Session 8
HIVE BASICS
Assignment 1

- ○ By Prachi Mohite

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.

Solution Approach –

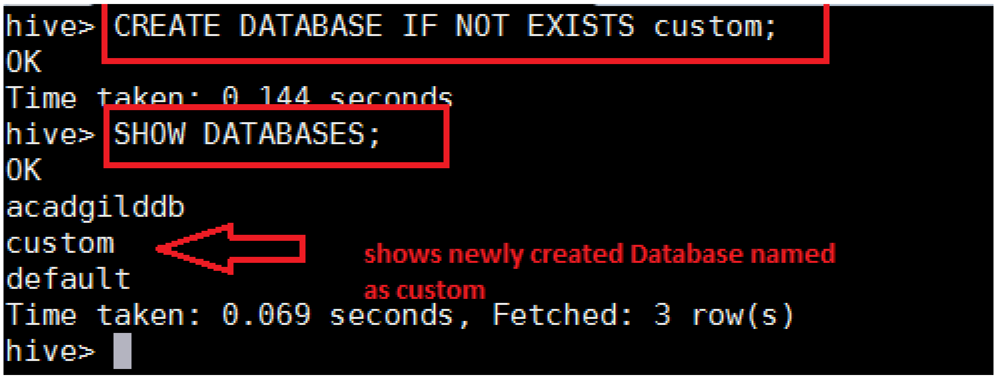
To execute the HIVE commands we are using HIVE Command line. It has two modes of interaction

1. **Interactive Mode**
    a. Here we can submit the actual hive commands (queries) on HIVE CLI directly
2. **Non Interactive Mode**
    a. Here we need to execute the HIVE script
    b. e.g HIVE –f name_of_script.q

Create a Database in HIVE

Command used for the same

- CREATE DATABASE custom;
    - This command will throw exception if database is already created.
- CREARE DATABASE IF NOT EXISTS custom;
    - This command will create database if the database does not exist.



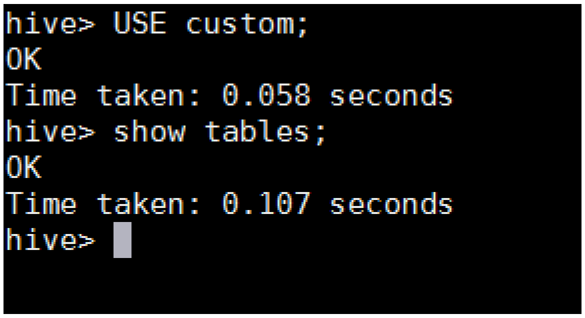
```
hive> CREATE DATABASE IF NOT EXISTS custom;
OK
Time taken: 0.144 seconds
hive> SHOW DATABASES;
OK
acadgilddb
custom
default
Time taken: 0.069 seconds, Fetched: 3 row(s)
hive>
```

shows newly created Database named as custom

Create a table named temperature_data inside custom having below fields:

1. date (mm-dd-yyyy) format
2. zip code
3. temperature

- To create table inside ‘custom’ database we have to choose database ‘custom’ as active database and the command used for the same is
  - USE Database_Name;
  - Show tables is used to get the list of tables belonging to database.



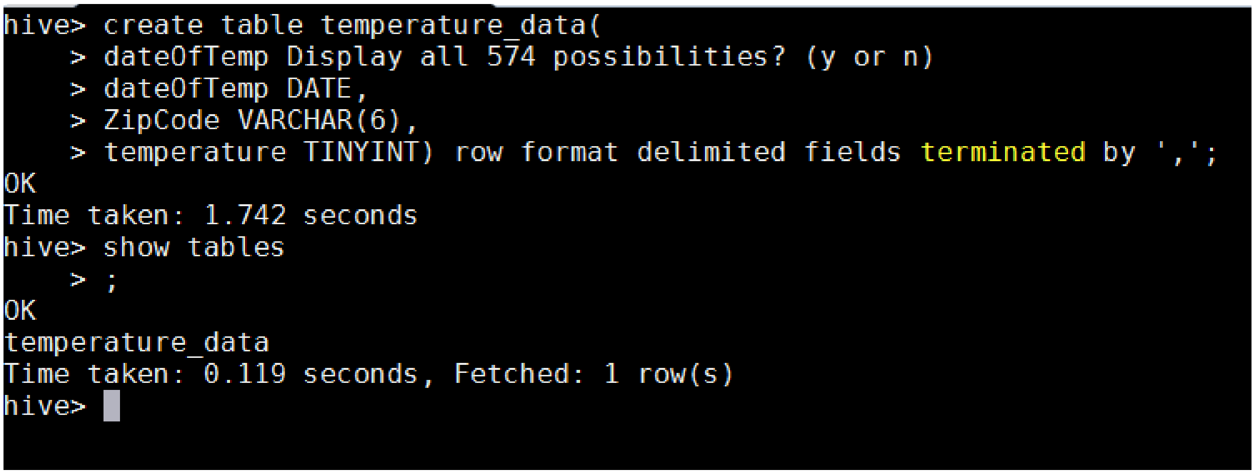
```
hive> USE custom;
OK
Time taken: 0.058 seconds
hive> show tables;
OK
Time taken: 0.107 seconds
hive>
```

- To create table we need to use the below command

```
CREATE TABLE  temperature_data  (
 dateofMeasurement DATE,
 zip_code VARCHAR(6),
temperature TINYINT ) row format delimited fields terminated by ',';
```

DataTypes used

- Date - to capture the date when temperature was measured
- VARCHAR(6) – to store the ZIP Code (as per standards it has 6 digit value)
- TINYINT – for temperature as value is not beyond 100 and TINYINT is 1 byte signed integer (-128 to 127)



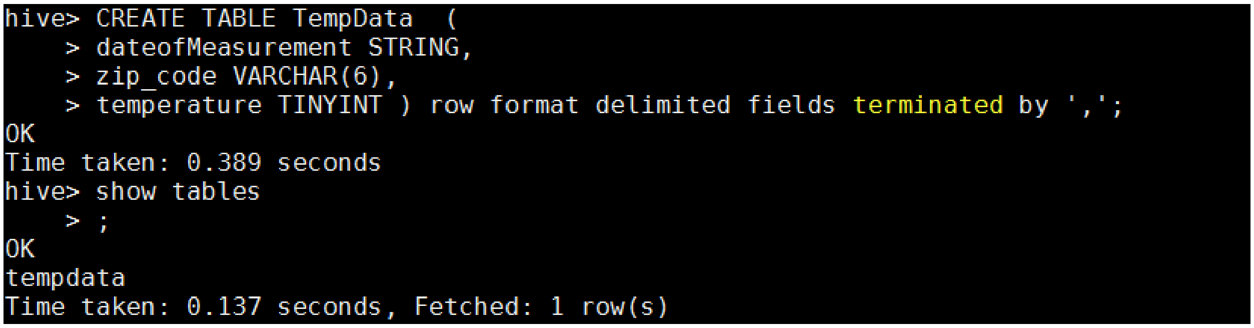
```
hive> create table temperature_data(
    > dateOfTemp Display all 574 possibilities? (y or n)
    > dateOfTemp DATE,
    > ZipCode VARCHAR(6),
    > temperature TINYINT) row format delimited fields terminated by ',';
OK
Time taken: 1.742 seconds
hive> show tables
    > ;
OK
temperature_data
Time taken: 0.119 seconds, Fetched: 1 row(s)
hive> 
```

