define the work flow and then execute it. This is with the assumption that  MySQL, Oozie and Hadoop have been installed, configured and work properly.**[**Here**](http://www.thecloudavenue.com/2013/10/installation-and-configuration-of.html)**are the instructions for installing and configuring Oozie.  
  
- The work flow requires more than 2 map slots in the cluster, so if the work flow is executed on a single node cluster the following has to be included in the `mapred-site.xml`.**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8** | **<property>**  **<name>mapred.tasktracker.map.tasks.maximum</name>**  **<value>4</value>**  **</property>**  **<property>**  **<name>mapred.tasktracker.reduce.tasks.maximum</name>**  **<value>4</value>**  **</property>** |

**- Create the file `oozie-clickstream-examples/input-data/clickstream/clickstream.txt` in HDFS with the below content. Note than the last record is an invalid record which is filtered by Pig when the work flow is executed. The first field is the userId and the second field is the site visited by the user.**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11** | **1,www.bbc.com**  **1,www.abc.com**  **1,www.gmail.com**  **2,www.cnn.com**  **2,www.eenadu.net**  **2,www.stackoverflow.com**  **2,www.businessweek.com**  **3,www.eenadu.net**  **3,www.stackoverflow.com**  **3,www.businessweek.com**  **A,www.thecloudavenue.com** |

**- Create a user table in MySQL**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7** | **CREATE TABLE user (**  **user\_id INTEGER NOT NULL PRIMARY KEY,**  **name CHAR(32) NOT NULL,**  **age INTEGER,**  **country VARCHAR(32),**  **gender CHAR(1)**  **);** |

**- And insert some data into it**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1**  **2**  **3** | **insert into user values (1,"Tom",20,"India","M");**  **insert into user values (2,"Rick",5,"India","M");**  **insert into user values (3,"Rachel",15,"India","F");** |

**- Extract the `oozie-4.0.0/oozie-sharelib-4.0.0.tar.gz` file from the Oozie installation folder and copy the**[**mysql-connector-java-\*.jar**](http://dev.mysql.com/downloads/connector/j/)**to the `share/lib/sqoop` folder. This jar is required for Sqoop to connect to the MySQL database and get the user data.  
  
- Copy the above mentioned `share` folder into HDFS.**[**Here**](http://blog.cloudera.com/blog/2012/12/how-to-use-the-sharelib-in-apache-oozie/)**is the significance of sharelib in Oozie. These are the common libraries which are used across different actions in Oozie.**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1** | **bin/hadoop fs -put /home/vm4learning/Code/share/ /user/vm4learning/share/** |

**- Create the work flow file in HDFS (oozie-clickstream-examples/apps/cs/workflow.xml). Note that the connect string for the Oozie has to be modified appropriately.**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52**  **53**  **54**  **55**  **56**  **57**  **58**  **59**  **60**  **61**  **62**  **63**  **64**  **65**  **66**  **67**  **68**  **69**  **70**  **71**  **72**  **73**  **74**  **75**  **76**  **77**  **78**  **79**  **80**  **81**  **82**  **8** | **<?xml version="1.0" encoding="UTF-8"?>**  **<workflow-app xmlns="uri:oozie:workflow:0.2" name="cs-wf-fork-join">**  **<start to="fork-node"/>**    **<fork name="fork-node">**  **<path start="sqoop-node" />**  **<path start="pig-node" />**  **</fork>**    **<action name="sqoop-node">**  **<sqoop xmlns="uri:oozie:sqoop-action:0.2">**  **<job-tracker>${jobTracker}</job-tracker>**  **<name-node>${nameNode}</name-node>**  **<prepare>**  **<delete path="${nameNode}/${examplesRootDir}/input-data/user"/>**  **</prepare>**    **<configuration>**  **<property>**  **<name>mapred.job.queue.name</name>**  **<value>${queueName}</value>**  **</property>**  **</configuration>**  **<command>import --connect jdbc:**[**mysql://localhost/clickstream**](mysql://localhost/clickstream) **--table user --target-dir ${examplesRootDir}/input-data/user -m 1</command>**  **</sqoop>**  **<ok to="joining"/>**  **<error to="fail"/>**  **</action>**    **<action name="pig-node">**  **<pig>**  **<job-tracker>${jobTracker}</job-tracker>**  **<name-node>${nameNode}</name-node>**  **<prepare>**  **<delete path="${nameNode}${examplesRootDir}/intermediate"/>**  **</prepare>**  **<configuration>**  **<property>**  **<name>mapred.job.queue.name</name>**  **<value>${queueName}</value>**  **</property>**  **<property>**  **<name>mapred.compress.map.output</name>**  **<value>true</value>**  **</property>**  **</configuration>**  **<script>filter.pig</script>**  **<param>INPUT=${examplesRootDir}/input-data/clickstream</param>**  **<param>OUTPUT=${examplesRootDir}/intermediate</param>**  **</pig>**  **<ok to="joining"/>**  **<error to="fail"/>**  **</action>**    **<join name="joining" to="hive-node"/>**    **<action name="hive-node">**  **<hive xmlns="uri:oozie:hive-action:0.2">**  **<job-tracker>${jobTracker}</job-tracker>**  **<name-node>${nameNode}</name-node>**  **<prepare>**  **<delete path="${nameNode}/${examplesRootDir}/finaloutput"/>**  **</prepare>**  **<configuration>**  **<property>**  **<name>mapred.job.queue.name</name>**  **<value>${queueName}</value>**  **</property>**  **</configuration>**  **<script>script.sql</script>**  **<param>CLICKSTREAM=${examplesRootDir}/intermediate/</param>**  **<param>USER=${examplesRootDir}/input-data/user/</param>**  **<param>OUTPUT=${examplesRootDir}/finaloutput</param>**  **</hive>**  **<ok to="end"/>**  **<error to="fail"/>**  **</action>**    **<kill name="fail">**  **<message>Sqoop failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>**  **</kill>**  **<end name="end"/>**  **</workflow-app>** |

**- Create the job.properties file in HDFS (oozie-clickstream-examples/apps/cs/job.properties).**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9** | **nameNode=hdfs://localhost:9000**  **jobTracker=localhost:9001**  **queueName=default**    **examplesRoot=oozie-clickstream-examples**  **examplesRootDir=/user/${user.name}/${examplesRoot}**    **oozie.use.system.libpath=true**  **oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/apps/cs** |

**- Create the Hive script file in HDFS (oozie-clickstream-examples/apps/cs/script.sql). The below mentioned query will find the top 3 url's visited by users whose age is less than 16.**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7** | **DROP TABLE clickstream;**  **CREATE EXTERNAL TABLE clickstream (userid INT, url STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '${CLICKSTREAM}';**    **DROP TABLE user;**  **CREATE EXTERNAL TABLE user (user\_id INT, name STRING, age INT, country STRING, gender STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '${USER}';**    **INSERT OVERWRITE DIRECTORY '${OUTPUT}' SELECT url,count(url) c FROM user u JOIN clickstream c ON (u.user\_id=c.userid) where u.age<16 group by url order by c DESC LIMIT 3;** |

**- Create the Pig script file in HDFS (oozie-clickstream-examples/apps/cs/filter.pig).**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

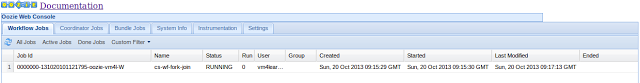
|  |  |
| --- | --- |
| **1**  **2**  **3** | **clickstream = load '$INPUT' using PigStorage(',') as (userid:int, url:chararray);**  **SPLIT clickstream INTO good\_records IF userid is not null,  bad\_records IF userid is null;**  **STORE good\_records into '$OUTPUT';** |

**- Execute the Oozie workflow as below. Note that the `job.properties` file should be present in the local file system and not in HDFS.**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1** | **bin/oozie job -oozie http://localhost:11000/oozie -config /home/vm4learning/Code/oozie-clickstream-examples/apps/cs/job.properties -run** |

**- Initially the job will be in the `RUNNING` state and finally will reach the `SUCCEEDED` state. The progress of the work flow can be monitored from Oozie console at http://localhost:11000/oozie/.**



**- The output should appear as below in the `oozie-clickstream-examples/finaloutput/000000\_0` file in HDFS.**

[**?**](http://www.thecloudavenue.com/2013/10/executing-oozie-workflow-with-pig-hive.html)

|  |  |
| --- | --- |
| **1**  **2**  **3** | **www.businessweek.com 2**  **www.eenadu.net 2**  **www.stackoverflow.com 2** |

**Here are some final thoughts on Oozie.  
  
- It's better to test the individual actions like Hive, Pig and Sqoop independent of Ooize and later integrate them in the Oozie work flow.  
  
