EXPERIMENT NO: 6

AIM: Create HIVE database and descriptive analytics (basic statistics)

Theory:

Hive is a powerful data warehousing and SQL-like query language tool in the Hadoop ecosystem. Developed by Facebook and later open-sourced, Hive is designed to make it easier for users to query and analyze large datasets stored in Hadoop's distributed file system (HDFS) without requiring extensive programming skills. In this 400-word explanation, we'll explore the key features and use cases of Hive.

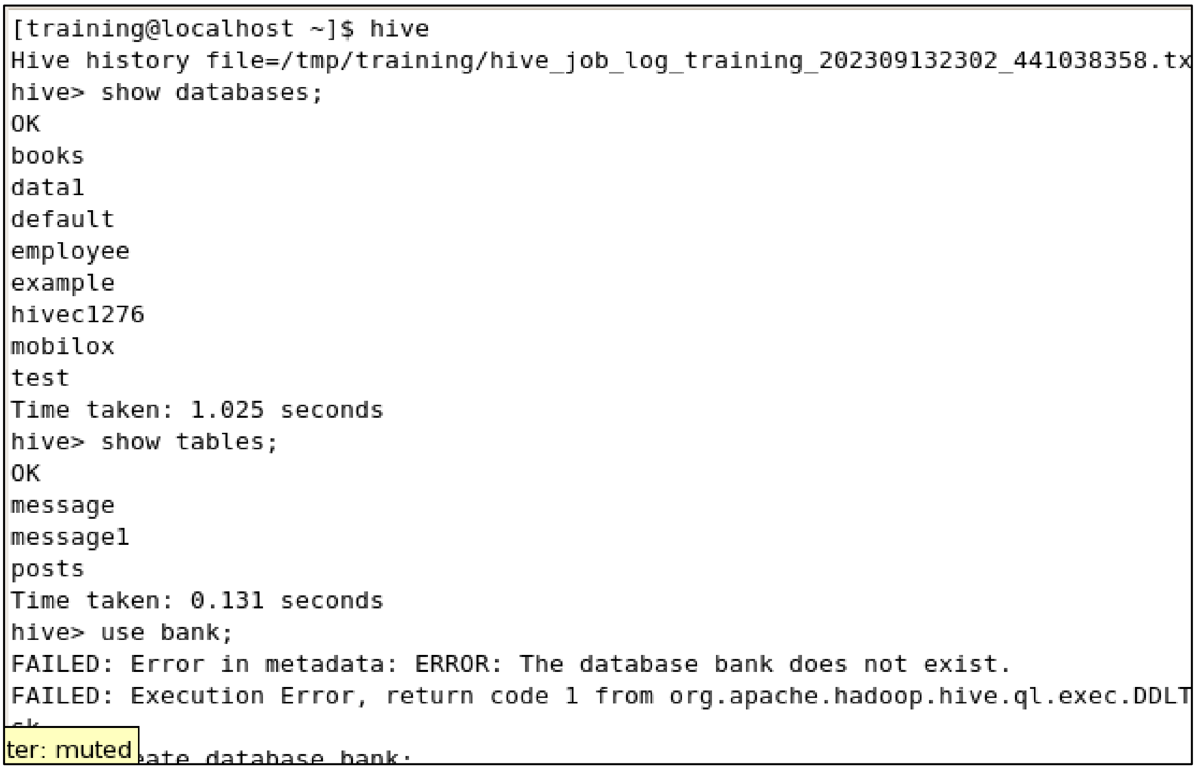
Key Features and Concepts:

1. **SQL-like Query Language**: Hive Query Language (HQL) resembles SQL, making it familiar to analysts and data scientists. Users can write queries to extract, transform, and analyze data in a manner similar to traditional relational databases.
2. **Schema-on-Read**: Unlike traditional databases with a strict schema-on-write, Hive follows a schema-on-read approach. This means that data is read from HDFS as is, and you can apply the schema when querying the data. This flexibility is particularly useful for handling semi-structured and unstructured data.
3. **Data Storage**: Hive organizes data into tables, databases, and partitions. It can store data in various formats, such as Text, Parquet, ORC, and more. This flexibility allows you to choose the best storage format for your data.
4. **Extensibility**: Hive supports custom user-defined functions (UDFs) and user-defined aggregates (UDAs) in multiple programming languages, such as Java, Python, and more, allowing you to perform complex data processing within your queries.
5. **Optimization**: Hive includes a query optimizer that can optimize query execution plans, improving query performance.
6. **Integration with Hadoop Ecosystem**: Hive seamlessly integrates with other Hadoop ecosystem components like HBase, Pig, and Spark, making it a valuable tool for a wide range of data processing tasks.

