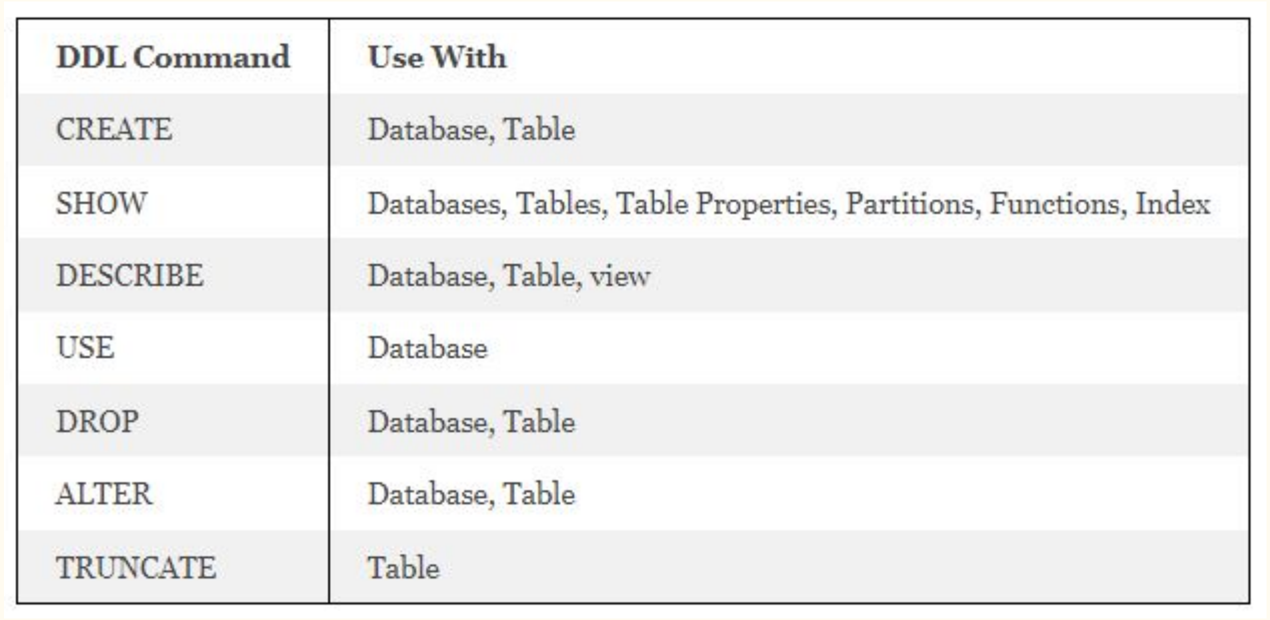
Chapter 3: HDFS, Hive and HiveQL, HBase (part 2)

By Ms. Tina D’abreo

Content

- HDFS-Overview, Installation and Shell, Java API
- Hive Architecture and Installation, Comparison with Traditional Database,
- **HiveQL Querying Data, Sorting And Aggregating,**
- Map Reduce Scripts, Joins & Sub queries
- HBase concepts, Advanced Usage, Schema Design, Advance Indexing, PIGGrunt – pig data model – Pig Latin – developing and testing Pig Latin scripts
- Zookeeper , how it helps in monitoring a cluster
- Build Applications with Zookeeper and HBase

Hive QL - DDL Statements



DDL Command	Use With
CREATE	Database, Table
SHOW	Databases, Tables, Table Properties, Partitions, Functions, Index
DESCRIBE	Database, Table, view
USE	Database
DROP	Database, Table
ALTER	Database, Table
TRUNCATE	Table

Hive QL - DDL

- ***Commands on Database or schema***
  - CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name

    [COMMENT database_comment]

    [LOCATION hdfs_path]

    [WITH DBPROPERTIES (property_name=property_value, ...)];
  - SHOW (DATABASES|SCHEMAS);
  - USE database_name;
  - DROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];
    - RESTRICT – The default behavior is RESTRICT, where DROP DATABASE will fail if the database is not empty
    - CASCADE – Use CASCADE option, if you wanted to drop all tables before dropping the database.
  - ALTER (DATABASE|SCHEMA) database_name SET DBPROPERTIES (property_name=property_value, ...);

Hive QL - DDL

- ***Commands on Database or schema***
  - DROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];
    - RESTRICT – The default behavior is RESTRICT, where DROP DATABASE will fail if the database is not empty
    - CASCADE – Use CASCADE option, if you wanted to drop all tables before dropping the database.