- The Oozie error messages very cryptic and the MapReduce log files need to be looked to figure out the actual error.  
  
- The Web UI which comes with Oozie is very rudimentary and clumsy, need to look into some of the alternatives.  
  
- The XML for creating the work flows is very verbose and is very error prone. Any UI for creating workflows for Oozie would be very helpful.  
  
Will look into the alternatives for some of the above problems mentioned in a future blog entry.**

**In this article, I am going to walk over easy to follow examples and show how to create Hive User Defined Functions (UDF) and User Defined Aggregate Functions (UDAFs), package into a JAR, and test in Hive CLI. So let’s begin.**

**In my Sqoop import article, I imported a customers table. Similarly, I have imported an orders table, which I used in my**[**Hive Joins**](https://letsdobigdata.wordpress.com/2016/03/02/hive-joins/)**article. Also, I am using a dummy table for UDF verification. You can find the relevant Sqoop commands on**[**GitHub**](https://github.com/hardik-pandya/HiveUDFs/blob/master/sqoop/sqoop_commands.txt)**.**

**Hive supports of a lot of built-in SQL-like functions in HiveQL. But just in case, if there is a need to write your own UDF, no one is stopping you.**

## **UDF (User Defined Function)**

**Here I am going to show how to write a simple “trim-like” function called “Strip” – of course, you can write something fancier, but my goal here is to take away something in a short amount of time. So let’s begin.**

## **How to Write a UDF function in Hive?**

1. **Create a Java class for the User Defined Function which extends ora.apache.hadoop.hive.sq.exec.UDF and implements more than one evaluate() methods. Put in your desired logic and you are almost there.**
2. **Package your Java class into a JAR file (I am using Maven)**
3. **Go to Hive CLI, add your JAR, and verify your JARs is in the Hive CLI classpath**
4. **CREATE TEMPORARY FUNCTION in Hive which points to your Java class**
5. **Use it in Hive SQL and have fun!**

**There are better ways to do this, by writing your own GenericUDF to deal with non-primitive types like arrays and maps – but I am not going to cover it in this article.**

**I will go into detail for each one.**

## **Create Java Class for a User Defined Function**

**As you can see below I am calling my Java class “Strip”. You can call it anything, but the important point is that it extends the UDF interface and provides two evaluate() implementations.**

**evaluate(Text str, String stripChars) - will trim specified characters in stripChars from first argument str.**

**evaluate(Text str) - will trim leading and trailing spaces.**

**package org.hardik.letsdobigdata;**

**import org.apache.commons.lang.StringUtils;**

**import org.apache.hadoop.hive.ql.exec.UDF;**

**import org.apache.hadoop.io.Text;**

**public class Strip extends UDF {**

**private Text result = new Text();**

**public Text evaluate(Text str, String stripChars) {**

**if(str == null) {**

**return null;**

**}**

**result.set(StringUtils.strip(str.toString(), stripChars));**

**return result;**

**}**

**public Text evaluate(Text str) {**

**if(str == null) {**

**return null;**

**}**

**result.set(StringUtils.strip(str.toString()));**

**return result;**

**}**

**}**

## **Package Your Java Class into a JAR**

**There is a pom.xml attached in GitHub. Please make sure you have Maven installed. If you are working with a GitHub clone, go to your shell:**

**$ cd HiveUDFs**

**and run "mvn clean package". This will create a JAR file which contains our UDF class. Copy the JAR's path.**

## **Go to the Hive CLI and Add the UDF JAR**

**hive> ADD /home/cloudera/workspace/HiveUDFs/target/HiveUDFs-0.0.1-SNAPSHOT.jar;**

**Added [/home/cloudera/workspace/HiveUDFs/target/HiveUDFs-0.0.1-SNAPSHOT.jar] to class path**

**Added resources: [/home/cloudera/workspace/HiveUDFs/target/HiveUDFs-0.0.1-SNAPSHOT.jar]**

## **Verify JAR is in Hive CLI Classpath**

**You should see your jar in the list.**

**hive> list jars;**

**/usr/lib/hive/lib/hive-contrib.jar**

**/home/cloudera/workspace/HiveUDFs/target/HiveUDFs-0.0.1-SNAPSHOT.jar**

## **Create Temporary Function**

**It does not have to be a temporary function. You can create your own function, but just to keep things moving, go ahead and create a temporary function.**

**You may want to add ADD JAR and CREATE TEMPORARY FUNCTION to .hiverc file so they will execute at the beginning of each Hive session.**

## **UDF Output**

**The first query strips ‘ha’ from string ‘hadoop’ as expected (2 argument evaluate() in code). The second query strips trailing and leading spaces as expected.**

**hive> CREATE TEMPORARY FUNCTION STRIP AS 'org.hardik.letsdobigdata.Strip';**

**hive> select strip('hadoop','ha') from dummy;**

**OK**

**doop**

**Time taken: 0.131 seconds, Fetched: 1 row(s)**

**hive> select strip(' hiveUDF ') from dummy;**

**OK**

**hiveUDF**

**If you have made this far, congratulations! That was our UDF in action! You can follow the code on**[**GitHub**](https://github.com/hardik-pandya/HiveUDFs/blob/master/src/main/java/org/hardik/letsdobigdata/Strip.java)**.**

## **UDAF (User Defined Aggregated Function)**

**Now, equipped with our first UDF knowledge, we will move to a next step. When we say aggregation, COUNT, AVG, SUM, MIN, and MAX come to our mind.**

**I am picking a very simple aggregation function AVG/MEAN, where I am going to work with the “orders” table imported using**[**Sqoop**](https://github.com/hardik-pandya/HiveUDFs/blob/master/sqoop/sqoop_commands.txt)**. Once you import it into Hive, it will look like the below (or you can use LOAD DATA INPATH – it is totally up to you.)**

**hive> select \* from orders;**

**OK**

**orders.order\_id orders.order\_date orders.customer\_id orders.amount**

**101 2016-01-01 7 3540**

**102 2016-03-01 1 240**

**103 2016-03-02 6 2340**

**104 2016-02-12 3 5000**

**105 2016-02-12 3 5500**

**106 2016-02-14 9 3005**

**107 2016-02-14 1 20**

**108 2016-02-29 2 2000**

**109 2016-02-29 3 2500**

**110 2016-02-27 1 200**

**The goal of our UDAF is to find the average amount of orders for all customers in the orders table.**

**We are looking for Query:**

**SELECT CUSTOMER\_ID, AVG(AMOUNT) FROM ORDERS GROUP BY CUSTOMER\_ID;**

**I am going to replace AVG function with “MEAN” function**

**But before I begin, let’s stop and think as we are entering the MapReduce world. One of the bottlenecks you want to avoid is moving too much data from the Map to the Reduce phase.**

**An aggregate function is more difficult to write than a regular UDF. Values are aggregated in chunks (across many maps or many reducers), so the implementation has to be capable of combining partial aggregations into final results.**

**At a high-level, there are two parts to implementing a Generic UDAF:**

1. **evaluator – The evaluator class actually implements the UDAF logic.**
2. **resolver – The resolver class handles type checking and operator overloading (if you want it), and helps Hive find the correct evaluator class for a given set of argument types.**

**We are not creating a GenericUDAF. We are creating our one-time aggregation function, so we do not have to worry about a resolver. I am planning write on GenericUDF/GenericUDAF, though. It may be some other day, but soon. :)**

## **How to Write UDAF?**

1. **Create a Java class which extends org.apache.hadoop.hive.ql.exec.hive.UDAF;**
2. **Create an inner class which implements UDAFEvaluator;**
3. **Implement five methods ()**
   1. **init() – The init() method initializes the evaluator and resets its internal state. We are using new Column() in the code below to indicate that no values have been aggregated yet.**
   2. **iterate() – this method is called every time there is a new value to be aggregated. The evaulator should update its internal state with the result of performing the aggregation (we are doing sum – see below). We return true to indicate that the input was valid.**
   3. **terminatePartial() – this method is called when Hive wants a result for the partial aggregation. The method must return an object that encapsulates the state of the aggregation.**
   4. **merge() – this method is called when Hive decides to combine one partial aggregation with another.**
   5. **terminate() – this method is called when the final result of the aggregation is needed.**
4. **Compile and package the JAR**
5. **CREATE TEMPORARY FUNCTION in hive CLI**
6. **Run Aggregation Query and Verify Output!!!**

