Aim: Use Sqoop to load data from RDBMS (weblog/transactions dara) and analyze it using Hive.

Theory:

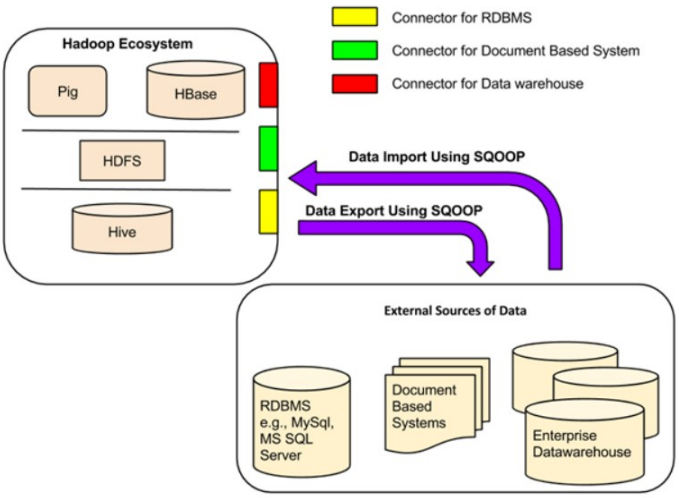
Introduction to Sqoop:

Sqoop (SQL-to-Hadoop) is an open-source data transfer tool that facilitates the efficient movement of data between Apache Hadoop and structured data stores, such as relational databases. It was developed by the Apache Software Foundation and is widely used in the big data ecosystem to import data from relational databases into Hadoop's HDFS (Hadoop Distributed File System) and export data from Hadoop to relational databases.

The primary use case for Sqoop is to bridge the gap between the traditional relational database world and the Hadoop ecosystem, allowing organizations to leverage the power of Hadoop for processing and analysis while still being able to work with their existing data stored in relational databases.

Architecture of Sqoop:

Using its connectors, sqoop facilitates data migration between Hadoop and external storage systems. These connectors enable Sqoop to work with various widely-used relational databases, such as MySQL, PostgreSQL, Oracle, SQL Server, and DB2. Each connector establishes communication with the corresponding DBMS it is associated with. A generic JDBC connector is also available for connecting to any database that adheres to the JDBC standard. Furthermore, Sqoop Big Data offers specialized connectors optimized for PostgreSQL and MySQL, leveraging database-specific APIs for enhanced performance.



Advantages of Sqoop:

1. It entails data transfer from numerous structured sources, like Oracle, Postgres, etc.
2. Due to the parallel data transport, it is quick and efficient.
3. Many procedures can be automated, which increases efficiency.
4. Integration with Kerberos security authentication is feasible.
5. Direct data loading is possible from HBase and Hive.
6. It is a powerful tool with a sizable support network.
7. As a result of its ongoing development and contributions, it is frequently updated.

Issues in Sqoop:

<u>Data load using Scripts</u>

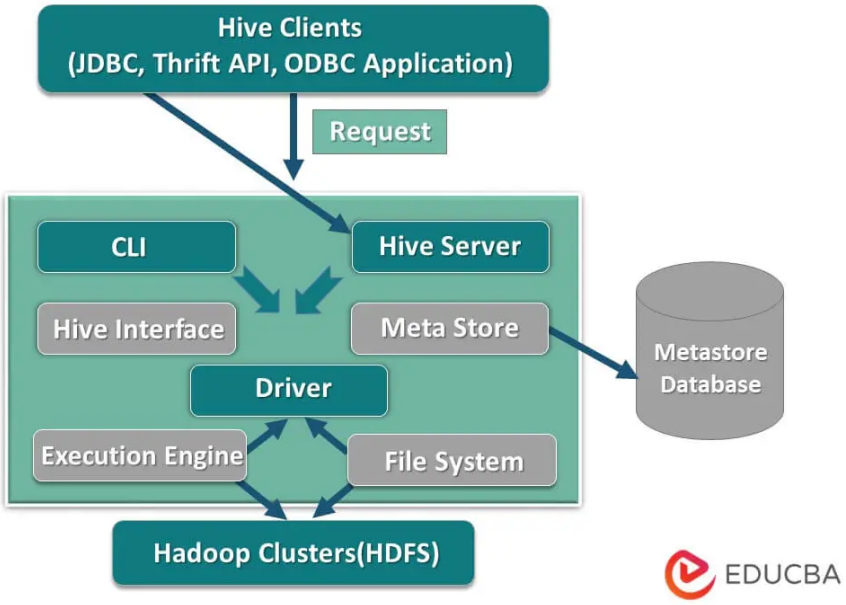
The traditional approach of using scripts to load data is not suitable for bulk data load into Hadoop; this approach is inefficient and very time-consuming.

<u>Direct access to external data via Map-Reduce application</u>

Providing direct access to the data residing at external systems(without loading into Hadoop) for map-reduce applications complicates these applications. So, this approach is not feasible.

Introduction to Hive:
Apache Hive is an open-source data warehousing and SQL-like query language tool built on top of the Hadoop ecosystem. It was developed by the Apache Software Foundation to provide a high-level interface for querying and analyzing large datasets stored in Hadoop's distributed storage system, HDFS (Hadoop Distributed File System). Hive is particularly useful for users familiar with SQL as it allows them to work with big data using a familiar query language.

Architecture of Hive:



Hive Clients

They include Thrift application to execute easy hive commands which are available for python, ruby, C++, and drivers. These client application benefits for executing queries on the hive. Hive has three types of client categorization: thrift clients, JDBC, and ODBC clients.

