

# Hive Installation Guide and HQL Hands On - H2

This guide is the CLASS component for the Hive Installation and HQL Hands On. Before you start, please make sure you have completed the HOME part found [here](#). Follow the instructions in this file to complete the hands-on/tutorial tasks.

**ALL COMMANDS MUST BE RUN ONLY AFTER CLONING THE REPO AND FROM WITHIN THE LOCALLY CLONED FOLDER ONLY**

## Few useful commands

### To start the Hive CLI

```
start-hive.sh
```

### If you are running into issues with starting Hadoop, try the following

```
stop-all.sh  
sudo rm -rf ~/dfsdata/  
sudo rm -rf ~/tmpdata/  
hdfs namenode -format  
start-all.sh
```

### To remove all data from HDFS which is involved with Hive

```
hdfs dfs -rm -r -f /user*
```

(Screenshot numbers begin from 2 since screenshot 1a was from the HOME component)

# TASK 1 - Create tables, partitions and buckets in hive.

Find the dataset "netflix1.csv" provided in the repo [here](#). You would find this dataset in the cloned repo as well.

attribute	Description	DataType
show_id	unique values	String
type	Content(Tv show or Movie)	String
title	Title	String
director	Director of the show or movie	String
country	Country of the show or movie	String
release_year	Content release year	int

## TABLE

Create a table with the above schema in the Hive shell.

```
create table netflix(show_id String,type String,title String,director
String,country String,release_year int,primary key (show_id) disable
novalidate) row format delimited fields terminated by ',';
```

Now load the CSV file into the netflix table created in the Hive shell.

```
load data local inpath 'PATH_TO_netflix1.csv' into table netflix;
```

**Make sure you replace the PATH\_TO\_netflix1.csv with the actual path to the file**

Since the dataset is huge lets query the first 3 records in the database.

When you perform "select \* from tablename", Hive fetches the whole data from file as a FetchTask rather than a mapreduce task which just dumps the data as it is without doing anything on it. This is similar to "hadoop dfs -text ".Therefore when queried using SELECT, FILTER, LIMIT, this property skips mapreduce and uses FETCH task. As a result Hive can execute query without running mapreduce task as

shown below.

```
select * from netflix limit 3;
```

**Take a screenshot of the terminal output and name it 2a.png**(If you have multiple screenshots because all content did not fit in one, you can name them 2a1.png, 2a2.png, 2a3.png, etc.)

```
Initialization script completed
schematool completed
SLF4J: class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/han/hive/apache_hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/han/hadoop-3.3.3/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/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]
Hive Session ID = aed1c5ab-97ad-4241-944c-634654be7744

Logging initialized using configuration in jar:file:/home/han/hive/apache_hive/lib/hive-common-3.1.3.jar!/hive-log4j2.properties Async: true
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.
Hive Session ID = 23401b1c-b0b9-46b1-b273-b1fe56525850
hive> create table netflix(show_id String,type String,title String,director String,country String,release_year int,primary key (show_id) disable novalidate) row format delimited fields terminated by ',';
OK
Time taken: 2.367 seconds
hive> load data local inpath '/home/han/Desktop/BD22/UE20CS322-H2/netflix1.csv' into table netflix;
Loading data to table default.netflix
OK
Time taken: 1.307 seconds
hive> select * from netflix limit 3;
OK
s1      Movie    Dick Johnson Is Dead    Kirsten Johnson United States    2020
s3      TV Show  Ganglands    Julien Leclercq France    2021
s6      TV Show  Mldnight Mass    Mike Flanagan    United States    2021
Time taken: 2.866 seconds, Fetched: 3 row(s)
hive>
```

Hive is a data warehouse database for Hadoop, all database and table data files are stored at HDFS location /user/hive/warehouse by default.

