1. Brief description of the Data

- The New York City Taxi and TLC (Taxi and Limousine Commission) is the agency responsible for issuing license and regulating rules for the taxi cabs, hire-vehicles, commuter vans etc. in New York City. TLC issues license to drivers after scrutinizing background of driver, safe driving records and on completion of 24 hours driving training.
- Dataset contains total of 60,44,050 rows and 20 columns. The details are as below.
- This data dictionary describes SHL trip data. For a dictionary describing Green taxi data, or a map of the TLC Taxi Zones.

Sl. No.	Field Name	Description	Datatype
1	VENDORID	A code indicating the LPEP providerthat provided the record. **1**= Creative Mobile Technologies, LLC; **2**= VeriFone Inc.	NUMBER
2	LPEP_PICKUP_DATETIME	The date and time when the meter was engaged.	DATE
3	LPEP_DROPOFF_DATETIME	The date and time when the meter was disengaged.	DATE
4	STORE_AND_FWD_FLAG	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server. Y= store and forward trip N= not a store and forward trip	CHAR
5	RATECODEID	The final rate code in effect at the endof the trip. **1**= Standard rate **2**=JFK **3**=Newark **4**=Nassau or Westchester **5**=Negotiated fare **6**=Group ride	NUMBER
6	PULOCATIONID	TLC Taxi Zone in which the taximeter was engaged	NUMBER
7	DOLOCATIONID	TLC Taxi Zone in which the taximeter was disengaged	NUMBER
8	PASSENGER_COUNT	The number of passengers in the vehicle. This is a driver-entered value.	NUMBER
9	TRIP_DISTANCE	The elapsed trip distance in miles reported by the taximeter.	FLOAT
10	FARE_AMOUNT	The time-and-distance fare calculated by the meter.	FLOAT
11	EXTRA	Miscellaneous extras and surcharges. Currently, this only includes the $0.50and $1 rush hour and overnight charges.	FLOAT

VARUN NAGENDRA
ORACLE ASSIGNMENT

Sl. No.	Field Name	Description	Datatype
12	MTA_TAX	$0.50 MTA tax that is automatically triggered based on the metered rate in use.	FLOAT
13	TIP_AMOUNT	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.	FLOAT
14	TOLLS_AMOUNT	The total amount charged to passengers. Does not include cash tips	FLOAT
15	EHAIL_FEE		FLOAT
16	IMPROVEMENT_SURCHARGE	$0.30 improvement surcharge assessed on hailed trips at the flagdrop. The improvement surcharge began being levied in 2015.	FLOAT
17	TOTAL_AMOUNT	The total amount charged to passengers. Does not include cash tips.	FLOAT
18	PAYMENT_TYPE	A numeric code signifying how the passenger paid for the trip. **1**= Credit card **2**= Cash **3**= No charge **4**= Dispute **5**= Unknown **6**= Voided trip	NUMBER
19	TRIP_TYPE	A code indicating whether the trip wasa street-hail or a dispatch that is automatically assigned based on the metered rate in use but can be alteredby the driver. **1**= Street-hail **2**=Dispatch	NUMBER
20	CONGESTION_SURCHARGE		FLOAT

2. Any anomalies you identified in the provided dataset and a brief description of how identified them and why do you think they are anomalies.

- Yes the anomalies are present in the provided data set.
- Anomalies identification was done by comparing the data set to the data dictionary.
- The anomalies are present in the given data set are NULL Values and the extra added values to the columns in which are not present in the data dictionary.

