NAME: sahil shaikh Mushtaq Ahmed
ROLL NO: 47

Experiment 7: Part 2 – Joins, Sorting, Subqueries using

JOINS

JOIN is a clause that is used for combining specific fields from two tables by using values common to each one. It is used to combine records from two or more tables in the database.

There are different types of joins given as follows:

- JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

JOIN

JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.

LEFT OUTER JOIN

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.

A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

RIGHT OUTER JOIN

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.

- **FULL OUTER JOIN**

  The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer tables that fulfil the JOIN condition. The joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side.

SUB QUERIES:

A Query present within a Query is known as a sub query. The main query will depend on the values returned by the subqueries.

Subqueries can be classified into two types

- Subqueries in FROM clause
- Subqueries in WHERE clause **When to use:**

- To get a particular value combined from two column values from different tables
- Dependency of one table values on other tables
- Comparative checking of one column values from other tables

SORTING

The SORT BY syntax is similar to the syntax of **ORDER BY** in SQL language.

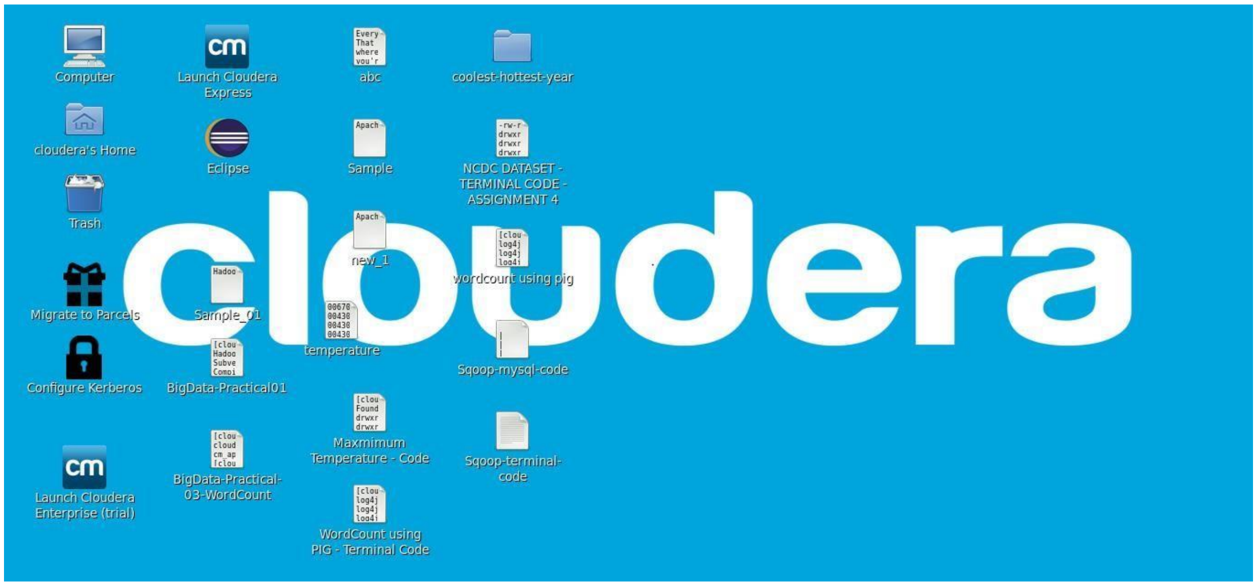
Hive supports **SORT BY** which sorts the data per reducer. The difference between "order by" and "sort by" is that the former guarantees total order in the output while the latter only guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by" may give partially ordered final results.

Hive uses the columns in SORT BY to sort the rows before feeding the rows to a reducer. The sort order will be dependent on the column types. If the column is of numeric type, then the sort order is also in numeric order. If the column is of string type, then the sort order will be lexicographical order.

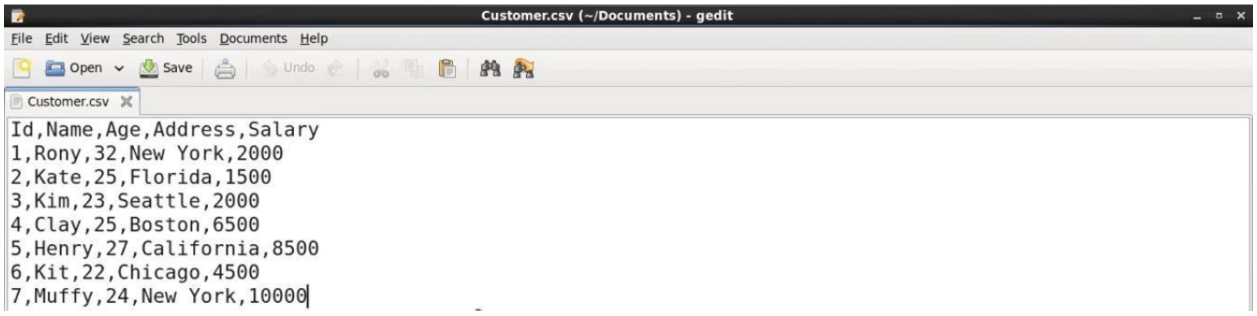
Steps: Joins, Sorting, Subqueries using HiveQL