Hive Query Language (HQL):

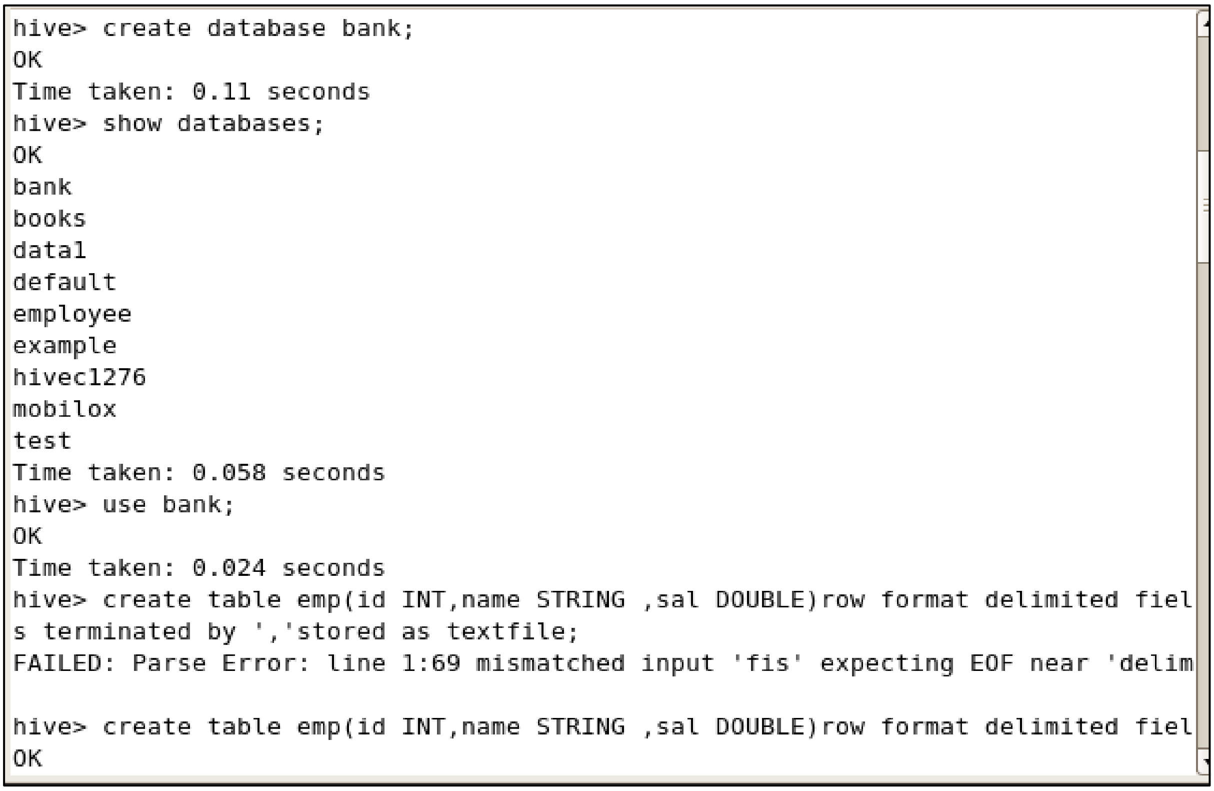
HQL is a SQL-like language used for querying and manipulating data in Hive. It includes familiar SQL commands like SELECT, JOIN, GROUP BY, and ORDER BY. However, it also incorporates some Hive-specific extensions to work with distributed data and the schema-on-read approach.

