**Scenario Based Question:**

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

* If your query is a simple select query then no reducers are called.
* If your query has something like aggregation along with group by or order by and lets say you are using MR as your execution engine then reducers will be called.

**Suppose I have installed Apache Hive on top of my Hadoop cluster using default metastore 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.

**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?**

We can solve this problem of query latency by partitioning the table according to each month. So, for each month we will be scanning only the partitioned data instead of whole data sets.

As we know, we can’t partition an existing non-partitioned table directly. So, we will be taking following steps to solve the very problem:

1. Create a partitioned table, say partitioned\_transaction:

CREATE TABLE partitioned\_transaction (cust\_id INT, amount FLOAT, country STRING) PARTITIONED BY (month STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ ;

2. Enable dynamic partitioning in Hive:

SET hive.exec.dynamic.partition = true;

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

3. Transfer the data from the non – partitioned table into the newly created partitioned table:

INSERT OVERWRITE TABLE partitioned\_transaction PARTITION (month) SELECT cust\_id, amount, country, month FROM transaction\_details;

Now, we can perform the query using each partition and therefore, decrease the query time.

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

ALTER TABLE partitioned\_transaction ADD PARTITION (month=’Dec’) LOCATION  ‘/partitioned\_transaction’;

**I 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?**

SET hive.exec.dynamic.partition = true;

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

**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?**

SerDe stands for serializer/deserializer. A SerDe allows us to convert the unstructured bytes into a record that we can process using Hive. SerDes are implemented using Java. Hive comes with several built-in SerDes and many other third-party SerDes are also available.

Hive provides a specific SerDe for working with CSV files. We can use this SerDe for the sample.csv by issuing following commands:

CREATE EXTERNAL TABLE sample

(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’;

**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?**

One can use the SequenceFile format which will group these small files together to form a single sequence file. The steps that will be followed in doing so are as follows:

* Create a temporary table:

CREATE TABLE temp\_table (id INT, name STRING, e-mail STRING, country STRING)

ROW FORMAT FIELDS DELIMITED TERMINATED BY ‘,’ STORED AS TEXTFILE;

* Load the data into temp\_table:

LOAD DATA INPATH ‘/input’ INTO TABLE temp\_table;

* Create a table that will store data in SequenceFile format:

CREATE TABLE sample\_seqfile (id INT, name STRING, e-mail STRING, country STRING)

ROW FORMAT FIELDS DELIMITED TERMINATED BY ‘,’ STORED AS SEQUENCEFILE;

* Transfer the data from the temporary table into the sample\_seqfile table:

INSERT OVERWRITE TABLE sample SELECT \* FROM temp\_table;

Hence, a single SequenceFile is generated which contains the data present in all of the input files and therefore, the problem of having lots of small files is finally eliminated.

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

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

Filename is missing.

**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)**

**Now perform different joins operations on top of these tables**

**(Inner JOIN, LEFT OUTER JOIN ,RIGHT OUTER JOIN ,FULL OUTER JOIN)**

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

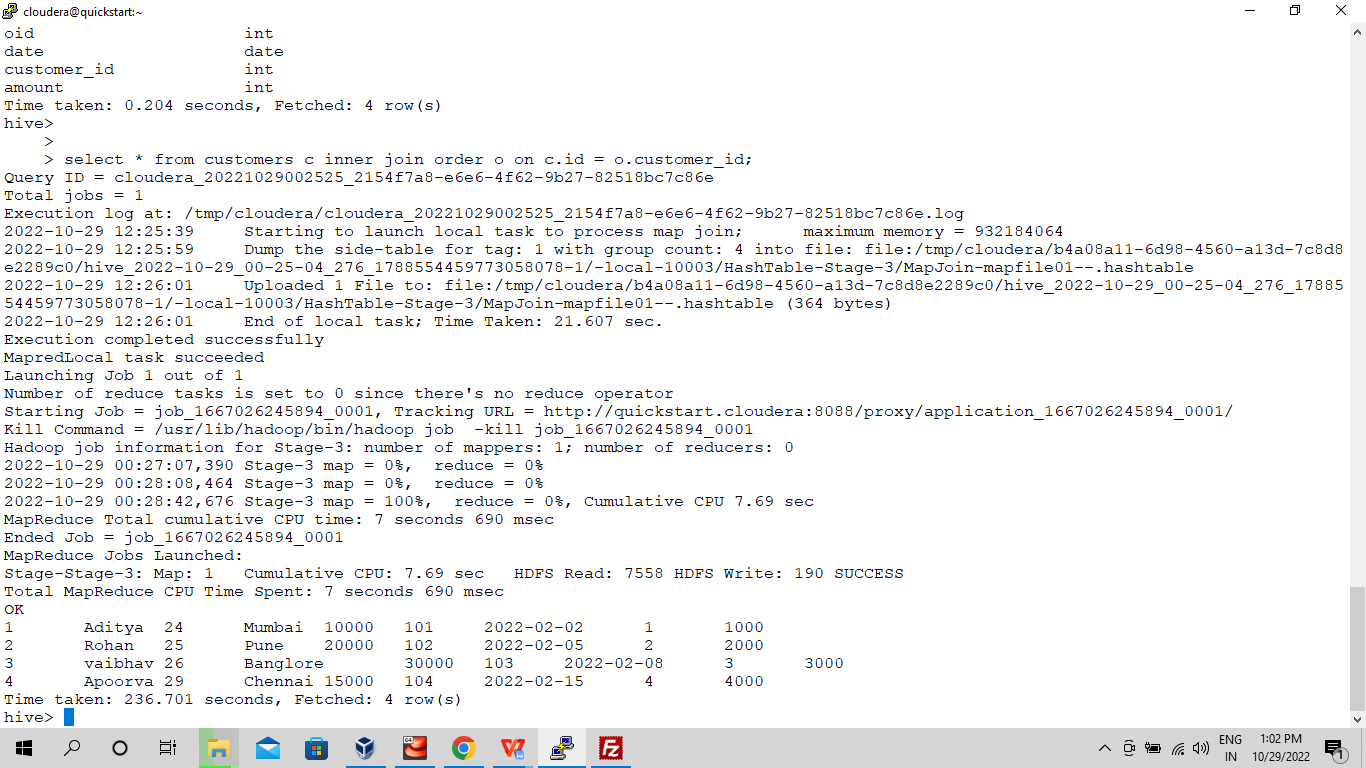
> fields terminated by ",";

hive> load data local inpath "/home/cloudera/data/customers.csv" into table customers;

hive> load data local inpath "/home/cloudera/data/orders.csv" into table order;