*In order to **drop the database**, the database should be empty meaning it should **not contain tables, views, or any other objects**. If you try to drop the database that has tables, you will get an error Database <dbname> is not empty. One or more tables exist.*

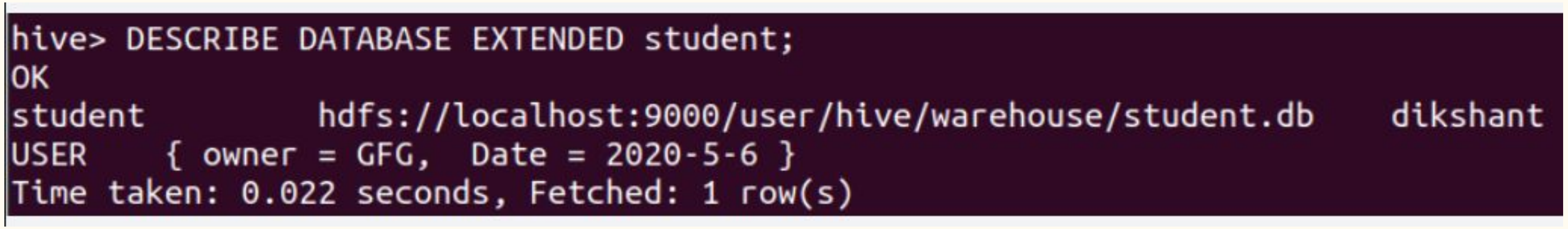
This can be changed using Cascade.

Hive QL - DDL

- *Commands on Database or schema*

```
ALTER (DATABASE|SCHEMA) <database_name> SET DBPROPERTIES
('<property_name>'='<property_value>',..);
```

```
ALTER DATABASE student SET DBPROPERTIES ( ' owner ' = ' GFG' , ' Date ' = ' 2020-5-6
');
```



```
hive> DESCRIBE DATABASE EXTENDED student;
OK
student         hdfs://localhost:9000/user/hive/warehouse/student.db    dikshant
USER    { owner = GFG,  Date = 2020-5-6 }
Time taken: 0.022 seconds, Fetched: 1 row(s)
```

