Assignment 1

NYC-Yellow Taxi

November 2nd 2021

Team #5 member: Nazanin Hashemipoor, Sofienne Srihi, Nasim Afzali Chali, Marie-Noel lepage, Benoit Tessier

In our team, all of us were working and participating. We all worked on code on our own and shared our progress with each other to solve issues and obtain a final working code. In parallel we produced and reviewed the documentation to supplement the exercise.

Scenario:

The New York City Taxi & Limousine Commission (TLC) has provided a dataset of trips made by the taxis in New York City. Based on real data, the data used was collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP).

The purpose of this dataset is to get a better understanding of the taxi system so that the city of New York can improve the efficiency of in-city commutes.

Used Dataset:

"Real TLC Data is published Every month. A second CSV file Contains lookup information"

Data storage:

In this part, we use Hadoop to store our data in HDFS.

Task 01: Store the input files.

- Unzip the data files in Mobaxterm
- We unzipped the files and we upload all of them in cloudera with Mobaextern

$tar -xf /home/cloudera/downloads/yellow_tripdata_2020-01.tar.gz
$tar -xf /home/cloudera/downloads/yellow_tripdata_2020-02.tar.gz

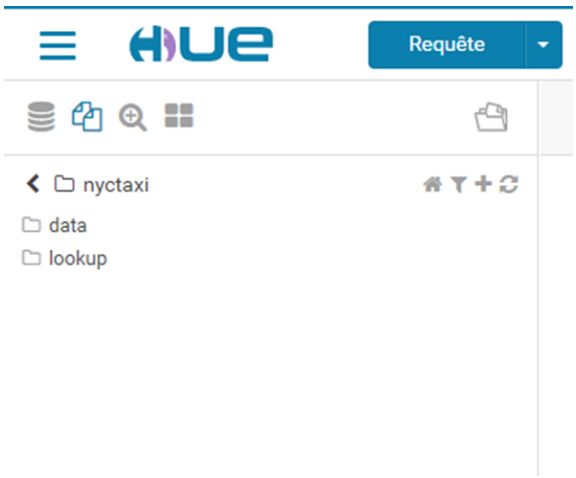
- Create a new Directory and sub-directories

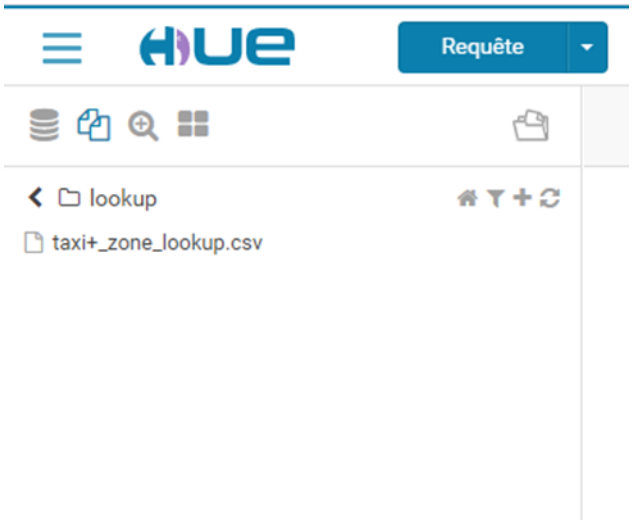
  hdfs dfs -mkdir -p /nyctaxi/data /nyctaxi/lookup

- Get the list of the sub-directories

  hdfs dfs -ls /nyctaxi

```
Found 2 items
drwxr-xr-x   - root supergroup          0 2021-10-28 19:24 /nyctaxi/data
drwxr-xr-x   - root supergroup          0 2021-10-28 19:25 /nyctaxi/lookup
```





- Place data lookup into the directory

  hdfs dfs -put /home/cloudera/Downloads/taxi+_zone_lookup.csv /nyctaxi/lookup

  hdfs dfs -put /home/cloudera/Downloads/yellow_tripdata_2020.csv /nyctaxi/data*

- Get the list of the file in the directory

  hdfs dfs -ls -R -h /nyctaxi/

```
drwxr-xr-x   - root supergroup          0 2021-10-28 19:24 /nyctaxi/data
-rw-r--r--   1 root supergroup    566.1 M 2021-10-28 19:24 /nyctaxi/data/yellow_tripdata_2020-01.csv
-rw-r--r--   1 root supergroup    557.1 M 2021-10-28 19:24 /nyctaxi/data/yellow_tripdata_2020-02.csv
drwxr-xr-x   - root supergroup          0 2021-10-28 19:25 /nyctaxi/lookup
-rw-r--r--   1 root supergroup     12.0 K 2021-10-28 19:25 /nyctaxi/lookup/taxi+_zone_lookup.csv
```

Data Normalization:

In this part, we use the Pig Script to normalize, Clean, and filter input data.

