Delhivery:

Delhivery is the largest and fastest-growing fully integrated player in India by revenue in Fiscal 2021. They aim to build the operating system for commerce, through a combination of world-class infrastructure, logistics operations of the highest quality, and cutting-edge engineering and technology capabilities.

Basic data cleaning and exploration:

- **Importing Libraries:**

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import scipy.stats as spy
```

- Loading dataset:

```
[ ]  data= pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181")
     data.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	...	cut
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	0
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	

5 rows × 24 columns

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   data                            144867 non-null  object 
 1   trip_creation_time              144867 non-null  object 
 2   route_schedule_uuid             144867 non-null  object 
 3   route_type                      144867 non-null  object 
 4   trip_uuid                       144867 non-null  object 
 5   source_center                   144867 non-null  object 
 6   source_name                     144574 non-null  object 
 7   destination_center              144867 non-null  object 
 8   destination_name                144606 non-null  object 
 9   od_start_time                   144867 non-null  object 
 10  od_end_time                     144867 non-null  object 
 11  start_scan_to_end_scan          144867 non-null  float64
 12  is_cutoff                       144867 non-null  bool   
 13  cutoff_factor                   144867 non-null  int64  
 14  cutoff_timestamp                144867 non-null  object 
 15  actual_distance_to_destination  144867 non-null  float64
 16  actual_time                     144867 non-null  float64
 17  osrm_time                       144867 non-null  float64
 18  osrm_distance                   144867 non-null  float64
 19  factor                          144867 non-null  float64
 20  segment_actual_time             144867 non-null  float64
 21  segment_osrm_time               144867 non-null  float64
 22  segment_osrm_distance           144867 non-null  float64
 23  segment_factor                  144867 non-null  float64
dtypes: bool(1), float64(10), int64(1), object(12)
```

```
data.shape
```

```
(144867, 24)
```

```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
start_scan_to_end_scan	144867.0	961.262986	1037.012769	20.000000	161.000000	449.000000	1634.000000	7898.000000
cutoff_factor	144867.0	232.926567	344.755577	9.000000	22.000000	66.000000	286.000000	1927.000000
actual_distance_to_destination	144867.0	234.073372	344.990009	9.000045	23.355874	66.126571	286.708875	1927.447705
actual_time	144867.0	416.927527	598.103621	9.000000	51.000000	132.000000	513.000000	4532.000000
osrm_time	144867.0	213.868272	308.011085	6.000000	27.000000	64.000000	257.000000	1686.000000
osrm_distance	144867.0	284.771297	421.119294	9.008200	29.914700	78.525800	343.193250	2326.199100
factor	144867.0	2.120107	1.715421	0.144000	1.604264	1.857143	2.213483	77.387097
segment_actual_time	144867.0	36.196111	53.571158	-244.000000	20.000000	29.000000	40.000000	3051.000000
segment_osrm_time	144867.0	18.507548	14.775960	0.000000	11.000000	17.000000	22.000000	1611.000000
segment_osrm_distance	144867.0	22.829020	17.860660	0.000000	12.070100	23.513000	27.813250	2191.403700
segment_factor	144867.0	2.218368	4.847530	-23.444444	1.347826	1.684211	2.250000	574.250000

```python
data.describe(include="object").T
```

	count	unique	top	freq
trip_creation_time	144316	14787	2018-10-01 05:04:55.268931	101
route_schedule_uuid	144316	1497	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	1812
trip_uuid	144316	14787	trip-153837029526866991	101
source_center	144316	1496	IND000000ACB	23267
source_name	144316	1496	Gurgaon_Bilaspur_HB (Haryana)	23267
destination_center	144316	1466	IND000000ACB	15192
destination_name	144316	1466	Gurgaon_Bilaspur_HB (Haryana)	15192
segment_key	144316	26222	trip-153755502932196495_IND160002AAC_IND562132AAA	81

```python
data.isna().sum()
```

```
data                               0
trip_creation_time                 0
route_schedule_uuid                0
route_type                         0
trip_uuid                          0
source_center                      0
source_name                      293
destination_center                 0
destination_name                 261
od_start_time                      0
od_end_time                        0
start_scan_to_end_scan             0
is_cutoff                          0
cutoff_factor                      0
cutoff_timestamp                   0
actual_distance_to_destination     0
actual_time                        0
osrm_time                          0
osrm_distance                      0
factor                             0
segment_actual_time                0
segment_osrm_time                  0
segment_osrm_distance              0
segment_factor                     0
dtype: int64
```

