Assignment-8

Task 1

Create a database named 'custom'.

Create a table named temperature_data inside custom having below fields:

1. date (mm-dd-yyyy) format

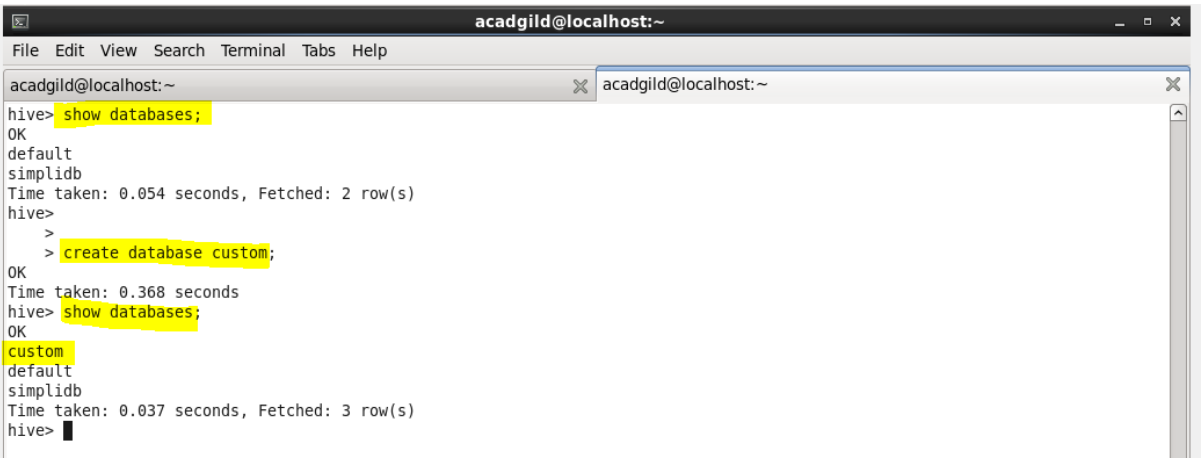
2. zip code

3. temperature

The table will be loaded from comma-delimited file.

Load the dataset.txt (which is ',' delimited) in the table.

Database with name '**custom'** has been created. Created database has been shown in the below screenshot along with the command to create the database.



A table has been created with name **TEMPERATURE_DATA** as shown in the below screenshot.

Data present in the dataset_Session_14 has been loaded into the table created.

```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Downloads/dataset_Session_14.txt' into table TEMPERATURE_DATA;
Loading data to table default.temperature_data
OK
Time taken: 2.702 seconds
hive>
```

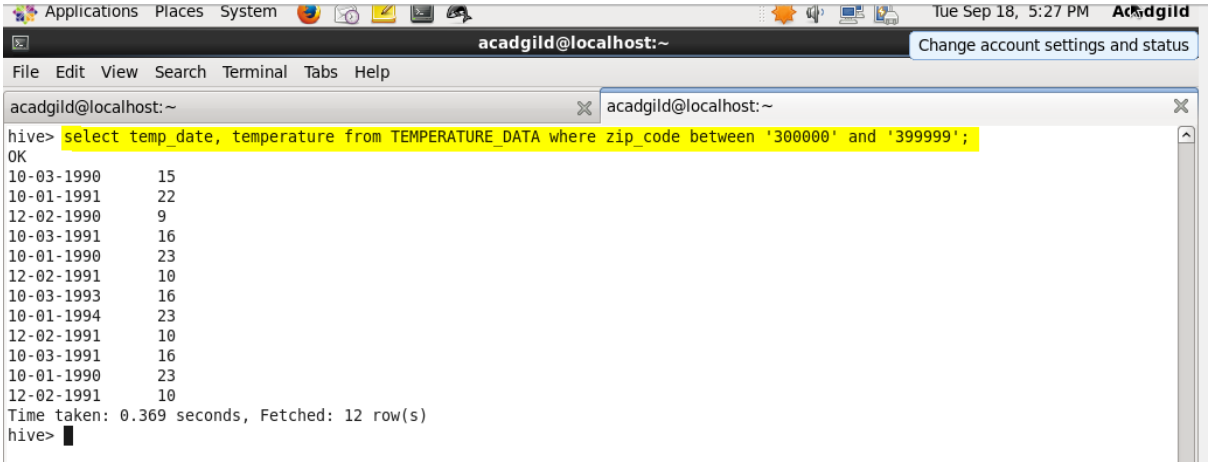
Contents of table after loading the data into the table has been shown in the below screenshot.

```
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: 0.368 seconds, Fetched: 20 row(s)
hive>
```

