In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import t
import scipy.stats as stats
from statistics import mean, median, mode, stdev
import warnings
warnings.filterwarnings("ignore")
from scipy.stats import norm
import math
from numpy import cov
from math import sqrt
from scipy.stats import f_oneway
import calendar
import scipy.integrate as integrate
import scipy.special
from scipy.stats import levene
```

In [2]:

```
dv = pd.read_csv(r"C:\Users\Acer\Downloads\delhivery_data.csv")
```

In [3]:

```
1 dv
```

Out[3]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	so
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND

144867 rows × 24 columns

In [4]:

```
1 dv.describe()
```

Out[4]:

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	14
count	144867.000000	144867.000000	144867.000000	144867.000000	
mean	961.262986	232.926567	234.073372	416.927527	
std	1037.012769	344.755577	344.990009	598.103621	
min	20.000000	9.000000	9.000045	9.000000	
25%	161.000000	22.000000	23.355874	51.000000	
50%	449.000000	66.000000	66.126571	132.000000	
75%	1634.000000	286.000000	286.708875	513.000000	
max	7898.000000	1927.000000	1927.447705	4532.000000	

In [5]:

```
1 dv.isnull().sum()
```

Out[5]:

```
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 293
- destination_name 261

In [6]:

```
1 dv.columns
```

Out[6]:

```
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'],
      dtype='object')
```

Exploring each column

data - tells whether the data is testing or training data

In [7]:

```
1 dv.head()
```

Out[7]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_c
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812

5 rows × 24 columns

In [8]:

```
1 dv.data.unique()
```

Out[8]:

```
array(['training', 'test'], dtype=object)
```

In [9]:

```
1 len(dv.data)
```

Out[9]:

```
144867
```

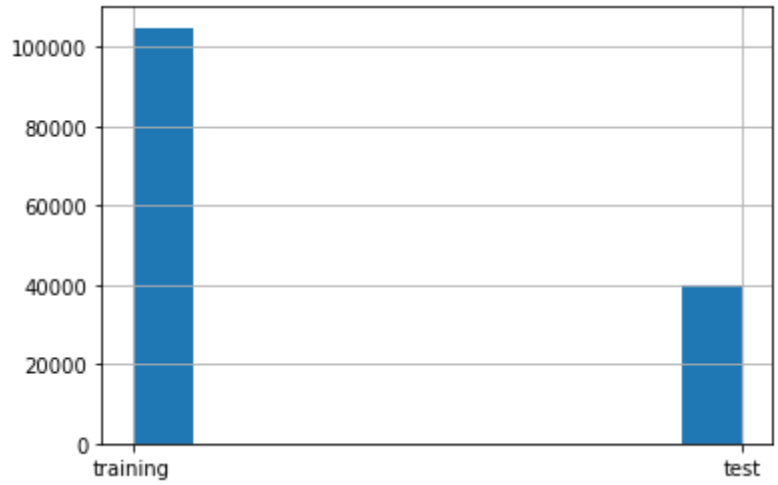
In [10]:

```
1 print(dv.data.value_counts())
2 dv.data.hist()
```

```
training    104858
test         40009
Name: data, dtype: int64
```

Out[10]:

```
<AxesSubplot:>
```



- train and test data imbalance in numbers but zero missing elements

trip_creation_time – Timestamp of trip creation

In [11]:

```
1 dv.head()
```

Out[11]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_c
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812

5 rows × 24 columns

In [12]:

```
1 print('year / time - HH-MM-SSSSSS')
2 dv.trip_creation_time.unique()
```

```
year / time - HH-MM-SSSSSS
```

Out[12]:

```
array(['2018-09-20 02:35:36.476840', '2018-09-23 06:42:06.021680',
       '2018-09-14 15:42:46.437249', ..., '2018-09-22 11:30:41.399439',
       '2018-09-17 11:35:28.838714', '2018-09-20 16:24:28.436231'],
      dtype=object)
```

In [13]:

```
1 print('alot of same trip creation time - makes sense because multiple orders in sec pos
2 dv.trip_creation_time.nunique()
```

```
alot of same trip creation time - makes sense because multiple orders in sec
possible
```

Out[13]:

```
14817
```

In [14]:

```
print(dv.trip_creation_time.value_counts())
# dv.trip_creation_time.hist()
```

```
2018-09-28 05:23:15.359220    101
2018-10-02 06:05:53.086094    101
2018-09-27 04:47:19.425867    101
2018-09-22 04:55:04.835022    101
2018-09-29 05:04:57.639067    101
                             ...
2018-09-27 18:08:18.207639      1
2018-09-28 17:31:07.690205      1
2018-09-29 14:56:33.655170      1
2018-09-19 04:35:44.776558      1
2018-09-14 17:04:32.989471      1
Name: trip_creation_time, Length: 14817, dtype: int64
```

In [ ]:

```

```

route_schedule_uuid – Unique Id for a particular route schedule

In [15]:

```
dv.head()
```

Out[15]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_c
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812

5 rows × 24 columns

In [16]:

```
dv.route_schedule_uuid.unique()

```

Out[16]:

```
array(['thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3297ef',
       'thanos::sroute:ff52ef7a-4d0d-4063-9bfe-cc211728881b',
       'thanos::sroute:a16bfa03-3462-4bce-9c82-5784c7d315e6', ...,
       'thanos::sroute:72cf9feb-f4e3-4a55-b92a-0b686ee8fabc',
       'thanos::sroute:5e08be79-8a4c-4a91-a514-5350403c0e31',
       'thanos::sroute:a3c30562-87e5-471c-9646-0ed49c150996'],
      dtype=object)
```

In [17]:

```
dv.route_schedule_uuid.nunique()

```

Out[17]:

```
1504
```

In [18]:

```
144867 // 1504
```

Out[18]:

```
96
```

In [19]:

```
dv.route_schedule_uuid.value_counts()
```

Out[19]:

```
thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069fbcea9    1812
thanos::sroute:0456b740-1dad-4929-bbe0-87d8843f5a10    1608
thanos::sroute:dca6268f-741a-4d1a-b1b0-aab13095a366    1605
thanos::sroute:a1b25549-1e77-498f-8538-00292e5bd5a2    1285
thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e5720d    1280
                                                       ...
thanos::sroute:d563d17e-2123-40a4-9eec-40018966caba       1
thanos::sroute:036f372d-28d8-4d19-877c-6277077ad09e       1
thanos::sroute:e00eb6aa-d792-4b28-81fa-fdee413ef326       1
thanos::sroute:889b9cf5-da6a-48ce-b3bd-6983c8090164       1
thanos::sroute:404cbabf-d2a5-4e46-bf79-8b3c518f082b       1
Name: route_schedule_uuid, Length: 1504, dtype: int64
```

In [ ]:

```

```

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)

In [20]:

```
1 dv.head()
```

Out[20]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_c
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812

5 rows × 24 columns

In [21]:

```
1 dv.route_type.unique()
2
```

Out[21]:

```
array(['Carting', 'FTL'], dtype=object)
```

In [270]:

```
1 route_type = dv.groupby("route_schedule_uuid")["route_type"].unique().reset_index()["ro
2 route_type
```

Out[270]:

```
Carting    922
FTL        582
Name: route_type, dtype: int64
```

In [269]:

```
print('more through the truks than carting ')
dv.groupby("route_schedule_uuid")["route_type"].unique().reset_index()["route_type"].ap
```

```
more through the truks than carting
```

Out[269]:

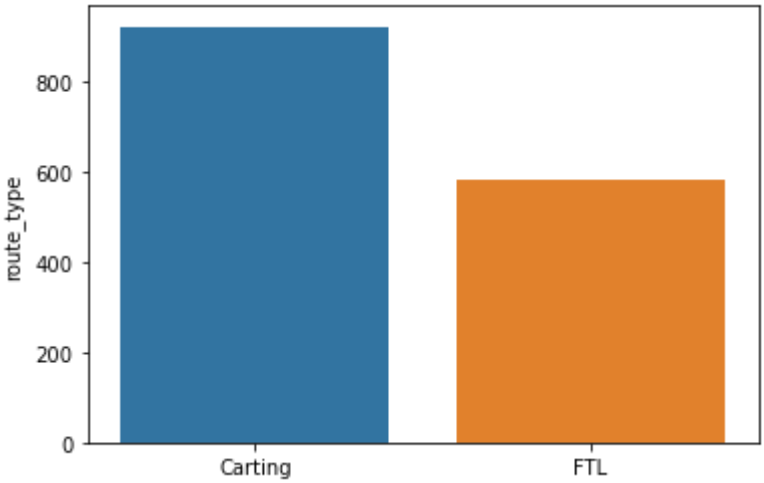
```
Carting    922
FTL        582
Name: route_type, dtype: int64
```

In [272]:

```
sns.barplot(x= route_type.index,
            y = route_type)
```

Out[272]:

```
<AxesSubplot:ylabel='route_type'>
```



Observations

From 1504 total different routes , we have
922 (61%) of the routes are Carting , which consists of small vehicles and
582 (38.69%) of total routes are FTL : which are Full Truck Load get to the destin
ation sooner. as no otther pickups or drop offs along the way .

trip_uuid - Unique ID given to a particular trip (A trip may include different source and destination centers)

In [23]:

```
1 dv.trip_uuid.value_counts()
```

Out[23]:

```
trip-153811219535896559    101
trip-153846035308581166    101
trip-153802363942560700    101
trip-153759210483476123    101
trip-153819749763881430    101
                          ...
trip-153807169820740041      1
trip-153815586768995663      1
trip-153823299365493206      1
trip-153733174477629450      1
trip-153694467298919626      1
Name: trip_uuid, Length: 14817, dtype: int64
```

In [24]:

```
1 dv.trip_uuid.nunique()
```

Out[24]:

```
14817
```

In [25]:

```
1 print('on a average atleast 9 trasit can be present inside a total source to destinatic
2 144867 // 14817
```

```
on a average atleast 9 trasit can be present inside a total source to destin
ation travel
```

Out[25]:

```
9
```

Observations

we have 14817 different trips happended between source to destinations.

source_center - Source ID of trip origin

In [26]:

```
1 dv.source_center.unique()
2
```

Out[26]:

```
array(['IND388121AAA', 'IND388620AAB', 'IND421302AAG', ...,
       'IND361335AAA', 'IND562132AAC', 'IND639104AAB'], dtype=object)
```

In [27]:

```
dv.source_center.nunique()

```

Out[27]:

```
1508
```

In [28]:

```
dv.source_center.value_counts()

```

Out[28]:

```
IND000000ACB    23347
IND562132AAA     9975
IND421302AAG     9088
IND411033AAA     4061
IND501359AAE     3340
                ...
IND741121AAA        1
IND207123AAA        1
IND242001AAA        1
IND222001AAA        1
IND741101AAB        1
Name: source_center, Length: 1508, dtype: int64
```

In [ ]:

```

```

Observations

- looks like source id and source name are the same things , name is the name of warehouse and id is the unique id for the same but there is a difference of 10

source_name - Source Name of trip origin(has null rows)

In [29]:

```
1 dv.head(5)
```

Out[29]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_c
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812

5 rows × 24 columns

In [30]:

```
1 dv.source_name.unique()
2
```

Out[30]:

```
array(['Anand_VUNagar_DC (Gujarat)', 'Khambhat_MotvdDPP_D (Gujarat)',
       'Bhiwandi_Mankoli_HB (Maharashtra)', ...,
       'Dwarka_StnRoad_DC (Gujarat)', 'Bengaluru_Nelmngla_L (Karnataka)',
       'Kulithalai_AnnaNGR_D (Tamil Nadu)'], dtype=object)
```

In [31]:

```
1 dv.source_name.nunique()
2
```

Out[31]:

1498

In [32]:

```
print('by the look of it , it must be the source of the inventory')
dv.source_name.value_counts()
```

```
by the look of it , it must be the source of the inventory
```

Out[32]:

```
Gurgaon_Bilaspur_HB (Haryana)            23347
Bangalore_Nelmngla_H (Karnataka)          9975
Bhiwandi_Mankoli_HB (Maharashtra)         9088
Pune_Tathawde_H (Maharashtra)             4061
Hyderabad_Shamshbd_H (Telangana)          3340
                                         ...
Shahjhnpur_NavdaCln_D (Uttar Pradesh)        1
Soro_UttarDPP_D (Orissa)                     1
Kayamkulam_Bhrnikvu_D (Kerala)               1
Krishnanagar_AnadiDPP_D (West Bengal)        1
Faridabad_Old (Haryana)                      1
Name: source_name, Length: 1498, dtype: int64
```

In [ ]:

```

```

destination_cente – Destination ID

In [33]:

```
dv.destination_center.unique()

```

Out[33]:

```
array(['IND388620AAB', 'IND388320AAA', 'IND411033AAA', ...,
       'IND600004AAA', 'IND134203AAA', 'IND400701AAA'], dtype=object)
```

In [34]:

```
dv.destination_center.nunique()

```

Out[34]:

```
1481
```

In [35]:

```
dv.destination_center.value_counts()

```

Out[35]:

```
IND000000ACB    15192
IND562132AAA    11019
IND421302AAG     5492
IND501359AAE     5142
IND712311AAA     4892
                ...
IND520011AAA        1
IND741201AAC        1
IND400705AAA        1
IND110046AAA        1
IND504215AAA        1
Name: destination_center, Length: 1481, dtype: int64
```

In [ ]:

```

```

destination_name – Destination Name(null vals)

In [36]:

```
dv.destination_name.unique()

```

Out[36]:

```
array(['Khambhat_MotvdDPP_D (Gujarat)', 'Anand_Vaghasi_IP (Gujarat)',
       'Pune_Tathawde_H (Maharashtra)', ...,
       'Chennai_Mylapore (Tamil Nadu)', 'Naraingarh_Ward2DPP_D (Haryana)',
       'Mumbai_Ghansoli_DC (Maharashtra)'], dtype=object)
```

In [37]:

```
dv.destination_name.nunique()

```

Out[37]:

```
1468
```

In [38]:

```
dv.destination_name.value_counts()

```

Out[38]:

```
Gurgaon_Bilaspur_HB (Haryana)        15192
Bangalore_Nelmngla_H (Karnataka)     11019
Bhiwandi_Mankoli_HB (Maharashtra)     5492
Hyderabad_Shamshbd_H (Telangana)      5142
Kolkata_Dankuni_HB (West Bengal)      4892
                                     ...
Hyd_Trimulgherry_Dc (Telangana)          1
Vijayawada (Andhra Pradesh)              1
Baghpat_Barout_D (Uttar Pradesh)         1
Mumbai_Sanpada_CP (Maharashtra)          1
Basta_Central_DPP_1 (Orissa)             1
Name: destination_name, Length: 1468, dtype: int64
```

In [ ]:

```

```

the destination center and destination id just like the source are the same but a difference of 13 missing values

In [ ]:

```

```

od_start_time – Trip start time

In [39]:

```
dv.od_start_time
```

Out[39]:

```
0         2018-09-20 03:21:32.418600
1         2018-09-20 03:21:32.418600
2         2018-09-20 03:21:32.418600
3         2018-09-20 03:21:32.418600
4         2018-09-20 03:21:32.418600
                     ...
144862    2018-09-20 16:24:28.436231
144863    2018-09-20 16:24:28.436231
144864    2018-09-20 16:24:28.436231
144865    2018-09-20 16:24:28.436231
144866    2018-09-20 16:24:28.436231
Name: od_start_time, Length: 144867, dtype: object
```

In [40]:

```
1 dv.od_start_time.unique()
```

Out[40]:

```
array(['2018-09-20 03:21:32.418600', '2018-09-20 04:47:45.236797',
       '2018-09-23 06:42:06.021680', ..., '2018-09-22 11:30:41.399439',
       '2018-09-17 11:35:28.838714', '2018-09-20 16:24:28.436231'],
      dtype=object)
```

In [41]:

```
1 dv.od_start_time.nunique()
```

Out[41]:

```
26369
```

In [42]:

```
1 dv.od_start_time.value_counts()
```

Out[42]:

```
2018-09-21 18:37:09.322207    81
2018-10-03 04:55:30.039225    79
2018-09-30 05:56:48.299467    79
2018-09-26 05:33:10.899941    79
2018-09-14 07:13:24.396869    79
                              ..
2018-09-25 04:58:40.930230     1
2018-09-22 05:19:27.704727     1
2018-09-25 09:05:19.576386     1
2018-09-20 02:16:44.645390     1
2018-09-27 02:59:59.877566     1
Name: od_start_time, Length: 26369, dtype: int64
```

- there has been same start time for various products.