To check all database and table files stored open a new terminal and use the following command

```
hdfs dfs -ls /user/hive/warehouse
hdfs dfs -ls /user/hive/warehouse/netflix
```

**Take a screenshot of the terminal output and name it 2b.png**

```
sagarika@ubuntu: ~
sagarika@ubuntu: ~$ hdfs dfs -ls /user/hive/warehouse
Found 1 items
drwxr-xr-x - sagarika supergroup 0 2022-09-02 05:05 /user/hive/warehouse/netflix
sagarika@ubuntu:~$ hdfs dfs -ls /user/hive/warehouse/netflix
Found 1 items
-rw-r--r-- 1 sagarika supergroup 541863 2022-09-02 05:05 /user/hive/warehouse/netflix/netflix1.csv
sagarika@ubuntu:~$
```

## PARTITION

Hive organizes tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as type, country etc. Using partition, it is easy to query a portion of the data.

For example, a table named Employee contains employee data such as id, name, dept, and yoj (i.e., year of joining). Suppose you need to retrieve the details of all employees who joined in 2012. A query searches the whole table for the required information. However, if you partition the employee data with the year and store it in a separate file, it reduces the query processing time.

Firstly enable dynamic partition using the commands in the Hive shell as follows:

```
set hive.exec.dynamic.partition=True;
set hive.exec.dynamic.partition.mode=nonstrict;
```

To create a partitioned table we have to follow the below command in the Hive shell:

```
create table netflix_partition(title String,director String,country
String,release_year int) partitioned by (type String);
```

Now we will load the data into the partitioned table using the following command in the Hive shell:

```
insert into table netflix_partition partition(type='Movie') select
title,director,country,release_year from netflix where type='Movie';
insert into table netflix_partition partition(type='TV Show') select
title,director,country,release_year from netflix where type='TV Show';
```

Let's check the partitioned table

```
select * from netflix_partition limit 3;
```

**Take a screenshot of the terminal output and name it 2c.png**(If you have multiple screenshots because all content did not fit in one, you can name them 2c1.png, 2c2.png, 2c3.png, etc.)

```

hive> create table netflix_partition(title String,director String,country String,release_year int) partitioned by (type String);
OK
Time taken: 0.178 seconds
hive> insert into table netflix_partition partition(type='Movie') select title,director,country,release_year from netflix where type='Movie';
Query ID = han_20220906234927_172cd1d2-fee1-4a38-ba3f-b3b419cf34d
Total jobs = 3
Launching Job 1 out of 3
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_1662477242847_0019, Tracking URL = http://sachin:8088/proxy/application_1662477242847_0019/
Kill Command = /home/han/hadoop-3.3.3/bin/mapred job -kill job_1662477242847_0019
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-09-06 23:49:47,392 Stage-1 map = 0%, reduce = 0%
2022-09-06 23:49:59,395 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 10.51 sec
2022-09-06 23:50:10,335 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 17.91 sec
MapReduce Total cumulative CPU time: 17 seconds 910 msec
Ended Job = job_1662477242847_0019
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://127.0.0.1:9000/user/hive/warehouse/netflix_partition/type=Movie/.hive-staging_hive_2022-09-06_23-49-28_033_1552963472564916125-1/-ext-10000
Loading data to table default.netflix_partition partition (type=Movie)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 17.91 sec HDFS Read: 559852 HDFS Write: 304492 SUCCESS
Total MapReduce CPU Time Spent: 17 seconds 910 msec
OK
Time taken: 45.968 seconds
hive> insert into table netflix_partition partition(type='Tv shows') select title,director,country,release_year from netflix where type='Tv shows';
Query ID = han_20220906235025_2b732f63-37f7-41b8-ac5f-7e4519c9ae8a
Total jobs = 3
Launching Job 1 out of 3
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_1662477242847_0020, Tracking URL = http://sachin:8088/proxy/application_1662477242847_0020/
Kill Command = /home/han/hadoop-3.3.3/bin/mapred job -kill job_1662477242847_0020
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-09-06 23:50:45,492 Stage-1 map = 0%, reduce = 0%
2022-09-06 23:50:58,816 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 10.11 sec
2022-09-06 23:51:09,625 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 17.87 sec
MapReduce Total cumulative CPU time: 17 seconds 870 msec
Ended Job = job_1662477242847_0020
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://127.0.0.1:9000/user/hive/warehouse/netflix_partition/type=Tv shows/.hive-staging_hive_2022-09-06_23-50-25_579_7921135060984760601-1/-ext-10000
Loading data to table default.netflix_partition partition (type=Tv shows)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 17.87 sec HDFS Read: 559911 HDFS Write: 152 SUCCESS
Total MapReduce CPU Time Spent: 17 seconds 870 msec
OK
Time taken: 46.844 seconds
hive> select * from netflix_partition limit 3;
OK
Dick Johnson Is Dead Kirsten Johnson United States 2020 Movie
Confessions of an Invisble Girl Bruno Garotti Brazil 2021 Movie
Sankofa Haile Gerima United States 1993 Movie
Time taken: 0.353 seconds, Fetched: 3 row(s)
hive>