Hive Services

To process all the queries hive has various services. All the functions are easily defined by the user in the hive.Let’s see all those services in brief:

Command-line Interface (CLI): User interface for executing Hive commands and queries. Web interfaces (HWI) can also be used for query submission.
Hive Driver: Receives queries from various sources, interacts with ODBC/JDBC drivers, and manages query execution stages.
Compiler: Parses and performs semantic analysis on queries, creating abstract syntax trees and optimizing query execution plans.
Execution Engine: Processes queries by executing tasks, managing task dependencies, and executing DAG stages.
Metastore: Central repository storing metadata about tables, partitions, and HDFS file storage. Manages structured information and acts as a namespace for tables.

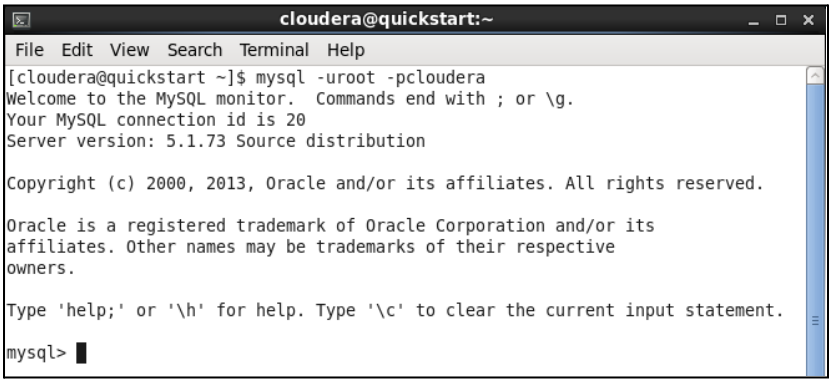
Characteristics of Hive:

1. Databases and tables are built before loading the data.
2. Hive as a data warehouse is built to manage and query only structured data which is residing under tables.
3. At the time of handling structured data, MapReduce lacks optimization and usability functions such as UDFs whereas Hive framework have optimization and usability.
4. Programming in Hadoop deals directly with the files. So, Hive can partition the data with directory structures to improve performance on certain queries.
5. Hive is compatible for the various file formats which are TEXTFILE, SEQUENCEFILE, ORC, RCFILE, etc.
6. Hive uses derby database in single user metadata storage and it uses MYSQL for multiple user Metadata or shared Metadata.

Implementation:

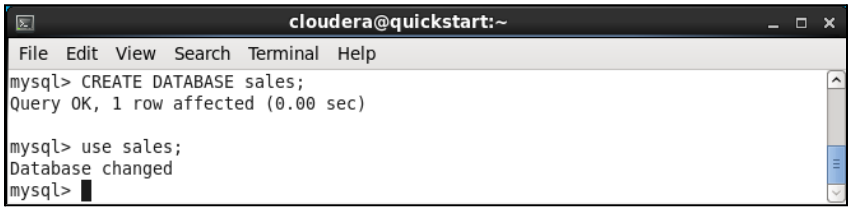
1. Entering into the MySQL command line

Command: mysql -uroot -pcloudera



2. Creating a 'sales' database in mysql

Command: CREATE DATABASE sales;
use sales;



3. Creating a table in the mysql

Command: CREATE TABLE sales (invoice_id INT(10) NOT NULL PRIMARY KEY, city VARCHAR(20), `product_line` VARCHAR(20), unit_price FLOAT, quantity INT(10), tax FLOAT, total FLOAT, `date` DATETIME, `time` DATETIME, payment VARCHAR(20), cogs FLOAT, rating INT);

desc sales;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
mysql> use sales;
Database changed
mysql> CREATE TABLE sales(invoice_id INT(10) not null primary key, city VARCHAR(
20), product_line VARCHAR(20), unit_price FLOAT, quantity INT(10), tax FLOAT, to
tal FLOAT, date datetime, time datetime,payment VARCHAR(20), cogs FLOAT, rating
INT);
Query OK, 0 rows affected (2.40 sec)

mysql> desc sales;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| invoice_id   | int(10)     | NO   | PRI | NULL    |       |
| city         | varchar(20) | YES  |     | NULL    |       |
| product_line | varchar(20) | YES  |     | NULL    |       |
| unit_price   | float       | YES  |     | NULL    |       |
| quantity     | int(10)     | YES  |     | NULL    |       |
| tax          | float       | YES  |     | NULL    |       |
| total        | float       | YES  |     | NULL    |       |
| date         | datetime    | YES  |     | NULL    |       |
| time         | datetime    | YES  |     | NULL    |       |
| payment      | varchar(20) | YES  |     | NULL    |       |
| cogs         | float       | YES  |     | NULL    |       |
| rating       | int(11)     | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
12 rows in set (0.95 sec)
```

4. Importing the values into the table

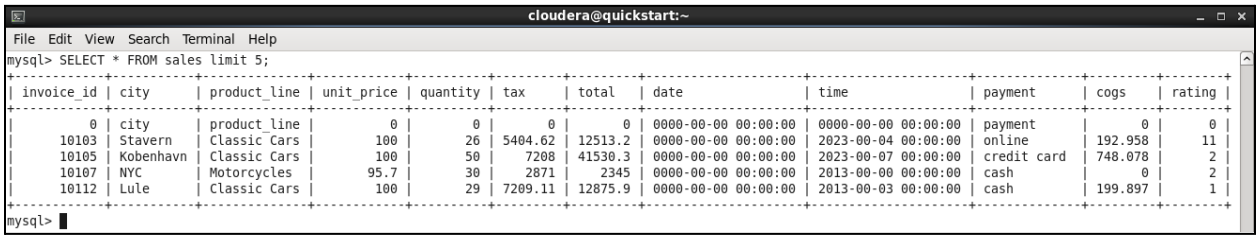
Command: LOAD Data Local Infile '/home/cloudera/Desktop/sales.csv' into table sales Fields Terminated By ' , ' Lines Terminated By '\n' ;

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
mysql> LOAD Data Local Infile '/home/cloudera/Desktop/sales.csv' into table sales
Fields Terminated By ',' Lines Terminated By '\n';
Query OK, 67 rows affected, 27368 warnings (0.22 sec)
Records: 2824  Deleted: 0  Skipped: 2757  Warnings: 19223

mysql>
```

