Big Data Hadoop and Spark Developer

Apache Hive

Learning Objectives

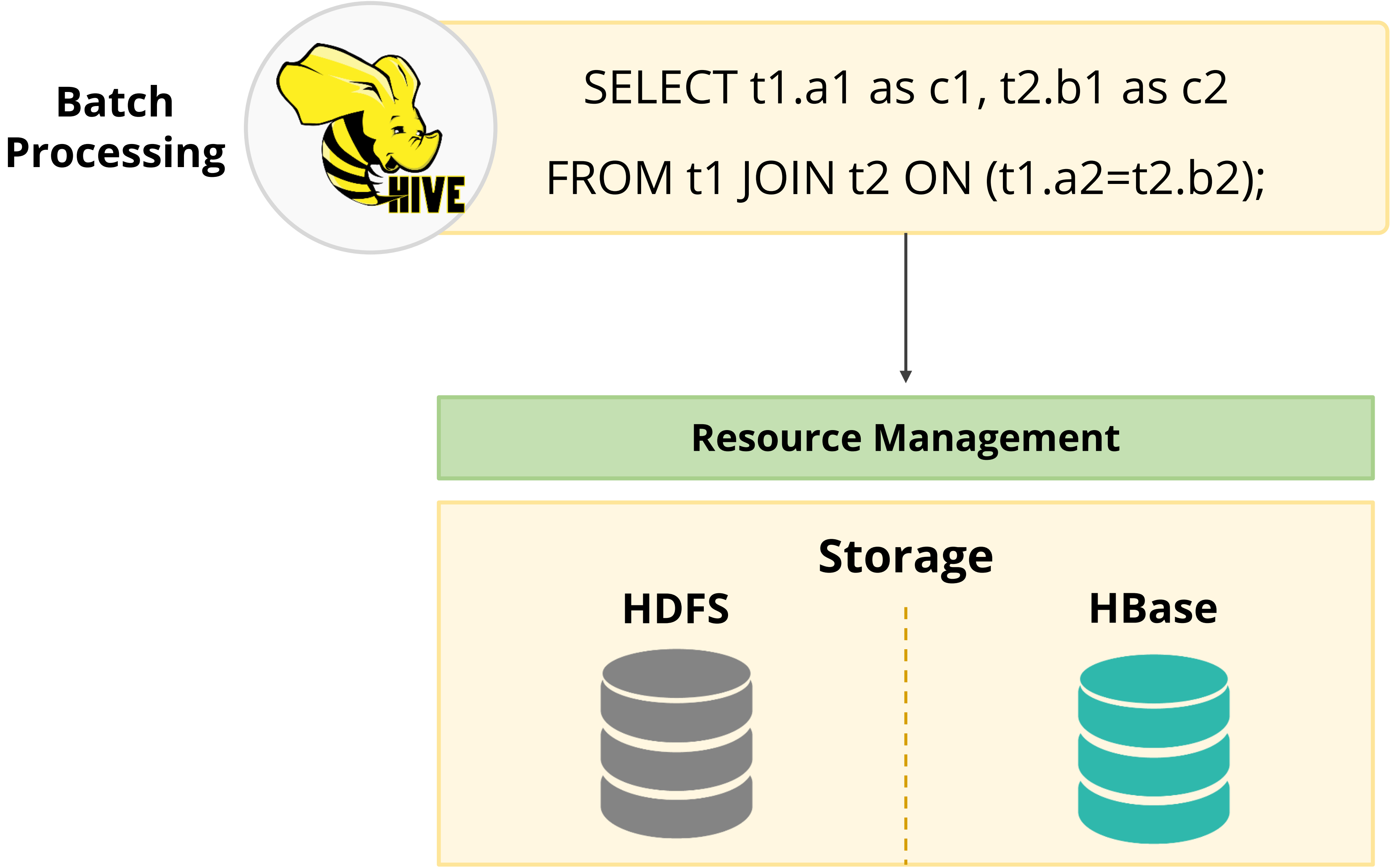
By the end of this lesson, you will be able to:

- Define Hive and its architecture
- Create and manage tables using Hue Web UI and Beeline
- Understand various file formats supported in Hive
- Use HiveQL DDL to create tables and execute queries

Hive: SQL over Hadoop MapReduce

Apache Hive

Hive provides a SQL like interface for users to extract data from the Hadoop system.



Batch Processing

```
SELECT t1.a1 as c1, t2.b1 as c2
FROM t1 JOIN t2 ON (t1.a2=t2.b2);
```

Features of Hive

- Originally developed by Facebook around 2007
- Is an open-source Apache project
- High level abstraction layer on top of MapReduce and Apache Spark
- Uses HiveQL
- Suitable for structured data

Case Study

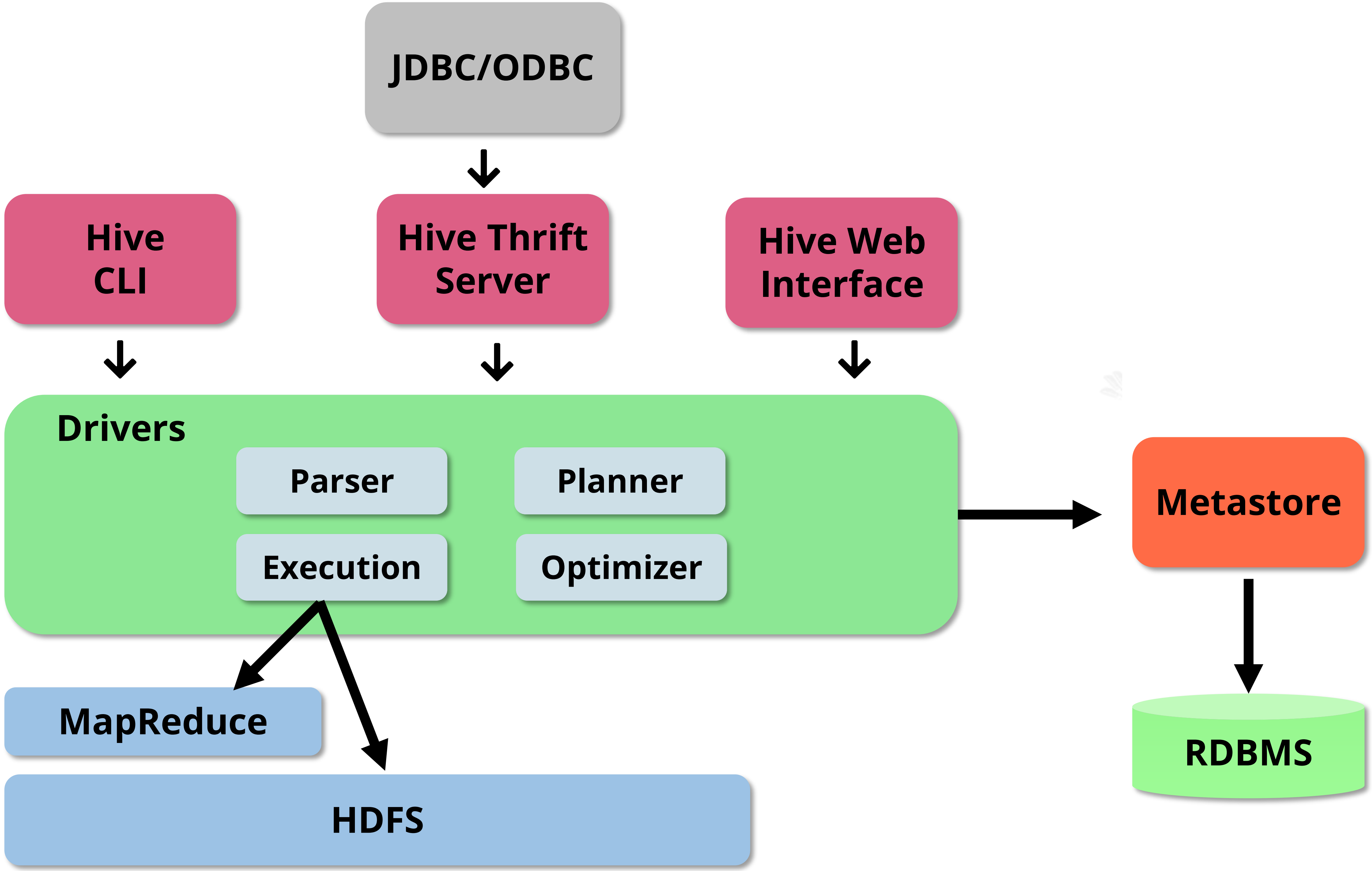
A leading online education company uses Hive and Impala to analyze social media coverage.

The organization analyzes positive, negative, and neutral reviews using Hive.

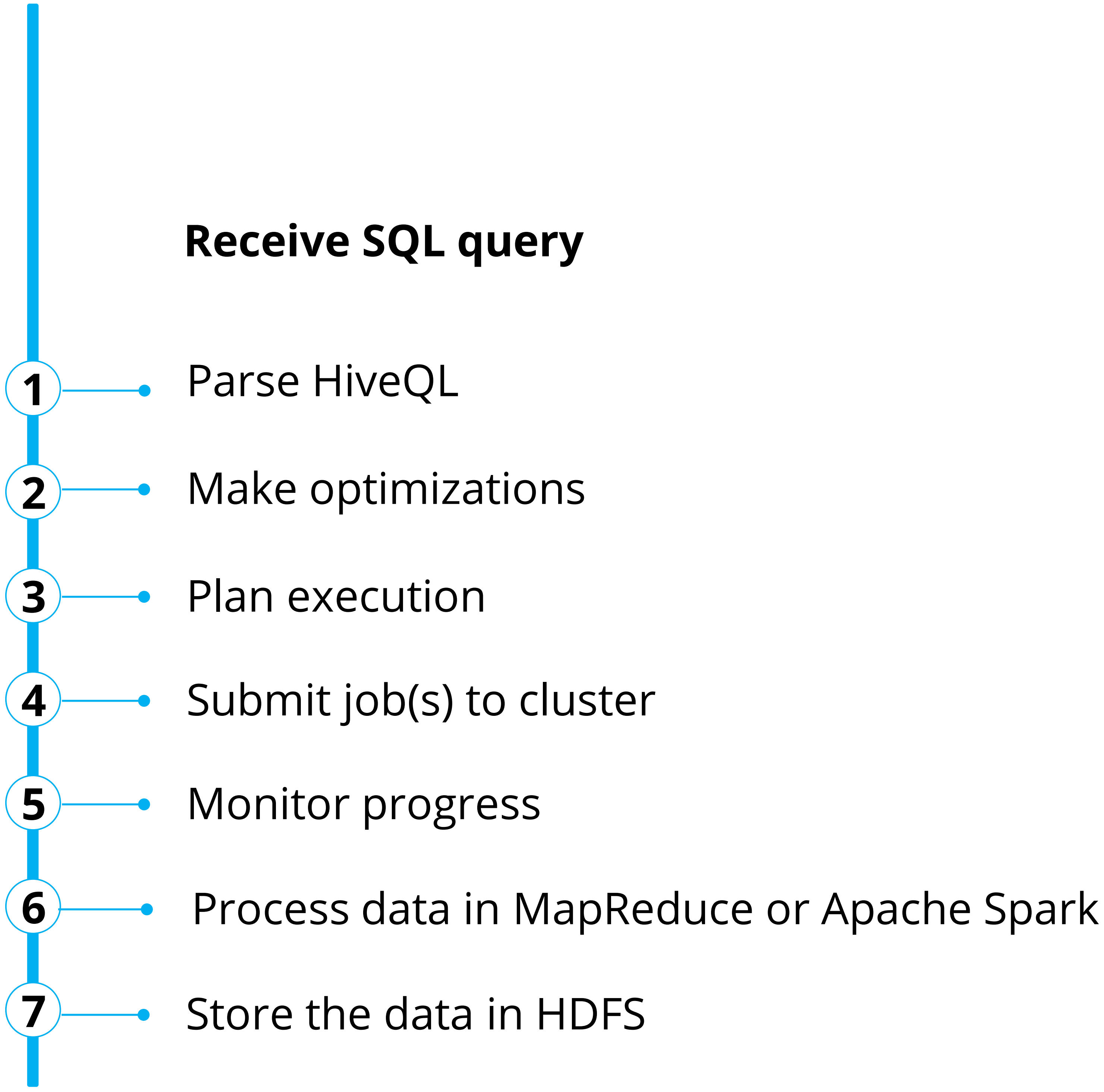
Hive Architecture

Hive Architecture

The major components of Hive architecture are: **Hadoop core components**, **Metastore**, **Driver**, and **Hive clients**.



Job Execution Flow in Hive



Receive SQL query

1 Parse HiveQL

2 Make optimizations

3 Plan execution

4 Submit job(s) to cluster

5 Monitor progress

6 Process data in MapReduce or Apache Spark

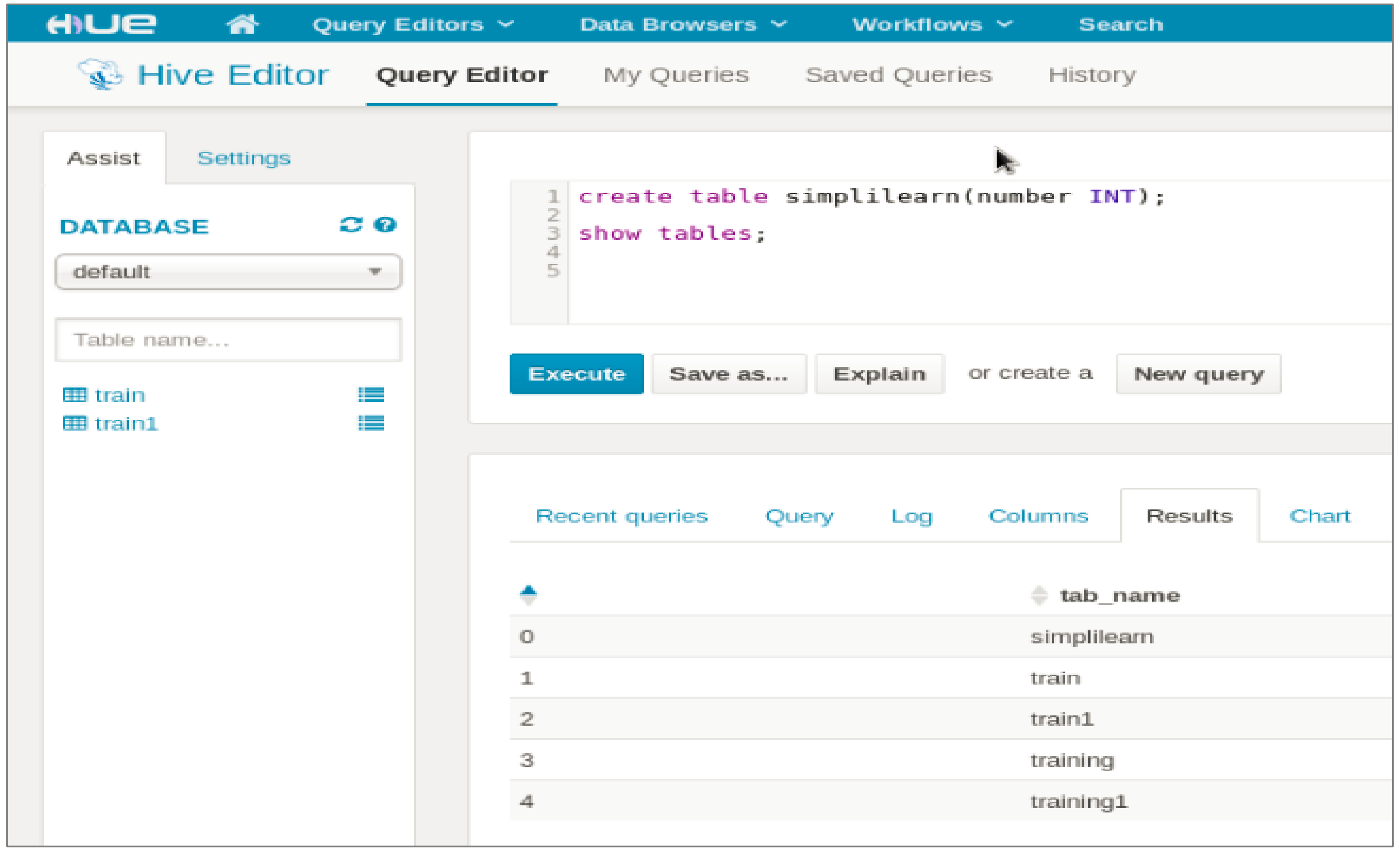
7 Store the data in HDFS

Interfaces to Run Hive Queries

Hive offers many interfaces for running queries.

- **Command-line shell**
  - Hive: Beeline
- **Hue Web UI**
  - Hive Query Editor
- **Metastore Manager**
  - ODBC / JDBC

Hive Query Editor

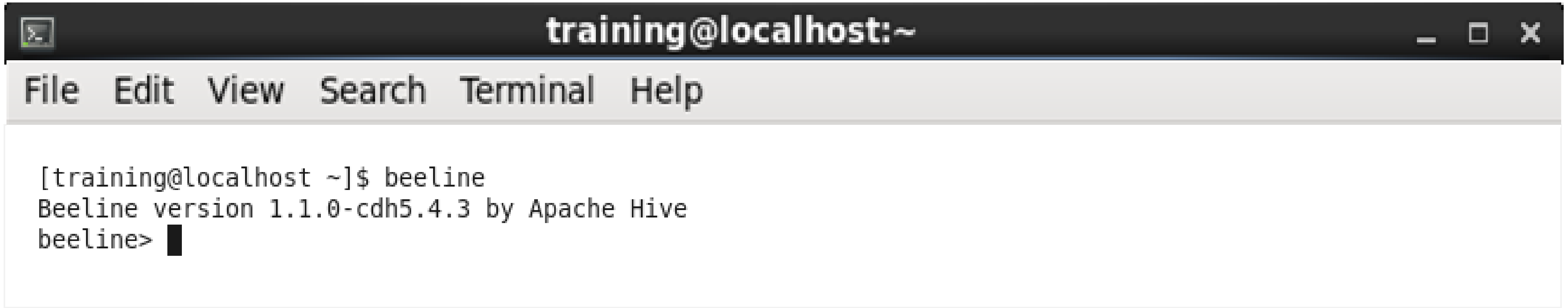


```
create table simplilearn(number INT);

show tables;
```

	tab_name
0	simplilearn
1	train
2	train1
3	training
4	training1

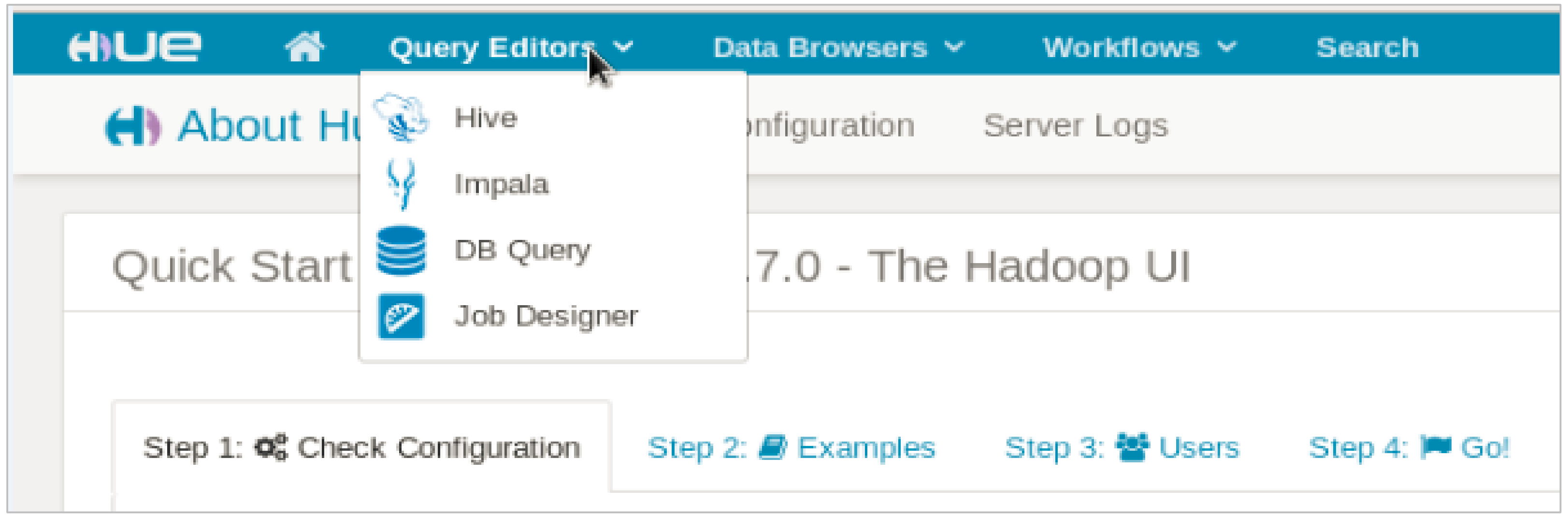
Connecting with Hive

Hive can be run using Beeline



```
[training@localhost ~]$ beeline
Beeline version 1.1.0-cdh5.4.3 by Apache Hive
beeline> 
```

Hue can be used to write a Hive query from a UI

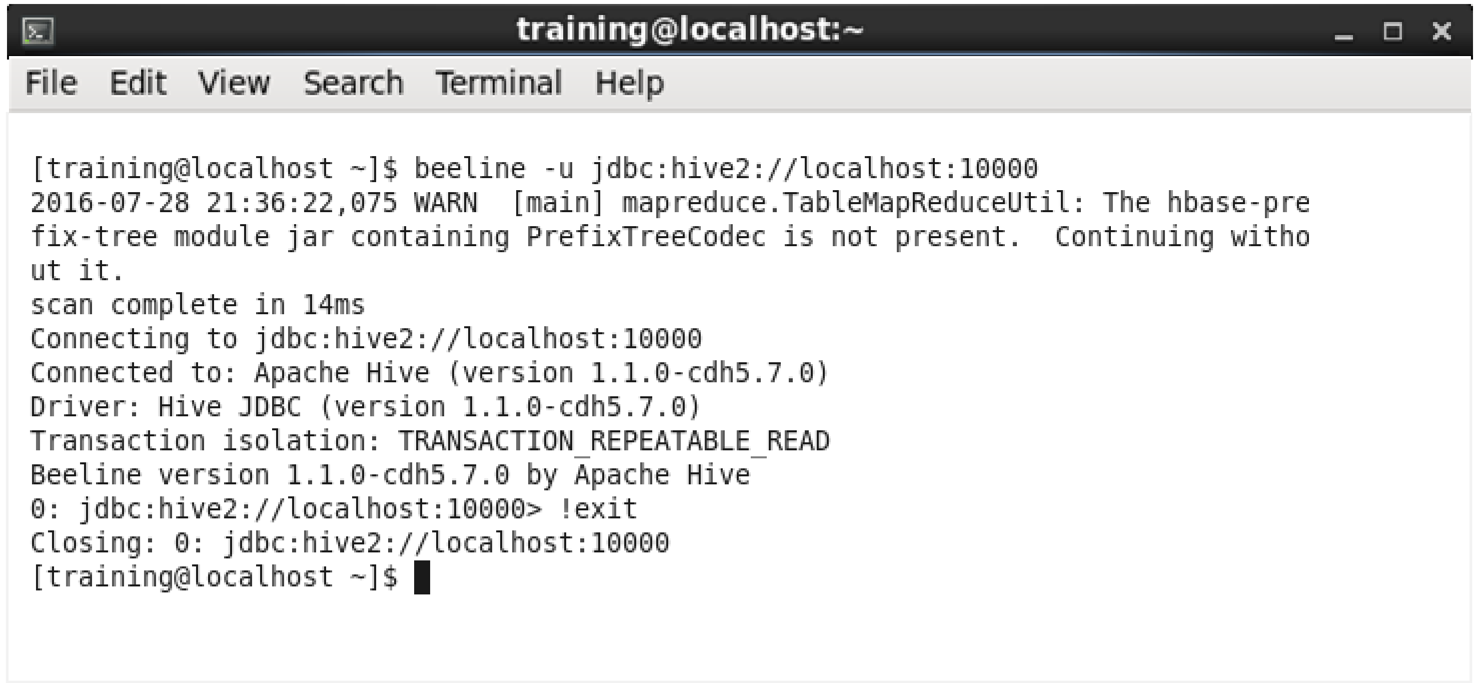


Running Hive Queries Using Beeline

‘!’ is used to execute Beeline commands.

