Program – 8

AIM: 1) To install and run Hive. 2) Use Hive to CREATE, ALTER, DROP databases, tables, views, functions, indexes.

Introduction & Theory

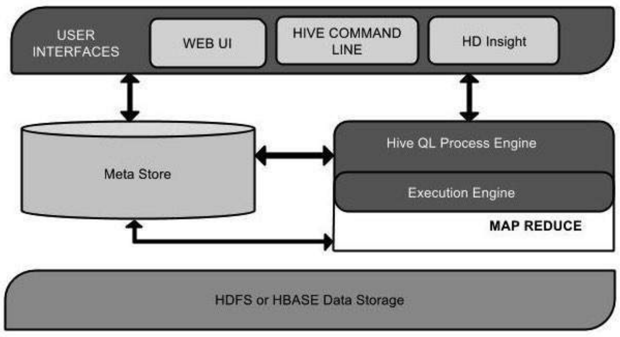
About Hive

Apache Hive is a data warehouse infrastructure that facilitates querying and managing large data sets which resides in distributed storage system. It is built on top of Hadoop and developed by Facebook. Hive provides a way to query the data using a SQL-like query language called HiveQL(Hive query Language).

Internally, a compiler translates HiveQL statements into MapReduce jobs, which are then submitted to Hadoop framework for execution.
Hive looks very much similar like traditional database with SQL access. However, because Hive is based on Hadoop and MapReduce operations, there are several key differences:

As Hadoop is intended for long sequential scans and Hive is based on Hadoop, you would expect queries to have a very high latency. It means that Hive would not be appropriate for those applications that need very fast response times, as you can expect with a traditional RDBMS database.

Finally, Hive is read-based and therefore not appropriate for transaction processing that typically involves a high percentage of write operations.



User Interface:

Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).

Meta Store:

Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.

HiveQL Process Engine:

HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.

Program – 8

Execution Engine:

The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavour of MapReduce.

HDFS/HBASE:

Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Hive Features:

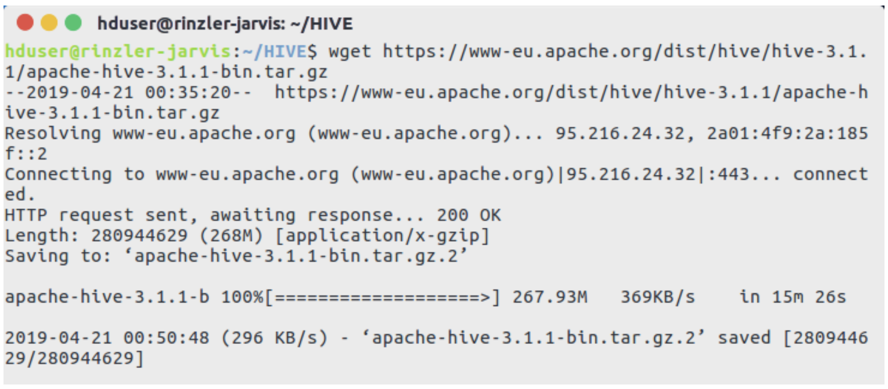
- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Installing Pig

Prerequisites:

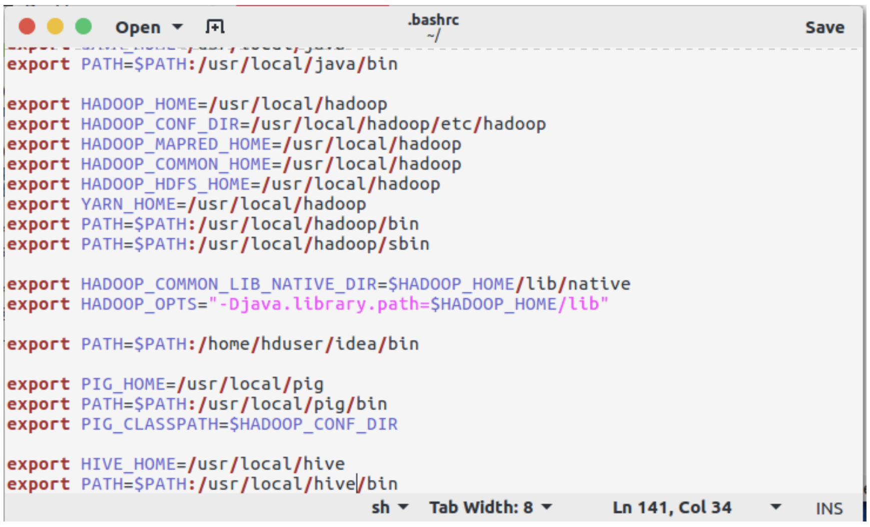
1. Java
2. Hadoop

1. Download the Hive Files from Apache.



```
hduser@rinzler-jarvis: ~/HIVE
hduser@rinzler-jarvis:~/HIVE$ wget https://www-eu.apache.org/dist/hive/hive-3.1.
1/apache-hive-3.1.1-bin.tar.gz
--2019-04-21 00:35:20--  https://www-eu.apache.org/dist/hive/hive-3.1.1/apache-h
ive-3.1.1-bin.tar.gz
Resolving www-eu.apache.org (www-eu.apache.org)... 95.216.24.32, 2a01:4f9:2a:185
f::2
Connecting to www-eu.apache.org (www-eu.apache.org)|95.216.24.32|:443... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 280944629 (268M) [application/x-gzip]
Saving to: ‘apache-hive-3.1.1-bin.tar.gz.2’

apache-hive-3.1.1-b 100%[===================>] 267.93M   369KB/s    in 15m 26s

2019-04-21 00:50:48 (296 KB/s) - ‘apache-hive-3.1.1-bin.tar.gz.2’ saved [2809446
29/280944629]
```

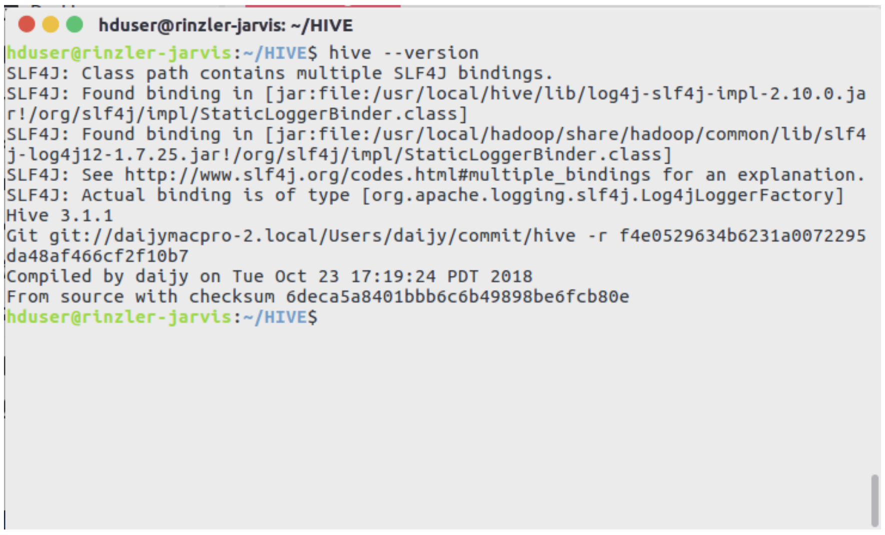
2. Extract the files to a convenient location. (/usr/local).
3. Edit the system variable to include the Pig files.



```
export PATH=$PATH:/usr/local/java/bin

export HADOOP_HOME=/usr/local/hadoop
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export HADOOP_MAPRED_HOME=/usr/local/hadoop
export HADOOP_COMMON_HOME=/usr/local/hadoop
export HADOOP_HDFS_HOME=/usr/local/hadoop
export YARN_HOME=/usr/local/hadoop
export PATH=$PATH:/usr/local/hadoop/bin
export PATH=$PATH:/usr/local/hadoop/sbin

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

export PATH=$PATH:/home/hduser/idea/bin

export PIG_HOME=/usr/local/pig
export PATH=$PATH:/usr/local/pig/bin
export PIG_CLASSPATH=$HADOOP_CONF_DIR

export HIVE_HOME=/usr/local/hive
export PATH=$PATH:/usr/local/hive/bin
```