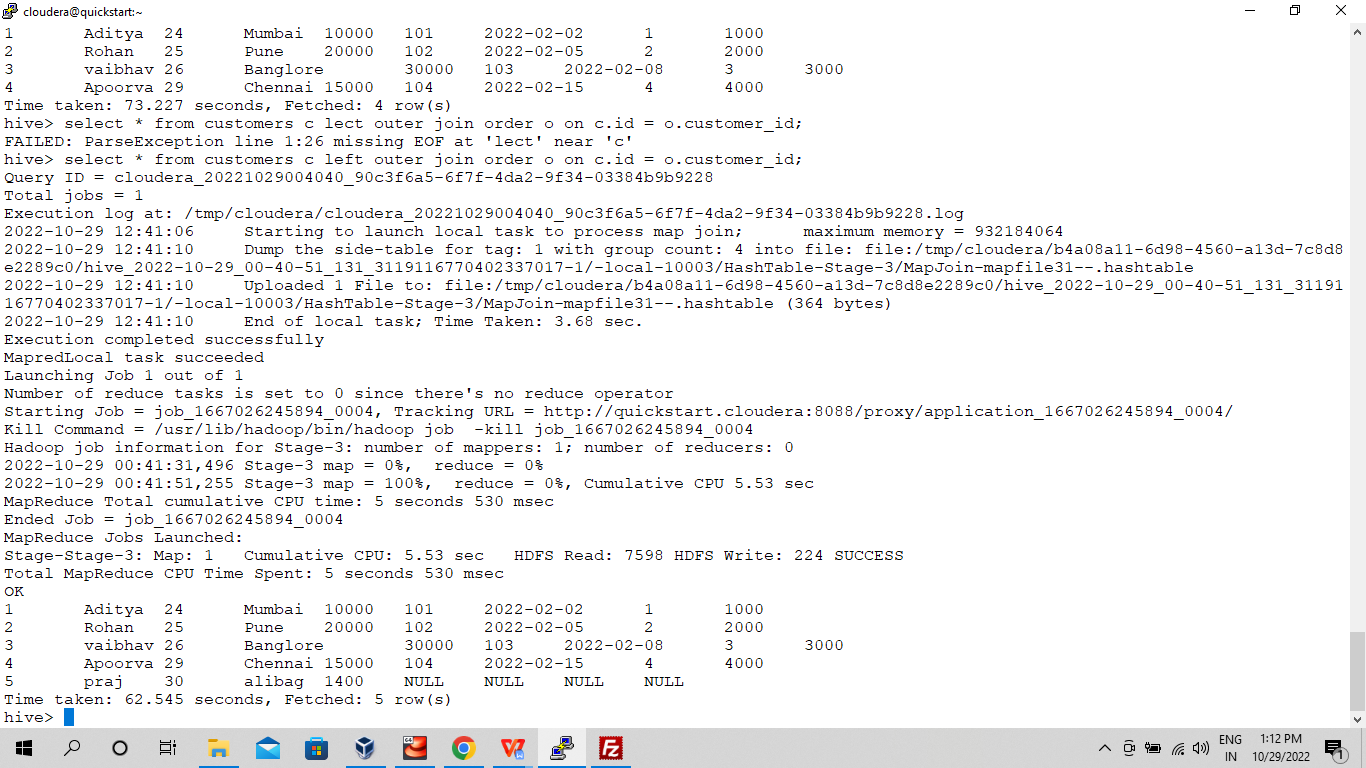
**Inner Join -**

hive> select \* from customers c inner join order o on c.id = o.customer\_id;