Below are a few commands for running Beeline:

- !exit – to exit the shell
- !help – to show list of all commands
- !verbose – to show added details of queries



```
[training@localhost ~]$ beeline -u jdbc:hive2://localhost:10000
2016-07-28 21:36:22,075 WARN  [main] mapreduce.TableMapReduceUtil: The hbase-pre
fix-tree module jar containing PrefixTreeCodec is not present.  Continuing witho
ut it.
scan complete in 14ms
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 1.1.0-cdh5.7.0)
Driver: Hive JDBC (version 1.1.0-cdh5.7.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 1.1.0-cdh5.7.0 by Apache Hive
0: jdbc:hive2://localhost:10000> !exit
Closing: 0: jdbc:hive2://localhost:10000
[training@localhost ~]$ 
```

Running Beeline from Command Line

Below are the command lines for running Beeline

To execute file using the –u option

```
 beeline –u … -f
simplilearn.hql
```

To use HiveQL directly from the command line using the -e option

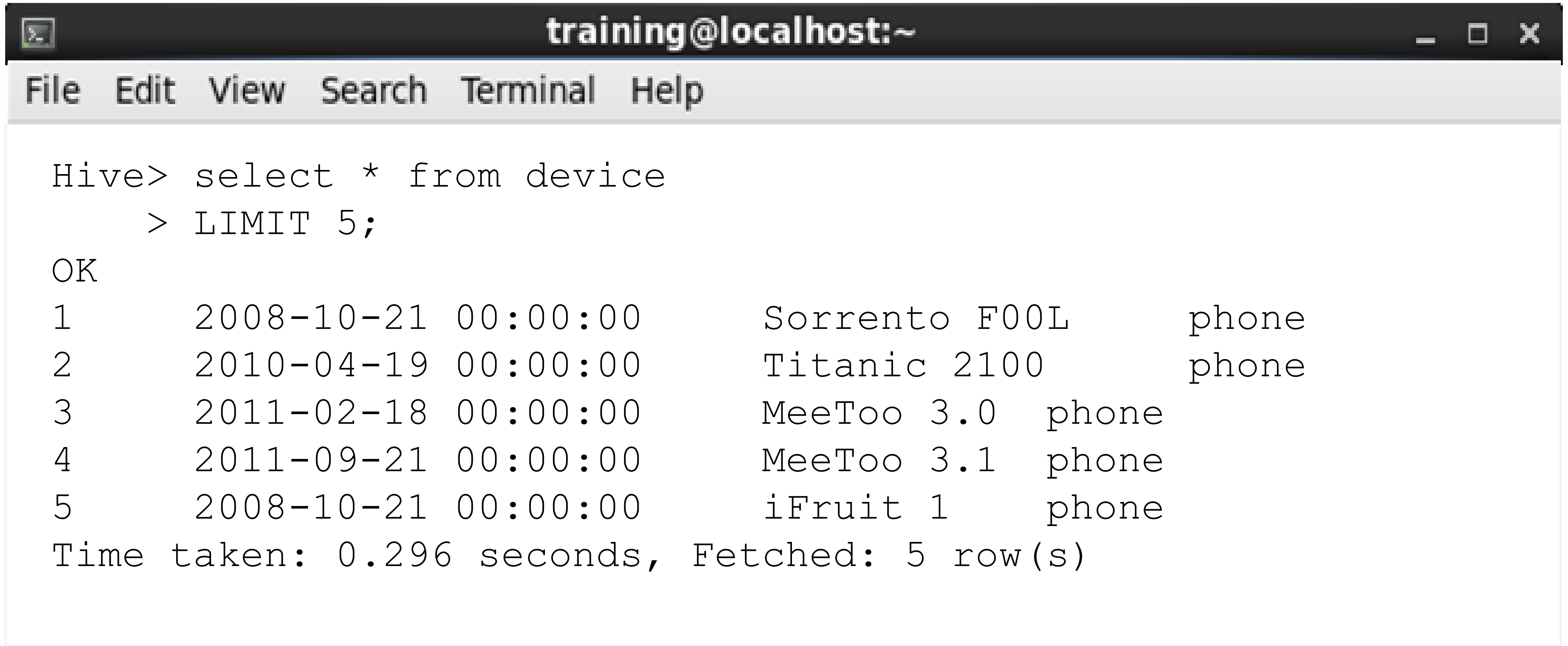
```
beeline –u ... -e
'SELECT * FROM users`
```

To continue running script even after an error

```
beeline –u … -
force=TRUE
```

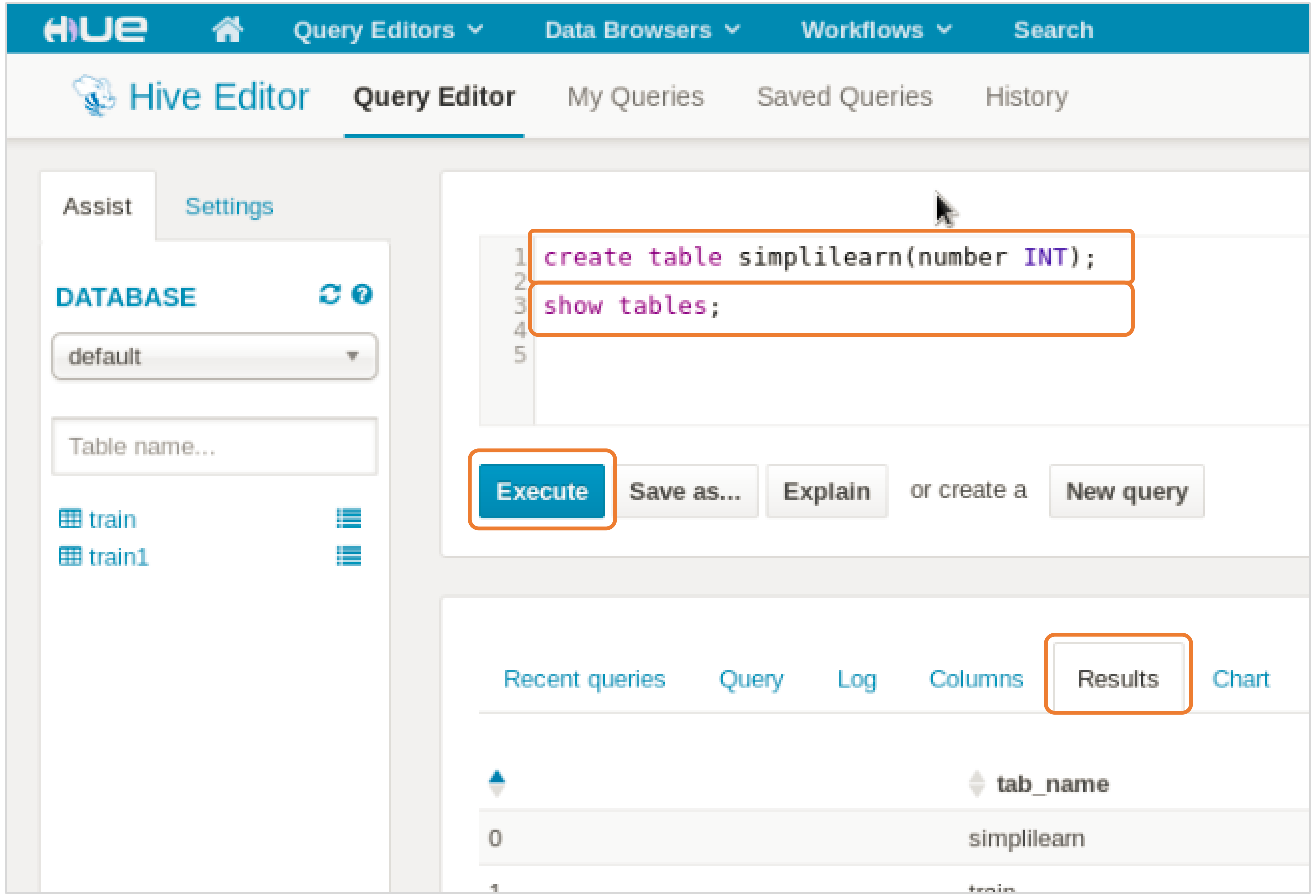
Running Hive query

SQL commands are terminated with a semicolon (;)



```
Hive> select * from device
    > LIMIT 5;
OK
1       2008-10-21 00:00:00     Sorrento F00L     phone
2       2010-04-19 00:00:00     Titanic 2100      phone
3       2011-02-18 00:00:00     MeeToo 3.0  phone
4       2011-09-21 00:00:00     MeeToo 3.1  phone
5       2008-10-21 00:00:00     iFruit 1    phone
Time taken: 0.296 seconds, Fetched: 5 row(s)
```

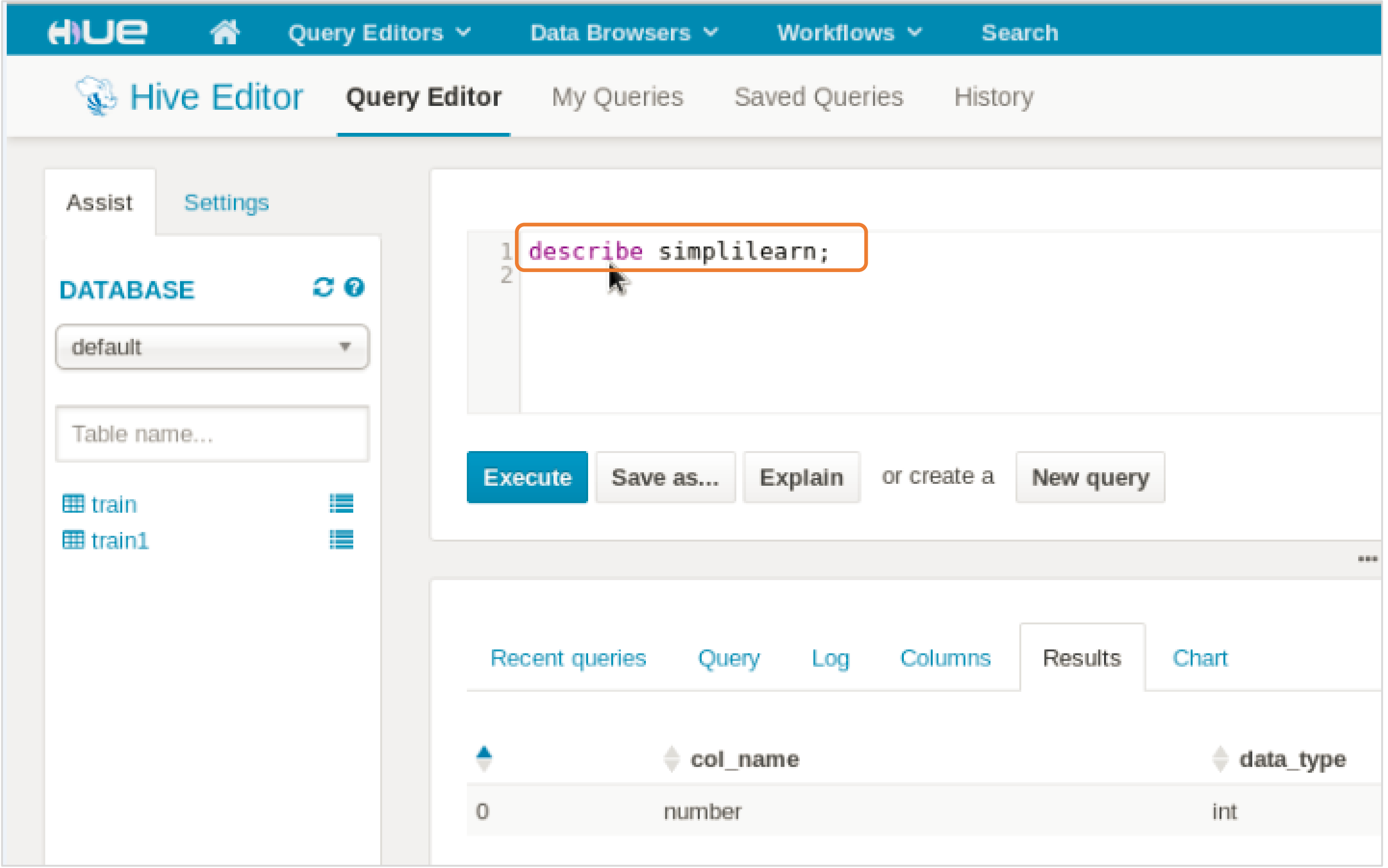
Hive Editors in Hue



```
create table simplilearn(number INT);

show tables;
```

```
        tab_name
0       simplilearn
1       train
```

Diagram 1



```
describe simplilearn;
```

```
        col_name        data_type
0       number          int
```

Diagram 2

Hive Metastore

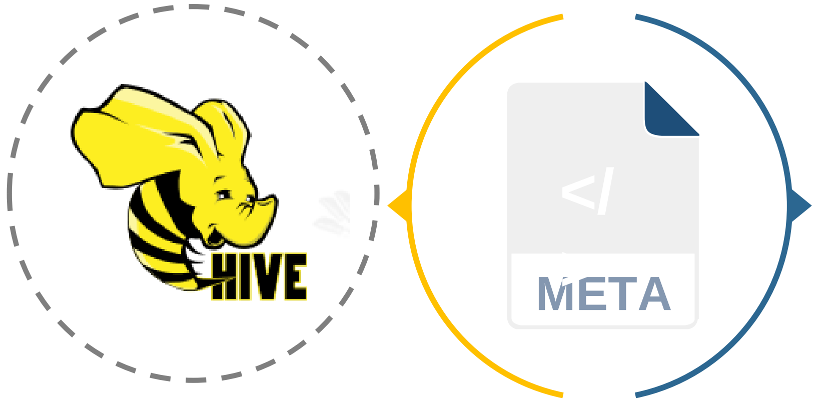
Managing Data with Hive

Hive uses Metastore service to store metadata for Hive tables.

- A table is an HDFS directory containing zero or more files

Default path: **/user/hive/warehouse/<*table_name*>**

- Table supports many formats for data storage and retrieval

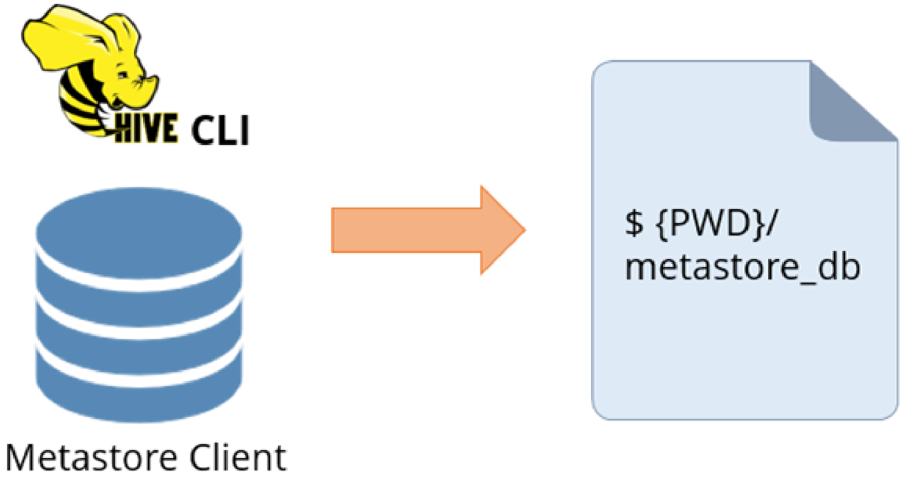


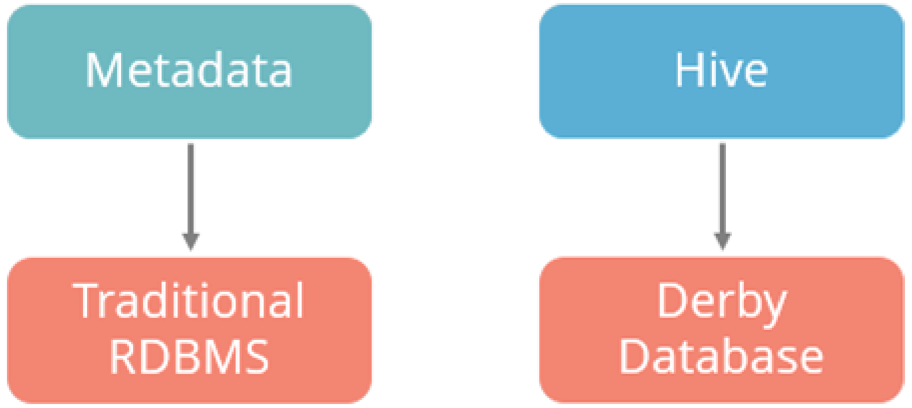
- Metastore stores the created metadata
  - Contained in an RDBMS such as MySQL

- Hive Tables are stored in HDFS and the relevant metadata is stored in the Metastore

What Is Hive Metastore?

The Metastore is the component that stores the system catalog which contains metadata about tables, columns, and partitions.

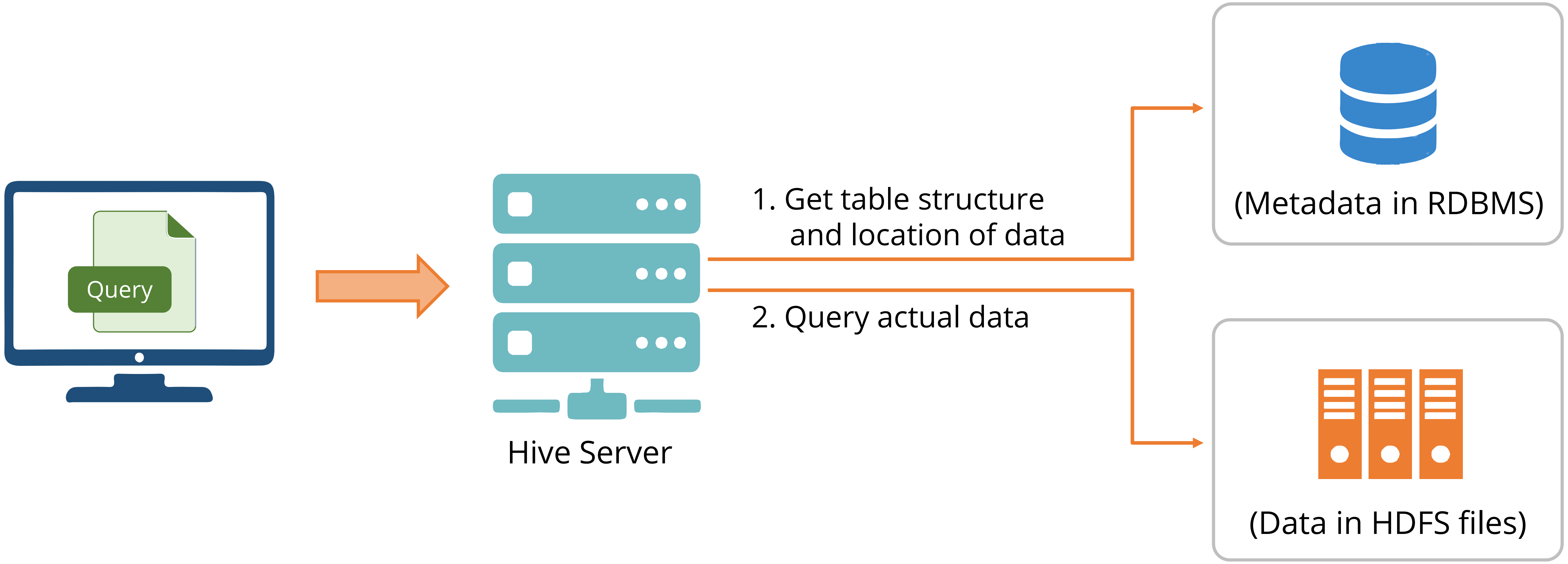




Parameter	Description	Example
Javax.jdo.option.ConnectionURL	JDBC connection URL along with database name containing metadata	jdbc:derby:;databaseName=metastore_db;create=true
Javax.jdo.option.ConnectionDriverName	JDBC driver name. Embedded Derby for Single user mode.	Org.apache.derby.jdbc.EmbeddedDriver
Javax.jdo.option.ConnectionUserName	User name for Derby database	APP
Javax.jdo.option.ConnectionPassword	Password	mine

Use of Metastore in Hive

- Hive uses metastore to get table structure and location of data
- The server queries actual data which is stored in HDFS



Data Warehouse Directory Structure

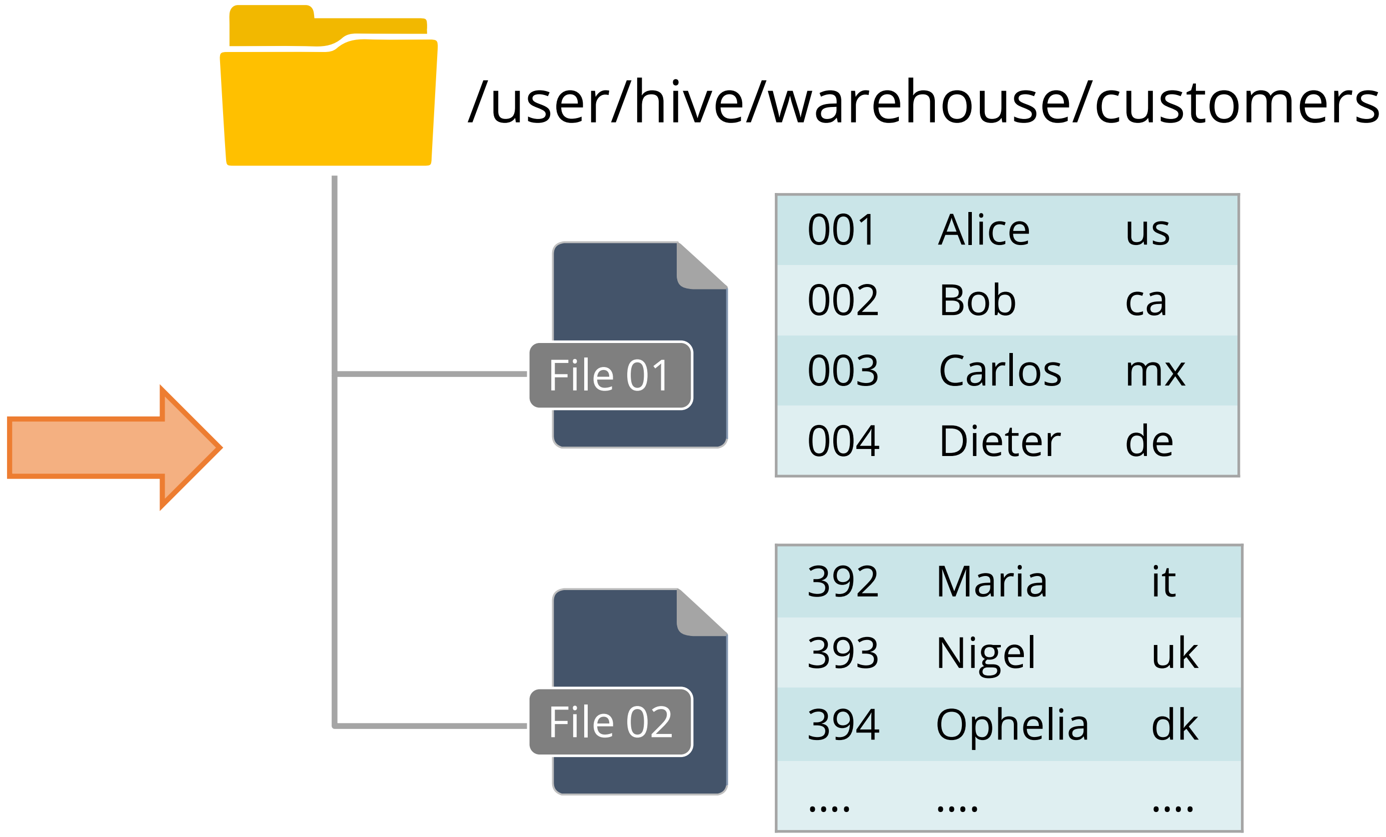
- By default, all data gets stored in

```
/user/hive/warehouse
```

- Each table is a directory within the default location having one or more files

Customers Table

customer_id	name	country
001	Alice	us
002	Bob	ca
003	Carlos	mx
….	….	….
392	Maria	it
393	Nigel	uk
394	Ophelia	dk
….	….	….



