**Scenario Based questions:**

1) Will the reducer work or not if you use “Limit 1” in any HiveQL query?

**--> I think Reducer will work, because as per Hive documentation -- Limit indicates the number of rows to be returned. The rows returned are chosen at random.**

2) Suppose I have installed Apache Hive on top of my Hadoop cluster using default metasto re configuration. Then, what will happen if we have multiple clients trying to access Hive at the same time?

**--> The default metastore configuration allows only one Hive session to be opened at a time for accessing the metastore. Therefore, if multiple clients try to access the metastore at the same time, they will get an error. One has to use a standalone metastore, i.e. Local or remote metastore configuration in Apache Hive for allowing access to multiple clients concurrently.**

3) Suppose, I create a table that contains details of all the transactions done by the customers: CREATE TABLE transaction\_details (cust\_id INT, amount FLOAT, month STRING, country STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ ;

Now, after inserting 50,000 records in this table, I want to know the total revenue generated for each month. But, Hive is taking too much time in processing this query.

How will you solve this problem and list the steps that I will be taking in order to do so?

--> **This is problem of query latency. so, we can solve this problem of query latency by partitioning the table according to each month.**

**so we will be scanning only the partitioned data instead of whole data sets.**

**as we can not directly create partition on existing table we need to create**

**partition table.**

1**. create a partioned table .**

**hive> create table transaction\_details\_partitioned**

**(**

**cust\_id int,**

**amount float)**

**partitioned by(month string,country string)**

**row format delimited fileds** terminated by ',';

2. to enable dynamic partition need to set few commands in hive shell

**hive> set hive.exec.dynamic.partition = true;**

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

3. now load data infact transfer data from non partitioned table to newly created partition table

**hive> Insert overwrite table transaction\_details\_partitioned partition(month,country) select cust\_id,amount,month,country from transaction\_details;**

#####4. now later we can drop even old table and new table can be renamed to old table name

**hive> alter table transaction\_details\_partitioned rename to transaction\_details;**

#####5. now we can perform query using each partition and query processing time will be reduce and performance is improve

How can you add a new partition for the month December in the above partitioned table?

-->

**hive> alter table transaction\_details\_partitioned add partition(month = 'Dec') Location ‘user/hive/warehouse/transaction\_details\_partitioned/’**

4) am inserting data into a table based on partitions dynamically. But, I received an error – FAILED ERROR IN SEMANTIC ANALYSIS: Dynamic partition strict mode requires at least one static partition column. How will you remove this error?

**--> we can remove error by running below commands:**

**hive> set hive.exec.dynamic.partition = true;**

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

5) Suppose, I have a CSV file – ‘sample.csv’ present in ‘/temp’ directory with the following entries:

id first\_name last\_name email gender ip\_address

How will you consume this CSV file into the Hive warehouse using built-in SerDe?

**hive> create external table sample\_csv**

**(id int,**

**first\_name string,**

**last\_name string,**

**email string,**

**gender string,**

**ip\_address string)**

**row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'**

**stored as textfile**

**location '/temp';**

**we can run any query on this table now**

**hive> select \* form sample\_csv where gender = 'male';**

6) Suppose, I have a lot of small CSV files present in the input directory in HDFS and I want to create a single Hive table corresponding to these files. The data in these files are in the format: {id, name, e-mail, country}. Now, as we know, Hadoop performance degrades when we use lots of small files.

So, how will you solve this problem where we want to create a single Hive table for lots of small files without degrading the performance of the system?

* create a temporary table:

**hive> create table temp\_tbl(**

**ind int,**

**name string,**

**e\_mail string,**

**country string)**

**row format delimited**

**fields terminated by ','**

**stroed as textfile;**

load data into these file using below command :

**hive> load data inpath '/input\_directory/...' into table temp\_tbl;**

create a table that will store data in sequenceFile format:

**hive> create table seq\_tbl(**

**ind int,**

**name string,**

**e\_mail string,**

**country string)**

**row format delimited**

**fields terminated by ','**

**stroed as sequencefile;**

transfer data from temporary table into this table:

**hive> Insert overwrite table seq\_tbl select \* from temp\_tbl;**

7) LOAD DATA LOCAL INPATH ‘Home/country/state/’ OVERWRITE INTO TABLE address;

The following statement failed to execute. What can be the cause?

