HDFS Theoretical Questions

General Theoretical Questions

Question 1: List all big data-specific formats that you know.

Answer 1: There are many file formats, which are being used to deal big-data. Plain text file, Sequence file, Avro, Parquet and ORC. Below a short description is provided for each big-data type:

Plain text file: Data is saved in *txt* or *csv* formats, they are used in the non-Hadoop environment as well. Heavy disk space/storage can be occupied by these plain text file formats. Thus, a robust compression is required.

Sequence file: This file type was mainly developed for MapReduce. Each entry of the data is encoded with a key and a value. Hence the data is stored in a binary format and occupies lesser storage space on the disk than the file formats which are text-based. It is also compatible with block-level compression, which is one of the advantages of the Sequence file.

Avro file: While dealing with the big-data files, Avro format is a good choice. On top of a normal file-format, it also supports serialization-deserialization and blocks compression. It is a row based and a splittable format. Avro is only a machine-readable file format.

Parquet and *ORC file*: ORC stands for Optimized Row Columnar. Unlike the Avro file, it also stores the data in columnar format. Hence, the horizontal and vertical partition of the data is possible. However, these file formats are also machine-readable only. Because these file formats are columnar, they outperform in terms of storage optimization than any other file formats.

Question 2: Why the compression of data matters for Hadoop?

Answer 2: Necessity of compression in a typical Hadoop eco-system is because of the huge volumes of data, it must deal with. Reduction in- (1) space needed for storing data and (2) data transfer speed to or from the disk, could be two significant advantages while dealing with big-data. These two advantages could be introduced with data compression. For example, MapReduce job for a compressed-large volume results into a low-latency task.

However, the data compression rate must be tuned with a trade-off between compression and speed of computation. Compressed data first gets decompressed before any other operation on the cluster. Thus, it increases CPU utilization along with the compression. The more compression is set for data, the more resources are used to first decompress it.

YARN application/commands

Question 3: What is YARN? What are YARN's two most important functions?

Answer 3: YARN stands for Yet Another Resource Negotiator. YARN sits in between of Hadoop File System (HDFS) and processing layer. It is essentially used for the two most important function, mentioned below:

(1) Resource Management: YARN allocates resources such as memory, to the application.

(2) Job Scheduling: It supports multiple scheduling methods for submitting the job to be processed in sequence. Some of the state-of-the-art scheduling methods are FIFO, Capacity Scheduler and Fair Scheduler.

Question 4: List all running applications

Answer 4: The Running application can be visualized with both: YARN UI and HDFS terminal. Figure 1 shows a screenshot of all the running application with YARN UI. Figure 2 lists all the running application with HDFS terminal.

