Scenario Based questions:

Question1:
Will the reducer work or not if you use “Limit 1” in any HiveQL query?

Answer:

Yes, using "LIMIT 1" in a HiveQL query will work with reducers.

When a query is executed in Hive, it is broken down into multiple tasks called "**MapReduce**" tasks. The "Map" tasks process the data and emit intermediate results, while the "Reduce" tasks aggregate these intermediate results to produce the final output.

The "LIMIT" clause in a HiveQL query limits the number of records returned by the query. When "LIMIT 1" is used, Hive will only return the first record that satisfies the query conditions.

The use of "LIMIT 1" in a HiveQL query does not affect the execution of reducers. The reducers will still be used to aggregate the intermediate results, but since there is only one record to be returned, there will be only one reducer task. The mapper task will be executed to read and filter the data, and the reducer task will be executed to aggregate the data and produce the final output.

Question 2

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?

Answer

If multiple clients try to access Hive at the same time, they will be able to do so without any issues. Apache Hive is designed to handle multiple concurrent connections from different clients.

When you install Hive on a Hadoop cluster with default metastore configuration, the metastore is configured to use an embedded Derby database. This embedded database is capable of handling concurrent connections from multiple clients.

When a client connects to Hive, it creates a session with the HiveServer2 process, which is responsible for managing client connections and executing queries. Each client session is isolated and does not interfere with other sessions. HiveServer2 can handle multiple client sessions simultaneously, and each session is managed independently.

Therefore, even if multiple clients try to access Hive at the same time, they will be able to do so without any issues, as Hive is designed to handle concurrent connections from multiple clients. However, if the workload on the cluster becomes too high, it may impact the performance of Hive, and users may experience longer query execution times. In such cases, it may be necessary to scale the cluster up or optimize the queries to improve performance.

Question 3

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?

Answer

We can solve this problem with the help of concepts of partitioning or bucketing

By Creating Partitions

Lets create a base table first as shown below

```
hive> create table customer_transaction(
    > cust_id int,
    > amount int,
    > month string,
    > country string
    > )
    > row format delimited
    > fields terminated by ','
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.246 seconds
hive> load data local inpath 'file:///app/customer_transaction.csv' into
table customer_transaction;
```

```
Using Base Table to form dynamic Partitioned Table
hive> create table customer_transaction_partition(
    > cust_id int,
    > amount int,
    > country string)
    > partitioned by (month string);
OK
Time taken: 0.193 seconds
hive> insert overwrite table customer_transaction_partition partition
(month)
    > select cust_id,amount,country,month from customer_transaction;
FAILED: SemanticException [Error 10096]: Dynamic partition strict mode
requires at least one static partition column. To turn this off set
hive.exec.dynamic.partition.mode=nonstrict
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> insert overwrite table customer_transaction_partition partition
(month)
    > select cust_id,amount,country,month from customer_transaction;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230305190500_331dc2a4-c226-4ab6-97fd-54f845283a89
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2023-03-05 19:05:06,696 Stage-1 map = 0%,  reduce = 0%
2023-03-05 19:05:07,720 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local2125662914_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory
hdfs://namenode:8020/user/hive/warehouse/hive_db.db/customer_transaction_pa
rtition/.hive-staging_hive_2023-03-05_19-05-00_921_6658704204825331977-1/-e
xt-10000
Loading data to table hive_db.customer_transaction_partition partition
(month=null)

Loaded : 6/6 partitions.
         Time taken to load dynamic partitions: 2.031 seconds
         Time taken for adding to write entity : 0.003 seconds
```

```
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 2496 HDFS Write: 1521 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
cust_id amount  country month
Time taken: 10.218 seconds
hive>
```

