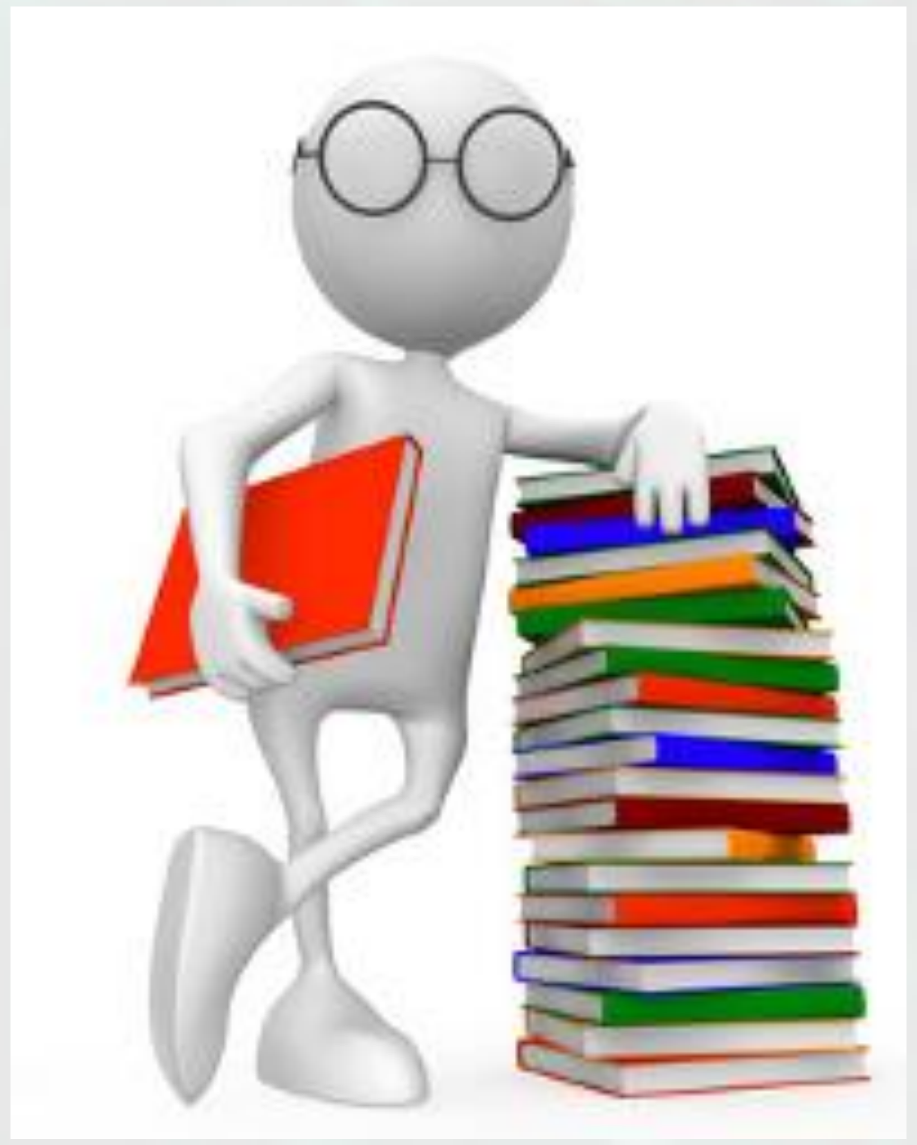
Master Executive di II Livello

BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

HiveQL

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

Topics



- What is Hive ?
- What Hive is not
- Hive features
- Hive architecture
- Limitations of Hive
- Hive QL
- Hive Datatypes
- Hive partitioning
- Basic Hive QL commands
- Hive SELECT
- Wordcount in Hive QL

What is Hive ?

- Is a SQL like Querying tool to query the data stored in HDFS and other Filesystems that integrate with Hadoop
- Developed by Facebook and later on taken by Apache
- It processes Structured data that can be stored into tables
- Efficient for Batch processing
- Is a lens between Map reduce and HDFS
- Provides us various storage file formats like Parquet, Sequence file, ORC file, Text file with significant compression.

What Hive is Not ?

