Green Taxi:

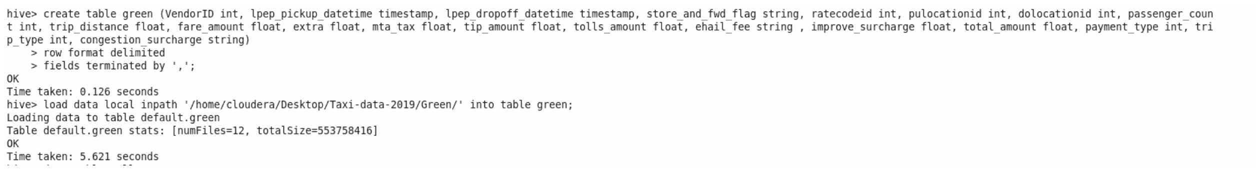
Loading data into Hive Tables for Green:

Creating tables for taxi data:

```
create table green (VendorID int, lpep_pickup_datetime timestamp,
lpep_dropoff_datetime timestamp, store_and_fwd_flag string, ratecodeid
int, pulocationid int, dolocationid int, passenger_count int,
trip_distance float, fare_amount float, extra float, mta_tax float,
tip_amount float, tolls_amount float, ehail_fee string ,
improve_surcharge float, total_amount float, payment_type int,
trip_type int, congestion_surcharge string)
row format delimited
fields terminated by ',';
```

Loading data from csv for all months of 2019:

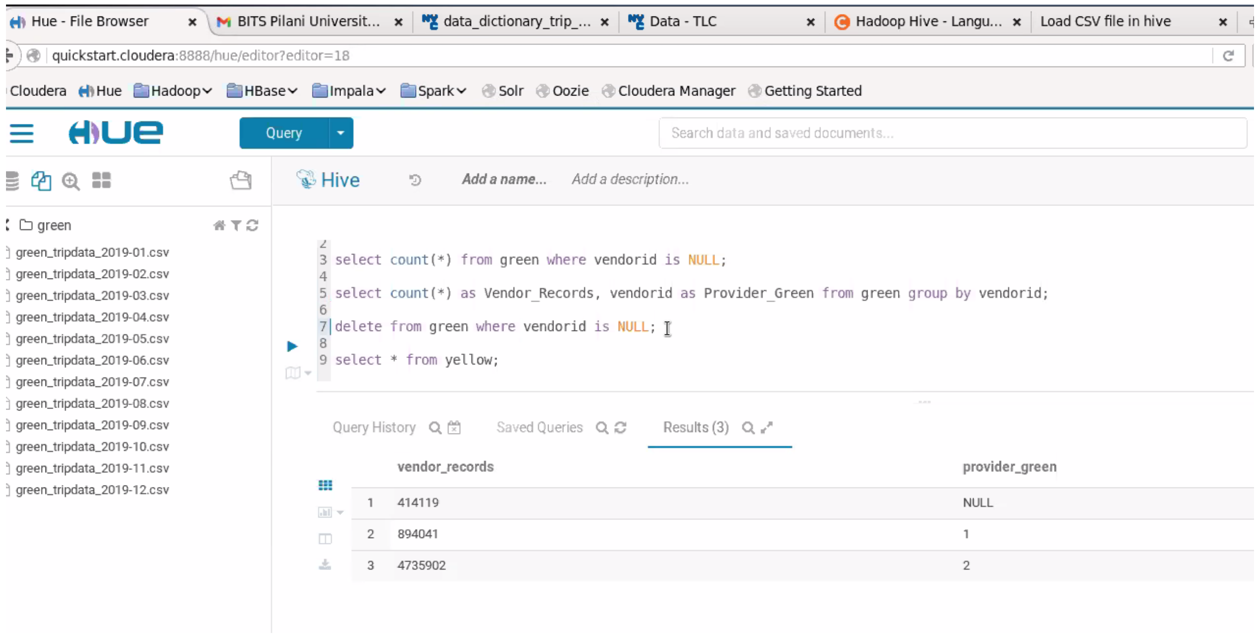
```
load data local inpath '/home/cloudera/Desktop/Taxi-data-2019/Green/' into table
green;
```



```
hive> create table green (VendorID int, lpep_pickup_datetime timestamp, lpep_dropoff_datetime timestamp, store_and_fwd_flag string, ratecodeid int, pulocationid int, dolocationid int, passenger_coun
t int, trip_distance float, fare_amount float, extra float, mta_tax float, tip_amount float, tolls_amount float, ehail_fee string , improve_surcharge float, total_amount float, payment_type int, tri
p_type int, congestion_surcharge string)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.126 seconds
hive> load data local inpath '/home/cloudera/Desktop/Taxi-data-2019/Green/' into table green;
Loading data to table default.green
Table default.green stats: [numFiles=12, totalSize=553758416]
OK
Time taken: 5.621 seconds
```

Total records provided by each vendor:

```
select count(*) as Vendor_Records, vendorid as Provider_Green from green group by
vendorid;
```



green_tripdata_2019-01.csv
green_tripdata_2019-02.csv
green_tripdata_2019-03.csv
green_tripdata_2019-04.csv
green_tripdata_2019-05.csv
green_tripdata_2019-06.csv
green_tripdata_2019-07.csv
green_tripdata_2019-08.csv
green_tripdata_2019-09.csv
green_tripdata_2019-10.csv
green_tripdata_2019-11.csv
green_tripdata_2019-12.csv

```
select count(*) from green where vendorid is NULL;

select count(*) as Vendor_Records, vendorid as Provider_Green from green group by vendorid;

delete from green where vendorid is NULL;

select * from yellow;
```