Load data into table from a file

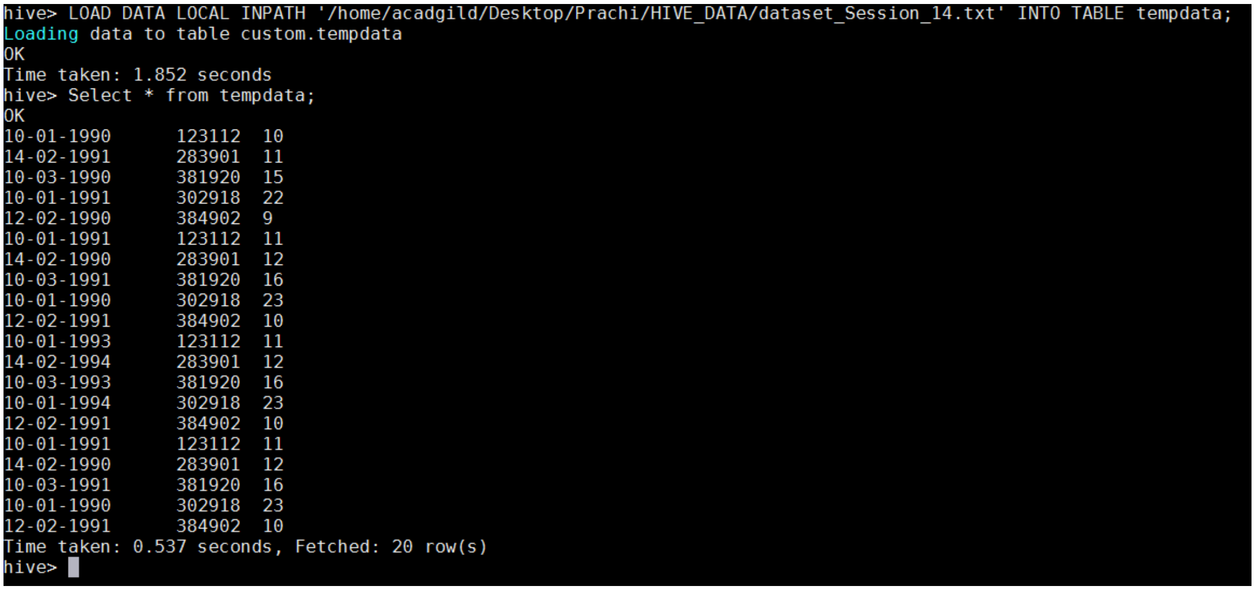
- This can be achieved by using load command
- LOAD DATA LOCAL INPATH '*file_path*' INTO TABLE <*table_name*>
- As this Table has date column in it we have two approaches to deal with reading / loading date column
    - Load data in temporary table where date column is stored as string which then loaded in actual table with column having data type as 'Date'
    - Load data in actual table with string column and while performing the actions use date / timestamping built in functions.

Approach A for creating a Table and loading Data

- Create a temp table with string column to hold date values
  CREATE TABLE TempData (
  dateofMeasurement STRING,
  zip_code VARCHAR(6),
  temperature TINYINT) row format delimited fields terminated by ',';

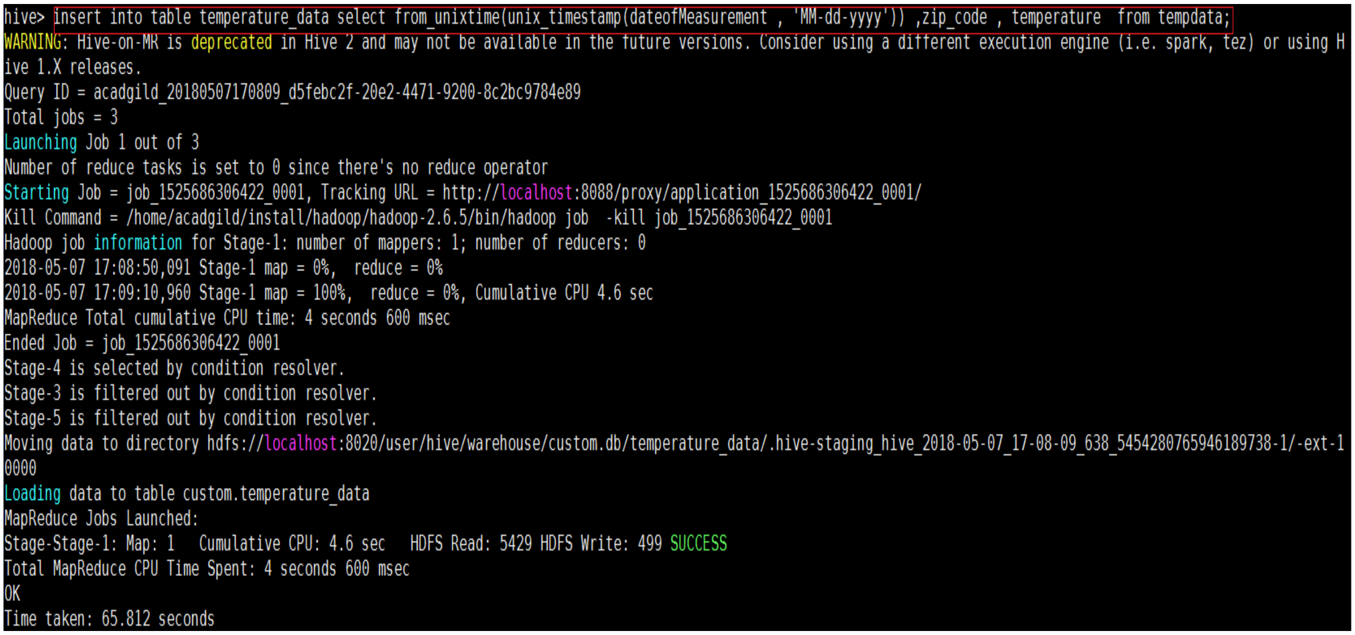


```
hive> CREATE TABLE TempData  (
    > dateofMeasurement STRING,
    > zip_code VARCHAR(6),
    > temperature TINYINT ) row format delimited fields terminated by ',';
OK
Time taken: 0.389 seconds
hive> show tables
    > ;
OK
tempdata
Time taken: 0.137 seconds, Fetched: 1 row(s)
```



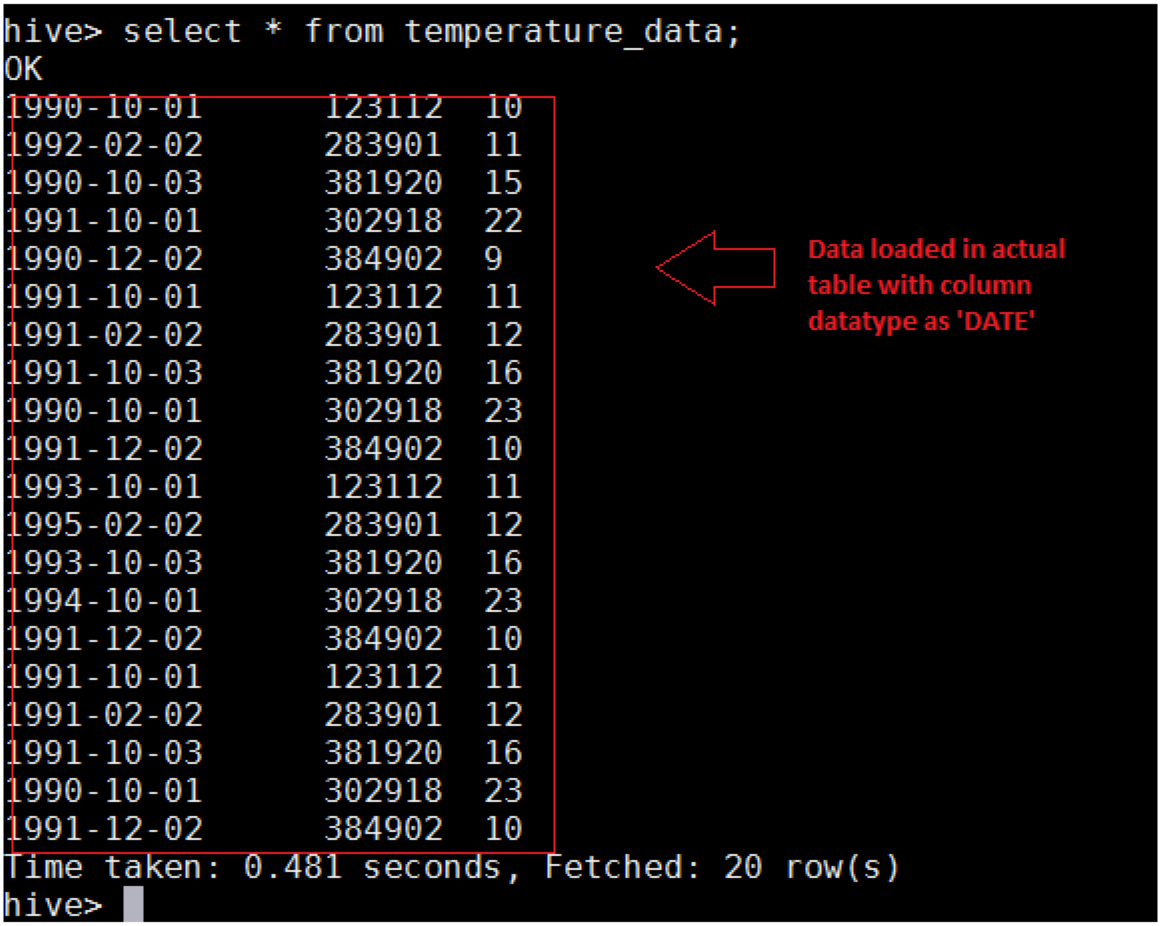
```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Desktop/Prachi/HIVE_DATA/dataset_Session_14.txt' INTO TABLE tempdata;
Loading data to table custom.tempdata
OK
Time taken: 1.852 seconds
hive> Select * from tempdata;
OK
10-01-1990      123112  10
14-02-1991      283901  11
10-03-1990      381920  15
10-01-1991      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: 0.537 seconds, Fetched: 20 row(s)
hive> 
```

- Once temp table is created load data from temp table to actual temperature_data table.



```
hive> insert into table temperature_data select from_unixtime(unix_timestamp(dateofMeasurement , 'MM-dd-yyyy')) ,zip_code , temperature  from tempdata;
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 H
ive 1.X releases.
Query ID = acadgild_20180507170809_d5febc2f-20e2-4471-9200-8c2bc9784e89
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_1525686306422_0001, Tracking URL = http://localhost:8088/proxy/application_1525686306422_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525686306422_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-05-07 17:08:50,091 Stage-1 map = 0%,  reduce = 0%
2018-05-07 17:09:10,960 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.6 sec
MapReduce Total cumulative CPU time: 4 seconds 600 msec
Ended Job = job_1525686306422_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://localhost:8020/user/hive/warehouse/custom.db/temperature_data/.hive-staging_hive_2018-05-07_17-08-09_638_5454280765946189738-1/-ext-1
0000
Loading data to table custom.temperature_data
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 4.6 sec   HDFS Read: 5429 HDFS Write: 499 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 600 msec
OK
Time taken: 65.812 seconds
```

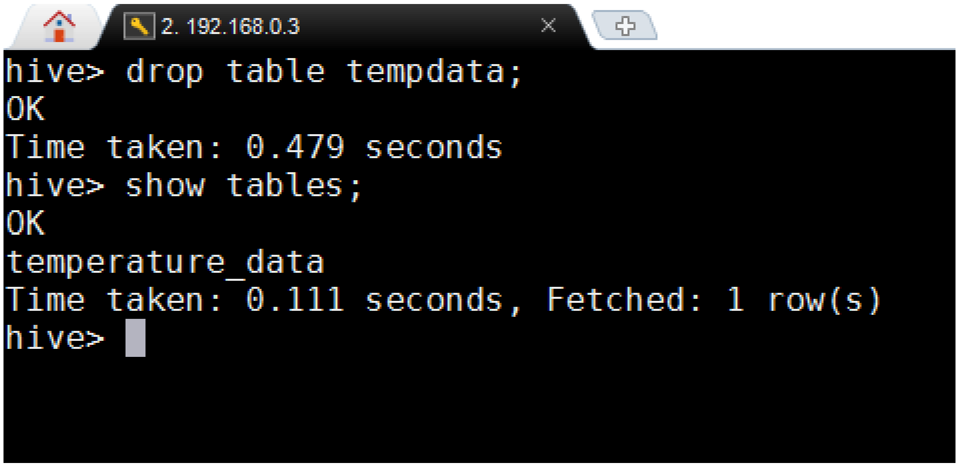
Verify if data loaded properly by executing the select command



```
hive> select * from temperature_data;
OK
1990-10-01      123112  10
1992-02-02      283901  11
1990-10-03      381920  15
1991-10-01      302918  22
1990-12-02      384902  9
1991-10-01      123112  11
1991-02-02      283901  12
1991-10-03      381920  16
1990-10-01      302918  23
1991-12-02      384902  10
1993-10-01      123112  11
1995-02-02      283901  12
1993-10-03      381920  16
1994-10-01      302918  23
1991-12-02      384902  10
1991-10-01      123112  11
1991-02-02      283901  12
1991-10-03      381920  16
1990-10-01      302918  23
1991-12-02      384902  10
Time taken: 0.481 seconds, Fetched: 20 row(s)
hive>
```

Data loaded in actual table with column datatype as 'DATE'

Drop the temp table

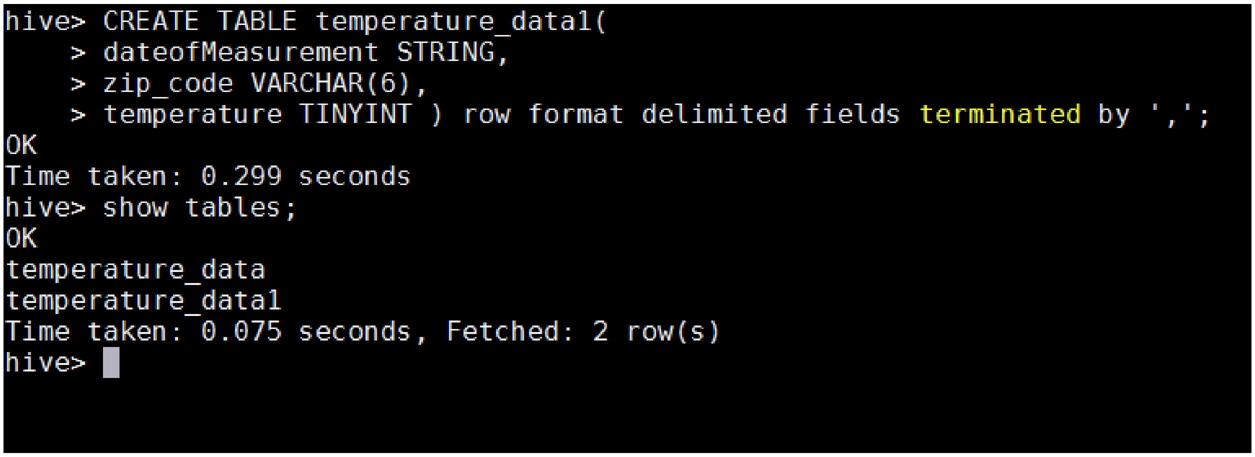


```
hive> drop table tempdata;
OK
Time taken: 0.479 seconds
hive> show tables;
OK
temperature_data
Time taken: 0.111 seconds, Fetched: 1 row(s)
hive>
```

Approach B: Load data into temperature data table with date column as string and while performing operations use date functions.

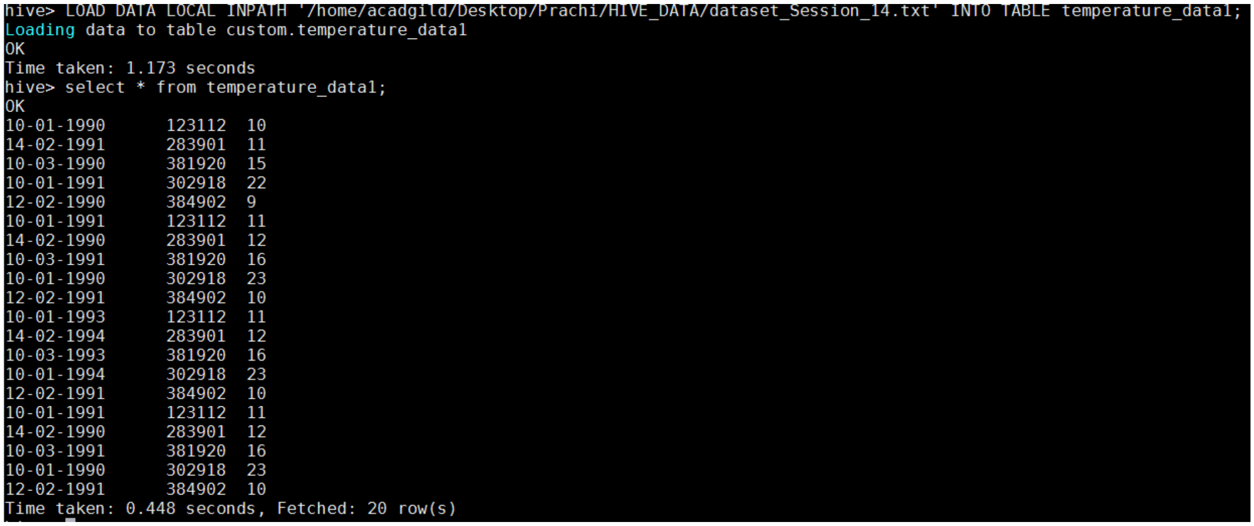
We will be using approach B for this assignment

- Created new table named as ‘temperature_data1’ where date will be saved as string
  CREATE TABLE temperature_data1(
  dateofMeasurement STRING,
  zip_code VARCHAR(6),
  temperature TINYINT) row format delimited fields terminated by ',';



```
hive> CREATE TABLE temperature_data1(
    > dateofMeasurement STRING,
    > zip_code VARCHAR(6),
    > temperature TINYINT ) row format delimited fields terminated by ',';
OK
Time taken: 0.299 seconds
hive> show tables;
OK
temperature_data
temperature_data1
Time taken: 0.075 seconds, Fetched: 2 row(s)
hive> 
```

Loaded data from file and verified if the data is loaded

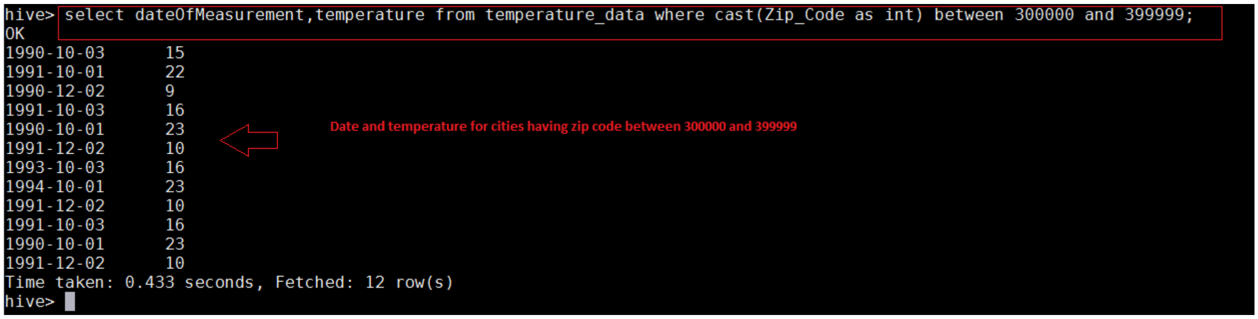


```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Desktop/Prachi/HIVE_DATA/dataset_Session_14.txt' INTO TABLE temperature_data1;
Loading data to table custom.temperature_data1
OK
Time taken: 1.173 seconds
hive> select * from temperature_data1;
OK
10-01-1990      123112  10
14-02-1991      283901  11
10-03-1990      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: 0.448 seconds, Fetched: 20 row(s)
```

Task 2

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

- Here we have to write a select statement with where clause
- As we have taken ZipCode as varchar(6), we need to use convert function (builtin) which will convert Varchar to int for comparision



```
hive> select dateOfMeasurement,temperature from temperature_data where cast(Zip_Code as int) between 300000 and 399999;
OK
1990-10-03      15
1991-10-01      22
1990-12-02      9
1991-10-03      16
1990-10-01      23
1991-12-02      10
1993-10-03      16
1994-10-01      23
1991-12-02      10
1991-10-03      16
1990-10-01      23
1991-12-02      10
Time taken: 0.433 seconds, Fetched: 12 row(s)
hive> 
```

Date and temperature for cities having zip code between 300000 and 399999

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

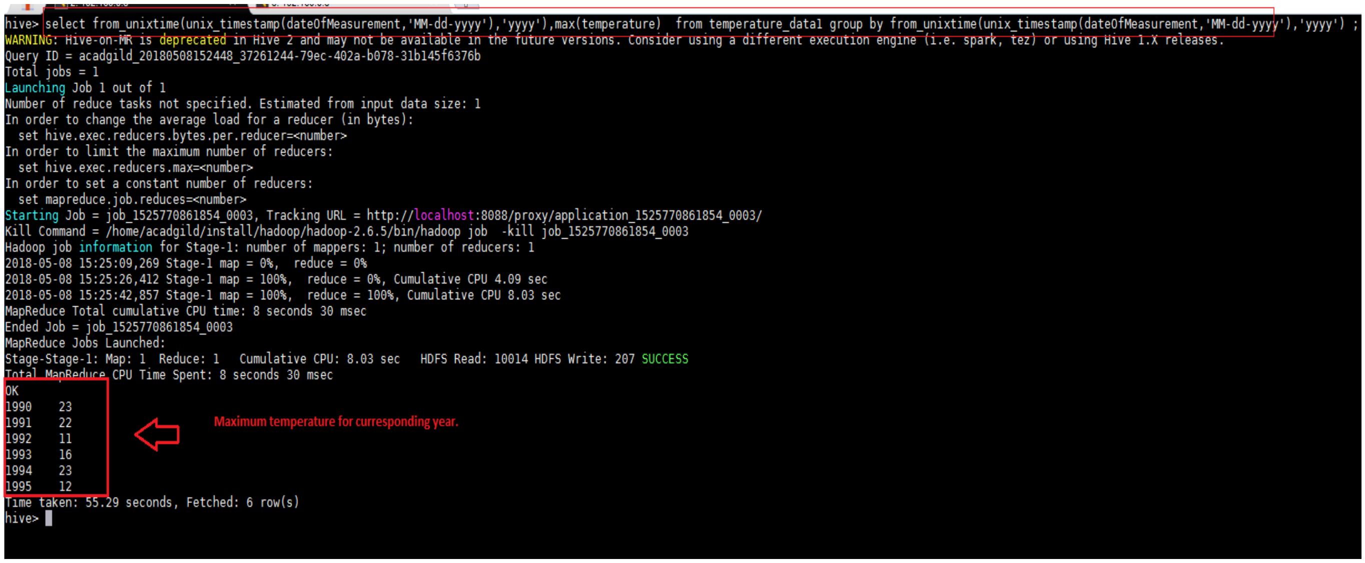
- Get the maximum temperature of every year, we need to group the table based on year of the date when the temperature is taken.

Grouping on Table temperature_data1 where date column is saved as string

- Here we have to convert string date to date and get the year from date this is achieved with below date time functions
  - Unix_timestamp(stringDate,datePattern)
    - Unix_timestamp(dateOfMeasurement,'MM-dd-yyyy')
  - The above functions returns the seconds which then converted to datetime column with date format 'yyyy'
    - From_unixtime(seconds,datePattern)
    - From_unixtime (Unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy')

- Group on the year and get the maximum temperature of the year
  - Grouping is done on the extracted year using group by function
    - Group By From_unixtime (Unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy')
- Maximum Temperature is retrieved with the help of MAX function

  - Select From_unixtime (Unix_timestamp(dateOfMeasurement,’MM-dd-yyyy’),’yyyy’) as year , MAX(temperature) from temperature_data1 group by From_unixtime (Unix_timestamp(dateOfMeasurement,’MM-dd-yyyy’),’yyyy’)



```
hive> select from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy'),max(temperature)  from temperature_data1 group by from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy') ;
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 = acadgild_20180508152448_37261244-79ec-402a-b078-31b145f6376b
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_1525770861854_0003, Tracking URL = http://localhost:8088/proxy/application_1525770861854_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525770861854_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-05-08 15:25:09,269 Stage-1 map = 0%,  reduce = 0%
2018-05-08 15:25:26,412 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.09 sec
2018-05-08 15:25:42,857 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.03 sec
MapReduce Total cumulative CPU time: 8 seconds 30 msec
Ended Job = job_1525770861854_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.03 sec   HDFS Read: 10014 HDFS Write: 207 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 30 msec
OK
1990    23
1991    22
1992    11
1993    16
1994    23
1995    12
Time taken: 55.29 seconds, Fetched: 6 row(s)
hive>
```

Maximum temperature for curresponding year.

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

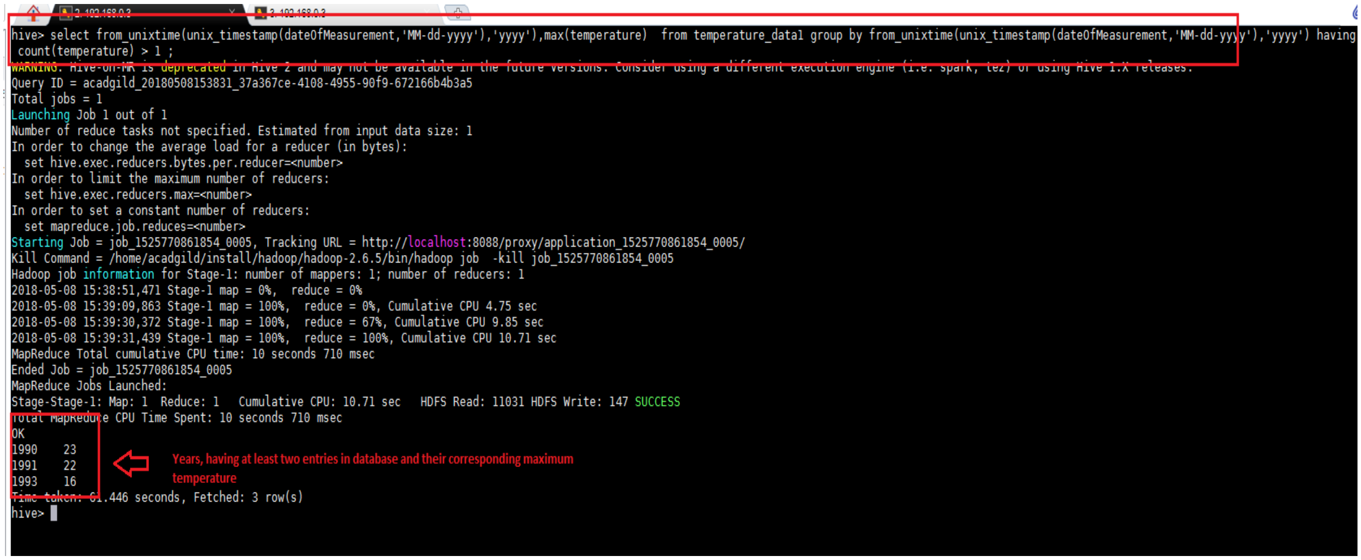
Solution Approach

- Get the maximum temperature of every year, we need to group the table based on year of the date when the temperature is taken.
- Also we need to check the count of the year is greater than 1 (i.e. having at least 2 entries)
- Here we have to convert string date to date and get the year from date this is achieved with below date time functions
  - Unix_timestamp(stringDate,datePattern)
    - Unix_timestamp(dateOfMeasurement,’MM-dd-yyyy’)
  - The above functions returns the seconds which then converted to datetime column with date format ‘yyyy’
    - From_unixtime(seconds,datePattern)
    - From_unixtime (Unix_timestamp(dateOfMeasurement,’MM-dd-yyyy’),’yyyy’)

- Group on the year and get the maximum temperature of the year
  - Grouping is done on the extracted year using group by function

    - Group By From_unixtime (Unix_timestamp(dateOfMeasurement,’MM-dd-yyyy’),’yyyy’)

- Maximum Temperature is retrieved with the help of MAX function
- Now we have to check it should have at least 2 entries for a year which can be achieved using Having clause with group by statement.
    - select from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy'),max(temperature) from temperature_data1 group by from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy') having count(temperature) > 1 ;



```
hive> select from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy'),max(temperature)  from temperature_data1 group by from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy') having
 count(temperature) > 1 ;
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 = acadgild_20180508153831_37a367ce-4108-4955-90f9-672166b4b3a5
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_1525770861854_0005, Tracking URL = http://localhost:8088/proxy/application_1525770861854_0005/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525770861854_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-05-08 15:38:51,471 Stage-1 map = 0%,  reduce = 0%
2018-05-08 15:39:09,863 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.75 sec
2018-05-08 15:39:30,372 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 9.85 sec
2018-05-08 15:39:31,439 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.71 sec
MapReduce Total cumulative CPU time: 10 seconds 710 msec
Ended Job = job_1525770861854_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.71 sec   HDFS Read: 11031 HDFS Write: 147 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 710 msec
OK
1990    23
1991    22
1993    16
Time taken: 61.446 seconds, Fetched: 3 row(s)
hive>
```

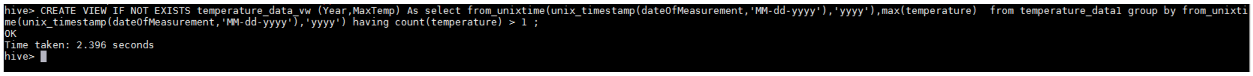
Years, having at least two entries in database and their corresponding maximum temperature

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

Views are generated based on user requirements. We can save any result set data as a view. The usage of view in Hive is same as that of the view in SQL. It is a standard RDBMS concept. We can execute all DML operations on a view.

We can create a view at the time of executing a SELECT statement.

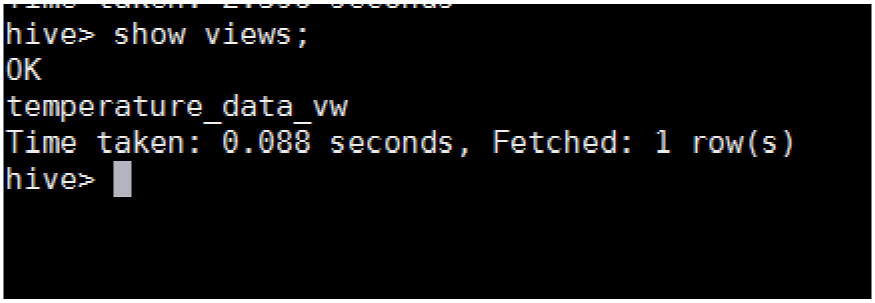
CREATE VIEW IF NOT EXISTS temperature_data_vw (Year,MaxTemp) As select from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy'),max(temperature) from temperature_data1 group by from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy') having count(temperature) > 1 ;



```
hive> CREATE VIEW IF NOT EXISTS temperature_data_vw (Year,MaxTemp) As select from_unixtime(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy'),max(temperature)  from temperature_data1 group by from_unixti
me(unix_timestamp(dateOfMeasurement,'MM-dd-yyyy'),'yyyy') having count(temperature) > 1 ;
OK
Time taken: 2.396 seconds
hive>
```

To Verify list of views

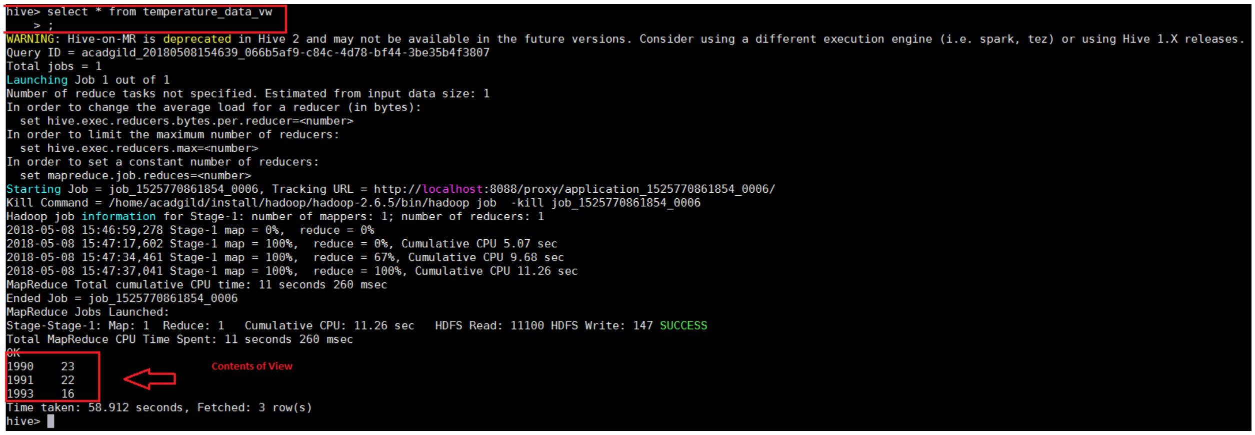
- show views; command is used



```
hive> show views;
OK
temperature_data_vw
Time taken: 0.088 seconds, Fetched: 1 row(s)
hive>
```

To verify the data inside view

- select command is used



```
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 execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180508154639_066b5af9-c84c-4d78-bf44-3be35b4f3807
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_1525770861854_0006, Tracking URL = http://localhost:8088/proxy/application_1525770861854_0006/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525770861854_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-05-08 15:46:59,278 Stage-1 map = 0%,  reduce = 0%
2018-05-08 15:47:17,602 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.07 sec
2018-05-08 15:47:34,461 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 9.68 sec
2018-05-08 15:47:37,041 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 11.26 sec
MapReduce Total cumulative CPU time: 11 seconds 260 msec
Ended Job = job_1525770861854_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 11.26 sec   HDFS Read: 11100 HDFS Write: 147 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 260 msec
OK
1990    23
1991    22
1993    16
Time taken: 58.912 seconds, Fetched: 3 row(s)
hive>
```

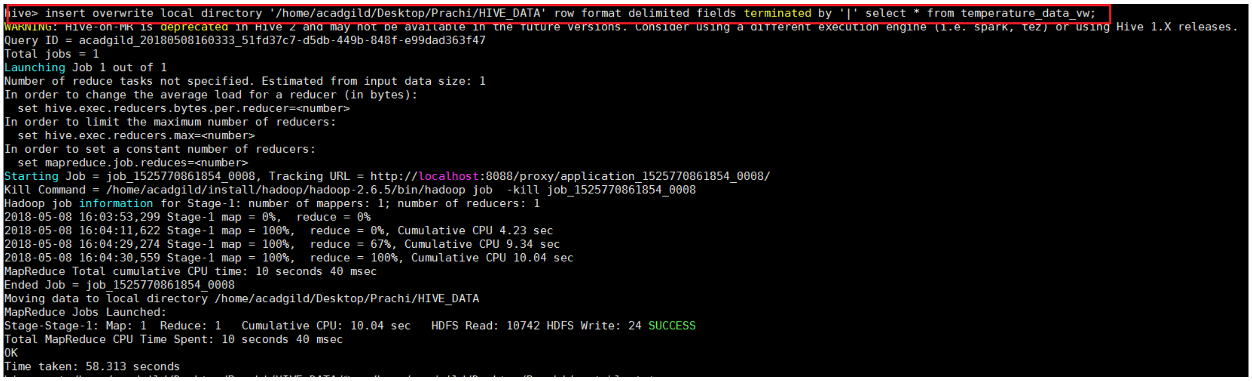
2.5 Export contents from temperature_data_vw to a file in local file system, such that each field is '|' delimited.

Approach One (Hive Insert Overwrite a Directory):

This approach writes the contents of a Hive table to a local path (linux) in as many files as it needs. It then uses a Linux "cat" command to merge all files to one csv.

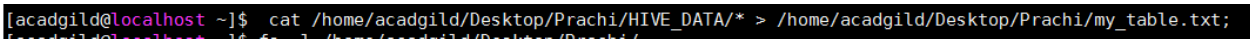
1. Command issued to Hive that selects all records from a table in Hive, separates the fields/columns by a "|", and writes the file to a local directory (wiping anything previously in that path).
2. Cat command issued to get/merge all part files (remember, the output was from a Map/Reduce job) in directory into a single .csv file.

- insert overwrite local directory ‘/home/acadgild/Desktop/Prachi/HIVE_DATA/’ row format delimited fields terminated by ‘|’ select * from temperature_data_vw;



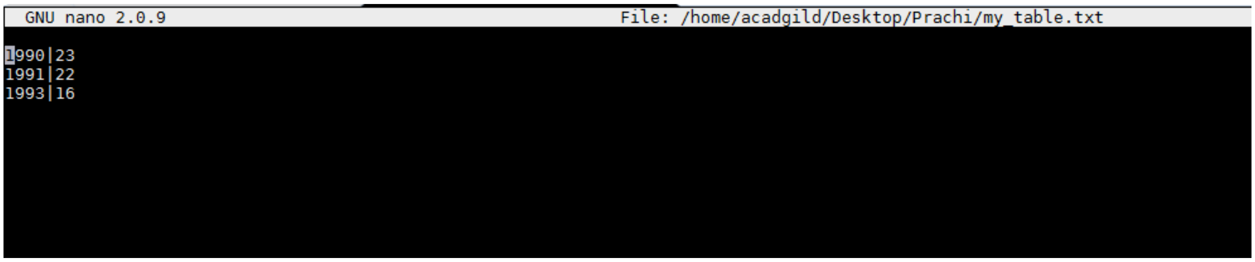
```
hive> insert overwrite local directory '/home/acadgild/Desktop/Prachi/HIVE_DATA' row format delimited fields terminated by '|' select * from temperature_data_vw;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180508160333_51fd37c7-d5db-449b-848f-e99dad363f47
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_1525770861854_0008, Tracking URL = http://localhost:8088/proxy/application_1525770861854_0008/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525770861854_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-05-08 16:03:53,299 Stage-1 map = 0%,  reduce = 0%
2018-05-08 16:04:11,622 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.23 sec
2018-05-08 16:04:29,274 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 9.34 sec
2018-05-08 16:04:30,559 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.04 sec
MapReduce Total cumulative CPU time: 10 seconds 40 msec
Ended Job = job_1525770861854_0008
Moving data to local directory /home/acadgild/Desktop/Prachi/HIVE_DATA
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.04 sec   HDFS Read: 10742 HDFS Write: 24 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 40 msec
OK
Time taken: 58.313 seconds
```

- Cat command issued to get/merge all part files



```
[acadgild@localhost ~]$  cat /home/acadgild/Desktop/Prachi/HIVE_DATA/* > /home/acadgild/Desktop/Prachi/my_table.txt;
```

- Final exported output



```
  GNU nano 2.0.9                          File: /home/acadgild/Desktop/Prachi/my_table.txt

1990|23
1991|22
1993|16
```

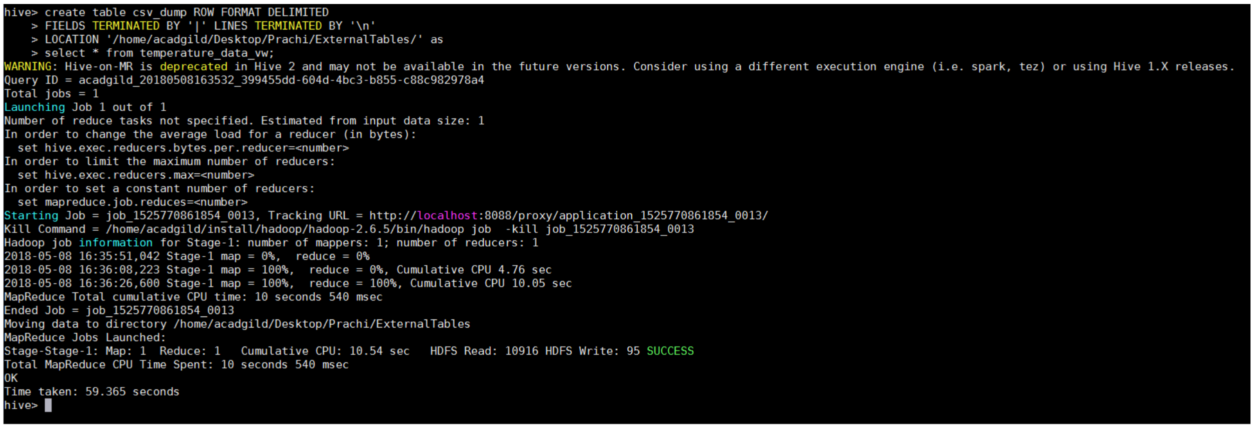
Approach Two (Hive CSV Dump External Table):