```

To check the partitions stored in Hadoop data warehouse, use the following command on the terminal:

```

hdfs dfs -ls /user/hive/warehouse/netflix_partition/type=Movie
hdfs dfs -ls /user/hive/warehouse/netflix_partition/type=Tv\ Show

```

Take a screenshot of the terminal output and name it 2d.png

```
sagarika@ubuntu:~$ hdfs dfs -ls /user/hive/warehouse/netflix_partition/type=Movie
Found 2 items
-rw-r--r-- 1 sagarika supergroup 422016 2022-09-02 19:45 /user/hive/warehouse/netflix_partition/type=Movie/000000_0
-rw-r--r-- 1 sagarika supergroup 541863 2022-09-02 20:47 /user/hive/warehouse/netflix_partition/type=Movie/netflix1.csv
sagarika@ubuntu:~$ hdfs dfs -ls /user/hive/warehouse/netflix_partition/
Found 3 items
drwxr-xr-x - sagarika supergroup 0 2022-09-02 20:46 /user/hive/warehouse/netflix_partition/type=Movie
drwxr-xr-x - sagarika supergroup 0 2022-09-02 20:50 /user/hive/warehouse/netflix_partition/type=TV shows
drwxr-xr-x - sagarika supergroup 0 2022-09-02 05:35 /user/hive/warehouse/netflix_partition/type=__HIVE_DEFAULT_PARTITION__
sagarika@ubuntu:~$
```

## BUCKETS

Tables or partitions are sub-divided into buckets, to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table.

The command below allows the correct number of reducers and the cluster by column to be automatically selected based on the table in the Hive shell:

```
set hive.enforce.bucketing=True;
```

Create bucketing on country column on top of partitioning by type and insert data in the Hive shell:

```
CREATE TABLE netflix_bucket(title String,director String,country String)
PARTITIONED BY(type String) CLUSTERED BY (country) INTO 10 BUCKETS;
insert into table netflix_bucket partition(type='Movie') select
title,director,country from netflix where type='Movie';
```