--> **the local inpath statement should contain a file and not a directory.here file is missing.**

8) Is it possible to add 100 nodes when we already have 100 nodes in Hive? If yes, how?

--> **Yes, we can add the nodes by following the below steps:**

**Step 1: Take a new system; create a new username and password**

**Step 2: Install SSH and with the master node setup SSH connections**

**Step 3: Add ssh public\_rsa id key to the authorized keys file**

**Step 4: Add the new DataNode hostname, IP address, and other details in /etc/hosts slaves file:**

**192.168.1.102 slave3.in slave3**

**Step 5: Start the DataNode on a new node**

**Step 6: Login to the new node like suhadoop or:**

**ssh -X hadoop@192.168.1.103**

**Step 7: Start HDFS of the newly added slave node by using the following command:**

**./bin/hadoop-daemon.sh start data node**

**Step 8: Check the output of the jps command on the new node**

=====================================================================================

**Hive Practical questions:**

**Hive Join operations**

Create a table named CUSTOMERS(ID | NAME | AGE | ADDRESS | SALARY)

Create a Second table ORDER(OID | DATE | CUSTOMER\_ID | AMOUNT)

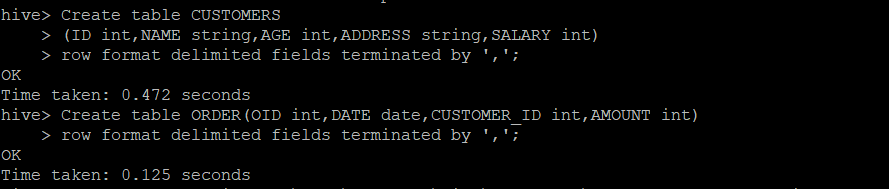
**hive> Create table CUSTOMERS**

**(ID int,NAME string,AGE int,ADDRESS string,SALARY int)**

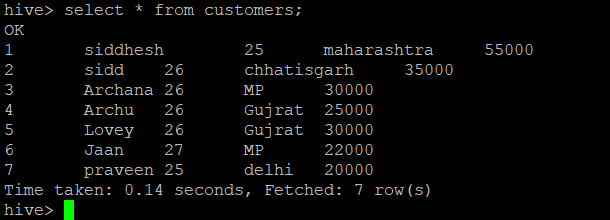
**row format delimited fields terminated by ',';**

**hive> Create table ORDER(OID int,DATE date,CUSTOMER\_ID int,AMOUNT int)**

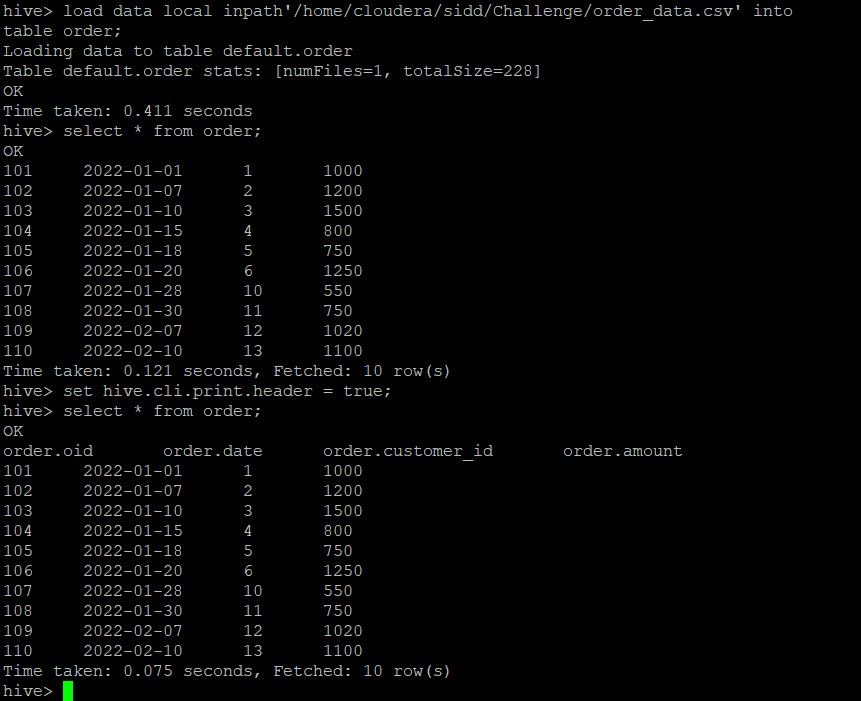
**row format delimited fields terminated by ',';**