1. Open the cloudera.

NAME: sahil shaikh Mushtaq Ahmed
ROLL NO: 47

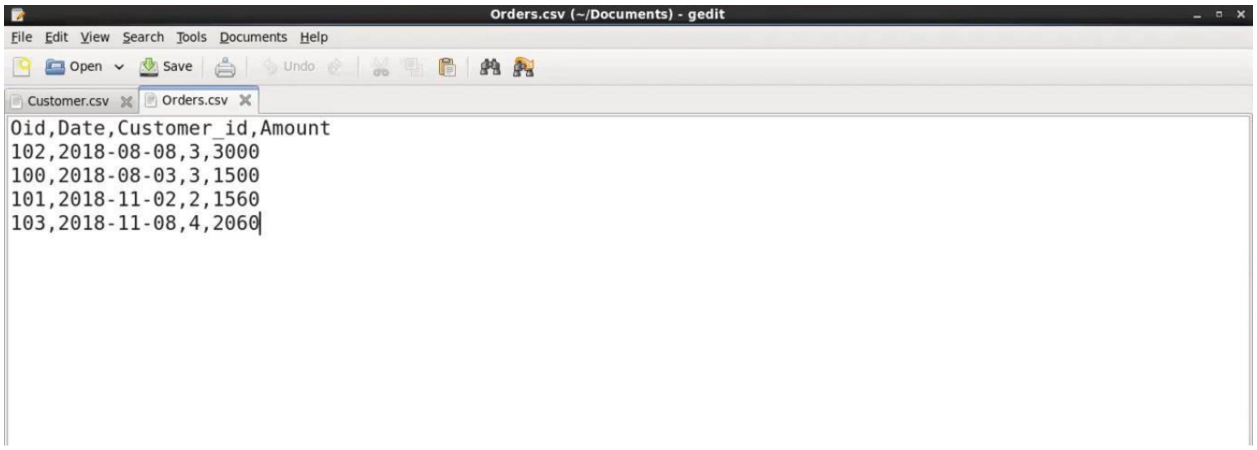


2. First we will create the **Customer.csv** file.


Customer.csv (~/Documents) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
Customer.csv

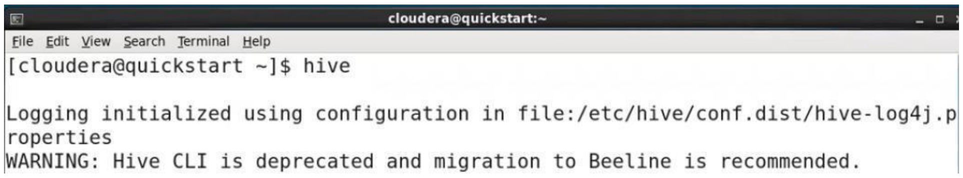
```
Id,Name,Age,Address,Salary
1,Rony,32,New York,2000
2,Kate,25,Florida,1500
3,Kim,23,Seattle,2000
4,Clay,25,Boston,6500
5,Henry,27,California,8500
6,Kit,22,Chicago,4500
7,Muffy,24,New York,10000
```

3. Then creating **Orders.csv** file.


Orders.csv (~/Documents) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
Customer.csv
Orders.csv

```
Oid,Date,Customer_id,Amount
102,2018-08-08,3,3000
100,2018-08-03,3,1500
101,2018-11-02,2,1560
103,2018-11-08,4,2060
```

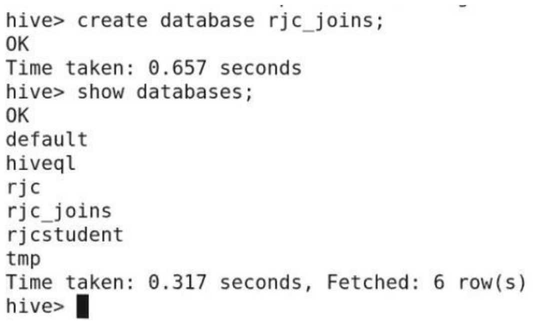
4. Open the terminal, Now we use **hive** command to enter the **hive shell prompt** and in hive shell we could execute all of the hive commands.



```
[cloudera@quickstart ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.p
roperties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
```

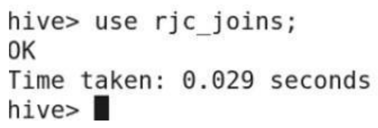
5. Now we will be creating a new database named as rjc_joins using below command, **create database rjc_joins;** And then showing the databases. **show databases;**



```
hive> create database rjc_joins;
OK
Time taken: 0.657 seconds
hive> show databases;
OK
default
hiveql
rjc
rjc_joins
rjcstudent
tmp
Time taken: 0.317 seconds, Fetched: 6 row(s)
hive> 
```

As we can see **rjc_joins** database is created.

6. Now to work inside this database we use below command; **use rjc_joins;**

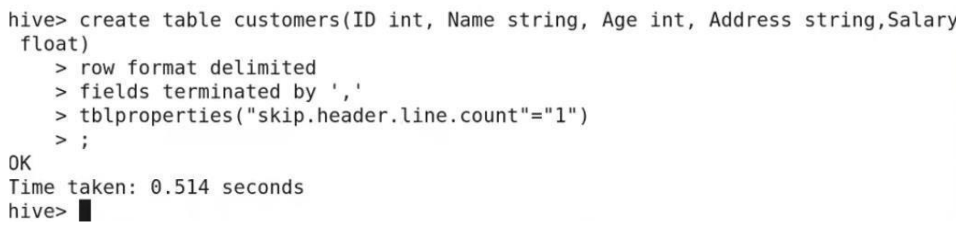


```
hive> use rjc_joins;
OK
Time taken: 0.029 seconds
hive> 
```

7. Now we will create two tables in one table we will load the **Customer.csv** file and in the other table we will load **Orders.csv** file.

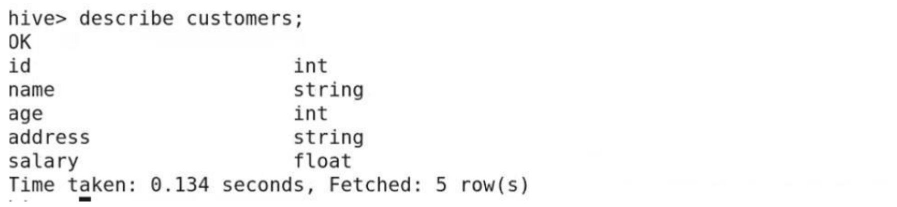
create table customers(ID int, Name string, Age int, Address string, Salary float)

- **row format delimited**
- **fields terminated by ‘,’**
- **tblproperties(“skip.header.line.count” =”1”);**



```
hive> create table customers(ID int, Name string, Age int, Address string,Salary
 float)
    > row format delimited
    > fields terminated by ','
    > tblproperties("skip.header.line.count"="1")
    > ;
OK
Time taken: 0.514 seconds
hive> 
```

Now we will see the schema of the table using describe command, **describe customers;**

