

DELHIVERY - Business CaseStudy

Feature Engineering

Analysed by : **Atharva Samindre**



Introduction:

-  **Delhivery**, established in 2011, is India's foremost logistics and supply chain service provider, offering a comprehensive range of solutions including express parcel transportation, warehousing, and last-mile delivery.
- Leveraging advanced technology and a vast delivery network, Delhivery efficiently manages nationwide movement of goods, earning trust across businesses of all sizes for its dedication to innovation and customer satisfaction.
- As the largest fully integrated player in India by revenue in Fiscal 2021, Delhivery aims to lead the industry by pioneering the commerce operating system, driven by top-tier infrastructure, logistics operations, and innovative data intelligence initiatives led by its Data team.

◆ Why this case study?

Delhivery aims to establish itself as the premier player in the logistics industry. This case study is of paramount importance as it aligns with the company's core objectives and operational excellence.

It provides a practical framework for understanding and processing data, which is integral to their operations. By leveraging data engineering pipelines and data analysis techniques, Delhivery can achieve several critical goals.

First, it allows them to ensure data integrity and quality by addressing missing values and structuring the dataset appropriately.

Second, it enables the extraction of valuable features from raw data, which can be utilized for building accurate forecasting models.

Moreover, it facilitates the identification of patterns, insights, and actionable recommendations crucial for optimizing their logistics operations.

By conducting hypothesis testing and outlier detection, Delhivery can refine their processes and further enhance the quality of service they provide.

How can you help here?

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

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

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

Features of the dataset:

- Column Profiling:

Feature	Description
data	tells whether the data is testing or training data
trip_creation_time	Timestamp of trip creation
route_schedule_uuid	Unique ID for a particular route schedule
route_type	Transportation type
a. 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
b. Carting	Handling system consisting of small vehicles (carts)
trip_uuid	Unique ID given to a particular trip (A trip may include different source and destination centers)
source_center	Source ID of trip origin
source_name	Source Name of trip origin
destination_center	Destination ID
destination_name	Destination Name
od_start_time	Trip start time
od_end_time	Trip end time
start_scan_to_end_scan	Time taken to deliver from source to destination
is_cutoff	Unknown field
cutoff_factor	Unknown field
cutoff_timestamp	Unknown field

Feature	Description
actual_distance_to_destination	Distance in kms between source and destination warehouse
actual_time	Actual time taken to complete the delivery (Cumulative)
osrm_time	An open-source routing engine time calculator which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) and gives the time (Cumulative)
osrm_distance	An open-source routing engine which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) (Cumulative)
factor	Unknown field
segment_actual_time	This is a segment time. Time taken by the subset of the package delivery
segment_osrm_time	This is the OSRM segment time. Time taken by the subset of the package delivery
segment_osrm_distance	This is the OSRM distance. Distance covered by subset of the package delivery
segment_factor	Unknown field

```
In [1]: # importing the required modules and packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re
from scipy.stats import norm,zscore,boxcox,probplot
from scipy.stats import ttest_ind,ttest_rel,mannwhitneyu,wilcoxon
from scipy.stats import shapiro,levene,kstest,anderson
import statsmodels.api as sm
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler , MinMaxScaler , OneHotEncoder
import warnings
warnings.filterwarnings('ignore')
```

```
In [2]: # pd_reading the data
delhivery_data = pd.read_csv('delhivery_data.csv')
```

```
In [3]: # setting the option of displaying all the columns
pd.set_option('display.max_columns', 50)
```

```
In [4]: # making a deep copy for backup
dd = delhivery_data.copy()
dd.head()
```

Out[4]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_cer
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	trip- 153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620/
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	trip- 153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620/
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	trip- 153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620/
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	trip- 153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620/
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	trip- 153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620/



👀 Exploration of data :

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

```
Out[5]: (144867, 24)
```

```
In [6]: dd.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')
```

```
In [7]: dd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column           Non-Null Count   Dtype  
 ---  -- 
 0   data              144867 non-null    object  
 1   trip_creation_time 144867 non-null    object  
 2   route_schedule_uuid 144867 non-null    object  
 3   route_type          144867 non-null    object  
 4   trip_uuid           144867 non-null    object  
 5   source_center        144867 non-null    object  
 6   source_name          144574 non-null    object  
 7   destination_center   144867 non-null    object  
 8   destination_name     144606 non-null    object  
 9   od_start_time        144867 non-null    object  
 10  od_end_time         144867 non-null    object  
 11  start_scan_to_end_scan 144867 non-null    float64
 12  is_cutoff            144867 non-null    bool    
 13  cutoff_factor        144867 non-null    int64  
 14  cutoff_timestamp     144867 non-null    object  
 15  actual_distance_to_destination 144867 non-null    float64
 16  actual_time          144867 non-null    float64
 17  osrm_time            144867 non-null    float64
 18  osrm_distance        144867 non-null    float64
 19  factor               144867 non-null    float64
 20  segment_actual_time  144867 non-null    float64
 21  segment_osrm_time    144867 non-null    float64
 22  segment_osrm_distance 144867 non-null    float64
 23  segment_factor       144867 non-null    float64
dtypes: bool(1), float64(10), int64(1), object(12)
memory usage: 25.6+ MB
```

📊 Statistical Summary

```
In [8]: dd.describe().T
```

Out[8]:

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

In [9]: `dd.describe(include=object).T`

Out[9]:

		count	unique	top	freq
	data	144867	2	training	104858
	trip_creation_time	144867	14817	2018-09-28 05:23:15.359220	101
	route_schedule_uuid	144867	1504	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	1812
	route_type	144867	2	FTL	99660
	trip_uuid	144867	14817	trip-153811219535896559	101
	source_center	144867	1508	IND000000ACB	23347
	source_name	144574	1498	Gurgaon_Bilaspur_HB (Haryana)	23347
	destination_center	144867	1481	IND000000ACB	15192
	destination_name	144606	1468	Gurgaon_Bilaspur_HB (Haryana)	15192
	od_start_time	144867	26369	2018-09-21 18:37:09.322207	81
	od_end_time	144867	26369	2018-09-24 09:59:15.691618	81
	cutoff_timestamp	144867	93180	2018-09-24 05:19:20	40

📋 📋 Duplicate Detection

In [10]: `dd[dd.duplicated()]`Out[10]: `data trip_creation_time route_schedule_uuid route_type trip_uuid source_center source_name destination_center destination_name`

💡 Insights

- The dataset does not contain any duplicates.

◆ ? Null Detection

```
In [11]: dd.isna().any()
```

```
Out[11]: data           False
          trip_creation_time  False
          route_schedule_uuid  False
          route_type            False
          trip_uuid              False
          source_center          False
          source_name             True
          destination_center     False
          destination_name        True
          od_start_time           False
          od_end_time              False
          start_scan_to_end_scan  False
          is_cutoff                False
          cutoff_factor            False
          cutoff_timestamp          False
          actual_distance_to_destination  False
          actual_time               False
          osrm_time                 False
          osrm_distance              False
          factor                     False
          segment_actual_time       False
          segment_osrm_time          False
          segment_osrm_distance      False
          segment_factor              False
          dtype: bool
```

```
In [12]: dd.isnull().sum()
```

```
Out[12]: 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
```

```
In [13]: def missing_data(df):
    total_missing_df = df.isnull().sum().sort_values(ascending=False)
    percent_missing_df = (df.isnull().sum()/df.isna().count()*100).sort_values(ascending=False) # ----> /Len(dd)
    missing_data_df = pd.concat([total_missing_df, percent_missing_df], axis=1, keys=['Total', 'Percent'])
    return missing_data_df

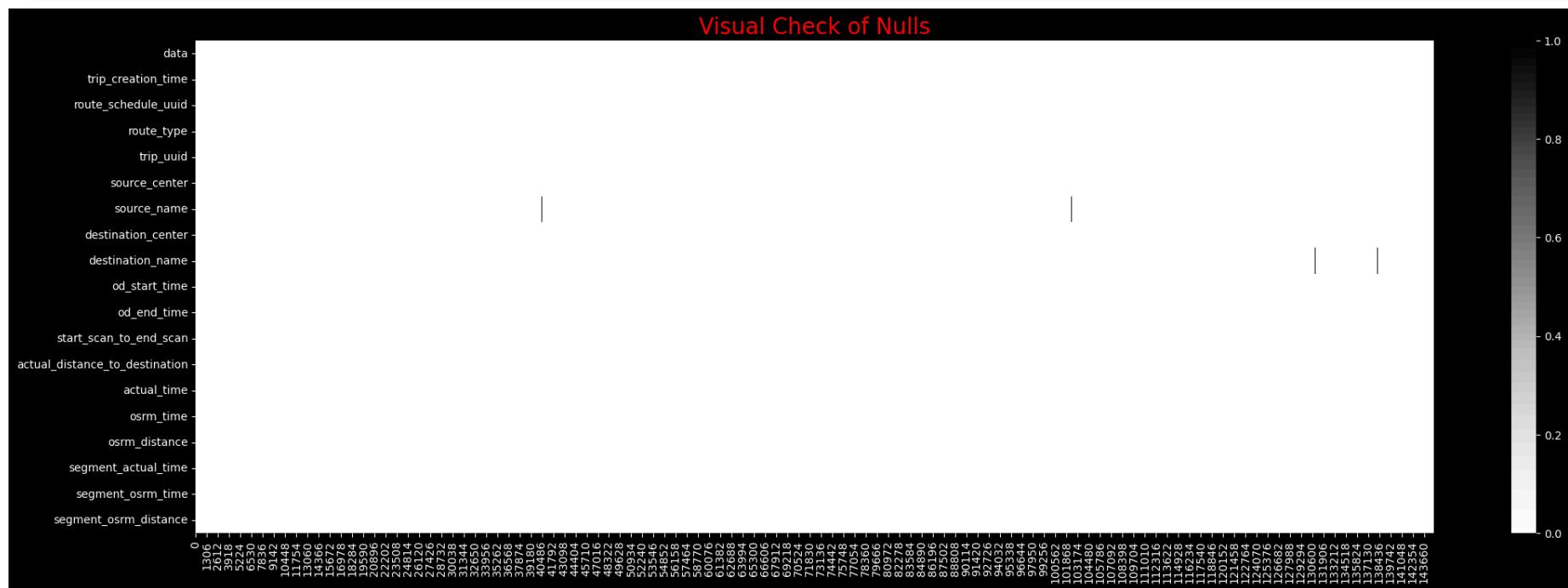
missing_pct = missing_data(dd)
missing_pct[missing_pct['Total'] > 0]
```

Out[13]:

	Total	Percent
source_name	293	0.202254
destination_name	261	0.180165

In [16]:

```
plt.figure(figsize=(25,8))
plt.style.use('dark_background')
sns.heatmap(dd.isnull().T,cmap='Greys')
plt.title('Visual Check of Nulls',fontsize=20,color='r')
plt.show()
```



In [15]: # Dropping unknown fields

```
unknown_fields = ['is_cutoff', 'cutoff_factor', 'cutoff_timestamp', 'factor', 'segment_factor']
dd = dd.drop(columns = unknown_fields)
```

In [17]: dd.sample()

Out[17]:	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination
133488	test	2018-09-30 05:56:48.299467	thanos::sroute:6be6529b-f2ad-4714-b7ab-ac58f24...	FTL	trip-153828700829921150	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND60(



In [18]: dd.shape

Out[18]: (144867, 19)

```
In [19]: #checking the unique values for columns
for _ in dd.columns:
    print()
    print(f'Total Unique Values in {_} column are :- {dd[_].nunique()}')
    print(f'Unique Values in {_} column are :-\n{dd[_].unique()}')
    print()
    print('-'*120)
```

Total Unique Values in data column are :- 2
Unique Values in data column are :-
['training' 'test']

Total Unique Values in trip_creation_time column are :- 14817
Unique Values in trip_creation_time column are :-
['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']

Total Unique Values in route_schedule_uuid column are :- 1504
Unique Values in route_schedule_uuid column are :-
['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']

Total Unique Values in route_type column are :- 2
Unique Values in route_type column are :-
['Carting' 'FTL']

Total Unique Values in trip_uuid column are :- 14817
Unique Values in trip_uuid column are :-
['trip-153741093647649320' 'trip-153768492602129387'
'trip-153693976643699843' ... 'trip-153761584139918815'
'trip-153718412883843340' 'trip-153746066843555182']

Total Unique Values in source_center column are :- 1508
Unique Values in source_center column are :-

```
['IND388121AAA' 'IND388620AAB' 'IND421302AAG' ... 'IND361335AAA'  
'IND562132AAC' 'IND639104AAB']
```

Total Unique Values in source_name column are :- 1498

Unique Values in source_name column are :-

```
['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)']
```

Total Unique Values in destination_center column are :- 1481

Unique Values in destination_center column are :-

```
['IND388620AAB' 'IND388320AAA' 'IND411033AAA' ... 'IND600004AAA'  
'IND134203AAA' 'IND400701AAA']
```

Total Unique Values in destination_name column are :- 1468

Unique Values in destination_name column are :-

```
['Khambhat_MotvdDPP_D (Gujarat)' 'Anand_Vaghasi_IP (Gujarat)'  
'Pune_Tathawde_H (Maharashtra)' ... 'Chennai_Mylapore (Tamil Nadu)'  
'Naraingarh_Ward2DPP_D (Haryana)' 'Mumbai_Ghansoli_DC (Maharashtra)']
```

Total Unique Values in od_start_time column are :- 26369

Unique Values in od_start_time column are :-

```
['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']
```

Total Unique Values in od_end_time column are :- 26369

Unique Values in od_end_time column are :-

```
['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']
```

Total Unique Values in start_scan_to_end_scan column are :- 1915
Unique Values in start_scan_to_end_scan column are :-
[86. 109. 302. ... 2476. 1161. 2949.]

Total Unique Values in actual_distance_to_destination column are :- 144515
Unique Values in actual_distance_to_destination column are :-
[10.43566024 18.9368423 27.63727904 ... 66.16359134 73.68066734
70.03901016]

Total Unique Values in actual_time column are :- 3182
Unique Values in actual_time column are :-
[14. 24. 40. ... 3169. 3318. 2980.]

Total Unique Values in osrm_time column are :- 1531
Unique Values in osrm_time column are :-
[11. 20. 28. ... 1340. 1439. 1312.]

Total Unique Values in osrm_distance column are :- 138046
Unique Values in osrm_distance column are :-
[11.9653 21.7243 32.5395 ... 97.0933 111.2709 88.7319]

Total Unique Values in segment_actual_time column are :- 747
Unique Values in segment_actual_time column are :-
[1.400e+01 1.000e+01 1.600e+01 2.100e+01 6.000e+00 1.500e+01
2.800e+01 2.600e+01 3.800e+01 3.700e+01 4.100e+01 2.300e+01
4.600e+01 3.000e+01 5.000e+01 9.300e+01 6.200e+01 4.900e+01
2.700e+01 3.500e+01 6.700e+01 2.000e+01 5.100e+01 9.400e+01
1.900e+01 1.200e+01 1.800e+01 1.100e+01 2.000e+00 1.300e+01]

2.400e+01	5.700e+01	1.000e+00	0.000e+00	2.200e+01	2.500e+01
4.500e+01	8.000e+00	3.600e+01	4.200e+01	4.400e+01	7.500e+01
7.800e+01	2.900e+01	3.400e+01	3.200e+01	6.000e+01	4.300e+01
7.900e+01	4.000e+01	6.900e+01	5.800e+01	5.200e+01	4.800e+01
5.500e+01	5.400e+01	4.700e+01	9.000e+00	8.700e+01	1.700e+01
6.800e+01	5.600e+01	3.300e+01	4.000e+00	5.300e+01	2.000e+02
6.500e+01	3.100e+01	8.800e+01	7.000e+00	8.400e+01	8.200e+01
3.900e+01	1.010e+02	1.900e+02	2.020e+02	9.700e+01	8.600e+01
2.910e+02	1.620e+02	4.310e+02	1.060e+02	1.530e+02	8.900e+01
7.300e+01	3.000e+00	1.360e+02	6.600e+01	6.300e+01	5.000e+00
4.220e+02	7.600e+01	8.300e+01	1.230e+02	7.200e+01	1.320e+02
6.850e+02	1.038e+03	6.100e+01	5.900e+01	1.710e+02	1.410e+02
7.000e+01	7.700e+01	1.250e+02	9.200e+01	7.100e+01	6.400e+01
1.040e+02	1.120e+02	9.000e+01	9.800e+01	3.030e+02	1.240e+02
8.100e+01	1.730e+02	9.100e+01	2.200e+02	1.750e+02	2.920e+02
1.170e+02	4.680e+02	6.940e+02	1.090e+02	1.300e+02	3.710e+02
6.110e+02	-2.600e+01	1.480e+02	1.070e+02	5.040e+02	1.150e+02
8.000e+01	1.790e+02	1.080e+02	9.600e+01	1.000e+02	8.500e+01
4.930e+02	4.440e+02	4.240e+02	7.600e+02	1.030e+02	2.320e+02
1.490e+02	2.050e+02	9.420e+02	1.270e+02	7.400e+01	1.660e+02
9.500e+01	1.440e+02	2.220e+02	1.540e+02	1.210e+02	1.840e+02
3.250e+02	1.020e+02	5.270e+02	1.110e+02	5.390e+02	1.590e+02
5.860e+02	3.460e+02	1.180e+02	3.190e+02	2.690e+02	2.950e+02
6.580e+02	2.410e+02	2.960e+02	9.900e+01	1.160e+02	1.140e+02
1.520e+02	-2.100e+01	2.130e+02	1.050e+02	1.220e+02	1.670e+02
-5.000e+00	1.780e+02	1.136e+03	1.190e+02	1.820e+02	2.120e+02
2.930e+02	1.870e+02	1.350e+02	9.010e+02	1.600e+02	2.240e+02
2.790e+02	1.280e+02	6.370e+02	1.310e+02	1.340e+02	5.590e+02
1.580e+02	1.200e+02	5.580e+02	3.940e+02	2.280e+02	2.770e+02
2.040e+02	2.297e+03	1.630e+02	1.130e+02	5.700e+02	2.720e+02
1.510e+02	7.080e+02	1.380e+02	2.810e+02	8.330e+02	-1.000e+00
5.020e+02	1.100e+02	1.570e+02	1.650e+02	2.080e+02	1.910e+02
3.480e+02	1.500e+02	1.430e+02	2.430e+02	2.330e+02	1.470e+02
3.550e+02	1.370e+02	6.590e+02	2.620e+02	2.440e+02	6.200e+02
1.810e+02	1.560e+02	2.700e+02	1.420e+02	6.270e+02	2.480e+02
2.510e+02	1.770e+02	1.860e+02	1.390e+02	2.880e+02	2.530e+02
2.230e+02	1.400e+02	1.330e+02	2.150e+02	3.470e+02	3.560e+02
2.670e+02	1.720e+02	1.290e+02	6.800e+02	3.450e+02	1.450e+02
4.760e+02	3.300e+02	1.880e+02	3.950e+02	3.850e+02	7.190e+02
1.039e+03	5.510e+02	4.590e+02	4.890e+02	2.100e+02	4.740e+02
1.700e+02	9.900e+02	1.550e+02	2.170e+02	3.230e+02	4.850e+02

3.180e+02	1.690e+02	4.190e+02	6.360e+02	1.850e+02	1.260e+02
3.020e+02	6.350e+02	2.030e+02	1.980e+02	3.600e+02	2.090e+02
2.290e+02	7.430e+02	1.117e+03	3.790e+02	3.090e+02	2.500e+02
1.460e+02	2.780e+02	3.140e+02	1.760e+02	2.340e+02	6.300e+02
3.930e+02	1.890e+02	4.160e+02	4.610e+02	2.630e+02	2.380e+02
1.140e+03	4.210e+02	2.260e+02	2.760e+02	1.610e+02	2.060e+02
1.153e+03	4.270e+02	3.390e+02	8.410e+02	5.220e+02	2.140e+02
1.830e+02	4.490e+02	6.120e+02	1.017e+03	1.640e+02	2.160e+02
3.520e+02	4.140e+02	4.470e+02	6.710e+02	2.210e+02	3.800e+02
2.940e+02	2.070e+02	2.370e+02	3.900e+02	3.750e+02	2.010e+02
1.800e+02	1.847e+03	3.880e+02	9.430e+02	5.160e+02	9.930e+02
2.640e+02	5.150e+02	9.470e+02	6.700e+02	1.680e+02	4.280e+02
4.040e+02	3.980e+02	2.270e+02	2.890e+02	3.990e+02	7.320e+02
5.560e+02	5.980e+02	1.050e+03	2.350e+02	7.660e+02	4.920e+02
6.460e+02	3.120e+02	6.950e+02	2.580e+02	3.590e+02	2.360e+02
3.540e+02	6.010e+02	6.400e+02	6.410e+02	5.070e+02	3.720e+02
2.750e+02	5.180e+02	4.400e+02	2.390e+02	5.100e+02	1.716e+03
7.410e+02	9.370e+02	3.860e+02	2.180e+02	2.650e+02	2.820e+02
2.990e+02	1.990e+02	6.890e+02	6.870e+02	3.220e+02	4.460e+02
3.810e+02	4.950e+02	5.930e+02	2.300e+02	9.340e+02	1.143e+03
3.840e+02	9.940e+02	3.160e+02	8.770e+02	4.150e+02	1.086e+03
1.960e+02	8.790e+02	8.960e+02	6.600e+02	5.240e+02	3.360e+02
5.440e+02	3.380e+02	6.620e+02	7.050e+02	3.110e+02	2.680e+02
1.211e+03	5.630e+02	1.981e+03	6.670e+02	3.320e+02	2.310e+02
8.430e+02	8.220e+02	-5.800e+01	5.730e+02	3.000e+02	-2.110e+02
1.740e+02	6.440e+02	3.570e+02	2.190e+02	2.600e+02	1.940e+02
2.110e+02	2.450e+02	2.032e+03	2.860e+02	2.710e+02	4.200e+02
5.430e+02	3.200e+02	5.570e+02	4.320e+02	2.460e+02	5.010e+02
6.480e+02	3.690e+02	5.250e+02	3.920e+02	6.430e+02	7.570e+02
4.780e+02	7.670e+02	2.850e+02	2.250e+02	7.170e+02	1.970e+02
9.580e+02	5.890e+02	4.620e+02	4.050e+02	1.036e+03	6.560e+02
2.351e+03	4.520e+02	6.530e+02	2.740e+02	1.077e+03	5.210e+02
4.830e+02	3.370e+02	4.500e+02	2.610e+02	9.500e+02	6.830e+02
3.500e+02	2.400e+02	3.910e+02	9.910e+02	2.590e+02	2.281e+03
5.910e+02	1.083e+03	3.280e+02	1.124e+03	1.207e+03	3.640e+02
5.360e+02	4.330e+02	5.870e+02	4.510e+02	7.110e+02	2.550e+02
2.980e+02	-1.200e+01	8.500e+02	9.350e+02	2.660e+02	5.480e+02
4.900e+02	2.464e+03	3.650e+02	5.750e+02	6.420e+02	-3.600e+01
1.008e+03	7.800e+02	2.540e+02	1.950e+02	3.820e+02	6.250e+02
6.510e+02	4.770e+02	5.060e+02	6.180e+02	4.350e+02	6.690e+02
9.920e+02	6.470e+02	3.270e+02	9.180e+02	1.067e+03	4.720e+02

1.125e+03	7.370e+02	4.130e+02	1.046e+03	6.970e+02	6.390e+02
4.290e+02	5.850e+02	3.080e+02	3.780e+02	5.400e+02	4.670e+02
5.170e+02	8.690e+02	5.660e+02	2.520e+02	4.340e+02	3.510e+02
6.230e+02	3.060e+02	7.680e+02	4.110e+02	1.065e+03	-4.200e+01
5.110e+02	3.490e+02	5.340e+02	5.090e+02	4.560e+02	1.055e+03
2.490e+02	4.410e+02	-5.100e+01	9.020e+02	3.010e+02	3.350e+02
3.050e+02	9.530e+02	9.050e+02	6.650e+02	4.090e+02	4.980e+02
5.690e+02	5.670e+02	5.950e+02	5.740e+02	7.990e+02	3.130e+02
6.280e+02	4.600e+02	4.870e+02	1.630e+03	7.700e+02	2.120e+03
2.800e+02	4.750e+02	4.800e+02	6.020e+02	4.660e+02	3.530e+02
6.980e+02	1.131e+03	7.620e+02	7.180e+02	9.950e+02	8.090e+02
2.730e+02	3.340e+02	2.970e+02	4.640e+02	4.170e+02	-2.440e+02
6.000e+02	4.970e+02	8.140e+02	9.320e+02	8.700e+02	3.580e+02
1.097e+03	6.260e+02	3.680e+02	7.550e+02	1.061e+03	1.032e+03
4.230e+02	8.910e+02	3.240e+02	3.420e+02	6.820e+02	1.152e+03
5.880e+02	4.100e+02	9.680e+02	7.710e+02	3.290e+02	7.310e+02
-3.000e+00	9.590e+02	4.020e+02	1.166e+03	8.150e+02	4.960e+02
4.390e+02	9.160e+02	1.182e+03	8.490e+02	2.830e+02	8.940e+02
5.940e+02	5.280e+02	2.470e+02	2.900e+02	7.120e+02	1.041e+03
2.420e+02	1.128e+03	-7.400e+01	7.070e+02	6.290e+02	3.330e+02
5.600e+02	6.770e+02	6.100e+02	9.820e+02	1.920e+02	8.550e+02
3.610e+02	6.210e+02	6.740e+02	1.001e+03	9.610e+02	7.330e+02
6.880e+02	7.200e+02	3.310e+02	2.570e+02	2.408e+03	5.310e+02
1.025e+03	8.400e+02	1.584e+03	1.246e+03	7.470e+02	8.930e+02
6.310e+02	6.780e+02	4.010e+02	5.970e+02	9.270e+02	1.167e+03
1.115e+03	3.260e+02	7.150e+02	1.930e+02	1.395e+03	9.670e+02
5.370e+02	6.040e+02	3.210e+02	3.770e+02	8.510e+02	1.016e+03
1.031e+03	4.550e+02	6.220e+02	3.890e+02	5.830e+02	-4.800e+01
4.180e+02	1.047e+03	6.930e+02	9.700e+02	7.040e+02	7.850e+02
-2.000e+00	2.491e+03	9.460e+02	4.650e+02	2.541e+03	1.122e+03
3.051e+03	9.740e+02	9.780e+02	9.040e+02	-1.600e+01	8.530e+02
4.790e+02	1.148e+03	5.720e+02	4.250e+02	5.530e+02	4.060e+02
-7.000e+00	3.070e+02	1.093e+03	4.630e+02	9.390e+02	5.450e+02
1.325e+03	9.150e+02	5.460e+02	7.530e+02	5.290e+02	4.370e+02
5.200e+02	8.300e+02	1.677e+03	1.020e+03	7.480e+02	4.880e+02
6.130e+02	9.510e+02	3.740e+02	7.360e+02	9.330e+02	5.790e+02
2.625e+03	7.520e+02	-1.500e+01	1.192e+03	6.640e+02	1.320e+03
2.870e+02	3.700e+02	1.104e+03]			

Total Unique Values in segment_osrm_time column are :- 214

Unique Values in segment_osrm_time column are :-

```
[1.100e+01 9.000e+00 7.000e+00 1.200e+01 5.000e+00 6.000e+00 1.000e+01  
2.400e+01 2.700e+01 2.600e+01 1.400e+01 1.500e+01 3.000e+01 1.800e+01  
3.800e+01 3.700e+01 2.500e+01 1.700e+01 2.200e+01 3.600e+01 3.200e+01  
1.600e+01 7.000e+01 3.500e+01 4.500e+01 1.300e+01 0.000e+00 8.000e+00  
2.000e+00 1.900e+01 2.300e+01 2.800e+01 2.000e+01 2.100e+01 3.300e+01  
3.400e+01 8.100e+01 3.000e+00 4.400e+01 1.000e+00 4.000e+00 3.900e+01  
4.000e+01 2.900e+01 5.300e+01 3.100e+01 7.500e+01 7.900e+01 9.700e+01  
4.800e+01 4.100e+01 4.300e+01 5.000e+01 5.400e+01 6.800e+01 4.200e+01  
4.600e+01 6.000e+01 5.800e+01 7.600e+01 4.900e+01 1.300e+02 7.800e+01  
6.700e+01 6.400e+01 5.500e+01 5.100e+01 1.420e+02 7.700e+01 7.100e+01  
5.600e+01 6.600e+01 5.900e+01 9.200e+01 6.200e+01 5.200e+01 5.700e+01  
8.000e+01 4.700e+01 7.400e+01 6.500e+01 8.800e+01 7.200e+01 6.900e+01  
7.300e+01 8.400e+01 8.200e+01 8.300e+01 1.540e+02 9.100e+01 6.100e+01  
9.400e+01 1.220e+02 6.300e+01 2.180e+02 9.800e+01 8.700e+01 3.830e+02  
9.000e+01 1.080e+02 9.300e+01 8.600e+01 8.900e+01 1.020e+02 1.600e+02  
2.210e+02 1.050e+02 1.330e+02 1.000e+02 1.060e+02 4.070e+02 8.500e+01  
1.340e+02 4.690e+02 1.800e+02 2.340e+02 1.490e+02 1.010e+02 1.450e+02  
1.140e+02 1.840e+02 2.270e+02 1.740e+02 1.320e+02 9.900e+01 9.600e+01  
1.310e+02 1.110e+02 1.040e+02 1.750e+02 2.300e+02 9.500e+01 1.250e+02  
2.950e+02 1.560e+02 1.160e+02 1.460e+02 1.410e+02 1.030e+02 1.170e+02  
2.310e+02 2.540e+02 2.200e+02 2.330e+02 1.810e+02 1.210e+02 1.270e+02  
3.700e+02 3.750e+02 1.500e+02 1.070e+02 1.610e+02 2.320e+02 1.090e+02  
1.200e+02 1.100e+02 9.970e+02 1.790e+02 1.130e+02 1.660e+02 9.960e+02  
1.240e+02 2.150e+02 1.570e+02 3.620e+02 1.430e+02 1.150e+02 1.280e+02  
1.700e+02 1.440e+02 2.350e+02 1.510e+02 3.560e+02 1.180e+02 1.390e+02  
1.710e+02 1.290e+02 1.190e+02 1.690e+02 1.630e+02 2.040e+02 1.480e+02  
1.830e+02 4.810e+02 3.410e+02 3.280e+02 2.130e+02 1.890e+02 1.910e+02  
1.400e+02 1.470e+02 2.080e+02 2.860e+02 2.160e+02 1.720e+02 1.380e+02  
1.670e+02 2.940e+02 1.230e+02 1.260e+02 2.110e+02 1.611e+03 2.190e+02  
2.490e+02 1.850e+02 1.580e+02 3.240e+02 1.770e+02 4.530e+02 1.520e+02  
1.760e+02 7.370e+02 1.730e+02 1.032e+03]
```

Total Unique Values in segment_osrm_distance column are :- 113799

Unique Values in segment_osrm_distance column are :-

```
[11.9653 9.759 10.8152 ... 20.7053 18.8885 8.8088]
```

Changing the Datatype of Columns

In [20]: dd.sample()

Out[20]:	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination
124444	training	2018-09-23 02:44:13.665024	thanos::sroute:f01c8bbd-655d-42ea-9abf-60d5040...	FTL	trip-153767065366477819	IND821115AAB	Sasaram_Central_L_2 (Bihar)	IND20...

```
In [21]: dd.dtypes
```

```
Out[21]: data          object
         trip_creation_time    object
         route_schedule_uuid   object
         route_type            object
         trip_uuid              object
         source_center          object
         source_name             object
         destination_center     object
         destination_name        object
         od_start_time          object
         od_end_time             object
         start_scan_to_end_scan float64
         actual_distance_to_destination float64
         actual_time              float64
         osrm_time                float64
         osrm_distance            float64
         segment_actual_time      float64
         segment_osrm_time        float64
         segment_osrm_distance    float64
dtype: object
```

```
In [22]: # Converting the datatypes to category for columns like data and route_type as they only have 2 values.  
dd['data'] = dd['data'].astype('category')
```

```
dd['route_type'] = dd['route_type'].astype('category')

# Converting time columns to datetime format
datetime_cols = ['trip_creation_time', 'od_start_time', 'od_end_time']
for _ in datetime_cols:
    dd[_] = pd.to_datetime(dd[_])
```

In [23]: `dd.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   data             144867 non-null   category
 1   trip_creation_time 144867 non-null   datetime64[ns]
 2   route_schedule_uuid 144867 non-null   object  
 3   route_type        144867 non-null   category
 4   trip_uuid         144867 non-null   object  
 5   source_center      144867 non-null   object  
 6   source_name        144574 non-null   object  
 7   destination_center 144867 non-null   object  
 8   destination_name   144606 non-null   object  
 9   od_start_time     144867 non-null   datetime64[ns]
 10  od_end_time       144867 non-null   datetime64[ns]
 11  start_scan_to_end_scan 144867 non-null   float64
 12  actual_distance_to_destination 144867 non-null   float64
 13  actual_time        144867 non-null   float64
 14  osrm_time          144867 non-null   float64
 15  osrm_distance      144867 non-null   float64
 16  segment_actual_time 144867 non-null   float64
 17  segment_osrm_time  144867 non-null   float64
 18  segment_osrm_distance 144867 non-null   float64
dtypes: category(2), datetime64[ns](3), float64(8), object(6)
memory usage: 19.1+ MB
```

```
float_cols = []
for _ in dd.columns:
    if isinstance(dd[_], 'float64'):
        float_cols.append(_)
float_cols
```

see y it didnt work

```
In [24]: float_cols = []
for _ in dd.columns:
    if dd[_].dtype=='float64':
        float_cols.append(_)
float_cols
```

```
Out[24]: ['start_scan_to_end_scan',
          'actual_distance_to_destination',
          'actual_time',
          'osrm_time',
          'osrm_distance',
          'segment_actual_time',
          'segment_osrm_time',
          'segment_osrm_distance']
```

```
In [25]: # reducing the float64 to float32 to save memory
for _ in float_cols:
    dd[_] = dd[_].astype('float32')
```

```
In [26]: dd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   data              144867 non-null   category
 1   trip_creation_time 144867 non-null   datetime64[ns]
 2   route_schedule_uuid 144867 non-null   object  
 3   route_type          144867 non-null   category
 4   trip_uuid           144867 non-null   object  
 5   source_center        144867 non-null   object  
 6   source_name          144574 non-null   object  
 7   destination_center   144867 non-null   object  
 8   destination_name     144606 non-null   object  
 9   od_start_time        144867 non-null   datetime64[ns]
 10  od_end_time         144867 non-null   datetime64[ns]
 11  start_scan_to_end_scan 144867 non-null   float32
 12  actual_distance_to_destination 144867 non-null   float32
 13  actual_time          144867 non-null   float32
 14  osrm_time            144867 non-null   float32
 15  osrm_distance        144867 non-null   float32
 16  segment_actual_time  144867 non-null   float32
 17  segment_osrm_time    144867 non-null   float32
 18  segment_osrm_distance 144867 non-null   float32
dtypes: category(2), datetime64[ns](3), float32(8), object(6)
memory usage: 14.6+ MB
```

💡 Insights:

- Earlier the dataset was using 25.6+ MB of memory but now it has been reduced to 14.6 + MB. Around 40.63 % reduction in the memory usage.

