Big Data- Hadoop- Assignment 3

Exploring Pig:

Task 1:

Write a program to implement wordcount using Pig.

Step 1: Creating a file with some data:

```
[acadgild@localhost pig_test]$ vi word_count.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost pig_test]$ cat word_count.txt
Apache Pig is a highevelProcedural langauage
Pig runs on Hadoop
It makes use of HDFS and MapReduce
```

```
grunt> word_lines = LOAD 'word_count.txt' AS (lines:chararray);
2018-07-16 21:27:53,484 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-16 21:27:53,484 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
```

```
grunt> words = FOREACH word_lines GENERATE FLATTEN(TOKENIZE(lines)) as word;
grunt> grouped_words = GROUP words by word;
```

```
grunt> wordcount = FOREACH grouped_words GENERATE group, COUNT(words);
grunt> DUMP wordcount;
2018-07-16 21:33:10,984 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: GROUP_BY
2018-07-16 21:33:11,036 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-16 21:33:11,036 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-07-16 21:33:11,036 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Ke
y [pig.schematuple] was not set... will not generate code.
```

OUTPUT:

```
(a,1)
(It,1)
(is,1)
(of,1)
(on,1)
(Pig,2)
(and,1)
(use,1)
(HDFS,1)
(runs,1)
(makes,1)
(Apache,1)
(Hadoop,1)
(MapReduce,1)
(langauage,1)
(highevelProcedural,1)
```

Big Data- Hadoop- Assignment 3

Task 2:

We have employee_details and employee_expenses files. Use local mode while running Pig and write Pig Latin script to get below results:

Step 1: Running Pig in Local mode

```
[acadgild@localhost pig_test]$ pig -x local
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/sha
re/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/s
lf4j-log4j12-1.7.5.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.slf4j.impl.Log4jLoggerFactory]
18/07/17 09:12:57 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
18/07/17 09:12:57 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2018-07-17 09:12:57,855 [main] INFO  org.apache.pig.Main - Apache Pig version 0.
16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2018-07-17 09:12:57,855 [main] INFO  org.apache.pig.Main - Logging error message
s to: /home/acadgild/pig_test/pig_1531798977852.log
2018-07-17 09:12:58,034 [main] INFO  org.apache.pig.impl.util.Utils - Default bo
otup file /home/acadgild/.pigbootup not found
2018-07-17 09:12:58,664 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
```

Grunt Shell opens:

```
grunt>
```

employee_details (EmpID,Name,Salary,Rating)

```
[acadgild@localhost pig_test]$ vi employee_details.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost pig_test]$ cat employee_details.txt
101,Amitabh,20000,1
102,Shahrukh,10000,2
103,Akshay,11000,3
104,Anubhav,5000,4
105,Pawan,2500,5
106,Aamir,25000,1
107,Salman,17500,2
108,Ranbir,14000,3
109,Katrina,1000,4
110,Priyanka,2000,5
111,Tushar,500,1
112,Ajay,5000,2
113,Jubeen,1000,1
114,Madhuri,2000,2
```

Big Data- Hadoop- Assignment 3

employee_expenses(EmpID,Expense)

```
[acadgild@localhost pig_test]$ vi employee_expenses.txt
[acadgild@localhost pig_test]$ cat employee_expenses.txt
101     200
102     100
110     400
114     200
119     200
105     100
101     100
104     300
```

Step 3: Loading the “employee_details file”

```
grunt> emp1 = LOAD 'employee_details.txt' USING PigStorage(',') AS (emp_id:int,
emp_name:chararray, emp_salary:int,emp_rating:int);
2018-07-17 09:25:59,703 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 09:25:59,703 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> dump emp1;
2018-07-17 09:26:05,495 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: UNKNOWN
2018-07-17 09:26:05,562 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 09:26:05,566 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-07-17 09:26:05,566 [main] WARN  org.apache.pig.data.SchemaTupleBackend - Sc
hemaTupleBackend has already been initialized
2018-07-17 09:26:05,567 [main] INFO  org.apache.pig.newplan.logical.optimizer.Lo
```

```
(101,Amitabh,20000,1)
(102,Shahrukh,10000,2)
(103,Akshay,11000,3)
(104,Anubhav,5000,4)
(105,Pawan,2500,5)
(106,Aamir,25000,1)
(107,Salman,17500,2)
(108,Ranbir,14000,3)
(109,Katrina,1000,4)
(110,Priyanka,2000,5)
(111,Tushar,500,1)
(112,Ajay,5000,2)
(113,Jubeen,1000,1)
(114,Madhuri,2000,2)
grunt> describe emp1;
emp1: {emp_id: int,emp_name: chararray,emp_salary: int,emp_rating: int}
```

