About 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.

The Data team builds intelligence and capabilities using this data that helps them to widen the gap between the quality, efficiency, and profitability of their business versus their competitors.

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

Basic data cleaning and exploration:

- Handle missing values in the data.
- Analyze the structure of the data.
- Try merging the rows using the hint mentioned above.

Build some features to prepare the data for actual analysis. Extract features from the below fields:

- Destination Name: Split and extract features out of destination. City-place-code (State)
- Source Name: Split and extract features out of destination. City-place-code (State)
- Trip_creation_time: Extract features like month, year and day etc

In-depth analysis and feature engineering:

- Calculate the time taken between od_start_time and od_end_time and keep it as a feature. Drop the original columns, if required
- Compare the difference between Point a. and start_scan_to_end_scan. Do hypothesis testing/ Visual analysis to check.
- Do hypothesis testing/ visual analysis between actual_time aggregated value and OSRM time aggregated value (aggregated values are the values you'll get after merging the rows on the basis of trip_uuid)
- Do hypothesis testing/ visual analysis between actual_time aggregated value and segment actual time aggregated value (aggregated values are the values you'll get after merging the rows on the basis of trip_uuid)

- Do hypothesis testing/ visual analysis between osrm distance aggregated value and segment osrm distance aggregated value (aggregated values are the values you'll get after merging the rows on the basis of trip_uuid)
- Do hypothesis testing/ visual analysis between osrm time aggregated value and segment osrm time aggregated value (aggregated values are the values you'll get after merging the rows on the basis of trip_uuid)
- Find outliers in the numerical variables (you might find outliers in almost all the variables), and check it using visual analysis
- Handle the outliers using the IQR method.
- Do one-hot encoding of categorical variables (like route_type)
- Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler.

In [539]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
import statsmodels.api as sm
from scipy.stats import norm
from scipy.stats import t
import plotly.express as px
```

In [540]:
```
pd.set_option('display.max_columns',None)
```

In [541]:
```
data=pd.read_csv(r"C:\Users\Sweta.Singh\Downloads\delhivery_data.csv")
```

In [542]:
```
data
```

Out[542]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end_time	start_scan_to_
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
...	...	...	...	...	...	...	...	...	...	...	...	...
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	

144867 rows × 24 columns

In [543]:
```
data.head(5)
```

Out[543]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end_time	start_scan_to_end_s
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	

Understanding shape and structure of data:

In [544]:
```
data.shape
```

Out[544]:
```
(144867, 24)
```

- 144867 total records
- 24 columns

In [545]:
```
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)
memory usage: 25.6+ MB
```

In [546]:
```
data.isna().sum()
```

Out[546]:
```
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
```

- source_name and destination_name has 293 and 261 missing values

Changing data type for data and time related features:

In [547]:
```
data["od_end_time"]=pd.to_datetime(data["od_end_time"])
data["od_start_time"]=pd.to_datetime(data["od_start_time"])
data["trip_creation_time"]=pd.to_datetime(data["trip_creation_time"])
```

In [548]:
```
data.head(5)
```

Out[548]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end_time	start_scan_to_end_s
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	

In [549]:
```
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  datetime64[ns]
 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  datetime64[ns]
 10  od_end_time                     144867 non-null  datetime64[ns]
 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), datetime64[ns](3), float64(10), int64(1), object(9)
memory usage: 25.6+ MB
```

Extracting Trip Creation Information from trip creation time:

In [550]:
```
data["trip_creation_time"].dt.month_name().value_counts()
```

Out[550]:
```
September    127349
October       17518
Name: trip_creation_time, dtype: int64
```

In [551]:
```
data["trip_creation_time"].dt.year.value_counts()
```

Out[551]:
```
2018    144867
Name: trip_creation_time, dtype: int64
```

- delivery trip data is given from september and october 2018.

In [552]:
```
data["trip_creation_day"]=(data["trip_creation_time"].dt.day_name())
data["trip_creation_month"]=(data["trip_creation_time"].dt.month_name())
data["trip_creation_year"]=(data["trip_creation_time"].dt.year)
```

In [553]:
```
data["trip_creation_day"].value_counts(normalize=True)*100
```

Out[553]:
```
Wednesday    18.452788
Thursday     14.137795
Friday       13.972816
Tuesday      13.778845
Saturday     13.761588
Monday       13.560714
Sunday       12.335453
Name: trip_creation_day, dtype: float64
```

- Wednesday seems to have relatively higher records of data compare to other days.

Understanding the structure

In [554]:
```
data.nunique()
```

Out[554]:
```
data                                2
trip_creation_time              14817
route_schedule_uuid              1504
route_type                          2
trip_uuid                       14817
source_center                    1508
source_name                      1498
destination_center               1481
destination_name                 1468
od_start_time                   26369
od_end_time                     26369
start_scan_to_end_scan           1915
is_cutoff                           2
cutoff_factor                     501
cutoff_timestamp                93180
actual_distance_to_destination  144515
actual_time                      3182
osrm_time                        1531
osrm_distance                  138046
factor                          45641
segment_actual_time               747
segment_osrm_time                 214
segment_osrm_distance          113799
segment_factor                   5675
trip_creation_day                   7
trip_creation_month                 2
trip_creation_year                  1
dtype: int64
```

- We have 14817 different trips happended between source to destinations.
- Total 1504 dehlivery routes.
- 1504 unique source centers.
- 1481 unique destination centers.

There are two different kind of routes.

In [555]:
```
data.groupby("trip_uuid")["route_type"].unique().reset_index()["route_type"].apply(lambda x:x[0]).value_counts()
```

Out[555]:
```
Carting    8908
FTL        5909
Name: route_type, dtype: int64
```

In [556]:
```
data.groupby("trip_uuid")["route_type"].unique().reset_index()["route_type"].apply(lambda x:x[0]).value_counts(normalize = True)*100
```

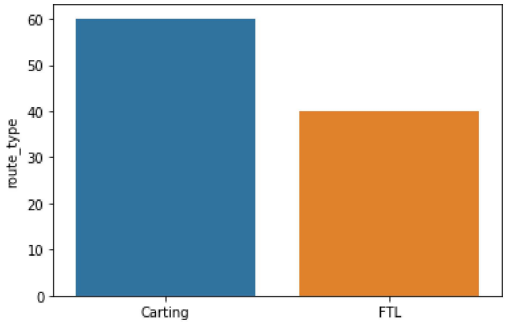
Out[556]:
```
Carting    60.120132
FTL        39.879868
Name: route_type, dtype: float64
```

In [557]:
```
routeType_plot= (data.groupby("trip_uuid")["route_type"].unique().reset_index()["route_type"].apply(lambda x:x[0]).value_counts(normalize = True)*100)
routeType_plot
```

Out[557]:
```
Carting    60.120132
FTL        39.879868
Name: route_type, dtype: float64
```

In [558]:
```
sns.barplot(x=routeType_plot.index,
            y= routeType_plot)
```

Out[558]:
```
<AxesSubplot:ylabel='route_type'>
```



- Out of 14817 total different trips, we have 8908 (60%) of the trip-routes are Carting, which consists of small vehicles and 5909 (40%) of total trip-routes are FTL: which are full truck load get to the destination sooner as no other pickups or drop offs along the way.

Exploratory Data Analysis

Busiest Route Analysis

- Number of Trips between cities , sorted highest to lowest
  - What are the top 20 cities with the highest frequency of trips as both source and destination?

In [703]:
```
Number_of_trips_between_cities = data1.groupby(["source_city_state",
                                                "destination_city_state"])["trip_uuid"].nunique().sort_values(ascending=False).reset_index()
Number_of_trips_between_cities.head(25)
```

Out[703]:

	source_city_state	destination_city_state	trip_uuid
0	Bengaluru Karnataka	Bengaluru Karnataka	1369
1	Bhiwandi Maharashtra	Mumbai Maharashtra	512
2	Mumbai Maharashtra	Mumbai Maharashtra	361
3	Hyderabad Telangana	Hyderabad Telangana	308
4	Mumbai Maharashtra	Bhiwandi Maharashtra	282
5	Delhi Delhi	Gurgaon Haryana	248
6	Gurgaon Haryana	Delhi Delhi	237
7	Mumbai Hub Maharashtra	Mumbai Maharashtra	227
8	Chennai Tamil Nadu	Chennai Tamil Nadu	205
9	MAA Tamil Nadu	Chennai Tamil Nadu	204
10	Chennai Tamil Nadu	MAA Tamil Nadu	141
11	Bengaluru Karnataka	HBR Karnataka	133
12	Ahmedabad Gujarat	Ahmedabad Gujarat	131
13	Pune Maharashtra	PNQ Maharashtra	122
14	Jaipur Rajasthan	Jaipur Rajasthan	111
15	Delhi Delhi	Delhi Delhi	109
16	Pune Maharashtra	Bhiwandi Maharashtra	107
17	Pune Maharashtra	Pune Maharashtra	101
18	Chandigarh Chandigarh	Chandigarh Punjab	100
19	Kolkata West Bengal	CCU West Bengal	96
20	Gurgaon Haryana	Sonipat Haryana	92
21	Sonipat Haryana	Gurgaon Haryana	86
22	Chandigarh Punjab	Chandigarh Chandigarh	84
23	HBR Karnataka	Bengaluru Karnataka	79
24	Bengaluru Karnataka	BLR Karnataka	78

- The cities of Mumbai in Maharashtra, Delhi, Gurgaon in Haryana, Bengaluru in Karnataka, Hyderabad in Telangana, Chennai in Tamil Nadu, Ahmedabad in Gujarat, Pune in Maharashtra, Chandigarh in Chandigarh, and Kolkata in West Bengal have the highest number of trips within the city.

In [704]:
```
Number_of_trips_between_cities.loc[Number_of_trips_between_cities["source_city_state"] != Number_of_trips_between_cities["destination_city_state"]].head(25)
```

Out[704]:

	source_city_state	destination_city_state	trip_uuid
1	Bhiwandi Maharashtra	Mumbai Maharashtra	512
4	Mumbai Maharashtra	Bhiwandi Maharashtra	282
5	Delhi Delhi	Gurgaon Haryana	248
6	Gurgaon Haryana	Delhi Delhi	237
7	Mumbai Hub Maharashtra	Mumbai Maharashtra	227
9	MAA Tamil Nadu	Chennai Tamil Nadu	204
10	Chennai Tamil Nadu	MAA Tamil Nadu	141
11	Bengaluru Karnataka	HBR Karnataka	133
13	Pune Maharashtra	PNQ Maharashtra	122
16	Pune Maharashtra	Bhiwandi Maharashtra	107
18	Chandigarh Chandigarh	Chandigarh Punjab	100
19	Kolkata West Bengal	CCU West Bengal	96
20	Gurgaon Haryana	Sonipat Haryana	92
21	Sonipat Haryana	Gurgaon Haryana	86
22	Chandigarh Punjab	Chandigarh Chandigarh	84
23	HBR Karnataka	Bengaluru Karnataka	79
24	Bengaluru Karnataka	BLR Karnataka	78
26	Del Delhi	Gurgaon Haryana	76
27	Bhiwandi Maharashtra	Pune Maharashtra	72
28	Ludhiana Punjab	Chandigarh Punjab	71
30	Chandigarh Punjab	Gurgaon Haryana	66
31	Gurgaon Haryana	Bengaluru Karnataka	66
32	LowerParel Maharashtra	Mumbai Maharashtra	65
34	Mumbai Hub Maharashtra	Bhiwandi Maharashtra	63
35	PNQ Maharashtra	Pune Maharashtra	62

- The cities with the highest number of trips between their source and destination states are Delhi to Gurgaon, Gurgaon to Bengaluru, Bhiwandi/Mumbai to Pune, and Sonipat to Gurgaon.
- Additionally, it has been noticed that many deliveries are made to airports, such as Chennai to the Chennai International Airport, Pune to the Pune Airport, Kolkata to the Kolkata International Airport, and Bengaluru to the Bengaluru International Airport.

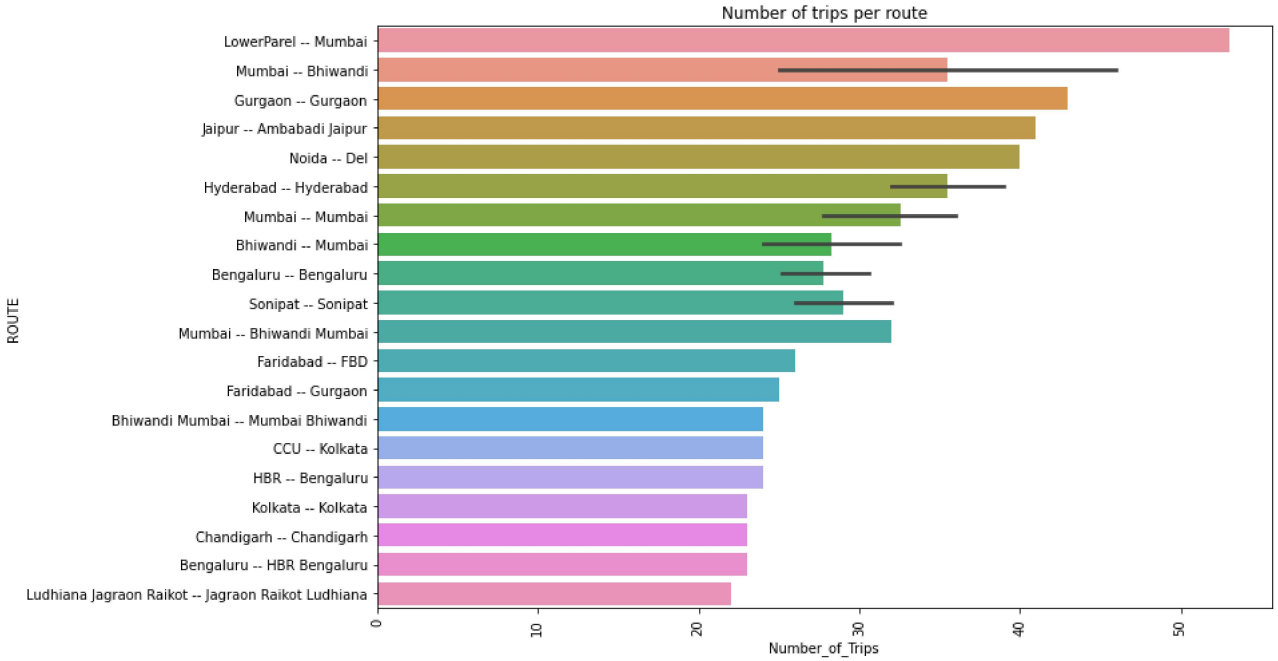
In [705]:
```
route_records[["ROUTE","Number_of_Trips",
               "Average_Actual_distance_to_destination",
               "#source_cities",
               "#destination_cities"]].sort_values(by="Number_of_Trips",ascending=False).head(25)
```

Out[705]:

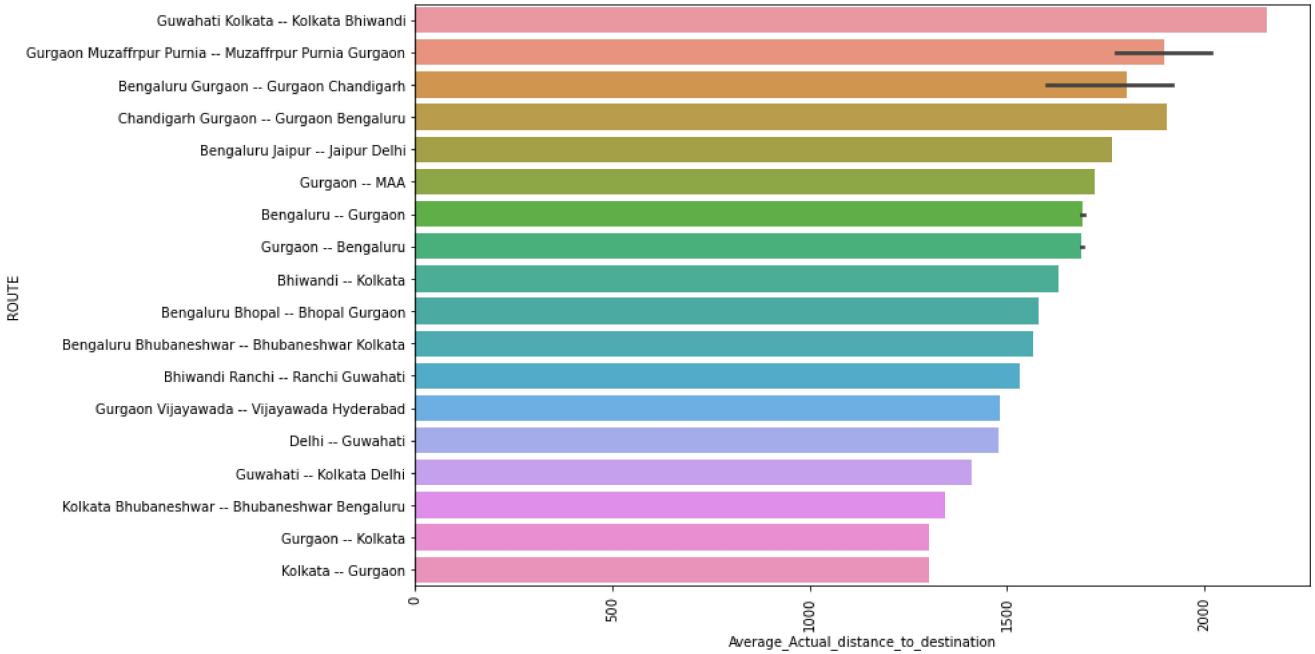
	ROUTE	Number_of_Trips	Average_Actual_distance_to_destination	#source_cities	#destination_cities
1465	LowerParel -- Mumbai	53	16.428868	1	1
1426	Mumbai -- Bhiwandi	46	20.199445	1	1
808	Gurgaon -- Gurgaon	43	29.740842	1	1
679	Jaipur -- Ambabadi Jaipur	41	15.348495	1	2
1257	Noida -- Del	40	10.882902	1	1
1368	Hyderabad -- Hyderabad	39	35.695641	1	1
1273	Mumbai -- Mumbai	37	13.882863	1	1
1359	Mumbai -- Mumbai	36	17.526251	1	1
1303	Bhiwandi -- Mumbai	35	21.241534	1	1
700	Mumbai -- Mumbai	34	15.906614	1	1
751	Mumbai -- Mumbai	33	15.668726	1	1
1060	Bengaluru -- Bengaluru	33	28.067004	1	1
793	Sonipat -- Sonipat	32	11.691243	1	1
972	Hyderabad -- Hyderabad	32	21.835579	1	1
1184	Mumbai -- Bhiwandi Mumbai	32	21.601109	1	2
874	Bengaluru -- Bengaluru	30	28.055789	1	1
1177	Bhiwandi -- Mumbai	30	21.396002	1	1
1354	Bengaluru -- Bengaluru	27	27.967087	1	1
921	Faridabad -- FBD	26	9.677121	1	1
1480	Sonipat -- Sonipat	26	12.182486	1	1
1041	Mumbai -- Bhiwandi	25	19.942191	1	1
877	Faridabad -- Gurgaon	25	47.091622	1	1
833	Bhiwandi -- Mumbai	25	21.531705	1	1
1249	Bengaluru -- Bengaluru	25	28.019668	1	1
869	Bengaluru -- Bengaluru	24	41.396497	1	1

Routes with the greatest number of trips between and within the source and destinations.

In [706]:
```
plt.figure(figsize=(12,8))