Hive QL - DDL

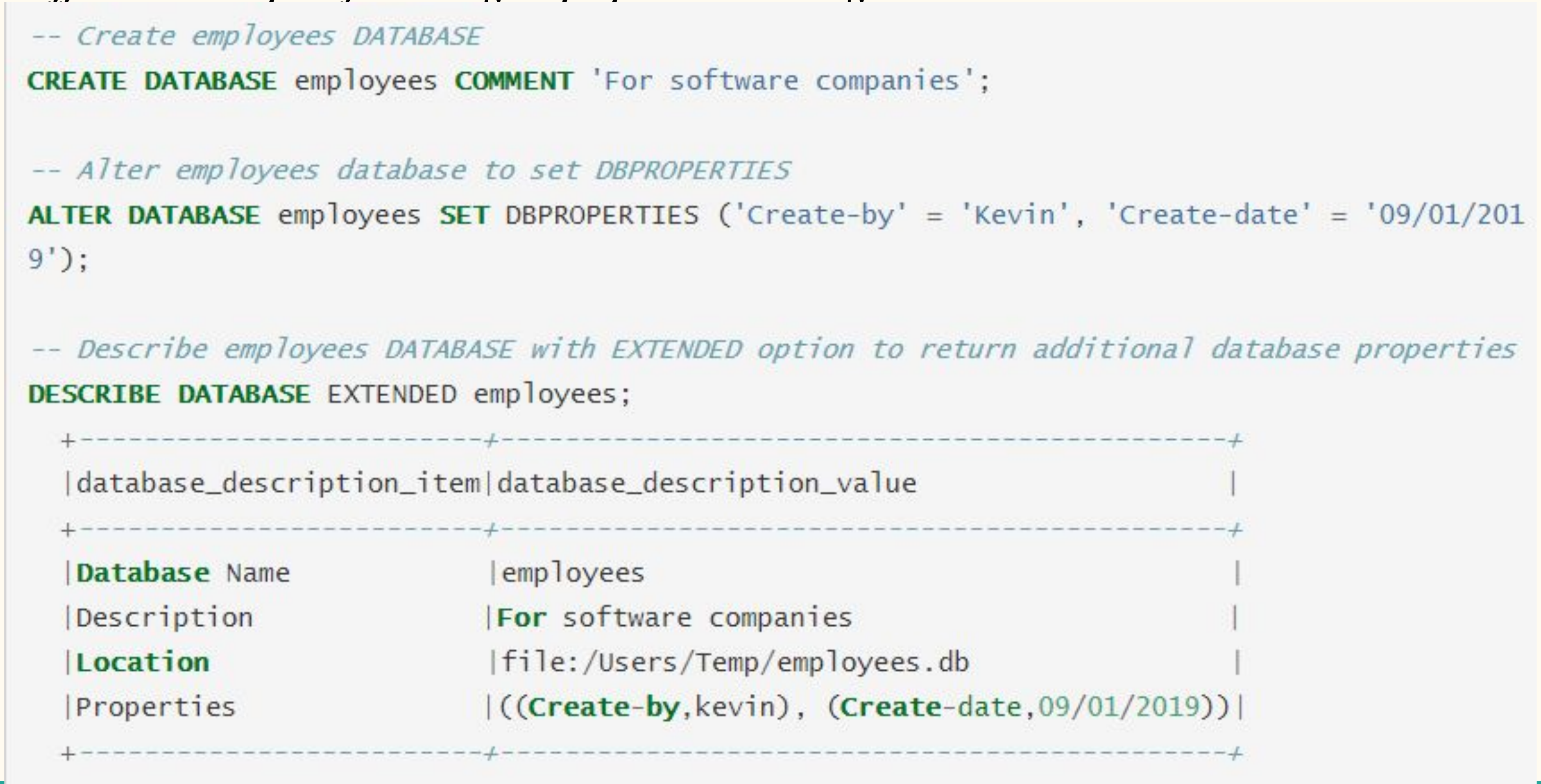
- ***Using Describe without DBproperties and EXTENDED***

```
-- Create employees DATABASE
CREATE DATABASE employees COMMENT 'For software companies';

-- Describe employees DATABASE.
-- Returns Database Name, Description and Root location of the filesystem
-- for the employees DATABASE.
DESCRIBE DATABASE employees;
  +-------------------------+-----------------------------+
  |database_description_item|database_description_value   |
  +-------------------------+-----------------------------+
  |Database Name            |employees                    |
  |Description              |For software companies       |
  |Location                 |file:/Users/Temp/employees.db|
  +-------------------------+-----------------------------+
```

Hive QL - DDL

- ***Difference in output after setting DBproperties and using EXTENDED with DESCRIBE***