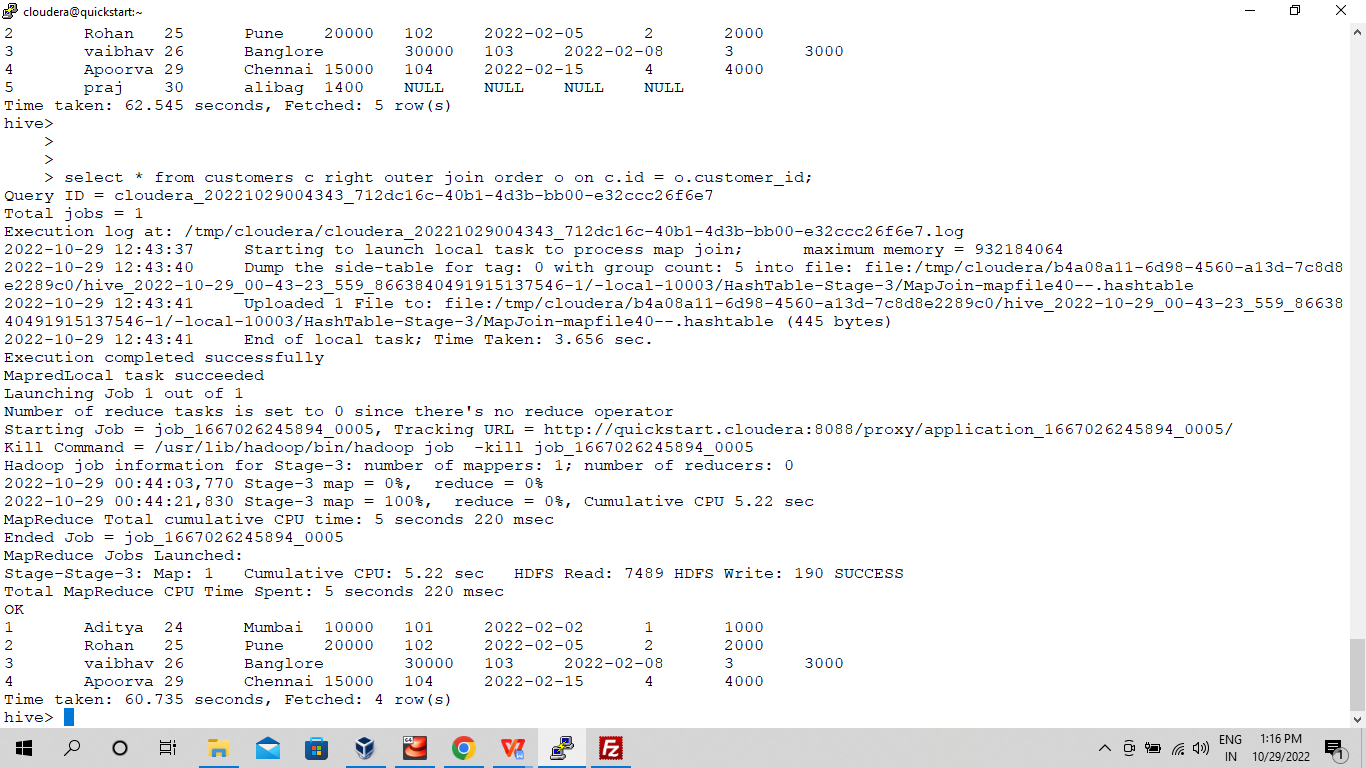
**Left Outer Join -**

hive> select \* from customers c left outer join order o on c.id = o.customer\_id;



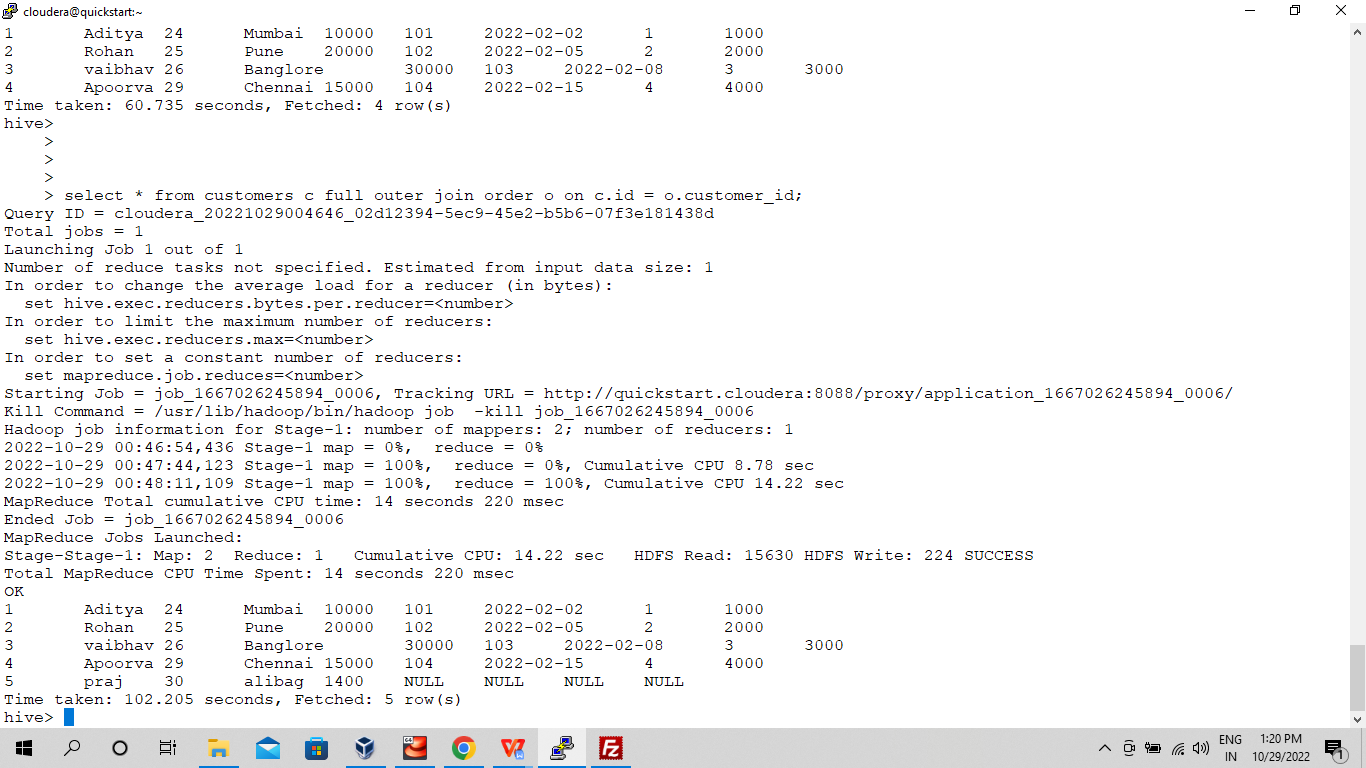
**Right Outer Join -**

hive > select \* from customers c right outer join order o on c.id = o.customer\_id;



**Full Outer Join -**

hive> select \* from customers c full outer join order o on c.id = o.customer\_id;