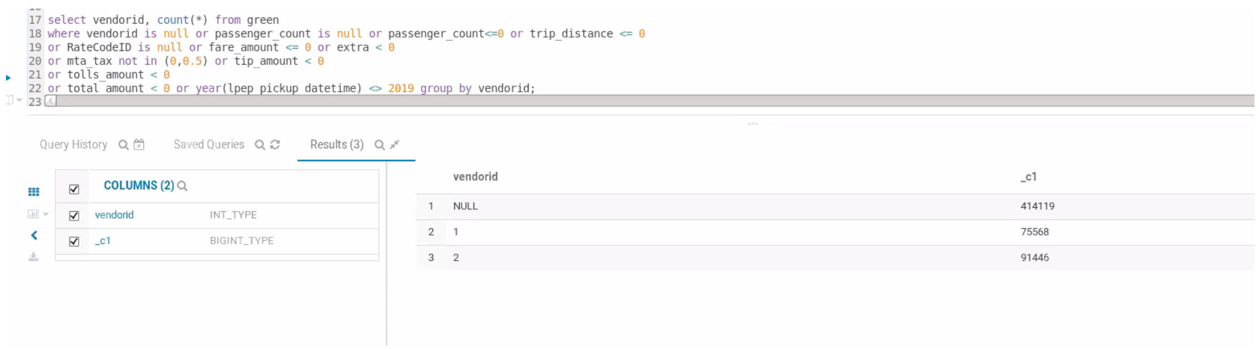
	vendor_records	provider_green
1	414119	NULL
2	894041	1
3	4735902	2

From above result, We see there are some null records for Vendor

1) Which vendor provides the most useful data?

Below query is for getting the vendor who is giving more inappropriate data(i.e nulls or negative values in CSV):

```
select vendorid, count(*) from green
where vendorid is null or passenger_count is null or passenger_count<=0 or
trip_distance <= 0
or RateCodeID is null or fare_amount <= 0 or extra < 0
or mta_tax not in (0,0.5) or tip_amount < 0
or tolls_amount < 0
or total_amount < 0 or year(lpep_pickup_datetime) <> 2019 group by vendorid;
```



```
select vendorid, count(*) from green
where vendorid is null or passenger_count is null or passenger_count<=0 or trip_distance <= 0
or RateCodeID is null or fare_amount <= 0 or extra < 0
or mta_tax not in (0,0.5) or tip_amount < 0
or tolls_amount < 0
or total_amount < 0 or year(lpep_pickup_datetime) <> 2019 group by vendorid;
```

vendorid	INT_TYPE
_c1	BIGINT_TYPE

	vendorid	_c1
1	NULL	414119
2	1	75568
3	2	91446

From above we can see there is no much difference but vendor 2= VeriFone Inc. is providing more useful data

Cleaning the data for further processing:

Now creating another table to load valid and appropriate data by removing nulls and negatives:

Creating table with partition by month:

```
create table if not exists green_partition (
       vendorid int,
       lpep_pickup_datetime string,
       lpep_dropoff_datetime string,
       store_and_fwd_flag string,
       ratecodeid int,
       pulocationid int,
       dolocationid int,
       passenger_count int,
       trip_distance double,
       fare_amount double,
       extra double,
       mta_tax double,
       tip_amount double,
       tolls_amount double,
       improvement_surcharge double,
       total_amount double,
       payment_type int
       ) partitioned by (mnth int);

SET hive.exec.max.dynamic.partitions=100000;
SET hive.exec.max.dynamic.partitions.pernode=100000;
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;
```

Loading data to Green_partition table:

```
insert overwrite table green_partition partition(mnth)
select vendorid, lpep_pickup_datetime, lpep_dropoff_datetime, store_and_fwd_flag,
ratecodeid,  pulocationid, dolocationid, passenger_count,
trip_distance,
fare_amount, extra, mta_tax,
tip_amount, tolls_amount, improve_surcharge, total_amount, payment_type,
month(lpep_pickup_datetime) as mnth
from green
where passenger_count>0 and trip_distance > 0
and RateCodeID is not null and fare_amount > 0 and extra >= 0
and mta_tax in (0,0.5) and tip_amount >= 0
and tolls_amount >= 0 and improve_surcharge = 0.30
and total_amount >= 0 and year(lpep_pickup_datetime) = 2019;
```

Counting the number of rows in which any column has NULL values

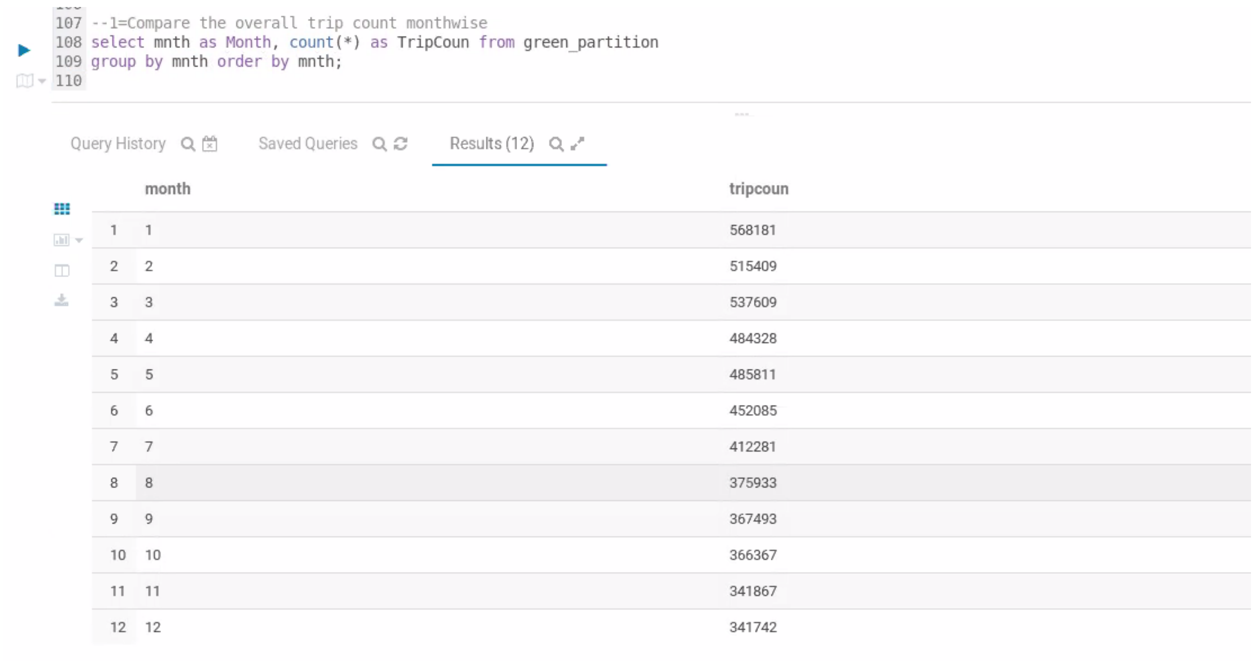
```
select count(*) as Total_NULL_records from green_partition
where vendorid is null
or tpep_pickup_datetime is null
or tpep_pickup_datetime is null
or passenger_count is null
or trip_distance is null
or ratecodeid is null
or store_and_fwd_flag is null
or pulocationid is null
or dolocationid is null
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 mnth is null;
```

There are no nulls in the data

2) Find the month wise trip count, average distance and average passenger count from the trips completed by yellow and green taxis in 2019. Summary visualizations will be preferred for better analysis.