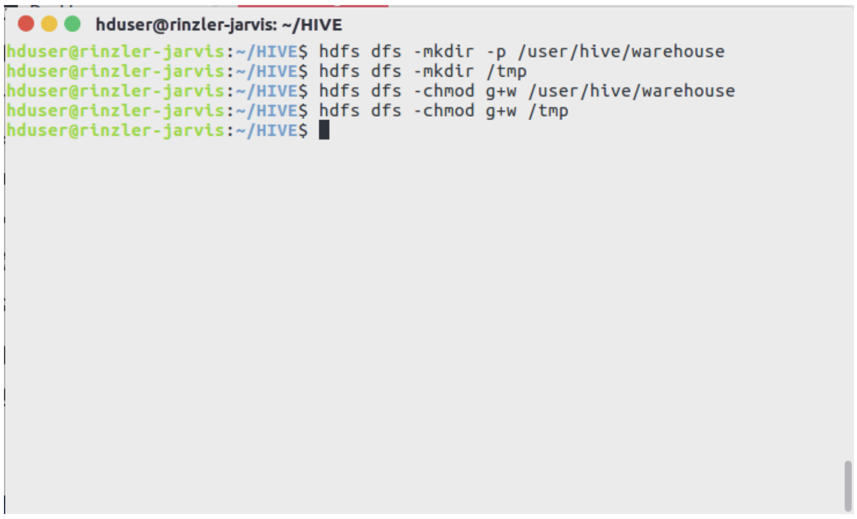
Program – 8

4. Check Pig version to check if its working properly.



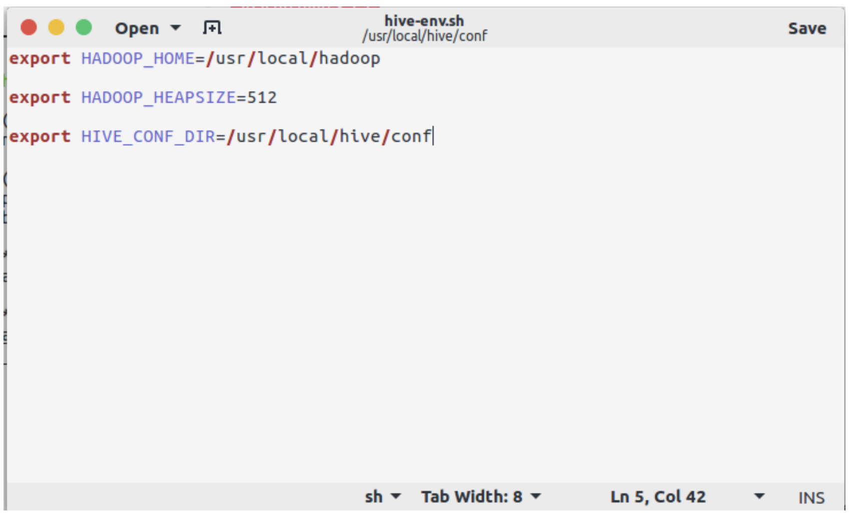
```
hduser@rinzler-jarvis: ~/HIVE
hduser@rinzler-jarvis:~/HIVE$ hive --version
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive 3.1.1
Git git://daijymacpro-2.local/Users/daijy/commit/hive -r f4e0529634b6231a0072295
da48af466cf2f10b7
Compiled by daijy on Tue Oct 23 17:19:24 PDT 2018
From source with checksum 6deca5a8401bbb6c6b49898be6fcb80e
hduser@rinzler-jarvis:~/HIVE$
```

5. Create Hive directories within HDFS and give them read/write permissions. The directory 'warehouse' is the location to store the table or data related to hive.



```
hduser@rinzler-jarvis: ~/HIVE
hduser@rinzler-jarvis:~/HIVE$ hdfs dfs -mkdir -p /user/hive/warehouse
hduser@rinzler-jarvis:~/HIVE$ hdfs dfs -mkdir /tmp
hduser@rinzler-jarvis:~/HIVE$ hdfs dfs -chmod g+w /user/hive/warehouse
hduser@rinzler-jarvis:~/HIVE$ hdfs dfs -chmod g+w /tmp
hduser@rinzler-jarvis:~/HIVE$
```