You can check this in Hadoop Environment also if these partitions are created or not

```
# hadoop fs -ls
/user/hive/warehouse/hive_db.db/customer_transaction_partition
Found 6 items
drwxrwxr-x   - root supergroup          0 2023-03-05 19:05
/user/hive/warehouse/hive_db.db/customer_transaction_partition/month=Jan
drwxrwxr-x   - root supergroup          0 2023-03-05 19:05
/user/hive/warehouse/hive_db.db/customer_transaction_partition/month=Jul
drwxrwxr-x   - root supergroup          0 2023-03-05 19:05
/user/hive/warehouse/hive_db.db/customer_transaction_partition/month=Jun
drwxrwxr-x   - root supergroup          0 2023-03-05 19:05
/user/hive/warehouse/hive_db.db/customer_transaction_partition/month=Mar
drwxrwxr-x   - root supergroup          0 2023-03-05 19:05
/user/hive/warehouse/hive_db.db/customer_transaction_partition/month=May
drwxrwxr-x   - root supergroup          0 2023-03-05 19:05
/user/hive/warehouse/hive_db.db/customer_transaction_partition/month=Oct
```

By Creating Buckets

Bucketing is another technique that can improve the performance of the query by reducing the amount of data scanned. It involves dividing the data into smaller buckets based on a **hash function** of the column being bucketed.

Let's create bucket table with cluster id on month

```
hive> create table customer_transaction_bucket
    > (
    > cust_id int,
    > amount int,
    > month string,
    > country string
    > )
    > clustered by (month)
    > sorted by (month)
    > into 3 buckets;
OK
Time taken: 0.123 seconds
hive> insert overwrite table customer_transaction_bucket select * from
customer_transaction;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230305191643_278a7a24-f839-4574-9de1-47464864adda
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
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>
Job running in-process (local Hadoop)
2023-03-05 19:16:45,671 Stage-1 map = 100%,  reduce = 0%
2023-03-05 19:16:46,721 Stage-1 map = 100%,  reduce = 67%
2023-03-05 19:16:47,746 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local561407860_0002
Loading data to table hive_db.customer_transaction_bucket
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 14404 HDFS Write: 7290 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
customer_transaction.cust_id    customer_transaction.amount
customer_transaction.month      customer_transaction.country
Time taken: 5.205 seconds
```

To confirm for bucket creation in hadoop

```
# hadoop fs -ls /user/hive/warehouse/hive_db.db/customer_transaction_bucket
Found 3 items
-rwxrwxr-x   3 root supergroup        127 2023-03-05 19:16
/user/hive/warehouse/hive_db.db/customer_transaction_bucket/000000_0
-rwxrwxr-x   3 root supergroup         72 2023-03-05 19:16
/user/hive/warehouse/hive_db.db/customer_transaction_bucket/000001_0
-rwxrwxr-x   3 root supergroup        131 2023-03-05 19:16
/user/hive/warehouse/hive_db.db/customer_transaction_bucket/000002_0
# hadoop fs -ls
/user/hive/warehouse/hive_db.db/customer_transaction_bucket/000000_0
-rwxrwxr-x   3 root supergroup        127 2023-03-05 19:16
/user/hive/warehouse/hive_db.db/customer_transaction_bucket/000000_0
# hadoop fs -cat
/user/hive/warehouse/hive_db.db/customer_transaction_bucket/000000_0
190765MarIndia
223907MarUK
924528OctUSA
912596OctUK
867906OctUSA
896507OctIndia
746172OctUK
87169OctUK
#
```

Question 4

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

Answer

By Static Partitioning Method

```
hive> create table customer_transaction_static(
    > cust_id int,
    > amount int,
    > country string)
    > partitioned by (month string);
```

```
OK
Time taken: 0.212 seconds
hive> insert overwrite table customer_transaction_static partition
(month='Dec')
    > select cust_id,amount,country from customer_transaction where
month='Dec';
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230305192728_ce1edccc-9fa9-4234-9780-edbed28345b5
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
```

In Hadoop we can confirm the creation by using below command

```
# hadoop fs -ls /user/hive/warehouse/hive_db.db/customer_transaction_static
Found 1 items
drwxrwxr-x   - root supergroup          0 2023-03-05 19:27
/user/hive/warehouse/hive_db.db/customer_transaction_static/month=Dec
#
```

Question 5

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?

Answer

Disable dynamic partitioning strict mode: We can disable dynamic partitioning strict mode by setting the configuration property "hive.exec.dynamic.partition.mode" to "nonstrict". This will allow us to insert data into partitions without specifying all the partition columns.

Question 6

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?

