Business Case: Delhivery

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.

Business Problem Statement

The company wants to understand and process the data coming out of data engineering pipelines:

• Clean, sanitize and manipulate data to get useful features out of raw fields

• Make sense out of the raw data and help the data science team to build forecasting models on it

Importing Libraries:

In [201]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import figure
import warnings
warnings.filterwarnings('ignore')
import statsmodels.api as sm
from scipy.stats import norm
from scipy.stats import t
from scipy.stats import ttest_ind
from scipy import stats
```

In [4]:
```
data = pd.read_csv("delhivery_data.txt")
data
```

Out[4]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destina
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_M
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_M
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_M
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_M
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_M
...	...	...	...	...	...	...	...	...	
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_E
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_E
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_E
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_E
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_E

144867 rows × 24 columns

In [5]:
```
data.shape
```

Out[5]: (144867, 24)

In [6]:
```
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 [7]:
```
data.isna().sum()
```

Out[7]:
```
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, destination_name having missing values

Changing data type for data and time related features :

In [19]:
```
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"])
```

Extracting Trip Creation Informations from Trip Creation time :

In [21]:
```
data["trip_creation_time"].dt.month_name().value_counts()
```

Out[21]:
```
September    127349
October       17518
Name: trip_creation_time, dtype: int64
```

In [22]:
```
data["trip_creation_time"].dt.year.value_counts()
```

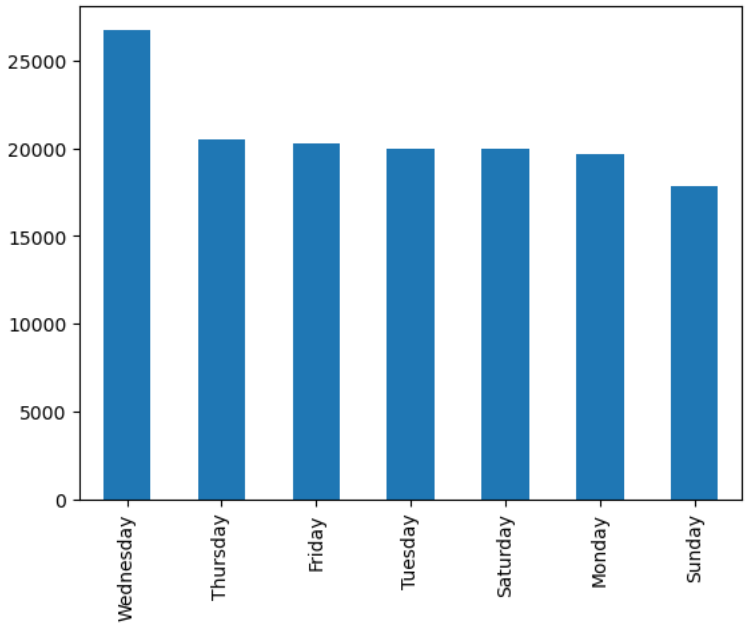
Out[22]:
```
2018    144867
Name: trip_creation_time, dtype: int64
```

- delivery trip data is given from September and October 2018

In [24]:
```
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 [28]:
```
data["trip_creation_day"].value_counts().plot(kind = "bar" )
```

Out[28]: <AxesSubplot:>



In [29]:
```
data["trip_creation_day"].value_counts(normalize=True)*100
```

Out[29]:
```
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

In [30]:
```
data.nunique()
```

Out[30]:
```
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
```

- there is 14817 different trips happened between source to destination.
- there is total 1504 delivery routes.
- 1508 unique source centers
- 1481 unique destination centres

There are two different kind of routes are there

In [32]:
```
data.groupby("trip_uuid")["route_type"].unique().reset_index()["route_type"].apply(lambda x:x[0]).value_counts()
```

Out[32]:
```
Carting    8908
FTL        5909
Name: route_type, dtype: int64
```

In [33]:
```
data.groupby("trip_uuid")["route_type"].unique().reset_index()["route_type"].apply(lambda x:x[0]).value_counts(normalize = True)*
```

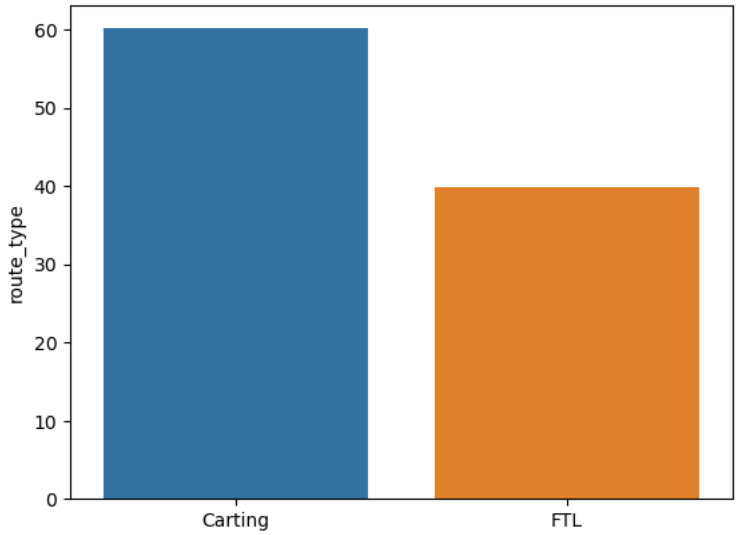
Out[33]:
```
Carting    60.120132
FTL        39.879868
Name: route_type, dtype: float64
```

In [34]:
```
route_plot= (data.groupby("trip_uuid")["route_type"].unique().reset_index()["route_type"].apply(lambda x:x[0]).value_counts(norma
route_plot
```

Out[34]:
```
Carting    60.120132
FTL        39.879868
Name: route_type, dtype: float64
```

In [36]:
```
sns.barplot(x= route_plot.index,
            y = route_plot)
```

Out[36]:
```
<AxesSubplot:ylabel='route_type'>
```



- From 14,817 total different trips , we have
- 8908 (60%) of the trip-routes are Carting , which consists of small vehicles
- 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

Analyzing records for one particular trip id :

In [37]:
```
data[data["trip_uuid"]=="trip-153741093647649320"]
```

Out[37]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destinatic
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_Motv
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_Motv
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_Motv
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_Motv
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_Motv
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_Va
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_Va
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_Va
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_Va
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_Va

10 rows × 27 columns

In [ ]:
```
from above one particular trip record , trip is segmented between different drop locations .

we can observe
trip is taking stops between mentioned source and destination centers(warehouses).
od-end-time and od-start-time are the time when the that particular trip was ended and started .

start-scan-to-end-scan is the time duration of trips are being scanned when start and end.
start-scan-to-end-scan time is given cummulative. which is not given per trip segments.


trip cut off False ,shows the record of trip when trip changes from one warehouse to another. between source to destination.


Actual-time given is the time to complete the entire delivery from source to destination (given cumulatively )



osrm -time is an open rourse routing engine time calculator which computes the shortest path between points in a given map and gi


Actual-distnace-to-destination is the actual distance between warehouses , given cummulative during the trip .
every time cutoff is False , distance count starts from begining.

Segmment actual time,  is the actual time taken between two stops in between trips. given per each segment (taken between subset

segment osrm time is the osrm segment time , taken between subset of package delivery
```

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

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

In [38]:
```
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(")","")
```

In [39]:
```
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 [40]:
```
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 [41]:
```
data
```

Out[41]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destina
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_M
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_M
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_M
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_M
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_M
...	...	...	...	...	...	...	...	...	
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_E
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_E
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_E
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_E
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_E

144867 rows × 35 columns

Time_taken_btwn_odstart_and_od_end VS start_scan_to_end_scan :

In [42]:
```
data["time_taken_btwn_odstart_and_od_end"] = ((data["od_end_time"]-data["od_start_time"])/pd.Timedelta(1,unit="hour"))
```

Converting given time duration features into hours .

- start_scan_to_end_scan
- actual_time
- osrm_time
- segment_actual_time
- segment_osrm_time

In [44]:
```
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
data["segment_osrm_time"] = data["segment_osrm_time"]/60
```

In [45]:
```
data
```

Out[45]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destina
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_M
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_M
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_M
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_M
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_M
...	...	...	...	...	...	...	...	...	
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_E
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_E
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_E
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_E
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_E

144867 rows × 36 columns

In [46]:
```
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_btwn_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 [47]:
```
data.isna().sum()
```

Out[47]:
```
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_btwn_odstart_and_od_end      0
dtype: int64
```

In [49]:
```
data.shape
```

Out[49]:
```
(144867, 36)
```

Data cleaning :

In [50]:
```
data['source_state'].unique()
```

Out[50]:
```
array(['Gujarat', 'Maharashtra', 'Karnataka', 'Punjab', 'Haryana',
       'Uttarakhand', 'Tamil Nadu', 'Rajasthan', nan, 'Telangana',
       'Madhya Pradesh', 'Uttar Pradesh', 'Himachal Pradesh', 'Kerala',
       'Andhra Pradesh', 'Bihar', 'Jharkhand', 'Hub Maharashtra', 'Assam',
       'West Bengal', 'Orissa', 'Delhi', 'Nagar_DC Rajasthan',
       'Jammu & Kashmir', 'Alipore_DPC West Bengal', 'Chandigarh',
       'Chhattisgarh', 'Vadgaon Sheri DPC Maharashtra', 'Goa',
       '02_DPC Uttar Pradesh', 'MP Nagar Madhya Pradesh', 'Road Punjab',
       'Pondicherry', 'Layout PC Karnataka', 'Mandakni Madhya Pradesh',
       'Dadra and Nagar Haveli', 'DC Maharashtra', 'Arunachal Pradesh',
       'Antop Hill Maharashtra', 'City Madhya Pradesh',
       'Pashan DPC Maharashtra', 'Nagaland', 'Meghalaya', 'DC Rajasthan',
       'West _Dc Maharashtra', 'Nagar Uttar Pradesh',
       '_NAD Andhra Pradesh', 'Avenue_DPC West Bengal', 'Tripura',
       'Mizoram', 'Rahatani DPC Maharashtra', 'Balaji Nagar Maharashtra',
       'Goa Goa', 'Kothanur_L Karnataka', 'Mahim Maharashtra'],
      dtype=object)
```