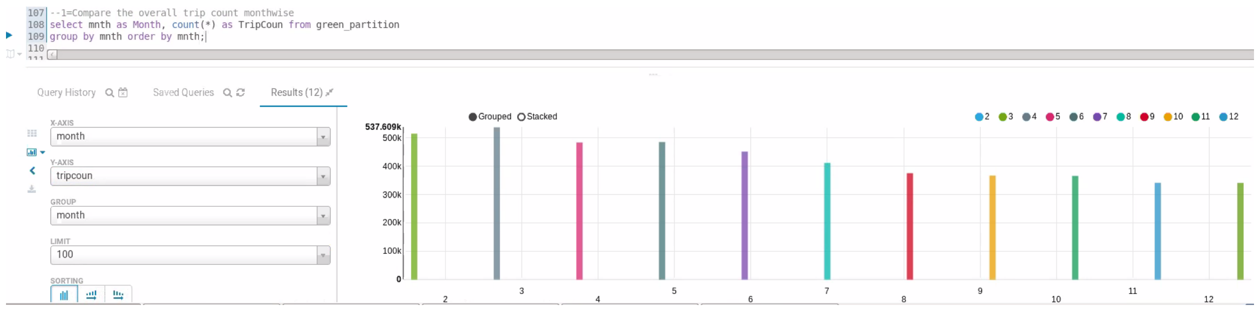
- Compare the overall trip count monthwise: Based on the results we can see most number of trips are done in early months of the year and are getting reduced by the end o the year

```
select mnth as Month, count(*) as TripCoun from green_partition
group by mnth order by mnth;
```



```
--1=Compare the overall trip count monthwise
select mnth as Month, count(*) as TripCoun from green_partition
group by mnth order by mnth;
```

	month	tripcoun
1	1	568181
2	2	515409
3	3	537609
4	4	484328
5	5	485811
6	6	452085
7	7	412281
8	8	375933
9	9	367493
10	10	366367
11	11	341867
12	12	341742



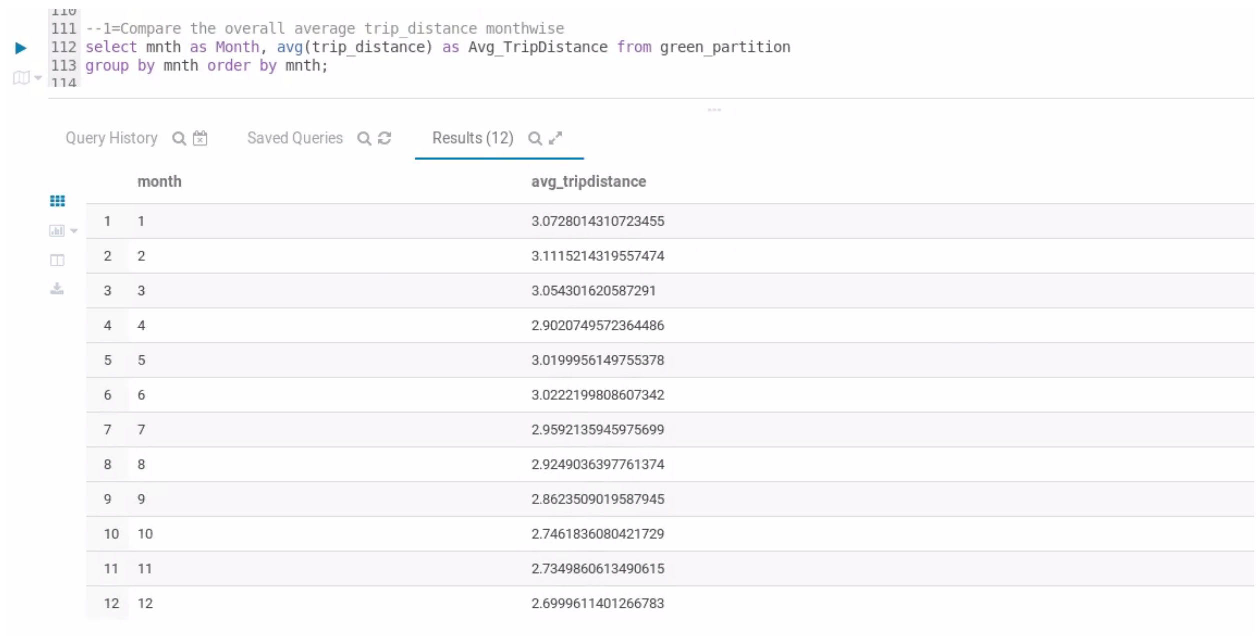
```
--1=Compare the overall trip count monthwise
select mnth as Month, count(*) as TripCoun from green_partition
group by mnth order by mnth;
```

Query History | Saved Queries | Results (12)

X-AXIS: month
Y-AXIS: tripcoun
GROUP: month
LIMIT: 100

- Compare the overall average trip_distance monthwise: Based on the results, we can see it is almost similar 2 - 3 miles