X = route_records[["ROUTE", "Number_of_Trips",
                ]].sort_values(by="Number_of_Trips",ascending=False).head(35)
sns.barplot(y = X["ROUTE"],
            x= X["Number_of_Trips"])
plt.title("Number of trips per route")
plt.xticks(rotation = 90)
plt.show()
```



In [707]:
```
plt.figure(figsize=(12,8))

X = route_records[["ROUTE", "Average_Actual_distance_to_destination",
                ]].sort_values(by="Average_Actual_distance_to_destination",ascending=False).head(25)
sns.barplot(y = X["ROUTE"],
            x = X["Average_Actual_distance_to_destination"])
plt.xticks(rotation = 90)
plt.show()
```



- The bar chart and table indicate that the highest number of trips occur within specific cities.
- In terms of average distance between destinations, the longest routes are found to be Guwahati to Mumbai, Bangalore to Chandigarh, Bangalore to Delhi, and Bangalore to Gurgaon.

Routes with the Highest Traffic and Greatest Length

In [710]:
```
Busiest_and_Longest_Routes  = route_records[(route_records["Average_Actual_distance_to_destination"] > route_records["Average_Actual_distance_to_destination"].quantile(0.75))
             & (route_records["Number_of_Trips"] > route_records["Number_of_Trips"].quantile(0.75))].sort_values(by="Average_Actual_distance_to_destination"
                                                                                                                 ,ascending=False)
```

In [711]:
```
Busiest_and_Longest_Routes_top25 = Busiest_and_Longest_Routes[["source_cities",
                                                               "destination_cities",
                                                               "Number_of_Trips",
                                                               "Average_Actual_distance_to_destination"]].head(25)
Busiest_and_Longest_Routes_top25
```

Out[711]:

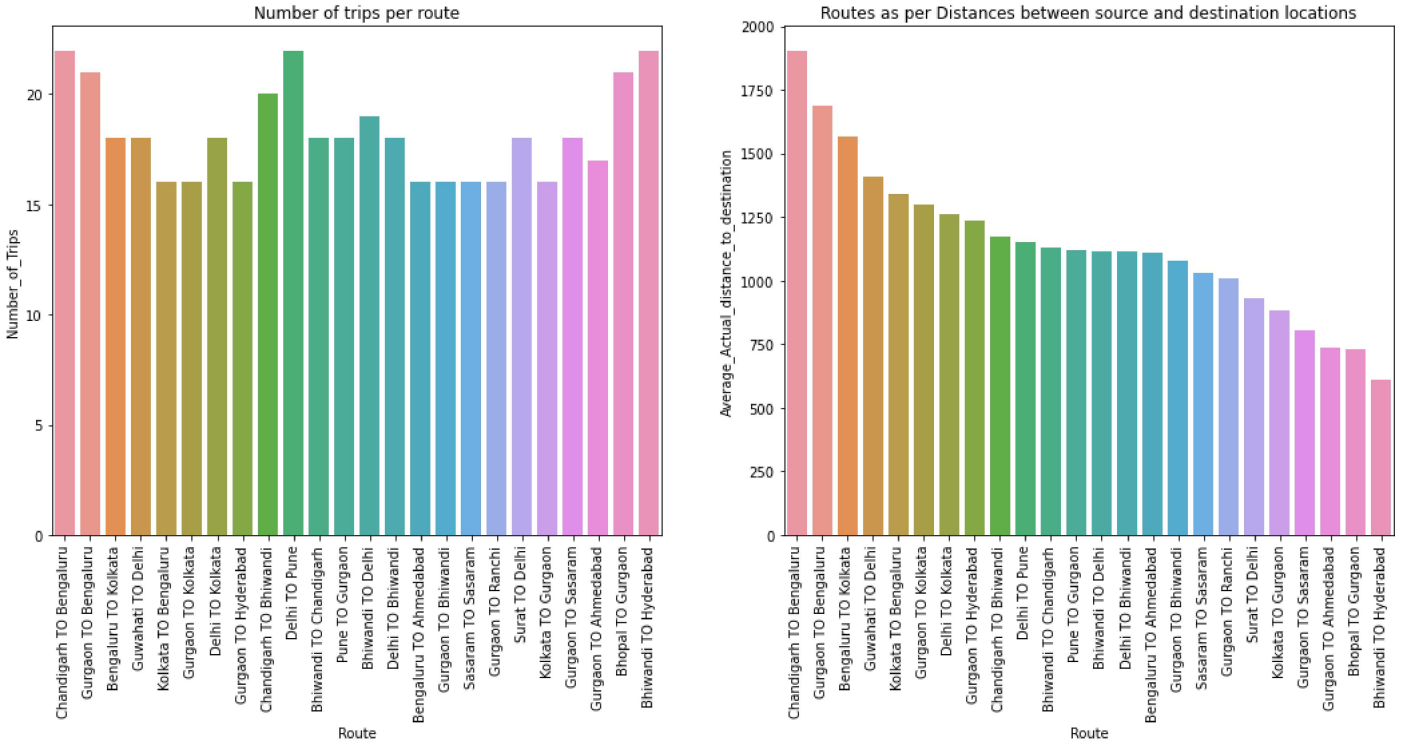
	source_cities	destination_cities	Number_of_Trips	Average_Actual_distance_to_destination
629	Chandigarh Gurgaon	Gurgaon Bengaluru	22	1905.766051
995	Gurgaon	Bengaluru	21	1689.873158
991	Gurgaon	Bengaluru	21	1689.791894
512	Bengaluru Bhubaneshwar	Bhubaneshwar Kolkata	18	1567.577507
745	Guwahati	Kolkata Delhi	18	1411.208424
624	Kolkata Bhubaneshwar	Bhubaneshwar Bengaluru	16	1342.143081
752	Gurgaon	Kolkata	16	1300.572161
588	Delhi Gurgaon	Gurgaon Kolkata	18	1263.113211
826	Gurgaon	Hyderabad	16	1236.572072
541	Chandigarh Gurgaon	Gurgaon Bhiwandi	20	1170.817927
442	Delhi Gurgaon	Gurgaon Pune	22	1151.514940
445	Bhiwandi Sonipat	Sonipat Chandigarh	18	1129.609705
739	Pune	Gurgaon	18	1120.729446
1377	Bhiwandi	Delhi	19	1114.214670
1049	Delhi	Bhiwandi	18	1114.182197
313	Bengaluru Kolhapur Surat	Kolhapur Surat Ahmedabad	16	1110.015339
1219	Gurgaon	Bhiwandi	16	1078.076312
197	Sasaram Kanpur Kolkata Dhanbad	Kanpur Gurgaon Dhanbad Sasaram	16	1028.024726
1136	Gurgaon	Ranchi	16	1010.953223
1286	Surat	Delhi	18	931.980821
439	Kolkata Ranchi	Ranchi Gurgaon	16	881.621264
1108	Gurgaon	Sasaram	18	804.210670
1454	Gurgaon	Ahmedabad	17	735.550450
223	Bhopal Kanpur Auraiya Etawah	Kanpur Auraiya Etawah Gurgaon	21	731.634456
863	Bhiwandi	Hyderabad	22	607.514619

- The above table displays the top city-to-city routes with the highest number of trips over long distances, including: Chandigarh to Bengaluru, Gurgaon to Bengaluru, Bengaluru to Kolkata, Guwahati to Delhi, Delhi to Kolkata, Chandigarh to Gurgaon, Gurgaon to Hyderabad, Bengaluru to Ahmedabad, Surat to Delhi, and Gurgaon to Ahmedabad.

In [712]:
```
Busiest_and_Longest_Routes_top25["Route"] = Busiest_and_Longest_Routes_top25["source_cities"].str.split(" ").apply(lambda x:x[0]) + " TO " + Busiest_and_Longest_Routes_top25["
```

In [713]:
```
Busiest_and_Longest_Routes_top25.drop(["source_cities","destination_cities"],axis = 1,inplace=True)
```

In [715]:
```
plt.figure(figsize=(18,7))

plt.subplot(121)
plt.title("Number of trips per route")
sns.barplot(x=Busiest_and_Longest_Routes_top25["Route"],
            y = Busiest_and_Longest_Routes_top25["Number_of_Trips"])
plt.xticks(rotation = 90)
plt.subplot(122)
plt.title("Routes as per Distances between source and destination locations")
sns.barplot(x=Busiest_and_Longest_Routes_top25["Route"],
            y= Busiest_and_Longest_Routes_top25["Average_Actual_distance_to_destination"])
plt.xticks(rotation = 90)
plt.show()
```



- The charts above depict the locations with the highest number of trips between them, which have long distances between their source and destination points.

In [716]:
```
route_records.columns
```

Out[716]:
```
Index(['SouceToDestination_city', 'ROUTE', '#source_cities',
       '#destination_cities', 'Number_of_Trips',
       'Average_Actual_distance_to_destination', '#source_states',
       '#destination_states', 'destination_states', 'source_states',
       'source_cities', 'route_type', 'destination_cities'],
      dtype='object')
```

Routes with the highest number of cities passed through

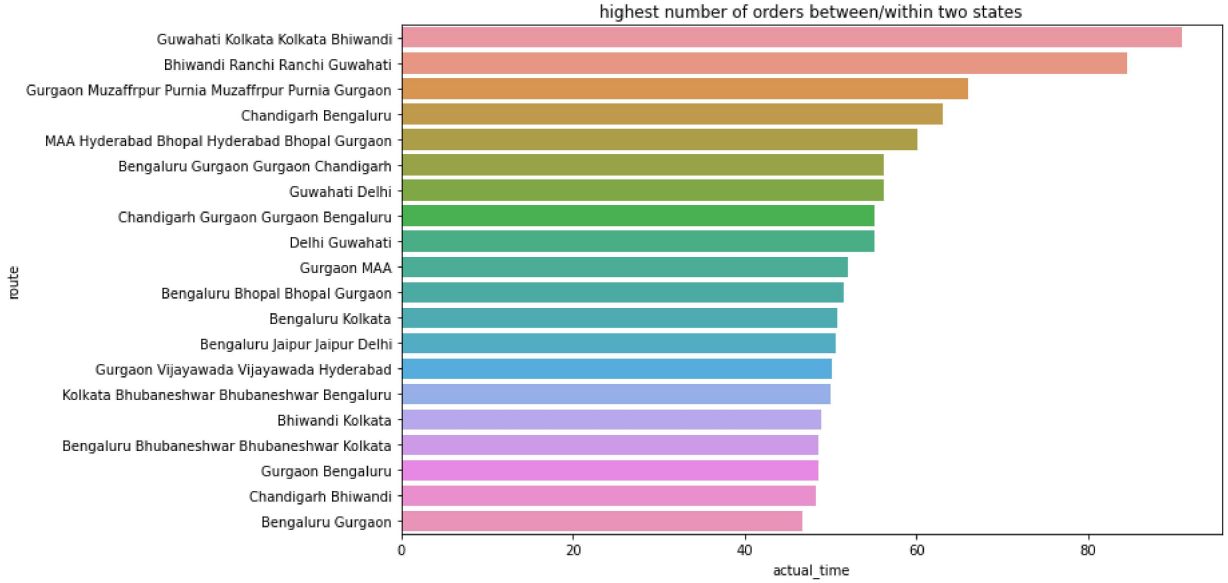
In [717]:
```
route_records[["SouceToDestination_city","Number_of_Trips",
               "Average_Actual_distance_to_destination",
               "#source_cities",
               "#destination_cities"]].sort_values(by=["#source_cities",
                                                       "#destination_cities",
                                                       "Number_of_Trips"]
                                                   ,ascending=False).head(25)
```

Out[717]:

	SouceToDestination_city	Number_of_Trips	Average_Actual_distance_to_destination	#source_cities	#destination_cities
0	Guwahati TO LakhimpurN	14	281.596486	13	11
2	Jaipur TO Tarnau	20	351.611796	10	10
1	Guwahati TO Tura	12	332.602225	10	10
3	Mangalore TO Udupi	9	195.257193	9	9
4	Ajmer TO Raipur	20	178.737233	9	8
5	Mainpuri TO Tilhar	12	207.247057	8	8
8	Hassan TO Koppa	21	200.497832	7	7
15	Shrirampur TO Sangamner	20	204.509529	7	7
7	Musiri TO Tiruchi	19	219.845121	7	7
9	Bijnor TO Bijnor	17	209.400685	7	7
10	Dausa TO Lalsot	17	232.408310	7	7
17	Tinusukia TO Dibrugarh	16	111.098543	7	7
12	Pondicherry TO Pondicherry	12	230.253602	7	7
14	Mysore TO Mysore	12	154.324190	7	7
6	Golaghat TO Guwahati	11	258.546587	7	7
13	Varanasi TO Varanasi	8	82.545019	7	7
16	Vijayawada TO Suryapet	8	407.029391	7	7
11	Hyderabad TO Miryalguda	7	420.603709	7	7
27	Srikakulam TO Bobbili	22	154.495283	6	6
36	Pukhrayan TO Kanpur	22	139.834945	6	6
48	Dhule TO Shirpur	22	150.016233	6	6
30	Madhupur TO Madhupur	21	252.072259	6	6
38	Kamareddy TO Kamareddy	21	177.923330	6	6
42	Noida TO Khurja	21	208.714043	6	6
20	Junagadh TO Veraval	19	179.538596	6	6

The top 20 longest routes, based on the average actual time taken to travel from one city to another.

In [718]:
```
Longest_route_as_per_actual_trip_time = trip_records.groupby(["source_city",
                     "destination_city"])["actual_time"].mean().sort_values(ascending=False).head(20).reset_index()
Longest_route_as_per_actual_trip_time["route"] = Longest_route_as_per_actual_trip_time["source_city"] + " " + Longest_route_as_per_actual_trip_time["destination_city"]
Longest_route_as_per_actual_trip_time.drop(["source_city",
                                            "destination_city"],axis = 1,inplace=True)
Longest_route_as_per_actual_trip_time
plt.figure(figsize=(11,7))
sns.barplot(y = Longest_route_as_per_actual_trip_time["route"],
            x = Longest_route_as_per_actual_trip_time["actual_time"],)
plt.title("highest number of orders between/within two states")
plt.show()
```

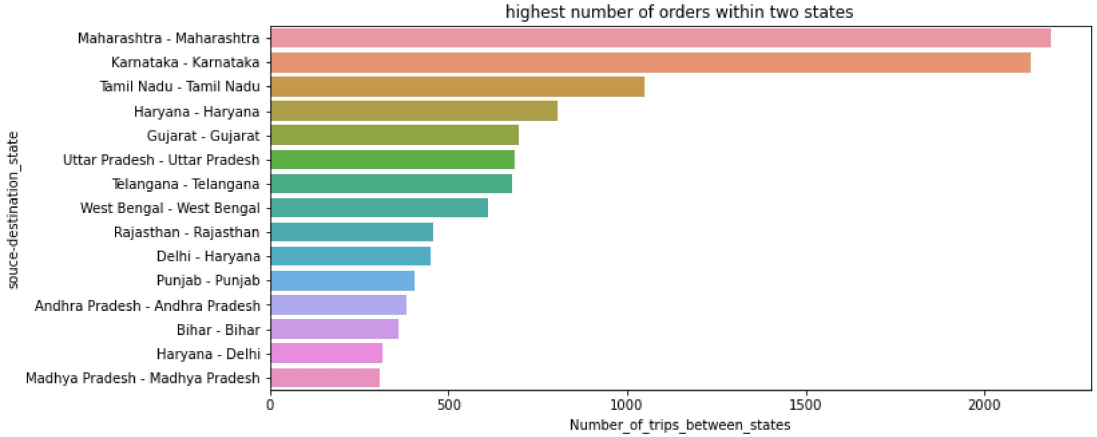


What is the highest number of trips that occur between or within two states?

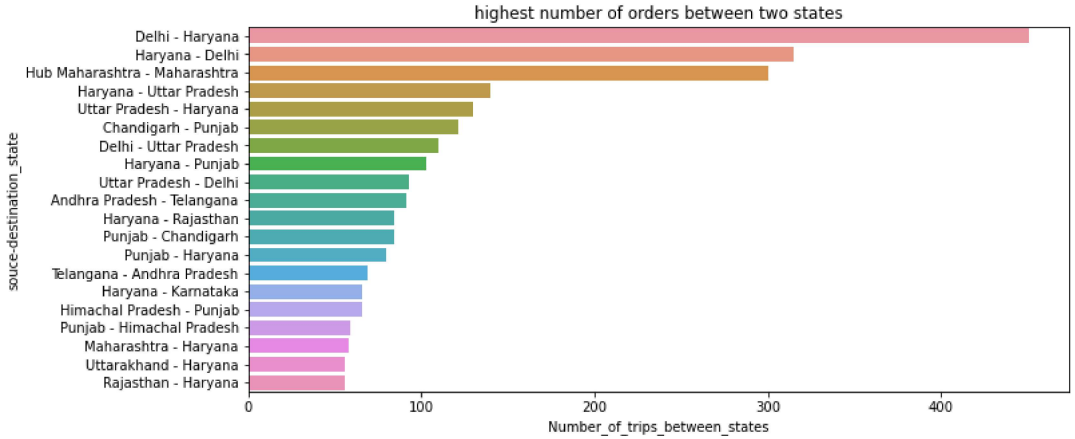
In [719]:
```
highest_order_between_states = data.groupby(["source_state",
                                             "destination_state"])["trip_uuid"].nunique().sort_values(ascending=False).reset_index()
```

In [720]:
```
HOBS  = highest_order_between_states.head(15)
HOBS["souce-destination"] = HOBS["source_state"] + " - " + HOBS["destination_state"]
HOBS.drop(["source_state","destination_state"],axis = 1, inplace=True)
HOBS.columns = ["Number_of_trips_between_states","souce-destination_state"]