Answer
Step 1
Moving File from local to hadoop /tmp location

```
# hadoop fs -put /app/sample.csv /tmp
# hadoop fs -ls /tmp
Found 2 items
drwx-wx-wx   - root supergroup          0 2023-02-04 09:51 /tmp/hive
-rw-r--r--   3 root supergroup       3726 2023-03-05 19:55 /tmp/sample.csv
#
```

Step 2

Creating a table in hive and referring this /tmp location for creating an internal table

```
hive> create table sample
    > (
    > id int,
    > first_name string,
    > last_name string,
    > email string,
    > gender string,
    > ip_address string)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 1.367 seconds
hive> load data inpath '/tmp/sample.csv' into table sample;
Loading data to table hive_db.sample
OK
Time taken: 2.418 seconds
hive> select * from sample limit 10;
OK
sample.id       sample.first_name       sample.last_name
sample.email    sample.gender   sample.ip_address
```

```
NULL    first_name      last_name       email   gender  ip_address
1       Kim     Hale    kimhale@example.com     Female  85.247.112.83
2       Cole    Harrison        coleharrison@example.com        Male
125.126.116.46
3       Eliana  Hobbs   elianahobbs@example.com Female  222.126.22.7
4       Khalid  Mcgrath khalidmcgrath@example.com       Male
102.163.80.153
5       Erin    Cole    erincole@example.com    Female  108.190.110.178
6       Katelin Hess    katelinhess@example.com Female  138.40.238.88
7       Selena  Gilbert selenagilbert@example.com       Female
47.152.211.88
8       Mylo    Pierce  mylopierce@example.com  Male    218.47.186.225
9       Destiny Mccarthy        destinymccarthy@example.com     Female
156.67.116.171
Time taken: 0.523 seconds, Fetched: 10 row(s)
hive>
```

Step 3
Confirming if the data is in hive warehouse

```
# hadoop fs -ls /user/hive/warehouse/hive_db.db/sample
Found 1 items
-rwxrwxr-x   3 root supergroup       3726 2023-03-05 19:55
/user/hive/warehouse/hive_db.db/sample/sample.csv
```

Question 7

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?

Answer

Mostly we can first combine all single files into one and point that file to create an external table in hive

Question 8

LOAD DATA LOCAL INPATH ‘Home/country/state/’
OVERWRITE INTO TABLE address;

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

Answer

Check for file path mostly it should point to some file like sample.csv
As there could be many files inside state directory

Question 9

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

Answer

Yes we can do it by adding nodes in hadoop cluster and configuring them

<u>**Hive Practical questions:**</u>

Hive Join operations

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

Lets Create a sample data

```
1,John Doe,35,"123,Main St, Anytown,USA",75000.0
2,Jane Smith,27,"456 Oak Ave, Anytown, USA",55000.0
3,Bob Brown,45,"789 Pine Rd, Anytown, USA",100000.0
```

Creating customers table with csv Serde library and loading data from sample data

```
hive> create table customers (
    > id int
    > ,
    > name string,
    > age int,
    > address string,
    > salary int)
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > with serdeproperties (
    > "separatorChar"=",",
    > "quoteChar"="\""
    > )
    > stored as textfile;
OK
Time taken: 0.569 seconds
hive> load data local inpath 'file:///app/customers.csv' into table
customers;
Loading data to table hive_db.customers
OK
Time taken: 1.185 seconds
hive> select * from customers;
OK
customers.id    customers.name  customers.age   customers.address
customers.salary
1       John Doe        35      123,Main St, Anytown,USA        75000.0
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0
3       Bob Brown       45      789 Pine Rd, Anytown, USA       100000.0
Time taken: 0.298 seconds, Fetched: 3 row(s)
hive> select address from customers;
OK
address
123,Main St, Anytown,USA
456 Oak Ave, Anytown, USA
789 Pine Rd, Anytown, USA
Time taken: 0.164 seconds, Fetched: 3 row(s)
hive>
```

Creating second table orders
OID | DATE | CUSTOMER_ID | AMOUNT
1,2022-01-01,1,1000.0
2,2022-01-02,1,2000.0
3,2022-01-03,2,1500.0
4,2022-01-04,3,5000.0
5,2022-01-05,2,3000.0

```
hive> create table orders (
    > oid int,
    > date_time DATE,
    > customer_id int,
    > amount float
    > )
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.362 seconds
hive> load data local inpath 'file:///app/orders.csv' into table orders;
Loading data to table hive_db.orders
OK
Time taken: 1.015 seconds
hive>
```