```
In [27]: # Time period of data
dd['trip_creation_time'].max(), dd['trip_creation_time'].min() , dd['trip_creation_time'].max()-dd['trip_creation_time'].min()

Out[27]: (Timestamp('2018-10-03 23:59:42.701692'),
Timestamp('2018-09-12 00:00:16.535741'),
Timedelta('21 days 23:59:26.165951'))
```

```
In [28]: # Time period of data
dd['od_start_time'].max(), dd['od_start_time'].min(), dd['od_start_time'].max() - dd['od_start_time'].min()
```

```
Out[28]: (Timestamp('2018-10-06 04:27:23.392375'),
Timestamp('2018-09-12 00:00:16.535741'),
Timedelta('24 days 04:27:06.856634'))
```

```
In [29]: # Time period of data
dd['od_end_time'].max(), dd['od_end_time'].min(), dd['od_end_time'].max() - dd['od_end_time'].min()
```

```
Out[29]: (Timestamp('2018-10-08 03:00:24.353479'),
Timestamp('2018-09-12 00:50:10.814399'),
Timedelta('26 days 02:10:13.539080'))
```

```
In [30]: data_time_frame = dd['od_end_time'].max() - dd['trip_creation_time'].min()
data_time_frame
```

```
Out[30]: Timedelta('26 days 03:00:07.817738')
```

Null Treatment:

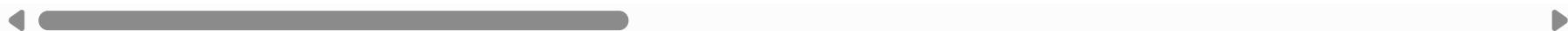
Replace null values in 'source_name' and 'destination_name' columns with 'unknown' through scikit imputation

```
columns_to_impute = ['source_name', 'destination_name']
imputer = SimpleImputer(strategy='constant', fill_value='unknown')
dd[columns_to_impute] = imputer.fit_transform(dd[columns_to_impute])
```

but 'unknown' transactions will be more and to be omitted while analyzing ... Hence no use of imputing

```
In [31]: dd[(dd.source_name.isna())&(dd.destination_name.isna())]
```

Out[31]:		data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center
68006	training	2018-09-26 22:21:56.619259	thanos::sroute:cfb575b8- df26-48f5-8427- 6f48f9d...	FTL	trip- 153800051661903546	IND331022A1B	NaN	IND331001A1	
68007	training	2018-09-26 22:21:56.619259	thanos::sroute:cfb575b8- df26-48f5-8427- 6f48f9d...	FTL	trip- 153800051661903546	IND331022A1B	NaN	IND331001A1	
68008	training	2018-09-26 22:21:56.619259	thanos::sroute:cfb575b8- df26-48f5-8427- 6f48f9d...	FTL	trip- 153800051661903546	IND331022A1B	NaN	IND331001A1	



In [32]: dd[dd.source_name.isna()]

Out[32]:

		data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_ce
112	training		2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4ef4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014.
113	training		2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4ef4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014.
114	training		2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4ef4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014.
115	training		2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4ef4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014.
116	training		2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4ef4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014.
...
144484	test		2018-10-03 09:06:06.690094	thanos::sroute:cbeff3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003.
144485	test		2018-10-03 09:06:06.690094	thanos::sroute:cbeff3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003.
144486	test		2018-10-03 09:06:06.690094	thanos::sroute:cbeff3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003.
144487	test		2018-10-03 09:06:06.690094	thanos::sroute:cbeff3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003.

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_ce
144488	test	2018-10-03 09:06:06.690094	thanos::sroute:cbeff3b6a- 79ea-4d5e-a215- b558a70...	FTL	trip- 153855756668984584	IND282002AAD	NaN	IND474003.

293 rows × 19 columns

```
In [33]: dd[dd.destination_name.isna()]
```

Out[33]:

		data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination
110	training		2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4ef4201d0...	FTL	trip-153786558437756691	IND342601AAA	Piparcity_BsstdDPP_D (Rajasthan)	IND
111	training		2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4ef4201d0...	FTL	trip-153786558437756691	IND342601AAA	Piparcity_BsstdDPP_D (Rajasthan)	IND
982	test		2018-10-01 20:56:18.155260	thanos::sroute:d0ebdacd-e09b-47d3-be77-c9c4a05...	FTL	trip-153842737815495661	IND573103AAA	Arsikere_HsnRdDPP_D (Karnataka)	IND
983	test		2018-10-01 20:56:18.155260	thanos::sroute:d0ebdacd-e09b-47d3-be77-c9c4a05...	FTL	trip-153842737815495661	IND573103AAA	Arsikere_HsnRdDPP_D (Karnataka)	IND
4882	training		2018-09-24 07:18:06.087341	thanos::sroute:2f43f11e-d3ba-4590-9355-82928e1...	FTL	trip-153777348608709328	IND202001AAB	Aligarh_KhirByps_I (Uttar Pradesh)	IND
...
144478	test		2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND
144479	test		2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND
144480	test		2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND
144481	test		2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destina
144482	test	2018-10-03 09:06:06.690094	thanos::sroute:cbeff3b6a- 79ea-4d5e-a215- b558a70...	FTL	trip- 153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND

261 rows × 19 columns

In [34]: `ddd = dd.copy()`In [35]: `missing_source_name = ddd.loc[ddd['source_name'].isnull(), 'source_center'].unique()
missing_source_name`Out[35]: `array(['IND342902A1B', 'IND577116AAA', 'IND282002AAD', 'IND465333A1B',
 'IND841301AAC', 'IND509103AAC', 'IND126116AAA', 'IND331022A1B',
 'IND505326AAB', 'IND852118A1B'], dtype=object)`In [36]: `missing_destination_name = ddd.loc[ddd['destination_name'].isnull(), 'destination_center'].unique()
missing_destination_name`Out[36]: `array(['IND342902A1B', 'IND577116AAA', 'IND282002AAD', 'IND465333A1B',
 'IND841301AAC', 'IND505326AAB', 'IND852118A1B', 'IND126116AAA',
 'IND509103AAC', 'IND221005A1A', 'IND250002AAC', 'IND331001A1C',
 'IND122015AAC'], dtype=object)`In [37]: `# checking if element of one np.array isin another np.array

#np.all(df.loc[df['source_name'].isnull(), 'source_center'].isin(missing_destination_name))

np.in1d(missing_source_name, missing_destination_name).all()`Out[37]: `False`In [38]: `for _ in missing_source_name:
 unique_source_name = ddd.loc[ddd['source_center'] == _, 'source_name'].unique()
 if pd.isna(unique_source_name):
 print("Source Center :", _, "-" * 5, "Source Name :", 'NA')`

```

else :
    print("Source Center :", _, "-" * 5, "Source Name :", unique_source_name)

```

```

Source Center : IND342902A1B ----- Source Name : NA
Source Center : IND577116AAA ----- Source Name : NA
Source Center : IND282002AAD ----- Source Name : NA
Source Center : IND465333A1B ----- Source Name : NA
Source Center : IND841301AAC ----- Source Name : NA
Source Center : IND509103AAC ----- Source Name : NA
Source Center : IND126116AAA ----- Source Name : NA
Source Center : IND331022A1B ----- Source Name : NA
Source Center : IND505326AAB ----- Source Name : NA
Source Center : IND852118A1B ----- Source Name : NA

```

```

In [39]: for _ in missing_destination_name:
    unique_destination_name = ddd.loc[ddd['destination_center'] == _, 'destination_name'].unique()
    if pd.isna(unique_destination_name):
        print("Destination Center :", _, "-" * 5, "Destination Name :", 'NA')
    else :
        print("Destination Center :", _, "-" * 5, "Destination Name :", unique_destination_name)

```

```

Destination Center : IND342902A1B ----- Destination Name : NA
Destination Center : IND577116AAA ----- Destination Name : NA
Destination Center : IND282002AAD ----- Destination Name : NA
Destination Center : IND465333A1B ----- Destination Name : NA
Destination Center : IND841301AAC ----- Destination Name : NA
Destination Center : IND505326AAB ----- Destination Name : NA
Destination Center : IND852118A1B ----- Destination Name : NA
Destination Center : IND126116AAA ----- Destination Name : NA
Destination Center : IND509103AAC ----- Destination Name : NA
Destination Center : IND221005A1A ----- Destination Name : NA
Destination Center : IND250002AAC ----- Destination Name : NA
Destination Center : IND331001A1C ----- Destination Name : NA
Destination Center : IND122015AAC ----- Destination Name : NA

```

```

In [40]: count = 1
for i in missing_destination_name:
    ddd.loc[ddd['destination_center'] == i, 'destination_name'] = ddd.loc[ddd['destination_center'] == i,
                                                                'destination_name'].replace(np.nan, f'location_{count}')
    count += 1

```

```
In [41]: d = {}
for i in missing_source_name:
    d[i] = ddd.loc[ddd['destination_center'] == i, 'destination_name'].unique()
for idx, val in d.items():
    if len(val) == 0:
        d[idx] = [f'location_{count}']
        count += 1
d2 = {}
for idx, val in d.items():
    d2[idx] = val[0]
for i, v in d2.items():
    print(i, v)
```

```
IND342902A1B location_1
IND577116AAA location_2
IND282002AAD location_3
IND465333A1B location_4
IND841301AAC location_5
IND509103AAC location_9
IND126116AAA location_8
IND331022A1B location_14
IND505326AAB location_6
IND852118A1B location_7
```

```
In [45]: for i in missing_source_name:
    ddd.loc[ddd['source_center'] == i, 'source_name'] = ddd.loc[ddd['source_center'] == i, 'source_name'].replace(np.nan, d2[i])
```

```
In [46]: ddd.source_name.value_counts()
```

```
Out[46]: source_name
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
...
Badkulla_Central_DPP_1 (West Bengal) 1
Kasganj_BnkrGate_D (Uttar Pradesh)  1
Shahjhnpur_NavdaCln_D (Uttar Pradesh) 1
Jaunpur_Katghara_D (Uttar Pradesh)  1
Krishnanagar_AnadiDPP_D (West Bengal) 1
Name: count, Length: 1508, dtype: int64
```

```
In [47]: ddd.destination_name.value_counts()
```

```
Out[47]: destination_name
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
...
Vijayawada (Andhra Pradesh)        1
Ranaghat_ArickDPP_D (West Bengal)  1
Mumbai_Sanpada_CP (Maharashtra)    1
Delhi_Lajwanti (Delhi)             1
Luxettipet_ShivaDPP_D (Telangana)  1
Name: count, Length: 1481, dtype: int64
```

even if we replace these nulls with some values, those are not gonna have any impact on the data.... so we can drop it as well..

```
In [49]: 261+290
```

```
Out[49]: 551
```

```
In [50]: len(delhivery_data)
```

```
Out[50]: 144867
```

```
In [51]: 144867 - 551
```

```
Out[51]: 144316
```

```
In [52]: df = dd.dropna()
```

```
In [53]: df.isna().sum().any()
```

```
Out[53]: False
```

```
In [54]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 144316 entries, 0 to 144866
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   data              144316 non-null   category
 1   trip_creation_time 144316 non-null   datetime64[ns]
 2   route_schedule_uuid 144316 non-null   object  
 3   route_type          144316 non-null   category
 4   trip_uuid           144316 non-null   object  
 5   source_center        144316 non-null   object  
 6   source_name          144316 non-null   object  
 7   destination_center   144316 non-null   object  
 8   destination_name     144316 non-null   object  
 9   od_start_time        144316 non-null   datetime64[ns]
 10  od_end_time          144316 non-null   datetime64[ns]
 11  start_scan_to_end_scan 144316 non-null   float32
 12  actual_distance_to_destination 144316 non-null   float32
 13  actual_time          144316 non-null   float32
 14  osrm_time            144316 non-null   float32
 15  osrm_distance         144316 non-null   float32
 16  segment_actual_time   144316 non-null   float32
 17  segment_osrm_time     144316 non-null   float32
 18  segment_osrm_distance 144316 non-null   float32
dtypes: category(2), datetime64[ns](3), float32(8), object(6)
memory usage: 15.7+ MB
```

💡 Insights:

- Only two fields have a tiny fraction of missing values, less than 0.05% of the whole dataset.
- Since we have plenty of data to work with, we're choosing to just get rid of the missing values instead of trying to guess them using methods like using the average or most common value.
- I'm dropping the missing values to keep things simple and not mess up how the features are spread out. But if a lot more data was missing, we could have used other methods like guessing based on what's there or using the most common values.

🕵️ Exploratory Data Analysis

```
In [55]: cp = ['gray', 'red', 'dimgrey', 'tomato', 'dimgray', 'orangered', 'k', 'salmon', 'gray', 'red', 'dimgrey', 'tomato', 'dimgray', 'orangered'
```

```
In [255... df.sample()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_
92389	training	2018-09-14 23:36:50.771430	thanos::sroute:3f001d56-e933-4ba1-a7cf-26828f7...	Carting	trip-153696821077115152	IND515201AAA	Hindupur_Parigi_D (Andhra Pradesh)	IND5153

```
In [270... plt.figure(figsize=(20,4))
plt.suptitle('Pie Percentage distribution', fontsize=13, fontfamily='serif', fontweight='bold', backgroundcolor=cp[-1], color='w')

plt.subplot(141)
plt.pie(df['data'].value_counts(), labels=df['data'].value_counts().index, colors=cp, counterclock=True, explode=(0.02,0.02),
        textprops={'color': 'k', 'fontsize':10}, shadow=True, radius=1, wedgeprops=dict(edgecolor='r', linewidth=0.1, width=0.25))
plt.title('Data-Pie', fontsize=10, fontfamily='serif', fontweight='bold', backgroundcolor=cp[0], color='w')

plt.subplot(142)
a = sns.barplot(x=df['data'].value_counts().index, y=df['data'].value_counts(), palette=cp)
a.bar_label(a.containers[0], label_type='edge', fmt='%.d')
plt.title('Data-Pie', fontsize=10, fontfamily='serif', fontweight='bold', backgroundcolor=cp[0], color='w')
```

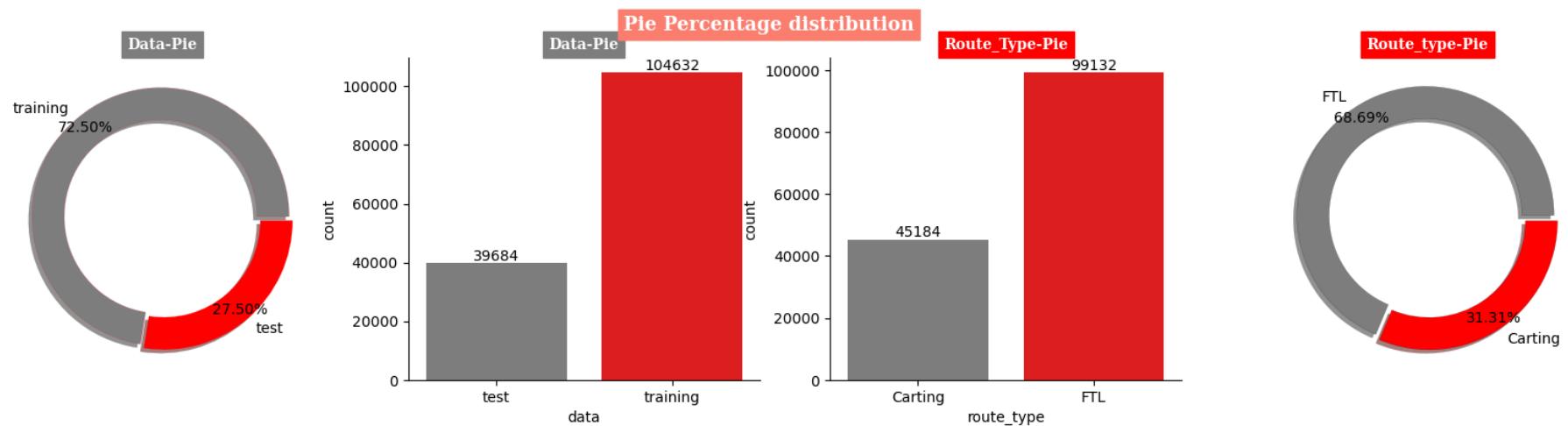
```

plt.subplot(143)
b = sns.barplot(x=df['route_type'].value_counts().index, y=df['route_type'].value_counts(), palette=cp)
b.bar_label(b.containers[0], label_type='edge', fmt='%d')
plt.title('Route_Type-Pie', fontsize=10, fontfamily='serif', fontweight='bold', backgroundcolor=cp[1], color='w')

plt.subplot(144)
plt.pie(df['route_type'].value_counts(), labels=df['route_type'].value_counts().index, colors=cp, counterclock=True, explode=(textprops={'color': 'k', 'fontsize': 10}, shadow=True, radius=1, wedgeprops=dict(edgecolor='k', linewidth=0.1, width=0.25)))
plt.title('Route_type-Pie', fontsize=10, fontfamily='serif', fontweight='bold', backgroundcolor=cp[1], color='w')

sns.despine()
plt.show()

```



```

In [57]: plt.figure(figsize=(25,13))
plt.style.use('default')
plt.style.use('seaborn-bright')

plt.subplot(221)
sns.boxplot(data=df,y='data',x='osrm_time',palette=cp)
plt.title('Data Vs Osrm_time', fontsize=14, fontfamily='serif', fontweight='bold', backgroundcolor='k', color='w')

plt.subplot(222)
sns.boxplot(data=df,y='route_type',x='actual_time',palette=cp)
plt.title('Route_type Vs actual_time', fontsize=14, fontfamily='serif', fontweight='bold', backgroundcolor='r', color='w')

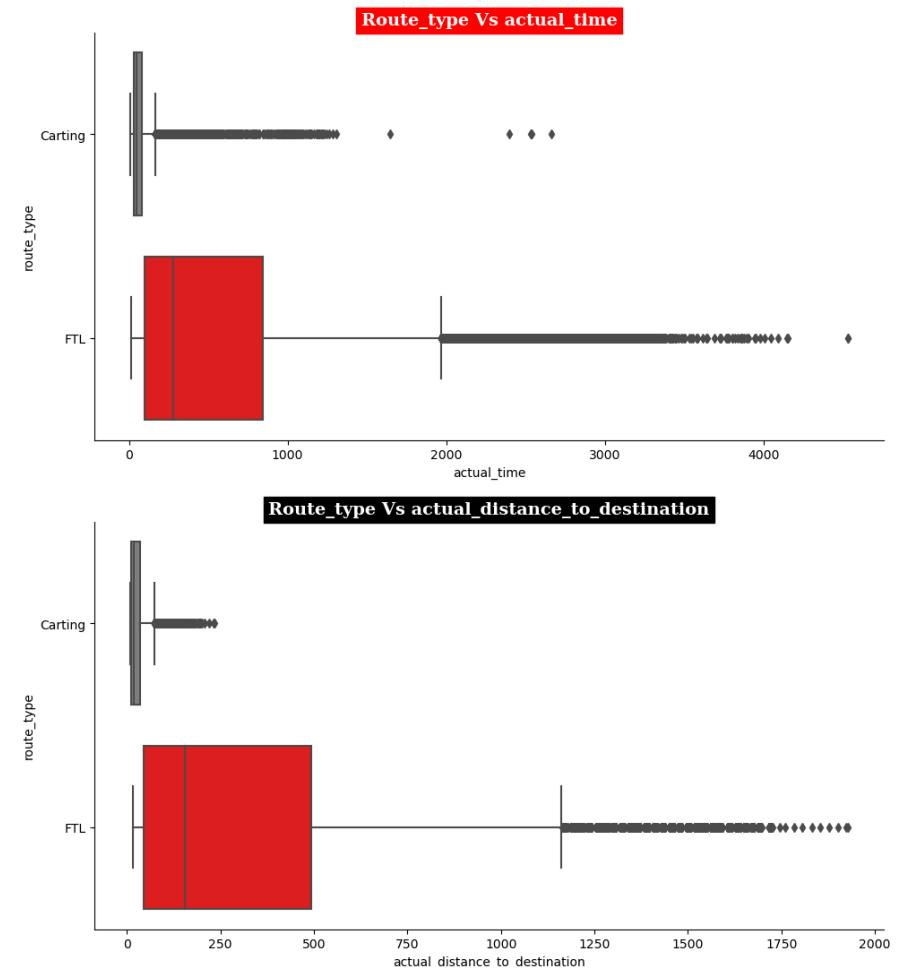
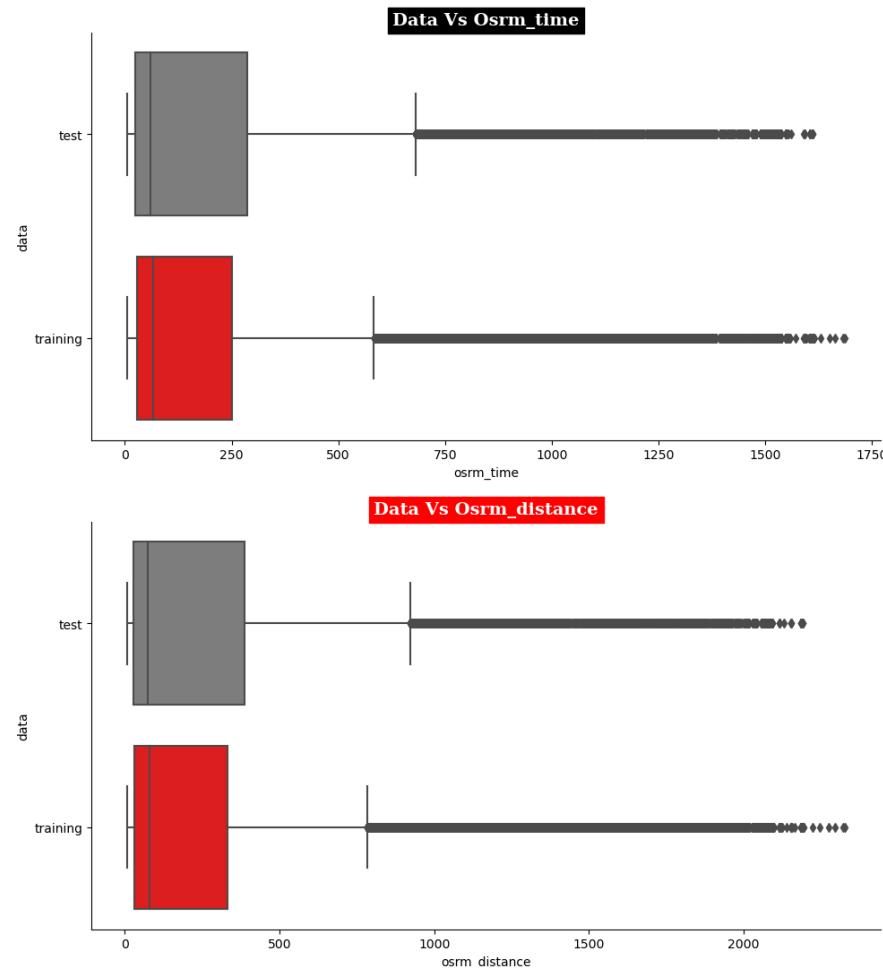
```

```

plt.subplot(223)
sns.boxplot(data=df,y='data',x='osrm_distance',palette=cp)
plt.title('Data Vs Osrm_distance',fontsize=14,fontfamily='serif',fontweight='bold',backgroundcolor='r',color='w')

plt.subplot(224)
sns.boxplot(data=df,y='route_type',x='actual_distance_to_destination',palette=cp)
plt.title('Route_type Vs actual_distance_to_destination',fontsize=14,fontfamily='serif',fontweight='bold',backgroundcolor='k',sns.despine())
plt.show()

```



💡 Observations:

- Both training and test data have the same range of osrm time recorded
 - FTL route type has more actual time compared to Carting. This can also be since FTL is used a lot more than carting in the data available to us
 - Both training and test data have the same range of osrm distance recorded
 - FTL route type has more actual distance compared to Carting. This can also be since FTL is used a lot more than carting in the data available to us
-

2.Merging of rows and aggregation of fields

Merging of rows and aggregation of fields

- Since delivery details of one package are divided into several rows (think of it as connecting flights to reach a particular destination). Now think about how we should treat their fields if we combine these rows? What aggregation would make sense if we merge. What would happen to the numeric fields if we merge the rows.

```
In [58]: # Grouping by segment
# Creating a unique identifier for each segment of a trip

segment_cols = ['segment_actual_time', 'segment_osrm_distance', 'segment_osrm_time']

df['segment_key'] = df['trip_uuid'] + '+' + df['source_center'] + '+' + df['destination_center']

for col in segment_cols:
    df[col + '_sum'] = df.groupby('segment_key')[col].cumsum()

df[['segment_key', 'segment_actual_time', 'segment_actual_time_sum', 'segment_osrm_distance', 'segment_osrm_distance_sum', 'segm
```

Out[58]:

		segment_key	segment_actual_time	segment_actual_time_sum	segment_osrm_distance	segm
0	153741093647649320+IND388121AAA+IND388620AAB	trip-	14.0	14.0	11.965300	
1	153741093647649320+IND388121AAA+IND388620AAB	trip-	10.0	24.0	9.759000	
2	153741093647649320+IND388121AAA+IND388620AAB	trip-	16.0	40.0	10.815200	
3	153741093647649320+IND388121AAA+IND388620AAB	trip-	21.0	61.0	13.022400	
4	153741093647649320+IND388121AAA+IND388620AAB	trip-	6.0	67.0	3.915300	
...
144862	153746066843555182+IND131028AAB+IND000000ACB	trip-	12.0	92.0	8.185800	
144863	153746066843555182+IND131028AAB+IND000000ACB	trip-	26.0	118.0	17.372499	
144864	153746066843555182+IND131028AAB+IND000000ACB	trip-	20.0	138.0	20.705299	
144865	153746066843555182+IND131028AAB+IND000000ACB	trip-	17.0	155.0	18.888500	
144866	153746066843555182+IND131028AAB+IND000000ACB	trip-	268.0	423.0	8.808800	

144316 rows × 7 columns

In [59]: # Aggregating at segment Level & Creating a dictionary for aggregation at segment Level

```
segment_dict = {
    'trip_uuid' : 'first',
```

```
'data': 'first',
'route_type': 'first',
'trip_creation_time': 'first',
'source_name': 'first',
'destination_name': 'last',
'od_start_time': 'first',
'od_end_time': 'last',
'start_scan_to_end_scan': 'first',
'actual_distance_to_destination': 'last',
'actual_time': 'last',
'osrm_time': 'last',
'osrm_distance': 'last',
'segment_actual_time' : 'sum',
'segment_osrm_time' : 'sum',
'segment_osrm_distance' : 'sum',
'segment_actual_time_sum': 'last',
'segment_osrm_time_sum': 'last',
'segment_osrm_distance_sum': 'last',
}

# Grouping by segment_key and aggregating
segment_agg_data = df.groupby('segment_key').agg(segment_dict).reset_index()
segment_agg_data = segment_agg_data.sort_values(by=['segment_key','od_end_time'])
segment_agg_data
```

Out[59]:

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_i
0	trip-153671041653548748+IND209304AAA+IND000000ACB	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Centra (Uttar Pra
1	trip-153671041653548748+IND462022AAA+IND209304AAA	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Bhopal_Trnsp (Madhya Pra
2	trip-153671042288605164+IND561203AAB+IND562101AAA	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaD (Karna
3	trip-153671042288605164+IND572101AAA+IND561203AAB	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Tumkur_Veers (Karna
4	trip-153671043369099517+IND00000ACB+IND160002AAC	trip-153671043369099517	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur (Har
...
26217	trip-153861115439069069+IND628204AAA+IND627657AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Tirchchndr_Shnmgp (Tamil N
26218	trip-153861115439069069+IND628613AAA+IND627005AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Peikulam_SriVnkt (Tamil N
26219	trip-153861115439069069+IND628801AAA+IND628204AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Eral_Busstand_D (N
26220	trip-153861118270144424+IND583119AAA+IND583101AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Sandur_WrdN1D (Karna
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (Karna

26222 rows × 20 columns



◆ Understanding:

- || The rows have been merged based on the unique segment_key, which is a combination of trip_uuid, source_center, and destination_center.
 - || The aggregated dataset reflects the total values for each segment of the trip.
-

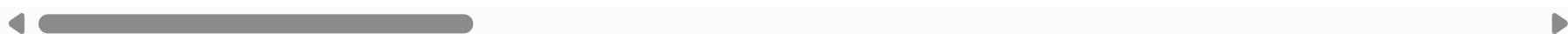
📁 Feature Engineering

```
In [60]: # 1. Calculating time difference between od_start_time and od_end_time  
segment_agg_data['od_total_time']=(segment_agg_data['od_end_time'] - segment_agg_data['od_start_time'])  
segment_agg_data['od_time_diff_hour'] = (segment_agg_data['od_total_time']).dt.total_seconds()/3600  
segment_agg_data
```

Out[60]:

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_i
0	trip-153671041653548748+IND209304AAA+IND000000ACB	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Centra (Uttar Pra
1	trip-153671041653548748+IND462022AAA+IND209304AAA	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Bhopal_Trnsp (Madhya Pra
2	trip-153671042288605164+IND561203AAB+IND562101AAA	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaD (Karna
3	trip-153671042288605164+IND572101AAA+IND561203AAB	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Tumkur_Veers (Karna
4	trip-153671043369099517+IND00000ACB+IND160002AAC	trip-153671043369099517	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur (Har
...
26217	trip-153861115439069069+IND628204AAA+IND627657AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Tirchchndr_Shnmgp (Tamil N
26218	trip-153861115439069069+IND628613AAA+IND627005AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Peikulam_SriVnkt (Tamil N
26219	trip-153861115439069069+IND628801AAA+IND628204AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Eral_Busstand_D (N
26220	trip-153861118270144424+IND583119AAA+IND583101AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Sandur_WrdN1D (Karna
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (Karna

26222 rows × 22 columns



In [61]:

segment_agg_data.sample()

Out[61]:

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_na
4965	trip-153704666810685062+IND173025AAA+IND160002AAC	trip-153704666810685062	training	FTL	2018-09-15 21:24:28.108007	PaontSahib_Gurudwa (Himachal Prade



```
In [63]: # de = segment_agg_data.drop(columns=['source_city', 'source_state', 'source_place', 'destination_place', 'destination_city', 'dest
de = segment_agg_data.copy()
```

In [64]: `de.sample()`

Out[64]:

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_na
4962	trip-153704666399260818+IND421302AAG+IND401104AAA	trip-153704666399260818	training	Carting	2018-09-15 21:24:23.992906	Bhiwandi_Mankoli_HF (Maharashtra



```
In [ ]: # # could have done this --- but some major error ... check it....
# sad = segment_agg_data.copy()
# sad["source_city"] = sad["source_name"].str.split(" ",n=1,expand=True)[0].str.split("_",n=1,expand=True)[0]
# sad["source_state"] = sad["source_name"].str.split(" ",n=1,expand=True)[1].str.replace("(, "").str.replace(")", "")
# sad["destination_city"] = sad["destination_name"].str.split(" ",n=1,expand=True)[0].str.split("_",n=1,expand=True)[0]
# sad["destination_state"] = sad["destination_name"].str.split(" ",n=1,expand=True)[1].str.replace("(, "").str.replace(")", "")

# sad["source_place"] = sad["source_name"].str.split("_",n=2,expand=True)[1]
# sad["destination_place"] = sad["destination_name"].str.split("_",n=2,expand=True)[1]
```

In [65]: `# using regex pattern to separate the city, place, state`

```
def extract_info(name):
    pattern = r'^^(?P<city>[\^s_]+)_?(?P<place>[^\\(\\)]*)\\s?\\((?P<state>[A-Za-z\\s&]+)\\)$'
    match = re.match(pattern, name)
    if match:
        city = match.group('city').strip()
        place = match.group('place').strip() if match.group('place') else city
        state = match.group('state').strip()
        return city, place, state
```

```
    else:  
        return None, None, None
```

```
In [66]: de[['source_city', 'source_place', 'source_state']] = de['source_name'].apply(lambda x: pd.Series(extract_info(x)))
```

```
In [67]: de[['destination_city', 'destination_place', 'destination_state']] = de['destination_name'].apply(lambda x: pd.Series(extract_
```

```
In [68]: de
```

Out[68]:

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_i
0	trip-153671041653548748+IND209304AAA+IND000000ACB	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Centra (Uttar Pra
1	trip-153671041653548748+IND462022AAA+IND209304AAA	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Bhopal_Trnsp (Madhya Pra
2	trip-153671042288605164+IND561203AAB+IND562101AAA	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaD (Karna
3	trip-153671042288605164+IND572101AAA+IND561203AAB	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Tumkur_Veers (Karna
4	trip-153671043369099517+IND00000ACB+IND160002AAC	trip-153671043369099517	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur (Har
...
26217	trip-153861115439069069+IND628204AAA+IND627657AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Tirchchndr_Shnmgp (Tamil N
26218	trip-153861115439069069+IND628613AAA+IND627005AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Peikulam_SriVnkt (Tamil N
26219	trip-153861115439069069+IND628801AAA+IND628204AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Eral_Busstand_D (N
26220	trip-153861118270144424+IND583119AAA+IND583101AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Sandur_WrdN1D (Karna
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (Karna

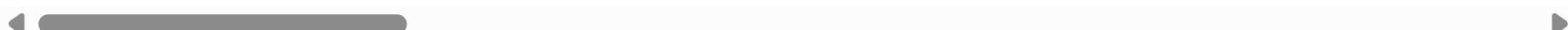
26222 rows × 28 columns

In [69]: `de[(de['source_place']=='') | (de['destination_place']=='')]`

Out[69]:

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_name
7	trip-153671052974046625+IND583101AAA+IND583201AAA	trip-153671052974046625	training	FTL	2018-09-12 00:02:09.740725	Bellary_D (Karnataka)
9	trip-153671052974046625+IND583201AAA+IND583119AAA	trip-153671052974046625	training	FTL	2018-09-12 00:02:09.740725	Hospet (Karnataka)
19	trip-153671110078355292+IND121004AAB+IND121001AAA	trip-153671110078355292	training	Carting	2018-09-12 00:11:40.783923	FBD_Balabgarh_DP (Haryana)
33	trip-153671173668736946+IND110043AAA+IND110078AAA	trip-153671173668736946	training	Carting	2018-09-12 00:22:16.687619	Delhi_Nangli_I (Delhi)
80	trip-153671320807895983+IND121004AAB+IND121102AAA	trip-153671320807895983	training	Carting	2018-09-12 00:46:48.079257	FBD_Balabgarh_DP (Haryana)
...
26118	trip-153860849934816308+IND110078AAA+IND110043AAA	trip-153860849934816308	test	Carting	2018-10-03 23:14:59.348414	Janakpuri (Delhi)
26153	trip-153860958923357924+IND842003AAB+IND482002AAA	trip-153860958923357924	test	Carting	2018-10-03 23:33:09.233829	Jabalpur_Adhartal_I (Madhya Pradesh)
26180	trip-153861007249500192+IND842001AAA+IND846004AAA	trip-153861007249500192	test	FTL	2018-10-03 23:41:12.495257	Muzaffarpur_Bbganj (Bihar)
26181	trip-153861007249500192+IND846004AAA+IND847103AAA	trip-153861007249500192	test	FTL	2018-10-03 23:41:12.495257	Darbhanga (Bihar)
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (Karnataka)

782 rows × 28 columns



In [70]:

```
de.loc[de['source_place']=='','source_place']=de['source_city']
de.loc[de['destination_place']=='','destination_place']=de['destination_city']
```

In [71]:

```
de[de.source_place.isna()]
```

```
Out[71]: segment_key  trip_uuid  data  route_type  trip_creation_time  source_name  destination_name  od_start_time  od_end_time  start_scan_to_
```



```
In [72]: de.isna().sum()
```

```
Out[72]: segment_key          0  
trip_uuid            0  
data                0  
route_type           0  
trip_creation_time   0  
source_name          0  
destination_name     0  
od_start_time        0  
od_end_time          0  
start_scan_to_end_scan 0  
actual_distance_to_destination 0  
actual_time          0  
osrm_time            0  
osrm_distance        0  
segment_actual_time   0  
segment_osrm_time     0  
segment_osrm_distance 0  
segment_actual_time_sum 0  
segment_osrm_time_sum 0  
segment_osrm_distance_sum 0  
od_total_time         0  
od_time_diff_hour    0  
source_city           0  
source_place          0  
source_state          0  
destination_city       0  
destination_place      0  
destination_state      0  
dtype: int64
```

```
In [73]: #de = de.drop(columns=['source_city', 'source_state', 'source_place', 'destination_city', 'destination_place', 'destination_state'])
```

```
In [74]: de.loc[de.source_city=='Bangalore', 'source_city']='Bengaluru'
```

```
de.loc[de.destination_city=='Bangalore','destination_city']='Bengaluru'
```

```
In [75]: np.set_printoptions(threshold=np.inf)
```

```
In [76]: de['source_city'].unique()
```

```
Out[76]: array(['Kanpur', 'Bhopal', 'Doddablpur', 'Tumkur', 'Gurgaon', 'Bengaluru',
   'Mumbai', 'Bellary', 'Sandur', 'Hospet', 'Chennai', 'HBR', 'Surat',
   'Delhi', 'Pune', 'FBD', 'Shirala', 'Ratnagiri', 'Kolhapur',
   'Hyderabad', 'Anantapur', 'Thirumalagiri', 'Gulbarga', 'Aland',
   'Sindagi', 'Indi', 'Jaipur', 'Allahabad', 'Guwahati', 'Unnao',
   'Narsinghpur', 'Gadarwara', 'Shrirampur', 'Nashik', 'Sinnar',
   'Sangamner', 'Shirdi', 'Kopargaon', 'Vaijiapur', 'Hoogly',
   'Hooghly', 'Kolkata', 'Madakasira', 'Pavagada', 'Sonari',
   'Medchal', 'Dindigul', 'Kodaikanal', 'Batlagundu', 'Palani',
   'Oddnchtram', 'Jalandhar', 'Nakodar', 'Kapurthala', 'Faridabad',
   'Chandigarh', 'Deoli', 'Pandharpur', 'Atapadi', 'CCU', 'Bhandara',
   'Kurnool', 'Palwal', 'Bhiwandi', 'Bhatinda', 'TalwandiSabo',
   'Mansa', 'Jhunir', 'RoopNagar', 'AnandprShb', 'Bantwal', 'Kadaba',
   'Sullia', 'Chittapur', 'Sedam', 'Chincholi', 'Lalru', 'Kadi',
   'Mehsana', 'Shahdol', 'Dola', 'Gangakher', 'Parli', 'Ambajogai',
   'Nanded', 'Loha', 'Durgapur', 'Bankura', 'Barjora', 'Vapi',
   'Jamjodhpur', 'Porbandar', 'Junagadh', 'Jetpur', 'Dhoraji',
   'Khammam', 'Nalgonda', 'Miryalguda', 'Suryapet', 'Choutuppal',
   'Vijayawada', 'Vadnagar', 'Palanpur', 'Deesa', 'Jabalpur',
   'Talala', 'Veraval', 'Una', 'Kodinar', 'Gundlupet', 'Tirumakudalu',
   'Chamarjngr', 'Malavalli', 'Kollegal', 'Mysore', 'Hunsur',
   'HDKote', 'Gonikoppal', 'Patan', 'Bhabhar', 'Vizag', 'Rajamundry',
   'Goa', 'Sawantwadi', 'Kankavali', 'Sonipat', 'Moradabad',
   'Rudrapur', 'Himmatnagar', 'Khedbrahma', 'Modasa', 'Jamshedpur',
   'Pondicherry', 'Cuddalore', 'Chidambaram', 'Sirkazhi', 'Karaikal',
   'Thiruvarur', 'Nagapttinm', 'MAA', 'Anand', 'Khambhat', 'Udgir',
   'Latur', 'Degloor', 'Nadiad', 'Umreth', 'Villupuram', 'Virudhchlm',
   'Pennadam', 'Chinnasalem', 'Panruti', 'Neyveli', 'Purulia', 'Hura',
   'Rghunthpur', 'Jhalda', 'Bhubaneshwar', 'Bamangola', 'Meham',
   'Tiruppattur', 'Ambur', 'Kotdwara', 'Haridwar', 'Medak',
   'Narsapur', 'Kamareddy', 'Yellareddy', 'Bodhan', 'Banswada',
   'Dhrangadhra', 'Halvd', 'Gandhidham', 'Gangavathi', 'Koppal',
   'Ghumarwin', 'JognderNgr', 'Jahu', 'ChandroknaRD', 'Kharagpur',
   'AmaDub', 'Agra', 'Sitapur', 'Biswan', 'Lakhimpur', 'Gola',
   'Canacona', 'Bilimora', 'SultnBthry', 'Lucknow', 'Vellore', 'Bhuj',
   'Anjar', 'Dinhata', 'BOM', 'Margherita', 'Boisar', 'Dahanu',
   'Chodavaram', 'Tezpur', 'Bomdila', 'Koduru', 'Rajampet',
   'Tirupati', 'Puttur', 'Srikalahsti', 'Gudur', 'Venktagiri', 'Pen',
   'Roha', 'Mahad', 'AMD', 'Ahmedabad', 'Faizabad', 'Gosainganj',
   'Gandhinagar', 'Muzaffrpur', 'Phagwara', 'Betul', 'Itarsi',
```

'Pandhurna', 'Panskura', 'Haldia', 'Tamluk', 'Karur', 'BLR',
'Rasipurm', 'Sankari', 'Jorhat', 'Bokakhat', 'PNQ', 'Aligarh',
'Mainpuri', 'Shikohabad', 'Firozabad', 'Srikakulam', 'Rajam',
'Palakonda', 'Parvathipuram', 'Bobbili', 'Tekkali', 'Palasa',
'Narasnpeta', 'Paralakhemundi', 'Dehradun', 'Hajo', 'NOI',
'Jassur', 'Dalhousie', 'Chamba', 'Baharampur', 'Dhulian',
'Hoskote', 'Shajapur', 'Shujalpur', 'Pachore', 'OK', 'Ludhiana',
'GreaterThane', 'Janakpuri', 'Amritsar', 'Tirupur', 'Attur',
'Salem', 'Tirchngode', 'Darjeeling', 'Mirik', 'Tiruchi', 'Noida',
'Rangia', 'Dhubri', 'Bilasipara', 'Kokrajhar', 'Tura', 'Ajmer',
'Pali', 'Jodhpur', 'Amroha', 'Gajraula', 'Rampur', 'Jhansi',
'Dholpur', 'Gwalior', 'Thirthurpondi', 'Pushpavanam', 'Ranchi',
'Guna', 'Ashokngr', 'Raver', 'Hassan', 'Jairampur', 'Chamoli',
'Karnaprayag', 'Gohpur', 'Dhemaji', 'Silapathar', 'Gopiganj',
'Varanasi', 'Dharmapuri', 'Hubli', 'Bailhongal', 'Gokak', 'Gadag',
'Rona', 'Bagalkot', 'Ramdurg', 'Duddhi', 'Renukoot', 'Anpara',
'Singrauli', 'Panipat', 'Sasaram', 'Tangi', 'Davangere',
'Ranebennur', 'Vrindavan', 'Chittaurgarh', 'Rajsamand', 'Solapur',
'Pratapgarh', 'Del', 'Moga', 'Muktsar', 'Jagatsghpr', 'Paradip',
'Mangalore', 'Barshi', 'Osmanabad', 'Vinukonda', 'Ongole',
'Kanigiri', 'Kavali', 'Achrol', 'LowerParel', 'Sagara', 'Sirsi',
'Deoband', 'Muzaffrngr', 'Tikamgarh', 'Satna', 'Ghaziabad',
'Chhapra', 'BiharSarif', 'Rajgir', 'Jehanabad', 'Nawada',
'Pallakad', 'Vadakkencherry', 'Thrissur', 'Aluva', 'Kanakapura',
'Ramanagara', 'Mandy', 'Srirangapatna', 'Roorkee', 'Rishikesh',
'Aurangabad', 'Surathkal', 'Vadodara', 'Barh', 'Sheikhpora',
'Godhra', 'Coimbatore', 'Bhadراك', 'Karanjia', 'Kendujhar', 'Joda',
'Narnaul', 'Rewari', 'Bhiwadi', 'Hisar', 'Ganga', 'Hanumangarh',
'Bihta', 'Arrah', 'Jhajjar', 'Silchar', 'Kabuganj', 'Bardhaman',
'Asansol', 'Rupnarayanpur', 'Midnapore', 'Sillod', 'Jalna',
'Nellore', 'Katwa', 'Chapra', 'Nakashipara', 'Plassey',
'Thamarassery', 'Safidon', 'Kaithal', 'Kurukshtetra', 'Pehowa',
'Udaipur', 'Kaikaluru', 'Eluru', 'Machilipatnam', 'Gudivada',
'Nazirpur', 'Nowda', 'Domkal', 'Devarakonda', 'Haliya',
'Kalwakurthy', 'Kadthal', 'Vikarabad', 'Rampurhat', 'Morgram',
'Kandi', 'Khargram', 'Ghanashyampur', 'Visakhapatnam', 'Lalgola',
'Murshidabad', 'Jangipur', 'Ragunthgnj', 'Sagardighi',
'AhmedNagar', 'Ashti', 'Parner', 'Bilaspur', 'Pilibhit',
'Hailakandi', 'Badarpur', 'Teok', 'BilaspurHP', 'Mandi',
'SundarNgr', 'Kakinada', 'Chitradurga', 'Hiriyur', 'Amalapuram',
'Madhubani', 'Jaynagar', 'Kalka', 'Solan', 'Nalagarh', 'Buldhana',

