DELIVERABLE-2

Data Understanding

Understanding the nature of the data

As per the dataset obtained, it consists of 13,69,765 rows and 19 columns in total.

PULocationID – It consists of the pickup locations of the customers.

DOLocationID – It consists of the drop-off locations.

Passenger count – It consists of the passenger count in a ride.

Trip distance – It consists of the total distance of a trip

Alongside there are several other columns such as RatecodeID, store_and_fwd_flag,vendorID, payment type, fare_amount, taxes, tolls, and some other columns.

In [3]:
```
#Looking through the data
data.head()
```

Out[3]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payme
0	1.0	2021-01-01 00:30:10	2021-01-01 00:36:12	1.0	2.10	1.0	N	142	43	
1	1.0	2021-01-01 00:51:20	2021-01-01 00:52:19	1.0	0.20	1.0	N	238	151	
2	1.0	2021-01-01 00:43:30	2021-01-01 01:11:06	1.0	14.70	1.0	N	132	165	
3	1.0	2021-01-01 00:15:48	2021-01-01 00:31:01	0.0	10.60	1.0	N	138	132	
4	2.0	2021-01-01 00:31:49	2021-01-01 00:48:21	1.0	4.94	1.0	N	68	33	

In [4]:
```
#describing the statistical information of the data
data.describe()
```

Out[4]:

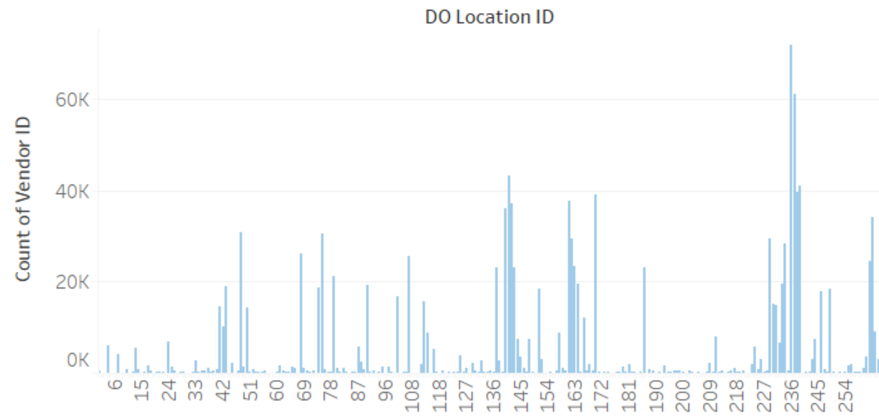
	VendorID	passenger_count	trip_distance	RatecodeID	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	
count	1.271413e+06	1.271413e+06	1.369765e+06	1.271413e+06	1.369765e+06	1.369765e+06	1.271413e+06	1.369765e+06	1.369765e+06	1.369765e+06	1
mean	1.676925e+00	1.411508e+00	4.631982e+00	1.035081e+00	1.652472e+02	1.614956e+02	1.280521e+00	1.209662e+01	9.705085e-01	4.930411e-01	1
std	4.676513e-01	1.059833e+00	3.939042e+02	5.994840e-01	6.783849e+01	7.210800e+01	4.916921e-01	1.291338e+01	1.231256e+00	7.632070e-02	2
min	1.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	-4.900000e+02	-5.500000e+00	-5.000000e-01	-1
25%	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.240000e+02	1.070000e+02	1.000000e+00	6.000000e+00	0.000000e+00	5.000000e-01	0
50%	2.000000e+00	1.000000e+00	1.700000e+00	1.000000e+00	1.620000e+02	1.620000e+02	1.000000e+00	8.500000e+00	0.000000e+00	5.000000e-01	1
75%	2.000000e+00	1.000000e+00	3.020000e+00	1.000000e+00	2.360000e+02	2.360000e+02	2.000000e+00	1.350000e+01	2.500000e+00	5.000000e-01	2
max	2.000000e+00	8.000000e+00	2.631633e+05	9.900000e+01	2.650000e+02	2.650000e+02	4.000000e+00	6.960500e+03	8.250000e+00	5.000000e-01	1

```
In [5]: #getting info of all the columns
        data.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 1369765 entries, 0 to 1369764
        Data columns (total 19 columns):
         #   Column                 Non-Null Count    Dtype
        ---  ------                 --------------    -----
         0   VendorID               1271413 non-null  float64
         1   tpep_pickup_datetime   1369765 non-null  object
         2   tpep_dropoff_datetime  1369765 non-null  object
         3   passenger_count        1271413 non-null  float64
         4   trip_distance          1369765 non-null  float64
         5   RatecodeID             1271413 non-null  float64
         6   store_and_fwd_flag     1271413 non-null  object
         7   PULocationID           1369765 non-null  int64
         8   DOLocationID           1369765 non-null  int64
         9   payment_type           1271413 non-null  float64
         10  fare_amount            1369765 non-null  float64
         11  extra                  1369765 non-null  float64
         12  mta_tax                1369765 non-null  float64
         13  tip_amount             1369765 non-null  float64
         14  tolls_amount           1369765 non-null  float64
         15  improvement_surcharge  1369765 non-null  float64
         16  total_amount           1369765 non-null  float64
         17  congestion_surcharge   1369765 non-null  float64
         18  ID                     1369765 non-null  int64
        dtypes: float64(13), int64(3), object(3)
        memory usage: 198.6+ MB

In [6]: #checking number of rows and columns
        data.shape

Out[6]: (1369765, 19)
```

