Ex No: 6.a **Hive Databases -> Tables, Views**

AIM:
To write a script to Hive Databases -> Tables, Views,

Create Database Statement

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 **userdb**:

```
hive> CREATE DATABASE [IF NOT EXISTS] userdb;
```

or

```
hive> CREATE SCHEMA userdb;
```

The following query is used to verify a databases list:

```
hive> SHOW DATABASES;
default
userdb
```

JDBC Program

The JDBC program to create a database is given below.

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;

public class HiveCreateDb {
   private static String driverName =
"org.apache.hadoop.hive.jdbc.HiveDriver";

   public static void main(String[] args) throws SQLException {
      // Register driver and create driver instance

      Class.forName(driverName);
      // get connection
      Connection con = 
DriverManager.getConnection("jdbc:hive://localhost:10000/default", "", "");
      Statement stmt = con.createStatement();

      stmt.executeQuery("CREATE DATABASE userdb");
      System.out.println("Database userdb created successfully.");

      con.close();
   }
}
```

Save the program in a file named HiveCreateDb.java. The following commands are used to compile and execute this program.

```
$ javac HiveCreateDb.java
$ java HiveCreateDb
```

Output:

```
Database userdb created successfully.
```

Creating a View

You can create a view at the time of executing a SELECT statement. The syntax is as follows:

```
CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], 
...) ]
[COMMENT table_comment]
AS SELECT ...
```

Example

Let us take an example for view. Assume employee table as given below, with the fields Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than Rs 30000. We store the result in a view named **emp_30000.**

ID	Name	Salary	Designation	Dept
1201	Gopal	45000	Technical manager	TP
1202	Manisha	45000	Proofreader	PR
1203	Masthanvali	40000	Technical writer	TP
1204	Krian	40000	Hr Admin	HR
1205	Kranthi	30000	Op Admin	Admin

The following query retrieves the employee details using the above scenario:

```
hive> CREATE VIEW emp_30000 AS
SELECT * FROM employee
WHERE salary>30000;
```

Dropping a View

Use the following syntax to drop a view:

```
DROP VIEW view_name
```

The following query drops a view named as emp_30000:

```
hive> DROP VIEW emp_30000;
```

Ex No: 6.b **Hive Databases-> Functions and Indexes**

AIM:
To write a script to Hive Databases -> **Functions and Indexes**

The following queries demonstrate some built-in functions:

round() function

```
hive> SELECT round(2.6) from temp;
```

On successful execution of query, you get to see the following response:

```
3.0
```

floor() function

```
hive> SELECT floor(2.6) from temp;
```

On successful execution of the query, you get to see the following response:

```
2.0
```

ceil() function

```
hive> SELECT ceil(2.6) from temp;
```

On successful execution of the query, you get to see the following response:

```
3.0
```

Aggregate Functions

Hive supports the following built-in **aggregate functions**. The usage of these functions is as same as the SQL aggregate functions.

Return Type	Signature	Description
BIGINT	count(*), count(expr),	count(*) - Returns the total number of retrieved rows.
DOUBLE	sum(col), sum(DISTINCT col)	It returns the sum of the elements in the group or the sum of the distinct values of the column in the group.
DOUBLE	avg(col), avg(DISTINCT col)	It returns the average of the elements in the group or the average of the distinct values of the column in the group.

DOUBLE	min(col)	It returns the minimum value of the column in the group.
DOUBLE	max(col)	It returns the maximum value of the column in the group.

Creating an 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. Its syntax is as follows:

```
CREATE INDEX index_name
ON TABLE base_table_name (col_name, ...)
AS 'index.handler.class.name'
[WITH DEFERRED REBUILD]
[IDXPROPERTIES (property_name=property_value, ...)]
[IN TABLE index_table_name]
[PARTITIONED BY (col_name, ...)]
[
   [ ROW FORMAT ...] STORED AS ...
   | STORED BY ...
]
[LOCATION hdfs_path]
[TBLPROPERTIES (...)]
```

Example

Let us take an example for index. Use the same employee table that we have used earlier with the fields Id, Name, Salary, Designation, and Dept. Create an index named index_salary on the salary column of the employee table.

The following query creates an index:

```
hive> CREATE INDEX inedx_salary ON TABLE employee(salary)
AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler';
```

It is a pointer to the salary column. If the column is modified, the changes are stored using an index value.

Dropping an 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;
```