'Akola', 'Shegaon', 'Karad', 'Islampur', 'Chiplun', 'JoguGadwal',
'Raichur', 'Modinagar', 'Meerut', 'Madhepura', 'Supaul',
'Triveninganj', 'Saharsa', 'Purnia', 'Simrahi', 'Araria',
'Narpatganj', 'Sitamari', 'Sheohar', 'Benipatti', 'Atmakur',
'Bngnpalle', 'Nandyal', 'Dandeli', 'Mallapur', 'Chikodi', 'Athani',
'Rohtak', 'Jind', 'Gohana', 'Patiala', 'Sunam', 'Nasirabad',
'Asind', 'Darbhanga', 'Benipur', 'Jhanjharpur', 'Channaraya',
'Naugchia', 'Katihar', 'Puttaprthi', 'Kadiri', 'Dharmavram',
'Rayachoti', 'Pulivendula', 'Ambala', 'YamunaNagar', 'PaontSahib',
'Kishangarh', 'Degana', 'Makrana', 'Merta', 'Parbatsar', 'Korba',
'Pithorgarh', 'Nainital', 'Almora', 'Ranikhet', 'Deoghar', 'Alwar',
'Bharatpur', 'Gorakhpur', 'Deoria', 'Kaptanganj', 'Padrauna',
'Salempur', 'Dhaka', 'Bettiah', 'Narktiganj', 'Bhatpara',
'Mungeli', 'Dumka', 'Jamtara', 'Pakur', 'Bangana', 'Sujanpur',
'Nadaun', 'Bahadurgarh', 'Kanth', 'Thakurdwara', 'Bijnor',
'Dhampur', 'Najibabad', 'Nichlaul', 'Warangal', 'Khanpur',
'Rayaparthi', 'Aonla', 'Sambhal', 'Chandausi', 'Dhar', 'Bagnan',
'Kolaghat', 'Ghatal', 'Naraingarh', 'Kashipur', 'Ramnagar',
'Ratanpura', 'Bariya', 'Sikandarpur', 'Gondia', 'Motihari',
'Zahirabad', 'Narayankhed', 'Humnabad', 'Bidar', 'Gooty',
'Kalyandurg', 'Guntakal', 'Rayadurgam', 'Samana', 'Patran',
'Bhadrachalam', 'Manuguru', 'Sathupally', 'Baraut', 'Bassi',
'Dausa', 'Lalsot', 'Gangapur', 'Hindaun', 'Karauli', 'Sikar',
'Nawalgarh', 'Jamnagar', 'Khambhalia', 'Bhatiya', 'Kakdwip',
'Fatepur', 'Amravati', 'Kalluvathukal', 'Attingal',
'Kazhakkottam', 'Surendranagar', 'Limbdhi', 'Chotila', 'Dudu',
'Phulera', 'Renwal', 'Buxar', 'Dumraon', 'DalsinghSarai', 'Rusera',
'Manjhaul', 'Khurja', 'Jewar', 'Anupshahar', 'Pahasu',
'Trivandrum', 'Neyatinkra', 'Etawah', 'Auraiya', 'Jasai',
'Bhagalpur', 'Banka', 'Panaji', 'Raikot', 'Jagraon', 'GZB',
'Hapur', 'Pilkhuwa', 'Dholi', 'Machhiwara', 'Chhata', 'Luxettipet',
'Vemulawada', 'Metpally', 'Jagtial', 'Mancherial', 'Karimnagar',
'Kottayam', 'Tiruvalla', 'Parakkdavu', 'Khurdha', 'Rajpura',
'SirhindFatehgarh', 'Khanna', 'Pthnmthitt', 'Kollam', 'Punalur',
'Kottarakkara', 'Adoor', 'Dhule', 'Sakri', 'Shindkheda', 'Shahada',
'Nandurbar', 'Dehrisone', 'Brahmapuri', 'Gadchiroli', 'Ramagundam',
'Gomoh', 'Chalisgaon', 'Wardha', 'Barnala', 'Hindupur',
'Ghatampur', 'Upleta', 'Mangrol', 'Banda', 'Fatehpur', 'Akbarpur',
'Pukhrayan', 'Orai', 'Bhanvad', 'Mejia', 'Basti', 'Dibrugarh',
'Khonsa', 'Digboi', 'Mussoorie', 'Kalpetta', 'Phalodi', 'Barmer',
'Pokhran', 'Gangarampr', 'Changlang', 'Sidhi', 'Churhat', 'Vansda',

'Bishnupur', 'Guskhara', 'Panagarh', 'Bolpur', 'Nanoor',
'Mainaguri', 'Mathabhang', 'Jammu', 'Kathua', 'Samba', 'Bhusawal',
'Burhanpur', 'Narsinghgarh', 'Hyd', 'Nalbari', 'Sendhwa',
'Sholinghur', 'Ranipet', 'Kalpakkam', 'Talegaon', 'Rajgurunagar',
'Kittur', 'SrinagarUK', 'Berhampur', 'Bishwanath', 'LakhimpurN',
'Likabali', 'Shillong', 'Shimoga', 'Saundatti', 'Chomu', 'Gonda',
'Balrampur', 'Tulsipur', 'Manapparai', 'Melur', 'Kandukur',
'Ghazipur', 'Paota', 'Kotputli', 'Masaurhi', 'Guruvayoor',
'Chetpet', 'Polur', 'Arani', 'Wai', 'Satara', 'Khandala',
'Bhiwani', 'Udupi', 'Kundapura', 'Ankola', 'Bhatkal', 'Honnavar',
'Kumta', 'Karkala', 'Moodbidri', 'GGN', 'Patancheru', 'Alappuzha',
'Haripad', 'Kozhikode', 'Vadakara', 'Koyilandy', 'Kallachi',
'Kumbakonam', 'Mylduthuri', 'Mannargudi', 'Rameswram',
'Muthukulathur', 'Narikkudi', 'Shirur', 'Ooty', 'Mahabubnagar',
'Bijnapally', 'Jadcherla', 'Pattukotai', 'Aranthangi',
'Peravurani', 'Ponnamaravathi', 'Karaikudi', 'Karimganj',
'Srisailam', 'Lalpet', 'Thuraiyur', 'Musiri', 'Perambalur',
'Ariyalur', 'Jayamkondan', 'Karwar', 'Parwanoo', 'Forbesganj',
'Raniganj', 'Madurai', 'Rajpalayam', 'Sivakasi', 'Maharajapuram',
'Sankaran', 'Sathyamangalam', 'Gobicheti', 'Anthiyour', 'Cjb',
'Ratlam', 'Jaora', 'Nabha', 'Usilampatti', 'Kattappana', 'Khurai',
'Mungaoli', 'Nuzvid', 'Vuyyuru', 'Koppa', 'Krishnarajpet',
'Nagamangala', 'Ramanathapura', 'Madikeri', 'Khatauli', 'Tiptur',
'Arsikere', 'Draksharamam', 'Weir', 'Kherli', 'Kolar',
'Chintamani', 'Mulbagal', 'Bethamangala', 'Bangarapet', 'Pollachi',
'Udumalpet', 'Haveri', 'NeemKaThana', 'Baheri', 'Raxaul', 'Chakia',
'Dharapuram', 'Kangayam', 'Tirpur', 'Mohania', 'Naugarh',
'Arakkonam', 'Kanchipuram', 'Kallikkad', 'Malemruvathur',
'Mahbubabad', 'Yellandu', 'Tirunelveli', 'Vallior', 'Marthandam',
'Bhavnagar', 'Amreli', 'Thisayanvilai', 'Tirchchndr', 'Peikulam',
'Eral', 'Kusumnchi', 'Kodad', 'Godda', 'Dhanbad', 'Mahadevpur',
'Bhupalpally', 'Mulugu', 'Jammikunta', 'Narsingpur', 'Amd',
'Manthani', 'SikandraRao', 'Etah', 'Jalesar', 'Mahoba', 'Bhind',
'Valsad', 'Navsari', 'Bagepalli', 'Chikblapur', 'Chhaygaon',
'Manamelkudi', 'Oriyur', 'Thiruvadanai', 'Sivaganga', 'Kanti',
'Saraiya', 'Malegaon', 'Manmad', 'Sindhanur', 'Murbad', 'Asangaon',
'Bhuvanagiri', 'Kanker', 'Unjha', 'Jaleswar', 'Malda', 'Balipara',
'Pathankot', 'Bhubaneswar', 'Sehore', 'Ashta', 'Erode', 'Gotan',
'Jalgaon', 'Silvassa', 'Mau', 'Balasore', 'Udala', 'Nagaur',
'Tarnau', 'Bhilwara', 'Ghanpur', 'Jangaon', 'Hazaribag',
'JhumriTlya', 'Dharwad', 'Chhatarpur', 'Panna', 'Dahod',

'Bareilly', 'Faridpur', 'Mandapeta', 'Tuni', 'Jeypore', 'Koraput',
'Wanaparthy', 'Ramnathpuram', 'Paramakudi', 'Kekri', 'Bijainagar',
'Beawar', 'Bilara', 'Sankaramangalam', 'Shoranur', 'Arimbur',
'Pazhayannur', 'Chikhli', 'Mehkar', 'Khamgaon', 'Ratia', 'Cumbum',
'Theni', 'Sakleshpur', 'Mudigere', 'Kotagiri', 'HazratJandaha',
'Samastipur', 'Nawanshahr', 'Bina', 'Ukkadagatri', 'Malur',
'Lonavala', 'Avinashi', 'Jhunjhunu', 'Khetri', 'AurangabadBR',
'Ambah', 'Chandpur', 'Madhupur', 'Sahebganj', 'Shirpur', 'Dadri',
'Husnabad', 'Kaman', 'Bewar', 'Manvi', 'Raipur', 'Bellampalli',
'Kaghaznagar', 'Chinnur', 'Harraiya', 'Bareli', 'Nanjangud',
'Mothkur', 'Chhindwara', 'Lakhnadon', 'Hansi', 'Salur', 'Phulpur',
'Soraon', 'Mananthavady', 'Maihar', 'Rewa', 'Srivijaynagar',
'Suratgarh', 'Thoppur', 'Bongaigaon', 'Barpeta', 'Chopan',
'Addanki', 'Chimkurthy', 'Markapur', 'Bikaner', 'Didwana',
'Challakere', 'Molakalmuru', 'Chrkhidri', 'Blr', 'Sirsa',
'Nandigama', 'Dhone', 'Bhadra', 'Tirurangadi', 'Malappuram',
'Baddi', 'Mariyam', 'Pupri', 'Sonepur', 'Karnal', 'Malerkotla',
'Giridih', 'Sahatwar', 'Sagar', 'Kushinagar', 'Pangodu', 'Shamli',
'Katni', 'Barhi', 'Rath', 'Konch', 'Hamirpur', 'Umaria', 'Koratla',
'Sitarganj', 'Anakapalle', 'Narsipatnam', 'Akhnoor', 'MirzapurWB',
'Golaghat', 'Sivasagar', 'Pasighat', 'Marakkanam', 'Dohrighat',
'Ghosi', 'Dhekiajuli', 'Mangaldoi', 'Reengus', 'Nokha', 'Bokaro',
'Peterbar', 'Ramgarh', 'Tandur', 'Shadnagar', 'Badnaur', 'Seoni',
'Mandla', 'Dharamshala', 'DehraGopipur', 'Amb', 'Kilimanoor',
'Nabarangpr', 'Palibirsighpr', 'Pilani', 'Jagdishpur', 'Dwarka',
'Ambasamdrum', 'Tenkasi', 'Kolasib', 'Haldwani', 'Bheemunipatnam',
'Vizianagaram', 'Kanhagad', 'Manjeshwar', 'Anandnagar',
'Maharajganj', 'RaisingNgr', 'Anupgarh', 'Balotra', 'Jaisalmer',
'Vidisha', 'GanjBasoda', 'Malvan', 'Tezu', 'Piparcity', 'Lakhipur',
'Jaunpur', 'Banswara', 'Sujangarh', 'Kunnathund', 'Belgaum',
'Khanapur', 'Chalakudy', 'Irinjlkuda', 'Angamaly', 'Jath',
'Bijapur', 'Coonoor', 'Chikmagalur', 'Palampur', 'Fatehabad',
'Barwala', 'Bhadaur', 'Tinusukia', 'Chabua', 'DoomDooma', 'Namsai',
'Dalkhola', 'Dharmanagr', 'Airport', 'MughalSarai', 'Manthuka',
'Kayamkulam', 'Aizawl', 'Champhai', 'Tirur', 'Cochin', 'Uchila',
'Shevgaon', 'Paithan', 'Beed', 'Georai', 'Nandikotkr',
'Betamcherla', 'Nilambur', 'HanumanJNC', 'Bikramgang', 'Dinara',
'Itahar', 'Kaliyaganj', 'Manikchak', 'Sihora', 'Rajkot',
'Bulndshahr', 'Chaksu', 'Majalgaon', 'Mandsaur', 'Sitamau',
'Kirauli', 'Roing', 'Krishnagiri', 'Pachora', 'Erandol', 'Amalner',
'Pataudi', 'Naharlagun', 'Bhadohi', 'Cherthala', 'Madnapalle',

```
'Mokokchung', 'Pappadahandi', 'Lalitpur', 'Farrukhbad', 'Shahabad',
'Jalalabad', 'Tilhar', 'Garhshnker', 'Hajipur', 'Paranpur',
'Gauribidanur', 'Wankaner', 'Morbi', 'BariSadri', 'Neemuch',
'Sangareddy', 'Islampure', 'Nakhatrana', 'Chandauli', 'Dhrmsthal',
'Jowai', 'Chiraiyakot', 'Jiyanpur', 'Siwan', 'Rawatsar', 'Loharu',
'Kozhenchery', 'Printhlmna', 'Aliganj', 'Chhibramau', 'Taranagar',
'Khed', 'Mundakayam', 'Nimbahera', 'Sangola', 'Asifabad',
'Arambag', 'Berhampore', 'Manbazar', 'Shahganj', 'Azamgarh',
'Saraimeer', 'Neemrana', 'Barbil', 'Razole', 'Sultana', 'Budhana',
'Raiganj', 'Radhanpur', 'Moranhat', 'Khatra', 'Simlapal',
'Ghatkesar', 'Betnoti', 'Bongaon', 'Nedumangad', 'Karukachal',
'Soro', 'Kamarpukur', 'Keshiary', 'Contai', 'Manglamaro',
'Puranpur', 'Palakollu', 'Falna', 'Thirthahalli', 'Tarkeshwar',
'Hathras', 'Nipani', 'Kulithalai', 'Brajrajanagar', 'Sundargarh',
'Sambalpur', 'Jalore', 'Bhinmal', 'Bhilad', 'Pilibanga',
'Khanakul', 'Kothanalloor', 'Nirjuli', 'Sidhmukh', 'Kullu',
'Manteswar', 'Ranaghat', 'Gudalur', 'Kasganj', 'Helencha',
'Dabhoi', 'Balangir', 'Bargarh', 'Sirohi', 'Gondal', 'Morena',
'Nagpur', 'Baripada', 'Palitana', 'Samsi', 'Sumerpur', 'Vadipatti',
'Giddarbaha', 'Fazilka', 'Phusro', 'Tonk', 'SawaiMadhopur',
'Munger', 'Pauri', 'Sankerko', 'Nohar', 'Krishnanagar',
'Kuthuparamba', 'KharagpurBR', 'Jamui', 'DhrmpuriTS', 'Anuppur',
'Baruipur', 'Tirtol', 'Daurala', 'Jharsuguda', 'Central', 'Abohar',
'RampuraPhul', 'Hoshangabad', 'Cuttack', 'Kendrapara', 'Rajgarh',
'Durg', 'Balurghat', 'Chamorshi', 'Barasat', 'Gujiliam', 'Basta',
'Shahjhnpur', 'Tadipatri', 'Cuddapah', 'Badvel', 'Proddatur',
'Angul', 'Phulbani', 'Ambegaon', 'Chanchal', 'Malout',
'Uthangarai', 'Badlapur', 'Balugaon', 'Mahasamund', 'Badkulla',
'Kapadvanj'], dtype=object)
```

```
In [77]: de['source_place'].unique()
```

```
Out[77]: array(['Central_H_6', 'Trnsport_H', 'ChikaDPP_D', 'Veersagr_I',  
   'Bilaspur_HB', 'Nelmgla_H', 'Hub', 'Dc', 'WrdN1DPP_D', 'Hospet',  
   'Poonamallee', 'Porur_DPC', 'Chrompet_DPC', 'Layout PC',  
   'Bagaluru_D', 'Central_D_12', 'Central_I_4', 'Lajpat_IP',  
   'North_D_3', 'Balabgarh_DPC', 'Central_DPP_3', 'MjgaonRd_D',  
   'Shivaji_I', 'Shamshbd_H', 'KamaStrt_I', 'Xroad_D', 'Nehrugnj_I',  
   'RazaviRd_D', 'KalyanNg_D', 'SindgiRD_D', 'Central_I_7',  
   'VarunCly_DC', 'Central_H_1', 'Nangli_IP', 'North', 'VikasRam_D',  
   'KndliDPP_D', 'MPward_D', 'Central_D_9', 'DavkharRd_D',  
   'TgrniaRD_I', 'Central_D_1', 'JawanChk_D', 'SaiBansi_D',  
   'NkshtrPz_D', 'YeolaRD_D', 'Bandel_D', 'DC', 'North_I_4',  
   'RTCStand_D', 'PnukndRD_D', 'Central_DPP_1', 'KGAirprt_HB',  
   'North_D_2', 'MROoffce_D', 'Athithnr_DC', 'RTOofice_D',  
   'RjndraRd_D', 'Palani_D', 'DPC', 'ChowkDPP_D', 'Mthurard_L',  
   'Mullanpr_DC', 'Mehmdpur_H', 'Mohali', 'Central_DPP_2',  
   'RajCmplx_D', 'Vidyangr_D', 'Beliaghata_DPC', 'RjnaiDPP_D',  
   'KaremDPP_D', 'AbbasNgr_I', 'Palwal', 'Mankoli_HB', 'Wardno3_D',  
   'GreenVly_D', 'BhaRDDPP_D', 'Airport_H', 'Gateway_HB',  
   'Tathawde_H', 'ChotiHvl_DC', 'PnjPiara_D', 'Kharar_DC',  
   'Trmltmpl_D', 'AnugrDPP_D', 'Srirampt_D', 'GlbrgaRD_D',  
   'DBRCmplx_D', 'Mainroad_D', 'OnkarDPP_D', 'KaranNGR_D',  
   'Panchot_IP', 'Sohagpur_D', 'ChainDPP_D', 'Chrompet_L',  
   'Busstand_D', 'JingeNgr_D', 'BnsllNgr_D', 'Aswningr_I',  
   'SivjiCWK_D', 'Central_I_1', 'KeranDPP_D', 'IndEstat_I', 'Court_D',  
   'JmnvadRd_DC', 'NSTRoad_I', 'HydRoad_DC', 'Ragvendr_D',  
   'Krishna_D', 'Nagaram_D', 'Rynapadu_H', 'BsstdDPP_D',  
   'HawaiPlr_DC', 'Adhartal_IP', 'DumDum_DPC', 'Bomsndra_HB',  
   'SsnRdDPP_D', 'Mamlatdr_DC', 'NCplxDPP_D', 'Swamylt_D',  
   'Narasipr_D', 'BrDbleRd_D', 'SulthnRd_D', 'NarsipuraRd_D',  
   'Yadvgiri_IP', 'RathaDPP_D', 'BegurRD_D', 'Thomas_D',  
   'Central_D_2', 'TirupDPP_D', 'Gajuwaka_L', 'AtoNgrRd_I',  
   'LaxmiNgr_D', 'NrdawDPP_D', 'Old City', 'Kundli_H', 'UdhamNgr_H',  
   'Patelfli_D', 'Central_I_3', 'Vasanthm_I', 'KtsiGrsm_D',  
   'ARBNorth_DC', 'Pngktgudi_D', 'Thalthr_dc', 'Bypasrd_D',  
   'Sttyapar_D', 'Poonamallee_HB', 'VUNagar_DC', 'MotvdDPP_D',  
   'NlgaonRd_D', 'Srwnsngr_D', 'Dakor_DC', 'Vaghasi_IP', 'Bnnrghta_L',  
   'Thirumtr_IP', 'SelamRd_D', 'EastmnRD_D', 'VketRoad_D',  
   'GandhiRd_D', 'NJVNgr_D', 'GariDPP_D', 'barkarRd_D', 'GMukrDPP_D',  
   'MP Nagar', 'Central_D_3', 'Jogshwri_I', 'GModDPP_D', 'KoilStrt_D',  
   'CotnGren_M', 'Nzbadrd_D', 'Haridwar', 'Dwaraka_D', 'Devenply_I',
```

'JKRoad_D', 'SuryaDPP_D', 'Menagrdn_D', 'NvygRDPP_D', 'CrossRD_D',
'Sector1A_IP', 'PhrmPlza_D', 'Banikatt_D', 'Gndhichk_D', 'Dhelu_D',
'Sulgwan_D', 'Bulabeda_D', 'Chowk_D', 'PatelNGR_D', 'Maheva_D',
'BkgnRoad_D', 'CharRsta_D', 'Kollgpra_D', 'Peenya_IP',
'GndhiNgr_IP', 'Sanpada_I', 'WrdN4DPP_D', 'Sakinaka_RP',
'CivilHPL_D', 'OstwlEmp_D', 'KetyDPP_D', 'Gajuwaka', 'BaljiDPP_D',
'Mhbhirab_D', 'CTRoad_D', 'MGRoad_D', 'RSRoad_D', 'Balajicly_I',
'Artmclny_D', 'SDKNgr_D', 'Bngisheb_D', 'TirupthiRd_D',
'BljiMrkt_D', 'DataSagr_D', 'Govndsgsgr_D', 'Dankuni_HB', 'Rakhial',
'East_H_1', 'Memnagar', 'East_I_21', 'Mithakal_D', 'TrnspNgr_D',
'Pakrela_D', 'Bbganj_I', 'Bilaspur_RP', 'Lovely_D', 'PatelWrd_D',
'DivrsnRd_D', 'Mataward_D', 'CottonGreen_DPC', 'Pawane_L', 'Karur',
'JPNagar_Pc', 'Knrpatti_D', 'Trchngrd_D', 'Kengeri_IP', 'KHRoad_I',
'RicMilRd_D', 'MlnprDPP_D', 'MiraRd_IP', 'Pashan DPC',
'KhirByps_I', 'Agraroad_I', 'Katrmira_D', 'Sudmpuri_D', 'Potheri',
'Kuslpram_I', 'SamathBv_D', 'VadaiDPP_D', 'ColegRd_D',
'AmvdiDPP_D', 'HudcoDPP_D', 'Ward14_D', 'Nayapalli', 'Nirjanpur_L',
'Uppal_I', 'Jalukbari', 'Sardala_D', 'RPC', 'Chndivli_PC',
'AadiDPP_D', 'Banikhet_D', 'Bangotu_D', 'Chuanpur_I', 'RatanDPP_D',
'Hoodi_IP', 'Kadugodi_D', 'MhprraRD_D', 'AstrrdDPP_D', 'Shop3DPP_D',
'Chikdply_I', 'Mayapuri_PC', 'Mylapore', 'GillChwk_DC', 'Anjur_C',
'Kishangarh_DPC', 'Janakpuri', 'Rohini_DPC', 'Panvel_D',
'MilrGanj_HB', 'Koliplm_I', 'Ghansoli_DC', 'Bhogal', 'KaaduRd_D',
'Patparganj_DPC', 'Salem', 'Mhdhvpur_D', 'Hillcard_DC',
'WardNo3_D', 'Tiruchi', 'Sec 02_DPC', 'Sec-83_DC', 'Kadipur',
'Chukhndi_D', 'Hosp1Rd_D', 'Tetulttol_D', 'Wardno5_D',
'HplrdDPP_D', 'Sraccmplx_D', 'FoySGRRD_I', 'Nayagaon_I', 'Basni_I',
'SamitiRd_D', 'JawaharN_D', 'RoshnBgh_I', 'AmzonDev_V',
'Mundhawa_L', 'Vadgaon Sheri DPC', 'Alwal_L', 'LVMColge_D',
'GtRoad_D', 'HrihrNgr_I', 'Madaprum_D', 'Thmpulam_D', 'HnumnDPP_D',
'Raghogrh_D', 'Bypassrd_D', 'Truptingr_D', 'Faridabad', 'Peenya_L',
'Pandrnga_I', 'PunjabiB_L', 'Mdiclcly_D', 'CGRoad_D', 'Umalodge_D',
'Hejunagr_D', 'Wardno13_D', 'MahmурGj_IP', 'Adargchi_IP',
'NamoNagr_D', 'Bsavangr_D', 'Cdoscldr_D', 'GadagRD_D', 'Sector4_D',
'GndhiNGR_D', 'MChwkDPP_D', 'Govind_D', 'Parasi_D', 'Waidhan_D',
'Ward8DPP_D', 'PC', 'Central_I_2', 'SriDPP_D', 'SriRmNGR_D',
'Raiprvlg_L', 'MathuraRD_D', 'Panipat', 'kankroli_D', 'Nimachrd_D',
'Okhla_PC', 'Bownplly_C', 'Narynpur_C', 'Madhavaram_DPC', 'HUB',
'Chikdply_C', 'Markndpr_D', 'Udyabata_D', 'Sector02_C',
'Kuntikna_H', 'Kurdwudi_D', 'Kakdepot_D', 'Kothapet_D',
'SubhVRTL_I', 'Tiglgndi_D', 'Kacheri_D', 'BgwriDPP_D', 'CP',

'Vardhard_D', 'Vidygiri_D', 'Sanpada_CP', 'Egmore_C',
'Begumpur_CP', 'Sodal_Road', 'Ramvlg_D', 'Dehradun', 'MhmodNgr_D',
'Beleghtha_CP', 'MndiRoad_D', 'MohanNgr_C', 'Prbhunth_D',
'Soghra_D', 'PnditNGR_D', 'Madarpur_D', 'ChndrNgr_D', 'Robinson_D',
'Poothole_D', 'Peedika_H', 'Alandur_C', 'VaniThtr_D', 'BMRd_DC',
'Bburynkpl_D', 'Karelibaug_DPC', 'MhimWest_C', 'Malahi_D',
'Bgtichk_D', 'Vasai_CP', 'Mdhavram_C', 'Sector63_L', 'Karayam_H',
'Sarubali_D', 'Sirjudin_D', 'YuktiDPP_D', 'IndstlAr_I', 'Nagar_DC',
'ShivNgar_D', 'Katira_D', 'Sirikona_H', 'Jantaclg_D', 'Bankura_D',
'Salanpur_D', 'Talkui_D', 'Jogeshwri_L', 'ZebaTWR_D', 'BhgyaNgr_D',
'Mhdiptnm_C', 'North_R_8', 'Pratpngr_D', 'BSarani_D', 'NagarDPP_D',
'MaxDPP_D', 'BtaiRoad_D', 'Kapshera_L', 'Chungam_D', 'HatRDDPP_D',
'WdN14DPP_D', 'Egmore_DPC', 'Mangri_I', 'Atapaka_D', 'Agraharm_DC',
'GvrCompx_D', 'ArhamDPP_D', 'DindIRD_D', 'PeroorRD_D',
'VidyaNGR_D', 'GndhiDPP_D', 'Wazirpur_L', 'Alwal_I', 'SrifoDPP_D',
'Kntgorya_D', 'DohalDPP_D', 'BoiDPP_D', 'NditaDPP_D',
'Gajuwaka_IP', 'KrsprDPP_D', 'Sirjudol_D', 'GopalDPP_D',
'UtBzrDPP_D', 'SavtaHTL_D', 'Hindcwk_D', 'MiraRoad_M',
'GrmNgriya_D', 'ViksClny_D', 'kalibari_D', 'Konapara_D',
'Indsarea_D', 'PlaceCol_D', 'Bhogpur_D', 'TahurDPP_D', 'Vijdurg_D',
'Bardivan_D', 'Wardno6_D', 'Pinjore_DC', 'MohanPrk_D',
'Thsil3PL_D', 'Gaurkshn_I', 'SChwkDPP_D', 'Mundhe_D', 'Shantanu_D',
'LxmiNiws_D', 'ColctrOf_D', 'LSRoad_DC', 'SikriKla_DC', 'Meerut',
'Krishnpr_D', 'SadRHsptl_D', 'Gangjala_D', 'Central_H_2',
'Bazar_D', 'Wardn13_D', 'Durma_D', 'Ward9DPP_D', 'WardNo1_D',
'IndraNgr_D', 'Enkndl_a_D', 'Srnvsngr_D', 'RKComplx_D',
'ShtDRDPP_D', 'VikrmMah_D', 'Rohtak', 'DvlalDPP_D', 'GainMrkt_L',
'BsStdDPP_D', 'Darbhanga', 'Jawahar_D', 'Nagar_D', 'patna_D',
'Vijayght_D', 'Shyndco_D', 'AgrohDPP_D', 'Gokulam_D', 'GVManu_D',
'SaiNgr_D', 'Madnpali_D', 'Ambala', 'Gurudwar_D', 'Jhilmil_L',
'BhukrdPP_D', 'Mwalibad_D', 'AjmhwdPP_D', 'Tilknagr_DC', 'Kumud_D',
'Sookhtal_D', 'SuzkiSrv_D', 'Subhash_D', 'Barmasia_D',
'NraynDPP_D', 'Nangli_L', 'Matriprm_IP', 'CCRoad_D', 'Subshngr_D',
'BawliDPPP_D', 'Banjaria_D', 'PchpkRD_D', 'BypassRd_D',
'TrengaRD_D', 'Panderia_D', 'Dudhani_D', 'D', 'Hatpada_D',
'Bhaleti_D', 'SainkSCL_D', 'Ward6_D', 'Dayanand_D', 'Fathuluh_D',
'NaginaRd_D', 'NaginaRD_D', 'KotdwrrRD_D', 'HunterRd_I',
'Nrsampt_D', 'Perkadrd_D', 'KdidmCLY_D', 'Khwssrai_D', 'Ganesh_D',
'Trimurti_D', 'Harop_D', 'KrshDPP_D', 'Ward2DPP_D', 'Vaishali_D',
'BhwniGnj_D', 'BajprDPP_D', 'MubarDPP_D', 'BgnprDPP_D', 'RamNgr_D',
'RajaBzr_D', 'Mohim_D', 'Datatrtya_D', 'EBroad_D', 'Bidar',

'RIICO_L', 'StatonRd_D', 'Shankrpa_D', 'Verpatem_D', 'RailGate_D',
'Kalol_DC', 'PODPP_D', 'MheshNGR_D', 'ITDARD_D', 'AskNagar_D',
'SrnprHwy_D', 'DmodrNGR_D', 'IndraCln_D', 'NWclyDPP_D',
'Ward6DPP_D', 'MndwrRod_D', 'HnsChowk_D', 'FatehpRd_I',
'NavldiDPP_D', 'JdswarRD_D', 'Kalyanpr_D', 'KlngrDPP_D', 'NH117_D',
'Antop Hill', 'Pariplly_D', 'Rdiosttn_D', 'NrainaRD_D',
'NarenaRD_D', 'ChomuRD_D', 'Sarswati_D', 'Nishangr_D',
'NgrNigam_DC', 'Wardno4_D', 'Purbari_D', 'Pnjbiyon_D', 'SJRoad_D',
'DcntCLY_D', 'Kanongyn_D', 'Pettah_D', 'Arlumodu_D', 'MhraChng_D',
'AryaNagr_D', 'Virar_DC', 'JNPT_D', 'Pbroad_DC', 'Goa',
'ZuariNgr_IP', 'Mohan_Nagar_DPC', 'Swargash_D', 'HapurRD_D',
'WardNo4_D', 'MnBzrDPP_D', 'GvrdnDPP_D', 'Kalyan West _Dc',
'Ambernath_Dc', 'ShivaDPP_D', 'GunjRDPP_D', 'Aravind_D',
'Mumbra_DC', 'Hitech_D', 'KamnHbRD_I', 'UzanBazr_DC', 'TKRoad_D',
'KeRoad_D', 'JatniDPP_D', 'Klskhrpt_D', 'PostofJN_D', 'Amankovl_D',
'Town_D', 'MIDCAvdn_I', 'DhuleRoad_D', 'KakaCplx_D', 'Nandrbar_D',
'DhuleRd_D', 'Dilliyan_D', 'Jaipur', 'Shahdara', 'DhnliRth_D',
'Pdmavati_D', 'KhsmiDPP_D', 'Balaji Nagar', 'BhadgDPP_D',
'RamaNgr_D', 'Parigi_D', 'StatinRD_D', 'AdrshSt_DC', 'keshod_DC',
'JilRDDPP_D', 'GayatriN_D', 'Kotwali_D', 'NehruNGR_D',
'Arulimod_D', 'Ajnari_D', 'Madhavaram_L', 'GndhiNgr_D', 'Bokule_H',
'JyotiNgr_D', 'BrlwgDPP_D', 'Selakui_D', 'Ward19_D', 'PalikDPP_D',
'Nehru3PL_D', 'SttinDPP_D', 'FulbaDPP_D', 'Padra_D', 'CikhliRD_D',
'StnRdDPP_D', 'Itachnda_D', 'WebeIDPP_D', 'MilpaDPP_D',
'Pshimpra_D', 'Uppal_L', 'BOB_D', 'Samarth_D', 'StRoad_D',
'Tolichwk_I', 'Chndivli_D', 'Vadodara', 'Sholinganallur_Dc',
'Sixmile', 'LB-Nagar_Dc', 'Panchkula', 'Bhgtpura_D', 'VishnuVhr_D',
'ArkonmRD_D', 'MBTRd_DC', 'Sadras_D', 'Central_DPP_4', 'Chakan_D',
'ColageRD_D', 'Srikot_D', 'Khajuria_I', 'ChrliDPP_D', 'SashPhkn_D',
'ZoomCDPP_D', 'Shillong', 'Shanthis_D', 'ShsmlDPP_D', 'KotwaliN_D',
'PBRDDPP_D', 'MduraiRD_D', 'LICOffce_D', 'Kaithwal_D',
'SmClyDPP_D', 'LXngrDPP_D', 'TrtllaRD_L', 'Blockrd_D',
'ManhrBld_D', 'Blmrgnst_D', 'StationRD_D', 'KrthiKyn_D',
'Satara_D', 'Idgah_P', 'Udupi', 'Kakrmath_D', 'KmkshBul_D',
'Prbhtngr_D', 'MarketRd_D', 'MrdiVlge_D', 'Ameenpur_I',
'Pazhvedu_D', 'Kumrpurm_D', 'Feroke_H', 'Mandodi_D', 'Pshrikvu_D',
'ZamQuatr_D', 'Mettu_DC', 'Thiruviz_D', 'Vadasari_D',
'Kdthdsttrt_D', 'VaikLSRT_D', 'MukkuRD_D', 'Davisdle_D', 'nagar_D',
'Badeplly_D', 'anthniyr_D', 'HsptlRod_D', 'SH71_D', 'Puduvalvu_D',
'Sbrmnprnm_D', 'Alngjuri_D', 'Sishumdr_D', 'PriyrNGR_D',
'Mthrapuri_D', 'ThryrRD_D', 'goplpurm_D', 'Kmrnjngr_D',

'Chithbrm_D', 'AmtlaDPP_D', 'JhumanCk_D', 'Kappalur_H',
'AshkTalk_D', 'Msstreet_DC', 'Krsnakcl_D', 'Kovil_D', 'Mlydpthr_D',
'Ward25_D', 'TherSRT_D', 'Kovaipudur_Dc', 'West_Dc', 'Khjurwli_DC',
'RtlamNka_D', 'Patiala', 'Sahni_D', 'MrutingR_D', 'Palikval_D',
'Talaiya_D', 'AshkngRd_D', 'Ward17_D', 'BhogdDPP_D', 'BMRd_D',
'MuruPost_D', 'MandyaRD_D', 'MuthpTmp_D', 'TilakNgr_D', 'YTRd_D',
'HsnRdDPP_D', 'Anaipeta_D', 'GodamDPP_D', 'Ambedkar_D', 'Sunku_D',
'FshryOFC_D', 'KolarRd_D', 'Venkatssa_DC', 'Artclgrd_D',
'GuttalRD_D', 'War5DPP_D', 'HoliCDPP_D', 'KairiyaT_D',
'Wardno10_D', 'Techrcly_D', 'Thvlrsrt_D', 'Palladam_DC',
'Bhabua_D', 'Torwa_DC', 'HousngBd_D', 'New Alipore_DPC',
'Mutvila_D', 'Chnglptu_DC', 'Achipkam_D', 'Yellanda_D',
'Sudimala_D', 'VdkkuSrt_I', 'Radhaprm_D', 'Nallur_D',
'ChtrGIDC_IP', 'MrktYrd_DC', 'UdnkdiRD_D', 'Shnmgrpm_D',
'SriVnktpm_D', 'SKRoad_D', 'VidyaNgr_D', 'PushPlza_D',
'Kalynpur_I', 'Kataram_D', 'JwahrNGR_D', 'LaxmiNGR_D', 'Darbe_DC',
'ConduDPP_D', 'JrjolDPP_D', 'Chandkheda_Dc', 'Chaitnya_D',
'MohnVRTL_D', 'JydevNGR_D', 'CivlLine_D', 'ChngiDPP_D',
'Mlkpura_D', 'RjnndrNgr_DC', 'EtwahDPP_D', 'TBCross_D',
'ShntiSgr_D', 'Kalyan', 'GwhRDDPP_D', 'Vepmpttu_DC', 'TmpleSrt_D',
'Vllyaprm_D', 'RamnadRD_D', 'Raiprkl_C', 'Rawlgaon_D',
'Malegaon_D', 'Varachha_DC', 'South_D_4', 'VasaviNg_D',
'SnkunDPP_D', 'Shahapur_D', 'HBColny_D', 'Mangol_DC',
'KrisnKunj_D', 'Lake Avenue_DPC', 'NaturDPP_D', 'Kolar Mandakni',
'Kadtmpy_H', 'krshnPly_DC', 'EragnDPP_D', 'Tejpal_I', 'BjbNgr_DC',
'PndrgNgr_DC', 'Chandigarh', 'Mangalam_D', 'ShantiNg_D', 'Erode',
'DKLogDPP_D', 'Samrvrni_D', 'BaliaMod_D', 'Ganeshwr_D',
'NagplDPP_D', 'KisanCo_D', 'Palakrty_D', 'Hanmkond_D',
'BodomBzr_DC', 'RadhaCpx_D', 'MaladWest_CP', 'Mohnprwa_D',
'Mnanthla_D', 'ShjnprRD_D', 'Mainrd_D', 'Eaglvari_D', 'Kelasahi_D',
'GhtimDPP_D', 'VallaDPP_D', 'TnhbBlkC_D', 'VagaiNgr_D',
'LxmntDPP_D', 'KdrShrRd_D', 'Veluthur_D', 'AlathurRD_D',
'KKndrDPP_D', 'SagarDPP_D', 'WamanDPP_D', 'SchdvDPP_D', 'LFRoad_D',
'Rathnam_D', 'RgvdrDPP_D', 'HesglDPP_D', 'CroslySRT_D',
'HajiprRD_D', 'Haripur_D', 'Srwdycwk_D', 'Tuminkte_D',
'NharuExt_D', 'NngrgnRd_D', 'Valluvar_D', 'Banshkri_DC',
'Mahindra_D', 'MrentTirh_D', 'NorprRD_D', 'Sitarmrd_D',
'BaraLoha_D', 'Rajula_DC', 'KrantiNgr_D', 'ICDCant_D',
'Greenmkt_D', 'Chandanagar_Dc', 'Mnanthla_H', 'BndhuTRH_D',
'Chandmari', 'APMCYard_D', 'Barwala', 'Katora_IP', 'BasthDPP_D',
'AsnsdhRD_D', 'MahuGDPP_D', 'SourvDPP_D', 'RPRoad_D', 'NagpurRd_D',

