

Q:Will the reducer work or not if you use “Limit 1” in any HiveQL query?

If we use a simple select * query then the reducer will not work as it will create a fetch task only. But if we use other queries such as group by or other aggregations then the reducer will work.

Q:Suppose I have installed Apache Hive on top of my Hadoop cluster using default metastore configuration. Then, what will happen if we have multiple clients trying to access Hive at the same time?

As the default metastore configuration in hive uses derby db only one connection is supported at a time.If we try to access using multiple clients it will give error.If we want multiple concurrent clients we have to use a standalone metastore, i.e. Local or remote metastore configuration in Apache Hive.

Q:Suppose, I create a table that contains details of all the transactions done by the customers: CREATE TABLE transaction_details (cust_id INT, amount FLOAT, month STRING, country STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,‘; Now, after inserting 50,000 records in this table, I want to know the total revenue generated for each month. But, Hive is taking too much time in processing this query. How will you solve this problem and list the steps that I will be taking in order to do so?

As hive is taking too much time in processing this query we can partition the table transaction_details on month column so now when query runs it only loads the data from the partitions and does not scan the full data.

Steps to Partition the table:

```
1.CREATE TABLE transaction_details_partitioned
(cust_id INT,
amount FLOAT,
country STRING)
PARTITIONED BY (month STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ‘,‘;
```

```
2.SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;
```

3.Load the data into partitioned table

```
INSERT OVERWRITE TABLE transaction_details_partitioned PARTITION (month) SELECT
cust_id, amount, country, month FROM transaction_details;
```

Q:How can you add a new partition for the month December in the above partitioned table?

```
ALTER TABLE transaction_details_partitioned ADD PARTITION (month='Dec') LOCATION  
'/transaction_details_partitioned;
```

Q:I am inserting data into a table based on partitions dynamically. But, I received an error – FAILED ERROR IN SEMANTIC ANALYSIS: Dynamic partition strict mode requires at least one static partition column. How will you remove this error?

```
SET hive.exec.dynamic.partition = true;
```

```
SET hive.exec.dynamic.partition.mode = nonstrict;
```

Q:Suppose, I have a CSV file – ‘sample.csv’ present in ‘/temp’ directory with the following entries:

id first_name last_name email gender ip_address

How will you consume this CSV file into the Hive warehouse using built-in SerDe?

```
CREATE EXTERNAL TABLE sample_csv  
(id int,  
first_name string,  
last_name string,  
email string,  
gender string,  
ip_address string)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'  
STORED AS TEXTFILE LOCATION '/temp';
```

Now we can execute queries on this table.

Q:Suppose, I have a lot of small CSV files present in the input directory in HDFS and I want to create a single Hive table corresponding to these files. The data in these files are in the format: {id, name, e-mail, country}. Now, as we know, Hadoop performance degrades when we use lots of small files.

So, how will you solve this problem where we want to create a single Hive table for lots of small files without degrading the performance of the system?

One can use the SequenceFile format which will group these small files together to form a single sequence file. The steps that will be followed in doing so are as follows:

We need to create a sequence file to resolve this issue.

Steps to create a sequence file

- Create a temporary table:

```
CREATE TABLE temp_table (  
  id INT,  
  name STRING,  
  email STRING,  
  country STRING)  
DELIMITED TERMINATED BY ','  
STORED AS TEXTFILE;
```

- Load the data into temp_table:

```
LOAD DATA INPATH '/input' INTO TABLE temp_table;
```

- Create a table that will store data in SequenceFile format:

```
CREATE TABLE seq_file_table  
(id INT,  
name STRING,  
email STRING,  
country STRING)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
STORED AS SEQUENCEFILE;
```

- Load the data from the temporary table into the sample_seqfile table:

```
INSERT OVERWRITE TABLE seq_file_table SELECT * FROM temp_table;
```

```
LOAD DATA LOCAL INPATH 'Home/country/state/'  
OVERWRITE INTO TABLE address;
```

The following statement failed to execute. What can be the cause?

As we are loading data from local, the path should contain a file path not a directory path.

Eg: LOAD DATA LOCAL INPATH 'Home/country/state/file_name.txt' OVERWRITE INTO TABLE address;

Q: Is it possible to add 100 nodes when we already have 100 nodes in Hive? If yes, how?