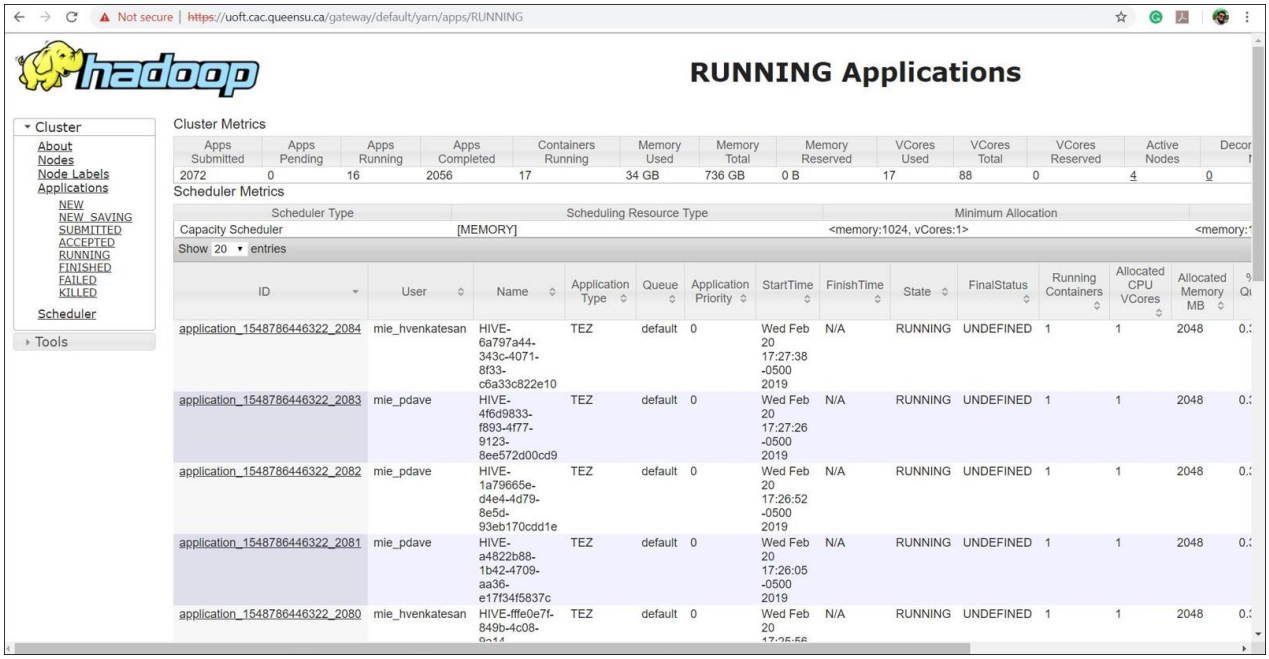

Figure 1. Running the application with YARN UI.

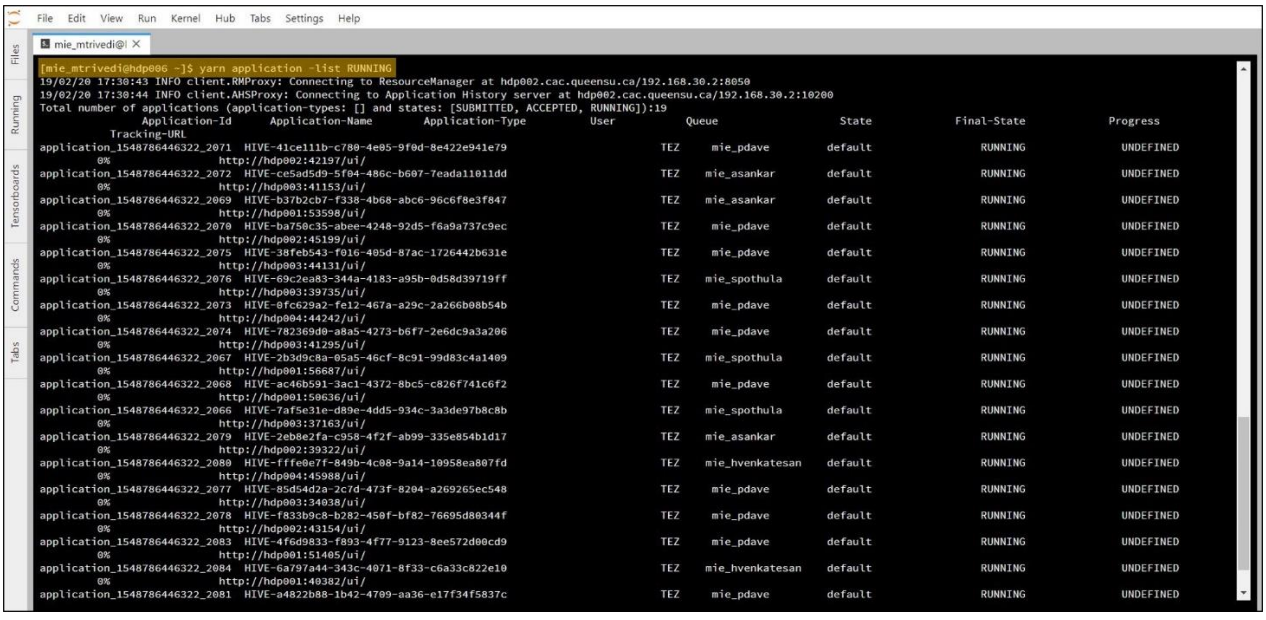

Figure 2. List of all the running application with HDFS terminal (Highlighted yellow is the command).