Big Data- Hadoop- Assignment 3

Step 4: Loading “employee_expenses.txt” file

```
grunt> emp_expl = LOAD 'employee_expenses.txt' AS (emp_id:int, expenses:int);
2018-07-17 09:29:42,742 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 09:29:42,742 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> dump emp_expl;
2018-07-17 09:29:56,572 [main] INFO  org.apache.pig.tools.pigstats.ScriptState -
 Pig features used in the script: UNKNOWN
2018-07-17 09:29:56,630 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 09:29:56,630 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-07-17 09:29:56,630 [main] WARN  org.apache.pig.data.SchemaTupleBackend - Sc
hemaTupleBackend has already been initialized
2018-07-17 09:29:56,634 [main] INFO  org.apache.pig.newplan.logical.optimizer.Lo
gicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalc
ulator, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeF
```

```
(101,200)
(102,100)
(110,400)
(114,200)
(119,200)
(105,100)
(101,100)
(104,300)
grunt> describe emp_expl;
emp_expl: {emp_id: int,expenses: int}
```

a) **Top 5 employees (employee id and employee name) with highest rating. (In case two employees have same rating, employee with name coming first in dictionary should get preference)**

```
grunt> empl_with_high_rating = ORDER empl by emp_rating DESC, emp_name ASC;
grunt> empl_limit_five = LIMIT empl_with_high_rating 5;
grunt> dump empl_limit_five;
```

Output:

```
(105,Pawan,2500,5)
(110,Priyanka,2000,5)
(104,Anubhav,5000,4)
(109,Katrina,1000,4)
(103,Akshay,11000,3)
```

Big Data- Hadoop- Assignment 3

(b) Top 3 employees (employee id and employee name) with highest salary, whose employee id is an odd number. (In case two employees have same salary, employee with name coming first in dictionary should get preference

```
grunt> empl_salary_order = ORDER empl by emp_salary DESC;
grunt> emp_empl_id = FILTER empl by emp_id % 2 ==1;
grunt> emp_high_salary = FOREACH emp_empl_id generate emp_id,emp_name;
grunt> emp_limit_three = LIMIT emp_high_salary 3;
grunt> dump emp_limit_three;
```

Output:

```
(101,Amitabh)
(103,Akshay)
(105,Pawan)
```

(c) Employee (employee id and employee name) with maximum expense (In case two employees have same expense, employee with name coming first in dictionary should get preference)

```
grunt> empl = LOAD 'employee_details.txt' USING PigStorage(',') AS (emp_id:int,
emp_name:chararray, emp_salary:int);
2018-07-17 11:20:54,028 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 11:20:54,028 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> emp_expenses = LOAD 'employee_expenses.txt' USING PigStorage(',') AS (emp
_id:int, emp_expense:int);
2018-07-17 11:21:08,291 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-07-17 11:21:08,291 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> describe empl;
empl: {emp_id: int,emp_name: chararray,emp_salary: int}
grunt> describe emp_expenses;
emp_expenses: {emp_id: int,emp_expense: int}
```

```
grunt> join_emp_empexpense = join empl by emp_id,emp_expenses by emp_id;
grunt> max_expense = ORDER join_emp_empexpense by emp_expenses::emp_expense desc
;
grunt> Limit_maxepnse = LIMIT max_expense 1;
grunt> max_expense_final = foreach Limit_maxepnse generate empl::emp_id,empl::em
p_name;
grunt> dump max_expense_final;
```

OUTPUT:

```
(110,Priyanka)
```

Big Data- Hadoop- Assignment 3

(d) List of employees (employee id and employee name) having entries in employee_expenses file.

```
grunt> emp_with_exp = JOIN emp1 BY emp_id, emp_expenses BY emp_id;
grunt> emp_with_exp_limit = FOREACH emp_with_exp GENERATE emp1::emp_id, emp1::em
p_name;
grunt> emp_with_exp_distinct_data = DISTINCT emp_with_exp_limit;
grunt> dump emp_with_exp_distinct_data
```