Yes, we can add the nodes by following the below steps:

Step 1: Take a new system; create a new username and password

Step 2: Install SSH and with the master node setup SSH connections

Step 3: Add ssh public_rsa id key to the authorized_keys file

Step 4: Add the new DataNode hostname, IP address, and other details in /etc/hosts slaves file:

192.168.1.102 slave3.in slave3

Step 5: Start the DataNode on a new node

Step 6: Login to the new node like suhadoop or:

ssh -X hadoop@192.168.1.103

Step 7: Start HDFS of the newly added slave node by using the following command:

./bin/hadoop-daemon.sh start data node

Hive Practical Questions:

Hive Join operations

Create a table named CUSTOMERS(ID | NAME | AGE | ADDRESS | SALARY)


Create a Second table ORDER(OID | DATE | CUSTOMER_ID | AMOUNT)

Now perform different joins operations on top of these tables

(Inner JOIN, LEFT OUTER JOIN ,RIGHT OUTER JOIN ,FULL OUTER JOIN)

cloudera@quickstart:~


```
hive> select * from customers;
OK
customers.id    customers.name  customers.age  customers.address  customers.salary
1      Shashank      28      Gurgaon 100000
2      Sudhanshu     32      Bangalore      786665
3      Prajwal 23      Nagpur 12231
4      Vansh 15      Nagpur 787878
5      Prakash 35      Nagpur 121212
6      Ram 32      Delhi 23322
Time taken: 0.166 seconds, Fetched: 6 row(s)
hive> █
```

 cloudera@quickstart:~

```
hive> select * from orders;
OK
orders.oid      orders.order_date  orders.customer_id  orders.amount
1      2022-08-10      1      10000.0
2      2022-09-01      2      890.0
3      2022-01-01      3      90.0
4      2022-08-01      4      8700.0
5      2022-08-10      5      1000.0
6      2022-09-01      2      890.0
7      2022-01-01      3      90.0
8      2022-08-01      4      8700.0
Time taken: 0.164 seconds, Fetched: 8 row(s)
hive> █
```

INNER JOIN

```
hive> select c.id,c.name,o.oid,o.order_date,o.amount from
> orders o
> join customers c
> on o.customer_id = c.id;
```

 cloudera@quickstart:~

```
hive> select c.id,c.name,o.oid,o.order_date,o.amount from
> orders o
> join customers c
> on o.customer_id = c.id;
Query ID = cloudera_20220916003535_09b53ebb-0f1b-476e-88e1-a5ba8fcef38
Total jobs = 1
```

```

c.id    c.name  o.oid    o.order_date    o.amount
1       Shashank    1        2022-08-10      10000.0
2       Sudhanshu  2        2022-09-01      890.0
3       Prajwal 3    2022-01-01      90.0
4       Vansh    4        2022-08-01      8700.0
5       Prakash 5    2022-08-10      1000.0
2       Sudhanshu  6        2022-09-01      890.0
3       Prajwal 7    2022-01-01      90.0
4       Vansh    8        2022-08-01      8700.0
Time taken: 86.685 seconds, Fetched: 8 row(s)
hive> █


```

Left Outer Join

```

hive> Select c.id,c.name,o.oid,o.order_date,o.amount
      > from customers c
      > left join orders o
      > on c.id = o.customer_id;

```

 cloudera@quickstart:~

```

hive> Select c.id,c.name,o.oid,o.order_date,o.amount
      > from customers c
      > left join orders o
      > on c.id = o.customer_id;
Query ID = cloudera_20220916005454_a1769bd8-22c9-484f-9811-8138d1ce1ce7

```

```

OK
c.id      c.name  o.oid    o.order_date    o.amount
1         Shashank 1        2022-08-10      10000.0
2         Sudhanshu 2        2022-09-01      890.0
2         Sudhanshu 6        2022-09-01      890.0
3         Prajwal 3        2022-01-01      90.0
3         Prajwal 7        2022-01-01      90.0
4         Vansh    4        2022-08-01      8700.0
4         Vansh    8        2022-08-01      8700.0
5         Prakash 5        2022-08-10      1000.0
6         Ram      NULL     NULL            NULL
Time taken: 43.445 seconds, Fetched: 9 row(s)
hive>

```

RIGHT OUTER JOIN

```

hive> Select c.id,c.name,o.oid,o.order_date,o.amount
> from orders o
> right join customers c
> on o.id = c.customer_id;