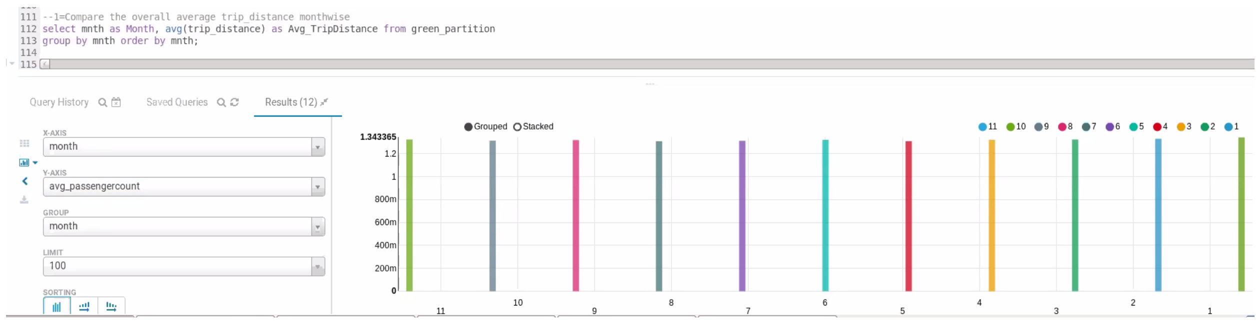
```
select mnth as Month, avg(trip_distance) as Avg_TripDistance from green_partition
group by mnth order by mnth;
```



```
--1=Compare the overall average trip_distance monthwise
select mnth as Month, avg(trip_distance) as Avg_TripDistance from green_partition
group by mnth order by mnth;
```

Query History | Saved Queries | Results (12)

	month	avg_tripdistance
1	1	3.0728014310723455
2	2	3.1115214319557474
3	3	3.054301620587291
4	4	2.9020749572364486
5	5	3.0199956149755378
6	6	3.0222199808607342
7	7	2.9592135945975699
8	8	2.9249036397761374
9	9	2.8623509019587945
10	10	2.7461836080421729
11	11	2.7349860613490615
12	12	2.6999611401266783



```
--1=Compare the overall average trip_distance monthwise
select mnth as Month, avg(trip_distance) as Avg_TripDistance from green_partition
group by mnth order by mnth;
```

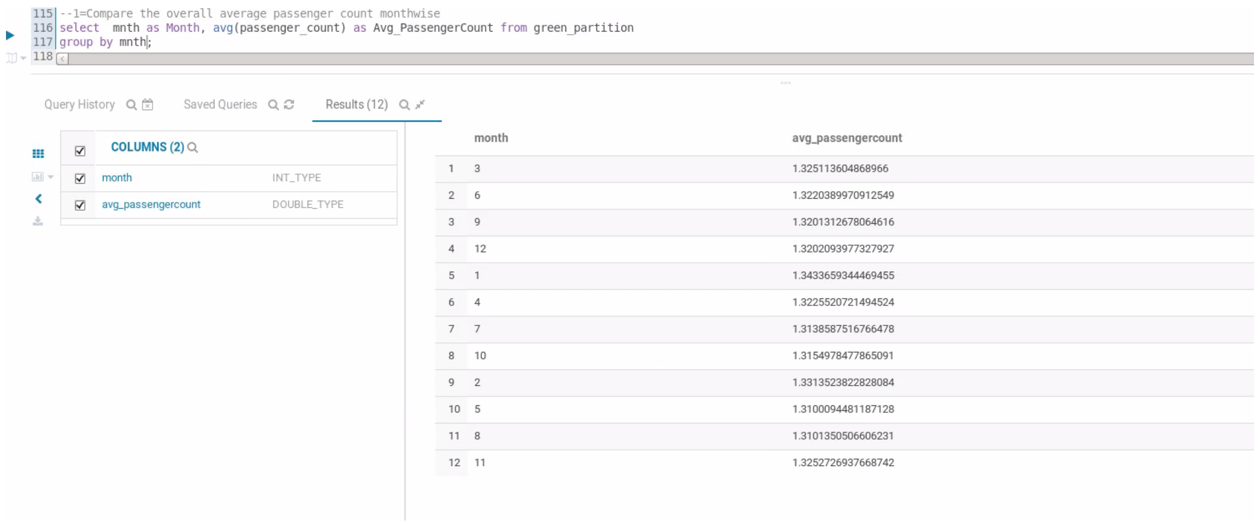
Query History | Saved Queries | Results (12)

X-AXIS: month
Y-AXIS: avg_passengercount
GROUP: month
LIMIT: 100

- Compare the overall average passenger count monthwise: From below we can see it is almost similar for all months

```
select  mnth as Month, avg(passenger_count) as Avg_PassengerCount from green_partition
```

```
group by mnth order by mnth desc;
```



```
--1=Compare the overall average passenger count monthwise
select  mnth as Month, avg(passenger_count) as Avg_PassengerCount from green_partition
group by mnth;
```

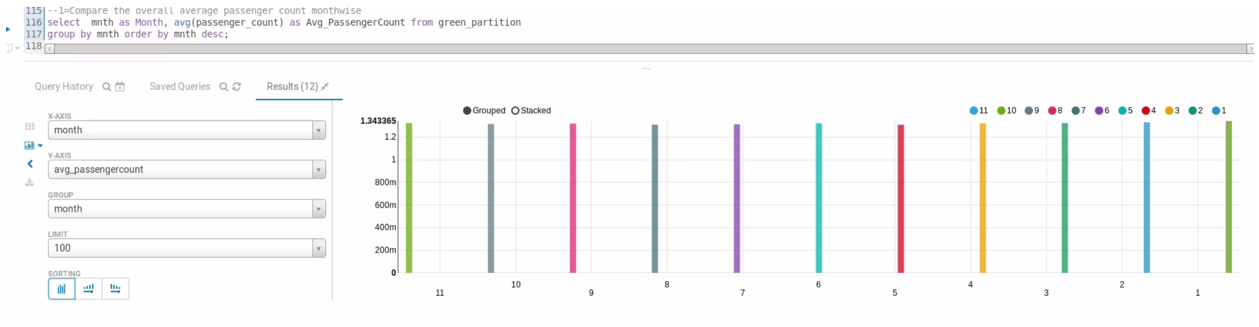
Query History | Saved Queries | Results (12)

COLUMNS (2)

month	INT_TYPE
avg_passengercount	DOUBLE_TYPE

	month	avg_passengercount
1	3	1.325113604868966
2	6	1.3220389970912549
3	9	1.3201312678064616
4	12	1.3202093977327927
5	1	1.3433659344469455
6	4	1.3225520721494524
7	7	1.3138587516766478
8	10	1.3154978477865091
9	2	1.3313523822828084
10	5	1.3100094481187128
11	8	1.3101350506606231
12	11	1.3252726937668742

```
Visual bar graph for average passenger count. It is almost same across months
```



```
--1=Compare the overall average passenger count monthwise
select  mnth as Month, avg(passenger_count) as Avg_PassengerCount from green_partition
group by mnth order by mnth desc;
```