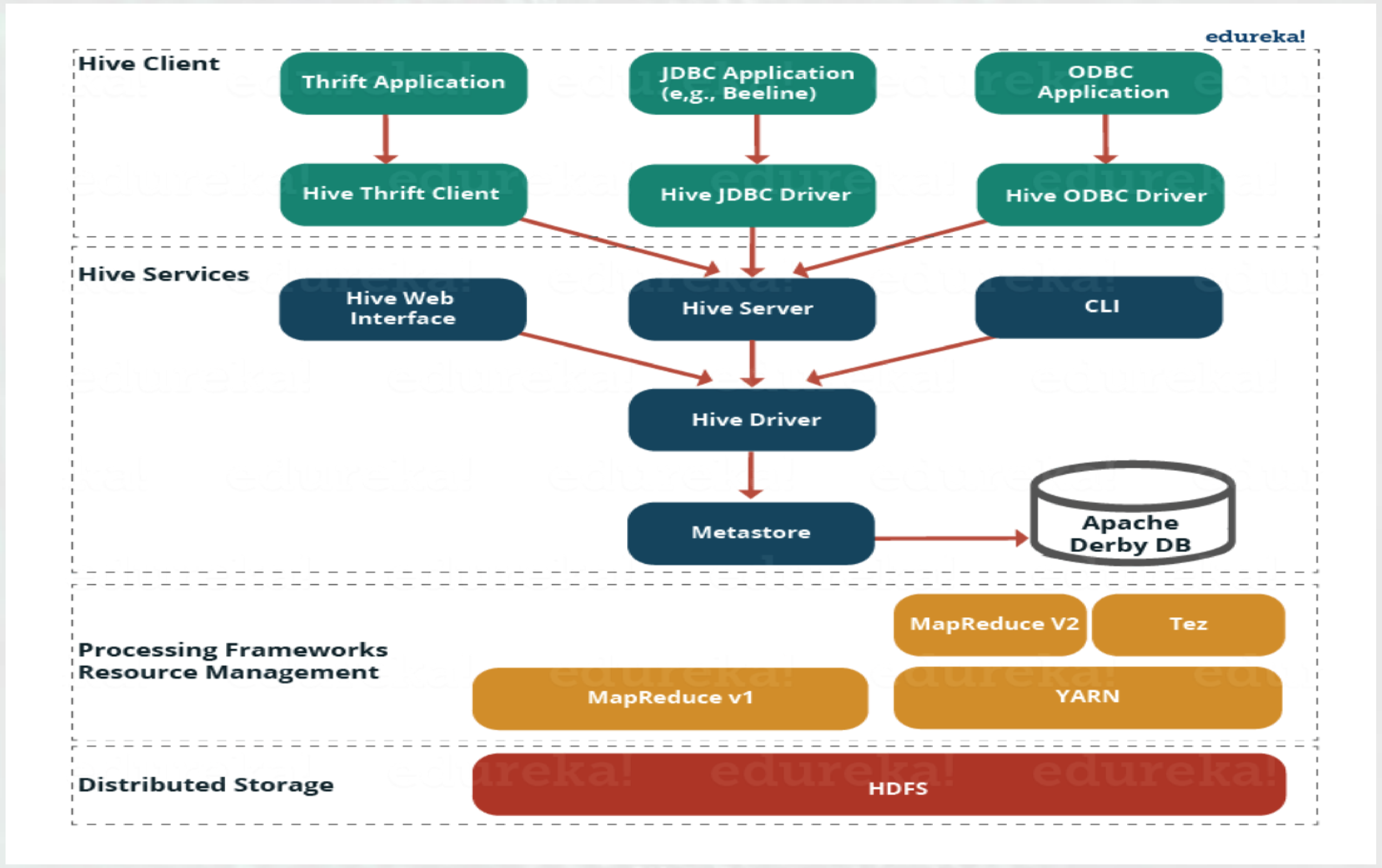
- Hive is not a Database. It just points to the data lying in HDFS.
- Hive is not a tool for OLTP. It is closer being as OLAP tool.
- It does not provide row level Insert, Update and Delete.
- Not used where fast response time is required as in RDBMS. Rather used where high latency is acceptable with batch processing.
- Does not support unstructured data like Audio, Video and Images.

Hive Features

- Indexing to provide acceleration, index type including compaction and bitmap index.
- Metadata storage in a relational database management system, significantly reducing the time to perform semantic checks during query execution.
- Built-in user-defined functions (UDFs) to manipulate dates, strings, and other data-mining tools.

Note: By default, Hive stores metadata in an embedded Apache Derby database, and other client/server databases like MySQL can optionally be used.

Hive Architecture



Hive Architecture(2)

- **Metastore**: Stores metadata for each of the tables such as their schema and location.
- **Driver**: Acts like a controller which receives the HiveQL statements. Monitors the life cycle and progress of the execution. The driver also acts as a collection point of data or query results obtained after the Reduce operation
- **Compiler**: Performs compilation of the HiveQL query, which converts the query to an execution plan. This plan contains the tasks and steps needed to be performed by the Hadoop MapReduce to get the output as translated by the query.