'De1RdDPP_D', 'ward9_D', 'Central_D_5', 'Shekhpur_D', 'BypassRD_D',
'MngalDPP_D', 'BhwanDPP_D', 'Hghsc1RD_D', 'Blbgarh_DC',
'Chpaguri_D', 'ShivBari_D', 'PigonDPP_D', 'Samyaprm_D',
'PreetDPP_D', 'Oilmilrd_D', 'MSRClrgRd_D', 'ITICollg_L',
'KatlaDPP_D', 'South_R_11', 'Shivangr_D', 'KnsgraRD_D',
'PuranDPP_D', 'Domlur', 'AnprnDPP_D', 'Chndrlpd_D', 'RmNyrDPP_D',
'GMndiDPP_D', 'Kooriyad_D', 'Munduprm_D', 'ByePass_D',
'SngihiRD_D', 'Sabalpur_D', 'Karnal', 'Shivalya_D', 'PnchmDPP_D',
'Bahreya_I', 'Jisw1DPP_D', 'Gobindgarh_DC', 'KarnalRd_D',
'Bargawan_DC', 'Mughlptra_D', 'GutmgrCl_D', 'Perungudi_DPC',
'BhemuDPP_D', 'Central_D_7', 'Patel_Nagar', 'SidculRd_D',
'Kothuru_D', 'Wardno8_D', 'ThthiCwk_D', 'VidyaDPP_D', 'BaruaRd_D',
'Babupaty_D', 'Bhankrot_DC', 'Trimulgherry_Dc', 'Surajpur_DC',
'TrnptNgr_L', 'MissonRd_D', 'MJRDPP_D', 'AzmrDPP_D', 'Jamalpur_D',
'LNBRoad_D', 'Shop2DPP_D', 'DeVDPP_D', 'KSCLny_DC', 'GagiDPP_D',
'HotelPrk_D', 'SnthiNGR_D', 'BypRDDPP_D', 'BhmrdDPP_D',
'Vijayawada', 'Kidwai_D', 'Mharajpr_D', 'GrudwrRd_D', 'Adrshngr_D',
'Krvnkuzy_D', 'PhdofDPP_D', 'CtyLgdDPP_D', 'Pilani', 'StnRoad_DC',
'Solaiprm_D', 'RailwyRd_D', 'Diakkawn_D', 'PiliKoti_D',
'VislkNgr_DC', 'SaiTempl_D', 'NcsRd_DC', 'Arangadi_D',
'LohiaDPP_D', 'PakriDPP_D', 'Bisno1DPP_D', 'PrmNrDPP_D',
'BsnoiHPL_D', 'Gopa3PL_D', 'ShrprDPP_D', 'MnRDDPP_D', 'B11vMarg_D',
'Farmnala_D', 'Katghara_D', 'KhandDPP_D', 'JngidDPP_D',
'Tolichwk_L', 'Pringla_D', 'SurbhiTh_D', 'Greens_D', 'CivilStn_D',
'Shivprasad_D', 'KirtiNgr_D', 'ModelTwn_P', 'Fairybnk_D',
'Cnsrvila_D', 'BhunaDPP_D', 'Ukkadam_D', 'BChwkDPP_D', 'Jharia_DC',
'Chrwpaty_D', 'CollegeRD_D', 'Pthrgoan_D', 'HelipadRD_D',
'DltprDPP_D', 'Ranakant_D', 'Vandalur_Dc', 'Airport',
'Rahatani_DPC', 'Bhrnikvu_D', 'HunthrVg_I', 'AwmpiVng_D',
'PmthuKlm_D', 'PanditRd_D', 'Chtrpuza_D', 'SantaNGR_D',
'JalnaRd_D', 'Bazarrd_D', 'UmarDPP_D', 'CmtNgRod_D', 'KasyaDPP_D',
'StteHW28_D', 'VijywdRD_D', 'Dehrird_D', 'Wardno7_D', 'NatunDPP_D',
'Enayetpr_D', 'East_I_20', 'KtnRdDPP_D', 'Mapusa', 'Kothriya_DC',
'Ymunpurm_D', 'TonkRoad_D', 'East_D_8', 'SmbjiCwk_D', 'Jabalpur',
'Indrapri_DC', 'YashDPP_D', 'AchneraRD_D', 'DibngVly_D',
'BnglorRd_D', 'PcrrdDPP_D', 'Murtingr_D', 'Central_D_15',
'JmpuCntr_D', 'Rajpura_D', 'Mulapali_D', 'LdnunDPP_D', 'PngnrRd_D',
'Tejpal_M', 'RjghatRd_D', 'Pnchlght_D', 'KDRoad_D', 'farukngr_D',
'SingCLNY_D', 'Jagrata_D', 'ThaneDPP_D', 'MdhsnDPP_D', 'Margao_Dc',
'Nijgan_D', 'Rawatpur_D', 'Gurukrpa_D', 'JivanDPP_D',
'AshokNagar_DC', 'BhmprDPP_D', 'KarjuDPP_D', 'Pothredy_D', '_NAD',

```
'ShbdnDPP_D', 'ClgRDDPP_D', 'Brpc', 'ShubsNGR_D', 'Belngdi_D',
'Ldthlabh_D', 'KamalDPP_D', 'IndEstat_L', 'HnmntNgr_D',
'ShivmDPP_D', 'BstndDPP_D', 'Cherukole_D', 'EmsPnmbi_D',
'KaimgnjRD_D', 'Bhainpura_D', 'FatprDPP_D', 'RajRdDPP_D',
'Mahad_D', 'MsjidDPP_D', 'PaikjNGR_D', 'BalibDPP_D', 'KcharaRD_D',
'ChtwrDPP_D', 'MsmcyDPP_D', 'ElngoNgr_C', 'Rcocmplx_D',
'PunjbiPd_D', 'Shanthi_D', 'Ponda_Dc', 'SirsaDPP_D', 'Mahim',
'Paldi_D', 'SubrtDPP_D', 'Santalpr_D', 'TiloIDPP_D', 'Idgah_L',
'KoralDPP_D', 'Nerul_D', 'Uppal_Dc', 'MhliaDPP_D', 'Bomsndra_L',
'SukntDPP_D', 'Arsprmbu_D', 'MnimlaRd_D', 'UttarDPP_D',
'ChatidPP_D', 'MdnpriDPP_D', 'Kanakpur_D', 'East', 'MdothdRD_D',
'CroadDPP_D', 'Sarjapur_D', 'SbhRDDPP_D', 'Central_D_4',
'Thiruvlr_DC', 'Sangetha_D', 'NadthiCx_D', 'Naraynpr_D', 'Hathras',
'AkkolRD_D', 'AnnaNGR_D', 'LamtidPP_D', 'DiyoDPP_D', 'MunplDPP_D',
'RoopNgr_D', 'RwStnDPP_D', 'BhrolDPP_D', 'Umargaon_DC',
'Ramnagar_D', 'Majoor_D', 'BageDPP_D', 'MnbzrDPP_D', 'AkhraBzr_D',
'MemariRD_D', 'ArickDPP_D', 'kalmpuza_D', 'BnkrGate_D',
'ColnyDPP_D', 'Muktsar_D', 'Patia', 'Rjndpara_D', 'BrezeDPP_D',
'GurpdDPP_D', 'Kothanur_L', 'Ricco_D', 'Palam', 'Prjapati_D',
'Gondkhry_H', 'South_D_20', 'Central_H_4', 'KalikDPP_D',
'Bareilly', 'STRdDPP_D', 'RatuaDPP_D', 'BazarDPP_D', 'lalangr_D',
'GndhiChk_D', 'Mirapati_L', 'RhmgjDPP_D', 'Central_L_8', 'Kaura_D',
'Khndyusn_D', 'AmbkaDPP_D', 'Nelmngla_L', 'IOTCEncl_L',
'AnadiDPP_D', 'JJCpxDPP_D', 'Idstr1Ar_D', 'Mehsouri_D',
'HanumDPP_D', 'GovndNgr_DC', 'JthriDPP_DC', 'Karelibaug_DC',
'Bnsibtla_D', 'MjlprDPP_D', 'Sardhnrd_D', 'MbRoad_D',
'Central_D_10', 'Old', 'SliprDPP_DC', 'Kapleswr_D', 'Kdvantra_D',
'SadulDPP_D', 'Pulgaon_DC', 'AkhirDPP_D', 'KrshnNgr_D', 'Parai_D',
'Selaiyur_DC', 'Chinchwad DC', 'NavdaCln_D', 'CBRoad_D', 'Sidrd_D',
'ThanaDPP_D', 'Krusphrma_D', 'Mahuva_DC', 'Manchar_D',
'BargaDPP_D', 'NapitDPP_D', 'RgstrOFC_D', 'Bhaluahi_D',
'Thikiri_D', 'RajpurRD_D'], dtype=object)
```

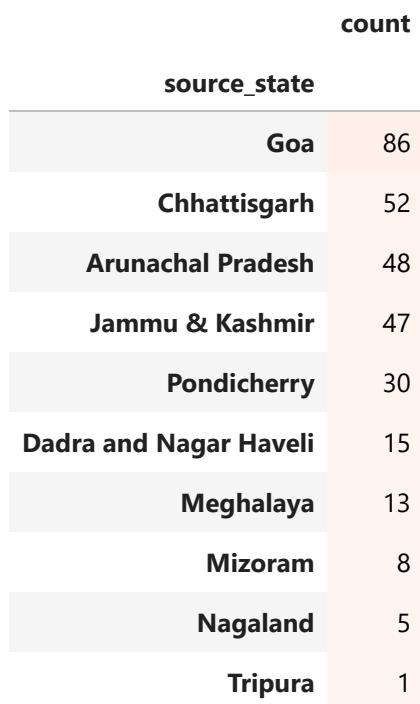
```
In [78]: de['source_state'].unique()
```

```
Out[78]: array(['Uttar Pradesh', 'Madhya Pradesh', 'Karnataka', 'Haryana',
   'Maharashtra', 'Tamil Nadu', 'Gujarat', 'Delhi', 'Telangana',
   'Andhra Pradesh', 'Rajasthan', 'Assam', 'West Bengal', 'Punjab',
   'Chandigarh', 'Goa', 'Uttarakhand', 'Jharkhand', 'Pondicherry',
   'Orissa', 'Himachal Pradesh', 'Kerala', 'Arunachal Pradesh',
   'Bihar', 'Meghalaya', 'Chhattisgarh', 'Jammu & Kashmir',
   'Dadra and Nagar Haveli', 'Mizoram', 'Tripura', 'Nagaland'],
  dtype=object)
```

```
In [253... de['source_state'].value_counts().to_frame().style.background_gradient(cmap='Reds')
```

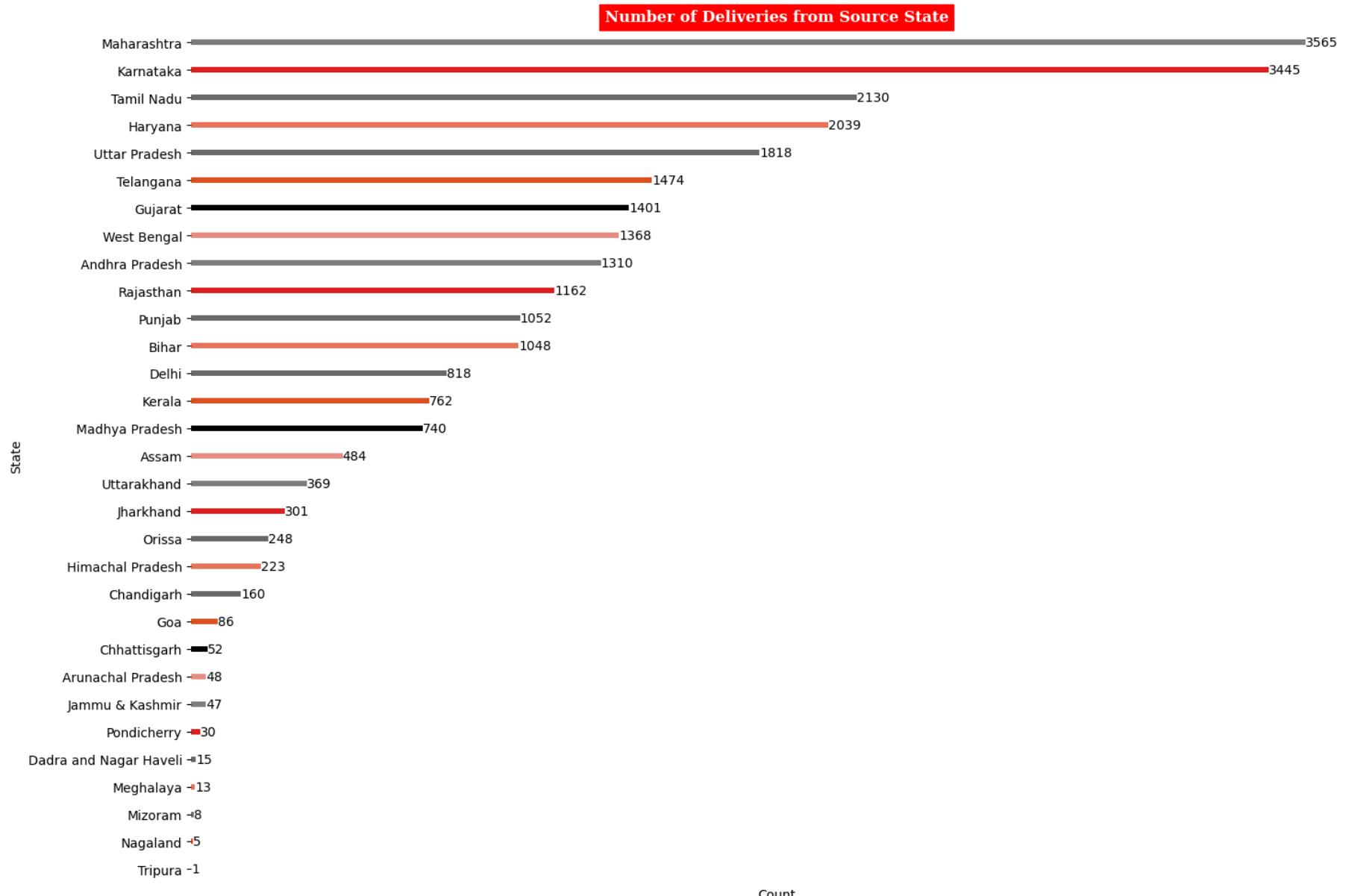
Out[253...]

source_state	count
Maharashtra	3565
Karnataka	3445
Tamil Nadu	2130
Haryana	2039
Uttar Pradesh	1818
Telangana	1474
Gujarat	1401
West Bengal	1368
Andhra Pradesh	1310
Rajasthan	1162
Punjab	1052
Bihar	1048
Delhi	818
Kerala	762
Madhya Pradesh	740
Assam	484
Uttarakhand	369
Jharkhand	301
Orissa	248
Himachal Pradesh	223
Chandigarh	160



```
In [80]: state_counts = de['source_state'].value_counts().to_frame().reset_index()
state_counts.columns = ['State', 'Count']

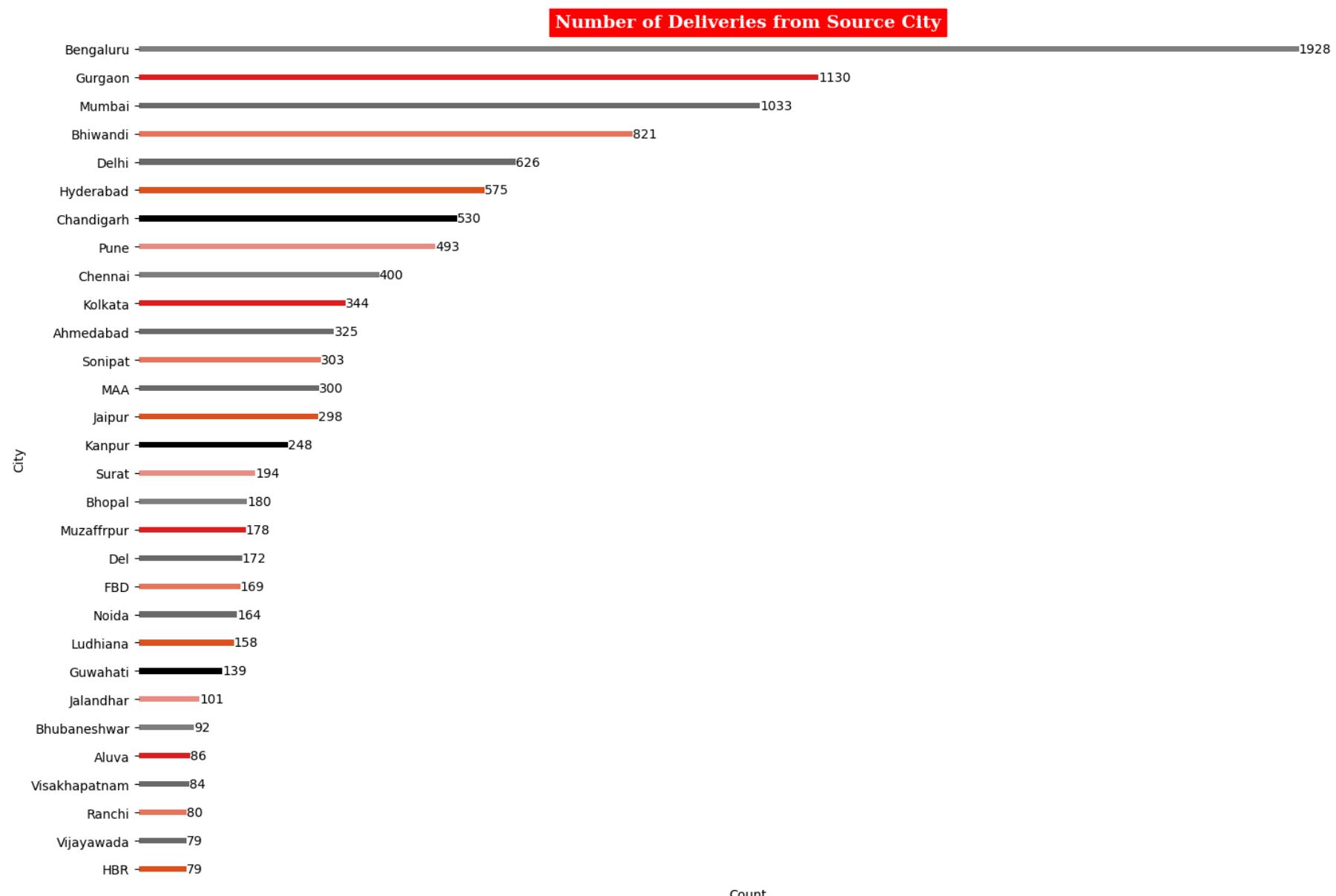
plt.figure(figsize=(15,10))
a = sns.barplot(y='State', x='Count', data=state_counts, palette=cp, width=0.2)
a.bar_label(a.containers[0], label_type='edge')
plt.xticks([])
plt.ylabel('State')
plt.xlabel('Count')
plt.title('Number of Deliveries from Source State', fontsize=12, fontfamily='serif', fontweight='bold', backgroundcolor='r', color='w')
plt.tight_layout()
sns.despine(bottom=True, left=True)
plt.show()
```



```
In [81]: city_counts = de['source_city'].value_counts().to_frame().reset_index()[:30]
city_counts.columns = ['City', 'Count']

plt.figure(figsize=(15,10))
```

```
a = sns.barplot(y='City', x='Count', data=city_counts, palette=cp, width=0.2)
a.bar_label(a.containers[0], label_type='edge')
plt.xticks([])
plt.ylabel('City')
plt.xlabel('Count')
plt.title('Number of Deliveries from Source City', fontsize=14, fontfamily='serif', fontweight='bold', backgroundcolor='r', color='w')
plt.tight_layout()
sns.despine(bottom=True, left=True)
plt.show()
```



💡 Insights:

Source State contributors

- **Maharashtra, Karnataka ,Tamil Nadu, Haryana, and Uttar Pradesh** are the top contributors where maximum bookings are recorded in this month indicating significant engagement.

Source City contributors

- Cities like **Bengaluru, Gurgaon, Mumbai, Bhiwandi, Delhi, Hyderabad** where the major no.of booking are recorded.

```
In [82]: de.describe().T
```

Out[82]:

	count	mean	min	25%	50%	75%	
trip_creation_time	26222	2018-09-22 13:58:56.740969728	2018-09-12 00:00:16.535741	2018-09-17 03:57:50.900417024	2018-09-22 03:33:30.255023104	2018-09-27 19:55:17.273207040	23:59
od_start_time	26222	2018-09-22 17:49:54.012840448	2018-09-12 00:00:16.535741	2018-09-17 07:43:36.525784320	2018-09-22 07:17:15.379571712	2018-09-27 23:22:20.105725952	04:27
od_end_time	26222	2018-09-22 22:49:00.449498112	2018-09-12 00:50:10.814399	2018-09-17 15:10:35.615369472	2018-09-22 14:46:05.410478848	2018-09-28 03:19:50.211971840	03:00
start_scan_to_end_scan	26222.0	298.553375	20.0	90.0	152.0	307.0	
actual_distance_to_destination	26222.0	92.533051	9.001351	21.65415	35.044329	65.557392	19
actual_time	26222.0	200.92659	9.0	51.0	84.0	167.0	
osrm_time	26222.0	90.785332	6.0	25.0	39.0	72.0	
osrm_distance	26222.0	114.975334	9.0729	27.71915	43.54355	85.443949	23
segment_actual_time	26222.0	199.095642	9.0	50.0	83.0	166.0	
segment_osrm_time	26222.0	101.793343	6.0	25.0	42.0	79.0	
segment_osrm_distance	26222.0	125.587128	9.0729	28.429099	45.797649	91.023573	26
segment_actual_time_sum	26222.0	199.095642	9.0	50.0	83.0	166.0	
segment_osrm_time_sum	26222.0	101.793343	6.0	25.0	42.0	79.0	
segment_osrm_distance_sum	26222.0	125.587128	9.0729	28.429099	45.797649	91.023573	26
od_total_time	26222	0 days 04:59:06.436657725	0 days 00:20:42.168787	0 days 01:30:58.665517750	0 days 02:32:20.203949500	0 days 05:07:17.158769	11:38
od_time_diff_hour	26222.0	4.985121	0.345047	1.516296	2.538946	5.121433	1

In [83]: `de.describe(include='object').T`

Out[83]:

	count	unique	top	freq
segment_key	26222	26222	trip-153671041653548748+IND209304AAA+IND000000ACB	1
trip_uuid	26222	14787	trip-153717306559016761	8
source_name	26222	1496	Gurgaon_Bilaspur_HB (Haryana)	1052
destination_name	26222	1466	Gurgaon_Bilaspur_HB (Haryana)	928
source_city	26222	1239	Bengaluru	1928
source_place	26222	1246	Bilaspur_HB	1052
source_state	26222	31	Maharashtra	3565
destination_city	26222	1236	Bengaluru	1863
destination_place	26222	1217	Bilaspur_HB	928
destination_state	26222	32	Karnataka	3497

In [84]: de['destination_city'].unique()

```
Out[84]: array(['Gurgaon', 'Kanpur', 'Chikblapur', 'Doddablpur', 'Chandigarh',  
    'Mumbai', 'Hospet', 'Bellary', 'Sandur', 'Chennai', 'Bengaluru',  
    'HBR', 'Surat', 'Delhi', 'PNQ', 'Faridabad', 'Kolhapur', 'Shirala',  
    'Ratnagiri', 'Anantapur', 'Hyderabad', 'Sindagi', 'Gulbarga',  
    'Indi', 'Aland', 'Jaipur', 'Satna', 'Janakpuri', 'Guwahati',  
    'Unnao', 'Gadarwara', 'Bareli', 'Shirdi', 'Sinnar', 'Sangamner',  
    'Shrirampur', 'Kopargaon', 'Vaijiapur', 'Nashik', 'Kolkata',  
    'Hoogly', 'Hooghly', 'Pavagada', 'Puttaprthi', 'Sivasagar',  
    'Medchal', 'Oddnchtram', 'Batlagundu', 'Vadipatti', 'Kodaikanal',  
    'Palani', 'Nakodar', 'Kapurthal', 'Jalandhar', 'Yavatmal',  
    'Atapadi', 'Sangola', 'Bhandara', 'Savner', 'Kurnool', 'Palwal',  
    'FBD', 'TalwandiSabo', 'Mansa', 'Jhunir', 'Bhatinda', 'Bhiwandi',  
    'Barnala', 'Murbad', 'AnandprShb', 'RoopNagar', 'Puttur', 'Kadaba',  
    'Chittapur', 'Sedam', 'Chincholi', 'Naraingarh', 'Ludhiana',  
    'Ahmedabad', 'Kadi', 'Dola', 'Jabalpur', 'MAA', 'Parli',  
    'Ambajogai', 'Pune', 'Loha', 'Gangakher', 'Barjora', 'Bishnupur',  
    'Bankura', 'Silvassa', 'Junagadh', 'Bhanvad', 'Porbandar',  
    'Dhoraji', 'Upleta', 'Jetpur', 'Choutuppal', 'Suryapet',  
    'Vijayawada', 'Nalgonda', 'Miryalguda', 'Khammam', 'Vadnagar',  
    'Palanpur', 'Deesa', 'Mehsana', 'Katni', 'Kodinar', 'Talala',  
    'Una', 'Chamarjngr', 'Mysore', 'Kollegal', 'Tirumakudalu',  
    'Malavalli', 'Krishnarajnjn', 'Gonikoppal', 'HDKote', 'Unjha',  
    'Bhabhar', 'Radhanpur', 'Rajamundry', 'Visakhapatnam',  
    'Sawantwadi', 'Kankavali', 'Bhopal', 'Bhubaneshwar', 'Allahabad',  
    'Moradabad', 'Rudrapur', 'Sonipat', 'Modasa', 'Khedbrahma',  
    'Himmatnagar', 'Sasaram', 'Ranchi', 'Cuddalore', 'Chidambaram',  
    'Sirkazhi', 'Karaikal', 'Nagaptinm', 'Pondicherry', 'Thiruvarur',  
    'GZB', 'Khambhat', 'Anand', 'Degloor', 'Udgir', 'Latur', 'Nanded',  
    'Noida', 'Umreth', 'Nadiad', 'Panruti', 'Pennadam', 'Chinnasalem',  
    'Villupuram', 'Neyveli', 'Virudhchlm', 'Jhalda', 'Purulia', 'Hura',  
    'Durgapur', 'Bhadrak', 'Goa', 'Balurghat', 'Meham', 'Hisar',  
    'Ambur', 'Tiruppattur', 'Haridwar', 'Kotdwara', 'Narsapur',  
    'Kamareddy', 'Bodhan', 'Medak', 'Banswada', 'Yellareddy',  
    'Dhrangadhra', 'Halvd', 'Gangavathi', 'Koppal', 'Jahu',  
    'BilaspurHP', 'JognderNgr', 'Kharagpur', 'Jhargram',  
    'ChandroknaRD', 'Kirauli', 'BLR', 'Lakhimpur', 'Sitapur', 'Gola',  
    'Dhaurahara', 'Mangalore', 'Canacona', 'Vansda', 'Mananthavady',  
    'Lucknow', 'Silchar', 'Nakhatrana', 'Bhuj', 'Pundibari',  
    'LowerParel', 'Changlang', 'Dahanu', 'Boisar', 'Chodavaram',  
    'Bhalukpong', 'Tezpur', 'Rajampet', 'Tirupati', 'Koduru', 'GGN',
```

'Srikalahsti', 'Venktagiri', 'Gudur', 'Roha', 'Mahad', 'Pen', 'CCU',
'Amdavad', 'AMD', 'Gosainganj', 'Akbarpur', 'Muzaffrpur', 'Purnia',
'Aurangabad', 'KN', 'Phagwara', 'Pandhurna', 'Betul', 'Sausar',
'Tamluk', 'Panskura', 'Haldia', 'Madurai', 'Dindigul', 'Namakkal',
'Erode', 'Aligarh', 'Mainpuri', 'Shikohabad', 'Firozabad', 'Rajam',
'Salur', 'Srikakulam', 'Palakonda', 'Parvathipuram', 'Narasnpeta',
'Palasa', 'Paralakhemundi', 'Tekkali', 'Nalasopara', 'Hajo', 'NOI',
'Dalhousie', 'Chamba', 'Pathankot', 'Baharampur', 'Dhulian',
'Malda', 'Malvan', 'Hoskote', 'Shujalpur', 'Shajapur', 'Ambabadi',
'OK', 'Amritsar', 'Coimbatore', 'Jasai', 'Tirchngode', 'Mettur',
'Kurseong', 'Darjeeling', 'Tiruchi', 'Dadri', 'Del', 'Rangia',
'Nalbari', 'Bilasipara', 'Lakhipur', 'Dhubri', 'Vellore', 'Ajmer',
'Pali', 'Jodhpur', 'Gotan', 'Gajraula', 'Rampur', 'Amroha',
'Dholpur', 'Lalitpur', 'Gwalior', 'Datia', 'Thirthurpondi',
'Pushpavanam', 'Dhanbad', 'Ashokngr', 'Guna', 'Burhanpur',
'Hassan', 'Margherita', 'SrinagarUK', 'Chamoli', 'Gohpur',
'Itanagar', 'Silapathar', 'Pasighat', 'Mirzapur', 'Ghazipur',
'Dharwad', 'Gokak', 'Hubli', 'Ramdurg', 'Gadag', 'Rona',
'Bagalkot', 'Renukoot', 'Anpara', 'Singrauli', 'Robertsganj',
'Panipat', 'Berhampur', 'Ranebennur', 'Haiveri', 'Alwar',
'Bhilwara', 'Udaipur', 'Gandhidham', 'Solapur', 'Belgaum',
'Muktsar', 'Moga', 'Jagatsghpr', 'Paradip', 'Kendrpara',
'Osmanabad', 'Barshi', 'Addanki', 'Kanigiri', 'Kandukur', 'Ongole',
'Bokaro', 'Sirsi', 'Sagara', 'Muzaffrngr', 'Dehradun', 'Deoband',
'Chhatarpur', 'Siwan', 'Nawada', 'Chandi', 'BiharSarif', 'Rajgir',
'Pallakad', 'Vadakkencherry', 'Thrissur', 'Kanakapura',
'Chanapatna', 'Mandya', 'Roorkee', 'Rishikesh', 'Manjeshwar',
'Surathkal', 'Jamshedpur', 'Vadodara', 'Sheikhpura', 'Bakhtiarpur',
'Godhra', 'Dahod', 'Tirupur', 'Kendujhar', 'Barbil', 'Karanjia',
'Rewari', 'Dharuhera', 'Neemrana', 'Hanumangarh', 'Sirsa', 'Ganga',
'Arrah', 'Arwal', 'Jhajjar', 'Bhiwani', 'Kabuganj', 'Kolasib',
'Bardhaman', 'Asansol', 'Rupnarayanpur', 'Midnapore', 'Jalna',
'Sillod', 'Nellore', 'Chapra', 'Nakashipara', 'Plassey',
'SultnBthry', 'Rawatsar', 'Kurukshetra', 'Assandh', 'Pehowa',
'Kaithal', 'Nuzvid', 'Kaikaluru', 'Gudivada', 'Machilipatnam',
'Nowda', 'Domkal', 'Nazirpur', 'Kadthal', 'Haliya', 'Devarakonda',
'Kalwakurthy', 'Khargram', 'Rampurhat', 'Morgram', 'Ghanashyampur',
'Kandi', 'Ragunthgnj', 'Lalgola', 'Sagardighi', 'Jangipur',
'AhmedNagar', 'Ashti', 'Parner', 'Bilaspur', 'Baheri', 'Puranpur',
'Hailakandi', 'Karimganj', 'Jorhat', 'SundarNgr', 'Kullu', 'Mandi',
'Hiriyur', 'Davangere', 'Chitradurga', 'Draksharamam', 'Madhubani',

'Jaynagar', 'Baddi', 'Solan', 'Parwanoo', 'Shegaon', 'Mehkar',
'Akola', 'Chiplun', 'Karad', 'Khed', 'Islampur', 'Raichur',
'Wanaparthy', 'JoguGadwal', 'Modinagar', 'Meerut', 'Saharsa',
'Triveninganj', 'Supaul', 'Madhepura', 'Araria', 'Raniganj',
'Simrahi', 'Sheohar', 'Pupri', 'Sitamari', 'Srisailam', 'Nandyal',
'Bngnpalle', 'Tiptur', 'Mallapur', 'Dandeli', 'Athani', 'Bijapur',
'Jind', 'Gohana', 'Rohtak', 'Patiala', 'Beawar', 'Bijainagar',
'Darbhanga', 'Benipur', 'Jhanjharpur', 'Krishnarajpet',
'Channaraya', 'Khagaria', 'Naugchia', 'Katihar', 'Ratia',
'Dharmavram', 'Kadiri', 'Rayachoti', 'Pulivendula', 'YamunaNagar',
'PaontSahib', 'Kishangarh', 'Parbatsar', 'Merta', 'Degana',
'Makrana', 'Raigarh', 'Ranikhet', 'Nainital', 'Pithorgarh',
'Almora', 'Godda', 'Bharatpur', 'Bayana', 'Deoria', 'Salempur',
'Gorakhpur', 'Kaptanganj', 'Kushinagar', 'Bettiah', 'Narktiganj',
'Dhaka', 'Mungeli', 'Kawardha', 'Pakur', 'Dumka', 'Bangana',
'Bhota', 'Nadaun', 'Bahadurgarh', 'Thakurdwara', 'Najibabad',
'Kanth', 'Dhampur', 'Khanpur', 'Mahbubabad', 'Warangal',
'Chandausi', 'Aonla', 'Sambhal', 'Ratlam', 'Kolaghat', 'Bagnan',
'Lalru', 'Kashipur', 'Ramnagar', 'Sikandarpur', 'Ratanpura',
'Sahatwar', 'Balaghat', 'Motihari', 'Raxaul', 'Zahirabad',
'Humnabad', 'Bidar', 'Narayankhed', 'Gooty', 'Guntakal',
'Rayadurgam', 'Kalyandurg', 'Gandhinagar', 'Amd', 'Patran',
'Samana', 'Manuguru', 'Sathupally', 'Shamli', 'Bassi', 'Dausa',
'Hindaun', 'Lalsot', 'Karauli', 'Gangapur', 'Nawalgarh', 'Pilani',
'Khambhalia', 'Bhatiya', 'Dwarka', 'Kakdwip', 'Ambah',
'Kilimanoor', 'Kalluvathukal', 'Attingal', 'Limbdi', 'Botad',
'Surendranagar', 'Dudu', 'Phulera', 'Renwal', 'Dumraon',
'Jagdishpur', 'Buxar', 'DalsinghSarai', 'Rusera', 'Manjhaul',
'Sikandrabd', 'Jewar', 'Pahasu', 'Anupshahar', 'Neyatinkra',
'Kallikkad', 'Etawah', 'Auraiya', 'Kahalgaon', 'Bhagalpur',
'Panaji', 'Jagraon', 'Raikot', 'Pilkhuwa', 'Hapur', 'Dholi',
'Samastipur', 'Nawanshahr', 'Kaman', 'Karimnagar', 'Metpally',
'Jagtial', 'DhrmpuriTS', 'Mancherial', 'Tiruvalla', 'Haripad',
'Mundakayam', 'Khurdha', 'Khanna', 'Rajpura', 'Adoor',
'Kottarakkara', 'Pthnmthitt', 'Punalur', 'Kollam', 'Shirpur',
'Dhule', 'Nandurbar', 'Sakri', 'Shahada', 'AurngbadBR', 'Kanti',
'Gadchiroli', 'Chamorshi', 'Pandharpur', 'Zirakpur', 'Aluva',
'Chalisgaon', 'Kannad', 'Deoli', 'Bhadaur', 'Hindupur', 'Hamirpur',
'Ghatampur', 'Mangrol', 'Veraval', 'Fatehpur', 'Mahoba', 'Banda',
'Pukhrayan', 'Orai', 'Konch', 'Jamjodhpur', 'Mejia', 'Rghunthpur',
'Duliajan', 'Dibrugarh', 'Digboi', 'Mussoorie', 'Tarkeshwar',

