Advanced Data Analytics using Hive of Titanic: Machine Learning from Disaster dataset

Calculate how many events occurred according to the event types

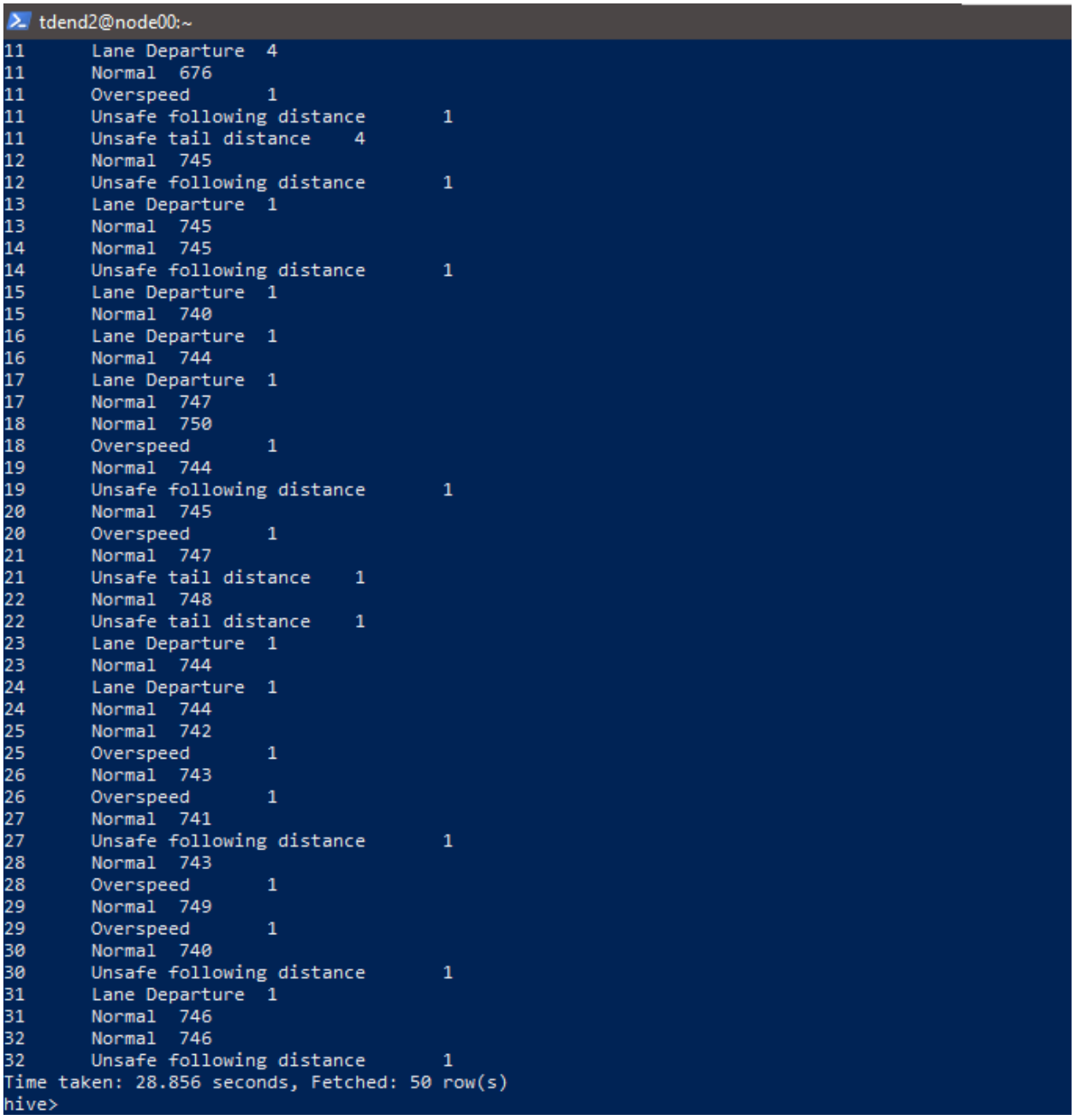
First, ran the below HiveQL query to calculate how many events occurred according to the event
types of each driver.

SELECT driverid, eventType, count(*) FROM csc534.tdend2_truck_event GROUP BY driverId, eventType;

```
tdend2@node00:~
hive> SELECT driverid, eventType, count(*)  FROM csc534.tdend2_truck_event GROUP BY driverId, eventType;
Query ID = tdend2_20241009170501_963f7c3b-1fc0-4290-b77f-148b6b584c20
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>
```

```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-09 17:05:14,853 Stage-1 map = 0%,  reduce = 0%
2024-10-09 17:05:22,064 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.74 sec
2024-10-09 17:05:29,244 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.14 sec
MapReduce Total cumulative CPU time: 8 seconds 140 msec
Ended Job = job_1722897143033_1257
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.14 sec   HDFS Read: 2283649 HDFS Write: 1610 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 140 msec
OK
10      Normal  727
10      Overspeed       2
10      Unsafe tail distance    1
11      Lane Departure  4
11      Normal  676
11      Overspeed       1
```

Last half of output:



To show header or column name in the outputs, ran below command:
set hive.cli.print.header=true;
set hive.resultset.use.unique.column.names=false;

Calculate the total events that occurred

Next, ran the below HiveQL query to calculate the total events that occurred per driver.

