Data Ingestion and Processing – Hive

Solution:

1. Start the AWS EC2 instance by logging in to AWS Management Console.
2. Change the user to **root** using command **sudo -i**.
3. Execute the command: **hive** as shown in below screenshot:

```
[root@ip-10-0-0-206 ~]# hive
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=512M; support was
removed in 8.0
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=512M; support was remove
d in 8.0

Logging initialized using configuration in jar:file:/opt/cloudera/parcels/CDH-5.15.1-1
.cdh5.15.1.p0.4/jars/hive-common-1.1.0-cdh5.15.1.jar!/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>
```

4. Create a new database **NewYork_Taxi** using below command:

```
create database NewYork_Taxi;
```

5. Use the newly created database using below command:

```
use NewYork_Taxi;
```

```
hive> create database NewYork_Taxi;
OK
Time taken: 1.949 seconds
hive> use NewYork_Taxi;
OK
Time taken: 0.037 seconds
hive>
```

6. Run below command to create an external table **yellow_trip_data**:

```
create external table yellow_trip_data(
vendorid int,
tpep_pickup_datetime timestamp,
tpep_dropoff_datetime timestamp,
passenger_count int,
trip_distance double,
ratecodeid int,
store_and_fwd_flag string ,
pulocationid int,
dolocationid int,
payment_type int,
fare_amount double,
extra double,
```

```
mta_tax double,
tip_amount double,
tolls_amount double,
improvement_surcharge double,
total_amount double )
row format delimited fields terminated by ','
location '/user/hive/warehouse/nyc_taxi_data'
tblproperties ("skip.header.line.count" = "1");
```

```
hive> create external table yellow_trip_data(
    > vendorid int,
    > tpep_pickup_datetime timestamp,
    > tpep_dropoff_datetime timestamp,
    > passenger_count int,
    > trip_distance double,
    > ratecodeid int,
    > store_and_fwd_flag string ,
    > pulocationid int,
    > dolocationid int,
    > payment_type int,
    > fare_amount double,
    > extra double,
    > mta_tax double,
    > tip_amount double,
    > tolls_amount double,
    > improvement_surcharge double,
    > total_amount double )
    > row format delimited fields terminated by ','
    > location '/user/hive/warehouse/nyc_taxi_data'
    > tblproperties ("skip.header.line.count" = "1");
OK
Time taken: 0.427 seconds
hive>
```

7. Checking all the records by running below query:

```
select * from yellow_trip_data limit 10;
```

```
hive> select * from yellow_trip_data limit 10;
OK
yellow_trip_data.vendorid       yellow_trip_data.tpep_pickup_datetime   yellow_trip_data.tpep_dropoff_datetime   yellow_trip_data.passenger_count        yellow_trip_data.trip_distance  yello
w_trip_data.ratecodeid  yellow_trip_data.store_and_fwd_flag      yellow_trip_data.pulocationid   yellow_trip_data.dolocationid   yellow_trip_data.payment_type   yellow_trip_data.fare_amounty
ellow_trip_data.extra   yellow_trip_data.mta_tax        yellow_trip_data.tip_amount     yellow_trip_data.tolls_amount   yellow_trip_data.improvement_surcharge  yellow_trip_data.total_amount
1       2017-11-01 00:01:00     2017-11-01 00:03:00     1       0.4     1       N       151     151     2       3.5     0.5     0.5     0.0     0.0     0.3     4.8
2       2017-11-01 00:31:00     2017-11-01 00:31:00     1       0.0     1       N       193     193     2       2.5     0.5     0.5     0.0     0.0     0.3     3.8
1       2017-11-01 00:25:00     2017-11-01 00:40:00     1       2.9     1       N       50      249     1       12.5    0.5     0.5     2.75    0.0     0.3     16.55
1       2017-11-01 00:17:00     2017-11-01 00:30:00     1       3.0     1       N       79      230     1       11.5    0.5     0.5     2.55    0.0     0.3     15.35
1       2017-11-01 00:38:00     2017-11-01 01:01:00     1       4.2     1       N       113     33      1       18.5    0.5     0.5     3.95    0.0     0.3     23.75
2       2017-11-01 00:02:00     2017-11-01 00:34:00     1       20.46   2       N       142     132     1       52.0    0.0     0.5     5.55    5.76    0.3     64.11
2       2017-11-01 00:10:00     2017-11-01 00:20:00     1       1.27    1       N       107     68      2       8.0     0.5     0.5     0.0     0.0     0.3     9.3
2       2017-10-31 23:59:00     2017-11-01 00:11:00     2       1.7     1       N       230     170     2       9.5     0.5     0.5     0.0     0.0     0.3     10.8
2       2017-11-01 00:07:00     2017-11-01 00:17:00     1       1.38    1       N       90      107     1       8.5     0.5     0.5     1.96    0.0     0.3     11.76
1       2017-11-01 00:10:00     2017-11-01 00:15:00     1       0.4     1       N       114     148     1       4.5     0.5     0.5     1.15    0.0     0.3     6.95
Time taken: 0.169 seconds, Fetched: 10 row(s)
hive>
```

8. Check total number of rows using below command:

```
select count(*) from yellow_trip_data;
```

```
hive> use NewYork_Taxi;
OK
Time taken: 0.983 seconds
hive> select count(*) from yellow_trip_data;
```

We see that total **1174569** rows are present in the dataset.

```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-01-11 07:11:26,109 Stage-1 map = 0%,  reduce = 0%
2022-01-11 07:11:34,857 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.85 sec
2022-01-11 07:11:42,424 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.24 sec
MapReduce Total cumulative CPU time: 7 seconds 240 msec
Ended Job = job_1641884581838_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.24 sec   HDFS Read: 109530614 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 240 msec
OK
1174569
Time taken: 41.773 seconds, Fetched: 1 row(s)
hive>
```

Basic Data Quality Checks:

1. **Question 1:** How many records has each TPEP provider provided? Write a query that summarises the number of records of each provider.