Step 1: starting with HIVE, viewing database and tables

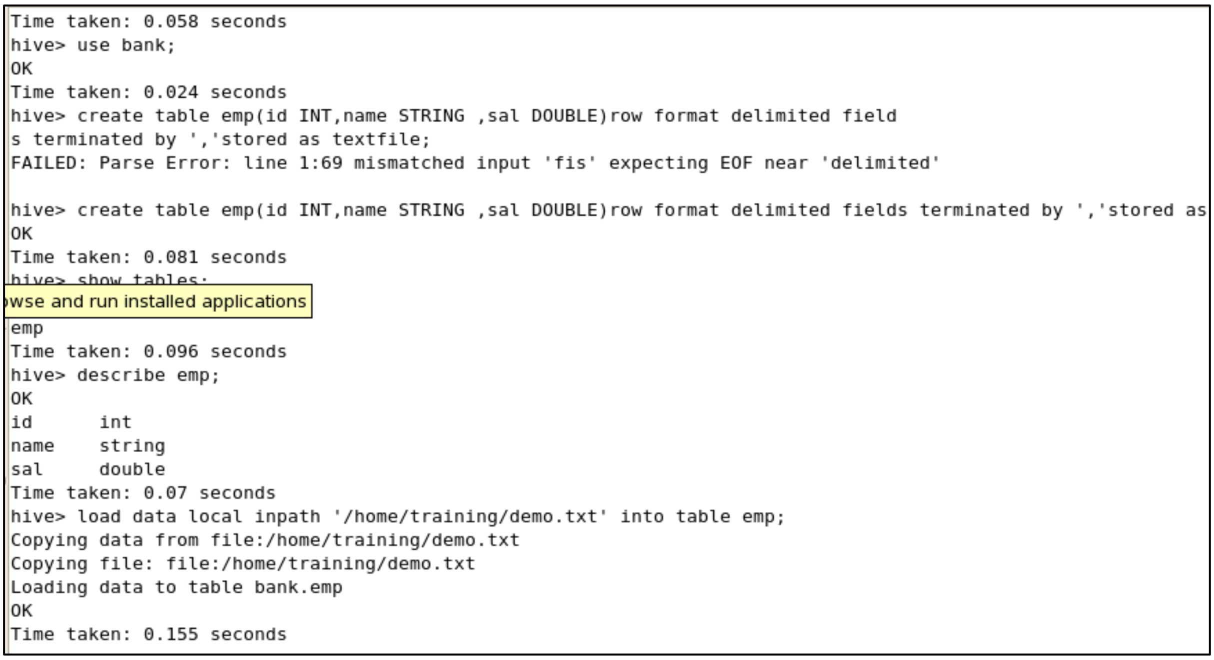


```
[training@localhost ~]$ hive
Hive history file=/tmp/training/hive_job_log_training_202309132302_441038358.tx
hive> show databases;
OK
books
data1
default
employee
example
hivec1276
mobilox
test
Time taken: 1.025 seconds
hive> show tables;
OK
message
message1
posts
Time taken: 0.131 seconds
hive> use bank;
FAILED: Error in metadata: ERROR: The database bank does not exist.
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLT
sk
ter: muted ate database bank;
```

Step 2: creating database and bank



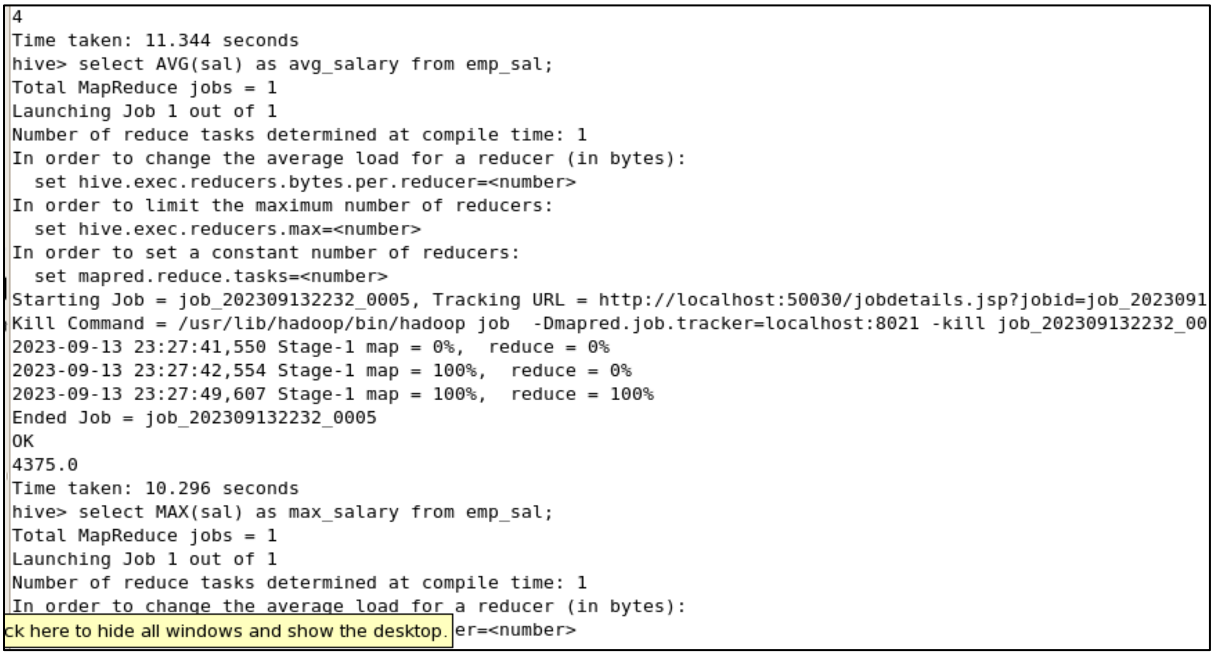
```
hive> create database bank;
OK
Time taken: 0.11 seconds
hive> show databases;
OK
bank
books
data1
default
employee
example
hivec1276
mobilox
test
Time taken: 0.058 seconds
hive> use bank;
OK
Time taken: 0.024 seconds
hive> create table emp(id INT,name STRING ,sal DOUBLE)row format delimited fiel
s terminated by ','stored as textfile;
FAILED: Parse Error: line 1:69 mismatched input 'fis' expecting EOF near 'delim

hive> create table emp(id INT,name STRING ,sal DOUBLE)row format delimited fiel
OK
```