In [51]:
```
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 [52]:
```
data['destination_state'].unique()
```

Out[52]:
```
array(['Gujarat', 'Maharashtra', 'Karnataka', 'Kerala', 'Punjab',
       'Uttarakhand', 'Tamil Nadu', 'Haryana', 'Rajasthan', nan,
       'Telangana', 'Uttar Pradesh', 'Delhi', 'Himachal Pradesh',
       'Hub Maharashtra', 'Andhra Pradesh', 'Bihar', 'Jharkhand', 'Assam',
       'Orissa', 'West Bengal', 'Pashan DPC Maharashtra',
       'Jammu & Kashmir', 'Madhya Pradesh', 'Avenue_DPC West Bengal',
       'Chandigarh', 'Chhattisgarh', 'Vadgaon Sheri DPC Maharashtra',
       '02_DPC Uttar Pradesh', 'Goa', 'MP Nagar Madhya Pradesh',
       'Pondicherry', 'Layout PC Karnataka', 'Mandakni Madhya Pradesh',
       'Arunachal Pradesh', 'Dadra and Nagar Haveli',
       'Nagar_DC Rajasthan', 'West _Dc Maharashtra',
       'Alipore_DPC West Bengal', 'Meghalaya', 'Rahatani DPC Maharashtra',
       'Nagar Uttar Pradesh', 'Kothanur_L Karnataka',
       'City Madhya Pradesh', 'Balaji Nagar Maharashtra', 'Tripura',
       'Mizoram', 'Daman & Diu', 'Nagaland', 'Goa Goa',
       'Antop Hill Maharashtra', 'West_Dc Maharashtra', 'Delhi Delhi'],
      dtype=object)
```

In [54]:
```
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 [55]:
```
data["destination_city"].replace({
    "del":"Delhi"
},inplace=True)
data["source_city"].replace({
    "del":"Delhi"
},inplace=True)
```

In [56]:
```
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 [57]:
```
data["source_city_state"] = data["source_city"] + " " + data["source_state"]
data["destination_city_state"] = data["destination_city"] + " " + data["destination_state"]
```

In [58]:
```
data["source_city_state"].nunique()
```

Out[58]: 1249

In [59]:
```
data["destination_city_state"].nunique()
```

Out[59]: 1242

In [60]:
```
data["source_state"].nunique()
```

Out[60]: 33

In [61]:
```
data["destination_state"].nunique()
```

Out[61]: 32

- Delhivery delivered in approximately 1250 cities and almost all the states all over in india

In [62]:
```
new_data = data.copy()
```

In [63]:
```
new_data.columns
```

Out[63]:
```
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_btwn_odstart_and_od_end',
       'source_city_state', 'destination_city_state'],
      dtype='object')
```

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

In [ ]:
```
# above data we impute after aggregating as per tripIDs.
```

In [64]:
```
new_data.drop(['source_center',"source_name","destination_center","destination_name","cutoff_timestamp"],axis = 1,inplace=True)
```

In [65]:
```
new_data.drop(["od_end_time","od_start_time"],axis = 1 , inplace=True)
```

In [66]:
```
new_data
```

Out[66]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	start_scan_to_end_scan	is_cutoff	cutoff_factor	actual_distance_to_
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	True	9	
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	True	18	
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	True	27	
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	True	36	
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	1.433333	False	39	
...	...	...	...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	True	45	
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	True	54	
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	True	63	
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	True	72	
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	7.116667	False	70	

144867 rows × 31 columns

Aggregating Data :

In [67]:
```
actual_time = new_data.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["actual_time"].max().reset_index().groupby("trip_uuid")["actual_time"].sum().reset_index
segment_osrm_time = new_data[["trip_uuid","segment_osrm_time"]].groupby("trip_uuid")["segment_osrm_time"].sum().reset_index()
segment_actual_time = new_data.groupby("trip_uuid")["segment_actual_time"].sum().reset_index()
osrm_time = new_data.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["osrm_time"].max().reset_index().groupby("trip_uuid")["osrm_time"].sum().reset_index()
time_taken_btwn_odstart_and_od_end = new_data.groupby("trip_uuid")["time_taken_btwn_odstart_and_od_end"].unique().reset_index()

time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"] = time_taken_btwn_odstart_and_od_end["time_taken_btwn_od
start_scan_to_end_scan = ((new_data.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 = new_data.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["osrm_distance"].max().reset_index().groupby("trip_uuid")["osrm_distance"].sum().reset_i
actual_distance_to_destination = new_data.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["actual_distance_to_destination"].max().reset_index().groupby("trip_uuid")["actual_dista
segment_osrm_distance = new_data[["trip_uuid",
                            "segment_osrm_distance"]].groupby("trip_uuid")["segment_osrm_distance"].sum().reset_index()
```

3.In-depth analysis and feature engineering:

Hypothesis Tests for time durations and distance related features :

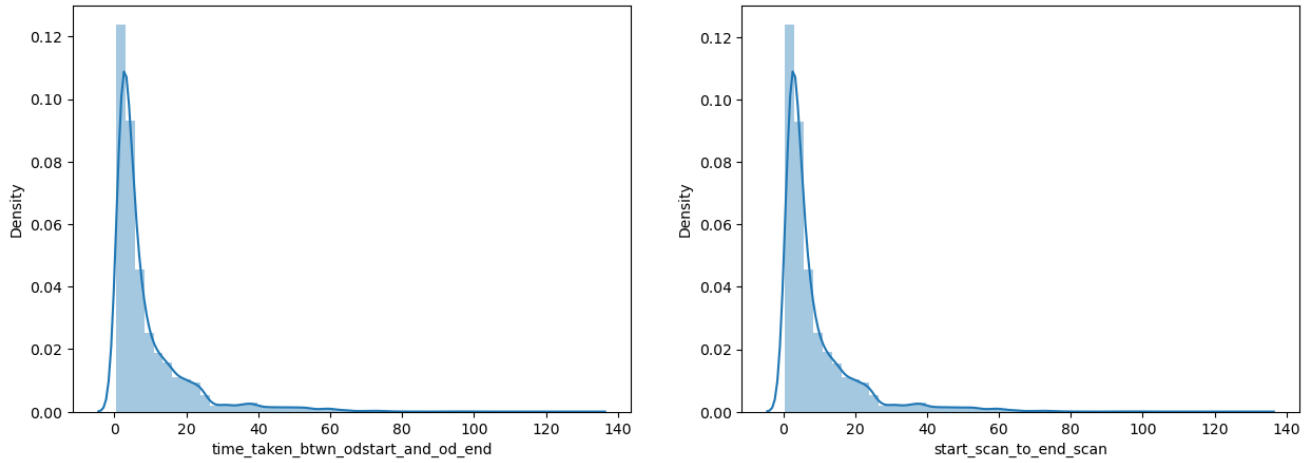
Analysing TimeTaken Between OdStart to OdEnd time & StartScan to EndScan :

H0: Mean of time taken between start and end, trip time = Mean of start and end, scan time

Ha: Mean of time taken between start and end, trip time != Mean of start and end, scan time¶

In [69]:
```
plt.figure(figsize=(15,5))
plt.subplot(121)
sns.distplot((time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"]))
plt.subplot(122)
sns.distplot((start_scan_to_end_scan["start_scan_to_end_scan"]))
```

Out[69]:
```
<AxesSubplot:xlabel='start_scan_to_end_scan', ylabel='Density'>
```



In [70]:
```
: Checking the distributions how closely and equally they are :
2samp(time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"], start_scan_to_end_scan["start_scan_to_end_scan"])
```

Out[70]:
```
KstestResult(statistic=0.004184382803536474, pvalue=0.9994337058695081)
```

In [71]:
```
for i in range(5):
    print(stats.ttest_ind((actual_time["actual_time"].sample(1000))
                ,(time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"].sample(1000))))
```

```
Ttest_indResult(statistic=-6.611291472463768, pvalue=4.8740033871078995e-11)
Ttest_indResult(statistic=-6.544170110074605, pvalue=7.580863526709161e-11)
Ttest_indResult(statistic=-6.378226115648503, pvalue=2.2196425991268515e-10)
Ttest_indResult(statistic=-4.870444321156885, pvalue=1.2009941687761395e-06)
Ttest_indResult(statistic=-7.0794350434866615, pvalue=1.9961263060332827e-12)
```

In [72]:
```
# You could use a two-sample t-test to test the hypothesis
t, p = stats.ttest_ind((actual_time["actual_time"].sample(1000))
                ,(time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"].sample(1000)))