### **MeanUDAF.java**

**package org.hardik.letsdobigdata;**

**import org.apache.commons.logging.Log;**

**import org.apache.commons.logging.LogFactory;**

**import org.apache.hadoop.hive.ql.exec.Description;**

**import org.apache.hadoop.hive.ql.exec.UDAF;**

**import org.apache.hadoop.hive.ql.exec.UDAFEvaluator;**

**import org.apache.hadoop.hive.ql.metadata.HiveException;**

**import org.hardik.letsdobigdata.MeanUDAF.MeanUDAFEvaluator.Column;**

**@Description(name = "Mean", value = "\_FUNC(double) - computes mean", extended = "select col1, MeanFunc(value) from table group by col1;")**

**public class MeanUDAF extends UDAF {**

**// Define Logging**

**static final Log LOG = LogFactory.getLog(MeanUDAF.class.getName());**

**public static class MeanUDAFEvaluator implements UDAFEvaluator {**

**/\*\***

**\* Use Column class to serialize intermediate computation**

**\* This is our groupByColumn**

**\*/**

**public static class Column {**

**double sum = 0;**

**int count = 0;**

**}**

**private Column col = null;**

**public MeanUDAFEvaluator() {**

**super();**

**init();**

**}**

**// A - Initalize evaluator - indicating that no values have been**

**// aggregated yet.**

**public void init() {**

**LOG.debug("Initialize evaluator");**

**col = new Column();**

**}**

**// B- Iterate every time there is a new value to be aggregated**

**public boolean iterate(double value) throws HiveException {**

**LOG.debug("Iterating over each value for aggregation");**

**if (col == null)**

**throw new HiveException("Item is not initialized");**

**col.sum = col.sum + value;**

**col.count = col.count + 1;**

**return true;**

**}**

**// C - Called when Hive wants partially aggregated results.**

**public Column terminatePartial() {**

**LOG.debug("Return partially aggregated results");**

**return col;**

**}**

**// D - Called when Hive decides to combine one partial aggregation with another**

**public boolean merge(Column other) {**

**LOG.debug("merging by combining partial aggregation");**

**if(other == null) {**

**return true;**

**}**

**col.sum += other.sum;**

**col.count += other.count;**

**return true;**

**}**

**// E - Called when the final result of the aggregation needed.**

**public double terminate(){**

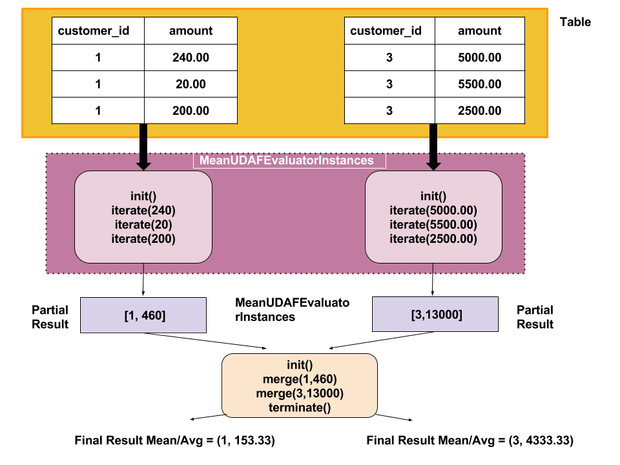
**LOG.debug("At the end of last record of the group - returning final result");**

**return col.sum/col.count;**

**}**

**}**

**}**



## **Package and ADD JAR**

**hive> ADD JAR /home/cloudera/workspace/HiveUDFs/target/HiveUDFs-0.0.1-SNAPSHOT.jar;  
Added [/home/cloudera/workspace/HiveUDFs/target/HiveUDFs-0.0.1-SNAPSHOT.jar] to class path  
Added resources: [/home/cloudera/workspace/HiveUDFs/target/StudentCourseMRJob-0.0.1-SNAPSHOT.jar]**

### **CREATE FUNCTION in HIVE**

**hive> CREATE TEMPORARY FUNCTION MeanFunc AS 'org.hardik.letsdobigdata.MeanUDAF';**

**OK**

## **Verify Output**

**Execute the below group by query. Our function is called MeanFunc**

**hive> select customer\_id, MeanFunc(amount) from orders group by customer\_id;**

**FAILED: SemanticException [Error 10001]: Line 1:42 Table not found 'orders'**

**hive> use sqoop\_workspace;**

**OK**

**Time taken: 0.247 seconds**

**hive> select customer\_id, MeanFunc(amount) from orders group by customer\_id;**

**Query ID = cloudera\_20160302030202\_fb24b7c1-4227-4640-afb9-4ccd29bd735f**

**Total jobs = 1**

**Launching Job 1 out of 1**

**Number of reduce tasks not specified. Estimated from input data size: 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:**

**set mapreduce.job.reduces=<number>**

**Starting Job = job\_1456782715090\_0020, Tracking URL = http://quickstart.cloudera:8088/proxy/application\_1456782715090\_0020/**

**Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job\_1456782715090\_0020**

**Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1**

**2016-03-02 03:03:16,703 Stage-1 map = 0%, reduce = 0%**

**2016-03-02 03:03:53,241 Stage-1 map = 50%, reduce = 0%, Cumulative CPU 3.31 sec**

**2016-03-02 03:03:55,593 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.9 sec**

**2016-03-02 03:04:09,201 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.18 sec**

**MapReduce Total cumulative CPU time: 6 seconds 180 msec**

**Ended Job = job\_1456782715090\_0020**

**MapReduce Jobs Launched:**

**Stage-Stage-1: Map: 2 Reduce: 1 Cumulative CPU: 6.18 sec HDFS Read: 12524 HDFS Write: 77 SUCCESS**

**Total MapReduce CPU Time Spent: 6 seconds 180 msec**

**OK**

**1 153.33333333333334**

**2 2000.0**

**3 4333.333333333333**

**6 2340.0**

**7 3540.0**

**9 3005.0**

**Time taken: 72.172 seconds, Fetched: 6 row(s)**

**Verify Individual customer\_id : As you can see, group by value matches – you can cross check manually. Thanks for your time and reading my blog – hope this is helpful!!!**

**hive> select \* from orders where customer\_id = 1;**

**OK**

**102 2016-03-01 1 240**

**107 2016-02-14 1 20**

**110 2016-02-27 1 200**

**Time taken: 0.32 seconds, Fetched: 3 row(s)**

**hive> select \* from orders where customer\_id = 2;**

**OK**

**108 2016-02-29 2 2000**

**Time taken: 0.191 seconds, Fetched: 1 row(s)**

**hive> select \* from orders where customer\_id = 3;**

**OK**

**104 2016-02-12 3 5000**

**105 2016-02-12 3 5500**

**109 2016-02-29 3 2500**

**Time taken: 0.093 seconds, Fetched: 3 row(s)**

**Find the perfect platform for a scalable self-service model to manage Big Data workloads in the Cloud.**[**Download the free O'Reilly eBook to learn more**](https://dzone.com/go?i=239230&u=http%3A%2F%2Fgo.qubole.com%2FLP---DataOps---Book-Offer.html%3Futm_campaign%3DCS-Dzone%26utm_content%3DDataOps_Book%26utm_medium%3DPartnerResource%26utm_term%3DQ317)**.**

### **Like This Article? Read More From DZone**

**Topics:**



[**Performance Analysis: The Cost of Stack Overflow Indexes**](https://dzone.com/articles/performance-analysis-the-cost-of-stackoverflow-ind?fromrel=true)



[**MapReduce Succinctly: An Instructional Implementation From Scratch of MapReduce Using Scala and Akka**](https://dzone.com/articles/mapreduce-succinctly-an-instructional-implementati?fromrel=true)



[**Working in Parallel: On the Complications of Parallel Algorithm Design**](https://dzone.com/articles/working-in-parallel?fromrel=true)



[**Free DZone Refcard**](https://dzone.com/refcardz/data-warehousing?fromrel=true)

[**Data Warehousing**](https://dzone.com/refcardz/data-warehousing?fromrel=true)

