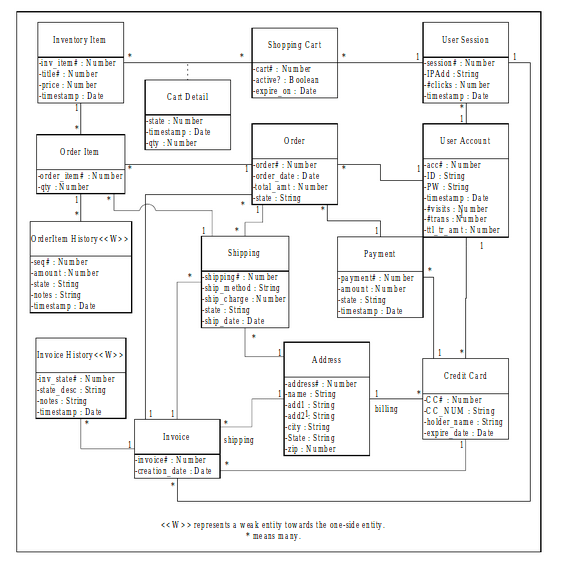
**Big Data Case Study: E-Commerce Data Analysis Using Hadoop**

*E-commerce* or *electronic commerce* is a transaction of buying or selling online. It draws on technologies such as mobile commerce, electronic funds transfer, supply chain management, and much more. The e-commerce field is increasing rapidly across the world. The e-commerce business in India will be worth 38 billion dollars by 2016 and it is estimated to reach 159 billion dollars by 2020.  
The e-commerce firms are growing rapidly all over the world with millions of transactions made every day. So, one needs to analyze that data and draw some useful insights from it.  
Here, we bring to you a business use case of an e-commerce company which wants to analyze their transactions and draw some useful insights out of it, which will be useful for their business development.  
Here is the database design, the E-R diagram, of an e-commerce firm which was found in an [IEEE research paper](https://www.researchgate.net/publication/2359510_Database_Design_for_Real-World_E-Commerce_Systems).  
  
Let’s start with the creation of tables according to this E-R diagram.  
Firstly, we will create a database to store our tables. We have named our database as **ecom.**

create database ecom;

use ecom;

We have now successfully created the database and have moved it into that database too. You can see the same in the screenshot below:  
  
Now, we will create all the tables that are required for our e-commerce website based on the E-R diagram.  
**Creating a table user\_account**

[Get Skilled in Big Data](https://acadgild.com/big-data/big-data-development-training-certification)

create table user\_account

(

ac\_no int,

user\_id varchar(30),

user\_pswd varchar(30),

date datetime,

no\_of\_visit int,

no\_of\_trnsc int,

trnsc\_amt int,

primary key(ac\_no)

);

**Inserting data**

insert into user\_account values(125,'a123','pp284','2011-04-14',5,6,100);

insert into user\_account values(126,'a124','rr999','2012-06-13',2,3,1000);

insert into user\_account values(127,'a125','ab888','2010-07-15',3,9,5000);

insert into user\_account values(128,'a126','bb900','2016-05-15',1,8,7000);

insert into user\_account values(129,'a127','rt007','2012-07-14',4,5,4000);

insert into user\_account values(130,'a128','ss008','2013-03-15',6,2,8000);

**Creating table user\_session\_details**

create table user\_session\_details

(

ac\_no int,

session\_no int,

ip\_address varchar(30),

no\_of\_clicks int,

date datetime,

primary key(session\_no),

foreign key(ac\_no) references user\_account(ac\_no)

);

**Inserting data**

insert into user\_session\_details values(130,11,'10.23.130.162',5,'2016-05-14');

insert into user\_session\_details values(130,10,'11.26.135.167',5,'2014-09-15');

insert into user\_session\_details values(125,9,'11.58.123.151',3,'2012-10-13');

insert into user\_session\_details values(127,3,'11.33.111.188',2,'2015-01-14');

insert into user\_session\_details values(129,15,'11.37.101.142',7,'2014-10-17');

insert into user\_session\_details values(128,8,'11.67.100.123',6,'2012-09-12');

**Creating table shopping\_cart\_details**

create table shopping\_cart\_details

(

ac\_no int,

cart\_num int,

active\_status boolean,

expire\_date datetime,

primary key(cart\_num),

foreign key(ac\_no) references user\_account(ac\_no)

