A1. Show all database in hive.

show databases;

show databases like ‘\*db\*’

describe database extended niit;

---------------------------------------------------------------------------------------------------------------------

B1. Create database

create database niit;

use niit;

---------------------------------------------------------------------------------------------------------------------

describe formatted txnrecords;

OK

# col\_name data\_type comment

txnno int

txndate string

custno int

amount double

category string

product string

city string

state string

spendby string

# Detailed Table Information

Database: niit

Owner: hduser

CreateTime: Sat Jun 10 22:23:58 IST 2017

LastAccessTime: UNKNOWN

Protect Mode: None

Retention: 0

Location: hdfs://localhost:54310/user/hive/warehouse/niit.db/txnrecords

Table Type: MANAGED\_TABLE

Table Parameters:

transient\_lastDdlTime 1497113638

# Storage Information

SerDe Library: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

InputFormat: org.apache.hadoop.mapred.TextInputFormat

OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat

Compressed: No

Num Buckets: -1

Bucket Columns: []

Sort Columns: []

Storage Desc Params:

field.delim ,

serialization.format ,

Time taken: 0.219 seconds, Fetched: 35 row(s)

---------------------------------------------------------------------------------------------------------------------

C1. Create transaction table

create table txnrecords(txnno INT, txndate STRING, custno Int, amount double, category string, product String, city string, state string, spendby string) row format delimited fields terminated by ',' stored as textfile;

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

C2. create Customer table.

create table customer (custno int, firstname string, lastname string, age int, profession string) row format delimited fields terminated by ',' stored as textfile;

---------------------------------------------------------------------------------------------------------------------

D1. Load the data into table from local file system

load data local inpath '/home/hduser/Downloads/txns1.txt' overwrite into table txnrecords;

will **delete** all content existing earlier and will place new file

load data local inpath '/home/hduser/Downloads/txns1.txt' into table txnrecords

will **not delete** old content

---------------------------------------------------------------------------------------------------------------------

E1. Describe metadata pr schema of table

hive> describe txnrecords;

OK

txnno int

txndate string

custno int

amount double

category string

product string

city string

state string

spendby string

E2. describe extended txnrecords;

OK

txnno int

txndate string

custno int

amount double

category string

product string

city string

state string

spendby string

Detailed Table Information Table(tableName:txnrecords, dbName:niit, owner:hduser, createTime:1497113638, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:txnno, type:int, comment:null), FieldSchema(name:txndate, type:string, comment:null), FieldSchema(name:custno, type:int, comment:null), FieldSchema(name:amount, type:double, comment:null), FieldSchema(name:category, type:string, comment:null), FieldSchema(name:product, type:string, comment:null), FieldSchema(name:city, type:string, comment:null), FieldSchema(name:state, type:string, comment:null), FieldSchema(name:spendby, type:string, comment:null)], location:hdfs://localhost:54310/user/hive/warehouse/niit.db/txnrecords, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, serdeInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{field.delim=,, serialization.format=,}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), storedAsSubDirectories:false), partitionKeys:[], parameters:{transient\_lastDdlTime=1497113638}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED\_TABLE)

**Time taken: 0.169 seconds, Fetched: 11 row(s)**

**ASCII( string str )**

The ASCII function converts the first character of the string into its numeric ascii value.

select ASCII('A');

---------------------------------------------------------------------------------------------------------------------

**CONCAT( string str1, string str2... )**

The CONCAT function concatenates all the stings.

select concat('Vish','\*','al');

**CONCAT\_WS( string delimiter, string str1, string str2... )**

The CONCAT\_WS function is similar to the CONCAT function. Here you can also provide the delimiter, which can be used in between the strings to concat.

select concat\_ws('\_','Vishal','Duratkar');

---------------------------------------------------------------------------------------------------------------------

**FIND\_IN\_SET( string search\_string, string source\_string\_list )**

The FIND\_IN\_SET function searches for the search string in the source\_string\_list and returns the position of the first occurrence in the source string list. Here the source string list should be comma delimited one. It returns 0 if the first argument contains comma.