```

```

hive> Select c.id,c.name,o.oid,o.order_date,o.amount
> from orders o
> right join customers c
> on o.customer_id = c.id;
Query ID = cloudera_20220916005858_9d174e62-584b-4ad9-a792-d0d7e7818089
Total jobs = 1

```


```

OK
c.id      c.name  o.oid    o.order_date    o.amount
1         Shashank 1        2022-08-10      10000.0
2         Sudhanshu 2        2022-09-01      890.0
2         Sudhanshu 6        2022-09-01      890.0
3         Prajwal 3        2022-01-01      90.0
3         Prajwal 7        2022-01-01      90.0
4         Vansh    4        2022-08-01      8700.0
4         Vansh    8        2022-08-01      8700.0
5         Prakash 5        2022-08-10      1000.0
6         Ram      NULL     NULL            NULL
Time taken: 37.616 seconds, Fetched: 9 row(s)
hive>

```

FULL OUTER JOIN

```
hive> Select c.id,c.name,o.oid,o.order_date,o.amount
> from orders o
> full join customers c
> on o.customer_id = c.id;
```

 cloudera@quickstart:~

```
hive> Select c.id,c.name,o.oid,o.order_date,o.amount
> from orders o
> full join customers c
> on o.customer_id = c.id;
```

Query ID = cloudera_20220916010000_294926c7-46a2-48ab-a7e5-4145f9cb3759

OK

c.id	c.name	o.oid	o.order_date	o.amount
1	Shashank	1	2022-08-10	10000.0
2	Sudhanshu	6	2022-09-01	890.0
2	Sudhanshu	2	2022-09-01	890.0
3	Prajwal	7	2022-01-01	90.0
3	Prajwal	3	2022-01-01	90.0
4	Vansh	8	2022-08-01	8700.0
4	Vansh	4	2022-08-01	8700.0
5	Prakash	5	2022-08-10	1000.0
6	Ram	NULL	NULL	NULL

Time taken: 93.762 seconds, Fetched: 9 row(s)

hive> █

Download a data from the given location -

<https://archive.ics.uci.edu/ml/machine-learning-databases/00360/>


1. Create a hive table as per given schema in your dataset


```

hive>
hive> create table bank(
  >   age int,
  >   job string,
  >   marital string,
  >   education string,
  >   default string,
  >   balance int,
  >   housing string,
  >   loan string,
  >   contact string,
  >   day int,
  >   month string,
  >   duration int,
  >   campaign int,
  >   padays int,
  >   previous int,
  >   poutcome string,
  >   y string)
  > row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  > with SERDEPROPERTIES(
  >   "separatorChar" = "\;",
  >   "quoteChar" = "\""
  > )
  > stored as textfile;
OK
Time taken: 0.268 seconds

```

2. try to place a data into table location

 cloudera@quickstart:~

```

hive> load data local inpath 'file:///home/cloudera/bank.csv' into table bank;
Loading data to table hive_class_b1.bank
Table hive_class_b1.bank stats: [numFiles=1, totalSize=461474]
OK
Time taken: 0.618 seconds
hive> █

```

3. Perform a select operation .

```
cloudera@quickstart~$ hive> select * from bank limit 10;
OK
bank.age      bank.job      bank.marital  bank.education  bank.default  bank.balance  bank.housing  bank.loan      bank.contact  bank.day      bank.month    bank.duration
bank.campaign bank.padays   bank.previous bank.poutcome   bank.y        contact day    month duration    campaign      pdays  previous  poutcome    y
age  job      marital education default balance housing loan  contact day    month duration    campaign      pdays  previous  poutcome    y
30   unemployed married primary no    1787  no    no    cellular  19    oct  79    1    -1    0    unknown no
33   services  married secondary no    4789  yes   yes   cellular  11    may  220   1    339   4    failure no
35   management single tertiary  no    1350  yes   no    cellular  16    apr  185   1    330   1    failure no
30   management married tertiary no    1476  yes   yes   unknown  3    jun  199   4    -1    0    unknown no
59   blue-collar married secondary no    0     yes   no    unknown  5    may  226   1    -1    0    unknown no
35   management single tertiary  no    747   no    no    cellular  23    feb  141   2    176   3    failure no
36   self-employed married tertiary no    307   yes   no    cellular  14    may  341   1    330   2    other no
39   technician married secondary no    147   yes   no    cellular  6     may  151   2    -1    0    unknown no
41   entrepreneur married tertiary no    221   yes   no    unknown  14    may  57    2    -1    0    unknown no
Time taken: 0.126 seconds, Fetched: 10 row(s)
hive>
```

