show databases;

use default;

show tables;

**Create Managed Table:**

create table storesales(RowID int,OrderID string,OrderDate date,ShipDate date,ShipMode string,CustomerID string,CustomerName string,Segment string,Country string,City string,State string,PostalCode int,Region string,ProductID string,Category string,SubCategory string,ProductName string,Sales double) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;

describe storesales;

describe formatted storesales;

**Load data from local:**

load data local inpath '/home/ganesh/Analytics/dataset' into table storesales;

Select \* from storesales limit 15;

# if dropped need to create again for future use.

drop table storesales;

**Create External Table:**

hdfs dfs -mkdir /user/hdfs/superstore

hdfs dfs -copyFromLocal superstore.csv /user/hdfs/superstore

create external table storesales\_ex(RowID int,OrderID string,OrderDate date,ShipDate date,ShipMode string,CustomerID string,CustomerName string,Segment string,Country string,City string,State string,PostalCode int,Region string,ProductID string,Category string,SubCategory string,ProductName string,Sales double) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE location '/user/hdfs/superstore/’;

Describe storesales\_ex;

Describe formatted storesales\_ex;

**Validate physical location of external table after dropping table:**

drop table storesales\_ex;

**Load data from HDFS:**

create external table superstoresales\_inpath(RowID int,OrderID string,OrderDate string,ShipDate string,ShipMode string,CustomerID string,CustomerName string,Segment string,Country string,City string,State string,PostalCode int,Region string,ProductID string,Category string,SubCategory string,ProductName string,Sales double) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE location '/user/hdfs/superstore\_inpath/';

load data inpath '/user/hdfs/superstore' into table superstoresales\_inpath;

**Dynamic Partition:**

cd /opt/hadoop/etc/hadoop

vi yarn-site.xml

<property>

<name>mapreduce.map.memory.mb</name>

<value>2048</value>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>2048</value>

</property>

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>4096</value>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb</name>

<value>5120</value>

</property>

**vi mapred-site.xml**

**<property>**

**<name>mapred.child.java.opts</name>**

**<value>-Xmx6144m</value>**

**</property>**

**<property>**

**<name>yarn.app.mapreduce.am.env</name>**

**<value>HADOOP\_MAPRED\_HOME=/opt/hadoop</value>**

**</property>**

**<property>**

**<name>mapreduce.map.env</name>**

**<value>HADOOP\_MAPRED\_HOME=/opt/hadoop</value>**

**</property>**

**<property>**

**<name>mapreduce.reduce.env</name>**

**<value>HADOOP\_MAPRED\_HOME=/opt/hadoop</value>**

**</property>**

create table storeinfo(RowID int,OrderID string,OrderDate string,ShipDate string,ShipMode string,CustomerID string,CustomerName string,Segment string,Country string,City string,State string,PostalCode int,Region string,ProductID string,Category string,SubCategory string,ProductName string,Sales double) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE location '/user/hdfs/storeinfo/';

load data local inpath '/home/skiganesh/partitionds' into table storeinfo;

create external table salespartition\_dynamic(RowID int,OrderID string,OrderDate string,ShipDate string,ShipMode string,CustomerID string,CustomerName string,Segment string,Country string,PostalCode int,Region string,ProductID string,Category string,SubCategory string,ProductName string,Sales double,City string) partitioned by (state string) stored as orc;

set hive.exec.dynamic.partition=true;

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

#set hive.exec.max.dynamic.partitions=105;

Insert into table salespartition\_dynamic partition(state) Select RowID,OrderID,OrderDate,ShipDate,ShipMode,CustomerID,CustomerName,Segment,Country,PostalCode,Region,ProductID,Category,SubCategory,ProductName,Sales,City,State from storeinfo;

**Static Partition:**

**Pyspark commands:**

**To get the pyspark shell:**

pyspark

**Setting Log Level as “Error”:**

spark.sparkContext.setLogLevel("ERROR");