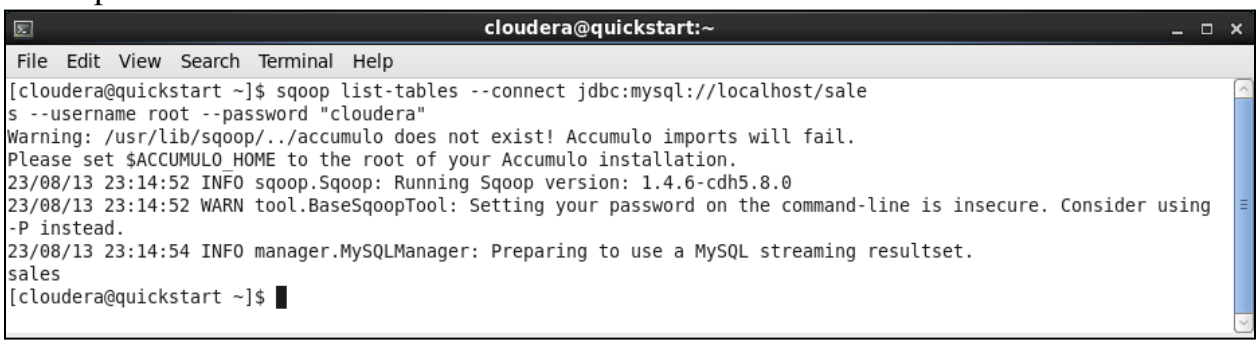
5. Checking the values

Command: SELECT * FROM sales;



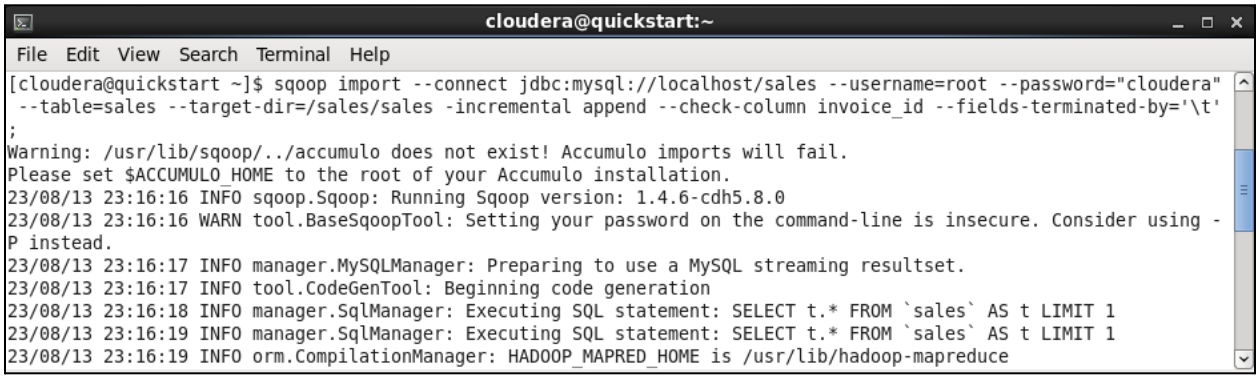
6. Listing the tables present in database sales on MySQL. We use Sqoop to list all the tables present in the MySQL database.

Command: sqoop list-tables --connect jdbc:mysql://localhost/sales --username root --password "cloudera"



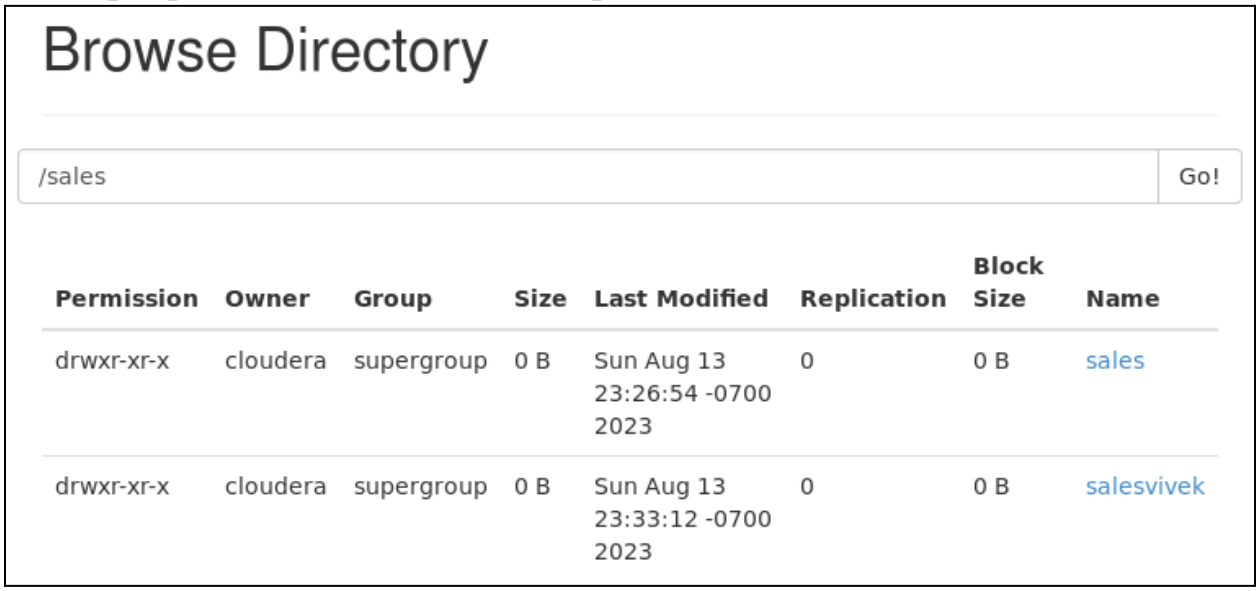
7. Import tables from rdbms to hdfs using sqoop