In [43]:

```
1 dv['od_start_time'] = pd.to_datetime(dv['od_start_time'])
2
```

In [ ]:

```
1
```

In [ ]:

```
1
```

od_start_time + od_end_time = actual_time

od_end_time – Trip end time

In [44]:

```
1 dv.od_end_time
```

Out[44]:

```
0         2018-09-20 04:47:45.236797
1         2018-09-20 04:47:45.236797
2         2018-09-20 04:47:45.236797
3         2018-09-20 04:47:45.236797
4         2018-09-20 04:47:45.236797
                     ...
144862    2018-09-20 23:32:09.618069
144863    2018-09-20 23:32:09.618069
144864    2018-09-20 23:32:09.618069
144865    2018-09-20 23:32:09.618069
144866    2018-09-20 23:32:09.618069
Name: od_end_time, Length: 144867, dtype: object
```

In [45]:

```
1 dv.od_end_time.unique()
```

Out[45]:

```
array(['2018-09-20 04:47:45.236797', '2018-09-20 06:36:55.627764',
       '2018-09-23 11:44:28.365845', ..., '2018-09-22 21:45:05.128533',
       '2018-09-17 13:32:21.128357', '2018-09-20 23:32:09.618069'],
      dtype=object)
```

In [46]:

```
1 dv.od_end_time.nunique()
```

Out[46]:

```
26369
```

In [47]:

```
1 dv.od_end_time.value_counts()
```

Out[47]:

```
2018-09-24 09:59:15.691618    81
2018-10-05 11:15:01.115906    79
2018-10-02 10:36:25.970169    79
2018-09-28 12:13:41.675546    79
2018-09-16 17:00:03.263746    79
                              ..
2018-09-25 13:57:24.614624     1
2018-09-22 06:11:51.820645     1
2018-09-25 15:47:35.535055     1
2018-09-20 04:04:45.620456     1
2018-09-27 15:47:50.386008     1
Name: od_end_time, Length: 26369, dtype: int64
```

- the start time and end time length match = this must be the time from warehouse to doorstep ???

In [48]:

```
dv['od_end_time'] = pd.to_datetime(dv['od_end_time'])
```

Calculate the time taken between od_start_time and od_end_time and keep it as a feature. Drop the original columns, if required

In [49]:

```
dv['od_end_time'] - dv['od_start_time']
```

Out[49]:

```
0        0 days 01:26:12.818197
1        0 days 01:26:12.818197
2        0 days 01:26:12.818197
3        0 days 01:26:12.818197
4        0 days 01:26:12.818197
                  ...
144862   0 days 07:07:41.181838
144863   0 days 07:07:41.181838
144864   0 days 07:07:41.181838
144865   0 days 07:07:41.181838
144866   0 days 07:07:41.181838
Length: 144867, dtype: timedelta64[ns]
```

In [50]:

```
dv['time_taken'] = (dv['od_end_time'] - dv['od_start_time'])/pd.Timedelta(1,unit = 'hou
dv['time_taken']
```

Out[50]:

```
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, Length: 144867, dtype: float64
```

Continuing with column analysing

start_scan_to_end_scan – Time taken to deliver from source to destination

In [51]:

```
1 dv
```

Out[51]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	so
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND

144867 rows × 25 columns

In [52]:

```
1 dv.start_scan_to_end_scan.max()
```

Out[52]:

```
7898.0
```

In [53]:

```
1 dv.start_scan_to_end_scan.unique()
```

Out[53]:

```
array([  86.,  109.,  302., ..., 2476., 1161., 2949.])
```

In [54]:

```
1 dv.start_scan_to_end_scan.nunique()
```

Out[54]:

```
1915
```

In [55]:

```
1 print('time taken to deliver from source warehouse to destination warehouse')
2 (dv.start_scan_to_end_scan /(60))
```

```
time taken to deliver from source warehouse to destination warehouse
```

Out[55]:

```
0         1.433333
1         1.433333
2         1.433333
3         1.433333
4         1.433333
            ...
144862    7.116667
144863    7.116667
144864    7.116667
144865    7.116667
144866    7.116667
Name: start_scan_to_end_scan, Length: 144867, dtype: float64
```

In [ ]:

```
1
```

In [56]:

```
1 dv.cutoff_timestamp
```

Out[56]:

```
0                2018-09-20 04:27:55
1                2018-09-20 04:17:55
2         2018-09-20 04:01:19.505586
3                2018-09-20 03:39:57
4                2018-09-20 03:33:55
                     ...
144862           2018-09-20 21:57:20
144863           2018-09-20 21:31:18
144864           2018-09-20 21:11:18
144865           2018-09-20 20:53:19
144866    2018-09-20 16:24:28.436231
Name: cutoff_timestamp, Length: 144867, dtype: object
```

In [ ]:

```
1
```

actual_distance_to_destination – Distance in Kms between source and destination warehouse

In [57]:

```
1 dv.actual_distance_to_destination
```

Out[57]:

```
0         10.435660
1         18.936842
2         27.637279
3         36.118028
4         39.386040
            ...
144862    45.258278
144863    54.092531
144864    66.163591
144865    73.680667
144866    70.039010
Name: actual_distance_to_destination, Length: 144867, dtype: float64
```

In [58]:

```
1 dv.actual_distance_to_destination.nunique()
```

Out[58]:

```
144515
```

In [59]:

```
1 dv.actual_distance_to_destination.value_counts()
```

Out[59]:

```
100.282892    2
19.122553     2
44.552586     2
27.297724     2
23.955638     2
             ..
18.495678     1
9.380550      1
10.583412     1
9.162213      1
70.039010     1
Name: actual_distance_to_destination, Length: 144515, dtype: int64
```

- does destination warehouse differ >>??? is middle warehouses considered destination at each split ??

In [ ]:

```
1
```

actual_time – Actual time taken to complete the delivery (Cumulative)

In [60]:

```
dv.actual_time

```

Out[60]:

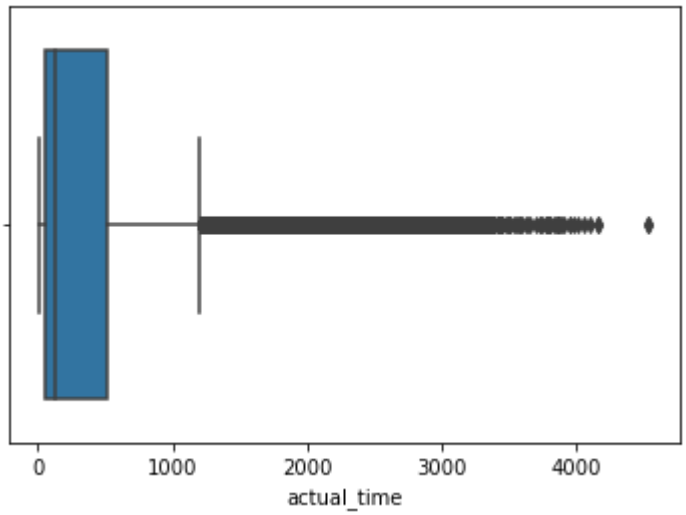
```
0          14.0
1          24.0
2          40.0
3          62.0
4          68.0
          ...
144862     94.0
144863    120.0
144864    140.0
144865    158.0
144866    426.0
Name: actual_time, Length: 144867, dtype: float64
```

In [61]:

```
sns.boxplot(dv.actual_time)
```

Out[61]:

```
<AxesSubplot:xlabel='actual_time'>
```



In [62]:

```
(dv.actual_time // (24*60)).min()
```

Out[62]:

```
0.0
```

In [63]:

```
dv.actual_time.max()
```

Out[63]:

```
4532.0
```

In [64]:

```
1 4532.0 // 60
```

Out[64]:

```
75.0
```

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)

In [65]:

```
1 dv.osrm_time
```

Out[65]:

```
0         11.0
1         20.0
2         28.0
3         40.0
4         44.0
          ...
144862    60.0
144863    76.0
144864    88.0
144865    98.0
144866    95.0
Name: osrm_time, Length: 144867, dtype: float64
```

In [66]:

```
1 dv.osrm_time.max()
```

Out[66]:

```
1686.0
```

- actual time is thrice the OSRM time.

In [ ]:

```
1
```

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)

In [67]:

```
1 dv.osrm_distance
```

Out[67]:

```
0          11.9653
1          21.7243
2          32.5395
3          45.5620
4          54.2181
            ...
144862     67.9280
144863     85.6829
144864     97.0933
144865    111.2709
144866     88.7319
Name: osrm_distance, Length: 144867, dtype: float64
```

In [68]:

```
1 dv.osrm_distance.max()
```

Out[68]:

```
2326.1991000000003
```

In [69]:

```
1 dv.osrm_distance.unique()
```

Out[69]:

```
array([ 11.9653,  21.7243,  32.5395, ...,  97.0933, 111.2709,  88.7319])
```

In [70]:

```
1 dv.osrm_distance.nunique()
```

Out[70]:

```
138046
```

In [ ]:

```
1
```

In [71]:

```
1 dv.factor
```

Out[71]:

```
0         1.272727
1         1.200000
2         1.428571
3         1.550000
4         1.545455
            ...
144862    1.566667
144863    1.578947
144864    1.590909
144865    1.612245
144866    4.484211
Name: factor, Length: 144867, dtype: float64
```

Observation from analysing each and every column

Observations

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-tiem 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 gives the time and osrm distance gives the shortest distance (given cumulatively)

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 of package delivery)

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

Compare the difference between Point a. and start_scan_to_end_scan. Do hypothesis testing/ Visual analysis to check.

In [72]:

```
dv_scantime = dv.groupby('trip_uuid')['start_scan_to_end_scan'].unique()/60
dv_scantime_reset = dv_scantime.reset_index()
dv_scantime_reset['start_scan_to_end_scan'].apply(sum)
```

Out[72]:

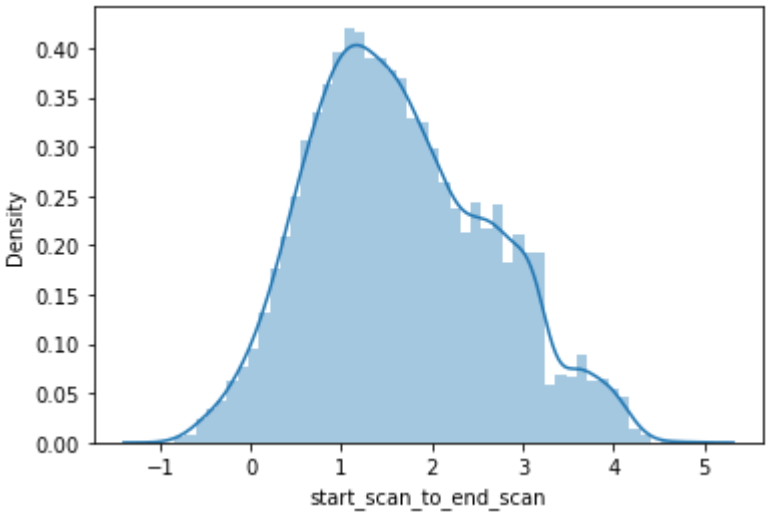
```
0        37.650000
1         3.000000
2        65.550000
3         1.666667
4        11.950000
           ...
14812     4.283333
14813     1.000000
14814     7.016667
14815     5.783333
14816     5.883333
Name: start_scan_to_end_scan, Length: 14817, dtype: float64
```

In [73]:

```
sns.distplot(np.log(dv_scantime_reset['start_scan_to_end_scan'].apply(sum)))
```

Out[73]:

```
<AxesSubplot:xlabel='start_scan_to_end_scan', ylabel='Density'>
```



In [74]:

```
# dv_scantime[]
```

In [75]:

```
dv_timetaken = dv.groupby('trip_uuid')['time_taken'].unique().reset_index()
dv_timetaken
```

Out[75]:

	trip_uuid	time_taken
0	trip-153671041653548748	[16.65842298, 21.0100736875]
1	trip-153671042288605164	[2.0463247669444447, 0.9805397955555556]
2	trip-153671043369099517	[51.662059856388886, 13.910648811388889]
3	trip-153671046011330457	[1.6749155866666667]
4	trip-153671052974046625	[2.5335485744444446, 1.3423885633333332, 8.096...
...	...	...
14812	trip-153861095625827784	[2.546464057777778, 1.7540180775]
14813	trip-153861104386292051	[1.0098420219444444]
14814	trip-153861106442901555	[2.895179575833333, 4.1401515375]
14815	trip-153861115439069069	[1.7609491794444445, 0.7362400538888889, 1.035...
14816	trip-153861118270144424	[1.1155594141666667, 4.7912334425]

14817 rows × 2 columns

In [76]:

```
dv_timetaken['time_taken'].apply(sum)
```

Out[76]:

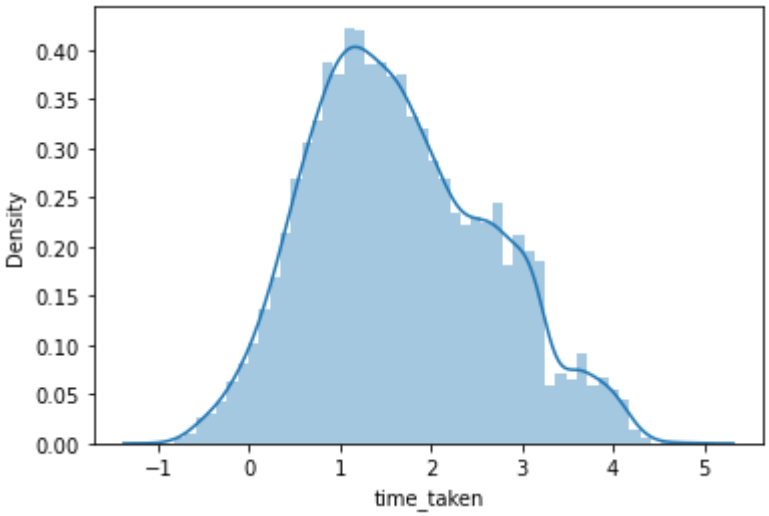
```
0        37.668497
1         3.026865
2        65.572709
3         1.674916
4        11.972484
           ...
14812     4.300482
14813     1.009842
14814     7.035331
14815     5.808548
14816     5.906793
Name: time_taken, Length: 14817, dtype: float64
```