About dataset:

- The given dataset has details about the delivery of the goods and it has 144876 rows and 24 columns.
- The data is given from the period '2018-09-12 00:00:16' to '2018-10-08 03:00:24'.
- There are about 14787 unique trip IDs, 1496 unique source centers, 1466 unique destination centers, 1260 unique source cities, 1256 unique destination cities.
- Most of the data is for testing than for training.
- Most common route type is Carting.
- The names of 10 unique location ids are missing in the data.
- There 293 null values in source name column and 261 null values in destination column.

Column profiling:

- data - tells whether the data is testing or training data
- trip_creation_time – Timestamp of trip creation
- route_schedule_uuid – Unique Id for a particular route schedule
- route_type – Transportation type
- FTL – Full Truck Load: FTL shipments get to the destination sooner, as the truck is making no other pickups or drop-offs along the way
- Carting: Handling system consisting of small vehicles (carts)
- trip_uuid - Unique ID given to a particular trip (A trip may include different source and destination centers)
- source_center - Source ID of trip origin
- source_name - Source Name of trip origin
- destination_cente – Destination ID

- destination_name – Destination Name
- od_start_time – Trip start time
- od_end_time – Trip end time
- start_scan_to_end_scan – Time taken to deliver from source to destination
- is_cutoff – Unknown field
- cutoff_factor – Unknown field
- cutoff_timestamp – Unknown field
- actual_distance_to_destination – Distance in Kms between source and destination warehouse
- actual_time – Actual time taken to complete the delivery (Cumulative)
- osrm_time – An open-source routing engine time calculator which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) and gives the time (Cumulative)
- osrm_distance – An open-source routing engine which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) (Cumulative)
- factor – Unknown field
- segment_actual_time – This is a segment time. Time taken by the subset of the package delivery
- segment_osrm_time – This is the OSRM segment time. Time taken by the subset of the package delivery
- segment_osrm_distance – This is the OSRM distance. Distance covered by subset of the package delivery
- segment_factor – Unknown field

- <u>**Merging rows:**</u>

Create a unique identifier for different segments of a trip based on the combination of the trip_uuid, source_center, destination_center and name it as segment_key.

```
[13] data["segment_key"]= data["trip_uuid"]+"_"+data["source_center"]+"_"+data["destination_center"]

data["segment_actual_time_sum"]= data.groupby("segment_key")["segment_actual_time"].cumsum()
data["segment_osrm_distance_sum"]= data.groupby("segment_key")["segment_osrm_distance"].cumsum()
data["segment_osrm_time_sum"]= data.groupby("segment_key")["segment_osrm_time"].cumsum()

data[["segment_key", "segment_actual_time_sum", "segment_osrm_distance_sum", "segment_osrm_time_sum"]]
```

	segment_key	segment_actual_time_sum	segment_osrm_distance_sum	segment_osrm_time_sum
0	trip-153741093647649320_IND388121AAA_IND388620AAB	14.0	11.9653	11.0
1	trip-153741093647649320_IND388121AAA_IND388620AAB	24.0	21.7243	20.0
2	trip-153741093647649320_IND388121AAA_IND388620AAB	40.0	32.5395	27.0
3	trip-153741093647649320_IND388121AAA_IND388620AAB	61.0	45.5619	39.0
4	trip-153741093647649320_IND388121AAA_IND388620AAB	67.0	49.4772	44.0
...	...	...	...	...
144862	trip-153746066843555182_IND131028AAB_IND000000ACB	92.0	65.3487	94.0
144863	trip-153746066843555182_IND131028AAB_IND000000ACB	118.0	82.7212	115.0
144864	trip-153746066843555182_IND131028AAB_IND000000ACB	138.0	103.4265	149.0
144865	trip-153746066843555182_IND131028AAB_IND000000ACB	155.0	122.3150	176.0
144866	trip-153746066843555182_IND131028AAB_IND000000ACB	423.0	131.1238	185.0