Now perform different joins operations on top of these tables
(Inner JOIN, LEFT OUTER JOIN ,RIGHT OUTER JOIN ,FULL OUTER JOIN)
Inner Join

```
hive> select * from customers
    > inner join orders on orders.customer_id=customers.id
    > ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230305210847_60aa8954-d7f6-45c2-b2ec-16c1943b26c2
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in
```

```
[jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLo
ggerBinder.class]
SLF4J: Found binding in
[jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
Execution log at:
/tmp/root/root_20230305210847_60aa8954-d7f6-45c2-b2ec-16c1943b26c2.log
2023-03-05 21:08:56     Starting to launch local task to process map join;
maximum memory = 477626368
2023-03-05 21:08:58     Dump the side-table for tag: 1 with group count: 3
into file:
file:/tmp/root/a8350add-bdd0-434a-ba2a-80e665326222/hive_2023-03-05_21-08-4
7_804_4027251762087833772-1/-local-10004/HashTable-Stage-3/MapJoin-mapfile0
1--.hashtable
2023-03-05 21:08:58     Uploaded 1 File to:
file:/tmp/root/a8350add-bdd0-434a-ba2a-80e665326222/hive_2023-03-05_21-08-4
7_804_4027251762087833772-1/-local-10004/HashTable-Stage-3/MapJoin-mapfile0
1--.hashtable (393 bytes)
2023-03-05 21:08:58     End of local task; Time Taken: 1.801 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2023-03-05 21:09:00,815 Stage-3 map = 100%,  reduce = 0%
Ended Job = job_local468983981_0001
MapReduce Jobs Launched:
Stage-Stage-3:  HDFS Read: 370 HDFS Write: 109 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
customers.id    customers.name  customers.age   customers.address
customers.salary        orders.oid      orders.date_time
orders.customer_id       orders.amount
1       John Doe        35      123,Main St, Anytown,USA        75000.0 1
2022-01-01      1       1000.0
1       John Doe        35      123,Main St, Anytown,USA        75000.0 2
2022-01-02      1       2000.0
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0 3
2022-01-03      2       1500.0
```

```
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0 5
2022-01-05      2       3000.0
3       Bob Brown       45      789 Pine Rd, Anytown, USA       100000.0
4       2022-01-04      3       5000.0
Time taken: 13.027 seconds, Fetched: 5 row(s)
hive>
```

Left outer join

```
hive> select * from customers
    > left outer join orders on orders.customer_id=customers.id;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230305211014_6a565eda-1510-4beb-8405-2b2f1ad429f5
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in
[jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLo
ggerBinder.class]
SLF4J: Found binding in
[jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
Execution log at:
/tmp/root/root_20230305211014_6a565eda-1510-4beb-8405-2b2f1ad429f5.log
2023-03-05 21:10:20     Starting to launch local task to process map join;
maximum memory = 477626368
2023-03-05 21:10:21     Dump the side-table for tag: 1 with group count: 3
into file:
file:/tmp/root/a8350add-bdd0-434a-ba2a-80e665326222/hive_2023-03-05_21-10-1
4_329_5222155641039588901-1/-local-10004/HashTable-Stage-3/MapJoin-mapfile1
1--.hashtable
2023-03-05 21:10:22     Uploaded 1 File to:
file:/tmp/root/a8350add-bdd0-434a-ba2a-80e665326222/hive_2023-03-05_21-10-1
4_329_5222155641039588901-1/-local-10004/HashTable-Stage-3/MapJoin-mapfile1
```

```
1--.hashtable (393 bytes)
2023-03-05 21:10:22     End of local task; Time Taken: 1.345 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2023-03-05 21:10:24,207 Stage-3 map = 100%,  reduce = 0%
Ended Job = job_local1100734693_0002
MapReduce Jobs Launched:
Stage-Stage-3:  HDFS Read: 522 HDFS Write: 109 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
customers.id    customers.name  customers.age   customers.address
customers.salary        orders.oid      orders.date_time
orders.customer_id       orders.amount
1       John Doe        35      123,Main St, Anytown,USA        75000.0 1
2022-01-01      1       1000.0
1       John Doe        35      123,Main St, Anytown,USA        75000.0 2
2022-01-02      1       2000.0
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0 3
2022-01-03      2       1500.0
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0 5
2022-01-05      2       3000.0
3       Bob Brown       45      789 Pine Rd, Anytown, USA       100000.0
4       2022-01-04      3       5000.0
Time taken: 9.884 seconds, Fetched: 5 row(s)
hive>
```