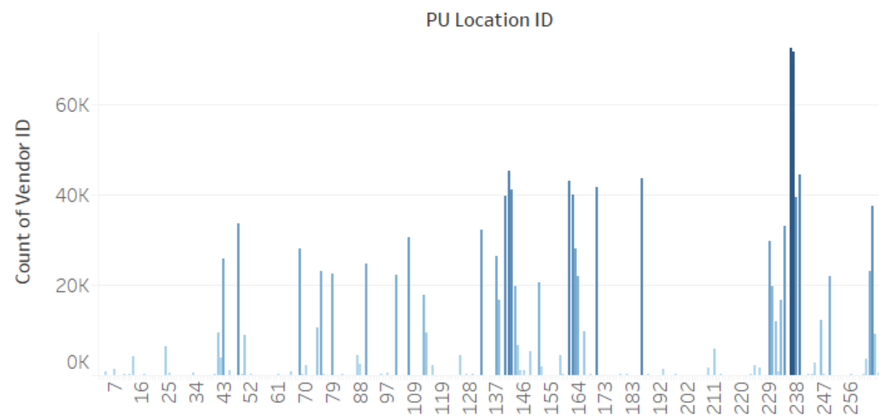
Exploratory data analysis:

- From the datasets, we can analyse that there are 258 unique pickup locations and 260 drop locations.
- On understanding, we can explain that Friday has the highest number of rides, Next comes Thursday and Wednesday, and the least number of rides are on Sunday.
- Amongst the rides, the most number of passenger count for rides are 1 and followed by 2.
- Amongst the drop-off locations, these are the descending order of locations with the highest number of drop-off 236,237,239,238,141
- Amongst the pick-up locations, these are in descending order of locations with the highest number of pick-ups 236, and 237,141,239,186.

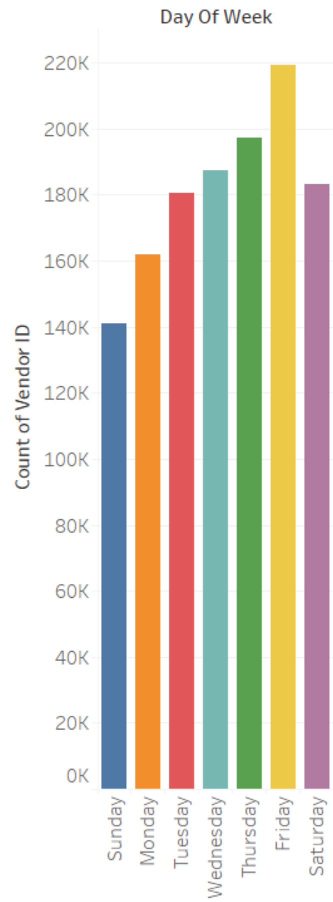
Based on Drop Off Locations



Based on pickup Location



Rides vs Days



Data Preparation

In this stage, we prepare our data for analysis. Here, we employ several data cleaning techniques.

Data Cleaning

Checking how many rows with atleast one column having Null value

In [7]:
```
data.isnull().sum() * 100 / len(data)
```

Out[7]:
```
VendorID                 7.18021
tpep_pickup_datetime     0.00000
tpep_dropoff_datetime    0.00000
passenger_count          7.18021
trip_distance            0.00000
RatecodeID               7.18021
store_and_fwd_flag       7.18021
PULocationID             0.00000
DOLocationID             0.00000
payment_type             7.18021
fare_amount              0.00000
extra                    0.00000
mta_tax                  0.00000
tip_amount               0.00000
tolls_amount             0.00000
improvement_surcharge    0.00000
total_amount             0.00000
congestion_surcharge     0.00000
ID                       0.00000
dtype: float64
```

Our data has around 7.1% rows with null values in columns 'RatecodeID', 'store_and_fwd_flag', 'congestion_surcharge', 'passenger_count'. As the amount of rows with Null values are less as compared to total rows we have, we will drop all rows having atleast one column with Null value.

