

## ASSIGNMENT -4

### Part A : Assignments based on the Hadoop

**4. Write an application using HBase and HiveQL for flight information system which will include**

- 1) Creating, Dropping, and altering Database tables**
- 2) Creating an external Hive table to connect to the HBase for Customer Information Table**
- 3) Load table with data, insert new values and field in the table, Join tables with Hive**
- 4) Create index on Flight information Table**
- 5) Find the average departure delay per day in 2008.**

-----  
HBase via Hive,

### **DOWNLOAD & COPY:**

- Download hive on below path (nearly 93 MB): <http://www.apache.org/dyn/closer.cgi/hive/>
- Extract the .tar.gz file in Downloads/ and rename it to hive/ and move the folder to **/usr/lib/** path:

```
sudo mv Downloads/hive /usr/lib
```

### **CHANGE THE OWNER:**

- Provide access to hive path by changing the owners and groups to hduser and hadoop respectively.

```
sudo chown -R hduser:hadoop /usr/lib/hive
```

### **CONFIGURE ENVIRONMENT VARIABLES:**

- Configure environment variables in .bashrc file.

```
su - hduser
```

```
vim ~/.bashrc
```

- Add following lines at the end of file

```
export HIVE_HOME=/usr/lib/hive/  
export PATH=$PATH:$HIVE_HOME/bin  
export HADOOP_USER_CLASSPATH_FIRST=true
```

- Apply the changes:

```
source ~/.bashrc
```

## MAKE DIRECTORIES:

- Create temporary and folder for data warehouse of hive in HDFS as well as change the permissions.

```
hadoop fs -mkdir /tmp
hadoop fs -mkdir -p /user/hive/warehouse
hadoop fs -chmod g+w /tmp
hadoop fs -chmod -R g+w /user/hive/warehouse
```

## CONFIGURE HIVE:

- To configure Hive with Hadoop, you need to edit the hive-env. sh file, which is placed in the \$HIVE\_HOME/conf directory. The following commands redirect to Hive config folder and copy the template file:

```
cd $HIVE_HOME/conf
cp hive-env.sh.template hive-env.sh
```

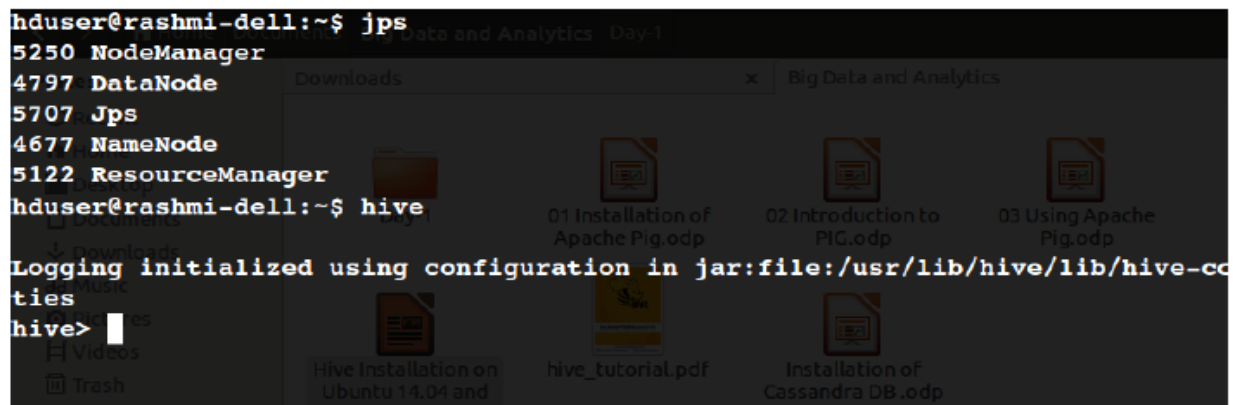
- Edit the hive-env.sh file by appending the following line:

```
export HADOOP_HOME=/usr/local/hadoop
```

## RUN HIVE:

Make sure that Hadoop services are running. Then type  
hive

hive



The screenshot shows a terminal window with the following output:

```
hduser@rashmi-dell:~$ jps
5250 NodeManager
4797 DataNode
5707 Jps
4677 NameNode
5122 ResourceManager
hduser@rashmi-dell:~$ hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-conf-ties
hive>
```

In the background, a file explorer window titled "Big Data and Analytics" is open, showing a "Downloads" folder. It contains several files, including "01 Installation of Apache Pig.odp", "02 Introduction to PIG.odp", "03 Using Apache Pig.odp", "Hive Installation on Ubuntu 14.04 and", "hive\_tutorial.pdf", and "Installation of Cassandra DB.odp".