Right Outer Join

```
hive> select * from customers
    > right outer join orders on orders.customer_id=customers.id;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230305211152_87d55b71-aed3-4da0-944d-f42f6ada19f1
Total jobs = 1
```

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in
[jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLo
ggerBinder.class]
SLF4J: Found binding in
[jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
Execution log at:
/tmp/root/root_20230305211152_87d55b71-aed3-4da0-944d-f42f6ada19f1.log
2023-03-05 21:11:59     Starting to launch local task to process map join;
maximum memory = 477626368
2023-03-05 21:12:00     Dump the side-table for tag: 0 with group count: 3
into file:
file:/tmp/root/a8350add-bdd0-434a-ba2a-80e665326222/hive_2023-03-05_21-11-5
2_887_5592232781083745007-1/-local-10004/HashTable-Stage-3/MapJoin-mapfile2
0--.hashtable
2023-03-05 21:12:00     Uploaded 1 File to:
file:/tmp/root/a8350add-bdd0-434a-ba2a-80e665326222/hive_2023-03-05_21-11-5
2_887_5592232781083745007-1/-local-10004/HashTable-Stage-3/MapJoin-mapfile2
0--.hashtable (485 bytes)
2023-03-05 21:12:00     End of local task; Time Taken: 1.4 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2023-03-05 21:12:02,752 Stage-3 map = 100%,  reduce = 0%
Ended Job = job_local610516821_0003
MapReduce Jobs Launched:
Stage-Stage-3:  HDFS Read: 631 HDFS Write: 109 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
customers.id    customers.name  customers.age   customers.address
customers.salary        orders.oid      orders.date_time
orders.customer_id       orders.amount
1       John Doe        35      123,Main St, Anytown,USA        75000.0 1
2022-01-01      1       1000.0
1       John Doe        35      123,Main St, Anytown,USA        75000.0 2
2022-01-02      1       2000.0
```

```
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0 3
2022-01-03      2       1500.0
3       Bob Brown       45      789 Pine Rd, Anytown, USA       100000.0
4       2022-01-04      3       5000.0
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0 5
2022-01-05      2       3000.0
Time taken: 9.874 seconds, Fetched: 5 row(s)
hive>
```

Full Outer Join

```
hive> select * from customers
    > full outer join orders on orders.customer_id=customers.id;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230305211245_411df96d-caac-424c-939b-cc1d646472f0
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>
Job running in-process (local Hadoop)
2023-03-05 21:12:47,199 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1792450364_0004
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 2567 HDFS Write: 327 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
customers.id    customers.name  customers.age   customers.address
customers.salary        orders.oid      orders.date_time
orders.customer_id       orders.amount
1       John Doe        35      123,Main St, Anytown,USA        75000.0 2
2022-01-02      1       2000.0
1       John Doe        35      123,Main St, Anytown,USA        75000.0 1
2022-01-01      1       1000.0
```

```
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0 5
2022-01-05      2       3000.0
2       Jane Smith      27      456 Oak Ave, Anytown, USA       55000.0 3
2022-01-03      2       1500.0
3       Bob Brown       45      789 Pine Rd, Anytown, USA       100000.0
4       2022-01-04      3       5000.0
Time taken: 1.646 seconds, Fetched: 5 row(s)
hive>
```

BUILD A DATA PIPELINE WITH HIVE

Download a data from the given location -
https://archive.ics.uci.edu/ml/machine-learning-databases/00360/

<u>1. Create a hive table as per given schema in your dataset</u>