4. Fetch the result of the select operation in your local as a csv file .

```
cloudera@quickstart~$ [cloudera@quickstart ~]$ hive -e "select * from hive_class_b1.bank" > ~/output.csv
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
OK
Time taken: 0.632 seconds, Fetched: 4522 row(s)
[cloudera@quickstart ~]$ ls
AirQualityUCI.csv  country_wise_latest.csv  Desktop  enterprise-deployment.json  kerberos  output.csv  sales_order_data.csv  test2.txt
array_data.csv    covid_19_clean_complete.csv  Documents  express-deployment.json    lib        parcels      sampledata.csv        usa_county_wise.csv
bank.csv          customers.csv               Downloads  full_grouped.csv          map_data.csv  Pictures     sample_output.csv     Videos
cloudera-manager  day_wise.csv               eclipse    hive-hcatalog-core-0.14.0.jar  Music       Public       temp                  workspace
cm.apl.py         dept_data.csv              employee.csv  json_data.json            orders.csv   sales_data.csv  Templates              worldometer_data.csv
[cloudera@quickstart ~]$
```

5. Perform group by operation .

hive> Select marital,count(1) cnt

> from bank

> group by marital;

cloudera@quickstart~

```
hive> Select marital,count(1) cnt
> from bank
> group by marital;
Query ID = cloudera_20220916030303_5a1bbaa1-116e-4b9d-94b6-df98e0b15d93
Total jobs = 1
Launching Job 1 out of 1
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_1663312363320_0007, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1663312363320_0007/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1663312363320_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-09-16 03:04:11,433 Stage-1 map = 0%, reduce = 0%
2022-09-16 03:04:38,775 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 27.91 sec
2022-09-16 03:04:50,524 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 30.19 sec
MapReduce Total cumulative CPU time: 30 seconds 190 msec
Ended Job = job_1663312363320_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 30.19 sec HDFS Read: 470579 HDFS Write: 48 SUCCESS
Total MapReduce CPU Time Spent: 30 seconds 190 msec
OK
marital cnt
divorced          528
marital 1
married 2797
single 1196
Time taken: 53.877 seconds, Fetched: 4 row(s)
hive>
```

7. Perform filter operation at least 5 kinds of filter examples .

select * from bank where job in ('services','blue-collar') limit 10;

```
cloudera@quickstart~
hive> select * from bank where job in ('services','blue-collar') limit 10;
OK
bank.age      bank.job      bank.marital  bank.education  bank.default  bank.balance  bank.housing  bank.loan      bank.contact  bank.day      bank.month      bank.duration
bank.campaign bank.padays  bank.previous bank.poutcome  bank.y        bank.y        bank.y        bank.y        bank.y        bank.y        bank.y        bank.y
33  services    married secondary no 4789 yes yes cellular 11 may 220 1 339 4 failure no
59  blue-collar married secondary no 0 yes no unknown 5 may 226 1 -1 0 unknown no
43  services    married primary no -88 yes yes cellular 17 apr 313 1 147 2 failure no
39  services    married secondary no 9374 yes no unknown 20 may 273 1 -1 0 unknown no
31  blue-collar married secondary no 360 yes yes cellular 29 jan 89 1 241 1 failure no
25  blue-collar single primary no -221 yes no unknown 23 may 250 1 -1 0 unknown no
31  services    married secondary no 132 no no cellular 7 jul 148 1 152 1 other no
44  services    single secondary no 106 no no unknown 12 jun 109 2 -1 0 unknown no
55  blue-collar married primary no 627 yes no unknown 5 may 247 1 -1 0 unknown no
32  blue-collar married secondary no 2089 yes no cellular 14 nov 132 1 -1 0 unknown yes
Time taken: 0.486 seconds, Fetched: 10 row(s)
hive>
```