144316 rows × 4 columns

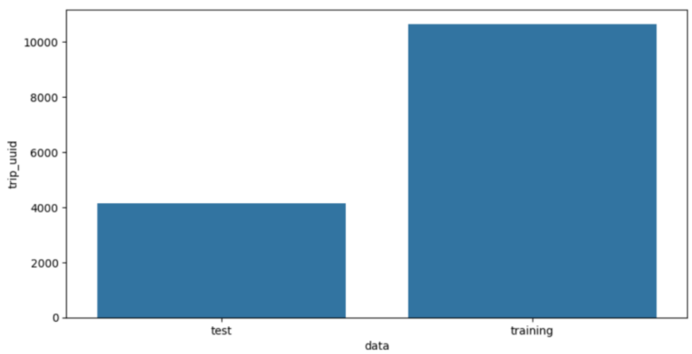
```
# Merging rows

create_segment_dict={ "data" : "first",
                      "route_type" : "first",
                      "trip_creation_time" : "first",
                      "trip_uuid" : "first",
                      "source_center" : "first",
                      "source_name" : "first",
                      "destination_center" : "first",
                      "destination_name" : "last",
                      "od_start_time" : "first",
                      "od_end_time" : "first",
                      "start_scan_to_end_scan" : "first",
                      "actual_distance_to_destination" : "last",
                      "actual_time" : "last",
                      "osrm_time" : "last",
                      "osrm_distance" : "last",
                      "segment_actual_time" : "sum",
                      "segment_osrm_time" : "sum",
                      "segment_osrm_distance" : "sum"}

segmented_data = data.groupby(by= "segment_key", as_index = False).agg(create_segment_dict)
segmented_data = segmented_data.sort_values(by=["segment_key","od_end_time"], ascending=True)
segmented_data.head()
```

```
# FeatureEngineering
# Calculate time taken between od_start_time and od_end_time

segmented_data["od_time_diff_hour"]= round(((segmented_data["od_end_time"] - segmented_data["od_start_time"]).dt.total_seconds()/3600),2)
segmented_data.head()
```

.me	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time	segment_osrm_time	segment_osrm_distance	od_time_diff_hour
-13 '44	1260.0	383.759164	732.0	329.0	446.5496	728.0	534.0	670.6205	21.01
-12 i69	999.0	440.973689	830.0	388.0	544.8027	820.0	474.0	649.8528	16.66
-12 i55	58.0	24.644021	47.0	26.0	28.1994	46.0	26.0	28.1995	0.98
-12 i91	122.0	48.542890	96.0	42.0	56.9116	95.0	39.0	55.9899	2.05
-14 i54	834.0	237.439610	611.0	212.0	281.2109	608.0	231.0	317.7408	13.91

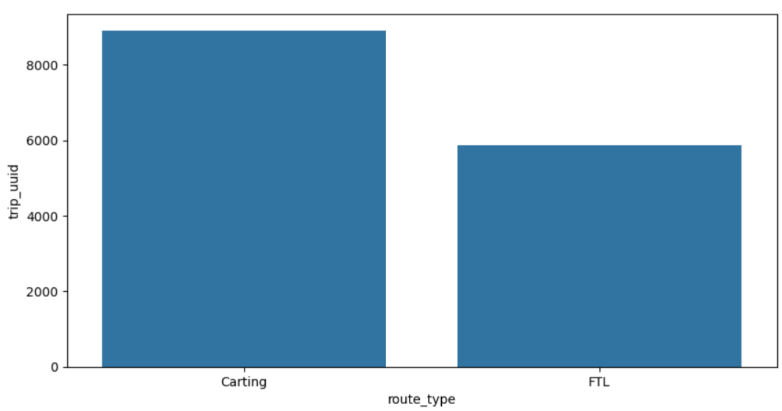
Business insights:

```
plt.figure(figsize = (10, 5))
sns.barplot(data= data_type, x="data", y="trip_uuid")
plt.show()
```



- Most of the data is related to training data.