**HIVE ,MAP REDUCE**

**Like**[**(3)**](https://dzone.com/articles/writing-custom-hive-udf-andudaf)

**Comme**

### **Oozie Example: Hive Actions**

**Running Hive through Oozie is pretty straight-forward and it's getting much simpler day-by-day. 1st time when I used it(old versions) I faced some issues mostly related to classpath though I resolved them. But when I used the recent versions (Hive 0.8+, Oozie 3.3.2+), I only faced 1 or 2 issues at most.**

**In this example, I'm going to execute a very simple Hive script through Oozie. I have a Hive table "temp" and it's currently empty. The script will load some data from HDFS to that specific hive table.**

|  |  |
| --- | --- |
|  | **hive> describe temp;** |
|  | **OK** |
|  | **id bigint None** |
|  | **name string None** |
|  | **Time taken: 0.261 seconds, Fetched: 2 row(s)** |

[**view raw**](https://gist.github.com/tmusabbir/7019600/raw/17dce78d5e033b5164dbab14aee85c49c0aa2930/hive-cmd)**[hive-cmd](https://gist.github.com/tmusabbir/7019600" \l "file-hive-cmd) hosted with ❤ by**[**GitHub**](https://github.com/)

**And here is the content of the script.hql:**

|  |  |
| --- | --- |
|  | **load data inpath '${INPUT\_PATH}' overwrite into table temp;** |

[**view raw**](https://gist.github.com/tmusabbir/7019600/raw/17dce78d5e033b5164dbab14aee85c49c0aa2930/script.hql)**[script.hql](https://gist.github.com/tmusabbir/7019600" \l "file-script-hql) hosted with ❤ by**[**GitHub**](https://github.com/)

**Now you need to setup your Oozie workflow app folder. You need one very important file to execute Hive action through Oozie which is hive-site.xml. When Oozie executes a Hive action, it needs Hive's configuration file. You can provide multiple configurations file in a single action. You can find your Hive configuration file from "/etc/hive/conf.dist/hive-site.xml" (default location). Copy that file and put it inside your workflow application path in HDFS. Here is the list of files that I have in my Oozie Hive action's workflow application folder.**

|  |  |
| --- | --- |
|  | **[ambari-qa@ip-10-0-0-51 hive-oozie]$ hadoop fs -lsr /user/ambari-qa/example/hive-oozie** |
|  | **-rwx------ 3 ambari-qa hdfs 3383 2013-03-01 21:28 /user/ambari-qa/example/hive-oozie/hive-site.xml** |
|  | **-rwx------ 3 ambari-qa hdfs 24 2013-03-01 21:29 /user/ambari-qa/example/hive-oozie/script.hql** |
|  | **-rwx------ 3 ambari-qa hdfs 1852 2013-03-01 21:29 /user/ambari-qa/example/hive-oozie/workflow.xml** |

[**view raw**](https://gist.github.com/tmusabbir/7019600/raw/17dce78d5e033b5164dbab14aee85c49c0aa2930/folder)**[folder](https://gist.github.com/tmusabbir/7019600" \l "file-folder) hosted with ❤ by**[**GitHub**](https://github.com/)

**And here is my workflow.xml file:**

|  |  |
| --- | --- |
|  | **<workflow-app xmlns="uri:oozie:workflow:0.4" name="hive-wf"****>** |
|  | **<start to="hive-node"****/>** |
|  |  |
|  | **<action name="hive-node"****>** |
|  | **<hive xmlns="uri:oozie:hive-action:0.2"****>** |
|  | **<job-tracker>${jobTracker}</job-tracker****>** |
|  | **<name-node>${nameNode}</name-node****>** |
|  | **<job-xml>hive-site.xml</job-xml****>** |
|  | **<configuration****>** |
|  | **<property****>** |
|  | **<name>mapred.job.queue.name</name****>** |
|  | **<value>${queueName}</value****>** |
|  | **</property****>** |
|  | **</configuration****>** |
|  | **<script>script.hql</script****>** |
|  | **<param>INPUT\_PATH=${inputPath}</param****>** |
|  | **</hive****>** |
|  | **<ok to="end"****/>** |
|  | **<error to="fail"****/>** |
|  | **</action****>** |
|  |  |
|  | **<kill name="fail"****>** |
|  | **<message>Hive failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message****>** |
|  | **</kill****>** |
|  | **<end name="end"****/>** |
|  | **</workflow-app>** |

[**view raw**](https://gist.github.com/tmusabbir/7019600/raw/17dce78d5e033b5164dbab14aee85c49c0aa2930/workflow.xml)**[workflow.xml](https://gist.github.com/tmusabbir/7019600" \l "file-workflow-xml) hosted with ❤ by**[**GitHub**](https://github.com/)

**Look at the <job-xml> tag, since I'm putting hive-site.xml in my application path, so I'm just passing the file name not the whole location. If you want to keep that file in some other location of your HDFS, then you can pass the whole HDFS path there too. In older version of Hive, user had to provide the hive-default.xml file by using property key oozie.hive.defaults while running Oozie Hive action, but from now on (Hive 0.8+) it's not required anymore.**

**Here I'm using another tag <param>, which is not required but I'm using it just to show how to pass parameter among hive script, job properties and workflow. If you are using any parameter variable inside your hive script, it needs to pass through the hive action. So you can do, either:**

* **<param>INPUT\_PATH=${inputPath}</param> (where inputPath can be passed through job properties) , Or**
* **<param>INPUT\_PATH=/user/ambari-qa/input/temp</param>**

**Inside my HDFS, "/hive-input/temp" folder contains files which need to be loaded to Hive table:**

|  |  |
| --- | --- |
|  | **[ambari-qa@ip-10-0-0-51 hive-oozie]$ hadoop fs -lsr /user/ambari-qa/example/hive-input/temp** |
|  | **-rwx------ 3 ambari-qa hdfs 3383 2013-03-01 22:18 /user/ambari-qa/example/hive-input/temp/r-000000** |
|  | **-rwx------ 3 ambari-qa hdfs 24 2013-03-01 22:19 /user/ambari-qa/example/hive-input/temp/r-000001** |
|  | **-rwx------ 3 ambari-qa hdfs 1852 2013-03-01 22:19 /user/ambari-qa/example/hive-input/temp/r-000002** |

[**view raw**](https://gist.github.com/tmusabbir/7019600/raw/17dce78d5e033b5164dbab14aee85c49c0aa2930/hive-dir)**[hive-dir](https://gist.github.com/tmusabbir/7019600" \l "file-hive-dir) hosted with ❤ by**[**GitHub**](https://github.com/)

**And here is my job.properties file:**

|  |  |
| --- | --- |
|  | **nameNode****=hdfs://ip-10-0-0-52:8020** |
|  | **jobTracker****=ip-10-0-0-53:50300** |
|  | **queueName****=default** |
|  | **exampleRoot****=example** |
|  |  |
|  | **oozie.use.system.libpath****=true** |
|  | **oozie.libpath****=/user/oozie/share/lib** |
|  |  |
|  | **oozie.wf.application.path****=${nameNode}/user/${user.name}/${exampleRoot}/hive-oozie** |
|  |  |
|  | **inputPath=${nameNode}/user/${user.name}/${exampleRoot}/hive-input/temp/\*** |

[**view raw**](https://gist.github.com/tmusabbir/7019600/raw/17dce78d5e033b5164dbab14aee85c49c0aa2930/job.properties)**[job.properties](https://gist.github.com/tmusabbir/7019600" \l "file-job-properties) hosted with ❤ by**[**GitHub**](https://github.com/)

**That's it! You can now run your Hive workflow by executing this on the client node:**

|  |  |
| --- | --- |
|  | **oozie job -oozie http://ip-10-0-0-51:11000/oozie -config job.properties -run** |

[**view raw**](https://gist.github.com/tmusabbir/7018058/raw/744c231e32e0724503bd243e98ebf870fe0be115/oozie)**[oozie](https://gist.github.com/tmusabbir/7018058" \l "file-oozie) hosted with ❤ by**[**GitHub**](https://github.com/)

**Two common issues:  
You might face some issues if the required jar files are not present inside "/user/oozie/share/lib/hive" folder (HDFS). One of the commons issue is not having the hcatalog\* jar files in that folder. In that case you will see something like this in the log:**

|  |  |
| --- | --- |
|  | **Failing Oozie Launcher, Main class [org.apache.oozie.action.hadoop.HiveMain], main() threw exception, org.apache.hadoop.hive.ql.metadata.HiveException: java.lang.ClassNotFoundException: org.apache.hcatalog.security.HdfsAuthorizationProvider** |
|  |  |
|  | **java.lang.RuntimeException: org.apache.hadoop.hive.ql.metadata.HiveException: java.lang.ClassNotFoundException: org.apache.hcatalog.security.HdfsAuthorizationProvider** |
|  | **at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:280)** |
|  | **at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:670)** |
|  | **at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:614)** |
|  | **at org.apache.oozie.action.hadoop.HiveMain.runHive(HiveMain.java:261)** |

