Session 8
HIVE BASICS
Assignment 1

TASK 1:

Create a database named 'custom'.
Create a table named temperature_data inside custom having below fields:
1. date (mm-dd-yyyy) format
2. zip code
3. temperature
The table will be loaded from comma-delimited file.
Load the dataset.txt (which is ',' delimited) in the table.

EXECUTION:

Create a database called "custom"

```
1.x releases.
hive> show databases
    > ;
OK
default
Time taken: 12.16 seconds, Fetched: 1 row(s)
hive> create database custom;
OK
Time taken: 0.254 seconds
hive> show databases;
OK
custom
default
Time taken: 0.074 seconds, Fetched: 2 row(s)
hive>
```

Create a table names "temperature_data" under custom database with columns as locationdate, zip code and temperature.

```
Time taken: 0.947 seconds
hive> create table temperature_data(locationdate string,zipcode string,temperatu
re int) row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 0.158 seconds
hive> show tables
    > ;
OK
employees
temperature_data
Time taken: 0.072 seconds, Fetched: 2 row(s)
hive>
```

Load data from "dataset.txt" in local file system to Hive table "temperature_data"

```
employees
temperature_data
Time taken: 0.072 seconds, Fetched: 2 row(s)
hive> LOAD DATA LOCAL INPATH '/home/acadgild/workspace/Hive/dataset.txt' OVERWRI
TE INTO TABLE temperature_data;
Loading data to table custom.temperature_data
OK
Time taken: 3.877 seconds
hive>
```

The data is loaded successfully in to Hive table. Select the loaded data from temperature_data table,

```
Time taken: 3.877 seconds
hive> select * from temperature_data;
OK
10-01-1990      123112  10
14-02-1991      283901  11
10-03-1990      381920  15
10-01-1991      302918  22
12-02-1990      384902  9
10-01-1991      123112  11
14-02-1990      283901  12
10-03-1991      381920  16
10-01-1990      302918  23
12-02-1991      384902  10
10-01-1993      123112  11
14-02-1994      283901  12
10-03-1993      381920  16
10-01-1994      302918  23
12-02-1991      384902  10
10-01-1991      123112  11
14-02-1990      283901  12
10-03-1991      381920  16
10-01-1990      302918  23
12-02-1991      384902  10
Time taken: 3.896 seconds, Fetched: 20 row(s)
hive>
```

TASK 2

- Fetch date and temperature from temperature_data where zip code is greater than 300000 and less than 399999.

Hive Query:

```
SELECT * FROM temperature_data WHERE zipcode > 300000 AND zipcode < 399999;
```

Output:

```
Time taken: 55.228 seconds, Fetched: 4 row(s)
hive> SELECT * FROM temperature_data WHERE zipcode > 300000 AND zipcode < 399999
;
OK
10-03-1990      381920  15
10-01-1991      302918  22
12-02-1990      384902  9
10-03-1991      381920  16
10-01-1990      302918  23
12-02-1991      384902  10
10-03-1993      381920  16
10-01-1994      302918  23
12-02-1991      384902  10
10-03-1991      381920  16
10-01-1990      302918  23
12-02-1991      384902  10
Time taken: 0.403 seconds, Fetched: 12 row(s)
hive>
```

- Calculate maximum temperature corresponding to every year from temperature_data table.

Hive Query:

```
SELECT YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-
yyyy')))),MAX(temperature) FROM temperature_data GROUP  BY
YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy'))));
```

Output:

```
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.31 sec   HDFS Read: 9618 HD
FS Write: 167 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 310 msec
OK
1990    23
1991    22
1993    16
1994    23
Time taken: 55.228 seconds, Fetched: 4 row(s)
hive>
```

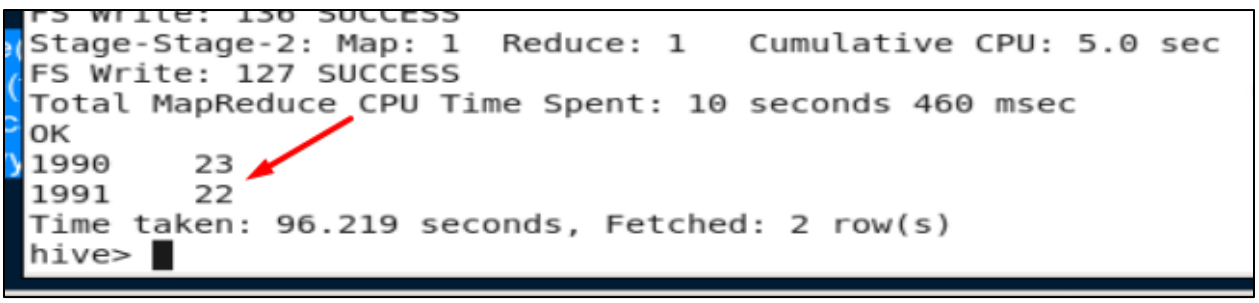
- Calculate maximum temperature from temperature_data table corresponding to those years which have at least 2 entries in the table.

  All the distinct years in dataset are having at least 2 entries. So, when we query the table to fetch records as per the condition provided in question, it returns all 4 years and the result is similar to previous question. So I have considered the condition for years which have at least 3 entries in the table.