# If p-value is less than 0.05, reject the null hypothesis
if p < 0.05:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
Reject H0
```

- from Kolmogorov–Smirnov test , p-value is 0.9943 , from which we can conclude tht both the distributions (time_taken_btwn_odstart_and_od_end and start_scan_to_end_scan) are closely similar.
- from 2 sample t-test, we can also conclude that average time_taken_btwn_odstart_and_od_end for population is also equal to average start_scan_to_end_scan for population.

In [73]:
```
dard deviation for Trip time:
d_end["time_taken_btwn_odstart_and_od_end"].mean(),time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"].std()
```

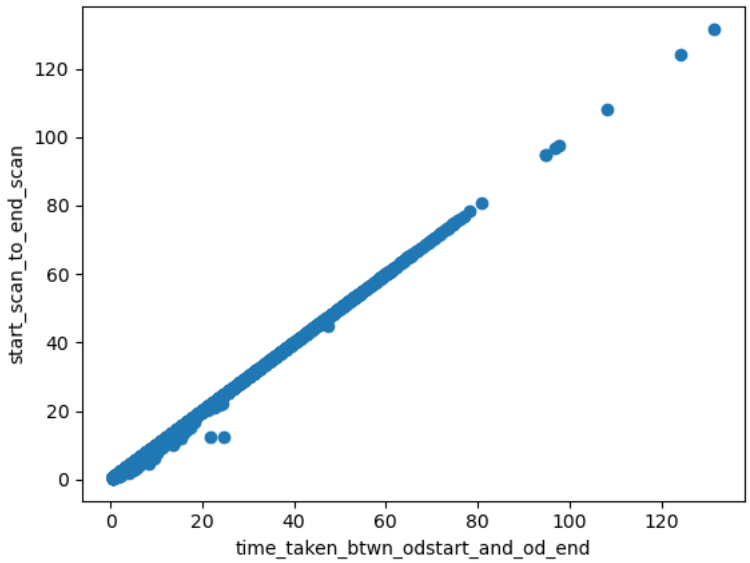
Out[73]: (8.861857235305067, 10.981665759990623)

In [74]:
```
# Also checking mean and standard deviation for Scan time :
start_scan_to_end_scan["start_scan_to_end_scan"].mean(),start_scan_to_end_scan["start_scan_to_end_scan"].std()
```

Out[74]: (8.835777597804325, 10.97628639143973)

- variance and mean both are closely similar for trip start and end time and scan time

In [76]:
```
# Visual analysis
# scatter plot to visualize the relationship between the two columns
plt.scatter(time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"], start_scan_to_end_scan["start_scan_to_end_s
plt.xlabel('time_taken_btwn_odstart_and_od_end')
plt.ylabel('start_scan_to_end_scan')
plt.show()
```



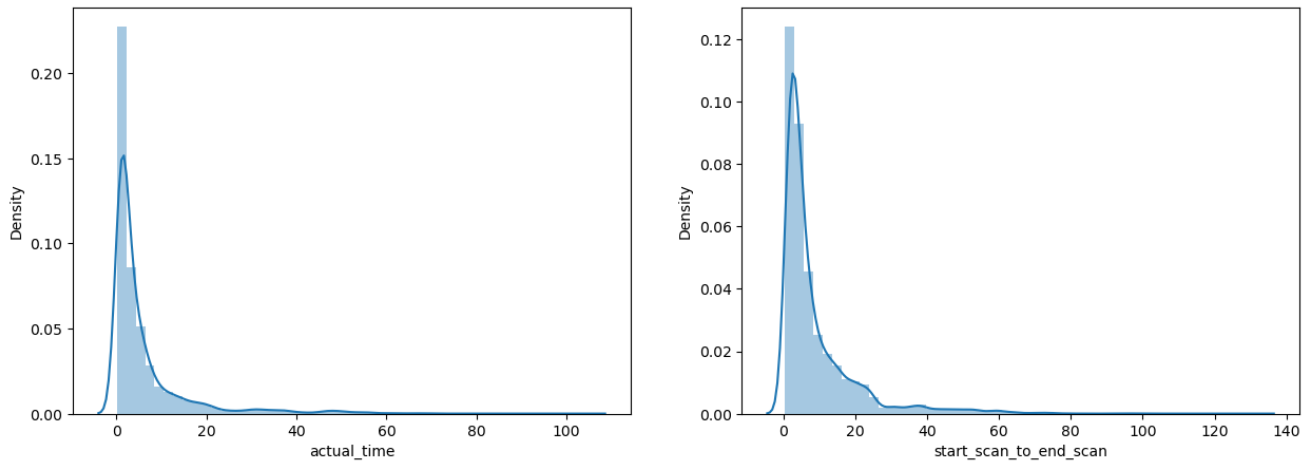
Analysing Actual Time taken to complete the delivery & start-scan-end-scan

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 [77]:
```
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"]))
```

Out[77]: <AxesSubplot:xlabel='start_scan_to_end_scan', ylabel='Density'>



In [78]:
```
# KS-test : Checking the distributions how closely and equally they are
stats.ks_2samp(actual_time["actual_time"],start_scan_to_end_scan["start_scan_to_end_scan"])
```

Out[78]: KstestResult(statistic=0.27387460349598436, pvalue=0.0)

In [79]:
```
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=-10.346881915408108, pvalue=3.4966461172722804e-25)
Ttest_indResult(statistic=-10.908438633500797, pvalue=9.4965418992526e-28)
Ttest_indResult(statistic=-12.325367596094628, pvalue=8.623519660253465e-35)
Ttest_indResult(statistic=-9.520911813345581, pvalue=1.2138213940652973e-21)
Ttest_indResult(statistic=-10.96408046988087, pvalue=5.203878774690686e-28)
Ttest_indResult(statistic=-12.249588899961294, pvalue=2.1513828051846042e-34)
Ttest_indResult(statistic=-9.967665756515583, pvalue=1.5988702087399364e-23)
```

In [80]:
```
# You could use a two-sample t-test to test the hypothesis
t, p = stats.ttest_ind((actual_time["actual_time"].sample(3000)), (start_scan_to_end_scan["start_scan_to_end_scan"].sample(3000)))

# If p-value is less than 0.05, reject the null hypothesis
if p < 0.05:
    print("Reject H0, average actual_time is less than population average start_scan_to_end_scan")
else:
    print("average actual_time is greater than or equal to population average start_scan_to_end_scan")
```

```
Reject H0, average actual_time is less than population average start_scan_to_end_scan
```

- from KS test for actual-time and start_scan_to_end_scan distributions are not same.
- t test of population average actual_time is less than population average start_scan_to_end_scan.

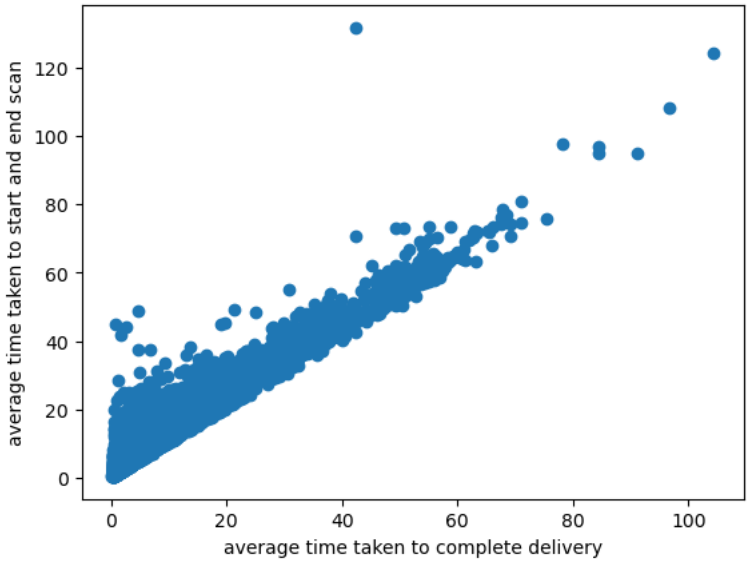
In [81]:
```
actual_time["actual_time"].mean(),actual_time["actual_time"].std()
```

Out[81]: (5.945176711435117, 9.35554782297388)

In [82]:
```
start_scan_to_end_scan["start_scan_to_end_scan"].mean(),start_scan_to_end_scan["start_scan_to_end_scan"].std()
```

Out[82]: (8.835777597804325, 10.97628639143973)

In [83]:
```
# Visual analysis
# scatter plot to visualize the relationship between the two columns
plt.scatter(actual_time["actual_time"],start_scan_to_end_scan["start_scan_to_end_scan"])
plt.xlabel('average time taken to complete delivery')
plt.ylabel('average time taken to start and end scan')
plt.show()
```



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 [84]:
```
stats.ks_2samp(actual_time["actual_time"],time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"])
```

Out[84]:
```
KstestResult(statistic=0.2765067152594992, pvalue=0.0)
```

In [85]:
```
for i in range(5):
    print(stats.ttest_ind((actual_time["actual_time"].sample(1000))
                ,(time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"].sample(1000))))
```

```
Ttest_indResult(statistic=-5.9700307499946055, pvalue=2.79991933480723e-09)
Ttest_indResult(statistic=-5.487077697636152, pvalue=4.605890805083997e-08)
Ttest_indResult(statistic=-5.5967969741848105, pvalue=2.484580639724411e-08)
Ttest_indResult(statistic=-8.134354899006857, pvalue=7.193829735578578e-16)
Ttest_indResult(statistic=-7.133520528828217, pvalue=1.3623059325083446e-12)
```

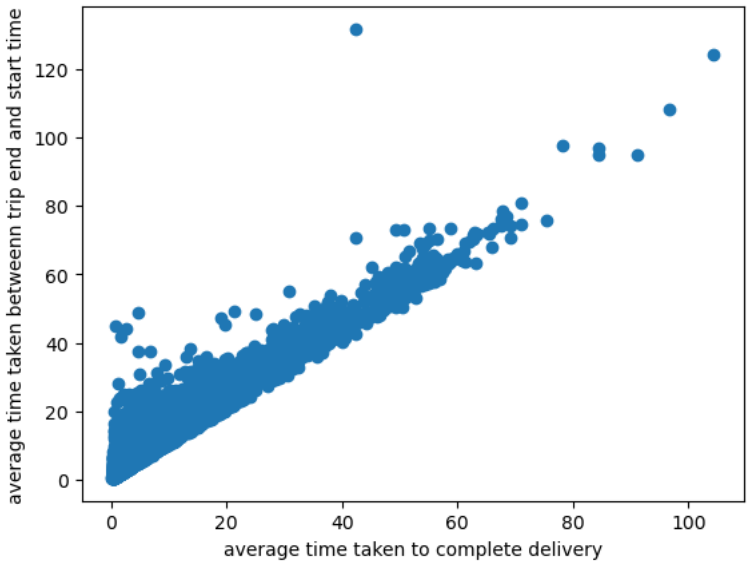
In [86]:
```
# You could use a two-sample t-test to test the hypothesis
t, p = stats.ttest_ind((actual_time["actual_time"].sample(1000)),
                       (time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"].sample(1000)))

# If p-value is less than 0.05, reject the null hypothesis
if p < 0.05:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
Reject H0
```

- from above kstest of distribution and two sample ttest ,
- we can conclude that population mean actual_time taken to complete delivery and population mean time_taken_btwn_od_start_and_od_end are also not same.

In [87]:
```
# Visual analysis
# scatter plot to visualize the relationship between the two columns
plt.scatter(actual_time["actual_time"],time_taken_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"])
plt.xlabel('average time taken to complete delivery')
plt.ylabel('average time taken betweenn trip end and start time')
plt.show()
```



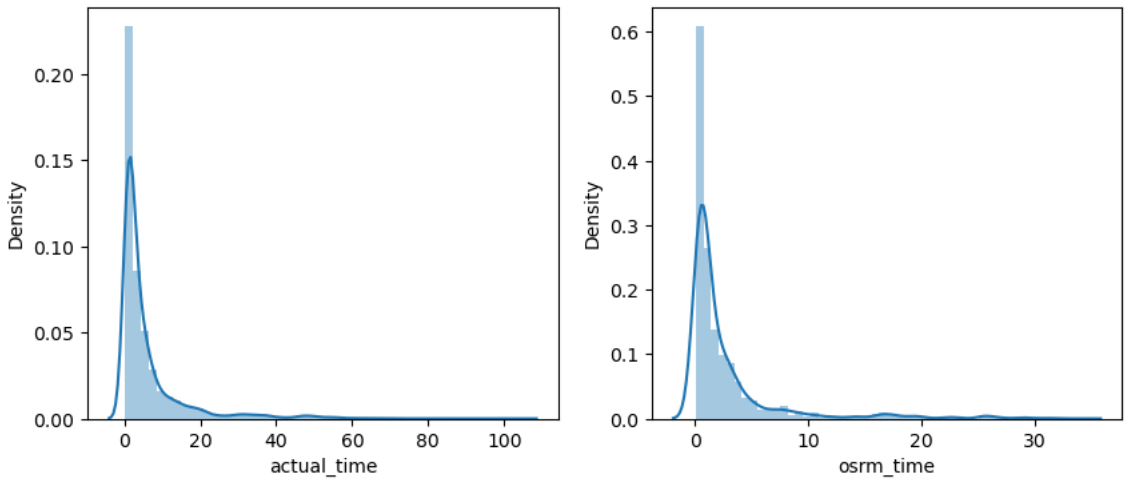
Analysing Actual Time taken to complete delivery from source to destination hub & OSRM measured time :

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 [88]:
```
plt.figure(figsize=(10,4))
plt.subplot(121)
sns.distplot(((actual_time["actual_time"])))
plt.subplot(122)
sns.distplot(((osrm_time["osrm_time"])))
```

Out[88]:
```
<AxesSubplot:xlabel='osrm_time', ylabel='Density'>
```



In [89]:
```
stats.ks_2samp(actual_time["actual_time"], osrm_time["osrm_time"])
```

Out[89]:
```
KstestResult(statistic=0.2945265573327934, pvalue=0.0)
```

In [90]:
```
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=21.113524303608532, pvalue=3.8079053841649383e-97)
Ttest_indResult(statistic=22.606355580556237, pvalue=1.0646924270822882e-110)
Ttest_indResult(statistic=22.35101183196233, pvalue=2.5394268977506855e-108)
Ttest_indResult(statistic=21.923688796334538, pvalue=2.1332975599994168e-104)
Ttest_indResult(statistic=21.80772665394172, pvalue=2.4108930805476576e-103)
```

In [91]:
```
# You could use a two-sample t-test to test the hypothesis
t, p = stats.ttest_ind(actual_time["actual_time"].sample(5000),
                osrm_time["osrm_time"].sample(5000),alternative='greater')