[**view raw**](https://gist.github.com/tmusabbir/7019600/raw/17dce78d5e033b5164dbab14aee85c49c0aa2930/common-issue)**[common-issue](https://gist.github.com/tmusabbir/7019600" \l "file-common-issue) hosted with ❤ by**[**GitHub**](https://github.com/)

**In that case, you need to manually copy those required jar files into that folder. You can do that by following:**

|  |  |
| --- | --- |
|  | **[root@ip-10-0-0-51 ~]# su -l oozie -c "hadoop fs -put /usr/lib/hcatalog/share/hcatalog/hcatalog-core-0.11.0.1.3.2.0-111.jar /user/oozie/share/lib/hive"** |
|  | **[root@ip-10-0-0-51 ~]# su -l oozie -c "hadoop fs -put /usr/lib/hcatalog/share/hcatalog/hcatalog-server-extensions-0.11.0.1.3.2.0-111.jar /user/oozie/share/lib/hive"** |
|  | **[root@ip-10-0-0-51 ~]# su -l oozie -c "hadoop fs -chmod 775 /user/oozie/share/lib/hive/hcatalog-core-0.11.0.1.3.2.0-111.jar"** |
|  | **[root@ip-10-0-0-51 ~]# su -l oozie -c "hadoop fs -chmod 775 /user/oozie/share/lib/hive/hcatalog-server-extensions-0.11.0.1.3.2.0-111.jar"** |

[**view raw**](https://gist.github.com/tmusabbir/7019600/raw/17dce78d5e033b5164dbab14aee85c49c0aa2930/issue-res)**[issue-res](https://gist.github.com/tmusabbir/7019600" \l "file-issue-res) hosted with ❤ by**[**GitHub**](https://github.com/)

**Another common issue you might face is:**

**SemanticException [Error 10001]: Table not found**

**Even though you can see your table is exists, you might see this error when running through Oozie. Most of the time it happens when your Hive is not properly pointing to the right metastore. Most of the time, the problem goes away when you copy the correct hive-site.xml into hive lib folder inside HDFS. Make sure you check your hive-site.xml file to see all properties are correctly set. Like,  "hive.metastore.uris", "javax.jdo.option.ConnectionUR", "javax.jdo.option.ConnectionDriverName". But me and other users (**[**Hive action failing in oozie**](http://hadoop-common.472056.n3.nabble.com/Hive-Action-Failing-in-Oozie-td4004208.html)**) also found out that the above error message is ambiguous and doesn't give much insight. If the expected jar files are not present in the share lib folder, hive also throws the same error message! So be careful about what you have in the classpath when running hive through Oozie.**

**Note: For privacy purpose, I had to modify several lines on this post from my original post. So if you find something is not working or facing any issues, please do not hesitate to contact me.**

**Here a sample XML is as following  
  
Company.xml**

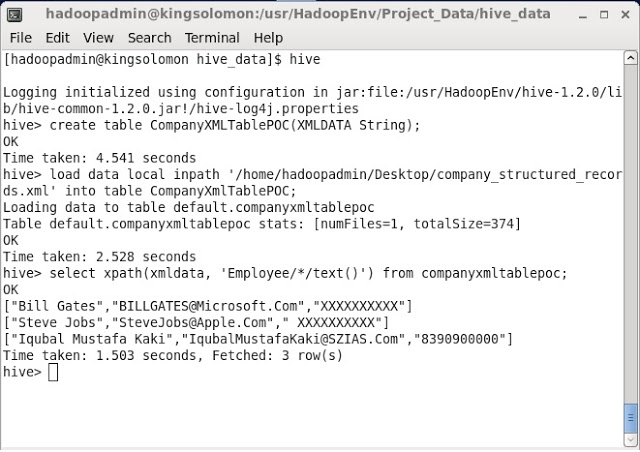
|  |
| --- |
| **<Employee>**  **<Name>Bill Gates</Name>**  **<Email>BILLGATES@Microsoft.Com</Email>**  **<Contact>XXXXXXXXXX</Contact>**  **</Employee>**  **<Employee>**  **<Name>Steve Jobs</Name>**  **<Email>SteveJobs@Apple.Com</Email>**  **<Contact> XXXXXXXXXX</Contact>**  **</Employee>**  **<Employee>**  **<Name>Iqubal Mustafa Kaki</Name>**  **<Email>IqubalMustafaKaki@SZIAS.Com</Email>**  **<Contact>8390900000</Contact>**  **</Employee>** |

**Here we will load XML data into hive table by using XPATH() .**

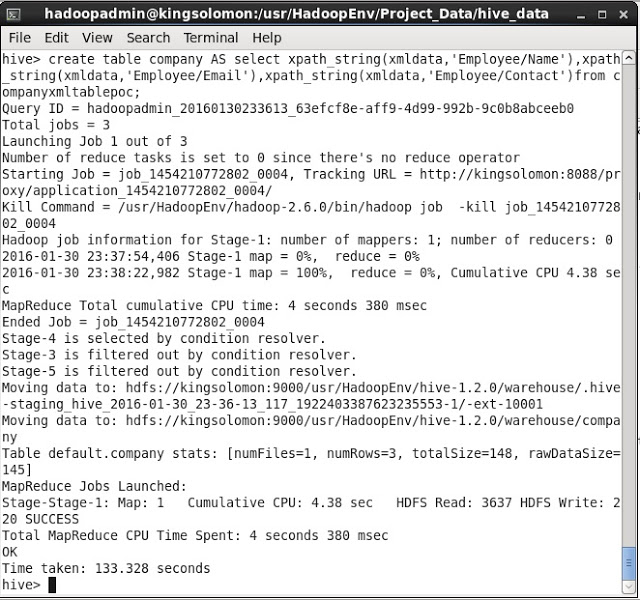
**Thus we will explore XPATH() which will used for converting XML data to String Array.**

**Steps for loading XML Data into Hive Table**

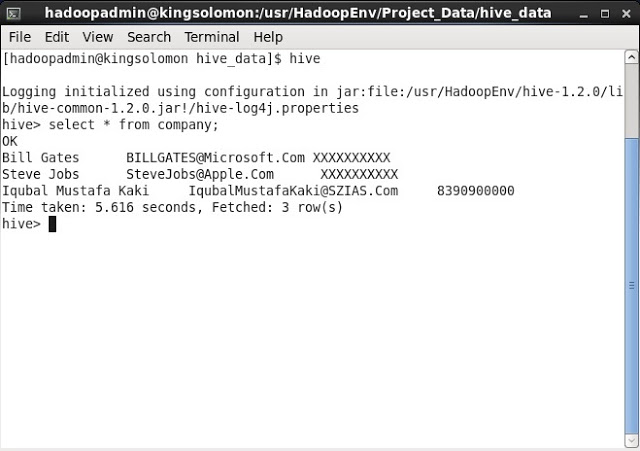
**Step:1 Structured the XML, by executing below command  
  
$ cat Company.xml | tr -d '&' | tr '\n' ' ' | tr '\r' ' ' | sed 's|</Employee>|</Employee>\n|g'| grep -v '^\s\*$' > company\_structured\_records.xml  
  
  
Now we have company\_structured\_records.xml file for the further operation.  
  
Step:2 Create a Hive table and load the XML file into the table  
  
hive> create table CompanyXMLTablePOC(XMLDATA String);  
  
hive> load data local inpath '/home/hadoopadmin/Desktop/company\_structured\_records.xml' into table CompanyXmlTablePOC;  
  
  
Step:3 Convert the XML Data into Array Format Using XPATH()  
  
hive> select xpath(xmldata, 'Employee/\*/text()') from CompanyXmlTablePOC;  
  
you will get the result result as below.  
  