Imputation of the rows containing null value

In [8]:
```
# Performing imputation with mode
data['VendorID'] = data['VendorID'].fillna(data['VendorID'].mode()[0])
data['payment_type'] = data['payment_type'].fillna(data['payment_type'].mode()[0])
data['store_and_fwd_flag'] = data['store_and_fwd_flag'].fillna(data['store_and_fwd_flag'].mode()[0])
data['passenger_count'] = data['passenger_count'].fillna(data['passenger_count'].mode()[0])
data['RatecodeID'] = data['RatecodeID'].fillna(data['RatecodeID'].mode()[0])

data.isnull().sum() * 100 / len(data)
```

Out[8]:
```
VendorID                 0.0
tpep_pickup_datetime     0.0
tpep_dropoff_datetime    0.0
passenger_count          0.0
trip_distance            0.0
RatecodeID               0.0
store_and_fwd_flag       0.0
PULocationID             0.0
DOLocationID             0.0
payment_type             0.0
fare_amount              0.0
extra                    0.0
mta_tax                  0.0
tip_amount               0.0
tolls_amount             0.0
improvement_surcharge    0.0
total_amount             0.0
congestion_surcharge     0.0
ID                       0.0
dtype: float64
```

Correcting datatype of tpep_pickup_datetime and tpep_dropoff_datetime column

In [9]:
```
data['tpep_pickup_datetime'] = pd.to_datetime(data['tpep_pickup_datetime'])
data['tpep_dropoff_datetime'] = pd.to_datetime(data['tpep_dropoff_datetime'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1271413 entries, 0 to 1271412
Data columns (total 19 columns):
VendorID                 1271413 non-null float64
tpep_pickup_datetime     1271413 non-null datetime64[ns]
tpep_dropoff_datetime    1271413 non-null datetime64[ns]
passenger_count          1271413 non-null float64
trip_distance            1271413 non-null float64
RatecodeID               1271413 non-null float64
store_and_fwd_flag       1271413 non-null object
PULocationID             1271413 non-null int64
DOLocationID             1271413 non-null int64
payment_type             1271413 non-null float64
fare_amount              1271413 non-null float64
extra                    1271413 non-null float64
mta_tax                  1271413 non-null float64
tip_amount               1271413 non-null float64
tolls_amount             1271413 non-null float64
improvement_surcharge    1271413 non-null float64
total_amount             1271413 non-null float64
congestion_surcharge     1271413 non-null float64
ID                       1271413 non-null int64
dtypes: datetime64[ns](2), float64(13), int64(3), object(1)
memory usage: 194.0+ MB
```

Checking anomalies with Pickup time and drop time

Pickup time will be always before drop-off time. We need to find rows with anomalies with trip duration.

In [10]:
```
pd_time_anomaly = len(data[data['tpep_pickup_datetime'] >= data['tpep_dropoff_datetime']])
print(f"Total rows with anomalies with trip duration: {pd_time_anomaly}")
```

```
Total rows with anomalies with trip duration: 794
```

In [11]:
```
# Removing all rows anomalies with trip duration
data = data[data['tpep_pickup_datetime'] < data['tpep_dropoff_datetime']]
pd_time_anomaly = len(data[data['tpep_pickup_datetime'] > data['tpep_dropoff_datetime'] ])
print(f"Now, total rows with anomalies with trip duration: {pd_time_anomaly}")
```

```
Now, total rows with anomalies with trip duration: 0
```

Changing categorical variable into numeric/boolean ¶

All input and output variables in machine learning models must be numeric. This means that if our data contains categorical data, we must convert it to numbers before fitting and evaluating a model. Here, 'store_and_fwd_flag' is Y/N. So, we will convert it to boolean values 0 or 1.

In [24]:
```
data['store_and_fwd_flag'] = data['store_and_fwd_flag'].replace({'N': 0, 'Y': 1})
data.head()
```

Out[24]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payme
0	1.0	2021-01-01 00:30:10	2021-01-01 00:36:12	1.0	2.10	1.0	0	142	43	
1	1.0	2021-01-01 00:51:20	2021-01-01 00:52:19	1.0	0.20	1.0	0	238	151	
2	1.0	2021-01-01 00:43:30	2021-01-01 01:11:06	1.0	14.70	1.0	0	132	165	
4	2.0	2021-01-01 00:31:49	2021-01-01 00:48:21	1.0	4.94	1.0	0	68	33	
5	1.0	2021-01-01 00:16:29	2021-01-01 00:24:30	1.0	1.60	1.0	0	224	68	