OUTPUT:

```
(101,Amitabh)
(102,Shahrukh)
(104,Anubhav)
(105,Pawan)
(110,Priyanka)
(114,Madhuri)
```

(e) List of employees (employee id and employee name) having no entry in employee_expenses file.

```
grunt> emp_without_exp = JOIN emp1 BY emp_id LEFT OUTER, emp_expenses BY emp_id;
grunt> emp_without_exp_filter = FILTER emp_without_exp BY emp_expenses::emp_id i
s null;
grunt> emp_without_exp_filter_data = FOREACH emp_without_exp_filter GENERATE emp
1::emp_id, emp1::emp_name;
grunt> dump emp_without_exp_filter_data;
```

Output:

```
(103,Akshay)
(106,Aamir)
(107,Salman)
(108,Ranbir)
(109,Katrina)
(111,Tushar)
(112,Ajay)
(113,Jubeen)
```

Big Data- Hadoop- Assignment 3

Task 3:

Implement the use case present in below blog link and share the complete steps along with

screenshot(s) from your end.

https://acadgild.com/blog/aviation-data-analysis-using-apache-pig/

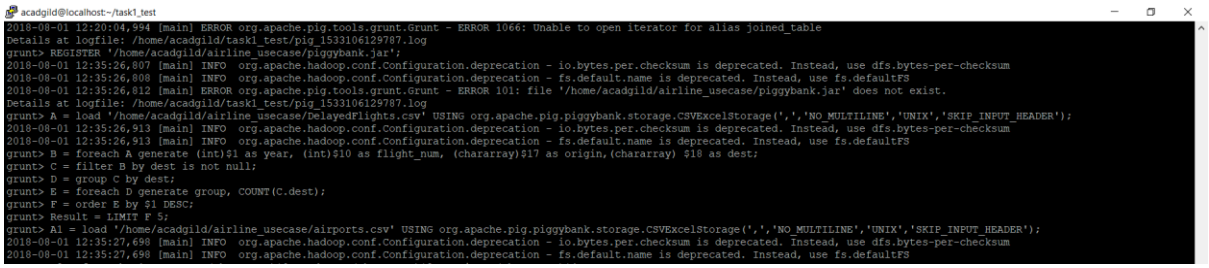
Problem Statement 1

Find out the top 5 most visited destinations.

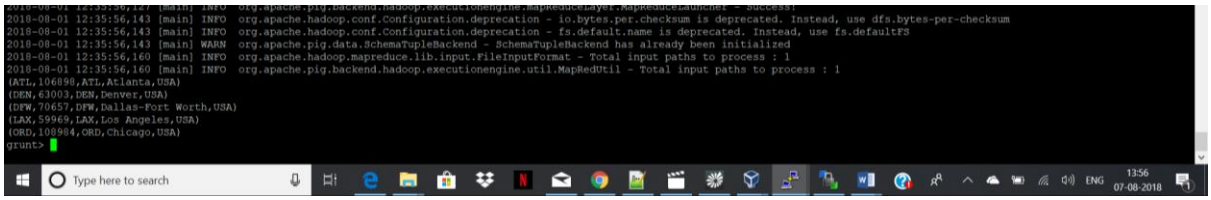
Code

```
REGISTER '/home/acadgild/airline_usecase/piggybank.jar';

A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX',
'SKIP_INPUT_HEADER');

B = foreach A generate (int)$1 as year, (int)$10 as flight_num,
(chararray)$17 as origin,(chararray) $18 as dest;

C = filter B by dest is not null;

D = group C by dest;

E = foreach D generate group, COUNT(C.dest);

F = order E by $1 DESC;

Result = LIMIT F 5;

A1 = load '/home/acadgild/airline_usecase/airports.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX',
'SKIP_INPUT_HEADER');

A2 = foreach A1 generate (chararray)$0 as dest, (chararray)$2 as city,
(chararray)$4 as country;

joined_table = join Result by $0, A2 by dest;

dump joined_table;
```