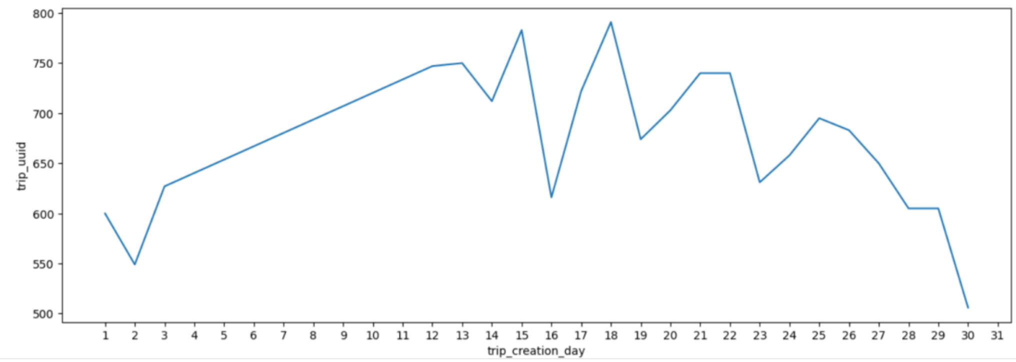
```
[27] plt.figure(figsize = (10, 5))
     sns.barplot(data= data_route, x="route_type", y="trip_uuid")
     plt.show()
```



- Most of the route type is of carting type.

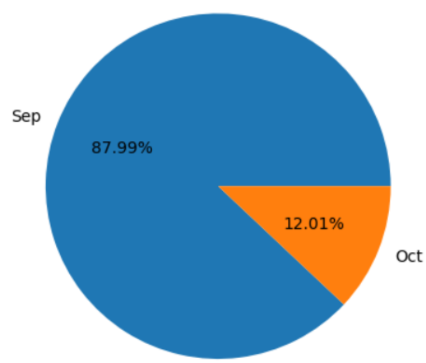
```python
plt.figure(figsize = (15, 5))
sns.lineplot(data = data_day,
             x = data_day['trip_creation_day'],
             y = data_day['trip_uuid'])
plt.xticks(np.arange(1, 32))
plt.plot()
```

```
[]
```

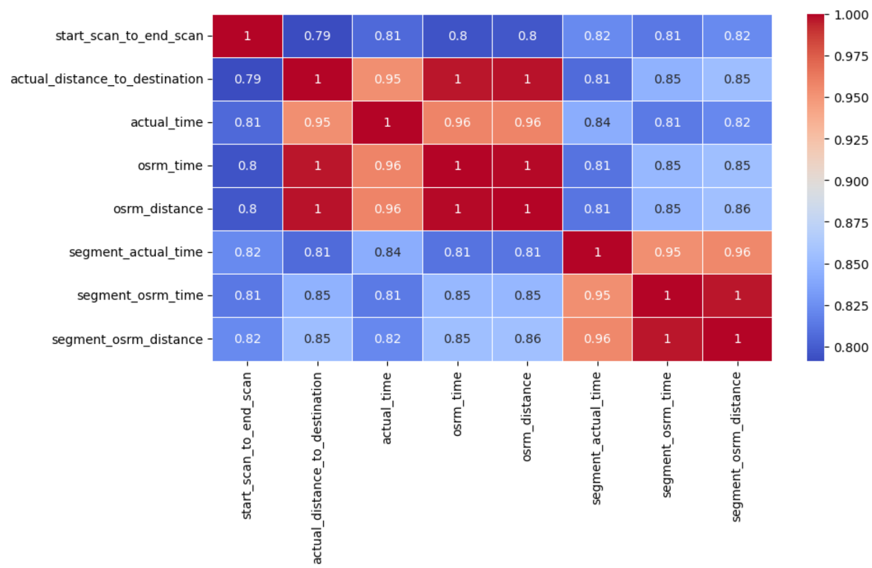



```python
plt.pie(x= data_month["trip_uuid"], labels= ["Sep", "Oct"], autopct= "%.2f%%")
plt.plot()
```

