Data Analytics with DW/OLAP using Hive - Truck IoT Data

Create Hive Tables

Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data queries and analysis. We use Hive as a data warehouse/OLAP tool for analyzing data.

Create Hive tables

Apache Hive provides an SQL interface to query data stored in various databases and file systems that integrate with Hadoop. Hive enables analysts familiar with SQL to run queries on large volumes of data. Hive has three main functions: data summarization, query, and analysis. Hive provides tools that enable easy data extraction, transformation, and loading (ETL).

Dataset

Created Hive tables and populated them with the Trucking IoT Data used in previous Hands-on Exercises as shown below:

Trucking IoT Data.

• Dataset: https://www.cloudera.com/content/dam/www/marketing/tutorials/beginnersguide-to-apache-pig/assets/driver_data.zip

• Related GitHub project: https://github.com/hortonworks-gallery/iot-truck-streaming

The dataset, Trucking IoT, contains the following files:

• drivers.csv

o This has driver information. It contains records showing driverId, name, ssn, location, certified, and wage-plan.

• timesheet.csv

o This contains records showing driverId, week, hours-logged, and miles-logged.

• truck_event_text_partition.csv

o This contains records showing driverId, truckId, eventTime, eventType, longitude, latitude, eventKey, CorrelationId, driverName, routeId, routeName, and eventDate

The dataset is located in /home/data/CSC534BDA/datasets/Truck-IoT of the cluster's Linux file system (Not in the HDFS).

```
tdend2@node00:~
[tdend2@node00 ~]$  ls -alFh /home/data/CSC534BDA/datasets/Truck-IoT
total 2.2M
drwxrwxr-x 2 sslee777 sslee777   84 Sep 15  2020 ./
drwxrwxr-x 6 sslee777 sslee777   76 Oct 22  2021 ../
-rw-rw-r-- 1 sslee777 sslee777 2.0K Sep 15  2020 drivers.csv
-rw-rw-r-- 1 sslee777 sslee777  26K Sep 15  2020 timesheet.csv
-rw-rw-r-- 1 sslee777 sslee777 2.2M Sep 15  2020 truck_event_text_partition.csv
[tdend2@node00 ~]$
```

Start Hive CLI

First, start the Hive CLI by typing hive

List existing databases:

```
tdend2@node00:~
hive> show databases;
OK
csc533
csc534
csc572
default
sslee777
Time taken: 1.657 seconds, Fetched: 5 row(s)
hive>
```

use the **csc534** database

```
tdend2@node00:~
hive> use csc534;
OK
Time taken: 0.04 seconds
hive>
```

Create Hive Tables

Check Schema

Check the schema of the tables, verified the first 5 rows. To see **drivers**.csv, used the 'head' Linux commands. We can see the schema (at least field/column names) in the first line: driverId, name, ssn, location, certified, and wage-plan. We can decide the field/column types based on the data later

```
tdend2@node00:~
[tdend2@node00 ~]$ head /home/data/CSC534BDA/datasets/Truck-IoT/drivers.csv
driverId,name,ssn,location,certified,wage-plan
10,George Vetticaden,621011971,244-4532 Nulla Rd.,N,miles
11,Jamie Engesser,262112338,366-4125 Ac Street,N,miles
12,Paul Coddin,198041975,Ap #622-957 Risus. Street,Y,hours
13,Joe Niemiec,139907145,2071 Hendrerit. Ave,Y,hours
14,Adis Cesir,820812209,Ap #810-1228 In St.,Y,hours
15,Rohit Bakshi,239005227,648-5681 Dui- Rd.,Y,hours
16,Tom McCuch,363303105,P.O. Box 313- 962 Parturient Rd.,Y,hours
17,Eric Mizell,123808238,P.O. Box 579- 2191 Gravida. Street,Y,hours
18,Grant Liu,171010151,Ap #928-3159 Vestibulum Av.,Y,hours
[tdend2@node00 ~]$
```

Timesheet.csv

```
[tdend2@node00 ~]$ head /home/data/CSC534BDA/datasets/Truck-IoT/timesheet.csv
driverId,week,hours-logged,miles-logged
10,1,70,3300
10,2,70,3300
10,3,60,2800
10,4,70,3100
10,5,70,3200
10,6,70,3300
10,7,70,3000
10,8,70,3300
10,9,70,3200
[tdend2@node00 ~]$
```

The truck_event_text_partition.csv is shown below.