HDFS commands

Question 5: Create a folder/directory 'Lab1_results' in your own HDFS directory.
Answer 5: Command used to list the directories/files in a home folder, before generating *Lab1_results* directory: *hdfs dfs -ls*. Figure 3 and Figure 4 shows the listing of all the directories before generating Lab1_results directory. All the commands are implemented in HDFS terminal only.

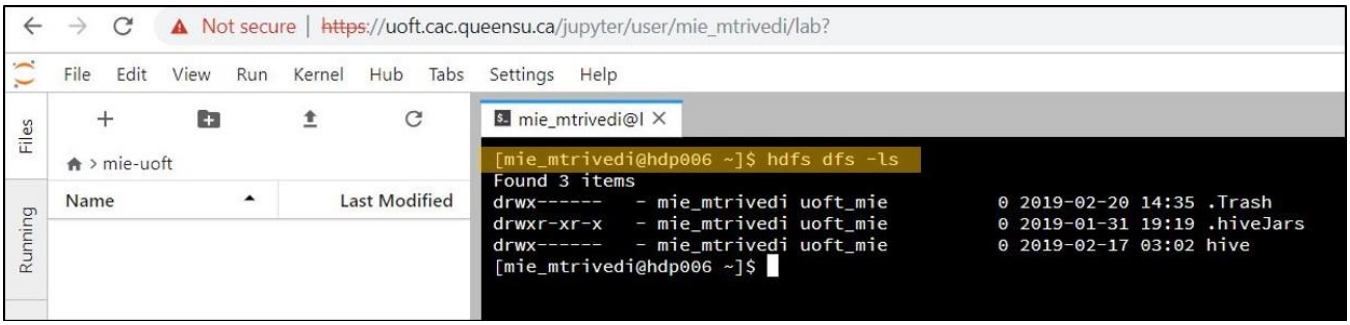

Figure 3. Listing of all the directories/files in present HDFS directory with HDFS terminal.

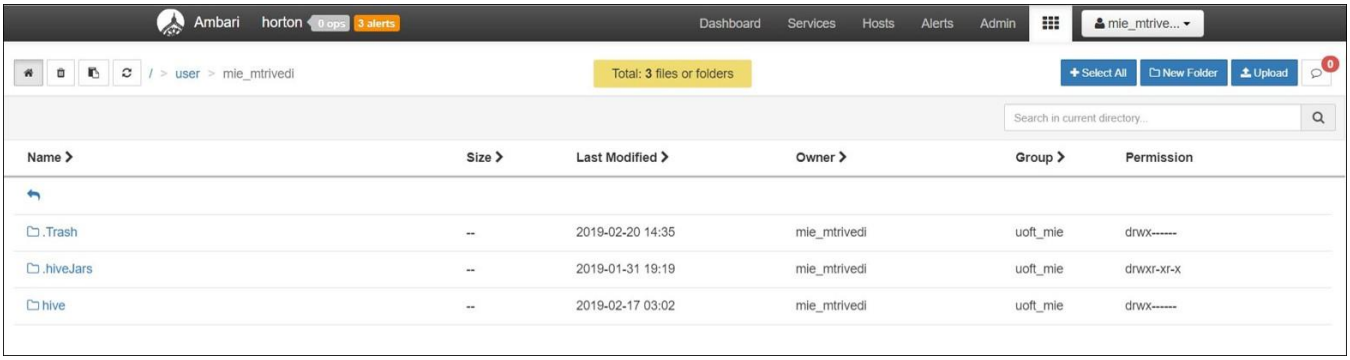

Figure 4. Listing of all the directories/files with HDFS UI.