# If p-value is less than 0.05, reject the null hypothesis
if p < 0.05:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
Reject H0
```

- from two sample t test we can conclude that population mean actual time taken to complete delivery from source to warehouse and osrm estimate mean time for population are not same.
- actual time is higher than the osrm estimated time for delivery.

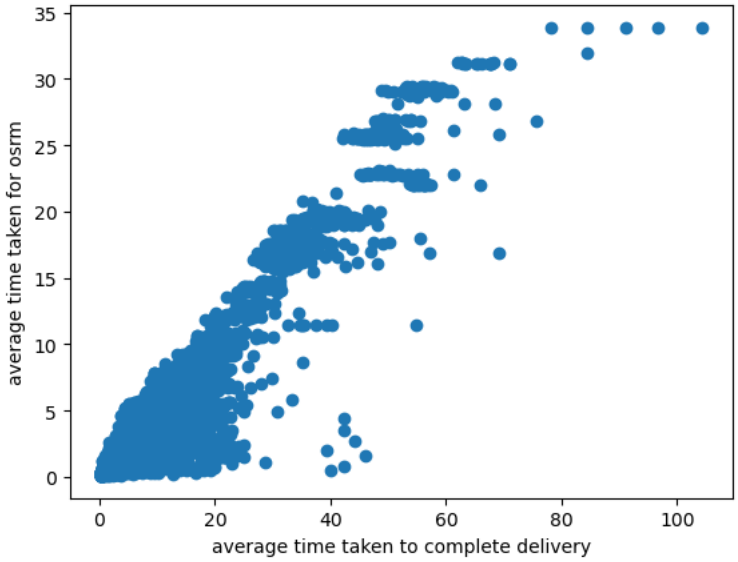
In [93]:
```
actual_time["actual_time"].mean(),actual_time["actual_time"].std()
```

Out[93]: (5.945176711435117, 9.35554782297388)

In [94]:
```
osrm_time["osrm_time"].mean(),osrm_time["osrm_time"].std()
```

Out[94]: (2.697313896200314, 4.537654251845703)

In [95]:
```
# Visual analysis
# scatter plot to visualize the relationship between the two columns
plt.scatter(actual_time["actual_time"], osrm_time["osrm_time"])
plt.xlabel('average time taken to complete delivery')
plt.ylabel('average time taken for osrm')
plt.show()
```



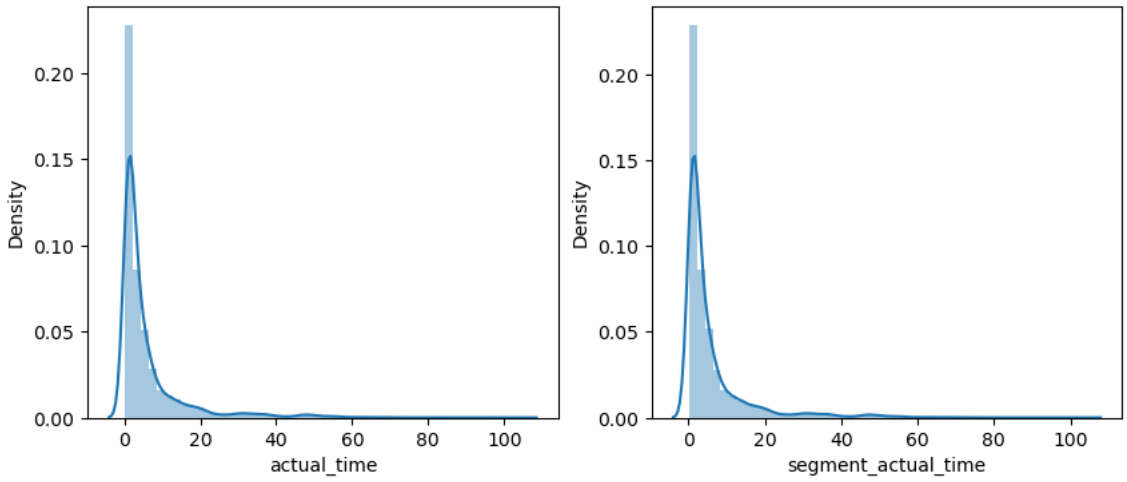
Analysing Actual Time taken to complete delivery from source to destination hub & Segment Actual Time :

H0: Actual time = segment actual time

Ha: Actual time != segment actual time

In [97]:
```
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"])))
```

Out[97]:
```
<AxesSubplot:xlabel='segment_actual_time', ylabel='Density'>
```



In [98]:
```
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=-0.5924671403345311, pvalue=0.5535601522825845)
Ttest_indResult(statistic=0.4487812049306625, pvalue=0.6536057109571575)
Ttest_indResult(statistic=0.1202163851052729, pvalue=0.9043157649106701)
Ttest_indResult(statistic=0.06496857199957115, pvalue=0.948201188394096)
Ttest_indResult(statistic=-0.5905906189600484, pvalue=0.5548170032105051)
Ttest_indResult(statistic=0.5307277561404781, pvalue=0.5956270982792182)
Ttest_indResult(statistic=0.4503093133630965, pvalue=0.652503699698777)
```

In [99]:
```
# You could use a two-sample t-test to test the hypothesis
t, p = stats.ttest_ind((actual_time["actual_time"].sample(3000)),
                (segment_actual_time["segment_actual_time"].sample(3000)))

# If p-value is less than 0.05, reject the null hypothesis
if p < 0.05:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
Fail to reject H0
```

- from two sample ttest , we can conclude that population average for Actual Time taken to complete delivery trip and segment actual time are same.

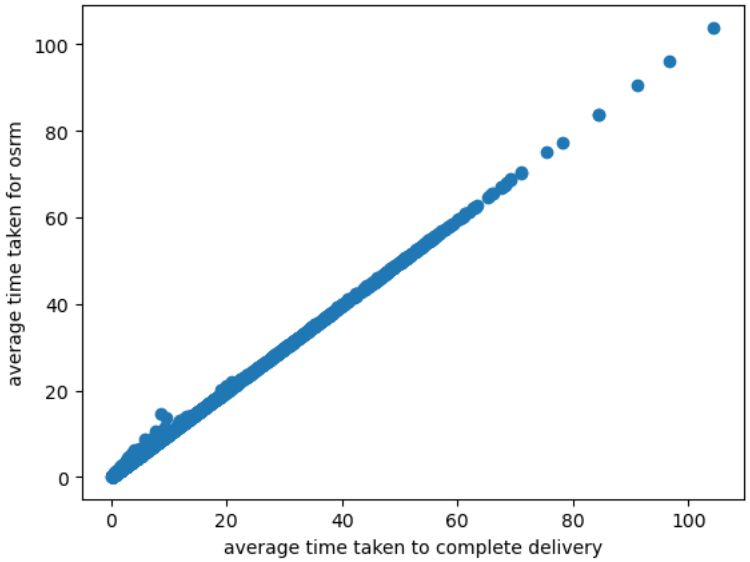
In [100]:
```
actual_time["actual_time"].mean(),actual_time["actual_time"].std()
```

Out[100]:
```
(5.945176711435117, 9.35554782297388)
```

In [101]:
```
segment_actual_time["segment_actual_time"].mean(),segment_actual_time["segment_actual_time"].std()
```

Out[101]:
```
(5.898204764797215, 9.270799413152762)
```

In [102]:
```
# Visual analysis
# scatter plot to visualize the relationship between the two columns
plt.scatter(actual_time["actual_time"], segment_actual_time["segment_actual_time"])
plt.xlabel('average time taken to complete delivery')
plt.ylabel('average time taken for osrm')
plt.show()
```



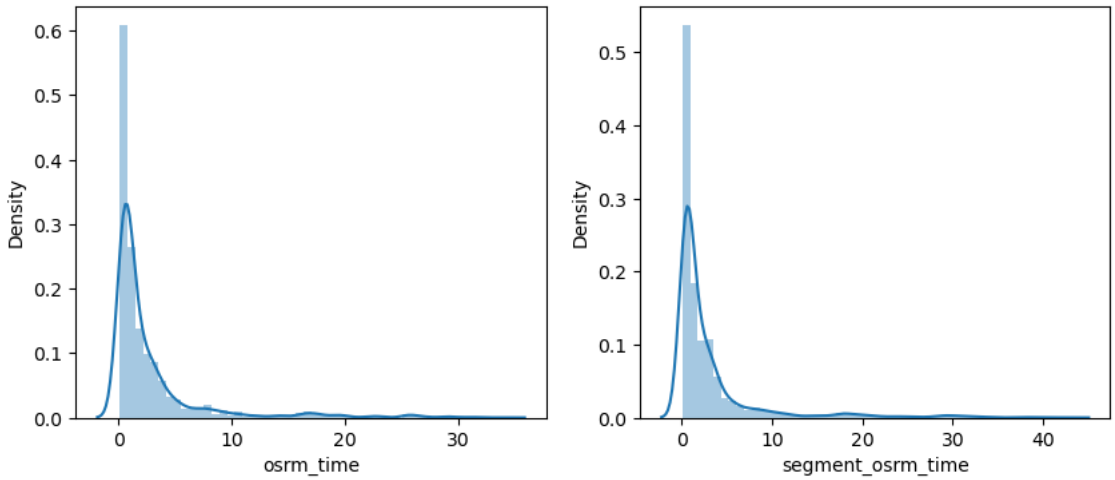
Analysing osrm Time & segment-osrm-time :

H0: segment actual time <= OSRM time

Ha: segment actual time > OSRM time

In [104]:
```
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"])))
```

Out[104]: <AxesSubplot:xlabel='segment_osrm_time', ylabel='Density'>



In [105]:
```
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=-3.189336597939127, pvalue=0.0007166674108004285)
Ttest_indResult(statistic=-3.301003757304989, pvalue=0.0004845131000488376 5)
Ttest_indResult(statistic=-3.7341664098076977, pvalue=9.504185111456117e-05)
Ttest_indResult(statistic=-4.009772956093486, pvalue=3.0757489986457994e-05)
Ttest_indResult(statistic=-2.352242811129341, pvalue=0.009346366939404044)
Ttest_indResult(statistic=-2.1325073237820846, pvalue=0.016502830499565527)
Ttest_indResult(statistic=-1.702686044727531, pvalue=0.04433932515648669)
```

In [106]:
```
# We can use a two-sample t-test to test the hypothesis
t, p = stats.ttest_ind((osrm_time["osrm_time"].sample(3000)),
                (segment_osrm_time["segment_osrm_time"].sample(3000)),alternative ="less")