*head /home/data/CSC534BDA/datasets/TruckIoT/**truck_event_text_partition**.csv*

```
tdend2@node00:~
[tdend2@node00 ~]$ head /home/data/CSC534BDA/datasets/Truck-IoT/truck_event_text_partition.csv
driverId,truckId,eventTime,eventType,longitude,latitude,eventKey,CorrelationId,driverName,routeId,routeName,eventDate
14,25,59:21.4,Normal,-94.58,37.03,14|25|9223370572464814373,3.66E+18,Adis Cesir,160405074,Joplin to Kansas City Route 2,2016-05-27-22
18,16,59:21.7,Normal,-89.66,39.78,18|16|9223370572464814089,3.66E+18,Grant Liu,1565885487,Springfield to KC Via Hanibal,2016-05-27-22
27,105,59:21.7,Normal,-90.21,38.65,27|105|9223370572464814070,3.66E+18,Mark Lochbihler,1325562373,Springfield to KC Via Columbia Route 2,2016-05-27-22
11,74,59:21.7,Normal,-90.2,38.65,11|74|9223370572464814123,3.66E+18,Jamie Engesser,1567254452,Saint Louis to Memphis Route2,2016-05-27-22
22,87,59:21.7,Normal,-90.04,35.19,22|87|9223370572464814101,3.66E+18,Nadeem Asghar,1198242881, Saint Louis to Chicago Route2,2016-05-27-22
22,87,59:22.3,Normal,-90.37,35.21,22|87|9223370572464813486,3.66E+18,Nadeem Asghar,1198242881, Saint Louis to Chicago Route2,2016-05-27-22
23,68,59:22.4,Normal,-89.91,40.86,23|68|9223370572464813450,3.66E+18,Adam Diaz,160405074,Joplin to Kansas City Route 2,2016-05-27-22
11,74,59:22.5,Normal,-89.74,39.1,11|74|9223370572464813355,3.66E+18,Jamie Engesser,1567254452,Saint Louis to Memphis Route2,2016-05-27-22
20,41,59:22.5,Normal,-93.36,41.69,20|41|9223370572464813344,3.66E+18,Chris Harris,160779139,Des Moines to Chicago Route 2,2016-05-27-22
[tdend2@node00 ~]$
```

Create Tables

Now we will create the tables in Hive. We already know the schema of the tables. You can find the syntax for creating tables in Hive below:

https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStartedCreatingHiveTables

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-CreateTableCreate/Drop/TruncateTable

hive> CREATE TABLE [db_name.]table_name [(col_name data_type...)] [options ...]

We also add some options, including (case insensitive)

• Define delimiter: comma

o ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ LINES TERMINATED BY ‘\n’

• Specify a file format for a Hive table (e.g., textfile, Avro, parquet, etc.)

o STORED AS TEXTFILE
• The dataset has a header in the file, so we don't want to store the header as a row.
o TBLPROPERTIES('skip.header.line.count'='1')
See details in https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL.

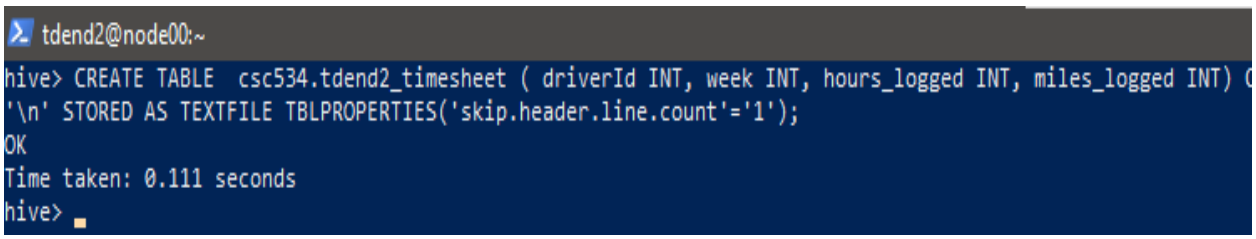
Create trucks table

CREATE TABLE csc534.**tdend2_drivers** (
driverId INT,
name STRING,
ssn BIGINT,
location STRING,
certified STRING,
wageplan STRING)
COMMENT 'drivers'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
TBLPROPERTIES('skip.header.line.count'='1');