  Hive Query:

```
SELECT YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-
yyyy')))),MAX(temperature) FROM temperature_data WHERE
YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy')))) IN
(SELECT YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-
yyyy')))) FROM temperature_data GROUP BY
YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy'))))
HAVING COUNT(*) > 2) GROUP BY
YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy'))));
```

  Output:

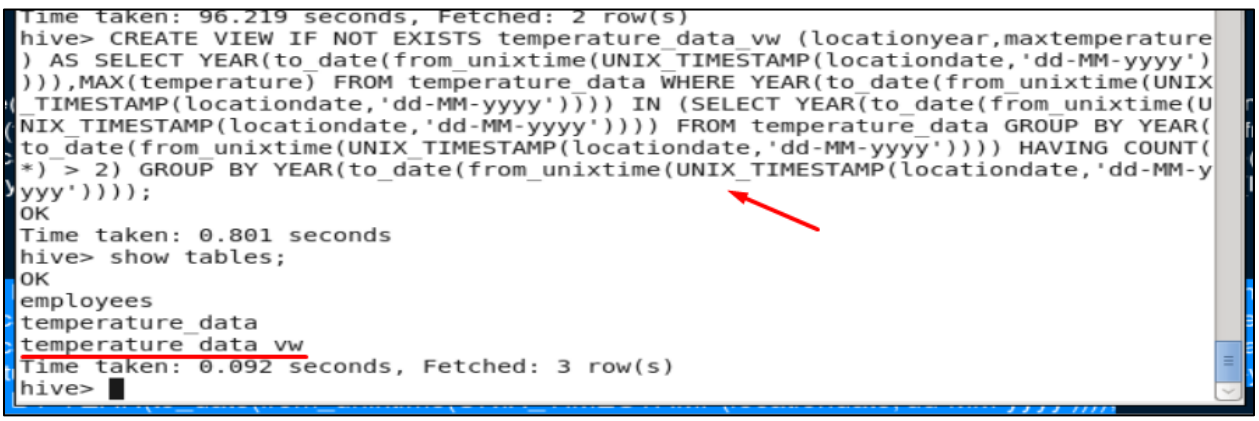


- Create a view on the top of last query, name it temperature_data_vw.

Hive Query:

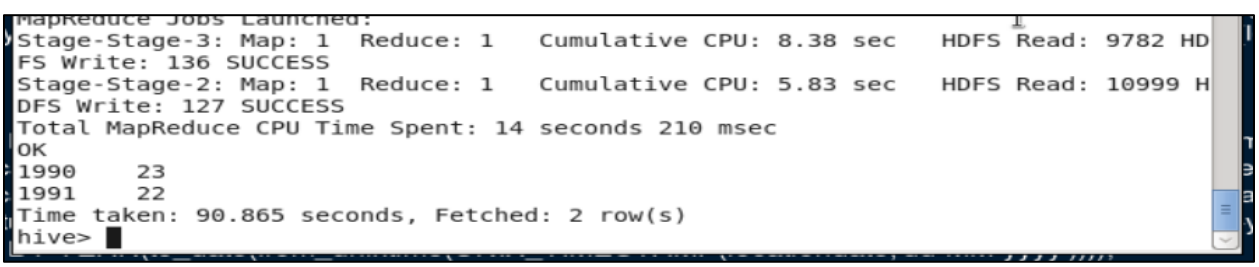
```
CREATE VIEW IF NOT EXISTS temperature_data_vw (locationyear,maxtemperature)
AS SELECT YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-
yyyy')))),MAX(temperature) FROM temperature_data WHERE
YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy')))) IN
(SELECT YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-
yyyy')))) FROM temperature_data GROUP BY
YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy'))))
HAVING COUNT(*) > 2) GROUP BY
YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy'))));
```

Output:



```
Time taken: 96.219 seconds, Fetched: 2 row(s)
hive> CREATE VIEW IF NOT EXISTS temperature_data_vw (locationyear,maxtemperature
) AS SELECT YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy')
))),MAX(temperature) FROM temperature_data WHERE YEAR(to_date(from_unixtime(UNIX
_TIMESTAMP(locationdate,'dd-MM-yyyy')))) IN (SELECT YEAR(to_date(from_unixtime(U
NIX_TIMESTAMP(locationdate,'dd-MM-yyyy')))) FROM temperature_data GROUP BY YEAR(
to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-yyyy')))) HAVING COUNT(
*) > 2) GROUP BY YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(locationdate,'dd-MM-y
yyy'))));
OK
Time taken: 0.801 seconds
hive> show tables;
OK
employees
temperature_data
temperature_data_vw
Time taken: 0.092 seconds, Fetched: 3 row(s)
hive>
```

Selecting the view "temperature_data_vw" returns the below data,



```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 8.38 sec   HDFS Read: 9782 HD
FS Write: 136 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.83 sec   HDFS Read: 10999 H
DFS Write: 127 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 210 msec
OK
1990    23
1991    22
Time taken: 90.865 seconds, Fetched: 2 row(s)
hive>
```

- Export contents from temperature_data_vw to a file in local file system, such that each file is '|' delimited.

Hive Query:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/acadgild/workspace/Hive/export' ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' SELECT * FROM temperature_data_vw;
```

Output:

```
Time taken: 20.544 seconds
hive> INSERT OVERWRITE LOCAL DIRECTORY '/home/acadgild/workspace/Hive/export' RO
W FORMAT DELIMITED FIELDS TERMINATED BY '|' SELECT * FROM temperature_data_vw;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20190114154738_5f5ec2d6-6cb5-4e4a-897e-89bb63690cce
Total jobs = 2
Launching Job 1 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_1547449962557_0010, Tracking URL = http://localhost:8088/prox
y/application_1547449962557_0010/
```

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost export]$ ls
000000_0
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost export]$ cat 000000_0
1990|23
1991|22
[acadgild@localhost export]$
```