```
Date: The date the measurements were taken
Time: The time the measurements were taken
CMC: Carbon monoxide concentration in mg/m^3
TIOC: Tin oxide concentration in a gas sensor in mV
NMHC: Non-methane hydrocarbons concentration in microg/m^3
BC: Benzene concentration in microg/m^3
TC: Titania concentration in a gas sensor in mV
NO: Nitrogen oxides concentration in ppb
T0: Tungsten oxide concentration in a gas sensor in mV
NDO: Nitrogen dioxide concentration in microg/m^3
TO: Tungsten oxide concentration in a gas sensor in mV
PT08.S5(O3): Indium oxide concentration in a gas sensor in mV
T: Temperature in Celsius
RH: Relative humidity in %
AH: Absolute humidity in mg/L
```

<u>2. try to place a data into table location</u>

```
hive> load data local inpath 'file:///app/kaggle.csv' into table mytable;
```

3. Perform a select operation .

```
hive> select * from mytable limit 1;
OK
mytable.datetime        mytable.time    mytable.co_gt   mytable.pt08_s1_co
mytable.nmhc_gt mytable.c6_h6_gt        mytable.pt_08_s2_nmhc
mytable.nox_gt  mytable.pt_08_s3_nox    mytable.no2_gt  mytable.pt08_s4_no2
mytable.pt08_s5_o3      mytable.t       mytable.rhmytable.ah
10-03-2004      18:00:00        2.6     1360.0  150.0   11.9    1046.0
166.0   1056.0  113.0   1692.0  1268.0  13.6    48.9    0.7578
Time taken: 0.569 seconds, Fetched: 1 row(s)
hive>
```

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

```
INSERT OVERWRITE LOCAL DIRECTORY '/app' ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' SELECT * FROM mytable;
```

5. Perform group by operation .

```
hive> select datetime,sum(co_gt)
    > from mytable
    > group by datetime;
```

6.Question Missing

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

```
a)hive> select * from mytable where split(datetime,'-')[0]=3;
```

```
b)select * from mytable where no2_gt between 10 and 15 ;
c)select * from mytable where split(time,':')[0]>12;
d)select datetime,time,co_gt from mytable limit 100;
e)select * from mytable where no2_gt in (11.9,9.4);
```

8. show an example of regex operation

```
hive> SELECT datetime,regexp_replace(datetime, '-', '/') AS date_string
from mytable;
```

9. alter table operation

```
hive> alter table mytable change datetime datetimenew string;
OK
Time taken: 0.622 seconds
hive> select * from mytable limit 1;
OK
mytable.datetimenew     mytable.time    mytable.co_gt   mytable.pt08_s1_co
mytable.nmhc_gt mytable.c6_h6_gt        mytable.pt_08_s2_nmhc
mytable.nox_gt  mytable.pt_08_s3_nox    mytable.no2_gt  mytable.pt08_s4_no2
mytable.pt08_s5_o3      mytable.t       mytable.rhmytable.ah
10-03-2004      18:00:00        2.6     1360.0  150.0   11.9    1046.0
166.0   1056.0  113.0   1692.0  1268.0  13.6    48.9    0.7578
Time taken: 0.217 seconds, Fetched: 1 row(s)
hive>
```

10 . drop table operation

```
hive> drop table mytable;
OK
Time taken: 0.948 seconds
hive>
```

11.Question Missing

12 . order by operation .

```
hive> select * from mytable
    > order by datetime desc limit 1;
```

13 . where clause operations you have to perform .

```
select * from mytable where no2_gt in (11.9,9.4);
```

14 . sorting operation you have to perform .

```
hive> select * from mytable
    > order by datetime desc limit 1;
```

15 . distinct operation you have to perform .

```
hive>Select distinct datetime from mytable;
```

16 . like an operation you have to perform .

```
select * from mytable
    > where co_gt like '%3%';
```

17 . union operation you have to perform .

```
hive> select distinct string(datetime) from mytable
    > union
    > select current_date() from mytable;
```

18 . table view operation you have to perform .

```
hive> create view mytable_view as
    > select datetime,time
    > from mytable;
OK
datetime        time
Time taken: 4.242 seconds
hive> select * from mytable_view limit 20;
OK
mytable_view.datetime   mytable_view.time
10-03-2004      18:00:00
10-03-2004      19:00:00
```