   ⇨ **Solution:** Run below command:

```
select vendorid, count(*) as No_of_Records from yellow_trip_data
group by vendorid
order by vendorid;
```

```
hive> select vendorid, count(*) as No_of_Records from yellow_trip_data
    > group by vendorid
    > order by vendorid;
```

   ⇨ **Analysis:** The TPEP Provider 1 has 527386 records and Provider 2 has 647183 records. Below is the result:

```
Total MapReduce CPU Time Spent: 16 seconds 360 msec
OK
1       527386
2       647183
Time taken: 70.089 seconds, Fetched: 2 row(s)
hive>
```

2. **Question 2:** The data provided is for months November and December only. Check whether the data is consistent, and if not, identify the data quality issues. Mention all data quality issues in comments.

   ⇨ **Solution:** Run below command:

```
select year(tpep_pickup_datetime) as trip_year,
month(tpep_pickup_datetime) as Trip_Month
from yellow_trip_data
where month(tpep_pickup_datetime) not in (11, 12)
or year(tpep_pickup_datetime) != 2017;
```

```
hive> select year(tpep_pickup_datetime) as trip_year,
    > month(tpep_pickup_datetime) as Trip_Month
    > from yellow_trip_data
    > where month(tpep_pickup_datetime) not in (11, 12)
    > or year(tpep_pickup_datetime) != 2017;
```

⇨ **Analysis:** Here, we are analyzing **November** and **December** month data of the year **2017**. After running abover query, we saw that data from other year and months are also available. So, it is a data quality issue. Below is the result:

```
Total MapReduce CPU Time Spent: 8 seconds 250 msec
OK
trip_year       trip_month
2017    10
2017    10
2017    10
2017    10
2017    10
2017    10
2009    1
2008    12
2008    12
2003    1
2018    1
2018    1
2018    1
2018    1
Time taken: 27.916 seconds, Fetched: 14 row(s)
hive>
```

3. **Question 3**: You might have encountered unusual or erroneous rows in the dataset. Can you conclude which vendor is doing a bad job in providing the records using different columns of the dataset? Summarise your conclusions based on every column where these errors are present. For example, there are unusual passenger count, i.e., 0 which is unusual.

⇨ **Solution:** Basically, we are going to check each column of the record and analyse if anything is going out of the track. Here, we are mainly concentrating on the records of year **2017** and months **November** and **December**. So, below are the checks for inconsistency:

- **Checking if NULL value is present in any column:**

```
select count(*) as Null_Records_count from yellow_trip_data
where vendorid is null
or tpep_pickup_datetime is null
or tpep_pickup_datetime is null
or passenger_count is null
or trip_distance is null
or ratecodeid is null
or store_and_fwd_flag is null
or pulocationid is null
or dolocationid is null
or payment_type is null
```

```
or fare_amount is null
or extra is null
or mta_tax is null
or tip_amount is null
or tolls_amount is null
or improvement_surcharge is null
or total_amount is null;
```

```
hive> select count(*) as Null_Records_count from yellow_trip_data
    > where vendorid is null
    > or tpep_pickup_datetime is null
    > or tpep_pickup_datetime is null
    > or passenger_count is null
    > or trip_distance is null
    > or ratecodeid is null
    > or store_and_fwd_flag is null
    > or pulocationid is null
    > or dolocationid is null
    > or payment_type is null
    > or fare_amount is null
    > or extra is null
    > or mta_tax is null
    > or tip_amount is null
    > or tolls_amount is null
    > or improvement_surcharge is null
    > or total_amount is null;
```

We see that there is no Null value present in the data set.

```
Total MapReduce CPU Time Spent: 15 seconds 620 msec
OK
null_records_count
0
Time taken: 42.922 seconds, Fetched: 1 row(s)
hive>
```

- **Checking count of customer in column "passenger_count":**

```
select distinct passenger_count from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12);
```

```
hive> select distinct passenger_count from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12);
```

We see below output of the query:

```
Total MapReduce CPU Time Spent: 16 seconds 460 msec
OK
passenger_count
0
2
4
6
8
1
3
5
7
9
Time taken: 40.203 seconds, Fetched: 10 row(s)
hive>
```

One row has value **0**. So, it is an erroneous record.