SELECT driverId, count(*) AS total_events FROM csc534.tdend2_truck_event GROUP BY driverid;

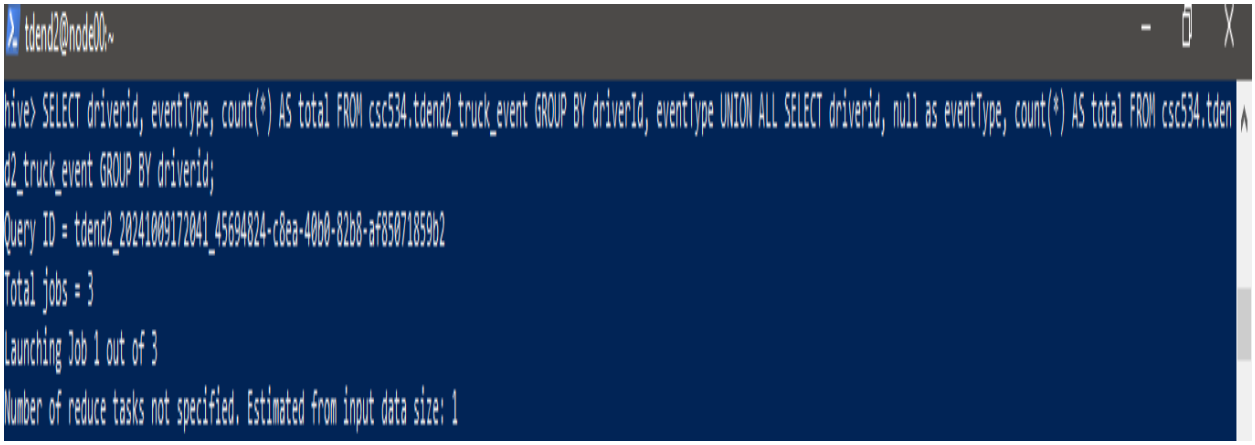
```
tdend2@node00:~
hive>  set hive.cli.print.header=true;
hive>  SELECT driverId, count(*) AS total_events FROM csc534.tdend2_truck_event GROUP BY driverid;
Query ID = tdend2_20241009171433_27af75c6-b3c8-4799-83c3-9454086d39c8
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. 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>
24/10/09 17:14:33 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
24/10/09 17:14:33 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
Starting Job = job_1722897143033_1258, Tracking URL = http://node00.sun:8088/proxy/application_1722
Kill Command = /opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/lib/hadoop/bin/hadoop job  -ki
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-09 17:14:43,198 Stage-1 map = 0%,  reduce = 0%
2024-10-09 17:14:51,396 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.66 sec
2024-10-09 17:14:59,583 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.07 sec
MapReduce Total cumulative CPU time: 7 seconds 70 msec
Ended Job = job_1722897143033_1258
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.07 sec   HDFS Read: 2283400 HDFS Write: 524 HD
Total MapReduce CPU Time Spent: 7 seconds 70 msec
OK
driverid        total_events
10      730
11      686
12      746
13      746
```

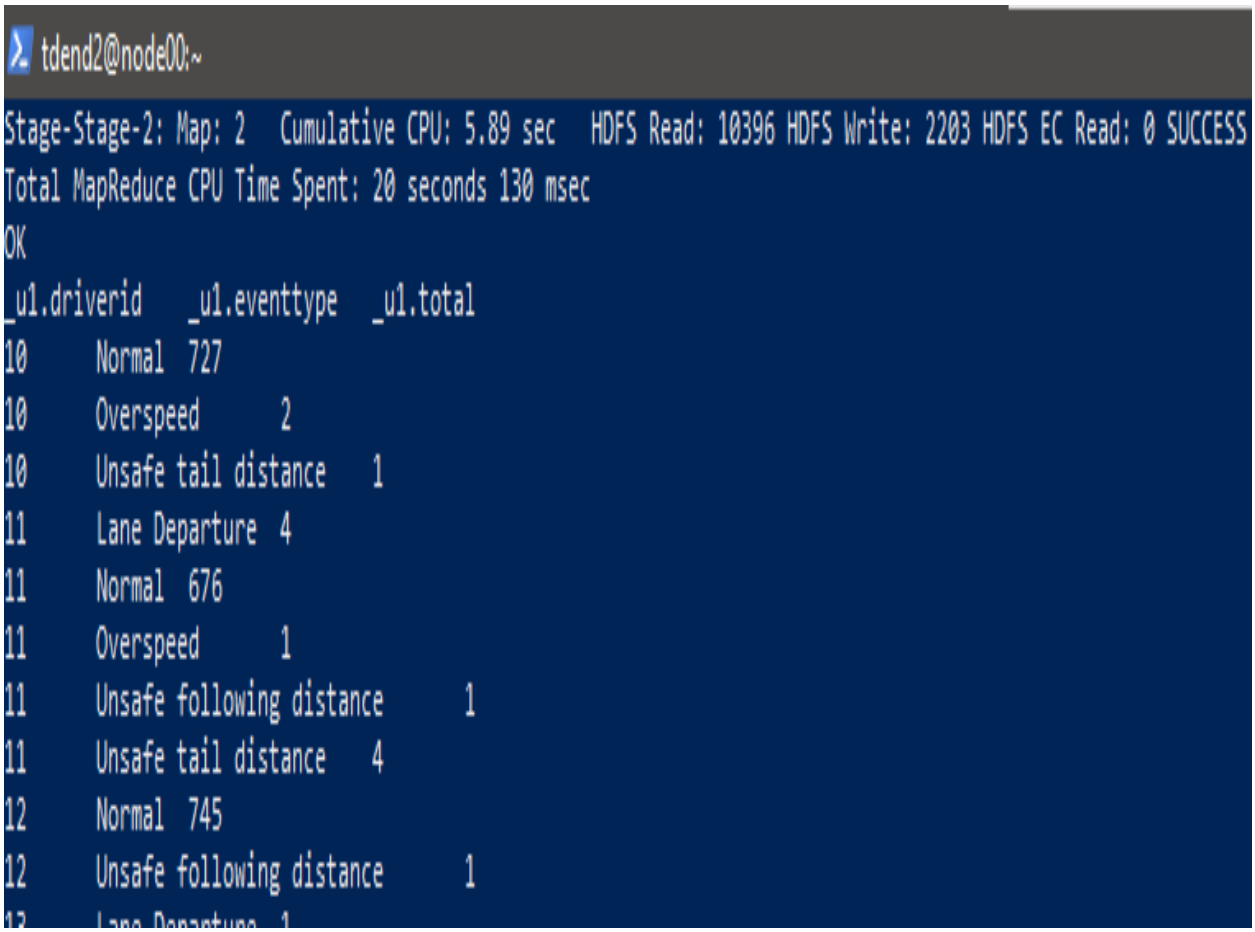
Total events based on driver id with header :

```
tdend2@node00:~
driverid        total_events
10      730
11      686
12      746
13      746
14      746
15      741
16      745
17      748
18      751
19      745
20      746
21      748
22      749
23      745
24      745
25      743
26      744
27      742
28      744
29      750
30      741
31      747
32      747
Time taken: 27.223 seconds, Fetched: 23 row(s)
hive>
```

