

EXP NO : 5

## **Create tables in Hive and write queries to access the data in the table**

### **AIM:**

To create tables in Hive and write queries to access the data in the table.

### **PROCEDURE:**

#### **Step 1: Download and Install Hive**

##### **1. Download Hive:**

**Download Hive from the official website:**

```
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

##### **2. Extract Hive:**

```
tar -xvf apache-hive-3.1.2-bin.tar.gz
```

##### **3. Move Hive Directory:**

```
sudo mv apache-hive-3.1.2-bin /usr/local/hive
```

##### **4. Set Hive Environment Variables:**

**Edit .bashrc to configure Hive:**

```
nano ~/.bashrc
```

**Add the following lines:**

```
export HIVE_HOME=/usr/local/hive
```

```
export PATH=$PATH:$HIVE_HOME/bin
```

**Apply the changes:**

```
source ~/.bashrc
```

##### **5. Configure Hive:**

**Configure Hive to use MySQL as its metastore by editing the Hive configuration file (hive-site.xml):**

```
nano $HIVE_HOME/conf/hive-site.xml
```

**Add the following configuration for MySQL connection:**

```
<property>

  <name>javax.jdo.option.ConnectionURL</name>

  <value>jdbc:mysql://localhost/metastore</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionDriverName</name>

  <value>com.mysql.jdbc.Driver</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionUserName</name>

  <value>root</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionPassword</name>

  <value>password</value>

</property>
```

**6. Start Hive:**

**Once everything is configured, start Hive by simply typing:**

```
hive
```

**Step 2: Create a Database and Table in Hive****1. Create a Database:**

**In the Hive terminal, create a new database:**

```
CREATE DATABASE financials;
```

**2. Use the Database:**

```
USE financials;
```

**3. Create a Table:**

**Create a table to store financial data:**

```
CREATE TABLE finance_table (  
    id INT,  
    name STRING  
)
```

**4. Insert Data into the Table:**

**Insert sample data into the finance\_table:**

```
INSERT INTO TABLE finance_table VALUES (1, 'Alice'), (2, 'Bob'), (3, 'Charlie');
```

**Step 3: Store the Output in HDFS****1. Create a Partitioned Table:**

**For optimized storage, create a partitioned table by year:**

```
CREATE TABLE partitioned_finance_table (  
    id INT,  
    name STRING  
)  
  
PARTITIONED BY (year INT)
```

**2. Insert Data into the Partitioned Table:**

```
INSERT INTO partitioned_finance_table PARTITION (year=2023) VALUES (1, 'Alice'), (2,  
'Bob');
```

```
INSERT INTO partitioned_finance_table PARTITION (year=2024) VALUES (3, 'Charlie');
```

### 3. Create a Bucketed Table:

**Create a bucketed table to improve query performance:**

```
CREATE TABLE bucketed_finance_table (  
    id INT,  
    name STRING  
)  
  
CLUSTERED BY (id) INTO 4 BUCKETS
```

### 4. Insert Data into the Bucketed Table:

```
INSERT INTO TABLE bucketed_finance_table VALUES (1, 'Alice'), (2, 'Bob'), (3,  
'Charlie');
```

---

## Step 4: View the Output in HDFS

### 1. Create an ORC Table:

**Use ORC (Optimized Row Columnar) format for efficient storage:**

```
CREATE TABLE orc_finance_table (  
    id INT,  
    name STRING  
)
```

### 2. Insert Data into the ORC Table:

```
INSERT INTO TABLE orc_finance_table SELECT * FROM finance_table;
```

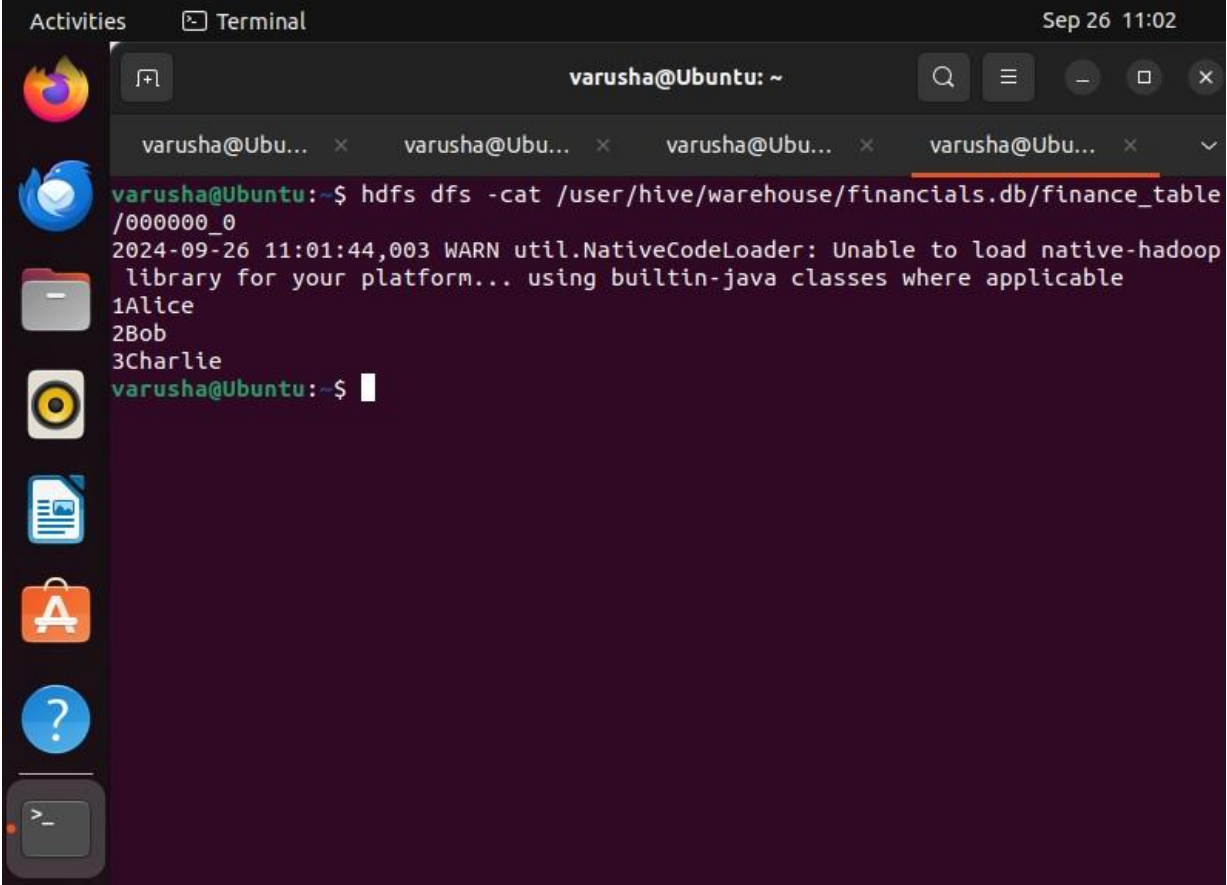
### 3. View the Output in HDFS:

**You can view the output by navigating to the HDFS directory where Hive stores the data. Use the following command to view the stored data:**

```
hdfs dfs -ls /user/hive/warehouse/financials.db/finance_table
```

**To view the contents of the ORC table:**

```
hdfs dfs -cat /user/hive/warehouse/financials.db/orc_finance_table/000000_0
```

**OUTPUT:**

A screenshot of a Linux terminal window titled 'Terminal' with the user 'varusha@Ubuntu'. The terminal shows the command `hdfs dfs -cat /user/hive/warehouse/financials.db/finance_table/000000_0` being executed. The output displays a warning message from 'util.NativeCodeLoader' and a list of names: '1Alice', '2Bob', and '3Charlie'. The prompt `varusha@Ubuntu:~$` is visible at the bottom.

```
varusha@Ubuntu:~$ hdfs dfs -cat /user/hive/warehouse/financials.db/finance_table/000000_0
2024-09-26 11:01:44,003 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
1Alice
2Bob
3Charlie
varusha@Ubuntu:~$
```

**RESULT:**

Thus, to create tables in Hive and write queries to access the data in the table was completed successfully.