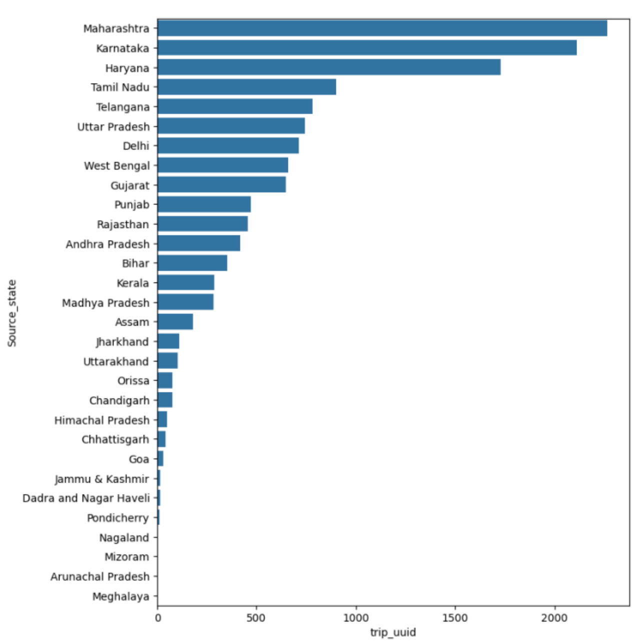
```
[]
```

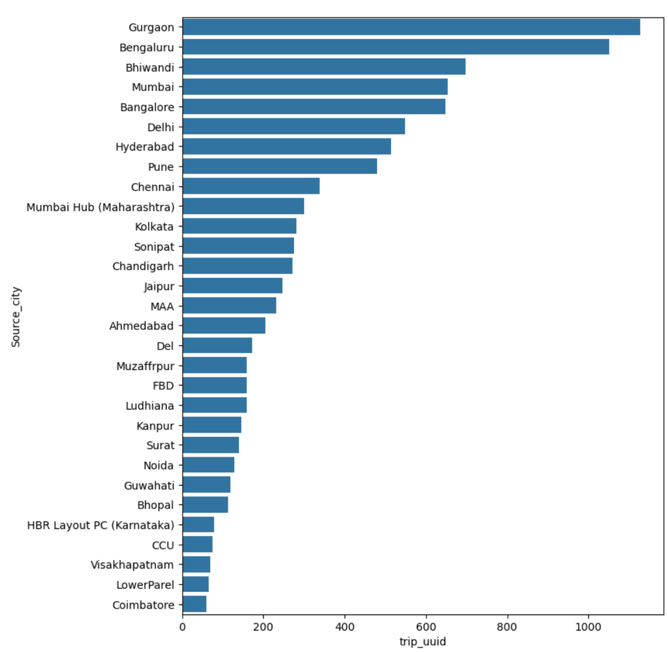


- From the above line plot it can be inferred that most of the trips are created in the mid of the month. That means customers usually make more orders in the mid of the month.
- Also most of the data is from the month of September.

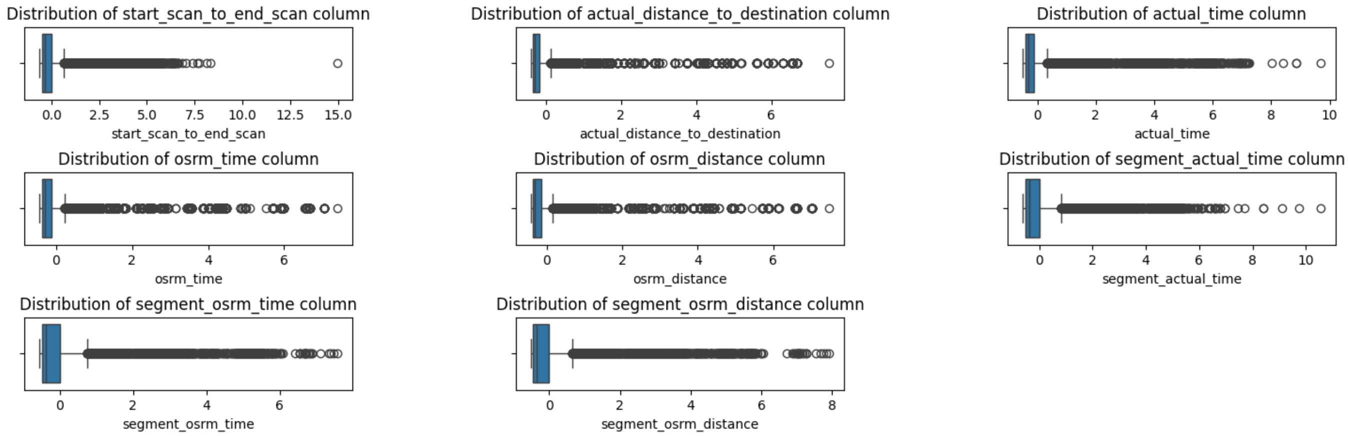


- From the above heat map we can get that high correlation exists between the numerical columns.