VARUN NAGENDRA
ORACLE ASSIGNMENT

- There are having 323382 values are missing in the vendorid.

select vendorid,count(*) from greentaxi group by vendorid;

```
SQL> select vendorid,count(*) from greentaxi group by vendorid;

  VENDORID   COUNT(*)
---------- ----------
            323382
         1     850947
         2    4519426

SQL>
```

- The NULL values are identified in the Store_and_fwd_flag;

select Store_and_fwd_flag,count(*) from greentaxi group by Store_and_fwd_flag;

```
SQL> select Store_and_fwd_flag,count(*) from greentaxi group by Store_and_fwd_flag;

S   COUNT(*)
- ----------
      323382
Y      13677
N    5356696

SQL>
```

- In RateCodeID having only 6 types provided from the data dictionary but new one is 99 is present of 51 in the data set and having the NULL values are 323382 as well.

select RateCodeID,count(*) from greentaxi group by RateCodeID;

```
SQL> select RateCodeID,count(*) from greentaxi group by RateCodeID;

RATECODEID   COUNT(*)
---------- ----------
            323382
         5     248101
        99         51
         6         54
         3       2483
         1    5105245
         2      10464
         4       3975

8 rows selected.
```

- According to the Data Dictionary the payment type is having 6 types and having 323382 NULL values in the data.

select Payment_type,count(*) from greentaxi group by Payment_type;

```
SQL> select Payment_type,count(*) from greentaxi group by Payment_type;

PAYMENT_TYPE   COUNT(*)
------------ ----------
             323382
           5       192
           3     28143
           1   3072398
           2   2257654
           4     11986

6 rows selected.
```

- The Trip_type is having 323739 NULL values in the provided data set.

Select Trip_type ,count(*) from greentaxi group by Trip_type;

```
SQL> select Trip_type,count(*) from greentaxi group by Trip_type;

TRIP_TYPE   COUNT(*)
--------- ----------
            323739
        1    5125740
        2     244276
```

- 0 and NULL are the anomalies in the passenger count

select passenger_count,count(*) from greentaxi group by passenger_count;

```
SQL> select passenger_count,count(*) from greentaxi group by passenger_count;

PASSENGER_COUNT   COUNT(*)
--------------- ----------
              1    4620855
                    323382
              5     156505
              8        119
              7         83
              9         34
              6      84796
              2     396206
              4      27456
              3      73150
              0      11169

11 rows selected.
```

- In Ehail_Fee there are 5693401 NULL values in dataset.

select Ehail_fee,count(*) from greentaxi group by Ehail_fee;

```
SQL> select Ehail_fee,count(*) from greentaxi group by Ehail_fee;

 EHAIL_FEE   COUNT(*)
--------- ----------
            5693401
      1.95          3
         0        351
```

- 869762 Null Values are present in the congestion_surcharge.

select congestion_surcharge,count(*) from greentaxi group by congestion_surcharge;

```
SQL> select congestion_surcharge,count(*) from greentaxi group by congestion_surcharge;

CONGESTION_SURCHARGE   COUNT(*)
-------------------- ----------
                          869762
                -2.5          1
                 .75         68
               -2.75         69
                   2          2
               -.75           1
                  .3          1
                 2.5       4574
                   0    4126071
                2.75     693206

10 rows selected.
```

VARUN NAGENDRA
ORACLE ASSIGNMENT

Create the table structure with appropriate data types.

```
create table greentaxi
(
  vendorid number
, lpep_pickup_datetime date
, lpep_dropoff_datetime date
, store_and_fwd_flag char(1byte)
, ratecodeid number(*, 0)
, pulocationid number (*, 0)
, dolocationid number(*, 0)
, passenger_count number(*, 0)
, trip_distance float(126)
, fare_amount float(126)
, extra float (126)
, mta_tax float (126)
, tip_amount float (126)
, tolls_amount float (126)
, ehail_fee float (126)
, improvement_surcharge float (126)
, total_amount float (126)
, payment_type number(*, 0)
, trip_type number(*, 0)
, congestion_surcharge float (126)
);
```