# If p-value is less than 0.05, we reject the null hypothesis
if p < 0.05:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
Reject H0
```

- from ttest , we can conclude that average of osrm Time & segment-osrm-time for population is not same.
- Population Mean osrm time is less than Population Mean segment osrm time.

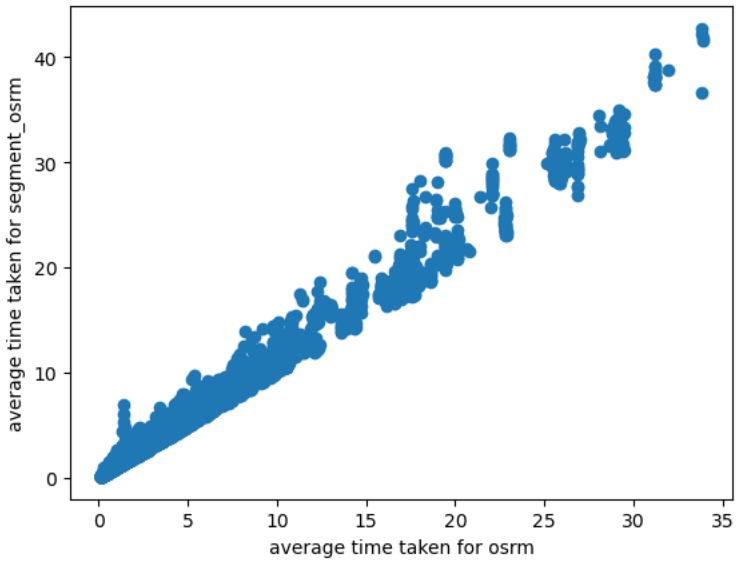
In [107]:
```
osrm_time["osrm_time"].mean(),osrm_time["osrm_time"].std()
```

Out[107]:
```
(2.697313896200314, 4.537654251845703)
```

In [108]:
```
segment_osrm_time["segment_osrm_time"].mean(),segment_osrm_time["segment_osrm_time"].std()
```

Out[108]:
```
(3.0158297901059705, 5.242367441693007)
```

In [109]:
```
# Visual analysis
# scatter plot to visualize the relationship between the two columns
plt.scatter(osrm_time["osrm_time"], segment_osrm_time["segment_osrm_time"])
plt.xlabel('average time taken for osrm')
plt.ylabel('average time taken for segment_osrm')
plt.show()
```



Analysing Distances measures :

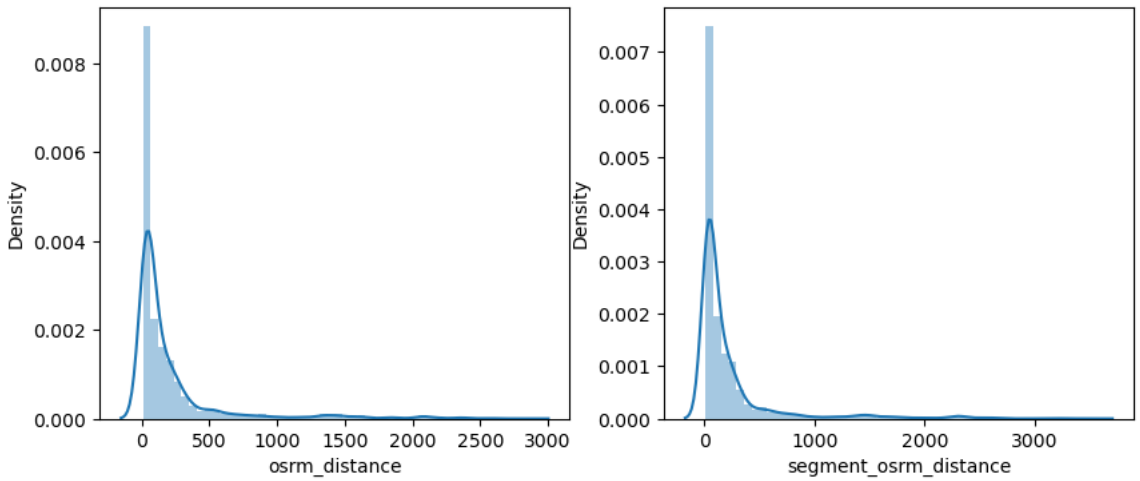
Analysing and Visulizing OSRM Estimated distance and Segment-osrm-distance :

H0 : Segment OSRM distance <= OSRM distance

Ha : Segment OSRM distance > OSRM distance

In [110]:
```
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"])))
```

Out[110]: <AxesSubplot:xlabel='segment_osrm_distance', ylabel='Density'>



In [111]:
```
stats.ks_2samp(osrm_distance["osrm_distance"],segment_osrm_distance["segment_osrm_distance"])
```

Out[111]: KstestResult(statistic=0.03948167645272321, pvalue=1.8042208791084262e-10)

In [112]:
```
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=-1.708648235545984, pvalue=0.04377358288223859)
Ttest_indResult(statistic=-3.708382173311544, pvalue=0.00010485816689210049)
Ttest_indResult(statistic=-3.067588732953408, pvalue=0.0010818518075808527)
Ttest_indResult(statistic=-2.7365768446368173, pvalue=0.0031095872633100582)
Ttest_indResult(statistic=-2.3727222997963238, pvalue=0.00883816997636293)
Ttest_indResult(statistic=-2.601825350138051, pvalue=0.0046432885070805925)
Ttest_indResult(statistic=-3.751779209315119, pvalue=8.828834257315083e-05)
```

In [113]:
```
# We can use a two-sample t-test to test the hypothesis
t, p = stats.ttest_ind(osrm_distance["osrm_distance"].sample(5000),
                segment_osrm_distance["segment_osrm_distance"].sample(5000),alternative="less")
# If p-value is less than 0.05, we reject the null hypothesis
if p < 0.05:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
Reject H0
```

- from KS test , we can conclude the distributions of segment osrm distance and osrm distance are not same!
- from two sample one sided ttest, we can conclude that Average of osrm distance for population is less than average of segment osrm distance

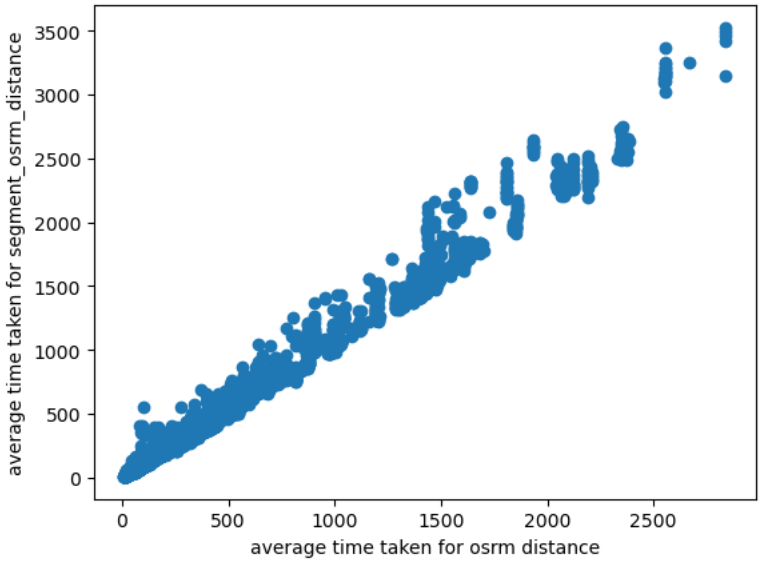
In [114]:
```
osrm_distance["osrm_distance"].mean(),osrm_distance["osrm_distance"].std()
```

Out[114]: (204.83672531551625, 370.74927471335496)

In [115]:
```
segment_osrm_distance["segment_osrm_distance"].mean(),segment_osrm_distance["segment_osrm_distance"].std()
```

Out[115]: (223.20116128771042, 416.6283742907418)

In [116]:
```
# Visual analysis
# scatter plot to visualize the relationship between the two columns
plt.scatter(osrm_distance["osrm_distance"], segment_osrm_distance["segment_osrm_distance"])
plt.xlabel('average time taken for osrm distance')
plt.ylabel('average time taken for segment_osrm_distance')
plt.show()
```



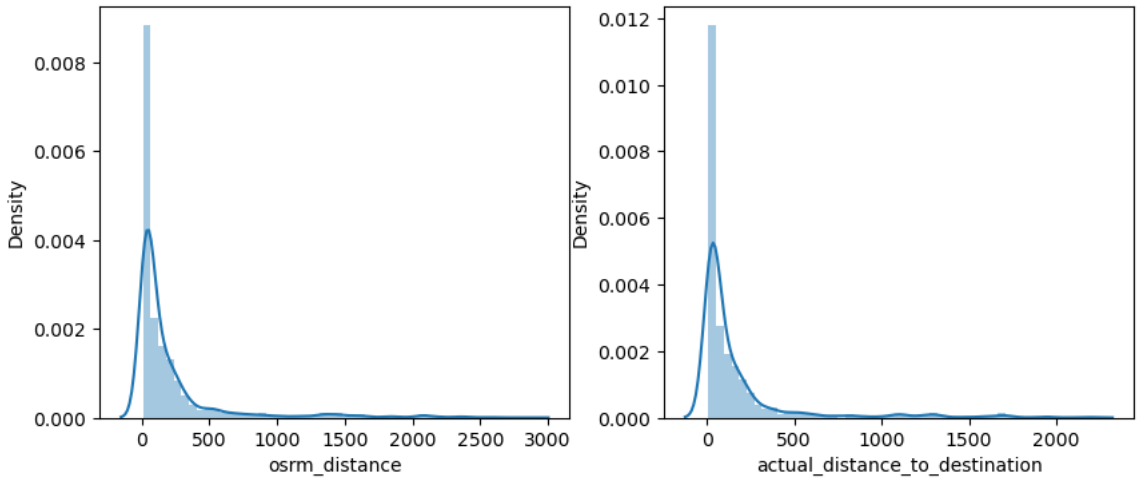
Analysing and Visulizing OSRM Estimated distance and Actual Distance between source and destination warehouse :

H0 : Mean OSRM distance <= Mean Actual distnace

Ha : Mean OSRM distance > Mean Actual distnace

In [118]:
```
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"])))
```

Out[118]: <AxesSubplot:xlabel='actual_distance_to_destination', ylabel='Density'>



In [119]:
```
stats.ks_2samp(osrm_distance["osrm_distance"],actual_distance_to_destination["actual_distance_to_destination"])
```

Out[119]: KstestResult(statistic=0.11837753931295136, pvalue=6.578385372142345e-91)

In [120]:
```
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=5.271305079215503, pvalue=6.9140918330326e-08)
Ttest_indResult(statistic=6.738042809493213, pvalue=8.468944757445392e-12)
Ttest_indResult(statistic=5.483477481375194, pvalue=2.1359716705057126e-08)
Ttest_indResult(statistic=4.311402936170675, pvalue=8.18885304635954e-06)
Ttest_indResult(statistic=6.046394132148361, pvalue=7.671082502963416e-10)
```

In [121]:
```
# We can use a two-sample t-test to test the hypothesis
t, p = stats.ttest_ind(osrm_distance["osrm_distance"].sample(5000),
                actual_distance_to_destination["actual_distance_to_destination"].sample(5000),alternative="greater")
# If p-value is less than 0.05, we reject the null hypothesis
if p < 0.05:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
Reject H0
```

- From left sided ttest , - we can conclude that population OSRM estimated distance is higher than the actual distance from source to destination warehouse.

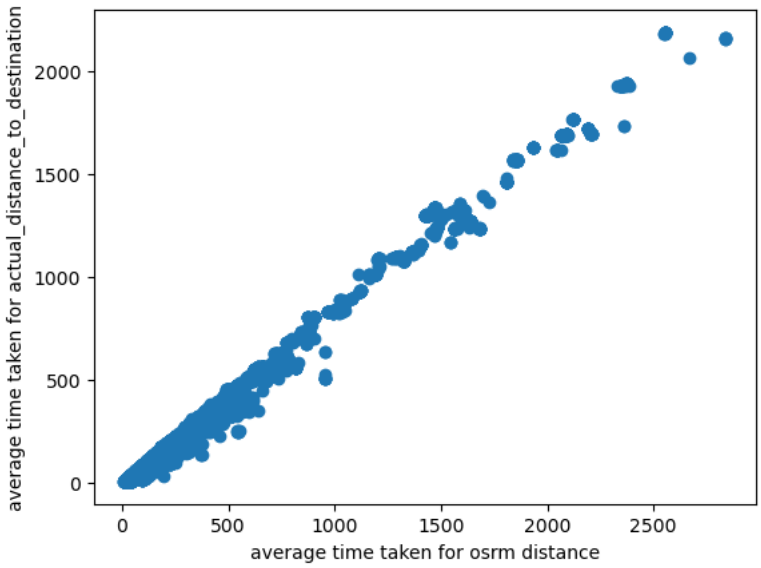
In [122]:
```
osrm_distance["osrm_distance"].mean(),osrm_distance["osrm_distance"].std()
```

Out[122]:
```
(204.83672531551625, 370.74927471335496)
```

In [123]:
```
e_to_destination["actual_distance_to_destination"].mean(),actual_distance_to_destination["actual_distance_to_destination"].std()
```

Out[123]:
```
(164.4733217454422, 305.5408288910492)
```

In [124]:
```
# Visual analysis
# scatter plot to visualize the relationship between the two columns
plt.scatter(osrm_distance["osrm_distance"], actual_distance_to_destination["actual_distance_to_destination"])
plt.xlabel('average time taken for osrm distance')
plt.ylabel('average time taken for actual_distance_to_destination')
plt.show()
```



Merging All the numerical Fields as per TripID:

In [125]:
```
actual_distance_to_destination
osrm_distance
segment_osrm_distance
```

Out[125]:

	trip_uuid	segment_osrm_distance
0	trip-153671041653548748	1320.4733
1	trip-153671042288605164	84.1894
2	trip-153671043369099517	2545.2678
3	trip-153671046011330457	19.8766
4	trip-153671052974046625	146.7919
...	...	...
14812	trip-153861095625827784	64.8551
14813	trip-153861104386292051	16.0883
14814	trip-153861106442901555	104.8866
14815	trip-153861115439069069	223.5324
14816	trip-153861118270144424	80.5787

14817 rows × 2 columns

In [126]:
```
time_taken_btwn_odstart_and_od_end
start_scan_to_end_scan
actual_time
segment_actual_time
osrm_time
segment_osrm_time
```

Out[126]:

	trip_uuid	segment_osrm_time
0	trip-153671041653548748	16.800000
1	trip-153671042288605164	1.083333
2	trip-153671043369099517	32.350000
3	trip-153671046011330457	0.266667
4	trip-153671052974046625	1.916667
...	...	...
14812	trip-153861095625827784	1.033333
14813	trip-153861104386292051	0.183333
14814	trip-153861106442901555	1.466667
14815	trip-153861115439069069	3.683333
14816	trip-153861118270144424	1.116667

14817 rows × 2 columns

In [127]:
```
distances = segment_osrm_distance.merge(actual_distance_to_destination.merge(osrm_distance,
                                                                             on="trip_uuid"),
                                        on="trip_uuid")
```

In [128]:
```
time = segment_osrm_time.merge(osrm_time.merge(segment_actual_time.merge(actual_time.merge(time_taken_btwn_odstart_and_od_end.mer
                                    on="trip_uuid",
                                    ),on="trip_uuid"),on="trip_uuid"),on="trip_uuid"),on="trip_uuid")
```

In [129]:
```
Merge1 = time.merge(distances,on="trip_uuid",
                                    )
```

In [153]:
```
Merge1
```

Out[153]:

	trip_uuid	segment_osrm_time	osrm_time	segment_actual_time	actual_time	time_taken_btwn_odstart_and_od_end	start_scan_to_end_scan	s
0	trip-153671041653548748	16.800000	12.383333	25.800000	26.033333	37.668497	37.650000	
1	trip-153671042288605164	1.083333	1.133333	2.350000	2.383333	3.026865	3.000000	
2	trip-153671043369099517	32.350000	29.016667	55.133333	55.783333	65.572709	65.550000	
3	trip-153671046011330457	0.266667	0.250000	0.983333	0.983333	1.674916	1.666667	
4	trip-153671052974046625	1.916667	1.950000	5.666667	5.683333	11.972484	11.950000	
...	...	...	...	...	...	...	...	
14812	trip-153861095625827784	1.033333	1.033333	1.366667	1.383333	4.300482	4.283333	
14813	trip-153861104386292051	0.183333	0.200000	0.350000	0.350000	1.009842	1.000000	
14814	trip-153861106442901555	1.466667	0.900000	4.683333	4.700000	7.035331	7.016667	
14815	trip-153861115439069069	3.683333	3.066667	4.300000	4.400000	5.808548	5.783333	
14816	trip-153861118270144424	1.116667	1.133333	4.566667	4.583333	5.906793	5.883333	

14817 rows × 10 columns

Merging Location details and route_type and Numerical data on TripID :

In [155]:
```
city = new_data.groupby("trip_uuid")[["source_city",
                                      "destination_city"]].aggregate({
        "source_city":pd.unique,
    "destination_city":pd.unique,
})

