Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

Experiment 7 – Joins, Sorting, Subqueries using HiveQL

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

Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

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

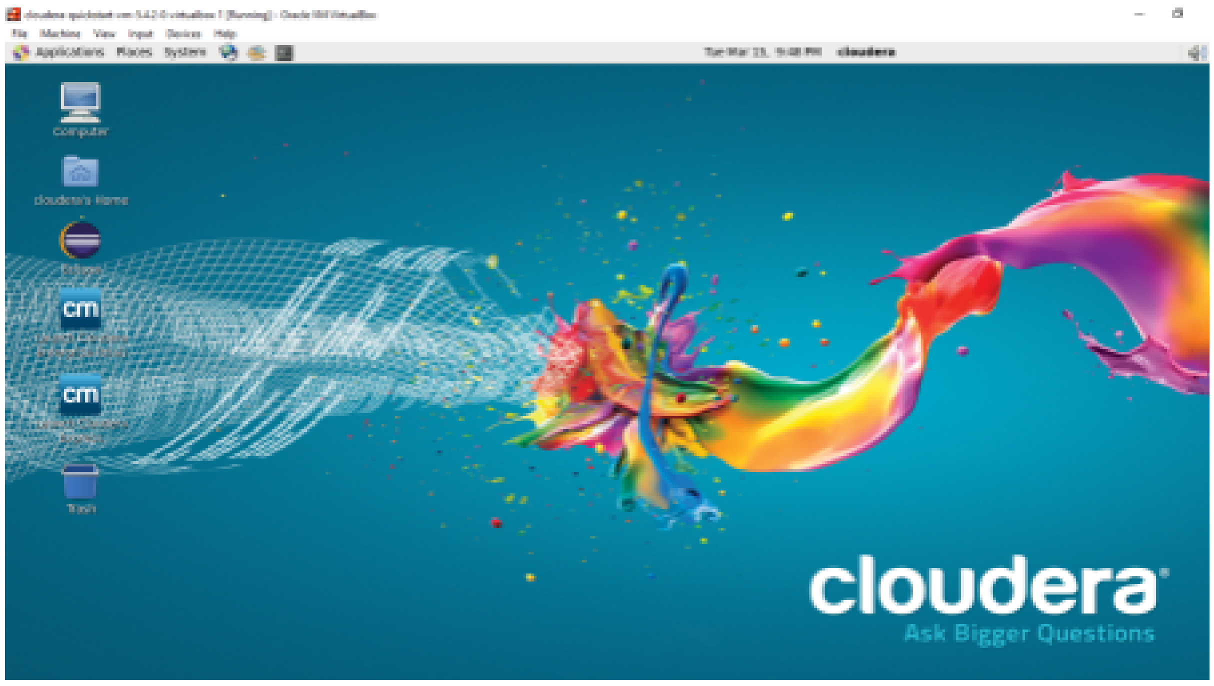
The SORT BY syntax is similar to the syntax of ORDER BY in SQL language.

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

Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

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.

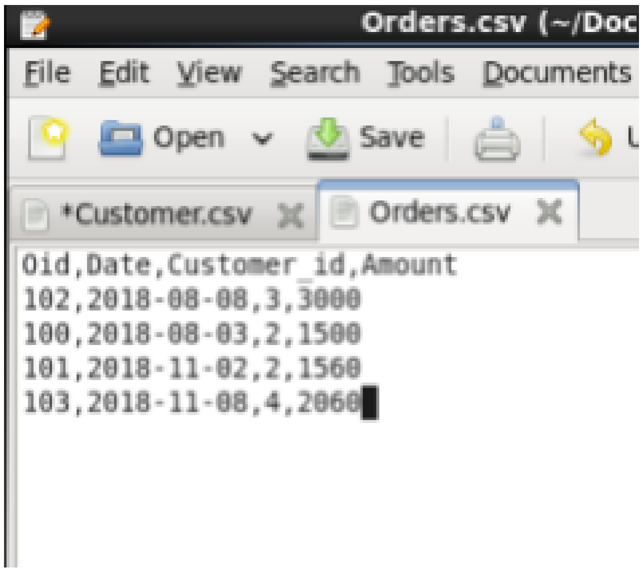
Steps: Joins, Sorting, Subqueries using HiveQL
1. Open the cloudera.



Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

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

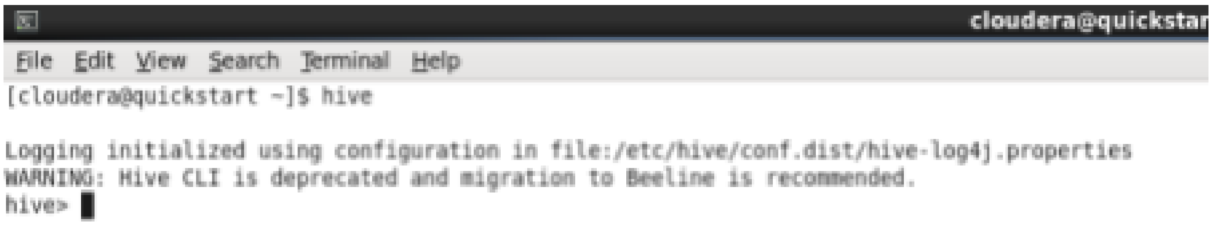
3. Then creating Orders.csv file.



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

Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

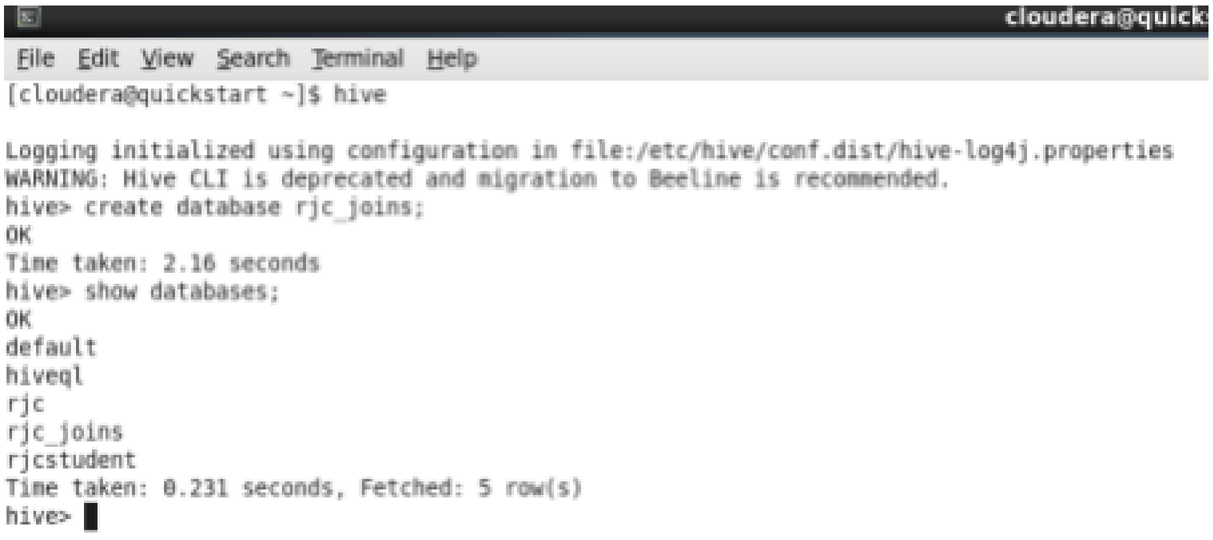
4. Open the terminal, Now we use hive command to enter the hive shell prompt and in hive
shell we could execute all of the hive commands.



```
[cloudera@quickstart ~]$ hive

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

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;

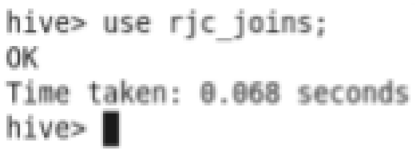


```
[cloudera@quickstart ~]$ hive

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 database rjc_joins;
OK
Time taken: 2.16 seconds
hive> show databases;
OK
default
hiveql
rjc
rjc_joins
rjcstudent
Time taken: 0.231 seconds, Fetched: 5 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;

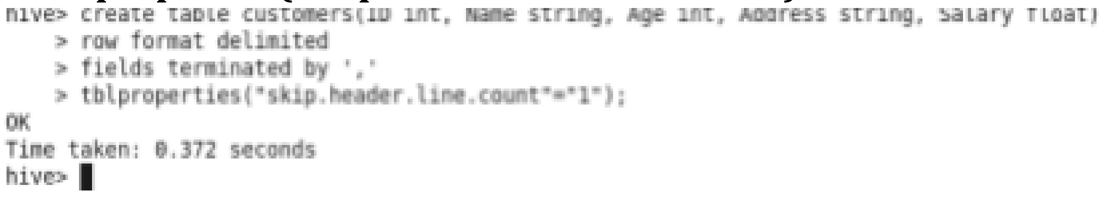


```
hive> use rjc_joins;
OK
Time taken: 0.068 seconds
hive>
```

Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

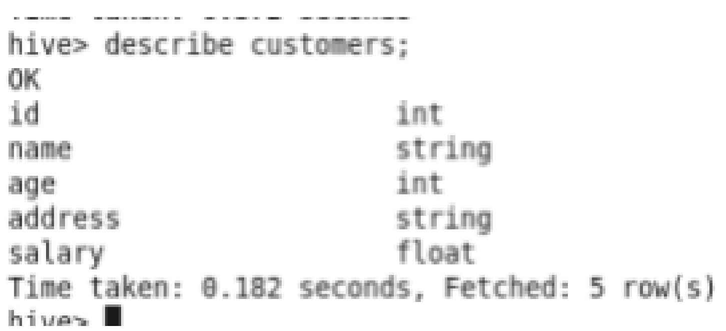
7. Now we will create two tables in one table we will load the Customer.csv file and in the other table we will load Orders.csv file.
create table customers(ID int, Name string, Age int, Address string, Salary float)

- ➢ **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: 0.372 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: 0.182 seconds, Fetched: 5 row(s)
hive>
```

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

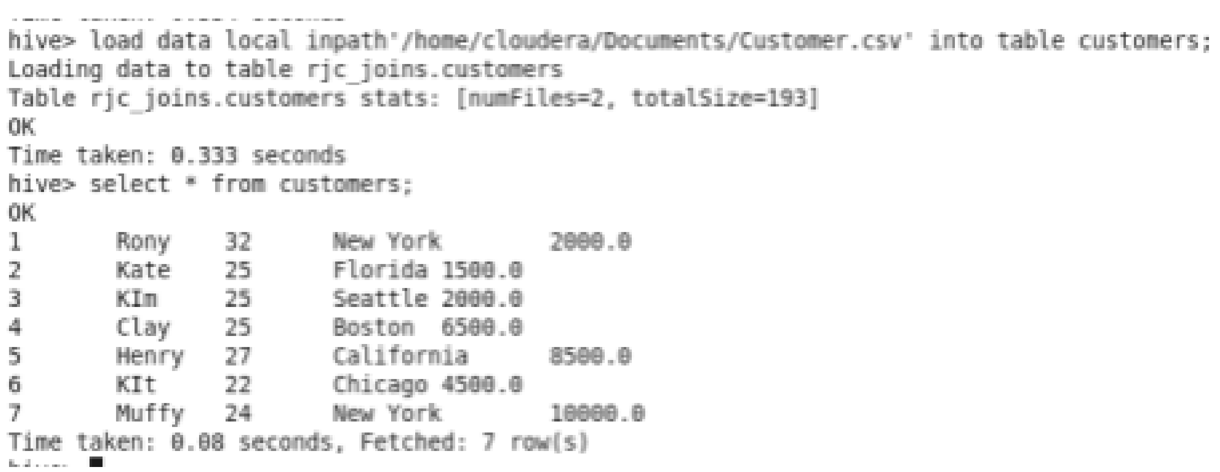
customers;

Select * from customers;

Name: Pooja Pathak

Roll No: 14

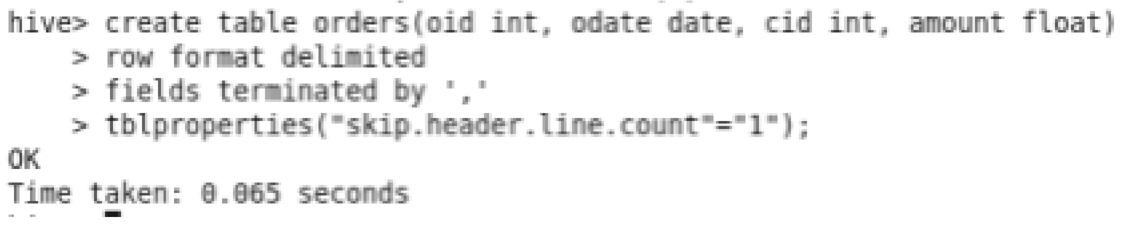
MSc Part-I Sem-2


```
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=2, totalSize=193]
OK
Time taken: 0.333 seconds
hive> select * from customers;
OK
1       Rony    32      New York        2000.0
2       Kate    25      Florida 1500.0
3       KIm     25      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: 0.08 seconds, Fetched: 7 row(s)
```

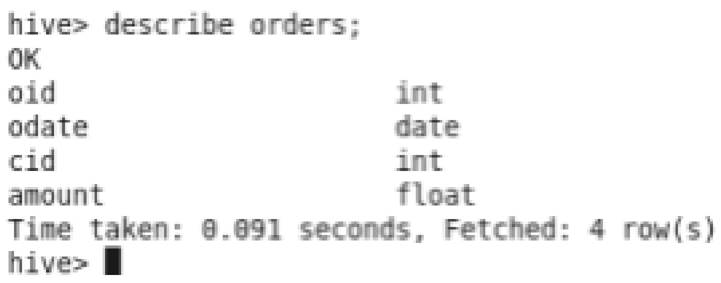
8. Creating a second table named as orders using below command, create table orders(oid int, odate date, cid int, amount float) ➢ row format delimited

➢ 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: 0.065 seconds
```

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: 0.091 seconds, Fetched: 4 row(s)
hive>
```

Now loading data in the orders table from Orders.csv file which present inside

/home/cloudera/Documents directory.

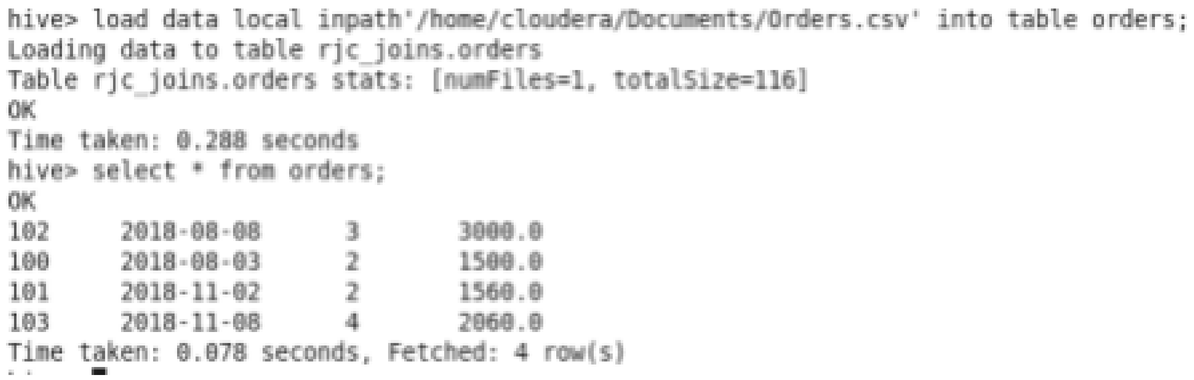
load data local inpath '/home/cloudera/Documents/Orders.csv' into table orders;

Select * from orders;

Name: Pooja Pathak

Roll No: 14

MSc Part-I Sem-2


```
hive> load data local inpath'/home/cloudera/Documents/Orders.csv' into table orders;
Loading data to table rjc_joins.orders
Table rjc_joins.orders stats: [numFiles=1, totalSize=116]
OK
Time taken: 0.288 seconds
hive> select * from orders;
OK
102     2018-08-08      3       3000.0
100     2018-08-03      2       1500.0
101     2018-11-02      2       1560.0
103     2018-11-08      4       2060.0
Time taken: 0.078 seconds, Fetched: 4 row(s)
```

9. Join:

Now First we apply the normal joins on the two tables using below command, we want to

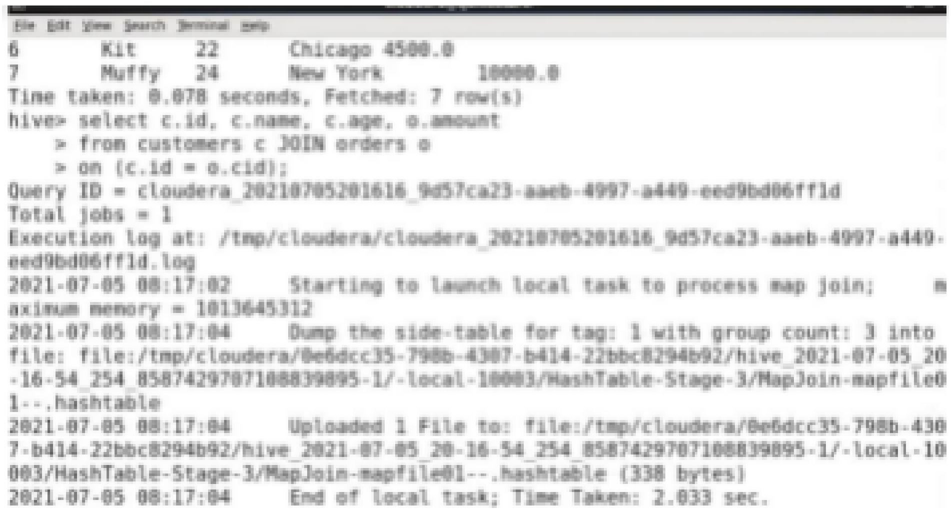
retrieve customer id, name, age from customers table and amount from the orders table

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

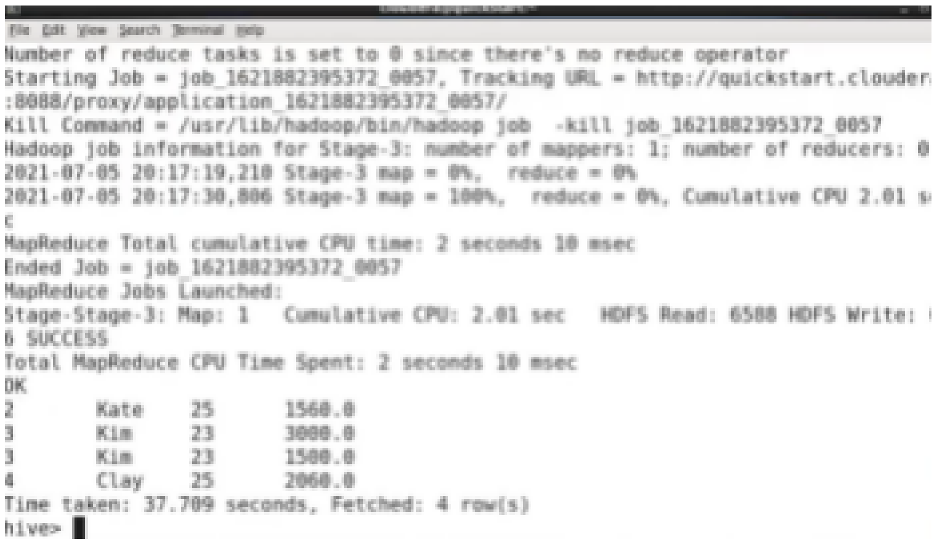
➢ on (c.id = o.cid);

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.
```

Name:
Pooja Pathak
Roll No: 14
MSc Part-I Sem-2



```
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1621882395372_0057, Tracking URL = http://quickstart.clouder
: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 s
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 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

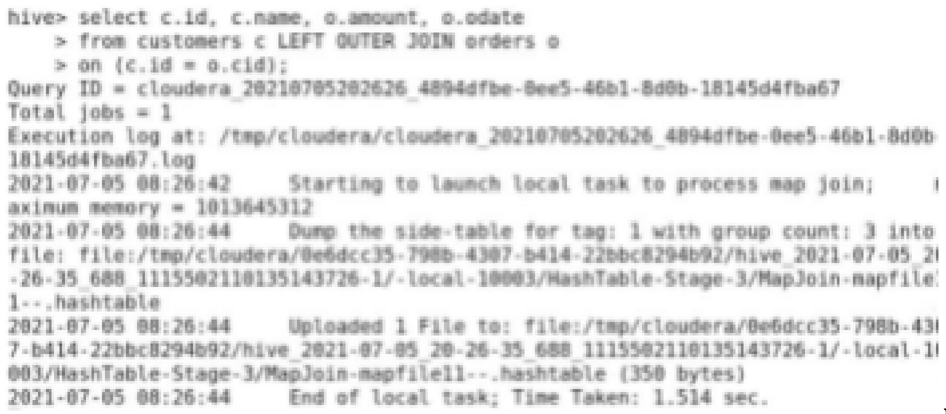
The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.

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;     
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_2
-26-35_688_1115502110135143726-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile
1--.hashtable
2021-07-05 08:26:44     Uploaded 1 File to: file:/tmp/cloudera/0e6dcc35-798b-43
7-b414-22bbc8294b92/hive_2021-07-05_20-26-35_688_1115502110135143726-1/-local-1
003/HashTable-Stage-3/MapJoin-mapfile11--.hashtable (350 bytes)
2021-07-05 08:26:44     End of local task; Time Taken: 1.514 sec.
```

Name:
Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

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

Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

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

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**

Select max(salary) from customers where customers.salary not in(select max(salary)
from customers);
Mapreduce task is performed

Name: Pooja Pathak

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

8500. 13. 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.

Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

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

Name: Pooja Pathak
Roll No: 14
MSc Part-I Sem-2

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

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