## Session: 16 The Hive Data-ware House

### 1. Introduction to Hive

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, but later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.

**Features of Hive**

- 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.

It is not built for Online Transactional Processing (OLTP) workloads. It is frequently used for data warehousing tasks like data encapsulation, Ad-hoc Queries, and analysis of huge datasets.

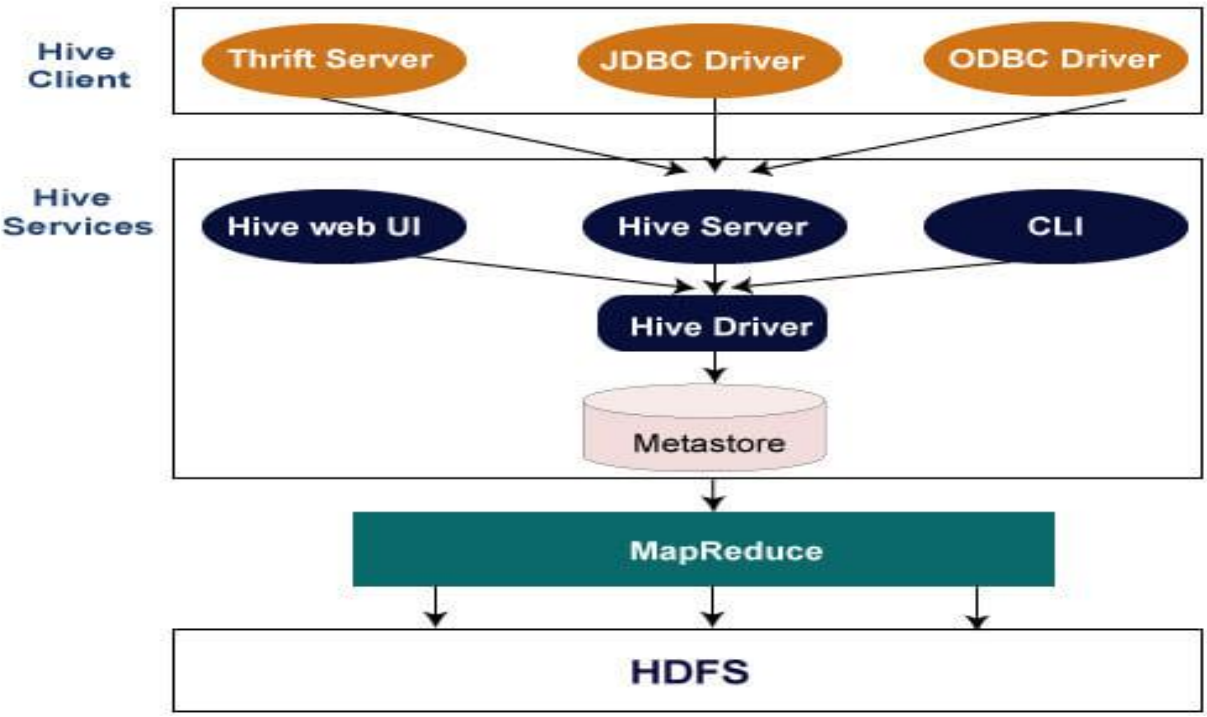
Apache Hive is a data warehouse software project that is built on top of the Hadoop ecosystem. It provides an SQL-like interface to query and analyze large datasets stored in Hadoop's distributed file system (HDFS)

It is designed to enhance scalability, extensibility, performance, fault-tolerance and loose-coupling with its input formats.

Hive uses a language called HiveQL, which is similar to SQL, to allow users to express data queries, transformations, and analyses in a familiar syntax. HiveQL statements are compiled into MapReduce jobs, which are then executed on the Hadoop cluster to process the data.

Hive can be used for a variety of data processing tasks, such as data warehousing, ETL (extract, transform, load) pipelines, and ad-hoc data analysis. It is widely used in the big data industry, especially in companies that have adopted the Hadoop ecosystem as their primary data processing platform.

## 2. Hive architecture




### 1. Hive Client

Hive allows writing applications in various languages, including Java, Python, and C++. It supports different types of clients such as:-

**Thrift Server** - It is a cross-language service provider platform that serves the request from all those programming languages that supports Thrift.