select find\_in\_set('hc','hao,mn,hc,ha,hef');

OK

3

---------------------------------------------------------------------------------------------------------------------

**LENGTH( string str )**

The LENGTH function returns the number of characters in a string.

select length('Vishal');

OK

6

---------------------------------------------------------------------------------------------------------------------

**LOWER( string str ),  LCASE( string str )**

The LOWER or LCASE function converts the string into lower case letters.

select lower('VISHal');

OK

vishal

---------------------------------------------------------------------------------------------------------------------

**LPAD( string str, int len, string pad )**

The LPAD function returns the string with a length of len characters left-padded with pad.

select lpad('duratkar',8,'VS');

OK

duratkar // no append as length already crossed

hive> select lpad('duratkar',10,'VS');

OK

Vsduratkar //Single VS appended

hive> select lpad('duratkar',9,'VS');

OK

Vduratkar // only V is appended as length is crossed

---------------------------------------------------------------------------------------------------------------------

**LTRIM( string str )**

The LTRIM function removes all the trailing spaces from the string.

select ltrim(' vishal');

OK

vishal

---------------------------------------------------------------------------------------------------------------------

**REPEAT( string str, int n )**

The REPEAT function repeats the specified string n times.

select repeat('Vishal',2);

OK

VishalVishal

---------------------------------------------------------------------------------------------------------------------

**RPAD( string str, int len, string pad )**

The RPAD function returns the string with a length of len characters right-padded with pad.

select rpad('duratkar',10,'VS');

OK

duratkarVS

hive> select rpad('duratkar',9,'VS');

OK

duratkarV

hive> select rpad('duratkar',11,'VS');

OK

duratkarVSV

---------------------------------------------------------------------------------------------------------------------

**REVERSE( string str )**

The REVERSE function gives the reversed string

select reverse('Vishal');

OK

lahsiV

---------------------------------------------------------------------------------------------------------------------

**RTRIM( string str )**

The RTRIM function removes all the leading spaces from the string.

select rtrim('Vishal ');

OK

Vishal

---------------------------------------------------------------------------------------------------------------------

**SPACE( int number\_of\_spaces )**

The SPACE function returns the specified number of spaces.

select space(4);

OK

---------------------------------------------------------------------------------------------------------------------

**SPLIT( string str, string pat )**

The SPLIT function splits the string around the pattern pat and returns an array of strings. You can specify regular expressions as patterns.

select SPLIT('hive:hadoop:pig:mapreduce',':');

OK

["hive","hadoop","pig","mapreduce"]

---------------------------------------------------------------------------------------------------------------------

**SUBSTR( string source\_str, int start\_position [,int length]),SUBSTRING( string source\_str, int start\_position [,int length] )**

The SUBSTR or SUBSTRING function returns a part of the source string from the start position with the specified length of characters. If the length is not given, then it returns from the start position to the end of the string.

select substr('hadoop',2);

OK

adoop

hive> select substr('hadoop',2,2);

OK

ad

---------------------------------------------------------------------------------------------------------------------

**TRIM( string str )**  
  
The TRIM function removes both the trailing and leading spaces from the string.

select trim(' vishal ');

OK

vishal

---------------------------------------------------------------------------------------------------------------------

**UPPER( string str ), UCASE( string str )**  
  
The UPPER or UCASE function converts the string into upper case letters.

select upper('abcd');

OK

ABCD

**Numeric and Mathematical Functions in Hive**

The Numerical functions are listed below in alphabetical order. Use these functions in SQL queries.   
**ABS**( double n )  
  
The ABS function returns the absolute value of a number.

select abs(-100);

OK

100

**ACOS**( double n )  
  
The ACOS function returns the arc cosine of value n. This function returns Null if the value n is not in the range of -1<=n<=1.

select acos(2);

OK

NaN

hive> select acos(0.2);

OK

1.369438406004566

**ASIN**( double n )  
  
The ASIN function returns the arc sin of value n. This function returns Null if the value n is not in the range of -1<=n<=1.