plt.figure(figsize=(11,5))
sns.barplot(y = HOBS["souce-destination_state"],
            x = HOBS["Number_of_trips_between_states"],)
plt.title("highest number of orders within two states")
plt.show()
```

```
C:\Users\Sweta.Singh\AppData\Local\Temp\ipykernel_9808\603003609.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  HOBS["souce-destination"] = HOBS["source_state"] + " - " + HOBS["destination_state"]
C:\Users\Sweta.Singh\AppData\Local\Temp\ipykernel_9808\603003609.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  HOBS.drop(["source_state","destination_state"],axis = 1, inplace=True)
```



In [721]:
```
HOBS = data.groupby(["source_state","destination_state"])["trip_uuid"].nunique().sort_values(ascending=False).reset_index()
HOBS = HOBS[HOBS["source_state"]!=HOBS["destination_state"]].head(20)

HOBS["souce-destination"] = HOBS["source_state"] + " - " + HOBS["destination_state"]
HOBS.drop(["source_state","destination_state"],axis = 1, inplace=True)
HOBS.columns = ["Number_of_trips_between_states","souce-destination_state"]

plt.figure(figsize=(11,5))
sns.barplot(y = HOBS["souce-destination_state"],
            x = HOBS["Number_of_trips_between_states"],)
plt.title("highest number of orders between two states")
plt.show()
```



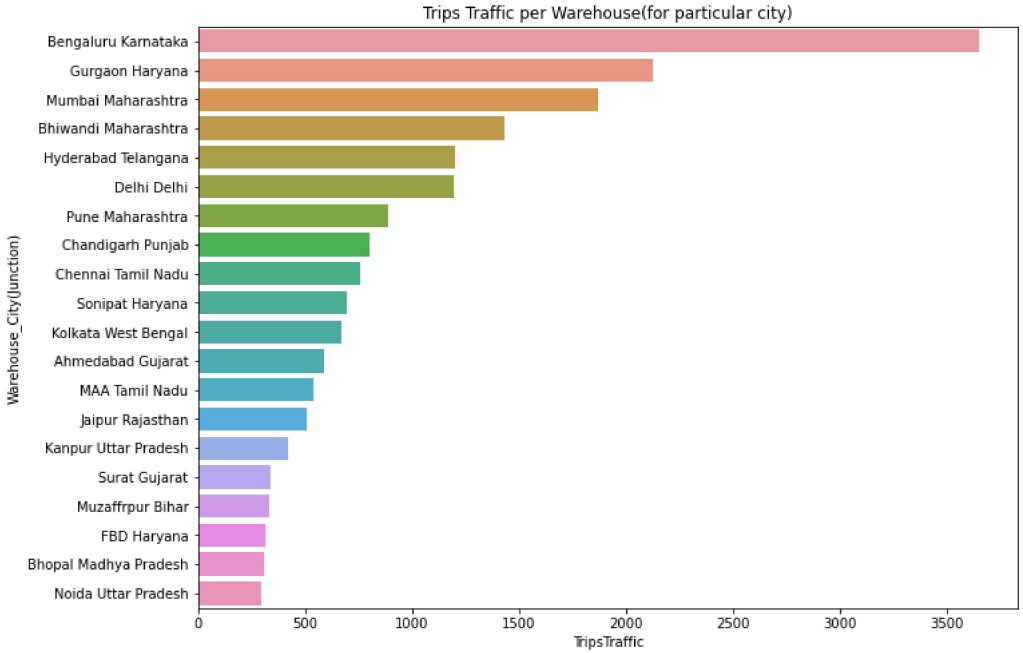
- The busiest route is Delhi to Haryana, with over 400 trips, followed by other busy routes such as Haryana to Uttar Pradesh, Chandigarh to Punjab, and Delhi to Uttar Pradesh. Within states, Maharashtra, Karnataka, and Tamil Nadu are some of the states that have more than 1000 trips."

The top 20 warehouses with the highest volume of traffic

In [722]:
```
destination_traffic = data.groupby(["destination_city_state"])["trip_uuid"].nunique().reset_index()
source_traffic = data.groupby(["source_city_state"])["trip_uuid"].nunique().reset_index()
transactions = source_traffic.merge(destination_traffic,
                                    left_on="source_city_state"
                                    ,right_on="destination_city_state")
transactions.columns = ["source_city_state","#Trips_s","destination_city_state","#Trips_d"]
transactions["TripsTraffic"] = transactions["#Trips_s"]+transactions["#Trips_d"]
transactions.drop(["#Trips_s","#Trips_d","destination_city_state"],axis = 1,inplace=True)
transactions.columns = ["Warehouse_City(Junction)","TripsTraffic"]
```

In [723]:
```
T = transactions.sort_values(by=["TripsTraffic"],ascending=False).head(20)
```

In [724]:
```
plt.figure(figsize=(11,8))
sns.barplot(y = T["Warehouse_City(Junction)"],
            x = T["TripsTraffic"])
plt.title("Trips Traffic per Warehouse(for particular city)")
plt.show()
```



- The top 20 busiest warehouses, as measured by trip traffic at the junction, are:

    - Bengaluru Karnataka,
    - Gurgaon Haryana,
    - Mumbai Maharashtra,
    - Bhiwandi Maharashtra,
    - Hyderabad Telangana,
    - Delhi Delhi,
    - Pune Maharashtra,
    - Chandigarh Punjab,
    - Chennai Tamil Nadu,
    - Sonipat Haryana,
    - Kolkata West Bengal,
    - Ahmedabad Gujarat,
    - MAA Tamil Nadu,
    - Jaipur Rajasthan,
    - Kanpur Uttar Pradesh,
    - Surat Gujarat,
    - Muzaffrpur Bihar,
    - FBD Haryana,
    - Bhopal Madhya Pradesh,
    - Noida Uttar Pradesh.

In [725]:
```
trip_records.groupby(["source_state","destination_state"])["trip_uuid"].count().sort_values(ascending=False).head(15).reset_index()
```

Out[725]:

	source_state	destination_state	trip_uuid
0	Maharashtra	Maharashtra	2085
1	Karnataka	Karnataka	2002
2	Tamil Nadu	Tamil Nadu	996
3	Haryana	Haryana	771
4	Telangana	Telangana	627
5	Gujarat	Gujarat	624
6	West Bengal	West Bengal	610
7	Uttar Pradesh	Uttar Pradesh	529
8	Rajasthan	Rajasthan	400
9	Delhi	Haryana	385
10	Andhra Pradesh	Andhra Pradesh	344
11	Punjab	Punjab	342
12	Bihar	Bihar	330
13	Haryana	Delhi	307
14	Hub Maharashtra	Maharashtra	300

Understanding Features and Feature Engineering:

Analyzing records for one particulat trip id:

In [559]:
```
data[data["trip_uuid"]=="trip-153741093647649320"]
```

Out[559]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end_time	start_scan_to_er
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
5	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	2018-09-20 06:36:55.627764	
6	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	2018-09-20 06:36:55.627764	
7	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	2018-09-20 06:36:55.627764	
8	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	2018-09-20 06:36:55.627764	
9	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	2018-09-20 06:36:55.627764	

- This trip record is divided into various segments, each representing a different drop location.
- From the record, it can be seen that the trip includes stops at different warehouses, which are designated as the source and destination centers.
- The od-start-time and od-end-time are the times when the specific trip in the record started and ended, respectively.
- The start-scan-to-end-scan time is the duration of the trip while it was being scanned, from start to end.
- The start-scan-to-end-scan time is given as a cumulative total, not per individual trip segment.
- A trip cut-off of "False" in the record indicates a change in the trip from one warehouse to another, between the source and destination.
- The actual-time is the total time it takes to complete the delivery from the source to the destination, and is given as a cumulative total.
- The osrm-time is calculated using an open-source routing engine which computes the quickest route between points on a map and gives the time taken. The osrm-distance is the shortest distance between the points, both given as cumulative totals.
- The actual-distance-to-destination is the total distance traveled between warehouses during the trip, given cumulatively.
- Every time the trip cut-off is "False", the distance count starts from the beginning.
- The segment actual time is the actual time taken between two stops during the trip, given for each segment. It represents the time taken for a subset of the package delivery.
- The segment osrm time is the time calculated by the open-source routing engine for each segment of the trip, representing a subset of the package delivery.

Extracting Features like city - place - code -state from source and destination name columns

In [560]:
```
data["source_city"]=data["source_name"].str.split(" ",n=1,expand=True)[0].str.split("_",n=1,expand=True)[0]
data["source_state"]=data["source_name"].str.split(" ",n=1,expand=True)[1].str.replace("(","").str.replace(")","")
data["destination_city"]=data["destination_name"].str.split(" ",n=1,expand=True)[0].str.split("_",n=1,expand=True)[0]
data["destination_state"]=data["destination_name"].str.split(" ",n=1,expand=True)[1].str.replace("(","").str.replace(")","")
```

```
C:\Users\Sweta.Singh\AppData\Local\Temp\ipykernel_9808\486606178.py:2: FutureWarning: The default value of regex will change from True to False in a future version. In additi
on, single character regular expressions will *not* be treated as literal strings when regex=True.
  data["source_state"]=data["source_name"].str.split(" ",n=1,expand=True)[1].str.replace("(","").str.replace(")","")
C:\Users\Sweta.Singh\AppData\Local\Temp\ipykernel_9808\486606178.py:4: FutureWarning: The default value of regex will change from True to False in a future version. In additi
on, single character regular expressions will *not* be treated as literal strings when regex=True.
  data["destination_state"]=data["destination_name"].str.split(" ",n=1,expand=True)[1].str.replace("(","").str.replace(")","")
```

In [561]:
```
data["source_place"]=data["source_name"].str.split("_",n=2,expand=True)[1]
data["destination_place"]=data["destination_name"].str.split("_",n=2,expand=True)[1]
```

In [562]:
```
data["source_pincode"]=data["source_center"].apply(lambda x:x[3:9])
data["destination_pincode"]=data["destination_center"].apply(lambda x:x[3:9])
```

In [563]:
```
data
```

Out[563]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end_time	start_scan_to_
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
...	...	...	...	...	...	...	...	...	...	...	...	...
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	

144867 rows × 35 columns

Time_taken_between_odstart_and_od_end V/S start_scan_to_end_scan:

In [564]:
```
data["Time_taken_between_odstart_and_od_end"]=((data["od_end_time"]-data["od_start_time"])/pd.Timedelta(1,unit="hour"))
```

In [565]:
```
data["Time_taken_between_odstart_and_od_end"]
```

Out[565]:
```
0         1.436894
1         1.436894
2         1.436894
3         1.436894
4         1.436894
             ...
144862    7.128106
144863    7.128106
144864    7.128106
144865    7.128106
144866    7.128106
Name: Time_taken_between_odstart_and_od_end, Length: 144867, dtype: float64
```

Converting given time duration features into hours .

- start_scan_to_end_scan
- actual_time
- osrm_time
- segment_actual_time
- segment_osrm_time

In [566]:
```
data["start_scan_to_end_scan"]=data["start_scan_to_end_scan"]/60
data["actual_time"]=data["actual_time"]/60
data["osrm_time"]=data["osrm_time"]/60
data["segment_actual_time"]=data["segment_actual_time"]/60
```

In [567]:
```
data
```

Out[567]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end_time	start_scan_to_
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
...	...	...	...	...	...	...	...	...	...	...	...	...
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	

144867 rows × 36 columns

In [568]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 36 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   data                                  144867 non-null  object
 1   trip_creation_time                    144867 non-null  datetime64[ns]
 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  datetime64[ns]
 10  od_end_time                           144867 non-null  datetime64[ns]
 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
 24  trip_creation_day                     144867 non-null  object
 25  trip_creation_month                   144867 non-null  object
 26  trip_creation_year                    144867 non-null  int64
 27  source_city                           144574 non-null  object
 28  source_state                          144574 non-null  object
 29  destination_city                      144606 non-null  object
 30  destination_state                     144606 non-null  object
 31  source_place                          142467 non-null  object
 32  destination_place                     142165 non-null  object
 33  source_pincode                        144867 non-null  object
 34  destination_pincode                   144867 non-null  object
 35  Time_taken_between_odstart_and_od_end  144867 non-null  float64
dtypes: bool(1), datetime64[ns](3), float64(11), int64(2), object(19)
memory usage: 38.8+ MB
```

In [569]:
```
data.isna().sum()
```

Out[569]:
```
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
trip_creation_day                         0
trip_creation_month                       0
trip_creation_year                        0
source_city                             293
source_state                            293
destination_city                        261
destination_state                       261
source_place                           2400
destination_place                      2702
source_pincode                            0
destination_pincode                       0
Time_taken_between_odstart_and_od_end     0
dtype: int64
```