Directory *Lab1_results* is generated with *hdfs dfs -mkdir Lab1_results* in the present working directory. Figure 5 shows the usage of *mkdir* command to generate *Lab1_results* along with the listing of all the directories after generating *Lab1_results*.

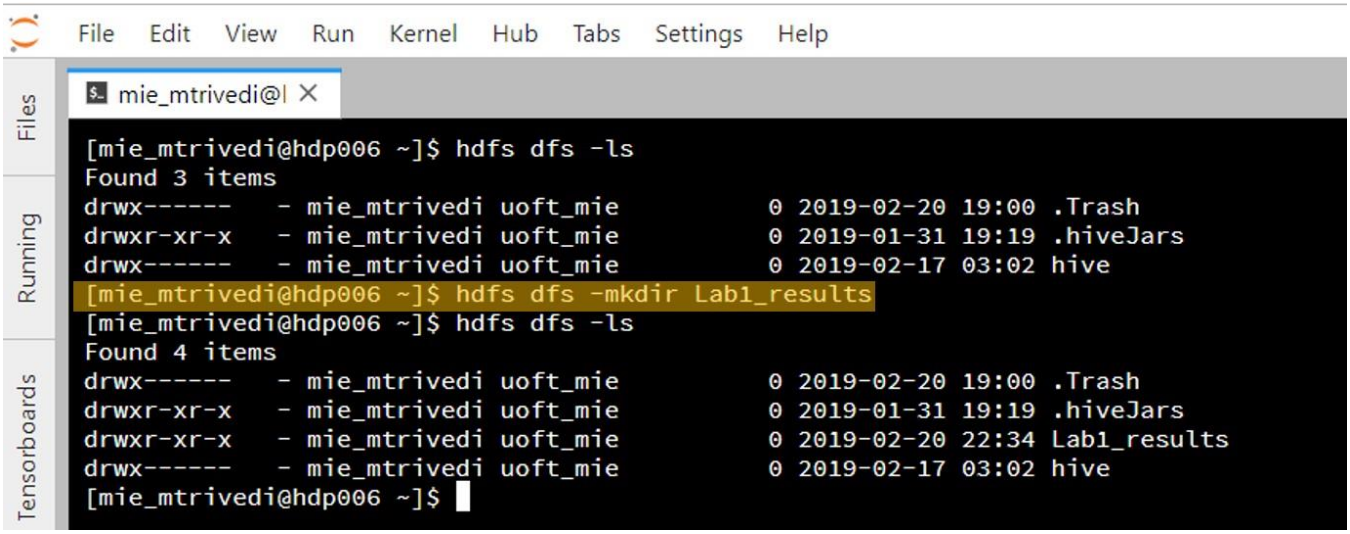

Figure 5. Command to generate a directory *Lab1_results*, with *mkdir* command in HDFS.