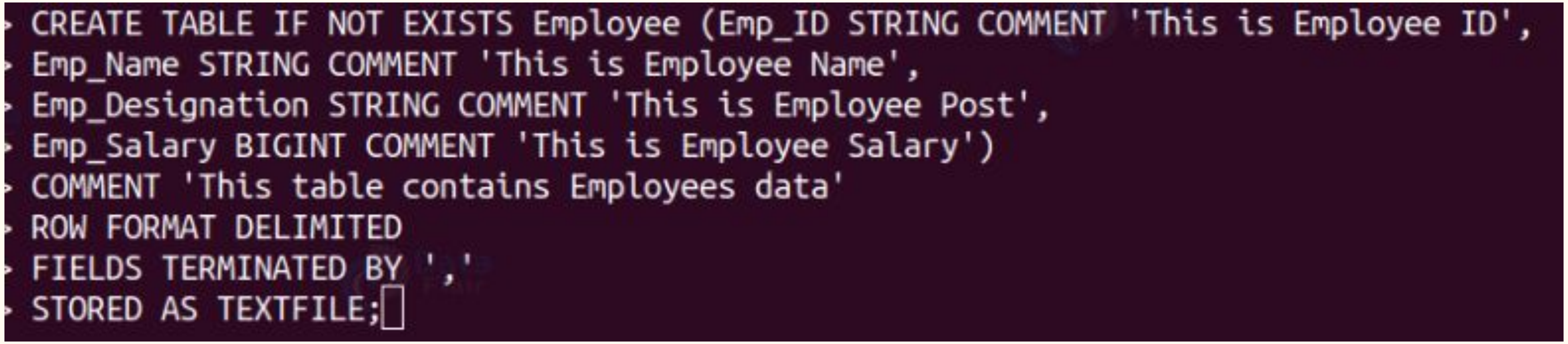
```
-- Create employees DATABASE
CREATE DATABASE employees COMMENT 'For software companies';

-- Alter employees database to set DBPROPERTIES
ALTER DATABASE employees SET DBPROPERTIES ('Create-by' = 'Kevin', 'Create-date' = '09/01/201
9');

-- Describe employees DATABASE with EXTENDED option to return additional database properties
DESCRIBE DATABASE EXTENDED employees;
 +-------------------------+---------------------------------------------+
 |database_description_item|database_description_value                   |
 +-------------------------+---------------------------------------------+
 |Database Name            |employees                                    |
 |Description              |For software companies                       |
 |Location                 |file:/Users/Temp/employees.db                |
 |Properties               |((Create-by,kevin), (Create-date,09/01/2019))|
 +-------------------------+---------------------------------------------+
```

Hive QL - DDL

- *Commands on Tables*
  - CREATE TABLE [IF NOT EXISTS] [db_name.] table_name [(col_name data_type [COMMENT col_comment], ... [COMMENT col_comment])] [COMMENT table_comment]
    [ROW FORMAT row_format]
    [STORED AS file_format]
    [LOCATION hdfs_path];



```
CREATE TABLE IF NOT EXISTS Employee (Emp_ID STRING COMMENT 'This is Employee ID',
Emp_Name STRING COMMENT 'This is Employee Name',
Emp_Designation STRING COMMENT 'This is Employee Post',
Emp_Salary BIGINT COMMENT 'This is Employee Salary')
COMMENT 'This table contains Employees data'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

Hive QL - DDL

- *Commands on Tables*
  - SHOW TABLES [IN database_name];
  - DESCRIBE [EXTENDED|FORMATTED] [db_name.] table_name[.col_name ([.field_name])];
  - DROP TABLE [IF EXISTS] table_name [PURGE];
  - Alter
    - Rename – ALTER TABLE table_name RENAME TO new_table_name;
    - Add – ALTER TABLE table_name ADD COLUMNS (column1 datatype, column2) ;
    - ALTER TABLE table_name SET TBLPROPERTIES (‘property_key’=’property_new_value’);
  - TRUNCATE TABLE table_name;

Hive QL - DML

- Insert

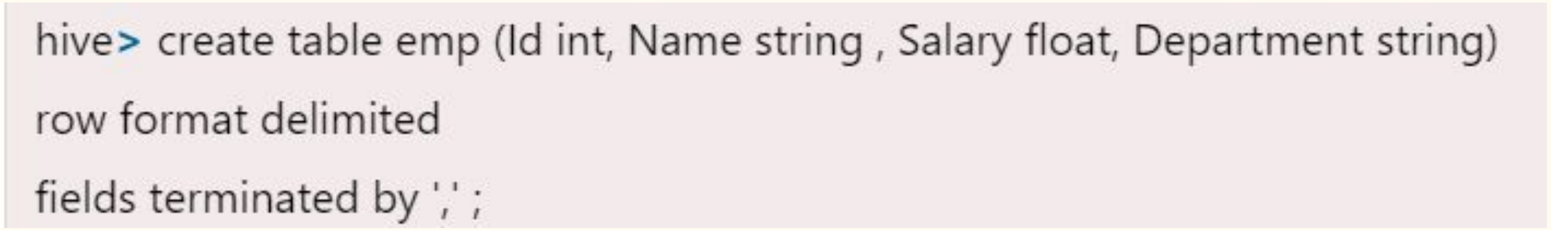
```
INSERT INTO TABLE student VALUES ('Dikshant',1,'95'),('Akshat', 2 , '96'),
('Dhruv',3,'90');
```

- Load
  - Hive provides us the functionality to load pre-created table entities either from our local file system or from HDFS. The LOAD DATA statement is used to load data into the hive table.
  - The LOCAL Switch specifies that the data we are loading is available in our Local File System. If the LOCAL switch is not used, the hive will consider the location as an HDFS path location.
  - The OVERWRITE switch allows us to overwrite the table data.

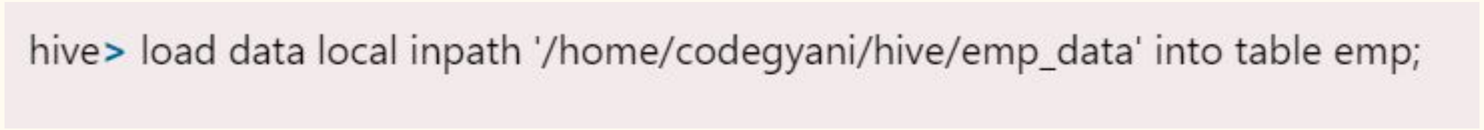
```
LOAD DATA [LOCAL] INPATH '<The table data location>' [OVERWRITE] INTO TABLE
<table_name>;
```