Command: sqoop import --connect jdbc:mysql://localhost/sales --username root --password "cloudera" --table sales --target-dir /sales/sales --incremental append --check-column invoice_id --fields-terminated-by '\t'



8. checking if tables are imported properly

Url: http://quickstart.cloudera:50070/explorer.html#/sales



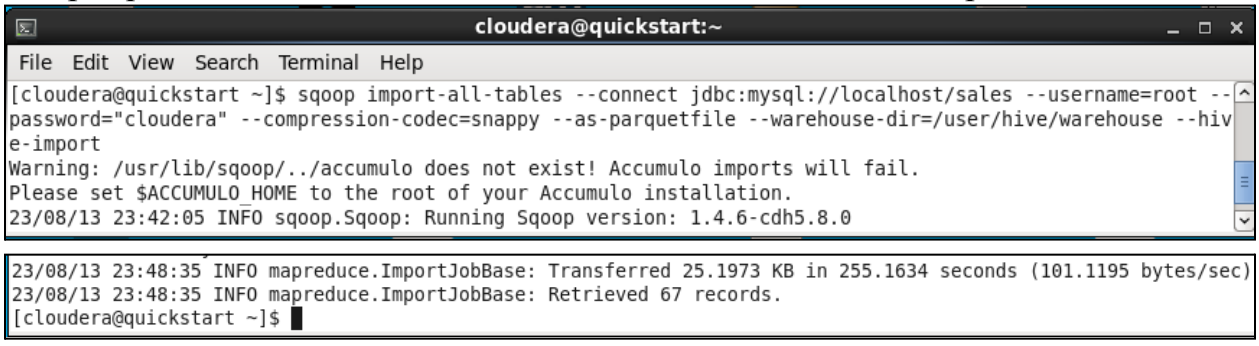
Browse Directory

/sales Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	cloudera	supergroup	0 B	Sun Aug 13 23:26:54 -0700 2023	0	0 B	sales
drwxr-xr-x	cloudera	supergroup	0 B	Sun Aug 13 23:33:12 -0700 2023	0	0 B	salesvivek

9. Import tables from HDFS to Hive

Command: sqoop import-all-tables --connect jdbc:mysql://localhost/sales --username root --password "cloudera" --compression-codec=snappy --as-parquetfile --warehouse-dir=/user/hive/warehouse --hive-import



cloudera@quickstart:~

File Edit View Search Terminal Help

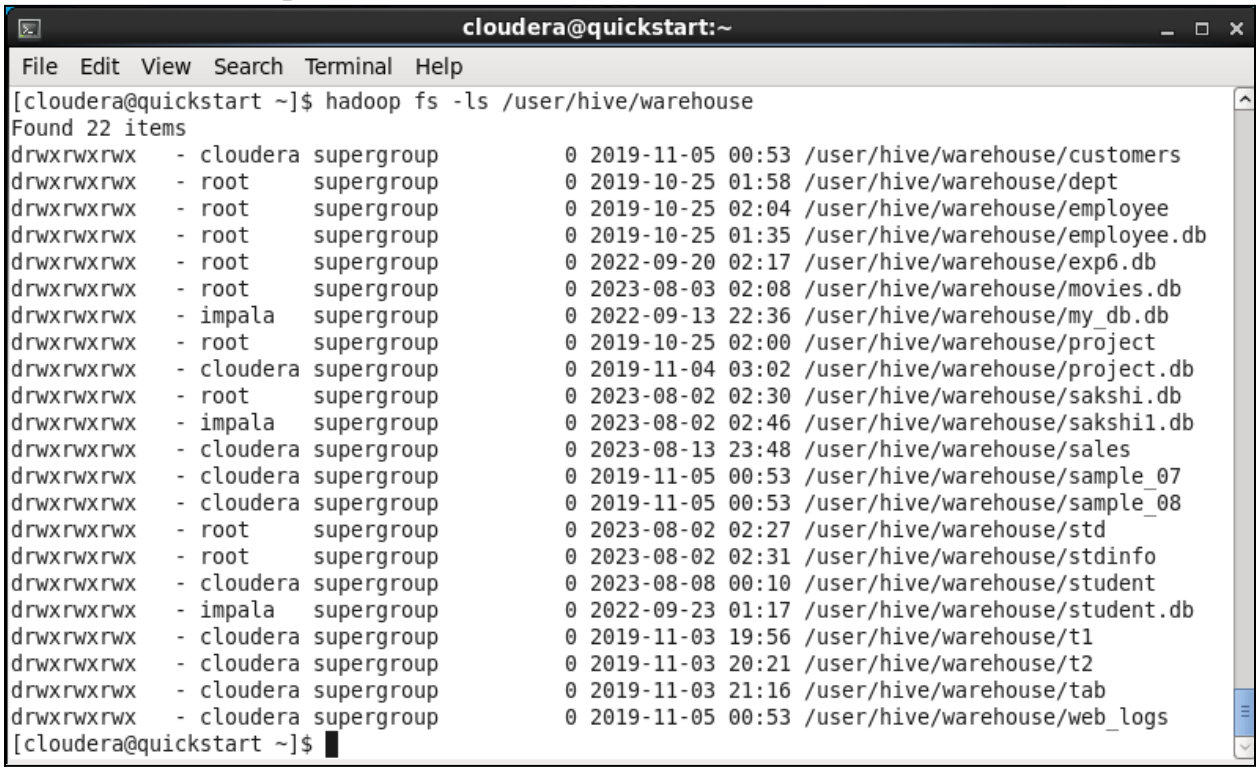
```
[cloudera@quickstart ~]$ sqoop import-all-tables --connect jdbc:mysql://localhost/sales --username=root --
password="cloudera" --compression-codec=snappy --as-parquetfile --warehouse-dir=/user/hive/warehouse --hiv
e-import
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
23/08/13 23:42:05 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.8.0
```



```
23/08/13 23:48:35 INFO mapreduce.ImportJobBase: Transferred 25.1973 KB in 255.1634 seconds (101.1195 bytes/sec)
23/08/13 23:48:35 INFO mapreduce.ImportJobBase: Retrieved 67 records.
[cloudera@quickstart ~]$
```