In [77]:

```
1 sns.distplot(np.log(dv_timetaken['time_taken'].apply(sum)))
```

Out[77]:

```
<AxesSubplot:xlabel='time_taken', ylabel='Density'>
```



In [ ]:

```
1
```

In [79]:

```
1 stats.ttest_ind(np.log(dv_scantime_reset['start_scan_to_end_scan'].apply(sum)),np.log(
```

Out[79]:

```
Ttest_indResult(statistic=-0.4066930332152617, pvalue=0.6842363952379178)
```

Observations

```
(time_taken_btwn_odstart_and_od_end and start_scan_to_end_scan) are closly simila
r.
from 2 sample t-test ,
we can also conclude that Average time_taken_btwn_odstart_and_od_end for populatio
n is
also equal to Average start_scan_to_end_scan for population.
```

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)

In [ ]:

```
1
```

In [80]:

```
1 (dv.groupby(["trip_uuid","time_taken"])["actual_time"].max()).reset_index()
2
```

Out[80]:

	trip_uuid	time_taken	actual_time
0	trip-153671041653548748	16.658423	830.0
1	trip-153671041653548748	21.010074	732.0
2	trip-153671042288605164	0.980540	47.0
3	trip-153671042288605164	2.046325	96.0
4	trip-153671043369099517	13.910649	611.0
...	...	...	...
26364	trip-153861115439069069	1.035253	51.0
26365	trip-153861115439069069	1.518130	90.0
26366	trip-153861115439069069	1.760949	60.0
26367	trip-153861118270144424	1.115559	42.0
26368	trip-153861118270144424	4.791233	233.0

26369 rows × 3 columns

In [81]:

```
1 actual_time = ((dv.groupby(["trip_uuid","time_taken"])["actual_time"].max()).reset_inde
2 actual_time
```

Out[81]:

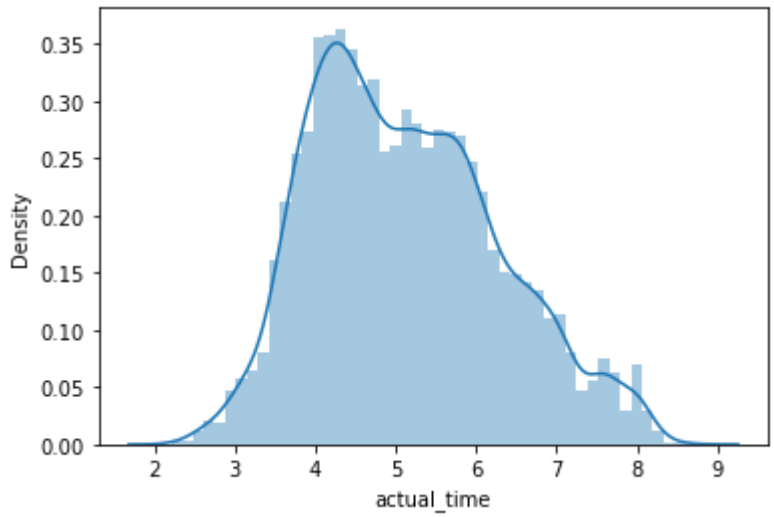
```
trip_uuid
trip-153671041653548748    1562.0
trip-153671042288605164     143.0
trip-153671043369099517    3347.0
trip-153671046011330457      59.0
trip-153671052974046625     341.0
                            ...
trip-153861095625827784      83.0
trip-153861104386292051      21.0
trip-153861106442901555     282.0
trip-153861115439069069     264.0
trip-153861118270144424     275.0
Name: actual_time, Length: 14817, dtype: float64
```

In [82]:

```
1 sns.distplot(np.log(actual_time))
```

Out[82]:

```
<AxesSubplot:xlabel='actual_time', ylabel='Density'>
```



In [83]:

```
1 (dv.groupby(["trip_uuid","time_taken"])["osrm_time"].max()).reset_index()
```

Out[83]:

	trip_uuid	time_taken	osrm_time
0	trip-153671041653548748	16.658423	394.0
1	trip-153671041653548748	21.010074	349.0
2	trip-153671042288605164	0.980540	26.0
3	trip-153671042288605164	2.046325	42.0
4	trip-153671043369099517	13.910649	212.0
...	...	...	...
26364	trip-153861115439069069	1.035253	41.0
26365	trip-153861115439069069	1.518130	48.0
26366	trip-153861115439069069	1.760949	50.0
26367	trip-153861118270144424	1.115559	26.0
26368	trip-153861118270144424	4.791233	42.0

26369 rows × 3 columns

In [84]:

```
osrm_time = ((dv.groupby(["trip_uuid","time_taken"])["osrm_time"].max()).reset_index())
osrm_time
```

Out[84]:

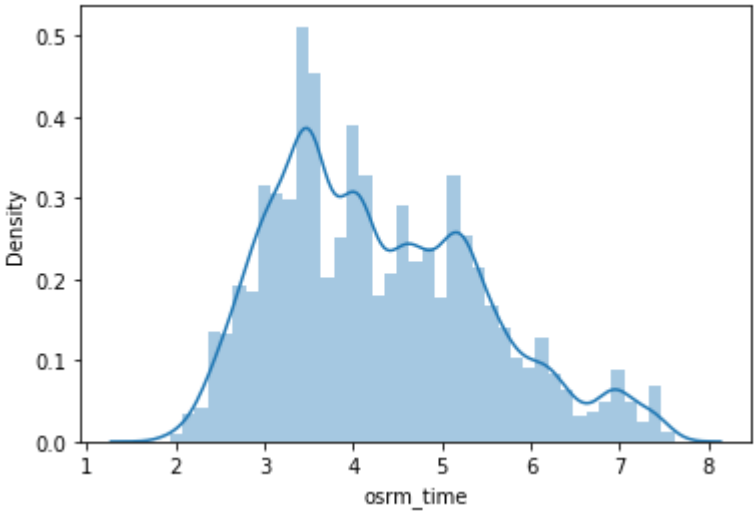
```
trip_uuid
trip-153671041653548748     743.0
trip-153671042288605164      68.0
trip-153671043369099517    1741.0
trip-153671046011330457      15.0
trip-153671052974046625     117.0
                           ...
trip-153861095625827784      62.0
trip-153861104386292051      12.0
trip-153861106442901555      54.0
trip-153861115439069069     184.0
trip-153861118270144424      68.0
Name: osrm_time, Length: 14817, dtype: float64
```

In [85]:

```
sns.distplot(np.log(osrm_time))
```

Out[85]:

```
<AxesSubplot:xlabel='osrm_time', ylabel='Density'>
```



In [ ]:

```

```

In [86]:

```
stats.ttest_ind(np.log(actual_time),np.log(osrm_time))
```

Out[86]:

```
Ttest_indResult(statistic=60.65444898882128, pvalue=0.0)
```

Observations

from ttestwe can conclude , tht population mean actual time taken to complete deli
vert from source to warehouse
and orsm estimate mean time for population are not same.
actual time is higher than the osrm estimated time for delivery.

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)

In [88]:

```
# actual_time , segment_actual_time
```

In [89]:

```
dv[dv['trip_uuid'] == 'trip-153741093647649320'][['trip_uuid' ,'time_taken', 'actual_ti
```

Out[89]:

	trip_uuid	time_taken	actual_time	segment_actual_time
0	trip-153741093647649320	1.436894	14.0	14.0
1	trip-153741093647649320	1.436894	24.0	10.0
2	trip-153741093647649320	1.436894	40.0	16.0
3	trip-153741093647649320	1.436894	62.0	21.0
4	trip-153741093647649320	1.436894	68.0	6.0
5	trip-153741093647649320	1.819553	15.0	15.0
6	trip-153741093647649320	1.819553	44.0	28.0
7	trip-153741093647649320	1.819553	65.0	21.0
8	trip-153741093647649320	1.819553	76.0	10.0
9	trip-153741093647649320	1.819553	102.0	26.0

In [90]:

```
# segment_actual_time - This is a segment time. Time taken by the subset of the package
# actual_time - Actual time taken to complete the delivery (Cumulative)
```

In [91]:

```
actual_time = ((dv.groupby(["trip_uuid","time_taken"])["actual_time"].max()).reset_inde
actual_time
```

Out[91]:

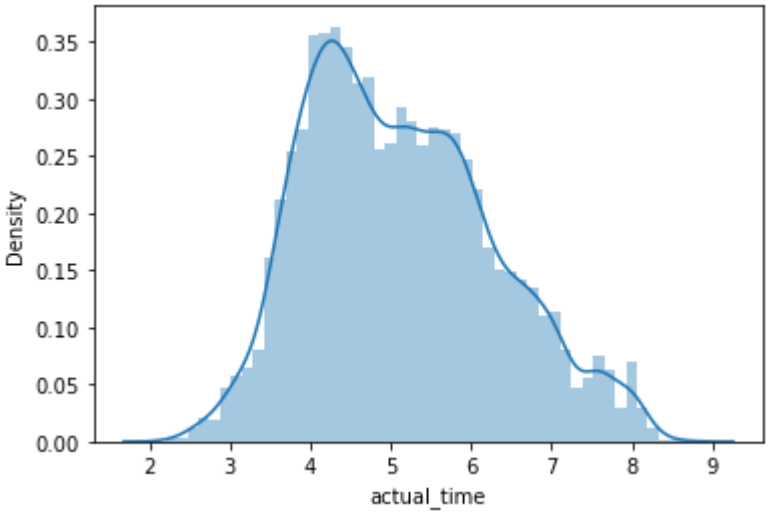
```
trip_uuid
trip-153671041653548748    1562.0
trip-153671042288605164     143.0
trip-153671043369099517    3347.0
trip-153671046011330457      59.0
trip-153671052974046625     341.0
                            ...
trip-153861095625827784      83.0
trip-153861104386292051      21.0
trip-153861106442901555     282.0
trip-153861115439069069     264.0
trip-153861118270144424     275.0
Name: actual_time, Length: 14817, dtype: float64
```

In [92]:

```
sns.distplot(np.log(actual_time))
```

Out[92]:

```
<AxesSubplot:xlabel='actual_time', ylabel='Density'>
```



In [93]:

```
segment_actual_time = ((dv.groupby(["trip_uuid","time_taken"])["segment_actual_time"].s
segment_actual_time
```

Out[93]:

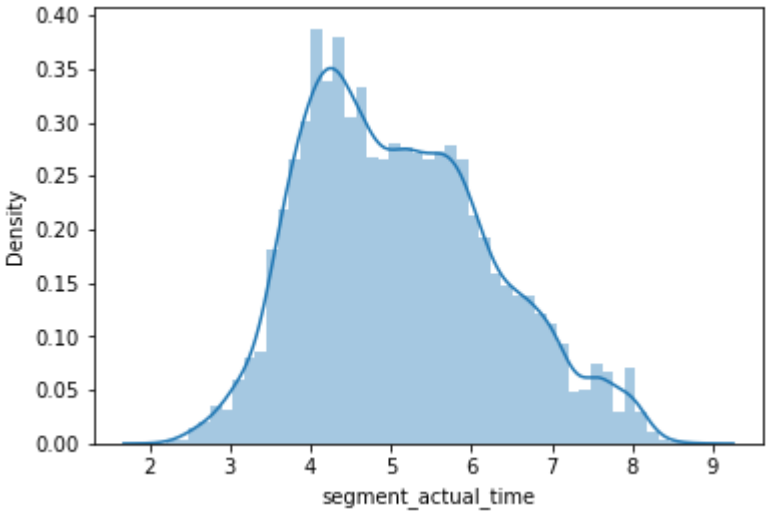
```
trip_uuid
trip-153671041653548748    1548.0
trip-153671042288605164     141.0
trip-153671043369099517    3308.0
trip-153671046011330457      59.0
trip-153671052974046625     340.0
                            ...
trip-153861095625827784      82.0
trip-153861104386292051      21.0
trip-153861106442901555     281.0
trip-153861115439069069     258.0
trip-153861118270144424     274.0
Name: segment_actual_time, Length: 14817, dtype: float64
```

In [94]:

```
sns.distplot(np.log(segment_actual_time))
```

Out[94]:

```
<AxesSubplot:xlabel='segment_actual_time', ylabel='Density'>
```



In [ ]:

```

```

In [95]:

```
stats.ttest_ind(np.log(actual_time),np.log(segment_actual_time))
```

Out[95]:

```
Ttest_indResult(statistic=0.7701458929626459, pvalue=0.4412194950197108)
```

In [ ]:

```
1
```

In [ ]:

```
1
```

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)

In [96]:

```
1 print('osrm_distance - An open-source routing engine which computes the shortest path b
2 dv.osrm_distance
```

```
osrm_distance - An open-source routing engine which computes the shortest pa
th between points in a given map (Includes usual traffic, distance through m
ajor and minor roads) (Cumulative)
```

Out[96]:

```
0          11.9653
1          21.7243
2          32.5395
3          45.5620
4          54.2181
            ...
144862     67.9280
144863     85.6829
144864     97.0933
144865    111.2709
144866     88.7319
Name: osrm_distance, Length: 144867, dtype: float64
```

In [97]:

```
1 print('segment_osrm_distance - This is the OSRM distance. Distance covered by subset of
2 dv.segment_osrm_distance
```

```
segment_osrm_distance - This is the OSRM distance. Distance covered by subse
t of the package delivery
```

Out[97]:

```
0         11.9653
1          9.7590
2         10.8152
3         13.0224
4          3.9153
            ...
144862     8.1858
144863    17.3725
144864    20.7053
144865    18.8885
144866     8.8088
Name: segment_osrm_distance, Length: 144867, dtype: float64
```

In [98]:

```
1 dv[dv['trip_uuid'] == 'trip-153741093647649320'][['trip_uuid' ,'time_taken', 'osrm_dist
```

Out[98]:

	trip_uuid	time_taken	osrm_distance	segment_osrm_distance	is_cutoff
0	trip-153741093647649320	1.436894	11.9653	11.9653	True
1	trip-153741093647649320	1.436894	21.7243	9.7590	True
2	trip-153741093647649320	1.436894	32.5395	10.8152	True
3	trip-153741093647649320	1.436894	45.5620	13.0224	True
4	trip-153741093647649320	1.436894	54.2181	3.9153	False
5	trip-153741093647649320	1.819553	12.1171	12.1171	True
6	trip-153741093647649320	1.819553	21.2890	9.1719	True
7	trip-153741093647649320	1.819553	35.8252	14.5362	True
8	trip-153741093647649320	1.819553	47.1900	11.3648	True
9	trip-153741093647649320	1.819553	53.2334	6.0434	False

In [ ]:

```
1
```

In [99]:

```
1 osrm_distance_time = ((dv.groupby(["trip_uuid","time_taken"])["osrm_distance"].max()).r
2 osrm_distance_time
```

Out[99]:

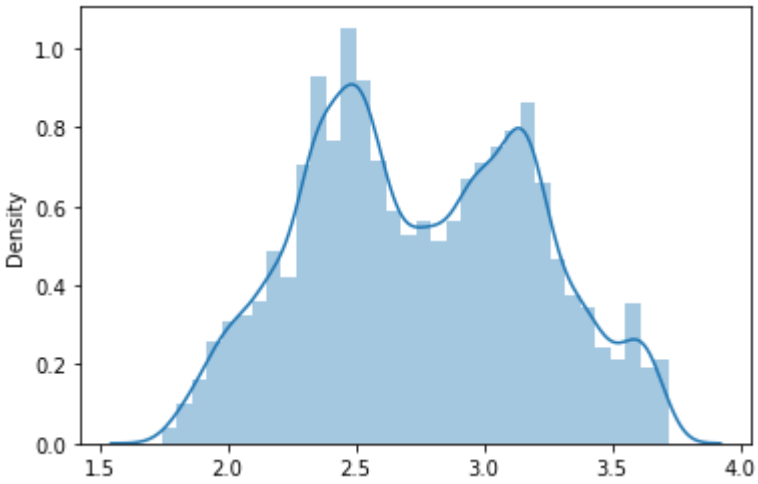
```
trip_uuid
trip-153671041653548748     991.3523
trip-153671042288605164      85.1110
trip-153671043369099517    2372.0852
trip-153671046011330457      19.6800
trip-153671052974046625     146.7918
                              ...
trip-153861095625827784      73.4630
trip-153861104386292051      16.0882
trip-153861106442901555      63.2841
trip-153861115439069069     177.6635
trip-153861118270144424      80.5787
Name: osrm_distance, Length: 14817, dtype: float64
```

In [100]:

```
1 sns.distplot(stats.boxcox(osrm_distance_time)[0])
```

Out[100]:

```
<AxesSubplot:ylabel='Density'>
```



In [ ]:

```
1
```

In [101]:

```
1 segment_osrm_distance_time = ((dv.groupby(["trip_uuid","time_taken"])["segment_osrm_dis
2 segment_osrm_distance_time
```

Out[101]:

```
trip_uuid
trip-153671041653548748    1320.4733
trip-153671042288605164      84.1894
trip-153671043369099517    2545.2678
trip-153671046011330457      19.8766
trip-153671052974046625     146.7919
                              ...
trip-153861095625827784      64.8551
trip-153861104386292051      16.0883
trip-153861106442901555     104.8866
trip-153861115439069069     223.5324
trip-153861118270144424      80.5787
Name: segment_osrm_distance, Length: 14817, dtype: float64
```

In [ ]:

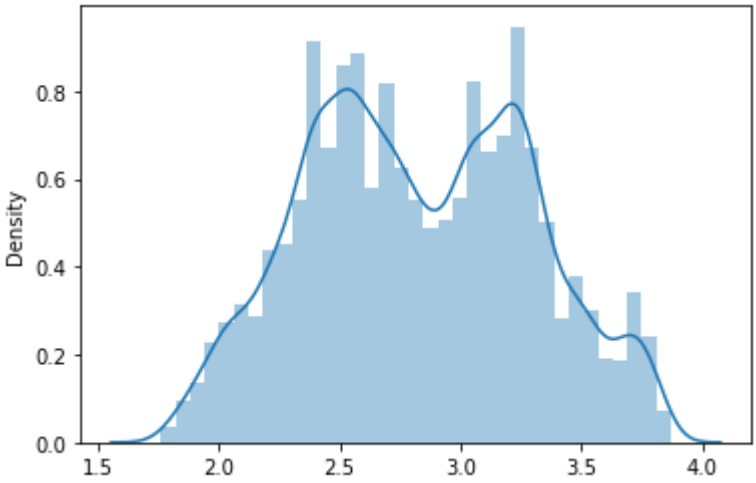
```
1
```

In [102]:

```
1 sns.distplot(stats.boxcox(segment_osrm_distance_time)[0])
```

Out[102]:

```
<AxesSubplot:ylabel='Density'>
```



In [ ]:

```
1
```

In [103]:

```
1 stats.ttest_ind(stats.boxcox(osrm_distance_time)[0],stats.boxcox(segment_osrm_distance_
```

Out[103]:

```
Ttest_indResult(statistic=-13.827029249141647, pvalue=2.388064035118428e-43)
```

In [ ]:

```
1
```

In [104]:

```
1 stats.ttest_ind(np.log(osrm_distance_time),np.log(segment_osrm_distance_time))
```

Out[104]:

```
Ttest_indResult(statistic=-4.1698517028537525, pvalue=3.056639395816076e-05)
```

In [ ]:

```
1
```

Observations

```
from two sample ttest , we can conclude that Population average for Actual Time ta
ken to complete delivery trip and
segment osrm distance are not same.
```

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)

In [105]:

```
dv[dv['trip_uuid'] == 'trip-153741093647649320'][['trip_uuid' ,'time_taken', 'osrm_time
```

Out[105]:

	trip_uuid	time_taken	osrm_time	segment_osrm_time	is_cutoff
0	trip-153741093647649320	1.436894	11.0	11.0	True
1	trip-153741093647649320	1.436894	20.0	9.0	True
2	trip-153741093647649320	1.436894	28.0	7.0	True
3	trip-153741093647649320	1.436894	40.0	12.0	True
4	trip-153741093647649320	1.436894	44.0	5.0	False
5	trip-153741093647649320	1.819553	11.0	11.0	True
6	trip-153741093647649320	1.819553	17.0	6.0	True
7	trip-153741093647649320	1.819553	29.0	11.0	True
8	trip-153741093647649320	1.819553	39.0	10.0	True
9	trip-153741093647649320	1.819553	45.0	6.0	False

In [106]:

```
print('osrm_time - An open-source routing engine time calculator which computes the sho
dv.osrm_time
```

```
osrm_time - An open-source routing engine time calculator which computes the
shortest path between points in a given map (Includes usual traffic, distanc
e through major and minor roads) and gives the time (Cumulative)
```

Out[106]:

```
0         11.0
1         20.0
2         28.0
3         40.0
4         44.0
          ...
144862    60.0
144863    76.0
144864    88.0
144865    98.0
144866    95.0
Name: osrm_time, Length: 144867, dtype: float64
```

In [107]:

```
osrm_time = ((dv.groupby(["trip_uuid","time_taken"])["osrm_time"].max()).reset_index())
osrm_time
```

Out[107]:

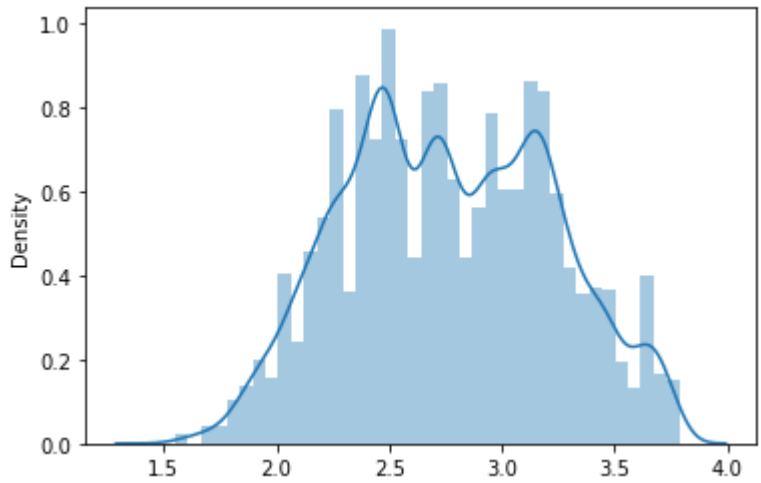
```
trip_uuid
trip-153671041653548748     743.0
trip-153671042288605164      68.0
trip-153671043369099517    1741.0
trip-153671046011330457      15.0
trip-153671052974046625     117.0
                           ...
trip-153861095625827784      62.0
trip-153861104386292051      12.0
trip-153861106442901555      54.0
trip-153861115439069069     184.0
trip-153861118270144424      68.0
Name: osrm_time, Length: 14817, dtype: float64
```

In [108]:

```
sns.distplot(stats.boxcox(osrm_time)[0])
```

Out[108]:

```
<AxesSubplot:ylabel='Density'>
```



In [ ]:

```

```

In [109]:

```
print('segment_osrm_time - This is the OSRM segment time. Time taken by the subset of t
dv.segment_osrm_time
```

```
segment_osrm_time - This is the OSRM segment time. Time taken by the subset
of the package delivery
```

Out[109]:

```
0         11.0
1          9.0
2          7.0
3         12.0
4          5.0
          ...
144862    12.0
144863    21.0
144864    34.0
144865    27.0
144866     9.0
Name: segment_osrm_time, Length: 144867, dtype: float64
```

In [110]:

```
segment_osrm_time = ((dv.groupby(["trip_uuid","time_taken"])["segment_osrm_time"].sum()
segment_osrm_time
```

Out[110]:

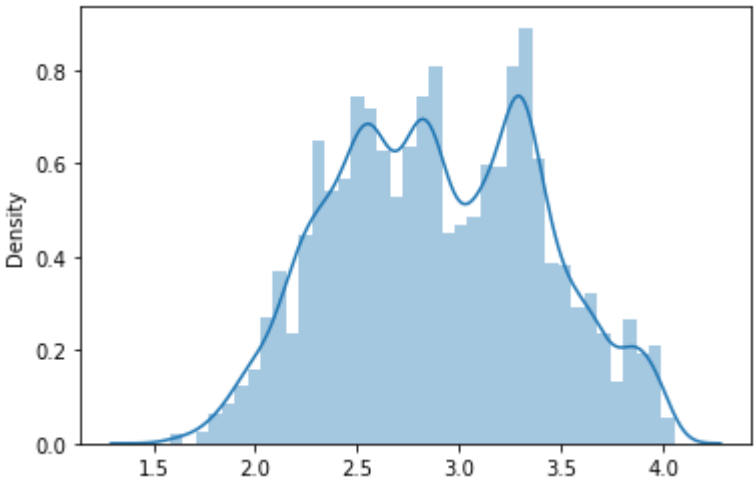
```
trip_uuid
trip-153671041653548748    1008.0
trip-153671042288605164      65.0
trip-153671043369099517    1941.0
trip-153671046011330457      16.0
trip-153671052974046625     115.0
                            ...
trip-153861095625827784      62.0
trip-153861104386292051      11.0
trip-153861106442901555      88.0
trip-153861115439069069     221.0
trip-153861118270144424      67.0
Name: segment_osrm_time, Length: 14817, dtype: float64
```

In [111]:

```
1 sns.distplot(stats.boxcox(segment_osrm_time)[0])
```

Out[111]:

```
<AxesSubplot:ylabel='Density'>
```



In [ ]:

```
1
```

In [112]:

```
1 stats.ttest_ind(stats.boxcox(osrm_time)[0],stats.boxcox(segment_osrm_time)[0])
```

Out[112]:

```
Ttest_indResult(statistic=-23.42127912444594, pvalue=3.2149793519334874e-12
0)
```

Observation

from two sample ttest , we can conclude that Population average for OSRM Time taken to complete delivery trip and segment osrm time are not same.

Find outliers in the numerical variables (you might find outliers in almost all the variables), and check it using visual analysis

In [113]:

```
1 dv.describe()
```

Out[113]:

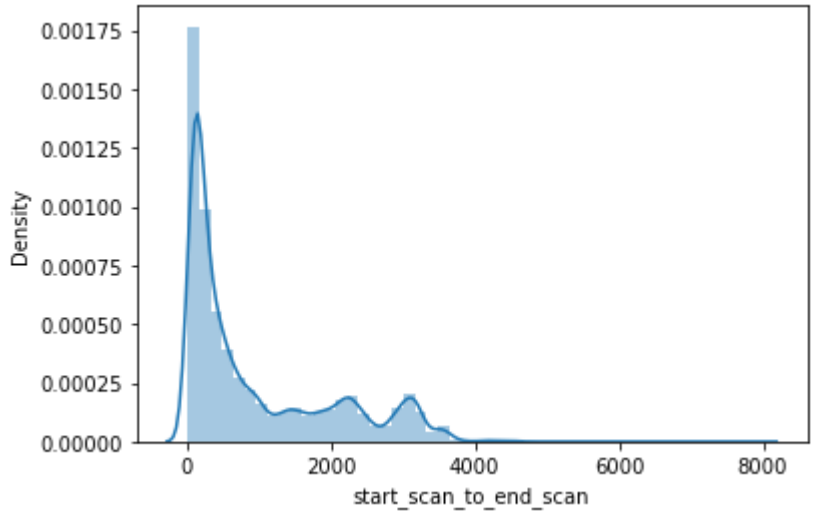
	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	14
count	144867.000000	144867.000000	144867.000000	144867.000000	
mean	961.262986	232.926567	234.073372	416.927527	
std	1037.012769	344.755577	344.990009	598.103621	
min	20.000000	9.000000	9.000045	9.000000	
25%	161.000000	22.000000	23.355874	51.000000	
50%	449.000000	66.000000	66.126571	132.000000	
75%	1634.000000	286.000000	286.708875	513.000000	
max	7898.000000	1927.000000	1927.447705	4532.000000	

In [114]:

```
1 sns.distplot(dv['start_scan_to_end_scan'])
```

Out[114]:

```
<AxesSubplot:xlabel='start_scan_to_end_scan', ylabel='Density'>
```



In [ ]:

```
1
```

In [ ]:

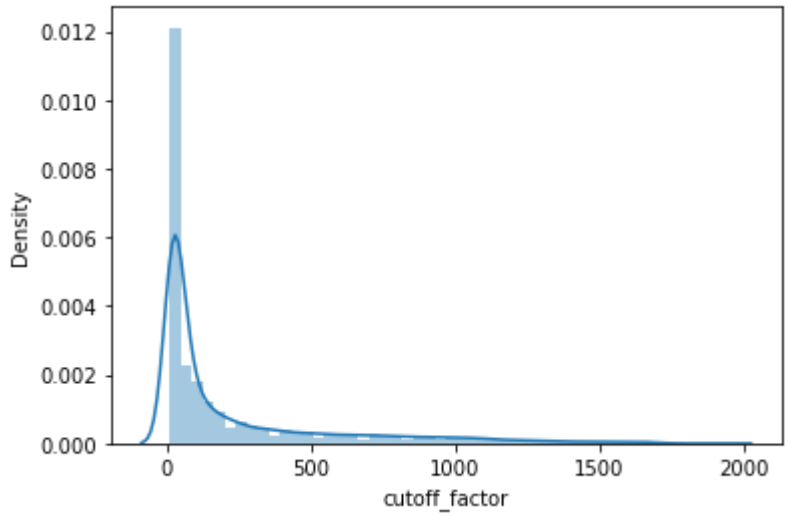
```
1
```

In [115]:

```
1 sns.distplot(dv['cutoff_factor'])
```

Out[115]:

<AxesSubplot:xlabel='cutoff_factor', ylabel='Density'>



In [ ]:

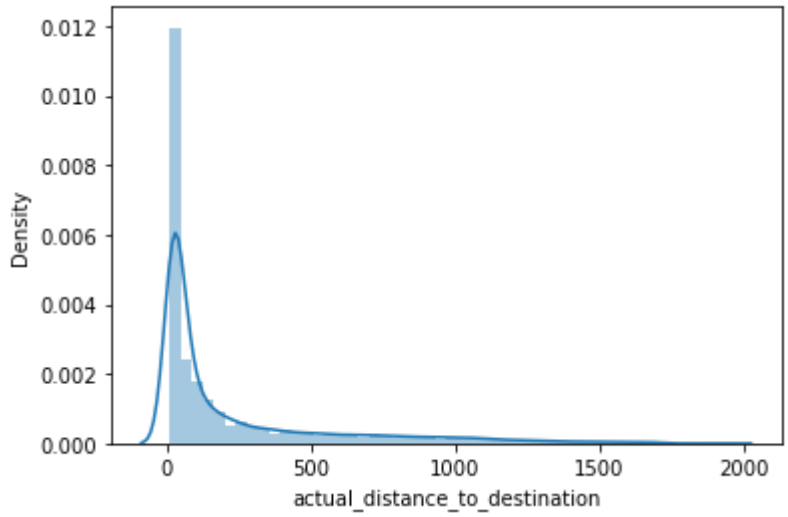
```
1
```

In [116]:

```
1 sns.distplot(dv['actual_distance_to_destination'])
```

Out[116]:

<AxesSubplot:xlabel='actual_distance_to_destination', ylabel='Density'>



In [ ]:

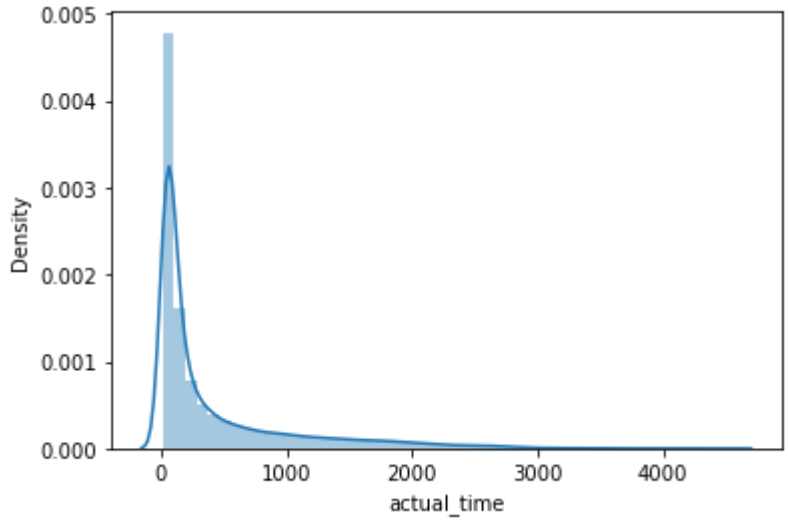
```
1
```

In [117]:

```
1 sns.distplot(dv['actual_time'])
```

Out[117]:

```
<AxesSubplot:xlabel='actual_time', ylabel='Density'>
```

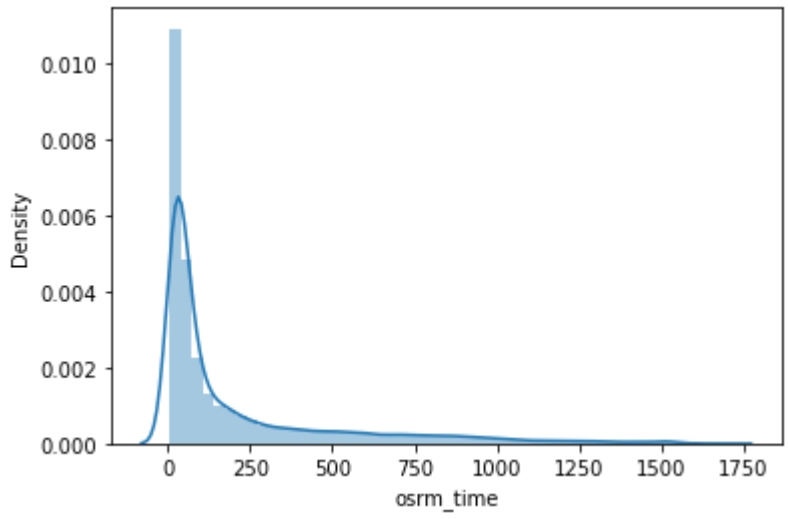


In [119]:

```
1 sns.distplot(dv['osrm_time'])
```

Out[119]:

```
<AxesSubplot:xlabel='osrm_time', ylabel='Density'>
```



In [ ]:

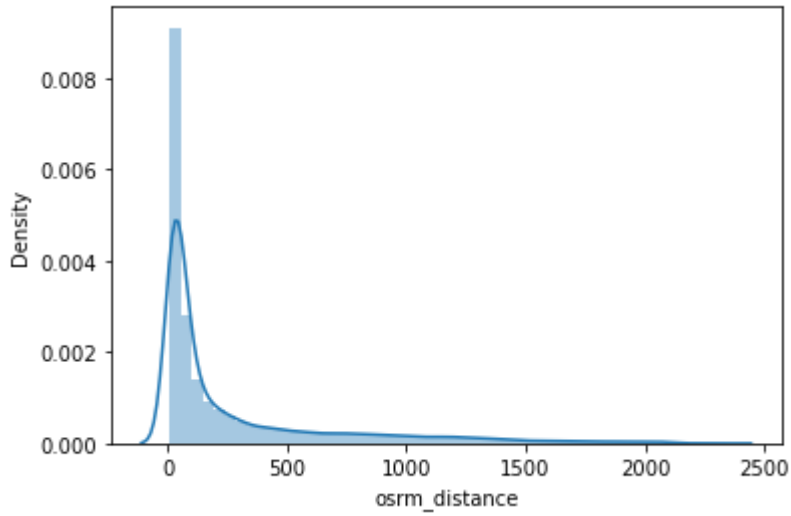
```
1
```

In [120]:

```
1 sns.distplot(dv['osrm_distance'])
```

Out[120]:

<AxesSubplot:xlabel='osrm_distance', ylabel='Density'>



In [ ]:

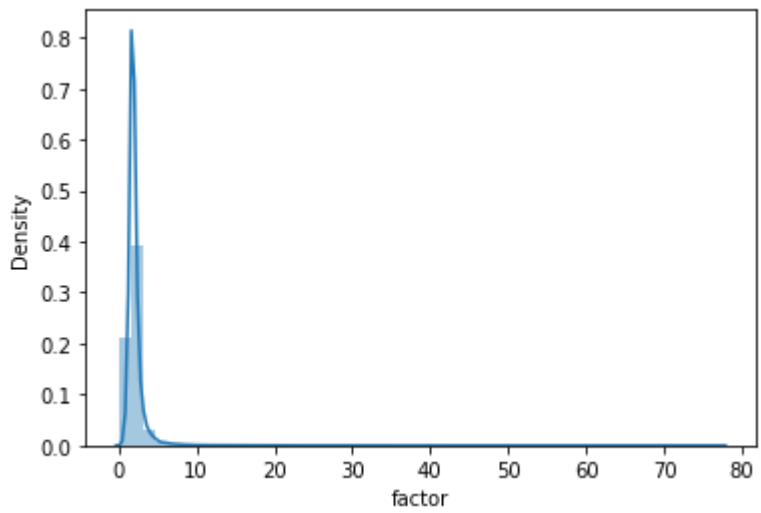
```
1
```

In [121]:

```
1 sns.distplot(dv['factor'])
```

Out[121]:

<AxesSubplot:xlabel='factor', ylabel='Density'>



In [ ]:

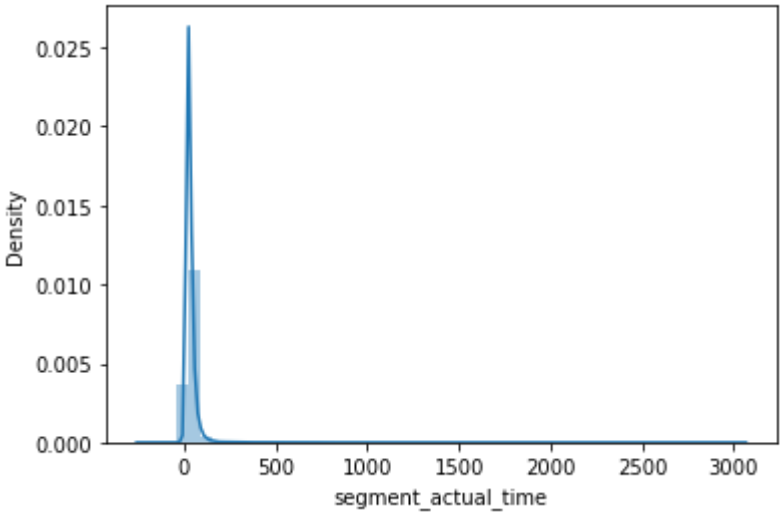
```
1
```

In [122]:

```
1 sns.distplot(dv['segment_actual_time'])
```

Out[122]:

```
<AxesSubplot:xlabel='segment_actual_time', ylabel='Density'>
```



In [ ]:

```
1
```

In [ ]:

```
1
```

In [ ]:

```
1
```

extracting process to get source city and destination city

In [123]:

```
1 dv.head()
```

Out[123]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_c
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812

- orders orders from which state ?
- we need uniq id col and uniqe name cols to start with

In [124]:

```
1 dv.route_schedule_uuid
```

Out[124]:

```
0         thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...
1         thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...
2         thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...
3         thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...
4         thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...
                              ...
144862    thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...
144863    thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...
144864    thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...
144865    thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...
144866    thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...
Name: route_schedule_uuid, Length: 144867, dtype: object
```

In [125]:

```
1 dv.source_name
```

Out[125]:

```
0         Anand_VUNagar_DC (Gujarat)
1         Anand_VUNagar_DC (Gujarat)
2         Anand_VUNagar_DC (Gujarat)
3         Anand_VUNagar_DC (Gujarat)
4         Anand_VUNagar_DC (Gujarat)
                    ...
144862    Sonipat_Kundli_H (Haryana)
144863    Sonipat_Kundli_H (Haryana)
144864    Sonipat_Kundli_H (Haryana)
144865    Sonipat_Kundli_H (Haryana)
144866    Sonipat_Kundli_H (Haryana)
Name: source_name, Length: 144867, dtype: object
```

In [ ]:

```
1
```

In [126]:

```
1 dv["source_city"] = dv["source_name"].str.extract(r'([^_]+)')
2 dv.source_city
```

Out[126]:

```
0           Anand
1           Anand
2           Anand
3           Anand
4           Anand
           ...
144862    Sonipat
144863    Sonipat
144864    Sonipat
144865    Sonipat
144866    Sonipat
Name: source_city, Length: 144867, dtype: object
```

In [127]:

```
dv["source_state"] = dv["source_name"].str.extract(r'.*\((.*)\).*')
dv["source_state"]
```

Out[127]:

```
0         Gujarat
1         Gujarat
2         Gujarat
3         Gujarat
4         Gujarat
           ...
144862    Haryana
144863    Haryana
144864    Haryana
144865    Haryana
144866    Haryana
Name: source_state, Length: 144867, dtype: object
```

In [128]:

```
dv["source_code"] = (dv["source_name"].str.split("_",n=2,expand = True)[2]).str.split('
dv["source_code"]
```

Out[128]:

```
0         DC
1         DC
2         DC
3         DC
4         DC
          ..
144862     H
144863     H
144864     H
144865     H
144866     H
Name: source_code, Length: 144867, dtype: object
```

In [129]:

```
dv["source_place"] = dv["source_name"].str.split("_",n=2,expand = True)[1]
dv["source_place"]
```

Out[129]:

```
0         VUNagar
1         VUNagar
2         VUNagar
3         VUNagar
4         VUNagar
           ...
144862     Kundli
144863     Kundli
144864     Kundli
144865     Kundli
144866     Kundli
Name: source_place, Length: 144867, dtype: object
```

In [ ]:

```
1
```

In [130]:

```
1 dv["destination_city"] = dv["destination_name"].str.extract(r'([^_]+)')
2
```

In [131]:

```
1 dv["destination_state"] = dv["destination_name"].str.extract(r'.*\((.*)\).*')
```

In [132]:

```
1 dv["destination_code"] = (dv["destination_name"].str.split("_",n=2,expand = True)[2]).s
```

In [133]:

```
1 dv["destination_place"] = dv["destination_name"].str.split("_",n=2,expand = True)[1]
```

In [134]:

```
1 dv["source_city"].replace({"Bangalore":"Bengaluru"},inplace=True)
2
3
4 dv["destination_city"].replace({"Bangalore":"Bengaluru"},inplace=True)
```

Check from where most orders are coming from (State, Corridor etc)

In [290]:

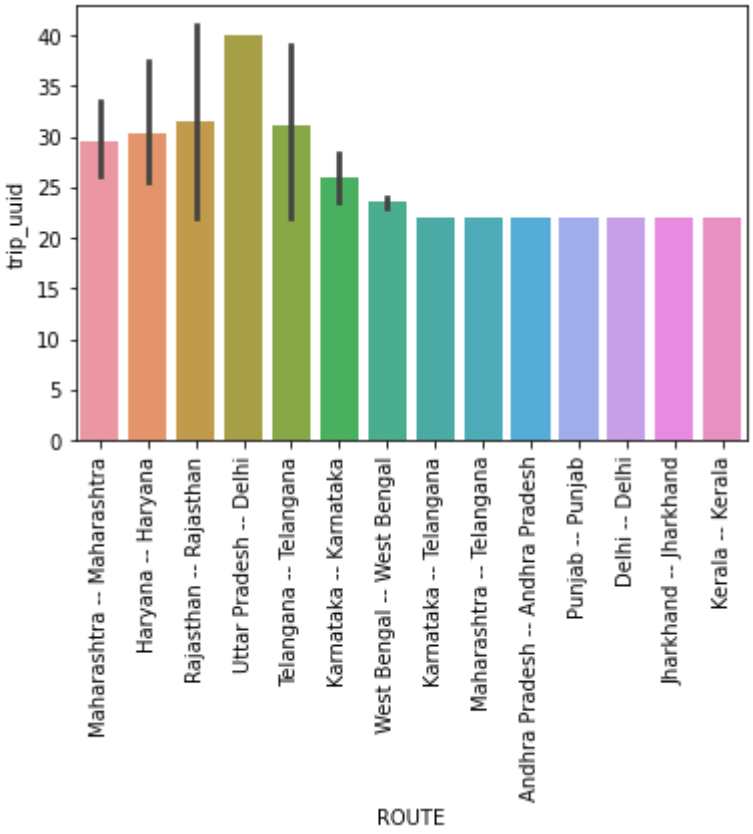
```
corr_data = dv.groupby(['source_state','route_schedule_uuid', 'destination_state' ])['t
corr_data["ROUTE"] = corr_data["source_state"] + " -- " + corr_data["destination_state"
corr_data.drop(['source_state', 'destination_state' ,'route_schedule_uuid'] ,axis = 1,
X = corr_data.head(50)

X.head(10)

```

Out[290]:

	trip_uuid	ROUTE
0	53	Maharashtra -- Maharashtra
1	46	Maharashtra -- Maharashtra
2	43	Haryana -- Haryana
3	41	Rajasthan -- Rajasthan
4	40	Uttar Pradesh -- Delhi
5	39	Telangana -- Telangana
6	37	Maharashtra -- Maharashtra
7	36	Maharashtra -- Maharashtra
8	35	Maharashtra -- Maharashtra
9	34	Maharashtra -- Maharashtra

In [291]:

```
sns.barplot(x = X["ROUTE"],
            y= X["trip_uuid"])
plt.xticks(rotation = 90)
plt.show()
```



In [299]:

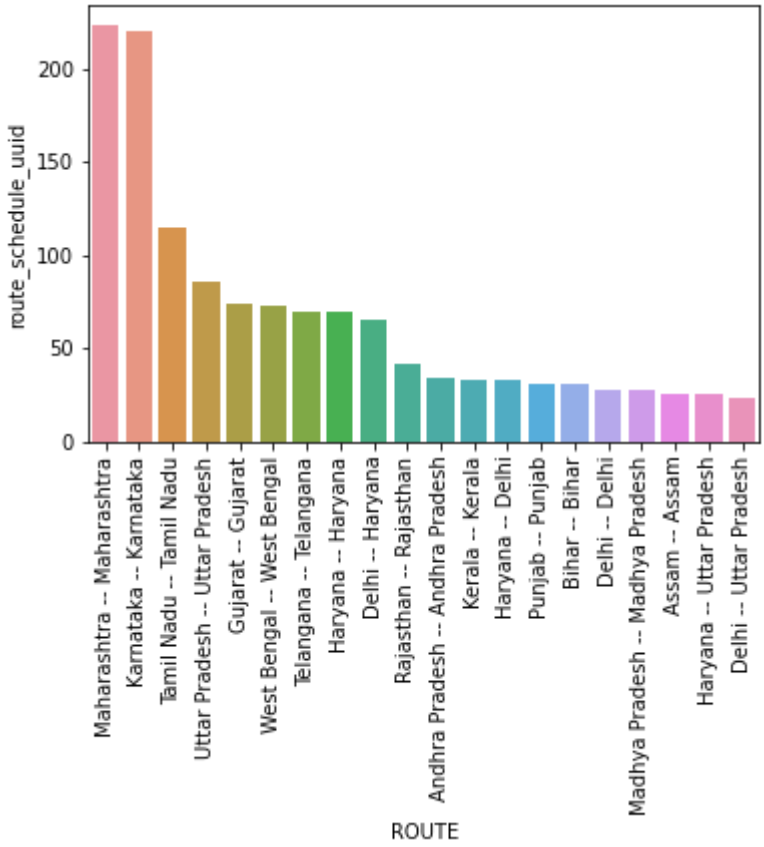
```
Y = dv.groupby(['source_state','destination_state' ])['route_schedule_uuid'].nunique().
Y["ROUTE"] = Y["source_state"] + " -- " + Y["destination_state"]
Y.drop(['source_state', 'destination_state'] ,axis = 1, inplace =True)
Y
```

Out[299]:

	route_schedule_uuid	ROUTE
0	223	Maharashtra -- Maharashtra
1	220	Karnataka -- Karnataka
2	115	Tamil Nadu -- Tamil Nadu
3	86	Uttar Pradesh -- Uttar Pradesh
4	74	Gujarat -- Gujarat
5	73	West Bengal -- West Bengal
6	69	Telangana -- Telangana
7	69	Haryana -- Haryana
8	65	Delhi -- Haryana
9	41	Rajasthan -- Rajasthan
10	34	Andhra Pradesh -- Andhra Pradesh
11	33	Kerala -- Kerala
12	33	Haryana -- Delhi
13	31	Punjab -- Punjab
14	31	Bihar -- Bihar
15	27	Delhi -- Delhi
16	27	Madhya Pradesh -- Madhya Pradesh
17	25	Assam -- Assam
18	25	Haryana -- Uttar Pradesh
19	23	Delhi -- Uttar Pradesh

In [300]:

```
sns.barplot(x = Y["ROUTE"],
            y= Y["route_schedule_uuid"])
plt.xticks(rotation = 90)
plt.show()
```



In [287]:

```
1 dv.groupby(['source_state','source_city','destination_city', 'destination_state' ])['rc
```

Out[287]:

```
source_state   source_city                        destination_city
destination_state
Karnataka      Bengaluru                          Bengaluru
Karnataka            142
Delhi          Delhi                              Gurgaon
Haryana              38
Maharashtra    Mumbai                             Bhiwandi
Maharashtra          38
               Bhiwandi                           Mumbai
Maharashtra          37
Telangana      Hyderabad                          Hyderabad
Telangana            33
Tamil Nadu     Chennai                            Chennai
Tamil Nadu           32
Haryana        Gurgaon                            Delhi
Delhi                25
Maharashtra    Mumbai                             Mumbai
Maharashtra          23
Tamil Nadu     Chennai                            MAA
Tamil Nadu           21
               MAA                                Chennai
Tamil Nadu           20
Delhi          Delhi                              Delhi
Delhi                17
Gujarat        Surat                              Surat
Gujarat              14
Maharashtra    Pune                               Bhiwandi
Maharashtra          13
West Bengal    Kolkata                            CCU
West Bengal          13
Karnataka      Bengaluru                          BLR
Karnataka            13
Delhi          Del                                Gurgaon
Haryana              13
Maharashtra    Pune                               Pune
Maharashtra          12
Rajasthan      Jaipur                             Jaipur
Rajasthan            11
Maharashtra    Mumbai                             Mumbai Hub (Maharashtra)
Maharashtra          11
Karnataka      Bengaluru                          HBR Layout PC (Karnatak
a)            Karnataka             11
Maharashtra    Mumbai Hub (Maharashtra)           Mumbai
Maharashtra          11
Chandigarh     Chandigarh                         Chandigarh
Punjab                8
West Bengal    Kolkata                            Kolkata
West Bengal           8
Assam          Guwahati                           Guwahati
Assam                 7
Maharashtra    PNQ Vadgaon Sheri DPC (Maharashtra)  Pune
Maharashtra           7
Delhi          Delhi                              Sonipat
Haryana               7
```

```
Haryana        Gurgaon                          Noida
Uttar Pradesh          6
                                                Sonipat
Haryana                6
Maharashtra    Bhiwandi                         Pune
Maharashtra            6
Punjab         Ludhiana                         Chandigarh
Punjab                 6
Haryana        Gurgaon                          Chandigarh
Punjab                 6
Gujarat        Ahmedabad                        Surat
Gujarat                6
Haryana        Sonipat                          Gurgaon
Haryana                6
Uttar Pradesh  Allahabad                        Allahabad
Uttar Pradesh          6
Karnataka      Bengaluru                        Gurgaon
Haryana                6
Uttar Pradesh  Kanpur                           Gurgaon
Haryana                6
Maharashtra    Pune                             PNQ Vadgaon Sheri DPC (M
aharashtra)  Maharashtra            6
West Bengal    CCU                              Kolkata
West Bengal            6
Haryana        Sonipat                          Roorkee
Uttarakhand            5
Delhi          Delhi                            Kanpur
Uttar Pradesh          5
               Del                              Delhi
Delhi                  5
Haryana        GGN                              Gurgaon
Haryana                5
Karnataka      HBR Layout PC (Karnataka)        Bengaluru
Karnataka              5
Maharashtra    Bhiwandi                         Bengaluru
Karnataka              5
Telangana      Hyderabad                        Pune
Maharashtra            5
Uttarakhand    Haridwar (Uttarakhand)           Rishikesh
Uttarakhand            5
Haryana        Gurgaon                          Kanpur
Uttar Pradesh          5
Punjab         Chandigarh                       Chandigarh
Chandigarh             5
Delhi          Delhi                            Noida
Uttar Pradesh          5
Haryana        Gurgaon                          Gurgaon
Haryana                5
Name: route_schedule_uuid, dtype: int64
```

In [ ]:

```
1
```

Busiest corridor, avg distance between them, avg time taken

In [141]:

```
1 dv.actual_distance_to_destination
```

Out[141]:

```
0         10.435660
1         18.936842
2         27.637279
3         36.118028
4         39.386040
            ...
144862    45.258278
144863    54.092531
144864    66.163591
144865    73.680667
144866    70.039010
Name: actual_distance_to_destination, Length: 144867, dtype: float64
```

In [142]:

```
1 dv[dv['trip_uuid']=='trip-153671043369099517'][['trip_uuid' ,'time_taken', 'osrm_time'
```

Out[142]:

	trip_uuid	time_taken	osrm_time	actual_distance_to_destination	is_cutoff	s
131339	trip-153671043369099517	51.662060	27.0	26.908354	True	
131340	trip-153671043369099517	51.662060	42.0	45.637104	True	
131341	trip-153671043369099517	51.662060	66.0	70.279536	True	
131342	trip-153671043369099517	51.662060	81.0	90.572441	True	
131343	trip-153671043369099517	51.662060	97.0	112.664478	True	
...	...	...	...	...	...	
131423	trip-153671043369099517	13.910649	155.0	182.128883	True	
131424	trip-153671043369099517	13.910649	166.0	199.340785	True	
131425	trip-153671043369099517	13.910649	182.0	221.523161	True	
131426	trip-153671043369099517	13.910649	198.0	242.309306	True	
131427	trip-153671043369099517	13.910649	212.0	237.439610	False	

89 rows × 9 columns

In [143]:

```
1 dv[dv['is_cutoff']==False][['trip_uuid' ,'time_taken', 'osrm_time' , 'actual_distance_t
```

Out[143]:

trip_uuid	time_taken	osrm_time	actual_distance_to_destination	is_cutoff
trip-153671041653548748	37.668497	717.0	824.732854	0
trip-153671042288605164	3.026865	68.0	73.186911	0
trip-153671043369099517	65.572709	1740.0	1927.404273	0
trip-153671046011330457	1.674916	15.0	17.175274	0
trip-153671052974046625	11.972484	117.0	127.448500	0
...	...	...	...	...
trip-153861095625827784	4.300482	62.0	57.762332	0
trip-153861104386292051	1.009842	12.0	15.513784	0
trip-153861106442901555	7.035331	48.0	38.684839	0
trip-153861115439069069	5.808548	179.0	134.723836	0
trip-153861118270144424	5.906793	68.0	66.081533	0

14703 rows × 4 columns

In [144]:

```
a = dv.groupby("trip_uuid")[["source_state","destination_state"]].aggregate({"source_st
a
```

Out[144]:

trip_uuid	source_state	destination_state
trip-153671041653548748	{Madhya Pradesh, Uttar Pradesh}	{Uttar Pradesh, Haryana}
trip-153671042288605164	{Karnataka}	{Karnataka}
trip-153671043369099517	{Karnataka, Haryana}	{Punjab, Haryana}
trip-153671046011330457	{Maharashtra}	{Maharashtra}
trip-153671052974046625	{Karnataka}	{Karnataka}
...	...	...
trip-153861095625827784	{Chandigarh, Punjab}	{Punjab}
trip-153861104386292051	{Haryana}	{Haryana}
trip-153861106442901555	{Uttar Pradesh}	{Uttar Pradesh}
trip-153861115439069069	{Tamil Nadu}	{Tamil Nadu}
trip-153861118270144424	{Karnataka}	{Karnataka}

14817 rows × 2 columns

Observation

- The above table shows the source state and destination state for a single trip
- As you can see, for a single trip_uuid there are multiple check points.

In [145]:

```
b = ((dv.groupby(["trip_uuid","is_cutoff"])["actual_distance_to_destination"].max()).re
b
```

Out[145]:

```
trip_uuid
trip-153671041653548748     881.003772
trip-153671042288605164      93.725419
trip-153671043369099517    3362.306060
trip-153671046011330457      28.529648
trip-153671052974046625     103.938814
                               ...
trip-153861095625827784      58.839137
trip-153861104386292051      25.130640
trip-153861106442901555      37.799141
trip-153861115439069069      74.104604
trip-153861118270144424      62.703556
Name: actual_distance_to_destination, Length: 14817, dtype: float64
```

In [197]:

```
c = a.merge(b ,on = 'trip_uuid')
df = pd.DataFrame(c)
df

```

Out[197]:

trip_uuid	source_state	destination_state	actual_distance_to_destination
trip-153671041653548748	{Madhya Pradesh, Uttar Pradesh}	{Uttar Pradesh, Haryana}	881.003772
trip-153671042288605164	{Karnataka}	{Karnataka}	93.725419
trip-153671043369099517	{Karnataka, Haryana}	{Punjab, Haryana}	3362.306060
trip-153671046011330457	{Maharashtra}	{Maharashtra}	28.529648
trip-153671052974046625	{Karnataka}	{Karnataka}	103.938814
...	...	...	...
trip-153861095625827784	{Chandigarh, Punjab}	{Punjab}	58.839137
trip-153861104386292051	{Haryana}	{Haryana}	25.130640
trip-153861106442901555	{Uttar Pradesh}	{Uttar Pradesh}	37.799141
trip-153861115439069069	{Tamil Nadu}	{Tamil Nadu}	74.104604
trip-153861118270144424	{Karnataka}	{Karnataka}	62.703556

14817 rows × 3 columns

Observations

- we can clearly see that when ever the trip is long wrt distance the number of check points are also more and vice versa

In [198]:

```
# df[['destination', 'dest1']] = df['destination_state'].str.split(',', 1, expand=True)
# # df.drop('dest1', axis=1 , inplace = True)
# # df
```

In [ ]:

```
# df[['source', 'B']] = df['destination_state'].str.split(',', 1, expand=True)
# df
```

In [159]:

```
df['source_state'] = df['source_state'].astype(str)
df['destination_state'] = df['destination_state'].astype(str)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 14817 entries, trip-153671041653548748 to trip-153861118270144424
Data columns (total 3 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   source_state                    14817 non-null  object 
 1   destination_state               14817 non-null  object 
 2   actual_distance_to_destination  14817 non-null  float64
dtypes: float64(1), object(2)
memory usage: 463.0+ KB
```

Total Orders

In [171]:

```
r = dv.groupby('trip_uuid')['route_schedule_uuid'].nunique().sort_values(ascending = Fa
r
```

Out[171]:

	trip_uuid	route_schedule_uuid
0	trip-153671041653548748	1
1	trip-153791331656620454	1
2	trip-153791340002649773	1
3	trip-153791341220571147	1
4	trip-153791345208672550	1
...	...	...
14812	trip-153730385530436915	1
14813	trip-153730387256906443	1
14814	trip-153730391233462064	1
14815	trip-153730396484038671	1
14816	trip-153861118270144424	1

14817 rows × 2 columns

In [172]:

```
s = dv.groupby('route_schedule_uuid')['trip_uuid'].count().sort_values(ascending = Fals
s
```

Out[172]:

	route_schedule_uuid	trip_uuid
0	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	1812
1	thanos::sroute:0456b740-1dad-4929-bbe0-87d8843...	1608
2	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	1605
3	thanos::sroute:a1b25549-1e77-498f-8538-00292e5...	1285
4	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	1280
...	...	...
1499	thanos::sroute:b45dbffe-dd2c-4edb-a9e6-e35065d...	1
1500	thanos::sroute:d29fd731-9f1f-490c-922e-6d79d16...	1
1501	thanos::sroute:889b9cf5-da6a-48ce-b3bd-6983c80...	1
1502	thanos::sroute:404cbabf-d2a5-4e46-bf79-8b3c518...	1
1503	thanos::sroute:036f372d-28d8-4d19-877c-6277077...	1

1504 rows × 2 columns

In [173]:

```
dv[dv['trip_uuid'] == 'trip-153784927255069118']['route_schedule_uuid'].count()
```

Out[173]:

```
101
```

In [174]:

```
source_package = dv.groupby('source_state')['route_schedule_uuid'].nunique().sort_value
source_package.rename(columns = {'source_state':'states', 'route_schedule_uuid':'no_of_
source_package
```