OK  
["Bill Gates","BILLGATES@Microsoft.Com","XXXXXXXXXX"]  
["Steve Jobs","SteveJobs@Apple.Com"," XXXXXXXXXX"]  
["Iqubal Mustafa Kaki","IqubalMustafaKaki@SZIAS.Com","8390900000"]  
Time taken: 1.503 seconds, Fetched: 3 row(s)**



**Step:4  Create the HIVE Table and Insert Data from Above Created Table  
  
hive> create table company AS select xpath\_string(xmldata,'Employee/Name'),xpath\_string(xmldata,'Employee/Email'),xpath\_string(xmldata,'Employee/Contact')from companyxmltablepoc;**



**hive> select \* from company;  
OK  
Bill Gates BILLGATES@Microsoft.Com XXXXXXXXXX  
Steve Jobs SteveJobs@Apple.Com  XXXXXXXXXX  
Iqubal Mustafa Kaki IqubalMustafaKaki@SZIAS.Com 8390900000  
  
Time taken: 5.616 seconds, Fetched: 3 row(s)**



DistributedCache is a facility provided by the Map-Reduce framework to cache files needed by applications. Once you cache a file for your job, hadoop framework will make it available on each and every data nodes (in file system, not in memory) where you map/reduce tasks are running. Then you can access the cache file as local file in your Mapper Or Reducer job. Now you can easily read the cache file and populate some collection (e.g Array, Hashmap etc.) in your code.

Refer <https://hadoop.apache.org/docs/r2.2.0/api/org/apache/hadoop/filecache/DistributedCache.html>

Let me know if still you have some questions.

You can read the cache file as local file in your UDF code. After reading the file using JAVA APIs just populate any collection (In memory).

Refere URL <http://www.lichun.cc/blog/2013/06/use-a-lookup-hashmap-in-hive-script/>

To better understand how partitioning and bucketing works, please take a look at how data is stored in hive. Let's say you have a table

CREATE TABLE mytable ( name string, city string, employee\_id int ) PARTITIONED BY (year STRING, month STRING, day STRING) CLUSTERED BY (employee\_id) INTO 256 BUCKETS

You insert some data into a partition for 2015-12-02. Hive will then store data in a directory hierarchy, such as:

/user/hive/warehouse/mytable/y=2015/m=12/d=02

As such, it is important to be careful when **partitioning**. As a general rule of thumb, when choosing a field for partitioning, the field should not have a high cardinality - the term 'cardinality' refers to the number of possible values a field can have. For instance, if you have a 'country' field, the countries in the world are about 300, so cardinality would be ~300. For a field like 'timestamp\_ms', which changes every millisecond, cardinality can be billions. The cardinality of the field relates to the number of directories that could be created on the file system. As an example, if you partition by employee\_id and you have millions of employees, you may end up having millions of directories in your file system.

**Clustering**, aka bucketing, on the other hand, will result in a fixed number of files, since you specify the number of buckets. What hive will do is to take the field, calculate a hash and assign a record to that bucket.

**FAQ**  
*What happens if you use e.g. 256 buckets and the field you're bucketing on has a low cardinality (for instance, it's a US state, so can be only 50 different values?*   
You'll have 50 buckets with data, and 206 buckets with no data.  
  
*Can partitions dramatically cut the amount of data that is being queried?*  
In the example table, if you want to query only from a certain date forward, the partitioning by year/month/day is going to dramatically cut the amount of IO.  
  
*Can bucketing can speed up joins with other tables that have exactly the same bucketing?*  
In the above example, if you're joining two tables on the same employee\_id, hive can do the join bucket by bucket (even better if they're already sorted by employee\_id since it's going to do a mergesort which works in linear time).

So, **bucketing** works well when the field has **high cardinality** and data is **evenly distributed** among buckets. **Partitioning** works best when the **cardinality** of the partitioning field is **not too high.**

Also, you can partition on multiple fields, with an order (year/month/day is a good example), while you can bucket on only one field.  
  
We strongly recommend not to use Bucketing.

Windowing allows you to create a window on a set of data further allowing aggregation surrounding that data. Windowing in Hive is introduced from Hive 0.11. In this blog, we will be giving a demo on the windowing functions available in Hive.

Windowing in Hive includes the following functions

* Lead
  + The number of rows to lead can optionally be specified. If the number of rows to lead is not specified, the lead is one row.
  + Returns null when the lead for the current row extends beyond the end of the window.
* Lag

The number of rows to lag can optionally be specified. If the number of rows to lag is not specified, the lag is one row.

Returns null when the lag for the current row extends before the beginning of the window.

* FIRST\_VALUE
* LAST\_VALUE

The OVER clause

* OVER with standard aggregates:
  + COUNT
  + SUM
  + MIN
  + MAX
  + AVG

OVER with a PARTITION BY statement with one or more partitioning columns.

* OVER with PARTITION BY and ORDER BY with one or more partitioning and/or ordering columns.

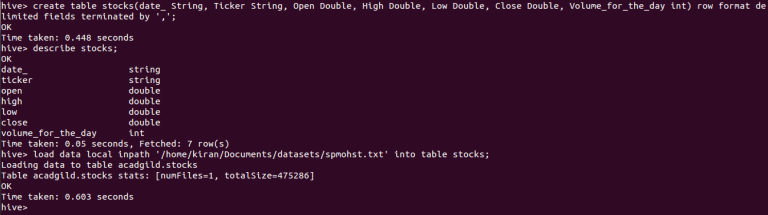
Analytics functions

* RANK
* ROW\_NUMBER
* DENSE\_RANK
* CUME\_DIST
* PERCENT\_RANK
* NTILE

To give you a brief idea of these windowing functions in Hive, we will be using stock market data. You can download the sample stocks data [from here](https://drive.google.com/open?id=0ByJLBTmJojjzbVhvSnQwNlhXUWs) and load into your stocks table.

Now we will create a table to load this stock market data as shown below.

|  |  |
| --- | --- |
| 1 | create table stocks (date\_ String, Ticker String, Open Double, High Double, Low Double, Close Double, Volume\_for\_the\_day int) row format delimited fields terminated by ','; |



Let us dive deeper into the window functions in Hive.

**Lag**

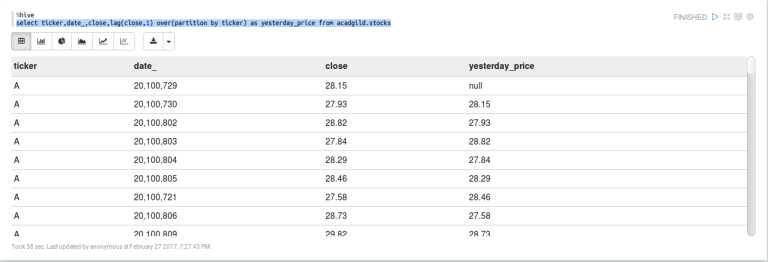
This function returns the values of the previous row. You can specify an integer offset which designates the row position else it will take the default integer offset as 1.

Here is the sample function for ***lag***

|  |  |
| --- | --- |
| 1 | select ticker,date\_,close,lag(close,1) over(partition by ticker) as yesterday\_price from acadgild.stocks |

Here using lag we can display the yesterday’s closing price of the ticker. Lag is to be used with over function, inside the over function you can use partition or order by classes.

In the below screenshot, you can see the closing price of the stock for the day and the yesterday’s price.



**Lead**

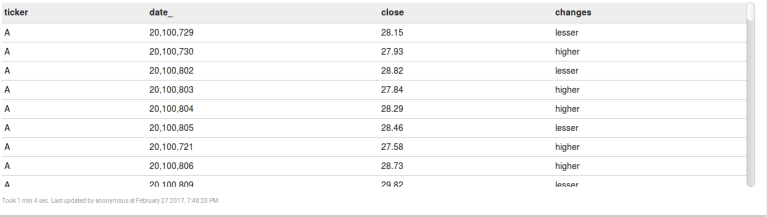
This function returns the values from the following rows. You can specify an integer offset which designates the row position else it will take the default integer offset as 1.

Here is the sample function for ***lead***

Now using the lead function, we will find that whether the following day’s closing price is higher or lesser than today’s and that can be done as follows.

|  |  |
| --- | --- |
| 1 | select ticker,date\_,close,case(lead(close,1) over(partition by ticker)-close)>0 when true then "higher" when false then "lesser" end as Changes from acadgild.stocks |

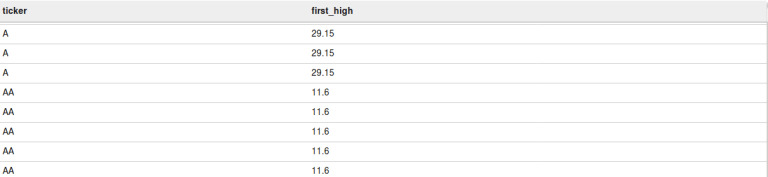
In the below screenshot, you can see the result.