10. Check if it created in hive

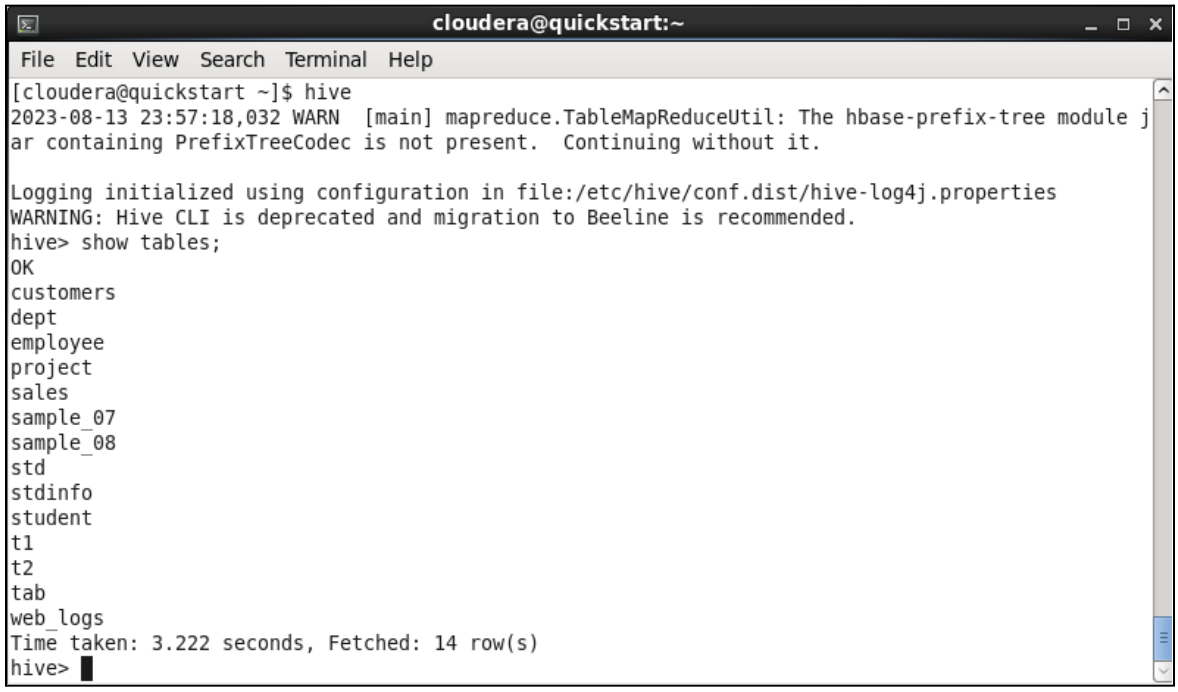
Command: hadoop fs -ls /user/hive/warehouse



11. Loging into hive

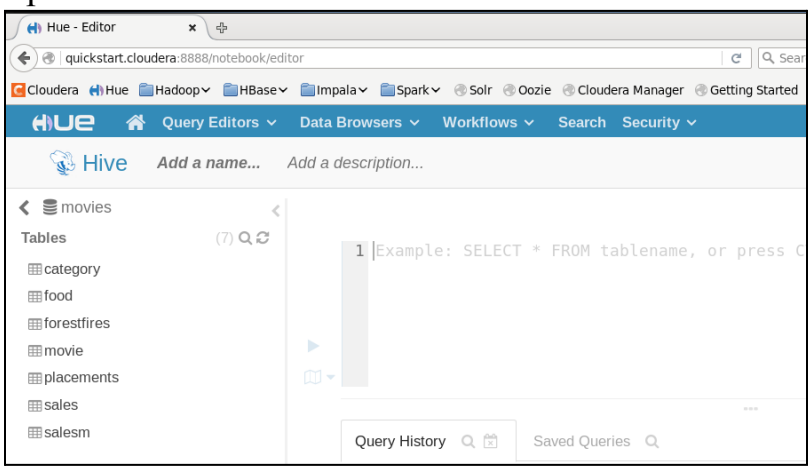
Command: hive

Show tables;