In [570]:
```
data.shape
```

Out[570]:
```
(144867, 36)
```

Data Cleaning

In [571]:
```
data["source_state"]=data["source_state"].replace({"Goa Goa":"Goa",
                            "Layout PC Karnataka":"Karnataka",
                            "Vadgaon Sheri DPC Maharashtra":"Maharashtra",
                            "Pashan DPC Maharashtra":"Maharashtra",
                            "City Madhya Pradesh":"Madhya Pradesh",
                            "02_DPC Uttar Pradesh":"Uttar Pradesh",
                            "Nagar_DC Rajasthan":"Rajasthan",
                            "Alipore_DPC West Bengal":"West Bengal",
                             "Mandakni Madhya Pradesh":"Madhya Pradesh",
                             "West _Dc Maharashtra":"Maharashtra",
                             "DC Rajasthan":"Rajasthan",
                             "MP Nagar Madhya Pradesh":"Madhya Pradesh",
                             "Antop Hill Maharashtra":"Maharashtra",
                             "Avenue_DPC West Bengal":"West Bengal",
                             "Nagar Uttar Pradesh":"Uttar Pradesh",
                             "Balaji Nagar Maharashtra":"Maharashtra",
                             "Kothanur_L Karnataka":"Karnataka",
                             "Rahatani DPC Maharashtra":"Maharashtra",
                             "Mahim Maharashtra":"Maharashtra",
                             "DC Maharashtra":"Maharashtra",
                             "_NAD Andhra Pradesh":"Andhra Pradesh",
                                                  })
```

In [572]:
```
data["destination_state"] = data["destination_state"].replace({"Goa Goa":"Goa",
                            "Layout PC Karnataka":"Karnataka",
                            "Vadgaon Sheri DPC Maharashtra":"Maharashtra",
                            "Pashan DPC Maharashtra":"Maharashtra",
                            "City Madhya Pradesh":"Madhya Pradesh",
                            "02_DPC Uttar Pradesh":"Uttar Pradesh",
                            "Nagar_DC Rajasthan":"Rajasthan",
                            "Alipore_DPC West Bengal":"West Bengal",
                             "Mandakni Madhya Pradesh":"Madhya Pradesh",
                             "West _Dc Maharashtra":"Maharashtra",
                             "DC Rajasthan":"Rajasthan",
                             "MP Nagar Madhya Pradesh":"Madhya Pradesh",
                             "Antop Hill Maharashtra":"Maharashtra",
                             "Avenue_DPC West Bengal":"West Bengal",
                             "Nagar Uttar Pradesh":"Uttar Pradesh",
                             "Balaji Nagar Maharashtra":"Maharashtra",
                             "Kothanur_L Karnataka":"Karnataka",
                             "Rahatani DPC Maharashtra":"Maharashtra",
                             "Mahim Maharashtra":"Maharashtra",
                             "DC Maharashtra":"Maharashtra",
                             "_NAD Andhra Pradesh":"Andhra Pradesh",
                            "Delhi Delhi":"Delhi",
                            "West_Dc Maharashtra":"Maharashtra",
                            "Hub Maharashtra":"Maharashtra"
                                                  })
```

In [573]:
```
data["destination_city"].replace({
    "del":"Delhi"
},inplace=True)
data["source_city"].replace({
    "del":"Delhi"
},inplace=True)
data["source_city"].replace({
    "Bangalore":"Bengaluru"
        },inplace=True)
data["destination_city"].replace({
    "Bangalore":"Bengaluru"
        },inplace=True)
data["destination_city"].replace({
    "AMD":"Ahmedabad"
        },inplace=True)
data["destination_city"].replace({
    "Amdavad":"Ahmedabad"
        },inplace=True)
data["source_city"].replace({
    "AMD":"Ahmedabad"
        },inplace=True)
data["source_city"].replace({
    "Amdavad":"Ahmedabad"
        },inplace=True)
```

In [574]:
```
data["source_city_state"]=data["source_city"]+" "+data["source_state"]
data["destination_city_state"]=data["destination_city"]+" "+data["destination_state"]
```

In [575]:
```
data["source_city_state"].nunique()
```

Out[575]: 1249

In [576]:
```
data["destination_city_state"].nunique()
```

Out[576]: 1242

In [577]:
```
data["source_state"].nunique()
```

Out[577]: 33

In [578]:
```
data["destination_state"].nunique()
```

Out[578]: 32

In [579]:
```
data
```

Out[579]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end_time	start_scan_to_
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
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	2018-09-20 04:47:45.236797	
...	...	...	...	...	...	...	...	...	...	...	...	...
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	

144867 rows × 38 columns

- Delhivery covers nearly 1250 cities and all states across India.

In [580]:
```
data1=data.copy()
```

In [581]:
```
data1.columns
```

Out[581]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'is_cutoff', 'cutoff_factor',
       'cutoff_timestamp', 'actual_distance_to_destination', 'actual_time',
       'osrm_time', 'osrm_distance', 'factor', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'segment_factor',
       'trip_creation_day', 'trip_creation_month', 'trip_creation_year',
       'source_city', 'source_state', 'destination_city', 'destination_state',
       'source_place', 'destination_place', 'source_pincode',
       'destination_pincode', 'Time_taken_between_odstart_and_od_end',
       'source_city_state', 'destination_city_state'],
      dtype='object')
```

In [582]:
```
##data1[["source_city","source_state","destination_city","destination_state","source_city_state","destination_city_state"]].fillna()
```

In [583]:
```
data1.drop(['source_center',"source_name","destination_center","destination_name","cutoff_timestamp"],axis = 1,inplace=True)
```

In [584]:
```
data1.drop(["od_end_time","od_start_time"],axis = 1 , inplace=True)
```

In [585]:
```
data1
```

Out[585]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	start_scan_to_end_scan	is_cutoff	cutoff_factor	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	fact
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	True	9	10.435660	0.233333	0.183333	11.9653	1.2727
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	True	18	18.936842	0.400000	0.333333	21.7243	1.2000
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	True	27	27.637279	0.666667	0.466667	32.5395	1.4285
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	True	36	36.118028	1.033333	0.666667	45.5620	1.5500
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	False	39	39.386040	1.133333	0.733333	54.2181	1.5454
...	...	...	...	...	...	...	...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	True	45	45.258278	1.566667	1.000000	67.9280	1.5666
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	True	54	54.092531	2.000000	1.266667	85.6829	1.5789
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	True	63	66.163591	2.333333	1.466667	97.0933	1.5909
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	True	72	73.680667	2.633333	1.633333	111.2709	1.6122
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	False	70	70.039010	7.100000	1.583333	88.7319	4.4842

144867 rows × 31 columns

Aggregating Data

In [586]:
```
actual_time = data1.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["actual_time"].max().reset_index().groupby("trip_uuid")["actual_time"].sum().reset_index()
segment_osrm_time = data1[["trip_uuid","segment_osrm_time"]].groupby("trip_uuid")["segment_osrm_time"].sum().reset_index()
segment_actual_time = data1.groupby("trip_uuid")["segment_actual_time"].sum().reset_index()
osrm_time = data1.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["osrm_time"].max().reset_index().groupby("trip_uuid")["osrm_time"].sum().reset_index()
Time_taken_between_odstart_and_od_end = data1.groupby("trip_uuid")["Time_taken_between_odstart_and_od_end"].unique().reset_index()
Time_taken_between_odstart_and_od_end["Time_taken_between_odstart_and_od_end"] = time_taken_btwn_odstart_and_od_end["Time_taken_between_odstart_and_od_end"].apply(sum)
start_scan_to_end_scan = ((data1.groupby("trip_uuid")["start_scan_to_end_scan"].unique())).reset_index()
start_scan_to_end_scan["start_scan_to_end_scan"] = start_scan_to_end_scan["start_scan_to_end_scan"].apply(sum)

osrm_distance = data1.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["osrm_distance"].max().reset_index().groupby("trip_uuid")["osrm_distance"].sum().reset_index()
actual_distance_to_destination = data1.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["actual_distance_to_destination"].max().reset_index().groupby("trip_uuid")["actual_distance_to_destination"].sum().reset_index()
segment_osrm_distance = data1[["trip_uuid",
                               "segment_osrm_distance"]].groupby("trip_uuid")["segment_osrm_distance"].sum().reset_index()
```

Hypothesis Tests for time durations and distance related features

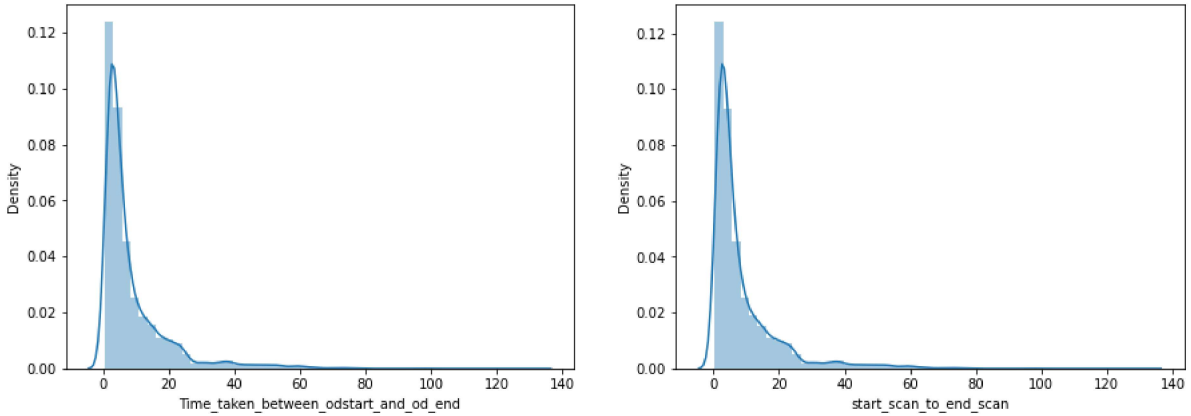
Analysing TimeTaken Between OdStart and OdEnd time & StartScanToEndScan

- H0: Mean of time taken betweenn trip end ans start time = Mean of start and end scan time
- Ha: Mean of time taken betweenn trip end ans start time != Mean of start and end scan time

In [587]:
```
plt.figure(figsize=(15,5))
plt.subplot(121)
sns.distplot((Time_taken_between_odstart_and_od_end["Time_taken_between_odstart_and_od_end"]))
plt.subplot(122)
sns.distplot((start_scan_to_end_scan["start_scan_to_end_scan"]))
```

```
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[587]: <AxesSubplot:xlabel='start_scan_to_end_scan', ylabel='Density'>



KS-test : Evaluating the similarity of the distributions.

In [589]:
```
import scipy.stats as stats
```

In [590]:
```
stats.ks_2samp(Time_taken_between_odstart_and_od_end["Time_taken_between_odstart_and_od_end"],
               start_scan_to_end_scan["start_scan_to_end_scan"])
```

Out[590]: KstestResult(statistic=0.004184382803536474, pvalue=0.9994337058695081)

In [591]:
```
for i in range(5):
    print(stats.ttest_ind((Time_taken_between_odstart_and_od_end["Time_taken_between_odstart_and_od_end"].sample(3000))
                ,(start_scan_to_end_scan["start_scan_to_end_scan"].sample(3000))))
```

```
Ttest_indResult(statistic=0.4204067260058777, pvalue=0.6742034157723639)
Ttest_indResult(statistic=0.6502269388597826, pvalue=0.5155705458933009)
Ttest_indResult(statistic=1.4140721449111169, pvalue=0.15739262277226873)
Ttest_indResult(statistic=-0.8512184027874402, pvalue=0.39468202333110614)
Ttest_indResult(statistic=1.1273340886591856, pvalue=0.2596463503478458)
```

- Based on the results of the Kolmogorov–Smirnov test, with a p-value of 0.9943, we can deduce that the distributions of time taken between order start and end, and start scan to end scan are highly similar.
- Additionally, the two-sample t-test confirms that the average time taken between order start and end for the population is equal to the average start scan to end scan for the population.

checking mean and standard deviation for timetaken and scan times

In [593]:
```
Time_taken_between_odstart_and_od_end["Time_taken_between_odstart_and_od_end"].mean(),Time_taken_between_odstart_and_od_end["Time_taken_between_odstart_and_od_end"].std()
# (8.861857235305067, 10.981665759990623)
```

Out[593]: (8.861857235305067, 10.981665759990623)

In [594]:
```
start_scan_to_end_scan["start_scan_to_end_scan"].mean(),start_scan_to_end_scan["start_scan_to_end_scan"].std()
# (8.835777597804325, 10.97628639143973)
```

Out[594]: (8.835777597804325, 10.97628639143973)

- The variance and means of the scan time and trip start and end time taken are similar to each other.

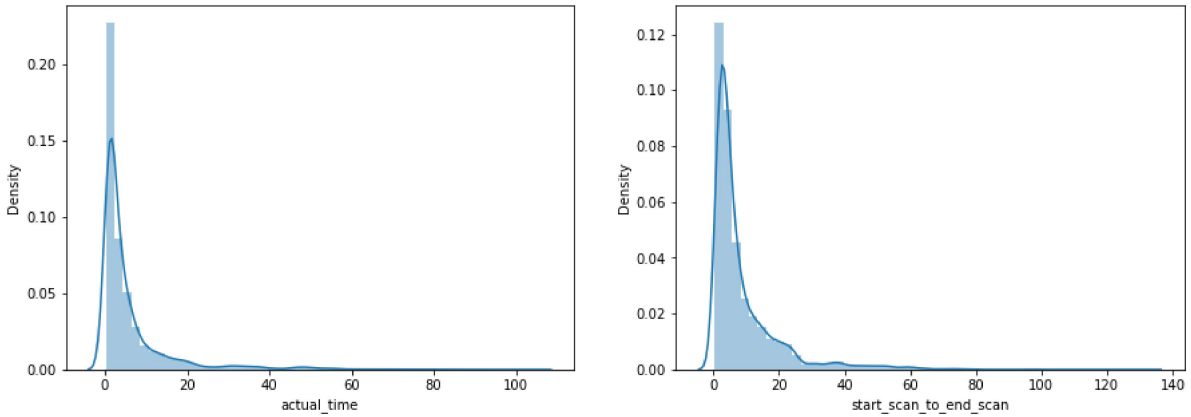
Analysing Actual Time taken to complete the delivery and start-scan-end-scan

- H0: Mean of start and end scan time <= Mean of Actual time taken to complete delivery
- Ha: Mean of start and end scan time > Mean of Actual time taken to complete delivery

In [596]:
```
plt.figure(figsize=(15,5))
plt.subplot(121)
sns.distplot((actual_time["actual_time"]))
plt.subplot(122)
sns.distplot((start_scan_to_end_scan["start_scan_to_end_scan"]))
```

```
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[596]: <AxesSubplot:xlabel='start_scan_to_end_scan', ylabel='Density'>



In [597]:
```
stats.ks_2samp(actual_time["actual_time"],start_scan_to_end_scan["start_scan_to_end_scan"])
```

Out[597]: KstestResult(statistic=0.27387460349598436, pvalue=0.0)

In [598]:
```
for i in range(7):
    print(stats.ttest_ind((actual_time["actual_time"].sample(3000))
                          ,(start_scan_to_end_scan["start_scan_to_end_scan"].sample(3000)),alternative="less"))
```

```
Ttest_indResult(statistic=-11.018917576127023, pvalue=2.8683931169328064e-28)
Ttest_indResult(statistic=-9.337886512595215, pvalue=6.7740123475045206e-21)
Ttest_indResult(statistic=-11.739056314858257, pvalue=8.863482053916008e-32)
Ttest_indResult(statistic=-11.289602321757089, pvalue=1.455012334097118e-29)
Ttest_indResult(statistic=-11.558361818781144, pvalue=7.046939710534293e-31)
Ttest_indResult(statistic=-12.275051707937202, pvalue=1.5833054024463153e-34)
Ttest_indResult(statistic=-11.088623423765833, pvalue=1.3398283629213325e-28)
```

- The results of the KS test indicate that the distributions of actual time and start scan to end scan are not the same.
- Furthermore, the t-test results show that the average actual time for the population is less than the average start scan to end scan for the population.

In [599]:
```
actual_time["actual_time"].mean(),actual_time["actual_time"].std()
```

Out[599]:
```
(5.945176711435117, 9.35554782297388)
```

In [600]:
```
start_scan_to_end_scan["start_scan_to_end_scan"].mean(),start_scan_to_end_scan["start_scan_to_end_scan"].std()
```

Out[600]:
```
(8.835777597804325, 10.97628639143973)
```

Analysing Actual Time & TimeTaken between start and end trip time.

- H0: Mean of Actual time taken to complete delivery = Mean of time taken betweenn trip end and start time
- Ha: Mean of Actual time taken to complete delivery != Mean of time taken betweenn trip end and start time

In [601]:
```
stats.ks_2samp(actual_time["actual_time"],Time_taken_between_odstart_and_od_end["Time_taken_between_odstart_and_od_end"])
```

Out[601]:
```
KstestResult(statistic=0.2765067152594992, pvalue=0.0)
```

In [602]:
```
for i in range(5):
    print(stats.ttest_ind((actual_time["actual_time"].sample(1000)),
    (Time_taken_between_odstart_and_od_end["Time_taken_between_odstart_and_od_end"].sample(1000))))
```