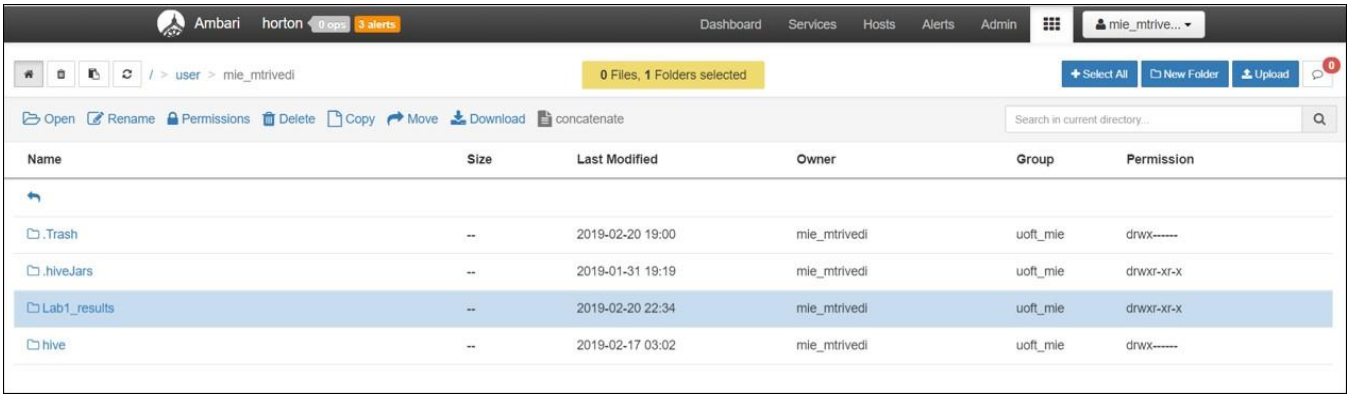

Figure 6. Checking HDFS UI for Lab1_results after generating it with HDFS command.

Question 6: Where in HDFS is MIE_Lecture4.ys_game file located? Provide path of the file.

Answer 6: Command *hdfs dfs -find / -name ys_game* could be used to find the location of this file with respect to the root directory. Figure 7 shows the usage of *find* command in HDFS terminal.