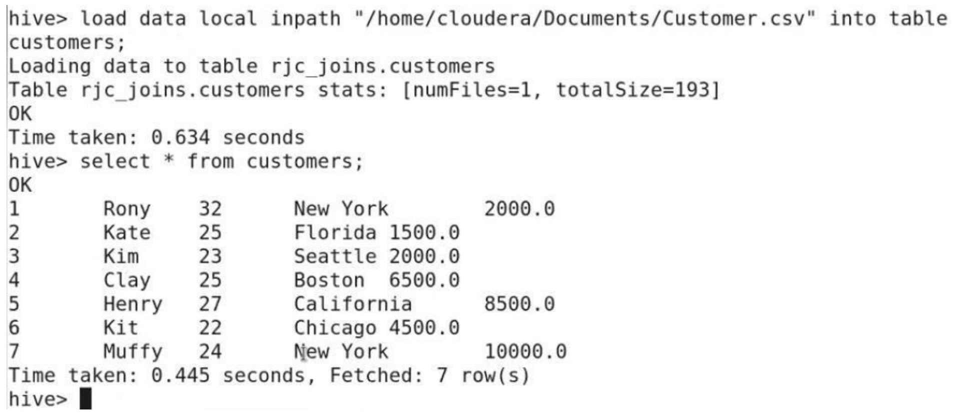


```
hive> describe customers;
OK
id                      int
name                    string
age                     int
address                 string
salary                  float
Time taken: 0.134 seconds, Fetched: 5 row(s)
```

Now loading data in the **customers** table from **Customer.csv** file which present inside /home/cloudera/Documents directory.

load data local inpath '/home/cloudera/Documents/Customer.csv' into table customers;

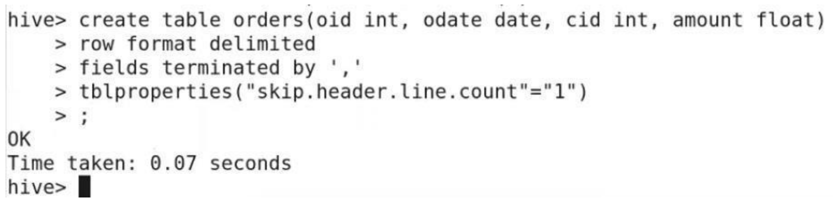
Select * from customers;



```
hive> load data local inpath "/home/cloudera/Documents/Customer.csv" into table
customers;
Loading data to table rjc_joins.customers
Table rjc_joins.customers stats: [numFiles=1, totalSize=193]
OK
Time taken: 0.634 seconds
hive> select * from customers;
OK
1       Rony    32      New York        2000.0
2       Kate    25      Florida 1500.0
3       Kim     23      Seattle 2000.0
4       Clay    25      Boston  6500.0
5       Henry   27      California      8500.0
6       Kit     22      Chicago 4500.0
7       Muffy   24      New York        10000.0
Time taken: 0.445 seconds, Fetched: 7 row(s)
hive> 
```

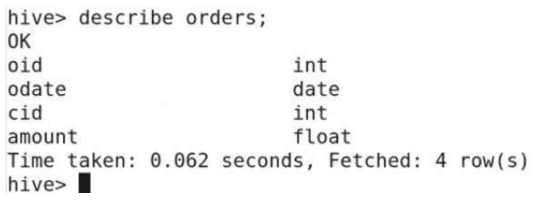
8. Creating a second table named as **orders** using below command, **create table orders(oid int, odate date, cid int, amount float)**

- **row format delimited**
- **fields terminated by ','**
- **tblproperties("skip.header.line.count" ="1");**



```
hive> create table orders(oid int, odate date, cid int, amount float)
    > row format delimited
    > fields terminated by ','
    > tblproperties("skip.header.line.count"="1")
    > ;
OK
Time taken: 0.07 seconds
hive> █
```

Now we will see the schema of the table using describe command, **describe orders;**

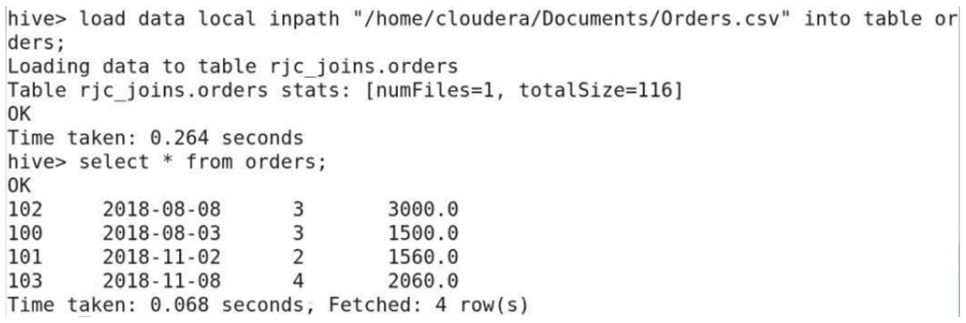


```
hive> describe orders;
OK
oid                     int
odate                   date
cid                     int
amount                  float
Time taken: 0.062 seconds, Fetched: 4 row(s)
hive> █
```

Now loading data in the **orders** table from **Orders.csv** file which present inside /home/cloudera/Documents directory.

load data local inpath '/home/cloudera/Documents/Orders.csv' into table orders;

Select * from orders;