- **Checking which vendor is doing bad for column "passenger_count":**

```
select distinct vendorid from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and passenger_count = 0;
```

```
hive> select distinct vendorid from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and passenger_count = 0;
```

As per the result of below query, we see that both the vendors are doing bad because there are records with passanger count 0.

```
Total MapReduce CPU Time Spent: 14 seconds 770 msec
OK
vendorid
2
1
Time taken: 35.04 seconds, Fetched: 2 row(s)
hive>
```

- **Checking "trip_distance" column for 0 or negative values:**

```
select vendorid, count(*) from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and (trip_distance = 0 or trip_distance < 0)
group by vendorid;
```

```
hive> select vendorid, count(*) from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and (trip_distance = 0 or trip_distance < 0)
    > group by vendorid;
```

Total **7401** records (3184 + 4217) are found and both the vendors are doing bad. Below is the screenshot of the result:

```
Total MapReduce CPU Time Spent: 15 seconds 200 msec
OK
vendorid        _c1
2       3184
1       4217
Time taken: 45.309 seconds, Fetched: 2 row(s)
hive>
```

- **Checking "ratecodeid" column if any record is present apart form 1,2,3,4,5,6:**

```
select distinct ratecodeid from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12);
```

```
hive> select distinct ratecodeid from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12);
```

There is a value **99** apart from 1,2,3,4,5,6. So, records are not consistent in this column.

```
Total MapReduce CPU Time Spent: 15 seconds 530 msec
OK
ratecodeid
2
4
6
1
3
5
99
Time taken: 37.865 seconds, Fetched: 7 row(s)
hive>
```

- **Checking which vedor is doing bad for column "ratecodeid":**

```
select distinct vendorid from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and ratecodeid = 99;
```

```
hive> select distinct vendorid from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and ratecodeid = 99;
```

We could see both vendors are doing bad here because both have **ratecodeid** as **99**.

```
Total MapReduce CPU Time Spent: 15 seconds 510 msec
OK
vendorid
2
1
Time taken: 41.652 seconds, Fetched: 2 row(s)
hive>
```

- **Checking "store_and_fwd_flag" column if any record present apart from Y and N:**

```
select distinct Store_and_fwd_flag
from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12);
```

```
hive> select distinct Store_and_fwd_flag
    > from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12);
```

We see in the result that only **Y** and **N** records are present in this column.

```
Total MapReduce CPU Time Spent: 13 seconds 140 msec
OK
store_and_fwd_flag
N
Y
Time taken: 40.256 seconds, Fetched: 2 row(s)
hive>
```

- **Checking the column "payment_type" if any value other that 1 to 6:**

```
select distinct payment_type from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12);
```

```
hive> select distinct payment_type from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12);
```

There is no issue in this column because values are in range. Below is the result:

```
Total MapReduce CPU Time Spent: 13 seconds 420 msec
OK
payment_type
2
4
1
3
Time taken: 30.666 seconds, Fetched: 4 row(s)
hive>
```

- **Checking "fare_amount" column if there is any negative or 0 value:**

```
select count(*) from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and (fare_amount = 0 or fare_amount < 0);
```

```
hive> select count(*) from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and (fare_amount = 0 or fare_amount < 0);
```

As per the result, we see that there are 870 rows with the value 0.

```
Total MapReduce CPU Time Spent: 10 seconds 540 msec
OK
_c0
870
Time taken: 32.452 seconds, Fetched: 1 row(s)
hive>
```

- **Checking which vendor is doing bad for column "fare_amount":**

```
select distinct vendorid from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and fare_amount = 0;
```

```
hive> select distinct vendorid from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and fare_amount = 0;
```

Both the vendors are doing bad here as fare amount is showing 0 for them. Which is erroneous.

```
Total MapReduce CPU Time Spent: 13 seconds 710 msec
OK
vendorid
2
1
Time taken: 32.336 seconds, Fetched: 2 row(s)
hive>
```

- **Checking "extra" column if there are any values other that 0, 0.50 and 1:**

```
select vendorid, count(*) from  yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and extra not in (0,0.50,1)
group by vendorid;
```

```
hive> select vendorid, count(*) from  yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and extra not in (0,0.50,1)
    > group by vendorid;
```

We see that there are **4856** (3033 + 1823) rows where records are other that 0, 0.5 and 1. So, inconsistency is here and both the vendors are doing bad.

```
Total MapReduce CPU Time Spent: 14 seconds 970 msec
OK
vendorid        _c1
2       3033
1       1823
Time taken: 32.779 seconds, Fetched: 2 row(s)
hive>
```

- **Check the column "mta_tax" if there are values other than 0.50:**

```
select count(*) from  yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and mta_tax != 0.50;
```

```
hive> select count(*) from  yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and mta_tax != 0.50;
```

We see in the result; total **5743** rows are erroneous.

```
Total MapReduce CPU Time Spent: 10 seconds 930 msec
OK
_c0
5743
Time taken: 29.062 seconds, Fetched: 1 row(s)
hive>
```

- **Checking the column "tip_amount" if any record has negative value:**

```
select vendorid, count(*) from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and tip_amount < 0
group by vendorid;
```

```
hive> select vendorid, count(*) from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and tip_amount < 0
    > group by vendorid;
```

Total 4 rows have negative values, which are erroneous and vendor 2 doing bad job here.

```
Total MapReduce CPU Time Spent: 14 seconds 120 msec
OK
vendorid        _c1
2       4
Time taken: 34.5 seconds, Fetched: 1 row(s)
hive>
```