'Kalpetta', 'Pokhran', 'Phalodi', 'Rewa', 'Sidhi', 'Bilimora',
'Bolpur', 'Guskhara', 'Nanoor', 'Mathabhang', 'CoochBehar',
'Dinhata', 'Kathua', 'Samba', 'Raver', 'Jalgaon', 'Narsinghgarh',
'Pachore', 'Vapi', 'Hyd', 'Barpeta', 'Sholinghur', 'Talegaon',
'Rajgurunagar', 'Jairampur', 'Kittur', 'Karnaprayag', 'Tangi',
'Dhemaji', 'Shillong', 'Bailhongal', 'Chomu', 'Sikar', 'Balrampur',
'Tulsipur', 'Gonda', 'Melur', 'Chittaurgarh', 'Kavali', 'Banswara',
'Ghosu', 'Achrol', 'Paota', 'Jehanabad', 'Edappal', 'Guruvayoor',
'Attur', 'Chetpet', 'Arani', 'Polur', 'Khandala', 'Wai', 'Loharu',
'Uchila', 'Kundapura', 'Bhatkal', 'Honnavar', 'Kumta', 'Ankola',
'Dhrmsthal', 'Karkala', 'Tumkur', 'Kottayam', 'Alappuzha',
'Koyilandy', 'Kallachi', 'Vadakara', 'Kuthuparamba', 'Mylkduthuri',
'Mannargudi', 'Thirukkattupli', 'Muthukulathur', 'Narikkudi',
'Coonoor', 'Gudalur', 'Mahabubnagar', 'Bijnapally', 'Jadcherla',
'Peravurani', 'Karaikudi', 'Aranthangi', 'Pudukkotti',
'Ponnamaravathi', 'Badarpur', 'Betamcherla', 'Kulithalai',
'Perambalur', 'Thuraiyur', 'Ariyalur', 'Jayamkondan', 'Lalpet',
'Karwar', 'Kalka', 'Forbesganj', 'Narpatganj', 'Sivakasi',
'Maharajapuram', 'Sankaran', 'Rajpalayam', 'Gobicheti',
'Anthiyour', 'Jaora', 'Nabha', 'Malerkotla', 'Kattappana', 'Rehli',
'Bina', 'HanumanJNC', 'Vuyyuru', 'Madikeri', 'Koppa',
'Ramanathapuram', 'Razole', 'Amalapuram', 'Kherli', 'Kolar',
'Mulbagal', 'Chintamani', 'Bangarapet', 'Bethamangala',
'Udumalpet', 'Pollachi', 'Ukkadagatri', 'Khandela', 'Pilibhit',
'Baghpat', 'Chakia', 'Kangayam', 'Tirpur', 'Dharapuram', 'Dinara',
'Champa', 'Bansi', 'Kanchipuram', 'Arakkonam', 'Fatepur',
'Nedumangad', 'Malemruvathur', 'Yellandu', 'Rayaparthi', 'Vallior',
'Nagarcoil', 'Tirunelveli', 'Amreli', 'Eral', 'Peikulam',
'Thisayanvilai', 'Tiruchchndr', 'Kodad', 'Kusumnchi', 'Sahebganj',
'Deoghar', 'Jamtara', 'Parkal', 'Mulugu', 'Mahadevpur',
'Bhupalpally', 'Sullia', 'Krishnagiri', 'Husnabad', 'Narsinghpur',
'Ramagundam', 'SikandraRao', 'Etah', 'Jalesar', 'Ghaziabad',
'Navsari', 'Valsad', 'Haldwani', 'Gauribidanur', 'Bagepalli',
'Chabua', 'Chhaygaon', 'Sivaganga', 'Manamelkudi', 'Oriyur',
'Thiruvadanai', 'Saraiya', 'Satana', 'Malegaon', 'Manmad',
'Siruguppa', 'Asangaon', 'Bhuvanagiri', 'Mothkur', 'Central',
'Mahasamund', 'Chandpur', 'Patan', 'Madakasira', 'Salem',
'Bamangola', 'Balipara', 'Jassur', 'Jammu', 'Sehore', 'Ashta',
'Perundurai', 'Marakkanam', 'Bhusawal', 'Mau', 'Betnoti',
'Balasore', 'Tarnau', 'Didwana', 'Bantwal', 'Rajsamand', 'Jangaon',
'JhumriTlya', 'Gopalganj', 'Shimoga', 'Tikamgarh', 'Panna',

'Jhabua', 'Trivandrum', 'Faridpur', 'Shirur', 'Mokokchung', 'Tuni',
'Kakinada', 'Nabarangpr', 'Jeypore', 'Rameswram', 'Ramnthurpum',
'Paramakudi', 'Nasirabad', 'Kekri', 'Bilara', 'Badnaur',
'Patancheru', 'Shoranur', 'Pazhayannur', 'Sankaramangalam',
'Buldhana', 'Chikhli', 'Fatehabad', 'Cumbum', 'Sakleshpur',
'Nagamangala', 'Sathyamangalam', 'Ooty', 'HazratJandaha',
'Machhiwara', 'Khurai', 'Malur', 'Lonavala', 'Baraut', 'Avinashi',
'Jhunjhunu', 'Shrimadhopur', 'Sultana', 'NeemKaThana', 'DehriSone',
'Bhind', 'Bijnor', 'Madhupur', 'Bhavnagar', 'Shindkheda',
'Jammikunta', 'Sangareddy', 'Manvi', 'Sindhanur', 'Raipur',
'Vemulawada', 'Kaghaznagar', 'Asifabad', 'Manthani', 'Chinnur',
'Basti', 'Gangarmpur', 'Panagarh', 'Nanjangud', 'Gundlupet',
'Thirumalagiri', 'Seoni', 'Chhindwara', 'Lakhnadon', 'Hansi',
'Bobbili', 'Soraon', 'Phulpur', 'Maihar', 'Churhat', 'Anupgarh',
'Srivijaynagar', 'Dharmapuri', 'Kokrajhar', 'Bongaigaon',
'Ghanpur', 'Chopan', 'Markapur', 'Vinukonda', 'North', 'Bikaner',
'Nagaur', 'Sujangarh', 'Challakere', 'Molakalmuru', 'Udupi',
'Satara', 'Joda', 'Narnaul', 'Nandigama', 'Nipani', 'Nohar',
'Sidhmukh', 'Malappuram', 'Printhlmna', 'Katwa', 'Bareilly',
'Benipatti', 'Dighwara', 'Karnal', 'Sangrur', 'Bariya', 'Budhana',
'Barhi', 'Rath', 'Sujanpur', 'Shahdol', 'Anakapalle', 'Narsiptnm',
'Golaghat', 'Bishwanath', 'Ranipet', 'Kalpakkam', 'Jiyanpur',
'Dohrighat', 'Mangaldoi', 'Dhekiajuli', 'Reengus', 'Nokha',
'Peterbar', 'Hazaribag', 'Ramgarh', 'Mandapeta', 'Shadnagar',
'Vikarabad', 'Tandur', 'Kotagiri', 'Narsingpur', 'Nandikotkr',
'Mandla', 'Palampur', 'Kangra', 'DehraGopipur', 'Pangodu',
'Pappadahandi', 'Anuppur', 'PaliBirsighpr', 'Rajgarh', 'Rajkot',
'Tenkasi', 'Ambasamdrm', 'Mohania', 'Bheemunipatnam',
'Vizianagaram', 'Kasaragod', 'Moodbidri', 'Anandnagar',
'Maharajganj', 'Nichlaul', 'RaisingNgr', 'Jaisalmer', 'Barmer',
'GanjBasoda', 'Sagar', 'Akhnoor', 'Piparcity', 'Tura', 'Jaunpur',
'Jalalpur', 'Pratapgarh', 'Cochin', 'Arimbur', 'Khanapur',
'Irinjlkuda', 'Chalakudy', 'Jath', 'Padrauna', 'Theni', 'Mudigere',
'Barwala', 'Chaksu', 'Chimkurthy', 'Giridih', 'DoomDooma',
'Tinusukia', 'Namsai', 'Tezu', 'Islampure', 'Dharmanagr',
'Agartala', 'Chandauli', 'Manthuka', 'Karungaply', 'Champhai',
'Aizawl', 'Kozhikode', 'Kunnathund', 'Paithan', 'Georai',
'Shevgaon', 'Beed', 'Koraput', 'Atmakur', 'Dhone', 'Teok',
'Mariani', 'Arsikere', 'Areacode', 'Eluru', 'SirhindFatehgarh',
'Bikramgang', 'Bellmpalli', 'Raiganj', 'Kaliyaganj', 'Paranpur',
'Sihora', 'Shivpuri', 'Gondal', 'Khurja', 'Tonk', 'Parbhani',

```
'Suratgarh', 'Mandsaur', 'Sitamau', 'Agra', 'Amalner', 'Pachora',
'ErandoI', 'Pataudi', 'LakhimpurN', 'Gopiganj', 'Baripada',
'MughalSarai', 'Palamaner', 'Umerkote', 'Shamshabad', 'Dhuri',
'Farrukhbad', 'Shahabad', 'Shahjhnpur', 'Aliganj', 'Jalalabad',
'Palakollu', 'Samsi', 'Wankaner', 'BariSadri', 'Pilibanga',
'Vishakhapatnam', 'Lodhan', 'Hoshangabad', 'Bokakhat', 'Moranhat',
'Anjar', 'Barauni', 'Gahmar', 'Jowai', 'Varanasi', 'Chiraiyakot',
'ChrkhiDdri', 'Kozhenchery', 'Chhibramau', 'Karukachal',
'Parakkadavu', 'Nimbahera', 'Neemuch', 'MirzapurWB', 'Berhampore',
'Manbazar', 'Khatra', 'Kothanalloor', 'Badlapur', 'Saraimeer',
'Shanganj', 'Kerala', 'Buhana', 'Tallada', 'Itahar', 'Manikchak',
'Sonari', 'Balotra', 'Simlapal', 'Ghatkesar', 'Helencha',
'Jaleswar', 'Soro', 'Manglamaro', 'Keshiary', 'Contai', 'Howrah',
'Udala', 'Chikmagalur', 'Kasganj', 'Umaria', 'MarwarJn', 'Jalore',
'Tirthahalli', 'Khanakul', 'Chikodi', 'Khamgaon', 'Sundargarh',
'Rajgangpur', 'Bargarh', 'Bhinmal', 'Sirohi', 'Dabhoi',
'RampuraPhul', 'Bhilad', 'Arambag', 'Vaikom', 'Manteswar',
'Nabadwip', 'Gangarampr', 'Bongaon', 'Morena', 'Peddapalli',
'Roing', 'Fazilka', 'Cjb', 'Sonepur', 'Sambalpur', 'Balangir',
'Sumerpur', 'Jasdan', 'Khalilabad', 'Nagpur', 'Aunrihar',
'Mungaoli', 'Chanchal', 'Falna', 'Gujiliam', 'Malout', 'Gomoh',
'Kamarpukur', 'AmaDubI', 'Azamgarh', 'Phusro', 'Tirurangadi',
'SawaiMadhopur', 'Sultanganj', 'Giddarbaha', 'Bhadra', 'Hanskiali',
'Munger', 'KharagpurBR', 'Baruipur', 'Khatauli', 'Ambegaon',
'Abohar', 'Nirsa', 'Itarsi', 'Cuttack', 'Taranagar', 'Kanker',
'Luxettipet', 'Basta', 'Bhadohi', 'Tilhar', 'Tadipatri',
'Proddatur', 'Badvel', 'Cuddapah', 'Angul', 'Phulbani', 'Khetri',
'Daman', 'Uthangarai', 'Durg', 'Thachnttukra', 'Krishnanagar',
'Ranaghat', 'Kapadvanj', 'Lunawada'], dtype=object)
```

```
In [85]: de['destination_place'].unique()
```

```
Out[85]: array(['Bilaspur_HB', 'Central_H_6', 'ShntiSgr_D', 'ChikadPP_D',
   'Mehmdpur_H', 'MiraRd_IP', 'Hospet', 'Dc', 'WrdN1DPP_D',
   'Sriperumbudur_Dc', 'Poonamallee', 'Vandalur_Dc', 'NwYlhnska_DC',
   'Layout_PC', 'Central_I_4', 'Central_D_3', 'Bhogal',
   'Rahatani_DPC', 'Faridabad', 'Shivaji_I', 'Central_DPP_3',
   'MjgaonRd_D', 'KamaStrt_I', 'Nelmngla_H', 'Uppal_I', 'KalyanNg_D',
   'Nehrugnj_I', 'SindgiRD_D', 'RazaviRd_D', 'Bhankrot_DC',
   'Central_I_7', 'Central_I_2', 'Janakpuri', 'Hub', 'VikasRam_D',
   'MPward_D', 'SourvDPP_D', 'Varachha_DC', 'SaiBansi_D',
   'Central_D_1', 'JawanChk_D', 'DavkharRd_D', 'NkshterPz_D',
   'YeolaRD_D', 'TgrniaRD_I', 'North_I_4', 'Bandel_D', 'DC',
   'PnukndRD_D', 'Gokulam_D', 'Babupaty_D', 'Bomsndra_HB',
   'MROoffce_D', 'Alwal_I', 'Palani_D', 'RT0ofice_D', 'lalaNGR_D',
   'Athithnr_DC', 'RjnndraRd_D', 'ChowkDPP_D', 'DPC', 'Mohali',
   'Mullanpr_DC', 'Sanpada_I', 'JajuDPP_D', 'Vidyangr_D',
   'Central_DPP_2', 'Dankuni_HB', 'KaremDPP_D', 'Wagodha_D',
   'Shamshbd_H', 'AbbasNgr_I', 'Palwal', 'Balabgarh_DPC',
   'Wardno3_D', 'GreenVly_D', 'BhaRDDPP_D', 'Mankoli_HB',
   'SnkunDPP_D', 'PnjPiara_D', 'ChotiHvl_DC', 'Kharar_DC', 'Darbe_DC',
   'AnugrDPP_D', 'GlbrgaRD_D', 'DBRCmplx_D', 'Mainroad_D',
   'Ward2DPP_D', 'MilrGanj_HB', 'East_H_1', 'KaranNGR_D',
   'ChainDPP_D', 'Adhartal_IP', 'Poonamallee_HB', 'JirgeNgr_D',
   'BnsllNgr_D', 'Tathawde_H', 'SivjiCWK_D', 'Busstand_D',
   'Central_DPP_1', 'StnRdDPP_D', 'BhowmDPP_D', 'Samrvrni_D',
   'JmnvdadRd_DC', 'AdrshSt_DC', 'Nagaram_D', 'Krishna_D',
   'Rynapadu_H', 'HydRoad_DC', 'Ragvendr_D', 'NSTRoad_I',
   'BsstdDPP_D', 'HawaiPlr_DC', 'Panchot_IP', 'Bargawan_DC',
   'KGAAirprt_HB', 'keshod_DC', 'NCplxDPP_D', 'SsnRdDPP_D',
   'Mamlatdr_DC', 'BrDbleRd_D', 'Yadvgiri_IP', 'NarsipuraRd_D',
   'Narasipr_D', 'SulthnRd_D', 'Jogeshwri_L', 'JublierD_D',
   'Thomas_D', 'BegurRD_D', 'TirupDPP_D', 'Santalpr_D', 'AtoNgrRd_I',
   'Gajuwaka_IP', 'LaxmiNgr_D', 'NrdaawDPP_D', 'Trnsport_H',
   'Central_H_1', 'UdhamNgr_H', 'Kundli_H', 'Patelfli_D',
   'Rohini_DPC', 'KtsiGrsm_D', 'ARBNorth_DC', 'Pngktgudi_D',
   'Thalthru_DC', 'Sttyapar_D', 'Vasanthm_I', 'Bypasrd_D',
   'Mohan_Nagar_DPC', 'Madhavaram_L', 'MotvdDPP_D', 'Vaghasi_IP',
   'NlgaonRd_D', 'Srwnsngr_D', 'Aswningr_I', 'Sec_02_DPC', 'Dakor_DC',
   'GandhiRd_D', 'EastmnRD_D', 'VketRoad_D', 'Thirumtr_IP',
   'NJVNgr_D', 'SelamRd_D', 'GMukrDPP_D', 'GariDPP_D', 'Central_I_1',
   'MP_Nagar', 'Phaphamu_DC', 'Porur_DPC', 'Perungudi_DPC',
```

'AkhirDPP_D', 'GModDPP_D', 'IndstlAr_I', 'Raiprvlg_L', 'Jhilmil_L',
'Central_D_2', 'KoilStrt_D', 'Haridwar', 'Nzbadrd_D', 'Xroad_D',
'Devenply_I', 'SuryaDPP_D', 'Dwaraka_D', 'Menagrdn_D', 'JKRoad_D',
'Mayapuri_PC', 'Hoodi_IP', 'NvygRDPP_D', 'CrossRD_D', 'PhrmPlza_D',
'Banikatt_D', 'Sulgwan_D', 'Indsarea_D', 'Dhelu_D', 'UBamddPP_D',
'AchneraRD_D', 'JPNagar_Pc', 'KHRoad_I', 'Maheva_D', 'Chowk_D',
'BkgnRoad_D', 'TahsilRD_D', 'Kishangarh_DPC', 'Kuntikna_H',
'CharRsta_D', 'CottonGreen_DPC', 'CikhliRD_D', 'PunjabiB_L',
'Kengeri_IP', 'Indira_Nagar', 'Peenya_IP', 'Sirikona_H',
'Khandeshwar_Dc', 'ClgRDDPP_D', 'Alwal_L', 'StatonRD_D',
'Pashan_DPC', 'CP', 'KetyDPP_D', 'OstwlEmp_D', 'BaljiDPP_D',
'Khenewa_D', 'Mhbhirab_D', 'RSRoad_D', 'Balajicly_I', 'Artmclny_D',
'MGRoad_D', 'SDKNgr_D', 'TirupthiRd_D', 'Bngisheb_D', 'Sector63_L',
'DataSagr_D', 'Govndsgsr_D', 'BljiMrkt_D', 'Bnnrghta_L',
'Beliaghata_DPC', 'Airport_H', 'Lake_Avenue_DPC', 'East',
'Memnagar', 'Mumbra_DC', 'Satellite', 'Pakrela_D', 'Auliayapr_D',
'Ulhasngr_DC', 'Pawane_L', 'Kalyan_West_Dc', 'Bbganj_I',
'Central_H_2', 'Mthurard_L', 'New_Alipore_DPC', 'RPC', 'Peenya_L',
'Lovely_D', 'Mataward_D', 'PatelWrd_D', 'Kappalur_H',
'Central_D_12', 'Rkcomplx_DC', 'Erode', 'Chrompet_L',
'Central_D_9', 'KhirByps_I', 'Agraroad_I', 'Katrmira_D',
'Sudmpuri_D', 'Jaipur', 'Chrompet_DPC', 'SamathBv_D', 'ward9_D',
'Kuslpram_I', 'VadaiDPP_D', 'HudcoDPP_D', 'Ward14_D', 'AmvdiDPP_D',
'Nalasopara', 'Sixmile', 'Chandmari', 'Sardala_D', 'VarunCly_DC',
'Banikhet_D', 'Bangotu_D', 'Chuanpur_I', 'RatanDPP_D',
'krshnPly_DC', 'BllvMarg_D', 'Central_D_10', 'AstrdDPP_D',
'MhptraRD_D', 'NgrNigam_DC', 'Egmore_DPC', 'Nangli_IP',
'Vadgaon_Sheri_DPC', 'Karayam_H', 'JNPT_D', 'Lajpat_IP',
'Madhavaram_DPC', 'Mdhvvpur_D', 'RTOroad_D', 'ABPath_D',
'Hillcard_DC', 'Samyaprm_D', 'Sec-83_DC', 'Blbgarh_DC', 'Manesar',
'ICDCant_D', 'B_RPC', 'HosptlRd_D', 'Bhgtpura_D', 'Wardno5_D',
'WrdN4DPP_D', 'Tetultol_D', 'GndhiNgr_IP', 'FoySGRRD_I',
'Nayagaon_I', 'Basni_I', 'DKLogDPP_D', 'JawaharN_D', 'RoshnBgh_I',
'SamitiRd_D', 'Kundli_P', 'TrtlldiaRD_L', 'GtRoad_D', 'RjghatRd_D',
'HrihrNgr_I', 'TownDPP_D', 'Madaprum_D', 'Thmpulam_D',
'Kalynpur_I', 'Bypassrd_D', 'HnumnDPP_D', 'Raghogrh_D', 'StRoad_D',
'Pandrnga_I', 'CivilHPL_D', 'Srikot_D', 'CGRoad_D', 'WageDPP_D',
'Hejunagr_D', 'MissonRd_D', 'JangiRd_D', 'Kaithwal_D',
'Bsavangr_D', 'Adargchi_IP', 'Jogshwri_I', 'GndhiNGR_D',
'Cdosclrd_D', 'GadagRD_D', 'Sector4_D', 'Govind_D', 'Parasi_D',
'Waidhan_D', 'Ward8DPP_D', 'ArtoDPP_D', 'Panipat', 'Khajuria_I',

'SriRmNGR_D', 'GuttalRD_D', 'Mangri_I', 'Sector1A_IP',
'Shamshbd_P', 'Dankuni_P', 'Markndpr_D', 'Udyabata_D',
'Kapleswr_D', 'Okhla_PC', 'Nangli_L', 'Kakdepot_D', 'Kurduwdi_D',
'Oilmilrd_D', 'Tiglgndi_D', 'LICOffce_D', 'SubhVRTL_I', 'HUB',
'Patparganj_DPC', 'Chndivli_PC', 'KSCLny_DC', 'Vidygiri_D',
'Vardhard_D', 'Mankoli_GW', 'Chrompet_PC', 'Bilaspur_P',
'Pandesra_Gateway', 'MhmodNgr_D', 'Dehradun', 'Ramvlg_D',
'HnmntNgr_D', 'SH78_D', 'Soghra_D', 'PnditNGR_D', 'ChndrNgr_D',
'Robinson_D', 'Poothole_D', 'VaniThtr_D', 'IOTCEncl_L',
'Tolichwk_I', 'Central_I_3', 'Karelibaug_DPC', 'Bgtwthck_D',
'BypassRD_D', 'Kapshera_L', 'RIICO_L', 'Koliplm_I', 'Sirjudin_D',
'PunjbiPd_D', 'Sarubali_D', 'YuktiDPP_D', 'Rcocmplx_D',
'AnprnDPP_D', 'Nagar_DC', 'Katira_D', 'Rozapar_D', 'Jantaclg_D',
'Diakkawn_D', 'Bankura_D', 'Salanpur_D', 'Talkui_D', 'BhgyaNgr_D',
'ZebaTWR_D', 'North_R_8', 'Aurangabad', 'NagarDPP_D', 'MaxDPP_D',
'BtaiRoad_D', 'Kollgpra_D', 'ShivmDPP_D', 'MhrjaDPP_D',
'WdN14DPP_D', 'Ward17_D', 'Atapaka_D', 'GvrCompx_D', 'ArhamDPP_D',
'GndhiDPP_D', 'PeroorRD_D', 'DindirD_D', 'VidyaNGR_D',
'Mundhawa_L', 'BoiDPP_D', 'SrifoDPP_D', 'Kntgorya_D', 'NditaDPP_D',
'DohalDPP_D', 'GopalDPP_D', 'KrsprDPP_D', 'UtBzrDPP_D',
'SavtaHTL_D', 'Hindcwk_D', 'GrmNgriya_D', 'HoliCDPP_D',
'MdothdRD_D', 'kalibari_D', 'Alngjuri_D', 'RicMilRd_D',
'Bhogpur_D', 'AkhraBzr_D', 'PlaceCol_D', 'TahurDPP_D',
'Anaipeta_D', 'Bardivan_D', 'Wardno6_D', 'ByePass_D', 'MohanPrk_D',
'SChwkDPP_D', 'SagarDPP_D', 'Gaurkshn_I', 'LxmiNiws_D', 'Mundhe_D',
'Mahad_D', 'Shantanu_D', 'LSRoad_DC', 'VallaDPP_D', 'ColctrOf_D',
'SikriKla_DC', 'Meerut', 'Wazirpur_L', 'Gangjala_D', 'SadrHsptl_D',
'Krishnpr_D', 'Wardn13_D', 'Bazar_D', 'Ward9DPP_D', 'SngihiRD_D',
'Durma_D', 'Sishumdr_D', 'Srnvsngr_D', 'Enkndl_a_D', 'YTRd_D',
'ShtDRDPP_D', 'RKComplx_D', 'VikrmMah_D', 'KirtiNgr_D',
'DvlalDPP_D', 'Rohtak', 'Patiala', 'GainMrkt_L', 'Darbhanga',
'Jawahar_D', 'Nagar_D', 'MuruPost_D', 'patna_D', 'GodamDPP_D',
'Vijayght_D', 'Shyndco_D', 'AgrohDPP_D', 'SchdvDPP_D', 'SaiNgr_D',
'GMManu_D', 'Madnpali_D', 'RTCStand_D', 'Gurudwar_D', 'AjmhwdPP_D',
'BhukrdPP_D', 'Mwalibad_D', 'MithmdRd_D', 'Subhash_D',
'Sookhtal_D', 'Kumud_D', 'SuzkiSrv_D', 'PushPlza_D', 'NravnDPP_D',
'LalBagh_D', 'CCRoad_D', 'Banjaria_D', 'Matriprm_IP', 'Subshngr_D',
'KasyaDPP_D', 'East_I_21', 'BypassRd_D', 'PchpkrrD_D',
'Panderia_D', 'TrnsptNGR_D', 'Hatpada_D', 'Dudhani_D', 'Bhaleti_D',
'BpassDPP_D', 'Ward6_D', 'Fathuluh_D', 'KotdwrrD_D', 'Dayanand_D',
'NaginaRD_D', 'Nrsampt_D', 'Yellanda_D', 'HunterRd_I', 'Ganesh_D',

'KdidmCLY_D', 'Khwsrrai_D', 'Khjurwli_DC', 'Harop_D', 'OnkarDPP_D',
'BajprDPP_D', 'Vaishali_D', 'BhwniGnj_D', 'MubarDPP_D',
'PnchmDPP_D', 'Kosmi_D', 'RajaBzr_D', 'KairiyaT_D', 'Mohim_D',
'EBroad_D', 'Bidar', 'Datatrya_D', 'StatonRd_D', 'Verpatem_D',
'RailGate_D', 'Shankrpa_D', 'Kalol_DC', 'Chandkheda_Dc',
'MheshNGR_D', 'PODPP_D', 'AskNagar_D', 'KarnalRd_D', 'DmodrNGR_D',
'IndraCln_D', 'MndwrRod_D', 'NWclyDPP_D', 'HnsChowk_D',
'Ward6DPP_D', 'NavldiDPP_D', 'Pilani', 'JdswarRD_D', 'Kalyanpr_D',
'StnRoad_DC', 'KlngrDPP_D', 'MrenTirh_D', 'Tejpal_I', 'Krvnkuzy_D',
'Pariplyy_D', 'NrainaRD_D', 'NarenaRD_D', 'ChomuRD_D',
'Nishangr_D', 'Wardno13_D', 'Sarswati_D', 'Wardno4_D', 'Purbari_D',
'GTRoad_D', 'SJRoad_D', 'Kanongyn_D', 'DcntCLY_D', 'Arlumodu_D',
'Mutvila_D', 'MhraChng_D', 'AryaNagr_D', 'Kharghar_D',
'NdiaTola_D', 'Pbroad_DC', 'ZuariNgr_IP', 'Goa', 'HapurRD_D',
'Swargash_D', 'WardNo4_D', 'Haripur_D', 'KamnHbRD_I', 'GunjRDPP_D',
'Aravind_D', 'HanumDPP_D', 'Hitech_D', 'TKRoad_D', 'Kumrpurm_D',
'MunplDPP_D', 'Gobindgarh_DC', 'Town_D', 'Amankovl_D',
'Klskrpt_D', 'PostofJN_D', 'KrantiNgr_D', 'MIDCAvdn_I',
'DhuleRd_D', 'DhuleRoad_D', 'Nandrbar_D', 'Mahindra_D',
'Khar West _Dc', 'RajCmplx_D', 'Balaji Nagar', 'Peedika_H',
'BhadgDPP_D', 'KolheDPP_D', 'BChwkDPP_D', 'Parigi_D', 'KanpurRd_D',
'StatinRD_D', 'Rawatpur_D', 'JilRDDPP_D', 'Kotwali_D', 'Mlkpura_D',
'GayatriN_D', 'Arulimod_D', 'Ajnari_D', 'NehruNGR_D', 'Court_D',
'GovndNgr_DC', 'barkarRd_D', 'Bokule_H', 'JyotiNgr_D',
'BrlwgDPP_D', 'Chikdply_I', 'Naraynpr_D', 'ModelTwn_P', 'Ward19_D',
'SttinDPP_D', 'PalikDPP_D', 'Padra_D', 'WebelDPP_D', 'Itachnda_D',
'Pshimpra_D', 'Psthrjhr_D', 'BOB_D', 'Truptingr_D', 'Virar_DC',
'Skynet_INT', 'Shop3DPP_D', 'Vepmpttu_DC', 'IndEstat_I',
'Paschim_DC', 'LB-Nagar_Dc', 'CotnGren_M', 'MiraRoad_M',
'ShivBari_D', 'Vadodara', 'ArkonmRD_D', 'Central_DPP_4',
'Chakan_D', 'Mdiclclly_D', 'ColageRD_D', 'Umalodge_D', 'SriDPP_D',
'WardNo3_D', 'Shillong', 'NamoNagr_D', 'ShsmldPP_D', 'FatehpRd_I',
'PBRDDPP_D', 'KotwaliN_D', 'MduraiRD_D', 'Kacheri_D', 'KhandDPP_D',
'Jamalpur_D', 'BgwriDPP_D', 'SmClyDPP_D', 'Madarpur_D',
'Nirjanpur_L', 'PonaniRD_D', 'ManhrBld_D', 'DumDum_DPC',
'KaaduRd_D', 'Blmrgnst_D', 'KrthiKyn_D', 'StationRD_D', 'Satara_D',
'BstndDPP_D', 'KmkshBul_D', 'Prbhtngr_D', 'Kakrmath_D',
'Beltnngdi_D', 'MarketRd_D', 'Veersagr_I', 'Pazhvedu_D',
'Pshrikvu_D', 'ZamQuatr_D', 'Mandodi_D', 'IdstrlAr_D',
'Thiruviz_D', 'Vadasari_D', 'Poondi_D', 'VaiklsRT_D', 'MukkuRD_D',
'Fairybnk_D', 'kalmpuza_D', 'nagar_D', 'Badeplyy_D', 'SH71_D',

'Sbrmnprm_D', 'HsptlRod_D', 'Mrthndpr_D', 'Puduvalvu_D',
'Konapara_D', 'CmtNgRod_D', 'AnnaNGR_D', 'goplpurm_D',
'Mthrapuri_D', 'Kmrajngr_D', 'Chithbrm_D', 'PriyrNGR_D',
'Pinjore_DC', 'AmtlaDPP_D', 'JhumanCk_D', 'Msstreet_DC',
'Krsnakcl_D', 'Kovil_D', 'AshkTalk_D', 'Ward25_D', 'TherSRT_D',
'Ukkadam_D', 'RtlamNka_D', 'Sahni_D', 'Palikval_D', 'Ward7DPP_D',
'Srvdyckw_D', 'VijywdRD_D', 'BhogdDPP_D', 'MuthpTmp_D', 'BMRd_D',
'Shanthy_D', 'Vijdurg_D', 'Ambedkar_D', 'Sunku_D', 'KolarRd_D',
'FshryOFC_D', 'Artclgrd_D', 'Venkatsa_DC', 'Tuminkte_D',
'PlsrdDPP_D', 'ViksClny_D', 'Barout_D', 'Wardno10_D', 'ThvIrsrt_D',
'Palladam_DC', 'Techrcly_D', 'Wardno7_D', 'Brplicwk_D',
'GangDPP_D', 'Banshkri_DC', 'HousngBd_D', 'NH117_D', 'Arsprmbu_D',
'Chnglptu_DC', 'Achipkam_D', 'Sudimala_D', 'Perkadrd_D',
'Radhaprm_D', 'AsrplmRd_DC', 'VdkkuSrt_I', 'MrktYrd_DC',
'Rajula_DC', 'SriVnktpm_D', 'UdnkdiRD_D', 'Shnmgprm_D',
'VidyaNgr_D', 'SKRoad_D', 'Uppal_L', 'BaraLoha_D', 'Barmasia_D',
'D', 'AmbedDPP_D', 'LaxmiNGR_D', 'Kataram_D', 'JwahrNGR_D',
'Srirampt_D', 'BnglorRd_D', 'Greenmkt_D', 'KndlIDPP_D',
'Pdmavati_D', 'JydevNGR_D', 'CivlLine_D', 'ChngiDPP_D',
'RjndrNgr_DC', 'PiliKoti_D', 'Gurukrpa_D', 'TBCross_D',
'CollgeRD_D', 'North', 'GwhRDDPP_D', 'Thiruvlr_DC', 'TmpleSrt_D',
'Vllyaprm_D', 'RamnadRD_D', 'ThrbadRD_D', 'Rawlgaon_D',
'Malegaon_D', 'South_D_12', 'Central_D_7', 'Wrd12DPP_D',
'Shahapur_D', 'HBColny_D', 'Ponda_Dc', 'RajpurRD_D', 'NorprRD_D',
'ThanaDPP_D', 'Salem', 'PC', 'EragnDPP_D', 'Antop Hill',
'AadiDPP_D', 'TrnptNgr_L', 'Trimulgerry_Dc', 'Panvel_D',
'Mangalam_D', 'ShantiNg_D', 'Viveka_DC', 'MJRDPP_D', 'Samarth_D',
'BaliaMod_D', 'MhliaDPP_D', 'Ganeshwr_D', 'KatlaDPP_D',
'Trmltmpl_D', 'kankroli_D', 'Mehmdpur_P', 'Hanmkond_D',
'RadhaCpx_D', 'MndiRoad_D', 'Mohnprwa_D', 'Ward11_D', 'Mnanthla_H',
'ShjnprRD_D', 'Eaglvari_D', 'PhdofDPP_D', 'Kelasahi_D',
'Kdthdstrt_D', 'TnhbBlkC_D', 'VagaiNgr_D', 'LxmntDPP_D',
'BhmrdDPP_D', 'Ameenpur_I', 'KdrShrRd_D', 'AlathurRD_D',
'Thsil3PL_D', 'KKndrDPP_D', 'BhunaDPP_D', 'LFRoad_D', 'RgvdrDPP_D',
'MandyarD_D', 'Mlydpthr_D', 'Davisdle_D', 'HajiprRD_D',
'MnBzrDPP_D', 'Talaiya_D', 'NharuExt_D', 'NngrgnRd_D',
'SrnnprHwy_D', 'Margao_Dc', 'Tejpali_M', 'Valluvar_D', 'RLSTNDPP_D',
'War5DPP_D', 'Dilliyan_D', 'BhrolDPP_D', 'NaginaRd_D',
'Sitarmrd_D', 'Kadugodi_D', 'Mahuva_DC', 'Shahdara', 'KakaCplx_D',
'ConduDPP_D', 'Pothredy_D', 'APMCYard_D', 'VasaviNg_D', 'Kalyan',
'Barwala', 'Central_D_5', 'PaikjNGR_D', 'Chaitnya_D', 'AsnsdhRD_D',

'GndhiNgr_D', 'KhdimDPP_D', 'RPRoad_D', 'Swamylyt_D', 'Kidwai_D',
'NagpurRd_D', 'DelRdDPP_D', 'ColegRd_D', 'Shekhpur_D',
'MngalDPP_D', 'PrmNrDPP_D', 'BhwanDPP_D', 'MohnVRTL_D',
'MohanNgr_C', 'PigonDPP_D', 'Chpaguri_D', 'Manikndm_H',
'Palakrty_D', 'PreetDPP_D', 'Kothapet_D', 'ChtrGIDC_IP', 'Delhi',
'ITICollg_L', 'KisanCo_D', 'LdnunDPP_D', 'Mhdiptnm_C',
'Shivangr_D', 'KnsgraRD_D', 'Udupi', 'Sector02_C', 'Bomsndra_PC',
'Vijayawada', 'Chndrlpd_D', 'AkkolRD_D', 'MnbzrDPP_D',
'Munduprm_D', 'EmsPnmbi_D', 'BSarani_D', 'Bareilly', 'WardNo1_D',
'SadarHPL_D', 'Karnal', 'BgnprDPP_D', 'Mughlpra_D', 'SainkSCL_D',
'Sohagpur_D', 'Sholinganallur_Dc', 'MdhsnDPP_D', 'Kadipur',
'Kothuru_D', 'Wardno8_D', 'BaruaRd_D', 'Potheri', 'ChrliDPP_D',
'Chatrpr_DC', 'MBTRd_DC', 'Sadras_D', 'AzmrddDPP_D', 'LNBRoad_D',
'Shop2DPP_D', 'DeVDPP_D', 'GagiDPP_D', 'BodomBzr_DC', 'HotelPrk_D',
'Mainrd_D', 'Pandriba_L', 'SnthiNGR_D', 'CroslySRT_D',
'BypRDDPP_D', 'JrjolDPP_D', 'UmarDPP_D', 'Mharajpr_D',
'Cnsrvila_D', 'Gurkhari_D', 'East_L_23', 'JthriDPP_DC',
'CtyLgDPP_D', 'SadulDPP_D', 'PedakRd_P', 'RailwyRd_D',
'Solaiprm_D', 'Bhabua_D', 'Chukhndi_D', 'Todapur_DC',
'VislkNgr_DC', 'SaiTempl_D', 'NcsRd_DC', 'Nullipad_D',
'NapitDPP_D', 'MrdiVlge_D', 'LohiadDPP_D', 'PakriDPP_D',
'BisnolDPP_D', 'Gopa3PL_D', 'Nehru3PL_D', 'MnRDDPP_D', 'Bahreya_I',
'ThthiCwk_D', 'HSR_Layout_PC', 'Sracomplx_D', 'Katghara_D',
'Mhimapur_D', 'Nimachrd_D', 'Kdvantra_D', 'Chtrpuza_D',
'Veluthur_D', 'Greens_D', 'SurbhiTh_D', 'Shivprsad_D',
'BawliDPPP_D', 'Rathnam_D', 'HesglDPP_D', 'TonkRoad_D',
'MSRClgRd_D', 'Shivalya_D', 'JatnidDPP_D', 'Mangol_DC',
'Pthrgoan_D', 'Chrwpaty_D', 'HelipadRD_D', 'Farmnala_D',
'ShbdnDPP_D', 'Ranakant_D', 'Chmpmura_I', 'North_D_3',
'ShubsNGR_D', 'Mrtrklgr_D', 'AwmpiVng_D', 'HunthrVg_I', 'Feroke_H',
'Pringla_D', 'SantaNGR_D', 'Bazarrd_D', 'JalnaRd_D', 'GhtimDPP_D',
'IndraNgr_D', 'RmNyrDPP_D', 'HsnRdDPP_D', 'Puthalam_D',
'Agraharm_DC', 'Dehrird_D', 'BasthDPP_D', 'SubrtDPP_D',
'FatprDPP_D', 'Nijgan_D', 'East_I_20', 'KtnRdDPP_D', 'Mapusa',
'Pnjbiyon_D', 'Patel Nagar', 'East_D_8', 'Rjnldrngr_D', 'Old City',
'YashDPP_D', 'Aliganj', 'Agra', 'Nerul_D', 'Murtingr_D',
'PcrrdDPP_D', 'BsStdDPP_D', 'SashPhkn_D', 'KalikDPP_D',
'Lakshmi_D', 'Whitefld_L', 'Lngrguda_D', 'DMComDPP_D',
'Pnchlght_D', 'KDRoad_D', 'NavdaCln_D', 'KaimgnjRD_D',
'farukngr_D', 'CroadDPP_D', 'RatuaDPP_D', 'JivanDPP_D',
'BhmpRdPP_D', 'Vishakhapatnam', 'GoalpDPP_D', 'MlnprDPP_D',