Hive Architecture(3)

- **Optimizer**: Performs various transformations on the execution plan to get an optimized DAG.
- **Executor**: After compilation and optimization, the executor executes the tasks. It interacts with the job tracker of Hadoop to schedule tasks to be run. It takes care of pipelining the tasks by making sure that a task with dependency gets executed only if all other prerequisites are run.
- **CLI, UI, and Thrift Server**: A command-line interface (CLI) provides a user interface for an external user to interact with Hive by submitting queries, instructions and monitoring the process status. Thrift server allows external clients to interact with Hive over a network, similar to the JDBC or ODBC protocols.

Limitations of Hive

- Hive is not designed for Online transaction processing (OLTP), it is only used for the Online Analytical Processing.(OLAP)
- Hive supports overwriting or apprehending data, but not updates and deletes.
- In Hive, sub queries are not supported.

Hive QL

- Based on SQL
- Offers basic support for indexes.
- HiveQL lacked support for transactions and materialized views, and only limited subquery support.
- Internally, a compiler translates HiveQL statements into a directed acyclic graph of MapReduce, Tez, or Spark jobs, which are submitted to Hadoop for execution.
- Support for insert, update, and delete with full ACID functionality

Hive - Datatypes

- All the data types in Hive are classified into four types:

  - Column Types

  - Literals

  - Null Values

  - Complex Types

Hive – Datatypes(2)

Column Types:

Column type are used as column data types of Hive. They are categorized further in to Integral types, String types, Time stamp, Dates and Decimals, Union types.

- **Integral**: TINYINT, SMALLINT, INT, BIGINT.
- **String**: VARCHAR(1-65355), CHAR(255).
- **Timestamp**: UNIX timestamp with optional nanosecond precision.
- **Dates**: DATE values are described in year/month/day format in the form {{YYYY-MM-DD}}.
- **Union**: Union is a collection of heterogeneous data types. Similar to dictionary in Python with key-value pairs.

Hive – Datatypes(2)

Literals:

- **Floating Point Types**: Floating point types are nothing but numbers with decimal points. Generally, this type of data is composed of DOUBLE data type.
- **Decimal Type**: Decimal type data is nothing but floating point value with higher range than DOUBLE data type. The range of decimal type is approximately -10^308 to 10^308..

Null Value:
Missing values are represented by the special value NULL.

Hive – Datatypes(3)

Complex Types:

- **Arrays:** Arrays in Hive are used the same way they are used in Java.
- **Maps:** Maps in Hive are similar to Java Maps.
- **Structs:** Structs in Hive is similar to using complex data with comment.

Hive – Partitioning

- **Partitions:** Hive organizes tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department. Using partition, it is easy to query a portion of the data.

- **Buckets:** Tables or partitions are sub-divided into buckets, to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table.

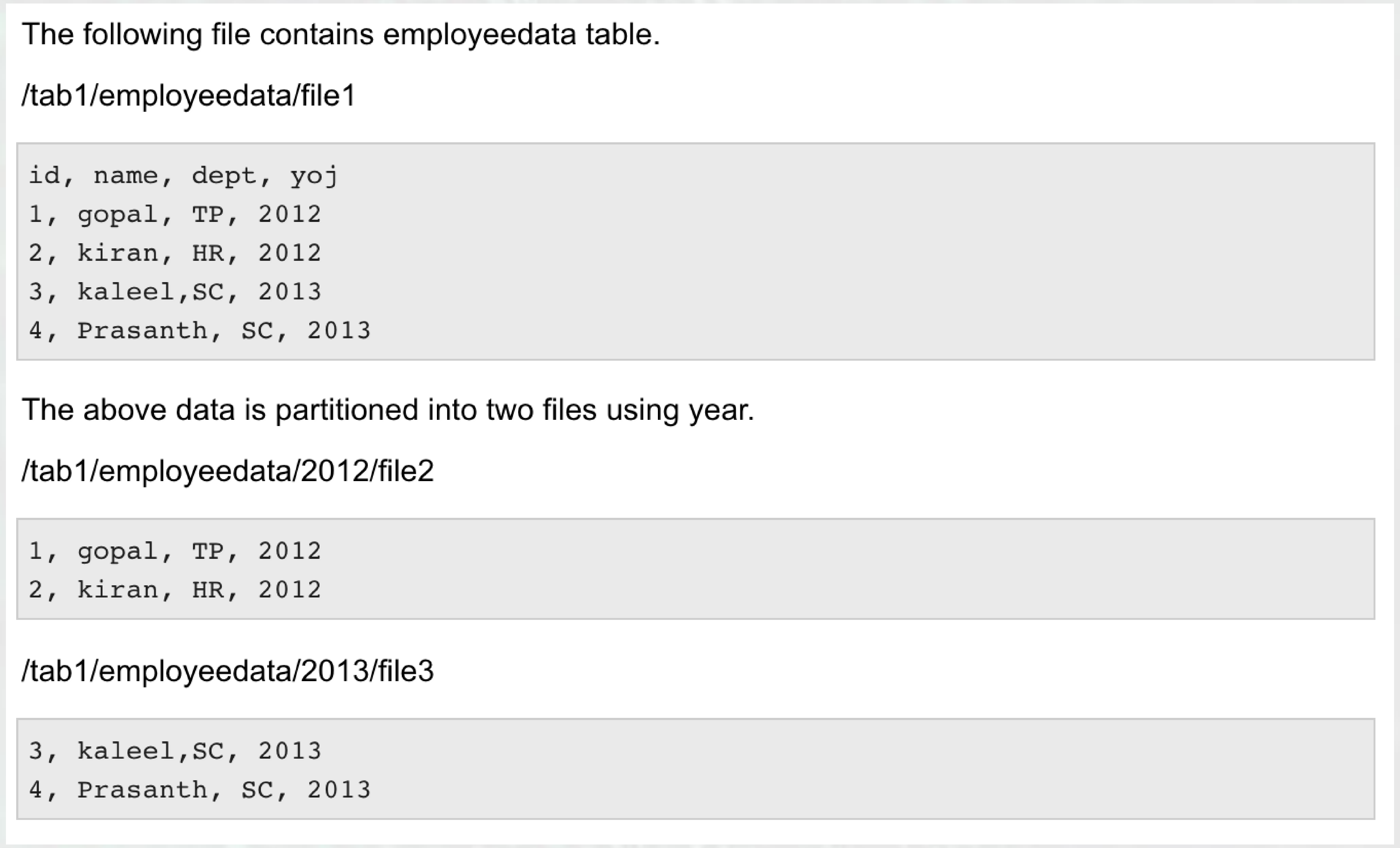
Hive – Partitioning(2)