Task 02: Pig Latin Script to Prepare TLC Data.

- Import the Piggybank library for use in the task below. Piggybank has a lot of functions that can use in our script and make our job easier.

  ```
  register /usr/lib/pig/piggybank.jar;
  ```

- Load the TLC data and remove the header line

  We use the CSExcelStorage function in the Piggybank library.

  ```
  TLC= LOAD '/nyctaxi/data/yellow_tripdata_2020-*.csv' Using
  org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX',
  'SKIP_INPUT_HEADER') AS
    (
    VendorID:chararray,
    tpep_pickup_datetime:chararray,
    tpep_dropoff_datetime:chararray,
    passenger_count:int,
    trip_distance:double,
    RatecodeID:int,
    store_and_fwd_flag:chararray,
    PULocationID:chararray,
    DOLocationID:chararray,
    payment_type:double,
    fare_amount:double,
    extra:double,
  ```

mta_tax:double,
tip_amount:double,
tolls_amount:double,
improvement_surcharge:double,
total_amount:double,
congestion_surcharge:double
);

- Split the Input data based on the Vendor ID Column

SPLIT TLC into VendorID_1 if (VendorID == '1'), VendorID_2 otherwise;

- Update the Split data to eliminate VendorID

Script1:
Vendor1= FOREACH VendorID_1 GENERATE 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, congestion_surcharge;

Script2:
Vendor2= FOREACH VendorID_2 GENERATE 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, congestion_surcharge;

- Store results into '/nycdata/" directory

Store Vendor1 INTO '/nycdata/VendorID_1' USING PigStorage(',');

Store Vendor2 INTO '/nycdata/VendorID_2' USING PigStorage(',');

Hdfs dfs -ls -R -h /nycdata

```
drwxr-xr-x   - cloudera supergroup          0 2021-10-31 13:11 /nycdata/VendorID_1
-rw-r--r--   1 cloudera supergroup          0 2021-10-31 13:11 /nycdata/VendorID_1/_SUCCESS
-rw-r--r--   1 cloudera supergroup     42.2 M 2021-10-31 13:10 /nycdata/VendorID_1/part-m-00000
-rw-r--r--   1 cloudera supergroup     43.2 M 2021-10-31 13:10 /nycdata/VendorID_1/part-m-00001
-rw-r--r--   1 cloudera supergroup     43.0 M 2021-10-31 13:10 /nycdata/VendorID_1/part-m-00002
-rw-r--r--   1 cloudera supergroup     42.9 M 2021-10-31 13:10 /nycdata/VendorID_1/part-m-00003
-rw-r--r--   1 cloudera supergroup     42.9 M 2021-10-31 13:10 /nycdata/VendorID_1/part-m-00004
-rw-r--r--   1 cloudera supergroup     42.8 M 2021-10-31 13:10 /nycdata/VendorID_1/part-m-00005
-rw-r--r--   1 cloudera supergroup     42.6 M 2021-10-31 13:11 /nycdata/VendorID_1/part-m-00006
-rw-r--r--   1 cloudera supergroup     42.5 M 2021-10-31 13:11 /nycdata/VendorID_1/part-m-00007
-rw-r--r--   1 cloudera supergroup     30.0 M 2021-10-31 13:11 /nycdata/VendorID_1/part-m-00008
drwxr-xr-x   - cloudera supergroup          0 2021-10-31 13:08 /nycdata/VendorID_2
-rw-r--r--   1 cloudera supergroup          0 2021-10-31 13:08 /nycdata/VendorID_2/_SUCCESS
-rw-r--r--   1 cloudera supergroup     88.6 M 2021-10-31 13:06 /nycdata/VendorID_2/part-m-00000
-rw-r--r--   1 cloudera supergroup     87.5 M 2021-10-31 13:06 /nycdata/VendorID_2/part-m-00001
-rw-r--r--   1 cloudera supergroup     87.7 M 2021-10-31 13:06 /nycdata/VendorID_2/part-m-00002
-rw-r--r--   1 cloudera supergroup     87.8 M 2021-10-31 13:06 /nycdata/VendorID_2/part-m-00003
```