);

**Inserting data**

insert into shopping\_cart\_details values(128,144,'0','2014-11-15');

insert into shopping\_cart\_details values(125,178,'0','2016-03-13');

insert into shopping\_cart\_details values(126,156,'1','2013-07-14');

insert into shopping\_cart\_details values(127,158,'1','2015-02-15');

insert into shopping\_cart\_details value(130,161,'1','2016-04-15');

insert into shopping\_cart\_details value(129,190,'1','2013-01-13');

**Creating table inventory details**

create table inventory\_item\_details

(

inven\_item\_no int,

title int,

price int,

date datetime,

primary key(inven\_item\_no)

);

**Inserting data**

insert into inventory\_item\_details values(188,78,1500,'2013-10-15');

insert into inventory\_item\_details values(122,54,5000,'2014-09-14');

insert into inventory\_item\_details values(167,56,4500,'2015-11-11');

insert into inventory\_item\_details values(134,66,4000,'2014-01-16');

insert into inventory\_item\_details values(156,35,3000,'2015-12-07');

insert into inventory\_item\_details values(177,88,6000,'2013-02-17');

**Creating table address**

create table address

(

adrss\_no int,

name varchar(30),

adrss\_1 varchar(30),

adrss\_2 varchar(30),

city varchar(30),

state varchar(30),

zip int,

primary key(adrss\_no)

);

**Inserting data**

insert into address values(13,'Blossoms','KS layout','Marathalli','Bangalore','Karnataka',560037);

insert into address values(15,'Prestige','Ecity','HSR Layout','Bangalore','Karnataka',560048);

insert into address values(16,'Orchid','vimaan nagar','Whitefield','Pune','Maharashtra',460056);

insert into address values(24,'Alpino','KS layout','connaught place','Delhi','New Delhi',460008);

insert into address values(18,'Galaxy','BTM Layout','Cunningham road','Bangalore','Karnataka',560027);

insert into address values(11,'Pratiksha','TR layout','Andheri East','Mumbai','Maharashtra',760037);

**Creating table credit\_card\_details**

create table credit\_card\_details

(

ac\_no int,

adrss\_no int,

ccard\_no int,

card\_no varchar(30),

holder\_name varchar(30),

expire\_date datetime,

primary key(ccard\_no),

foreign key(ac\_no) references user\_account(ac\_no),

foreign key (adrss\_no) references address(adrss\_no)

);

**Inserting data**

insert into credit\_card\_details values(126,13,345,'AB24563789','Shalini','2019-05-12');

insert into credit\_card\_details values(125,15,679,'HD54654458','Sakshi','2020-06-11');

insert into credit\_card\_details values(127,24,145,'KK98735476','Kiran','2021-11-18');

insert into credit\_card\_details values(128,11,211,'AA54673825','Alex','2025-08-14');

insert into credit\_card\_details values(129,18,978,'CB64738463','Rose','2023-01-16');

insert into credit\_card\_details values(130,16,667,'BB37845678','Lisa','2021-10-10');

**Creating table payment\_details**

create table payment\_details

(

ccard\_no int,

payment\_no int,

amt int,

state varchar(30),

date datetime,

primary key(payment\_no),

foreign key(ccard\_no) references credit\_card\_details(ccard\_no)

);

**Inserting data**

insert into payment\_details values(679,80,6600,'success','2014-11-16');

insert into payment\_details values(345,67,8000,'success','2014-06-21');

insert into payment\_details values(211,56,9000,'success','2015-02-25');

insert into payment\_details values(145,78,1000,'failed','2015-05-22');

insert into payment\_details values(667,12,5000,'success','2013-12-19');

insert into payment\_details values(978,24,2000,'failed','2015-07-11');

**Creating table invoice\_details**

create table invoice\_details

(

invoice\_num int,

creation\_date datetime,

session\_no int,

ccard\_no int,

adrss\_no int,

primary key(invoice\_num),

foreign key(session\_no) references user\_session\_details(session\_no),

foreign key(ccard\_no) references credit\_card\_details(ccard\_no),

foreign key(adrss\_no) references address(adrss\_no)

);