001	Alice	us
002	Bob	ca
003	Carlos	mx
004	Dieter	de

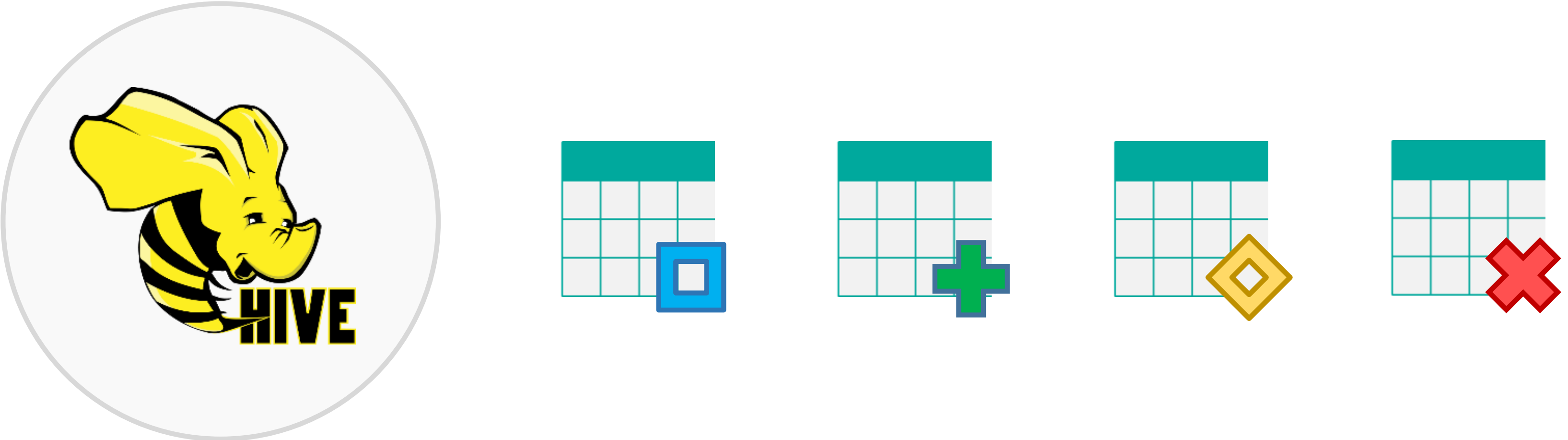
392	Maria	it
393	Nigel	uk
394	Ophelia	dk
….	….	….

In HDFS, Hive data can be split into more than one file.

Hive DDL and DML

Defining Database and Table

- Databases and tables are created and managed using the DDL (Data Definition Language) of HiveQL
- They are very similar to standard SQL DDL
- For example, Create/Drop/Alter/Use Database



Creating a Database

- To create a new database

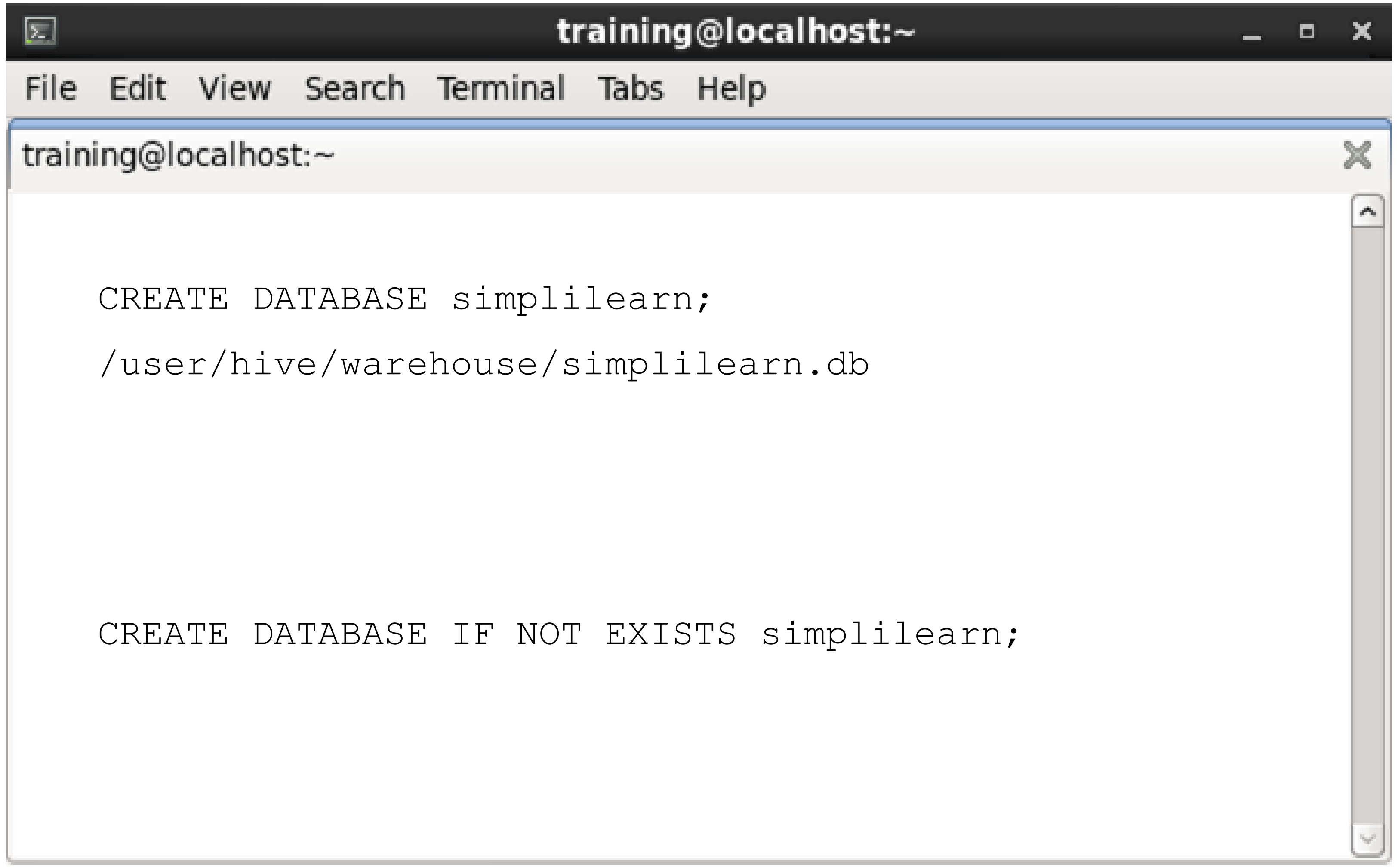
  CREATE DATABASE <dbname>;

  The above statement will add database definition to the metastore and will also create a storage directory in HDFS in the default location.

  For example: /user/hive/warehouse/simplilearn.db

- In order to avoid error in case database simplilearn already exists:

  CREATE DATABASE IF NOT EXISTS <dbname>;



```
CREATE DATABASE simplilearn;
/user/hive/warehouse/simplilearn.db



CREATE DATABASE IF NOT EXISTS simplilearn;
```

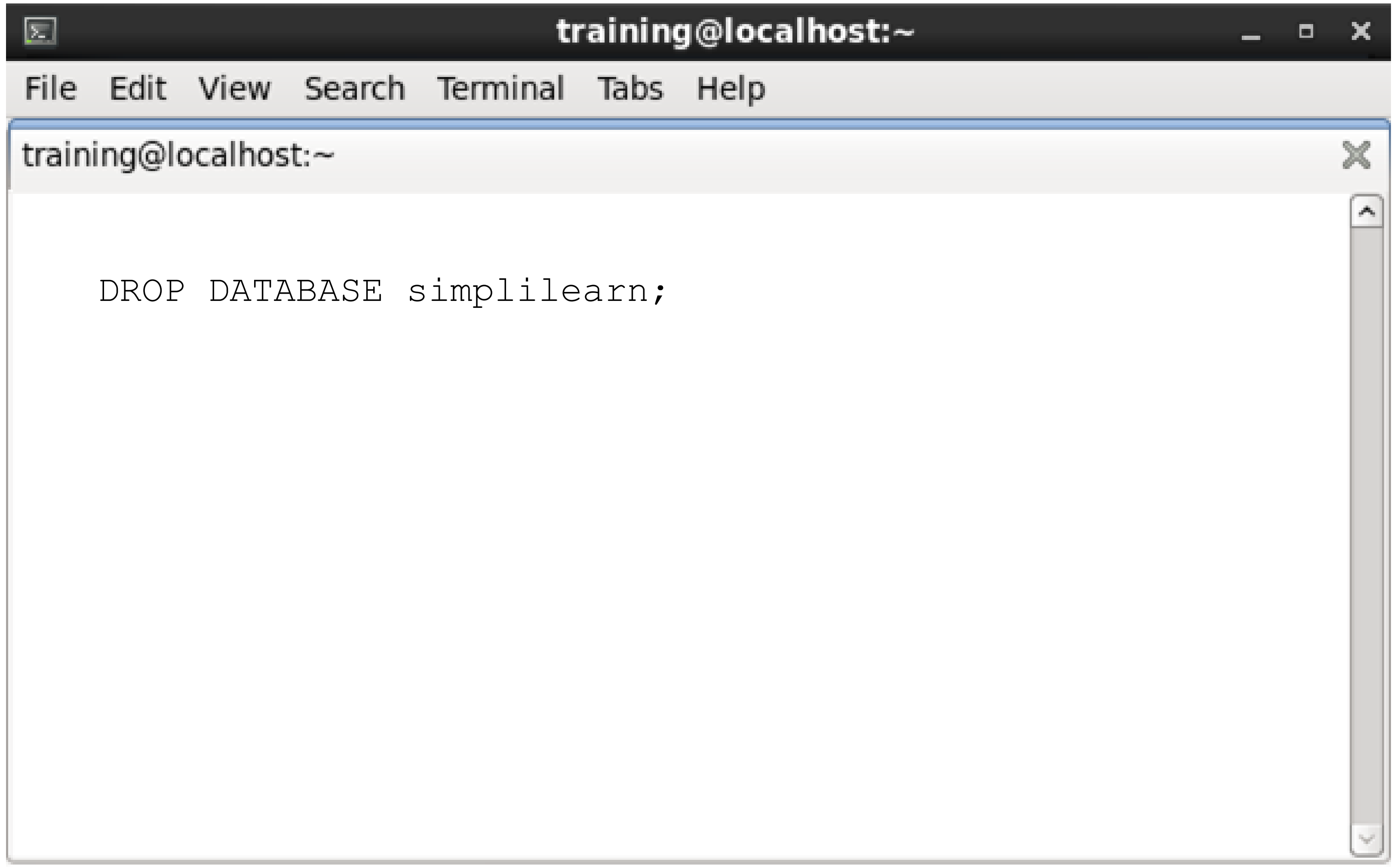
Deleting a Database

- Removing a database is similar to creating it
  - replace CREATE with DROP

    DROP DATABASE <dbname>;

    - In case the database already exists, you can DROP DATABASE IF EXISTS <dbname>;
- In order to remove database, if it has some table :

  DROP DATABASE <dbname> CASCADE;

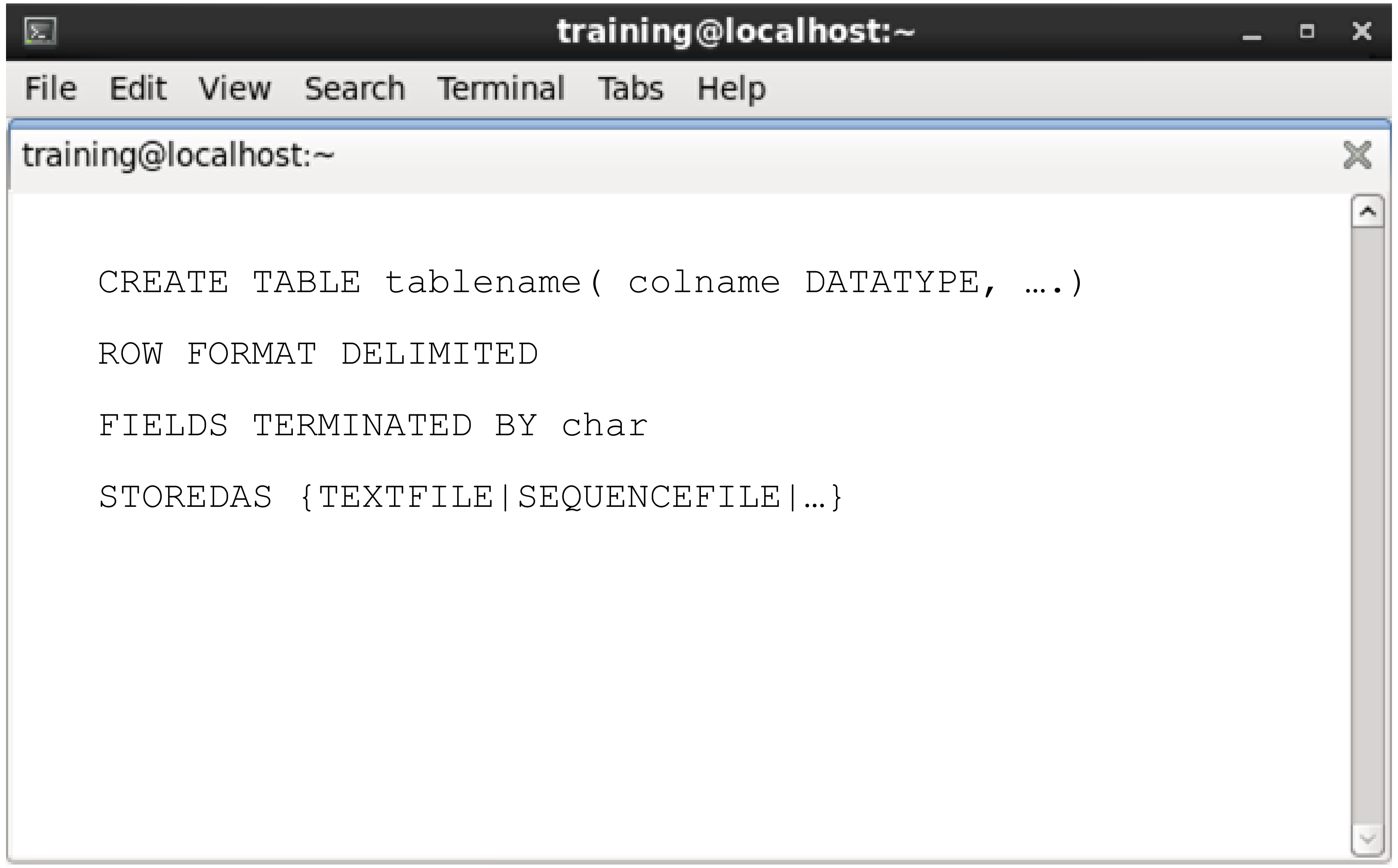


```
DROP DATABASE simplilearn;
```

This might remove data in HDFS.

Creating New Table

Syntax to create a new table



```
CREATE TABLE tablename( colname DATATYPE, ….)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY char
STOREDAS {TEXTFILE|SEQUENCEFILE|…}
```

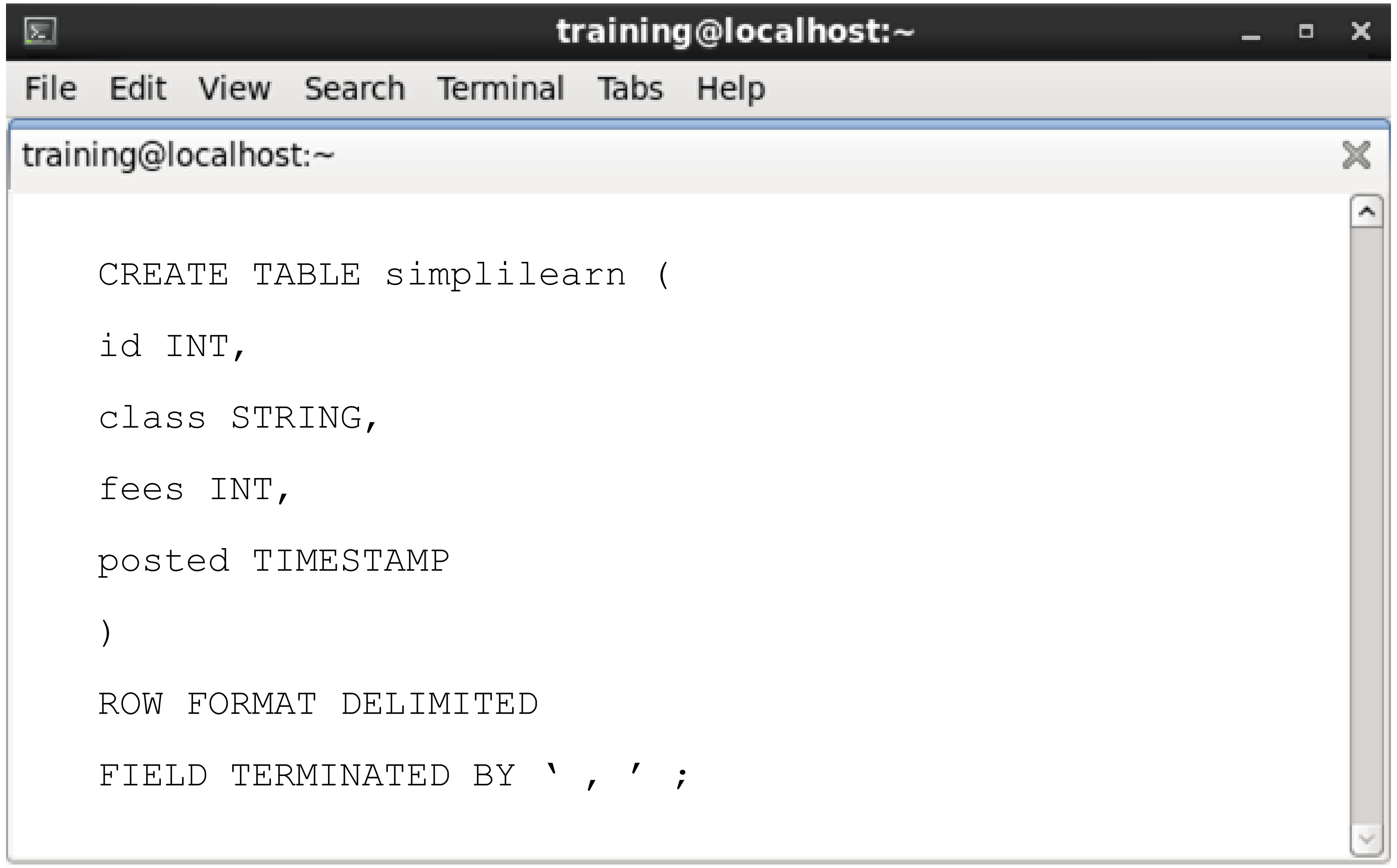
- Syntax creates a subdirectory in the database's warehouse directory in HDFS
  - Default database

    /user/hive/warehouse/tablename
  - Named database

    /user/hive/warehouse/dbname.db/tablename

Table Creation: Example

- The following example shows how to create a new table named simplilearn
  - Data is stored as text with four comma-separated fields per line

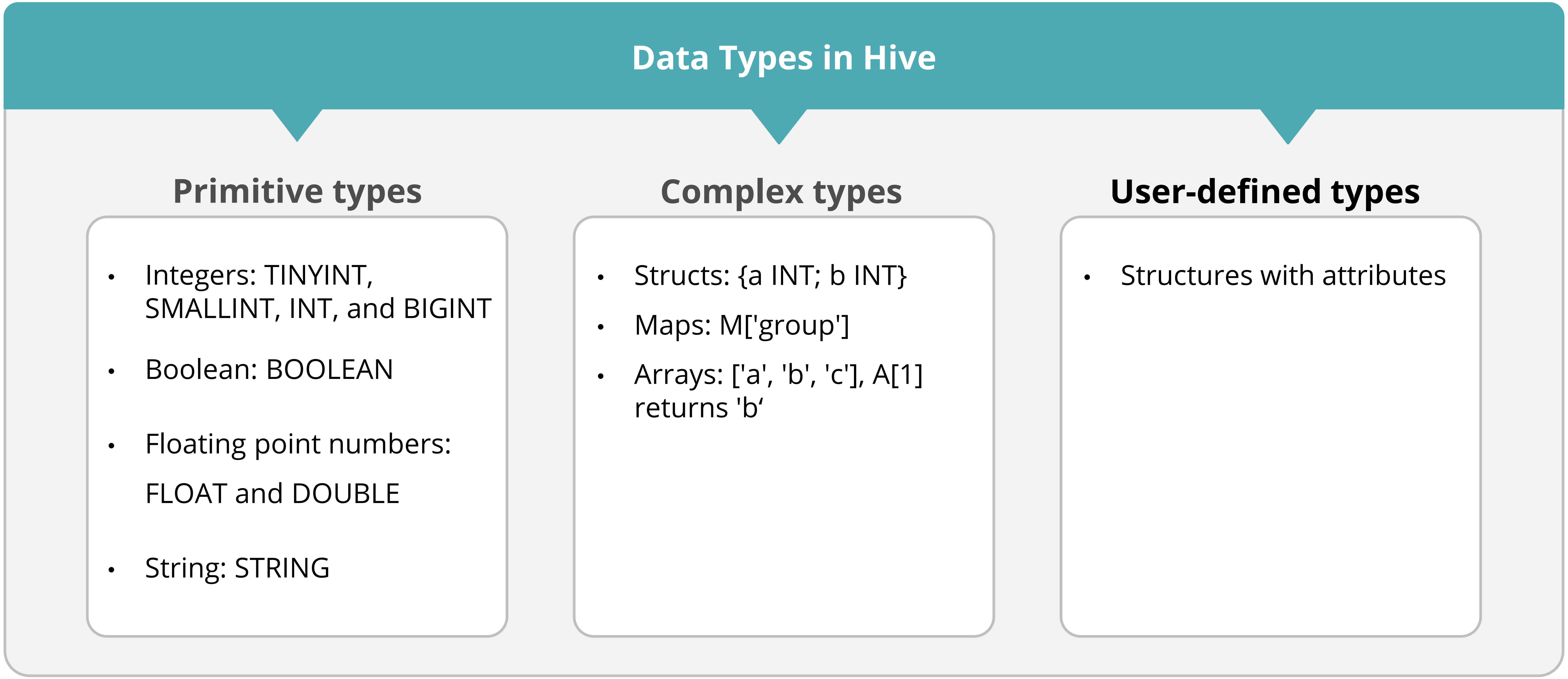


```
CREATE TABLE simplilearn (
id INT,
class STRING,
fees INT,
posted TIMESTAMP
)
ROW FORMAT DELIMITED
FIELD TERMINATED BY ‵ , ′ ;
```