**Take a screenshot of the terminal output and name it 2e.png**(If you have multiple screenshots, you can name them 2e1.png, 2e2.png, 2e3.png, etc.)

```
hive> CREATE TABLE netflix_bucket(title String,director String,country String) PARTITIONED BY(type String) CLUSTERED BY (country) INTO 10 BUCKETS;
OK
Time taken: 0.22 seconds
```



```

hive> insert into table netflix_bucket partition(type='Movie') select title,director,country from netflix where type='Movie';
Query ID = prerana_20220902210221_48426b33-e0be-4985-95b2-fadf85c349e0
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks determined at compile time: 10
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_1662128429665_0002, Tracking URL = http://prerana-VirtualBox:8088/proxy/application_1662128429665_0002/
Kill Command = /home/prerana/hadoop-3.3.3/bin/mapred job -kill job_1662128429665_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 10
2022-09-02 21:03:07,727 Stage-1 map = 0%, reduce = 0%
2022-09-02 21:03:35,708 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.46 sec
2022-09-02 21:04:36,677 Stage-1 map = 100%, reduce = 7%, Cumulative CPU 5.83 sec
2022-09-02 21:04:40,560 Stage-1 map = 100%, reduce = 13%, Cumulative CPU 7.18 sec
2022-09-02 21:04:43,257 Stage-1 map = 100%, reduce = 27%, Cumulative CPU 9.87 sec
2022-09-02 21:04:47,415 Stage-1 map = 100%, reduce = 40%, Cumulative CPU 12.55 sec
2022-09-02 21:04:53,962 Stage-1 map = 100%, reduce = 43%, Cumulative CPU 14.76 sec
2022-09-02 21:05:00,695 Stage-1 map = 100%, reduce = 47%, Cumulative CPU 17.28 sec
2022-09-02 21:05:05,990 Stage-1 map = 100%, reduce = 50%, Cumulative CPU 19.97 sec
2022-09-02 21:05:07,274 Stage-1 map = 100%, reduce = 57%, Cumulative CPU 25.17 sec
2022-09-02 21:05:08,458 Stage-1 map = 100%, reduce = 60%, Cumulative CPU 27.88 sec
2022-09-02 21:05:55,116 Stage-1 map = 100%, reduce = 67%, Cumulative CPU 29.22 sec
2022-09-02 21:06:00,432 Stage-1 map = 100%, reduce = 73%, Cumulative CPU 30.55 sec
2022-09-02 21:06:05,294 Stage-1 map = 100%, reduce = 80%, Cumulative CPU 31.87 sec
2022-09-02 21:06:06,673 Stage-1 map = 100%, reduce = 87%, Cumulative CPU 33.19 sec
2022-09-02 21:06:18,026 Stage-1 map = 100%, reduce = 90%, Cumulative CPU 35.84 sec
2022-09-02 21:06:20,465 Stage-1 map = 100%, reduce = 93%, Cumulative CPU 38.36 sec
2022-09-02 21:06:23,946 Stage-1 map = 100%, reduce = 97%, Cumulative CPU 40.86 sec
2022-09-02 21:06:25,065 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 43.37 sec
MapReduce Total cumulative CPU time: 43 seconds 370 msec
Ended Job = job_1662128429665_0002
Loading data to table default.netflix_bucket partition (type=Movie)
Launching Job 2 out of 2

```

```

2022-09-02 21:05:05,990 Stage-1 map = 100%, reduce = 50%, Cumulative CPU 19.97 sec
2022-09-02 21:05:07,274 Stage-1 map = 100%, reduce = 57%, Cumulative CPU 25.17 sec
2022-09-02 21:05:08,458 Stage-1 map = 100%, reduce = 60%, Cumulative CPU 27.88 sec
2022-09-02 21:05:55,116 Stage-1 map = 100%, reduce = 67%, Cumulative CPU 29.22 sec
2022-09-02 21:06:00,432 Stage-1 map = 100%, reduce = 73%, Cumulative CPU 30.55 sec
2022-09-02 21:06:05,294 Stage-1 map = 100%, reduce = 80%, Cumulative CPU 31.87 sec
2022-09-02 21:06:06,673 Stage-1 map = 100%, reduce = 87%, Cumulative CPU 33.19 sec
2022-09-02 21:06:18,026 Stage-1 map = 100%, reduce = 90%, Cumulative CPU 35.84 sec
2022-09-02 21:06:20,465 Stage-1 map = 100%, reduce = 93%, Cumulative CPU 38.36 sec
2022-09-02 21:06:23,946 Stage-1 map = 100%, reduce = 97%, Cumulative CPU 40.86 sec
2022-09-02 21:06:25,065 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 43.37 sec
MapReduce Total cumulative CPU time: 43 seconds 370 msec
Ended Job = job_1662128429665_0002
Loading data to table default.netflix_bucket partition (type=Movie)
Launching Job 2 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>
Starting Job = job_1662128429665_0003, Tracking URL = http://prerana-VirtualBox:8088/proxy/application_1662128429665_0003/
Kill Command = /home/prerana/hadoop-3.3.3/bin/mapred job -kill job_1662128429665_0003
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2022-09-02 21:07:11,573 Stage-3 map = 0%, reduce = 0%
2022-09-02 21:07:39,315 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 3.44 sec
2022-09-02 21:08:04,719 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 6.98 sec
MapReduce Total cumulative CPU time: 6 seconds 980 msec
Ended Job = job_1662128429665_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 10 Cumulative CPU: 43.37 sec HDFS Read: 1715404 HDFS Write: 1550 SUCCESS
Stage-Stage-3: Map: 1 Reduce: 1 Cumulative CPU: 6.98 sec HDFS Read: 15521 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 50 seconds 350 msec
OK
Time taken: 344.922 seconds
hive>