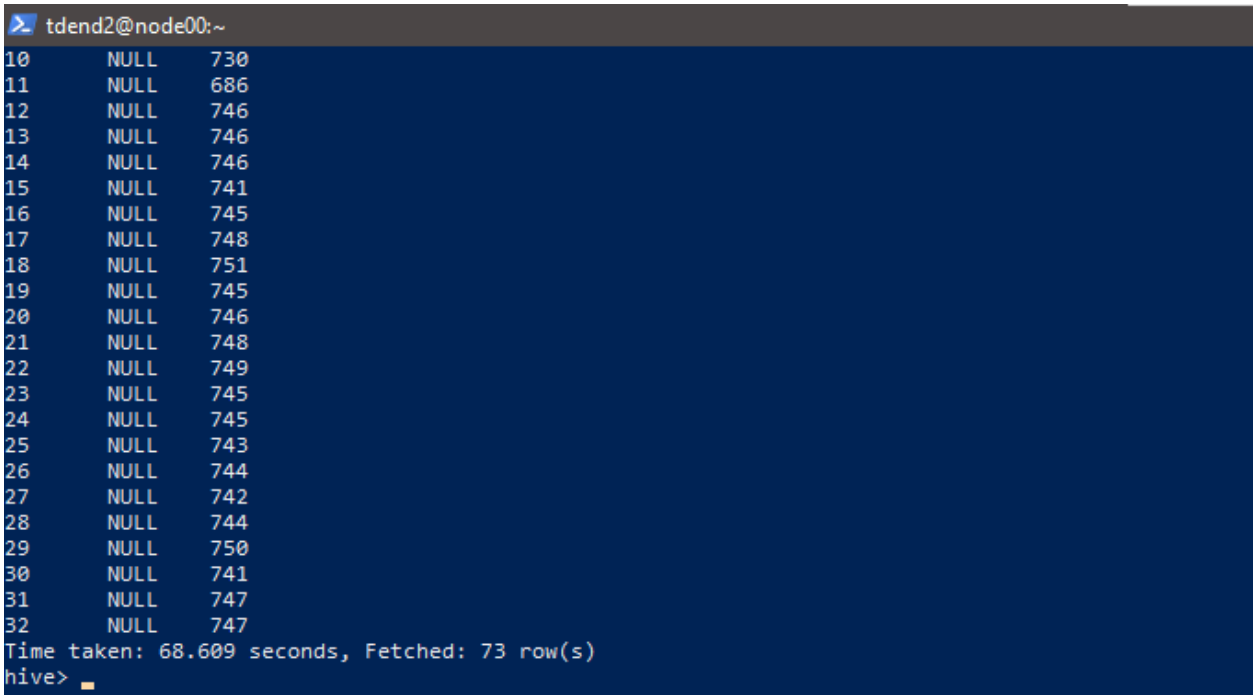
Connect both GROUP BY result sets
Finally, tried to connect both GROUP BY result sets with UNION ALL as below:
Also need to modify the queries to the schema of both sides of the union to be matched

SELECT driverid, eventType, count(*) AS total FROM csc534.tdend2_truck_event GROUP BY driverId, eventType UNION ALL SELECT driverid, null as eventType, count(*) AS total FROM csc534.tdend2_truck_event GROUP BY driverid;





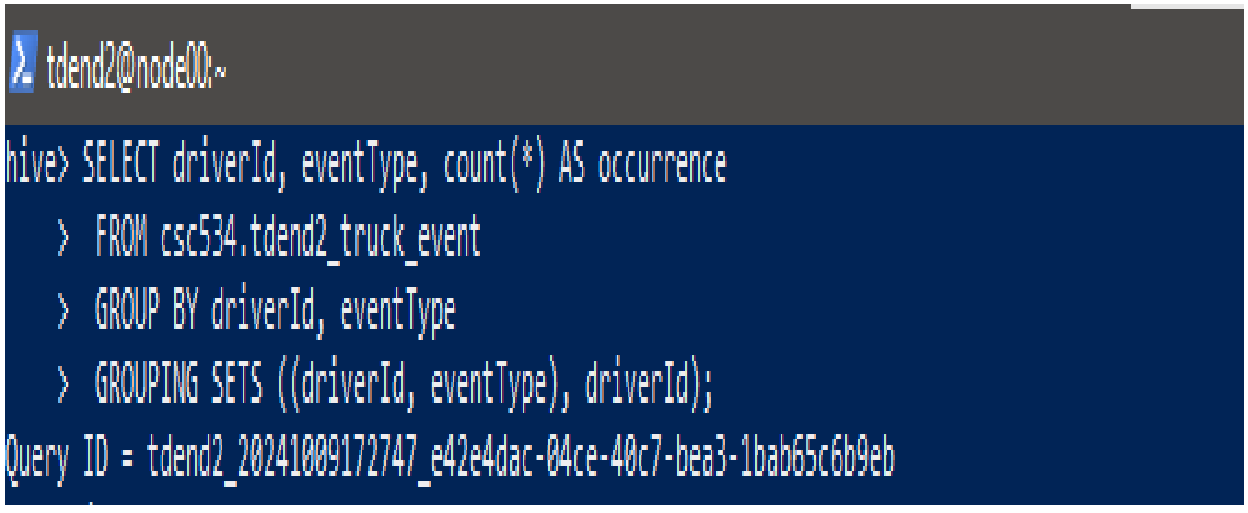
Last half of output:



It shows the driver's id, event type, and the total number of events from both query result sets. It took quite a long time compared to first query second
query, because it read the datasets twice. If we need to analyze Big data, it could be a problem.This is where GROUPING SETS come in.