- **Checking "tolls_amount" column if any negative values:**

```
select vendorid, count(*) from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and tolls_amount < 0
group by vendorid;
```

```
hive> select vendorid, count(*) from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and tolls_amount < 0
    > group by vendorid;
```

Total 3 rows have negative values and vendor 2 is doing bad job here.

```
Total MapReduce CPU Time Spent: 14 seconds 430 msec
OK
vendorid        _c1
2       3
Time taken: 34.545 seconds, Fetched: 1 row(s)
hive>
```

- **Checking column "improvement_surcharge" for any value other than 0 and 0.30:**

```
select vendorid, count(*) from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and improvement_surcharge not in (0,0.30)
group by vendorid;
```

```
hive> select vendorid, count(*) from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and improvement_surcharge not in (0,0.30)
    > group by vendorid;
```

Total 562 records have values other than 0 and 0.30. Vendor 2 is doing bad.

```
Total MapReduce CPU Time Spent: 13 seconds 910 msec
OK
vendorid        _c1
2       562
Time taken: 35.681 seconds, Fetched: 1 row(s)
hive>
```

- **Checking column "total_amount" for any negative values:**

```
select vendorid, count(*) from yellow_trip_data
where year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12)
and total_amount < 0
group by vendorid;
```

```
hive> select vendorid, count(*) from yellow_trip_data
    > where year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12)
    > and total_amount < 0
    > group by vendorid;
```

558 records have negative values and vendor 2 is doing bad here as well.

```
Total MapReduce CPU Time Spent: 14 seconds 400 msec
OK
vendorid        _c1
2       558
Time taken: 37.027 seconds, Fetched: 1 row(s)
hive>
```

Assumptions:

While performing this analysis, we will inevitably make several assumptions. As per the above analysis, we can make below assumptions:

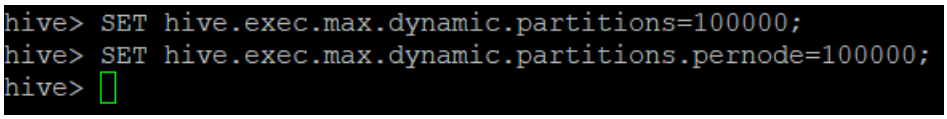
1. We are assuming that 0 passengers can't be there in "passenger_count" column. So, we will remove these rows.
2. We are assuming that distance can't be 0 or negative in "trip_distance" column. We'll remove those rows.
3. As per data dictionary, "RateCodeID" column contains only values 1 to 6. We are assuming that 99 is not a valid number. We'll remove such rows.
4. We are assuming that "fare_amount" column can't have the value 0. So, we'll remove those rows.
5. We are assuming that "Extra" column should not have negative values. So, we'll remove those rows.
6. We are assuming that as per data dictionary, "mta_tax" column should not contain any value other than 0 or 0.50. So, we'll remove those records.
7. We are assuming that "tip_amount" can't be negative. So, we'll remove those rows.
8. The column "toll_amount" should not contain negative values. So, assuming this, we will remove those rows.

9. We are assuming that the column "improvement_surcharge" should not contain any value other than 0.30. So, we'll remove those rows as well.
10. There should not be any negative value in the "total_amount" column. Assuming this, we'll remove such negative value rows. We can assume that
11. Total amount is 0 due to some offer or promocode.

ORC Partitioned Table Creation:

1. First, run below prerequisite commands to set dynamic partition configurations:

```
SET hive.exec.max.dynamic.partitions=100000;
SET hive.exec.max.dynamic.partitions.pernode=100000;
```



2. Creating ORC Partitioned table **yellow_trip_data_partition_orc** for analysis by running below query:

```
create external table yellow_trip_data_partition_orc(
vendorid int,
tpep_pickup_datetime timestamp,
tpep_dropoff_datetime timestamp,
passenger_count int,
trip_distance double,
ratecodeid int,
store_and_fwd_flag string ,
pulocationid int,
dolocationid int,
payment_type int,
fare_amount double,
extra double,
mta_tax double,
tip_amount double,
tolls_amount double,
improvement_surcharge double,
total_amount double ) partitioned by (yr int, mnth int)
stored as orc location '/user/hive/warehouse/nyc_taxi_data_ORC'
tblproperties ("orc.compress" = "SNAPPY");
```

```
hive> create external table yellow_trip_data_partition_orc(
    > vendorid int,
    > tpep_pickup_datetime timestamp,
    > tpep_dropoff_datetime timestamp,
    > passenger_count int,
    > trip_distance double,
    > ratecodeid int,
    > store_and_fwd_flag string ,
    > pulocationid int,
    > dolocationid int,
    > payment_type int,
    > fare_amount double,
    > extra double,
    > mta_tax double,
    > tip_amount double,
    > tolls_amount double,
    > improvement_surcharge double,
    > total_amount double ) partitioned by (yr int, mnth int)
    > stored as orc location '/user/hive/warehouse/nyc_taxi_data_ORC'
    > tblproperties ("orc.compress" = "SNAPPY");
```