```

To check the buckets stored in hadoop data warehouse, use the following command on the terminal:

```
hdfs dfs -ls /user/hive/warehouse/netflix_bucket/type=Movie
```

Take a screenshot of the terminal output and name it 2f.png

```
han@sachin:~/Desktop/BD22/UE20CS322-H2$ hdfs dfs -ls /user/hive/warehouse/netflix_bucket/type=Movie
Found 10 items
-rw-r--r-- 1 han supergroup 14237 2022-09-06 23:56 /user/hive/warehouse/netflix_bucket/type=Movie/000000_0
-rw-r--r-- 1 han supergroup 113867 2022-09-06 23:56 /user/hive/warehouse/netflix_bucket/type=Movie/000001_0
-rw-r--r-- 1 han supergroup 23287 2022-09-06 23:56 /user/hive/warehouse/netflix_bucket/type=Movie/000002_0
-rw-r--r-- 1 han supergroup 10311 2022-09-06 23:57 /user/hive/warehouse/netflix_bucket/type=Movie/000003_0
-rw-r--r-- 1 han supergroup 5485 2022-09-06 23:57 /user/hive/warehouse/netflix_bucket/type=Movie/000004_0
-rw-r--r-- 1 han supergroup 46603 2022-09-06 23:57 /user/hive/warehouse/netflix_bucket/type=Movie/000005_0
-rw-r--r-- 1 han supergroup 8528 2022-09-06 23:57 /user/hive/warehouse/netflix_bucket/type=Movie/000006_0
-rw-r--r-- 1 han supergroup 9748 2022-09-06 23:57 /user/hive/warehouse/netflix_bucket/type=Movie/000007_0
-rw-r--r-- 1 han supergroup 22963 2022-09-06 23:57 /user/hive/warehouse/netflix_bucket/type=Movie/000008_0
-rw-r--r-- 1 han supergroup 17074 2022-09-06 23:57 /user/hive/warehouse/netflix_bucket/type=Movie/000009_0
han@sachin:~/Desktop/BD22/UE20CS322-H2$
```

## TASK 2- HQL Map Join and Normal Join

### Problem statement :

Given two datasets, customers and orders, perform MapJoin and normal Join operations in Hive.

### CUSTOMERS

attribute	Description	DataType
customer_id	Customer ID	String
initals	Customer Initials	String
street	Street	String
country	Country	String

### ORDERS

attribute	Description	DataType
customer_id	Customer ID	String
order_id	Order ID	String
order_date	order date	String
total_cost	Total Cost	int

Similar to Task 1, create tables and load data into the created tables.



```
create table customers(customer_id int,initials String,street
String,country String);
create table orders(customer_id int,order_id String,order_date
date,total_cost int);
```

```
insert into customers values
(1,"GH","123 road","UK"),
(3,"JK","456 road","SP"),
(2,"NL","789 road","BZ"),
(4,"AJ","1011 road","AU"),
(5,"PK","1213 road","IN");
insert into orders values
(1,1,"2022-01-04",100),
(3,4,"2022-03-07",20),
(2,2,"2022-01-02",60),
(2,3,"2022-02-01",150);
```