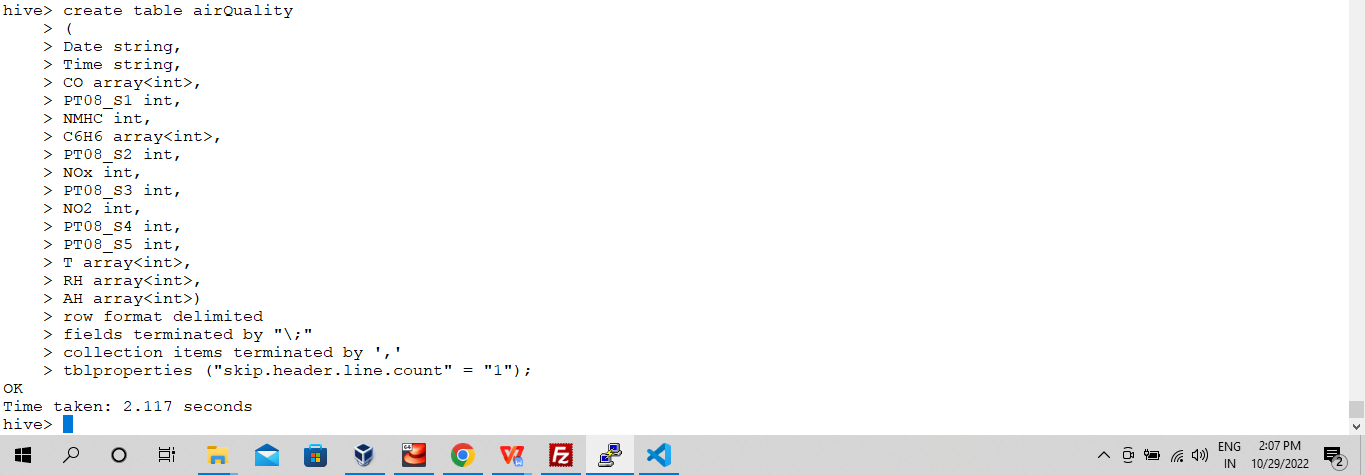


**BUILD A DATA PIPELINE WITH HIVE**

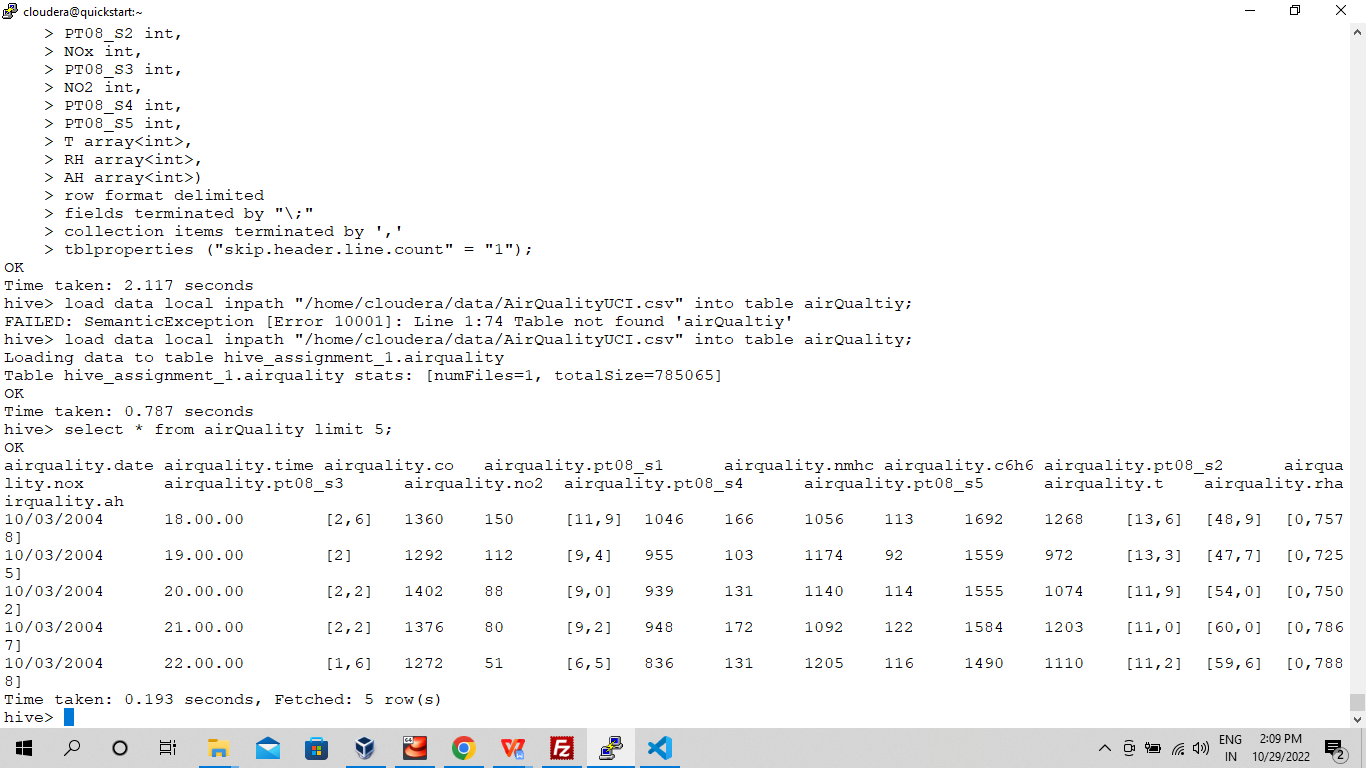
**Download a data from the given location -**