- From the above plots it can be seen that maximum trips originated from Maharashtra state followed by Karnataka and Haryana. City wise maximum trips originated from Gurgaon, Bengaluru and Bhiwandi.



- From the above box plots it can be clearly seen that there are outliers in all the numerical columns that need to be treated.

Hypothesis testing:

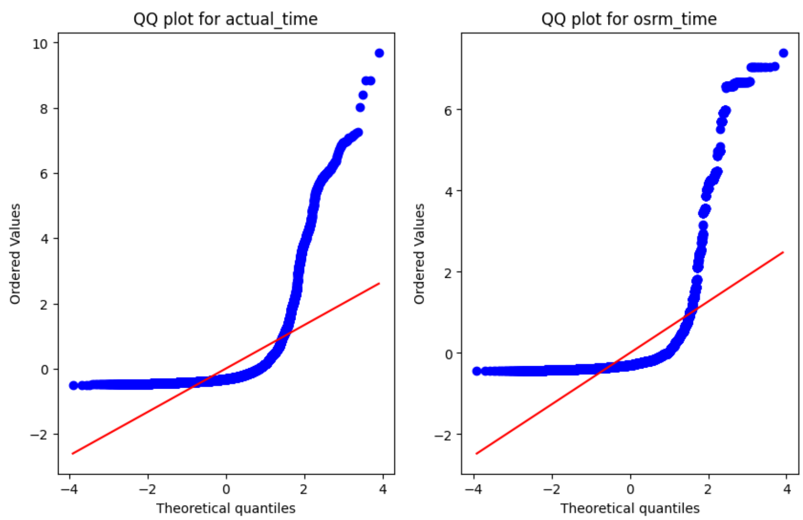
1. actual time aggregated value and OSRM time aggregated value.

H0 - actual time aggregated value and OSRM time aggregated value are same.

HA - actual time aggregated value and OSRM time aggregated value are different.

Assumptions of the test:

- **Normality**: From the below QQ plot we can observe that data does not follow normal distribution.



- **Equal Variance**: From levene's test we can conclude that the datas have different variance.

```
# Homogeneity of Variances using Lavene's test

# H0- Variance are significantly different
# HA- Variance are not significantly different

test_stat, p_value = spy.levene(trip_data["actual_time"], trip_data["osrm_time"])
print('p-value', p_value)
if p_value < 0.05:
    print("Variance are significantly different")
else:
    print("Variance are not significantly different")
```

```
p-value 1.7065702203571972e-134
Variance are significantly different
```

Since the samples are not normally distributed, T-Test cannot be applied here, we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

```
[51] # Since the samples do not follow any of the assumptions T-Test cannot be applied here.
     # we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

     t_stat, p_value = spy.mannwhitneyu(trip_data["actual_time"], trip_data["osrm_time"])
     print("p-value", p_value)
     if p_value < 0.05:
         print("The samples are not similar")
     else:
         print("The samples are similar")

     p-value 9.509176874996746e-61
     The samples are not similar
```

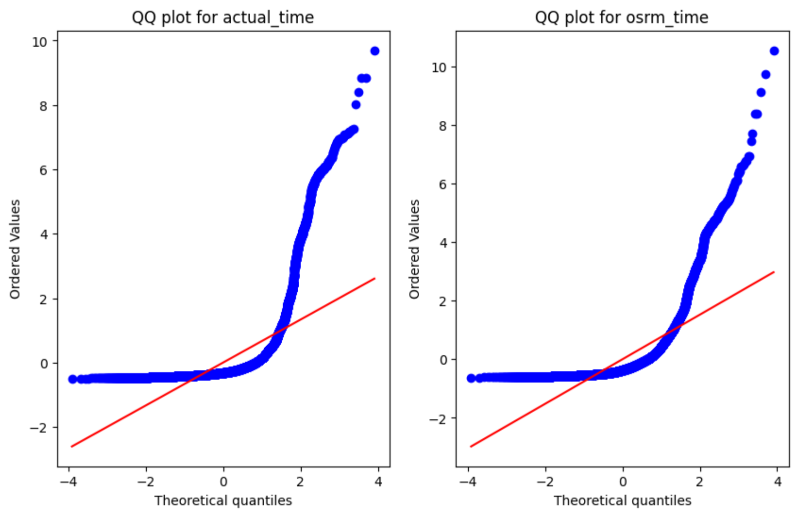
2. actual time aggregated value and segment actual time aggregated value.

