**EXPERIMENT 4**

| **Objective :**  **Theory:**  **Performance:**  **Deliverables:**  **Summary :** | Create HIVE Database and Descriptive analytics-basic statistics.  Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query and analysis. Using Hadoop was not easy for end users, especially for the ones who were not familiar with MapReduce framework. End users had to write map/reduce programs for simple tasks like getting raw counts or averages. Hive was created to make it possible for analysts with strong SQL skills (but meager Java programming skills) to run queries on the huge volumes of data to extract patterns and meaningful information. It provides an SQL-like language called HiveQL while maintaining full support for map/reduce. In short, a Hive query is converted to MapReduce tasks.    [Download](https://drive.google.com/file/d/1JfiudjZwsWzn6dXYf4RRKkcvdNsenYWh/view?usp=sharing) HDP Hortonworks sandbox from given link and import into Virtualbox.    Type in browser 127.0.0.1:8888 for accessing HDP  To explore HDP Sandbox navigate to 127.0.0.1:8000 by entering username as hue and password 1111    After Successful login you will get below screen    you can also login to your HDP sandbox by CLI, type  **user: sudo ssh root@127.0.0.1 -p 2222**  **password : hadoop**    To start service of Ambari run script as  **[root@sandbox ~]# ./start\_ambari.sh**    After Starting service of Ambari open UI of Ambari from browser 127.0.0.1:8080 and login with username and password as **admin**    After successful login you able to see Dashboard of Ambari which having all Hadoop ecosystem components monitoring as shown below:    You can start all Hadoop ecosystem components by selecting start all services from Action menu as shown:    If you need to start only selected services then you have to select that particular service and then from service action menu you need to start.  After successfully login we can use various Hadoop components such as hive, pig, Hcatalog etc components.  Hive allows us to run queries similar to SQL queries such as Create databases, Create tables etc.  Let’s demonstrate Hive using data set provided on Moodle  Upload the [data](https://drive.google.com/drive/folders/1reAkGV_qfGDshgy5A66mfICj0p6LwZEc?usp=sharing) to the file system using Ambari interface.  Open Query Editor in Hive.  Create a table in Hive to hold the data. Type the query into the Query Editor. Once the query is ready hit the **Execute** button at the bottom.  **CREATE TABLE temp\_drivers (col\_value STRING) STORED AS TEXTFILE**  ++++++++++++++    **Load the data file drivers.csv into the table temp\_drivers**  Load Data Inpath ‘/user/hue/HiveT/drivers.csv’ Overwrite Into Table temp\_drivers    **SELECT \* FROM temp\_drivers**    **CREATE TABLE drivers (driverId INT, name STRING, ssn BIGINT, location STRING, certified STRING, wageplan STRING)**  **Insert data into drivers table**  **insert overwrite table drivers**  **SELECT**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){1}', 1) driverId,**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){2}', 1) name,**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){3}', 1) ssn,**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){4}', 1) location,**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){5}', 1) certified,**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){6}', 1) wageplan**  **from temp\_drivers**  SELECT \* FROM drivers    Create temp\_timesheet and timesheet tables similarly  Create table in Hive to which contents of timesheet table will be transferred.  **CREATE TABLE temp\_timesheet (col\_value string) STORED AS TEXTFILE**  Load into this table the contents from hdfs file.  **LOAD DATA INPATH '/user/maria\_dev/timesheet.csv' OVERWRITE INTO TABLE temp\_timesheet**  **SELECT \* FROM temp\_timesheet LIMIT 10**  **CREATE TABLE timesheet (driverId INT, week INT, hours\_logged INT, miles\_logged INT)**  Copying data from temp\_timesheet table to timesheet table using following command.  **insert overwrite table timesheet**  **SELECT**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){1}', 1) driverId,**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){2}', 1) week,**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){3}', 1) hours\_logged,**  **regexp\_extract(col\_value, '^(?:([^,]\*),?){4}', 1) miles\_logged**  **from temp\_timesheet.**  **Now let’s process this data to find some meaningful insights.**  ***Find the total number of hours and miles logged in by each driver for a year.***  **SELECT driverId, sum(hours\_logged), sum(miles\_logged) FROM timesheet GROUP BY driverId**  ***Rewrite the query to identify the names of the drivers along with the total hours and miles logged for an year by the drivers.***  **SELECT d.driverId, d.name, t.total\_hours, t.total\_miles from drivers d**  **JOIN ( SELECT driverId, sum(hours\_logged)total\_hours, sum(miles\_logged)total\_miles FROM timesheet GROUP BY driverId ) t**  **ON (d.driverId = t.driverId)**  A well documented PDF enlisting all the steps along with the screen shots of the successful execution of each step.  In this experiment we uploaded data into HDFS Files View and create hive queries to manipulate data. We also demonstrated use of SQL queries in HIVE |
| --- | --- |