```
Ttest_indResult(statistic=-5.926756525949184, pvalue=3.6302781705386124e-09)
Ttest_indResult(statistic=-7.578313601278857, pvalue=5.3227840234202814e-14)
Ttest_indResult(statistic=-6.567472930258612, pvalue=6.506125361991376e-11)
Ttest_indResult(statistic=-6.865579860741747, pvalue=8.808209489821942e-12)
Ttest_indResult(statistic=-7.239168146033689, pvalue=6.410029091181238e-13)
```

- The results of the Kolmogorov-Smirnov test and the two-sample t-test suggest that the population means of "Actual time taken to complete delivery" and "time taken between order start and end" are not equal.

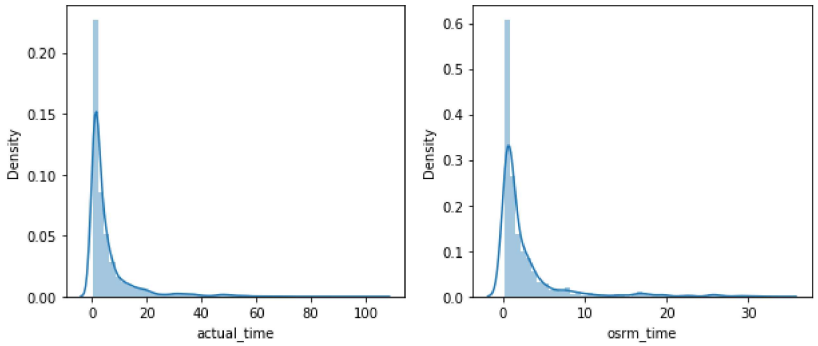
Analysing Actual Time taken to complete delivery from source to destination hub & OSRM measured time

- H0: Mean of OSRM time >= Mean of Actual time taken to complete delivery
- Ha: Mean of OSRM time < Mean of Actual time taken to complete delivery

In [603]:
```
plt.figure(figsize=(10,4))
plt.subplot(121)
sns.distplot(((actual_time["actual_time"])))
plt.subplot(122)
sns.distplot(((osrm_time["osrm_time"])))
```

```
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[603]: <AxesSubplot:xlabel='osrm_time', ylabel='Density'>



In [604]:
```
stats.ks_2samp(actual_time["actual_time"],
               osrm_time["osrm_time"])
```

Out[604]: KstestResult(statistic=0.2945265573327934, pvalue=0.0)

In [605]:
```
for i in range(5):
    print(stats.ttest_ind(actual_time["actual_time"].sample(5000),
               osrm_time["osrm_time"].sample(5000),alternative='greater'))
```

```
Ttest_indResult(statistic=23.017716268646225, pvalue=1.399727347623734e-114)
Ttest_indResult(statistic=21.100164333813876, pvalue=4.991019679454251e-97)
Ttest_indResult(statistic=21.67061281221638, pvalue=4.177078429042716e-102)
Ttest_indResult(statistic=22.46496336492036, pvalue=2.221944939896252e-109)
Ttest_indResult(statistic=21.936826732996103, pvalue=1.619641484601746e-104)
```

- The two-sample t-test results indicate that the population mean of actual delivery time from source to warehouse and the population mean estimated time are not equal.

- The actual delivery time exceeds the estimated time provided by OSRM.for delivery

In [606]:
```
actual_time["actual_time"].mean(),actual_time["actual_time"].std()
```

Out[606]: (5.945176711435117, 9.35554782297388)

In [607]:
```
osrm_time["osrm_time"].mean(),osrm_time["osrm_time"].std()
```

Out[607]: (2.697313896200314, 4.537654251845703)

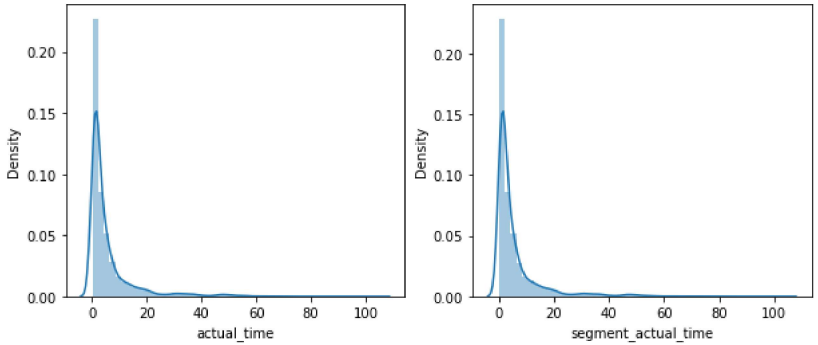
Analysing Actual Time taken to complete delivery from source to destination hub & Segment Actual Time

- H0: Actual time = segment actual time
- Ha: Actual time != segment actual time

In [608]:
```
plt.figure(figsize=(10,4))
plt.subplot(121)
sns.distplot(((actual_time["actual_time"])))
plt.subplot(122)
sns.distplot(((segment_actual_time["segment_actual_time"])))
```

```
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[608]:
```
<AxesSubplot:xlabel='segment_actual_time', ylabel='Density'>
```



In [609]:
```
for i in range(7):
    print(stats.ttest_ind((actual_time["actual_time"].sample(3000)),
                (segment_actual_time["segment_actual_time"].sample(3000))))
```

```
Ttest_indResult(statistic=0.13538180539800174, pvalue=0.8923145517714413)
Ttest_indResult(statistic=0.03794730440386378, pvalue=0.9697309595959158)
Ttest_indResult(statistic=-0.3322445607550838, pvalue=0.7397162081832163)
Ttest_indResult(statistic=-0.6052858485247405, pvalue=0.5450119825455728)
Ttest_indResult(statistic=-0.18699386576038485, pvalue=0.8516717670151503)
Ttest_indResult(statistic=1.191506202262887, pvalue=0.23350206748288724)
Ttest_indResult(statistic=0.485134096400858, pvalue=0.6275989448861131)
```

- The results of the two-sample t-test suggest that the population averages for "Actual Time taken to complete delivery trip" and "segment actual time" are equal.

In [610]:
```
actual_time["actual_time"].mean(),actual_time["actual_time"].std()
```

Out[610]:
```
(5.945176711435117, 9.35554782297388)
```

In [611]:
```
segment_actual_time["segment_actual_time"].mean(),segment_actual_time["segment_actual_time"].std()
```

Out[611]:
```
(5.898204764797215, 9.270799413152762)
```

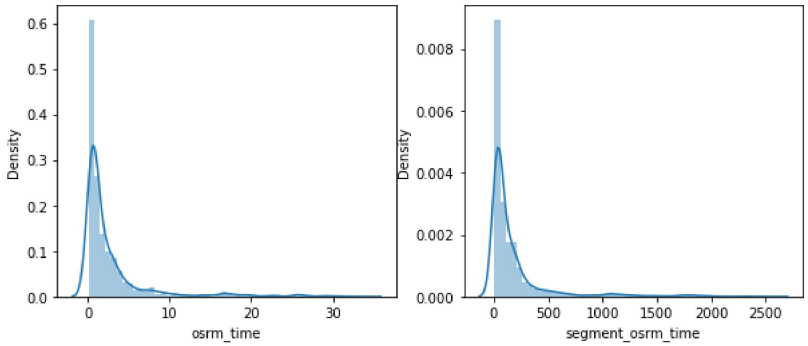
Analysing osrm Time & segment-osrm-time

- H0: segment actual time <= OSRM time
- Ha: segment actual time > OSRM time

In [612]:
```
plt.figure(figsize=(10,4))
plt.subplot(121)
sns.distplot(((osrm_time["osrm_time"])))
plt.subplot(122)
sns.distplot(((segment_osrm_time["segment_osrm_time"])))
```

```
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[612]: <AxesSubplot:xlabel='segment_osrm_time', ylabel='Density'>



In [613]:
```
for i in range(7):
    print(stats.ttest_ind((osrm_time["osrm_time"].sample(3000)),
                (segment_osrm_time["segment_osrm_time"].sample(3000)),alternative ="less"))
```

```
Ttest_indResult(statistic=-31.210002515802195, pvalue=1.3785643063240867e-198)
Ttest_indResult(statistic=-30.844757800529887, pvalue=2.4254533642499613e-194)
Ttest_indResult(statistic=-31.128067076308476, pvalue=1.2442973944286938e-197)
Ttest_indResult(statistic=-31.54790747472324, pvalue=1.513235057081208e-202)
Ttest_indResult(statistic=-30.980601023098988, pvalue=6.457322005305647e-196)
Ttest_indResult(statistic=-31.3546126411299, pvalue=2.8095996424413523e-200)
Ttest_indResult(statistic=-31.50709219113251, pvalue=4.568701216378814e-202)
```

- The results of the two-sample t-test indicate that the population average of "osrm Time" and "segment-osrm-time" are not equal.
- Population Mean osrm time is less than Population Mean segment osrm time.

In [614]:
```
osrm_time["osrm_time"].mean(),osrm_time["osrm_time"].std()
```

Out[614]: (2.697313896200314, 4.537654251845703)

In [615]:
```
segment_osrm_time["segment_osrm_time"].mean(),segment_osrm_time["segment_osrm_time"].std()
```

Out[615]: (180.94978740635756, 314.54204650157953)

Analysing Distances measures :

Analysing and Visulizing OSRM Estimated distance and Segment-osrm-distance :

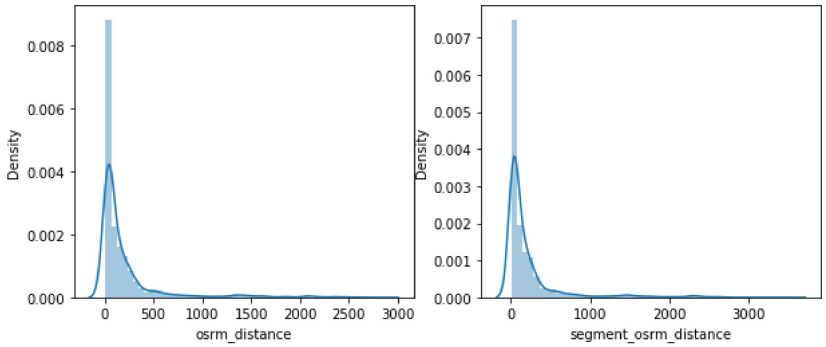
- H0 : Segment OSRM distnace <= OSRM distnace

- Ha : Segment OSRM distnace > OSRM distnace

In [616]:
```
plt.figure(figsize=(10,4))
plt.subplot(121)
sns.distplot(((osrm_distance["osrm_distance"])))
plt.subplot(122)
sns.distplot(((segment_osrm_distance["segment_osrm_distance"])))
```

```
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[616]:
```
<AxesSubplot:xlabel='segment_osrm_distance', ylabel='Density'>
```



In [617]:
```
stats.ks_2samp(osrm_distance["osrm_distance"],segment_osrm_distance["segment_osrm_distance"])
```

Out[617]:
```
KstestResult(statistic=0.03948167645272321, pvalue=1.8042208791084262e-10)
```

In [618]:
```
for i in range(7):
    print(stats.ttest_ind(osrm_distance["osrm_distance"].sample(5000),
                segment_osrm_distance["segment_osrm_distance"].sample(5000),alternative="less"))
```

```
Ttest_indResult(statistic=-1.9332166030074662, pvalue=0.026618860004863534)
Ttest_indResult(statistic=-2.679918450969975, pvalue=0.0036880378491931766)
Ttest_indResult(statistic=-0.9546188670500547, pvalue=0.16989677605741327)
Ttest_indResult(statistic=-2.333641115077961, pvalue=0.009817118356087118)
Ttest_indResult(statistic=-2.030319004731001, pvalue=0.02117527046006828)
Ttest_indResult(statistic=-2.264067030785325, pvalue=0.011795661241168204)
Ttest_indResult(statistic=-1.728884017164358, pvalue=0.04193036265870887)
```

In [619]:
```
osrm_distance["osrm_distance"].mean(),osrm_distance["osrm_distance"].std()
```

Out[619]:
```
(204.83672531551625, 370.74927471335496)
```

In [620]:
```
segment_osrm_distance["segment_osrm_distance"].mean(),segment_osrm_distance["segment_osrm_distance"].std()
```

Out[620]:
```
(223.20116128771042, 416.6283742907418)
```

- The Kolmogorov-Smirnov test indicates that the distributions of "segment osrm distance" and "osrm distance" are different.
- The results of the two-sample one-sided t-test suggest that the average of "osrm distance" for the population is less than the average of "segment osrm distance".

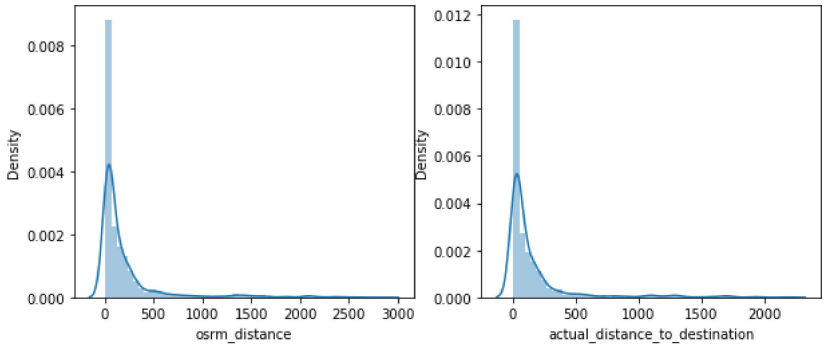
Analysing and Visulizing OSRM Estimated distance and Actual Distance between source and destination warehouse :

- H0 : Mean OSRM distance <= Mean Actual distnace
- Ha : Mean OSRM distance > Mean Actual distnace

In [621]:
```
plt.figure(figsize=(10,4))
plt.subplot(121)
sns.distplot(((osrm_distance["osrm_distance"])))
plt.subplot(122)
sns.distplot(((actual_distance_to_destination["actual_distance_to_destination"])))
```

```
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Sweta.Singh\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[621]: <AxesSubplot:xlabel='actual_distance_to_destination', ylabel='Density'>



In [622]:
```
stats.ks_2samp(osrm_distance["osrm_distance"],actual_distance_to_destination["actual_distance_to_destination"])
```

Out[622]: KstestResult(statistic=0.11837753931295136, pvalue=6.578385372142345e-91)

In [623]:
```
for i in range(5):
    print(stats.ttest_ind(osrm_distance["osrm_distance"].sample(5000),
                actual_distance_to_destination["actual_distance_to_destination"].sample(5000),alternative="greater"))
```

```
Ttest_indResult(statistic=5.579943675267602, pvalue=1.2343310873500265e-08)
Ttest_indResult(statistic=4.451251363529931, pvalue=4.314812658270748e-06)
Ttest_indResult(statistic=4.804018716871185, pvalue=7.888646231541627e-07)
Ttest_indResult(statistic=5.721111869596873, pvalue=5.443587959598414e-09)
Ttest_indResult(statistic=5.913169003916791, pvalue=1.7328899567226875e-09)
```

- The results of the left-tailed t-test indicate that the population mean of the OSRM estimated distance is greater than the actual distance from source to destination warehouse.

In [624]:
```
osrm_distance["osrm_distance"].mean(),osrm_distance["osrm_distance"].std()
```

Out[624]: (204.83672531551625, 370.74927471335496)

In [625]:
```
actual_distance_to_destination["actual_distance_to_destination"].mean(),actual_distance_to_destination["actual_distance_to_destination"].std()
```

Out[625]: (164.4733217454422, 305.5408288910492)

Combining all the numerical fields based on TripID.

In [626]:
```
distance = segment_osrm_distance.merge(actual_distance_to_destination.merge(osrm_distance,
                                                                            on="trip_uuid"),
                                       on="trip_uuid")
```

In [630]:
```
time = segment_osrm_time.merge(osrm_time.merge(segment_actual_time.merge(actual_time.merge(Time_taken_between_odstart_and_od_end.merge(start_scan_to_end_scan,
                                        on="trip_uuid",
                                        ),on="trip_uuid"),on="trip_uuid"),on="trip_uuid"),on="trip_uuid")
```

In [631]:
```
Merge=time.merge(distance,on="trip_uuid")
```

In [632]:
```
Merge
```

Out[632]:

trip_uuid	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance
trip-1041653548748	1008.0	12.383333	25.800000	26.033333	37.668497	37.650000	1320.4733	824.732854	991.3523
trip-1042288605164	65.0	1.133333	2.350000	2.383333	3.026865	3.000000	84.1894	73.186911	85.1110
trip-1043369099517	1941.0	29.016667	55.133333	55.783333	65.572709	65.550000	2545.2678	1932.273969	2372.0852
trip-1046011330457	16.0	0.250000	0.983333	0.983333	1.674916	1.666667	19.8766	17.175274	19.6800
trip-1052974046625	115.0	1.950000	5.666667	5.683333	11.972484	11.950000	146.7919	127.448500	146.7918
...	...	...	...	...	...	...	...	...	...
trip-1095625827784	62.0	1.033333	1.366667	1.383333	4.300482	4.283333	64.8551	57.762332	73.4630
trip-1104386292051	11.0	0.200000	0.350000	0.350000	1.009842	1.000000	16.0883	15.513784	16.0882
trip-1106442901555	88.0	0.900000	4.683333	4.700000	7.035331	7.016667	104.8866	38.684839	63.2841
trip-1115439069069	221.0	3.066667	4.300000	4.400000	5.808548	5.783333	223.5324	134.723836	177.6635
trip-1118270144424	67.0	1.133333	4.566667	4.583333	5.906793	5.883333	80.5787	66.081533	80.5787

10 columns

Combining the information about location and route type with the numerical data for each TripID.