Data Types

Data Types in Hive

The data types in Hive are as follows:



Data Types in Hive

Primitive types

- Integers: TINYINT, SMALLINT, INT, and BIGINT
- Boolean: BOOLEAN
- Floating point numbers: FLOAT and DOUBLE
- String: STRING

Complex types

- Structs: {a INT; b INT}
- Maps: M['group']
- Arrays: ['a', 'b', 'c'], A[1] returns 'b‘

User-defined types

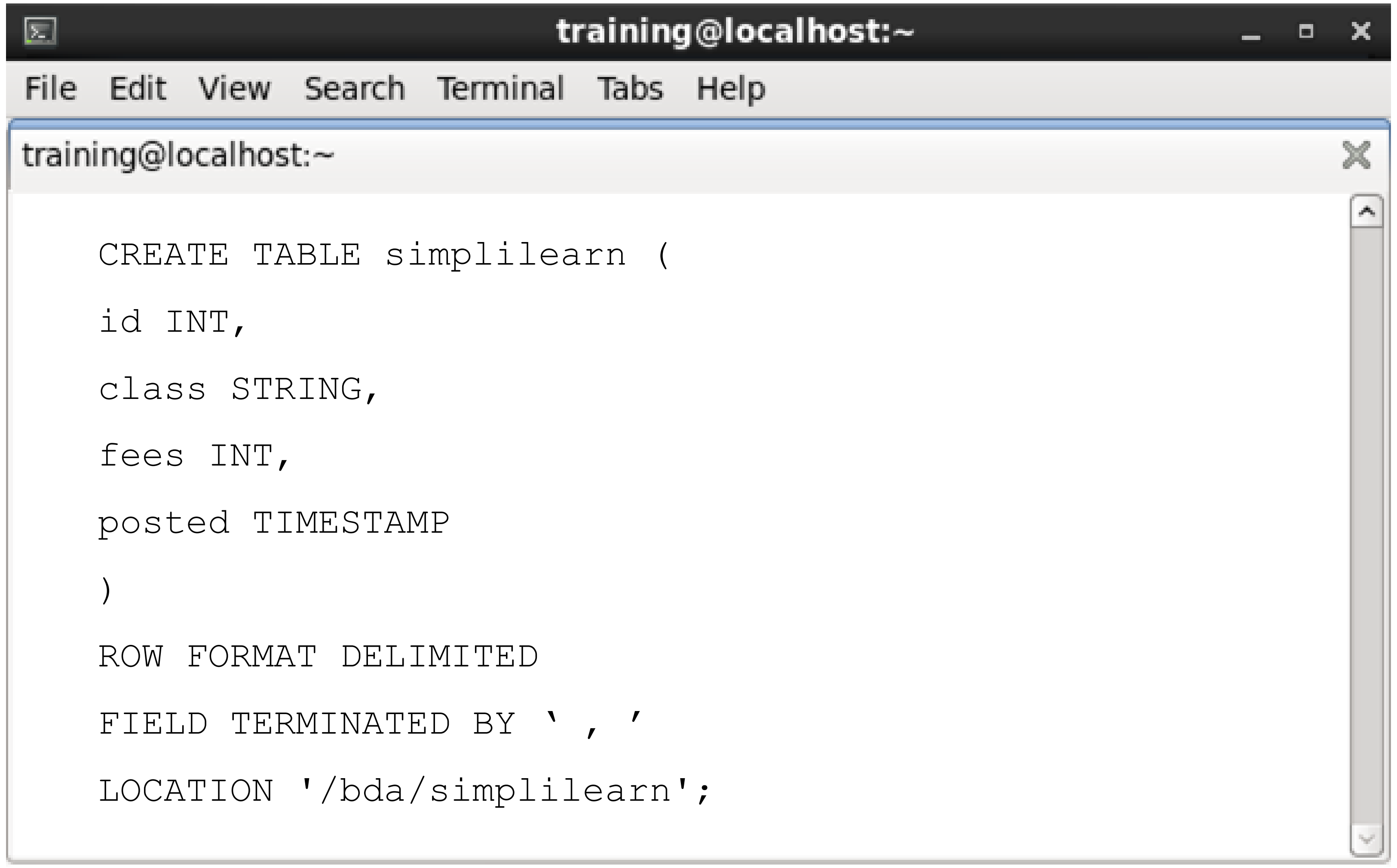
- Structures with attributes

Changing Table Data Location

- By default, table data is stored in the default warehouse location

  user/hive/warehouse

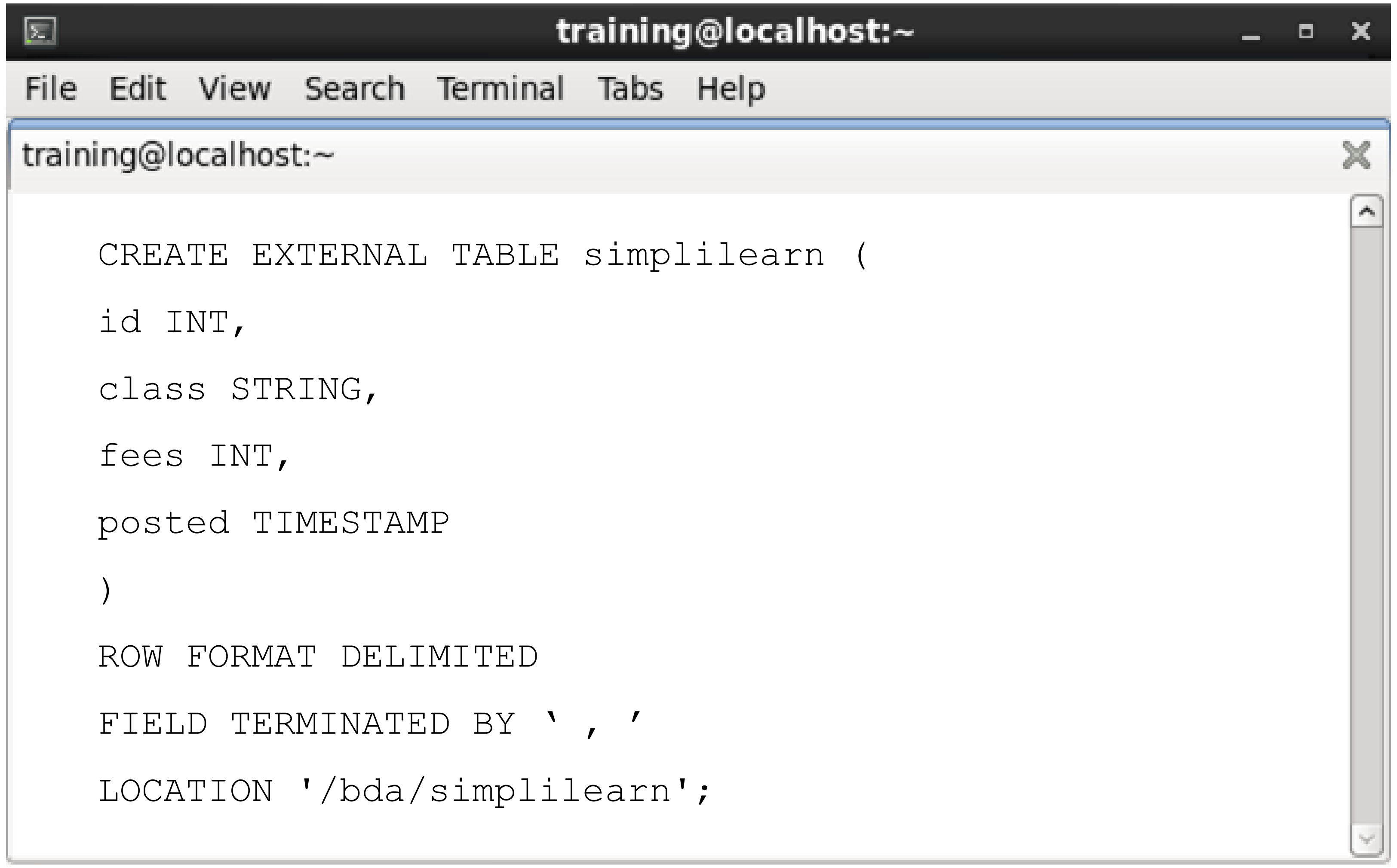
- Use LOCATION to specify the directory where you want to reside your data in HDFS



```
CREATE TABLE simplilearn (
id INT,
class STRING,
fees INT,
posted TIMESTAMP
)
ROW FORMAT DELIMITED
FIELD TERMINATED BY ‵ , ′
LOCATION '/bda/simplilearn';
```

External Managed Table

- Tables are “managed” or “internal” by default. When a table is removed, the data also gets deleted.
- Use EXTERNAL to create an external managed table
- Dropping an external table removes only its metadata

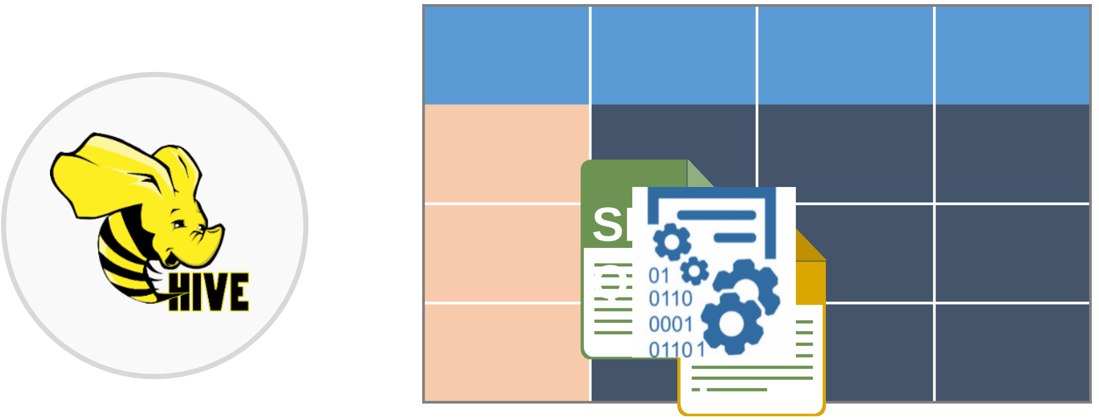


```
CREATE EXTERNAL TABLE simplilearn (
id INT,
class STRING,
fees INT,
posted TIMESTAMP
)
ROW FORMAT DELIMITED
FIELD TERMINATED BY ‘ , ’
LOCATION '/bda/simplilearn';
```

Validation of Data

Hive follows “schema on read”

- Unlike RDBMS, Hive does not validate data on insert
- Files are simply moved into place, which makes loading data into tables faster in Hive
- Errors in file formats are discovered when queries are performed



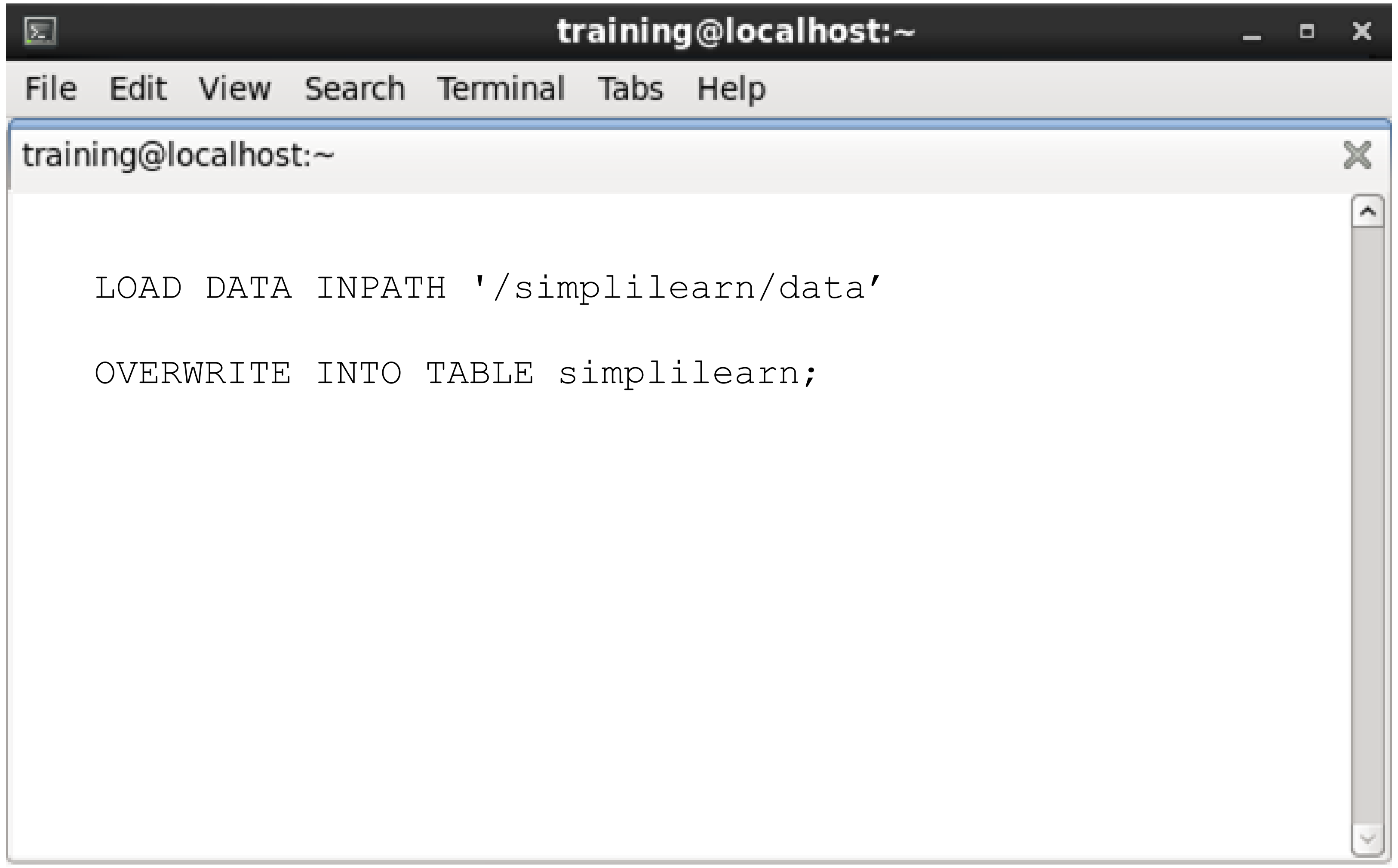
Missing data is represented as NULL.

Loading of Data

- Data can be moved from the HDFS file directly to Hive table

  hdfs dfs -mv /simplilearn/data /user/hive/warehouse/simplilearn/

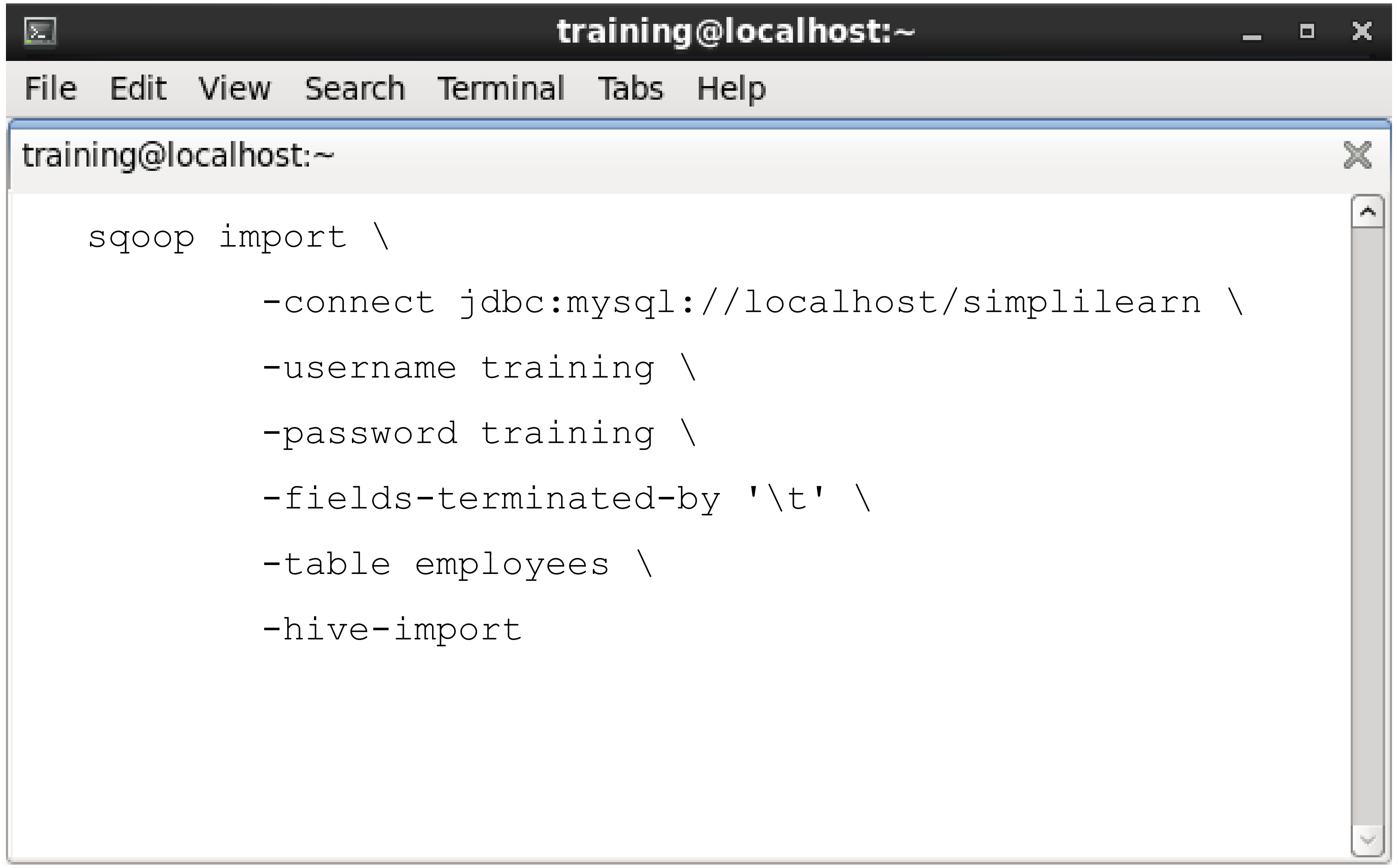
- Data can be loaded using the following query:



```
LOAD DATA INPATH '/simplilearn/data'

OVERWRITE INTO TABLE simplilearn;
```

Loading Data from RDBMS

- Sqoop provides support for importing data into Hive
- Using hive-import option in Sqoop, you can:
  - create a table in Hive metastore
  - import data from the RDBMS to the table’s directory in HDFS

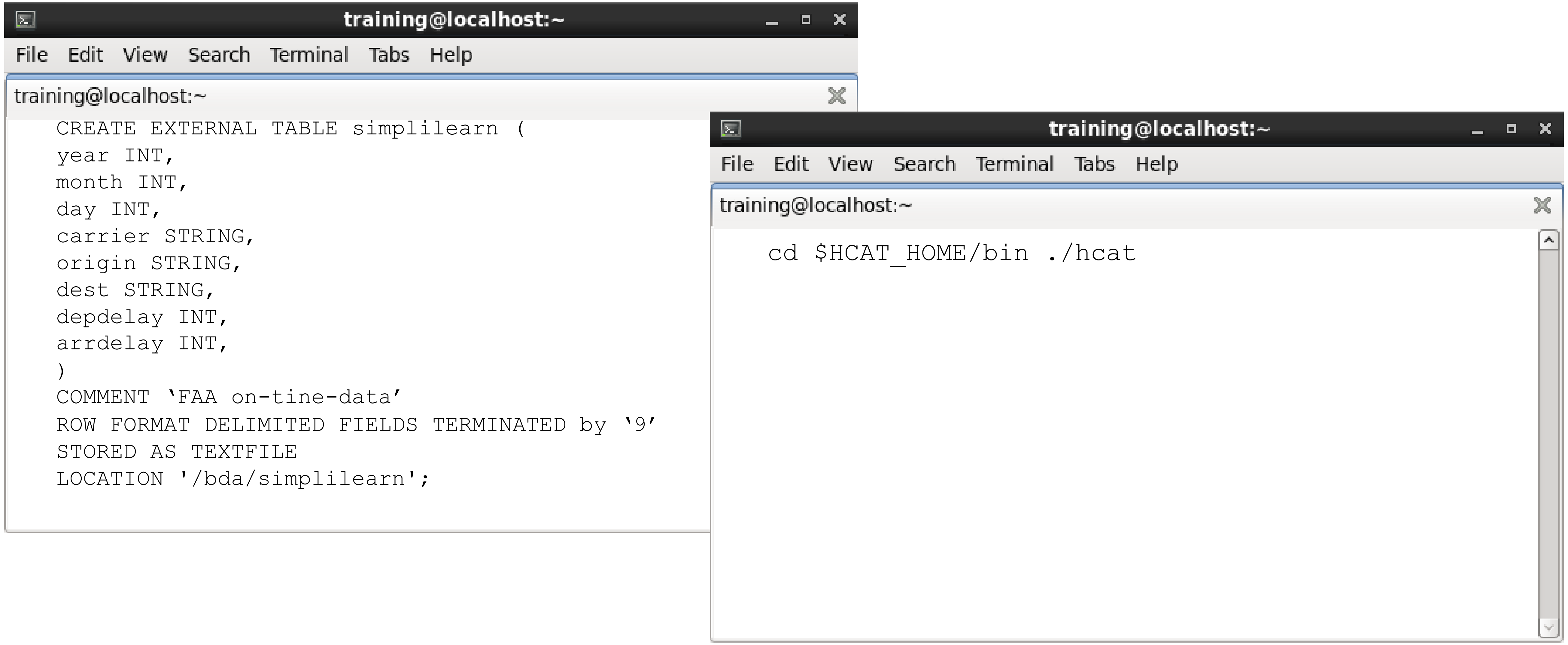


```
sqoop import \
        -connect jdbc:mysql://localhost/simplilearn \
        -username training \
        -password training \
        -fields-terminated-by '\t' \
        -table employees \
        -hive-import
```

hive-import creates a table accessible in Hive.