hive> select * from bank where education like 'primary' limit 10;

```

cloudera@quickstart:~
hive> select * from bank where education like 'primary' limit 10;
OK
bank.age      bank.job      bank.marital  bank.education  bank.default  bank.balance  bank.housing  bank.loan      bank.contact  bank.day      bank.month      bank.duration
bank.campaign  bank.padays    bank.previous  bank.poutcome  bank.y
30  unemployed  married primary no 1787 no no cellular 19 oct 79 1 -1 0 unknown no
43  services    married primary no -88 yes yes cellular 17 apr 313 1 147 2 failure no
25  blue-collar single primary no -221 yes no unknown 23 may 250 1 -1 0 unknown no
55  blue-collar married primary no 627 yes no unknown 5 may 247 1 -1 0 unknown no
78  retired divorced primary no 229 no no telephone 22 oct 97 1 -1 0 unknown yes
55  blue-collar married primary no 145 no no telephone 2 feb 59 3 5 2 other no
26  blue-collar married primary no 0 yes no unknown 21 may 425 1 -1 0 unknown no
22  entrepreneur single primary yes -849 yes yes cellular 4 feb 204 1 -1 0 unknown no
45  blue-collar divorced primary no 844 no no unknown 5 jun 1018 3 -1 0 unknown yes
41  blue-collar married primary no -516 no yes telephone 8 jul 554 3 -1 0 unknown no
Time taken: 0.151 seconds, Fetched: 10 row(s)
hive>

```

8. show and example of regex operation

```

cloudera@quickstart:~
hive> alter table bank rename to bank_data;
OK
Time taken: 0.295 seconds
hive> alter table bank_data change column loan loan string after balance;
OK
Time taken: 0.387 seconds
hive>

```

10.Drop table bank_data;

12 . order by operation .

```

hive>
hive>
>      Select marital,count(1) cnt
>      from bank_data
>      group by marital
>      order by cnt desc;
Query ID = cloudera_20220916034242_3785c037-bfae-4caa-8d39-4a7a92643350
Total jobs = 2
Launching Job 1 out of 2

```

```

OK
secondary      2306
tertiary        1350
primary 678
unknown 187
education       1
Time taken: 72.597 seconds, Fetched: 5 row(s)
hive>

```

14 . sorting operation you have to perform .

cloudera@quickstart:~

```
hive> Select marital,count(1) cnt
> from bank_data
> group by marital
> sort by cnt desc;
Query ID = cloudera_20220916034646_34497e96-b40a-4895-8d2a-4c4754d4a1b5
Total jobs = 2
```

```
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.93 sec
Total MapReduce CPU Time Spent: 16 seconds 870 msec
OK
secondary      2306
tertiary        1350
primary 678
unknown 187
education       1
Time taken: 58.047 seconds, Fetched: 5 row(s)
```

15 . distinct operation you have to perform .

Select distinct job from bank_data;

cloudera@quickstart:~

```
hive> select distinct job from bank_data;
Query ID = cloudera_20220916034848_93f03fa3-ea4d-442b-8e37-5e0562a51cb0
Total jobs = 1
Launching Job 1 out of 1
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_1663312363320_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1663312363320_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1663312363320_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-09-16 03:48:29,469 Stage-1 map = 0%, reduce = 0%
2022-09-16 03:48:38,390 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 7.93 sec
2022-09-16 03:48:47,942 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 10.06 sec
MapReduce Total cumulative CPU time: 10 seconds 60 msec
Ended Job = job_1663312363320_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 10.06 sec HDFS Read: 470444 HDFS Write: 126 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 60 msec
OK
admin.
blue-collar
entrepreneur
housemaid
job
management
retired
self-employed
services
student
technician
unemployed
unknown
Time taken: 31.514 seconds, Fetched: 13 row(s)
hive>
```


16 . like an operation you have to perform .

Select * from bank_data where job like 'student';

17 . union operation you have to perform .

```
hive> select * from bank_data where job='student'
> union all
> select * from bank_data where job='technician';
Query ID = cloudera_20220916035151_2934b14e-f02b-4e36-b477-2727d53f2277
Total jobs = 1
Launching Job 1 out of 1
```

18 . table view operation you have to perform .

 cloudera@quickstart:~

```
hive> create view client_jobs as
> select job,count(1) no_of_clients
> from bank_data
> group by job;
OK
Time taken: 0.405 seconds
```

Q:Create a python application that connects to the Hive database for extracting data, creating sub tables for data processing, drops temporary tables.fetch rows to python itself into a list of tuples and mimic the join or filter operations

<https://github.com/PrajwalMahale/hive/blob/main/main.py>