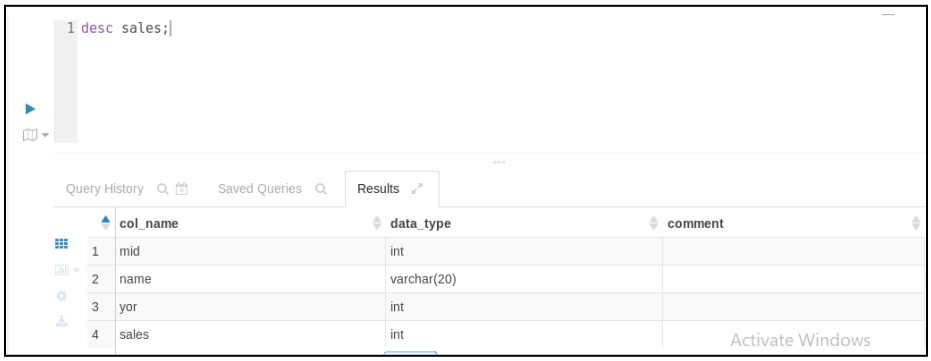


12. Executing queries

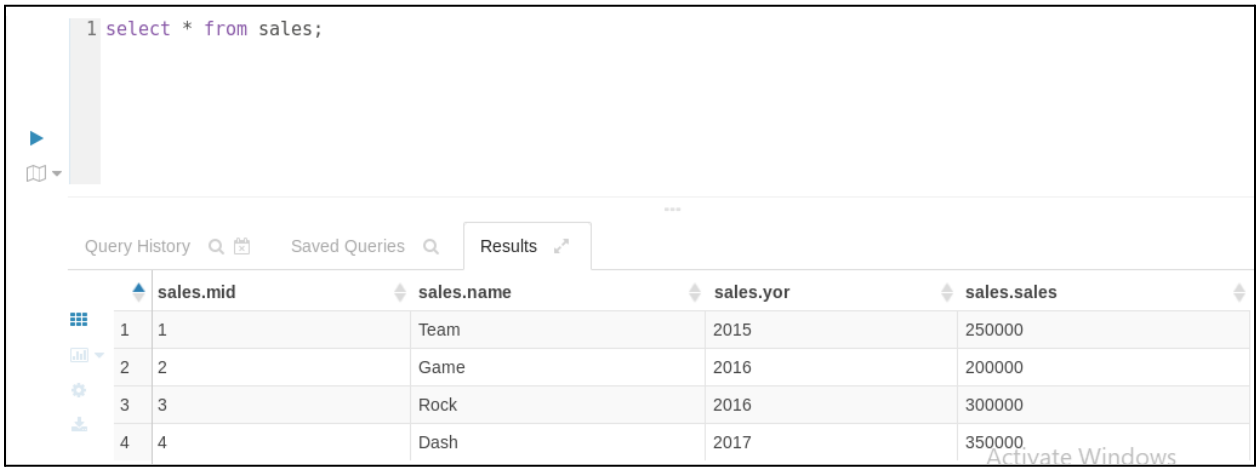
Open the Hue UI in the browser of the cloudera



a. desc sales;

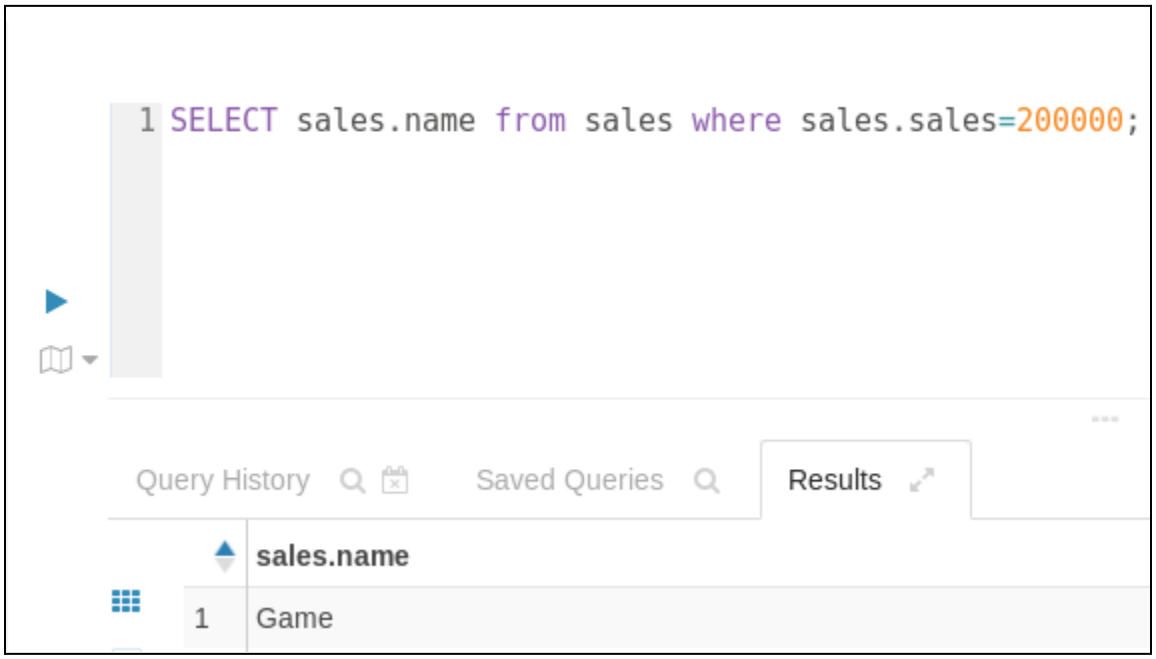


b. Select * from sales;