What Is HCatalog

- HCatalog is a Hive sub-project that provides access to the Metastore
- It allows to define tables using HiveQL DDL syntax
- It is accessible through command line and REST API
- It accesses tables created through HCatalog from Hive, MapReduce, Pig, and other tools


training@localhost:~
File Edit View Search Terminal Tabs Help
training@localhost:~
training@localhost:~
File Edit View Search Terminal Tabs Help
training@localhost:~

```
CREATE EXTERNAL TABLE simplilearn (
year INT,
month INT,
day INT,
carrier STRING,
origin STRING,
dest STRING,
depdelay INT,
arrdelay INT,
)
COMMENT ‘FAA on-tine-data’
ROW FORMAT DELIMITED FIELDS TERMINATED by ‘9’
STORED AS TEXTFILE
LOCATION '/bda/simplilearn';
```

```
cd $HCAT_HOME/bin ./hcat
```

Assisted Practice

Apache Hive

Duration: 15 mins

Problem Statement: In this demonstration, you will learn how to use Hive query editor for real-time analysis and data filtrations.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Assisted Practice

Apache Hive

Duration: 15 mins

Problem Statement: In this demonstration, you will learn how to use the Hive editor in web console.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Assisted Practice

Apache Hive

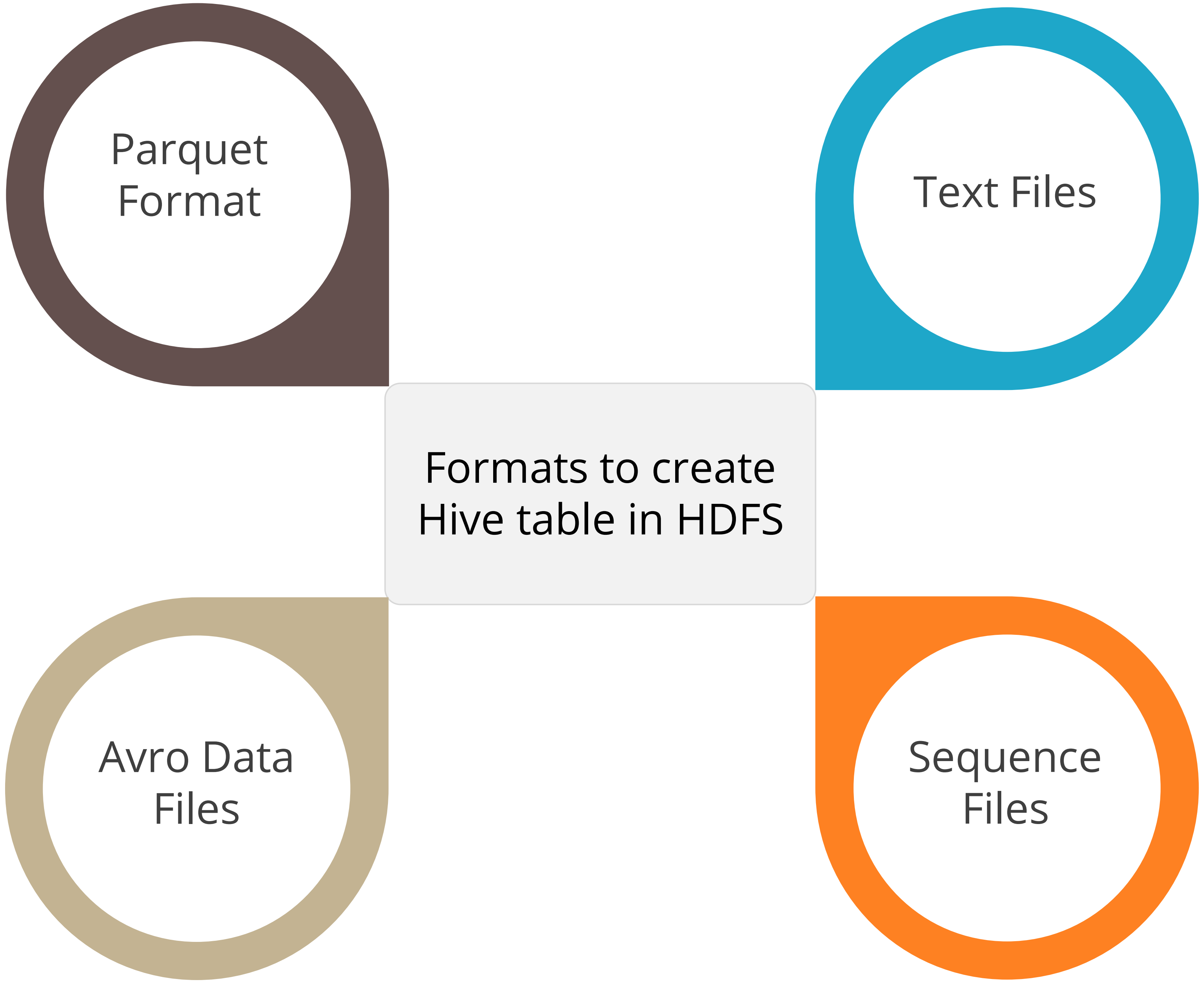
Duration: 15 mins

Problem Statement: In this demonstration, you will learn how to use Hive to import data from an external source and perform data representation and analysis.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

File Format Types

File Format Types



Formats to create Hive table in HDFS

Parquet Format

Text Files

Avro Data Files

Sequence Files

File Format: Text File Format

The most basic and human-readable file

Read or written in any programming language

Delimited by a comma or a tab

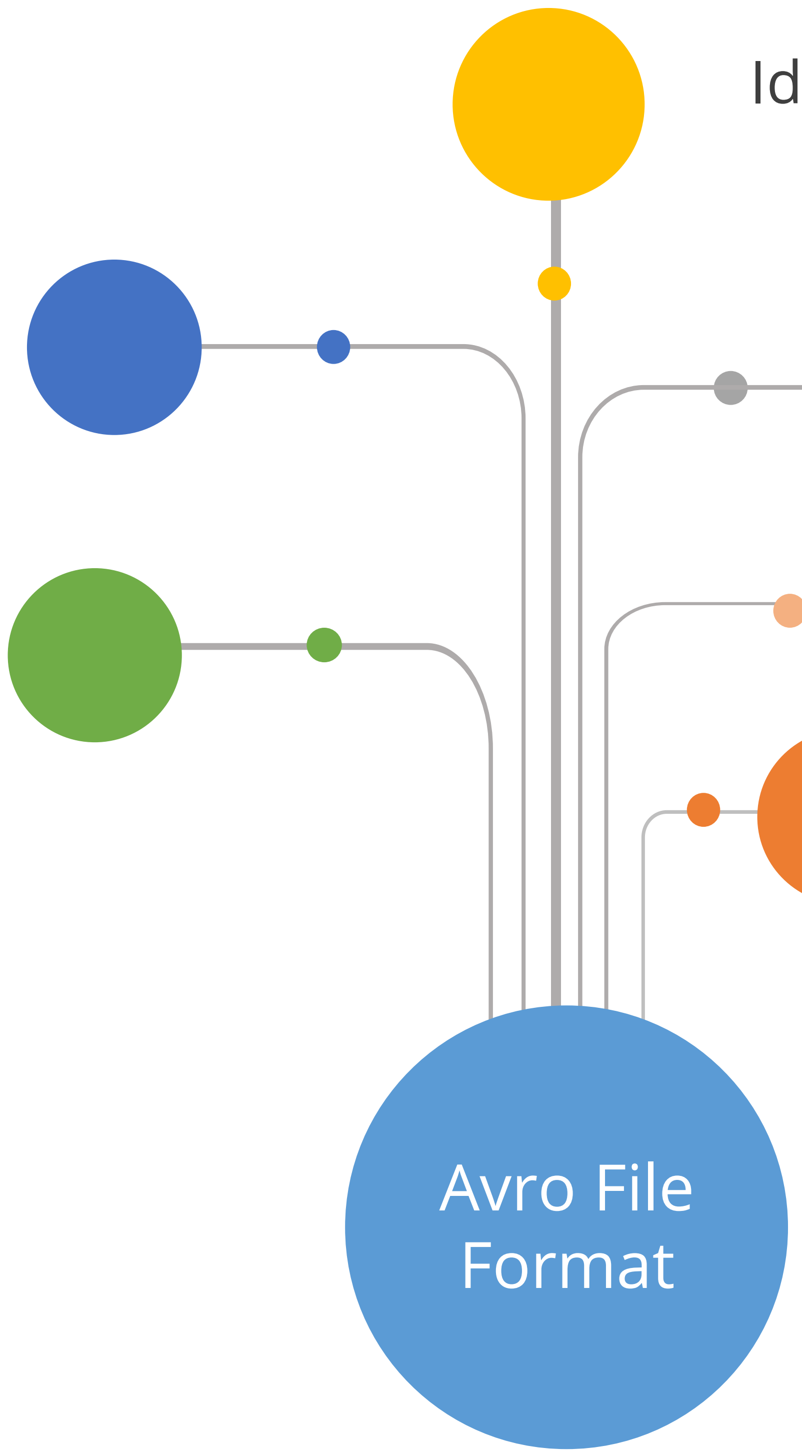
Consumes more space to store numeric value as string

Difficult to represent binary data

File Format: Sequence File Format

- Stores key-value pairs in a binary container format
- More efficient than a text file
- Not human-readable

File Format: Avro File Format



Ideal for long-term storage of data

Widely supported inside and outside Hadoop ecosystem

Can read from and write in many languages

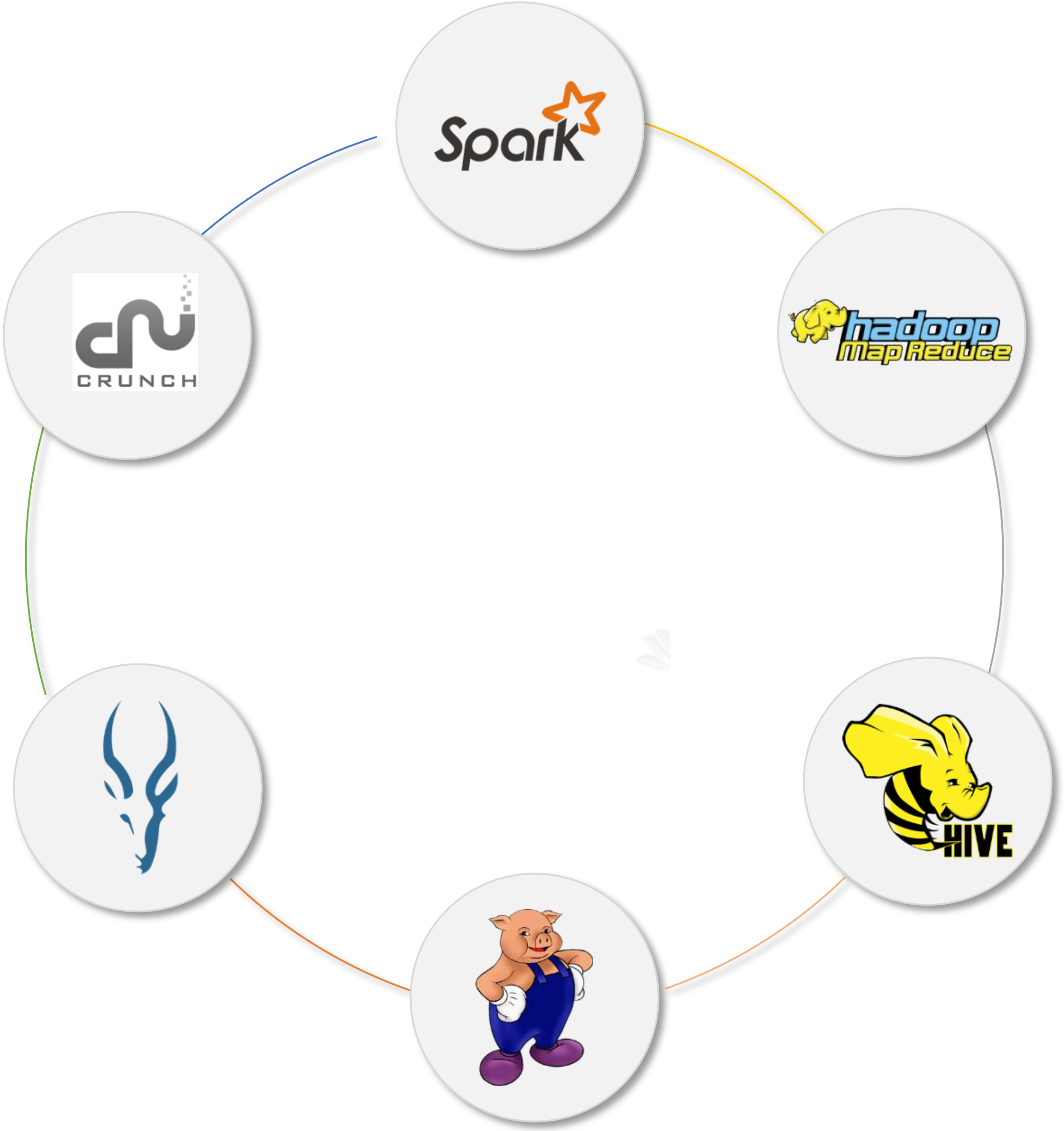
Efficient storage

Embeds schema metadata in the file

Considered the best choice for general-purpose storage in Hadoop

File Format: Parquet File Format

- Is a columnar format developed by Cloudera and Twitter
- Uses advanced optimizations described in Google’s Dremel paper
- Considered the most efficient for adding multiple records at a time



Data Serialization

What Is Data Serialization?

Data serialization is a way to represent data in the storage memory as a series of bytes.

How do you serialize the number 123456789?

It can be serialized as 4 bytes when stored as a Java int and 9 bytes when stored as a Java String.

Data Serialization Framework

- Efficient data serialization framework
- Widely supported throughout Hadoop and its ecosystem
- Supports Remote Procedure Calls (RPC)
- Offers compatibility

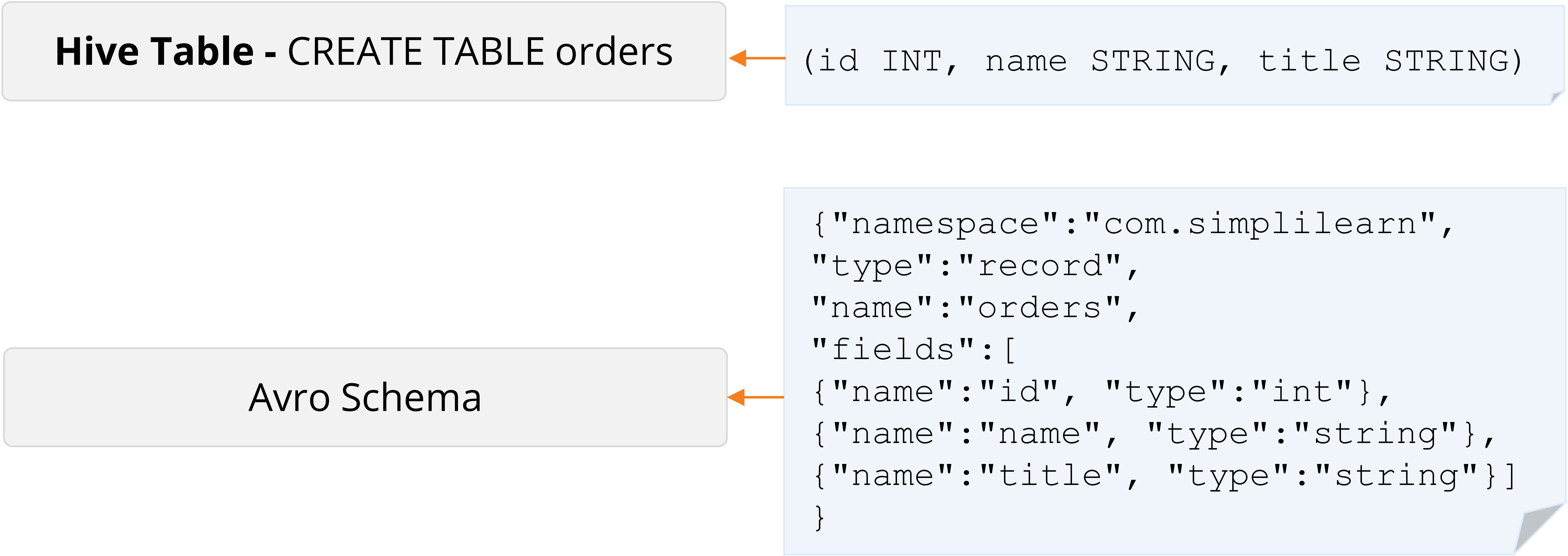
Data Types Supported in Avro

Name	Description
null	An absence of a value
boolean	A binary
int	32-bit signed integer
long	64-bit signed integer
float	Single-precision floating point value
double	Double-precision floating point value
bytes	Sequence of 8-bit unsigned bytes
string	Sequence of unicode characters

Complex Data Types Supported in Avro Schemas

Name	Description
record	A user-defined type composed of one or more named fields
enum	A specified set of values
array	Zero or more values of the same type
map	Set of key-value pairs; key is string while value is of specified type
union	Exactly one value matching a specified set of types
fixed	A fixed number of 8-bit unsigned bytes

Hive Table and Avro Schema



Hive Table - CREATE TABLE orders

```
(id INT, name STRING, title STRING)
```

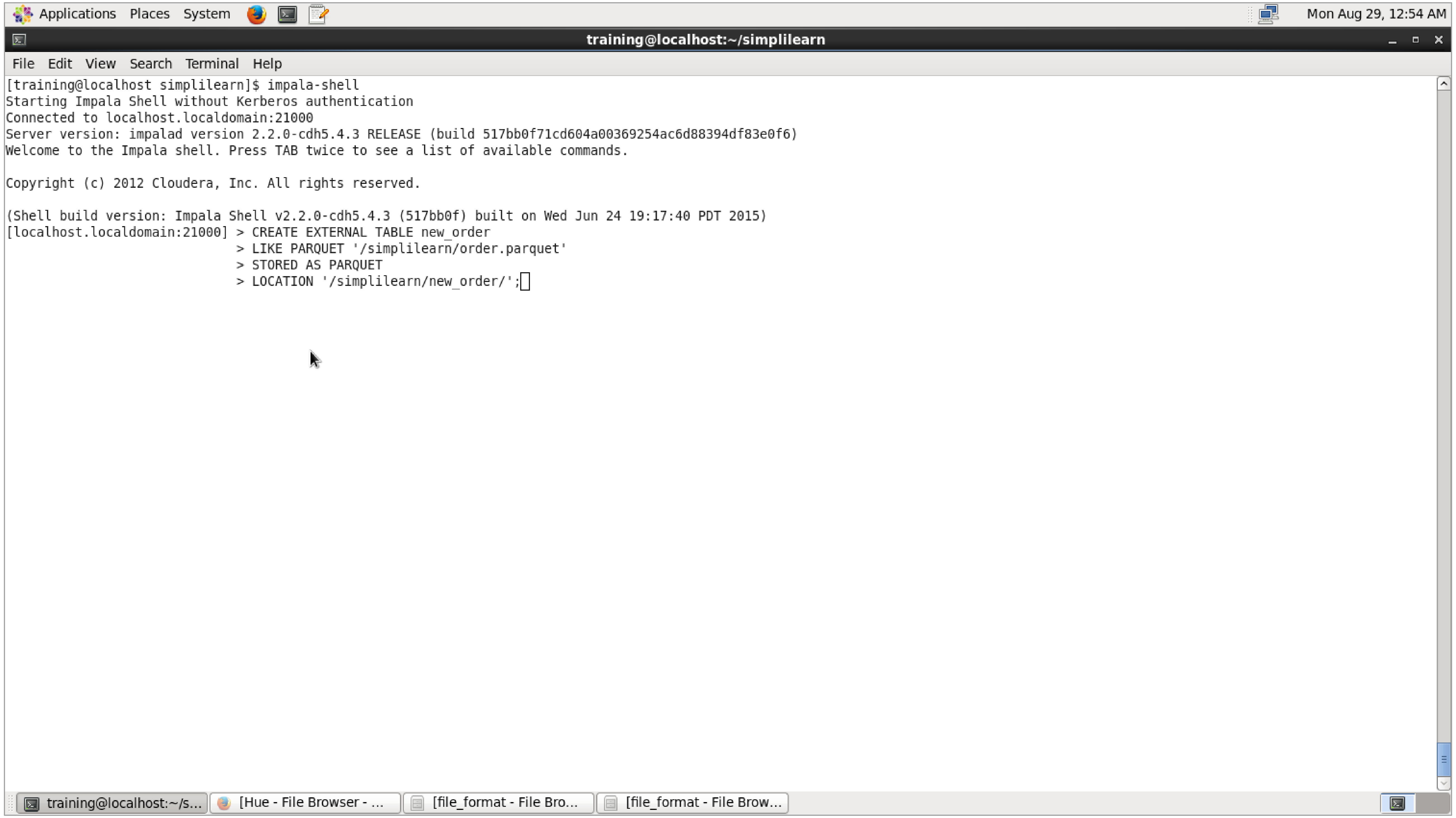
Avro Schema

```
{"namespace":"com.simplilearn",
"type":"record",
"name":"orders",
"fields":[
{"name":"id", "type":"int"},
{"name":"name", "type":"string"},
{"name":"title", "type":"string"}]
}
```

Other Avro Operations

```
{"namespace":"com.simplilearn",
"type":"record",
"name":"orders",
"fields":[
{"name":"id", "type":"int"},
{"name":"name", "type":"string", "default":"simplilearn"},
{"name":"title", "type":"string","default":"bigdata"}]
}
```