Task 2

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

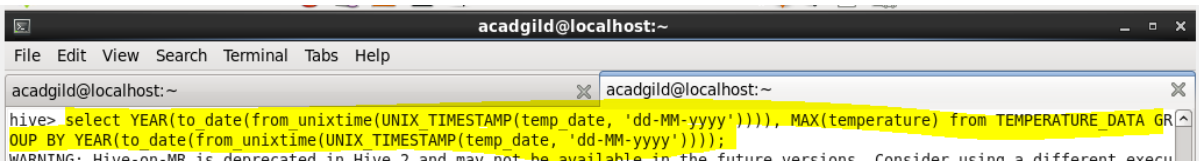
Date and temperature from table, if **zip_code** is between **300000** and **399999** has been fetched as shown in the below screenshot using between condition in the select query.



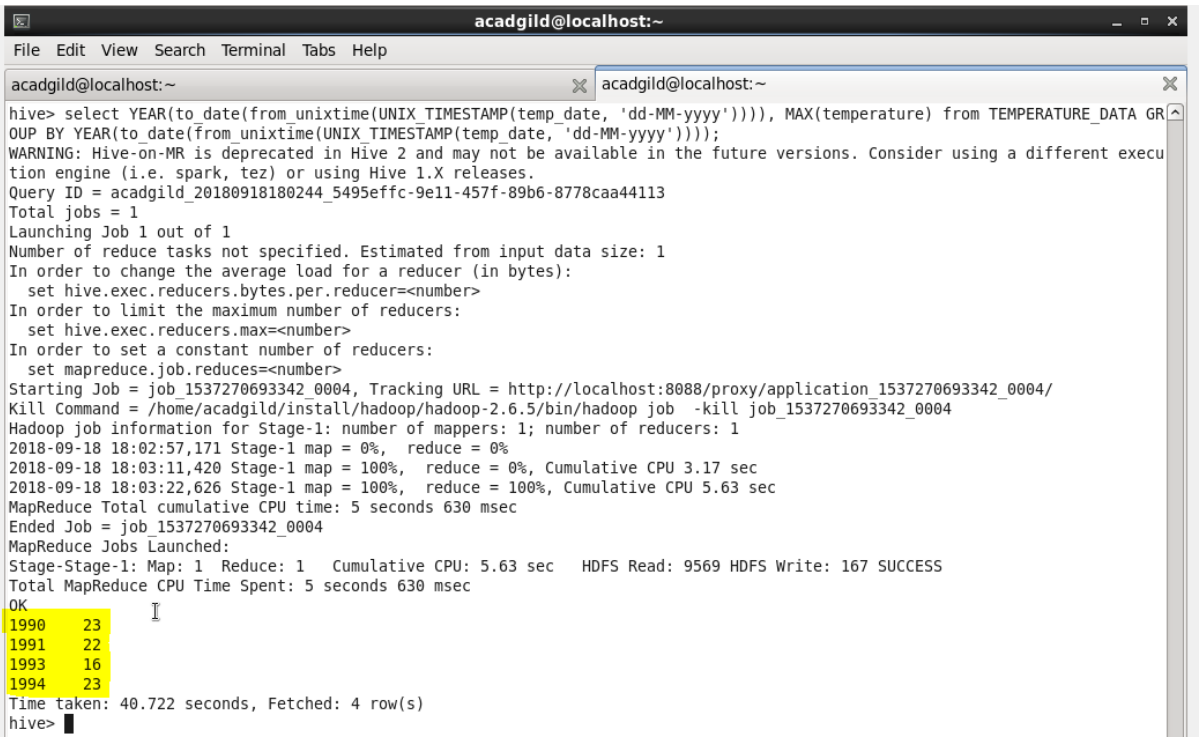
2) **Calculate maximum temperature corresponding to every year from temperature_data table.**

- Default date format in the table is dd-mm-yyyy, it has been converted into yyyy-mm-dd format using below command.
  to_date(from unixtime(UNIX_TIMESTAMP(column_name, 'dd-MM-yyyy')))
- Now year is taken using the command '**YEAR(yyyy-mm-dd)**'.
- Used **GROUP BY** condition to group the records based on year.
- Maximum temperature corresponding to every year is retrieved using '**MAX(temperature)'.**

Query used is shown in the below screenshot.



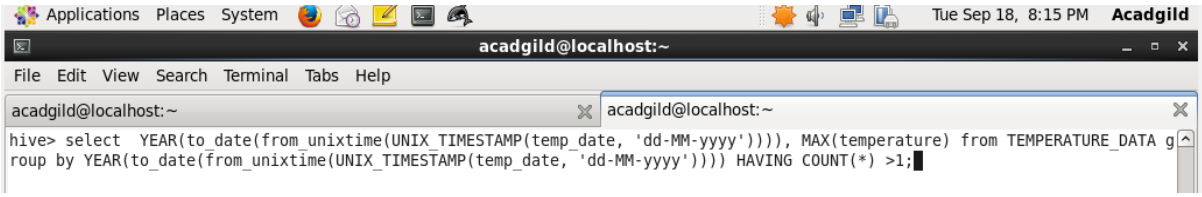
The required output is shown in the below screenshot.