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

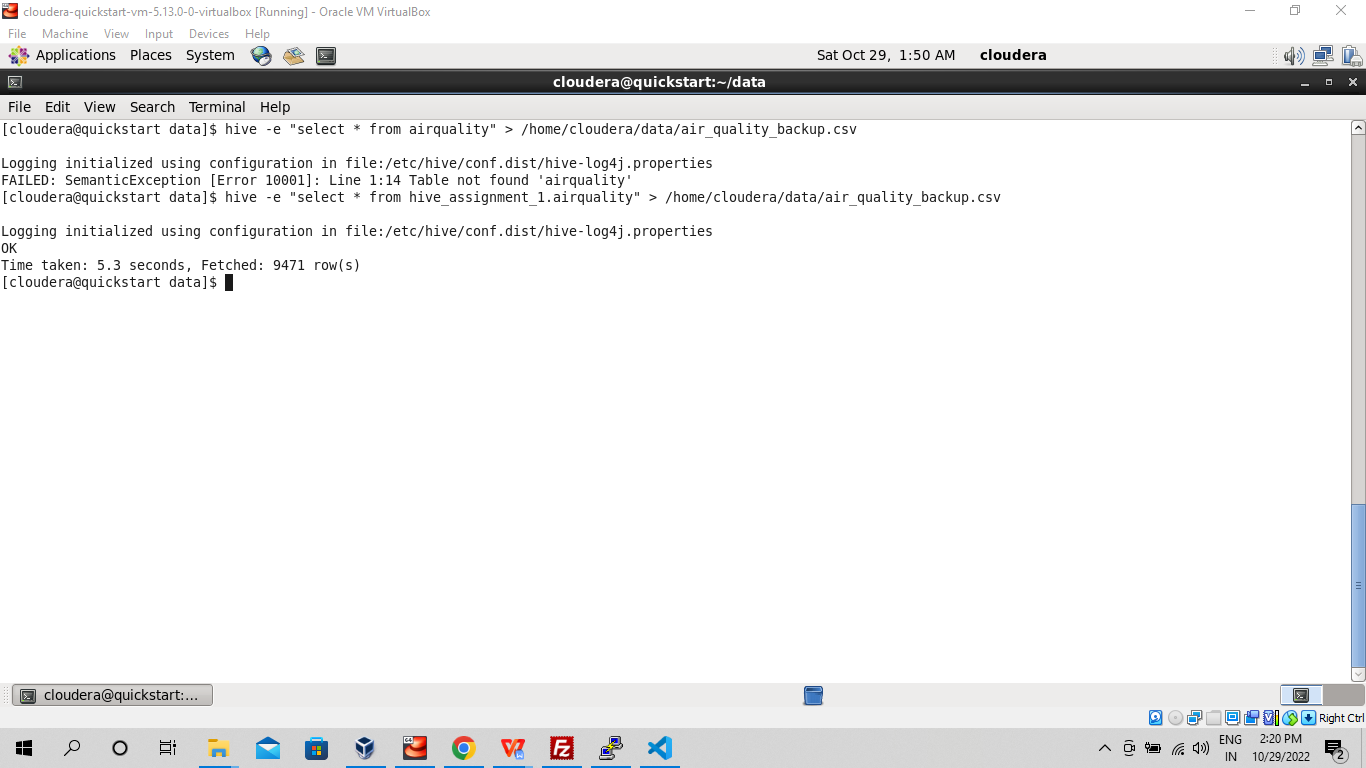
1. **Create a hive table as per given schema in your dataset**



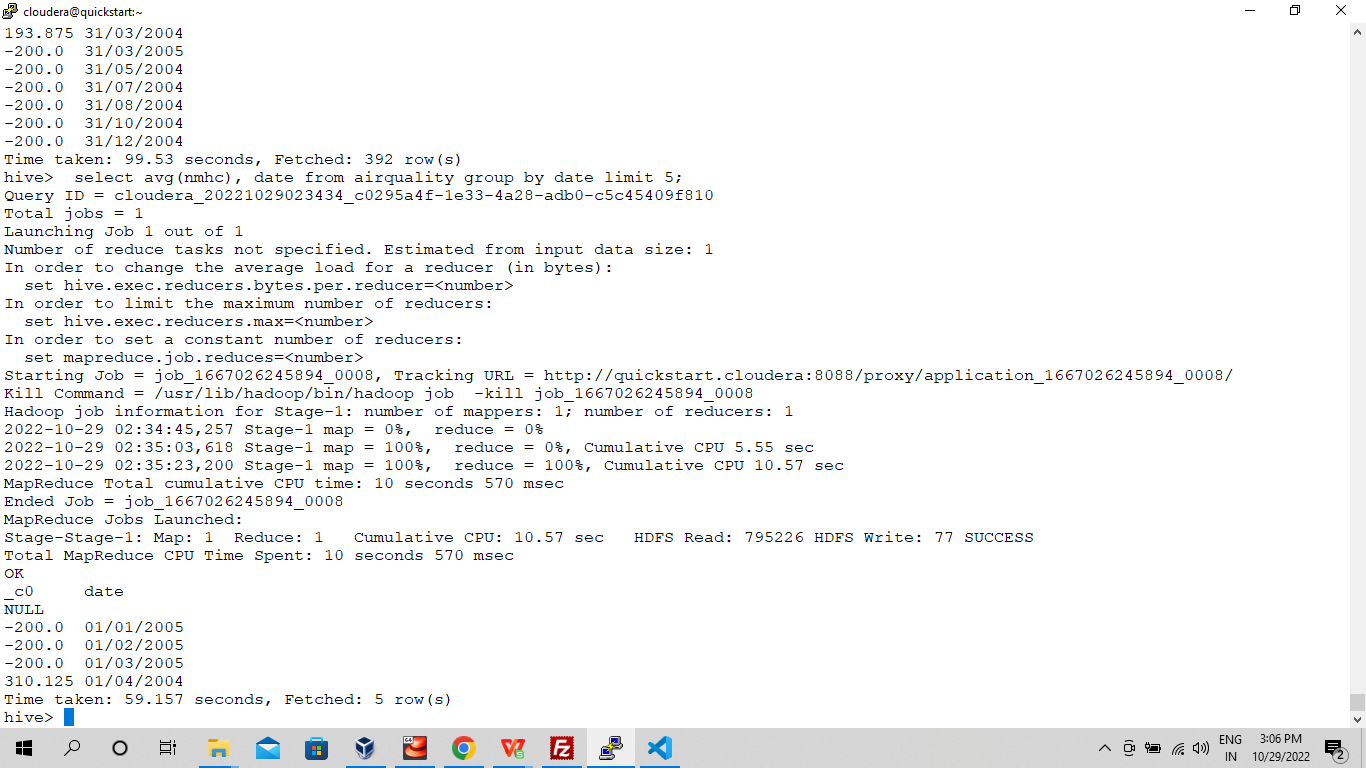
1. **try to place a data into table location**
2. **Perform a select operation**