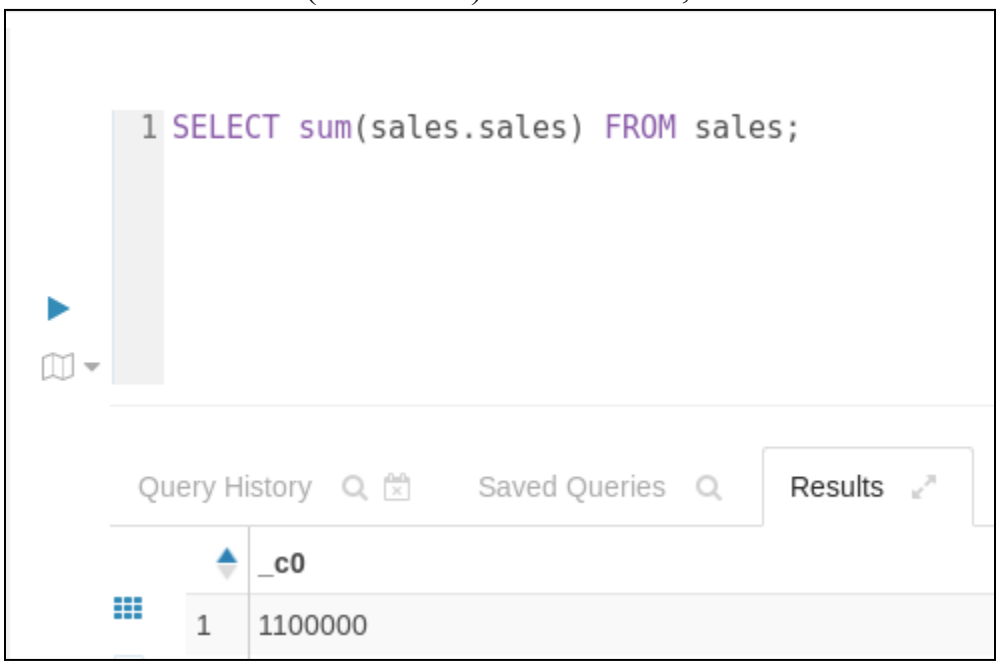


c. select count(mid), yor from sales GROUP BY yor;

d. SELECT sales.name from sales where sales.sales=200000;

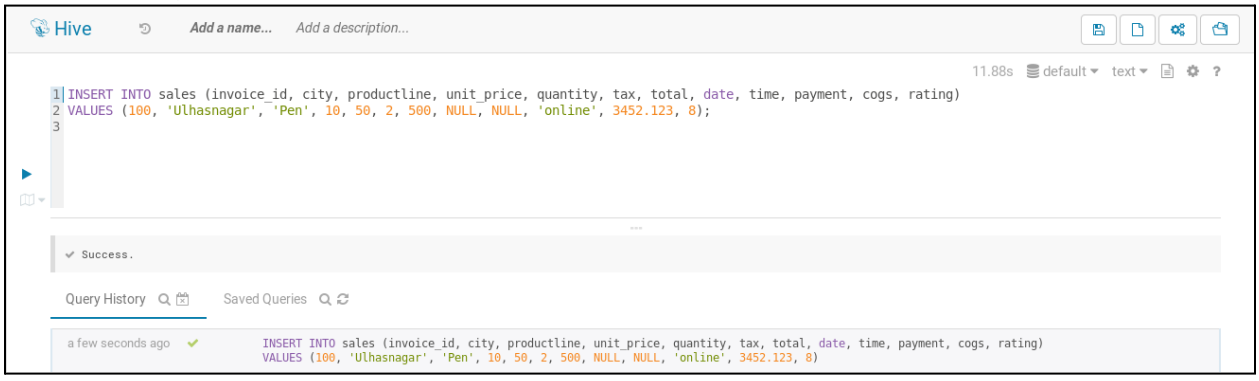


e. SELECT sum(sales.sales) FROM sales;

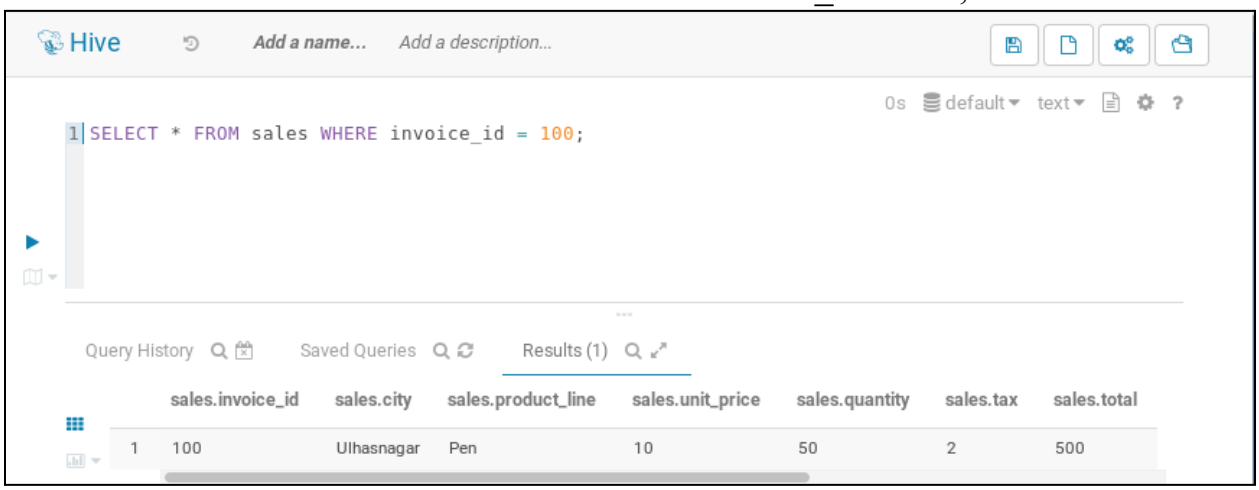


13. **Writing back to RDBMS:**

S1: INSERT INTO sales (invoice_id, city, productline, unit_price, quantity, tax, total, date, time, payment, cogs, rating)
VALUES (100, 'Ulhasnagar', 'Pen', 10, 50, 2, 500, NULL, NULL, 'online', 3452.123, 8);