H0 - actual time aggregated value and segment actual time value are same.

HA - actual time aggregated value and segment actual time value are different.

Assumptions of the test:

- **Normality**: From the below QQ plot we can observe that data does not follow normal distribution.



- **Equal Variance**: From levene's test we can conclude that the datas have different variance.

```
[66] # Homogeneity of Variances using Lavene's test

     # H0- Variance are significantly different
     # HA- Variance are not significantly different

     test_stat, p_value = spy.levene(trip_data["actual_time"], trip_data["segment_actual_time"])
     print("p-value", p_value)
     if p_value < 0.05:
         print("Variance are significantly different")
     else:
         print("Variance are not significantly different")

     p-value 2.1119134589517006e-23
     Variance are significantly different
```

Since the samples are not normally distributed, T-Test cannot be applied here, we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

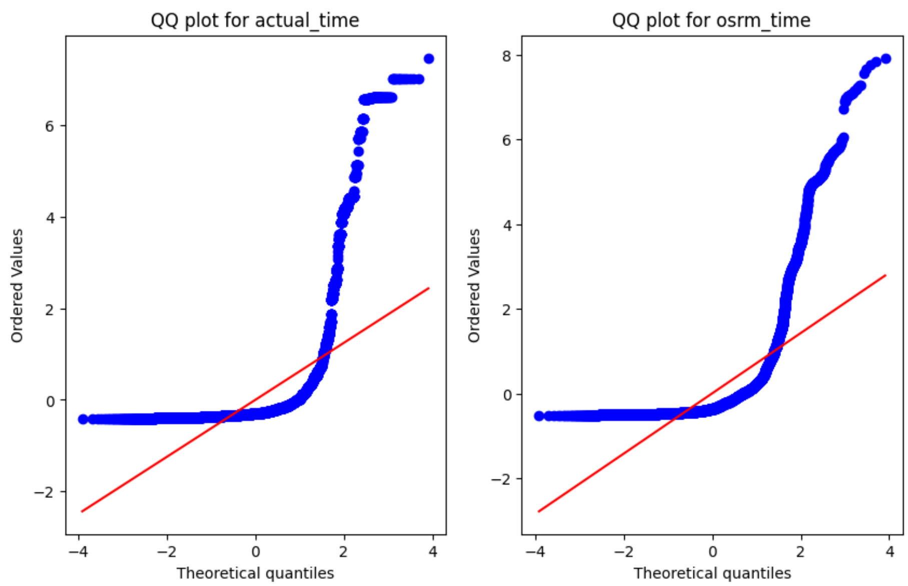
```
[55] # Since the samples do not follow any of the assumptions T-Test cannot be applied here.
     # we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

     t_stat, p_value = spy.mannwhitneyu(trip_data["actual_time"], trip_data["segment_actual_time"])
     print("p-value", p_value)
     if p_value < 0.05:
         print("The samples are not similar")
     else:
         print("The samples are similar")

     p-value 2.3269870249576406e-184
     The samples are not similar
```

3. OSRM distance aggregated value and segment OSRM distance aggregated value.

H0 - OSRM distance aggregated value and segment OSRM distance aggregated value are same.
HA - OSRM distance aggregated value and segment OSRM distance aggregated value are different.

Assumptions of the test:

- **Normality**: From the below QQ plot we can observe that data does not follow normal distribution.