**hive> load data local inpath'/home/cloudera/sidd/Challenge/customers\_data.csv' into table customers;**

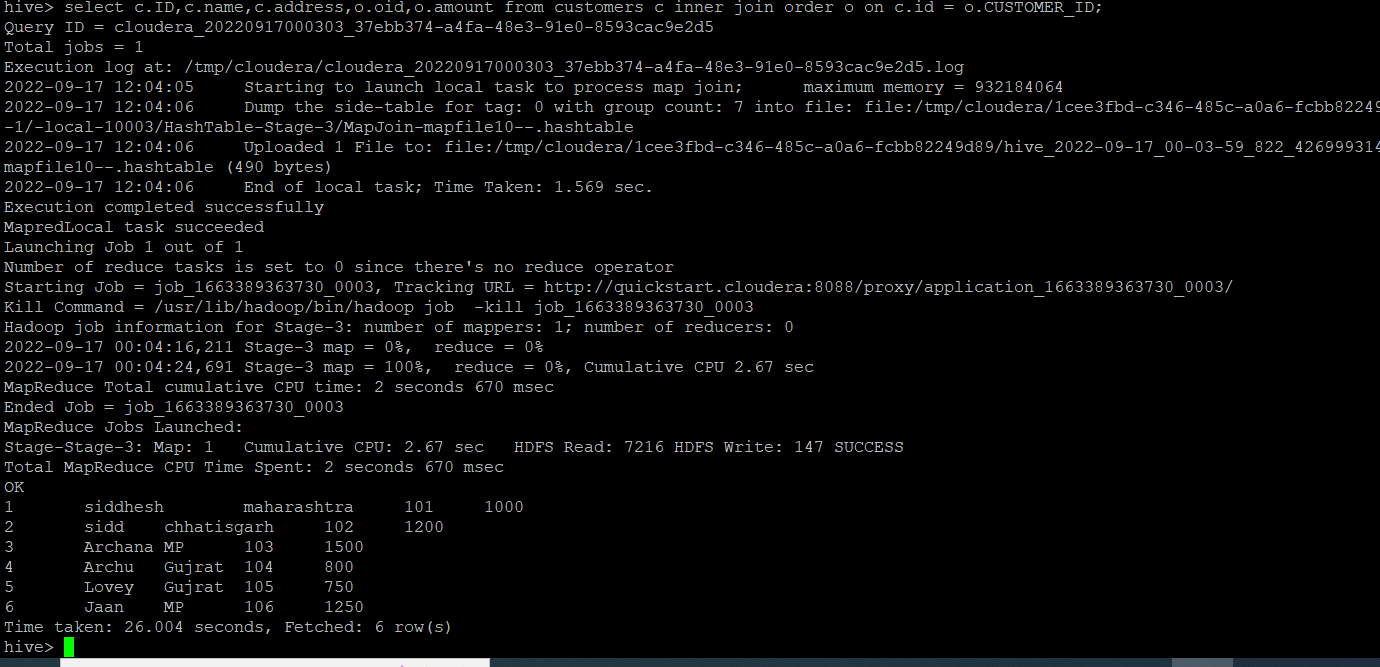


**hive> load data local inpath'/home/cloudera/sidd/Challenge/order\_data.csv' into table order;**



Inner Join:

**hive> select c.ID,c.name,c.address,o.oid,o.amount from customers c inner join order o on c.id = o.CUSTOMER\_ID;**

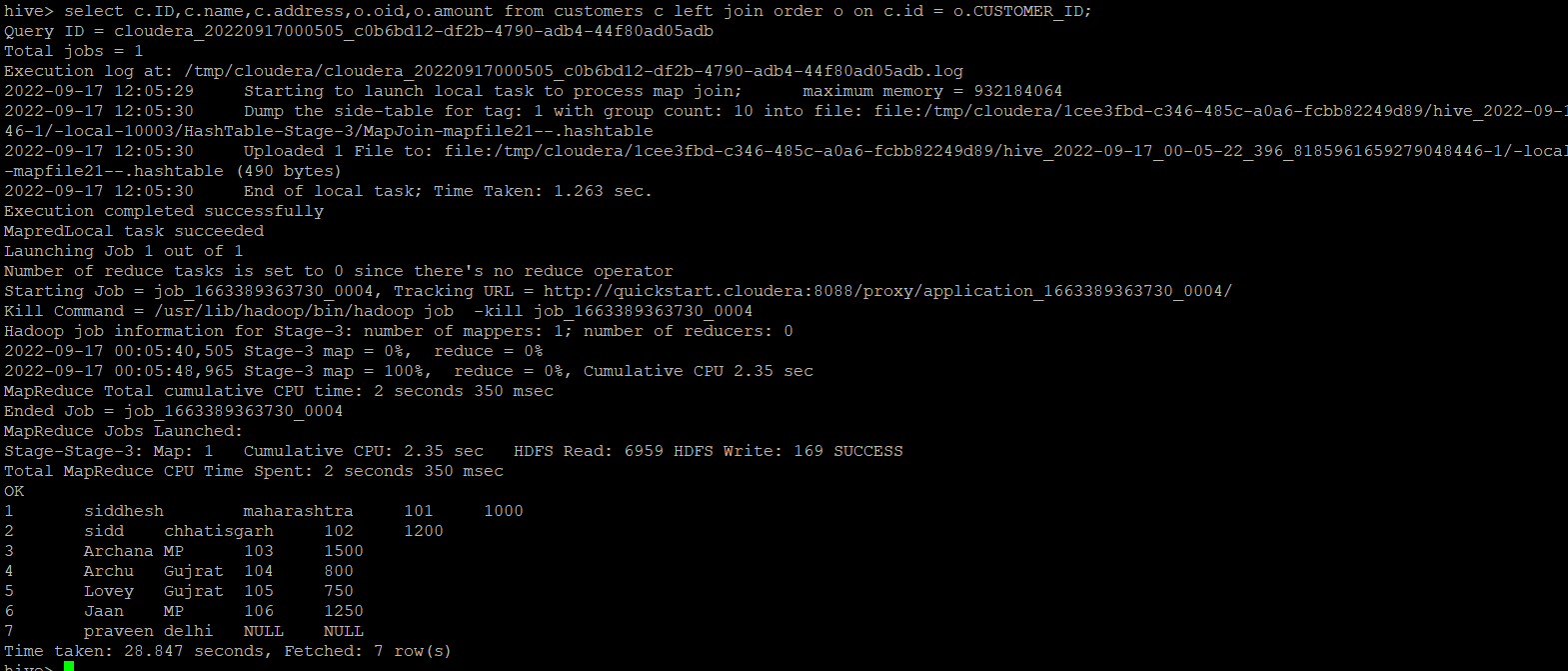


**Inner join using hints to improve performance:**

**hive> select /\*+ streamtable(o) \*/c.ID,c.name,c.address,o.oid,o.amount from customers c inner join order o on c.id = o.CUSTOMER\_ID;**