Took 4 sec. Last updated by anonymous at October 31 2021, 4:11:36 PM.

hdfs dfs -cat /nycdata/VendorID_1/part-m-00007 | head -n 5

```
2020-02-21 13:24:04,2020-02-21 13:48:15,1,5.0,1,N,90,238,1,21.0,2.5,0.5,4.85,0.0,0.3,29.15,2.5
2020-02-21 13:21:44,2020-02-21 13:58:39,1,8.8,1,N,138,246,1,32.0,2.5,0.5,8.25,6.12,0.3,49.67,2.5
2020-02-21 13:18:33,2020-02-21 13:28:33,1,1.7,1,N,237,263,1,8.5,2.5,0.5,1.77,0.0,0.3,13.57,2.5
2020-02-21 13:45:43,2020-02-21 13:55:42,1,1.5,1,N,237,233,1,8.5,2.5,0.5,2.36,0.0,0.3,14.16,2.5
2020-02-21 13:56:29,2020-02-21 14:04:54,1,2.5,1,N,233,145,1,9.0,2.5,0.5,3.65,6.12,0.3,22.07,2.5
```

- Report rows count for each split

  With hdfs:

  hdfs dfs -cat /nycdata/VendorID_1/part-m- | wc -l*

  hdfs dfs -cat /nycdata/VendorID_2/part-m- | wc -l*

```
%sh
hdfs dfs -cat /nycdata/VendorID_1/part-m-* | wc -l
```

```
4150850
```

Took 7 sec. Last updated by anonymous at October 31 2021, 2:23:47 PM.

```
%sh
hdfs dfs -cat /nycdata/VendorID_2/part-m-* | wc -l
```

```
8553513
```

With pig:

cnt_1 = Foreach (GROUP Vendor1 ALL) generate COUNT(Vendor1);

DUMP cnt_1;

```
%pig
DUMP cnt_1;
```

```
(4150850)
```

Took 4 min 56 sec. Last updated by anonymous at October 30 2021, 10:52:11 AM.

cnt_2 = Foreach (GROUP Vendor2 ALL) generate COUNT(Vendor2);

DUMP cnt_2;

```
%pig
DUMP cnt_2;
```

```
(8553513)
```

Took 1 min 34 sec. Last updated by anonymous at October 30 2021, 10:55:56 AM.

Task 03: Pig Latin Script to Prepare Lookup Data.

- Look up data, remove header line, and remove double quote character and replace it by an empty character

Lookup= LOAD '/nyctaxi/lookup/taxi+_zone_lookup.csv' Using org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER') AS

(LocationID:chararray,

Borough:chararray,

Zone:chararray,

service_zone:chararray

);

- Store output result on HDFS into '/nyclookup'/ directory

STORE Lookup INTO '/nyclookup/' USING PigStorage(',');

```
%sh

hdfs dfs -ls -R /nyclookup
```

```
-rw-r--r--   1 root supergroup          0 2021-10-28 13:35 /nyclookup/_SUCCESS
-rw-r--r--   1 root supergroup      10421 2021-10-28 13:35 /nyclookup/part-m-00000
```

Took 2 sec. Last updated by anonymous at October 28 2021, 4:35:41 PM. (outdated)

hdfs dfs -cat /nyclookup/part-m-00000 | head -n 5

```
1,EWR,Newark Airport,EWR
2,Queens,Jamaica Bay,Boro Zone
3,Bronx,Allerton/Pelham Gardens,Boro Zone
4,Manhattan,Alphabet City,Yellow Zone
5,Staten Island,Arden Heights,Boro Zone
```

Data Modeling:

Task 04: Creating TLC Table-Hive

- Create a database nyc

  %hive

  create database if not exists nyc

- Use database

  Use nyc

- Show database

  Show databases

- Create a new Hive user-managed table named nycTaxi

  create external table if not exists nycTaxi
  (
  tpep_pickup_datetime string,
  tpep_dropoff_datetime string,
  passenger_count int,
  trip_distance double,
  RatecodeID int,
  store_and_fwd_flag string,
  PULocationID string,
  DOLocationID string,
  payment_type int,

```
fare_amount double,
extra double,
mta_tax double,
tip_amount double,
tolls_amount double,
improvement_surcharge double,
total_amount double,
congestion_surcharge double
)
PARTITIONED BY (VendorID int)
row format delimited fields terminated by ','
STORED AS TEXTFILE
```

```
ALTER TABLE nycTaxi ADD PARTITION (VendorID = '1') LOCATION
'/nycdata/VendorID_1/'

ALTER TABLE nycTaxi ADD PARTITION (VendorID = '2') LOCATION
'/nycdata/VendorID_2/'
```

- Report table partitions number

```
show partitions nycTaxi
```

partition
vendorid=1
vendorid=2

- Use analyze table to get the compute statistics

```
ANALYZE TABLE nycTaxi PARTITION(VendorID = '1') COMPUTE STATISTICS

DESCRIBE formatted nycTaxi PARTITION(VendorID = '1')
```

col_name	data_type	comment
	numFiles	9
	numRows	4150850
	rawDataSize	386004613
	totalSize	390155463

ANALYZE TABLE nycTaxi PARTITION(VendorID = '2') COMPUTE STATISTICS

DESCRIBE formatted nycTaxi PARTITION(VendorID = '2')

col_name	data_type	comment
	numFiles	9
	numRows	8553513
	rawDataSize	804177988
	totalSize	812731501

- Report the rows count for partitioned tables.

```
select vendorid, count(*) as cnt
 from nycTaxi
where vendorid in (1,2)
group by vendorid
order by vendorid
```

VendorID 1=4150850

VendorID2=8553513

vendorid	cnt
1	4150850
2	8553513

- Show the first 10 rows from table

*select * from nycTaxi where vendorid in (1,2) limit 10*

nyctaxi.tpep_pickup_datetime	nyctaxi.tpep_dropoff_datetime	nyctaxi.passenger_count	nyctaxi.trip_distance	nyctaxi.ratecodeid	nyctaxi.store_and_fwd_flag	nycta
2020-01-01 00:28:15	2020-01-01 00:33:03	1	1.2	1	N	238
2020-01-01 00:35:39	2020-01-01 00:43:04	1	1.2	1	N	239
2020-01-01 00:47:41	2020-01-01 00:53:52	1	0.6	1	N	238
2020-01-01 00:55:23	2020-01-01 01:00:14	1	0.8	1	N	238
2020-01-01 00:29:01	2020-01-01 00:40:28	2	0.7	1	N	246
2020-01-01 00:55:11	2020-01-01 01:12:03	2	2.4	1	N	246
2020-01-01 00:37:15	2020-01-01 00:51:41	1	0.8	1	N	163
2020-01-01 00:56:27	2020-01-01 01:21:44	1	3.3	1	N	161

TASK 05: Creating Lookup table

- Create a new Hive user-managed table taxiLookup

```
create external table if not exists taxiLookup
    (
    LocationID string,
    Borough string,
    Zone string,
    service_zone string
    )
    row format delimited fields terminated by ','
LOCATION "/nyclookup"
```

- Analyze the table to get the compute statistics

```
ANALYZE TABLE taxiLookup COMPUTE STATISTICS
```

- Get the size of the HDFS table and count of the rows

```
DESCRIBE FORMATTED  taxiLookup
```

Number of rows:265
Totalsize:10421

col_name	data_type	comment
	numRows	265
	rawDataSize	10156
	totalSize	10421

TASK 06: Hive Partitioned Table

- Create a table partitioned by Month, Bucketed by Passenger Count and sorted by trip distance

Based on our data, we have 0 value for nine taxi passengers. We dropped 0 and null due to the not significance of data value, so we choosed 9 buckets.