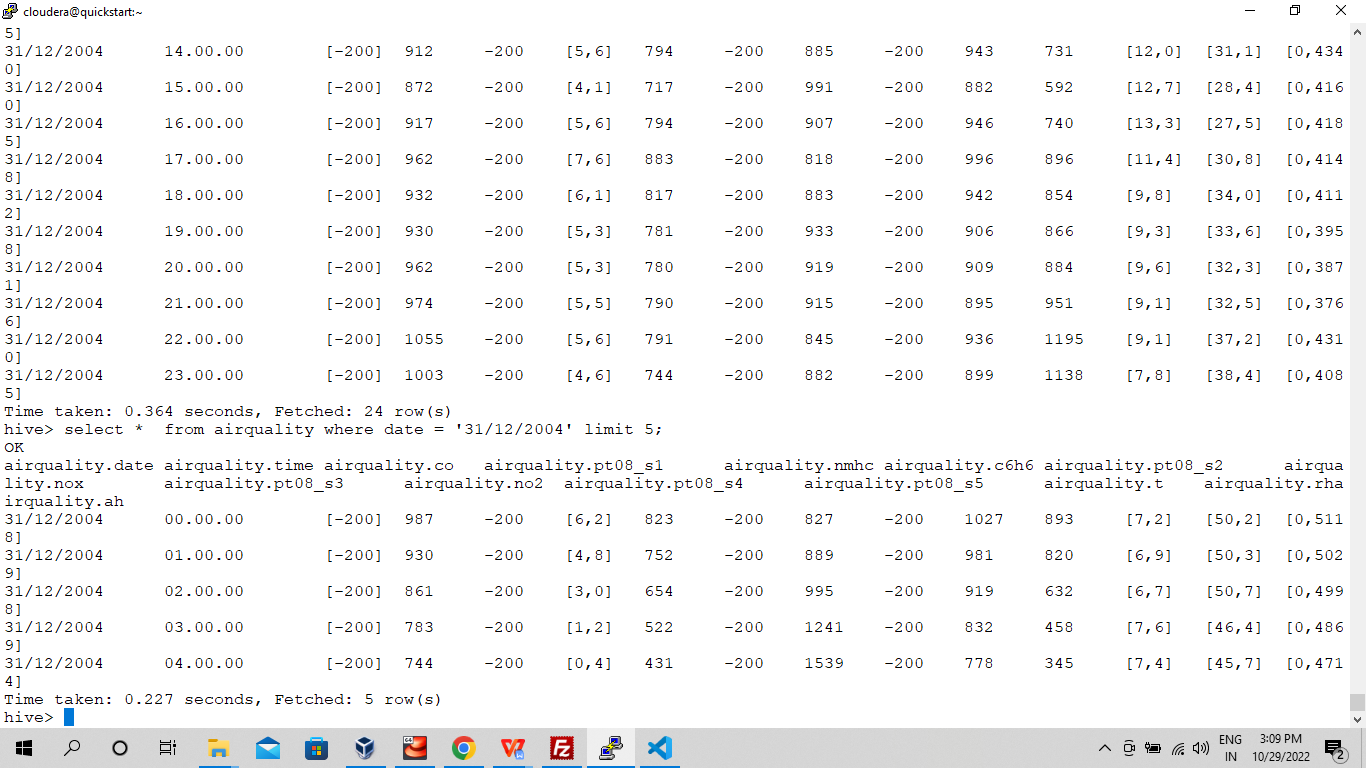
1. **Fetch the result of the select operation in your local as a csv file**