Query History | Saved Queries | Results (12)

X-AXIS: month
Y-AXIS: avg_passengercount
GROUP: month
LIMIT: 100
SORTING

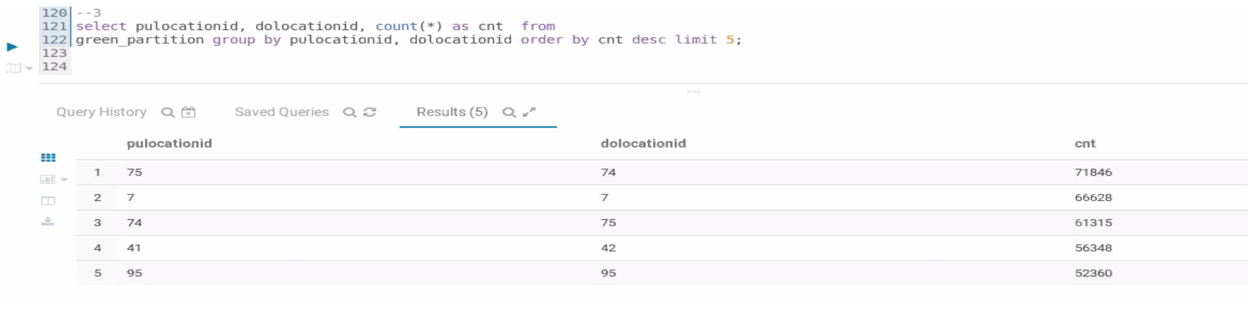
Loading Location Name data:

```
create table taxi_loc_data (LocationID int,Borough string,Zone string,service_zone
string)
row format delimited
fields terminated by ',';

load data local inpath '/home/cloudera/Desktop/Taxi-data-2019/taxi+_zone_lookup.csv'
into table taxi_loc_data;
```

3) Find out the five busiest routes served by the yellow and green taxis during 2019. The name of start and drop points to be provided.

```
select pulocationid, dolocationid, count(*) as cnt  from
green_partition group by pulocationid, dolocationid order by cnt desc limit 5;
```

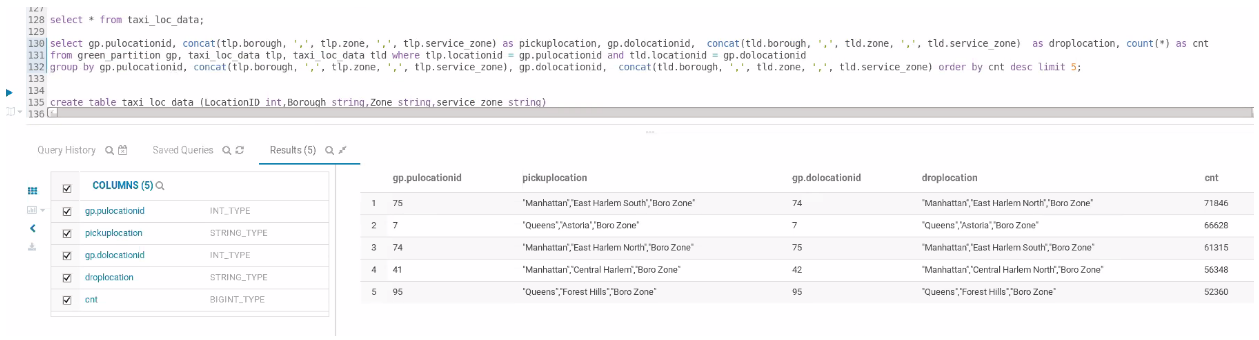


```
--3
select pulocationid, dolocationid, count(*) as cnt  from
green_partition group by pulocationid, dolocationid order by cnt desc limit 5;
```

	pulocationid	dolocationid	cnt
1	75	74	71846
2	7	7	66628
3	74	75	61315
4	41	42	56348
5	95	95	52360

Populating Location names:

```
select gp.pulocationid, concat(tlp.borough, ',', tlp.zone, ',', tlp.service_zone) as
pickuplocation, gp.dolocationid,  concat(tld.borough, ',', tld.zone, ',',
tld.service_zone)  as droplocation, count(*) as cnt
from green_partition gp, taxi_loc_data tlp, taxi_loc_data tld where tlp.locationid =
gp.pulocationid and tld.locationid = gp.dolocationid
group by gp.pulocationid, concat(tlp.borough, ',', tlp.zone, ',', tlp.service_zone),
gp.dolocationid,  concat(tld.borough, ',', tld.zone, ',', tld.service_zone) order by
cnt desc limit 5;
```



```
select * from taxi_loc_data;

select gp.pulocationid, concat(tlp.borough, ',', tlp.zone, ',', tlp.service_zone) as pickuplocation, gp.dolocationid,  concat(tld.borough, ',', tld.zone, ',', tld.service_zone)  as droplocation, count(*) as cnt
from green_partition gp, taxi_loc_data tlp, taxi_loc_data tld where tlp.locationid = gp.pulocationid and tld.locationid = gp.dolocationid
group by gp.pulocationid, concat(tlp.borough, ',', tlp.zone, ',', tlp.service_zone), gp.dolocationid,  concat(tld.borough, ',', tld.zone, ',', tld.service_zone) order by cnt desc limit 5;


create table taxi loc data (LocationID int,Borough string,Zone string,service zone string)
```