3. Now, let's Insert the data from external table **yellow_trip_data** into **yellow_trip_data_partition_orc**:

```
insert overwrite table yellow_trip_data_partition_orc partition (yr, mnth)
select vendorid, tpep_pickup_datetime, tpep_dropoff_datetime,
passenger_count, trip_distance, ratecodeid, store_and_fwd_flag,
pulocationid, dolocationid, payment_type, fare_amount, extra,
mta_tax, tip_amount, tolls_amount, improvement_surcharge,
total_amount,
year(tpep_pickup_datetime) as yr, month(tpep_pickup_datetime) as mnth
from yellow_trip_data
where passenger_count != 0
and trip_distance > 0
and RateCodeID != 99
and fare_amount > 0
and extra >= 0
and mta_tax in (0, 0.50)
and tip_amount >= 0
and tolls_amount >= 0
and improvement_surcharge = 0.30
and total_amount >= 0 and
year(tpep_pickup_datetime) = 2017
and month(tpep_pickup_datetime) in (11,12);
```

```
hive> insert overwrite table yellow_trip_data_partition_orc partition (yr, mnth)
    > select vendorid, tpep_pickup_datetime, tpep_dropoff_datetime,
    > passenger_count, trip_distance, ratecodeid, store_and_fwd_flag,
    > pulocationid, dolocationid, payment_type, fare_amount, extra,
    > mta_tax, tip_amount, tolls_amount, improvement_surcharge,
    > total_amount,
    > year(tpep_pickup_datetime) as yr, month(tpep_pickup_datetime) as mnth
    > from yellow_trip_data
    > where passenger_count != 0
    > and trip_distance > 0
    > and RateCodeID != 99
    > and fare_amount > 0
    > and extra >= 0
    > and mta_tax in (0, 0.50)
    > and tip_amount >= 0
    > and tolls_amount >= 0
    > and improvement_surcharge = 0.30
    > and total_amount >= 0 and
    > year(tpep_pickup_datetime) = 2017
    > and month(tpep_pickup_datetime) in (11,12);
```

We get below result of the above query:

```
Total MapReduce CPU Time Spent: 22 seconds 960 msec
OK
vendorid        tpep_pickup_datetime    tpep_dropoff_datetime   passenger_count trip_
distance        ratecodeid      store_and_fwd_flag      pulocationid    dolocationidp
ayment_type     fare_amount     extra   mta_tax tip_amount      tolls_amount    impro
vement_surcharge        total_amount    yr      mnth
Time taken: 42.657 seconds
hive>
```

4. Running the below query to check all records:

```
select * from yellow_trip_data_partition_orc limit 10;
```

```
hive> select * from yellow_trip_data_partition_orc limit 10;
OK
yellow_trip_data_partition_orc.vendorid yellow_trip_data_partition_orc.tpep_pickup_datetime     yellow_trip_data_partition_orc.tpep_dropoff_datetime    yellow_trip_data_partition_orc.passen
ger_count       yellow_trip_data_partition_orc.trip_distance    yellow_trip_data_partition_orc.ratecodeid       yellow_trip_data_partition_orc.store_and_fwd_flag       yellow_trip_data_part
ition_orc.pulocationid  yellow_trip_data_partition_orc.dolocationid      yellow_trip_data_partition_orc.payment_type     yellow_trip_data_partition_orc.fare_amount      yellow_trip_data_part
ition_orc.extra yellow_trip_data_partition_orc.mta_tax  yellow_trip_data_partition_orc.tip_amount      yellow_trip_data_partition_orc.tolls_amount     yellow_trip_data_partition_orc.improv
ement_surcharge yellow_trip_data_partition_orc.total_amount     yellow_trip_data_partition_orc.yr       yellow_trip_data_partition_orc.mnth
1       2017-11-01 00:01:00     2017-11-01 00:03:00     1       0.4     1       N       151     151     2       3.5     0.5     0.5     0.0     0.0     0.3     4.8     2017    11
1       2017-11-01 00:25:00     2017-11-01 00:40:00     1       2.9     1       N       50      249     1       12.5    0.5     0.5     2.75    0.0     0.3     16.55   2017    11
1       2017-11-01 00:17:00     2017-11-01 00:30:00     1       3.0     1       N       79      230     1       11.5    0.5     0.5     2.55    0.0     0.3     15.35   2017    11
1       2017-11-01 00:38:00     2017-11-01 01:01:00     1       4.2     1       N       113     33      1       18.5    0.5     0.5     3.95    0.0     0.3     23.75   2017    11
2       2017-11-01 00:02:00     2017-11-01 00:34:00     1       20.46   2       N       142     132     1       52.0    0.0     0.5     5.55    5.76    0.3     64.11   2017    11
2       2017-11-01 00:10:00     2017-11-01 00:20:00     1       1.27    1       N       107     68      2       8.0     0.5     0.5     0.0     0.0     0.3     9.3     2017    11
2       2017-11-01 00:07:00     2017-11-01 00:17:00     1       1.38    1       N       90      107     1       8.5     0.5     0.5     1.96    0.0     0.3     11.76   2017    11
1       2017-11-01 00:10:00     2017-11-01 00:15:00     1       0.4     1       N       114     148     1       4.5     0.5     0.5     1.15    0.0     0.3     6.95    2017    11
2       2017-11-01 00:05:00     2017-11-01 00:09:00     1       1.22    1       N       239     238     2       5.5     0.5     0.5     0.0     0.0     0.3     6.8     2017    11
2       2017-11-01 00:24:00     2017-11-01 00:31:00     1       4.66    1       N       124     132     2       14.0    0.5     0.5     0.0     0.0     0.3     15.3    2017    11
Time taken: 0.107 seconds, Fetched: 10 row(s)
hive>
```