• (10pts) Section 2.2

SELECT driverId, eventType, count(*) AS occurrence
FROM csc534.tdend2_truck_event
GROUP BY driverId, eventType
GROUPING SETS ((driverId, eventType), driverId);



```
tdend2@node00:~
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.15 sec   HDFS Read: 2284335 HDFS Write: 2116 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 150 msec
OK
driverid        eventtype       occurrence
10      NULL    730
10      Normal  727
10      Overspeed       2
10      Unsafe tail distance    1
11      NULL    686
11      Lane Departure  4
11      Normal  676
11      Overspeed       1
11      Unsafe following distance       1
11      Unsafe tail distance    4
12      NULL    746
12      Normal  745
12      Unsafe following distance       1
13      NULL    746
```

Last half of output:

```
29      Overspeed       1
30      NULL    741
30      Normal  740
30      Unsafe following distance       1
31      NULL    747
31      Lane Departure  1
31      Normal  746
32      NULL    747
32      Normal  746
32      Unsafe following distance       1
Time taken: 25.931 seconds, Fetched: 73 row(s)
hive>
```

We calculated how many events occurred according to each driver's event types and the total events that occurred per driver. Where the event column is null, we have the total sum of events that occurred to a driver across all event types.

Analyzing Driver Risk factor

Analyzed the risks associated with drivers in detail. We will use event and driver's mileage records to calculate the driver's risk factor.

Non-normal events and totals

First, filtered normal records from the events records (geolocation table), then created an unusual_event table. Ran the below query.

```
CREATE TABLE csc534.tdend2_unusual_events
STORED AS ORC
AS
SELECT driverId, count(*) AS occurrence
FROM csc534.tdend2_truck_event
WHERE eventType != 'Normal'
GROUP BY driverId;
```

```
tdend2@node00:~
hive> CREATE TABLE csc534.tdend2_unusual_events
    > STORED AS ORC
    > AS
    > SELECT driverId, count(*) AS occurrence
    > FROM csc534.tdend2_truck_event
    > WHERE eventType != 'Normal'
    > GROUP BY driverId;
Query ID = tdend2_20241009173432_9ba86569-b44b-4c64-94c0-84dd27afb830
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>
24/10/09 17:34:32 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
24/10/09 17:34:32 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
Starting Job = job_1722897143033_1268, Tracking URL = http://node00.sun:8088/proxy/application_17
Kill Command = /opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/lib/hadoop/bin/hadoop job
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-09 17:34:40,574 Stage-1 map = 0%,  reduce = 0%
2024-10-09 17:34:46,712 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.13 sec
2024-10-09 17:34:55,901 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.45 sec
MapReduce Total cumulative CPU time: 10 seconds 450 msec
Ended Job = job_1722897143033_1268
Moving data to directory hdfs://node00.sun:8020/user/hive/warehouse/csc534.db/tdend2_unusual_ever
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.45 sec   HDFS Read: 2284022 HDFS Write: 365
Total MapReduce CPU Time Spent: 10 seconds 450 msec
OK
driverid        occurrence
Time taken: 25.813 seconds
hive>
```

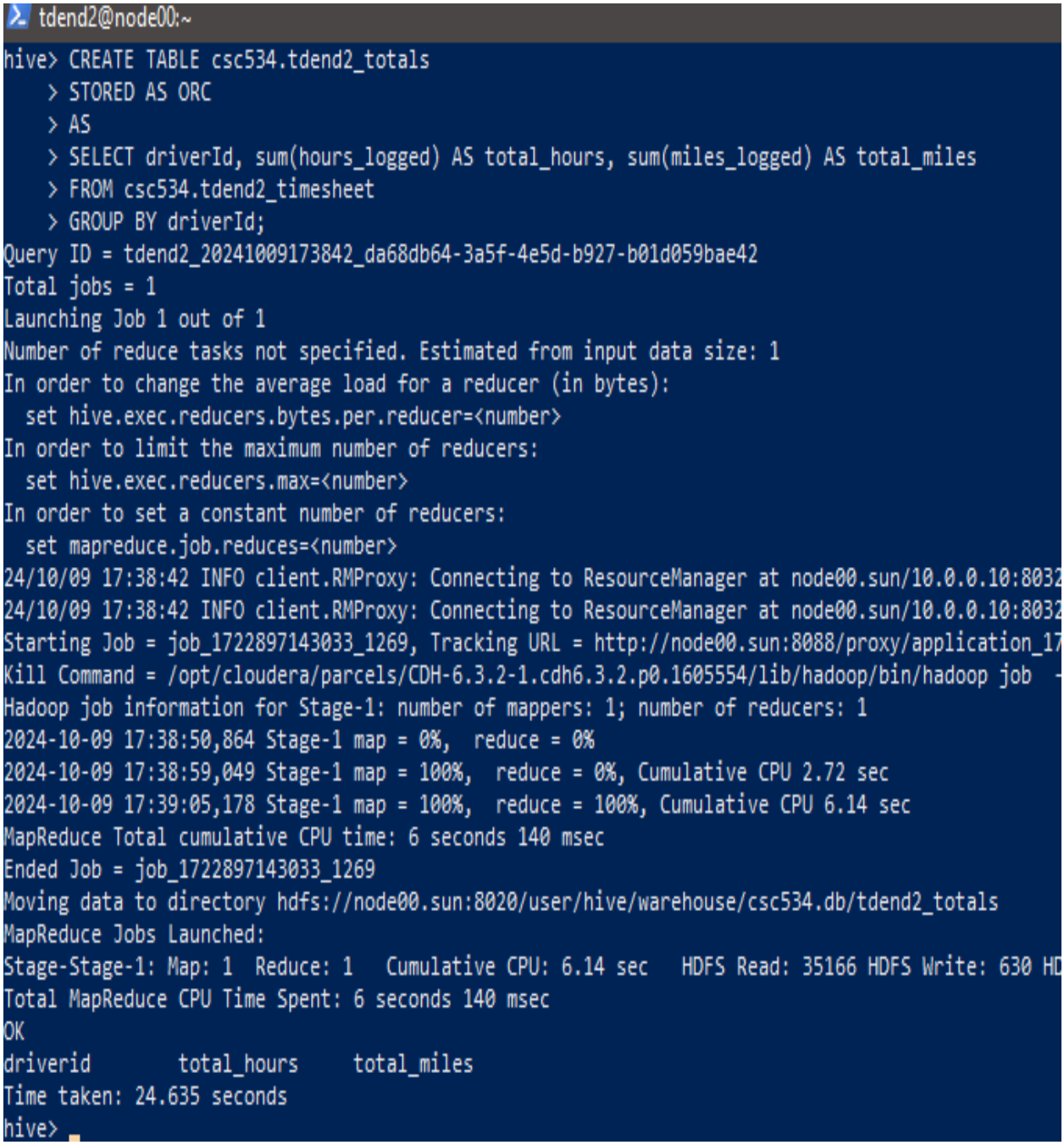
The resulting table will count each driver's total unusual or abnormal events

```
SELECT *
FROM csc534.tdend2_unusual_events
LIMIT 5;
```

```
tdend2@node00:~
hive> SELECT *
    >  FROM csc534.tdend2_unusual_events
    >  LIMIT 5;
OK
tdend2_unusual_events.driverid  tdend2_unusual_events.occurrence
10      3
11      10
12      1
13      1
14      1
Time taken: 0.077 seconds, Fetched: 5 row(s)
hive>
```