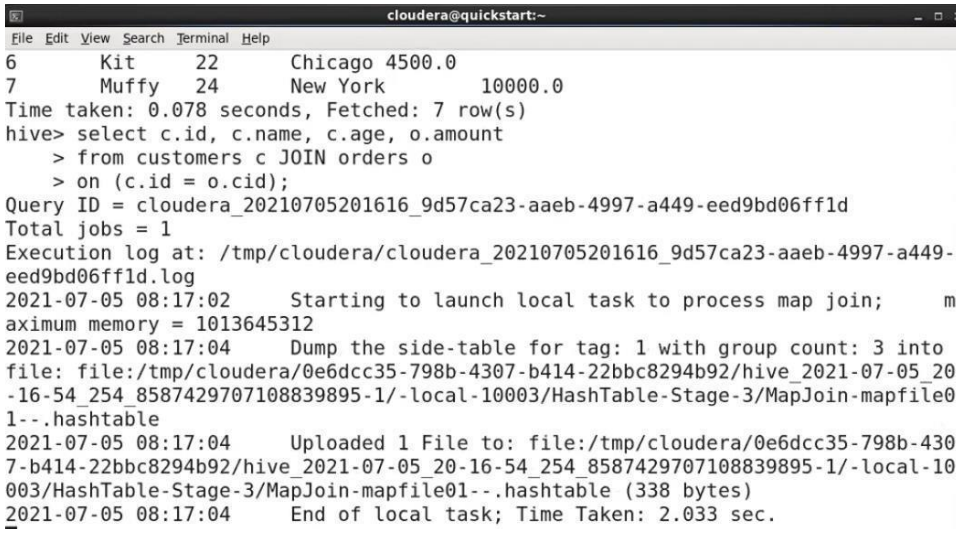
```
hive> load data local inpath "/home/cloudera/Documents/Orders.csv" into table or
ders;
Loading data to table rjc_joins.orders
Table rjc_joins.orders stats: [numFiles=1, totalSize=116]
OK
Time taken: 0.264 seconds
hive> select * from orders;
OK
102     2018-08-08      3       3000.0
100     2018-08-03      3       1500.0
101     2018-11-02      2       1560.0
103     2018-11-08      4       2060.0
Time taken: 0.068 seconds, Fetched: 4 row(s)
```

9. Join:

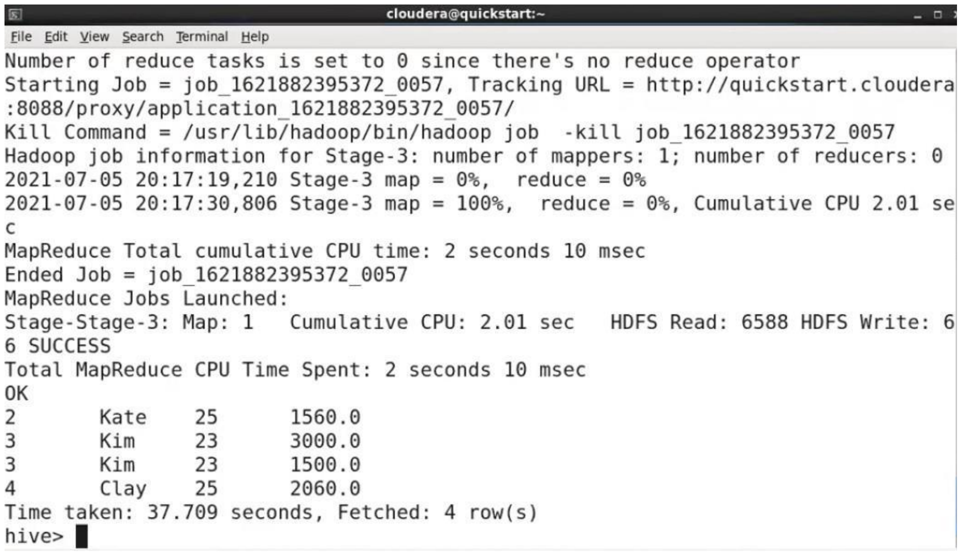
Now First we apply the normal joins on the two tables using below command, we want to retrieve customer id, name, age from customers table and amount from the orders table and join perform on id of the customers and orders table.

select c.id, c.name, c.age, o.amount
- **from customers c JOIN orders o**
- **on (c.id = o.cid);**

Mapreduce task is performed



```
6       Kit     22      Chicago 4500.0
7       Muffy   24      New York        10000.0
Time taken: 0.078 seconds, Fetched: 7 row(s)
hive> select c.id, c.name, c.age, o.amount
    > from customers c JOIN orders o
    > on (c.id = o.cid);
Query ID = cloudera_20210705201616_9d57ca23-aaeb-4997-a449-eed9bd06ff1d
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20210705201616_9d57ca23-aaeb-4997-a449-
eed9bd06ff1d.log
2021-07-05 08:17:02     Starting to launch local task to process map join;     m
aximum memory = 1013645312
2021-07-05 08:17:04     Dump the side-table for tag: 1 with group count: 3 into
file: file:/tmp/cloudera/0e6dcc35-798b-4307-b414-22bbc8294b92/hive_2021-07-05_20
-16-54_254_8587429707108839895-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile0
1--.hashtable
2021-07-05 08:17:04     Uploaded 1 File to: file:/tmp/cloudera/0e6dcc35-798b-430
7-b414-22bbc8294b92/hive_2021-07-05_20-16-54_254_8587429707108839895-1/-local-10
003/HashTable-Stage-3/MapJoin-mapfile01--.hashtable (338 bytes)
2021-07-05 08:17:04     End of local task; Time Taken: 2.033 sec.
```



```
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1621882395372_0057, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1621882395372_0057/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1621882395372_0057
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2021-07-05 20:17:19,210 Stage-3 map = 0%,  reduce = 0%
2021-07-05 20:17:30,806 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 2.01 se
c
MapReduce Total cumulative CPU time: 2 seconds 10 msec
Ended Job = job_1621882395372_0057
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 2.01 sec   HDFS Read: 6588 HDFS Write: 6
6 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 10 msec
OK
2       Kate    25      1560.0
3       Kim     23      3000.0
3       Kim     23      1500.0
4       Clay    25      2060.0
Time taken: 37.709 seconds, Fetched: 4 row(s)
hive> 
```

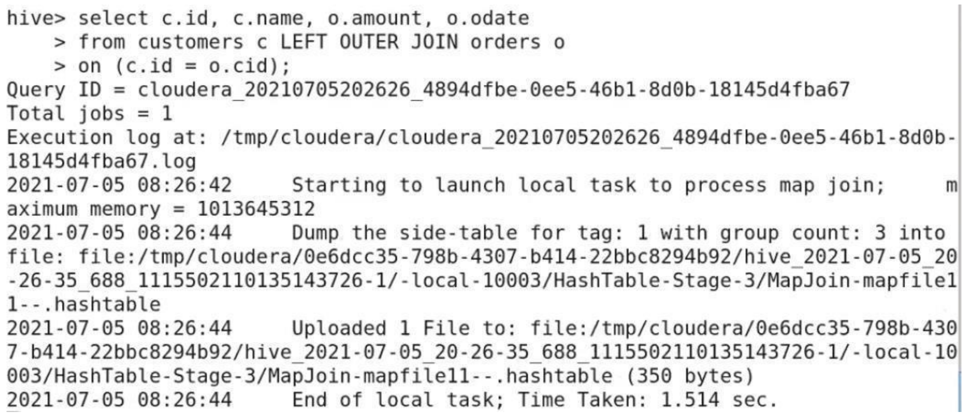
10. LEFT OUTER JOIN

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.

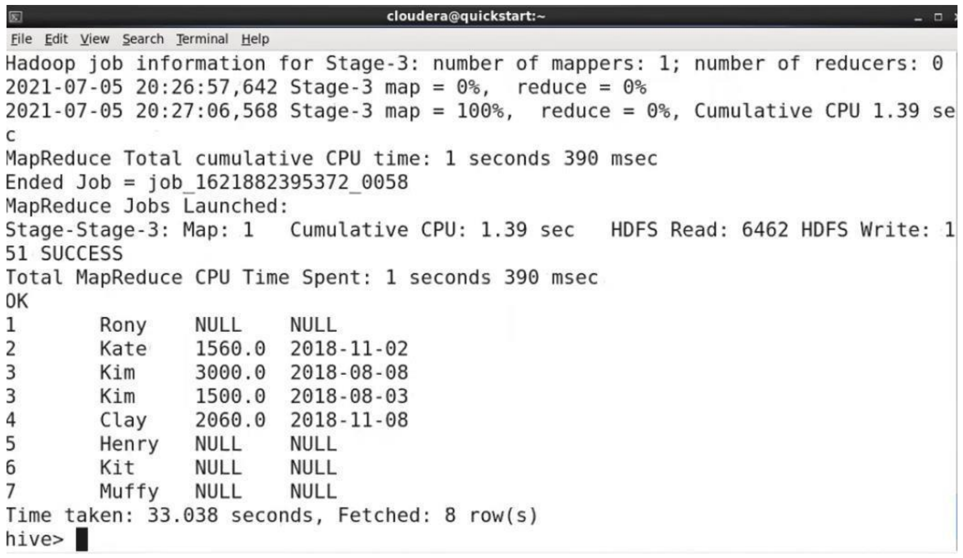
A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

select c.id, c.name, o.amount, o.odate
- **from customers c LEFT OUTER JOIN orders o**
- **on (c.id = o.cid);**

Mapreduce task is performed



```
hive> select c.id, c.name, o.amount, o.odate
    > from customers c LEFT OUTER JOIN orders o
    > on (c.id = o.cid);
Query ID = cloudera_20210705202626_4894dfbe-0ee5-46b1-8d0b-18145d4fba67
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20210705202626_4894dfbe-0ee5-46b1-8d0b-
18145d4fba67.log
2021-07-05 08:26:42     Starting to launch local task to process map join;     m
aximum memory = 1013645312
2021-07-05 08:26:44     Dump the side-table for tag: 1 with group count: 3 into
file: file:/tmp/cloudera/0e6dcc35-798b-4307-b414-22bbc8294b92/hive_2021-07-05_20
-26-35_688_1115502110135143726-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile1
1--.hashtable
2021-07-05 08:26:44     Uploaded 1 File to: file:/tmp/cloudera/0e6dcc35-798b-430
7-b414-22bbc8294b92/hive_2021-07-05_20-26-35_688_1115502110135143726-1/-local-10
003/HashTable-Stage-3/MapJoin-mapfile11--.hashtable (350 bytes)
2021-07-05 08:26:44     End of local task; Time Taken: 1.514 sec.
```



```
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2021-07-05 20:26:57,642 Stage-3 map = 0%,  reduce = 0%
2021-07-05 20:27:06,568 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 1.39 se
c
MapReduce Total cumulative CPU time: 1 seconds 390 msec
Ended Job = job_1621882395372_0058
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 1.39 sec   HDFS Read: 6462 HDFS Write: 1
51 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 390 msec
OK
1       Rony    NULL    NULL
2       Kate    1560.0  2018-11-02
3       Kim     3000.0  2018-08-08
3       Kim     1500.0  2018-08-03
4       Clay    2060.0  2018-11-08
5       Henry   NULL    NULL
6       Kit     NULL    NULL
7       Muffy   NULL    NULL
Time taken: 33.038 seconds, Fetched: 8 row(s)
hive> 
```

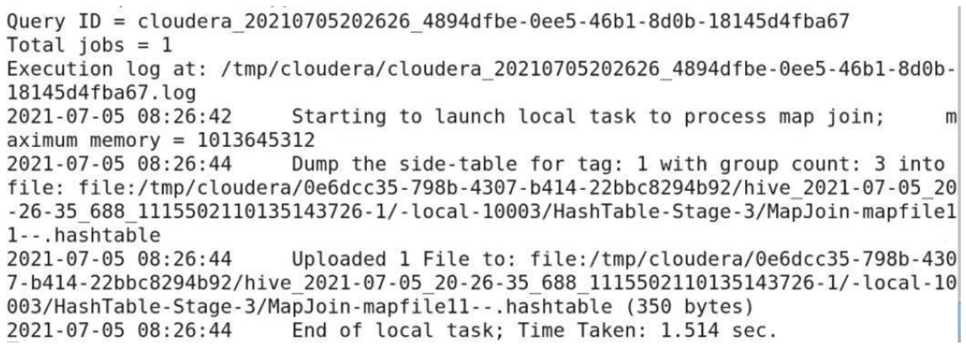
11. RIGHT OUTER JOIN

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.

select c.id, c.name, o.amount, o.odate
- **from customers c RIGHT OUTER JOIN orders o**
- **on (c.id = o.cid);**

Mapreduce task is performed



```
Query ID = cloudera_20210705202626_4894dfbe-0ee5-46b1-8d0b-18145d4fba67
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20210705202626_4894dfbe-0ee5-46b1-8d0b-
18145d4fba67.log
2021-07-05 08:26:42     Starting to launch local task to process map join;     m
aximum memory = 1013645312
2021-07-05 08:26:44     Dump the side-table for tag: 1 with group count: 3 into
file: file:/tmp/cloudera/0e6dcc35-798b-4307-b414-22bbc8294b92/hive_2021-07-05_20
-26-35_688_1115502110135143726-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile1
1--.hashtable
2021-07-05 08:26:44     Uploaded 1 File to: file:/tmp/cloudera/0e6dcc35-798b-430
7-b414-22bbc8294b92/hive_2021-07-05_20-26-35_688_1115502110135143726-1/-local-10
003/HashTable-Stage-3/MapJoin-mapfile11--.hashtable (350 bytes)
2021-07-05 08:26:44     End of local task; Time Taken: 1.514 sec.
```

12. Now we will be using the concept of **subqueries** for finding the second largest salary from the customers table.

<u>Sub queries:</u>

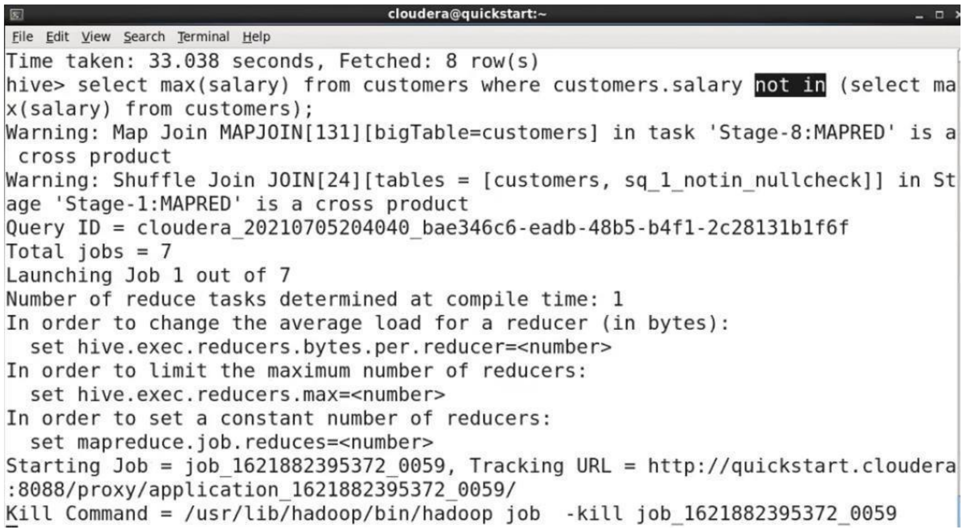
A Query present within a Query is known as a sub query. The main query will depend on the values returned by the subqueries.

Subqueries can be classified into two types

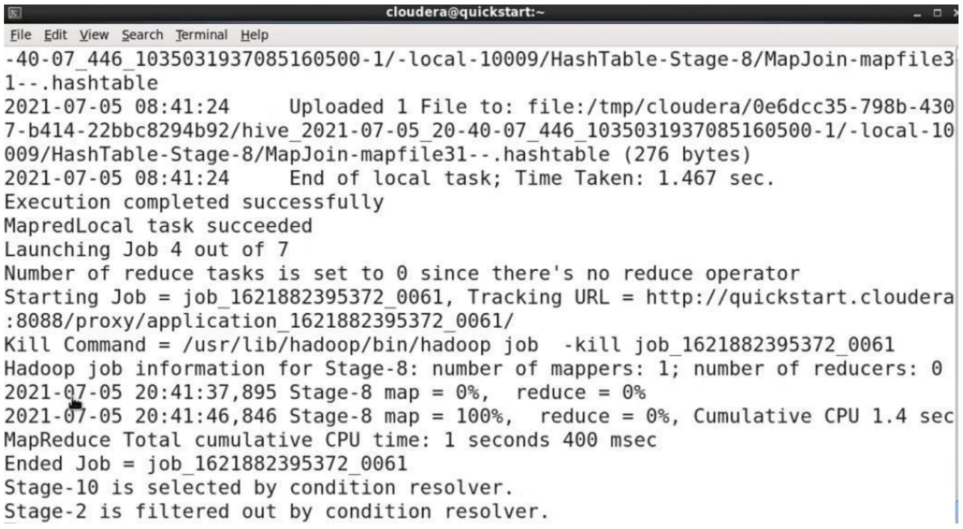
- Subqueries in FROM clause
- Subqueries in WHERE clause

Select max(salary) from customers where customers.salary not in(select max(salary) from customers);

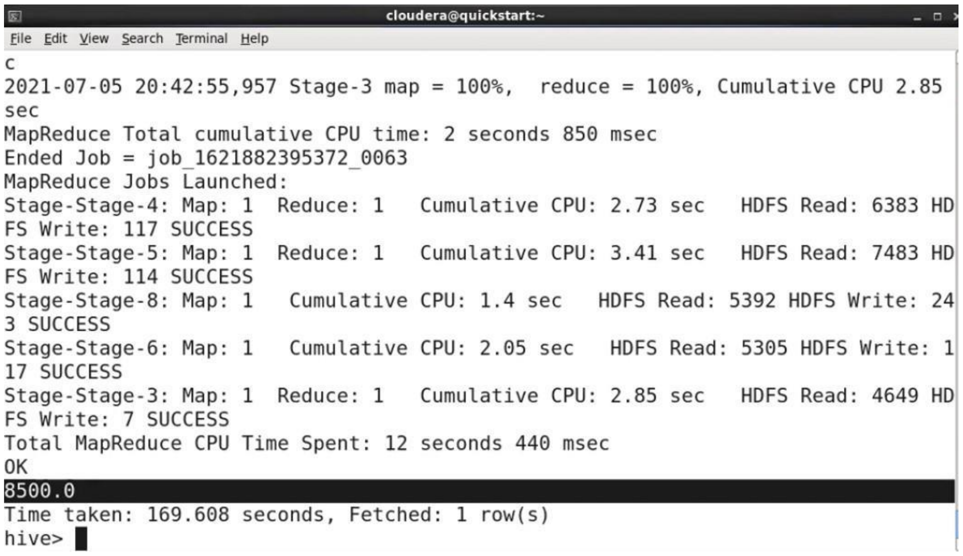
Mapreduce task is performed



```
Time taken: 33.038 seconds, Fetched: 8 row(s)
hive> select max(salary) from customers where customers.salary not in (select ma
x(salary) from customers);
Warning: Map Join MAPJOIN[131][bigTable=customers] in task 'Stage-8:MAPRED' is a
 cross product
Warning: Shuffle Join JOIN[24][tables = [customers, sq_1_notin_nullcheck]] in St
age 'Stage-1:MAPRED' is a cross product
Query ID = cloudera_20210705204040_bae346c6-eadb-48b5-b4f1-2c28131b1f6f
Total jobs = 7
Launching Job 1 out of 7
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 mapreduce.job.reduces=<number>
Starting Job = job_1621882395372_0059, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1621882395372_0059/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1621882395372_0059
```



```
-40-07_446_1035031937085160500-1/-local-10009/HashTable-Stage-8/MapJoin-mapfile3
1--.hashtable
2021-07-05 08:41:24     Uploaded 1 File to: file:/tmp/cloudera/0e6dcc35-798b-430
7-b414-22bbc8294b92/hive_2021-07-05_20-40-07_446_1035031937085160500-1/-local-10
009/HashTable-Stage-8/MapJoin-mapfile31--.hashtable (276 bytes)
2021-07-05 08:41:24     End of local task; Time Taken: 1.467 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 4 out of 7
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1621882395372_0061, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1621882395372_0061/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1621882395372_0061
Hadoop job information for Stage-8: number of mappers: 1; number of reducers: 0
2021-07-05 20:41:37,895 Stage-8 map = 0%,  reduce = 0%
2021-07-05 20:41:46,846 Stage-8 map = 100%,  reduce = 0%, Cumulative CPU 1.4 sec
MapReduce Total cumulative CPU time: 1 seconds 400 msec
Ended Job = job_1621882395372_0061
Stage-10 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
```



```
c
2021-07-05 20:42:55,957 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 2.85
sec
MapReduce Total cumulative CPU time: 2 seconds 850 msec
Ended Job = job_1621882395372_0063
MapReduce Jobs Launched:
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 2.73 sec   HDFS Read: 6383 HD
FS Write: 117 SUCCESS
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 3.41 sec   HDFS Read: 7483 HD
FS Write: 114 SUCCESS
Stage-Stage-8: Map: 1   Cumulative CPU: 1.4 sec   HDFS Read: 5392 HDFS Write: 24
3 SUCCESS
Stage-Stage-6: Map: 1   Cumulative CPU: 2.05 sec   HDFS Read: 5305 HDFS Write: 1
17 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.85 sec   HDFS Read: 4649 HD
FS Write: 7 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 440 msec
OK
8500.0
Time taken: 169.608 seconds, Fetched: 1 row(s)
hive>
```

As we can see from the above output the second largest salary is 8500.

13. <u>Sorting</u>

The SORT BY syntax is similar to the syntax of **ORDER BY** in SQL language.

Hive supports **SORT BY** which sorts the data per reducer. The difference between "order by" and "sort by" is that the former guarantees total order in the output while the latter only guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by" may give partially ordered final results.

Hive uses the columns in SORT BY to sort the rows before feeding the rows to a reducer. The sort order will be dependent on the column types. If the column is of numeric type, then the sort order is also in numeric order. If the column is of string type, then the sort order will be lexicographical order.

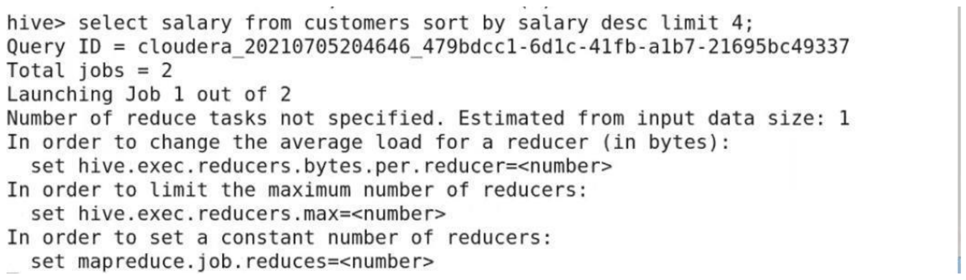
LIMIT can be used to minimize sort time.

Now finding the **fourth largest salary** from the customers table using **Sort by** clause.

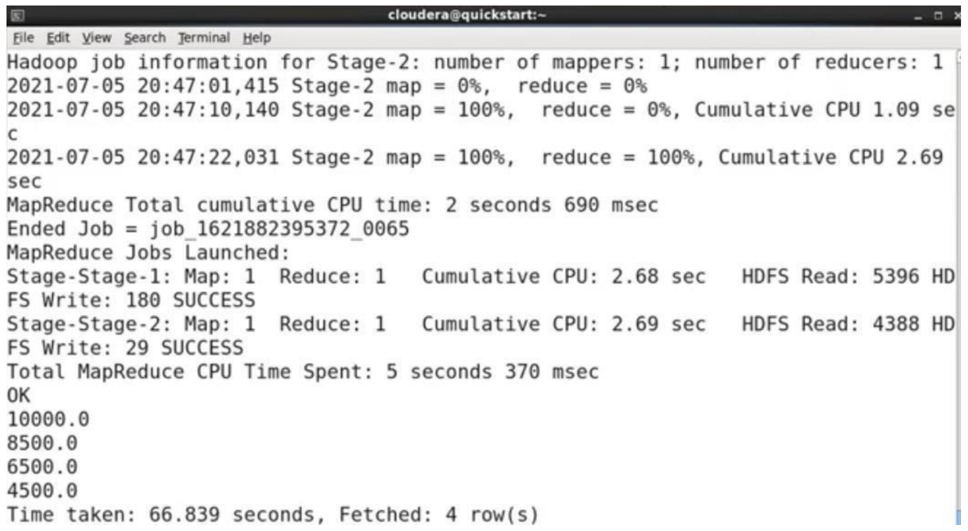
select salary from customers sort by salary desc limit 4;

It will give the only 4 records in the output after sorting them in descending order. This is not a complete syntax only we are showing what output it will give.

Mapreduce task is performed



```
hive> select salary from customers sort by salary desc limit 4;
Query ID = cloudera_20210705204646_479bdcc1-6d1c-41fb-a1b7-21695bc49337
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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 mapreduce.job.reduces=<number>
```



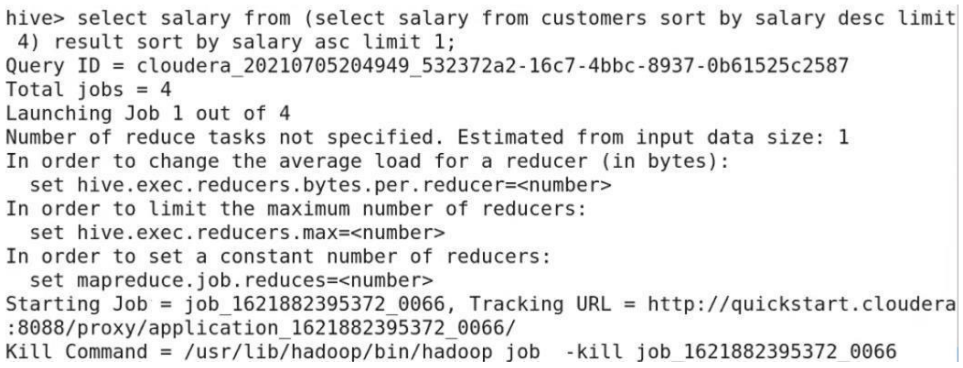
```
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2021-07-05 20:47:01,415 Stage-2 map = 0%,  reduce = 0%
2021-07-05 20:47:10,140 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.09 se
c
2021-07-05 20:47:22,031 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.69
sec
MapReduce Total cumulative CPU time: 2 seconds 690 msec
Ended Job = job_1621882395372_0065
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.68 sec   HDFS Read: 5396 HD
FS Write: 180 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.69 sec   HDFS Read: 4388 HD
FS Write: 29 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 370 msec
OK
10000.0
8500.0
6500.0
4500.0
Time taken: 66.839 seconds, Fetched: 4 row(s)
```

Now what records which we have got by executing the above queries now we will use this query as subqueries and we will now sort them in ascending order to find fourth largest salary of customer table.

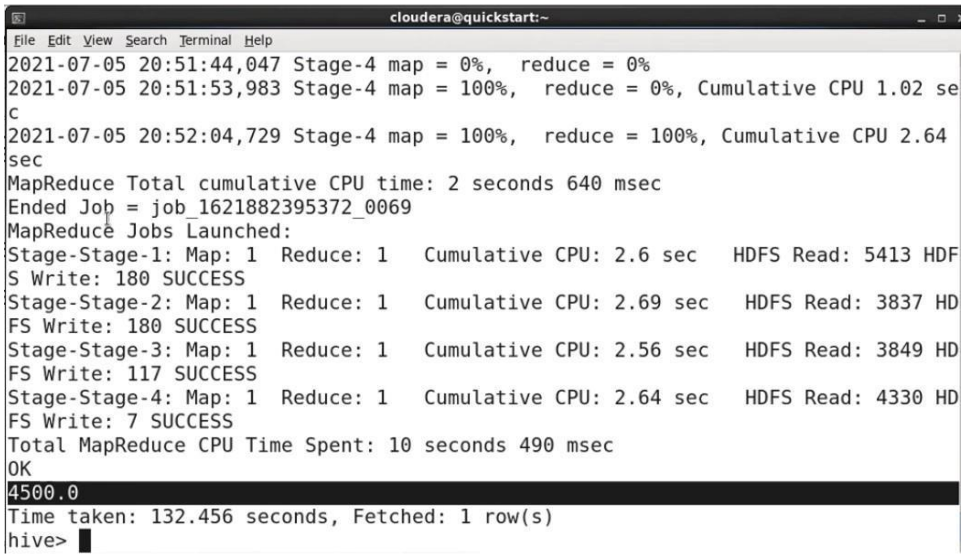
select salary from (select salary from customers sort by salary desc limit 4) result sort by salary asc limit 1;

Now whatever result we get from subquery we will store them in result table and then it will sort the result table in ascending order and as we want fourth largest salary so we are limiting it to 1.

Mapreduce task is performed



```
hive> select salary from (select salary from customers sort by salary desc limit
 4) result sort by salary asc limit 1;
Query ID = cloudera_20210705204949_532372a2-16c7-4bbc-8937-0b61525c2587
Total jobs = 4
Launching Job 1 out of 4
Number of reduce tasks not specified. Estimated from input data size: 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 mapreduce.job.reduces=<number>
Starting Job = job_1621882395372_0066, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1621882395372_0066/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1621882395372_0066
```



```
2021-07-05 20:51:44,047 Stage-4 map = 0%,  reduce = 0%
2021-07-05 20:51:53,983 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 1.02 se
c
2021-07-05 20:52:04,729 Stage-4 map = 100%,  reduce = 100%, Cumulative CPU 2.64
sec
MapReduce Total cumulative CPU time: 2 seconds 640 msec
Ended Job = job_1621882395372_0069
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.6 sec   HDFS Read: 5413 HDF
S Write: 180 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.69 sec   HDFS Read: 3837 HD
FS Write: 180 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.56 sec   HDFS Read: 3849 HD
FS Write: 117 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 2.64 sec   HDFS Read: 4330 HD
FS Write: 7 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 490 msec
OK
4500.0
Time taken: 132.456 seconds, Fetched: 1 row(s)
hive> 
```

Now we got the fourth largest salary i.e. 4500.0 as an output.