**Inserting data**

insert into invoice\_details values(188,'2014-02-05',11,345,15);

insert into invoice\_details values(101,'2014-04-07',10,145,13);

insert into invoice\_details values(111,'2015-03-10',03,211,16);

insert into invoice\_details values(197,'2015-06-15',09,978,11);

insert into invoice\_details values(125,'2014-07-21',08,679,24);

insert into invoice\_details values(167,'2015-01-11',15,667,18);

**Creating table order\_details**

create table order\_details

(

order\_num int,

payment\_no int,

ac\_no int,

invoice\_num int,

order\_date datetime,

total\_amt int,

state varchar(30),

primary key(order\_num),

foreign key(ac\_no) references user\_account(ac\_no),

foreign key(payment\_no) references payment\_details(payment\_no),

foreign key(invoice\_num) references invoice\_details(invoice\_num)

);

**Inserting data**

insert into order\_details values(16,80,125,188,'2015-08-14',6000,'placed');

insert into order\_details values(18,56,127,111,'2014-05-12',8000,'cancelled');

insert into order\_details values(21,67,126,101,'2015-01-10',9000,'cancelled');

insert into order\_details values(28,12,129,197,'2015-02-18',4000,'placed');

insert into order\_details values(39,78,128,125,'2014-01-16',7000,'placed');

insert into order\_details values(38,24,130,167,'2015-06-19',5500,'not placed');

**Creating table shipping\_details**

create table shipping\_details(

adrss\_no int,

order\_num int,

invoice\_num int,

shipping\_no int,

shipping\_method varchar(30),

shipping\_charge int,

state varchar(30),

shipping\_date datetime,

primary key(shipping\_no),

foreign key(adrss\_no) references address(adrss\_no),

foreign key(order\_num)references order\_details(order\_num),

foreign key(invoice\_num)references invoice\_details(invoice\_num)

);

**Inserting data**

insert into shipping\_details values(13,16,188,109,'EKART',1000,'delivered','2015-09-12');

insert into shipping\_details values(15,18,111,105,'FIRST FLIGHT',6000,'cancelled','2014-01-11');

insert into shipping\_details values(24,21,101,110,'DHL',2000,'cancelled','2015-04-09');

insert into shipping\_details values(16,39,125,111,'EKART',3000,'cancelled','2014-10-15');

insert into shipping\_details values(11,38,197,124,'INDIA POST',4000,'delivered','2015-08-14');

insert into shipping\_details values(18,28,167,178,'BLUE DART',5000,'delivered','2015-02-18');

**Creating table order\_item\_details**

create table order\_item\_details

(

order\_item\_no int,

quantity int,

order\_num int,

shipping\_no int,

inven\_item\_no int,

primary key(order\_item\_no),

foreign key(order\_num) references order\_details(order\_num),

foreign key(shipping\_no) references shipping\_details(shipping\_no),

foreign key(inven\_item\_no) references inventory\_item\_details(inven\_item\_no)

);

**Inserting data**

insert into order\_item\_details values(111,13,16,109,188);

insert into order\_item\_details values(112,10,21,105,167);

insert into order\_item\_details values(189,9,18,110,122);

insert into order\_item\_details values(127,8,39,124,156);

insert into order\_item\_details values(199,14,38,178,134);

insert into order\_item\_details values(146,15,28,111,177);

**Creating table invoice\_history**

create table invoice\_history

(

invoice\_state int,

state\_desc varchar(30),

notes varchar(30),

date datetime,

invoice\_num int,

foreign key(invoice\_num) references invoice\_details(invoice\_num)

);

**Inserting data**

insert into invoice\_history values(1,'Valid','Good','2012-09-24',188);

insert into invoice\_history values(0,'Invalid','Not good','2015-01-05',101);

insert into invoice\_history values(0,'Invalid','Not good','2012-09-11',111);

insert into invoice\_history values(1,'valid','Good','2012-09-09',197);

insert into invoice\_history values(1,'valid','Good','2012-09-14',167);

insert into invoice\_history values(0,'Invalid','Not good','2014-08-24',125);

**Creating table order\_items\_history**

create table order\_items\_history

(

seq\_no int,

amt int,

state varchar(30),

notes varchar(30),

date datetime,

order\_item\_no int,

primary key (seq\_no),

foreign key(order\_item\_no) references order\_item\_details(order\_item\_no)

);