## HADOOP ECOSYSTEM:

- The Hadoop ecosystem contains different subprojects (tools) such as Sqoop, Pig, and Hive that are used to help Hadoop modules.

- **Sqoop:** It is used to import and export data to and fro between HDFS and RDBMS.
- **Pig:** It is a procedural language platform used to develop a script for MapReduce operations.
- **Hive:** It is a platform used to develop SQL type scripts to do MapReduce operations.

## **DATABASE OPERATIONS:**

Hive is a database technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it. This chapter explains how to create Hive database. Hive contains a default database named **default**.

## **CREATE DATABASE:**

- Create Database is a statement used to create a database in Hive.
- A database in Hive is a namespace or a collection of tables.

The syntax for this statement is as follows:

**CREATE DATABASE|SCHEMA [IF NOT EXISTS]  
<database name>;**

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command.

- The following query is executed to create a database named mydb:

**hive> CREATE DATABASE [IF NOT EXISTS] mydb;**

or

**hive> CREATE SCHEMA mydb;**

- The following query is used to verify a databases list:

**hive> SHOW DATABASES;**

**default**

**mydb**

## **DROP DATABASE:**

- Drop Database is a statement that drops all the tables and deletes the database.
- Its syntax is as follows:

**DROP DATABASE Statement DROP  
(DATABASE|SCHEMA) [IF EXISTS]  
database\_name [RESTRICT|CASCADE];**

- The following queries are used to drop a database. Let us assume that the database name is mydb.

**hive> DROP DATABASE IF EXISTS mydb;**

## **CREATE TABLE:**

- Create Table is a statement used to create a table in Hive. The syntax and example are as follows:
- Syntax:

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF  
NOT EXISTS] [db_name.] table_name  
[(col_name data_type [COMMENT  
col_comment], ...)]  
[COMMENT table_comment]  
[ROW FORMAT row_format]  
[STORED AS file_format]
```

Example:::

Sr. No.	Field Name	Data type
1	Eid	Int
2	Name	String
3	Salary	Float
4	Designation	String

- The following query creates a table named employee using the above data.

```
hive> CREATE TABLE IF NOT EXISTS  
employee ( eid int, name String,  
> salary String, destination String)  
> COMMENT 'Employee details'  
> ROW FORMAT DELIMITED  
> FIELDS TERMINATED BY '\t'  
> LINES TERMINATED BY '\n'  
> STORED AS TEXTFILE;
```

## **ALTER TABLE:**

```
ALTER TABLE name RENAME TO new_name  
ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])  
ALTER TABLE name DROP [COLUMN] column_name  
ALTER TABLE name CHANGE column_name new_name  
new_type  
ALTER TABLE name REPLACE COLUMNS (col_spec[,  
col_spec ...])
```

## ALTER TABLE – RENAME TO:

ALTER TABLE employee RENAME TO emp;

## CHANGE STATEMENT:

The following table contains the fields of **employee** table and it shows the fields to be changed (in bold).

Field Name	Convert from Data Type	Change Field Name	Convert to Data Type
eid	int	eid	int
<b>name</b>	String	<b>ename</b>	String
salary	<b>Float</b>	salary	<b>Double</b>
designation	String	designation	String

## CHANGE STATEMENT EXAMPLE:

- **hive> ALTER TABLE employee CHANGE name ename String;**
- **hive> ALTER TABLE employee CHANGE salary salary Double;**

## ADD COLUMN STATEMENT:

- `hive> ALTER TABLE employee ADD COLUMNS (  
> dept STRING COMMENT 'Department name');`
- 

### **REPLACE STATEMENT:**

```
hive> ALTER TABLE employee REPLACE COLUMNS  
(  
> eid INT empid Int,  
> ename STRING name String);
```

---

### **DROP TABLE STATEMENT:**

- The syntax is as follows:
    - `DROP TABLE [IF EXISTS] table_name;`
  - The following query drops a table named employee:
    - `hive> DROP TABLE IF EXISTS employee;`
- 

### **INDEX:**

- An Index is nothing but a pointer on a particular column of a table.
- Creating an index means creating a pointer on a particular column of a table.
- `hive> CREATE INDEX index_yoj ON TABLE file(yoj)  
> AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'  
WITH DEFERRED REBUILD;`

### **DROP INDEX:**

- The following syntax is used to drop an index:

```
DROP INDEX <index_name> ON <table_name>
```

- The following query drops an index named index\_salary:

```
hive> DROP INDEX index_salary ON employee;
```

## Select...Order By:

- The ORDER BY clause is used to retrieve the details based on one column and sort the result set by ascending or descending order.

- Syntax:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
```

```
FROM table_reference
```

```
[WHERE where_condition]
```

```
[GROUP BY col_list]
```

```
[HAVING having_condition]
```

```
[ORDER BY col_list]]
```

```
[LIMIT number];
```

## Example:

```
hive> select * from file order by yoj;
Query ID = hduser_20160703164810_7d84d930-f1dd-4ed3-9410-1f09af20a74d
Total 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 mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2016-07-03 16:48:13,401 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1590275424_0005
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 6000 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
104      Parmeet  CS      2010
102      Rajesh   IT      2010
103      Awez     CS      2012
103      Suresh   CS      2012
Time taken: 2.462 seconds, Fetched: 4 row(s)
```

## Select...Group By:

- The GROUP BY clause is used to group all the records in a result set using a particular collection column. It is used to query a group of records.

- Syntax:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[ORDER BY col_list]
[LIMIT number];
```

Example:

```
hive> select dept, count(*) from file group by dept;
Query ID = hduser_20160703165351_da8962c1-3407-49bd-bd57-c463d2aab7ff
Total 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 mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2016-07-03 16:53:53,780 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1959421652_0007
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 6300 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
CS      3
IT      1
Time taken: 1.86 seconds, Fetched: 2 row(s)
```

## JOINS:

- JOINS is a clause that is used for combining specific fields from two tables by using values common to each one.
- It is used to combine records from two or more tables in the database.
- It is more or less similar to SQL JOINS.


Example::



```
hive> select * from customer;
OK
1      Kavita  24      Sangvi  34000
2      Chatur  23      Kothrud 35000
3      Fatema  31      Lohgad  20000
4      Rohan   27      Pune Station 22000
Time taken: 0.061 seconds, Fetched: 4 row(s)
```

```
hive> select * from orders;
OK
102     NULL    3      1200
104     NULL    3      3400
105     NULL    4      2150
106     NULL    2      3420
Time taken: 0.057 seconds, Fetched: 4 row(s)
```

```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT
> FROM CUSTOMER c JOIN ORDERS o
> ON (c.ID = o.c_id);
Query ID = hduser_20160703175303_ac7c2fcc-c9f2-
Total jobs = 1
```




```
Total MapReduce CPU Time Spent: 0 msec
OK
2      Chatur  23      3420
3      Fatema  31      1200
3      Fatema  31      3400
4      Rohan   27      2150
Time taken: 9.21 seconds, Fetched: 4 row(s)
```

## Left Outer Join:

- The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table.
- This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.
- A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

```
hive> select c.ID, c.NAME, o.AMOUNT  
> FROM CUSTOMER c  
> LEFT OUTER JOIN ORDERS o  
> ON (c.ID = o.C_ID);
```




```
MapReduce Jobs Launched:  
Stage-Stage-3: HDFS Read: 106 HDFS Write: 0 SUCCESS  
Total MapReduce CPU Time Spent: 0 msec  
OK  
1      Kavita  NULL  
2      Chatur  3420  
3      Fatema  1200  
3      Fatema  3400  
4      Rohan   2150  
Time taken: 11.194 seconds, Fetched: 5 row(s)
```

### **Right Outer Join:**

- The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table.
- If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.
- A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.

```
hive> select c.ID, c.NAME, o.AMOUNT  
> FROM CUSTOMER c  
> RIGHT OUTER JOIN ORDERS o  
> ON (c.ID = o.C_ID);
```



```
Stage-Stage-3: HDFS Read: 162 HDFS Write: 0 SUCCESS  
Total MapReduce CPU Time Spent: 0 msec  
OK  
3      Fatema    1200  
3      Fatema    3400  
4      Rohan     2150  
2      Chatur    3420  
Time taken: 18.488 seconds, Fetched: 4 row(s)
```

**Conclusion:** Thus we have learnt how to execute queries with hive and various operations related to database (like Creating Table/Database, Dropping Table ,Altering Table, Joins in Table, Index in Table)with HiveQL & HBase.