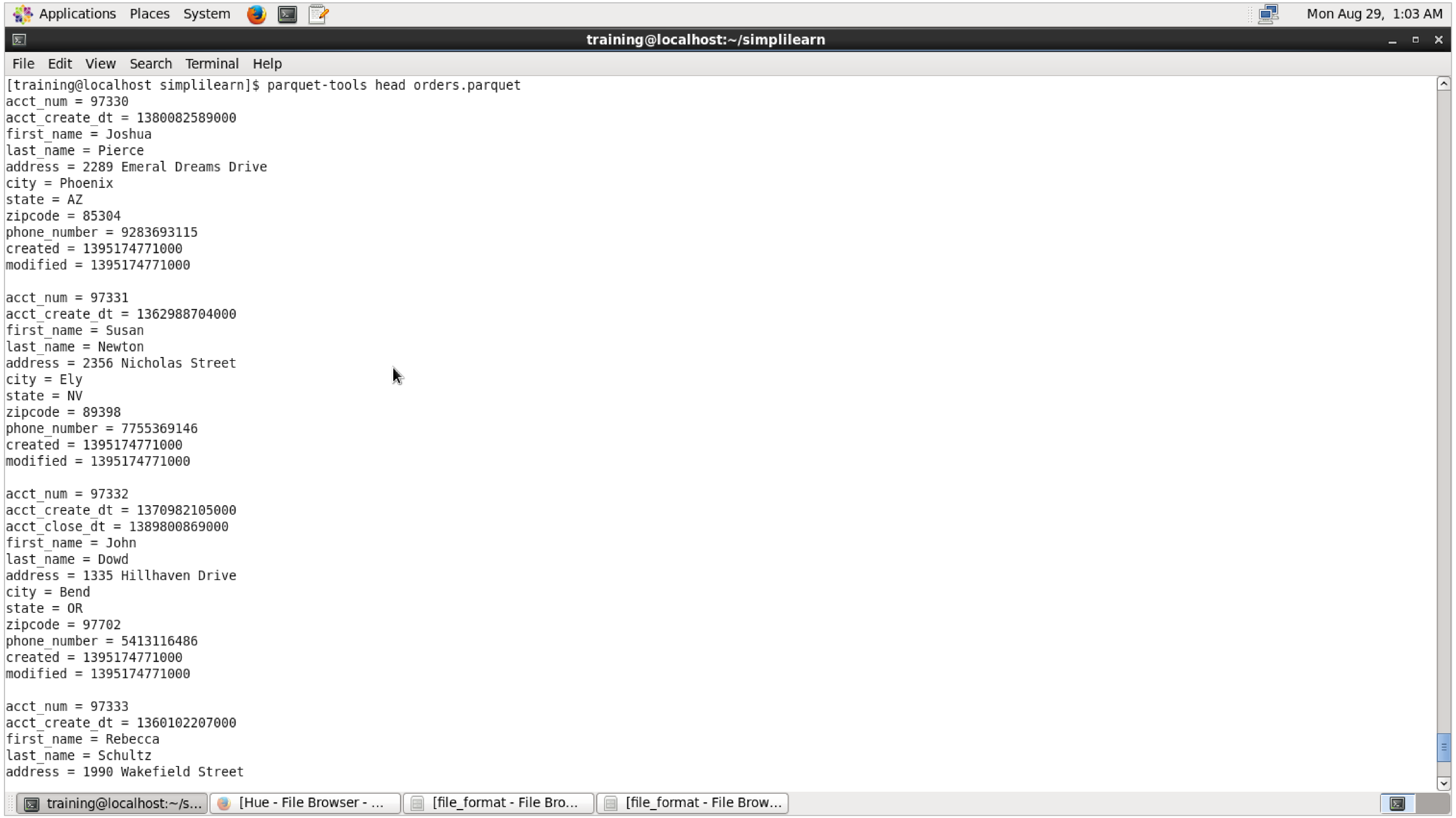
Create New Table with Parquet



Applications Places System
Mon Aug 29, 12:54 AM
training@localhost:~/simplilearn
File Edit View Search Terminal Help
training@localhost:~/s...
[Hue - File Browser - ...
[file_format - File Bro...
[file_format - File Brow...

```
[training@localhost simplilearn]$ impala-shell
Starting Impala Shell without Kerberos authentication
Connected to localhost.localdomain:21000
Server version: impalad version 2.2.0-cdh5.4.3 RELEASE (build 517bb0f71cd604a00369254ac6d88394df83e0f6)
Welcome to the Impala shell. Press TAB twice to see a list of available commands.

Copyright (c) 2012 Cloudera, Inc. All rights reserved.

(Shell build version: Impala Shell v2.2.0-cdh5.4.3 (517bb0f) built on Wed Jun 24 19:17:40 PDT 2015)
[localhost.localdomain:21000] > CREATE EXTERNAL TABLE new_order
                              > LIKE PARQUET '/simplilearn/order.parquet'
                              > STORED AS PARQUET
                              > LOCATION '/simplilearn/new_order/';
```

Reading Parquet Files Using Tools



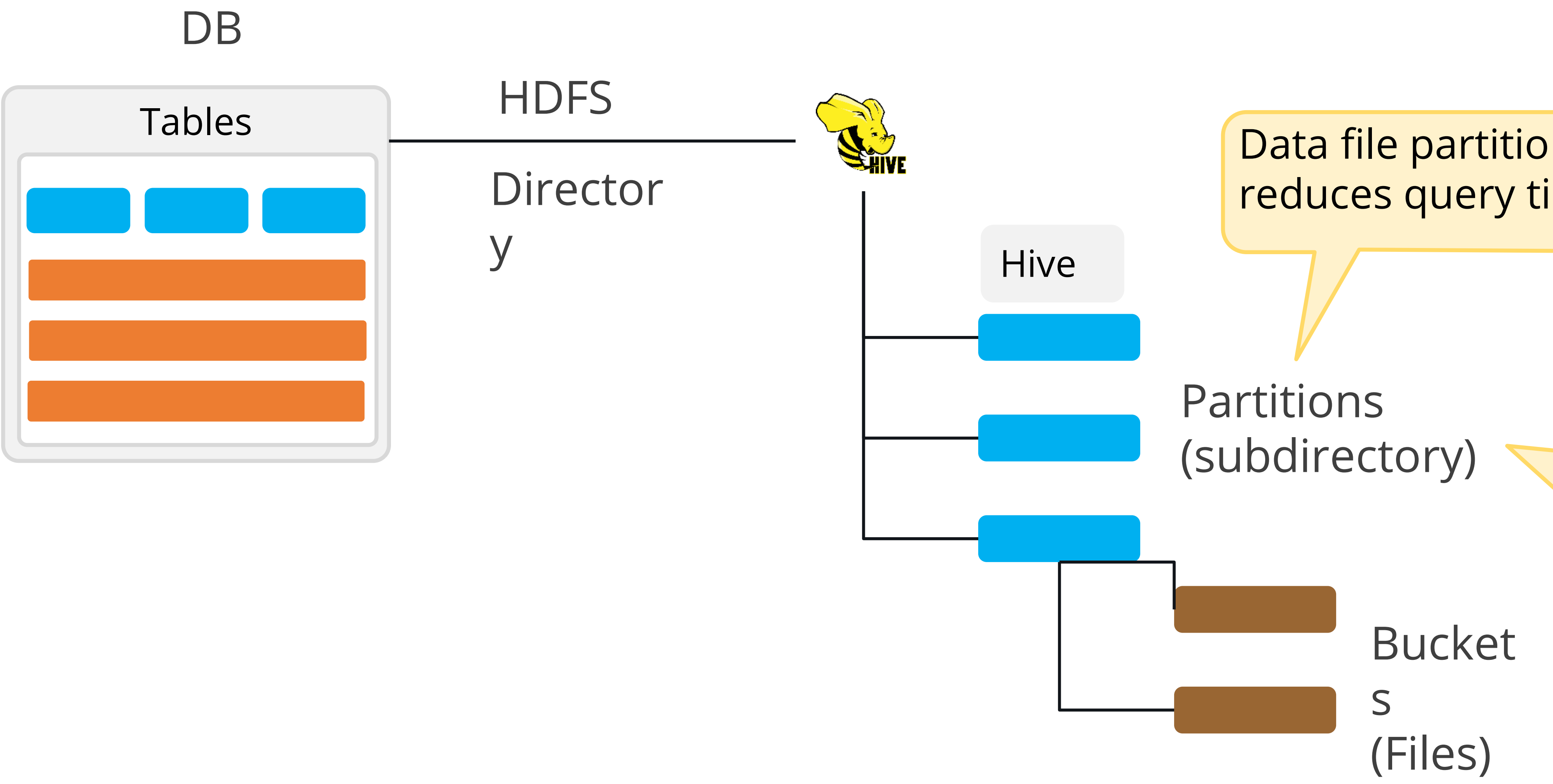
Applications Places System
Mon Aug 29, 1:03 AM
training@localhost:~/simplilearn
File Edit View Search Terminal Help
training@localhost:~/s...
[Hue - File Browser - ...
[file_format - File Bro...
[file_format - File Brow...

```
[training@localhost simplilearn]$ parquet-tools head orders.parquet
acct_num = 97330
acct_create_dt = 1380082589000
first_name = Joshua
last_name = Pierce
address = 2289 Emeral Dreams Drive
city = Phoenix
state = AZ
zipcode = 85304
phone_number = 9283693115
created = 1395174771000
modified = 1395174771000

acct_num = 97331
acct_create_dt = 1362988704000
first_name = Susan
last_name = Newton
address = 2356 Nicholas Street
city = Ely
state = NV
zipcode = 89398
phone_number = 7755369146
created = 1395174771000
modified = 1395174771000

acct_num = 97332
acct_create_dt = 1370982105000
acct_close_dt = 1389800869000
first_name = John
last_name = Dowd
address = 1335 Hillhaven Drive
city = Bend
state = OR
zipcode = 97702
phone_number = 5413116486
created = 1395174771000
modified = 1395174771000

acct_num = 97333
acct_create_dt = 1360102207000
first_name = Rebecca
last_name = Schultz
address = 1990 Wakefield Street
```

Hive Optimization: Partitioning, Bucketing, and Sampling

Data Storage

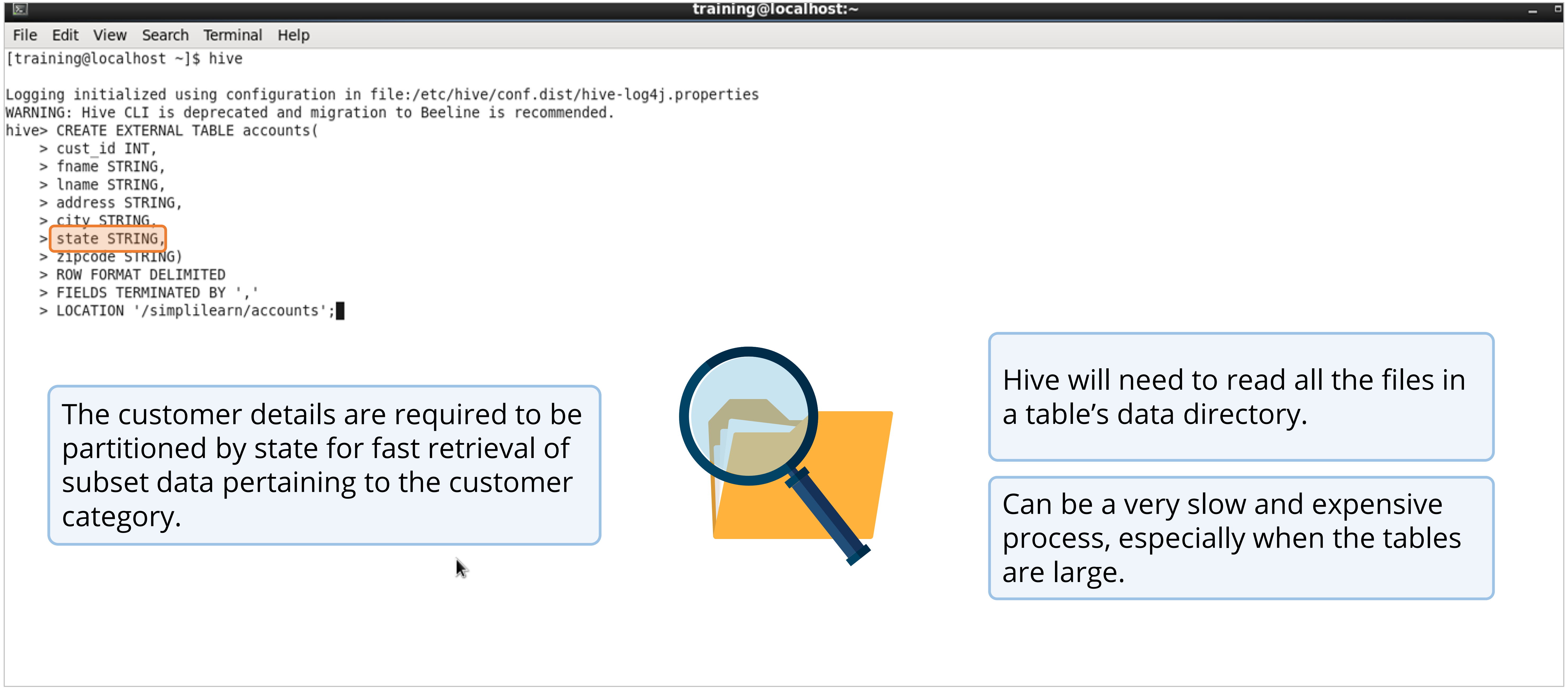
All files in a data set are stored in a single Hadoop Distributed File System or HDFS directory.



Data file partitioning reduces query time.

Subdirectories are created for each unique value of a partition column.

Example of a Non-Partitioned Table



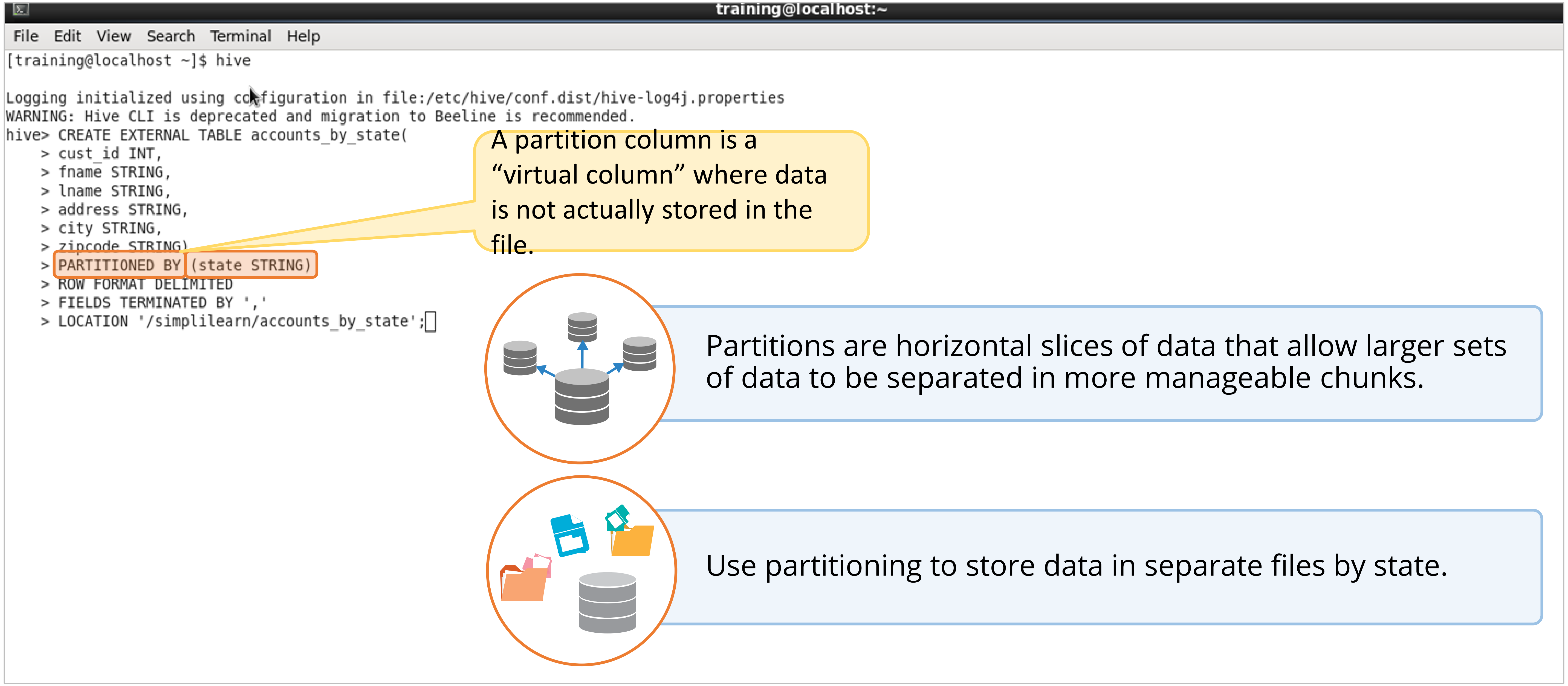
```
[training@localhost ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE EXTERNAL TABLE accounts(
    > cust_id INT,
    > fname STRING,
    > lname STRING,
    > address STRING,
    > city STRING,
    > state STRING,
    > zipcode STRING)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LOCATION '/simplilearn/accounts';
```

The customer details are required to be partitioned by state for fast retrieval of subset data pertaining to the customer category.

Hive will need to read all the files in a table’s data directory.

Can be a very slow and expensive process, especially when the tables are large.

Example of a Partitioned Table



```
[training@localhost ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE EXTERNAL TABLE accounts_by_state(
    > cust_id INT,
    > fname STRING,
    > lname STRING,
    > address STRING,
    > city STRING,
    > zipcode STRING)
    > PARTITIONED BY (state STRING)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LOCATION '/simplilearn/accounts_by_state';
```

A partition column is a “virtual column” where data is not actually stored in the file.

Partitions are horizontal slices of data that allow larger sets of data to be separated in more manageable chunks.

Use partitioning to store data in separate files by state.

Data Insertion

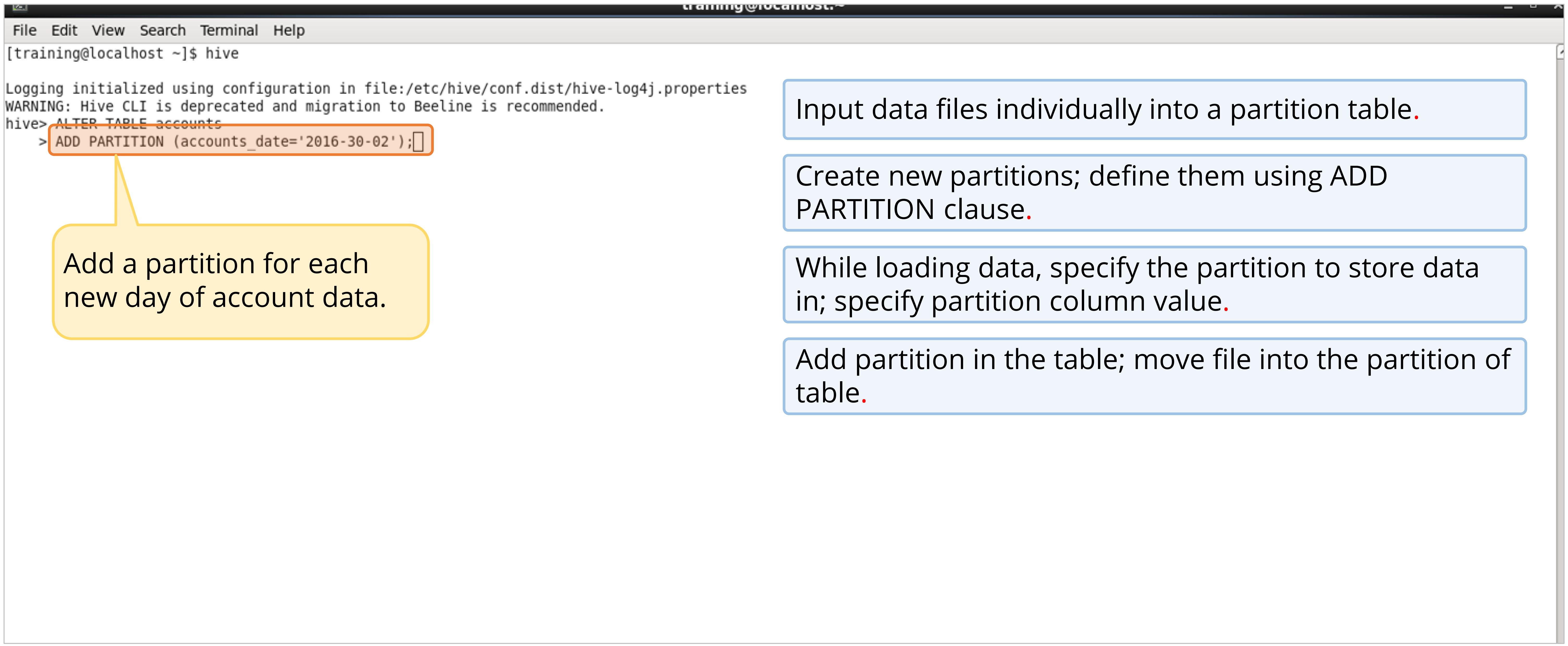
Data Insertion

Data insertion into partitioned tables can be done in two ways or modes:

Static partitioning

Dynamic partitioning

Static Partitioning



```
[training@localhost ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> ALTER TABLE accounts
    > ADD PARTITION (accounts_date='2016-30-02');
```

Add a partition for each new day of account data.

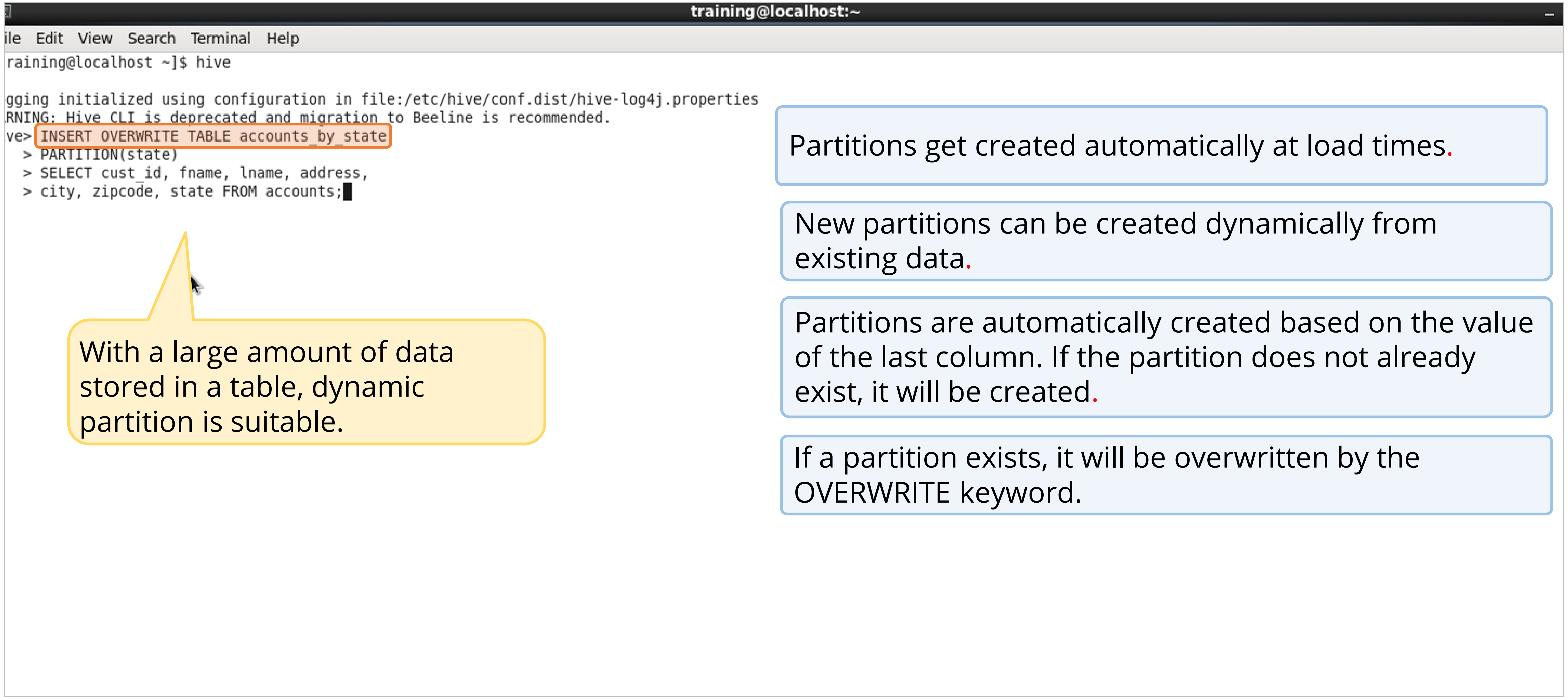
Input data files individually into a partition table.

Create new partitions; define them using ADD PARTITION clause.

While loading data, specify the partition to store data in; specify partition column value.

Add partition in the table; move file into the partition of table.

Dynamic Partitioning



```
raining@localhost ~]$ hive

gging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
RNING: Hive CLI is deprecated and migration to Beeline is recommended.
ve> INSERT OVERWRITE TABLE accounts_by_state
  > PARTITION(state)
  > SELECT cust_id, fname, lname, address,
  > city, zipcode, state FROM accounts;
```

With a large amount of data stored in a table, dynamic partition is suitable.