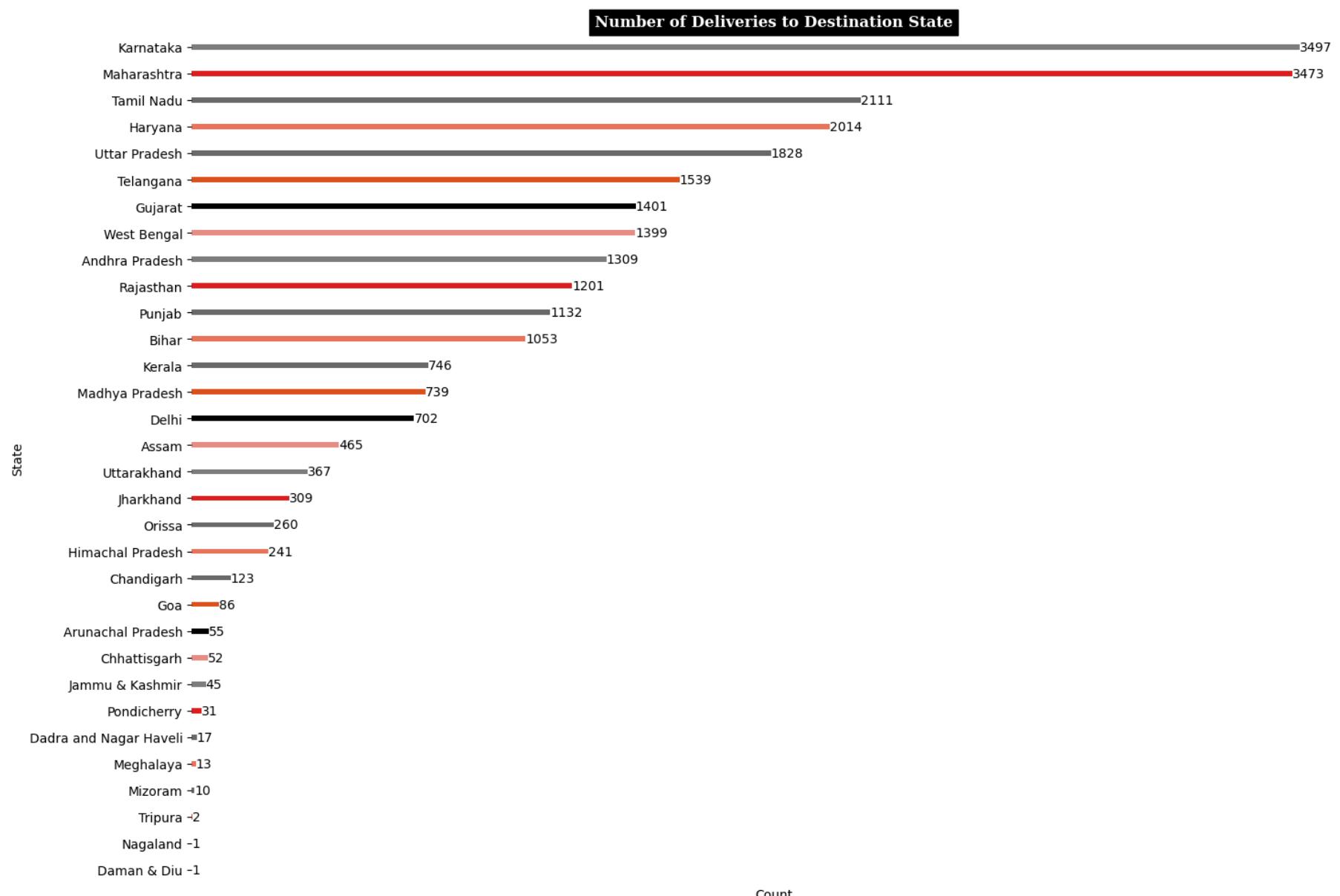
```
'TiloiDPP_D', 'Wardnor4_D', 'Ldthlabh_D', 'MahmurmGj_IP',
'KamalDPP_D', 'PuranDPP_D', 'Cherukole_D', 'Bhainpura_D',
'MnimlaRd_D', 'KeRoad_D', 'MsjidDPP_D', 'KarjuDPP_D', 'VidyaDPP_D',
'KoralDPP_D', 'Majoor_D', 'Bhaluahi_D', 'MsmcyDPP_D', 'KcharaRD_D',
'Cochin_L', 'CourtDPP_D', 'SirsaDPP_D', 'PtrkBunk_D', 'NatunDPP_D',
'Enayetpr_D', 'BsnoiHPL_D', 'Idgah_P', 'Chandigarh', 'ColnyDPP_D',
'Pettah_D', 'Mylapore', 'NaturDPP_D', 'UttarDPP_D', 'MdnpnDPP_D',
'Kanakpur_D', 'Salap_DC', 'Idgah_L', 'Kolar Mandakni', 'SohnaRd_D',
'NagplDPP_D', 'KeranDPP_D', 'BnkrGate_D', 'SndbrDPP_D',
'RoopNgr_D', 'Jharia_DC', 'Jabalpur', 'NadthiCx_D', 'Sangetha_D',
'Ramnagar_D', 'WamanDPP_D', 'DiyoDPP_D', 'JJCpxDPP_D',
'RwStnDPP_D', 'Ricco_D', 'Poonamallee_L', 'Umargaon_DC',
'BalibDPP_D', 'Kadtempty_D', 'KotyamRD_D', 'MemariRD_D',
'Srirmpur_D', 'Central_L_8', 'FulbaDPP_D', 'SukntDPP_D',
'Kondapur_D', 'VUNagar_DC', 'Prjapati_D', 'EtwahDPP_D',
'UdaynDPP_D', 'DibngVly_D', 'GndhiChk_D', 'Muktsar_D',
'Ghansoli_DC', 'Kovaipudur_Dc', 'Lajwanti', 'Central_H_4',
'BreezeDPP_D', 'Rjndpara_D', 'BazarDPP_D', 'MotiDPP_D',
'MrgnjDPP_D', 'Jaripatk_DC', 'Sarjapur_D', 'Kothanur_L',
'AshkngRd_D', 'JiswlDPP_D', 'West_Dc', 'BargaDPP_D', 'SbhRDDPP_D',
'Parai_D', 'KhsmiDPP_D', 'ChatiDPP_D', 'Bulabeda_D', 'ChtwrDPP_D',
'RhmgjDPP_D', 'Kooriyad_D', 'Bhandup West_Dc', 'GMndiDPP_D',
'GurpdDPP_D', 'Kaura_D', 'Bnsibtla_D', 'MjlprDPP_D', 'Varanasi',
'TilakNgr_D', 'Manchar_D', 'Old', 'ShantiDPP_D', 'Royapuram',
'DivrsnRd_D', 'SliprDPP_DC', 'RajRdDPP_D', 'KrisnKunj_D',
'ShivaDPP_D', 'East_D_7', 'Rajpura_D', 'SingCLNY_D', 'CBRoad_D',
'Sidrd_D', 'Krusphrma_D', 'Karelibaug_DC', 'RgstrOFC_D',
'Sanpada_CP', 'Bhilai_DC', 'Nattukal_D', 'AnadiDPP_D',
'ArickDPP_D', 'VrdhriRD_D'], dtype=object)
```

In [86]: `de['destination_state'].unique()`

Out[86]: array(['Haryana', 'Uttar Pradesh', 'Karnataka', 'Punjab', 'Maharashtra',
 'Tamil Nadu', 'Gujarat', 'Delhi', 'Andhra Pradesh', 'Telangana',
 'Rajasthan', 'Madhya Pradesh', 'Assam', 'West Bengal',
 'Chandigarh', 'Dadra and Nagar Haveli', 'Orissa', 'Uttarakhand',
 'Bihar', 'Jharkhand', 'Pondicherry', 'Goa', 'Himachal Pradesh',
 'Kerala', 'Arunachal Pradesh', 'Mizoram', 'Chhattisgarh',
 'Jammu & Kashmir', 'Meghalaya', 'Nagaland', 'Tripura',
 'Daman & Diu'], dtype=object)

```
In [87]: state_counts = de['destination_state'].value_counts().to_frame().reset_index()
state_counts.columns = ['State', 'Count']

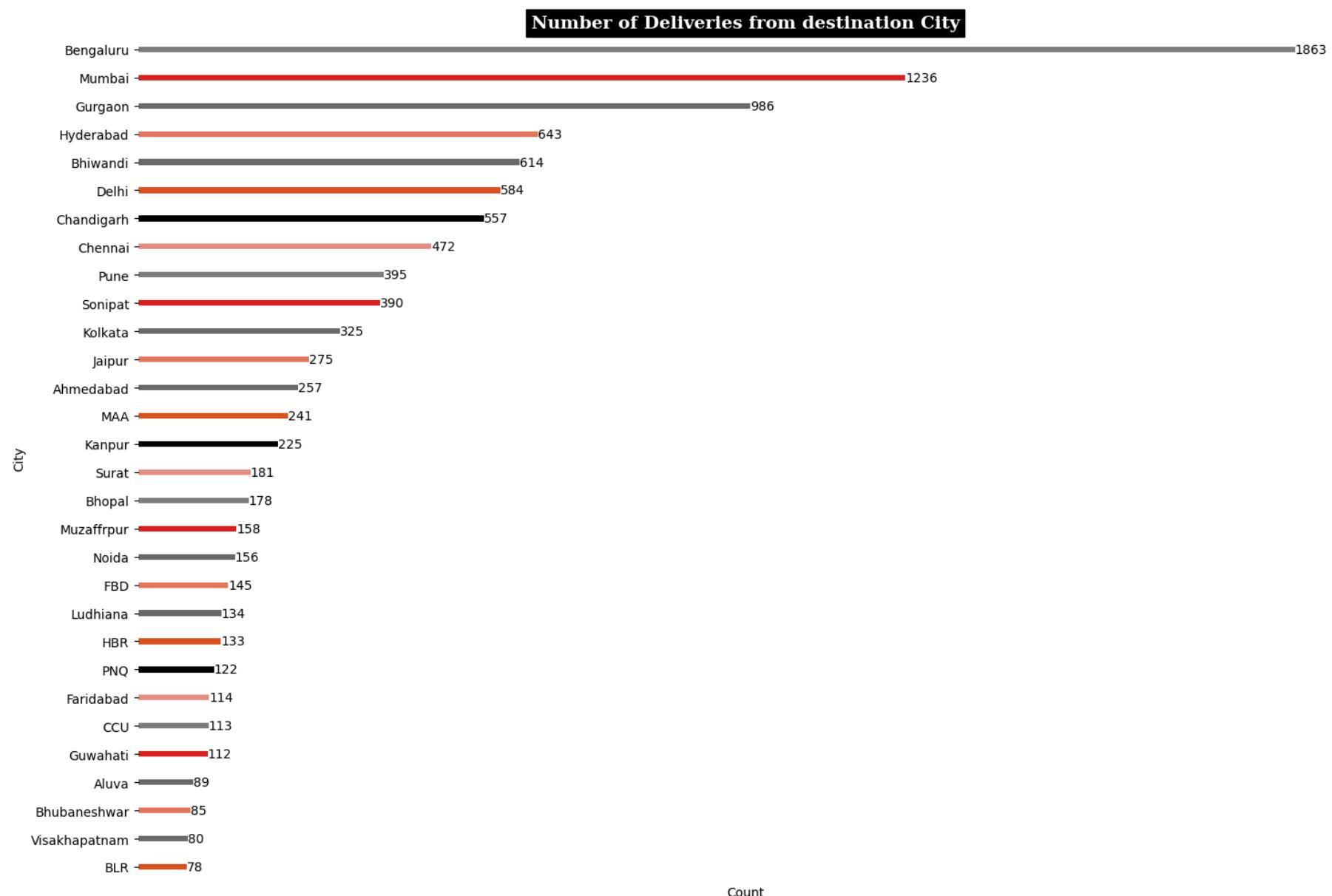
plt.figure(figsize=(15,10))
a = sns.barplot(y='State', x='Count', data=state_counts, palette=cp, width=0.2)
a.bar_label(a.containers[0], label_type='edge')
plt.xticks([])
plt.ylabel('State')
plt.xlabel('Count')
plt.title('Number of Deliveries to Destination State', fontsize=12, fontfamily='serif', fontweight='bold', backgroundcolor='k', col
plt.tight_layout()
sns.despine(bottom=True, left=True)
plt.show()
```



```
In [88]: city_counts = de['destination_city'].value_counts().to_frame().reset_index()[:30]
city_counts.columns = ['City', 'Count']

plt.figure(figsize=(15,10))
```

```
a = sns.barplot(y='City', x='Count', data=city_counts, palette=cp, width=0.2)
a.bar_label(a.containers[0], label_type='edge')
plt.xticks([])
plt.ylabel('City')
plt.xlabel('Count')
plt.title('Number of Deliveries from destination City', fontsize=14, fontfamily='serif', fontweight='bold', backgroundcolor='k', color='white')
plt.tight_layout()
sns.despine(bottom=True, left=True)
plt.show()
```



 **Insights:**

Destination State

- States like **Karnataka, Maharashtra, Tamil Nadu, Haryana**, and **Uttar Pradesh** where maximum packages are received in this month indicating significant engagement.

Destination City

- Cities like **Bengaluru, Mumbai, Gurgaon, Bhiwandi, Hyderabad, Delhi** where the major no.of booking are received.

```
In [89]: np.set_printoptions(threshold=np.inf)
```

```
In [90]: de['corridor'] = de['source_name'] + ' <--> ' + de['destination_name']
de['corridor'].value_counts()
```

```
Out[90]: corridor
Bangalore_Nelmngla_H (Karnataka) <--> Bengaluru_KGAirprt_HB (Karnataka)      151
Bangalore_Nelmngla_H (Karnataka) <--> Bengaluru_Bomsndra_HB (Karnataka)      127
Bengaluru_Bomsndra_HB (Karnataka) <--> Bengaluru_KGAirprt_HB (Karnataka)      121
Bengaluru_KGAirprt_HB (Karnataka) <--> Bangalore_Nelmngla_H (Karnataka)      108
Pune_Tathawde_H (Maharashtra) <--> Bhiwandi_Mankoli_HB (Maharashtra)          107
...
Ongole_SubhVRTL_I (Andhra Pradesh) <--> Kandukur_LICOffce_D (Andhra Pradesh)    1
Madnapalle_PngnrRd_D (Andhra Pradesh) <--> Palamaner_Lakshmi_D (Andhra Pradesh) 1
Dharmavram_SaiNgr_D (Andhra Pradesh) <--> Kadiri_GVManu_D (Andhra Pradesh)      1
Baharampur_Chuanpur_I (West Bengal) <--> Chapra_NagarDPP_D (West Bengal)     1
Jaipur_NgrNigam_DC (Rajasthan) <--> Jaipur_Central_D_1 (Rajasthan)           1
Name: count, Length: 2741, dtype: int64
```

```
In [91]: corridor_counts = de['corridor'].value_counts()[:50]

plt.figure(figsize=(18,12))
#corridor_counts.plot(kind='line', marker='d', color='r')
sns.lineplot(y=corridor_counts.index, x=corridor_counts.values, marker='d', color='r')
plt.title('Value Counts of Corridor', fontsize=20, fontfamily='serif', fontweight='bold', backgroundcolor='dimgray', color='w')
plt.ylabel('Corridor')
plt.xlabel('Count')
plt.tight_layout()
sns.despine()
plt.grid(True)

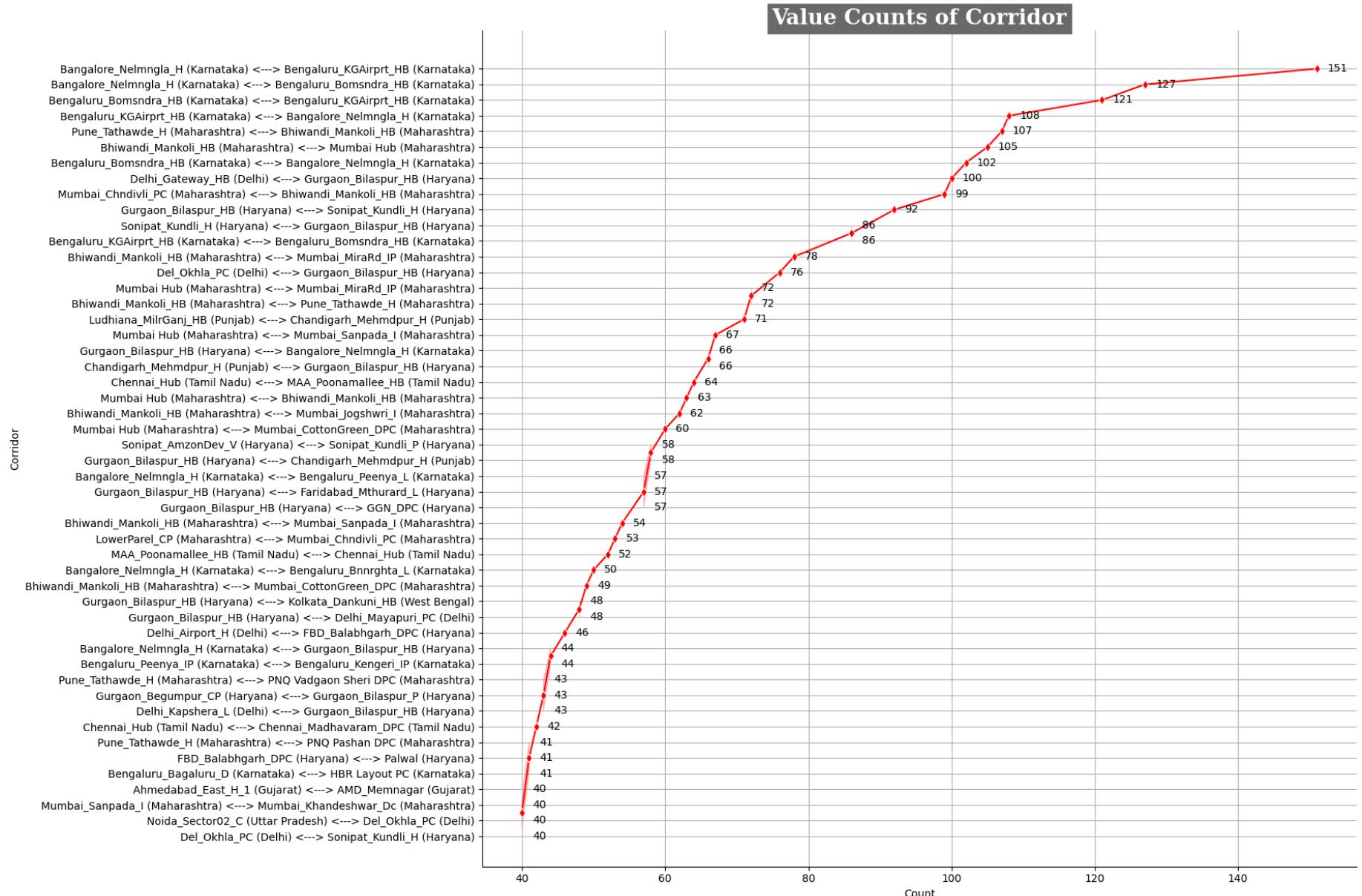
for i, count in enumerate(corridor_counts.values):
```

```

plt.text(count+1.5, corridor_counts.index[i], str(count), ha='left', va='center')

plt.show()

```



💡 Insights:

- The route between **Bangalore_Nelamangala_H** to **Bengaluru_KGAirport_HB,Bengaluru_Bomsndra_HB** sees the highest package volume, with 151 and 127 packages sent respectively.
- Bengaluru_Bommasandra_HB** to **Bengaluru_KGAirport_HB** is also popular, with 121 packages sent.
- Bengaluru_KGAirport_HB** to **Bangalore_Nelamangala_H** has moderate activity, with 108 packages sent.

4. The data indicates Bengaluru's importance as a transportation hub **Corridor** within ***Karnataka***, handling significant package traffic.

```
In [92]: de['state_corridor'] = de['source_state']+ '--'+de['source_city'] + ' <--> '+ de['destination_state']+ '--'+de['destination_city']
de['state_corridor'].value_counts()
```

```
Out[92]: state_corridor
Karnataka--Bengaluru <--> Karnataka--Bengaluru           1413
Maharashtra--Mumbai <--> Maharashtra--Mumbai             622
Maharashtra--Bhiwandi <--> Maharashtra--Mumbai            512
Maharashtra--Mumbai <--> Maharashtra--Bhiwandi            345
Telangana--Hyderabad <--> Telangana--Hyderabad            316
...
Gujarat--Jetpur <--> Gujarat--Dhoraji                  1
Andhra Pradesh--Anakapalle <--> Andhra Pradesh--Visakhapatnam 1
Andhra Pradesh--Narsiptnm <--> Andhra Pradesh--Anakapalle   1
West Bengal--MirzapurWB <--> West Bengal--Kolkata        1
Uttar Pradesh--Anandnagar <--> Uttar Pradesh--Gorakhpur    1
Name: count, Length: 2302, dtype: int64
```

```
In [93]: state_corridor_counts = de['state_corridor'].value_counts()[:25]

plt.figure(figsize=(18,8))
sns.lineplot(y=state_corridor_counts.index, x=state_corridor_counts.values, marker='s', color='k')
plt.title('Value Counts of state_city Corridor', fontsize=20, fontfamily='serif', fontweight='bold', backgroundcolor='tomato', color='white')
plt.ylabel('Corridor')
plt.xlabel('Count')
plt.tight_layout()
sns.despine()
plt.grid(True)

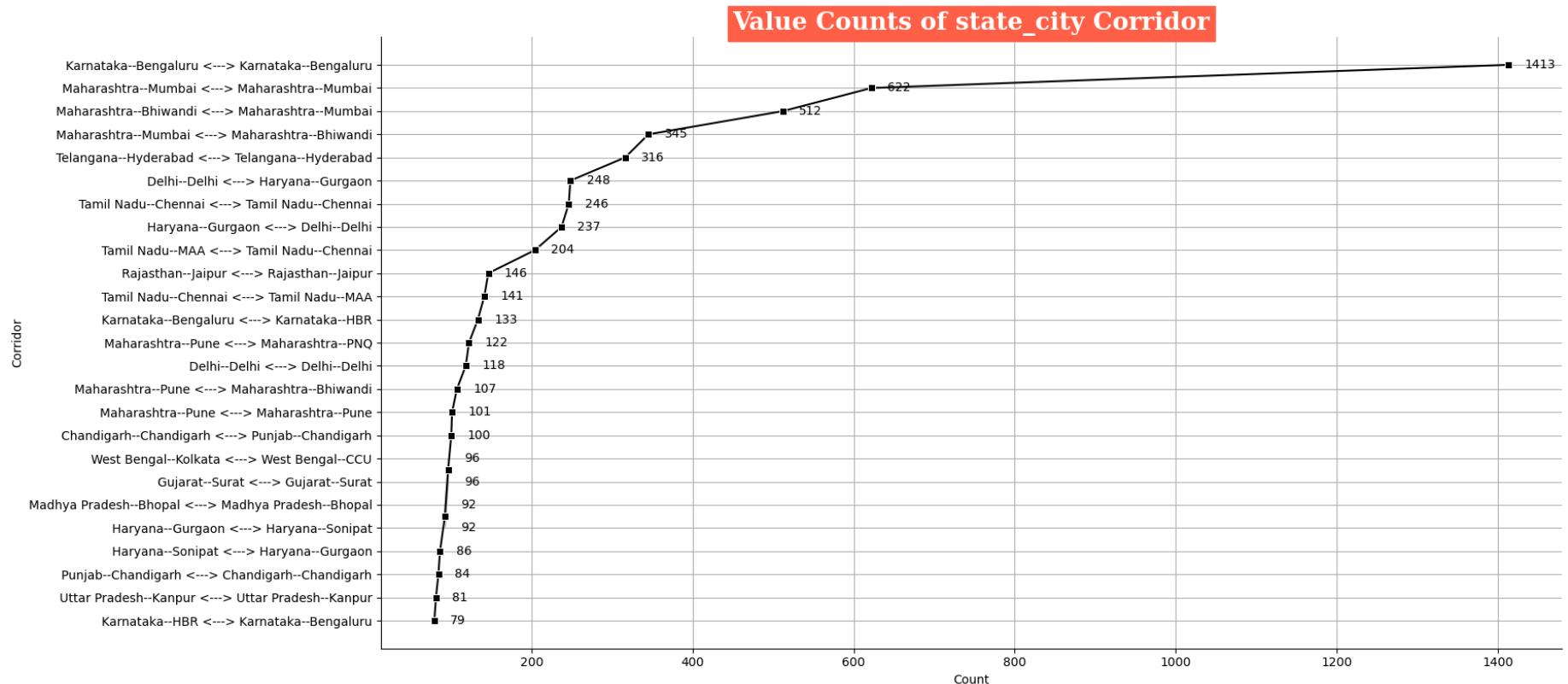
for i, count in enumerate(state_corridor_counts.values):
```

```

plt.text(count+20, state_corridor_counts.index[i], str(count), ha='left', va='center')

plt.show()

```



```

In [94]: de['city_corridor'] = de['source_city']+ '--'+de['source_place'] + ' <--> '+ de['destination_city']+ '--'+de['destination_place']
display(de['city_corridor'].value_counts())

```

```

city_corridor
Bengaluru--Nelmngla_H <---> Bengaluru--KGAirprt_HB      151
Bengaluru--Nelmngla_H <---> Bengaluru--Bomsndra_HB      127
Bengaluru--Bomsndra_HB <---> Bengaluru--KGAirprt_HB      121
Bengaluru--KGAirprt_HB <---> Bengaluru--Nelmngla_H      108
Pune--Tathawde_H <---> Bhiwandi--Mankoli_HB            107
...
Ongole--SubhVRTL_I <---> Kandukur--LICOffce_D          1
Madnapalle--PngnrRd_D <---> Palamaner--Lakshmi_D        1
Dharmavram--SaiNgr_D <---> Kadiri--GVManu_D           1
Baharampur--Chuanpur_I <---> Chapra--NagarDPP_D        1
Jaipur--NgrNigam_DC <---> Jaipur--Central_D_1          1
Name: count, Length: 2741, dtype: int64

```

```

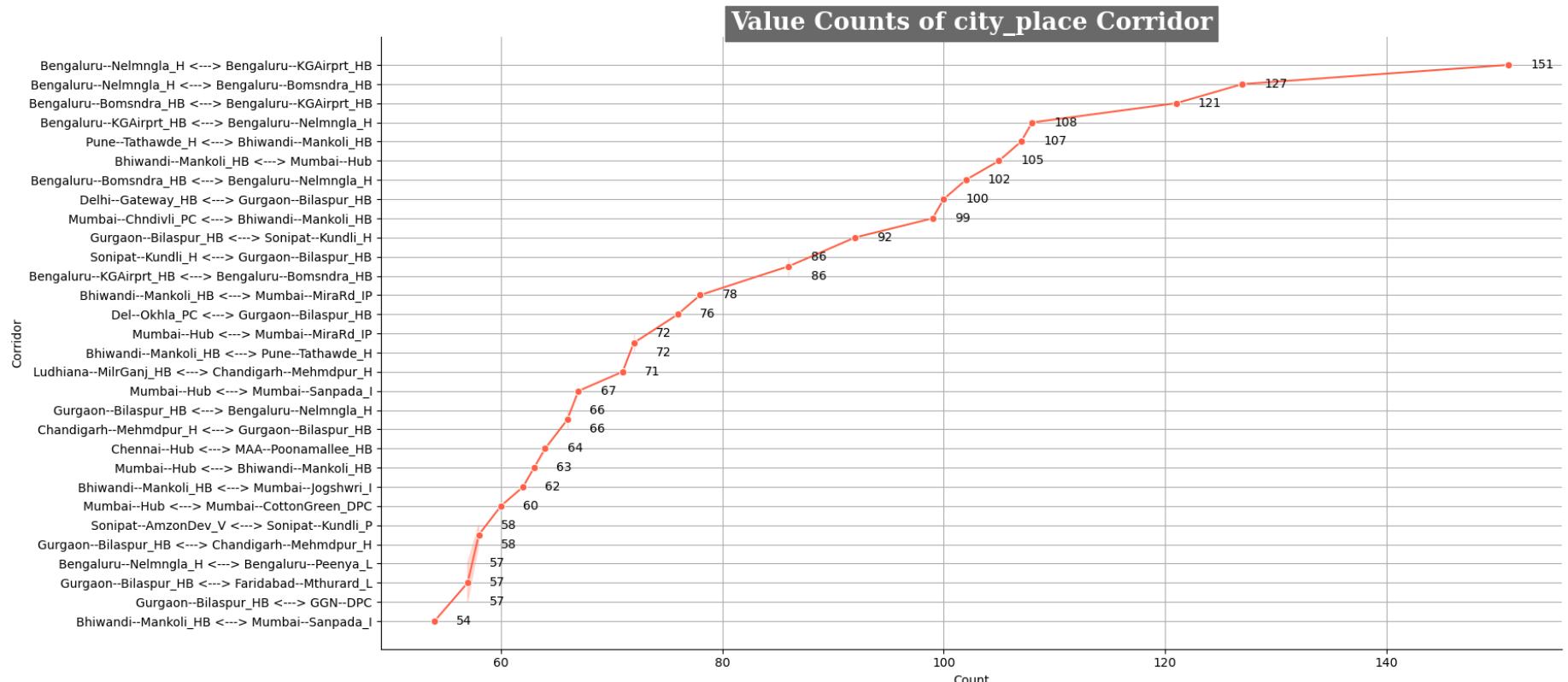
In [95]: city_corridor_counts = de['city_corridor'].value_counts()[:30]

plt.figure(figsize=(18,8))
sns.lineplot(y=city_corridor_counts.index, x=city_corridor_counts.values, marker='o', color='tomato')
plt.title('Value Counts of city_place Corridor', fontsize=20, fontfamily='serif', fontweight='bold', backgroundcolor='dimgray', col
plt.ylabel('Corridor')
plt.xlabel('Count')
plt.tight_layout()
sns.despine()
plt.grid(True)

for i, count in enumerate(city_corridor_counts.values):
    plt.text(count+2, city_corridor_counts.index[i], str(count), ha='left', va='center')

plt.show()

```



💡 Insights:

- Maharashtra, Karnataka, Haryana, and Tamil Nadu serve as key starting and ending locations for delivery services.
- Mumbai, Gurgaon, Delhi, and Bengaluru are major metropolitan centers from where many deliveries originate.
- A large proportion of nationwide deliveries are destined for Mumbai, Bengaluru, Gurgaon, and Delhi.

```
In [97]: # 4. Extracting features like month, year, day, etc. from Trip_creation_time
de['trip_creation_month'] = de['trip_creation_time'].dt.month
de['trip_creation_year'] = de['trip_creation_time'].dt.year
de['trip_creation_day'] = de['trip_creation_time'].dt.day
de['trip_creation_hour'] = de['trip_creation_time'].dt.hour
de['trip_creation_weekday'] = de['trip_creation_time'].dt.weekday
de['trip_creation_week'] = de['trip_creation_time'].dt.isocalendar().week
de
```

Out[97]:

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_i
0	trip-153671041653548748+IND209304AAA+IND000000ACB	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Centra (Uttar Pra
1	trip-153671041653548748+IND462022AAA+IND209304AAA	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Bhopal_Trnsp (Madhya Pra
2	trip-153671042288605164+IND561203AAB+IND562101AAA	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaD (Karna
3	trip-153671042288605164+IND572101AAA+IND561203AAB	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Tumkur_Veers (Karna
4	trip-153671043369099517+IND000000ACB+IND160002AAC	trip-153671043369099517	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur (Har
...
26217	trip-153861115439069069+IND628204AAA+IND627657AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Tirchchndr_Shnmgp (Tamil N
26218	trip-153861115439069069+IND628613AAA+IND627005AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Peikulam_SriVnktp (Tamil N

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_i
26219	trip-153861115439069069+IND628801AAA+IND628204AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Eral_Busstand_D (Karnataka)
26220	trip-153861118270144424+IND583119AAA+IND583101AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Sandur_WrdN1D (Karnataka)
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (Karnataka)

26222 rows × 37 columns

◆ In-Depth Analysis

In [98]: `new_df = de.copy()`In [99]: `new_df.columns`

```
Out[99]: Index(['segment_key', 'trip_uuid', 'data', 'route_type', 'trip_creation_time',
   'source_name', 'destination_name', 'od_start_time', 'od_end_time',
   'start_scan_to_end_scan', 'actual_distance_to_destination',
   'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
   'segment_osrm_time', 'segment_osrm_distance', 'segment_actual_time_sum',
   'segment_osrm_time_sum', 'segment_osrm_distance_sum', 'od_total_time',
   'od_time_diff_hour', 'source_city', 'source_place', 'source_state',
   'destination_city', 'destination_place', 'destination_state',
   'corridor', 'state_corridor', 'city_corridor', 'trip_creation_month',
   'trip_creation_year', 'trip_creation_day', 'trip_creation_hour',
   'trip_creation_weekday', 'trip_creation_week'],
  dtype='object')
```

In [100... new_df.sample(2)

		segment_key	trip_uuid	data	route_type	trip_creation_time	source_nar
3330	153694740553026744+IND574104AAA+IND574216AAA	trip-153694740553026744	trip-153694740553026744	training	Carting	2018-09-14 17:50:05.530477	Karkala_MarketRd. (Karnatakl
3755	153696604384005633+IND507117AAB+IND507303AAA	trip-153696604384005633	trip-153696604384005633	training	FTL	2018-09-14 23:00:43.840397	Manuguru_AskNagar. (Telangana)

In [101... create_trip_dict={

```
'data' : 'first',
'route_type' : 'first',
'od_start_time': 'first',
'od_end_time': 'last',
'od_time_diff_hour' : 'sum',
'trip_creation_time' : 'first',
'trip_creation_month' : 'first',
'trip_creation_year' : 'first',
```

```
'trip_creation_day' : 'first',
'trip_creation_hour' : 'first',
'trip_creation_weekday' : 'first',
'trip_creation_week' : 'first',
'start_scan_to_end_scan' : 'sum',
'actual_distance_to_destination' : 'sum',
'actual_time' : 'sum',
'osrm_time' : 'sum',
'osrm_distance' : 'sum',
'segment_actual_time': 'sum',
'segment_osrm_time': 'sum',
'segment_osrm_distance': 'sum',
'segment_actual_time_sum': 'sum',
'segment_osrm_time_sum': 'sum',
'segment_osrm_distance_sum': 'sum',
'source_name': 'first',
'source_city':'first',
'source_state':'first',
'source_place':'first',
'destination_name': 'first',
'destination_city':'first',
'destination_state':'first',
'destination_place':'first',
'corridor':'first',
'state_corridor':'first',
'city_corridor':'first'
}

trip_agg_df = new_df.groupby('trip_uuid').agg(create_trip_dict).reset_index()
trip_agg_df
```

Out[101...]

		trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creation_mont
0	153671041653548748	trip-training	training	FTL	2018-09-12 16:39:46.858469	2018-09-12 16:39:46.858469	37.668497	2018-09-12 00:00:16.535741	
1	153671042288605164	trip-training	training	Carting	2018-09-12 02:03:09.655591	2018-09-12 02:03:09.655591	3.026865	2018-09-12 00:00:22.886430	
2	153671043369099517	trip-training	training	FTL	2018-09-14 03:40:17.106733	2018-09-14 03:40:17.106733	65.572709	2018-09-12 00:00:33.691250	
3	153671046011330457	trip-training	training	Carting	2018-09-12 00:01:00.113710	2018-09-12 01:41:29.809822	1.674916	2018-09-12 00:01:00.113710	
4	153671052974046625	trip-training	training	FTL	2018-09-12 00:02:09.740725	2018-09-12 03:54:43.114421	11.972484	2018-09-12 00:02:09.740725	
...
14782	153861095625827784	trip-test	test	Carting	2018-10-03 23:55:56.258533	2018-10-04 06:41:25.409035	4.300482	2018-10-03 23:55:56.258533	1
14783	153861104386292051	trip-test	test	Carting	2018-10-03 23:57:23.863155	2018-10-04 00:57:59.294434	1.009842	2018-10-03 23:57:23.863155	1

		trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creation_mont
14784		trip-153861106442901555	test	Carting	2018-10-04 02:51:27.075797	2018-10-04 02:51:27.075797	7.035331	2018-10-03 23:57:44.429324	1
14785		trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	2018-10-04 02:29:04.272194	5.808548	2018-10-03 23:59:14.390954	1
14786		trip-153861118270144424	test	FTL	2018-10-04 03:58:40.726547	2018-10-04 03:58:40.726547	5.906793	2018-10-03 23:59:42.701692	1

14787 rows × 35 columns

In [102...]

```
numerical_columns = trip_agg_df.select_dtypes(include=[np.float32, np.float64])
numerical_columns
```

Out[102...]

	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_tir
0	37.668497	2259.0	824.732849	1562.0	717.0	991.352295	1548
1	3.026865	180.0	73.186905	143.0	68.0	85.111000	141
2	65.572709	3933.0	1927.404297	3347.0	1740.0	2354.066650	3308
3	1.674916	100.0	17.175274	59.0	15.0	19.680000	59
4	11.972484	717.0	127.448502	341.0	117.0	146.791794	340
...
14782	4.300482	257.0	57.762333	83.0	62.0	73.462997	82
14783	1.009842	60.0	15.513784	21.0	12.0	16.088200	21
14784	7.035331	421.0	38.684837	282.0	48.0	58.903702	28
14785	5.808548	347.0	134.723831	264.0	179.0	171.110306	258
14786	5.906793	353.0	66.081528	275.0	68.0	80.578705	274

14787 rows × 12 columns



In [103...]

numerical_columns.describe().T

Out[103...]

		count	mean	std	min	25%	50%	75%	max
	od_time_diff_hour	14787.0	8.840187	10.978880	0.391024	2.494975	4.661846	10.558962	131.642533
	start_scan_to_end_scan	14787.0	529.429016	658.254395	23.000000	149.000000	279.000000	632.000000	7898.000000
	actual_distance_to_destination	14787.0	164.090195	305.502808	9.002461	22.777099	48.287895	163.591255	2186.531738
	actual_time	14787.0	356.306000	561.517761	9.000000	67.000000	148.000000	367.000000	6265.000000
	osrm_time	14787.0	160.990936	271.459229	6.000000	29.000000	60.000000	168.000000	2032.000000
	osrm_distance	14787.0	203.887405	370.565460	9.072900	30.756900	65.302795	206.644203	2840.081055
	segment_actual_time	14787.0	353.059174	556.364441	9.000000	66.000000	147.000000	364.000000	6230.000000
	segment_osrm_time	14787.0	180.511597	314.678741	6.000000	30.000000	65.000000	184.000000	2564.000000
	segment_osrm_distance	14787.0	222.705444	416.845642	9.072900	32.578850	69.784203	216.560608	3523.632324
	segment_actual_time_sum	14787.0	353.059174	556.364441	9.000000	66.000000	147.000000	364.000000	6230.000000
	segment_osrm_time_sum	14787.0	180.511597	314.678741	6.000000	30.000000	65.000000	184.000000	2564.000000
	segment_osrm_distance_sum	14787.0	222.705444	416.845642	9.072900	32.578850	69.784203	216.560608	3523.632324

In [104...]

```
trip_agg_df.describe(include = object).T
```

Out[104...]

	count	unique	top	freq
trip_uuid	14787	14787	trip-153671041653548748	1
source_name	14787	930	Gurgaon_Bilaspur_HB (Haryana)	1052
source_city	14787	713	Bengaluru	1700
source_state	14787	29	Maharashtra	2714
source_place	14787	788	Bilaspur_HB	1052
destination_name	14787	1042	Gurgaon_Bilaspur_HB (Haryana)	745
destination_city	14787	851	Bengaluru	1633
destination_state	14787	32	Maharashtra	2569
destination_place	14787	866	Bilaspur_HB	745
corridor	14787	1737	Bangalore_Nelmngla_H (Karnataka) <---> Bengaluru	151
state_corridor	14787	1366	Karnataka--Bengaluru <---> Karnataka--Bengaluru	1333
city_corridor	14787	1737	Bengaluru--Nelmngla_H <---> Bengaluru--KGAirport	151

In [105...]

```
trip_df = trip_agg_df.copy()
```

In [106...]

