Assignment7.3

Explain the below concepts with an example in brief.

Hive Data Definitions

CREATE DATABASE/SCHEMA, TABLE, VIEW, FUNCTION, INDEX

CREATE TABLE page_view(viewTime INT, userid BIGINT,

page_url STRING, referrer_url STRING,

ip STRING COMMENT 'IP Address of the User')

COMMENT 'This is the page view table'

PARTITIONED BY(dt STRING, country STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\001'

STORED AS SEQUENCEFILE;

The above command creates a HIVE MANAGED table page_view

DROP DATABASE/SCHEMA, TABLE, VIEW, INDEX

DROP TABLE IF EXISTS page_view

DROP TABLE removes metadata and data for this table. The data is actually moved to the .Trash/Current directory if Trash is configured The metadata is completely lost.

When dropping an EXTERNAL table, data in the table will *NOT* be deleted from the file system.

TRUNCATE TABLE

TRUNCATE TABLE page_view

Removes all rows from a table or partition(s). The rows will be trashed if the filesystem Trash is enabled, otherwise they are deleted.Currently the target table should be native/managed table or an exception will be thrown. User can specify partial partition_spec for truncating multiple partitions at once and omitting partition_spec will truncate all partitions in the table.

ALTER DATABASE/SCHEMA, TABLE, VIEW

ALTER TABLE table_name RENAME TO new_table_name;

This statement lets you change the name of a table to a different name. You can add columns/partitions, change SerDe, add table and SerDe properties, or rename the table itself. Similarly, alter table partition statements allow you change the properties of a specific partition in the named table.

The column change command will only modify Hive's metadata, and will not modify data. Users should make sure the actual data layout of the table/partition conforms with the metadata definition.

MSCK REPAIR TABLE (or ALTER TABLE RECOVER PARTITIONS)

SHOW DATABASES/SCHEMAS, TABLES, TBLPROPERTIES, VIEWS, PARTITIONS, FUNCTIONS, INDEX[ES], COLUMNS, CREATE TABLE

SHOW DATABASES or SHOW SCHEMAS lists all of the databases defined in the metastore. The uses of SCHEMAS and DATABASES are interchangeable – they mean the same thing.

SHOW TABLES lists all the base tables and views in the current database (or the one explicitly named using the IN clause) with names matching the optional regular expression. Wildcards in the regular expression can only be '*' for any character(s) or '|' for a choice. Examples are 'page_view', 'page_v*', '*view|page*', all which will match the 'page_view' table. Matching tables are listed in alphabetical order. It is not an error if there are no matching tables found in metastore. If no regular expression is given then all tables in the selected database are listed.

SHOW VIEWS lists all the views in the current database.

SHOW PARTITIONS lists all the existing partitions for a given base table. Partitions are listed in alphabetical order.

SHOW TABLE EXTENDED will list information for all tables matching the given regular expression

DESCRIBE DATABASE/SCHEMA, table_name, view_name

PARTITION statements are usually options of TABLE statements, except for SHOW PARTITIONS.

Data Definition Language (DDL)

DDL statements are used to build and modify the tables and other objects in the database.

Example :

CREATE, DROP, TRUNCATE, ALTER, SHOW, DESCRIBE Statements.

Go to Hive shell by giving the command sudo hive and enter the command ‘create database<data base name>’ to create the new database in the Hive.

```
hive> create database retail;
OK
Time taken: 5.275 seconds
hive>
```

To list out the databases in Hive warehouse, enter the command ‘show databases’.

```
hive> show databases;
OK
default
retail
Time taken: 0.228 seconds
hive>
```

The database creates in a default location of the Hive warehouse. In Cloudera, Hive database store in a /user/hive/warehouse.

The command to use the database is USE <data base name>

```
hive> use retail;
OK
Time taken: 0.023 seconds
hive>
```

Copy the input data to HDFS from local by using the copy From Local command.

When we create a table in hive, it creates in the default location of the hive warehouse. – “/user/hive/warehouse”, after creation of the table we can move the data from HDFS to hive table.

The following command creates a table with in location of “/user/hive/warehouse/retail.db”

Note : retail.db is the database created in the Hive warehouse.

```
hive> create table txnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE,category STRING, product STRING, city STRIN
G, state STRING, spendby STRING) row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 1.163 seconds
hive>
```

Describe provides information about the schema of the table.

```
hive> describe txnrecords;
OK
txnno   int
txndate string
custno  int
amount  double
category        string
product string
city    string
state   string
spendby string
Time taken: 0.122 seconds
hive>
```

Hive Data Manipulations

Data Manipulation Language (DML)

DML statements are used to retrieve, store, modify, delete, insert and update data in the database.

Example :

LOAD, INSERT Statements.

Syntax :

LOAD data <LOCAL> inpath <file path> into table [tablename]

The Load operation is used to move the data into corresponding Hive table. If the keyword local is specified, then in the load command will give the local file system path. If the keyword local is not specified we have to use the HDFS path of the file.

Here are some examples for the LOAD data LOCAL command

```
hive> create table customer(custno string, firstname string, lastname string, age int,profession string) row format delimited
fields terminated by ',';
OK
Time taken: 0.102 seconds
hive>
```

```
hive> load data local inpath '/home/cloudera/Desktop/blog/custs' into table customer;
Copying data from file:/home/cloudera/Desktop/blog/custs
Copying file: file:/home/cloudera/Desktop/blog/custs
Loading data to table retail.customer
OK
Time taken: 0.227 seconds
hive>
```

After loading the data into the Hive table we can apply the Data Manipulation Statements or aggregate functions retrieve the data.

Example to count number of records:

Count aggregate function is used count the total number of the records in a table.

```
hive> select count(*) from txnrecords;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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 mapred.reduce.tasks=<number>
Starting Job = job_201402270420_0005, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201402270420_0005
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:8021 -kill job_201402270420_0005
2014-02-28 20:02:41,231 Stage-1 map = 0%,  reduce = 0%
2014-02-28 20:02:48,293 Stage-1 map = 50%,  reduce = 0%
2014-02-28 20:02:49,309 Stage-1 map = 100%,  reduce = 0%
2014-02-28 20:02:55,350 Stage-1 map = 100%,  reduce = 33%
2014-02-28 20:02:56,367 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201402270420_0005
OK
50000
Time taken: 19.027 seconds
hive>
```

‘create external’ Table :

The create external keyword is used to create a table and provides a location where the table will create, so that Hive does not use a default location for this table.
An EXTERNAL table points to any HDFS location for its storage, rather than default storage.

```
hive> create external table example_customer(custno string, firstname string, lastname string, age int,profession string) row
format delimited fields terminated by ',' LOCATION '/user/external';
OK
Time taken: 0.059 seconds
hive>
```

Insert Command:

The insert command is used to load the data Hive table. Inserts can be done to a table or a partition.

• INSERT OVERWRITE is used to overwrite the existing data in the table or partition.

• INSERT INTO is used to append the data into existing data in a table. (Note: INSERT INTO syntax is work from the version 0.8)

```
hive> from customer cus insert overwrite table example_customer select cus.custno,cus.firstname,cus.lastname,cus.age,cus.profe
ssion;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201402270420_0007, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201402270420_0007
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:8021 -kill job_201402270420_0007
2014-02-28 20:40:39,866 Stage-1 map = 0%,  reduce = 0%
2014-02-28 20:40:41,871 Stage-1 map = 100%,  reduce = 0%
2014-02-28 20:40:42,876 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201402270420_0007
Loading data to table retail.example_customer
Deleted hdfs://localhost/user/external
Table retail.example_customer stats: [num_partitions: 0, num_files: 0, num_rows: 0, total_size: 0]
9999 Rows loaded to example_customer
OK
Time taken: 5.786 seconds
hive>
```

Example for 'Partitioned By' and 'Clustered By' Command :

'Partitioned by' is used to divided the table into the Partition and can be divided in to buckets by using the 'Clustered By' command.

```
hive> create table txnrecsByCat(txnno INT, txndate STRING, custno INT, amount DOUBLE,product STRING, city STRING, state STRING
, spendby STRING) partitioned by (category STRING) clustered by (state) INTO 10 buckets row format delimited fields terminated
 by ',' stored as textfile;
OK
Time taken: 0.101 seconds
hive>
```

```
hive> from txnrecords txn INSERT OVERWRITE TABLE record PARTITION(category)select txn.txnno,txn.txndate,txn.custno,txn.amount,
txn.product,txn.city,txn.state,txn.spendby, txn.category;
FAILED: Error in semantic analysis: Dynamic partition strict mode requires at least one static partition column. To turn this
off set hive.exec.dynamic.partition.mode=nonstrict
```

When we insert the data Hive throwing errors, the dynamic partition mode is strict and dynamic partition not enabled (by Jeff at dresshead website). So we need to set the following parameters in Hive shell.

set hive.exec.dynamic.partition=true;

To enable dynamic partitions, by default, it's false

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

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

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

```
hive> from txnrecords txn INSERT OVERWRITE TABLE record PARTITION(category)select txn.txnno,txn.txndate,txn.custno,txn.amount,
txn.product,txn.city,txn.state,txn.spendby, txn.category;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 10
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 mapred.reduce.tasks=<number>
Starting Job = job_201402270420_0006, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_201402270420_0006
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:8021 -kill job_201402270420_0006
2014-02-28 20:18:22,243 Stage-1 map = 0%,  reduce = 0%
2014-02-28 20:18:29,289 Stage-1 map = 100%,  reduce = 0%
2014-02-28 20:18:39,352 Stage-1 map = 100%,  reduce = 10%
2014-02-28 20:18:40,360 Stage-1 map = 100%,  reduce = 20%
2014-02-28 20:18:49,412 Stage-1 map = 100%,  reduce = 40%
2014-02-28 20:18:58,456 Stage-1 map = 100%,  reduce = 50%
2014-02-28 20:18:59,459 Stage-1 map = 100%,  reduce = 60%
2014-02-28 20:19:09,506 Stage-1 map = 100%,  reduce = 80%
2014-02-28 20:19:18,547 Stage-1 map = 100%,  reduce = 90%
2014-02-28 20:19:19,554 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201402270420_0006
```

Partition is done by the category and can be divided in to buckets by using the ‘Clustered By’ command.

The ‘Drop Table’ statement deletes the data and metadata for a table. In the case of external tables, only the metadata is deleted.

```
hive> drop table customer;
OK
Time taken: 0.922 seconds
```

The ‘Drop Table’ statement deletes the data and metadata for a table. In the case of external tables, only the metadata is deleted.

Load data local inpath ‘aru.txt’ into table tablename and then we check employee1 table by using Select * from table name command

```
hive> load data local inpath 'aru.txt' into table employee1;
Copying data from file:/home/cloudera/aru.txt
Copying file: file:/home/cloudera/aru.txt
Loading data to table arushi.employee1
OK
Time taken: 0.434 seconds
hive> select * from employee1;
OK
Anu     10      5000.0  Bangalore
Alok    20      10000.0 Chennai
Amod    30      20000.0 Pune
Om      40      50000.0 Delhi
Time taken: 0.213 seconds
```

To count the number of records in table by using Select count(*) from txnrecords;

```
hive> select count(*) from employee1;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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 mapred.reduce.tasks=<number>
Starting Job = job_201312102209_0008, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0008
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0008
2013-12-11 00:58:36,125 Stage-1 map = 0%,  reduce = 0%
2013-12-11 00:58:39,154 Stage-1 map = 100%,  reduce = 0%
2013-12-11 00:58:46,204 Stage-1 map = 100%,  reduce = 33%
2013-12-11 00:58:47,214 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0008
OK
4
Time taken: 14.897 seconds
```

Aggregation :

Select count (DISTINCT category) from tablename;

This command will count the different category of ‘cate’ table. Here there are 3 different categories.

Suppose there is another table cate where f1 is field name of category.

```
hive> select * from cate;
OK
category1       1000
category2       200
category1       1000
category3       5000
category2       200
category1       1000
category2       200
category1       1000
category2       200
category3       5000
Time taken: 0.219 seconds
```

```
hive> select count(distinct f1) from cate;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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 mapred.reduce.tasks=<number>
Starting Job = job_201312102209_0010, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0010
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0010
2013-12-11 01:04:07,180 Stage-1 map = 0%,  reduce = 0%
2013-12-11 01:04:09,190 Stage-1 map = 100%,  reduce = 0%
2013-12-11 01:04:16,224 Stage-1 map = 100%,  reduce = 33%
2013-12-11 01:04:17,231 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0010
OK
3
Time taken: 13.577 seconds
```

Grouping :

Group command is used to group the result-set by one or more columns.

Select category, sum(amount) from txt records group by category

It calculates the amount of same category.

```
hive> select f1, sum(f2) from cate group by f1;
Total MapReduce 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 mapred.reduce.tasks=<number>
Starting Job = job_201312102209_0011, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0011
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0011
2013-12-11 01:05:36,284 Stage-1 map = 0%,  reduce = 0%
2013-12-11 01:05:38,292 Stage-1 map = 100%,  reduce = 0%
2013-12-11 01:05:45,326 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0011
OK
category1       4000
category2       800
category3       10000
Time taken: 12.453 seconds
```

The result one table is stored in to another table.

Create table newtablename as select * from oldtablename;

```
hive> create table result as select * from cate;
Total MapReduce jobs = 2
Launching Job 1 out of 2
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201312102209_0012, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0012
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0012
2013-12-11 01:09:44,943 Stage-1 map = 0%,  reduce = 0%
2013-12-11 01:09:46,957 Stage-1 map = 100%,  reduce = 0%
2013-12-11 01:09:47,970 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0012
Ended Job = 20115431, job is filtered out (removed at runtime).
Launching Job 2 out of 2
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201312102209_0013, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0013
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0013
2013-12-11 01:09:50,216 Stage-2 map = 0%,  reduce = 0%
2013-12-11 01:09:51,224 Stage-2 map = 100%,  reduce = 0%
2013-12-11 01:09:52,230 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0013
```

Join Command :

Here one more table is created in the name ‘mailid’

```
hive> create table mailid(name string, email string)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.081 seconds
```

```
hive> select * from mailid;
OK
anu     anu@gmail.com
om      om@yahoo.com
Anu     anu@gmail.com
Om      om@yahoo.com
Alok    alok@gmail.com
Time taken: 0.126 seconds
```

Join Operation:

A Join operation is performed to combining fields from two tables by using values common to each.

```
hive> select a.name,a.age,a.salary,b.email from employee1 a
    > join mailid b on a.name = b.name;
Total MapReduce 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 mapred.reduce.tasks=<number>
Starting Job = job_201312102209_0017, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0017
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0017
2013-12-11 01:30:34,669 Stage-1 map = 0%,  reduce = 0%
2013-12-11 01:30:36,677 Stage-1 map = 67%,  reduce = 0%
2013-12-11 01:30:38,688 Stage-1 map = 100%,  reduce = 0%
2013-12-11 01:30:44,721 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0017
OK
Alok    20      10000.0 alok@gmail.com
Anu     10      5000.0  anu@gmail.com
Om      40      50000.0 om@yahoo.com
```

Left Outer Join:

The result of a left outer join (or simply left join) for tables A and B always contains all records of the “left” table (A), even if the join-condition does not find any matching record in the “right” table (B).

```
hive> select a.name,a.age,a.salary,b.email from employee1 a
    > left outer join mailid b on a.name = b.name;
```

```
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 mapred.reduce.tasks=<number>
Starting Job = job_201312102209_0018, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0018
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0018
2013-12-11 01:35:13,880 Stage-1 map = 0%,  reduce = 0%
2013-12-11 01:35:15,887 Stage-1 map = 67%,  reduce = 0%
2013-12-11 01:35:17,897 Stage-1 map = 100%,  reduce = 0%
2013-12-11 01:35:22,920 Stage-1 map = 100%,  reduce = 33%
2013-12-11 01:35:23,926 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0018
OK
Alok    20      10000.0 alok@gmail.com
Amod    30      20000.0 NULL
Anu     10      5000.0  anu@gmail.com
Om      40      50000.0 om@yahoo.com
Time taken: 13.464 seconds
```

Right Outer Join:

A right outer join (or right join) closely resembles a left outer join, except with the treatment of the tables reversed. Every row from the “right” table (B) will appear in the joined table at least once.

```
hive> select a.name,a.age,a.salary,b.email from employee1 a
    > right outer join mailid b on a.name = b.name;
```

```
Total MapReduce 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 mapred.reduce.tasks=<number>
Starting Job = job_201312102209_0019, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0019
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0019
2013-12-11 01:37:53,768 Stage-1 map = 0%,  reduce = 0%
2013-12-11 01:37:55,775 Stage-1 map = 67%,  reduce = 0%
2013-12-11 01:37:57,789 Stage-1 map = 100%,  reduce = 0%
2013-12-11 01:38:03,817 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0019
OK
Alok    20      10000.0 alok@gmail.com
Anu     10      5000.0  anu@gmail.com
Om      40      50000.0 om@yahoo.com
NULL    NULL    NULL    anu@gmail.com
NULL    NULL    NULL    om@yahoo.com
```

Full Join:

The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.

```
hive> select a.name,a.age,a.salary,b.email from employee1 a
    > full join mailid b on a.name = b.name;
```

```
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 mapred.reduce.tasks=<number>
Starting Job = job_201312102209_0020, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_201312102209_0020
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:80
21 -kill job_201312102209_0020
2013-12-11 01:40:18,206 Stage-1 map = 0%,  reduce = 0%
2013-12-11 01:40:20,213 Stage-1 map = 67%,  reduce = 0%
2013-12-11 01:40:22,222 Stage-1 map = 100%,  reduce = 0%
2013-12-11 01:40:28,251 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201312102209_0020
OK
Alok    20      10000.0 alok@gmail.com
Amod    30      20000.0 NULL
Anu     10      5000.0  anu@gmail.com
Om      40      50000.0 om@yahoo.com
NULL    NULL    NULL    anu@gmail.com
NULL    NULL    NULL    om@yahoo.com
```

Once done with hive we can use quit command to exit from the hive shell.

```
hive> quit;
```

HiveQL Manipulations

Hive defines a simple SQL-like query language to querying and managing large datasets called Hive-QL (HQL). It's easy to use if you're familiar with SQL Language. Hive allows programmers who are familiar with the language to write the custom MapReduce framework to perform more sophisticated analysis.