Also, created another table, totals, with total hours and miles.

```
CREATE TABLE csc534.tdend2_totals
STORED AS ORC
AS
SELECT driverId, sum(hours_logged) AS total_hours, sum(miles_logged) AS total_miles
FROM csc534.tdend2_timesheet
GROUP BY driverId;
```



We can check the resulting table as shown below:

```
 SELECT *
FROM csc534.tdend2_totals
LIMIT 5;
```

```
tdend2@node00:~
hive> SELECT *
    > FROM csc534.tdend2_totals
    > LIMIT 5;
OK
tdend2_totals.driverid  tdend2_totals.total_hours       tdend2_totals.total_miles
10      3232    147150
11      3642    179300
12      2639    135962
13      2727    134126
14      2781    136624
Time taken: 0.066 seconds, Fetched: 5 row(s)
hive>
```

2.3.2 Perform JOIN operation

Performed a JOIN operation. The unusual_events table has the driver's ID and count of their respective non-normal events. The totals table shows the hours and miles each driver travelled.

```
CREATE TABLE csc534.tdend2_joined
STORED AS ORC
AS
SELECT u.driverId, u.occurrence, t.total_hours, t.total_miles
FROM csc534.tdend2_unusual_events u
JOIN csc534.tdend2_totals t
ON (u.driverId = t.driverId);
```

```
tdend2@node00:~
hive> CREATE TABLE csc534.tdend2_joined
    > STORED AS ORC
    > AS
    > SELECT u.driverId, u.occurrence, t.total_hours, t.total_miles
    > FROM csc534.tdend2_unusual_events u
    > JOIN csc534.tdend2_totals t
    > ON (u.driverId = t.driverId);
Query ID = tdend2_20241009174402_5dba689a-bd52-4f00-875b-5c10b15de9dd
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/jars/log4
SLF4J: Found binding in [jar:file:/opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/jars/slf4
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
```

```
tdend2@node00:~
Total MapReduce CPU Time Spent: 6 seconds 180 msec
OK
u.driverid      u.occurrence    t.total_hours   t.total_miles
Time taken: 30.668 seconds
hive>
```

The resulting data set will give us a driver's total miles and non-normal events.

```
SELECT *
FROM csc534.tdend2_joined
LIMIT 5;
```

```
tdend2@node00:~
hive> SELECT *
    > FROM csc534.tdend2_joined
    > LIMIT 5;
OK
tdend2_joined.driverid  tdend2_joined.occurrence        tdend2_joined.total_hours       tdend2_joined.total_miles
10      3       3232    147150
11      10      3642    179300
12      1       2639    135962
13      1       2727    134126
14      1       2781    136624
Time taken: 0.068 seconds, Fetched: 5 row(s)
hive>
```

Compute driver risk factor

We will associate a driver risk factor with every driver. The risk factor for each driver is the number of unusual/abnormal occurrences over the total number of miles. A high number of unusual/abnormal occurrences over a short number of miles indicates high risk.
Let's translate this intuition into an HiveQL query.

```
SELECT driverid, occurrence, total_hours, total_miles, total_miles/occurrence
AS riskfactor
FROM csc534.tdend2_joined
ORDER BY riskfactor ASC
LIMIT 5;
```

```
tdend2@node00:~
hive> SELECT driverid, occurrence, total_hours, total_miles, total_miles/occurrence
    > AS riskfactor
    > FROM csc534.tdend2_joined
    > ORDER BY riskfactor ASC
    > LIMIT 5;
Query ID = tdend2_20241009174824_f69294c6-8be2-49dd-8c03-04dd90577a95
```

```
tdend2@node00:~
Total MapReduce CPU Time Spent: 8 seconds 930 msec
OK
driverid        occurrence      total_hours     total_miles     riskfactor
11      10      3642    179300  17930.0
10      3       3232    147150  49050.0
13      1       2727    134126  134126.0
24      1       2647    134461  134461.0
20      1       2644    134564  134564.0
Time taken: 27.129 seconds, Fetched: 5 row(s)
hive>
```

The resulting data set will give us occurrence, total_hours, total miles, total unusual/abnormal events, and what is a risk for a particular driver.

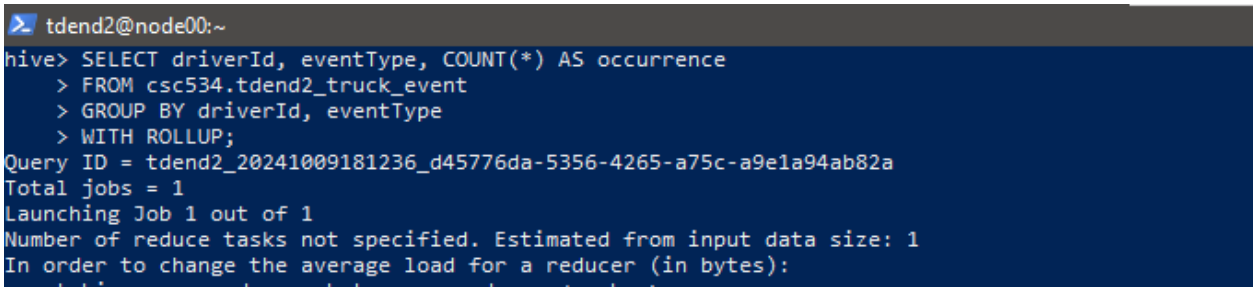
We exercised GROUPING SETS. There are additional ways to perform these aggregations, using CUBE or ROLLUP. These are almost like shortcuts. While CUBE returns all possible aggregation combinations, ROLLUP does it more hierarchically.