**FIRST\_VALUE**

It returns the value of the first row from that window. With the below query, you can see the first row high price of the ticker for all the days.

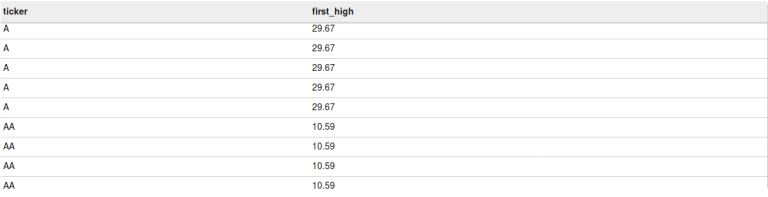
|  |  |
| --- | --- |
| 1 | select ticker,first\_value(high) over(partition by ticker) as first\_high from acadgild.stocks |



**LAST\_VALUE**

It is the reverse of FIRST\_VALUE. It returns the value of the last row from that window. With the below query, you can see the last row high price value of the ticker for all the days.

|  |  |
| --- | --- |
| 1 | select ticker,last\_value(high) over(partition by ticker) as first\_high from acadgild.stocks |



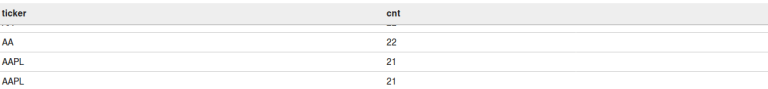
Let us now see the usage of the aggregate function using Over.

**Count**

It returns the count of all the values for the expression written in the over clause. From the below query, we can find the number of rows present for each ticker.

|  |  |
| --- | --- |
| 1 | select ticker,count(ticker) over(partition by ticker) as cnt from acadgild.stocks |

For each partition, the count of ticker will be calculated, you can see the same in the below screen shot.

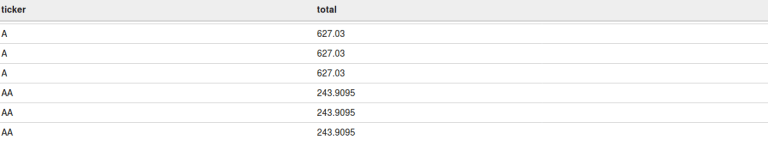


**Sum**

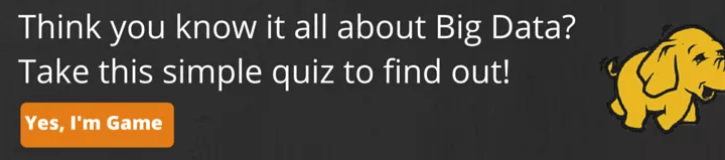
It returns the sum of all the values for the expression written in the over clause. From the below query, we can find the sum of all the closing stock prices for that particular ticker.

|  |  |
| --- | --- |
| 1 | select ticker,sum(close) over(partition by ticker) as total from acadgild.stocks |

For each ticker, the sum of all the closing prices will be calculated, you can see the same in the below screen shot.

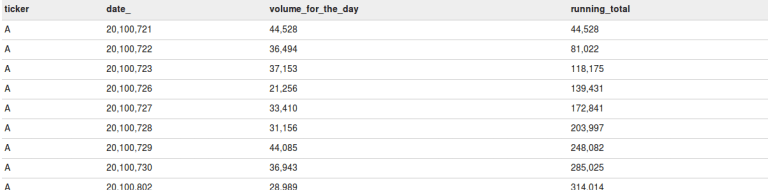


**Finding running total**



For suppose let us take if you want to get running total of the volume\_for\_the\_day for all the days for every ticker then you can do this with the below query.

|  |  |
| --- | --- |
| 1 | select ticker,date\_,volume\_for\_the\_day,sum(volume\_for\_the\_day) over(partition by ticker order by date\_) as running\_total from acadgild.stocks |

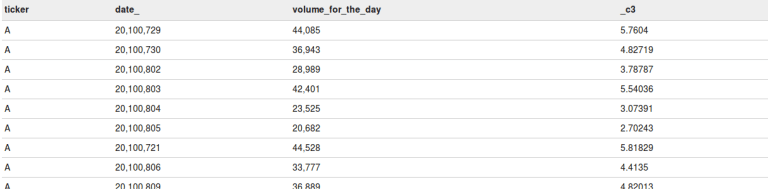


In the above screenshot, you can see the volume\_for\_the\_day for each day and the running total is the sum of volume\_for\_the\_day’s that are elapsed.

**Finding the percentage of each row value**

Now let’s take a scenario where you need to find the percentage of the volume\_for\_the\_day on the total volumes for that particular ticker and that can be done as follows.

|  |  |
| --- | --- |
| 1 | select ticker,date\_,volume\_for\_the\_day,(volume\_for\_the\_day\*100/(sum(volume\_for\_the\_day) over(partition by ticker))) from acadgild.stocks |

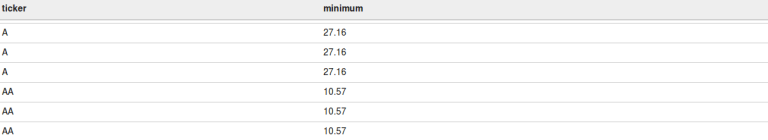


In the above screenshot, you can see that the percentage contribution of the volumes for the day is found based on the total volume for that ticker.

**Min**

It returns the minimum value of the column for the rows in that over clause. From the below query, we can find the minimum closing stock price for each particular ticker.

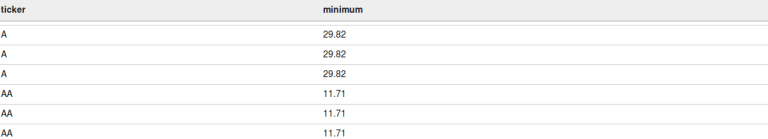
|  |  |
| --- | --- |
| 1 | select ticker, min(close) over(partition by ticker) as minimum from acadgild.stocks |



**Max**

It returns the maximum value of the column for the rows in that over clause. From the below query, we can find the maximum closing stock price for each particular ticker.

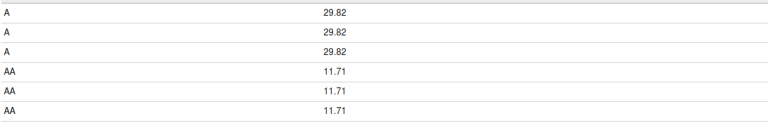
|  |  |
| --- | --- |
| 1 | select ticker, max(close) over(partition by ticker) as maximum from acadgild.stocks |



**AVG**

It returns the average value of the column for the rows that over clause returns. From the below query, we can find the average closing stock price for each particular ticker.

|  |  |
| --- | --- |
| 1 | select ticker, avg(close) over(partition by ticker) as maximum from acadgild.stocks |



Now let us work on some Analytic functions.

**Rank**

The rank function will return the rank of the values as per the result set of the over clause. If two values are same then it will give the same rank to those 2 values and then for the next value, the sub-sequent rank will be skipped.

The below query will rank the closing prices of the stock for each ticker. The same you can see in the below screenshot.

|  |  |
| --- | --- |
| 1 | select ticker,close,rank() over(partition by ticker order by close) as closing from acadgild.stocks |



**Row\_number**

Row number will return the continuous sequence of numbers for all the rows of the result set of the over clause.

From the below query, you will get the ticker, closing price and its row number for each ticker.

|  |  |
| --- | --- |
| 1 | select ticker,close,row\_number() over(partition by ticker order by close) as num from acadgild.stocks |



**Dense\_rank**

It is same as the rank() function but the difference is if any duplicate value is present then the rank will not be skipped for the subsequent rows. Each unique value will get the ranks in a sequence.

The below query will rank the closing prices of the stock for each ticker. The same you can see in the below screenshot.

|  |  |
| --- | --- |
| 1 | select ticker,close,dense\_rank() over(partition by ticker order by close) as closing from acadgild.stocks |

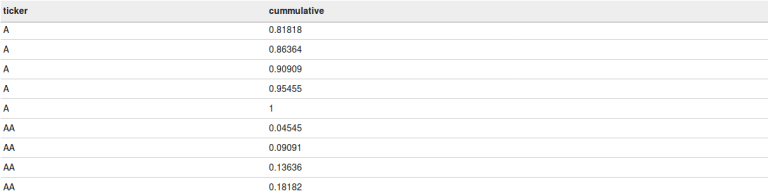


**Cume\_dist**

It returns the cumulative distribution of a value. It results from 0 to 1. For suppose if the total number of records are 10 then for the 1st row the cume\_dist will be 1/10 and for the second 2/10 and so on till 10/10.