Path of the file ys_game: /apps/hive/warehouse/mie_lecture4.db/ys_game

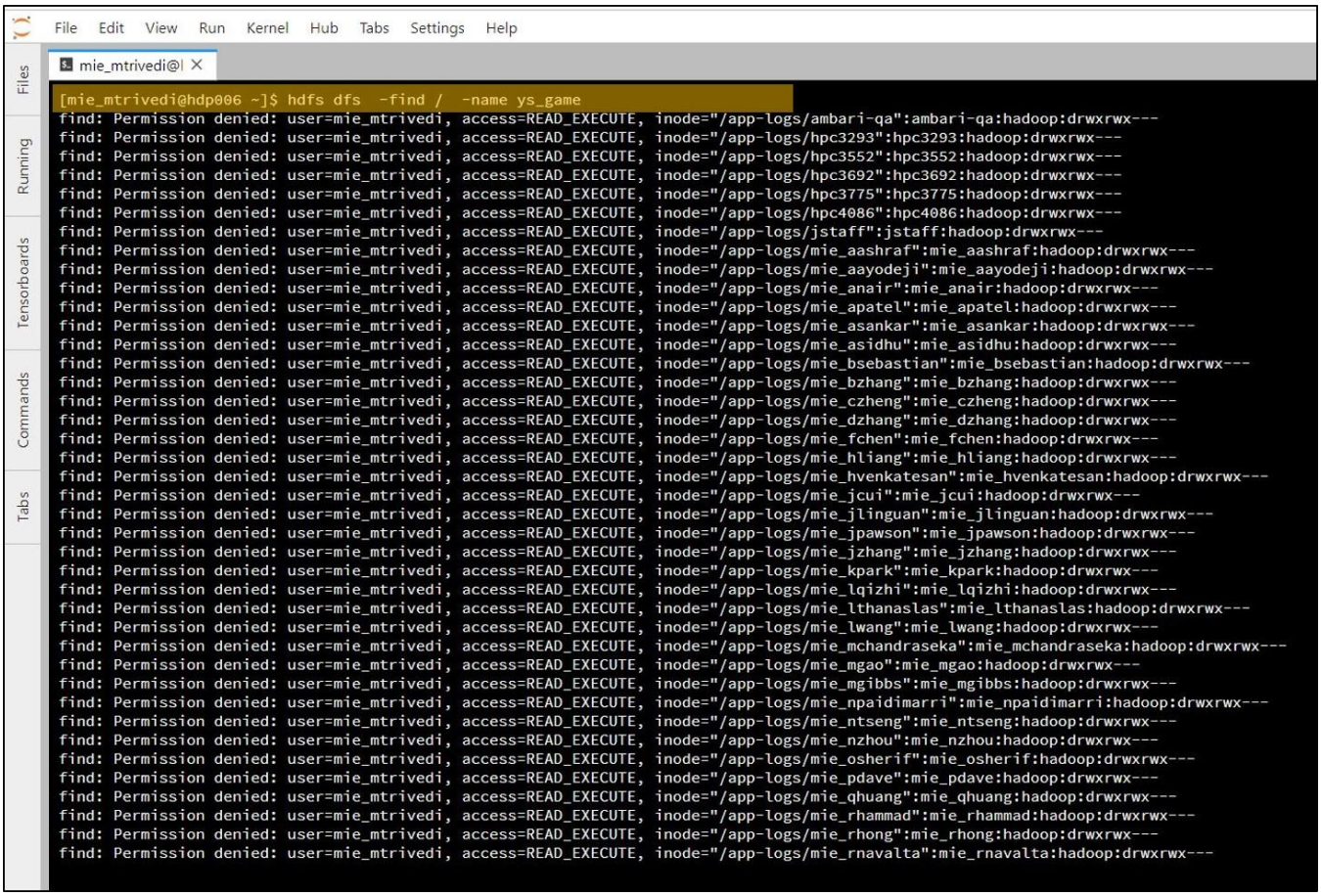

Figure 7. Command to locate *ys_game* file in HDFS terminal.

The output of *find* command lists the location of all the files with respect to its root directory. Scrolling down the output terminal window, *ys_game* could be located. Figure 8 shows the location of *ys_game* with the highlighted field.