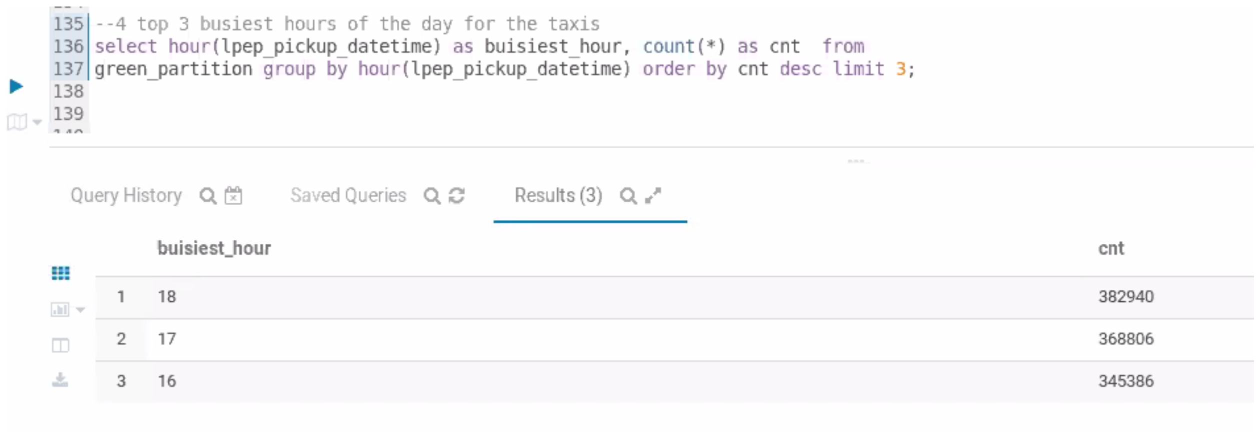
gp.pulocationid	INT_TYPE
pickuplocation	STRING_TYPE
gp.dolocationid	INT_TYPE
droplocation	STRING_TYPE
cnt	BIGINT_TYPE

	gp.pulocationid	pickuplocation	gp.dolocationid	droplocation	cnt
1	75	"Manhattan","East Harlem South","Boro Zone"	74	"Manhattan","East Harlem North","Boro Zone"	71846
2	7	"Queens","Astoria","Boro Zone"	7	"Queens","Astoria","Boro Zone"	66628
3	74	"Manhattan","East Harlem North","Boro Zone"	75	"Manhattan","East Harlem South","Boro Zone"	61315
4	41	"Manhattan","Central Harlem","Boro Zone"	42	"Manhattan","Central Harlem North","Boro Zone"	56348
5	95	"Queens","Forest Hills","Boro Zone"	95	"Queens","Forest Hills","Boro Zone"	52360

4) What are the top 3 busiest hours of the day for the taxis?

From Below we can see evening 4, 5, 6 PM are the busiest hours

```
select hour(lpep_pickup_datetime) as buisiest_hour, count(*) as cnt  from
green_partition group by hour(lpep_pickup_datetime) order by cnt desc limit 3;
```



```
--4 top 3 busiest hours of the day for the taxis
select hour(lpep_pickup_datetime) as buisiest_hour, count(*) as cnt  from
green_partition group by hour(lpep_pickup_datetime) order by cnt desc limit 3;
```

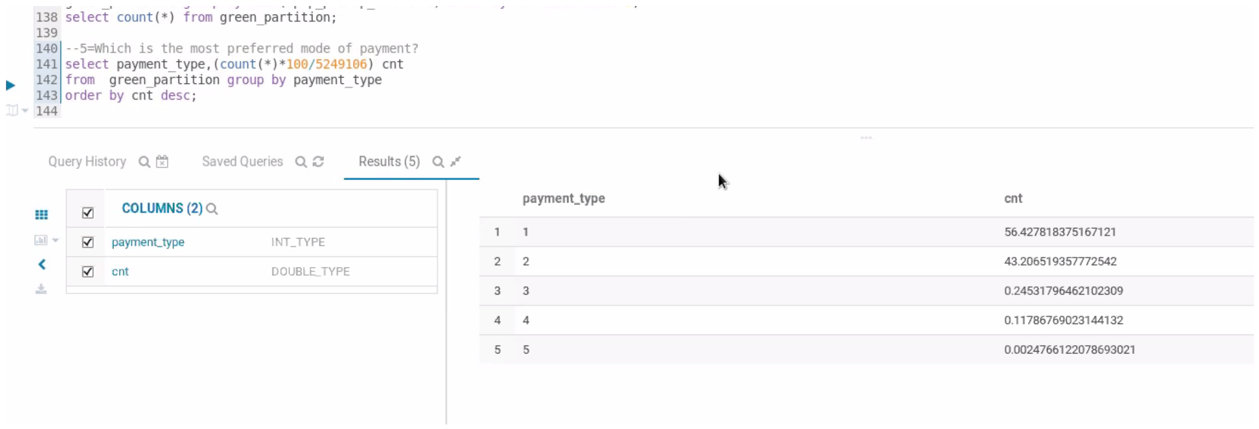
Query History   Saved Queries   Results (3)

	buisiest_hour	cnt
1	18	382940
2	17	368806
3	16	345386

5) What is the most preferred way of payment used by the passengers? What are the weekly trends observed for the methods of payments?

Mostly used payment mode is credit cards with 56%

```
select payment_type,(count(*)*100/5249106) cnt
from  green_partition group by payment_type
order by cnt desc;
```



```
select count(*) from green_partition;

--5=Which is the most preferred mode of payment?
select payment_type,(count(*)*100/5249106) cnt
from  green_partition group by payment_type
order by cnt desc;
```

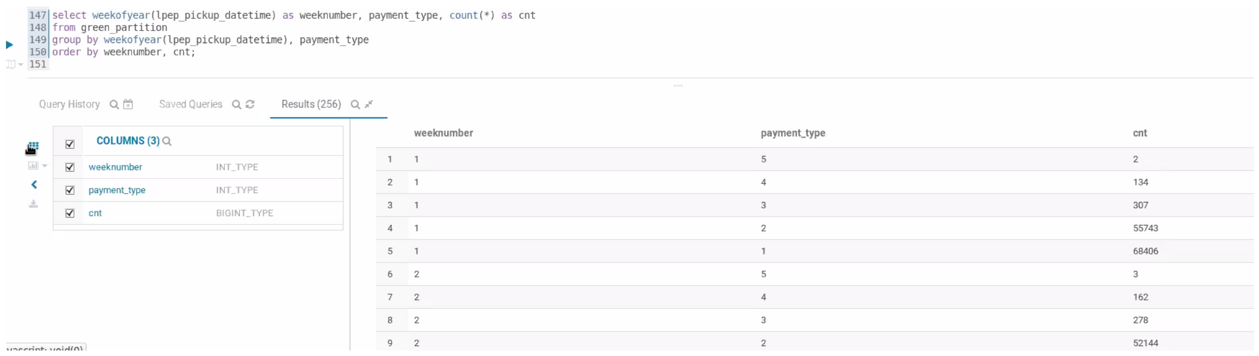
Query History   Saved Queries   Results (5)