Big Data- Hadoop- Assignment 3



OUTPUT



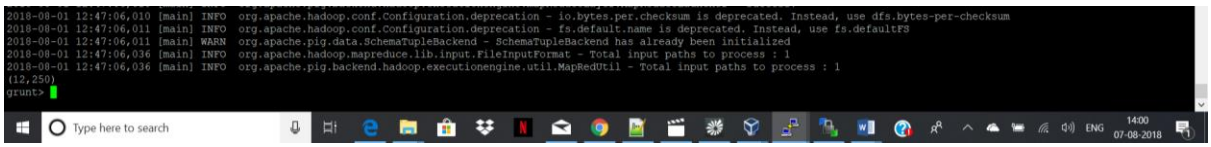
Problem Statement 2

Which month has seen the most number of cancellations due to bad weather?

```
REGISTER '/home/acadgild/airline_usecase/piggybank.jar';

A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX',
'SKIP_INPUT_HEADER');

B = foreach A generate (int)$2 as month,(int)$10 as flight_num,(int)$22 as
cancelled,(chararray)$23 as cancel_code;

C = filter B by cancelled == 1 AND cancel_code =='B';

D = group C by month;

E = foreach D generate group, COUNT(C.cancelled);

F= order E by $1 DESC;

Result = limit F 1;
```

```
dump Result;
```

OUTPUT



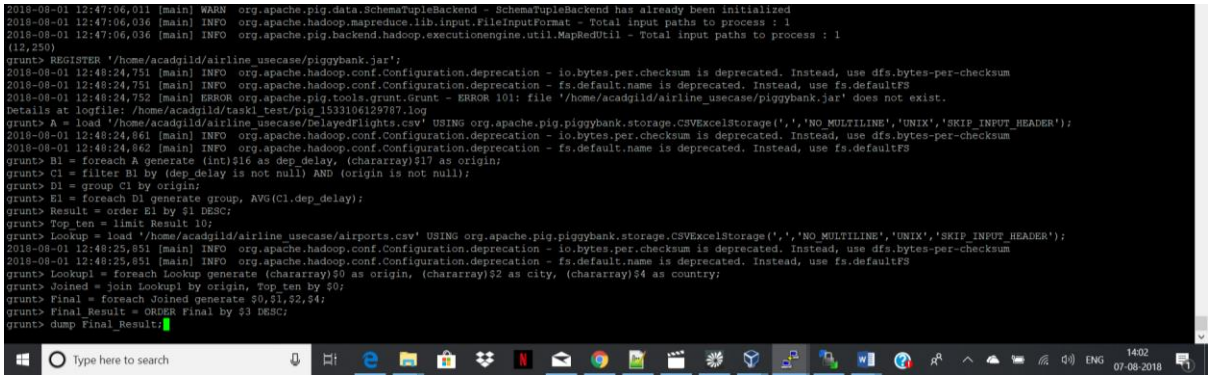
Problem Statement 3

Top ten origins with the highest AVG departure delay

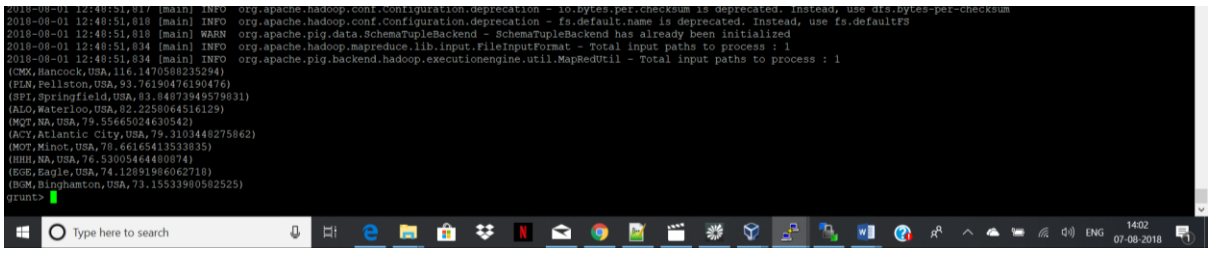
Code

```
REGISTER '/home/acadgild/airline_usecase/piggybank.jar';

A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX',
'SKIP_INPUT_HEADER');

B1 = foreach A generate (int)$16 as dep_delay, (chararray)$17 as origin;

C1 = filter B1 by (dep_delay is not null) AND (origin is not null);

D1 = group C1 by origin;

E1 = foreach D1 generate group, AVG(C1.dep_delay);

Result = order E1 by $1 DESC;

Top_ten = limit Result 10;

Lookup = load '/home/acadgild/airline_usecase/airports.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX',
'SKIP_INPUT_HEADER');

Lookup1 = foreach Lookup generate (chararray)$0 as origin, (chararray)$2 as
city, (chararray)$4 as country;

Joined = join Lookup1 by origin, Top_ten by $0;

Final = foreach Joined generate $0,$1,$2,$4;

Final_Result = ORDER Final by $3 DESC;
```

```
dump Final_Result;
```