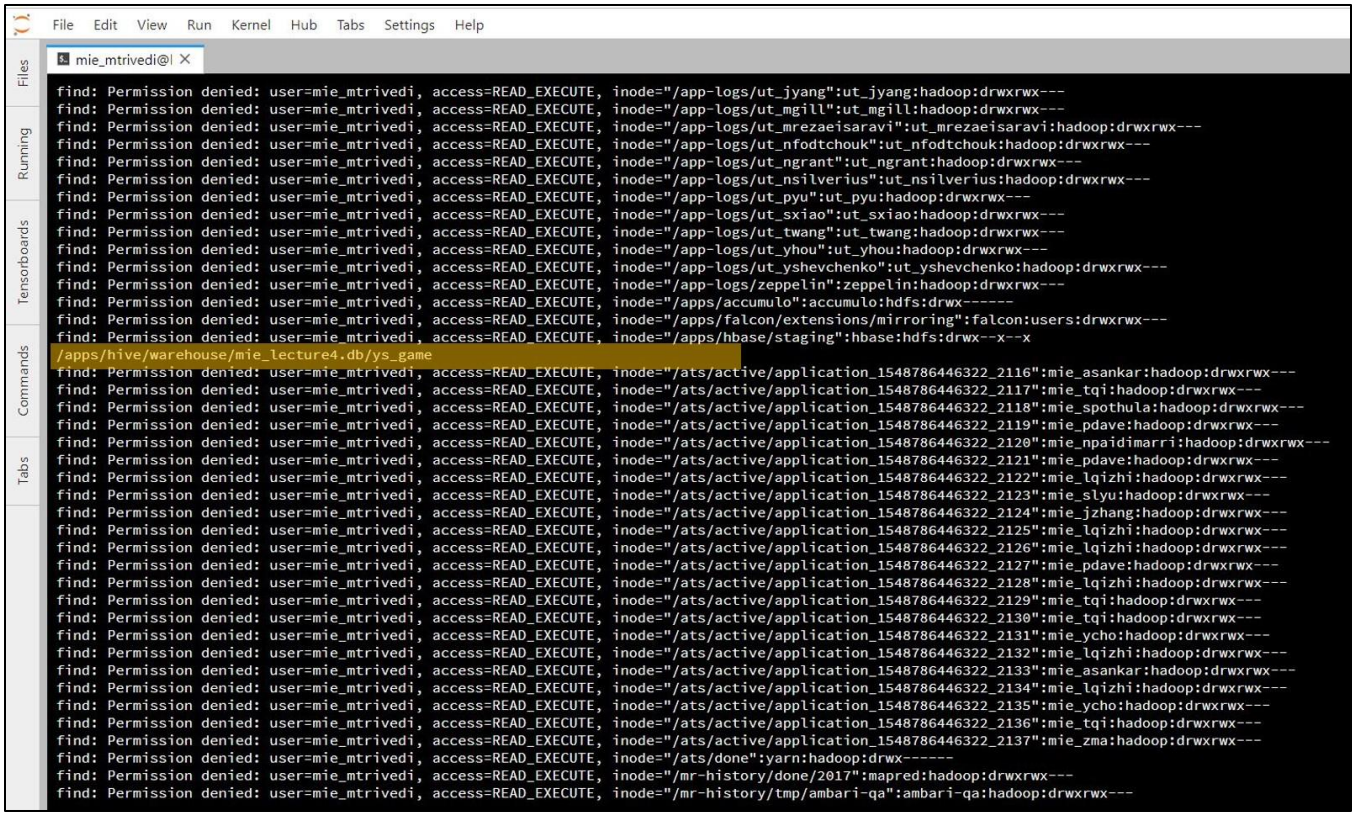

Figure 8. Global path of *ys_game* is highlighted with yellow color.

Question 7: What format is underlying Hive tables saved in? How can you find the format?

Answer 7: Through HDFS, entry to the Hive environment is made.

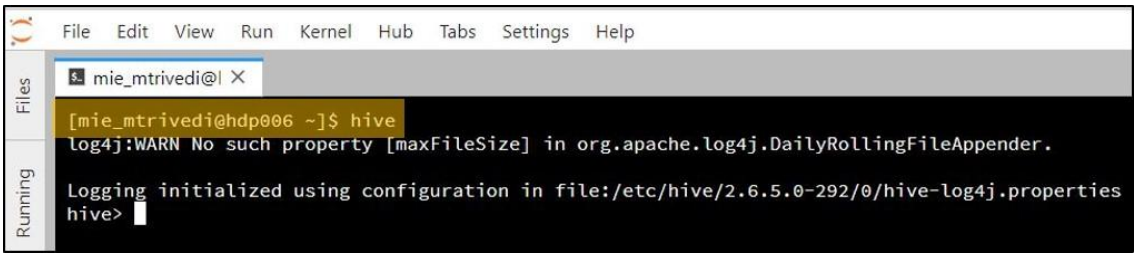

Figure 9. Entering the Hive with HDFS.

After entering the Hive, all the tables are listed with *show tables*.

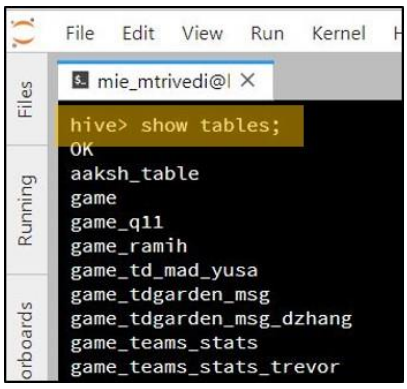

Figure 10. Listing of all the tables after entering in Hive environment with HDFS.

Once all the tables are listed, any of them is picked up and file format is checked with *desc formatted Table_name*. Here, table game is chosen for checking the file format and thus the command *desc formatted game* is used.