Rollup

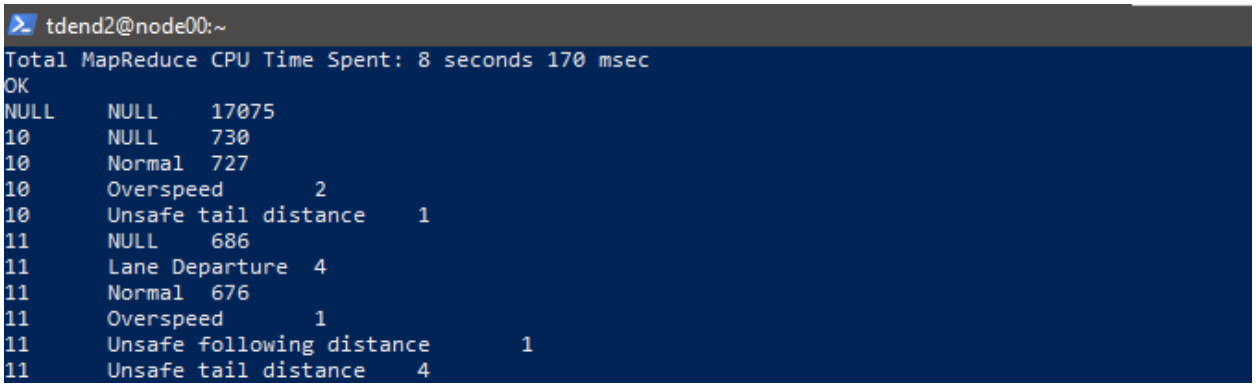
Modify/rewrite the grouping set query using ROLLUP. Run and show the results.Explain the presentation of the result data by comparing the grouping set query's presentation

Reference:
https://cwiki.apache.org/confluence/display/Hive/Enhanced+Aggregation%2C+Cube%2C+Grouping+and+Rollup

```
SELECT driverId, eventType, COUNT(*) AS occurrence
FROM csc534.tdend2_truck_event
GROUP BY driverId, eventType
WITH ROLLUP;
```



• The first 10 lines of query results only:



From the output, it can be observed that each row represents the 'driverid' and its corresponding 'event type' along with the sum of event_count for that combination. The difference in the results of grouping set and ROLL UP is that the first row in the result of ROLL UP is not obtained with GROUPING SET. The difference in the number of resultant rows of GROUPING SETS and ROLL UP is less.

Hence, the difference between GROUPING SETS and ROLLUP is clearly visible from the first half of the result especially the first 10 rows and found in the number of resultant rows and the order of it is GROUPING SETS<ROLL UP.

Cube

Write the grouping set query using CUBE. Run and show the results.

Explain the presentation of the result data by comparing both the grouping set query and the ROLLUP query's presentation.

• Replaced GROUPING SETS (...) with 'WITH CUBE

Reference:
https://cwiki.apache.org/confluence/display/Hive/Enhanced+Aggregation%2C+Cube%2C+Grouping+and+Rollup

```
SELECT driverId, eventType, count(*) AS occurrence
FROM csc534.tdend2_truck_event
GROUP BY driverId, eventType
WITH CUBE;
```

```
tdend2@node00:~
hive> SELECT driverId, eventType, count(*) AS occurrence
    > FROM csc534.tdend2_truck_event
    > GROUP BY driverId, eventType
    > WITH CUBE;
Query ID = tdend2_20241009181704_530db350-d732-46de-ab16-cabcc08f2490
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
```

• The first 10 lines of query results only.

```
tdend2@node00:~
Total MapReduce CPU Time Spent: 7 seconds 790 msec
OK
NULL    NULL    17075
NULL    Lane Departure  11
NULL    Normal  17041
NULL    Overspeed       9
NULL    Unsafe following distance       7
NULL    Unsafe tail distance    7
10      NULL    730
10      Normal  727
10      Overspeed       2
10      Unsafe tail distance    1
```

GROUPING SETS: Manually defines the exact sets of groupings you want.
ROLLUP: A hierarchical aggregation, showing subtotals and grand totals.
CUBE: Includes all possible combinations of the groupings, leading to more rows than ROLLUP or GROUPING SETS.

From the above cube output,each row represents 'driverid' and corresponding 'event type' along with sum of event-count for that combination. NULL values in the output represents totals or subtotals for combination where the corresponding column value is not included in the grouping set.The first row in the result of ROLL UP is not obtained with GROUPING SET but it is the common row for CUBE and ROLL UP in addition to other rows which are available in all these

results.In addition to this first row, other 5 rows with driver id 'NULL' are extra rows obtained in the result of CUBE.

The difference in the number of *resultant rows of GROUPING SETS and ROLL UP is not much but CUBE operation resulted in more number of rows than these operations as it* includes all possible combinations of the groupings, leading to more rows than ROLLUP or GROUPING SETS.

Hence, the difference among GROUPING SETS, ROLLUP and CUBE is clearly visible from the first half of the result especially the first 10 rows and found in the number of resultant rows and the order of it is GROUPING SETS<ROLL UP< CUBE.

Here we use a machine learning competition dataset from Kaggle, "**Titanic: Machine Learning from Disaster**". See the detail https://www.kaggle.com/c/titanic.