In [633]:
```
city = data.groupby("trip_uuid")[["source_city",
                                  "destination_city"]].aggregate({
        "source_city":pd.unique,
    "destination_city":pd.unique,
})

state = data.groupby("trip_uuid")[["source_state",
                                   "destination_state"]].aggregate({
        "source_state":pd.unique,
    "destination_state":pd.unique,
})

city_state = data.groupby("trip_uuid")[["source_city_state",
                                        "destination_city_state"]].aggregate({
        "source_city_state":pd.unique,
    "destination_city_state":pd.unique,
})

locations = city.merge(city_state.merge(state,on="trip_uuid"
                              ,how="outer"),
           on="trip_uuid",
           how="outer")
```

In [634]:
```
route_type = data1.groupby("trip_uuid")["route_type"].unique().reset_index()
```

In [635]:
```
Merged = route_type.merge(locations.merge(Merge,on="trip_uuid",
           how="outer"),
                 on="trip_uuid",
           how="outer"
                )
```

In [636]:
```
trip_records = Merged.copy()
```

In [637]:
```
trip_records["route_type"] = trip_records["route_type"].apply(lambda x:x[0])
```

In [638]:
```
route_to_merge = data.groupby("trip_uuid")["route_schedule_uuid"].unique().reset_index()
```

In [639]:
```
trip_records = trip_records.merge(route_to_merge,on="trip_uuid",how="outer")
```

In [640]:
```
trip_records["route_schedule_uuid"] = trip_records["route_schedule_uuid"].apply(lambda x:x[0])
```

In [641]:
```
trip_records
```

Out[641]:

_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance	route_schedule_uuid
1008.0	12.383333	25.800000	26.033333	37.668497	37.650000	1320.4733	824.732854	991.3523	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...
65.0	1.133333	2.350000	2.383333	3.026865	3.000000	84.1894	73.186911	85.1110	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...
1941.0	29.016667	55.133333	55.783333	65.572709	65.550000	2545.2678	1932.273969	2372.0852	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...
16.0	0.250000	0.983333	0.983333	1.674916	1.666667	19.8766	17.175274	19.6800	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...
115.0	1.950000	5.666667	5.683333	11.972484	11.950000	146.7919	127.448500	146.7918	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...
...	...	...	...	...	...	...	...	...	...
62.0	1.033333	1.366667	1.383333	4.300482	4.283333	64.8551	57.762332	73.4630	thanos::sroute:8a120994-f577-4491-9e4b-b7e4a14...
11.0	0.200000	0.350000	0.350000	1.009842	1.000000	16.0883	15.513784	16.0882	thanos::sroute:b30e1ec3-3bfa-4bd2-a7fb-3b75769...
88.0	0.900000	4.683333	4.700000	7.035331	7.016667	104.8866	38.684839	63.2841	thanos::sroute:5609c268-e436-4e0a-8180-3db4a74...
221.0	3.066667	4.300000	4.400000	5.808548	5.783333	223.5324	134.723836	177.6635	thanos::sroute:c5f2ba2c-8486-4940-8af6-d1d2a6a...
67.0	1.133333	4.566667	4.583333	5.906793	5.883333	80.5787	66.081533	80.5787	thanos::sroute:412fea14-6d1f-4222-8a5f-a517042...

In [643]:
```
# route_data['source'] = route_data['source'].str.strip("{''}")
```

In [644]:
```
trip_records[trip_records["trip_uuid"]=="trip-153852612674280168"]
```

Out[644]:

	trip_uuid	route_type	source_city	destination_city	source_city_state	destination_city_state	source_state	destination_state	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_tak
14199	trip-153852612674280168	FTL	[nan]	[Mathura]	[nan]	[Mathura Uttar Pradesh]	[nan]	[Uttar Pradesh]	92.0	1.266667	1.433333	1.433333	

In [645]:
```
trip_records.dropna(axis= 0,how = 'any',inplace = True)
```

In [646]:
```
trip_records["source_city"] = trip_records["source_city"].astype("str").str.strip("[]").str.replace("'","")
trip_records["destination_city"] = trip_records["destination_city"].astype("str").str.strip("[]").str.replace("'","")
trip_records["source_city_state"] = trip_records["source_city_state"].astype("str").str.strip("[]").str.replace("'","")
trip_records["destination_city_state"] = trip_records["destination_city_state"].astype("str").str.strip("[]").str.replace("'","")

trip_records["source_state"] = trip_records["source_state"].astype("str").str.strip("[]").str.replace("'","")
trip_records["destination_state"] = trip_records["destination_state"].astype("str").str.strip("[]").str.replace("'","")
```

Verifying the absence of any null values in the Trip Records Data.

In [647]:
```
trip_records.isna().sum()
```

Out[647]:
```
trip_uuid                               0
route_type                              0
source_city                             0
destination_city                        0
source_city_state                       0
destination_city_state                  0
source_state                            0
destination_state                       0
segment_osrm_time                       0
osrm_time                               0
segment_actual_time                     0
actual_time                             0
Time_taken_between_odstart_and_od_end   0
start_scan_to_end_scan                  0
segment_osrm_distance                   0
actual_distance_to_destination          0
osrm_distance                           0
route_schedule_uuid                     0
dtype: int64
```

In [648]:
```
trip_records.loc[trip_records.isnull().any(axis=1)]
```

Out[648]:

trip_uuid	route_type	source_city	destination_city	source_city_state	destination_city_state	source_state	destination_state	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odst

In [649]:
```
trip_records.corr()
```

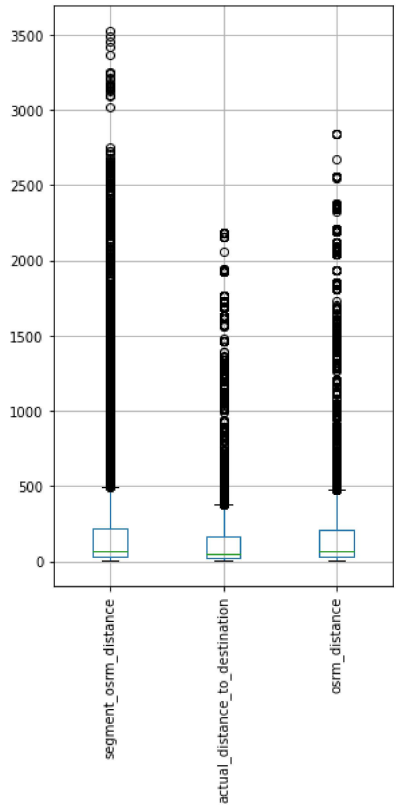
Out[649]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_
segment_osrm_time	1.000000	0.993508	0.953039	0.953800	0.918447	0.918493	0.996092	
osrm_time	0.993508	1.000000	0.957747	0.958613	0.926280	0.926469	0.991848	
segment_actual_time	0.953039	0.957747	1.000000	0.999920	0.961096	0.961107	0.956106	
actual_time	0.953800	0.958613	0.999920	1.000000	0.960958	0.961163	0.956949	
Time_taken_between_odstart_and_od_end	0.918447	0.926280	0.961096	0.960958	1.000000	0.999860	0.919156	
start_scan_to_end_scan	0.918493	0.926469	0.961107	0.961163	0.999860	1.000000	0.919288	
segment_osrm_distance	0.996092	0.991848	0.956106	0.956949	0.919156	0.919288	1.000000	
actual_distance_to_destination	0.987627	0.993556	0.953048	0.954082	0.918373	0.918671	0.993207	
osrm_distance	0.992050	0.997610	0.958341	0.959290	0.924093	0.924368	0.994921	

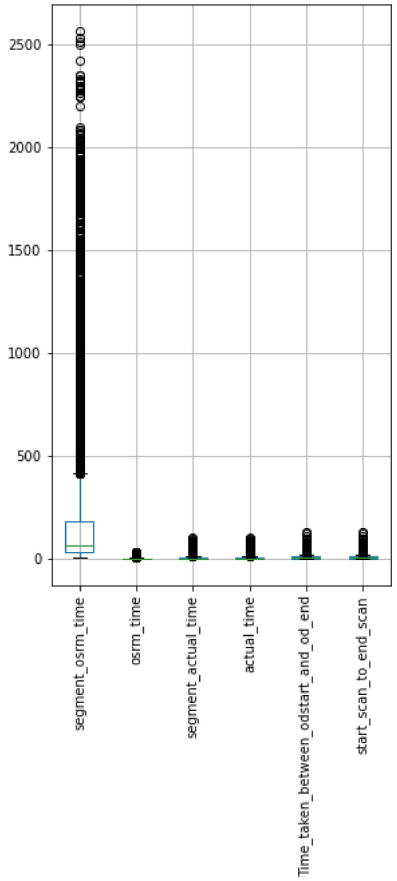
In [650]:
```
trip_records.to_csv("trip_records.csv")
```

Handling Outliers

In [651]:
```
plt.figure(figsize = (10,8))
plt.subplot(121)
trip_records[['segment_osrm_distance', 'actual_distance_to_destination',
       'osrm_distance']].boxplot()
plt.xticks(rotation =90)
plt.show()

# Time_taken_between_odstart_and_od_end
```



In [652]:
```
plt.figure(figsize = (10,8))
plt.subplot(121)
trip_records[['segment_osrm_time', 'osrm_time',
       'segment_actual_time', 'actual_time',
       'Time_taken_between_odstart_and_od_end', 'start_scan_to_end_scan']].boxplot()
plt.xticks(rotation =90)
plt.show()
```



In [653]:
```
outlier_treatment  = trip_records.copy()
```

In [654]:
```
outlier_treatment_num = outlier_treatment[['segment_osrm_time', 'osrm_time',
       'segment_actual_time', 'actual_time',
       'Time_taken_between_odstart_and_od_end', 'start_scan_to_end_scan',
        'segment_osrm_distance', 'actual_distance_to_destination',
        'osrm_distance']]
```

In [655]:
```
outlier_treatment_num
```

Out[655]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance
0	1008.0	12.383333	25.800000	26.033333	37.668497	37.650000	1320.4733	824.732854	991.3523
1	65.0	1.133333	2.350000	2.383333	3.026865	3.000000	84.1894	73.186911	85.1110
2	1941.0	29.016667	55.133333	55.783333	65.572709	65.550000	2545.2678	1932.273969	2372.0852
3	16.0	0.250000	0.983333	0.983333	1.674916	1.666667	19.8766	17.175274	19.6800
4	115.0	1.950000	5.666667	5.683333	11.972484	11.950000	146.7919	127.448500	146.7918
...	...	...	...	...	...	...	...	...	...
14812	62.0	1.033333	1.366667	1.383333	4.300482	4.283333	64.8551	57.762332	73.4630
14813	11.0	0.200000	0.350000	0.350000	1.009842	1.000000	16.0883	15.513784	16.0882
14814	88.0	0.900000	4.683333	4.700000	7.035331	7.016667	104.8866	38.684839	63.2841
14815	221.0	3.066667	4.300000	4.400000	5.808548	5.783333	223.5324	134.723836	177.6635
14816	67.0	1.133333	4.566667	4.583333	5.906793	5.883333	80.5787	66.081533	80.5787

14817 rows × 9 columns

After removing outliers from all numerical features :

In [659]:
```
trip_records_without_outliers = trip_records.loc[outlier_treatment_num[(np.abs(stats.zscore(outlier_treatment_num)) < 3).all(axis=1)].index]
trip_records_without_outliers
```

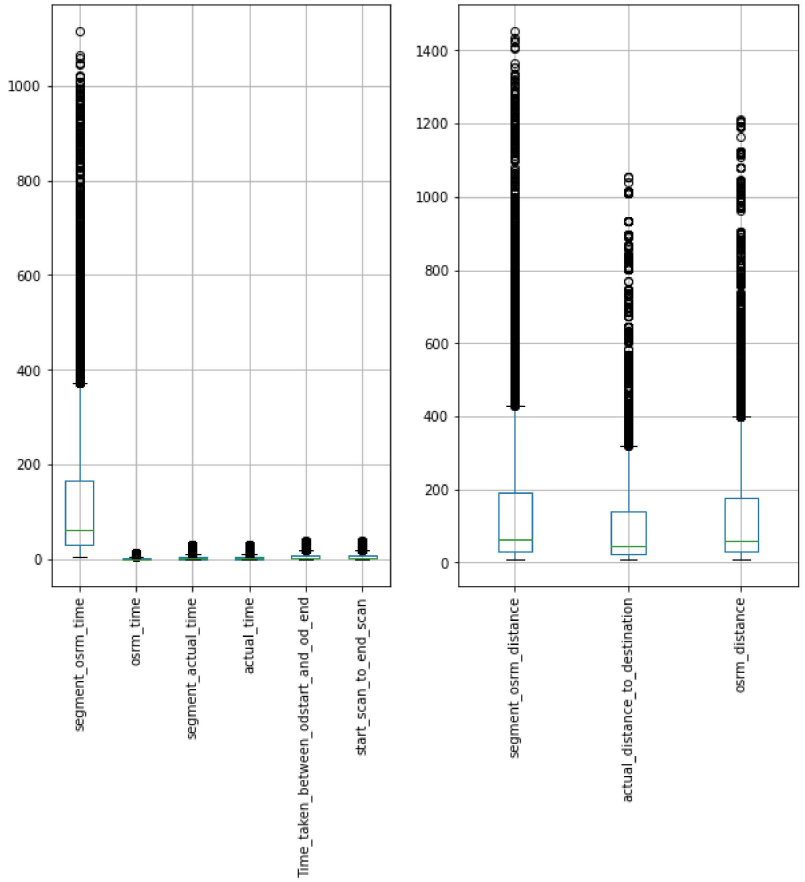
Out[659]:

	trip_uuid	route_type	source_city	destination_city	source_city_state	destination_city_state	source_state	destination_state	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_tal
0	trip-153671041653548748	FTL	Bhopal Kanpur	Kanpur Gurgaon	Bhopal Madhya Pradesh Kanpur Uttar Pradesh	Kanpur Uttar Pradesh Gurgaon Haryana	Madhya Pradesh Uttar Pradesh	Uttar Pradesh Haryana	1008.0	12.383333	25.800000	26.033333	
1	trip-153671042288605164	Carting	Tumkur Doddablpur	Doddablpur Chikblapur	Tumkur Karnataka Doddablpur Karnataka	Doddablpur Karnataka Chikblapur Karnataka	Karnataka	Karnataka	65.0	1.133333	2.350000	2.383333	
3	trip-153671046011330457	Carting	Mumbai	Mumbai	Mumbai Hub Maharashtra	Mumbai Maharashtra	Hub Maharashtra	Maharashtra	16.0	0.250000	0.983333	0.983333	
4	trip-153671052974046625	FTL	Bellary Hospet Sandur	Hospet Sandur Bellary	Bellary Karnataka Hospet Karnataka Sandur Karn...	Hospet Karnataka Sandur Karnataka Bellary Karn...	Karnataka	Karnataka	115.0	1.950000	5.666667	5.683333	
5	trip-153671055416136166	Carting	Chennai	Chennai	Chennai Tamil Nadu	Chennai Tamil Nadu	Tamil Nadu	Tamil Nadu	23.0	0.383333	1.000000	1.016667	
...	...	...	...	...	...	...	...	...	...	...	...	...	
14812	trip-153861095625827784	Carting	Chandigarh	Zirakpur Chandigarh	Chandigarh Punjab Chandigarh Chandigarh	Zirakpur Punjab Chandigarh Punjab	Punjab Chandigarh	Punjab	62.0	1.033333	1.366667	1.383333	
14813	trip-153861104386292051	Carting	FBD	Faridabad	FBD Haryana	Faridabad Haryana	Haryana	Haryana	11.0	0.200000	0.350000	0.350000	
14814	trip-153861106442901555	Carting	Kanpur	Kanpur	Kanpur Uttar Pradesh	Kanpur Uttar Pradesh	Uttar Pradesh	Uttar Pradesh	88.0	0.900000	4.683333	4.700000	
14815	trip-153861115439069069	Carting	Tirunelveli Eral Tirchchndr Thisayanvilai Peik...	Eral Tirchchndr Thisayanvilai Peikulam Tirunel...	Tirunelveli Tamil Nadu Eral Tamil Nadu Tirchch...	Eral Tamil Nadu Tirchchndr Tamil Nadu Thisayan...	Tamil Nadu	Tamil Nadu	221.0	3.066667	4.300000	4.400000	
14816	trip-153861118270144424	FTL	Hospet Sandur	Sandur Bellary	Hospet Karnataka Sandur Karnataka	Sandur Karnataka Bellary Karnataka	Karnataka	Karnataka	67.0	1.133333	4.566667	4.583333	

14160 rows × 18 columns

In [660]:
```
trip_records_without_outliers = trip_records_without_outliers[['trip_uuid','route_type','source_city_state', 'destination_city_state','segment_osrm_time', 'osrm_time',
       'segment_actual_time', 'actual_time',
       'Time_taken_between_odstart_and_od_end', 'start_scan_to_end_scan',
       'segment_osrm_distance', 'actual_distance_to_destination',
       'osrm_distance']]
```

In [661]:
```
plt.figure(figsize = (10,8))
plt.subplot(121)
trip_records_without_outliers[['segment_osrm_time', 'osrm_time',
       'segment_actual_time', 'actual_time',
       'Time_taken_between_odstart_and_od_end', 'start_scan_to_end_scan']].boxplot()
plt.xticks(rotation =90)
plt.subplot(122)
trip_records_without_outliers[['segment_osrm_distance', 'actual_distance_to_destination',
       'osrm_distance']].boxplot()
plt.xticks(rotation =90)
plt.show()
```



Processing Data for One hot encoding

- Combine location details into a single column and re-categorize the data based on the location with the highest number of trips being the top category.