**Inserting data**

insert into order\_items\_history values(10,7000,'Delivered','Good','2014-09-12',111);

insert into order\_items\_history values(11,8000,'cancelled','not good','2015-05-10',112);

insert into order\_items\_history values(12,9000,'Cancelled','not good','2014-10-08',127);

insert into order\_items\_history values(13,5000,'Cancelled','Not good','2015-03-14',189);

insert into order\_items\_history values(14,6000,'Delivered','Good','2015-06-11',199);

insert into order\_items\_history values(15,3000,'Delivered','Good','2014-07-13',146);

**Creating table card\_details**

create table cart\_details

(

c\_no int,

inven\_item\_no int,

state varchar(30),

cart\_num int,

date datetime,

quantity int,

primary key(c\_no),

foreign key(inven\_item\_no)references inventory\_item\_details(inven\_item\_no),

foreign key(cart\_num) references shopping\_cart\_details(cart\_num)

);

**Inserting data**

insert into cart\_details values(1,188,'0',144,'2016-11-15',13);

insert into cart\_details values(2,122,'0',178,'2011-03-13',9);

insert into cart\_details values(3,167,'1',156,'2015-07-14',10);

insert into cart\_details values(4,134,'1',158,'2015-02-15',14);

insert into cart\_details value(5,156,'1',161,'2016-06-15',8);

insert into cart\_details value(6,177,'1',190,'2015-01-13',15);

This way, we have created all the 14 tables in total accordance with the database design. We have also inserted some data into these tables.  
  
We have our databases ready now. The only thing remaining is the Big Data Analytics to be done on this data. So, first, we need to move this data into the HDFS. We do that using Sqoop.  
We will implement the Sqoop Incremental Import to import this data into the HDFS incrementally from MySQL.  
Below are the Sqoop scripts given to transfer this data into the HDFS incrementally. First, we will transfer the data that is present in the MySQL table. Next, when the MySQL table is updated, we need to specify the last value in the Sqoop import command so that all the newly added records will get imported into the HDFS.  
  
**Sqoop Incremental Import**  
**To transfer user\_account data**

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table user\_account --target-dir '/ecom/user\_account' --incremental append --check-column ac\_no -m 1 --driver com.mysql.jdbc.Driver

After running this job, you will get the “last value option” in the job metrics. Note down the last value of the check column record, and the next time you want to insert some new data, you need to mention the last value, here the last value is 130 and we are mentioning it in the Sqoop command as shown below.

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table user\_account --target-dir '/ecom/user\_account' --incremental append --check-column ac\_no --last-value 130 -m 1 --driver com.mysql.jdbc.Driver

**To transfer credit\_card\_details data**

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table credit\_card\_details --target-dir '/ecom/credit\_card\_details' --incremental append --check-column ccard\_no -m 1 --driver com.mysql.jdbc.Driver

After running this job, you will get the “last value option” in the job metrics, note down the last value of the check column record. The next time you want to insert the new data, you need to mention the last value. Here the last value is 978, and we will be mentioning it in the Sqoop command as shown below.

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table credit\_card\_details --target-dir '/ecom/credit\_card\_details' --incremental append --check-column ccard\_no --last-value 978 -m 1 --driver com.mysql.jdbc.Driver

**To transfer address data**

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table address --target-dir '/ecom/address' --incremental append --check-column adrss\_no -m 1 --driver com.mysql.jdbc.Driver

After running this job, you will get the “last value option” in the job metrics. Note down the last value of the check column record and the next time when you want to insert this new data, you need to mention the last value. Here, the last value is 24 and we are mentioning it in the Sqoop command as shown below:

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table address --target-dir '/ecom/address' --incremental append --check-column adrss\_no --last-value 24 -m 1 --driver com.mysql.jdbc.Driver

**To transfer Payment\_details data**

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table payment\_details --target-dir '/ecom/payment\_details' --incremental append --check-column payment\_no -m 1 --driver com.mysql.jdbc.Driver

After running this job, you will get the “last value option” in the job metrics. Note down the last value of the check column record and the next time when you want to insert this new data, you need to mention the last value. Here, the last value is 80, we are mentioning it in the Sqoop command as shown below:

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table payment\_details --target-dir '/ecom/payment\_details' --incremental append --check-column payment\_no --last-value 80 -m 1 --driver com.mysql.jdbc.Driver