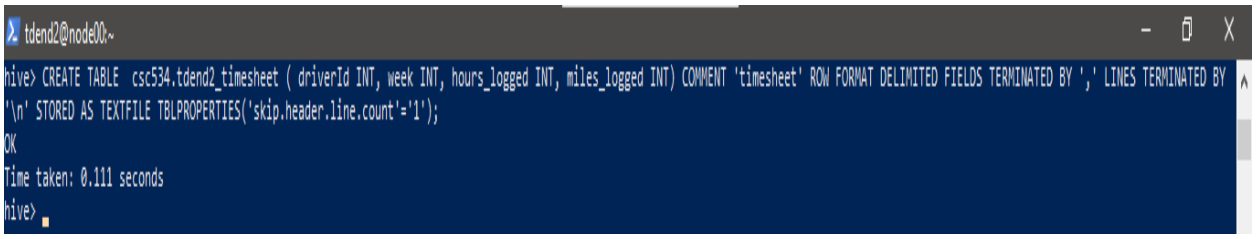
```
tdend2@node00:~
hive> CREATE TABLE csc534.tdend2_drivers (
    > driverId INT,
    > name STRING,
    > ssn BIGINT,
    > location STRING,
    > certified STRING,
    > wageplan STRING)
    > COMMENT 'drivers'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE
    > TBLPROPERTIES('skip.header.line.count'='1');
OK
Time taken: 0.437 seconds
hive>
```

To drop the above table, we can execute *DROP TABLE csc534.tdend2_drivers;*

Create the other two tables like above
• timesheet table
o using timesheet.csv
o Types of all columns - INT
• truck_event table
o using truck_event_text_partition.csv
o By seeing the records in csv, decided the types of columns.
CREATE TABLE csc534.**tdend2_timesheet** (driverId INT, week INT, hours_logged INT, miles_logged INT) COMMENT 'timesheet' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE TBLPROPERTIES('skip.header.line.count'='1');



Full command execution screenshot:



CREATE TABLE csc534.**tdend2_truck_event** (
driverId INT,
truckId INT,
eventTime VARCHAR(10), -- Time format (hh:mm.ss) needs to be a string
eventType VARCHAR(50), -- Event types like 'Normal', 'Severe', etc.
longitude DECIMAL(10, 5), -- Longitude with 5 decimal precision
latitude DECIMAL(10, 5), -- Latitude with 5 decimal precision
eventKey VARCHAR(100), -- Composite keys often have larger strings
CorrelationId VARCHAR(50), -- Exponential notation seen, so used string
driverName VARCHAR(100), -- Names can vary in length
routeId VARCHAR(20),
routeName VARCHAR(200),
eventDate DATE
)
COMMENT 'truck_event'

ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
TBLPROPERTIES('skip.header.line.count'='1');

```
hive> CREATE TABLE csc534.tdend2_truck_event (
    >     driverId INT,
    >     truckId INT,
    >     eventTime VARCHAR(10),  -- Time format (hh:mm.ss) needs to be a string
    >     eventType VARCHAR(50),  -- Event types like 'Normal', 'Severe', etc.
    >     longitude DECIMAL(10, 5),  -- Longitude with 5 decimal precision
    >     latitude DECIMAL(10, 5),   -- Latitude with 5 decimal precision
    >     eventKey VARCHAR(100),  -- Composite keys often have larger strings
    >     CorrelationId VARCHAR(50),  -- Exponential notation seen, so used string
    >     driverName VARCHAR(100),  -- Names can vary in length
    >     routeId VARCHAR(20),
    >     routeName VARCHAR(200),
    >     eventDate DATE
    > )
    > COMMENT 'truck_event'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE
    > TBLPROPERTIES('skip.header.line.count'='1');
OK
Time taken: 0.104 seconds
```

Check if they were created successfully by executing ***list***:

```
tdend2@node00:~
hive> show tables;
OK
abeer_drivers
abeer_timesheet
abeer_truck_event
mkhal44_drivers
mkhal44_timesheet
mkhal44_truck_event
operv2_drivers
operv2_timesheet
operv2_truck_event_text_partition
sslee777_drivers
sslee777_joined
sslee777_test
sslee777_timesheet
sslee777_titanic
sslee777_totals
sslee777_truck_event
sslee777_unusual_events
tdend2_drivers
tdend2_timesheet
tdend2_truck_event
vkond9_drivers
vkond9_timesheet
vkond9_truck_event
Time taken: 0.053 seconds, Fetched: 23 row(s)
hive>
```

The above created tables ***tdend2-drivers, tdend2-timesheet, tdend2_truck_event*** are seen here.