Out[174]:

	states	no_of_trips_from_each_state
0	Maharashtra	253
1	Karnataka	239
2	Haryana	178
3	Tamil Nadu	123
4	Delhi	120
5	Uttar Pradesh	109
6	Telangana	88
7	Gujarat	85
8	West Bengal	80
9	Rajasthan	51
10	Andhra Pradesh	50
11	Punjab	48
12	Madhya Pradesh	39
13	Kerala	35
14	Bihar	33
15	Assam	31
16	Orissa	23
17	Uttarakhand	19
18	Jharkhand	17
19	Chandigarh	9
20	Goa	9
21	Himachal Pradesh	9
22	Arunachal Pradesh	7
23	Chhattisgarh	5
24	Meghalaya	2
25	Mizoram	2
26	Jammu & Kashmir	2
27	Pondicherry	2
28	Dadra and Nagar Haveli	1
29	Nagaland	1
30	Tripura	1

In [175]:

```
destination_package = dv.groupby('destination_state')['route_schedule_uuid'].nunique().
destination_package.rename(columns = {'destination_state':'states', 'route_schedule_uui
destination_package
```

Out[175]:

	states	no_of_trips_to_each_state
0	Karnataka	250
1	Maharashtra	239
2	Haryana	198
3	Uttar Pradesh	125
4	Tamil Nadu	121
5	Telangana	87
6	Gujarat	86
7	West Bengal	83
8	Delhi	79
9	Punjab	53
10	Rajasthan	52
11	Andhra Pradesh	51
12	Madhya Pradesh	41
13	Kerala	36
14	Bihar	33
15	Assam	32
16	Orissa	24
17	Uttarakhand	19
18	Jharkhand	18
19	Goa	11
20	Himachal Pradesh	9
21	Chandigarh	6
22	Arunachal Pradesh	5
23	Chhattisgarh	5
24	Meghalaya	2
25	Mizoram	2
26	Jammu & Kashmir	2
27	Pondicherry	2
28	Dadra and Nagar Haveli	2
29	Daman & Diu	1
30	Nagaland	1
31	Tripura	1

In [176]:

```
sd = source_package.merge(destination_package , on = 'states')
sd['Busiest_state_by_trips'] = sd['no_of_trips_from_each_state'] + sd['no_of_trips_to_e
sd
```

Out[176]:

	states	no_of_trips_from_each_state	no_of_trips_to_each_state	Busiest_state_by_tri
0	Maharashtra	253	239	4
1	Karnataka	239	250	4
2	Haryana	178	198	3
3	Tamil Nadu	123	121	2
4	Delhi	120	79	1
5	Uttar Pradesh	109	125	2
6	Telangana	88	87	1
7	Gujarat	85	86	1
8	West Bengal	80	83	1
9	Rajasthan	51	52	1
10	Andhra Pradesh	50	51	1
11	Punjab	48	53	1
12	Madhya Pradesh	39	41	
13	Kerala	35	36	
14	Bihar	33	33	
15	Assam	31	32	
16	Orissa	23	24	
17	Uttarakhand	19	19	
18	Jharkhand	17	18	
19	Chandigarh	9	6	
20	Goa	9	11	
21	Himachal Pradesh	9	9	
22	Arunachal Pradesh	7	5	
23	Chhattisgarh	5	5	
24	Meghalaya	2	2	
25	Mizoram	2	2	
26	Jammu & Kashmir	2	2	
27	Pondicherry	2	2	
28	Dadra and Nagar Haveli	1	2	

	states	no_of_trips_from_each_state	no_of_trips_to_each_state	Busiest_state_by_tri
29	Nagaland	1	1	
30	Tripura	1	1	

Observations

- The busiest state wrt to trips are in the above order .
- we can infer that the busiest states are mostly metro states as well as industrial states.

In [177]:

```
1 dv.groupby('source_state')['time_taken' , 'trip_uuid'].max()
```

Out[177]:

source_state	time_taken	trip_uuid
Andhra Pradesh	44.109873	trip-153861028301961630
Arunachal Pradesh	20.027831	trip-153853799309447487
Assam	68.016825	trip-153855027033921568
Bihar	42.994144	trip-153861007249500192
Chandigarh	14.857787	trip-153861095625827784
Chhattisgarh	15.047255	trip-153859592654771797
Dadra and Nagar Haveli	1.805108	trip-153857331775400080
Delhi	61.716212	trip-153860849934816308
Goa	31.541874	trip-153835178030348861
Gujarat	35.781550	trip-153861089403973335
Haryana	75.595254	trip-153861104386292051
Himachal Pradesh	41.995165	trip-153860492150952876
Jammu & Kashmir	13.085807	trip-153855382748399399
Jharkhand	43.329073	trip-153861004148234782
Karnataka	61.521077	trip-153861118270144424
Kerala	22.530864	trip-153860848028848826
Madhya Pradesh	45.857752	trip-153861014185597051
Maharashtra	54.351372	trip-153861091843037040
Meghalaya	28.199533	trip-153786220794548811
Mizoram	55.448414	trip-153752546021401806
Nagaland	21.765250	trip-153833364602473905
Orissa	131.642533	trip-153859882058199771
Pondicherry	5.011213	trip-153801333300954216
Punjab	63.368436	trip-153861095625827784
Rajasthan	17.207654	trip-153860998196116365
Tamil Nadu	38.118282	trip-153861115439069069
Telangana	45.746066	trip-153860996602682925
Tripura	6.007843	trip-153724448912072562
Uttar Pradesh	42.659331	trip-153861106442901555
Uttarakhand	23.091634	trip-153859603936924192
West Bengal	70.120402	trip-153860891043685648

Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler.

In [184]:

```
1 dv
```

Out[184]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	so
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND
...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND

144867 rows × 33 columns

In [185]:

```
num_cols_max = ((dv.groupby(["trip_uuid","time_taken" ])[["actual_time" , 'osrm_time',
minn = num_cols_max.reset_index()
minn
```

Out[185]:

	trip_uuid	actual_time	osrm_time	osrm_distance	actual_distance_to_destinati
0	trip-153671041653548748	1562.0	743.0	991.3523	824.7328
1	trip-153671042288605164	143.0	68.0	85.1110	73.1869
2	trip-153671043369099517	3347.0	1741.0	2372.0852	1932.2739
3	trip-153671046011330457	59.0	15.0	19.6800	17.1752
4	trip-153671052974046625	341.0	117.0	146.7918	127.4485
...	...	...	...	...	
14812	trip-153861095625827784	83.0	62.0	73.4630	57.7623
14813	trip-153861104386292051	21.0	12.0	16.0882	15.5137
14814	trip-153861106442901555	282.0	54.0	63.2841	38.6848
14815	trip-153861115439069069	264.0	184.0	177.6635	134.7238
14816	trip-153861118270144424	275.0	68.0	80.5787	66.0815

14817 rows × 5 columns

In [186]:

```
num_cols_sum = ((dv.groupby(["trip_uuid","time_taken"])["segment_actual_time" , 'segmer
maxx = num_cols_sum.reset_index()
maxx
```

Out[186]:

	trip_uuid	segment_actual_time	segment_osrm_time	segment_osrm_distance
0	trip-153671041653548748	1548.0	1008.0	1320.4733
1	trip-153671042288605164	141.0	65.0	84.1894
2	trip-153671043369099517	3308.0	1941.0	2545.2678
3	trip-153671046011330457	59.0	16.0	19.8766
4	trip-153671052974046625	340.0	115.0	146.7919
...	...	...	...	...
14812	trip-153861095625827784	82.0	62.0	64.8551
14813	trip-153861104386292051	21.0	11.0	16.0883
14814	trip-153861106442901555	281.0	88.0	104.8866
14815	trip-153861115439069069	258.0	221.0	223.5324
14816	trip-153861118270144424	274.0	67.0	80.5787

14817 rows × 4 columns

In [246]:

```
numerical_df = minn.merge(maxx , on = 'trip_uuid')
numerical_df
```

Out[246]:

	trip_uuid	actual_time	osrm_time	osrm_distance	actual_distance_to_destinati
0	trip-153671041653548748	1562.0	743.0	991.3523	824.7328
1	trip-153671042288605164	143.0	68.0	85.1110	73.1869
2	trip-153671043369099517	3347.0	1741.0	2372.0852	1932.2739
3	trip-153671046011330457	59.0	15.0	19.6800	17.1752
4	trip-153671052974046625	341.0	117.0	146.7918	127.4485
...	...	...	...	...	
14812	trip-153861095625827784	83.0	62.0	73.4630	57.7623
14813	trip-153861104386292051	21.0	12.0	16.0882	15.5137
14814	trip-153861106442901555	282.0	54.0	63.2841	38.6848
14815	trip-153861115439069069	264.0	184.0	177.6635	134.7238
14816	trip-153861118270144424	275.0	68.0	80.5787	66.0815

14817 rows × 8 columns

Handle the outliers using the IQR method.

In [247]:

```
numerical_df['actual_time_in_hour'] = numerical_df['actual_time']/60
numerical_df['osrm_time_in_hour'] = numerical_df['osrm_time']/60
numerical_df['segment_actual_time_in_hour'] = numerical_df['segment_actual_time']/60
numerical_df['segment_osrm_time_in_hour'] = numerical_df['segment_osrm_time']/60
numerical_df.drop(['actual_time', 'osrm_time' ,'segment_actual_time' , 'segment_osrm_ti
nf = numerical_df.copy()
numerical_df
```

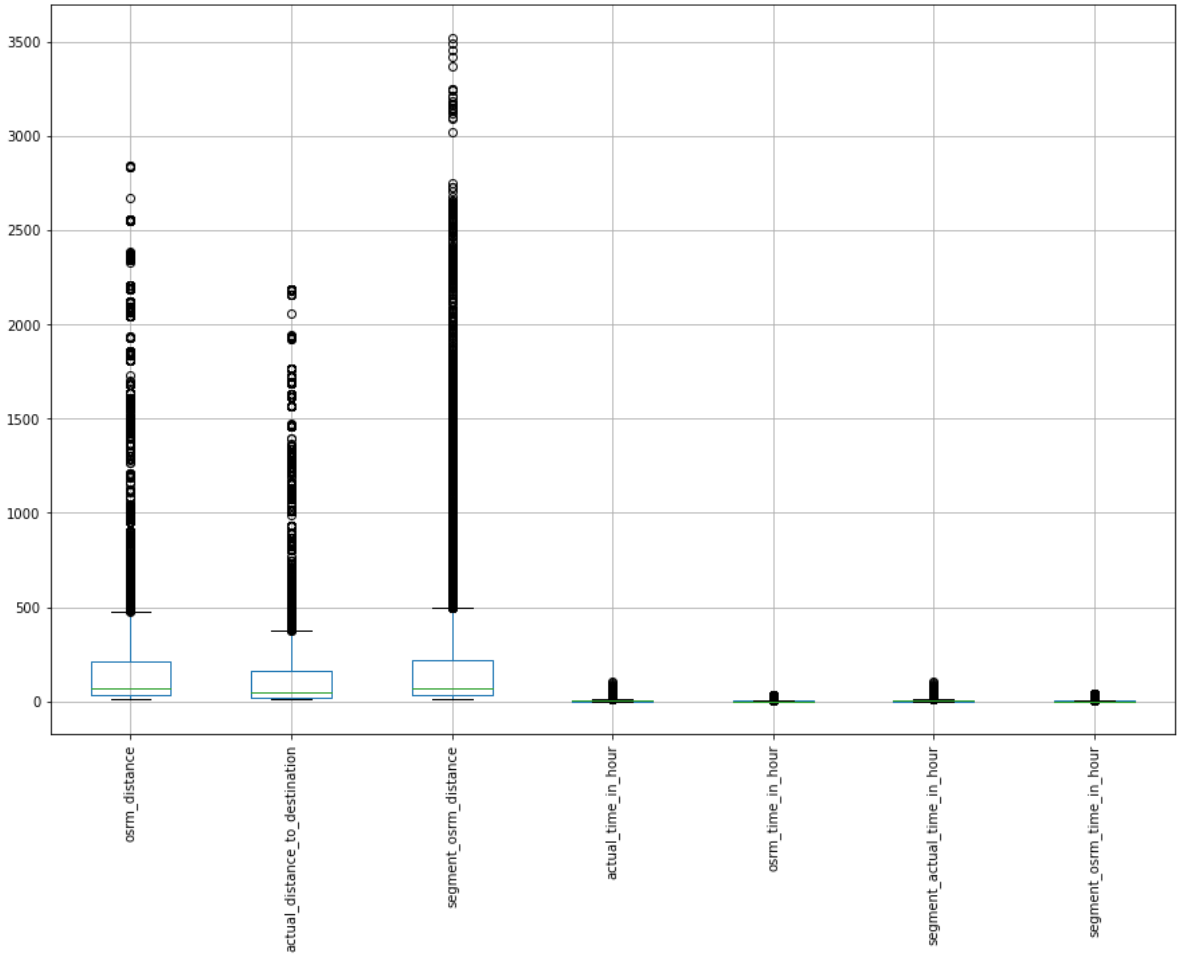
Out[247]:

	trip_uuid	osrm_distance	actual_distance_to_destination	segment_osrm_distar
0	trip-153671041653548748	991.3523	824.732854	1320.47
1	trip-153671042288605164	85.1110	73.186911	84.18
2	trip-153671043369099517	2372.0852	1932.273969	2545.26
3	trip-153671046011330457	19.6800	17.175274	19.87
4	trip-153671052974046625	146.7918	127.448500	146.79
...	...	...	...	
14812	trip-153861095625827784	73.4630	57.762332	64.85
14813	trip-153861104386292051	16.0882	15.513784	16.08
14814	trip-153861106442901555	63.2841	38.684839	104.88
14815	trip-153861115439069069	177.6635	134.723836	223.53
14816	trip-153861118270144424	80.5787	66.081533	80.57

14817 rows × 8 columns

In [248]:

```
numerical_df.boxplot(figsize = (15,10))
plt.xticks(rotation = 90)
plt.show()

```



In [250]:

```

nf.drop('trip_uuid', axis=1 , inplace = True)

```

In [251]:

```
1 nf
```

Out[251]:

	osrm_distance	actual_distance_to_destination	segment_osrm_distance	actual_time_in_ho
0	991.3523	824.732854	1320.4733	26.0333
1	85.1110	73.186911	84.1894	2.3833
2	2372.0852	1932.273969	2545.2678	55.7833
3	19.6800	17.175274	19.8766	0.9833
4	146.7918	127.448500	146.7919	5.6833
...	...	...	...	
14812	73.4630	57.762332	64.8551	1.3833
14813	16.0882	15.513784	16.0883	0.3500
14814	63.2841	38.684839	104.8866	4.7000
14815	177.6635	134.723836	223.5324	4.4000
14816	80.5787	66.081533	80.5787	4.5833

14817 rows × 7 columns

In [224]:

```
1 numerical_df.isna().sum()
```

Out[224]:

```
osrm_distance                   0
actual_distance_to_destination  0
segment_osrm_distance           0
actual_time_in_hour             0
osrm_time_in_hour               0
segment_actual_time_in_hour     0
segment_osrm_time_in_hour       0
dtype: int64
```