- **Equal Variance**: From levene's test we can conclude that the datas have different variance.

```
[65] # Homogeneity of Variances using Lavene's test

     # H0- Variance are significantly different
     # HA- Variance are not significantly different

     test_stat, p_value = spy.levene(trip_data["osrm_distance"], trip_data["segment_osrm_distance"])
     print("p-value", p_value)
     if p_value < 0.05:
         print("Variance are significantly different")
     else:
         print("Variance are not significantly different")
```

```
p-value 1.5684805255129562e-18
Variance are significantly different
```

Since the samples are not normally distributed, T-Test cannot be applied here, we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

```
[59] # Since the samples do not follow any of the assumptions T-Test cannot be applied here.
     # we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

     t_stat, p_value = spy.mannwhitneyu(trip_data["osrm_distance"], trip_data["segment_osrm_distance"])
     print("p-value", p_value)
     if p_value < 0.05:
         print("The samples are not similar")
     else:
         print("The samples are similar")
```

```
p-value 6.077882880733487e-272
The samples are not similar
```

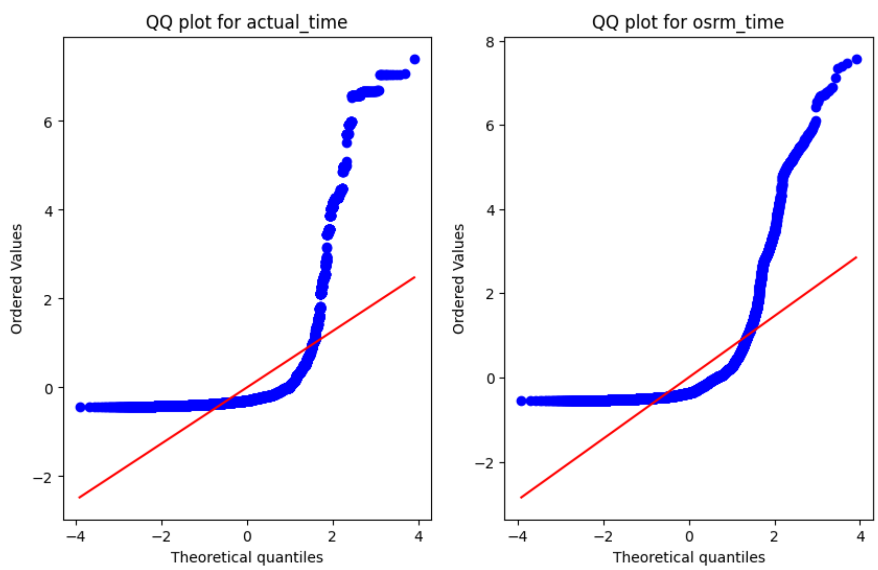
4. OSRM time aggregated value and segment OSRM time aggregated value.

H0 - OSRM time aggregated value and segment OSRM time aggregated value are same.

HA - OSRM time aggregated value and segment OSRM time aggregated value are different.

Assumptions of the test:

- **Normality**: From the below QQ plot we can observe that data does not follow normal distribution.



- **Equal Variance**: From levene's test we can conclude that the datas have different variance.

```
[64] # Homogeneity of Variances using Lavene's test

     # H0- Variance are significantly different
     # HA- Variance are not significantly different

     test_stat, p_value = spy.levene(trip_data["osrm_time"], trip_data["segment_osrm_time"])
     print("p-value", p_value)
     if p_value < 0.05:
         print("Variance are significantly different")
     else:
         print("Variance are not significantly different")
```

```
p-value 3.372487757874399e-21
Variance are significantly different
```

Since the samples are not normally distributed, T-Test cannot be applied here, we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

```
[63] # Since the samples do not follow any of the assumptions T-Test cannot be applied here.
     # we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

     t_stat, p_value = spy.mannwhitneyu(trip_data["osrm_time"], trip_data["segment_osrm_time"])
     print("p-value", p_value)
     if p_value < 0.05:
         print("The samples are not similar")
     else:
         print("The samples are similar")

     p-value 1.6475103625829434e-233
     The samples are not similar
```

Recommendations:

- OSRM time and actual time are different. Team needs to make sure this difference is reduced, so that better delivery time prediction can be made and it becomes convenient for the customer to expect an accurate delivery time.
- OSRM distance and actual distance covered are also not same. . Team needs to look into it as if maybe the delivery person is not following the predefined route which may lead to late deliveries or the OSRM devices is not properly predicting the route based on distance, traffic and other factors.
- Most of the orders are from/reaching to states like Maharashtra, Karnataka, Haryana and Tamil Nadu. The existing platforms should be improved to increase more customer interactions.
- The team should also focus on marketing in moderate states to improve the market hold.
- Customer profiling of the customers of these states has to be done to get to know why major orders are coming from these states and to improve customer's buying and delivery experience.