```
LOAD DATA LOCAL INPATH '/home/dikshant/Documents/data.csv' INTO TABLE student;
```

Hive QL - Group By and Having Clause



```
hive> create table emp (Id int, Name string , Salary float, Department string)
row format delimited
fields terminated by ',' ;
```

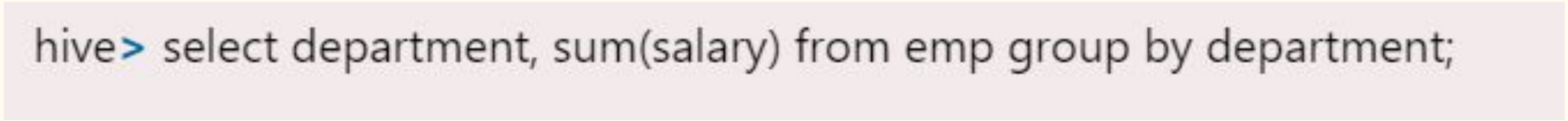


```
hive> load data local inpath '/home/codegyani/hive/emp_data' into table emp;
```

Hive QL - GroupBy and Having Clause

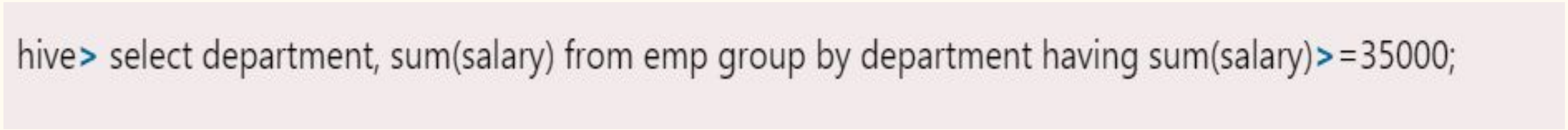
Group By clause is used to group the data from the multiple records based on one or more column.

It is generally used in conjunction with the aggregate functions (like SUM, COUNT, MIN, MAX and AVG) to perform an aggregation over each group.



```
hive> select department, sum(salary) from emp group by department;
```

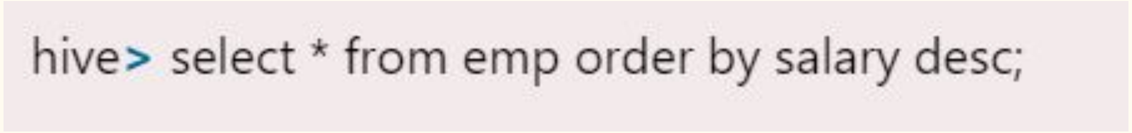
HAVING clause is used with GROUP BY clause. Its purpose is to apply constraints on the group of data produced by GROUP BY clause. Thus, it always returns the data where the condition is TRUE.