'DESCRIBE table' shows the list of columns, including partition columns for the given table:
DESCRIBE csc534.tdend2_drivers;

```
tdend2@node00:~
hive> DESCRIBE csc534.tdend2_drivers;
OK
driverid                int
name                    string
ssn                     bigint
location                string
certified               string
wageplan                string
Time taken: 0.071 seconds, Fetched: 6 row(s)
hive>
```

Show the list of the columns of the other two tables using DESCRIBE command

- **timesheet table**

DESCRIBE csc534.tdend2_timesheet;

```
tdend2@node00:~
hive> DESCRIBE csc534.tdend2_timesheet;
OK
driverid                int
week                    int
hours_logged            int
miles_logged            int
Time taken: 0.069 seconds, Fetched: 4 row(s)
hive>
```

- **truck_event table**

DESCRIBE csc534.tdend2_truck_event;

```
Select tdend2@node00:~
hive> DESCRIBE csc534.tdend2_truck_event;
OK
driverid                int
truckid                 int
eventtime               varchar(10)
eventtype               varchar(50)
longitude               decimal(10,5)
latitude                decimal(10,5)
eventkey                varchar(100)
correlationid           varchar(50)
drivername              varchar(100)
routeid                 varchar(20)
routename               varchar(200)
eventdate               date
Time taken: 0.067 seconds, Fetched: 12 row(s)
```

Load the other two datasets.

• timesheet table
o using timesheet.csv
• truck_event table
o using truck_event_text_partition.csv

2.2 Populate Hive tables
2.2.1 Loading datasets (files) into Hive tables
The Load operation moves the data into the corresponding Hive table. We need to give the local file system path if the keyword local is specified. If the keyword local is not specified, we need to use the HDFS path of the file.
Reference:
•https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStartedDMLOperations
• https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML#LanguageManualDML-Loadingfilesintotables

 # Loading trucks.csv into the trucks table in Hive
*LOAD DATA LOCAL INPATH '/home/data/CSC534BDA/datasets/Truck-IoT/**drivers**.csv' INTO TABLE csc534.**tdend2_drivers***;

```
tdend2@node00:~
hive> LOAD DATA LOCAL INPATH '/home/data/CSC534BDA/datasets/Truck-IoT/drivers.csv' INTO TABLE csc534.tdend2_drivers;
Loading data to table csc534.tdend2_drivers
OK
Time taken: 1.416 seconds
hive>
```

Populated timesheet table with data of timesheet.csv
LOAD DATA LOCAL INPATH '/home/data/CSC534BDA/datasets/Truck-IoT/**timesheet**.csv' INTO TABLE csc534.**tdend2_timesheet**;

```
tdend2@node00:~
hive> LOAD DATA LOCAL INPATH '/home/data/CSC534BDA/datasets/Truck-IoT/timesheet.csv' INTO TABLE csc534.tdend2_timesheet;
Loading data to table csc534.tdend2_timesheet
OK
Time taken: 0.731 seconds
hive>
```

Populated truck_event table with data of truck_evet-text_partition.csv
LOAD DATA LOCAL INPATH '/home/data/CSC534BDA/datasets/Truck-IoT/**truck_event_text_partition**.csv' INTO TABLE csc534.**tdend2_truck_event**;

```
tdend2@node00:~
hive> LOAD DATA LOCAL INPATH '/home/data/CSC534BDA/datasets/Truck-IoT/truck_event_text_partition.csv' INTO TABLE csc534.tdend2_truck_event;
Loading data to table csc534.tdend2_truck_event
OK
Time taken: 0.79 seconds
hive>
```