We will use simple descriptive statistics to understand and analyze the data using HiveQL.
Found the dataset in the cluster:

```
tdend2@node00:~
[tdend2@node00 ~]$  head /home/data/CSC534BDA/datasets/Titanic/titanic.csv
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
[tdend2@node00 ~]$
```

Step1
Load the data to your user space in HDFS

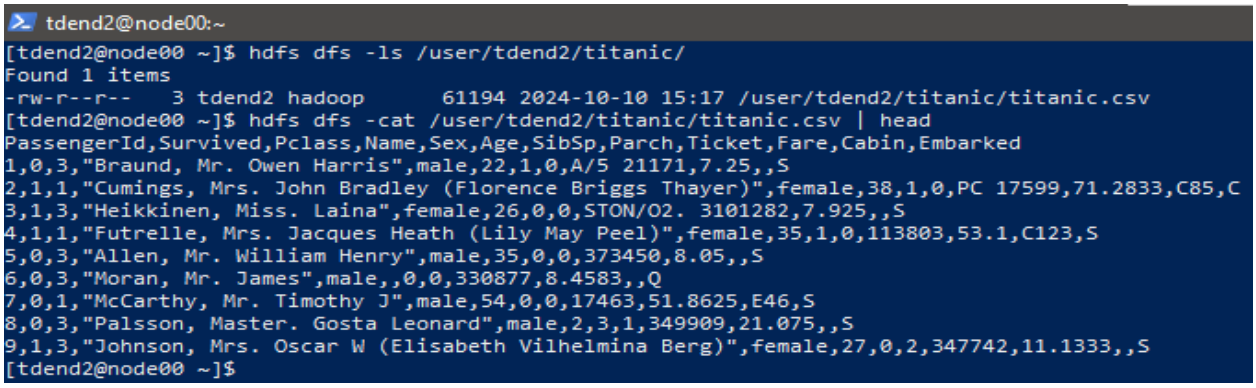
Created directory as follows to copy data from local file system to hdfs:
hdfs dfs -mkdir /user/tdend2/titanic
hdfs dfs -copyFromLocal /home/data/CSC534BDA/datasets/Titanic/titanic.csv /user/tdend2/titanic

```
tdend2@node00:~
[tdend2@node00 ~]$ hdfs dfs -mkdir /user/tdend2/titanic
```

```
tdend2@node00:~
[tdend2@node00 ~]$ hdfs dfs -ls /user/tdend2/titanic/
Found 1 items
-rw-r--r--   3 tdend2 hadoop      61194 2024-10-11 11:34 /user/tdend2/titanic/titanic.csv
[tdend2@node00 ~]$
```

Also , file can be uploaded using the below command to the desired directory:
hdfs dfs -put /home/data/CSC534BDA/datasets/Titanic/titanic.csv /user/tdend2/titanic/

Directory is created successfully and loaded contents of titanic as shown below:



As seen above, 12 columns exist in titanic.csv,which are as follows:
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked

Step2

Create a Hive table

• We may need to know how to handle double quotes.
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
In hive shell creed hive table ‘titanic’ using the above column names as follows:

```
CREATE TABLE csc534.tdend2_titanic (
   PassengerId INT,
   Survived INT,
   Pclass INT,
   Name STRING,
   Sex STRING,
   Age FLOAT,
   SibSp INT,
   Parch INT,
   Ticket STRING,
   Fare FLOAT,
   Cabin STRING,
   Embarked STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = ",",
   "quoteChar"     = "\""
   )
STORED AS TEXTFILE;
```

```
tdend2@node00:~
hive> CREATE TABLE csc534.tdend2_titanic (
    >     PassengerId INT,
    >     Survived INT,
    >     Pclass INT,
    >     Name STRING,
    >     Sex STRING,
    >     Age FLOAT,
    >     SibSp INT,
    >     Parch INT,
    >     Ticket STRING,
    >     Fare FLOAT,
    >     Cabin STRING,
    >     Embarked STRING
    > )
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > WITH SERDEPROPERTIES (
    >     "separatorChar" = ",",
    >     "quoteChar"     = "\""
    > )
    > STORED AS TEXTFILE;
OK
Time taken: 0.691 seconds
hive>
```

Step3
Load the data to Hive
We assumed the dataset is big, so loaded the data from HDFS, not the local filesystem.

LOAD DATA INPATH '/user/tdend2/titanic/titanic.csv' INTO TABLE csc534.tdend2_titanic;

```
tdend2@node00:~
hive> LOAD DATA INPATH '/user/tdend2/titanic/titanic.csv' INTO TABLE csc534.tdend2_titanic;
Loading data to table csc534.tdend2_titanic
OK
Time taken: 1.048 seconds
hive>
```

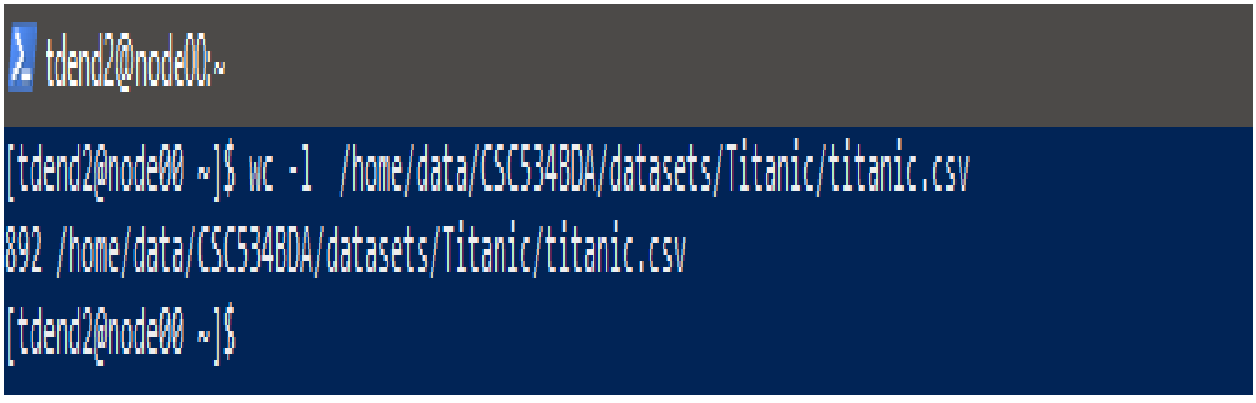
Verification: if loading was successful, comparing the line/row count of both source and target.
select count(*) from csc534.tdend2_titanic;

```
tdend2@node00:~
hive> select count(*) from csc534.tdend2_titanic;
Query ID = tdend2_20241011114635_46d90142-3baf-4891-947c-8db9ca4c2ef9
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>
24/10/11 11:46:35 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
24/10/11 11:46:35 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
Starting Job = job_1722897143033_1446, Tracking URL = http://node00.sun:8088/proxy/application_17
Kill Command = /opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/lib/hadoop/bin/hadoop job  -
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-11 11:46:45,330 Stage-1 map = 0%,  reduce = 0%
2024-10-11 11:46:50,475 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.43 sec
2024-10-11 11:46:58,672 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.43 sec
MapReduce Total cumulative CPU time: 5 seconds 890 msec
Ended Job = job_1722897143033_1446
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.89 sec   HDFS Read: 88918 HDFS Write: 103 HD
Total MapReduce CPU Time Spent: 5 seconds 890 msec
OK
891
Time taken: 25.121 seconds, Fetched: 1 row(s)
hive>
```