**To transfer Shipping\_details data**

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table shipping\_details --target-dir '/ecom/shipping\_details' --incremental append --check-column shipping\_no -m 1 --driver com.mysql.jdbc.Driver

After running this job, you will get the “last value option” in the job metrics. Note down the last value of the check column record and the next time when you want to insert this new data, you need to mention the last value. Here, the last value is 178, we are mentioning it in the Sqoop command as shown below:

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table shipping\_details --target-dir '/ecom/shipping\_details' --incremental append --check-column shipping\_no --last-value 178 -m 1 --driver com.mysql.jdbc.Driver

**To transfer order\_items\_history data**

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table order\_items\_history --target-dir '/ecom/order\_items\_history' --incremental append --check-column seq\_no -m 1 --driver com.mysql.jdbc.Driver

After running this job, you will get the “last value option” in the job metrics. Note down the last value of the check column record and the next time when you want to insert this new data, you need to mention the last value. Here, the last value is 15, we are mentioning it in the Sqoop command as shown below:

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table order\_items\_history --target-dir '/ecom/order\_items\_history' --incremental append --check-column seq\_no --last-value 15 -m 1 --driver com.mysql.jdbc.Driver

**To transfer Order\_item\_details data**

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table order\_item\_details --target-dir '/ecom/order\_item\_details' --incremental append --check-column order\_item\_no -m 1 --driver com.mysql.jdbc.Driver

After running this job, you will get the “last value option” in the job metrics. Note down the last value of the check column record and the next time when you want to insert this new data, you need to mention the last value. Here, the last value is 199, we are mentioning it in the Sqoop command as shown below:

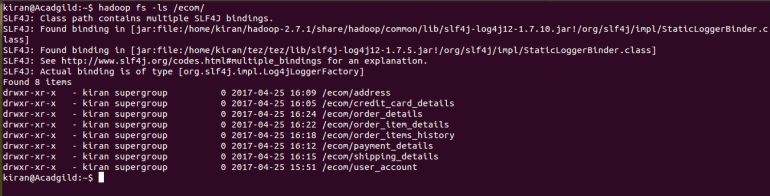
sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table order\_item\_details --target-dir '/ecom/order\_item\_details' --incremental append --check-column order\_item\_no --last-value 199 -m 1 --driver com.mysql.jdbc.Driver

**To transfer Order\_details data**

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table order\_details --target-dir '/ecom/order\_details' --incremental append --check-column order\_num -m 1 --driver com.mysql.jdbc.Driver

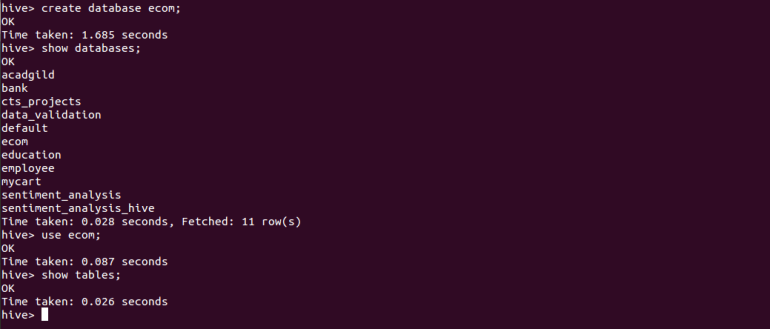
After running this job, you will get the “last value option” in the job metrics. Note down the last value of the check column record and the next time when you want to insert this new data, you need to mention the last value. Here, the last value is 39, we are mentioning it in the Sqoop command as shown below:

sqoop import --connect jdbc:mysql://localhost/ecom --username 'root' --P --table order\_details --target-dir '/ecom/order\_details' --incremental append --check-column order\_num --last-value 39 -m 1 --driver com.mysql.jdbc.Driver

We have successfully exported the that is required for our analysis. You can see the same in the screenshot below:  
  
Let’s see how to analyze this data using Hive.  
First, we will create a database in Hive as shown below:

create database ecom;

use ecom;

We have successfully created the database, we have also entered into the database.  
  
Now, we will create external tables in Hive referencing to the locations of the data that are present in the HDFS. Even if the data is updated, we can query the updated data in Hive as we are pointing the location of the data. If we delete the table too, our data will be present in that particular directory.  
**Creating user\_account table**

create external table user\_account(

ac\_no int,

user\_id STRING,

user\_pswd STRING,

date1 string,

no\_of\_visits int,

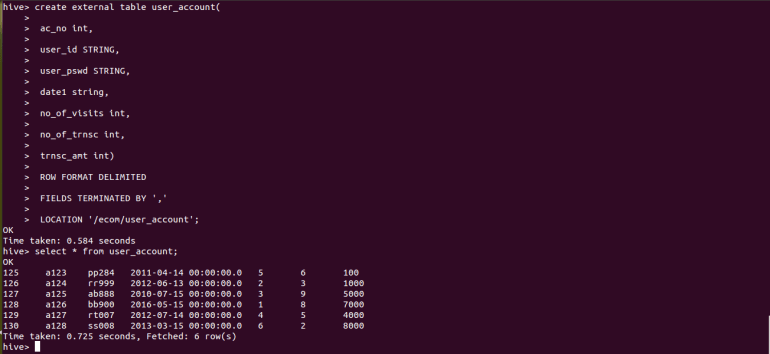
no\_of\_trnsc int,

trnsc\_amt int)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LOCATION '/ecom/user\_account';

  
**Creating order\_details table**

create external table order\_details(

order\_num int,

payment\_no int,

ac\_no int,

invoice\_num int,

order\_date string,

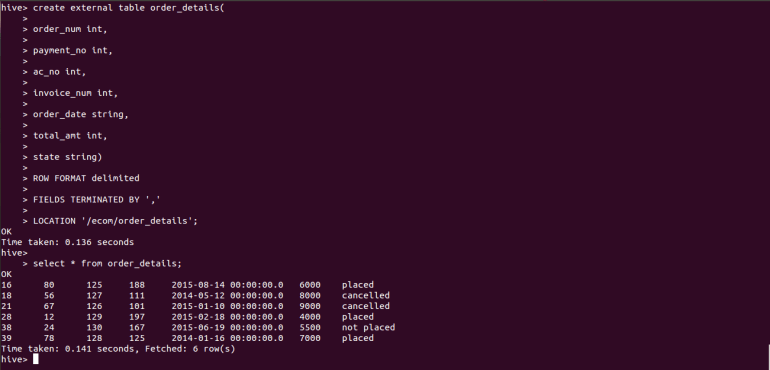
total\_amt int,

state string)

ROW FORMAT delimited

FIELDS TERMINATED BY ','

LOCATION '/ecom/order\_details';

  
**Creating order\_item\_details  table**

create external table order\_item\_details(

order\_item\_no int,

quantity int,

order\_num int,

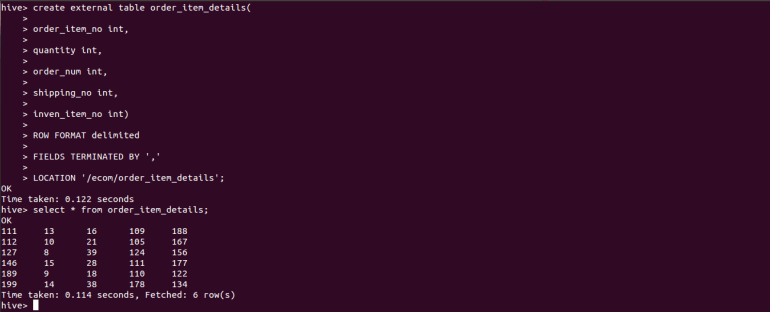
shipping\_no int,

inven\_item\_no int)

ROW FORMAT delimited

FIELDS TERMINATED BY ','

LOCATION '/ecom/order\_item\_details';

  
**Creating table order\_items\_history table**

create external table order\_items\_history(

seq\_no int,

amt int,

state string,

notes string,

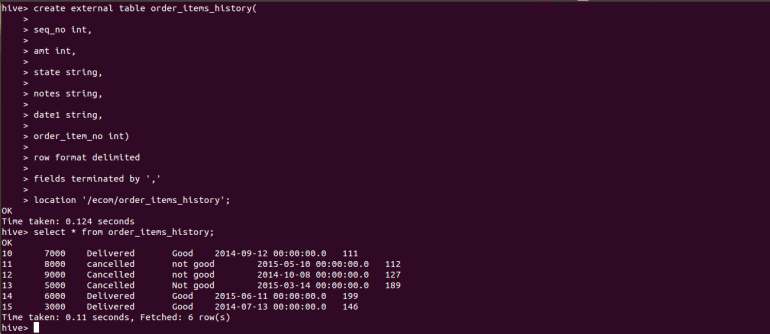
date1 string,

order\_item\_no int)

row format delimited

fields terminated by ','

location '/ecom/order\_items\_history';

  
**Creating shipping\_details table**

create external table shipping\_details(

adrss\_no int,

order\_num int,

invoice\_num int,

shipping\_no int,

shipping\_method string,

shipping\_charge int,

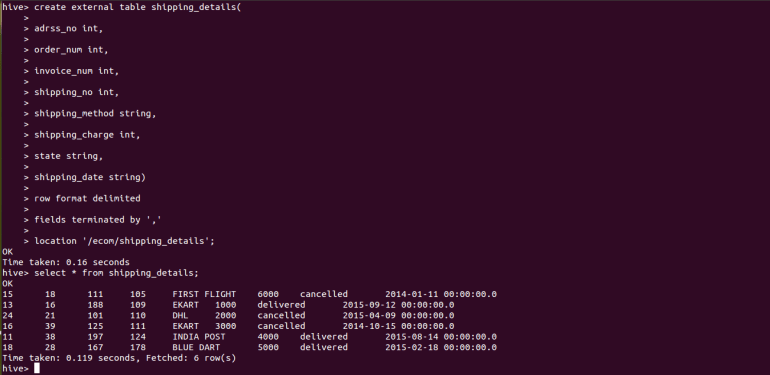
state string,

shipping\_date string)

row format delimited

fields terminated by ','

location '/ecom/shipping\_details';

  
**Creating payment\_details table**

create external table payment\_details(

ccard\_no int,

payment\_no int,

amt int,

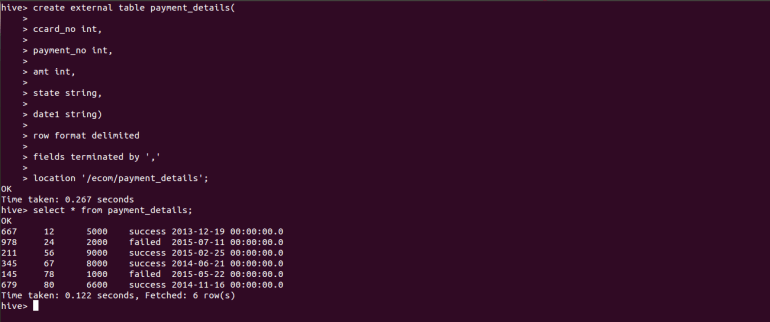
state string,

date1 string)

row format delimited

fields terminated by ','

location '/ecom/payment\_details';

  
**Creating address table**

create external table address(

adrss\_no int,

name string,

adrss\_1 string,

adrss\_2 string,

city string,

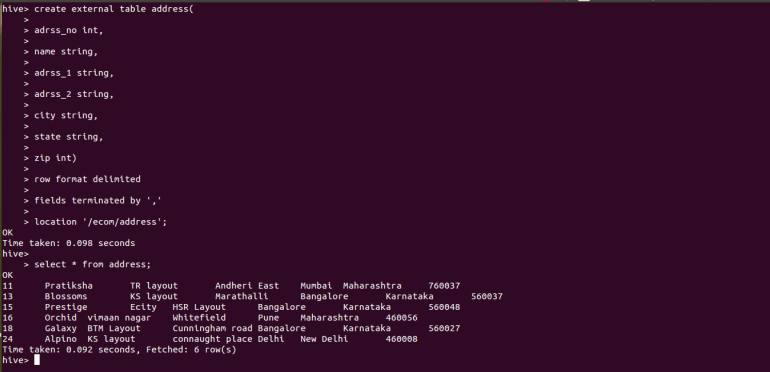
state string,

zip int)

row format delimited

fields terminated by ','

location '/ecom/address';

  
**Creating credit\_card\_details table**

create external table credit\_card\_details(

ac\_no int,

adrss\_no int,

ccard\_no int,

card\_no string,

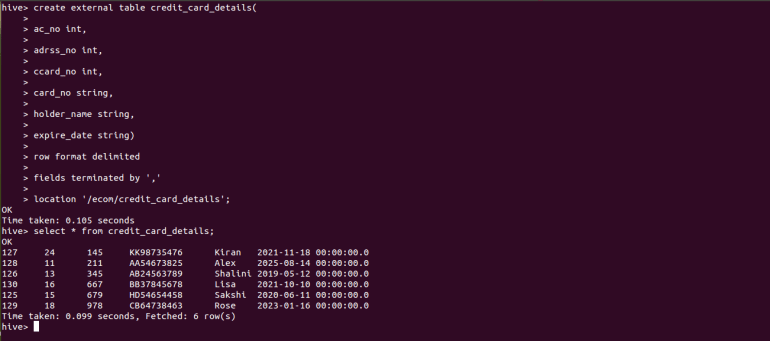
holder\_name string,

expire\_date string)

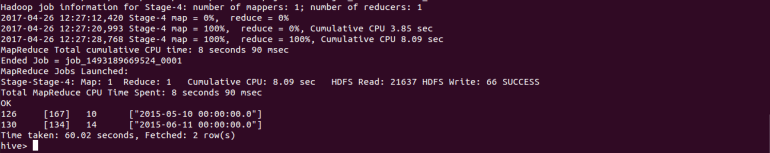
row format delimited

fields terminated by ','

location '/ecom/credit\_card\_details';

  
Here, we have created the 8 tables in Hive that are required for our analysis part, and the data is also populated into those tables. You can see the 8 tables in the screenshot below:  
  
Now, let’s jump to the analysis part.  
**1.Find the details of various items purchased by various customers in each quarter of the year separately.**  
In this problem statement, we need to find out the various items purchased by various customers in each quarter of the year. Here, we need to find out the item\_no from the inventory and the customer’s ac\_no from the user\_account and the date an item purchased was purchased on from the item history.  
Presented below is the Hive query to find out the details mentioned above.

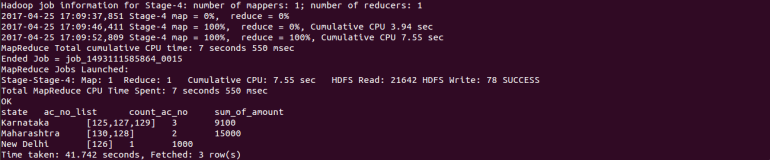
select account.ac\_no as ac\_no,COLLECT\_SET(order\_item\_details.inven\_item\_no) as item\_no\_list,SUM(order\_item\_details.quantity) as quantity,COLLECT\_SET(order\_items\_history.date1) as order\_date from user\_account account JOIN order\_details details ON (account.ac\_no=details.ac\_no) JOIN order\_item\_details ON (details.order\_num=order\_item\_details.order\_num) JOIN order\_items\_history ON (order\_item\_details.order\_item\_no=order\_items\_history.order\_item\_no) where order\_items\_history.date1>='2015-05-01' AND order\_items\_history.date1<='2015-08-31' group by account.ac\_no;

You can see the results in the screenshot below:  


1. **Find out which part of the world gets the most number of customers and total purchases as compared to the past.**

In this problem statement, we need to find out which part of the world has more number of customers and their total purchases in the past. For this, we will examine the details state-wise and find out which state has how many number of customers and those customers’ total purchases.  
We can find out about the states from the address table, customers will be counted from the user\_account table and their transactions will be calculated from the user\_account table itself.  
Given below is a Hive query that can be used to evaluate the above problem statement.

select addres.state as state,COLLECT\_SET(account.ac\_no) as ac\_no\_list,COUNT(account.ac\_no) as count\_ac\_no,SUM(account.trnsc\_amt) as sum\_of\_amount from user\_account account JOIN order\_details details ON account.ac\_no=details.ac\_no JOIN shipping\_details ship ON details.order\_num=ship.order\_num JOIN address addres ON ship.adrss\_no=addres.adrss\_no group by addres.state;

You can see the result in the screenshot below:  
  
When data is updated, even then we can run the same queries in Hive, and Hive gives us the updated results as the data in Hive gets updated automatically as we have given our dataset location referencing to an external path in the HDFS.