select passenger_count, count(*) as cnt
from nycTaxi
where vendorid in (1,2)
group by passenger_count
order by passenger_count

passenger_count	cnt
null	114276
0	237885
1	9042491
2	1873085
3	491854
4	239110
5	444693
6	260852

```
create table nycTaxi_part_bkt
(
tpep_pickup_datetime string,
   tpep_dropoff_datetime string,
   passenger_count int,
   trip_distance double,
   RatecodeID int,
   store_and_fwd_flag string,
   PULocationID string,
   DOLocationID string,
   payment_type int,
   fare_amount double,
   extra double,
   mta_tax double,
   tip_amount double,
   tolls_amount double,
   improvement_surcharge double,
   total_amount double,
   congestion_surcharge double)
   PARTITIONED BY (Month string)
   CLUSTERED BY(passenger_count) SORTED BY (trip_distance) INTO 9 BUCKETS
```

```
STORED as TEXTFILE
Location '/nycdata_part/
```

Change parameters to bucketing:

```
set hive.enforce.bucketing = true;
```

Change parameters to sorting:

```
set hive.enforce.sorting = true;
```

Data Analysis:

Task 07: Analysis I using Impala

In this section, we are checking whether the data is consistent or not.
We use the Impala queries to read the data from nycTaxi table.

- Refresh the metastore in cashes.
  invalidate metadata

- *Showing the creation of nycTaxi_part_bkt*

tab_name
nyctaxi
nyctaxi_part_bkt
taxilookup

- Summerize the number of records of each vendor

  SELECT vendorid, count(*) as sum_records from nycTaxi group BY vendorid order by vendorid

vendorid	sum_records
1	4150850
2	8553513

Vendorid1=4150850
VendorID2=8553513

- Report the total rows count that does not belong to January 2020 and February 2020

```
select count(*) as count from nycTaxi
where left(tpep_pickup_datetime,7) not in ('2020-01','2020-02')
and left(tpep_dropoff_datetime,7) not in ('2020-01','2020-02')
```

count
192

- Report The rows Counts for each date except January 2020 and February 2020. The same previous answer but with more details.

```
Select
coalesce(t.Date_pickup, 'TOTAL') as Date_pickup,
coalesce(t.Date_dropoff, '~TOTAL') as Date_dropoff,
t.Total as Total
From
(
Select strleft(tpep_pickup_datetime, 7) as Date_pickup, strleft(tpep_dropoff_datetime, 7) as
Date_dropoff, count(*) as Total
from nycTaxi
where strleft(tpep_pickup_datetime, 7) not in ('2020-01','2020-02')
and strleft(tpep_dropoff_datetime,7) not in ('2020-01','2020-02')
group by strleft(tpep_pickup_datetime, 7), strleft(tpep_dropoff_datetime, 7)
union
select null, null, count(*) as Total
from nycTaxi
where strleft(tpep_pickup_datetime, 7) not in ('2020-01','2020-02')
and strleft(tpep_dropoff_datetime,7) not in ('2020-01','2020-02')
) as t
order by t.Date_pickup, t.Date_dropoff
```

date_pickup	date_dropoff	total
2020-03	2020-03	33
2020-04	2020-04	6
2020-05	2020-05	11
2020-06	2020-06	1
2020-07	2020-07	8
2021-01	2021-01	3
tpep_pi	tpep_dr	1
TOTAL	TOTAL	192

- Show vendors record that is not in the timeframe

```
select strleft(tpep_pickup_datetime,7) as pickup_date,
strleft(tpep_dropoff_datetime,7) as dropoff_date, vendorid as vendor
from nycTaxi
where strleft(tpep_pickup_datetime,7) not in ('2020-01','2020-02') and
strleft(tpep_dropoff_datetime,7) not in ('2020-01','2020-02')
order by vendorid,pickup_date,dropoff_date
```

pickup_date	dropoff_date	vendor
2019-12	2019-12	2
2019-12	2019-12	2
2019-12	2019-12	2
2019-12	2019-12	2
2019-12	2019-12	2
2019-12	2019-12	2
2019-12	2019-12	2
2019-12	2019-12	2

According to the results, we figure out that records of vendor 2 are out of our timeframe.

- Identifying issues with the passenger count column

  We found that if there is no passenger the data will be inconsistent.