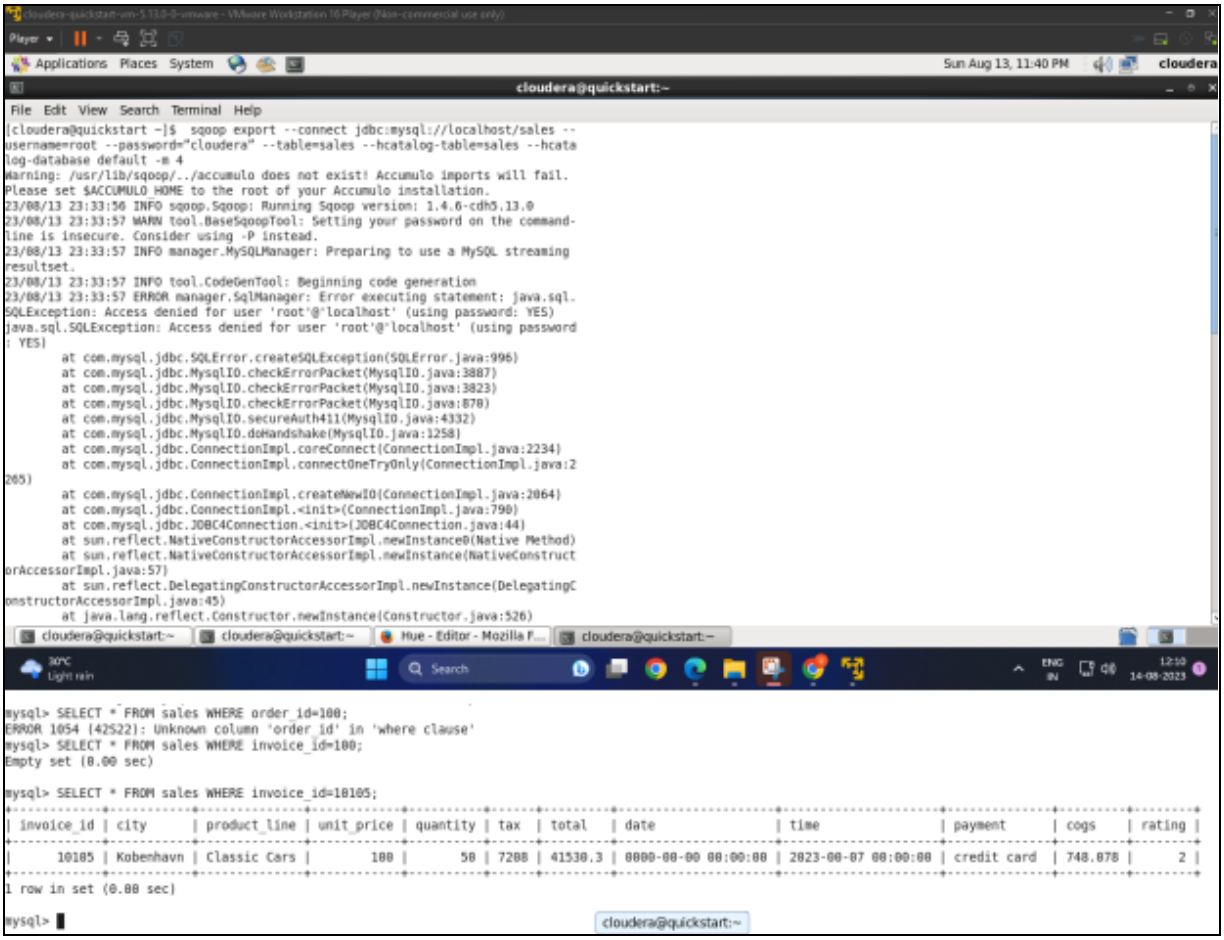
S2: Checking whether the tuple is added or not in the hive
Command: SELECT * FROM sales WHERE invoice_id = 100;



S3: Checking the data is not in mysql
Command: SELECT * FROM sales WHERE invoice_id = 100;

```
mysql> use sales;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM sales WHERE invoice_id=100;
Empty set (0.00 sec)
```


Applications Places System
Sun Aug 13, 11:40 PM
cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ sqoop export --connect jdbc:mysql://localhost/sales --username=root --password="cloudera" --table=sales --hcatalog-table=sales --hcatalog-database default -m 4
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
23/08/13 23:33:56 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
23/08/13 23:33:57 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
23/08/13 23:33:57 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
23/08/13 23:33:57 INFO tool.CodeGenTool: Beginning code generation
23/08/13 23:33:57 ERROR manager.SqlManager: Error executing statement: java.sql.SQLException: Access denied for user 'root'@'localhost' (using password: YES)
java.sql.SQLException: Access denied for user 'root'@'localhost' (using password: YES)
at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:996)
at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3887)
at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3823)
at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:870)
at com.mysql.jdbc.MysqlIO.secureAuth411(MysqlIO.java:4332)
at com.mysql.jdbc.MysqlIO.doHandshake(MysqlIO.java:1258)
at com.mysql.jdbc.ConnectionImpl.coreConnect(ConnectionImpl.java:2234)
at com.mysql.jdbc.ConnectionImpl.connectOneTryOnly(ConnectionImpl.java:2285)
at com.mysql.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:2064)
at com.mysql.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:790)
at com.mysql.jdbc.JDBC4Connection.<init>(JDBC4Connection.java:44)
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:57)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
at java.lang.reflect.Constructor.newInstance(Constructor.java:526)
cloudera@quickstart:~
cloudera@quickstart:~
Hue - Editor - Mozilla F...
cloudera@quickstart:~
30°C
Light rain
Search
ENG
IN
12:10
14-08-2023
mysql> SELECT * FROM sales WHERE order_id=100;
ERROR 1054 (42S22): Unknown column 'order_id' in 'where clause'
mysql> SELECT * FROM sales WHERE invoice_id=100;
Empty set (0.00 sec)
mysql> SELECT * FROM sales WHERE invoice_id=10105;
| invoice_id | city | product_line | unit_price | quantity | tax | total | date | time | payment | cogs | rating |
| 10105 | Kobenhavn | Classic Cars | 100 | 50 | 7208 | 41530.3 | 0000-00-00 00:00:00 | 2023-00-07 00:00:00 | credit card | 748.878 | 2 |
1 row in set (0.00 sec)
mysql>
cloudera@quickstart:~