COLUMNS (2)	
payment_type	INT_TYPE
cnt	DOUBLE_TYPE

	payment_type	cnt
1	1	56.427818375167121
2	2	43.206519357772542
3	3	0.24531796462102309
4	4	0.11786769023144132
5	5	0.0024766122078693021

Below is the Weekly trends followed by customers for payments:

```
select weekofyear(lpep_pickup_datetime) as weeknumber, payment_type, count(*) as cnt
from green_partition
group by weekofyear(lpep_pickup_datetime), payment_type
order by weeknumber, cnt;
```

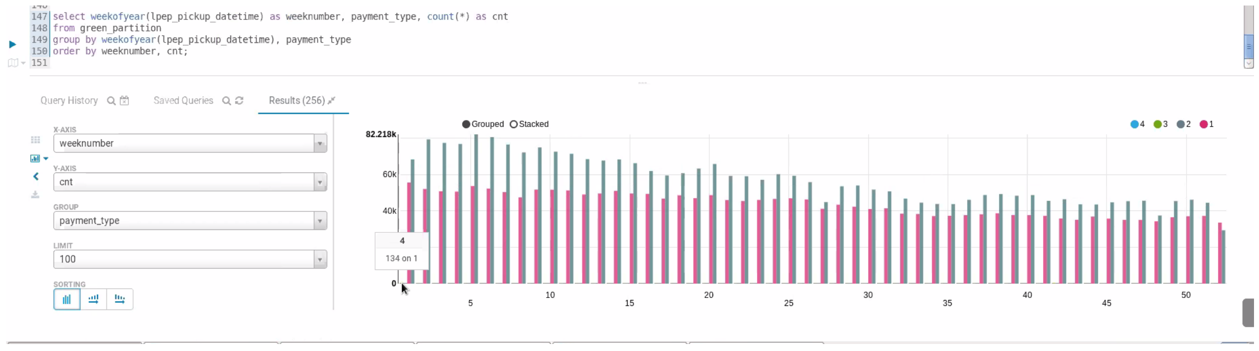


```
select weekofyear(lpep_pickup_datetime) as weeknumber, payment_type, count(*) as cnt
from green_partition
group by weekofyear(lpep_pickup_datetime), payment_type
order by weeknumber, cnt;
```

weeknumber	INT_TYPE
payment_type	INT_TYPE
cnt	BIGINT_TYPE

	weeknumber	payment_type	cnt
1	1	5	2
2	1	4	134
3	1	3	307
4	1	2	55743
5	1	1	68406
6	2	5	3
7	2	4	162
8	2	3	278
9	2	2	52144

View



```
select weekofyear(lpep_pickup_datetime) as weeknumber, payment_type, count(*) as cnt
from green_partition
group by weekofyear(lpep_pickup_datetime), payment_type
order by weeknumber, cnt;
```

X-AXIS: weeknumber
Y-AXIS: cnt
GROUP: payment_type
LIMIT: 100

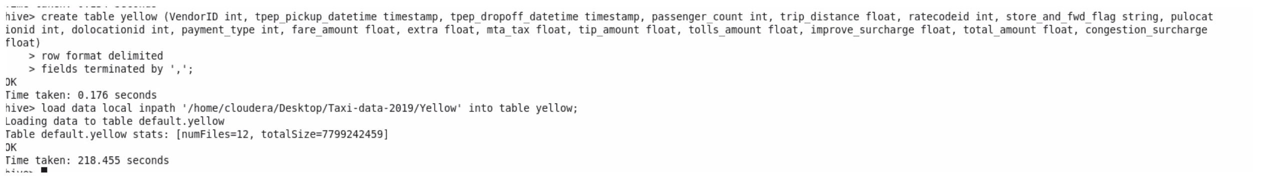
Yellow Taxi:

Creating table for yellow taxi data:

```
create table yellow (VendorID int, tpep_pickup_datetime timestamp,
tpep_dropoff_datetime timestamp, passenger_count int, trip_distance
float, ratecodeid int, store_and_fwd_flag string, pulocationid int,
dolocationid int, payment_type int, fare_amount float, extra float,
mta_tax float, tip_amount float, tolls_amount float, improve_surcharge
float, total_amount float, congestion_surcharge float)
row format delimited
fields terminated by ',';
```

Loading Data from csv for all months of 2019:

```
load data local inpath '/home/cloudera/Desktop/Taxi-data-2019/Yellow' into table
yellow;
```

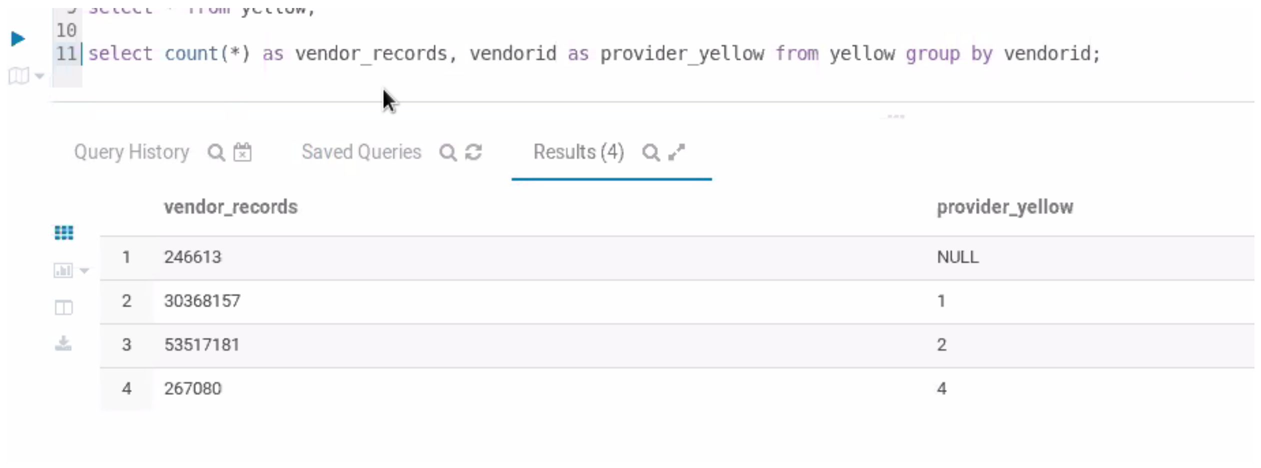


```
hive> create table yellow (VendorID int, tpep_pickup_datetime timestamp, tpep_dropoff_datetime timestamp, passenger_count int, trip_distance float, ratecodeid int, store_and_fwd_flag string, pulocat
ionid int, dolocationid int, payment_type int, fare_amount float, extra float, mta_tax float, tip_amount float, tolls_amount float, improve_surcharge float, total_amount float, congestion_surcharge
float)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.176 seconds
hive> load data local inpath '/home/cloudera/Desktop/Taxi-data-2019/Yellow' into table yellow;
Loading data to table default.yellow
Table default.yellow stats: [numFiles=12, totalSize=7799242459]
OK
Time taken: 218.455 seconds
```