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

city_state = new_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 [156]:
```
route_type = new_data.groupby("trip_uuid")["route_type"].unique().reset_index()
```

In [157]:
```
Merged = route_type.merge(locations.merge(Merge1,on="trip_uuid",
           how="outer"),
                 on="trip_uuid",
           how="outer"
                )
```

In [158]:
```
trip_records = Merged.copy()
```

In [159]:
```
trip_records["route_type"] = trip_records["route_type"].apply(lambda x:x[0])
```

In [160]:
```
route_to_merge = new_data.groupby("trip_uuid")["route_schedule_uuid"].unique().reset_index()
```

In [161]:
```
trip_records = trip_records.merge(route_to_merge,on="trip_uuid",how="outer")
```

In [162]:
```
trip_records["route_schedule_uuid"] = trip_records["route_schedule_uuid"].apply(lambda x:x[0])
```

In [163]:
```
trip_records
```

Out[163]:

	trip_uuid	route_type	source_city	destination_city	source_city_state	destination_city_state	source_state	destination_state	segment_osrm_
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]	16.8(
1	trip-153671042288605164	Carting	[Tumkur, Doddablpur]	[Doddablpur, Chikblapur]	[Tumkur Karnataka, Doddablpur Karnataka]	[Doddablpur Karnataka, Chikblapur Karnataka]	[Karnataka]	[Karnataka]	1.0(
2	trip-153671043369099517	FTL	[Bengaluru, Gurgaon]	[Gurgaon, Chandigarh]	[Bengaluru Karnataka, Gurgaon Haryana]	[Gurgaon Haryana, Chandigarh Punjab]	[Karnataka, Haryana]	[Haryana, Punjab]	32.3(
3	trip-153671046011330457	Carting	[Mumbai]	[Mumbai]	[Mumbai Hub Maharashtra]	[Mumbai Maharashtra]	[Hub Maharashtra]	[Maharashtra]	0.2(
4	trip-153671052974046625	FTL	[Bellary, Hospet, Sandur]	[Hospet, Sandur, Bellary]	[Bellary Karnataka, Hospet Karnataka, Sandur K...	[Hospet Karnataka, Sandur Karnataka, Bellary K...	[Karnataka]	[Karnataka]	1.9(
...	...	...	...	...	...	...	...	...	
14812	trip-153861095625827784	Carting	[Chandigarh]	[Zirakpur, Chandigarh]	[Chandigarh Punjab, Chandigarh Chandigarh]	[Zirakpur Punjab, Chandigarh Punjab]	[Punjab, Chandigarh]	[Punjab]	1.0(
14813	trip-153861104386292051	Carting	[FBD]	[Faridabad]	[FBD Haryana]	[Faridabad Haryana]	[Haryana]	[Haryana]	0.1(
14814	trip-153861106442901555	Carting	[Kanpur]	[Kanpur]	[Kanpur Uttar Pradesh]	[Kanpur Uttar Pradesh]	[Uttar Pradesh]	[Uttar Pradesh]	1.4(
14815	trip-153861115439069069	Carting	[Tirunelveli, Eral, Tirchchndr, Thisayanvilai,...	[Eral, Tirchchndr, Thisayanvilai, Peikulam, Ti...	[Tirunelveli Tamil Nadu, Eral Tamil Nadu, Tirc...	[Eral Tamil Nadu, Tirchchndr Tamil Nadu, Thisa...	[Tamil Nadu]	[Tamil Nadu]	3.6(
14816	trip-153861118270144424	FTL	[Hospet, Sandur]	[Sandur, Bellary]	[Hospet Karnataka, Sandur Karnataka]	[Sandur Karnataka, Bellary Karnataka]	[Karnataka]	[Karnataka]	1.1(