OUTPUT



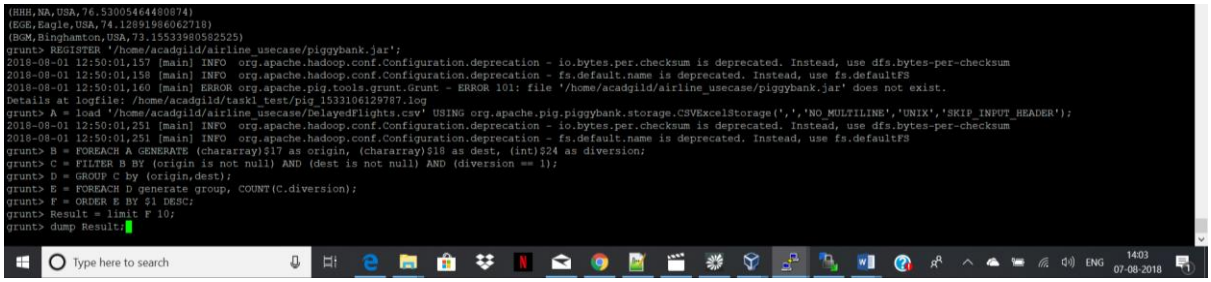
Problem Statement 4

Which route (origin & destination) has seen the maximum diversion?

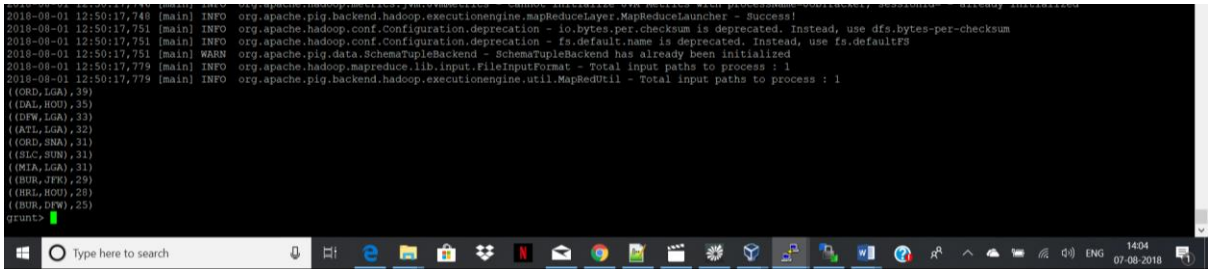
Code

```
REGISTER '/home/acadgild/airline_usecase/piggybank.jar';

A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX',
'SKIP_INPUT_HEADER');

B = FOREACH A GENERATE (chararray)$17 as origin, (chararray)$18 as dest,
(int)$24 as diversion;
```

```
C = FILTER B BY (origin is not null) AND (dest is not null) AND (diversion
== 1);

D = GROUP C by (origin,dest);

E = FOREACH D generate group, COUNT(C.diversion);

F = ORDER E BY $1 DESC;

Result = limit F 10;

dump Result;
```



OUTPUT



Hive Basics

Task 1:

Create a database named 'custom'.

Code used: `CREATE DATABASE custom;`

Big Data- Hadoop- Assignment 3

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.