select asin(-0.2);

OK

-0.2013579207903308

**BIN**( bigint n )  
  
The BIN function returns the number n in the binary format.

select bin(10);

OK

1010

txnrecords

**Array :-**

$ cat >arrayfile

1,abc,40000,a$b$c,hyd

2,def,3000,d$f,bang

create table tab7(id int, anme string, sal bigint, sub array<string>,city string)

>row format delimited

>fields terminated by ','

>collection items terminated by '$';

load data local inpath 'arrayfile' overwrite into table tab7;

select \* from tab7;

OK

1 abc 40000 ["a","b","c"] hyd

2 def 3000 ["d","f"] bang

select \* from tab7 where id=2;

OK

2 def 3000 ["d","f"] bang

select id, anme, sal, sub[2] from tab7;

OK

1 abc 40000 c

2 def 3000 NULL

**Map :-**

$ cat >mapfile

1,abc,40000,a$b$c,pf#500$epf#200,hyd

2,def,3000,d$f,pf#500,bang

create table tab10(id int, name string, sal bigint,sub array<string>, dud map<string,int>, city string)

> row format delimited

> fields terminated by ','

> collection items terminated by '$'

> map keys terminated by '#';

load data local inpath 'mapfile' overwrite into table tab10;

select \* from tab10;

OK

1 abc 40000 ["a","b","c"] {"pf":500,"epf":200} hyd

2 def 3000 ["d","f"] {"pf":500} bang

select dud["pf"] from tab10;

OK

500

500

select dud["pf"], dud["epf"] from tab10;

OK

500 200

500 NULL

**Struct:-**

create table tab11(id int, name string, sal bigint,sub array<string>, dud map<string,int>, addr struct<city:string, state:string, pin:bigint>)

> row format delimited

> fields terminated by ','

> collection items terminated by '$'

> map keys terminated by '#';

load data local inpath '/home/hduser/Downloads/Fw\_\_data\_files (2)/structfile.txt' overwrite into table tab11;

select \* from tab11;

OK

1 abc 40000 ["a","b","c"] {"bonus":500,"insurance":200} {"city":"city1","state":"state1","pin":111}

2 def 3000 ["d","f"] {"bonus":500} {"city":"city2","state":"state2","pin":222}

select addr.city from tab11;

OK

city1

city2

Q: Word count using hive.

create table wordCount(line string);

load data local inpath '/home/hduser/Downloads/wordCount.txt' into table wordCount;

select split(line,' ') as word from wordcount;

select explode(split(line,' ')) as word from wordcount;

select word, count(\*) from (select explode(split(line,' ')) as word from wordcount) a group by word;

F. Counting no of records.

select count(\*) from txnrecords;

G. Top 10 customer List.

select a.custno, b.firstname, b.lastname, b.age, b.profession, sum(a.amount) as amt

from txnrecords a, customer b

where a.custno = b.custno

group by a.custno, b.firstname, b.lastname, b.age, b.profession

order by amt desc limit 10;

H1. Create partitioned table (single bucket)

create table txnrecsByCat(txnno INT, txndate STRING, custno INT, amount DOUBLE,  
product STRING, city STRING, state STRING, spendby STRING)  
**partitioned by (category STRING)**  
row format delimited  
fields terminated by ','  
stored as textfile;

H2 . Load data into partition table (single bucket)

from txnrecords txn insert overwrite table txnrecsByCat

**partition(category)** select txn.txnno, txn.txndate,txn.custno,

txn.amount,txn.product,txn.city,txn.state, txn.spendby, **txn.category**

**distribute by category;**

I1. Create partitioned table (with multiple buckets)

create table txnrecsByCat2(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;

I2. Load data into partition table (with multiple buckets)

from txnrecords txn INSERT OVERWRITE TABLE txnrecsByCat2  
PARTITION(category) select txn.txnno, txn.txndate,txn.custno,  
txn.amount,txn.product,txn.city,txn.state, txn.spendby, **txn.category**  
**DISTRIBUTE By category**;