To check the data in the table, typed the following HiveQL query

From source file **'drivers', first five rows are shown below**:
head -n 5 /home/data/CSC534BDA/datasets/Truck-IoT/drivers.csv

```
tdend2@node00:~
[tdend2@node00 ~]$ head -n 5 /home/data/CSC534BDA/datasets/Truck-IoT/drivers.csv
driverId,name,ssn,location,certified,wage-plan
10,George Vetticaden,621011971,244-4532 Nulla Rd.,N,miles
11,Jamie Engesser,262112338,366-4125 Ac Street,N,miles
12,Paul Coddin,198041975,Ap #622-957 Risus. Street,Y,hours
13,Joe Niemiec,139907145,2071 Hendrerit. Ave,Y,hours
[tdend2@node00 ~]$
```

Skipped header and displayed the first five rows from the source file **'drivers'**:
`tail -n +2`: This command starts printing from the second line of the file, effectively skipping the header (the first line).
`| head -n 5`: This takes the output from the `tail` command and retrieves the first five lines from it.

```
tdend2@node00:~
[tdend2@node00 ~]$ tail -n +2 /home/data/CSC534BDA/datasets/Truck-IoT/drivers.csv | head -n 5
10,George Vetticaden,621011971,244-4532 Nulla Rd.,N,miles
11,Jamie Engesser,262112338,366-4125 Ac Street,N,miles
12,Paul Coddin,198041975,Ap #622-957 Risus. Street,Y,hours
13,Joe Niemiec,139907145,2071 Hendrerit. Ave,Y,hours
14,Adis Cesir,820812209,Ap #810-1228 In St.,Y,hours
[tdend2@node00 ~]$
```

*SELECT * FROM csc534.tdend2_drivers LIMIT 5;*

```
tdend2@node00:~
hive> SELECT * FROM csc534.tdend2_drivers LIMIT 5;
OK
10      George Vetticaden       621011971       244-4532 Nulla Rd.      N       miles
11      Jamie Engesser  262112338       366-4125 Ac Street      N       miles
12      Paul Coddin     198041975       Ap #622-957 Risus. Street       Y       hours
13      Joe Niemiec     139907145       2071 Hendrerit. Ave     Y       hours
14      Adis Cesir      820812209       Ap #810-1228 In St.     Y       hours
Time taken: 0.596 seconds, Fetched: 5 row(s)
hive>
```

Checked the total number of the rows of the populated tables as shown below:
SELECT COUNT(*) FROM csc534.tdend2_drivers;

```
tdend2@node00:~
hive> SELECT COUNT(*) FROM csc534.tdend2_drivers;
Query ID = tdend2_20241003174121_1a4c2758-d0b6-4a76-9a7b-d40f71449af9
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>
24/10/03 17:41:21 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
24/10/03 17:41:22 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
Starting Job = job_1722897143033_1010, Tracking URL = http://node00.sun:8088/proxy/application_1722897143033_1010/
Kill Command = /opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/lib/hadoop/bin/hadoop job  -kill job_1722897143033_1010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-03 17:41:32,173 Stage-1 map = 0%,  reduce = 0%
2024-10-03 17:41:37,337 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.91 sec
2024-10-03 17:41:45,534 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.71 sec
MapReduce Total cumulative CPU time: 6 seconds 710 msec
Ended Job = job_1722897143033_1010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.71 sec   HDFS Read: 10562 HDFS Write: 102 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 710 msec
OK
34
Time taken: 25.571 seconds, Fetched: 1 row(s)
hive>
```

When ran above count query, MapReduce job ran as observed from the above as Hive translates SQL queries into MapReduce tasks to process large datasets in a distributed manner across the Hadoop cluster since the data in Hive tables is usually stored in HDFS.

Also, Hive is designed to handle large datasets by leveraging Hadoop's distributed computing framework. For small datasets or local testing, you can switch Hive's execution engine from MapReduce to local mode by using:

SET hive.exec.mode.local.auto=true;

Row count from the source file **'drivers'** (using the wc -l command):
wc -l /home/data/CSC534BDA/datasets/Truck-IoT/drivers.csv

```
tdend2@node00:~
[tdend2@node00 ~]$ wc -l /home/data/CSC534BDA/datasets/Truck-IoT/drivers.csv
34 /home/data/CSC534BDA/datasets/Truck-IoT/drivers.csv
[tdend2@node00 ~]$
```

From source file **timesheet**, obtained first 5 rows as follows:

```
tdend2@node00:~
[tdend2@node00 ~]$ head -n 5 /home/data/CSC534BDA/datasets/Truck-IoT/timesheet.csv
driverId,week,hours-logged,miles-logged
10,1,70,3300
10,2,70,3300
10,3,60,2800
10,4,70,3100
[tdend2@node00 ~]$
```

Skipped header and displayed the first five rows from the source file **'timesheet'**:

```
tdend2@node00:~
[tdend2@node00 ~]$ tail -n +2 /home/data/CSC534BDA/datasets/Truck-IoT/timesheet.csv | head -n 5
10,1,70,3300
10,2,70,3300
10,3,60,2800
10,4,70,3100
10,5,70,3200
[tdend2@node00 ~]$
```

SELECT * FROM csc534.tdend2_timesheet LIMIT 5;

```
tdend2@node00:~
hive> SELECT * FROM csc534.tdend2_timesheet LIMIT 5;
OK
10      1       70      3300
10      2       70      3300
10      3       60      2800
10      4       70      3100
10      5       70      3200
Time taken: 0.076 seconds, Fetched: 5 row(s)
hive>
```

SELECT COUNT(*) FROM csc534.tdend2_timesheet;

```
tdend2@node00:~
hive> SELECT COUNT(*) FROM csc534.tdend2_timesheet;
Query ID = tdend2_20241003182916_b1e80726-6d25-4eea-8236-0d789f7262b8
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>
24/10/03 18:29:17 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
24/10/03 18:29:17 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
Starting Job = job_1722897143033_1012, Tracking URL = http://node00.sun:8088/proxy/application_1722897143033_1012/
Kill Command = /opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/lib/hadoop/bin/hadoop job  -kill job_1722897143033_1012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-03 18:29:26,571 Stage-1 map = 0%,  reduce = 0%
2024-10-03 18:29:34,802 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.5 sec
2024-10-03 18:29:39,933 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.98 sec
MapReduce Total cumulative CPU time: 5 seconds 980 msec
Ended Job = job_1722897143033_1012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.98 sec   HDFS Read: 34496 HDFS Write: 104 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 980 msec
OK
1768
Time taken: 25.327 seconds, Fetched: 1 row(s)
hive>
```

Timesheet has 1768 rows

Row count from the **source file 'timesheet'** (using the wc -l command):
wc -l /home/data/CSC534BDA/datasets/Truck-IoT/timesheet.csv

```
tdend2@node00:~
[tdend2@node00 ~]$ wc -l /home/data/CSC534BDA/datasets/Truck-IoT/timesheet.csv
1768 /home/data/CSC534BDA/datasets/Truck-IoT/timesheet.csv
[tdend2@node00 ~]$
```

From source file, showed both the first 5 rows

```
tdend2@node00:~
[tdend2@node00 ~]$ head -n 5 /home/data/CSC534BDA/datasets/Truck-IoT/truck_event_text_partition.csv
driverId,truckId,eventTime,eventType,longitude,latitude,eventKey,CorrelationId,driverName,routeId,routeName,eventDate
14,25,59:21.4,Normal,-94.58,37.03,14|25|9223370572464814373,3.66E+18,Adis Cesir,160405074,Joplin to Kansas City Route 2,2016-05-27-22
18,16,59:21.7,Normal,-89.66,39.78,18|16|9223370572464814089,3.66E+18,Grant Liu,1565885487,Springfield to KC Via Hanibal,2016-05-27-22
27,105,59:21.7,Normal,-90.21,38.65,27|105|9223370572464814070,3.66E+18,Mark Lochbihler,1325562373,Springfield to KC Via Columbia Route 2,2016-05-27-22
11,74,59:21.7,Normal,-90.2,38.65,11|74|9223370572464814123,3.66E+18,Jamie Engesser,1567254452,Saint Louis to Memphis Route2,2016-05-27-22
[tdend2@node00 ~]$
```