In [ ]:

```
1  # def remove_outlier(numerical_df):
2  #     inlier, outlier = [],[]
3  #     data_s = sorted(numerical_df)
4  #     q1,q3 = np.percentile(data_s,[25,75])
5  #     iqr = q1 - q3
6  #     upper_bound = q3 - (1.5*iqr)
7  #     lower_bound = q1 + (1.5*iqr)
8  #     for i in range(len(numerical_df)):
9  #         if numerical_df[i] > lower_bound and numerical_df[i] < upper_bound:
10 #             inlier.append(numerical_df[i])
11 #         else:
12 #             outlier.append(numerical_df[i])
13 #     return inlier,outlier
```

In [252]:

```
import numpy as np
from scipy import stats
nf[(np.abs(stats.zscore(nf)) < 3).all(axis=1)]
```

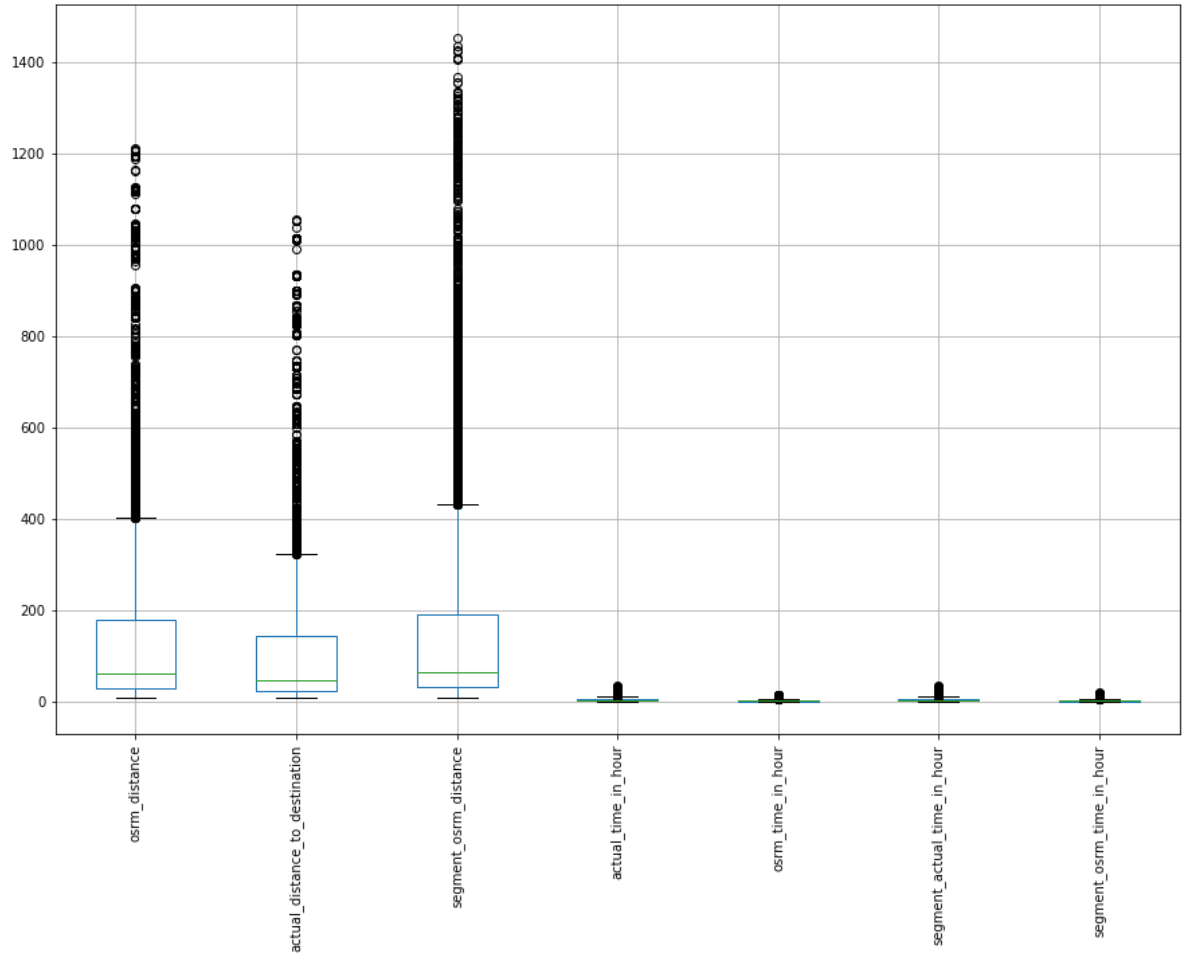
Out[252]:

	osrm_distance	actual_distance_to_destination	segment_osrm_distance	actual_time_in_ho
0	991.3523	824.732854	1320.4733	26.0333
1	85.1110	73.186911	84.1894	2.3833
3	19.6800	17.175274	19.8766	0.9833
4	146.7918	127.448500	146.7919	5.6833
5	28.0647	24.597048	28.0647	1.0166
...	...	...	...	
14812	73.4630	57.762332	64.8551	1.3833
14813	16.0882	15.513784	16.0883	0.3500
14814	63.2841	38.684839	104.8866	4.7000
14815	177.6635	134.723836	223.5324	4.4000
14816	80.5787	66.081533	80.5787	4.5833

14173 rows × 7 columns

In [253]:

```
nf[(np.abs(stats.zscore(nf)) < 3).all(axis=1)].boxplot(figsize = (15,10))
plt.xticks(rotation = 90)
plt.show()

```



Observations

- All the outliers lesser and larger than 3 std deviations are removed from all Numerical columns after creating a dataset called numeriacal df.

One Hot Encoding

In [227]:

```
ohe = dv.groupby('trip_uuid')['route_type'].unique().reset_index()
ohe
```

Out[227]:

	trip_uuid	route_type
0	trip-153671041653548748	[FTL]
1	trip-153671042288605164	[Carting]
2	trip-153671043369099517	[FTL]
3	trip-153671046011330457	[Carting]
4	trip-153671052974046625	[FTL]
...	...	...
14812	trip-153861095625827784	[Carting]
14813	trip-153861104386292051	[Carting]
14814	trip-153861106442901555	[Carting]
14815	trip-153861115439069069	[Carting]
14816	trip-153861118270144424	[FTL]

14817 rows × 2 columns

In [228]:

```
ohe['routetype'] = ohe['route_type'].apply(lambda x: ','.join(map(str, x)))
ohe
```

Out[228]:

	trip_uuid	route_type	routetype
0	trip-153671041653548748	[FTL]	FTL
1	trip-153671042288605164	[Carting]	Carting
2	trip-153671043369099517	[FTL]	FTL
3	trip-153671046011330457	[Carting]	Carting
4	trip-153671052974046625	[FTL]	FTL
...	...	...	...
14812	trip-153861095625827784	[Carting]	Carting
14813	trip-153861104386292051	[Carting]	Carting
14814	trip-153861106442901555	[Carting]	Carting
14815	trip-153861115439069069	[Carting]	Carting
14816	trip-153861118270144424	[FTL]	FTL

14817 rows × 3 columns

In [229]:

```
ohe.drop('route_type' ,axis = 1, inplace = True)
```

In [230]:

```
ohe
```

Out[230]:

	trip_uuid	routetype
0	trip-153671041653548748	FTL
1	trip-153671042288605164	Carting
2	trip-153671043369099517	FTL
3	trip-153671046011330457	Carting
4	trip-153671052974046625	FTL
...	...	...
14812	trip-153861095625827784	Carting
14813	trip-153861104386292051	Carting
14814	trip-153861106442901555	Carting
14815	trip-153861115439069069	Carting
14816	trip-153861118270144424	FTL

14817 rows × 2 columns

In [254]:

```
ohe_df = ohe.merge(numerical_df , on = 'trip_uuid')
ohe_df
```

Out[254]:

	trip_uuid	routetype	osrm_distance	actual_distance_to_destination	segment_
0	trip-153671041653548748	FTL	991.3523	824.732854	
1	trip-153671042288605164	Carting	85.1110	73.186911	
2	trip-153671043369099517	FTL	2372.0852	1932.273969	
3	trip-153671046011330457	Carting	19.6800	17.175274	
4	trip-153671052974046625	FTL	146.7918	127.448500	
...	...	...	...	...	
14812	trip-153861095625827784	Carting	73.4630	57.762332	
14813	trip-153861104386292051	Carting	16.0882	15.513784	
14814	trip-153861106442901555	Carting	63.2841	38.684839	
14815	trip-153861115439069069	Carting	177.6635	134.723836	
14816	trip-153861118270144424	FTL	80.5787	66.081533	

14817 rows × 9 columns

In [255]:

```
one_hot_encoded_data = pd.get_dummies(ohe_df, columns = ['routetype'])
one_hot_encoded_data
```

Out[255]:

	trip_uuid	osrm_distance	actual_distance_to_destination	segment_osrm_distar
0	trip-153671041653548748	991.3523	824.732854	1320.47
1	trip-153671042288605164	85.1110	73.186911	84.18
2	trip-153671043369099517	2372.0852	1932.273969	2545.26
3	trip-153671046011330457	19.6800	17.175274	19.87
4	trip-153671052974046625	146.7918	127.448500	146.79
...	...	...	...	
14812	trip-153861095625827784	73.4630	57.762332	64.85
14813	trip-153861104386292051	16.0882	15.513784	16.08
14814	trip-153861106442901555	63.2841	38.684839	104.88
14815	trip-153861115439069069	177.6635	134.723836	223.53
14816	trip-153861118270144424	80.5787	66.081533	80.57

14817 rows × 10 columns

STANDARD SCALER

In [260]:

```
1 nf
```

Out[260]:

	osrm_distance	actual_distance_to_destination	segment_osrm_distance	actual_time_in_ho
0	991.3523	824.732854	1320.4733	26.0333
1	85.1110	73.186911	84.1894	2.3833
2	2372.0852	1932.273969	2545.2678	55.7833
3	19.6800	17.175274	19.8766	0.9833
4	146.7918	127.448500	146.7919	5.6833
...	...	...	...	
14812	73.4630	57.762332	64.8551	1.3833
14813	16.0882	15.513784	16.0883	0.3500
14814	63.2841	38.684839	104.8866	4.7000
14815	177.6635	134.723836	223.5324	4.4000
14816	80.5787	66.081533	80.5787	4.5833

14817 rows × 7 columns

In [261]:

```
1 from sklearn.preprocessing import StandardScaler
2 from sklearn.preprocessing import MinMaxScaler
```

In [264]:

```
scaler = StandardScaler()
std_data = scaler.fit_transform(nf)
std_data = pd.DataFrame(std_data, columns=['osrm_distance',
 'actual_distance_to_destination',
 'segment_osrm_distance',
 'actual_time_in_hour',
 'osrm_time_in_hour',
 'segment_actual_time_in_hour',
 'segment_osrm_time_in_hour'])
std_data.head()
```

Out[264]:

	osrm_distance	actual_distance_to_destination	segment_osrm_distance	actual_time_in_hour	c
0	2.120596	2.160182	2.633784	2.146194	
1	-0.323634	-0.299454	-0.333670	-0.381473	
2	5.844580	5.784909	5.573660	5.325817	
3	-0.500108	-0.482768	-0.488040	-0.531102	
4	-0.157274	-0.121869	-0.183405	-0.028775	

Min Max Scaler

In [265]:

```
scaler = MinMaxScaler()
MinMax_data = scaler.fit_transform(nf)
MinMax_data = pd.DataFrame(MinMax_data, columns=['osrm_distance',
 'actual_distance_to_destination',
 'segment_osrm_distance',
 'actual_time_in_hour',
 'osrm_time_in_hour',
 'segment_actual_time_in_hour',
 'segment_osrm_time_in_hour'])
MinMax_data.head()
```

Out[265]:

	osrm_distance	actual_distance_to_destination	segment_osrm_distance	actual_time_in_hour	c
0	0.346972	0.374449	0.373134	0.248242	
1	0.026859	0.029463	0.021373	0.021419	
2	0.834689	0.882850	0.721625	0.533568	
3	0.003747	0.003752	0.003074	0.007992	
4	0.048647	0.054371	0.039185	0.053069	

Insights

- General Inference from Raw Data
  - Train and test data imbalance in numbers but zero missing elements
  - 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)
  - From 1504 total different routes , we have
    - 922 (61%) of the routes are Carting , which consists of small vehicles and
    - 582 (38.69%) of total routes are FTL : which are Full Truck Load get to the destination sooner. as no otther pickups or drop offs along the way .
  - we have 14817 different trips happended between source to destinations.
  - looks like source id and source name are the same things , name is the name of warehouse and id is the unique id for the same but there is a difference of 10
  - the destination center and destination id just like the source are the same but a difference of 13 missing values

- HYPOTHESIS TEST RESULTS
  - (time_taken_btwn_odstart_and_od_end and start_scan_to_end_scan) are closly 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.
  - mean actual time VS orsm estimate mean time
    - from ttestwe can conclude , tht population mean actual time taken to complete delivert from source to warehouse and orsm estimate mean time for population are not same.
      - actual time is higher than the osrm estimated time for delivery.
  - Actual Time Vs segment osrm distance
    - from two sample ttest , we can conclude that Population average for Actual Time taken to complete delivery trip and segment osrm distance are not same.
  - OSRM Time Vs segment osrm time
    - from two sample ttest , we can conclude that Population average for OSRM Time taken to complete delivery trip and segment osrm time are not same.
- Plotting and Aggregating
  - In[144] The above table shows the source state and destination state for a single trip . As you can see, for a single trip_uuid there are multiple check points.
  - In[197] we can clearly see that when ever the trip is long wrt distance the number of check points are also more and vice versa
  - In[176] The busiest state wrt to trips are in the above order .we can infer that the busiest states are mostly metro states as well as industrial states.
  - In[253] All the outliers lesser and larger than 3 std deviations are removed from all Numerical columns after creating a dataset called numeriacal df.

Plotting Insights

- the souce to destination city routes having largest numbers of trip happening having large distnaces :
  - Guwahati TO Bhiwandi, Bengaluru TO Chandigarh, Bengaluru TO Delhi,Gurgaon TO MAA Chennai Airport,Bhiwandi TO Kolkata, Bengaluru TO Kolkata, Gurgaon TO Hyderabad, Gurgaon TO Kolkata
- Routes which are busiest from source to destinations and states in which highest activities are noticed :
  - Delhi to Haryana is the busiest route, having more than 400 trips in between. Some of such busy routes are Haryana to Uttar Pradesh , Chandigarh to Punjab , Delhi to Uttar Pradesh .
- Within the state , Maharashtra , Karnataka, Tamil Nadu are some states having above 1000 trips.

- From above chart are some warehouse having Maximum traffic and hence busiest junctions.
  - Bengaluru Karnataka, Gurgaon Haryana, Mumbai Maharashtra, Hyderabad Telangana, 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.

Recommendation

- In the busiest corridors and Metro cities on which the average delivery time is larger, it would be appropreate to deliver the products via small carting vehicles than FLT's but keeping in mind the expenditure and the income generated we need to strike a balance between both based on the data aggregation done with respect to state , trips and average time and number of orders.
- Increasing the connectivity in Tier 2 and Tier 3 cities along with professional tie ups with several e-commerece giants can increase the revenue as well as the reputation on Connectivity across boarders.
- We can work on optimizing the scanning time on both ends which is start scanning time and end scanning time so that the delivery time can be equated to the OSRM delivery time.

In [ ]:

```
1
```