This cume\_dist will be calculated in accordance with the result set returned by the over clause. The below query will result in the cumulative of each record for every ticker.

|  |  |
| --- | --- |
| 1 | select ticker,cume\_dist() over(partition by ticker order by close) as cummulative from acadgild.stocks |



**Percent\_rank**

It returns the percentage rank of each row within the result set of over clause. Percent\_rank is calculated in accordance with the rank of the row and the calculation is as follows (rank-1)/(total\_rows\_in\_group – 1). If the result set has only one row then the percent\_rank will be 0.

The below query will calculate the percent\_rank for every row in each partition and you can see the same in the below screen shot.

|  |  |
| --- | --- |
| 1 | select ticker,close,percent\_rank() over(partition by ticker order by close) as closing from acadgild.stocks |



**Ntile**

It returns the bucket number of the particular value. For suppose if you say Ntile(5) then it will create 5 buckets based on the result set of the over clause after that it will place the first 20% of the records in the 1st bucket and so on till 5th bucket.

The below query will create 5 buckets for every ticker and the first 20% records for every ticker will be in the 1st bucket and so on.

|  |  |
| --- | --- |
| 1 | select ticker,ntile(5) over(partition by ticker order by close ) as bucket from acadgild.stocks |

In the below screenshot, you can see that 5 buckets will be created for every ticker and the least 20% closing prices will be in the first bucket and the next 20% will be in the second bucket and so on till 5th bucket for all the tickers.



This is how we can perform windowing operations in Hive.

We hope this blog helped you in understanding what are windowing functions and how to implement them in Hive. Keep visiting our site [www.acadgild.com](https://acadgild.com/) for more updates on Big Data and other technologies.

### Comparing ORC vs Parquet Data Storage Formats using Hive

CSV is the most familiar way of storing the data. In this blog I will try to compare the performance aspects of the ORC and the Parquet formats. There is a lot of literature on what these are, so less focus on the same.  
  
Similar to [Parquet](https://parquet.apache.org/) for storing the data in the column oriented format there is another format called [ORC](https://orc.apache.org/). Parquet had been aggressively promoted by Cloudera and ORC by Hortonworks. Here are some articles ([1](http://hortonworks.com/blog/orcfile-in-hdp-2-better-compression-better-performance/), [2](http://data-governance.blogspot.in/2016/02/the-format-war-for-hadoop-structured.html)) on Parquet vs ORC.  
  
The CSV data can be converted into ORC and Parquet formats using Hive. These are the steps involved. The same steps are applicable to ORC also. Simply, replace Parquet with ORC. Behind the scenes a MapReduce job will be run which will convert the CSV to the appropriate format.  
  
- Create a Hive table (ontime)  
- Map the ontime table to the CSV data  
- Create a Hive table ontime\_parquet and specify the format as Parquet  
- Move the table from the ontime table to the ontime\_parquet table  
  
In the [previous blog](http://www.thecloudavenue.com/2016/04/analyzing-airline-dataset-with-hive.html), we have seen how to convert CSV into Parquet using Hive. The procedure is more or less for ORC, just replace the `STORED AS PARQUET` to `STORED AS ORC` in the table definition as shown below and also specify the compressions codec to use.

[?](http://www.thecloudavenue.com/2016/10/comparing-orc-vs-parquet-data-storage.html)

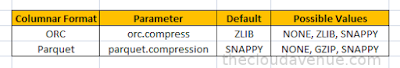
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | create external table ontime\_orc\_snappy (    Year INT,    Month INT,    DayofMonth INT,    DayOfWeek INT,    DepTime  INT,    CRSDepTime INT,    ArrTime INT,    CRSArrTime INT,    UniqueCarrier STRING,    FlightNum INT,    TailNum STRING,    ActualElapsedTime INT,    CRSElapsedTime INT,    AirTime INT,    ArrDelay INT,    DepDelay INT,    Origin STRING,    Dest STRING,    Distance INT,    TaxiIn INT,    TaxiOut INT,    Cancelled INT,    CancellationCode STRING,    Diverted STRING,    CarrierDelay INT,    WeatherDelay INT,    NASDelay INT,    SecurityDelay INT,    LateAircraftDelay INT  ) STORED AS PARQUET LOCATION '/user/bigdata/airline/input-orc-snappy-from-hive' TBLPROPERTIES ("orc.compress"="SNAPPY"); |

Then the data has to be moved from the regular Hive table (ontime) to the ontime\_orc\_snappy using the below command.

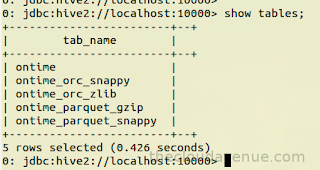
[?](http://www.thecloudavenue.com/2016/10/comparing-orc-vs-parquet-data-storage.html)

|  |  |
| --- | --- |
| 1 | INSERT OVERWRITE TABLE ontime\_parquet\_gzip SELECT \* FROM ontime; |

The property name for the and the default properties are mentioned in the below table. When not using the default compression codec then the property can be set on the table using the TBLPROPERTIES as shown in the above table creation command. **Note that ZLIB in ORC and GZIP in Parquet uses the same compression codec, just the property name is different.**



Four tables need to be created in Hive for the combination of orc/parquet and snappy/zlib/gzip compression as shown below.



Now that the tables have been created, the data can be moved from the ontime table to the remaining four tables. Four folders in HDFS will be created as shown below.



One the four tables I ran two queries on all the four tables. The first query was of type aggregation to find the number of delayed flights per origin as shown below.

[?](http://www.thecloudavenue.com/2016/10/comparing-orc-vs-parquet-data-storage.html)

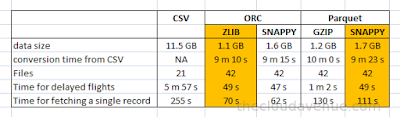
|  |  |
| --- | --- |
| 1 | select Origin, count(\*) from ontime\_parquet\_gzip where DepTime > CRSDepTime group by Origin; |

The second query is to fetch all the columns in a single row as shown below.

[?](http://www.thecloudavenue.com/2016/10/comparing-orc-vs-parquet-data-storage.html)

|  |  |
| --- | --- |
| 1 | select \* from ontime\_parquet\_gzip where origin = 'LNY' and AirTime = 16; |

Below is the comparison matrix which is of main interest.



**Here are a few things which I want to highlight**

- There is not much of storage savings when using using ORC and Parquet when using the same compression code like `SNAPPY vs SNAPPY` and `ZLIB vs GZIP`.

- The time for converting from CSV to ORC and Parquet format is very close, not much difference considering the total time it takes for the conversion.

- Hortonworks [blog](http://hortonworks.com/blog/orcfile-in-hdp-2-better-compression-better-performance/) says that the ORC format provides much better compression ratio when compared to Parquet. This is a bit misleading as the default properties are being used, ZLIB for ORC and SNAPPY for Parquet. By making sure that both the formats use the compression codec, there is not much significant difference in the compression ratio as shown in the above matrix. So, it would be better to focus on the features.

- For aggregation queries like `time for the the delayed flights` there is not such a drastic difference. Both the ORC and Parquet formats perform considerably well when compared to the CSV format.

- While fetching all the columns for a single now using a condition like "where origin = 'LNY' and AirTime = 16;", ORC has an edge over Parquet because the ORC format has a [light index](http://orc.apache.org/docs/indexes.html) along with each file. By using the indexes in ORC, the underlying MapRedeuce or Spark can avoid reading the entire block.

- The indexing in Parquet seems to be a good differentiator. Although the ORC has to create Index while creating the files, there is not significant difference for the conversion and also the size of the files for both the formats.

- The different Big Data vendors try to promote their own format without worrying much about the interoperability. The [Cloudera Certification](http://www.cloudera.com/training/certification/ccp-data-engineer.html) has topics about Parquet, while the [Hortonworks Certifications](http://hortonworks.com/training/class/hdp-certified-developer-hdpcd-exam/) has topics around ORC.

This has been a lengthy blog than I expected, so bye for now and see you soon.