Row count is 891

This matched with line count of titanic.csv(included header) which is available in local linux file system as shown below:

Row count of titanic.csv available in localfile system:
wc -l /home/data/CSC534BDA/datasets/Titanic/titanic.csv



892 lines are found, excluding header row of csv file, row count is 891.

• **I**f we drop(delete) your table, we should reload the data to HDFS because the LOAD command moves the data in space of HDFS to Hive storage of HDFS.

Write analytical queries (descriptive statistics) to answer the below questions.
Step4
Calculate the survivor counts and survival/death rates.
Survivor count and survival/death rates?

```
SELECT
   survived,
   COUNT(*) AS total,
   COUNT(*) / CAST(SUM(COUNT(*)) OVER () AS DOUBLE) AS survival_death_rate
FROM csc534.tdend2_titanic
GROUP BY survived;
```

```
tdend2@node00:~
hive> SELECT
    >     survived,
    >     COUNT(*) AS total,
    >     COUNT(*) / CAST(SUM(COUNT(*)) OVER () AS DOUBLE) AS survival_death_rate
    > FROM csc534.tdend2_titanic
    > GROUP BY survived;
Query ID = tdend2_20241012114058_73d1d7cb-d870-4602-96b8-85cbaf4f4faf
```

Result:

```
Total MapReduce CPU Time Spent: 12 seconds 880 msec
OK
survived        total   survival_death_rate
1       342     0.3838383838383838
0       549     0.6161616161616161
Time taken: 48.792 seconds, Fetched: 2 row(s)
hive>
```

0 -> non-survivors from survived column: death count is 549 and accounted to 61.62%
1 -> survivors from survived column: survivors count is 342 and accounted to 38.38%

From the above, it can be concluded that death rate of titanic passengers is more than those who survived(survival rate).

Step5 (10pts)
Calculate the survivor counts and survival/death rates by survived and Sex. Note: The sum of survival rates should be 1 or 100%.

```
SELECT
   survived,
   sex,
   COUNT(*) AS total,
   COUNT(*) / CAST(SUM(COUNT(*)) OVER (PARTITION BY Survived) AS DOUBLE) AS
survival_death_rate
FROM csc534.tdend2_titanic
GROUP BY survived, sex;
```

```
hive> SELECT
    >     survived,
    >     sex,
    >     COUNT(*) AS total,
    >     COUNT(*) / CAST(SUM(COUNT(*)) OVER (PARTITION BY Survived) AS DOUBLE) AS survival_death_rate
    > FROM csc534.tdend2_titanic
    > GROUP BY survived, sex;
Query ID = tdend2_20241012114511_1f01965b-140b-4e03-8e7d-b1ed0a418fdd
```

The output is as below:

```
Total MapReduce CPU Time Spent: 7 seconds 990 msec
OK
survived        sex     total   survival_death_rate
0       female  81      0.14754098360655737
0       male    468     0.8524590163934426
1       female  233     0.6812865497076024
1       male    109     0.31871345029239767
Time taken: 24.058 seconds, Fetched: 4 row(s)
hive>
```

0-> non-survivor(deceased)
1-> survivor

Survivor counts and survival/death rates by Sex:

0-> non-survivor(deceased) count is 81 and rate among female is 14.75%
0-> non-survivor(deceased) count is 468 and rate among male is 85.25%
1-> survivor count is 233 and rate among female is 68.13%
1-> survivor count is 109 and rate among male is 31.87%
From the above, it can be concluded that ***female sex(68.13%) is higher in survival rate (not death rate) whencompared to male survivors (31.87%)***

Step6
Write a query to produce similar results of Step 4 and Step 5 Combined.

```
SELECT
   survived,
   sex,
   COUNT(*) AS total,
   COUNT(*) / CAST(SUM(COUNT(*)) OVER (PARTITION BY Survived) AS DOUBLE) AS
survival_death_rate
FROM csc534.tdend2_titanic
GROUP BY survived, sex WITH CUBE;
```

```
hive> SELECT
    >     survived,
    >     sex,
    >     COUNT(*) AS total,
    >     COUNT(*) / CAST(SUM(COUNT(*)) OVER (PARTITION BY Survived) AS DOUBLE) AS survival_death_rate
    > FROM csc534.tdend2_titanic
    > GROUP BY survived, sex WITH CUBE;
Query ID = tdend2_20241012115344_28c94442-242b-4426-8caf-618a2d3c7823
```

The output is as below:

```
Total MapReduce CPU Time Spent: 7 seconds 980 msec
OK
survived        sex     total   survival_death_rate
NULL    NULL    891     0.5
NULL    female  314     0.17620650953984288
NULL    male    577     0.32379349046015715
0       NULL    549     0.5
0       female  81      0.07377049180327869
0       male    468     0.4262295081967213
1       NULL    342     0.5
1       female  233     0.3406432748538012
1       male    109     0.15935672514619884
Time taken: 25.062 seconds, Fetched: 9 row(s)
hive>
```

==========THE END==========