6. Set Hadoop path in hive-env.sh



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

export HADOOP_HEAPSIZE=512

export HIVE_CONF_DIR=/usr/local/hive/conf
```

Program – 8

7. Edit the hive-site.xml file.

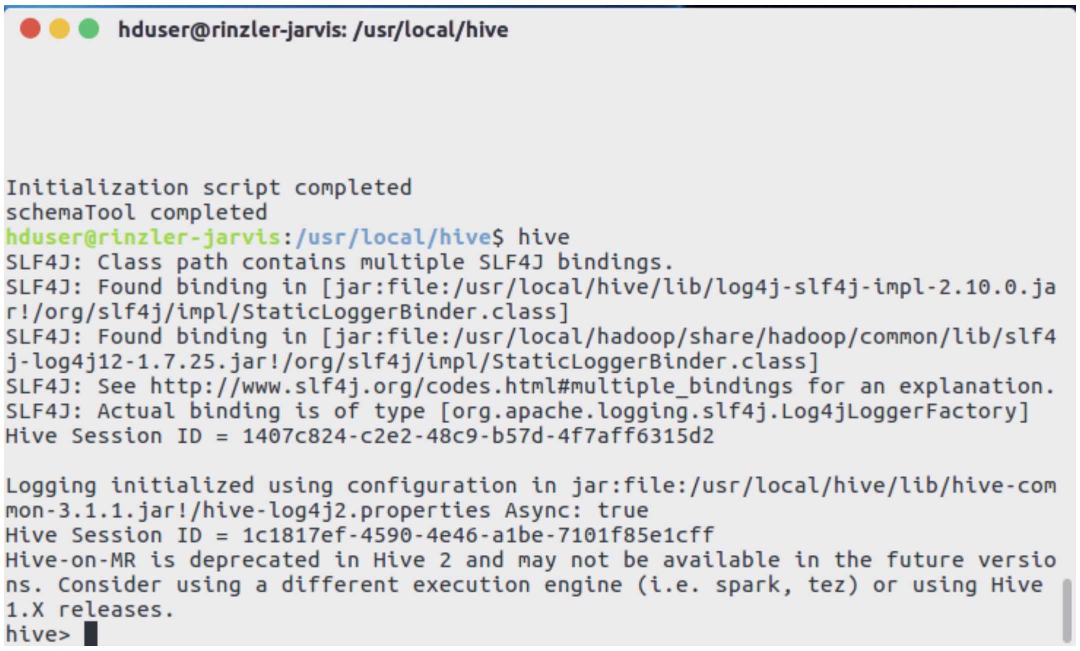
```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?><!—
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<configuration>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby:;databaseName=/usr/local/hive/metastore_db;create=true
</value>
        <description>
JDBC connect string for a JDBC metastore.
To use SSL to encrypt/authenticate the connection, provide database-
specific SSL flag in the connection URL.
For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
        </description>
    </property>
    <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/user/hive/warehouse</value>
        <description>location of default database for the
warehouse</description>
    </property>
    <property>
        <name>hive.metastore.uris</name>
        <value/>
        <description>Thrift URI for the remote metastore. Used by
metastore client to connect to remote metastore.</description>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>org.apache.derby.jdbc.EmbeddedDriver</value>
        <description>Driver class name for a JDBC
metastore</description>
    </property>
    <property>
        <name>javax.jdo.PersistenceManagerFactoryClass</name>
     <value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>
        <description>class implementing the jdo
persistence</description>
    </property>
</configuration>
```

Program – 8

8. By default, Hive uses Derby database. Initialize Derby database using:

```
bin/schematool -initSchema -dbType derby
```

9. Launch Hive



hduser@rinzler-jarvis: /usr/local/hive

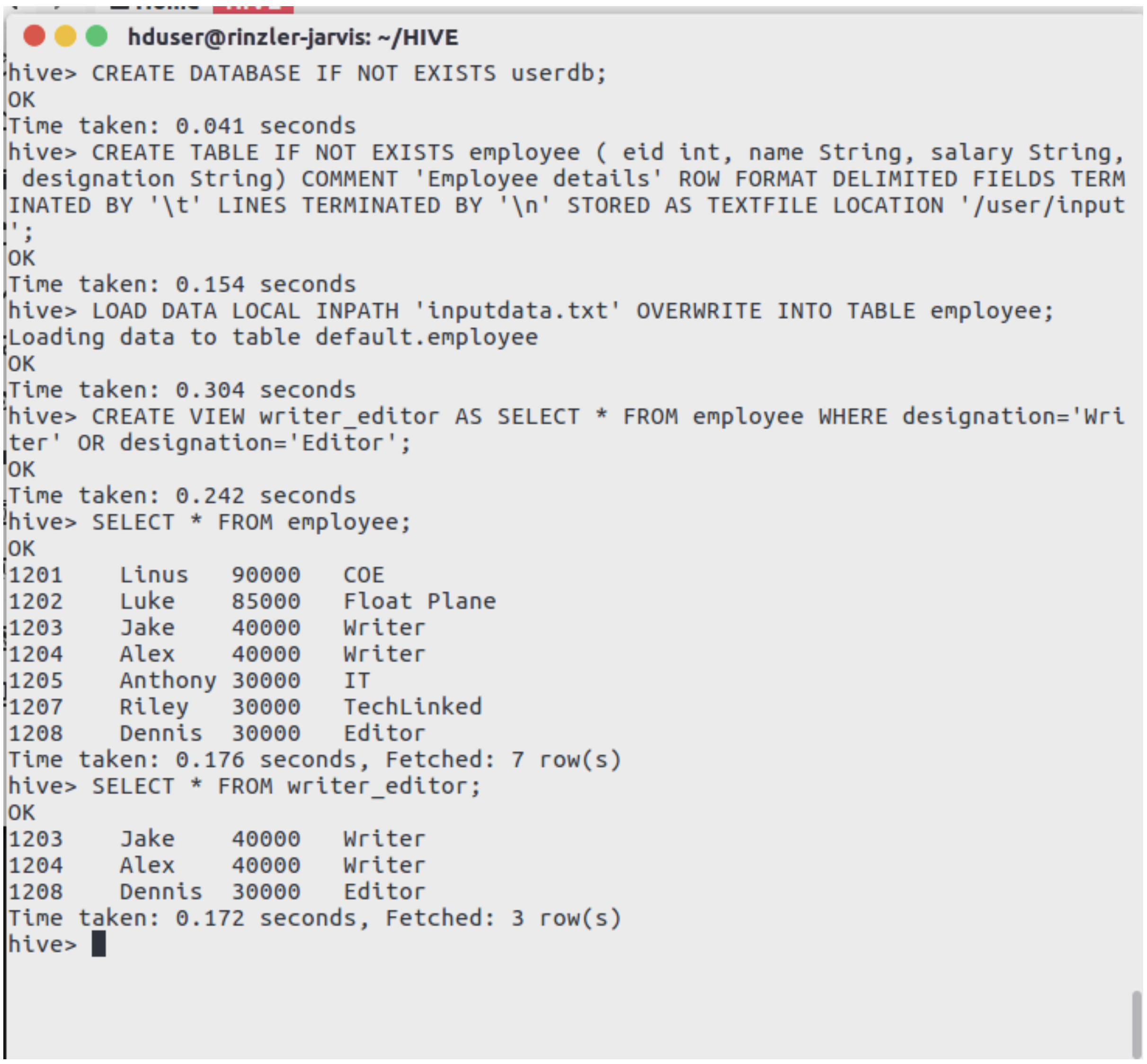
```
Initialization script completed
schemaTool completed
hduser@rinzler-jarvis:/usr/local/hive$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 1407c824-c2e2-48c9-b57d-4f7aff6315d2