**JDBC Driver -** It is used to establish a connection between hive and Java applications. The JDBC Driver is present in the class org.apache.hadoop.hive.jdbc.HiveDriver.

**ODBC Driver** - It allows the applications that support the ODBC protocol to connect to Hive.

**Hive Services**

The following are the services provided by Hive: -

**Hive CLI -** The Hive CLI (Command Line Interface) is a shell where we can execute Hive queries and commands.

Hive Web User Interface - The Hive Web UI is just an alternative of Hive CLI. It provides a web-based GUI for executing Hive queries and commands.

**Hive MetaStore -** It is a central repository that stores all the structure information of various tables and partitions in the warehouse. It also includes metadata of column and its type

information, the serializers and deserializers which is used to read and write data and the corresponding HDFS files where the data is stored.

**Hive Server -** It is referred to as Apache Thrift Server. It accepts the request from different clients and provides it to Hive Driver.

**Hive Driver -** It receives queries from different sources like web UI, CLI, Thrift, and JDBC/ODBC driver. It transfers the queries to the compiler.

**Hive Compiler** - The purpose of the compiler is to parse the query and perform semantic analysis on the different query blocks and expressions. It converts HiveQL statements into MapReduce jobs.

**Hive Execution Engine -** Optimizer generates the logical plan in the form of DAG of map-reduce tasks and HDFS tasks. In the end, the execution engine executes the incoming tasks in the order of their dependencies.

**Modes of Hive:** There are two modes of hive

**1. Local Mode –**

It is used, when the Hadoop is built under pseudo mode which has only one data node, when the data size is smaller in term of restricted to single local machine, and when processing will be faster on smaller datasets existing in the local machine.

**2.Map Reduce Mode –**

It is used, when Hadoop is built with multiple data nodes and data is divided across various nodes, it will function on huge datasets and query is executed parallelly, and to achieve enhanced performance in processing large datasets.

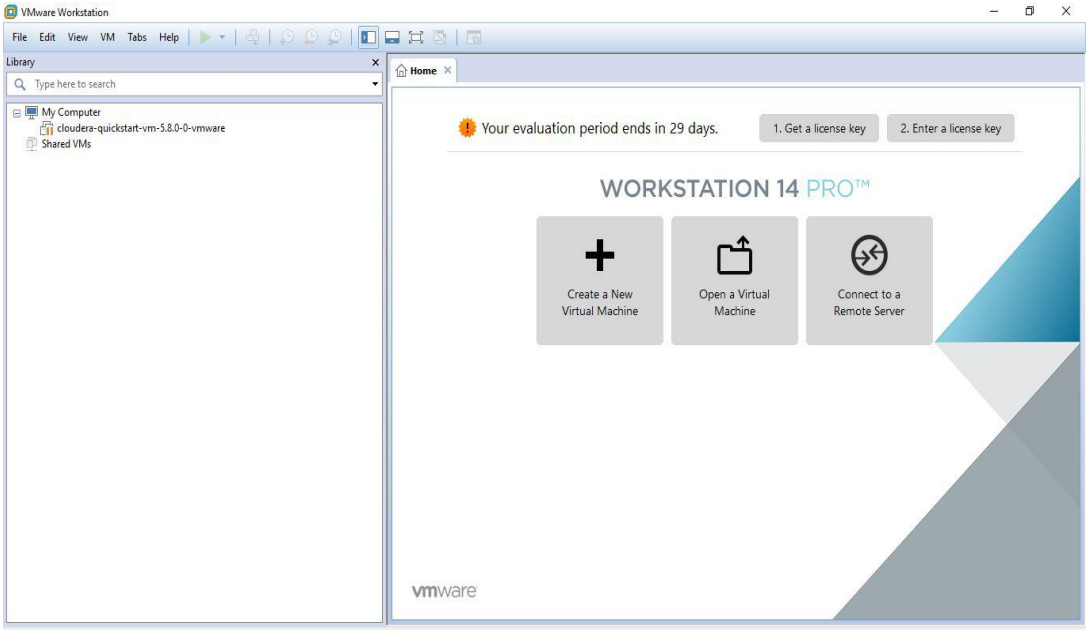
**Characteristics of Hive:**

- Databases and tables are built before loading the data.
- Hive as data warehouse is built to manage and query only structured data which is residing under tables.
- Programming in Hadoop deals directly with the files. So, Hive can partition the data with directory structures to improve performance on certain queries.
- Hive is compatible for the various file formats which are TEXTFILE, SEQUENCEFILE, ORC, RCFILE, etc.
- Hive uses derby database in single user metadata storage and it uses MYSQL for multiple user Metadata or shared Metadata.