Code used : create table temperature_data

(

date string,

zip_code int,

temperature int

)

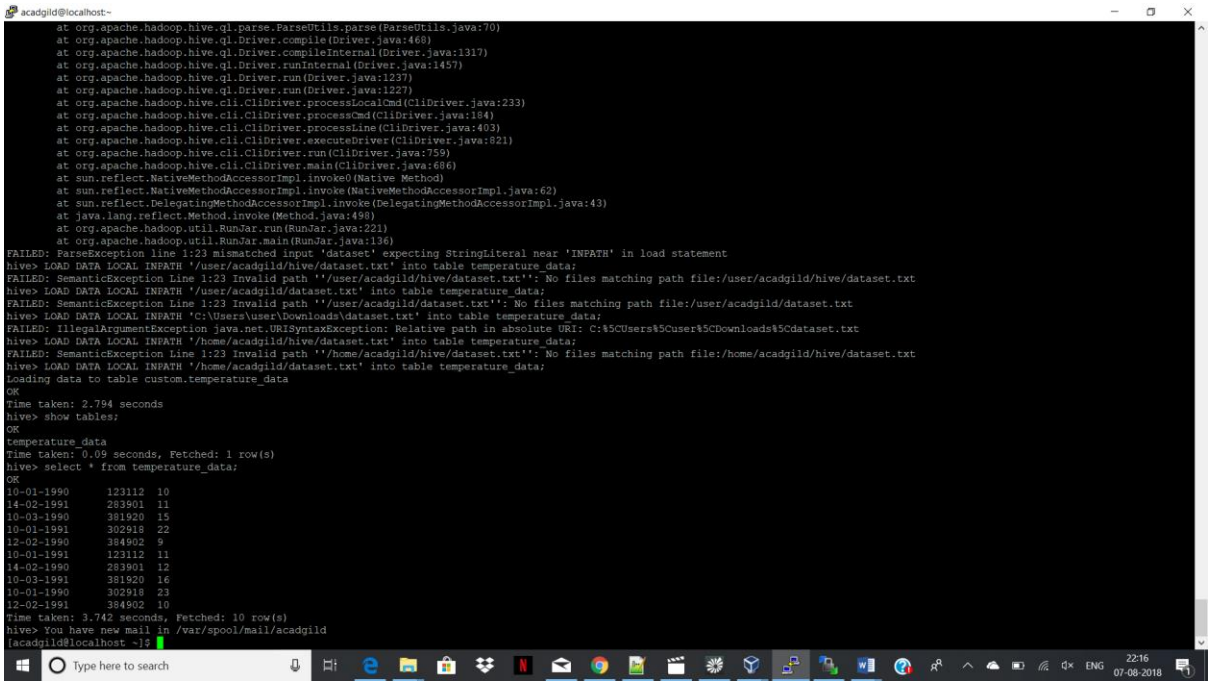
row format delimited

fields terminated by ',';

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

Code used: LOAD DATA LOCAL INPATH '/home/acadgild/dataset.txt' into table temperature_data;

<u>Solution for task1:</u>



```
acadgild@localhost:~
        at org.apache.hadoop.hive.ql.parse.ParseUtils.parse(ParseUtils.java:70)
        at org.apache.hadoop.hive.ql.Driver.compile(Driver.java:468)
        at org.apache.hadoop.hive.ql.Driver.compileInternal(Driver.java:1317)
        at org.apache.hadoop.hive.ql.Driver.runInternal(Driver.java:1457)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1237)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1227)
        at org.apache.hadoop.hive.cli.CliDriver.processLocalCmd(CliDriver.java:233)
        at org.apache.hadoop.hive.cli.CliDriver.processCmd(CliDriver.java:184)
        at org.apache.hadoop.hive.cli.CliDriver.processLine(CliDriver.java:403)
        at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java:821)
        at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:759)
        at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:686)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
FAILED: ParseException line 1:23 mismatched input 'dataset' expecting StringLiteral near 'INPATH' in load statement
hive> LOAD DATA LOCAL INPATH '/user/acadgild/hive/dataset.txt' into table temperature_data;
FAILED: SemanticException Line 1:23 Invalid path ''/user/acadgild/hive/dataset.txt''': No files matching path file:/user/acadgild/hive/dataset.txt
hive> LOAD DATA LOCAL INPATH '/user/acadgild/dataset.txt' into table temperature_data;
FAILED: SemanticException Line 1:23 Invalid path ''/user/acadgild/dataset.txt''': No files matching path file:/user/acadgild/dataset.txt
hive> LOAD DATA LOCAL INPATH 'C:\Users\user\Downloads\dataset.txt' into table temperature_data;
FAILED: IllegalArgumentException java.net.URISyntaxException: Relative path in absolute URI: C:%5CUsers%5Cuser%5CDownloads%5Cdataset.txt
hive> LOAD DATA LOCAL INPATH '/home/acadgild/hive/dataset.txt' into table temperature_data;
FAILED: SemanticException Line 1:23 Invalid path ''/home/acadgild/hive/dataset.txt''': No files matching path file:/home/acadgild/hive/dataset.txt
hive> LOAD DATA LOCAL INPATH '/home/acadgild/dataset.txt' into table temperature_data;
Loading data to table custom.temperature_data
OK
Time taken: 2.794 seconds
hive> show tables;
OK
temperature_data
Time taken: 0.09 seconds, Fetched: 1 row(s)
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
Time taken: 3.742 seconds, Fetched: 10 row(s)
hive> You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

Big Data- Hadoop- Assignment 3

Task 2:

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

Code used :

select day,temp from temperature_data where zip_code between 300000 and 399999;