14817 rows × 18 columns

In [164]:
```
# route_df['source'] = route_df['source'].str.strip("{''}")
```

In [165]:
```
trip_records.isna().sum()
```

Out[165]:
```
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_btwn_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 [166]:
```
trip_records.loc[trip_records.isnull().any(axis=1)]
```

Out[166]:

trip_uuid	route_type	source_city	destination_city	source_city_state	destination_city_state	source_state	destination_state	segment_osrm_time	osrm_time	s

In [167]:
```
trip_records.corr()
```

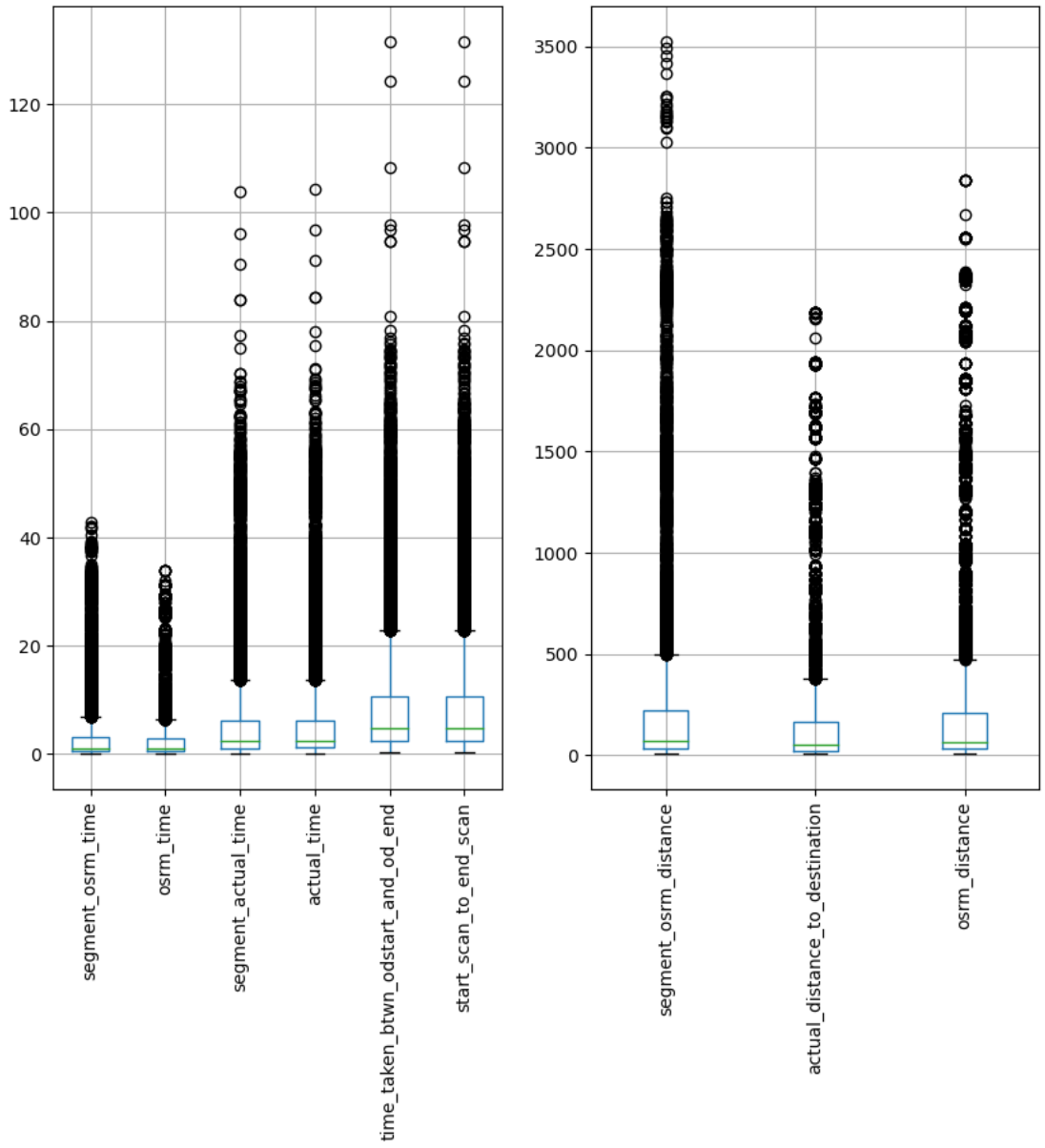
Out[167]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	time_taken_btwn_odstart_and_od_end	start_scan_to_enc
segment_osrm_time	1.000000	0.993508	0.953039	0.953800	0.918447	0.9
osrm_time	0.993508	1.000000	0.957747	0.958613	0.926280	0.9
segment_actual_time	0.953039	0.957747	1.000000	0.999920	0.961096	0.9
actual_time	0.953800	0.958613	0.999920	1.000000	0.960958	0.9
time_taken_btwn_odstart_and_od_end	0.918447	0.926280	0.961096	0.960958	1.000000	0.9
start_scan_to_end_scan	0.918493	0.926469	0.961107	0.961163	0.999860	1.0
segment_osrm_distance	0.996092	0.991848	0.956106	0.956949	0.919156	0.9
actual_distance_to_destination	0.987627	0.993556	0.953048	0.954082	0.918373	0.9
osrm_distance	0.992050	0.997610	0.958341	0.959290	0.924093	0.9

In [168]:
```
trip_records.to_csv("trip_records.csv")
```

Treating Outliers :

In [169]:
```
plt.figure(figsize = (10,8))
plt.subplot(121)
trip_records[['segment_osrm_time', 'osrm_time',
       'segment_actual_time', 'actual_time',
       'time_taken_btwn_odstart_and_od_end', 'start_scan_to_end_scan']].boxplot()
plt.xticks(rotation =90)
plt.subplot(122)
trip_records[['segment_osrm_distance', 'actual_distance_to_destination',
       'osrm_distance']].boxplot()
plt.xticks(rotation =90)
plt.show()
```



In [170]:
```
outlier_treatment  = trip_records.copy()
```

In [171]:
```
outlier_treatment_num = outlier_treatment[['segment_osrm_time', 'osrm_time',
       'segment_actual_time', 'actual_time',
       'time_taken_btwn_odstart_and_od_end', 'start_scan_to_end_scan',
        'segment_osrm_distance', 'actual_distance_to_destination',
        'osrm_distance']]
```

In [172]:
```
# outlier_treatment_num[(np.abs(stats.zscore(outlier_treatment_num)) < 3).all(axis=1)]
```

After removing outliers from all numerical features :

In [173]:
```
trip_records_without_outliers = trip_records.loc[outlier_treatment_num[(np.abs(stats.zscore(outlier_treatment_num)) < 3).all(axis
trip_records_without_outliers
```

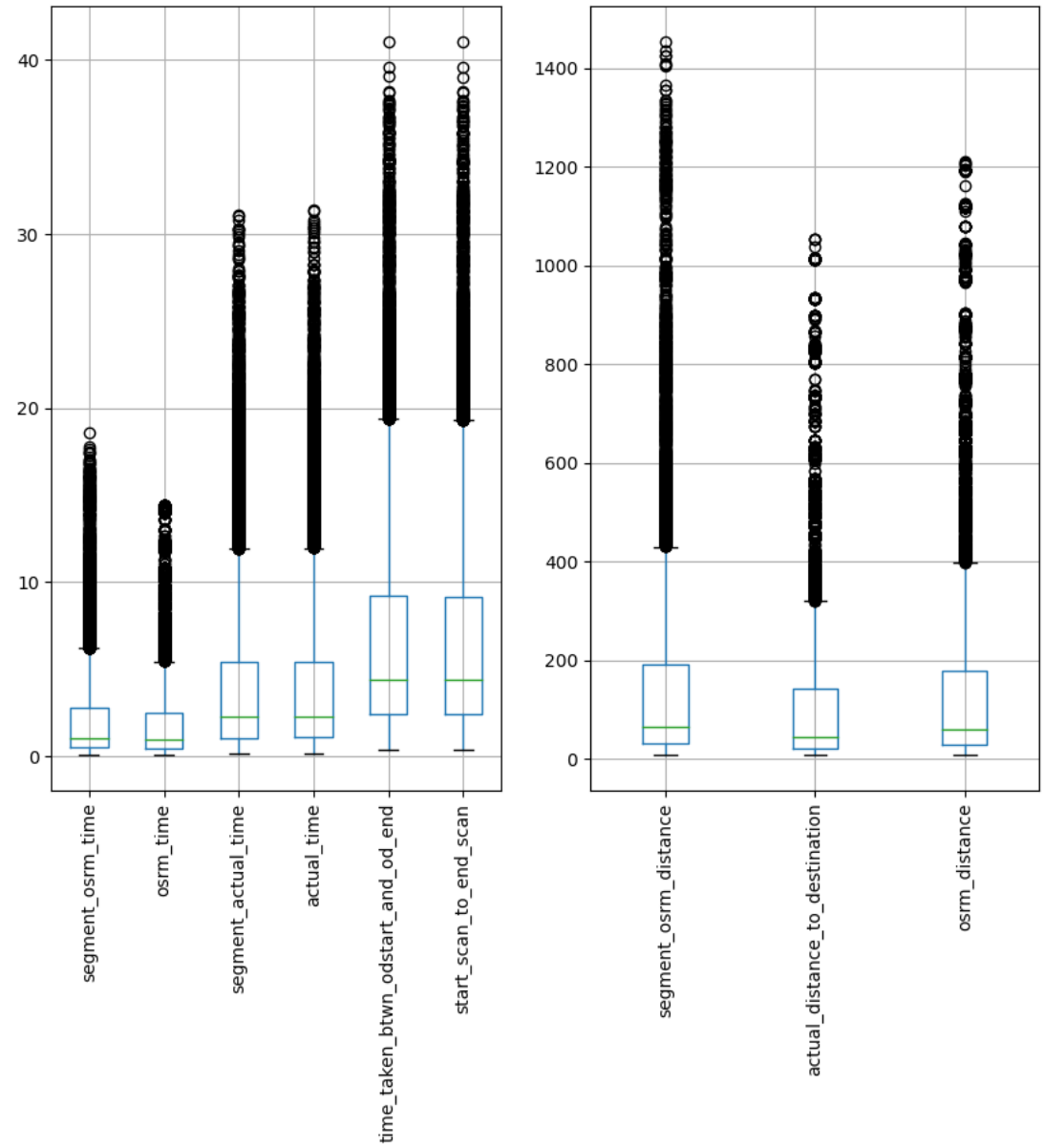
Out[173]:

	trip_uuid	route_type	source_city	destination_city	source_city_state	destination_city_state	source_state	destination_state	segment_osrm
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]	16.8(
1	trip-153671042288605164	Carting	[Tumkur, Doddablpur]	[Doddablpur, Chikblapur]	[Tumkur Karnataka, Doddablpur Karnataka]	[Doddablpur Karnataka, Chikblapur Karnataka]	[Karnataka]	[Karnataka]	1.0(
3	trip-153671046011330457	Carting	[Mumbai]	[Mumbai]	[Mumbai Hub Maharashtra]	[Mumbai Maharashtra]	[Hub Maharashtra]	[Maharashtra]	0.2(
4	trip-153671052974046625	FTL	[Bellary, Hospet, Sandur]	[Hospet, Sandur, Bellary]	[Bellary Karnataka, Hospet Karnataka, Sandur K...	[Hospet Karnataka, Sandur Karnataka, Bellary K...	[Karnataka]	[Karnataka]	1.91
5	trip-153671055416136166	Carting	[Chennai]	[Chennai]	[Chennai Tamil Nadu]	[Chennai Tamil Nadu]	[Tamil Nadu]	[Tamil Nadu]	0.3(
...	...	...	...	...	...	...	...	...	
14812	trip-153861095625827784	Carting	[Chandigarh]	[Zirakpur, Chandigarh]	[Chandigarh Punjab, Chandigarh Chandigarh]	[Zirakpur Punjab, Chandigarh Punjab]	[Punjab, Chandigarh]	[Punjab]	1.03
14813	trip-153861104386292051	Carting	[FBD]	[Faridabad]	[FBD Haryana]	[Faridabad Haryana]	[Haryana]	[Haryana]	0.1(
14814	trip-153861106442901555	Carting	[Kanpur]	[Kanpur]	[Kanpur Uttar Pradesh]	[Kanpur Uttar Pradesh]	[Uttar Pradesh]	[Uttar Pradesh]	1.4(
14815	trip-153861115439069069	Carting	[Tirunelveli, Eral, Tirchchndr, Thisayanvilai,...	[Eral, Tirchchndr, Thisayanvilai, Peikulam, Ti...	[Tirunelveli Tamil Nadu, Eral Tamil Nadu, Tirc...	[Eral Tamil Nadu, Tirchchndr Tamil Nadu, Thisa...	[Tamil Nadu]	[Tamil Nadu]	3.6(
14816	trip-153861118270144424	FTL	[Hospet, Sandur]	[Sandur, Bellary]	[Hospet Karnataka, Sandur Karnataka]	[Sandur Karnataka, Bellary Karnataka]	[Karnataka]	[Karnataka]	1.11

14160 rows × 18 columns

In [174]:
```
trip_records_without_outliers = trip_records_without_outliers[['trip_uuid','route_type','source_city_state', 'destination_city_st
       'segment_actual_time', 'actual_time',
       'time_taken_btwn_odstart_and_od_end', 'start_scan_to_end_scan',
       'segment_osrm_distance', 'actual_distance_to_destination',
       'osrm_distance']]
```

In [175]:
```
plt.figure(figsize = (10,8))
plt.subplot(121)
trip_records_without_outliers[['segment_osrm_time', 'osrm_time',
       'segment_actual_time', 'actual_time',
       'time_taken_btwn_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 encoding :