In [662]:
```
trip_records_without_outliers["destination_source_locations"] = trip_records_without_outliers["source_city_state"]+" "+trip_records_without_outliers["destination_city_state"]
trip_records_without_outliers.drop(["source_city_state","destination_city_state"],axis = 1,inplace=True)
```

```
C:\Users\Sweta.Singh\AppData\Local\Temp\ipykernel_9808\4230040880.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  trip_records_without_outliers["destination_source_locations"] = trip_records_without_outliers["source_city_state"]+" "+trip_records_without_outliers["destination_city_state"]
C:\Users\Sweta.Singh\AppData\Local\Temp\ipykernel_9808\4230040880.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  trip_records_without_outliers.drop(["source_city_state","destination_city_state"],axis = 1,inplace=True)
```

In [663]:
```
sc_dc = trip_records_without_outliers.groupby(["destination_source_locations"])["trip_uuid"].nunique().sort_values(ascending= False).reset_index()
```

In [664]:
```
sc_dc
```

Out[664]:

	destination_source_locations	trip_uuid
0	Bengaluru Karnataka Bengaluru Karnataka	1316
1	Bhiwandi Maharashtra Mumbai Maharashtra	437
2	Mumbai Maharashtra Mumbai Maharashtra	330
3	Hyderabad Telangana Hyderabad Telangana	308
4	Gurgaon Haryana Delhi Delhi	237
...	...	...
2520	Hyderabad Telangana Kadthal Telangana Kalwakur...	1
2521	Hyderabad Telangana Kadthal Telangana Kadthal ...	1
2522	Hyderabad Telangana Kadthal Telangana Haliya T...	1
2523	Hyderabad Telangana Kadthal Telangana Haliya T...	1
2524	nan nan	1

2525 rows × 2 columns

In [665]:
```
trip_records.groupby(['source_state','destination_state'])["trip_uuid"].nunique().sort_values(ascending= False).reset_index().head(30)
```

Out[665]:

	source_state	destination_state	trip_uuid
0	Maharashtra	Maharashtra	2085
1	Karnataka	Karnataka	2002
2	Tamil Nadu	Tamil Nadu	996
3	Haryana	Haryana	771
4	Telangana	Telangana	627
5	Gujarat	Gujarat	624
6	West Bengal	West Bengal	610
7	Uttar Pradesh	Uttar Pradesh	529
8	Rajasthan	Rajasthan	400
9	Delhi	Haryana	385
10	Andhra Pradesh	Andhra Pradesh	344
11	Punjab	Punjab	342
12	Bihar	Bihar	330
13	Haryana	Delhi	307
14	Hub Maharashtra	Maharashtra	300
15	Kerala	Kerala	252
16	Madhya Pradesh	Madhya Pradesh	248
17	Delhi	Delhi	189
18	Assam	Assam	174
19	Jharkhand	Jharkhand	143
20	Uttarakhand	Uttarakhand	101
21	Orissa	Orissa	96
22	Haryana	Uttar Pradesh	79
23	Haryana	Rajasthan	68
24	Uttar Pradesh	Delhi	63
25	Haryana	Punjab	60
26	Punjab Chandigarh	Chandigarh Punjab	55
27	Delhi	Uttar Pradesh	49
28	Haryana	Karnataka	47
29	Punjab	Haryana	45

In [666]:
```
def get_cat(H):
    if 0 <= H <= 50:
        return "Category 7"
    elif 51 <= H <= 100:
        return "Category 6"
    elif 101 <= H <= 200:
        return "Category 5"
    elif 201 <= H <= 300:
        return "Category 4"
    elif 301 <= H <= 400:
        return "Category 3"
    elif 401 <= H <= 500:
        return "Category 2"
    else:
        return "Category 1"
```

In [667]:
```
sc_dc["city"]  = pd.Series(map(get_cat,sc_dc["trip_uuid"]))
```

In [668]:
```
sc_dc["city"]
```

Out[668]:
```
0       Category 1
1       Category 2
2       Category 3
3       Category 3
4       Category 4
           ...
2520    Category 7
2521    Category 7
2522    Category 7
2523    Category 7
2524    Category 7
Name: city, Length: 2525, dtype: object
```

In [669]:
```
trip_records_for_encoding = sc_dc.merge(trip_records_without_outliers,
           on="destination_source_locations")
trip_records_for_encoding.drop(["destination_source_locations","trip_uuid_x"],axis = 1,inplace=True)
```

In [670]:
```
trip_records_for_encoding
```

Out[670]:

	city	trip_uuid_y	route_type	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_dis
0	Category 1	trip-153671240266732246	Carting	83.0	0.950000	3.183333	3.233333	4.407028	4.400000	76.1272	
1	Category 1	trip-153671256526597871	Carting	69.0	0.883333	2.666667	2.700000	4.063014	4.050000	59.1472	
2	Category 1	trip-153671590416667638	Carting	71.0	0.966667	3.316667	3.333333	4.076829	4.066667	65.6004	
3	Category 1	trip-153671843215737572	Carting	42.0	0.733333	1.316667	1.316667	4.915934	4.900000	41.8593	
4	Category 1	trip-153672019145222964	Carting	47.0	0.666667	1.750000	1.766667	3.248617	3.233333	39.3452	
...	...	...	...	...	...	...	...	...	...	...	...
14155	Category 7	trip-153808133667543285	FTL	118.0	1.983333	3.233333	3.250000	6.215731	6.183333	168.8821	
14156	Category 7	trip-153712521713367664	FTL	89.0	1.433333	2.716667	2.750000	5.625200	5.600000	123.3365	
14157	Category 7	trip-153816805956285514	FTL	175.0	2.866667	4.950000	4.983333	7.741082	7.716667	218.7102	
14158	Category 7	trip-153747533214526986	FTL	203.0	3.333333	10.950000	10.966667	13.940494	13.916667	278.2507	
14159	Category 7	trip-153800051661903546	FTL	48.0	0.816667	2.116667	2.133333	2.146146	2.133333	60.9205	

14160 rows × 12 columns

In [671]:
```
trip_records_for_encoding.drop(["trip_uuid_y"],axis = 1,inplace=True)
# trip_records_for_encoding.sample(15)
```

In [672]:
```
trip_records_for_encoding.sample(15)
```

Out[672]:

	city	route_type	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination
3221	Category 5	Carting	24.0	0.366667	0.783333	0.783333	1.634565	1.633333	20.6339	15.068582
12796	Category 7	FTL	170.0	2.633333	6.350000	6.383333	13.341480	13.316667	230.1376	173.125348
4699	Category 6	Carting	26.0	0.450000	0.733333	0.766667	2.286047	2.283333	24.8951	21.381948
390	Category 1	Carting	19.0	0.316667	0.533333	0.533333	1.212465	1.200000	13.2501	10.200625
9226	Category 7	Carting	57.0	0.983333	17.883333	17.900000	18.675169	18.666667	84.6001	70.381157
2376	Category 3	Carting	22.0	0.333333	0.683333	0.700000	3.607462	3.600000	25.6762	17.501620
11509	Category 7	FTL	747.0	11.300000	21.116667	21.333333	23.718354	23.700000	978.8525	713.454278
6691	Category 7	Carting	13.0	0.216667	0.400000	0.400000	1.074836	1.066667	10.9445	9.083165
8249	Category 7	FTL	960.0	14.183333	25.783333	26.033333	29.958639	29.950000	1315.8366	1010.827164
13541	Category 7	FTL	35.0	0.583333	0.983333	1.000000	1.498385	1.483333	49.1983	45.799225
10412	Category 7	FTL	482.0	7.983333	13.750000	13.883333	26.532838	26.500000	612.7635	513.269306
2483	Category 4	Carting	37.0	0.650000	0.900000	0.916667	2.883423	2.883333	38.7271	33.078091
8550	Category 7	FTL	326.0	5.350000	9.166667	9.250000	10.531498	10.516667	364.9302	254.786465
2940	Category 4	Carting	26.0	0.433333	4.850000	4.866667	5.262172	5.250000	29.2713	16.669036
5386	Category 6	Carting	22.0	0.366667	1.400000	1.400000	2.341975	2.333333	21.8943	16.686425

In [673]:
```
encoded_data = pd.get_dummies(trip_records_for_encoding,
             columns=["route_type","city"] )
```

In [674]:
```
encoded_data
```

Out[674]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance	route_type_
0	83.0	0.950000	3.183333	3.233333	4.407028	4.400000	76.1272	42.240305	53.1902	
1	69.0	0.883333	2.666667	2.700000	4.063014	4.050000	59.1472	28.706240	43.5805	
2	71.0	0.966667	3.316667	3.333333	4.076829	4.066667	65.6004	42.150597	62.8568	
3	42.0	0.733333	1.316667	1.316667	4.915934	4.900000	41.8593	32.569779	42.0797	
4	47.0	0.666667	1.750000	1.766667	3.248617	3.233333	39.3452	29.584603	35.5419	
...	...	...	...	...	...	...	...	...	...	
55	118.0	1.983333	3.233333	3.250000	6.215731	6.183333	168.8821	150.297914	168.2424	
56	89.0	1.433333	2.716667	2.750000	5.625200	5.600000	123.3365	84.553671	120.1657	
57	175.0	2.866667	4.950000	4.983333	7.741082	7.716667	218.7102	173.349660	230.1587	
58	203.0	3.333333	10.950000	10.966667	13.940494	13.916667	278.2507	194.623978	274.8427	
59	48.0	0.816667	2.116667	2.133333	2.146146	2.133333	60.9205	50.844665	60.9205	

60 rows × 18 columns

In [677]:
```
['segment_osrm_time', 'osrm_time',
       'segment_actual_time', 'actual_time',
       'Time_taken_between_odstart_and_od_end', 'start_scan_to_end_scan' ,'segment_osrm_distance', 'actual_distance_to_destination','osrm_distance' ]
```

Out[677]:
```
['segment_osrm_time',
 'osrm_time',
 'segment_actual_time',
 'actual_time',
 'Time_taken_between_odstart_and_od_end',
 'start_scan_to_end_scan',
 'segment_osrm_distance',
 'actual_distance_to_destination',
 'osrm_distance']
```

In [678]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

In [679]:
```
scaler = StandardScaler()
std_data = scaler.fit_transform(encoded_data[['segment_osrm_time',
 'osrm_time',
 'segment_actual_time',
 'actual_time',
 'Time_taken_between_odstart_and_od_end',
 'start_scan_to_end_scan',
 'segment_osrm_distance',
 'actual_distance_to_destination',
 'osrm_distance']])
std_data = pd.DataFrame(std_data, columns=['segment_osrm_time',
 'osrm_time',
 'segment_actual_time',
 'actual_time',
 'Time_taken_between_odstart_and_od_end',
 'start_scan_to_end_scan',
 'segment_osrm_distance',
 'actual_distance_to_destination',
 'osrm_distance'])
std_data.head()
```

Out[679]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance
0	-0.269133	-0.409683	-0.220225	-0.214843	-0.394178	-0.391956	-0.362747	-0.450888	-0.468190
1	-0.359785	-0.438916	-0.324535	-0.321822	-0.445632	-0.444397	-0.448864	-0.542288	-0.521446
2	-0.346835	-0.402374	-0.193306	-0.194785	-0.443566	-0.441900	-0.416136	-0.451494	-0.414618
3	-0.534615	-0.504692	-0.597087	-0.599297	-0.318061	-0.317039	-0.536543	-0.516196	-0.529763
4	-0.502239	-0.533926	-0.509601	-0.509034	-0.567441	-0.566761	-0.549293	-0.536356	-0.565995

In [680]:
```
scaler = MinMaxScaler()
MinMax_data = scaler.fit_transform(encoded_data[['segment_osrm_time','osrm_time','segment_actual_time','actual_time',
 'Time_taken_between_odstart_and_od_end','start_scan_to_end_scan','segment_osrm_distance','actual_distance_to_destination',
 'osrm_distance']])
MinMax_data = pd.DataFrame(MinMax_data,columns=['segment_osrm_time',
 'osrm_time','segment_actual_time','actual_time','Time_taken_between_odstart_and_od_end','start_scan_to_end_scan',
 'segment_osrm_distance','actual_distance_to_destination','osrm_distance'])
MinMax_data.head()
```

Out[680]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance
0	0.069369	0.059302	0.098113	0.098719	0.098792	0.098811	0.046420	0.031804	0.036747
1	0.056757	0.054651	0.081402	0.081644	0.090329	0.090201	0.034665	0.018854	0.028743
2	0.058559	0.060465	0.102426	0.101921	0.090669	0.090611	0.039132	0.031718	0.044799
3	0.032432	0.044186	0.037736	0.037353	0.111311	0.111111	0.022697	0.022551	0.027493
4	0.036937	0.039535	0.051752	0.051761	0.070296	0.070111	0.020957	0.019694	0.022047

In [681]:
```
std_data
```

Out[681]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance
0	-0.269133	-0.409683	-0.220225	-0.214843	-0.394178	-0.391956	-0.362747	-0.450888	-0.468190
1	-0.359785	-0.438916	-0.324535	-0.321822	-0.445632	-0.444397	-0.448864	-0.542288	-0.521446
2	-0.346835	-0.402374	-0.193306	-0.194785	-0.443566	-0.441900	-0.416136	-0.451494	-0.414618
3	-0.534615	-0.504692	-0.597087	-0.599297	-0.318061	-0.317039	-0.536543	-0.516196	-0.529763
4	-0.502239	-0.533926	-0.509601	-0.509034	-0.567441	-0.566761	-0.549293	-0.536356	-0.565995
...	...	...	...	...	...	...	...	...	...
14155	-0.042502	0.043440	-0.210131	-0.211500	-0.123651	-0.124754	0.107675	0.278861	0.169418
14156	-0.230282	-0.197738	-0.314441	-0.311792	-0.211977	-0.212156	-0.123317	-0.165131	-0.097018
14157	0.326583	0.430787	0.136448	0.136179	0.104495	0.104990	0.360386	0.434538	0.512552
14158	0.507888	0.635424	1.347789	1.336342	1.031740	1.033953	0.662356	0.578210	0.760187
14159	-0.495764	-0.468150	-0.435575	-0.435486	-0.732338	-0.731577	-0.439871	-0.392780	-0.425349

14160 rows × 9 columns

In [682]:
```
one_hot_encoded_data = encoded_data[["route_type_Carting","route_type_FTL","city_Category 1",
 "city_Category 2","city_Category 3","city_Category 4",
 "city_Category 5","city_Category 6","city_Category 7"]]
```

In [683]:
```
Standardized_Data = pd.concat([std_data,one_hot_encoded_data],axis = 1)
```

In [684]:
```
Min_Max_Scaled_Data = pd.concat([MinMax_data,one_hot_encoded_data],axis = 1)
```

In [685]:
```
Standardized_Data.sample(5)
```

Out[685]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance	route_t
2405	-0.418061	-0.373141	-0.600452	-0.589267	-0.483832	-0.481855	-0.444275	-0.417739	-0.436772	
3625	-0.683544	-0.702020	-0.765329	-0.766450	-0.741156	-0.739068	-0.632274	-0.618754	-0.648612	
1430	-0.677068	-0.680095	-0.704762	-0.706275	-0.680475	-0.679135	-0.611861	-0.590467	-0.616517	
2596	-0.424537	-0.380449	-0.506236	-0.502348	0.077005	0.080017	-0.440002	-0.398265	-0.434079	
3233	-0.741820	-0.753179	-0.724951	-0.726333	-0.812511	-0.811487	-0.689544	-0.674977	-0.698171	