**Connecing to hive default database using spark sql:**

spark.sql("use default");

spark.sql("show tables").show();

spark.sql("select count(1) from storesales").show();

californiadf=spark.sql("select \* from storesales where state='California'");

coloradodf=spark.sql("select \* from storesales where state='Colorado'");

californiadf.coalesce(1).write.csv('file:////home/skiganesh/storesale/california', header=True);

coloradodf.coalesce(1).write.csv('file:////home/skiganesh/storesale/colorado', header=True);

**Hive Queries:**

create table salepartitionstatic(RowID int,OrderID string,OrderDate string,ShipDate string,ShipMode string,CustomerID string,CustomerName string,Segment string,Country string,PostalCode int,Region string,ProductID string,Category string,SubCategory string,ProductName string,Sales double,City string) partitioned by (State string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;

Load data local inpath '/home/skiganesh/storesale/california/' into table salepartitionstatic partition(state='california');

Load data local inpath '/home/skiganesh/storesale/colorado/' into table salepartitionstatic partition(state='colorado');

**Bucketed Table:**

**Create Bucketed Table:**

set hive.exec.dynamic.partition=true;

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

create external table storesales\_bucket(RowID int,OrderID string,OrderDate date,ShipDate date,ShipMode string,CustomerID string,CustomerName string,Segment string,Country string,City string,State string,PostalCode int,Region string,ProductID string,Category string,SubCategory string,ProductName string,Sales double) CLUSTERED BY (State) SORTED BY (City) INTO 10 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE location '/user/hdfs/storesales\_bucket';

insert into table storesales\_bucket select \* from storeinfo;

**Lateral View:**

Create table electronics\_items(category string,brand array<string>) row format delimited fields terminated by ‘\t’ collection items terminated by ‘,’;

load data local inpath ‘/home/skiganesh/complexdatatypes/’ into table electronics\_items;

select category, brandcount from electronics\_items lateral view explode(brand) itemsview as brandcount;

Analytical Functions :

Create table emp (id int, name string, designation string, department string, salary int) row format delimited fields terminated by ‘,’ ;

Load data local inpath ‘’ into table emp;

ROW\_NUMBER()

Select id,name,department,salary, row\_number() over(partition\_by department order by salary desc) as s\_no from emp;

RANK()

select id,name,department,salary, rank () over(partition\_by department order by salary desc) as rank from emp;

DENSE\_RANK()

select id,name,department,salary, dense\_rank () over(partition\_by department order by salary desc) as rank from emp;

Inner Join:

select sales.\*, product.\* from sales join product on sales.id=product.pid;

Left Outer Join:

select sales.\*, product.\* from sales left outer join product on sales.id=product.pid;

Right Outer Join:

select sales.\*, product.\* from sales right outer join product on sales.id=product.pid;

Full Outer Join:

select sales.\*, product.\* from sales full outer join product on sales.id=product.pid;

**Complex Data Types:**

create table personal\_info(name string, friends array<string>, mobile map<string,bigint>, otherinfo struct<company:string, pin:int, married:string, salary:int>) row format delimited fields terminated by ‘\t’ collection items terminated by ‘,’ map keys terminated by ‘:’ lines terminated by ‘\n’;

load data local inpath ‘’ into table personal\_info;

select name,mobile[‘office’] from personal\_info;

select name, friend[0] from personal\_info;

select name,other\_info.company from personal\_info;

Common Table Express:

with q1 as (select category, brandcount from electronics\_items lateral view explode(brand) itemsview as brandcount) select category,count(brandcount) as count from q1 group by category;

ENABLE ACID PROPERTIES:

set hive.txn.manager =org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;

set hive.support.concurrency-true;

Table must be bucketed, declared as ORC format and has in it's table properties 'transactional'='true' (hive support ACID operations only for ORC format and transactional tables).

create table testTableNew(id int ,name string ) clustered by (id) into 2 buckets stored as orc TBLPROPERTIES('transactional'='true')