merging locations details into one columns and re categorise the data as per highest trips having location as top category

In [181]:
```
trip_records_without_outliers["destination_source_locations"] = trip_records_without_outliers["source_city_state"]+" "+trip_recor
trip_records_without_outliers.drop(["source_city_state","destination_city_state"],axis = 1,inplace=True)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in _na_arithmetic_op(left, right, op, is_cmp)
    162     try:
--> 163         result = func(left, right)
    164     except TypeError:

TypeError: unsupported operand type(s) for +: 'float' and 'str'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13612\4241698270.py in <module>
----> 1 trip_records_without_outliers["destination_source_locations"] = trip_records_without_outliers["source_city_state"]+" "+
trip_records_without_outliers["destination_city_state"]
      2 trip_records_without_outliers.drop(["source_city_state","destination_city_state"],axis = 1,inplace=True)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\common.py in new_method(self, other)
     68         other = item_from_zerodim(other)
     69
---> 70         return method(self, other)
     71
     72     return new_method

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\arraylike.py in __add__(self, other)
     98     @unpack_zerodim_and_defer("__add__")
     99     def __add__(self, other):
--> 100         return self._arith_method(other, operator.add)
    101
    102     @unpack_zerodim_and_defer("__radd__")

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\series.py in _arith_method(self, other, op)
   5637     def _arith_method(self, other, op):
   5638         self, other = ops.align_method_SERIES(self, other)
-> 5639         return base.IndexOpsMixin._arith_method(self, other, op)
   5640
   5641

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\base.py in _arith_method(self, other, op)
   1293
   1294         with np.errstate(all="ignore"):
-> 1295             result = ops.arithmetic_op(lvalues, rvalues, op)
   1296
   1297         return self._construct_result(result, name=res_name)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in arithmetic_op(left, right, op)
    220         _bool_arith_check(op, left, right)
    221
--> 222         res_values = _na_arithmetic_op(left, right, op)
    223
    224     return res_values

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in _na_arithmetic_op(left, right, op, is_cmp)
    168             # Don't do this for comparisons, as that will handle complex numbers
    169             #  incorrectly, see GH#32047
--> 170             result = _masked_arith_op(left, right, op)
    171         else:
    172             raise

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in _masked_arith_op(x, y, op)
    125
    126         if mask.any():
--> 127             result[mask] = op(xrav[mask], y)
    128
    129     np.putmask(result, ~mask, np.nan)

TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

Route analysis :

In [183]:
```
A = new_data.groupby("route_schedule_uuid")["route_type"].unique().reset_index()
B = new_data.groupby("route_schedule_uuid")["destination_city"].unique().reset_index()
B.columns = ["route_schedule_uuid","destination_cities"]
C = new_data.groupby("route_schedule_uuid")["source_city"].unique().reset_index()
C.columns = ["route_schedule_uuid","source_cities"]
D = new_data.groupby("route_schedule_uuid")["source_state"].unique().reset_index()
D.columns = ["route_schedule_uuid","source_states"]
E = new_data.groupby("route_schedule_uuid")["destination_state"].unique().reset_index()
E.columns = ["route_schedule_uuid","destination_states"]
F = new_data.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 [184]:
```
I = new_data.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 [185]:
```
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 [ ]:
```
# route_records.sort_values(by="Average_Actual_distance_to_destination",ascending=False)
```

In [186]:
```
route_records.isna().sum()
```

Out[186]:
```
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 [187]:
```
route_records
```

Out[187]:

	route_schedule_uuid	#source_cities	#destination_cities	Number_of_Trips	Average_Actual_distance_to_destination	#source_states	#destination_states
0	thanos::sroute:d010efca-d90d-4977-b987-eae68c5...	13	11	14	281.596486	2	2
1	thanos::sroute:4cbecb35-356b-4b68-bf3c-6225b5e...	10	10	12	332.602225	2	2
2	thanos::sroute:ae5c430f-6153-48d1-8fe5-d5f0bbc...	10	10	20	351.611796	1	1
3	thanos::sroute:f8968c72-5222-4d81-9eed-8a6d88f...	9	9	9	195.257193	1	2
4	thanos::sroute:ed5b80be-7abf-424d-b8cd-d81556a...	9	8	20	178.737233	1	1
...	...	...	...	...	...	...	...
1499	thanos::sroute:9e7bb811-593f-47bc-ac49-ba03ed8...	1	1	19	17.617532	1	1
1500	thanos::sroute:46b9641b-55b5-4b15-b039-2612a50...	1	1	15	10.137219	1	1
1501	thanos::sroute:b48f633d-15cb-4744-a0b9-21df0a9...	1	1	7	15.467701	1	1
1502	thanos::sroute:265efe06-3625-4fba-afee-07b5b64...	0	1	1	236.815038	0	1
1503	thanos::sroute:cfb575b8-df26-48f5-8427-6f48f9d...	0	0	1	50.844665	0	0

1504 rows × 12 columns

Exploratory Data Analysis : (getting some insights from preprocessed data) :

Busiest Route Analysis :

Number of Trips between cities , sorted highest to lowest

Top 20 source and destination cities wihc have high freqency of trips in between .

In [190]:
```
Number_of_trips_between_cities = new_data.groupby(["source_city_state",
                                                   "destination_city_state"])["trip_uuid"].nunique().sort_values(ascending=False).res
Number_of_trips_between_cities.head(25)
```

Out[190]:

	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

From above table, we can observe that Mumbai Maharashtra ,Delhi ,Gurgaon(Haryana),Bengaluru Karnataka ,Hyderabad Telangana,Chennai Tamil Nadu,Ahmedabad Gujarat,Pune Maharashtra,Chandigarh Chandigarh and Kolkata West Bengal are some cities have higest amount of trips happening states with in the city :

In [192]:
```
ies.loc[Number_of_trips_between_cities["source_city_state"] != Number_of_trips_between_cities["destination_city_state"]].head(25)
```

Out[192]:

	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

data not having equal source and destination states , source and destination cities having higest number of trips in between are :

```
delhi to gurgao
Gurgaon,Haryana TO Bengaluru,Karnataka
Bhiwandi/Mumbai,Maharashtra TO Pune Maharashtra
Sonipat TO    Gurgaon,Haryana
```

- it is also been observed that lots of deliveries are happening to airpots
- like : Chennai to MAA chennai international Airport , Pune to Pune Airport (PNQ),Kolkata to CCU West Bengal Kolkata International Airport , Bengluru to BLR-Bengaluru Internation Airport etc.

In [194]:
```
route_records[["Number_of_Trips",
               "Average_Actual_distance_to_destination",
               "#source_cities",
               "#destination_cities"]].sort_values(by="Number_of_Trips",ascending=False).head(25)
```

Out[194]:

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

Inferences and Recommendations :

- 14817 different trips happened between source to destinations during 2018 , September and October.
- 1504 delivery routes on which trips are happenig.
- we have 1508 unique source centers and 1481 unique destination centers

Hypothesis tests Results :

- from 2 sample t-test ,we can also conclude that
- Average time_taken_btwn_odstart_and_od_end for population is equal to Average start_scan_to_end_scan for population.
- population average actual_time is less than population average start_scan_to_end_scan.
- population mean Actual time taken to complete delivery and population mean time_taken_btwn_od_start_and_od_end are also not same.
- Mean of actual time is higher than Mean of the OSRM estimated time for delivery
- Population average for Actual Time taken to complete delivery trip and segment actual time are same.
- Average of OSRM Time & segment-osrm-time for population is not same.
- Population Mean osrm time is less than Population Mean segment osrm time.

Recommendations :

- It is recommended to use Carting (small vehicles) for delivery with in the city in order to reduce the delivery time, and Heavy trucks for long distance trips or heavy load. based on this , we can optimize the delivery time as well as increase the revenue as per requirements.
- Incresing the connectivity between cities can increase the revenue as well as the reputation on connectivity across borders.

In [ ]:

In [ ]:

In [ ]: