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



table, or NULL in case of no matching JOIN predicate.

A LEFT JOIN returns all the values from the left table, plus the matched values from the right

table.

table, the JOIN still returns a row in the result, but with NULL in each column from the right

matches in the right table. This means, if the ON clause matches 0 (zero) records in the right

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no

**LEFT OUTER JOIN**

keys of the tables.

OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign

JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as

**JOIN**



•

left table, or NULL in case of no matching join predicate.

A RIGHT JOIN returns all the values from the right table, plus the matched values from the

JOIN still returns a row in the result, but with NULL in each column from the left table.

no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are

**RIGHT OUTER JOIN**



There are different types of joins given as follows:

FULL OUTER JOIN

•

RIGHT OUTER JOIN

•

LEFT OUTER JOIN

•

JOIN

common to each one. It is used to combine records from two or more tables in the database.

JOIN is a clause that is used for combining specific fields from two tables by using values

**JOINS**

order is also in numeric order. If the column is of string type, then the sort order will be

sort order will be dependent on the column types. If the column is of numeric type, then the sort

Hive uses the columns in SORT BY to sort the rows before feeding the rows to a reducer. The

may give partially ordered final results.

guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by"

and "sort by" is that the former guarantees total order in the output while the latter only

Hive supports **SORT BY** which sorts the data per reducer. The difference between "order by"

lexicographical order.

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

**SORTING**

•

Comparative checking of one column values from other tables

•

Dependency of one table values on other tables

Open the cloudera.

**1.**

**Steps: Joins, Sorting, Subqueries using HiveQL**

The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer

**FULL OUTER JOIN**



tables that fulfil the JOIN condition. The joined table contains either all the records from both

•

To get a particular value combined from two column values from different tables

•

Subqueries in WHERE clause

**When to use:**

•

Subqueries in FROM clause

Subqueries can be classified into two types

values returned by the subqueries.

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

**SUB QUERIES:**

the tables, or fills in NULL values for missing matches on either side.



**3.**

Then creating **Orders.csv** file.

**2.**

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



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

**fields terminated by ‘,’**



**row format delimited**

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

other table we will load **Orders.csv** file.

**7.**

Now to work inside this database we use below command; **use rjc\_joins;**

**6.**



**tblproperties(“skip.header.line.count” =”1”);**



**hive** command to enter the **hive shell prompt** and in hive

Open the terminal, Now we use

**4.**

**5.**

As we can see **rjc\_joins** database is created.

**create database rjc\_joins;** And then showing the databases. **show databases;**

Now we will be creating a new database named as rjc\_joins using below command,

shell we could execute all of the hive commands.



**fields terminated by ‘,’**



**row format delimited**

**table orders(oid int, odate date, cid int, amount float)**

Creating a second table named as **orders** using below command, **create**

**8.**



**Select \* from customers;**



**tblproperties(“skip.header.line.count” =”1”);**

**customers;**

**load data local inpath ‘/home/cloudera/Documents/Customer.csv’ into table**

/home/cloudera/Documents directory.

Now loading data in the **customers** table from **Customer.csv** file which present inside

**customers;**

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



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

**on (c.id = o.cid);**



**from customers c JOIN orders o**

**select c.id, c.name, c.age, o.amount**

and join perform on id of the customers and orders table.

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

**9.**

**Join:**

**Mapreduce task is performed**



**Select \* from orders;**

**load data local inpath ‘/home/cloudera/Documents/Orders.csv’ into table orders;**

/home/cloudera/Documents directory.

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

**orders;**

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



**on (c.id = o.cid);**



**from customers c LEFT OUTER JOIN orders o**



**Mapreduce task is performed**

**select c.id, c.name, o.amount, o.odate**

right table, or NULL in case of no matching JOIN predicate.

A LEFT JOIN returns all the values from the left table, plus the matched values from the

the right table.

right table, the JOIN still returns a row in the result, but with NULL in each column from

no matches in the right table. This means, if the ON clause matches 0 (zero) records in the

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are

**10.**

**LEFT OUTER JOIN**



**from customers c RIGHT OUTER JOIN orders o**

**Mapreduce task is performed**

**on (c.id = o.cid);**





**select c.id, c.name, o.amount, o.odate**

the left table, or NULL in case of no matching join predicate.

A RIGHT JOIN returns all the values from the right table, plus the matched values from

the JOIN still returns a row in the result, but with NULL in each column from the left table.

are no matches in the left table. If the ON clause matches 0 (zero) records in the left table,

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there

**11.**

**RIGHT OUTER JOIN**



**Mapreduce task is performed**

**from customers);**

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

•

Subqueries in WHERE clause

•

Subqueries in FROM clause

Subqueries can be classified into two types

the values returned by the subqueries.

Now we will be using the concept of **subqueries** for finding the second largest salary

A Query present within a Query is known as a sub query. The main query will depend on

**Sub queries:**

from the customers table.

**12.**



The sort order will be dependent on the column types. If the column is of numeric type,

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.

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

**Sorting**

**13.**



**largest salary of customer table.**

**this query as subqueries and we will now sort them in ascending order to find fourth**

**Now what records which we have got by executing the above queries now we will use**

**Mapreduce task is performed**

**sort by salary asc limit 1;**

**select salary from (select salary from customers sort by salary desc limit 4) result**

then the sort order is also in numeric order. If the column is of string type, then the sort

not a complete syntax only we are showing what output it will give.

It will give the only 4 records in the output after sorting them in descending order. This is

**select salary from customers sort by salary desc limit 4;**

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

LIMIT can be used to minimize sort time.

order will be lexicographical order.



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

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**