## HIVE Installation

**Step 1:** Install VMware workstation on your system.

**Step 2:** Click on open a virtual device and open Hadoop framework.




**Step 3:** Now open the terminal and type **hive** at terminal.

```
[cloudera@quickstart ~]$ hive
```

## Features of Hive:

- It provides indexes, including bitmap indexes to accelerate the queries. Index type containing compaction and bitmap index as of 0.10.
- Metadata storage in a RDBMS, reduces the time to function semantic checks during query execution.
- Built in user-defined functions (UDFs) to manipulation of strings, dates, and other data-mining tools. Hive is reinforced to extend the UDF set to deal with the use-cases not reinforced by predefined functions.
- DEFLATE, BWT, snappy, etc are the algorithms to operation on compressed data which is stored in Hadoop Ecosystem.
- It stores schemas in a database and processes the data into the Hadoop File Distributed File System (HDFS).
- It is built for Online Analytical Processing (OLAP).

- It delivers various types of querying language which are frequently known as Hive Query Language (HVL or HiveQL).

**Advantages:**

**Scalability:** Apache Hive is designed to handle large volumes of data, making it a scalable solution for big data processing.

**Familiar SQL-like interface**: Hive uses a SQL-like language called HiveQL, which makes it easy for SQL users to learn and use.

**Integration with Hadoop ecosystem**: Hive integrates well with the Hadoop ecosystem, enabling users to process data using other Hadoop tools like Pig, MapReduce, and Spark.

**Supports partitioning and bucketing**: Hive supports partitioning and bucketing, which can improve query performance by limiting the amount of data scanned.

**User-defined functions:** Hive allows users to define their own functions, which can be used in HiveQL queries.

**Disadvantages:**

**Limited real-time processing:** Hive is designed for batch processing, which means it may not be the best tool for real-time data processing.

**Slow performance**: Hive can be slower than traditional relational databases because it is built on top of Hadoop, which is optimized for batch processing rather than interactive querying.

**Steep learning curve:** While Hive uses a SQL-like language, it still requires users to have knowledge of Hadoop and distributed computing, which can make it difficult for beginners to use.

**Lack of support for transactions:** Hive does not support transactions, which can make it difficult to maintain data consistency.

**Limited flexibility:** Hive is not as flexible as other data warehousing tools because it is designed to work specifically with Hadoop, which can limit its usability in other environments.

- Hive doesn’t support OLTP. Hive supports Online Analytical Processing (OLAP), but not Online Transaction Processing (OLTP).
- It doesn’t support subqueries.
- It has a high latency.

- Hive tables don’t support delete or update operations.

### 3. Comparison with Traditional Database

| Hive | Traditional database |
|---|---|
| **Schema on READ – it’s does not verify the schema while it’s loaded the data** | **Schema on WRITE** – table schema is enforced at data load time i.e if the data being loaded doesn’t conformed on schema in that case it will be rejected |
| It’s very easily scalable at low cost | Not much Scalable, costly scale up. |
| It’s based on Hadoop notation that is Write once and read many times | In traditional database we can read and write many time |
| Record level updates is not possible in Hive | Record level updates, insertions and deletes, transactions and indexes are possible |
| OLTP (On-line Transaction Processing) is not yet supported in Hive but it’s supported **OLAP (On-line Analytical Processing)** | Both OLTP (On-line Transaction Processing) and OLAP (On-line Analytical Processing) are supported in RDBMS. |

| RDBMS | HIVE |
|---|---|
| It is used to maintain database. | It is used to maintain data warehouse. |
| It uses SQL (Structured Query Language). | It uses HQL (Hive Query Language). |
| Schema is fixed in RDBMS. | Schema varies in it. |
| Normalized data is stored. | Normalized and de-normalized both type of data is stored. |
| Tables in RDBMS are sparse. | Table in hive are dense. |
| It doesn’t support partitioning. | It supports automation partition. |
| No partition method is used. | Sharding method is used for partition. |

**4. Basics of Hive Query Language.**

Hive is a data warehouse system that is used to query and analyze large datasets stored in the HDFS. Hive uses a query language called HiveQL, which is similar to SQL. There are key segments of Hive.