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-com
mon-3.1.1.jar!/hive-log4j2.properties Async: true
Hive Session ID = 1c1817ef-4590-4e46-a1be-7101f85e1cff
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive>
```

Hive Operations

```
CREATE DATABASE IF NOT EXISTS userdb;

CREATE TABLE IF NOT EXISTS employee ( eid int, name String, salary
String, designation String) COMMENT 'Employee details' ROW FORMAT
DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' STORED
AS TEXTFILE LOCATION '/user/input';

LOAD DATA LOCAL INPATH 'inputdata.txt' OVERWRITE INTO TABLE employee;


CREATE VIEW writer_editor AS SELECT * FROM employee WHERE
designation='Writer' or designation='Editor';


CREATE INDEX index_salary ON TABLE employee(salary) AS
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH
DEFERRED REBUILD;

SELECT * from employee;

SELECT * from writer_editor;
```

Program – 8

Output



```
hive> CREATE DATABASE IF NOT EXISTS userdb;
OK
Time taken: 0.041 seconds
hive> CREATE TABLE IF NOT EXISTS employee ( eid int, name String, salary String,
 designation String) COMMENT 'Employee details' ROW FORMAT DELIMITED FIELDS TERM
INATED BY '\t' LINES TERMINATED BY '\n' STORED AS TEXTFILE LOCATION '/user/input
';
OK
Time taken: 0.154 seconds
hive> LOAD DATA LOCAL INPATH 'inputdata.txt' OVERWRITE INTO TABLE employee;
Loading data to table default.employee
OK
Time taken: 0.304 seconds
hive> CREATE VIEW writer_editor AS SELECT * FROM employee WHERE designation='Wri
ter' OR designation='Editor';
OK
Time taken: 0.242 seconds
hive> SELECT * FROM employee;
OK
1201    Linus   90000   COE
1202    Luke    85000   Float Plane
1203    Jake    40000   Writer
1204    Alex    40000   Writer
1205    Anthony 30000   IT
1207    Riley   30000   TechLinked
1208    Dennis  30000   Editor
Time taken: 0.176 seconds, Fetched: 7 row(s)
hive> SELECT * FROM writer_editor;
OK
1203    Jake    40000   Writer
1204    Alex    40000   Writer
1208    Dennis  30000   Editor
Time taken: 0.172 seconds, Fetched: 3 row(s)
hive> 
```

Findings and Learnings:

1. We learned about Apache Hive.
2. We learned about the advantages of Apache Hive.
3. We compared SQL and Hive.
4. We learnt how to perform operations in Hive.