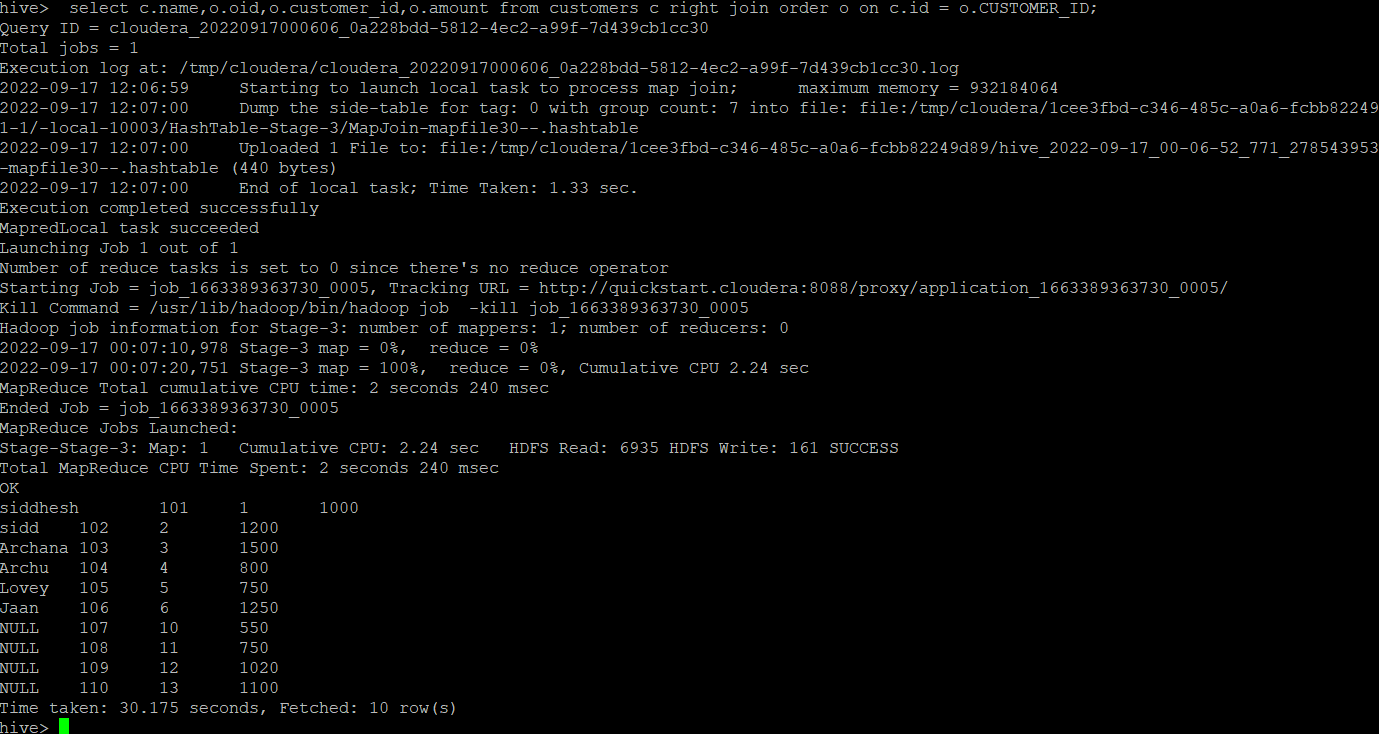
**Left join:**

**hive> select c.ID,c.name,c.address,o.oid,o.amount from customers c left join order o on c.id = o.CUSTOMER\_ID;**



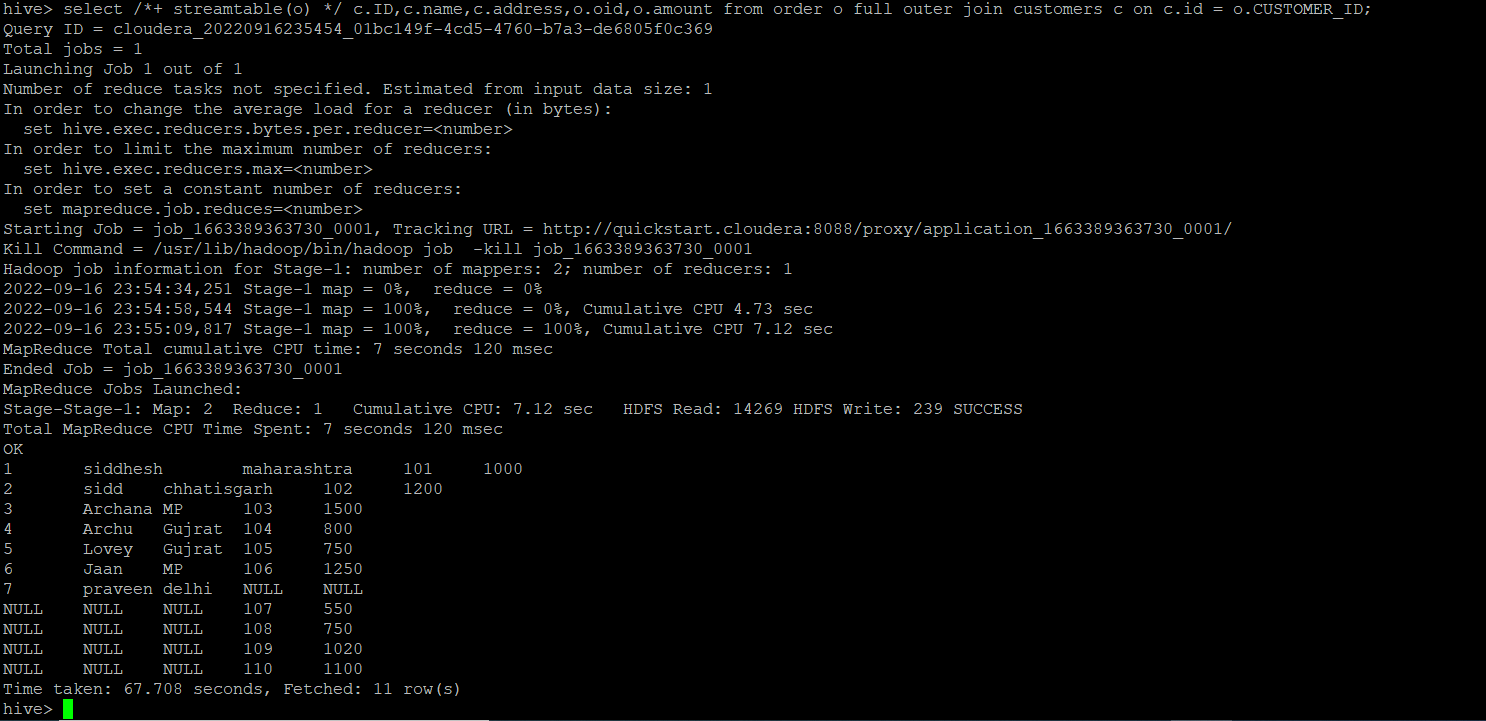
**Right join:**

**hive> select c.name,o.oid,o.customer\_id,o.amount from customers c right join order o on c.id = o.CUSTOMER\_ID;**



**Full Outer join:**

**hive> select /\*+ streamtable(o) \*/ c.ID,c.name,c.address,o.oid,o.amount from order o full outer join customers c on c.id = o.CUSTOMER\_ID;**