**Take a screenshot of the terminal output and name it 3a.png**(If you have multiple screenshots, you can name them 3a1.png, 3a2.png, 3a3.png, etc.)

```

hive> create table customers(customer_id int,initials String,street String,country String);
OK
Time taken: 0.102 seconds
hive> create table orders(customer_id int,order_id String,order_date date,total_cost int);
OK
Time taken: 0.107 seconds
hive> insert into customers values
> (1,"GH","123 road","UK"),
> (3,"JK","456 road","SP"),
> (2,"NL","789 road","BZ"),
> (4,"AJ","1011 road","AU"),
> (5,"PK","1213 road","IN");
Query ID = han_20220906211657_ef113cfa-df61-47a3-b705-3364a998a7d2
Total jobs = 3
Launching Job 1 out of 3
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_1662477242847_0005, Tracking URL = http://sachin:8088/proxy/application_1662477242847_0005/
Kill Command = /home/han/hadoop-3.3.3/bin/mapred job -kill job_1662477242847_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-09-06 21:17:13,318 Stage-1 map = 0%, reduce = 0%
2022-09-06 21:17:21,852 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.26 sec
2022-09-06 21:17:31,311 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 11.52 sec
MapReduce Total cumulative CPU time: 11 seconds 520 msec
Ended Job = job_1662477242847_0005
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://127.0.0.1:9000/user/hive/warehouse/customers/.hive-staging_hive_2022-09-06_21-16-57_844_4957968199257775503-1/-ext-10000
Loading data to table default.customers
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 11.52 sec HDFS Read: 18440 HDFS Write: 476 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 520 msec
OK
Time taken: 35.1 seconds
hive> insert into orders values
> (1,1,"2022-01-04",100),
> (3,4,"2022-03-07",20),
> (2,2,"2022-01-02",60),
> (2,3,"2022-02-01",150);
Query ID = han_20220906211741_cc0f7ba8-0db4-42d1-b465-b4ccb7632c05
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1662477242847_0006, Tracking URL = http://sachin:8088/proxy/application_1662477242847_0006/
Kill Command = /home/han/hadoop-3.3.3/bin/mapred job -kill job_1662477242847_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-09-06 21:17:55,415 Stage-1 map = 0%, reduce = 0%
2022-09-06 21:18:03,875 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.11 sec
MapReduce Total cumulative CPU time: 6 seconds 110 msec
Ended Job = job_1662477242847_0006
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://127.0.0.1:9000/user/hive/warehouse/orders/.hive-staging_hive_2022-09-06_21-17-41_157_8874374934678178964-1/-ext-10000
Loading data to table default.orders
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 6.11 sec HDFS Read: 6096 HDFS Write: 144 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 110 msec
OK
Time taken: 25.149 seconds
hive>

```

Let us first understand the functionality of normal join,

Whenever, we apply join operation, the job will be assigned to a Map Reduce task which consists of two stages– a ‘Map stage’ and a ‘Reduce stage’. A mapper’s job during Map Stage is to “read” the data from join tables and to “return” the ‘join key’ and ‘join value’ pair into an intermediate file. Further, in the shuffle stage, this intermediate file is then sorted and merged. The reducer’s job during reduce stage is to take this sorted result as input and complete the task of join.

```

select customers.initials,orders.order_id,orders.total_cost from
customers join orders on customers.customer_id=orders.customer_id;

```

As you can see in normal join all the tasks will be performed by both mapper and reducer. **Take a screenshot of the terminal output and name it 3b.png**(If you have multiple screenshots, you can name them 3b1.png, 3b2.png, 3b3.png, etc.)

```

hive> select customers.initials,orders.order_id,orders.total_cost from customers
  join orders on customers.customer_id=orders.customer_id;
Query ID = sagarika_20220903125504_26994844-1ba1-4587-90e7-f81b58f15661
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_1662229846370_0009, Tracking URL = http://ubuntu:8088/proxy/
application_1662229846370_0009/
Kill Command = /home/sagarika/hadoop-3.3.3/bin/mapred job -kill job_1662229846
370_0009
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2022-09-03 12:56:11,970 Stage-1 map = 0%,  reduce = 0%
2022-09-03 12:57:14,193 Stage-1 map = 0%,  reduce = 0%
2022-09-03 12:57:35,447 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 161.22
sec
2022-09-03 12:57:36,593 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 163.73
sec
2022-09-03 12:57:58,269 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 172.
49 sec
MapReduce Total cumulative CPU time: 2 minutes 52 seconds 490 msec
Ended Job = job_1662229846370_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 172.49 sec   HDFS Read: 1873
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1662229846370_0009, Tracking URL = http://ubuntu:8088/proxy/
application_1662229846370_0009/
Kill Command = /home/sagarika/hadoop-3.3.3/bin/mapred job -kill job_1662229846
370_0009
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2022-09-03 12:56:11,970 Stage-1 map = 0%,  reduce = 0%
2022-09-03 12:57:14,193 Stage-1 map = 0%,  reduce = 0%
2022-09-03 12:57:35,447 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 161.22
sec
2022-09-03 12:57:36,593 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 163.73
sec
2022-09-03 12:57:58,269 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 172.
49 sec
MapReduce Total cumulative CPU time: 2 minutes 52 seconds 490 msec
Ended Job = job_1662229846370_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 172.49 sec   HDFS Read: 1873
5 HDFS Write: 169 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 52 seconds 490 msec
OK
GH      1      100
NL      3      150
NL      2      60
JK      4      20
Time taken: 181.165 seconds, Fetched: 4 row(s)