- HIVE has two types of tables
  - Internal (Managed Tables)
    - When we load data into a Managed table, then Hive moves data into Hive warehouse directory.
    - if we drop the table his will delete the table metadata including its data. The data no longer exists anywhere. This is what it means for HIVE to manage the data.
  - External Tables
    - we can control the creation and deletion of the data. The location of the external data is specified at the table creation time.

- The important thing to notice is that when we drop an external table, Hive will leave the data untouched and only delete the metadata.

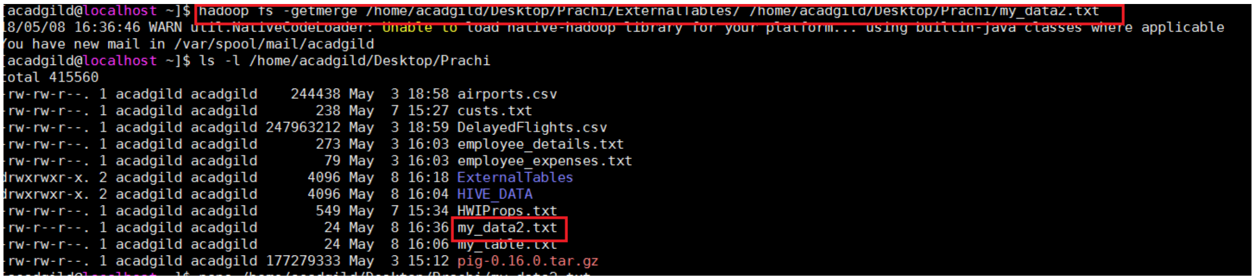
This approach writes a table’s contents to an internal Hive table called csv_dump, delimited by “|” (this can be specified by user) — stored in HDFS as usual. It then uses a hadoop filesystem command called “getmerge” that does the equivalent of Linux “cat” — it merges all files in a given directory, and produces a single file in another given directory (it can even be the same directory).

```
create table csv_dump ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|' LINES TERMINATED BY '\n'
LOCATION '/home/acadgild/Desktop/Prachi/ExternalTables/' as
select * from temperature_data_vw;
```

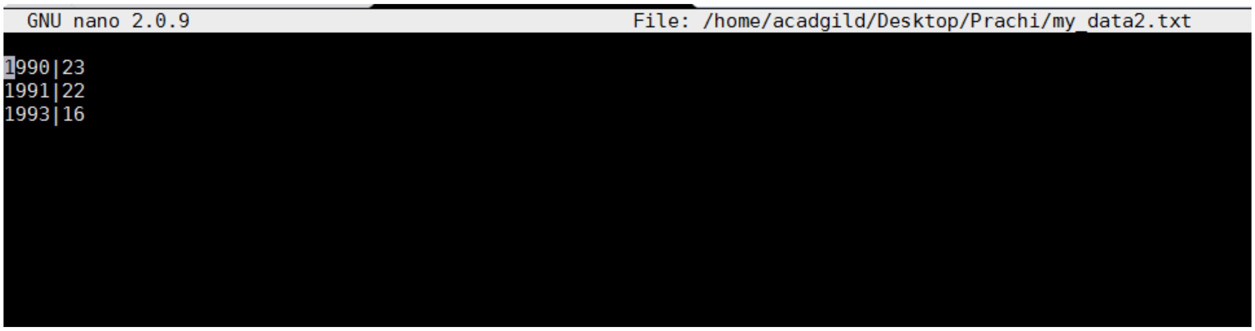


```
hive> create table csv_dump ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '|' LINES TERMINATED BY '\n'
    > LOCATION '/home/acadgild/Desktop/Prachi/ExternalTables/' as
    > select * from temperature_data_vw;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180508163532_399455dd-604d-4bc3-b855-c88c982978a4
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_1525770861854_0013, Tracking URL = http://localhost:8088/proxy/application_1525770861854_0013/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525770861854_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-05-08 16:35:51,042 Stage-1 map = 0%,  reduce = 0%
2018-05-08 16:36:08,223 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.76 sec
2018-05-08 16:36:26,600 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.05 sec
MapReduce Total cumulative CPU time: 10 seconds 540 msec
Ended Job = job_1525770861854_0013
Moving data to directory /home/acadgild/Desktop/Prachi/ExternalTables
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.54 sec   HDFS Read: 10916 HDFS Write: 95 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 540 msec
OK
Time taken: 59.365 seconds
hive> 
```

```
hadoop fs -getmerge /home/acadgild/Desktop/Prachi/ExternalTables/
/home/acadgild/Desktop/Prachi/my_data2.txt
```



```
[acadgild@localhost ~]$ hadoop fs -getmerge /home/acadgild/Desktop/Prachi/ExternalTables/ /home/acadgild/Desktop/Prachi/my_data2.txt
18/05/08 16:36:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ ls -l /home/acadgild/Desktop/Prachi
total 415560
-rw-rw-r--. 1 acadgild acadgild    244438 May  3 18:58 airports.csv
-rw-rw-r--. 1 acadgild acadgild       238 May  7 15:27 custs.txt
-rw-rw-r--. 1 acadgild acadgild 247963212 May  3 18:59 DelayedFlights.csv
-rw-rw-r--. 1 acadgild acadgild       273 May  3 16:03 employee_details.txt
-rw-rw-r--. 1 acadgild acadgild        79 May  3 16:03 employee_expenses.txt
drwxrwxr-x. 2 acadgild acadgild      4096 May  8 16:18 ExternalTables
drwxrwxr-x. 2 acadgild acadgild      4096 May  8 16:04 HIVE_DATA
-rw-rw-r--. 1 acadgild acadgild       549 May  7 15:34 HWIProps.txt
-rw-r--r--. 1 acadgild acadgild        24 May  8 16:36 my_data2.txt
-rw-rw-r--. 1 acadgild acadgild        24 May  8 16:06 my_table.txt
-rw-rw-r--. 1 acadgild acadgild 177279333 May  3 15:12 pig-0.16.0.tar.gz
[acadgild@localhost ~]$ nano /home/acadgild/Desktop/Prachi/my_data2.txt
```



```
1990|23
1991|22
1993|16
```