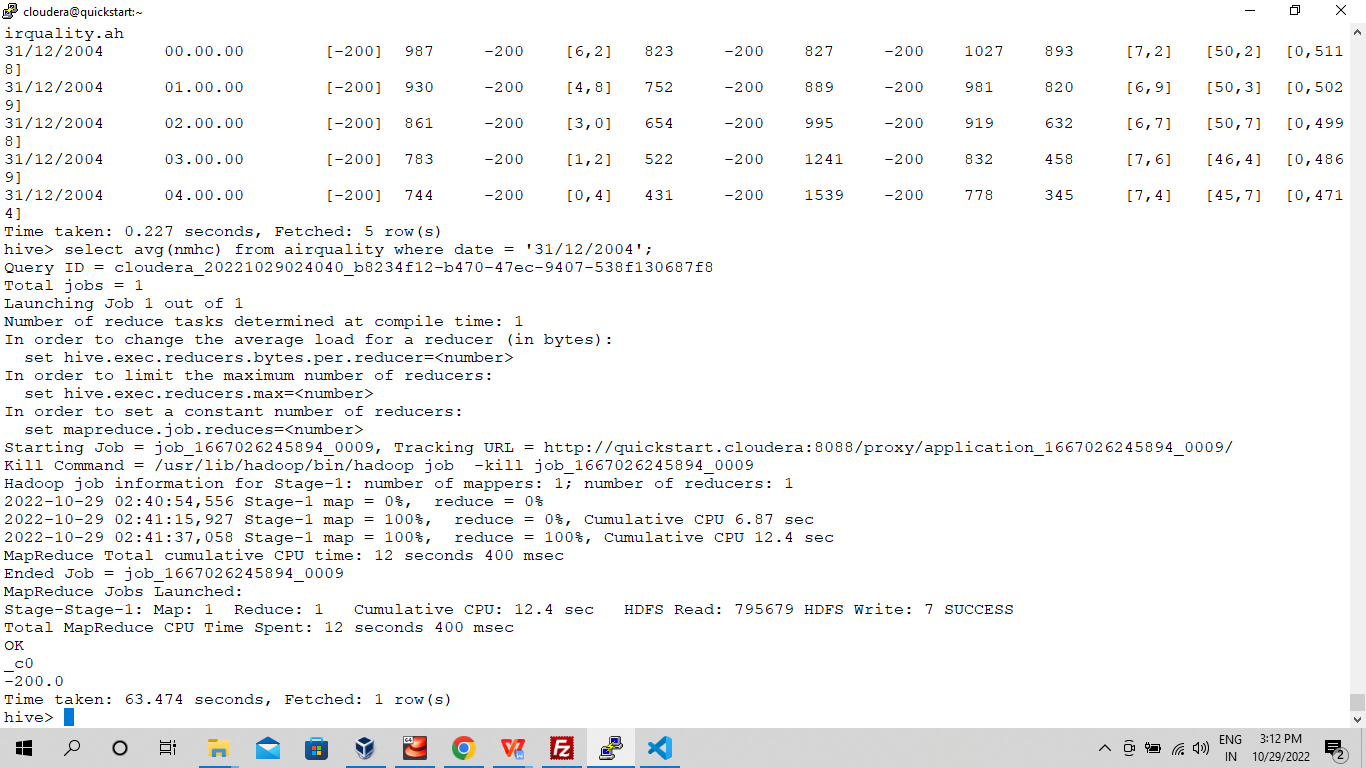


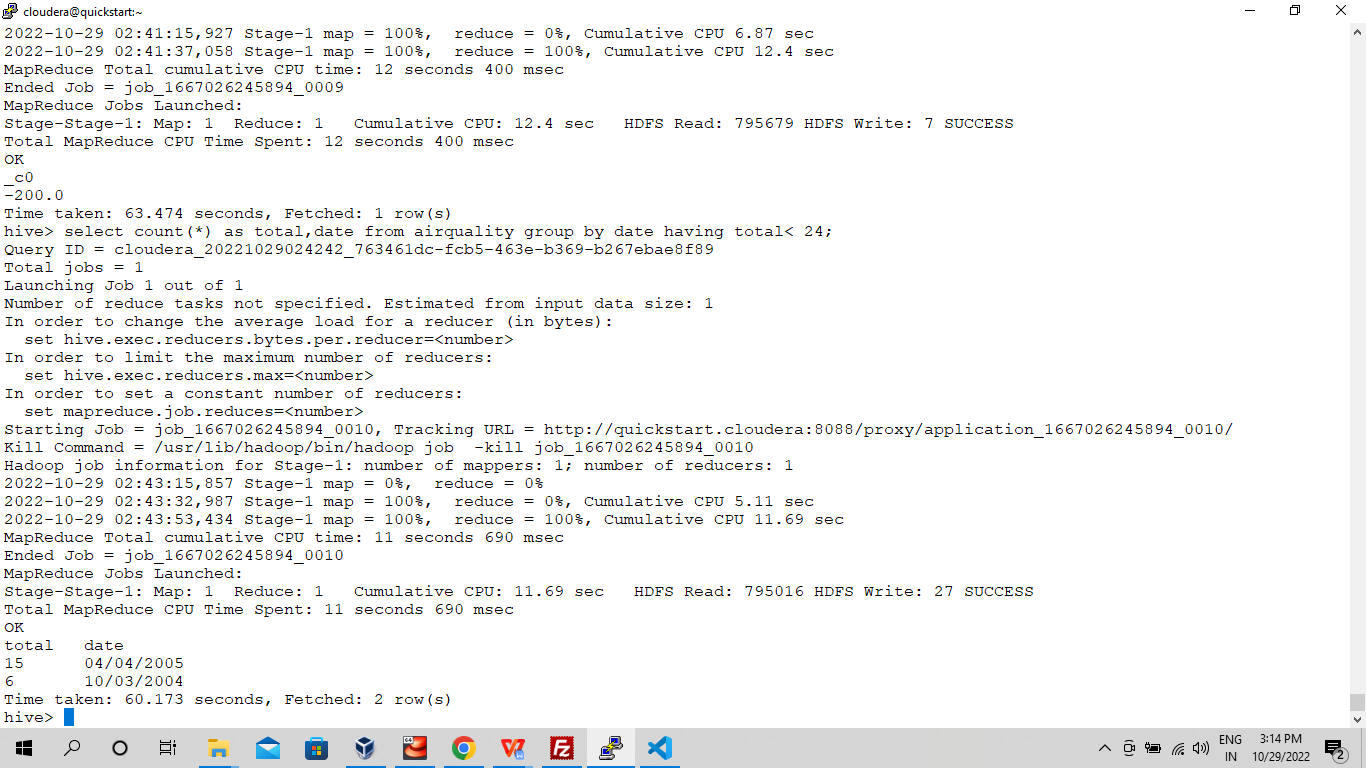
1. **Perform group by operation**

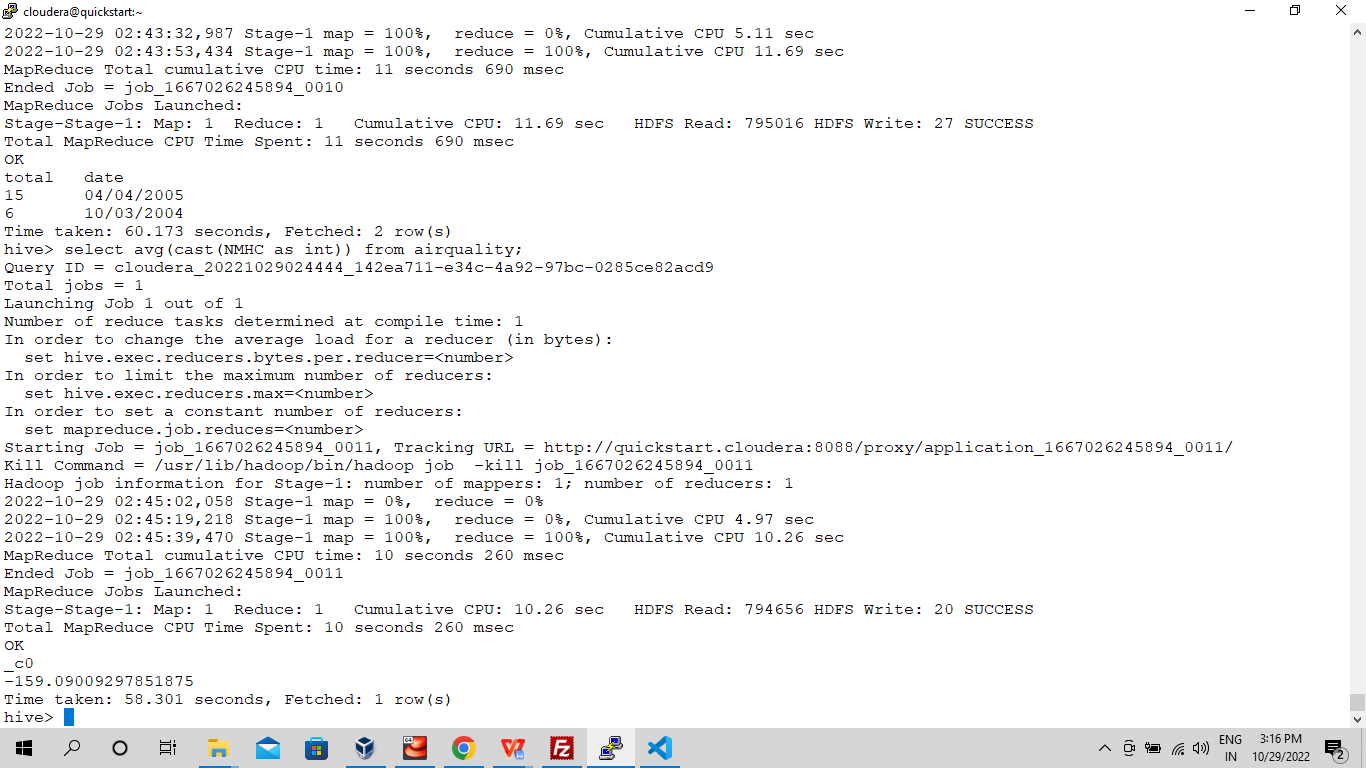


1. **Perform filter operation at least 5 kinds of filter examples**









1. **show and example of regex operation**
2. **alter table operation**

- Alter table airquality rename to air quality 1

1. **drop table operation**

**-** drop table airquality

1. **order by operation**

- select avg(nmhc), date from airquality group by date order by date limit 5;

1. **where clause operations you have to perform**

- select \* from airquality where date = “31/12/2004”;

1. **sorting operation you have to perform**