Partitions get created automatically at load times.

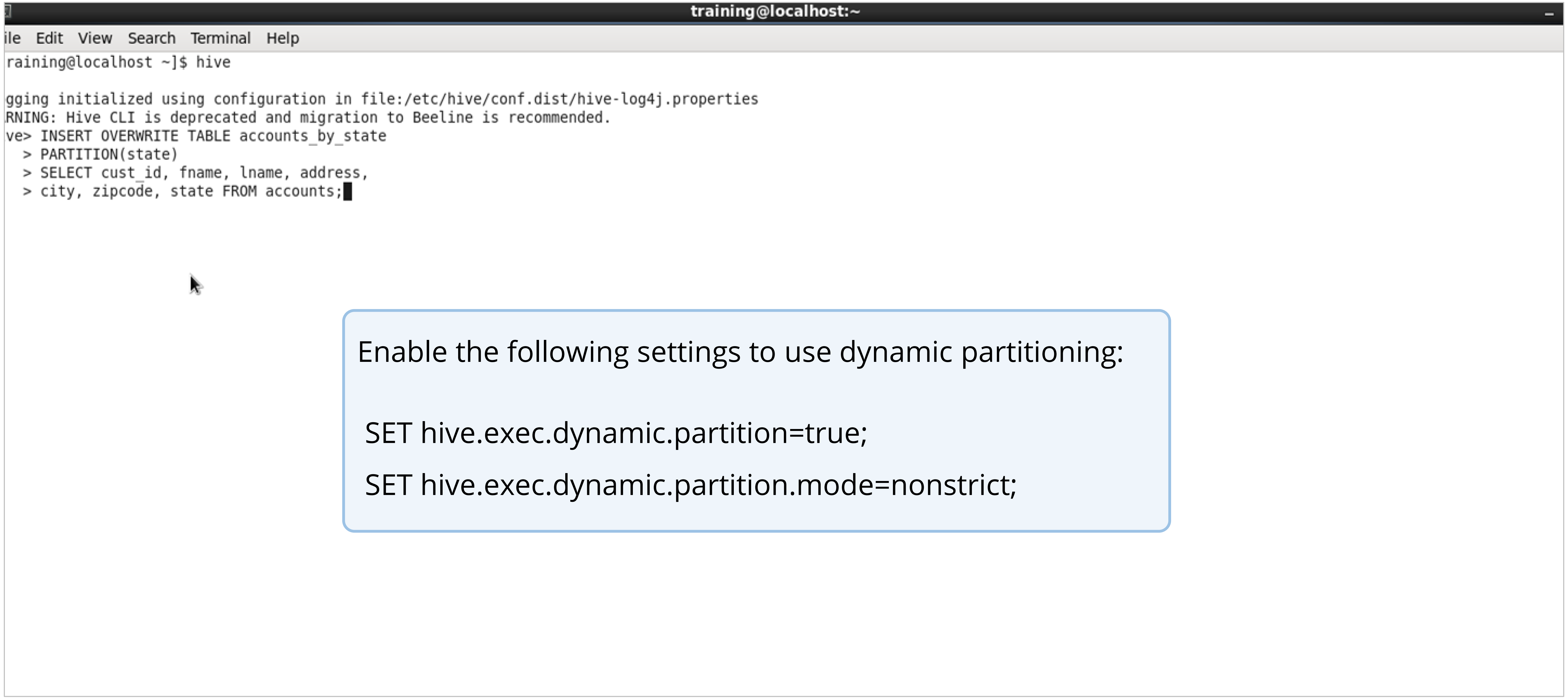
New partitions can be created dynamically from existing data.

Partitions are automatically created based on the value of the last column. If the partition does not already exist, it will be created.

If a partition exists, it will be overwritten by the OVERWRITE keyword.

Dynamic Partitioning in Hive

By default, dynamic partitioning is disabled in Hive to prevent accidental partition creation.



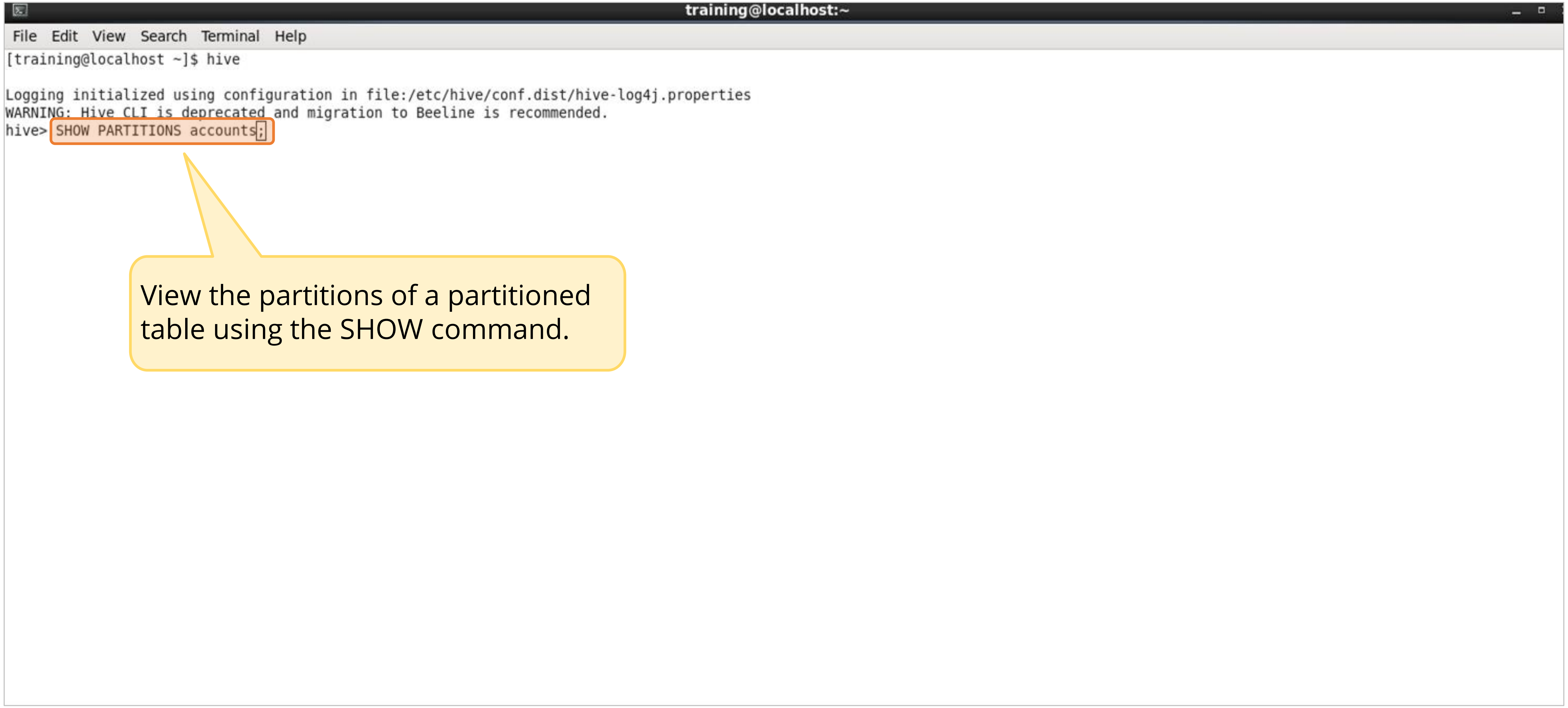
```
raining@localhost ~]$ hive

gging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
RNING: Hive CLI is deprecated and migration to Beeline is recommended.
ve> INSERT OVERWRITE TABLE accounts_by_state
  > PARTITION(state)
  > SELECT cust_id, fname, lname, address,
  > city, zipcode, state FROM accounts;
```

Enable the following settings to use dynamic partitioning:

```
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;
```

Viewing Partitions

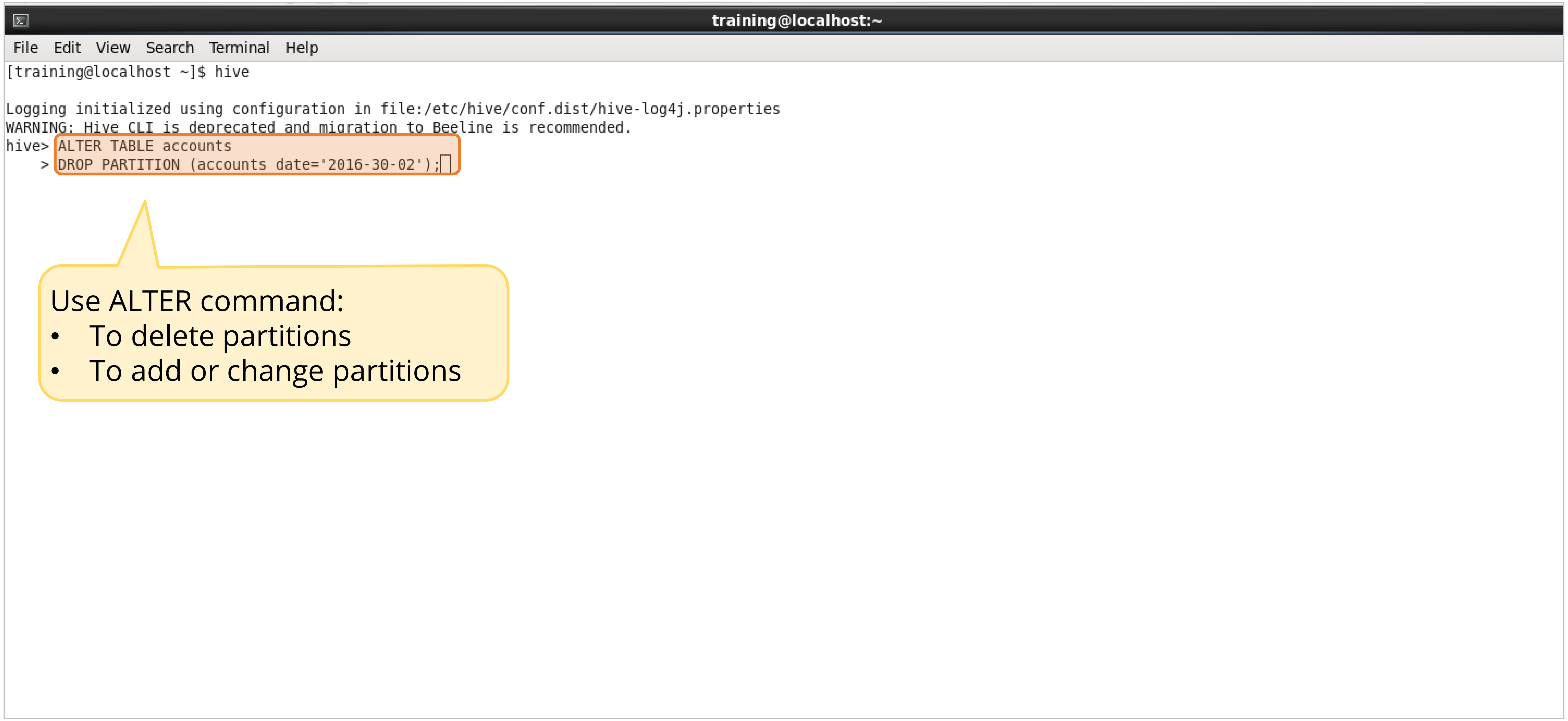
Commands that are supported on Hive partitioned tables to view and delete partitions.



```
[training@localhost ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> SHOW PARTITIONS accounts;
```

View the partitions of a partitioned table using the SHOW command.

Deleting Partitions



```
[training@localhost ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> ALTER TABLE accounts
    > DROP PARTITION (accounts date='2016-30-02');
```

Use ALTER command:

- To delete partitions
- To add or change partitions

When to Use Partitioning

Following are the instances when you need to use partitioning for tables:

- When reading the entire data set takes too long
- When queries almost always filter on the partition columns
- When there are a reasonable number of different values for partition columns

When Not to Use Partitioning

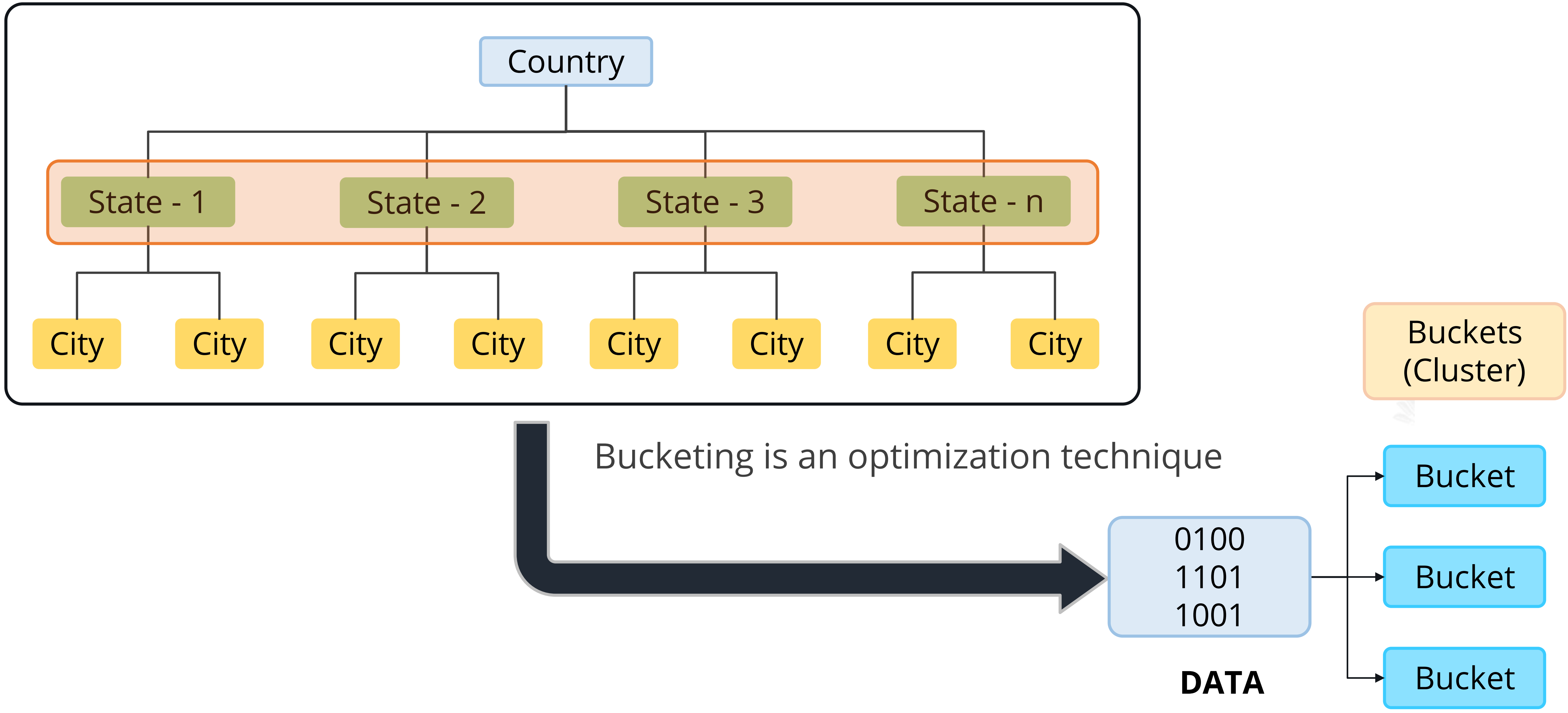
Following are the instances when you should avoid using a partitioning:

- When columns have too many unique rows
- When creating a dynamic partition as it can lead to high number of partitions
- When the partition is less than 20k

Bucketing

Bucketing in Hive

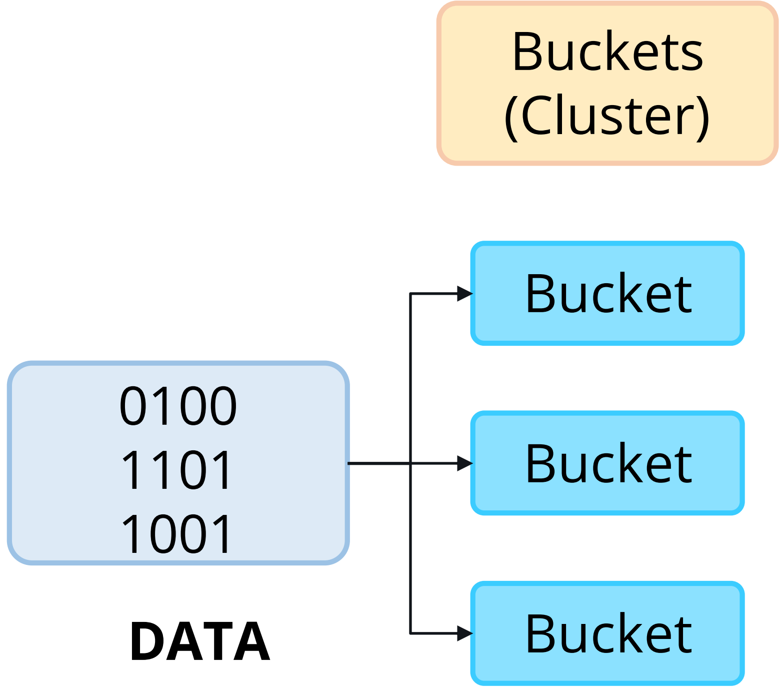
Partitioned column



Bucketing is an optimization technique

What Do Buckets Do?

Buckets distribute the data load into user-defined set of clusters by calculating the hash code of the key mentioned in the query.



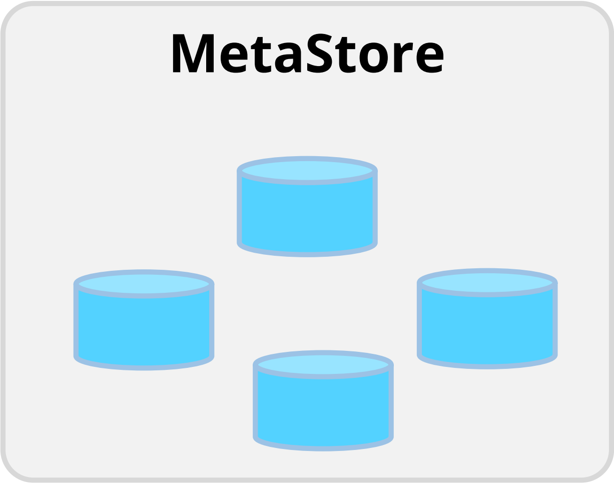
Syntax for creating a bucketed table

```
CREATE TABLE page_views( user_id INT, session_id BIGINT, url
STRING)
PARTIONED BY (day INT)
CLUSTERED BY (user_id) INTO 100;
```

The processor will first calculate the hash number of the user_id in the query and will look for only that bucket.

Hive Query Language: Introduction

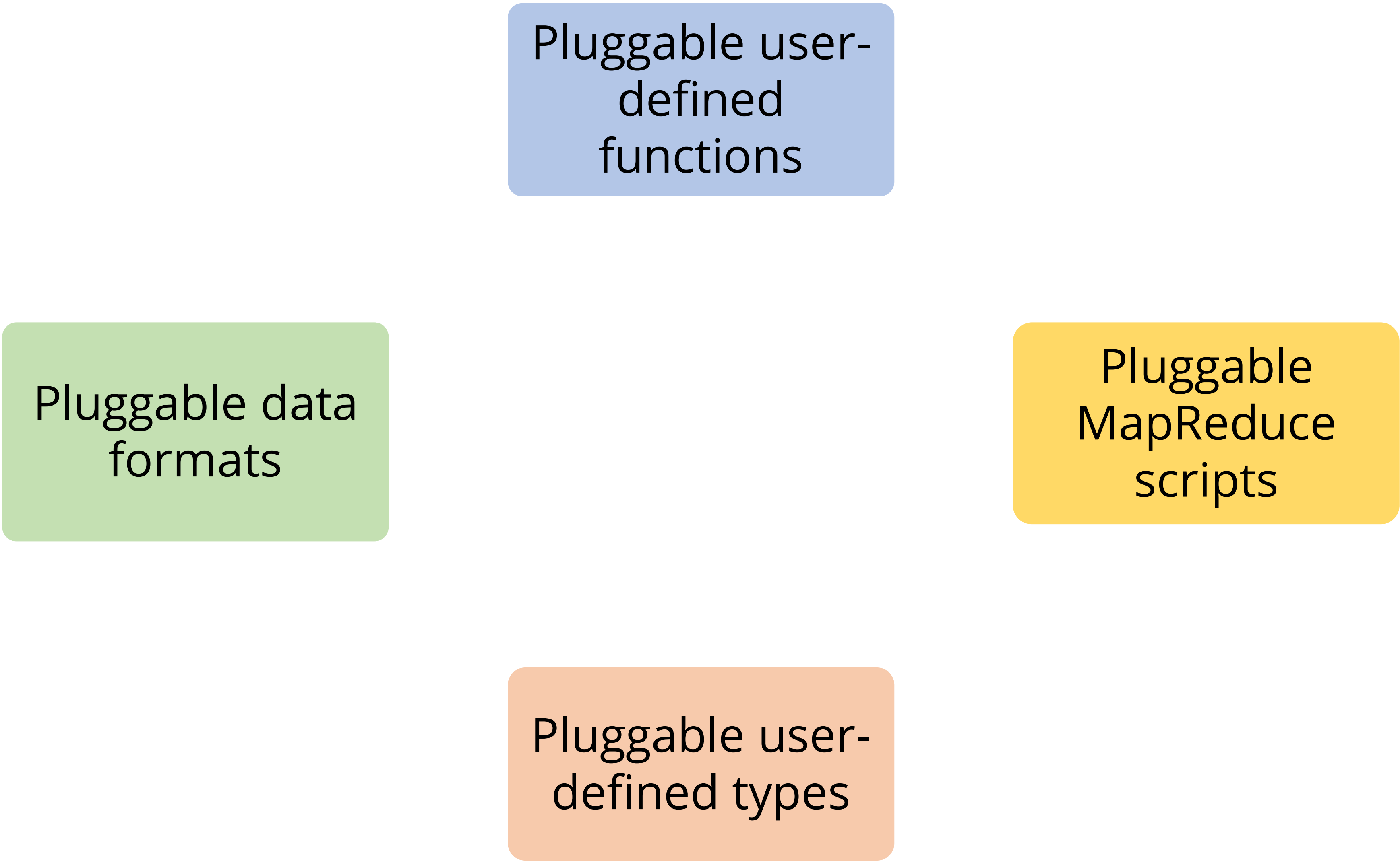
HiveQL is a SQL-like query language for Hive to process and analyze structured data in a Metastore.



```
SELECT
dt,
COUNT (DISTINCT (user_id))
FROM events
GROUP BY dt;
```

HiveQL: Extensibility

An important principle of HiveQL is its extensibility. HiveQL can be extended in multiple ways:



Pluggable user-defined functions

Pluggable data formats

Pluggable MapReduce scripts

Pluggable user-defined types

Hive Analytics: UDF and UDAF

User-Defined Function

Hive has the ability to define a function. UDFs extend the functionality of Hive, with a function written in Java, that can be evaluated in HiveQL statements.

All UDFs extend the Hive UDF class. After that, a UDF sub-class implements one or more methods named ‘evaluate’.

Evaluate should never be a void method. It can return null value, if required.