## HIVE Data Types

Hive data types are categorized in numeric types, string types, misc types, and complex types-

### Integer Types

| Type | Size | Range |
|---|---|---|
| TINYINT | 1-byte signed integer | -128 to 127 |
| SMALLINT | 2-byte signed integer | 32,768 to 32,767 |
| INT | 4-byte signed integer | 2,147,483,648 to 2,147,483,647 |
| BIGINT | 8-byte signed integer | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |

### Decimal Type

| Type | Size | Range |
|---|---|---|
| FLOAT | 4-byte | Single precision floating point number |
| DOUBLE | 8-byte | Double precision floating point number |

### Date/Time Types

**TIMESTAMP**

- It supports traditional UNIX timestamp with optional nanosecond precision.
- As Integer numeric type, it is interpreted as UNIX timestamp in seconds.
- As Floating-point numeric type, it is interpreted as UNIX timestamp in seconds with decimal precision.
- As string, it follows java.sql. Timestamp format "YYYY-MM-DD HH:MM:SS.fffffffff" (9 decimal place precision)

**Date:**

The Date value is used to specify a particular year, month and day, in the form YYYY--MM--DD.

### String Types

1. **STRING:** The string is a sequence of characters. It values can be enclosed within single quotes (') or double quotes (").

**2.Varchar**

The varchar is a variable length type whose range lies between 1 and 65535, which specifies that the maximum number of characters allowed in the character string.

**3.CHAR**

The char is a fixed-length type whose maximum length is fixed at 255.

## Complex Type

| Type | Size | Range |
|---|---|---|
| Struct | It is similar to C struct or an object where fields are accessed using the "dot" notation. | struct('CDAC','Delhi') |
| Map | It contains the key-value tuples where the fields are accessed using array notation. | map('first','cdac','last','delhi') |
| Array | It is a collection of similar type of values that indexable using zero-based integers. | array('cdac','delhi') |

## Create Database

1. **By using create database query**

   CREATE DATABASE database_name;

2. **To check whether database is created or not use**

   **Show databases;**

3. **Create data base by using owner and date**

   create database sunil1 **WITH DBPROPERTIES** ('creator'='sunil kumar','date'='2023-08-21');

4. **Get the extended information about the database and Display the HDFS path of a your database**.

   describe database extended **DBname**;

   describe database extended **sunil1;**

   describe database **sunil1;**

5. **drop the database**

   drop database **DBName;**

6. **Hive - Create Table**

   we can create a table by using the conventions similar to the SQL. It provides two types of table: -

   **A. Internal table**

   - The internal tables are also called managed tables as the lifecycle of their data is controlled by the Hive.
   - By default, these tables are stored in a subdirectory under the directory defined by **hive.metastore.warehouse.dir (i.e. /user/hive/warehouse).**
   - The internal tables are not flexible enough to share with other tools like Pig. If we try to drop the internal table, Hive deletes both table schema and data.

   **Create the table:**

   - create table sunil1.employee (Id int, Name string, Salary float);
   - describe sunil1.employee;
   - describe sunil1.new_employee;

   **B. External table**

   The external table allows us to create and access a table and a data externally. The '**external'** keyword is used to specify the external table, whereas the '**location'** keyword is used to determine the location of loaded data.