```

```
hive>
```

A table can be loaded into the memory completely within a mapper without using the Map/Reducer process. It reads the data from the smaller table and stores it in an in-memory hash table and then serializes it to a hash memory file, thus substantially reducing the time. It is also known as Map Side Join in Hive. Basically, it involves performing joins between 2 tables by using only the Map phase and skipping the Reduce phase. A time decrease in your queries' computation can be observed if they regularly use a small table joins. Map-side join helps in minimizing the cost that is incurred for sorting and merging in the shuffle and reduce stages. Map-side join also helps in improving the performance of the task by decreasing the time to finish the task.

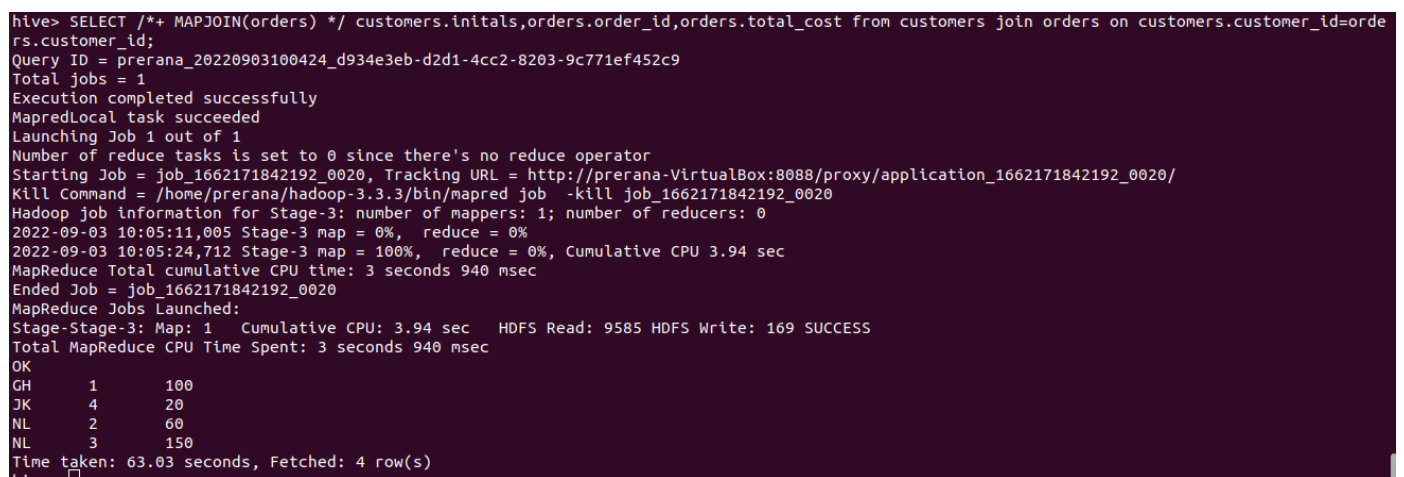
Before running the query, we have to set the below property to true in the Hive shell:

```
set hive.auto.convert.join=true;
```

Query:

```
SELECT /*+ MAPJOIN(orders) */  
customers.initials,orders.order_id,orders.total_cost from customers join  
orders on customers.customer_id=orders.customer_id;
```