5. Checking total number of rows in the new table. Run below query:

```
select count(*) from yellow_trip_data_partition_orc;
```

```
hive> select count(*) from yellow_trip_data_partition_orc;
```

We can see the below result that total **1159746** rows are present. If we compare from the original data, total **14823** rows (1174569 - 1159746) are erroneous.

```
Total MapReduce CPU Time Spent: 5 seconds 240 msec
OK
_c0
1159746
Time taken: 22.214 seconds, Fetched: 1 row(s)
hive>
```

6. Just to be double sure, checking Null values in the newly created ORC table:

```
select count(*) as Null_Records_count
from yellow_trip_data_partition_orc
where vendorid is null
or tpep_pickup_datetime is null
or tpep_pickup_datetime is null
or passenger_count is null
or trip_distance is null
or ratecodeid is null
or store_and_fwd_flag is null
or pulocationid is null
or dolocationid is null
or payment_type is null
or fare_amount is null
or extra is null
or mta_tax is null
or tip_amount is null
or tolls_amount is null
or improvement_surcharge is null
or total_amount is null
or yr is null
or mnth is null;
```

```
hive> select count(*) as Null_Records_count
    > from yellow_trip_data_partition_orc
    > where vendorid is null
    > or tpep_pickup_datetime is null
    > or tpep_pickup_datetime is null
    > or passenger_count is null
    > or trip_distance is null
    > or ratecodeid is null
    > or store_and_fwd_flag is null
    > or pulocationid is null
    > or dolocationid is null
    > or payment_type is null
    > or fare_amount is null
    > or extra is null
    > or mta_tax is null
    > or tip_amount is null
    > or tolls_amount is null
    > or improvement_surcharge is null
    > or total_amount is null
    > or yr is null
    > or mnth is null;
```

It is clear from the below result that there is no NULL values in the table. So, our dataset is good for analysis now.

```
Total MapReduce CPU Time Spent: 7 seconds 700 msec
OK
null_records_count
0
Time taken: 28.188 seconds, Fetched: 1 row(s)
hive>
```

Analysis-I:

1. **Compare the overall average fare per trip for November and December.**

   ⇨ **Solution:** Run the below query:

```
select mnth as Month,
round(avg(fare_amount), 2) as overall_avg
from yellow_trip_data_partition_orc
group by mnth;
```

```
hive> select mnth as Month,
    > round(avg(fare_amount), 2) as overall_avg
    > from yellow_trip_data_partition_orc
    > group by mnth;
```

   ⇨ **Analysis:** From the above query, we get that average fare per trip for November and December are **13.06** and **12.85** respectively. Blelow is the result:

```
Total MapReduce CPU Time Spent: 7 seconds 170 msec
OK
month   overall_avg
11      13.06
12      12.85
Time taken: 28.216 seconds, Fetched: 2 row(s)
hive>
```

2. **Explore the 'number of passengers per trip' - how many trips are made by each level of 'Passenger_count'?**

   ⇨ **Solution:** Below is the query:

```
select passenger_count,
count(passenger_count) Trip_Count
from yellow_trip_data_partition_orc
group by passenger_count
order by Trip_count desc;
```

```
hive> select passenger_count,
    > count(passenger_count) Trip_Count
    > from yellow_trip_data_partition_orc
    > group by passenger_count
    > order by Trip_count desc;
```

⇨ **Analysis:** After running above query, it is clear that most people travel **solo**. Below is the result:

```
Total MapReduce CPU Time Spent: 11 seconds 430 msec
OK
passenger_count trip_count
1       821262
2       175841
5       54342
3       50452
6       33037
4       24809
7       3
Time taken: 56.593 seconds, Fetched: 7 row(s)
hive>
```

3. **Which is the most preferred mode of payment?**

⇨ **Solution:** Run below query:

```
select payment_type,
count(*) as Number_of_Payment
from yellow_trip_data_partition_orc
group by payment_type
order by Number_of_Payment desc;
```

```
hive> select payment_type,
    > count(*) as Number_of_Payment
    > from yellow_trip_data_partition_orc
    > group by payment_type
    > order by Number_of_Payment desc;
```

⇨ **Analysis:** So, we get that most preferred mode of payment is **1 (Credit Card)**. Below is the result:

```
Total MapReduce CPU Time Spent: 10 seconds 830 msec
OK
payment_type    number_of_payment
1       782803
2       370832
3       4783
4       1328
Time taken: 55.815 seconds, Fetched: 4 row(s)
hive>
```

4. **What is the average tip paid per trip? Compare the average tip with the 25th, 50th and 75th percentiles and comment whether the 'average tip' is a representative statistic (of the central tendency) of 'tip amount paid'.**

   ⇨ **Solution:** Below is the query for **Average Tip**:

```
select round(avg(tip_amount), 2) as Average_Tip
from yellow_trip_data_partition_orc;
```

```
hive> select round(avg(tip_amount), 2) as Average_Tip
    > from yellow_trip_data_partition_orc;
```

Below are the queries to calculate **25th**, **50th** and **75th** percentile of tip amount:

```
select round(percentile_approx(tip_amount, 0.25), 2) as
25th_percentile_trip_amount,
round(percentile_approx(tip_amount, 0.50), 2) as
50th_percentile_trip_amount,
round(percentile_approx(tip_amount, 0.75), 2) as
75th_percentile_trip_amount
from yellow_trip_data_partition_orc;
```

```
hive> select round(percentile_approx(tip_amount, 0.25), 2) as 25th_percentile_trip_amount,
    > round(percentile_approx(tip_amount, 0.50), 2) as 50th_percentile_trip_amount,
    > round(percentile_approx(tip_amount, 0.75), 2) as 75th_percentile_trip_amount
    > from yellow_trip_data_partition_orc;
```

   ⇨ **Analysis:** Below are the analysis of the results obtained by above queries:
   - Average Tip per trip is **1.85**
   - 25th percentile of tip amount is **0**
   - 50th percentile of tip amount is **1.36**
   - 75th percentile of tip amount is **2.45**
   - So, as per the above data, average tip is a representative statistic (of the central tendency) of 'tip amount paid'.

```
Total MapReduce CPU Time Spent: 7 seconds 290 msec
OK
average_tip
1.85
Time taken: 30.025 seconds, Fetched: 1 row(s)
hive>
```

```
Total MapReduce CPU Time Spent: 9 seconds 190 msec
OK
25th_percentile_trip_amount     50th_percentile_trip_amount     75th_percentile_trip_amou
nt
0.0     1.36    2.45
Time taken: 27.927 seconds, Fetched: 1 row(s)
hive>
```

5. **Explore the 'Extra' (charge) variable - what fraction of total trips have an extra charge is levied?**

⇨ **Solution:** Below is the formula in the query:

```
select round((extra_charges_count * 100)/1159746, 2) as
extra_charge_percentage
from (
select count(*) as extra_charges_count
from yellow_trip_data_partition_orc
where extra > 0) Temp;
```

```
hive> select round((extra_charges_count * 100)/1159746, 2) as extra_charge_percentage
    > from (
    > select count(*) as extra_charges_count
    > from yellow_trip_data_partition_orc
    > where extra > 0) Temp;
```

⇨ **Analysis:** After running above query, we see that 46.33 % of total trips have an extra charge levied. Please find below result:

```
Total MapReduce CPU Time Spent: 6 seconds 880 msec
OK
extra_charge_percentage
46.33
Time taken: 28.122 seconds, Fetched: 1 row(s)
hive>
```

Analysis-II:

1. **What is the correlation between the number of passengers on any given trip, and the tip paid per trip? Do multiple travellers tip more compared to solo travellers?**

   ⇨ **Solution:** Below is the query to solve this problem:

```
select round(corr(passenger_count, tip_amount), 4)
from yellow_trip_data_partition_orc;
```

```
hive> select round(corr(passenger_count, tip_amount), 4)
    > from yellow_trip_data_partition_orc;
```

   ⇨ **Analysis:** As per the result of the above query, we got the correlation value as **-0.005**. It is very small value which does not proof any strong relationship between **passenger_count** and **tip_amount**.

```
Total MapReduce CPU Time Spent: 7 seconds 640 msec
OK
_c0
-0.005
Time taken: 27.379 seconds, Fetched: 1 row(s)
hive>
```

   Below is the query to check tip amount paid by solo and multiple travellers:

```
select passenger_count,
round(avg(tip_amount), 2) as Average_Tip
```

```
from yellow_trip_data_partition_orc
group by passenger_count;
```

```
hive> select passenger_count,
    > round(avg(tip_amount), 2) as Average_Tip
    > from yellow_trip_data_partition_orc
    > group by passenger_count;
```

We got below result:

```
Total MapReduce CPU Time Spent: 7 seconds 260 msec
OK
passenger_count average_tip
1       1.86
2       1.86
3       1.77
4       1.62
5       1.88
6       1.84
7       5.23
Time taken: 27.52 seconds, Fetched: 7 row(s)
hive>
```

⇨ **Analysis:** Below are the observations:

- As per above result, it is clear that average tip amount does not matter much with solo or multiple travellers.
- All level of travellers gives almost same average tip amount except the "level 7".

2. **Segregate the data into five segments of 'tip paid': [0-5), [5-10), [10-15) , [15-20) and >=20. Calculate the percentage share of each bucket.**

⇨ **Solution:** We have used CASE clause here to run the conditions in the query:

```
select Tip_bucket,
round((count(*)*100/1159746),4) No_of_tips from (
select (
        case when (tip_amount>=0 and tip_amount<5)   then '[0-5)'
             when (tip_amount>=5 and tip_amount<10)  then '[5-10)'
             when (tip_amount>=10 and tip_amount<15) then '[10-15)'
             when (tip_amount>=15 and tip_amount<20) then '[15-20)'
             when (tip_amount>=20)                   then '>=20'
         end) as Tip_bucket
from yellow_trip_data_partition_orc) as Temp
group by Tip_bucket
order by No_of_Tips desc;
```

```
hive> select Tip_bucket,
    > round((count(*)*100/1159746),4) No_of_tips from (
    > select (
    >         case when (tip_amount>=0 and tip_amount<5)   then '[0-5)'
    >              when (tip_amount>=5 and tip_amount<10)  then '[5-10)'
    >              when (tip_amount>=10 and tip_amount<15) then '[10-15)'
    >              when (tip_amount>=15 and tip_amount<20) then '[15-20)'
    >              when (tip_amount>=20)                   then '>=20'
    >          end) as Tip_bucket
    > from yellow_trip_data_partition_orc) as Temp
    > group by Tip_bucket
    > order by No_of_Tips desc;
```

After running above query, we see below result:

```
Total MapReduce CPU Time Spent: 12 seconds 940 msec
OK
tip_bucket      no_of_tips
[0-5)   92.178
[5-10)  5.6576
[10-15) 1.8461
[15-20) 0.2265
>=20    0.0917
Time taken: 54.098 seconds, Fetched: 5 row(s)
hive>
```

⇨ **Analysis:** As per above results, the bucket **[0-5)** shares highest percentage of tip share.

3. **Which month has a greater average 'speed' - November or December?**

⇨ **Solution:** Below is the query where first we converted miles into meters and applied speed = distance/Time formula:

```
select mnth,
round(avg(trip_distance * 1609.34/(unix_timestamp(tpep_dropoff_datetime) -
unix_timestamp(tpep_pickup_datetime))), 2) as `Average_Speed(Meter/Second)`
from yellow_trip_data_partition_orc
group by mnth;
```

```
hive> select mnth,
    > round(avg(trip_distance * 1609.34/(unix_timestamp(tpep_dropoff_datetime) - unix_timestamp(tpep_pickup_datetime))), 2) as `Average_Speed(Meter/Second)`
    > from yellow_trip_data_partition_orc
    > group by mnth;
```

⇨ Analysis: As per above query, December has greater average speed **4.96 meter/second**. Below is the result:

```
Total MapReduce CPU Time Spent: 9 seconds 340 msec
OK
mnth    average_speed(meter/second)
11      4.92
12      4.96
Time taken: 27.387 seconds, Fetched: 2 row(s)
hive>
```

4. **Analyse the average speed of the most happening days of the year, i.e., 31st December (New Year's Eve) and 25th December (Christmas) and compare it with the overall average.**
   ⇨ **Solution:** Use below formula in the query to obtain the required result:

```
select day(tpep_pickup_datetime) as Special_Day,
round(avg(trip_distance * 1609.34/(unix_timestamp(tpep_dropoff_datetime) -
unix_timestamp(tpep_pickup_datetime))), 2) as `Average_Speed(Meter/Second)`
from yellow_trip_data_partition_orc
where mnth = 12
and (day(tpep_pickup_datetime) = 25
or day(tpep_pickup_datetime) = 31)
group by day(tpep_pickup_datetime);
```

```
hive> select day(tpep_pickup_datetime) as Special_Day,
    > round(avg(trip_distance * 1609.34/(unix_timestamp(tpep_dropoff_datetime) - unix_tim
estamp(tpep_pickup_datetime))), 2) as `Average_Speed(Meter/Second)`
    > from yellow_trip_data_partition_orc
    > where mnth = 12
    > and (day(tpep_pickup_datetime) = 25
    > or day(tpep_pickup_datetime) = 31)
    > group by day(tpep_pickup_datetime);
```

Below is the result of above query:

```
Total MapReduce CPU Time Spent: 9 seconds 90 msec
OK
special_day     average_speed(meter/second)
25      6.82
31      5.92
Time taken: 28.107 seconds, Fetched: 2 row(s)
hive>
```

Overall Average Speed: Below is the query:

```
select round(avg(trip_distance * 1609.344 /
(unix_timestamp(tpep_dropoff_datetime) -
unix_timestamp(tpep_pickup_datetime))), 2) as Avg_overall_speed
from yellow_trip_data_partition_orc;
```

```
hive> select round(avg(trip_distance * 1609.344 / (unix_timestamp(tpep_dropoff_datetime) - unix_timestamp(tpep_pickup_datetime))), 2) as Avg_overall_speed
    > from yellow_trip_data_partition_orc;
```

As per the above query, we get average overall speed as **4.94 meter/sec**.

```
Total MapReduce CPU Time Spent: 8 seconds 480 msec
OK
avg_overall_speed
4.94
Time taken: 30.107 seconds, Fetched: 1 row(s)
hive>
```

   ⇨ **Analysis:** As per abover results, average speed on **25th December** is higher. But, average speed on 25th December and 31st December are higher than overall average speed.

Conclusion:

Data ingestion is the transportation of data from assorted sources to a storage medium where it can be accessed, used, and analysed. Here, in our case study, we imported data from Hadoop Distributed File System (HDFS) into Hive Tables. We performed various operations on external and partitioned tables. Below are the highlighted points:

1. Created a Hive table with the correct schema in the appropriate format to store the data.
2. Performed Data Quality Check and identified the unusual or erroneous values and checked the inconsistencies.
3. Used Hive's built-in function(s) to extract the appropriate segment of timestamp for easy and readable querying.
4. Made several domain related assumptions for the incorrect data and accomplished analysis accordingly. All the results were displayed correctly.
5. Performed various analysis using different queries and aggregation and answered all the asked questions.