   Hive>create **external** table cont_list(countryid int, countryName string,capital string, population int)row format delimited

   > fields terminated by ','

   > **location** '/hivedir'; [ in hdfs]

   Fetch data from table:

   Hive>select *from cont_list where countryid=4;

7. **Drop a Hive External Table**

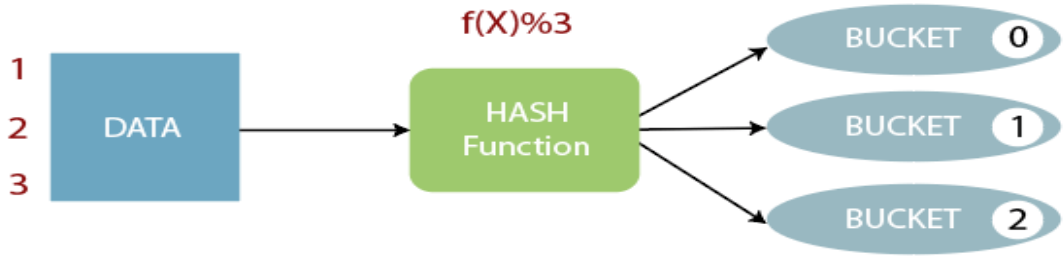
   **drop table [table_name];**

8. **Hive - Load Data**

Once the internal table has been created, the next step is to load the data into it. So, in Hive, we can easily load data from any file to the database.

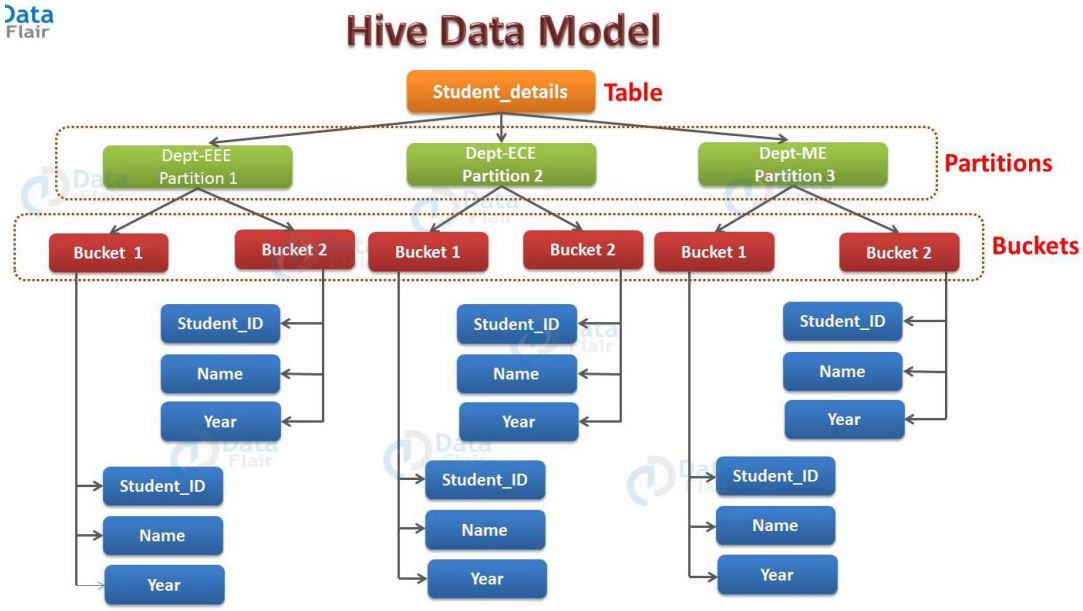
**Hive Partitioning and Bucketing?**

Apache Hive is an open-source data warehouse system used for querying and analyzing large datasets. Data in Apache Hive can be categorized into Table, Partition, and Bucket. The table in Hive is logically made up of the data being stored.




```
create table state_part(District string,Enrolments string) PARTITIONED
BY(state string);
```

```
create table emp_bucket(Id int, Name string , Salary float)
        clustered by (Id) into 3 buckets
        row format delimited
        fields terminated by ',' ;
```



**FILE FORMATS AND COMPRESSION IN APACHE HIVE**

HiveQL handles structured data only, much like SQL. In order to store the data in it, Hive has a derby database by default. The data will be stored as files in the backend framework while it shows the data in a structured format when it is retrieved. Some special file formats that Hive can handle are available, such as:

- Text File Format
- Sequence File Format
- RCFile (Row column file format)
- Avro Files
- ORC Files (Optimized Row Columnar file format)
- Parquet
- Custom INPUTFORMAT and OUTPUTFORMAT

**1) TEXT FILE FORMAT:**

Hive Text file format is a default storage format to load data from comma-separated values (CSV), tab-delimited, space-delimited, or text files that delimited by other special characters. You can use the text format to interchange the data with other client applications. The text file format is very common for most of the applications. Data is stored in lines, with each line being a record. Each line is terminated by a newline character (\n).The text file format storage option is defined by specifying "**STORED AS TEXTFILE**" at the end of the table creation.