```
FAILED: ParseException line 1:61 missing ) at '<=' near '<EOF>'
line 1:69 extraneous input ')' expecting EOF near '<EOF>'
hive> select day,temp from temperature_data where zip_code between 300000 and 399999;
OK
10-03-1990      15
10-01-1991      22
12-02-1990      9
10-03-1991      16
10-01-1990      23
12-02-1991      10
Time taken: 1.038 seconds, Fetched: 6 row(s)
hive> You have new mail in /var/spool/mail/acadgild
```

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

```
OK
10-01-1990      302918  23
10-01-1991      302918  22
Time taken: 33.127 seconds, Fetched: 2 row(s)
hive>
```

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

```
acadgild@localhost:~
MapReduce Total cumulative CPU time: 3 seconds 870 msec
Ended Job = job_1533069881276_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.87 sec   HDFS Read: 7802 HDFS Write: 416
 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 870 msec
OK
10-01-1990      302918  23
10-01-1991      302918  22
10-03-1991      381920  16
10-03-1990      381920  15
14-02-1990      283901  12
10-01-1991      123112  11
14-02-1991      283901  11
12-02-1991      384902  10
10-01-1990      123112  10
12-02-1990      384902  9
Time taken: 35.063 seconds, Fetched: 10 row(s)
```

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

```
hive> create view temperature_view as select * from temperature_data order by temp desc limit 2;
OK
Time taken: 1.195 seconds
hive> create view temperature_data_vm as select * from temperature_data order by temp desc limit 2;
OK
Time taken: 0.365 seconds
hive>
```

Big Data- Hadoop- Assignment 3

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

```
acadgild@localhost:~
OK
10-01-1990      302918  23
10-01-1991      302918  22
Time taken: 33.127 seconds, Fetched: 2 row(s)
hive> create view temperature_view as select * from temperature_data order by temp desc limit 2;
OK
Time taken: 1.195 seconds
hive> create view temperature_data_vm as select * from temperature_data order by temp desc limit 2;
OK
Time taken: 0.365 seconds
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/output'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '|'
    > SELECT * FROM emp_details;
FAILED: SemanticException [Error 10001]: Line 4:14 Table not found 'emp_details'
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/output'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '|'
    > SELECT * FROM emp_details;
FAILED: SemanticException [Error 10001]: Line 4:14 Table not found 'emp_details'
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/output'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '|'
    > SELECT * FROM temperature_view;
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_20180801170124_c7246508-84d6-4722-832c-8a65aae2768b
Total jobs = 1
Launching Job 1 out of 1
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_1533069881276_0004, Tracking URL = http://localhost:8088/proxy/application_1533069881276_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533069881276_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-01 17:01:36,606 Stage-1 map = 0%,  reduce = 0%
2018-08-01 17:01:45,632 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.78 sec
2018-08-01 17:01:56,894 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.06 sec
MapReduce Total cumulative CPU time: 4 seconds 60 msec
Ended Job = job_1533069881276_0004
Moving data to local directory /tmp/output
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.06 sec   HDFS Read: 7579 HDFS Write: 42 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 60 msec
OK
Time taken: 33.294 seconds
hive>
```