  With the script below we found 237885 records for 0 passengers and 114276 for passengers with null value.

```
select passenger_count, count(*) as cnt
from nycTaxi
group by passenger_count
order by passenger_count asc;
```

passenger_count	count
0	237885
null	114276

- Identifying issues with trip distance by checking the maximum distance and minimum distance in the column

```
select vendorid, min(trip_distance) as min_dis, max(trip_distance) as max_dis
from nycTaxi
group by vendorid
order by vendorid
```

vendorid	min_dis	max_dis
1	0.0	274.5
2	-30.62	210240.07

Based on shown results, the minimum distance = -30.62 which is a negative value does not make any sense. Also, the maximum distance = 210 240.07 is inaccurate and the 0 value will be ignored. We can consider them like wrong data.

Task 08: Loading data using Impala

- Populate nycTaxi_part_bkt table from nycTaxi while filtering erroneous records.

  We need to prepare our database:

  %impala

  -- Using database nyc in impala

  use nyc

  Respecting to the required conditions:

  Step 1: checking the condition.

  After checking we want to consider the taxi trip inter month when we populate the data. For this reason, when we populate table of January we pickup date in ('2019-12','2020-01') and dropoff date in ('2020-01'). With the code below, we will see that all the data are between Dec 31st and Jan 1st. We used the same pattern to populate the table of February.

  select
  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,congestion_surcharge
  from nycTaxi
  where (strleft(tpep_pickup_datetime, 7) in ('2019-12') and
  strleft(tpep_dropoff_datetime,7) in ('2020-01'))
  limit 5

tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	ratecodeid	store_and_fwd_flag	pulocationid	dolocationid	payment_type	fare_amount
2019-12-31 23:59:40	2020-01-01 00:09:06	2	2.19	1	N	231	158	1	9.5
2019-12-31 23:56:19	2020-01-01 00:15:43	1	3.74	1	N	162	158	1	15
2019-12-31 23:59:24	2020-01-01 00:00:47	6	0.07	1	N	263	263	2	3
2019-12-31 16:18:17	2020-01-01 15:37:27	1	1.22	1	N	142	48	2	12.5
2019-12-31 14:20:03	2020-01-01 13:46:02	6	4.43	1	N	231	142	1	19.5

We checked the loading condition.

We decided to take in consideration the lift pickup from JAN 31st to FEB 1st in our data.

```
%impala
select
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,
congestion_surcharge
from nycTaxi
where (strleft(tpep_pickup_datetime, 7) in ('2020-01') and  strleft(tpep_dropoff_datetime,7) in ('2020-02'))
limit 5
```

tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	ratecodeid	store_and_fwd_flag	pulocationid	dolocationid	payment_type	fare_amount	extra
2020-01-31 23:20:07	2020-02-01 00:32:03	1	8	1	N	142	112	1	42.5	3
2020-01-31 23:40:03	2020-02-01 00:16:30	1	5	1	N	114	106	1	24.5	3
2020-01-31 23:55:28	2020-02-01 00:01:51	2	1.2	1	N	229	170	1	6.5	3
2020-01-31 23:55:12	2020-02-01 00:00:13	3	0.7	1	N	234	90	2	5	3
2020-01-31 23:39:32	2020-02-01 00:06:12	1	4	1	N	163	232	1	18	3

Populate all the records belonging to January 2020 with respecting the conditions.

```
INSERT INTO TABLE nycTaxi_part_bkt
PARTITION(month = '01')
select
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,
congestion_surcharge
from nycTaxi
where (strleft(tpep_pickup_datetime, 7) in ('2019-12','2020-01') and
strleft(tpep_dropoff_datetime,7) in ('2020-01'))
and trip_distance >= 1
and trip_distance != (select max(trip_distance) from nyctaxi)
and passenger_count != 0
and total_amount > 0
```

Populating all the records belonging to Feb 2020

```
INSERT INTO TABLE nycTaxi_part_bkt PARTITION(month = '02')
Select
tpep_pickup_datetime,tpep_dropoff_datetime,passenger_count,trip_distance,RatecodeID,stor
e_and_fwd_flag,PULocationID,DOLocationID,payment_type,
fare_amount,extra,mta_tax,tip_amount,tolls_amount,improvement_surcharge,total_amount,
congestion_surcharge
from nycTaxi
where (strleft(tpep_pickup_datetime, 7) in ('2020-01','2020-02') and
strleft(tpep_dropoff_datetime,7) in ('2020-02'))
and trip_distance >= 1
and trip_distance != (select max(trip_distance) from nyctaxi)
and passenger_count != 0
and total_amount > 0
```