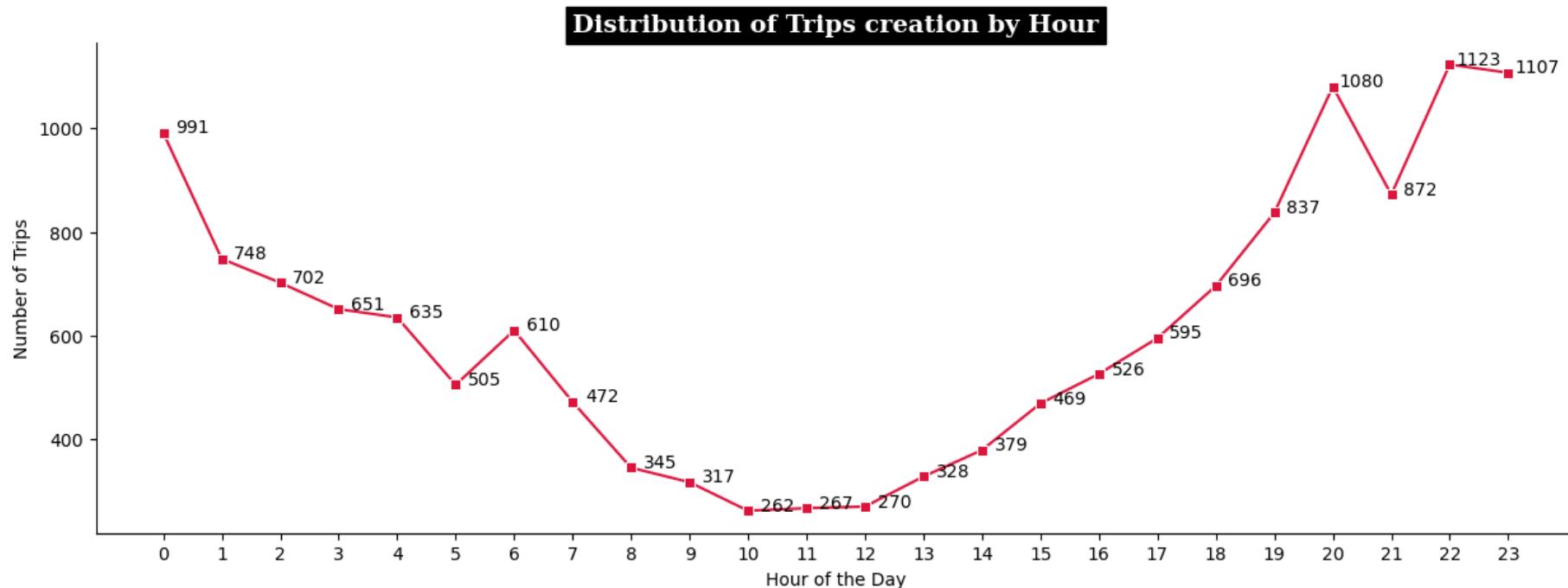
```
trip_creation_by_hour = trip_df.groupby(by='trip_creation_hour')['trip_uuid'].count().reset_index()

plt.figure(figsize=(15,5))
sns.lineplot(data=trip_creation_by_hour, x='trip_creation_hour', y='trip_uuid', marker='s', color='crimson')
plt.xticks(np.arange(0, 24))

for i, count in enumerate(trip_creation_by_hour['trip_uuid']):
    plt.text(trip_creation_by_hour['trip_creation_hour'][i]+0.5, count, count, ha='center')

plt.title('Distribution of Trips creation by Hour', fontsize=14, fontfamily='serif', fontweight='bold', backgroundcolor='k', color='white')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Trips')
```

```
sns.despine()
plt.show()
```



In [107...]: `trip_df.sample()`

Out[107...]:

	trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creation_month
7917	trip-153765583943264913	training	Carting	2018-09-22 22:37:19.432896	2018-09-22 23:19:42.965506	0.706537	2018-09-22 22:37:19.432896	9

In [108...]: `trip_df.trip_creation_year.value_counts()`

```
Out[108... trip_creation_year  
2018    14787  
Name: count, dtype: int64
```

```
In [109... trip_df.trip_creation_month.value_counts()
```

```
Out[109... trip_creation_month  
9      13011  
10     1776  
Name: count, dtype: int64
```

```
In [110... trip_df['trip_creation_month'].value_counts(normalize = True) * 100
```

```
Out[110... trip_creation_month  
9      87.98945  
10     12.01055  
Name: proportion, dtype: float64
```

```
In [111... trip_df.trip_creation_week.value_counts()
```

```
Out[111... trip_creation_week  
38     5001  
39     4402  
37     3608  
40     1776  
Name: count, dtype: Int64
```

```
In [112... trip_df.trip_creation_weekday.value_counts(ascending=True)
```

```
Out[112... trip_creation_weekday  
6      1753  
0      1980  
1      2035  
4      2057  
3      2103  
5      2128  
2      2731  
Name: count, dtype: int64
```

```
In [113... trip_df['trip_creation_day_week'] = trip_df['trip_creation_time'].dt.day_name()
```

```
In [114... trip_df.trip_creation_day.value_counts()
```

```
Out[114... trip_creation_day
```

```
18    791  
15    783  
13    750  
12    747  
21    740  
22    740  
17    722  
14    712  
20    703  
25    695  
26    683  
19    674  
24    658  
27    650  
23    631  
3     627  
16    616  
28    605  
29    605  
1     600  
2     549  
30    506
```

```
Name: count, dtype: int64
```

```
In [123... trip_df.sample()
```

```
Out[123...      trip_uuid      data route_type      od_start_time      od_end_time      od_time_diff_hour      trip_creation_time      trip_creation_month
```

	trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creation_month
8403	153772987770593762	trip-training	FTL	2018-09-24 00:14:22.226774	2018-09-24 02:42:00.976996	3.847563	2018-09-23 19:11:17.706293	9

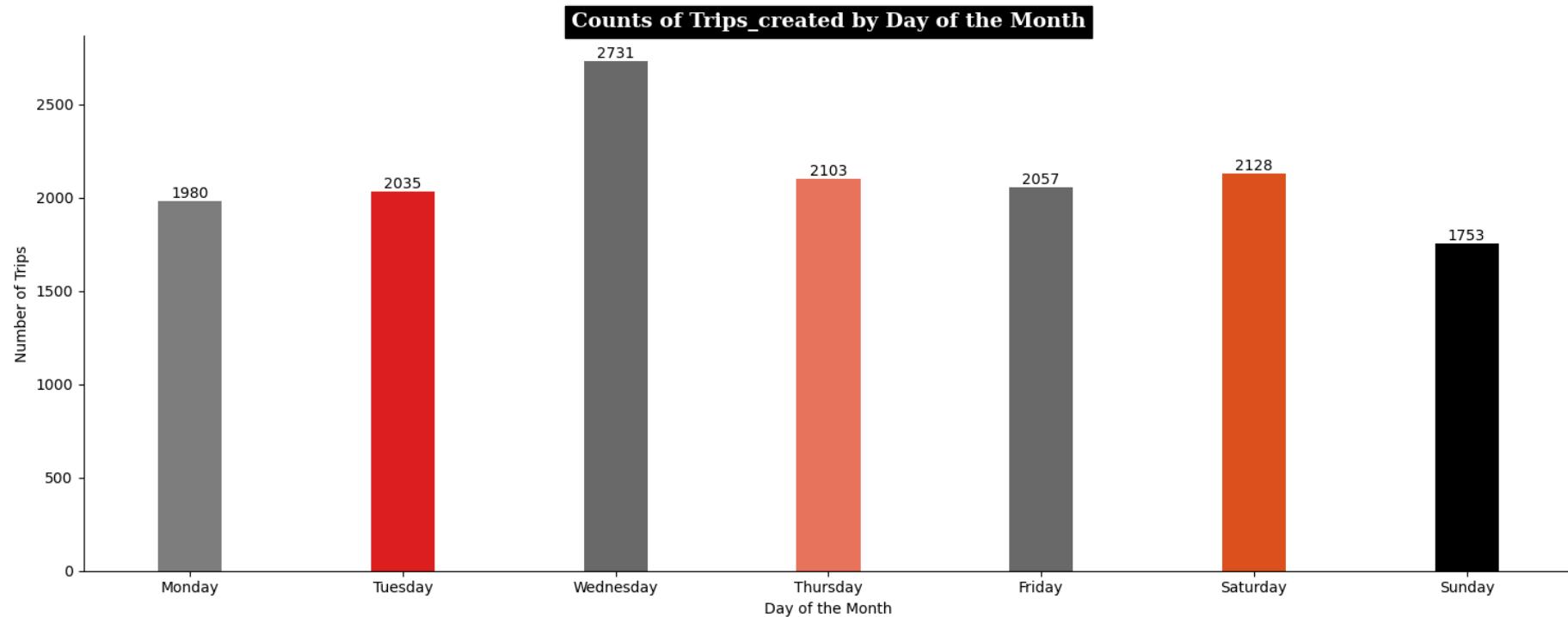


In [125...]

```
plt.figure(figsize=(15,6))

weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_counts = trip_df['trip_creation_day_week'].value_counts().reindex(weekday_order)

sns.barplot(x=day_counts.index, y=day_counts.values, palette=cp, width=0.3)
for i, count in enumerate(day_counts.values):
    plt.text(i, count, str(count), ha='center', va='bottom')
plt.title('Counts of Trips_created by Day of the Month', fontsize=14, fontfamily='serif', fontweight='bold', backgroundcolor='k', color='white')
plt.xlabel('Day of the Month')
plt.ylabel('Number of Trips')
plt.tight_layout()
sns.despine()
plt.show()
```



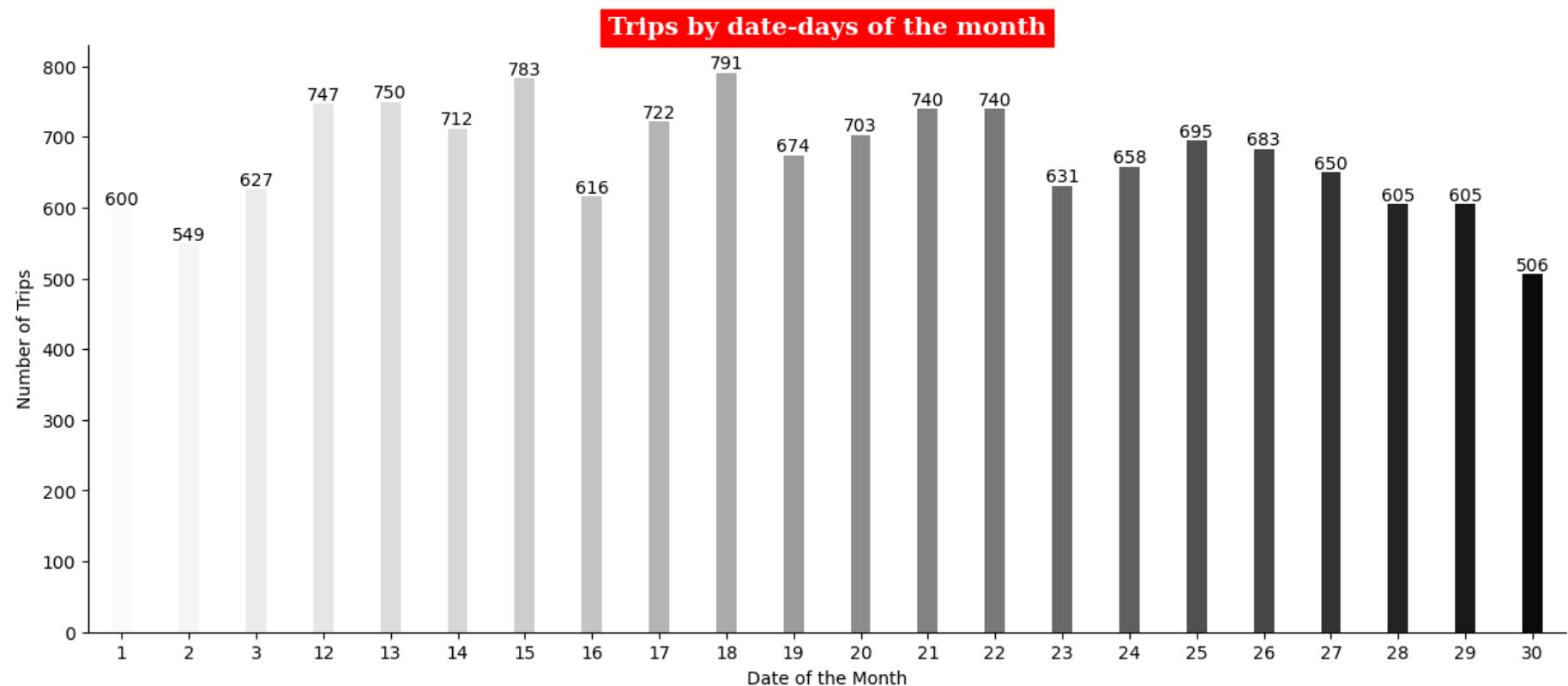
In [126...]

```
trip_df['trip_creation_dayofdate'] = trip_df['trip_creation_time'].dt.day
```

In [127...]

```
trips_by_dateday = trip_df.groupby(by = 'trip_creation_dayofdate')['trip_uuid'].count().to_frame().reset_index()

plt.figure(figsize = (15, 6))
sns.barplot(data = trip_df,x = trips_by_dateday['trip_creation_dayofdate'],y = trips_by_dateday['trip_uuid'], palette='Greys',w
for i, count in enumerate(trips_by_dateday['trip_uuid']):
    plt.text(i, count, str(count), ha='center', va='bottom')
plt.title('Trips by date-days of the month',fontsize=14,fontfamily='serif',fontweight='bold',backgroundcolor='r',color='w')
plt.xlabel('Date of the Month')
plt.ylabel('Number of Trips')
sns.despine()
plt.show()
```



Outlier treatment

In [128...]

numerical_columns

Out[128...]

	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time
--	-------------------	------------------------	--------------------------------	-------------	-----------	---------------	---------------------

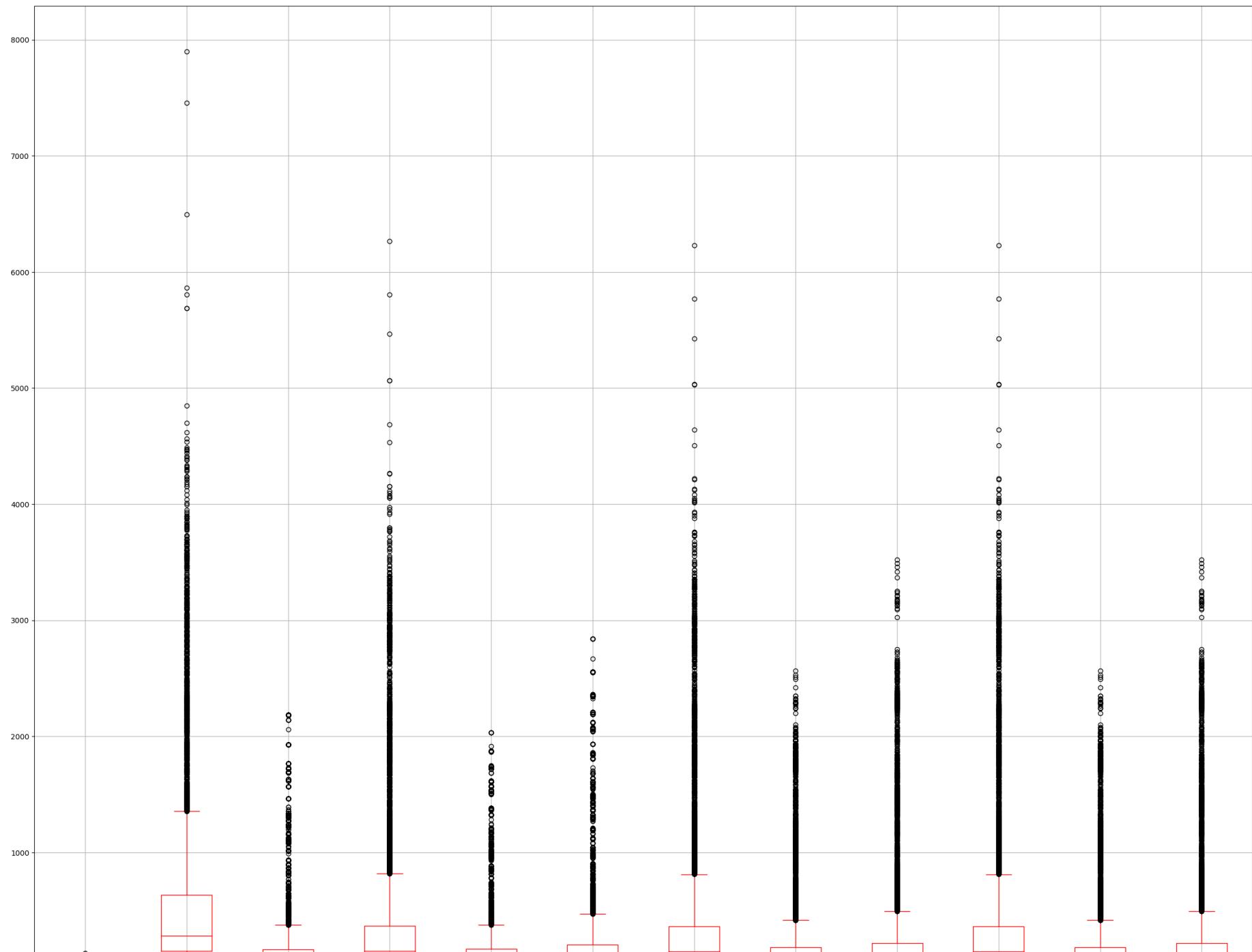
0	37.668497	2259.0	824.732849	1562.0	717.0	991.352295	1548
1	3.026865	180.0	73.186905	143.0	68.0	85.111000	141
2	65.572709	3933.0	1927.404297	3347.0	1740.0	2354.066650	3308
3	1.674916	100.0	17.175274	59.0	15.0	19.680000	59
4	11.972484	717.0	127.448502	341.0	117.0	146.791794	340
...
14782	4.300482	257.0	57.762333	83.0	62.0	73.462997	82
14783	1.009842	60.0	15.513784	21.0	12.0	16.088200	21
14784	7.035331	421.0	38.684837	282.0	48.0	58.903702	28
14785	5.808548	347.0	134.723831	264.0	179.0	171.110306	258
14786	5.906793	353.0	66.081528	275.0	68.0	80.578705	274

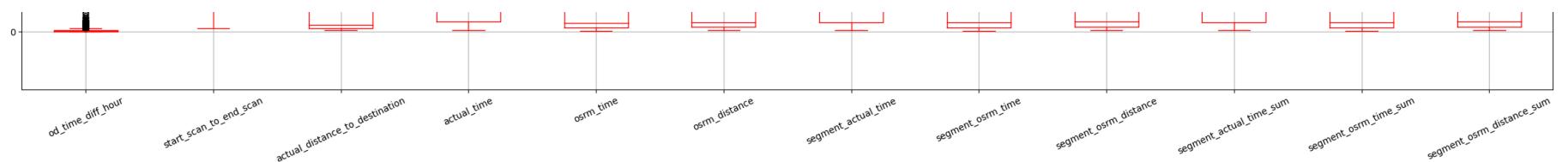
14787 rows × 12 columns



In [129...]

```
plt.figure(figsize=(30, 25))
numerical_columns.boxplot(rot=25, figsize=(35,20), color = 'r')
plt.grid('off')
plt.show()
```





In [130...]: `numerical_columns.columns`

Out[130...]: `Index(['od_time_diff_hour', 'start_scan_to_end_scan', 'actual_distance_to_destination', 'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time', 'segment_osrm_time', 'segment_osrm_distance', 'segment_actual_time_sum', 'segment_osrm_time_sum', 'segment_osrm_distance_sum'], dtype='object')`

In [131...]: `num_cols = numerical_columns.columns.tolist()`
`num_cols`

Out[131...]: `['od_time_diff_hour', 'start_scan_to_end_scan', 'actual_distance_to_destination', 'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time', 'segment_osrm_time', 'segment_osrm_distance', 'segment_actual_time_sum', 'segment_osrm_time_sum', 'segment_osrm_distance_sum']`

In [132...]: `# obtain the first quartile`
`Q1 = numerical_columns.quantile(0.25)`

`# obtain the third quartile`
`Q3 = numerical_columns.quantile(0.75)`

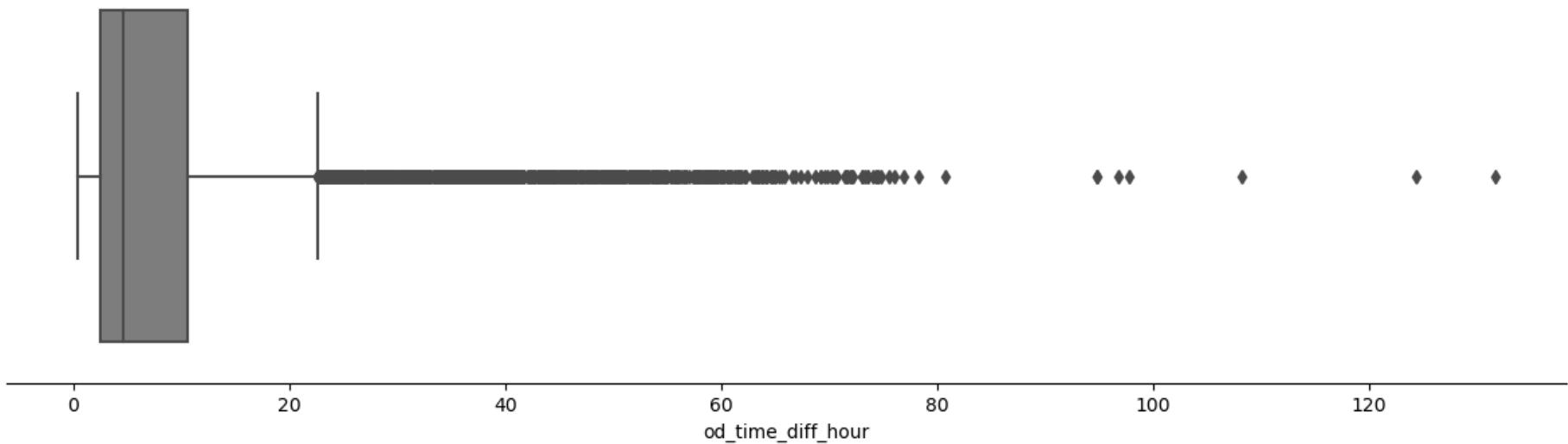
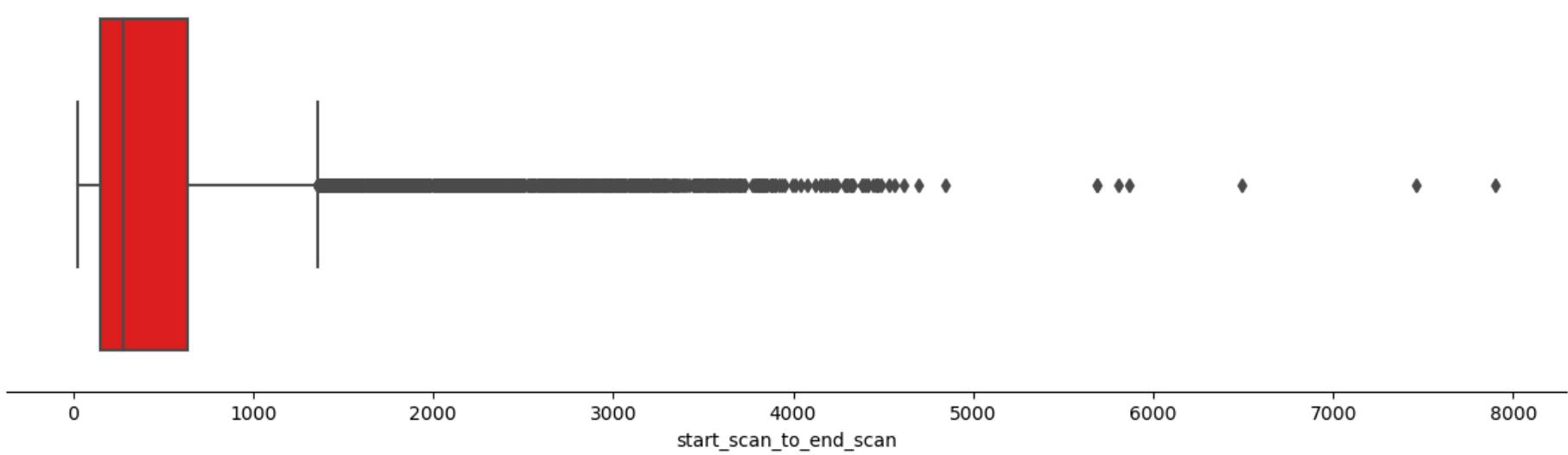
`# obtain the IQR`
`IQR = Q3 - Q1`

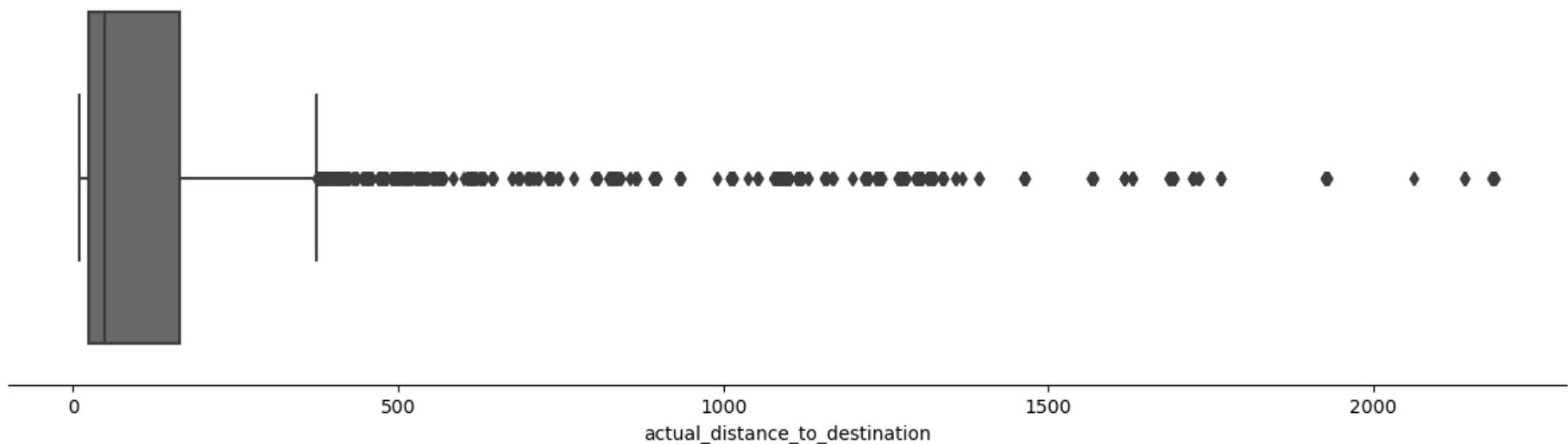
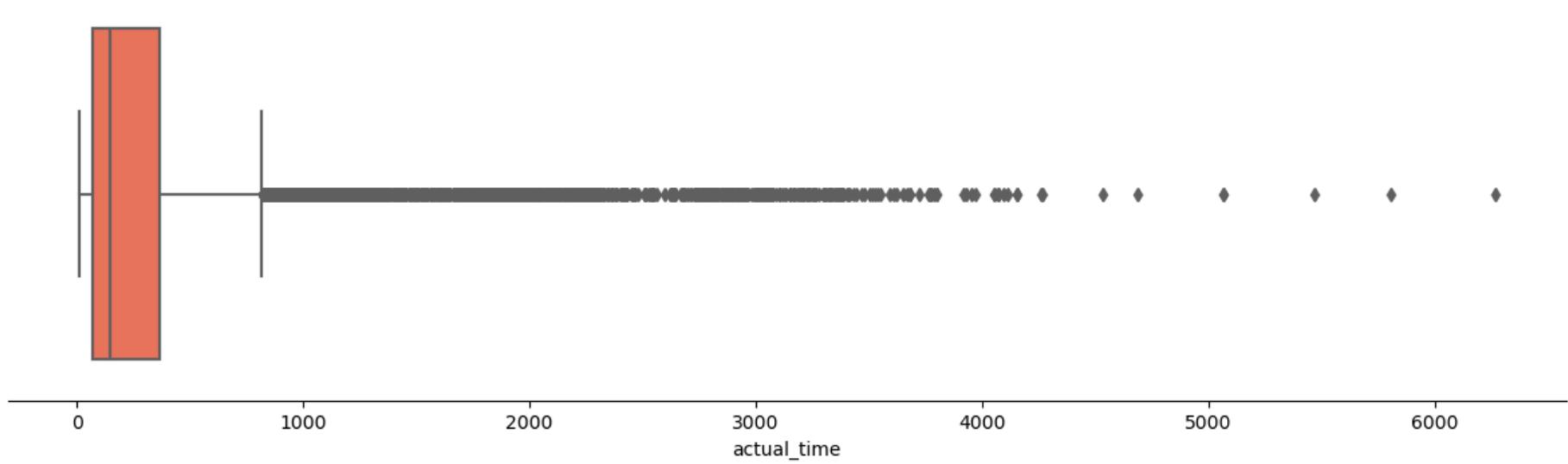
```
# print the IQR
print(IQR)
```

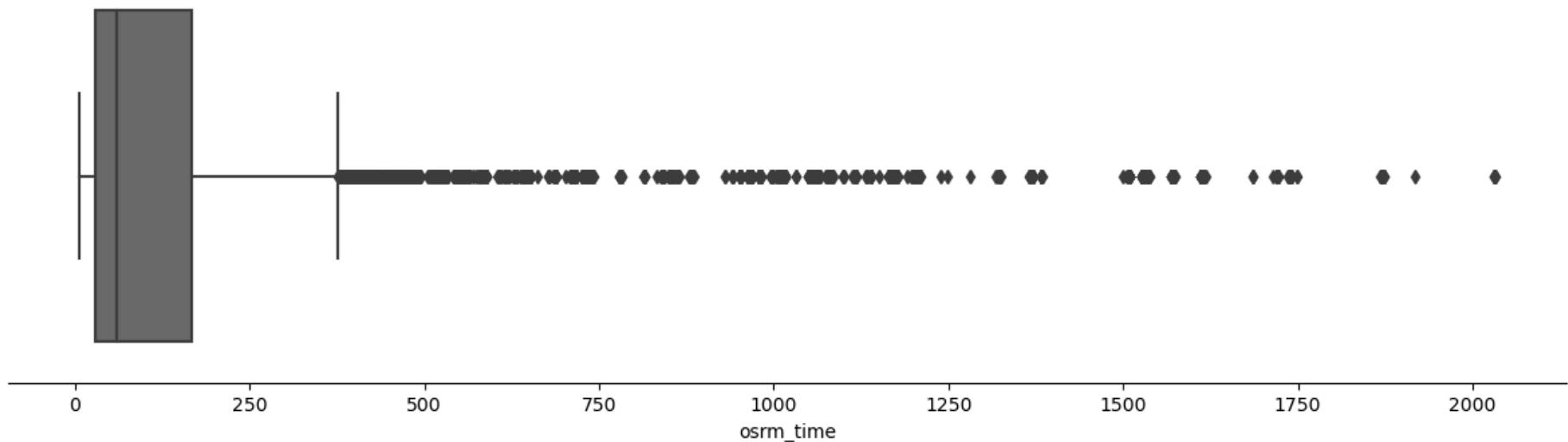
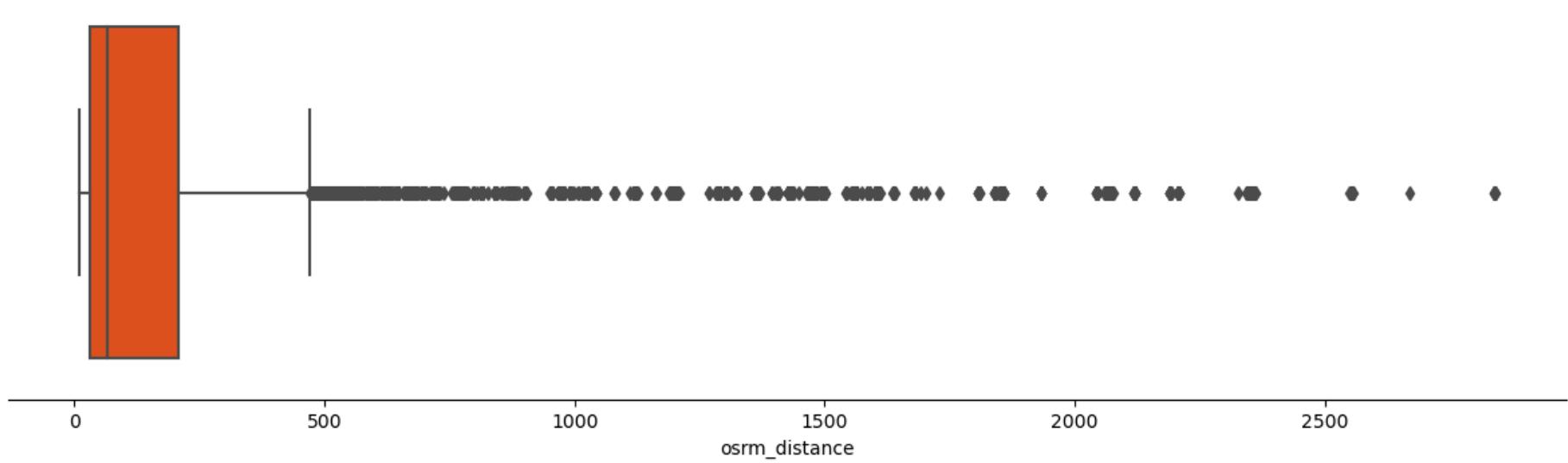
```
od_time_diff_hour           8.063987
start_scan_to_end_scan      483.000000
actual_distance_to_destination 140.814157
actual_time                 300.000000
osrm_time                   139.000000
osrm_distance                175.887303
segment_actual_time         298.000000
segment_osrm_time           154.000000
segment_osrm_distance        183.981758
segment_actual_time_sum     298.000000
segment_osrm_time_sum       154.000000
segment_osrm_distance_sum   183.981758
dtype: float64
```

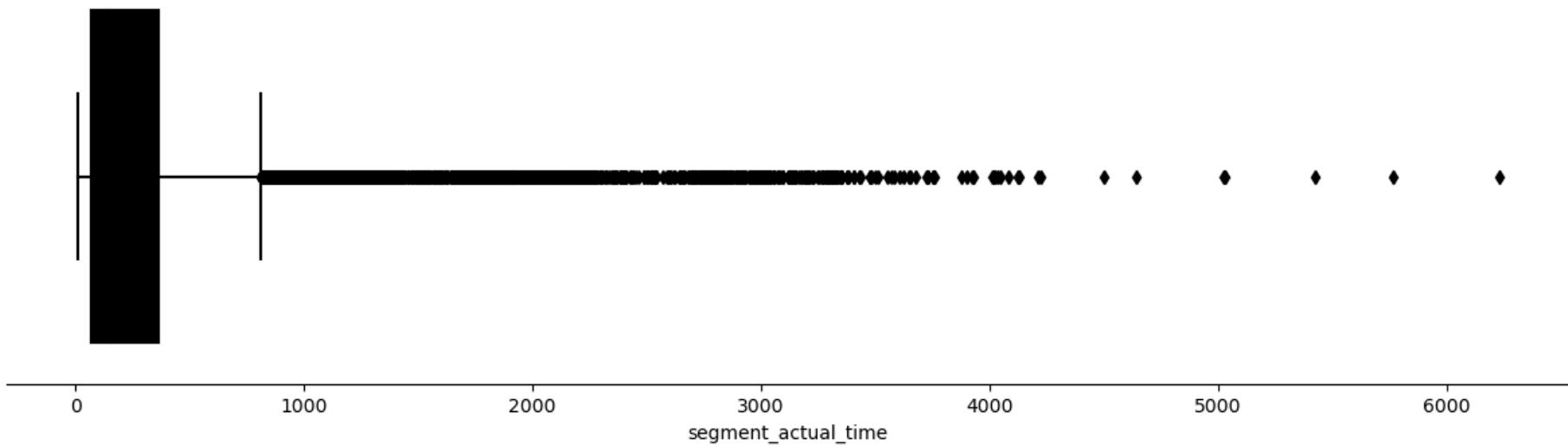
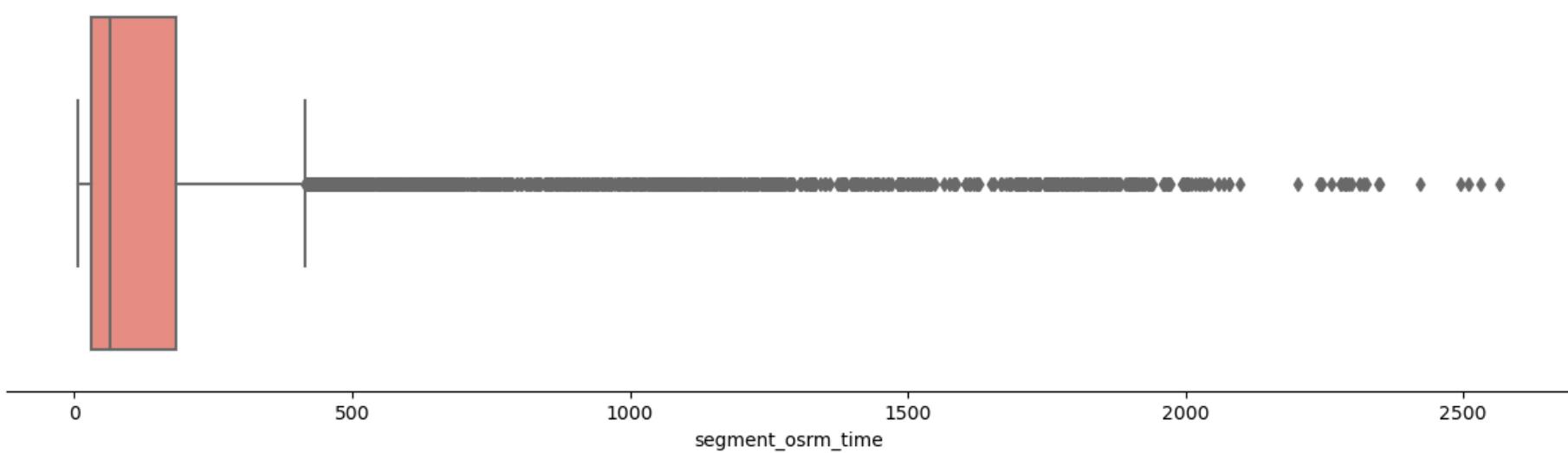
In [133...]