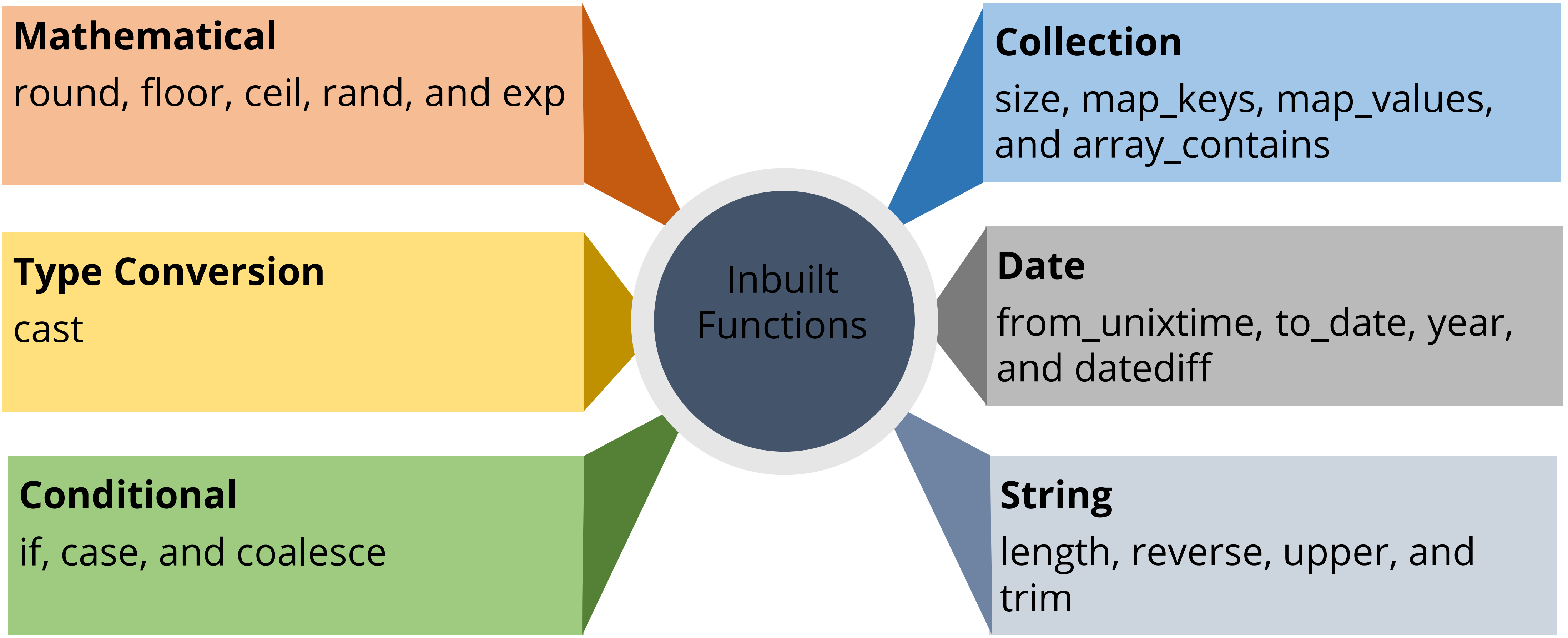
Code for Extending UDF

Here is a code that you can use to extend the User-Defined Function.

```
package com.example.hive.udf;

import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

public final class Lower extends UDF {
  public Text evaluate(final Text s) {
    if (s == null) { return null; }
    return new Text(s.toString().toLowerCase());
  }
}
```

Built-in Functions of Hive

Writing the functions in JAVA scripts creates its own UDF. Hive also provides some inbuilt functions that can be used to avoid own UDFs from being created.



Inbuilt Functions

Mathematical
round, floor, ceil, rand, and exp

Type Conversion
cast

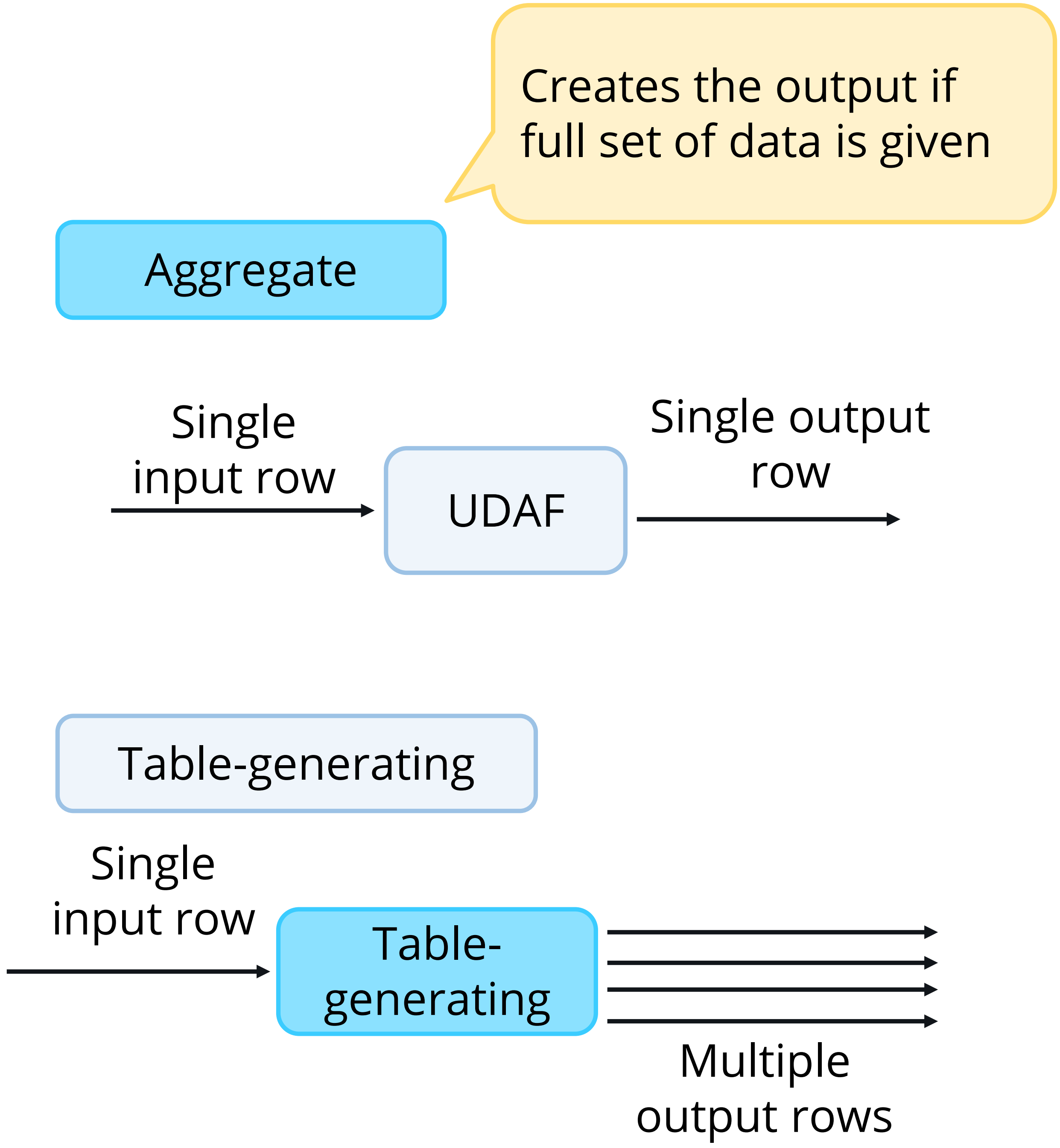
Conditional
if, case, and coalesce

Collection
size, map_keys, map_values, and array_contains

Date
from_unixtime, to_date, year, and datediff

String
length, reverse, upper, and trim

Other Functions of Hive



Creates the output if full set of data is given

Aggregate

Single input row

UDAF

Single output row

Table-generating

Single input row

Table-generating

Multiple output rows

Lateral view:

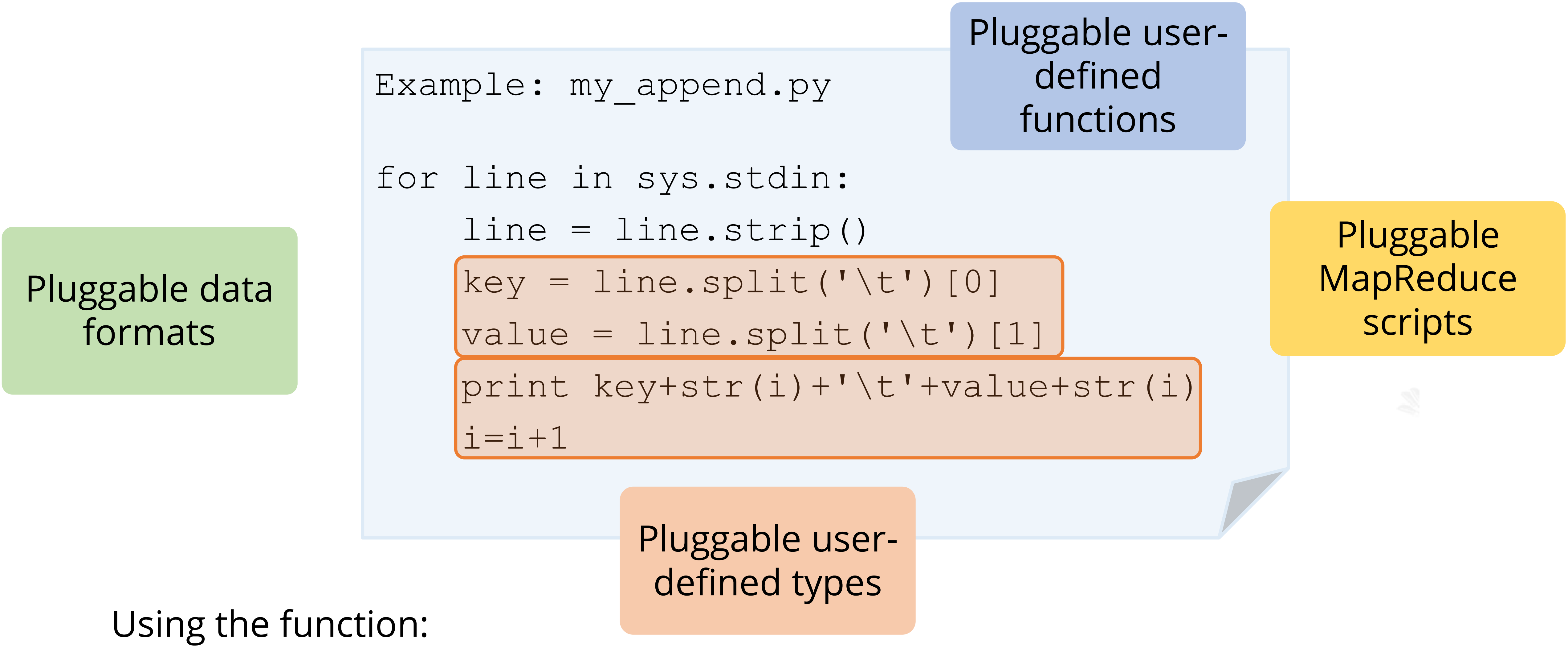
String pageid	Array<int> adid_list
"front_page"	[1,2,3]
"contact_page"	[3, 4, 5]

```
SELECT pageid, adid FROM pageAds
LATERAL VIEW explode(adid_list) adTable
AS adid;
```

String pageid	intadid
"front_page"	1
"front_page"	2
……	……

MapReduce Scripts

MapReduce scripts are written in scripting languages, such as Python.



```
Example: my_append.py

for line in sys.stdin:
    line = line.strip()
    key = line.split('\t')[0]
    value = line.split('\t')[1]
    print key+str(i)+'\t'+value+str(i)
    i=i+1
```

Pluggable user-defined functions

Pluggable data formats

Pluggable MapReduce scripts

Pluggable user-defined types

Using the function:

```
SELECT TRANSFORM (foo, bar) USING 'python ./my_append.py' FROM sample;
```

UDF/UDAF vs. MapReduce Scripts

Attribute	UDF/UDAF	MapReduce scripts
Language	Java	Any language
1/1 input/output	Supported via UDF	Supported
n/1 input/output	Supported via UDAF	Supported
1/n input/output	Supported via UDTF	Supported
Speed	Faster (in same process)	Slower (spawns new process)

Key Takeaways

You are now able to:

- Define Hive and its architecture
- Create and manage tables using Hue Web UI and Beeline
- Understand various file formats supported in Hive
- Use HiveQL DDL to create tables and execute queries

Knowledge Check

Knowledge Check

1

Deleting an individual record is possible in______ .

a. Hive

b. RDBMS

c. Both A and B

d. None of the above

Knowledge Check

1

Deleting an individual record is possible in______ .

a. Hive

b. RDBMS

c. Both A and B

d. None of the above

The correct answer is **b.**

Hive cannot delete individual records, but an RDBMS can.

Knowledge Check

2

In which HDFS directory is Hive table created by default?

a. /hive

b. /user/hive/

c. /user/hive/warehouse

d. All of the above

Knowledge Check

2

In which HDFS directory is Hive table created by default?

a. /hive

b. /user/hive/

c. /user/hive/warehouse

d. All of the above

The correct answer is **c.**

Hive table gets created by default in /user/hive/warehouse in HDFS directory.

Knowledge Check

3

Which of the following statements is true for sequential file format?

a. More efficient than a text file

b. Store key-value pairs in a binary container format

c. Not human-readable

d. All of the above

Knowledge Check

3

Which of the following statements is true for sequential file format?

a. More efficient than a text file

b. Store key-value pairs in a binary container format

c. Not human-readable

d. All of the above

The correct answer is **d.**

Sequential File format stores key-value pairs in a binary container format, is efficient than a text file, and it is not human–readable.

Knowledge Check

4

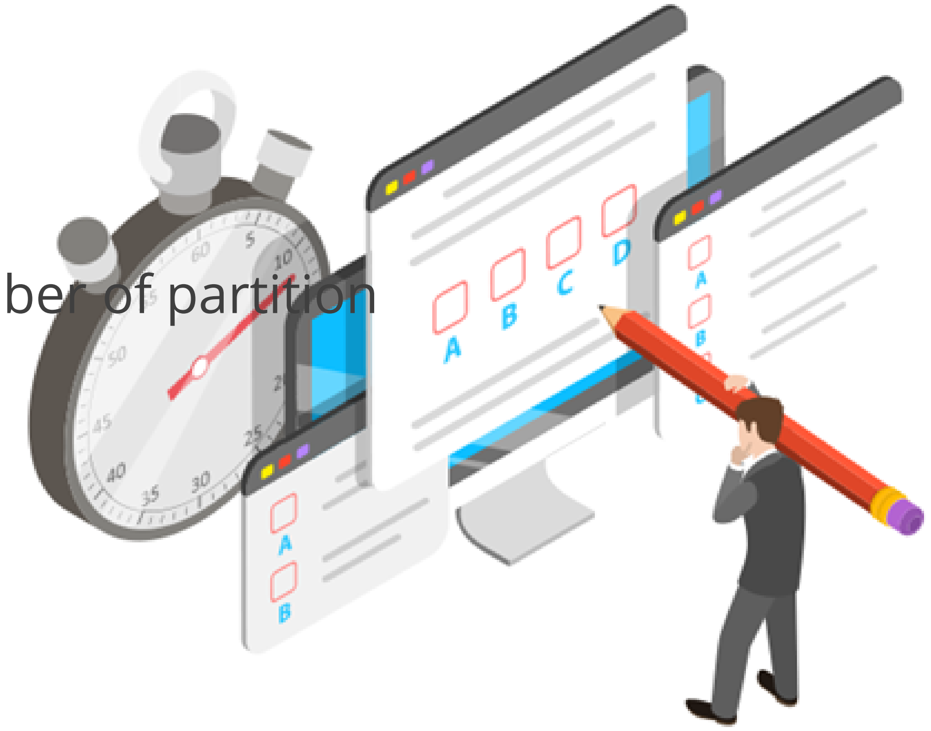
Which of the following statements is true about when not to use partitions?

a. Try to limit partition to less than 20k

b. Avoid partition on columns having too many unique rows

c. Be cautious while creating dynamic partition as it can lead to high number of partition

d. All of the above



Knowledge Check

4

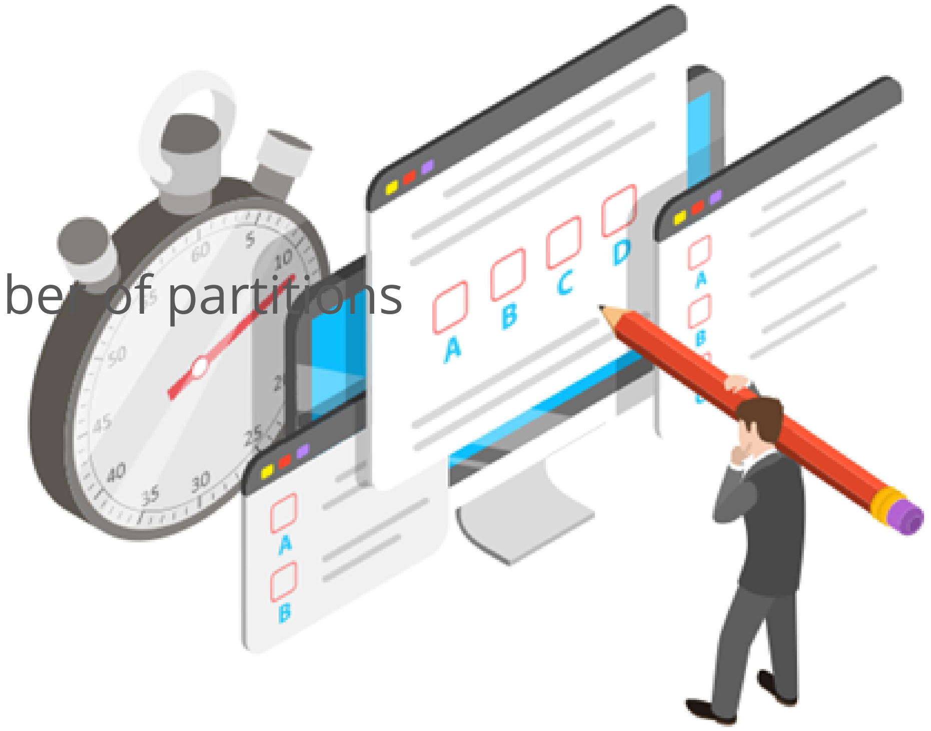
Which of the following statements is true about when not to use partitions?

a. Try to limit partition to less than 20k

b. Avoid partition on columns having too many unique rows

c. Be cautious while creating dynamic partition as it can lead to high number of partitions

d. All of the above



The correct answer is **d.**

We should avoid using partitions when columns have too many unique rows, while creating dynamic partition as it can lead to high number of partitions, and when limiting partition to less than 20k.

Knowledge Check

5

Which of the following is a way of representing data in memory as a series of bytes?

a. File Formatting

b. Data Serialization

c. Both A and B

d. None of the above

Knowledge Check

5

Which of the following is a way of representing data in memory as a series of bytes?

a. File Formatting

b. Data Serialization

c. Both A and B

d. None of the above

The correct answer is **b.**

Data Serialization is a way of representing data in memory as a series of bytes.

Lesson-End Project

Problem Statement:

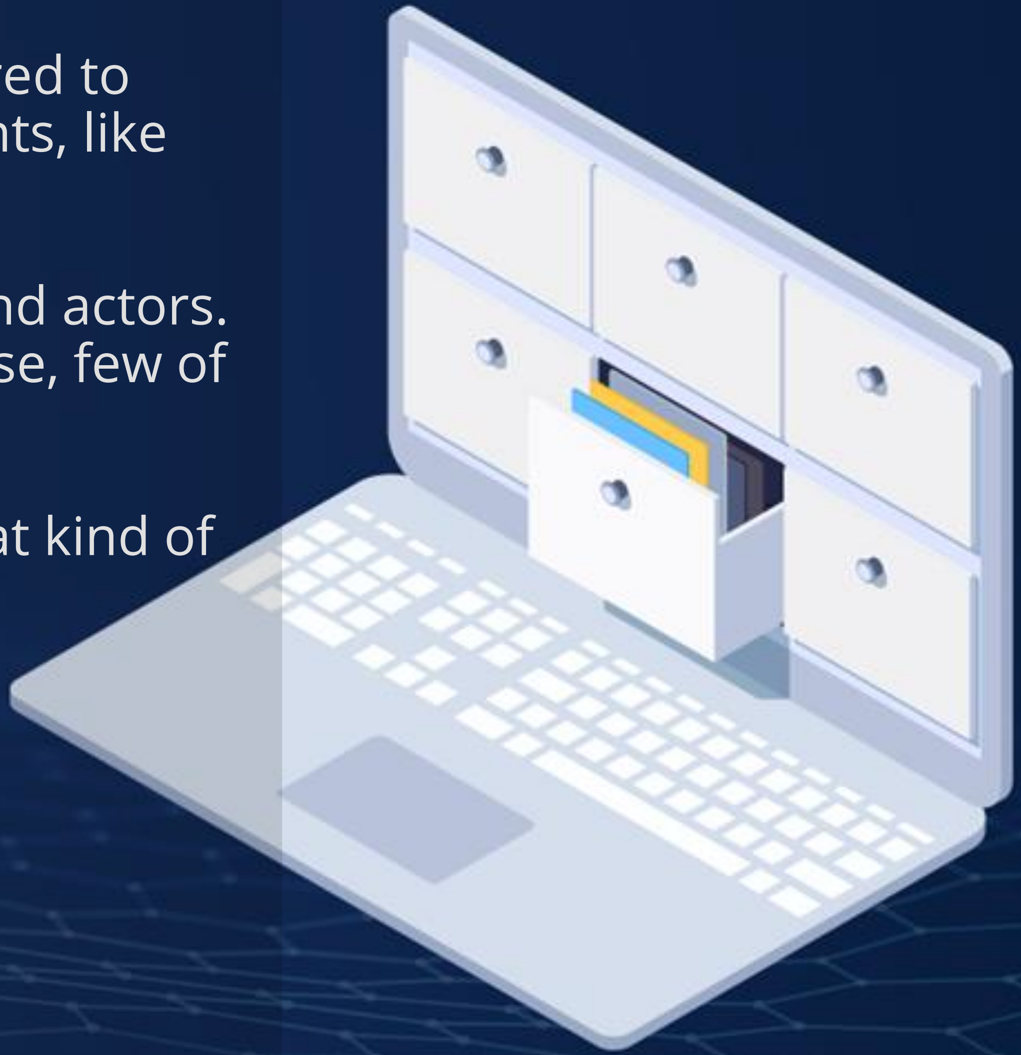
Everybody loves movies. Nowadays, movie releases per year has increased compared to earlier days because of an increase in the number of production houses. A few giants, like Netflix and Amazon, have started creating their content as well.

Hollywood is spreading its wings in most countries because of its graphics, story, and actors. In Hollywood, few directors have made great impact among audiences. Among these, few of them have received nominations and won awards.

Before watching a movie, people tend to validate the director's credentials like, what kind of movies he has made in the past and if he has won any awards.

The given data set has details about the movie directors and whether they have received nominations and won awards.

The dataset contains the following fields:
1. Director name
2. Ceremony
3. Year
4. Category
5. Outcome
6. Original language



Lesson-End Project

Find out the below insights:

1. Directors who were nominated and have won awards in the year 2011
2. Award categories available in the Berlin International Film Festival
3. Directors who won awards for making movies in French
4. Directors who have won awards more than 10 times

Thank You