**Take a screenshot of the terminal output and name it 3c.png.**



```
hive> SELECT /*+ MAPJOIN(orders) */ customers.initials,orders.order_id,orders.total_cost from customers join orders on customers.customer_id=orders.customer_id;  
Query ID = prerana_20220903100424_d934e3eb-d2d1-4cc2-8203-9c771ef452c9  
Total jobs = 1  
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  
Starting Job = job_1662171842192_0020, Tracking URL = http://prerana-VirtualBox:8088/proxy/application_1662171842192_0020/  
Kill Command = /home/prerana/hadoop-3.3.3/bin/mapred job -kill job_1662171842192_0020  
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0  
2022-09-03 10:05:11,005 Stage-3 map = 0%, reduce = 0%  
2022-09-03 10:05:24,712 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 3.94 sec  
MapReduce Total cumulative CPU time: 3 seconds 940 msec  
Ended Job = job_1662171842192_0020  
MapReduce Jobs Launched:  
Stage-Stage-3: Map: 1 Cumulative CPU: 3.94 sec HDFS Read: 9585 HDFS Write: 169 SUCCESS  
Total MapReduce CPU Time Spent: 3 seconds 940 msec  
OK  
CH      1      100  
JK      4       20  
NL      2       60  
NL      3      150  
Time taken: 63.03 seconds, Fetched: 4 row(s)
```

Verify the time taken for both Map Join and Normal Join.

## TASK 3 - Update ,delete entries in a Table and Query

Firstly you need to enable ACID Transactions to support transactional queries, one of the important property need to know is hive.txn.manager which is used to set Hive Transaction manager.

Below are the properties you need to enable ACID transactions.

```
SET hive.support.concurrency=true;
SET hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
# The following parameters are required for standalone hive metastore
SET hive.compactor.initiator.on=true;
SET hive.compactor.worker.threads=1;
```

To support ACID transactions you need to create a table with TBLPROPERTIES ('transactional'='true'); and the store type of the table should be ORC.

```
CREATE TABLE table_name (
  col_name data_type)
STORED AS ORC
TBLPROPERTIES ('transactional'='true');
```

Insert Records into the table

```
INSERT INTO table_name VALUES(col_value);
```

Hive UPDATE SQL query is used to update the existing records in a table. When WHERE clause not used, Hive updates all records in a table. By using WHERE clause you can specify a condition which records to update.

Update statement Syntax:

```
UPDATE tablename
SET column = value
WHERE expression;
```

(The WHERE clause is optional. If you omit the WHERE clause, all records in the table will be updated.)

Hive DELETE SQL query is used to delete the records from a table.

Delete statement Syntax:

```
DELETE FROM tablename  
WHERE expression;
```

(The WHERE clause is optional. If you omit the WHERE clause, all records in the table will be deleted.)

### Problem statement:

Create a Table with table name as "costs" and items with the following attributes

attribute	Description	DataType
id	item ID (primary key)	int
item_name	Item name	String
item_cost	Item Cost	double

Insert the below values into the table items



id	item_name	item_cost
1	chocolate	100
2	grape	50
3	chips	10
4	oranges	80
5	apples	90
6	chips	20
7	chocolate	90
8	grape	100
9	chips	40
10	oranges	70
11	apples	90
12	chips	20

Update item\_cost of chips to 30. **Take a screenshot of the terminal output and name it 4a.png.**

Delete all the rows with maximum item\_cost. **Take a screenshot of the terminal output and name it 4b.png.**

Write a query to find the total number of each item and check the number of mappers and reducers executed by that query. **Take a screenshot of the terminal output and name it 4c.png.**

# TASK 4 - Evaluation

This activity is graded and will be evaluated by two procedures. Both procedures are mandatory to be completed. **Deadline for this activity is 11:59pm on 7th September 2022.**

## Procedure 1 - Auto-Graded Evaluation

Run the following pyc file to evaluate your task. You can run this as many times until you get the right answer.

For RR Campus Students:

```
python3 eval-rr.pyc
```

For EC Campus Students:

```
python3 eval-ec.pyc
```

**Take a screenshot of the terminal output and name it 5a.png.**

## Procedure 2 - Google Form Evaluation

Submit the PDF containing all the screenshots specified throughout the activity in the Google Form. The PDF should be named as (SRN).pdf

Link for RR Campus Students: <https://forms.gle/oWqAwrv1y2S62w7NA>

Link for EC Campus Students: <https://forms.gle/ms6BzkKso1Ae4Uv67>