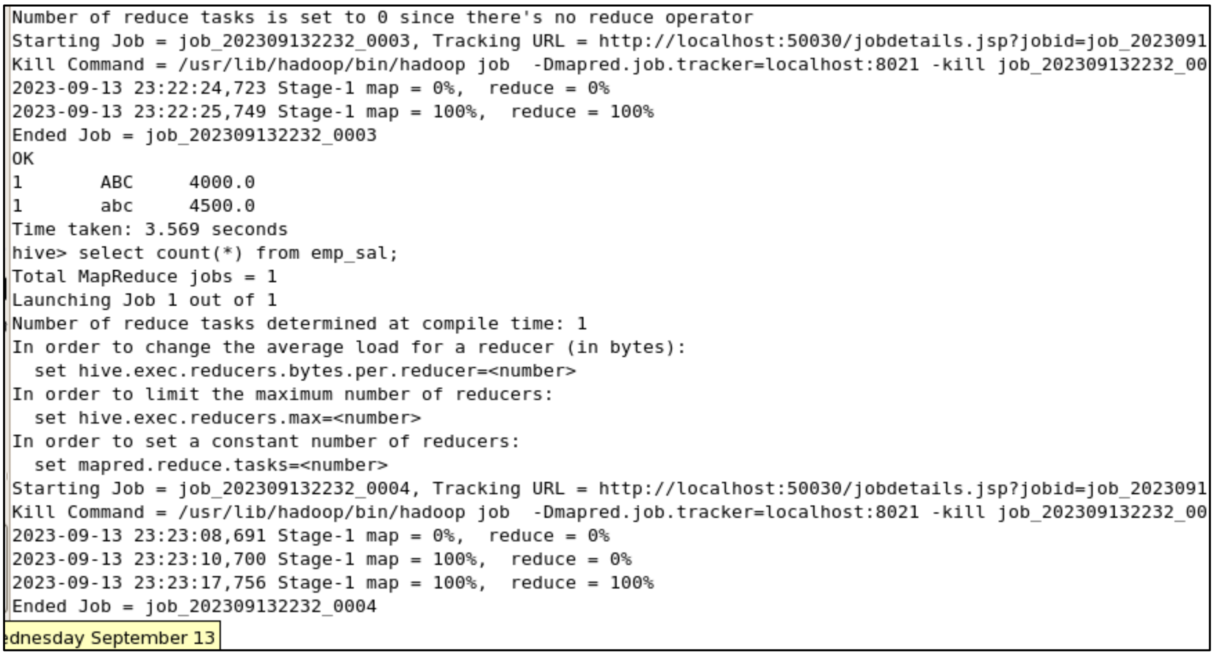
```
Time taken: 0.058 seconds
hive> use bank;
OK
Time taken: 0.024 seconds
hive> create table emp(id INT,name STRING ,sal DOUBLE)row format delimited field
s terminated by ','stored as textfile;
FAILED: Parse Error: line 1:69 mismatched input 'fis' expecting EOF near 'delimited'

hive> create table emp(id INT,name STRING ,sal DOUBLE)row format delimited fields terminated by ','stored as
OK
Time taken: 0.081 seconds
hive> show tables;
wse and run installed applications
emp
Time taken: 0.096 seconds
hive> describe emp;
OK
id      int
name    string
sal     double
Time taken: 0.07 seconds
hive> load data local inpath '/home/training/demo.txt' into table emp;
Copying data from file:/home/training/demo.txt
Copying file: file:/home/training/demo.txt
Loading data to table bank.emp
OK
Time taken: 0.155 seconds
```