Example:
A table named Tab1 contains employee data such as id, name, dept, and yoj (i.e., year of joining). Suppose you need to retrieve the details of all employees who joined in 2012.

Normal: A query searches the whole table for the required information.

However, if you partition the employee data with the year and store it in a separate file, it reduces the query processing time

Hive – Partitioning(3)



The following file contains employeedata table.

/tab1/employeedata/file1

```
id, name, dept, yoj
1, gopal, TP, 2012
2, kiran, HR, 2012
3, kaleel,SC, 2013
4, Prasanth, SC, 2013
```

The above data is partitioned into two files using year.

/tab1/employeedata/2012/file2

```
1, gopal, TP, 2012
2, kiran, HR, 2012
```

/tab1/employeedata/2013/file3

```
3, kaleel,SC, 2013
4, Prasanth, SC, 2013
```

Basic HiveQL Commands

- Find all the databases in HDFS –
  show databases;

- Switch to a specific database –
  use database_name; (Ex: use hive_demo;)

- Create a new table –

```
create table txnrecords( txnno INT,
                         txtdate STRING,
                         custno INT,
                         amount DOUBLE,
                         category STRING,
                         product STRING,
                         city STRING,
                         state STRING);
```

- Find the schema of a table –
  describe txnrecords;

HiveQL SELECT

- Syntax:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[CLUSTER BY col_list | [DISTRIBUTE BY col_list]
  [SORT BY col_list] ]
[LIMIT number];
```

Example:

```
SELECT * FROM employee WHERE salary>30000;
```

Word count in Hive QL

```
DROP TABLE IF EXISTS docs;

CREATE TABLE docs (line STRING);

LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;

CREATE TABLE word_counts AS

SELECT word, count(1) AS count FROM
   (SELECT explode(split(line, '\s')) AS word FROM docs) temp

GROUP BY word

ORDER BY word;
```

Word count in Hive QL(2)

DROP TABLE IF EXISTS docs;
CREATE TABLE docs (line STRING);

Explanation: Checks if table docs exists and drops it if it does. Creates a new table called docs with a single column of type STRING called line.

LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;

Explanation: Loads the specified file or directory (In this case "input_file") into the table. OVERWRITE specifies that the target table to which the data is being loaded into is to be re-written; Otherwise the data would be appended.

Word count in Hive QL(3)

```
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
   (SELECT explode(split(line, '\s')) AS word FROM docs) temp
GROUP BY word
ORDER BY word;
```

Explanation: Creates a table called word_counts with two columns: word and count. This query draws its input from the inner query (SELECT explode(split(line, '\s')) AS word FROM docs) temp". This query serves to split the input words into different rows of a temporary table aliased as temp. The GROUP BY WORD groups the results based on their keys. This results in the count column holding the number of occurrences for each word of the word column. The ORDER BY WORDS sorts the words alphabetically.

Master Executive di II Livello

BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

Grazie

fondazione INUIT TOR VERGATA