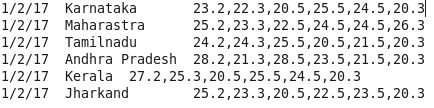
**Hive Questions –**

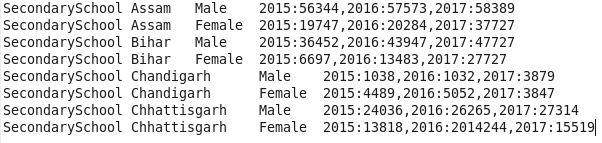
* we can run UNIX shell commands from Hive using the! Mark before the command .For example: !pwd at hive prompt will list the current directory.
* We can execute Hive queries from the script files by using the source command.  
  Example −

Hive> source /path/to/file/file\_with\_query.hql

* Date datatype in Hive – TIMESTAMP, DATE
* Numeric Data types – TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL
* String Data Types – Char, Varchar, String
* Misc Data Types – Boolean, Binary
* Collection data types in Hive – Array, Struct, Map (Also known as complex types) – These data types store multiple values in one row/column of a hive table..
* **Array:**  
  The first complex type is an array. It is nothing but a collection of items of similar data type. i.e, an array can contain one or more values of the same data type.  
  In our Array example, we will be using the dataset Temperature.txt where the fields are delimited by tab and the complex data type Array values are delimited by the comma.   
  **Dataset\_Temperature**  
  

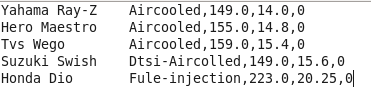
create table Temperature(date string,city string,MyTemp array<double>) row format delimited fields terminated by ‘\t’ collection items terminated by ‘,’;

To select a column and a value from the table we can use the below command.  
select city,MyTemp[0] from Temperature;

* **Map:**  
  A complex data type representing an arbitrary set of key-value pairs. The key part is a scalar type, while the value part can be a scalar or another complex type (ARRAY, STRUCT, or MAP).  
  In our Map example we will be using the dataset Schools.txt where the fields are delimited by tab, the complex type Map values are delimited by the comma.   
  **Dataset\_School\_Data**  
  

**Create table school (schooltype string, state string, gender char, total map<int,int>) row format delimited fields terminated by ‘\t’ collection items terminated by ‘,’ map keys terminated by ‘:’ ;**

select total[2016] from MySchools where state=’Assam’;

* **Struct:**  
  A complex data type, representing multiple fields of a single item.   
  In our Struct example, we will be using the dataset Bikes.txt where the fields are delimited by tab and the complex type Array values are delimited by the comma.  
  **Dataset\_Bikes**  
  

Create table Bike (name string, BikeFeatures struct<engine:string, cc:int>) row format delimited fields terminated by ‘\t’ collection items terminated by ‘,’;

select BikeFeatures.EngineType from MyBikes;

* Default Database in Hive - **Derby** is the default database.

**Indexing in HIVE**

* Indexes maintain the reference of the records.  Instead of searching all the records, we can refer to the index to search for a particular record.
* The major advantage of using indexing is; whenever we perform a query on a table that has an index, there is no need for the query to scan all the rows in the table. Further, it checks the index first and then goes to the particular column and performs the operation.
* Indexes are maintained in a separate table in Hive so that it won’t affect the data inside the table, which contains the data.
* Types of Indexes – Compact Indexing and Bitmap Indexing
* **Data in block indexing** - data will be able to identify which row is present in which block, so that when a query is triggered it can go directly into that block
* **Compact indexing** stores the pair of indexed column’s value and its blockid.
* **Bitmap indexing** stores the combination of indexed column value and list of rows as a bitmap.
* **Syntax to create Hive Index –**

CREATE INDEX index\_name ON TABLE table\_name (columns,....) AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;

ALTER INDEX index\_name on table\_name REBUILD;

* The org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler’ line specifies that a built in CompactIndexHandler will act on the created index, which means we are creating a compact index for the table.
* **WITH DEFERRED REBUILD** instructs Hive to first create an empty index; Step (B) is where you actually build the index with the **ALTER INDEX … REBUILD** command. Deferred index builds can be very useful in workflows where one process creates the tables and indexes, another loads the data and builds the indexes and a final process performs data analysis.
* 2nd command initiates the MR program
* Hive doesn’t provide automatic index maintenance, so you need to rebuild the index if you overwrite or append data to the table.
* Syntax to view the index –

show formatted index on table\_name;

* We can have any number of indexes on a particular table and any type of indexes.
* Syntax to create Bitmap index-

CREATE INDEX olympic\_index\_bitmap ON TABLE olympic (age)AS 'BITMAP' WITH DEFERRED REBUILD;

ALTER INDEX olympic\_index\_bitmap on olympic REBUILD;

* Syntax to drop an index –

**DROP INDEX IF EXISTS olympic\_index ON olympic;**

* With different types (compact,bitmap) of indexes on the same columns, for the same table, the index which is created first is taken as the index for that table on the specified columns.
* Indexes are advised to build on the columns on which you frequently perform operations.
* Building more number of indexes also degrade the performance of your query.
* Type of index to be created should be identified prior to its creation (if your data requires bitmap you should not create compact).This leads to increase in time for executing your query.
* **Cast Function** - This function is used to convert from one data type to another. The only type conversion function is CAST. The syntax of CAST is

CAST( expr as <type> )

The CAST function converts the expr into the specified type.

* **EXPLODE Function** - This function takes array as an input and outputs the elements of array into separate rows. The syntax of EXPLODE is

EXPLODE( ARRAY<A> )

* **COALESCE Function** - The COALESCE function returns the fist not NULL value from the list of values. If all the values in the list are NULL, then it returns NULL.

Example: COALESCE(NULL,NULL,5,NULL,4) returns 5

* **LOAD DATA** LOCAL INPATH './examples/files/kv1.txt' OVERWRITE INTO TABLE pokes; to load data in hive
* we can change the default location of Managed tables using the LOCATION keyword while creating the managed table.
* **To** **configure** **remote** **metastore** in Hive, hive-site.xml file has to be configured with the below property –

hive.metastore.uris

thrift: //node1 (or IP Address):9083

IP address and port of the metastore host