Skipped header and displayed the first five rows from the source file **'truckevent'**:

```
tdend2@node00:~
[tdend2@node00 ~]$ tail -n +2 /home/data/CSC534BDA/datasets/Truck-IoT/truck_event_text_partition.csv | head -n 5
14,25,59:21.4,Normal,-94.58,37.03,14|25|9223370572464814373,3.66E+18,Adis Cesir,160405074,Joplin to Kansas City
18,16,59:21.7,Normal,-89.66,39.78,18|16|9223370572464814089,3.66E+18,Grant Liu,1565885487,Springfield to KC Via
27,105,59:21.7,Normal,-90.21,38.65,27|105|9223370572464814070,3.66E+18,Mark Lochbihler,1325562373,Springfield to
11,74,59:21.7,Normal,-90.2,38.65,11|74|9223370572464814123,3.66E+18,Jamie Engesser,1567254452,Saint Louis to Memp
22,87,59:21.7,Normal,-90.04,35.19,22|87|9223370572464814101,3.66E+18,Nadeem Asghar,1198242881, Saint Louis to Chi
[tdend2@node00 ~]$
```

SELECT * FROM csc534.tdend2_truck_event LIMIT 5;

```
tdend2@node00:~
hive> SELECT *  FROM csc534.tdend2_truck_event LIMIT 5;
OK
14      25      59:21.4 Normal  -94.58000       37.03000        14|25|9223370572464814373
18      16      59:21.7 Normal  -89.66000       39.78000        18|16|9223370572464814089
27      105     59:21.7 Normal  -90.21000       38.65000        27|105|9223370572464814070
11      74      59:21.7 Normal  -90.20000       38.65000        11|74|9223370572464814123
22      87      59:21.7 Normal  -90.04000       35.19000        22|87|9223370572464814101
Time taken: 2.269 seconds, Fetched: 5 row(s)
hive>
```

SELECT COUNT(*) FROM csc534.tdend2_truck_event;

```
tdend2@node00:~
hive> SELECT COUNT(*) FROM csc534.tdend2_truck_event;
Query ID = tdend2_20241003182229_cf1b54ea-9ab5-4f7e-8a38-b239683aec7f
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>
24/10/03 18:22:30 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
24/10/03 18:22:30 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
Starting Job = job_1722897143033_1011, Tracking URL = http://node00.sun:8088/proxy/application_172
Kill Command = /opt/cloudera/parcels/CDH-6.3.2-1.cdh6.3.2.p0.1605554/lib/hadoop/bin/hadoop job  -k
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-03 18:22:40,883 Stage-1 map = 0%,  reduce = 0%
2024-10-03 18:22:49,124 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.18 sec
2024-10-03 18:22:54,254 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.6 sec
MapReduce Total cumulative CPU time: 6 seconds 600 msec
Ended Job = job_1722897143033_1011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.6 sec   HDFS Read: 2282944 HDFS Write: 105 HD
Total MapReduce CPU Time Spent: 6 seconds 600 msec
OK
17075
Time taken: 25.758 seconds, Fetched: 1 row(s)
hive>
```

Truckevent has 17075 rows

Row count from the **source file 'truck_event_text_partition'** (using the `wc -l` command):

```
tdend2@node00:~
[tdend2@node00 ~]$ wc -l /home/data/CSC534BDA/datasets/Truck-IoT/truck_event_text_partition.csv
17075 /home/data/CSC534BDA/datasets/Truck-IoT/truck_event_text_partition.csv
[tdend2@node00 ~]$
```

<u>Analytical Queries with HiveQL</u>

Query to aggregate the data

Now we have the data fields we want. The next step is to group the data by driverId, so we can find the sum of hours and miles logged score for a year. This query first groups all the records by
driverId and then selects the driver with the sum of the hours and miles logged runs for that year

SELECT driverId, sum(hours_logged), sum(miles_logged) FROM csc534.tdend2_timesheet GROUP BY driverId;

```
tdend2@node00:~
hive> SELECT driverId, sum(hours_logged), sum(miles_logged) FROM csc534.tdend2_timesheet GROUP BY driverId;
Query ID = tdend2_20241003183731_ce52ca7c-cc01-419d-8e3c-4d8aea782f60
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
```

Last half of result of above query:

```
MapReduce Total cumulative CPU time: 6 seconds 490 msec
Ended Job = job_1722897143033_1013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.49 sec   HDFS Read: 35481 HDFS Write: 1005 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 490 msec
OK
10      3232    147150
11      3642    179300
12      2639    135962
13      2727    134126
14      2781    136624
15      2734    138750
16      2746    137205
17      2701    135992
18      2654    137834
19      2738    137968
20      2644    134564
21      2751    138719
22      2733    137550
23      2750    137980
24      2647    134461
25      2723    139180
26      2730    137530
27      2771    137922
28      2723    137469
29      2760    138255
30      2773    137473
31      2704    137057
32      2736    137422
33      2759    139285
34      2811    137728
35      2728    138727
36      2795    138025
37      2694    137223
38      2760    137464
39      2745    138788
40      2700    136931
41      2723    138407
42      2697    136673
43      2750    136993
Time taken: 24.131 seconds, Fetched: 34 row(s)
hive>
```