Step 3: performing queries on the table

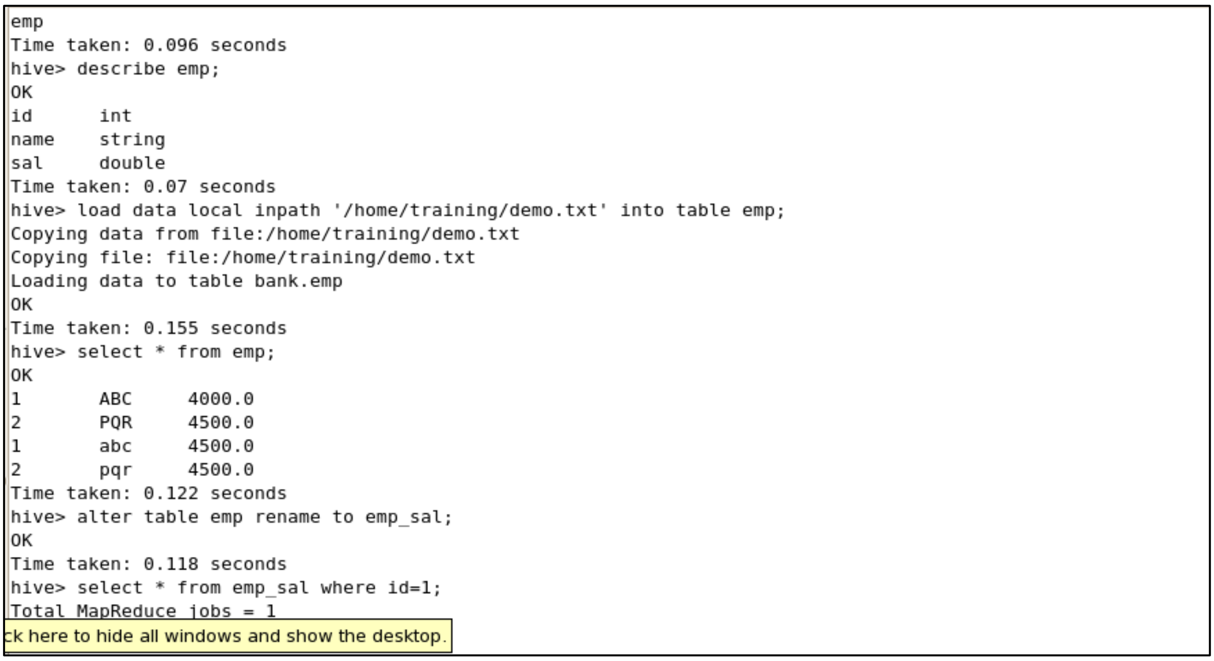
Various statistical queries like AVG () , MAX() atc



```
4
Time taken: 11.344 seconds
hive> select AVG(sal) as avg_salary from emp_sal;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_202309132232_0005, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_2023091
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:8021 -kill job_202309132232_00
2023-09-13 23:27:41,550 Stage-1 map = 0%,  reduce = 0%
2023-09-13 23:27:42,554 Stage-1 map = 100%,  reduce = 0%
2023-09-13 23:27:49,607 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_202309132232_0005
OK
4375.0
Time taken: 10.296 seconds
hive> select MAX(sal) as max_salary from emp_sal;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
ck here to hide all windows and show the desktop.er=<number>
```