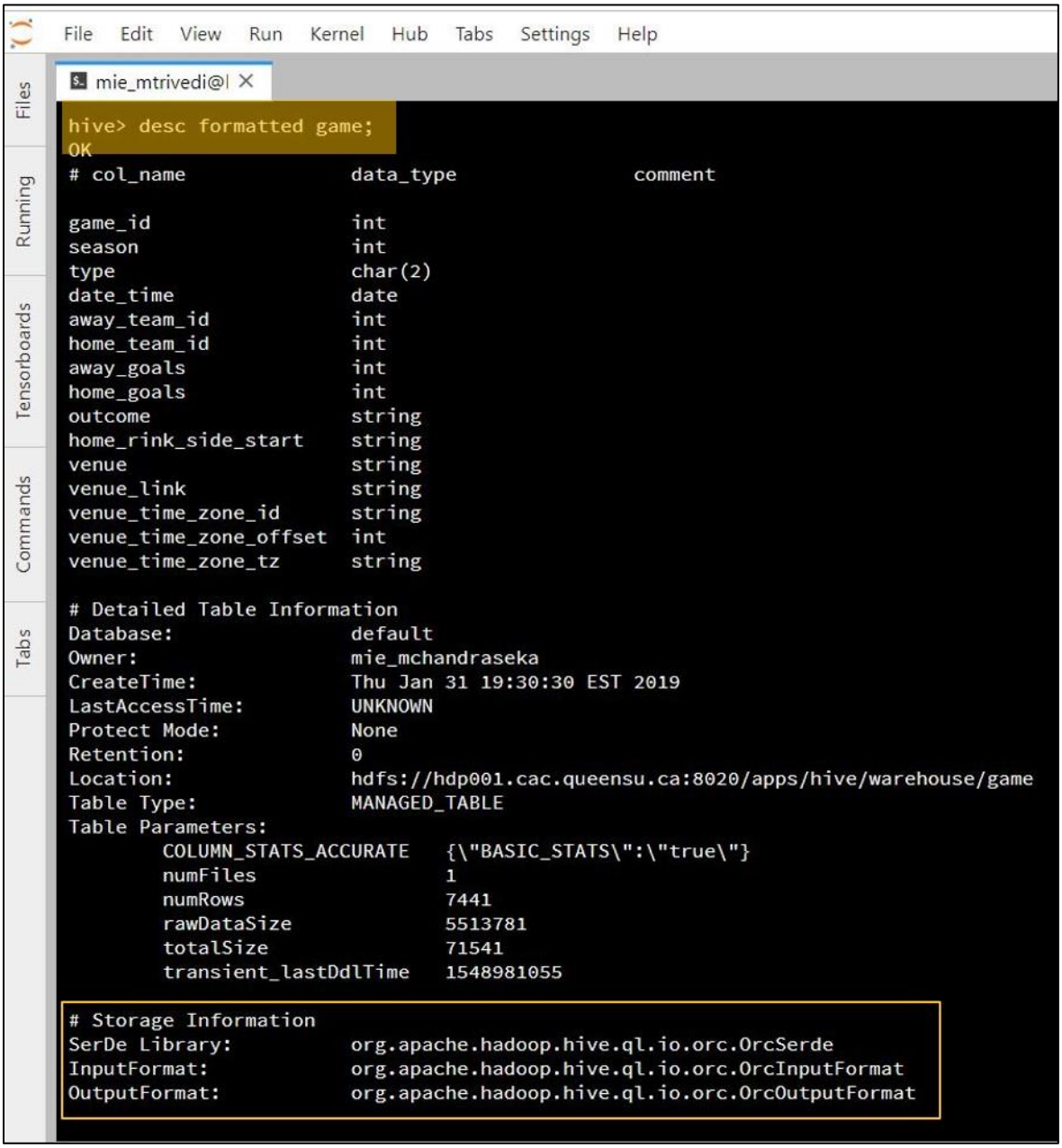

Figure 11. The file format is checked for the *game* table. The file format is bounded with a rectangle which is shown with yellow color.

Above described commands can be used to check the underlying format of all the files. As it can be seen from Figure 11, the file format of the *game* is ORC (Optimized Row Columnar).