Query to Join the data

Now we need to get the driverId(s) from the drivers table so we know who the driver(s) was. We can take the previous query and join it with the drivers records to get the final table which will have the driverId, name, and the sum of hours and miles logged

```
SELECT d.driverId, d.name, t.total_hours, t.total_miles
 FROM csc534.tdend2_drivers d
 JOIN (SELECT driverId, sum(hours_logged)total_hours, sum(miles_logged)total_miles
 FROM csc534.tdend2_timesheet
 GROUP BY driverId ) t
 ON (d.driverId = t.driverId);
```

First half of result:

```
tdend2@node00:~
hive> SELECT d.driverId, d.name, t.total_hours, t.total_miles
    > FROM csc534.tdend2_drivers d
    > JOIN (SELECT driverId, sum(hours_logged)total_hours, sum(miles_logged)total_miles
    > FROM csc534.tdend2_timesheet
    > GROUP BY driverId ) t
    > ON (d.driverId = t.driverId);
Query ID = tdend2_20241003184207_e829bfbf-b72b-4066-ad31-eae38c818354
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
24/10/03 18:42:08 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
24/10/03 18:42:08 INFO client.RMProxy: Connecting to ResourceManager at node00.sun/10.0.0.10:8032
Starting Job = job_1722897143033_1014, 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
2024-10-03 18:42:17,658 Stage-1 map = 0%,  reduce = 0%
2024-10-03 18:42:24,831 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.09 sec
2024-10-03 18:42:30,974 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 11.25 sec
MapReduce Total cumulative CPU time: 11 seconds 250 msec
Ended Job = job_1722897143033_1014
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2024-10-03 18:42:40,012 main ERROR Unable to invoke factory method in class class org.apache.hado
ang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

Last half of result:

```
tdend2@node00:~
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 0
2024-10-03 18:42:52,508 Stage-4 map = 0%,  reduce = 0%
2024-10-03 18:42:57,637 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 2.45 sec
MapReduce Total cumulative CPU time: 2 seconds 450 msec
Ended Job = job_1722897143033_1015
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 11.25 sec   HDFS Read: 34581 HDFS Write: 946 H
Stage-Stage-4: Map: 1   Cumulative CPU: 2.45 sec   HDFS Read: 6993 HDFS Write: 1456 HDFS EC Read:
Total MapReduce CPU Time Spent: 13 seconds 700 msec
OK
10      George Vetticaden       3232    147150
11      Jamie Engesser  3642    179300
12      Paul Coddin     2639    135962
13      Joe Niemiec     2727    134126
14      Adis Cesir      2781    136624
15      Rohit Bakshi    2734    138750
16      Tom McCuch      2746    137205
17      Eric Mizell     2701    135992
18      Grant Liu       2654    137834
19      Ajay Singh      2738    137968
20      Chris Harris    2644    134564
21      Jeff Markham    2751    138719
22      Nadeem Asghar   2733    137550
23      Adam Diaz       2750    137980
24      Don Hilborn     2647    134461
25      Jean-Philippe Playe     2723    139180
26      Michael Aube    2730    137530
27      Mark Lochbihler 2771    137922
28      Olivier Renault 2723    137469
29      Teddy Choi      2760    138255
30      Dan Rice        2773    137473
31      Rommel Garcia   2704    137057
32      Ryan Templeton  2736    137422
33      Sridhara Sabbella       2759    139285
34      Frank Romano    2811    137728
35      Emil Siemes     2728    138727
36      Andrew Grande   2795    138025
37      Wes Floyd       2694    137223
38      Scott Shaw      2760    137464
39      David Kaiser    2745    138788
40      Nicolas Maillard        2700    136931
41      Greg Phillips   2723    138407
42      Randy Gelhausen 2697    136673
43      Dave Patton     2750    136993
Time taken: 50.88 seconds, Fetched: 34 row(s)
hive>
```

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