```
hive> select department, sum(salary) from emp group by department having sum(salary)>=35000;
```

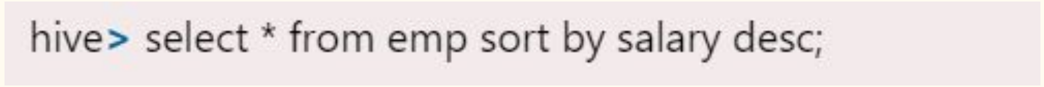
Hive QL - Order By and Sort By Clause

ORDER BY clause performs a complete ordering of the query result set. Hence, the complete data is passed through a single reducer. This may take much time in the execution of large datasets.



```
hive> select * from emp order by salary desc;
```

SORT BY clause is an alternative of ORDER BY clause. It orders the data within each reducer. Hence, it performs the local ordering, where each reducer's output is sorted separately.



```
hive> select * from emp sort by salary desc;
```

Hive QL - Group By and Having Clause

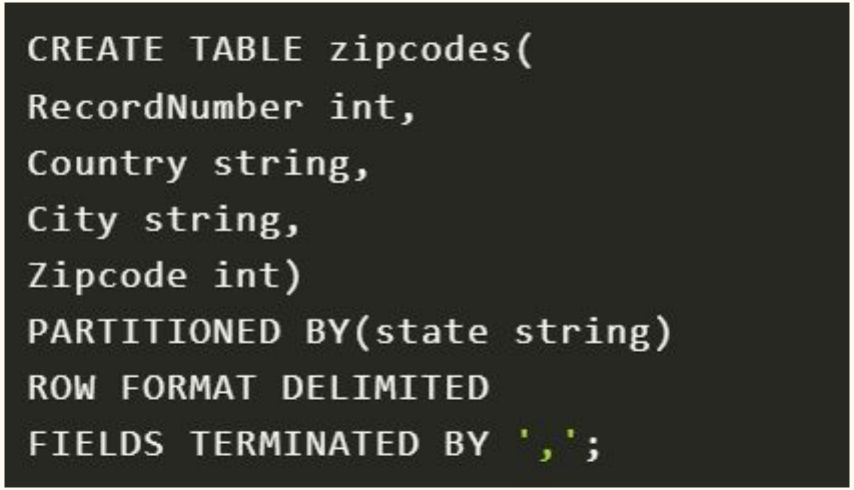
- Hive Aggregate Functions are the most used built-in functions that take a set of values and return a single value, when used with a group, it aggregates all values in each group and returns one value for each group.
- Like in SQL, Aggregate Functions in Hive can be used with or without GROUP BY functions however these aggregation functions are mostly used with GROUP BY.

Hive Partitioning

- Hive Partition is a way to organize large tables into smaller logical tables based on values of columns; one logical table (partition) for each distinct value.
- In Hive, tables are created as a directory on HDFS. A table can have one or more partitions that correspond to a sub-directory for each partition inside a table directory.
- Let’s assume you have a US census table which contains zip code, city, state and other columns.
    - Creating a partition on state splits the table into around 50 partitions, when searching for a zipcode with in a state (state=’CA’ and zipCode =’92704′) results in faster as it need to scan only in a state=CA partition directory.
- When creating partitions you have to be very cautious with the number of partitions it creates, as having too many partitions creates too many sub-directories in a table directory which bring unnecessarily and overhead to NameNode since it must keep all metadata for the file system in memory.

Hive Partitioning - Types

- Static & Dynamic Partitioning



```
CREATE TABLE zipcodes(
RecordNumber int,
Country string,
City string,
Zipcode int)
PARTITIONED BY(state string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

Hive Bucketing

- Hive Bucketing a.k.a (Clustering) is a technique to split the data into more manageable files, (By specifying the number of buckets to create). The value of the bucketing column will be hashed by a user-defined number into buckets.
- Bucketing can be created on just one column, you can also create bucketing on a partitioned table to further split the data which further improves the query performance of the partitioned table.
- Each bucket is stored as a file within the table’s directory or the partitions directories. Note that partition creates a directory and you can have a partition on one or more columns; these are some of the differences between Hive partition and bucket.
- From our example, we already have a partition on state which leads to around 50 subdirectories on a table directory, and creating a bucketing 10 on zipcode column creates 10 files for each partitioned subdirectory.

END (Part 1)