**-** select avg(nmhc), date from airquality group by date sort by date limit 5;

1. **distinct operation you have to perform**

- select distinct date from airquality limit 5;

1. **like an operation you have to perform**

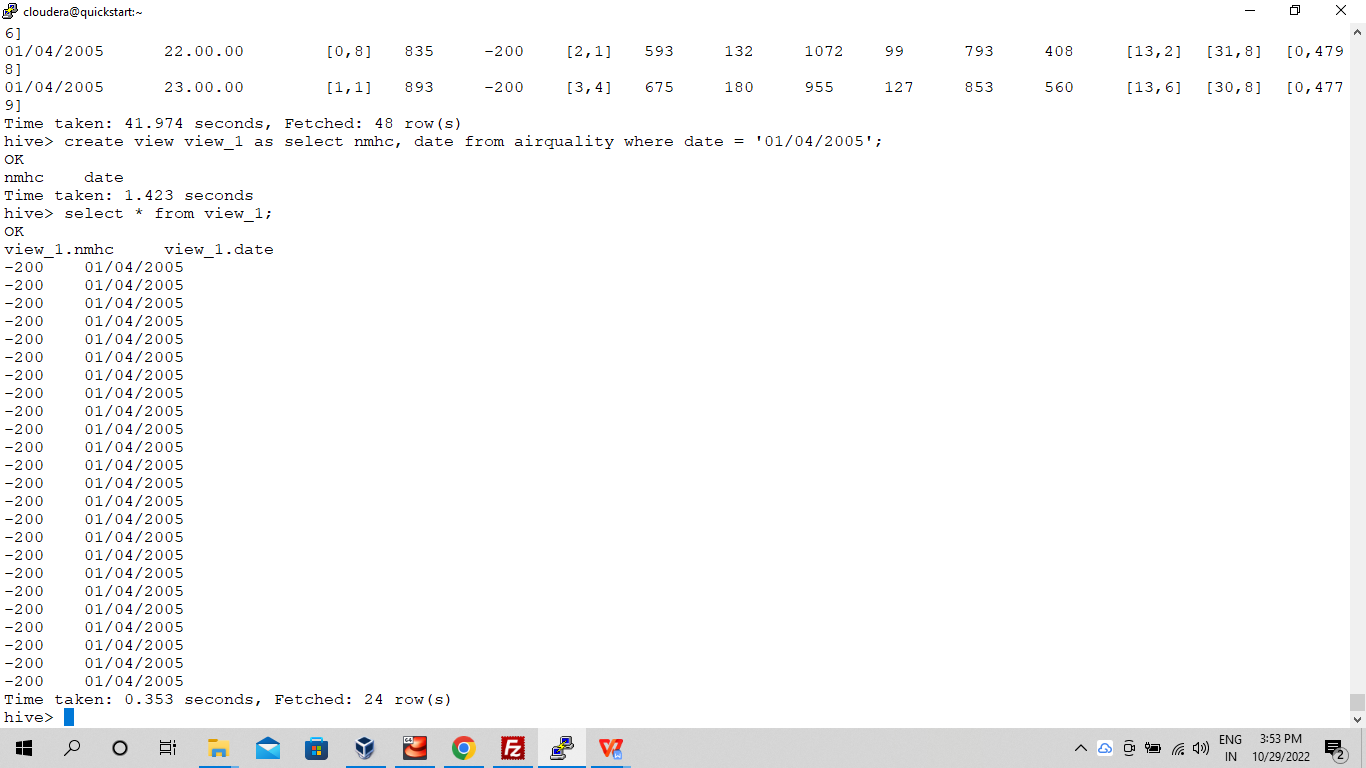
**-** select date from airquality where date like "01%" limit 5;

1. **union operation you have to perform**

**-** select \* from air\_quality where date = '01/04/2005' union all select \* from airquality where date= '31/03/2005';

1. **table view operation you have to perform**

- create view view\_1 as select nmhc, date from airquality where date = '01/04/2005';



**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.**

**-----**