VARUN NAGENDRA
ORACLE ASSIGNMENT

OUTPUT

```
Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> CREATE TABLE green_taxi
  2  (
  3    VENDORID NUMBER(*, 0)
  4  , LPEP_PICKUP_DATETIME DATE
  5  , LPEP_DROPOFF_DATETIME DATE
  6  , STORE_AND_FWD_FLAG CHAR(1 BYTE)
  7  , RATECODEID NUMBER(*, 0)
  8  , PULOCATIONID NUMBER(*, 0)
  9  , DOLOCATIONID NUMBER(*, 0)
 10  , PASSENGER_COUNT NUMBER(*, 0)
 11  , TRIP_DISTANCE FLOAT(126)
 12  , FARE_AMOUNT FLOAT(126)
 13  , EXTRA FLOAT(126)
 14  , MTA_TAX FLOAT(126)
 15  , TIP_AMOUNT FLOAT(126)
 16  , TOLLS_AMOUNT FLOAT(126)
 17  , EHAIL_FEE FLOAT(126)
 18  , IMPROVEMENT_SURCHARGE FLOAT(126)
 19  , TOTAL_AMOUNT FLOAT(126)
 20  , PAYMENT_TYPE NUMBER(*, 0)
 21  , TRIP_TYPE NUMBER(*, 0)
 22  , CONGESTION_SURCHARGE FLOAT(126)
 23  );

Table created.
```

Use Taxi Zone Lookup table from the city portal site to for drop and pickup location names

create table location(locid number ,borough varchar2(20) ,zone varchar2(20) ,service_zone varchar2(20));

OUTPUT

```
SQL> create table location(locid number,borough varchar2(20),zone varchar2(20),service_zone varchar2(20));

Table created.

SQL>
```

Green Taxi Payment Table

create table payment(id number,modeof_payment VARCHAR2(40));

insert into payment VALUES(1,'Credit card');
insert into payment VALUES(2,'Cash');
insert into payment VALUES(3,'No charge');
insert into payment VALUES(4,'Dispute');
insert into payment VALUES(5,'Unknown');
insert into payment VALUES(6,'Voided trip');

```
SQL> create table payment(id number,modeof_payment VARCHAR2(40));

Table created.

SQL> insert into payment VALUES(1,'Credit card');

1 row created.

SQL> insert into payment VALUES(2,'Cash');

1 row created.

SQL> insert into payment VALUES(3,'No charge');

1 row created.

SQL> insert into payment VALUES(4,'Dispute');

1 row created.

SQL> insert into payment VALUES(5,'Unknown');

1 row created.

SQL> insert into payment VALUES(6,'Voided trip');

1 row created.

SQL> select * from payment;

        ID MODEOF_PAYMENT
---------- ----------------------------------------
         1 Credit card
         2 Cash
         3 No charge
         4 Dispute
         5 Unknown
         6 Voided trip

6 rows selected.
```

3. Queries and Results

A. Find the month wise trip count, average distance and average passenger count from the trips completed by green taxis in 2019.

```
select to_char(lpep_dropoff_datetime,'MM'),

count (*) as  trip_count,

avg(trip_distance) as average_distance,

avg(passenger_count) as avgerage_passenger_count

from green_taxi

group by to_char(lpep_dropoff_datetime,'MM')

order by to_char(lpep_dropoff_datetime,'MM');
```

OUTPUT

VARUN NAGENDRA
ORACLE ASSIGNMENT

```
SQL> select to_char(lpep_dropoff_datetime,'MM') ,
  2  count(*) as  trip_count,
  3  avg(trip_distance) as average_distance,
  4  avg(passenger_count) as avgerage_passenger_count
  5  from greentaxi
  6  group by to_char(lpep_dropoff_datetime,'MM')
  7  order by to_char(lpep_dropoff_datetime,'MM');

TO TRIP_COUNT AVERAGE_DISTANCE AVGERAGE_PASSENGER_COUNT
-- ---------- ---------------- ----------------------
01     630744       3.43898819              1.31797528
02     575680        3.5054705               1.3065262
03     601088       3.45558214              1.30080787
04     514393       2.99751709              1.31340434
05     504813       2.97868839              1.30683243
06     471109       2.97809225              1.31800709
07     470731       3.32007565               1.3081531
08     449567       3.48231147              1.30269421
09     449237       3.52134192              1.31196647
10     476320       3.54281701              1.30833603
11     449505       2.89318937               1.3155307

TO TRIP_COUNT AVERAGE_DISTANCE AVGERAGE_PASSENGER_COUNT
-- ---------- ---------------- ----------------------
12     100568        2.5882917              1.31647823

12 rows selected.

SQL>
```