- Reporting the table stats (Rows count and size)

  compute stats nycTaxi_part_bkt

summary
Updated 2 partition(s) and 17 column(s).

show partitions nycTaxi_part_bkt

month	#Rows	#Files	Size	Bytes Cached	Cache Replication	Format	Incremen
01	4590384	1	396.37MB	NOT CACHED	NOT CACHED	TEXT	false
02	4509516	1	389.46MB	NOT CACHED	NOT CACHED	TEXT	false
Total	9099900	2	785.83MB	0B			

Task 08: Analysis II using Impala

- Comparing the overall average fare per trip for January and February

select strleft(tpep_pickup_datetime, 7) as pickup_date, round(avg(fare_amount),3) as avg_fare
from nyctaxi_part_bkt
group by strleft(tpep_pickup_datetime, 7)

date_dropoff	avg_fare
2020-01	14.674
2020-02	14.674

- Finding how many trips are made by each level of Passenger count

select passenger_count , count(*) as cnt
from nyctaxi_part_bkt
group by passenger_count

order by passenger_count

passenger_count	cnt
1	6623732
2	1402536
3	367976
4	179728
5	331369
6	194519
7	20
8	13

- Finding the most preferred mode of payment.
  The most popular one is Credit card.

select payment_type, count(*) as cnt
from nyctaxi_part_bkt
group by payment_type
order by count desc

payment_type	cnt
1	6951435
2	2109678
3	28116
4	10671

- The zone Percentage of the top 5 pull up location

*select a.pulocationid as location_id, b.zone, round(count(a.pulocationid) * 100.00 / sum(count(*)) over(), 2) as percentage*
from nyctaxi_part_bkt as a
left join taxilookup as b on (a.pulocationid=b.locationid)
group by a.pulocationid, b.zone
order by percentage desc
limit 5;

pulocationid	zone	zone_percentage
161	Midtown Center	4.36
132	JFK Airport	4.16
237	Upper East Side South	3.84
236	Upper East Side North	3.8
186	Penn Station/Madison Sq West	3.79

- The Zone percentage of the top 5 drop off location

```
select a.dolocationid as location_id, b.zone, round(count(a.dolocationid) * 100.00 /
sum(count(*)) over(), 2) as percentage
from nyctaxi_part_bkt as a
left join taxilookup  as b on (a.dolocationid=b.locationid)
group by a.dolocationid, b.zone
order by percentage desc
limit 5;
```

dolocationid	zone	zone_percentage
236	Upper East Side North	4.02
161	Midtown Center	3.59
237	Upper East Side South	3.42
162	Midtown East	2.91
170	Murray Hill	2.83

- The longest and shortest distance per zone.

```
select a.pulocationid as location_id, b.zone, min(a.trip_distance) as min_distance,
max(a.trip_distance) as max_distance
from nyctaxi_part_bkt as a
left join taxilookup  as b on (a.pulocationid=b.locationid)
group by a.pulocationid, b.zone
order by max(a.trip_distance) desc
limit 5;
```

location_id	zone	min_distance	max_distance
48	Clinton East	1	369.94
132	JFK Airport	1	262.88
140	Lenox Hill East	1	259.22
161	Midtown Center	1	211.7
208	Schuylerville/Edgewater Park	1	207.11

- Which month has a greater Average 'Speed'-January or February?

```
select strleft(tpep_dropoff_datetime, 7) as month, --tpep_dropoff_datetime,
tpep_pickup_datetime,
   round(sum(trip_distance) / sum((unix_timestamp(tpep_dropoff_datetime)-
unix_timestamp(tpep_pickup_datetime))/3600), 3) as avg_speed
from nyctaxi_part_bkt
group by strleft(tpep_dropoff_datetime, 7);
```

month	avg_speed
2020-01	11.615
2020-02	11.159

The average speed 11.615 is not reliable.
The greater average speed is in January. However, after looking into the data, we found out that there are a lot of incompatibility between the Time and Distance. Also, in some cases drop off was happening before Pickup.
All these inconsistencies will affect our results.