Total records provided by each vendor:

`select count(*) as vendor_records, vendorid as provider_yellow from yellow group by vendorid;`

We see there are some null records and VendorId 4 for Vendor which are invalid



```
select count(*) as vendor_records, vendorid as provider_yellow from yellow group by vendorid;
```

Query History Saved Queries Results (4)

	vendor_records	provider_yellow
1	246613	NULL
2	30368157	1
3	53517181	2
4	267080	4

1) Which vendor provides the most useful data?

Below query is for getting the vendor who is giving more inappropriate data(i.e nulls or negative values in CSV):

```
select vendorid, count(*) from yellow
where vendorid is null or passenger_count is null or passenger_count<=0 or
trip_distance <= 0
or RateCodeID is null or fare_amount <= 0 or extra < 0
or mta_tax not in (0,0.5) or tip_amount < 0
or tolls_amount < 0
or total_amount < 0 or year(lpep_pickup_datetime) <> 2019 group by vendorid;
```

From above we can see there is no much difference but vendor 2= VeriFone Inc. is providing more useful data

Cleaning the data for further processing:

Now creating another table to load valid and appropriate data by removing nulls and negatives:

Creating table with partition by month:

```
create table if not exists yellow_partition (
       vendorid int,
       tpep_pickup_datetime string,
       tpep_dropoff_datetime string,
       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 (mnth int);

SET hive.exec.max.dynamic.partitions=100000;
SET hive.exec.max.dynamic.partitions.pernode=100000;
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;
```

Loading data to Yellow_partition table:

Not able to run the yellow partition insertion and queries. It is taking too much time and System is crashing. All Yellow taxi queries are updated above. Results will be similar to Green taxi details

```
insert overwrite table yellow_partition partition(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, improve_surcharge, total_amount, month(tpep_pickup_datetime)
as mnth
from yellow
where vendorid is not null and vendorid <> 4 and
passenger_count>0 and trip_distance > 0
and RateCodeID is not null and fare_amount > 0 and extra >= 0
and mta_tax in (0,0.5) and tip_amount >= 0
and tolls_amount >= 0 and improve_surcharge = 0.30
and total_amount >= 0 and year(tpep_pickup_datetime) = 2019;
```

Counting the number of rows in which any column has NULL values

```
select count(*) as Total_NULL_records from yellow_partition
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 mnth is null;
```

There are no nulls in the data

2) Find the month wise trip count, average distance and average passenger count from the trips completed by yellow and green taxis in 2019. Summary visualizations will be preferred for better analysis.

- Compare the overall trip count monthwise: Based on the results we can see most number of trips are done in early months of the year and are getting reduced by the end of the year

```
select mnth as Month, count(*) as TripCoun from yellow_partition
group by mnth order by mnth;
```

- Compare the overall average trip_distance monthwise: Based on the results, we can see it is almost similar 2 - 3 miles

```
select mnth as Month, avg(trip_distance) as Avg_TripDistance from yellow_partition
group by mnth order by mnth;
```

- Compare the overall average passenger count monthwise: From below we can see it is almost similar for all months

```
select  mnth as Month, avg(passenger_count) as Avg_PassengerCount from
yellow_partition
group by mnth order by mnth desc;
```

Visual bar graph for average passenger count. It is almost same across months

3) Find out the five busiest routes served by the yellow and green taxis during 2019. The name of start and drop points to be provided.

```
select pulocationid, dolocationid, count(*) as cnt  from
yellow_partition group by pulocationid, dolocationid order by cnt desc limit 5;
```

Populating Location names:

```
select gp.pulocationid, concat(tlp.borough, ',', tlp.zone, ',', tlp.service_zone) as
pickuplocation, gp.dolocationid,  concat(tld.borough, ',', tld.zone, ',',
tld.service_zone)  as droplocation, count(*) as cnt
from green_partition gp, taxi_loc_data tlp, taxi_loc_data tld where tlp.locationid =
gp.pulocationid and tld.locationid = gp.dolocationid
group by gp.pulocationid, concat(tlp.borough, ',', tlp.zone, ',', tlp.service_zone),
gp.dolocationid,  concat(tld.borough, ',', tld.zone, ',', tld.service_zone) order by
cnt desc limit 5;
```

4) What are the top 3 busiest hours of the day for the taxis?
From Below we can see evening 4, 5, 6 PM are the busiest hours

```
select hour(tpep_pickup_datetime) as buisiest_hour, count(*) as cnt  from
yellow_partition group by hour(tpep_pickup_datetime) order by cnt desc limit 3;
```

5) What is the most preferred way of payment used by the passengers? What are

the weekly trends observed for the methods of payments?

Mostly used payment mode is credit cards with 56%

```
select payment_type,(count(*)*100/5249106) cnt
from  yellow_partition group by payment_type
order by cnt desc;
```

Below is the Weekly trends followed by customers for payments:

```
select weekofyear(tpep_pickup_datetime) as weeknumber, payment_type, count(*) as cnt
from yellow_partition
group by weekofyear(tpep_pickup_datetime), payment_type
order by weeknumber, cnt;
```

Not able to run the yellow partition insertion and queries. It is taking too much time and System is crashing. All Yellow taxi queries are updated above. Results will bi similar to Green taxi details