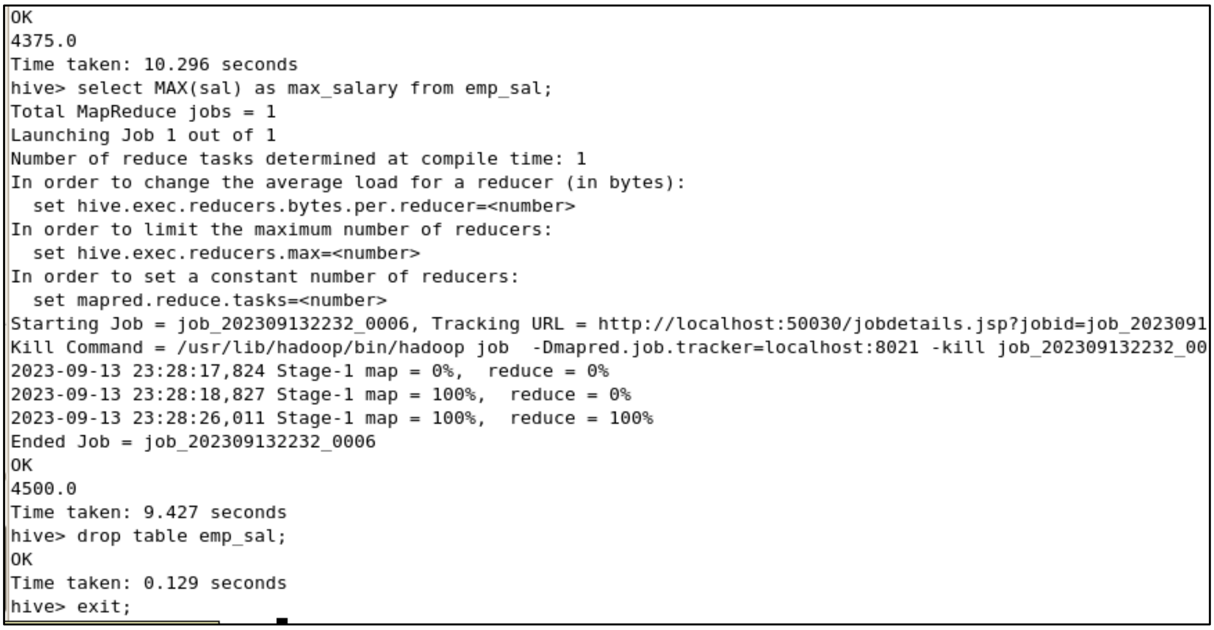
```
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_202309132232_0003, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_2023091
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:8021 -kill job_202309132232_00
2023-09-13 23:22:24,723 Stage-1 map = 0%,  reduce = 0%
2023-09-13 23:22:25,749 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_202309132232_0003
OK
1       ABC     4000.0
1       abc     4500.0
Time taken: 3.569 seconds
hive> select count(*) from emp_sal;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_202309132232_0004, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_2023091
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:8021 -kill job_202309132232_00
2023-09-13 23:23:08,691 Stage-1 map = 0%,  reduce = 0%
2023-09-13 23:23:10,700 Stage-1 map = 100%,  reduce = 0%
2023-09-13 23:23:17,756 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_202309132232_0004
```



```
emp
Time taken: 0.096 seconds
hive> describe emp;
OK
id      int
name    string
sal     double
Time taken: 0.07 seconds
hive> load data local inpath '/home/training/demo.txt' into table emp;
Copying data from file:/home/training/demo.txt
Copying file: file:/home/training/demo.txt
Loading data to table bank.emp
OK
Time taken: 0.155 seconds
hive> select * from emp;
OK
1       ABC     4000.0
2       PQR     4500.0
1       abc     4500.0
2       pqr     4500.0
Time taken: 0.122 seconds
hive> alter table emp rename to emp_sal;
OK
Time taken: 0.118 seconds
hive> select * from emp_sal where id=1;
Total MapReduce jobs = 1
```

Step 4:

EXIT from hive



```
OK
4375.0
Time taken: 10.296 seconds
hive> select MAX(sal) as max_salary from emp_sal;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_202309132232_0006, Tracking URL = http://localhost:50030/jobdetails.jsp?jobid=job_2023091
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=localhost:8021 -kill job_202309132232_00
2023-09-13 23:28:17,824 Stage-1 map = 0%,  reduce = 0%
2023-09-13 23:28:18,827 Stage-1 map = 100%,  reduce = 0%
2023-09-13 23:28:26,011 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_202309132232_0006
OK
4500.0
Time taken: 9.427 seconds
hive> drop table emp_sal;
OK
Time taken: 0.129 seconds
hive> exit;
```