In [686]:
```
Min_Max_Scaled_Data.sample(5)
```

Out[686]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	Time_taken_between_odstart_and_od_end	start_scan_to_end_scan	segment_osrm_distance	actual_distance_to_destination	osrm_distance	route_
5293	0.098198	0.108140	0.114825	0.114728	0.123375	0.123411	0.084803	0.088646	0.101884	
12330	0.069369	0.090698	0.270081	0.267876	0.217936	0.218122	0.066354	0.069843	0.079838	
9198	0.009910	0.012791	0.015633	0.015475	0.029311	0.029110	0.008691	0.001653	0.010457	
10613	0.130631	0.148837	0.155795	0.155816	0.185834	0.185732	0.143239	0.140831	0.148716	
13721	0.014414	0.019767	0.021563	0.021345	0.026535	0.026650	0.016328	0.020906	0.019646	

Route analysis

In [688]:
```
A = data1.groupby("route_schedule_uuid")["route_type"].unique().reset_index()
B = data1.groupby("route_schedule_uuid")["destination_city"].unique().reset_index()
B.columns = ["route_schedule_uuid","destination_cities"]
C = data1.groupby("route_schedule_uuid")["source_city"].unique().reset_index()
C.columns = ["route_schedule_uuid","source_cities"]
D = data1.groupby("route_schedule_uuid")["source_state"].unique().reset_index()
D.columns = ["route_schedule_uuid","source_states"]
E = data1.groupby("route_schedule_uuid")["destination_state"].unique().reset_index()
E.columns = ["route_schedule_uuid","destination_states"]
F = data1.groupby("route_schedule_uuid")[["source_state",
                                          "destination_state"]].nunique().sort_values(by="source_state",
                                                                                      ascending=False).reset_index()
F.columns = ["route_schedule_uuid","#source_states"
             ,"#destination_states"]
G = trip_records.groupby("route_schedule_uuid")["actual_distance_to_destination"].mean().reset_index()
G.columns = ["route_schedule_uuid","Average_Actual_distance_to_destination"]
H = trip_records["route_schedule_uuid"].value_counts().reset_index()
H.columns = ["route_schedule_uuid","Number_of_Trips"]
```

In [689]:
```
I = data1.groupby("route_schedule_uuid")[["source_city",
                                          "destination_city"]].nunique().sort_values(by="source_city",
                                                                                     ascending=False).reset_index()
I.columns = ["route_schedule_uuid","#source_cities"
             ,"#destination_cities"]
```

In [690]:
```
route_records = I.merge(H.merge(G.merge(F.merge(E.merge(D.merge(C.merge(A.merge(B,
        on ="route_schedule_uuid",
        how = "outer"),on ="route_schedule_uuid",
        how = "outer"),
       on ="route_schedule_uuid",
        how = "outer"),
       on ="route_schedule_uuid",
        how = "outer"),
       on ="route_schedule_uuid",
        how = "outer"),
       on ="route_schedule_uuid",
        how = "outer"),
       on ="route_schedule_uuid",
        how = "outer"),on ="route_schedule_uuid",
        how = "outer")
```

In [691]:
```
route_records.sort_values(by="Average_Actual_distance_to_destination",ascending=False)
```

Out[691]:

	route_schedule_uuid	#source_cities	#destination_cities	Number_of_Trips	Average_Actual_distance_to_destination	#source_states	#destination_states	destination_states	source_states	source_cities	route_type
602	thanos::sroute:3592c86e-c3d1-429b-917a-ebe9051...	2	2	5	2157.968312	2	2	[West Bengal, Maharashtra]	[Assam, West Bengal]	[Guwahati, Kolkata]	[FTL]
306	thanos::sroute:96a80600-40e1-436b-9161-fa68f9e...	3	3	13	2016.941915	2	2	[Bihar, Haryana]	[Haryana, Bihar]	[Gurgaon, Muzaffrpur, Purnia]	[FTL]
431	thanos::sroute:b3fd32d8-4027-4dc5-a425-eb8ccb8...	2	2	2	1939.975142	2	2	[Haryana, Punjab]	[Karnataka, Haryana]	[Bengaluru, Gurgaon]	[FTL]
629	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	2	2	22	1905.766051	2	2	[Haryana, Karnataka]	[Punjab, Haryana]	[Chandigarh, Gurgaon]	[FTL]
534	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	2	2	15	1869.546650	2	2	[Haryana, Punjab]	[Karnataka, Haryana]	[Bengaluru, Gurgaon]	[FTL]
...	...	...	...	...	...	...	...	...	...	...	...
1132	thanos::sroute:50fd07d6-9217-49dd-b510-3a68bf6...	1	1	1	9.042020	1	1	[Karnataka]	[Karnataka]	[Bengaluru]	[Carting]
964	thanos::sroute:dff6e62a-3e95-42e9-9794-b5125f8...	1	1	1	9.040986	1	1	[Tamil Nadu]	[Tamil Nadu]	[Salem]	[Carting]
1217	thanos::sroute:62545fd8-0a6d-4ede-87c9-71fdc9a...	1	1	2	9.032299	1	1	[Tamil Nadu]	[Tamil Nadu]	[Chennai]	[Carting]
654	thanos::sroute:fb962457-a420-4283-b216-55fd2ba...	1	1	1	9.027513	1	1	[Uttar Pradesh]	[Uttar Pradesh]	[Varanasi]	[Carting]
966	thanos::sroute:e00eb6aa-d792-4b28-81fa-fdee413...	1	1	1	9.006827	1	1	[Tamil Nadu]	[Tamil Nadu]	[Chennai]	[Carting]

1504 rows × 12 columns

In [692]:
```
route_records.isna().sum()
```

Out[692]:
```
route_schedule_uuid                   0
#source_cities                        0
#destination_cities                   0
Number_of_Trips                       0
Average_Actual_distance_to_destination    0
#source_states                        0
#destination_states                   0
destination_states                    0
source_states                         0
source_cities                         0
route_type                            0
destination_cities                    0
dtype: int64
```

In [693]:
```
route_records.dropna(inplace=True)
```

In [694]:
```
route_records["route_type"] = route_records["route_type"].astype("str").str.strip("[]").str.replace("'","")
route_records["source_cities"] = route_records["source_cities"].astype("str").str.strip("[]").str.replace("'","")
route_records["destination_cities"] = route_records["destination_cities"].astype("str").str.strip("[]").str.replace("'","")
route_records["source_states"] = route_records["source_states"].astype("str").str.strip("[]").str.replace("'","")

route_records["destination_states"] = route_records["destination_states"].astype("str").str.strip("[]").str.replace("'","")
```

In [695]:
```
route_records
```

Out[695]:

	route_schedule_uuid	#source_cities	#destination_cities	Number_of_Trips	Average_Actual_distance_to_destination	#source_states	#destination_states	destination_states	source_states	source_cities	route_type
0	thanos::sroute:d010efca-d90d-4977-b987-eae68c5...	13	11	14	281.596486	2	2	Assam Arunachal Pradesh	Assam Arunachal Pradesh	Guwahati LakhimpurN Dhemaji Likabali Tezpur Pa...	FTL
1	thanos::sroute:4cbecb35-356b-4b68-bf3c-6225b5e...	10	10	12	332.602225	2	2	Assam Meghalaya	Assam Meghalaya	Guwahati Rangia Kokrajhar Dhubri Bilasipara Tu...	FTL
2	thanos::sroute:ae5c430f-6153-48d1-8fe5-d5f0bbc...	10	10	20	351.611796	1	1	Rajasthan	Rajasthan	Jaipur Chomu Reengus Sikar Bikaner Didwana Suj...	FTL
3	thanos::sroute:f8968c72-5222-4d81-9eed-8a6d88f...	9	9	9	195.257193	1	2	Karnataka Goa	Karnataka	Mangalore Udupi Kundapura Bhatkal Honnavar Kum...	FTL
4	thanos::sroute:ed5b80be-7abf-424d-b8cd-d81556a...	9	8	20	178.737233	1	1	Rajasthan	Rajasthan	Ajmer Beawar Bilara Bijainagar Kekri Nasirabad...	FTL
...	...	...	...	...	...	...	...	...	...	...	...
1499	thanos::sroute:9e7bb811-593f-47bc-ac49-ba03ed8...	1	1	19	17.617532	1	1	Maharashtra	Maharashtra	Mumbai	Carting
1500	thanos::sroute:46b9641b-55b5-4b15-b039-2612a50...	1	1	15	10.137219	1	1	Maharashtra	Maharashtra	Mumbai	Carting
1501	thanos::sroute:b48f633d-15cb-4744-a0b9-21df0a9...	1	1	7	15.467701	1	1	Karnataka	Karnataka	Bengaluru	Carting
1502	thanos::sroute:265efe06-3625-4fba-afee-07b5b64...	0	1	1	236.815038	0	1	Uttar Pradesh	nan	nan	FTL
1503	thanos::sroute:cfb575b8-df26-48f5-8427-6f48f9d...	0	0	1	50.844665	0	0	nan	nan	nan	FTL

1504 rows × 12 columns

In [696]:
```
route_records["ROUTE"] = route_records["source_cities"] + " -- " + route_records["destination_cities"]
```

In [697]:
```
route_records.drop(["route_schedule_uuid"],axis = 1,inplace=True)
```

In [698]:
```
first_column = route_records.pop('ROUTE')
route_records.insert(0, 'ROUTE', first_column)
```

In [699]:
```
route_records["SouceToDestination_city"] = route_records["source_cities"].str.split(" ").apply(lambda x:x[0]) +" TO " +route_records["destination_cities"].str.split(" ").apply
```

In [700]:
```
first_column = route_records.pop('SouceToDestination_city')
route_records.insert(0, 'SouceToDestination_city', first_column)
```

In [701]:
```
route_records
```

Out[701]:

ROUTE	#source_cities	#destination_cities	Number_of_Trips	Average_Actual_distance_to_destination	#source_states	#destination_states	destination_states	source_states	source_cities	route_type	destination_cities
Guwahati LakhimpurN Dhemaji Likabali Tezpur Pa...	13	11	14	281.596486	2	2	Assam Arunachal Pradesh	Assam Arunachal Pradesh	Guwahati LakhimpurN Dhemaji Likabali Tezpur Pa...	FTL	Tezpur Dhemaji Silapathar Pasighat Mangaldoi I...
Guwahati Rangia Kokrajhar Dhubri Bilasipara Tu...	10	10	12	332.602225	2	2	Assam Meghalaya	Assam Meghalaya	Guwahati Rangia Kokrajhar Dhubri Bilasipara Tu...	FTL	Rangia Nalbari Dhubri Bilasipara Lakhipur Guwa...
Jaipur Chomu Reengus Sikar Bikaner Didwana Suj...	10	10	20	351.611796	1	1	Rajasthan	Rajasthan	Jaipur Chomu Reengus Sikar Bikaner Didwana Suj...	FTL	Chomu Reengus Sikar Bikaner Nokha Sujangarh Ja...
Mangalore Udupi Kundapura Bhatkal Honnavar Kum...	9	9	9	195.257193	1	2	Karnataka Goa	Karnataka	Mangalore Udupi Kundapura Bhatkal Honnavar Kum...	FTL	Uchila Kundapura Bhatkal Honnavar Kumta Ankola...
Ajmer Beawar Bilara Bijainagar Kekri Nasirabad...	9	8	20	178.737233	1	1	Rajasthan	Rajasthan	Ajmer Beawar Bilara Bijainagar Kekri Nasirabad...	FTL	Beawar Bilara Badnaur Kekri Nasirabad Ajmer Bi...
...	...	...	...	...	...	...	...	...	...	...	...
Mumbai -- Mumbai	1	1	19	17.617532	1	1	Maharashtra	Maharashtra	Mumbai	Carting	Mumbai
Mumbai -- Mumbai	1	1	15	10.137219	1	1	Maharashtra	Maharashtra	Mumbai	Carting	Mumbai
Bengaluru -- Bengaluru	1	1	7	15.467701	1	1	Karnataka	Karnataka	Bengaluru	Carting	Bengaluru
nan -- Mainpuri	0	1	1	236.815038	0	1	Uttar Pradesh	nan	nan	FTL	Mainpuri
nan -- nan	0	0	1	50.844665	0	0	nan	nan	nan	FTL	nan

In [702]:
```
route_records.to_csv("route_records.csv")
```

Inferences and Recommendations

Insights and Observations:

- There were a total of 14817 trips made between source and destination during the months of September and October in 2018.
- There are 1504 delivery routes where trips are taking place.
- The number of unique source centers is 1508 and the number of unique destination centers is 1481.
- Out of 14817 total trips, 60% or 8908 trips were Carting, which involved small vehicles. The remaining 40% or 5909 trips were FTL, which involved full truck loads and reached their destination faster as there were no additional pickups or drop-offs along the way.

From Exploratory Data Analysis:

- It can be noted that cities such as Mumbai (Maharashtra), Delhi, Gurgaon (Haryana), Bengaluru (Karnataka), Hyderabad (Telangana), Chennai (Tamil Nadu), Ahmedabad (Gujarat), Pune (Maharashtra), Chandigarh (Chandigarh), and Kolkata (West Bengal) experience a high volume of trips within their respective states.
- The cities with the highest number of trips between them are Delhi to Gurgaon, Gurgaon to Bengaluru, Bhiwandi/Mumbai to Pune Maharashtra, Sonipat to Gurgaon Haryana, where the source and destination states are not equal.
- Many deliveries are taking place to airports such as Chennai to Chennai International Airport (MAA), Pune to Pune Airport (PNQ), Kolkata to Kolkata International Airport (CCU West Bengal), Bengaluru to Bengaluru International Airport (BLR), etc.
- Based on the bar charts and analysis of calculated tables, it can be seen that the highest number of trips occur within specific cities. In terms of average distance between destinations, the longest routes are Guwahati to Mumbai, Bengaluru to Chandigarh, Bengaluru to Delhi, and Bengaluru to Gurgaon.
- From Guwahati to Bhiwandi, from Bengaluru to Chandigarh, from Bengaluru to Delhi, from Gurgaon to Chennai Airport, from Bhiwandi to Kolkata, from Bengaluru to Kolkata, from Gurgaon to Hyderabad, and from Gurgaon to Kolkata.
- The routes that pass through the most number of cities (more than 8) are Guwahati to Lakhimpur, Jaipur to Tarnau, Guwahati to Tura, Mangalore to Udupi, Ajmer to Raipur, and Mainpuri to Tilhar.
- The busiest route is the one between Delhi and Haryana, with over 400 trips. Other busy routes include Haryana to Uttar Pradesh, Chandigarh to Punjab, and Delhi to Uttar Pradesh.
- Some states in India, including Maharashtra, Karnataka, Tamil Nadu, Haryana, Telangana, Gujarat, West Bengal, and Uttar Pradesh, have recorded more than 1000 trips.
- The cities listed are Bengaluru in Karnataka, Gurgaon in Haryana, Mumbai in Maharashtra, Hyderabad in Telangana, Delhi, Pune in Maharashtra, Chandigarh in Punjab, Chennai in Tamil Nadu, Sonipat in Haryana, Kolkata in West Bengal, Ahmedabad in Gujarat, MAA in Tamil Nadu, Jaipur in Rajasthan, Kanpur in Uttar Pradesh, Surat in Gujarat, Muzaffrpur in Bihar, FBD in Haryana, Bhopal in Madhya Pradesh, and Noida in Uttar Pradesh.

Hypothesis tests Results

- The conclusion that can also be drawn from a two-sample t-test is that...
  - The average duration between the start and end of the odometer reading for the population is equivalent to the average duration from the beginning to the end of the scan for the population.
  - The average actual time in the population is less than the average of the start scan to end scan time in the population.
  - The population mean of the actual time taken to complete delivery and the population mean of the time elapsed between the start and end of the delivery are not equal.
  - The average actual delivery time exceeds the average estimated delivery time according to OSRM.
  - The average time taken to complete a delivery trip and the actual time taken for each segment are equivalent.
  - The average of OSRM time and segment-osrm-time for the population does not match.
  - The average osrm time is lower than the average segment osrm time.
  - The mean of the OSRM distances for the population is lower than the average of the segment's OSRM distances.
  - The estimated distance provided by OSRM for the population is greater than the real distance between the source and the destination warehouse.

Recommendations

- To optimize delivery time and increase revenue, it is recommended to use cars for intra-city deliveries and heavy trucks for long distance trips or heavy loads, as per the analysis.
- By improving the connectivity in tier 2 and tier 3 cities and establishing partnerships with leading e-commerce companies, revenue and the reputation for cross-border connectivity can be enhanced.
- We can improve the accuracy of the delivery time by optimizing the start and end scanning times, bringing them in line with the estimated delivery time from OSRM.

In [ ]:
```
```