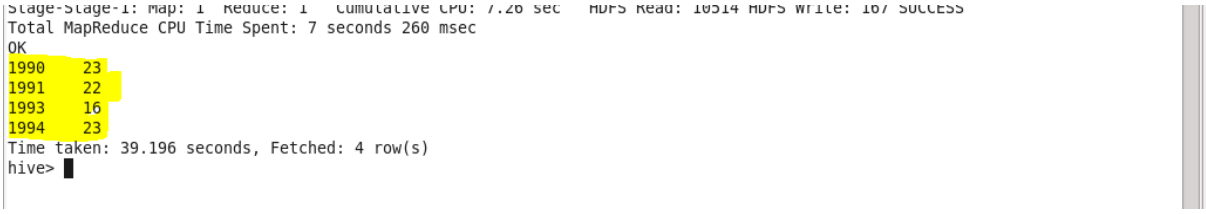
3) **Calculate maximum temperature from temperature_data table corresponding to those years which have at least 2 entries in the table.**

- Default date format in the table is dd-mm-yyyy, it has been converted into yyyy-mm-dd format using below command.
  to_date(from unixtime(UNIX_TIMESTAMP(column_name, 'dd-MM-yyyy')))
- Now year is taken using the command '**YEAR(yyyy-mm-dd)**'.
- Used **GROUP BY** condition to group the records based on year.
- To get Years which have at least two entries in the table, I have used **'HAVING COUNT(*) > 1'.**
- Maximum temperature corresponding to every year is retrieved using **'MAX(temperature)'.**

Query used is shown in the below screenshot



The required output is shown in the below screenshot.



4) **Create a view on the top of last query, name it temperature_data_vw.**

Created view for previous query as shown below.

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
acadgild@localhost:~    acadgild@localhost:~
hive> create view if not exists TEMPERATURE_DATA_VW as select  YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(temp_date, 'dd-MM-yy
yy')))), MAX(temperature) from TEMPERATURE_DATA group by YEAR(to_date(from_unixtime(UNIX_TIMESTAMP(temp_date, 'dd-MM-yyyy')))
) HAVING COUNT(*) >1;
OK
Time taken: 0.312 seconds
hive>
```

The contents of the query is shown in the below screenshot.

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
acadgild@localhost:~    acadgild@localhost:~
hive> select * from TEMPERATURE_DATA_VW;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180918202939_57ec141c-cfba-4f40-a034-91cb0917ad79
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_1537270693342_0013, Tracking URL = http://localhost:8088/proxy/application_1537270693342_0013/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537270693342_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-18 20:29:52,331 Stage-1 map = 0%,  reduce = 0%
2018-09-18 20:30:03,479 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.02 sec
2018-09-18 20:30:14,555 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.72 sec
MapReduce Total cumulative CPU time: 6 seconds 720 msec
Ended Job = job_1537270693342_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.72 sec   HDFS Read: 10587 HDFS Write: 167 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 720 msec
OK
1990    23
1991    22
1993    16
1994    23
Time taken: 37.411 seconds, Fetched: 4 row(s)
hive>
```

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

Exported contents of the view **TEMPERATURE_DATA_VW** to local file system with **'|'** delimited using the command shown in the below screenshot

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
acadgild@localhost:~        acadgild@localhost:~
hive> insert overwrite local directory '/home/acadgild/Desktop' row format delimited fields terminated by '|' select * from T
EMPERATURE_DATA_VW;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180918203612_5e414c43-0ec9-4b81-b9fb-d793275800a9
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_1537270693342_0014, Tracking URL = http://localhost:8088/proxy/application_1537270693342_0014/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537270693342_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-18 20:36:24,835 Stage-1 map = 0%,  reduce = 0%
2018-09-18 20:36:37,654 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.53 sec
2018-09-18 20:36:49,831 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.49 sec
MapReduce Total cumulative CPU time: 7 seconds 490 msec
Ended Job = job_1537270693342_0014
Moving data to local directory /home/acadgild/Desktop
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.49 sec   HDFS Read: 10187 HDFS Write: 32 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 490 msec
OK
Time taken: 38.9 seconds
hive>
```

Exported the file to desktop as shown in the below screenshot.

The contents of the exported file is as shown in the below screenshot.