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

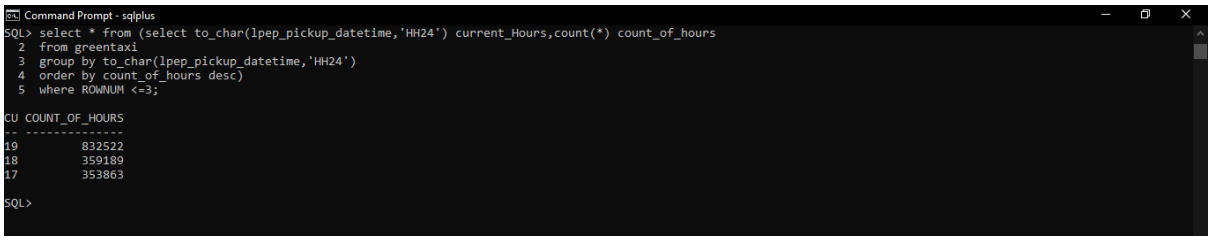
```
select * from
(
select a.zone as pickup_loc ,b.zone as dropoff_loc,
count(*) as no_of_trips from greentaxi g
join loc a on a.locationid = g.pulocationid
join loc b on b.locationid = g.dolocationid
group by a.zone,b.zone
order by no_of_trips desc
)
where rownum<=5;
```

```
SQL> select * from
  2  (
  3  select a.zone as pickup_loc,b.zone as dropoff_loc,
  4  count(*) as no_of_trips from greentaxi g
  5  join loc a on a.locationid = g.pulocationid
  6  join loc b on b.locationid = g.dolocationid
  7  group by a.zone,b.zone
  8  order by no_of_trips desc
  9  )
 10  where rownum<=5;

PICKUP_LOC
--------------------------------------------------
DROPOFF_LOC                                        NO_OF_TRIPS
-------------------------------------------------- -----------
East Harlem South
East Harlem North                                        72804

Astoria
Astoria                                                  69403

East Harlem North
East Harlem South                                        62616


PICKUP_LOC
--------------------------------------------------
DROPOFF_LOC                                        NO_OF_TRIPS
-------------------------------------------------- -----------
Central Harlem
Central Harlem North                                     57520

Forest Hills
Forest Hills                                             54712


SQL>
```

C. What are the top 3 busiest hours of the day for the taxis?

```
select * from (select to_char(lpep_pickup_datetime,'HH24') current_Hours,count(*)
count_of_hours

from greentaxi

group by to_char(lpep_pickup_datetime,'HH24')

order by count_of_hours desc

where ROWNUM<=3;
```

OUTPUT



D. What is the most preferred way of payment used by the passengers?

```
select * from
(
select p.modeof_payment as payment_type,count(*) as cnt
from greentaxi g
join payment p
on g.payment_type = p.id
group by p.modeof_payment
order by cnt desc
)
where rownum = 1;
```

OUTPUT

E. Write a PL/SQL block to read through each record and update ehail_fee to 0.5 (capture the time taken for execution)

```
set timing on;
begin
update greentaxi set ehail_fee= 0.5 ;
end;
/
```

F. Write a normal update statement to update ehail_fee to 0.75

```
set timing on;
update green_taxi set ehail_fee = 0.75;
```

G. Identify the time taken by e and f are provide your analysis on why each step took more/less time compared to other

The procedure **(e)** took **11:12.43** minutes to update the ehail_fee to 0.5 whereas update statement **(f)** completed in **53.13** seconds for updating the same ehail_fee to 0.75.

Procedure takes the row by row updating the records, and the time taken for the execution PL/SQL statement is more than the normal SQL statement.