**2) SEQUENCE FILE FORMAT:**

Flat files consisting of binary key-value pairs are sequence files. When converting queries to MapReduce jobs, Hive chooses to use the necessary key-value pairs for a given record. The key advantages of using a sequence file are that it incorporates two or more files into one file.The sequence file format storage option is defined by specifying "**STORED AS SEQUENCEFILE**" at the end of the table creation.

**3) RCFILE FORMAT:**

The row columnar file format is very much similar to the sequence file format. It is a data placement structure designed for MapReduce-based data warehouse systems. This also stores the data as key-value pairs and offers a high row-level compression rate. This will be used when there is a requirement to perform multiple rows at a time. RCFile format is supported by Hive version 0.6.0 and later.The RC file format storage option is defined by specifying "**STORED AS RCFILE**" at the end of the table creation.

**4) AVRO FILE FORMAT:**

Hive version 0.14.0 and later versions support Avro files. It is a row-based storage format for Hadoop which is widely used as a serialization platform. It's a remote procedure call and data serialization framework that uses JSON for defining data types and protocols and serializes data in a compact binary format to make it compact and efficient. This file format can be used in any of the Hadoop's tools like Pig and Hive.Avro is one of the common file formats in applications based on Hadoop. The option to store the data in the RC file format is defined by specifying **"STORED AS AVRO"** at the end of the table creation.

**5) ORC FILE FORMAT:**

The Optimized Row Columnar (ORC) file format provides a highly efficient way to store data in the Hive table. This file system was actually designed to overcome limitations of the other Hive file formats. ORC reduces I/O overhead by accessing only the columns that are required for the current query. It requires significantly fewer seek operations because all columns within a single group of row data are stored together on disk.The ORC file format storage option is defined by specifying "**STORED AS ORC**" at the end of the table creation.

**6) PARQUET:**

Parquet is a binary file format that is column driven. It is an open-source available to any project in the Hadoop ecosystem and is designed for data storage in an effective and efficient flat columnar format compared to row-based files such as CSV or TSV files. It only reads the necessary columns, which significantly reduces the IO and thus makes it highly efficient for large-scale query types. The Parquet table uses Snappy, which is a fast data compression and decompression library, as the default compression.

The parquet file format storage option is defined by specifying "**STORED AS PARQUET**" at the end of the table creation.

**7) CUSTOMER INPUTFORMAT & OUTPUTFORMAT:**

We can implement our own "inputformat" and "outputformat" incase the data comes in a different format. These "inputformat" and "outputformat" is similar to Hadoop MapReduce's input and output formats.

**Hive – Alter Table**

ALTER TABLE command can be used to perform alterations on the tables. We can modify multiple numbers of properties associated with the table schema in the Hive. Alteration on table modify's or changes its metadata and does not affect the actual data available inside the table. In general when we made some mistakes while creating the table structure then we use ALTER TABLE to change the characteristics of the schema. We can perform multiple operations with table schema like renaming the table name, add the column, change or replace the column name, etc.

**1. Renaming Table Name**

ALTER TABLE with RENAME is used to change the name of an already existing table in the hive.

Syntax:

ALTER TABLE <current_table_name> RENAME TO <new_table_name>;

**2. ADD Columns**

Syntax:

ALTER TABLE <table_name> ADD COLUMNS (<col-name> <data-type> COMMENT ”, <col-name> <data-type> COMMENT ”, ….. )

ALTER TABLE customer ADD COLUMNS ( contact BIGINT COMMENT ‘Store the customer contact number’);

**3. CHANGE Column**

CHANGE in ALTER TABLE is used to change the name or data type of an existing column or attribute.

**Syntax:**

ALTER TABLE <table_name> CHANGE <column_name> <new_column_name> <new_data_type>;

ALTER TABLE customer CHANGE demo_name customer_name STRING;

**4. REPLACE Column**

The REPLACE with ALTER TABLE is used to remove all the existing columns from the table in Hive. The attributes or columns which are added in the ALTER TABLE REPLACE statement will be replaced with the older columns.