BUILD A DATA PIPELINE WITH HIVE

Download a data from the given location -

https://archive.ics.uci.edu/ml/machine-learning-databases/00360/

1. Create a hive table as per given schema in your dataset

**create table Air\_quality(**

**Date date,**

**Time string,**

**CO array<int>,**

**PT08\_S1 int,**

**NMHC int,**

**C6H6 array<int>,**

**PT08\_S2 int,**

**NOx int,**

**PT08\_S3 int,**

**NO2 int,**

**PT08\_S4 int,**

**PT08\_S5 int,**

**T array<int>,**

**RH array<int>,**

**AH array<int>)**

**row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'**

**with serdeproperties (**

**"separatorChar" = "\;",**

**"quoteChar" = "\"",**

**"escapeChar" = "\\"**

**)**

**stored as textfile**

**tblproperties ("skip.header.line.count" = "1");**

**create table Air\_quality2(**

**Date string,**

**Time string,**

**CO array<int>,**

**PT08\_S1 int,**

**NMHC int,**

**C6H6 array<int>,**

**PT08\_S2 int,**

**NOx int,**

**PT08\_S3 int,**

**NO2 int,**

**PT08\_S4 int,**

**PT08\_S5 int,**

**T array<int>,**

**RH array<int>,**

**AH array<int>)**

**row format delimited**

**fields terminated by "\;"**

**collection items terminated by ','**

**tblproperties ("skip.header.line.count" = "1");**

2. try to place a data into table location

**hive> load data local inpath '/home/cloudera/sidd/Challenge/AirQualityUCI.csv' into table Air\_quality;**

**hive> load data local inpath '/home/cloudera/sidd/Challenge/AirQualityUCI.csv' into table Air\_quality2;**

3. Perform a select operation .

**hive> select \* from Air\_quality2 limit 5;**

4. Fetch the result of the select operation in your local as a csv file .

$ vim demo.hql

**select \* from challenge.sales\_order\_csv limit 5;**

and save

**$ echo 'select \* from hive\_class\_b1.sales\_order\_csv;' | cat >demo.hql**

**$ hive -e 'select \* from challenge.sales\_order\_csv limit 5' > /home/cloudera/arch/demo2.csv;**

**$ hive -f demo.hql > /home/cloudera/sidd/demo.csv;**

5. Perform group by operation .

**hive> select avg(CO),Date from Air\_quality group by Date;**

7. Perform filter operation at least 5 kinds of filter examples .

**select count(\*),date from Air\_quality group by date;**

**select count(\*) total,date from Air\_quality group by date having total< 24;**

**select \* from Air\_quality where date = '31/12/2004';**

**select avg(t[0]) from Air\_quality2 where date = '31/12/2004';**

**select avg(cast(NMHC as int)) from Air\_quality;**

8. show and example of regex operation

**Hive> Create table emp\_info(id int,email\_id string)**

**row format delimited**

**fields terminated by ','**

**tblproperties ("skip.header.line.count" = "1");**

**Hive> load data local inpath'/home/cloudera/sidd/Challenge/emp\_info.csv' into table emp\_info;**

**Hive> select id,regexp\_extract(email\_id,'@(.\*)',1) as email\_id from emp\_info;**

**Hive> select id,regexp\_replace(email\_id,'gmail', 'email') as email\_id from emp\_info;**

9. alter table operation

rename table:

**hive> alter table Air\_quality rename to AirQuality;**

Add column:

**hive> Alter table Air\_quality add columns(weather string);**

change column name:

**hive> Alter table Air\_quality change weather teamperature int;**

change column datatype:

**hive> Alter table Air\_quality change weather weather string;**

10 . drop table operation

**hive> drop table Air\_quality;**

12 . order by operation .

**hive> select avg(temperature),Date from Air\_quality group by Date order by Date;**

13 . where clause operations you have to perform .

**hive> select avg(temperature),Date from Air\_quality where Date ='10/1/2005' group by Date;**

14 . sorting operation you have to perform .

**select \* from Air\_quality where date = '10/1/2005' sort by t;**

15 . distinct operation you have to perform .

**hive> select distinct date from Air\_quality;**

16 . like an operation you have to perform .

**hive> select \* from AirQuality where weather like 'Rainy’;**

17 . union operation you have to perform .

**Hive> select \* from Air\_quality where date = '10/1/2005'**

**union all**

**select \* from Air\_quality where date = '10/1/2006'**

18 . table view operation you have to perform .

**Hive> create view vw\_airQuality as select \* from Air\_quality2 where date = '10/1/2005' ;**

**Hive> select \* from vw\_airQuality;**

hive operation with python

Create a python application that connects to the Hive database for extracting data, creating sub tables for data processing, drops temporary tables.fetch rows to python itself into a list of tuples and mimic the join or filter operations