**Syntax:**

ALTER TABLE <table_name> REPLACE COLUMNS (

<attribute_name> <data_type>,

<attribute_name> <data_type>,

.

.

.

);

## Command:

```
ALTER TABLE customer REPLACE COLUMNS (customer_name STRING);
```

**Advantages and Disadvantages of Hive Partitioning & Bucketing**

**a) Pros and Cons of Hive Partitioning**

**Pros:**

- It distributes execution load horizontally.
- In partition faster execution of queries with the low volume of data takes place. For example, search population from Vatican City returns very fast instead of searching entire world population.

**Cons:**

- There is the possibility of too many small partition creations- too many directories.
- Partition is effective for low volume data. But there some queries like group by on high volume of data take a long time to execute. For example, grouping population of China will take a long time as compared to a grouping of the population in Vatican City.
- There is no need for searching entire table column for a single record.

**b) Pros and Cons of Hive Bucketing**

**Pros:**

- It provides faster query response like portioning.
- In bucketing due to equal volumes of data in each partition, joins at Map side will be quicker.

**Cons:**

We can define a number of buckets during table creation. But loading of an equal volume of data has to be done manually by programmers.

**Operators and Functions**

- **Relational Operator in Hive**
- **Arithmetic Operator in Hive**

**HiveQL - Functions**

The Hive provides various in-built functions to perform mathematical and aggregate type operations. Here, we are going to execute such type of functions on the records of the below table:

**Mathematical Functions in Hive**

| Return type | Functions | Description |
|---|---|---|
| BIGINT | round(num) | It returns the BIGINT for the rounded value of DOUBLE num. |

| | | |
|---|---|---|
| BIGINT | floor(num) | It returns the largest BIGINT that is less than or equal to num. |
| BIGINT | ceil(num), ceiling(DOUBLE num) | It returns the smallest BIGINT that is greater than or equal to num. |
| DOUBLE | exp(num) | It returns exponential of num. |
| DOUBLE | ln(num) | It returns the natural logarithm of num. |
| DOUBLE | log10(num) | It returns the base-10 logarithm of num. |
| DOUBLE | sqrt(num) | It returns the square root of num. |
| DOUBLE | abs(num) | It returns the absolute value of num. |
| DOUBLE | sin(d) | It returns the sin of num, in radians. |
| DOUBLE | asin(d) | It returns the arcsin of num, in radians. |
| DOUBLE | cos(d) | It returns the cosine of num, in radians. |
| DOUBLE | acos(d) | It returns the arccosine of num, in radians. |
| DOUBLE | tan(d) | It returns the tangent of num, in radians. |
| DOUBLE | atan(d) | It returns the arctangent of num, in radians. |

```
hive> select Id, Name, sqrt(Salary) from employee_data ;
```

## Aggregate Functions in Hive

| Return Type | Operator | Description |
|---|---|---|
| BIGINT | count(*) | It returns the count of the number of rows present in the file. |
| DOUBLE | sum(col) | It returns the sum of values. |
| DOUBLE | sum(DISTINCT col) | It returns the sum of distinct values. |
| DOUBLE | avg(col) | It returns the average of values. |

| | | |
|---|---|---|
| DOUBLE | avg(DISTINCT col) | It returns the average of distinct values. |
| DOUBLE | min(col) | It compares the values and returns the minimum one form it. |
| DOUBLE | max(col) | It compares the values and returns the maximum one form it. |

| Return Type | Operator | Description |
|---|---|---|
| INT | length(str) | It returns the length of the string. |
| STRING | reverse(str) | It returns the string in reverse order. |
| STRING | concat(str1, str2, ...) | It returns the concatenation of two or more strings. |
| STRING | substr(str, start_index) | It returns the substring from the string based on the provided starting index. |
| STRING | substr(str, int start, int length) | It returns the substring from the string based on the provided starting index and length. |
| STRING | upper(str) | It returns the string in uppercase. |
| STRING | lower(str) | It returns the string in lowercase. |
| STRING | trim(str) | It returns the string by removing whitespaces from both the ends. |
| STRING | ltrim(str) | It returns the string by removing whitespaces from left-hand side. |
| TRING | rtrim(str) | It returns the string by removing whitespaces from right-hand side. |