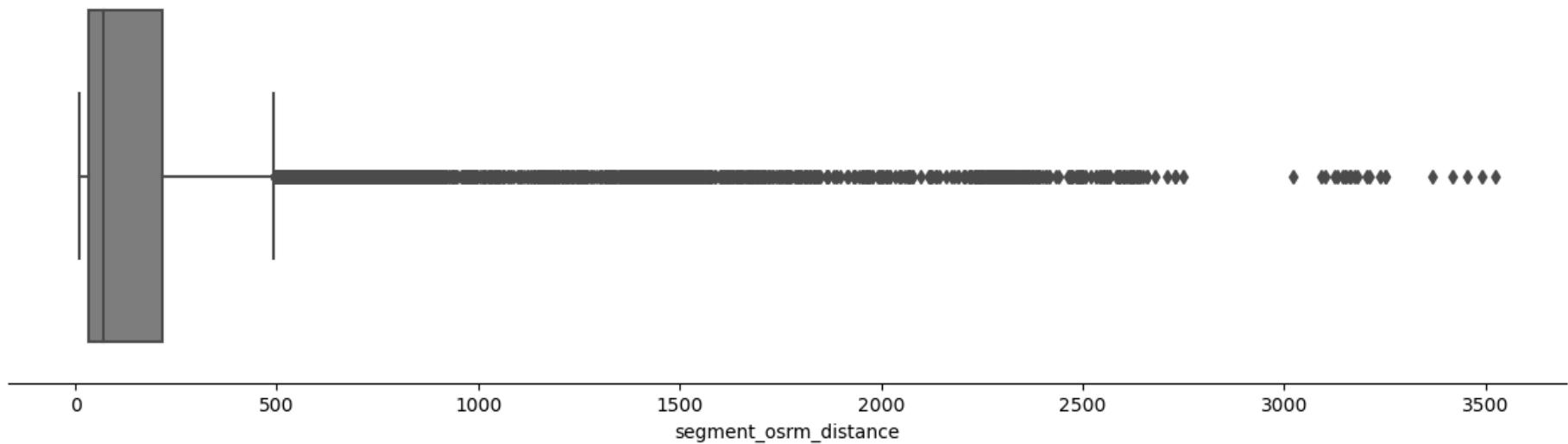
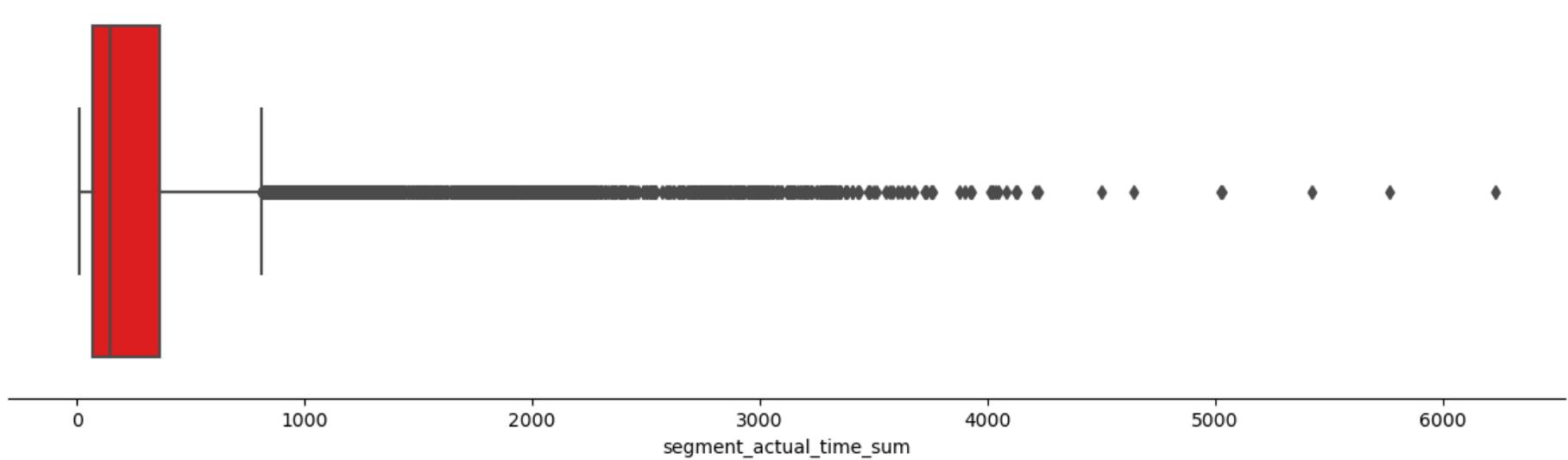
```
for i,col in enumerate(numerical_columns):
    plt.figure(figsize=(15,4))
    sns.boxplot(x=col, data=numerical_columns,color=cp[i])
    sns.despine(left=True)
    plt.yticks([])
    plt.title(f'Boxplot of {col}',fontfamily='serif',fontweight='bold',fontsize=12,backgroundcolor=cp[i],color='w')
    plt.show()
```

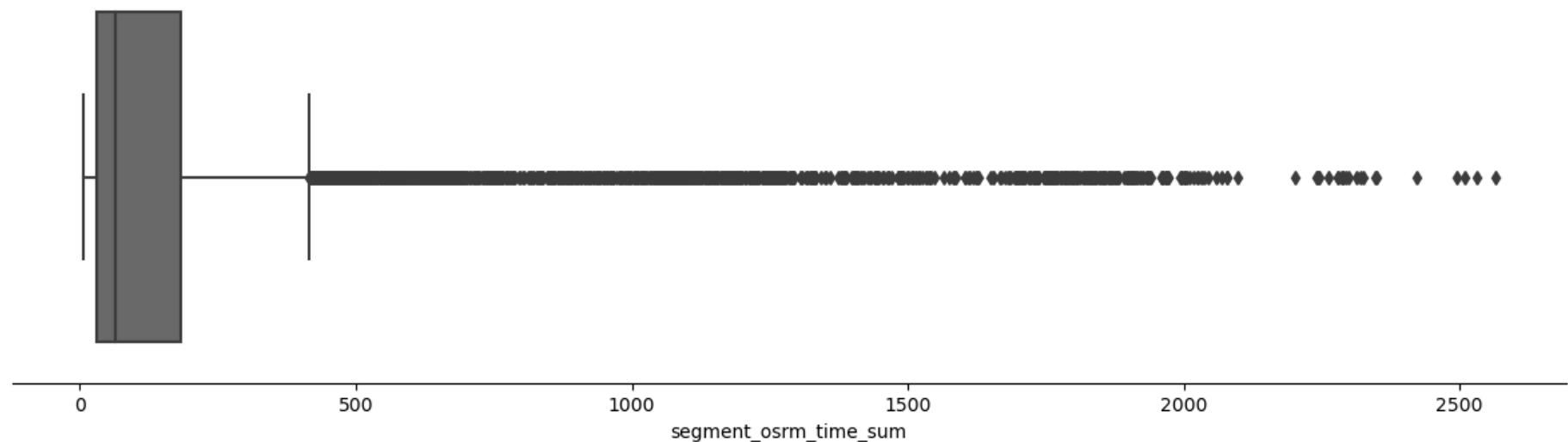
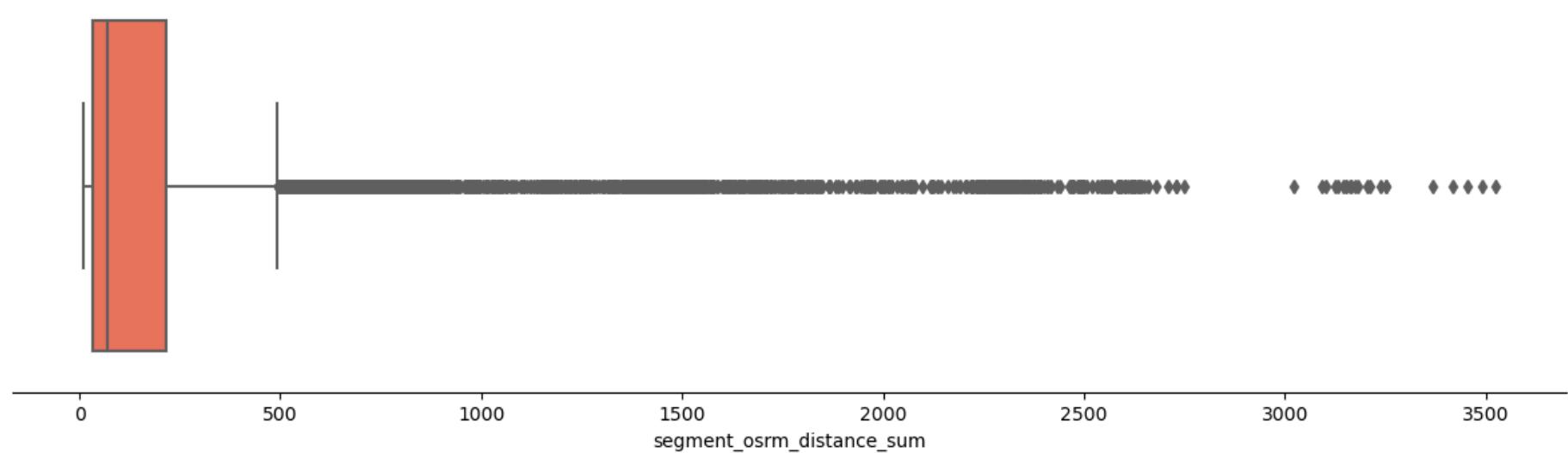
Boxplot of od_time_diff_hour**Boxplot of start_scan_to_end_scan**

Boxplot of actual_distance_to_destination**Boxplot of actual_time**

Boxplot of osrm_time**Boxplot of osrm_distance**

Boxplot of segment_actual_time**Boxplot of segment_osrm_time**

Boxplot of segment_osrm_distance**Boxplot of segment_actual_time_sum**

Boxplot of segment_osrm_time_sum**Boxplot of segment_osrm_distance_sum**

⭐ Outlier Removal

In [134...]

```
for i, col in enumerate(numerical_columns):

    data = trip_df[col]
    display(data.to_frame())

    Q1 = np.percentile(data, 25)
    Q3 = np.percentile(data, 75)
    IQR = Q3 - Q1

    lower_bound = Q1 - (1.5 * IQR)
    upper_bound = Q3 + (1.5 * IQR)

    clipped_data = np.clip(data, lower_bound, upper_bound)
    print(f'Clipped data of {col}')
    display(clipped_data.to_frame())
    print()

    # Plot boxplot of the clipped data
    plt.figure(figsize=(15, 4))
    plt.subplot(121)
    sns.boxplot(x=clipped_data, color=cp[i])
    sns.despine(left=True)
    plt.yticks([])
    plt.title(f'Boxplot of clipped {col}', fontfamily='serif', fontweight='bold', fontsize=12, backgroundcolor=cp[i], color='white')

    filtered_data = data.loc[(data >= lower_bound) | (data <= upper_bound)]
    print(f'Filtered data of {col}')
    display(filtered_data.to_frame())
    print()

    plt.subplot(122)
    sns.boxplot(x=filtered_data, color=cp[i])
    sns.despine(left=True)
    plt.yticks([])
    plt.title(f'Boxplot of filtered {col}', fontfamily='serif', fontweight='bold', fontsize=12, backgroundcolor=cp[i], color='white')

    plt.show()
```

od_time_diff_hour	
0	37.668497
1	3.026865
2	65.572709
3	1.674916
4	11.972484
...	...
14782	4.300482
14783	1.009842
14784	7.035331
14785	5.808548
14786	5.906793

14787 rows × 1 columns

Clipped data of od_time_diff_hour

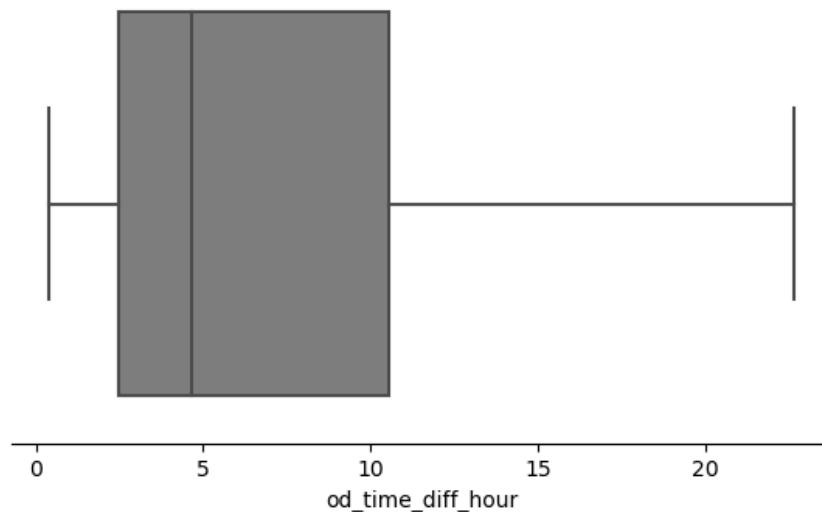
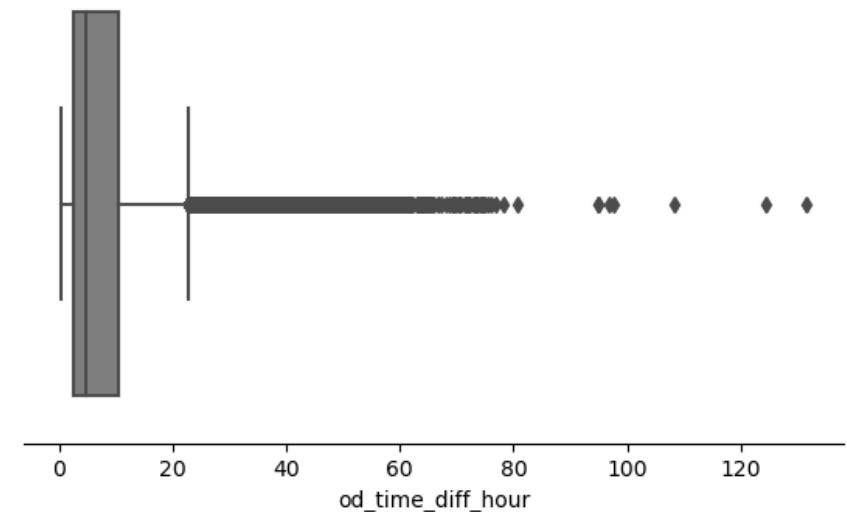
od_time_diff_hour	
0	22.654942
1	3.026865
2	22.654942
3	1.674916
4	11.972484
...	...
14782	4.300482
14783	1.009842
14784	7.035331
14785	5.808548
14786	5.906793

14787 rows × 1 columns

Filtered data of od_time_diff_hour

od_time_diff_hour
0 37.668497
1 3.026865
2 65.572709
3 1.674916
4 11.972484
... ...
14782 4.300482
14783 1.009842
14784 7.035331
14785 5.808548
14786 5.906793

14787 rows × 1 columns

Boxplot of clipped od_time_diff_hour**Boxplot of filtered od_time_diff_hour**

start_scan_to_end_scan	
0	2259.0
1	180.0
2	3933.0
3	100.0
4	717.0
...	...
14782	257.0
14783	60.0
14784	421.0
14785	347.0
14786	353.0

14787 rows × 1 columns

Clipped data of start_scan_to_end_scan

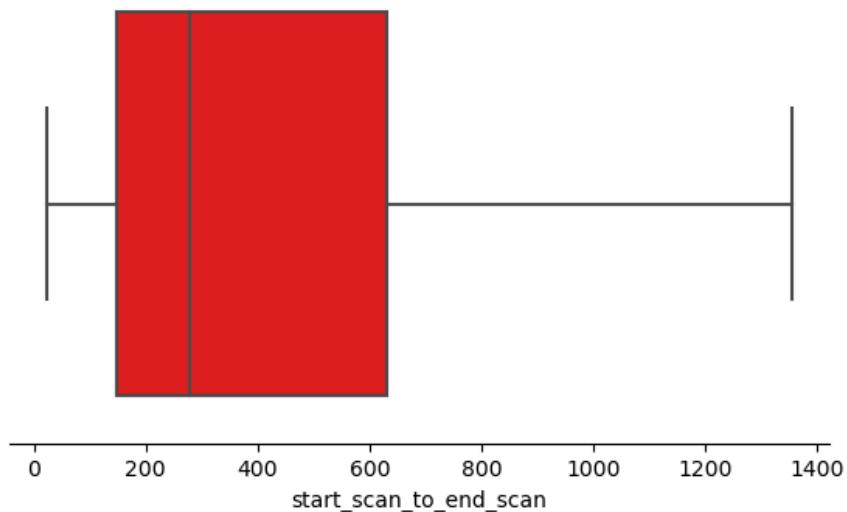
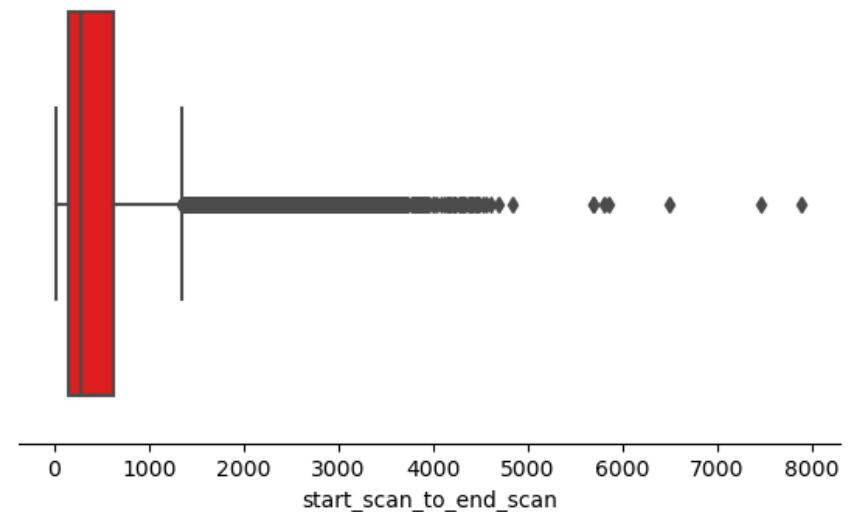
start_scan_to_end_scan	
0	1356.5
1	180.0
2	1356.5
3	100.0
4	717.0
...	...
14782	257.0
14783	60.0
14784	421.0
14785	347.0
14786	353.0

14787 rows × 1 columns

Filtered data of start_scan_to_end_scan

start_scan_to_end_scan	
0	2259.0
1	180.0
2	3933.0
3	100.0
4	717.0
...	...
14782	257.0
14783	60.0
14784	421.0
14785	347.0
14786	353.0

14787 rows × 1 columns

Boxplot of clipped start_scan_to_end_scan**Boxplot of filtered start_scan_to_end_scan**

actual_distance_to_destination
0 824.732849
1 73.186905
2 1927.404297
3 17.175274
4 127.448502
... ...
14782 57.762333
14783 15.513784
14784 38.684837
14785 134.723831
14786 66.081528

14787 rows × 1 columns

Clipped data of actual_distance_to_destination

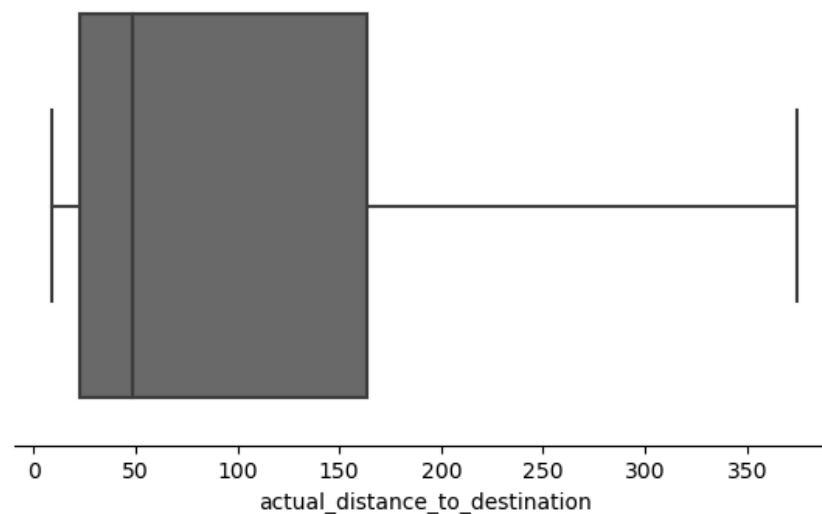
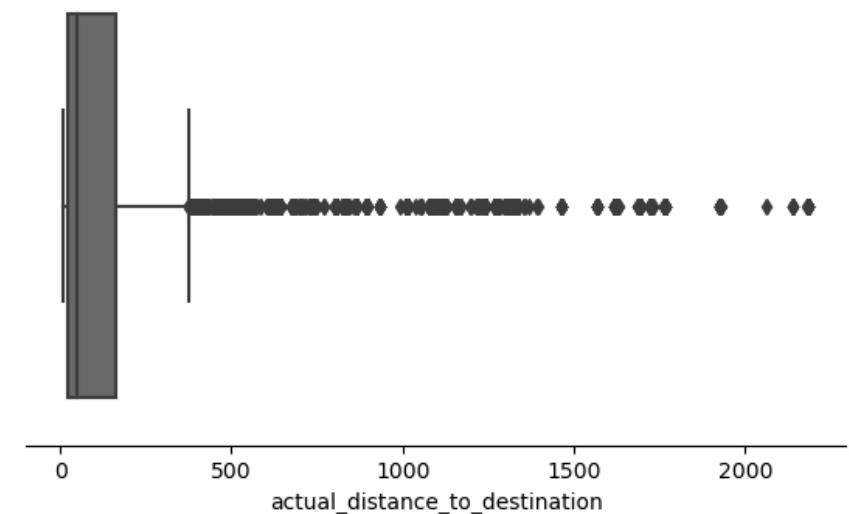
actual_distance_to_destination	
0	374.812490
1	73.186905
2	374.812490
3	17.175274
4	127.448502
...	...
14782	57.762333
14783	15.513784
14784	38.684837
14785	134.723831
14786	66.081528

14787 rows × 1 columns

Filtered data of actual_distance_to_destination

actual_distance_to_destination
0 824.732849
1 73.186905
2 1927.404297
3 17.175274
4 127.448502
... ...
14782 57.762333
14783 15.513784
14784 38.684837
14785 134.723831
14786 66.081528

14787 rows × 1 columns

Boxplot of clipped actual_distance_to_destination**Boxplot of filtered actual_distance_to_destination**

actual_time
0 1562.0
1 143.0
2 3347.0
3 59.0
4 341.0
... ...
14782 83.0
14783 21.0
14784 282.0
14785 264.0
14786 275.0

14787 rows × 1 columns

Clipped data of actual_time

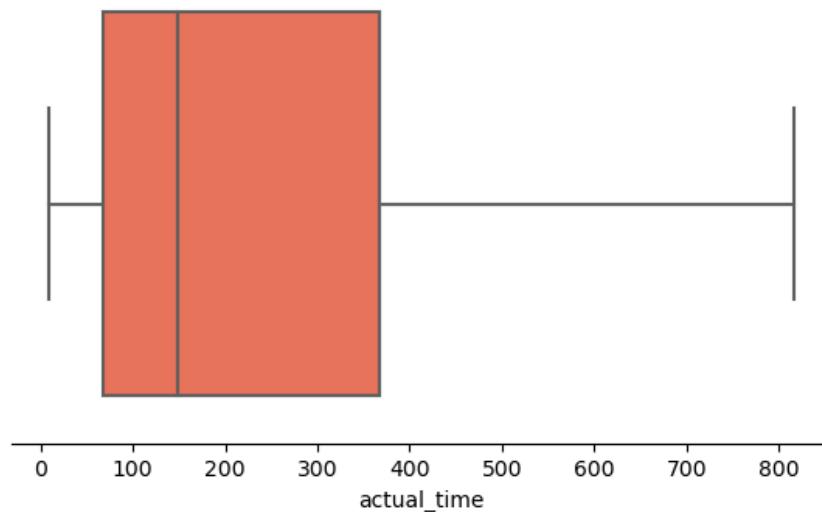
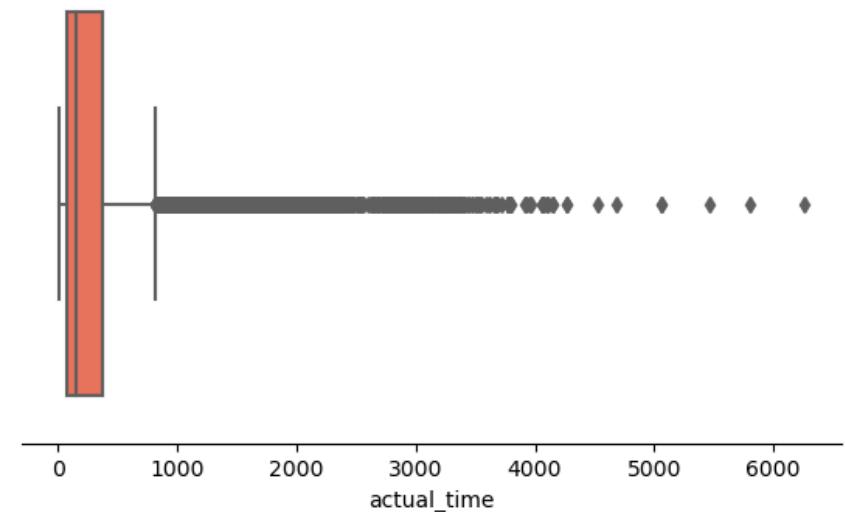
	actual_time
0	817.0
1	143.0
2	817.0
3	59.0
4	341.0
...	...
14782	83.0
14783	21.0
14784	282.0
14785	264.0
14786	275.0

14787 rows × 1 columns

Filtered data of actual_time

	actual_time
0	1562.0
1	143.0
2	3347.0
3	59.0
4	341.0
...	...
14782	83.0
14783	21.0
14784	282.0
14785	264.0
14786	275.0

14787 rows × 1 columns

Boxplot of clipped actual_time**Boxplot of filtered actual_time**

osrm_time	
0	717.0
1	68.0
2	1740.0
3	15.0
4	117.0
...	...
14782	62.0
14783	12.0
14784	48.0
14785	179.0
14786	68.0

14787 rows × 1 columns

Clipped data of osrm_time

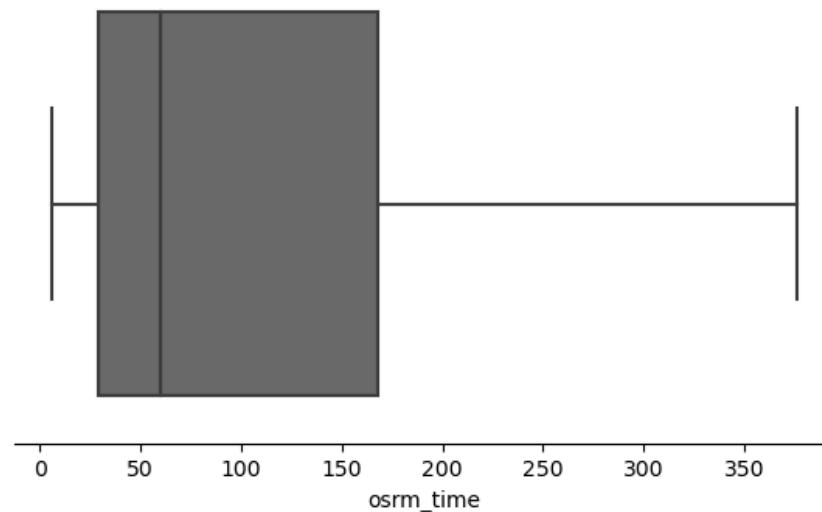
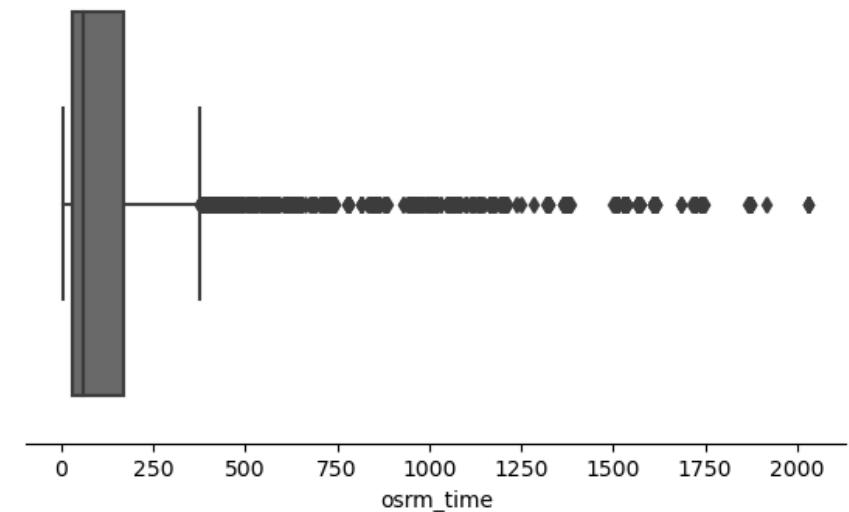
osrm_time	
0	376.5
1	68.0
2	376.5
3	15.0
4	117.0
...	...
14782	62.0
14783	12.0
14784	48.0
14785	179.0
14786	68.0

14787 rows × 1 columns

Filtered data of osrm_time

osrm_time	
0	717.0
1	68.0
2	1740.0
3	15.0
4	117.0
...	...
14782	62.0
14783	12.0
14784	48.0
14785	179.0
14786	68.0

14787 rows × 1 columns

Boxplot of clipped osrm_time**Boxplot of filtered osrm_time**

osrm_distance

0	991.352295
1	85.111000
2	2354.066650
3	19.680000
4	146.791794
...	...
14782	73.462997
14783	16.088200
14784	58.903702
14785	171.110306
14786	80.578705

14787 rows × 1 columns

Clipped data of osrm_distance

osrm_distance

0	470.475158
1	85.111000
2	470.475158
3	19.680000
4	146.791794
...	...
14782	73.462997
14783	16.088200
14784	58.903702
14785	171.110306
14786	80.578705

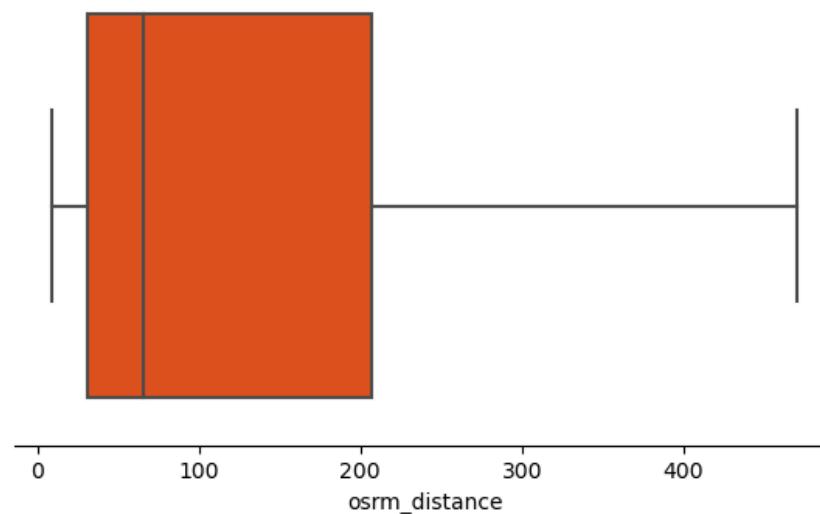
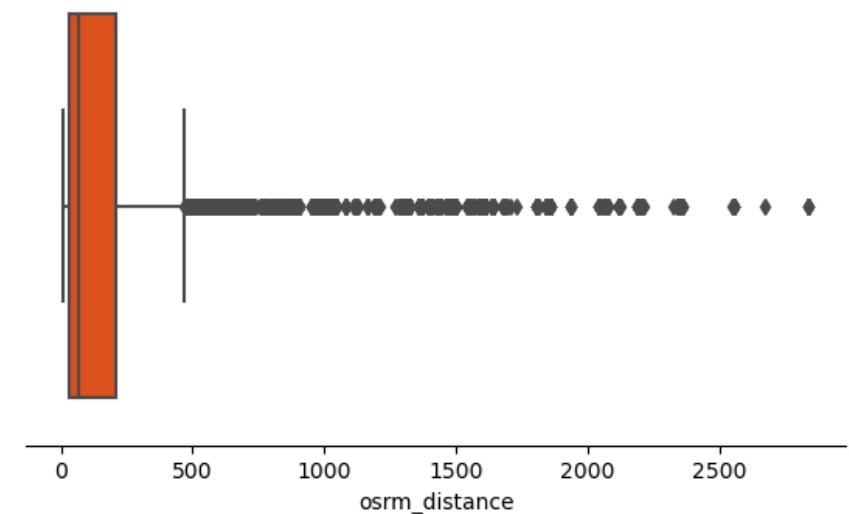
14787 rows × 1 columns

Filtered data of osrm_distance

osrm_distance

0	991.352295
1	85.111000
2	2354.066650
3	19.680000
4	146.791794
...	...
14782	73.462997
14783	16.088200
14784	58.903702
14785	171.110306
14786	80.578705

14787 rows × 1 columns

Boxplot of clipped osrm_distance**Boxplot of filtered osrm_distance**

segment_actual_time
0 1548.0
1 141.0
2 3308.0
3 59.0
4 340.0
... ...
14782 82.0
14783 21.0
14784 281.0
14785 258.0
14786 274.0

14787 rows × 1 columns

Clipped data of segment_actual_time

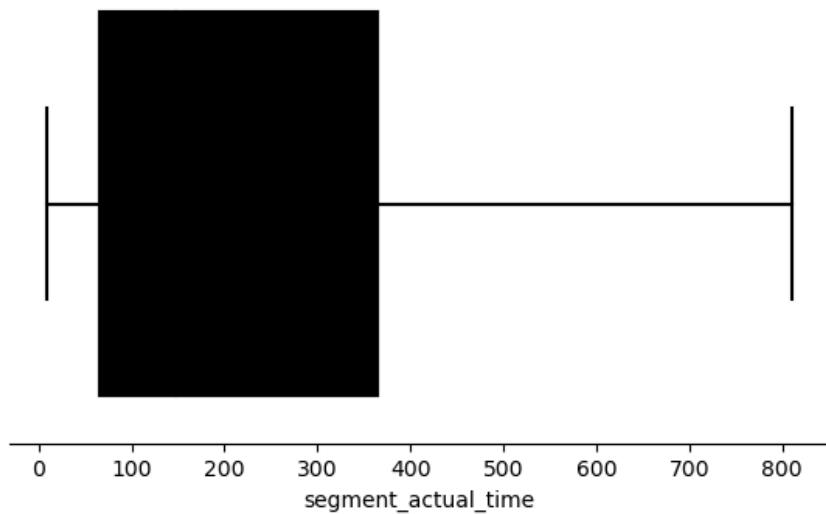
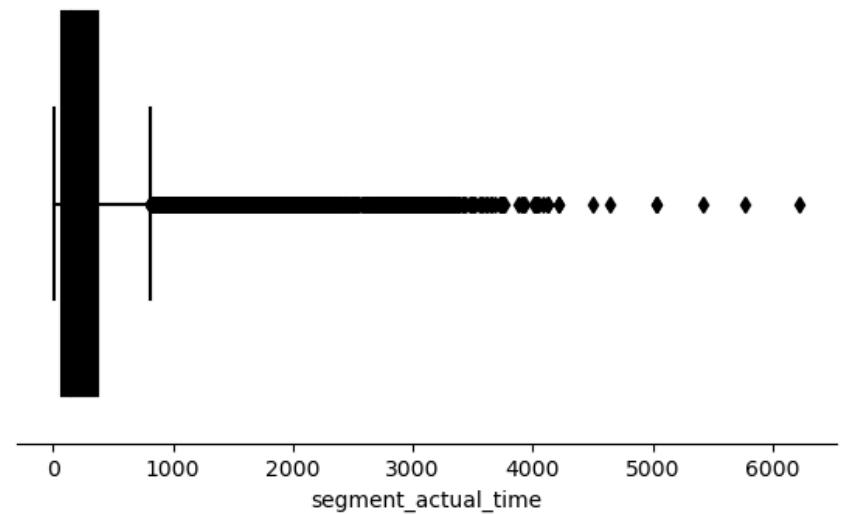
segment_actual_time
0 811.0
1 141.0
2 811.0
3 59.0
4 340.0
... ...
14782 82.0
14783 21.0
14784 281.0
14785 258.0
14786 274.0

14787 rows × 1 columns

Filtered data of segment_actual_time

segment_actual_time
0 1548.0
1 141.0
2 3308.0
3 59.0
4 340.0
... ...
14782 82.0
14783 21.0
14784 281.0
14785 258.0
14786 274.0

14787 rows × 1 columns

Boxplot of clipped segment_actual_time**Boxplot of filtered segment_actual_time**

segment_osrm_time
0 1008.0
1 65.0
2 1941.0
3 16.0
4 115.0
... ...
14782 62.0
14783 11.0
14784 88.0
14785 221.0
14786 67.0

14787 rows × 1 columns

Clipped data of segment_osrm_time

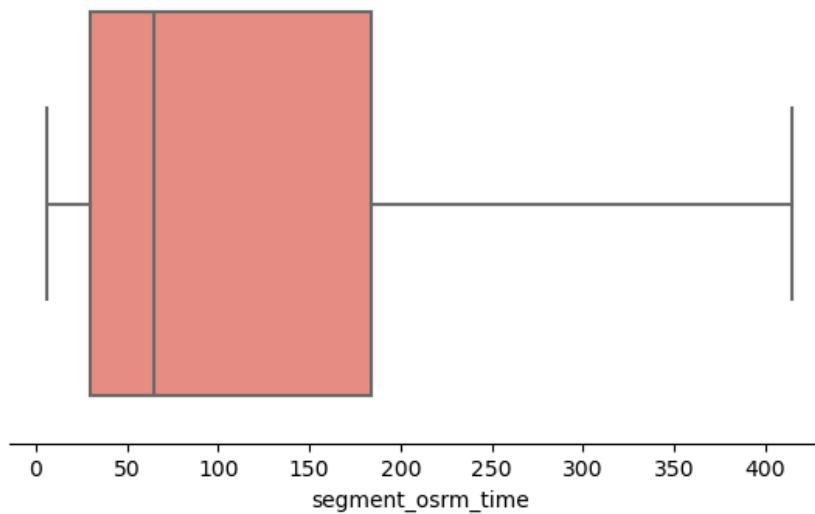
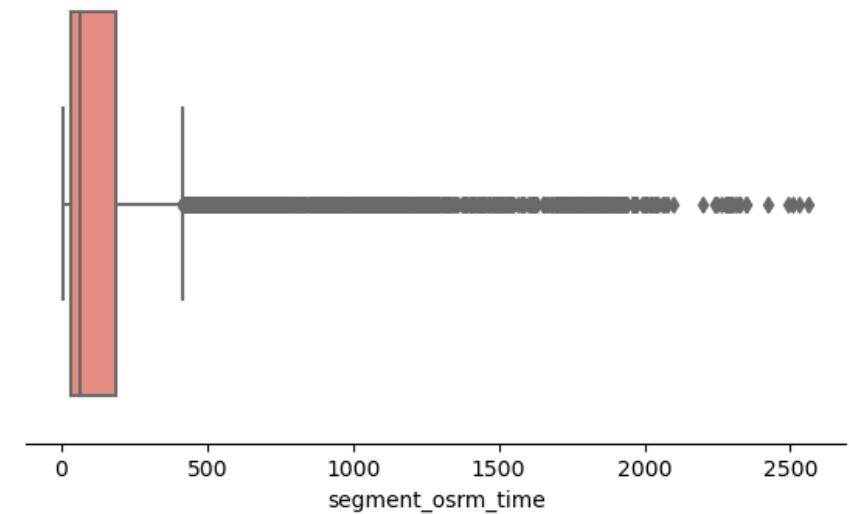
segment_osrm_time	
0	415.0
1	65.0
2	415.0
3	16.0
4	115.0
...	...
14782	62.0
14783	11.0
14784	88.0
14785	221.0
14786	67.0

14787 rows × 1 columns

Filtered data of segment_osrm_time

segment_osrm_time
0 1008.0
1 65.0
2 1941.0
3 16.0
4 115.0
... ...
14782 62.0
14783 11.0
14784 88.0
14785 221.0
14786 67.0

14787 rows × 1 columns

Boxplot of clipped segment_osrm_time**Boxplot of filtered segment_osrm_time**

segment_osrm_distance
0 1320.473267
1 84.189400
2 2545.267822
3 19.876600
4 146.791901
... ...
14782 64.855103
14783 16.088299
14784 104.886597
14785 223.532394
14786 80.578705

14787 rows × 1 columns

Clipped data of segment_osrm_distance

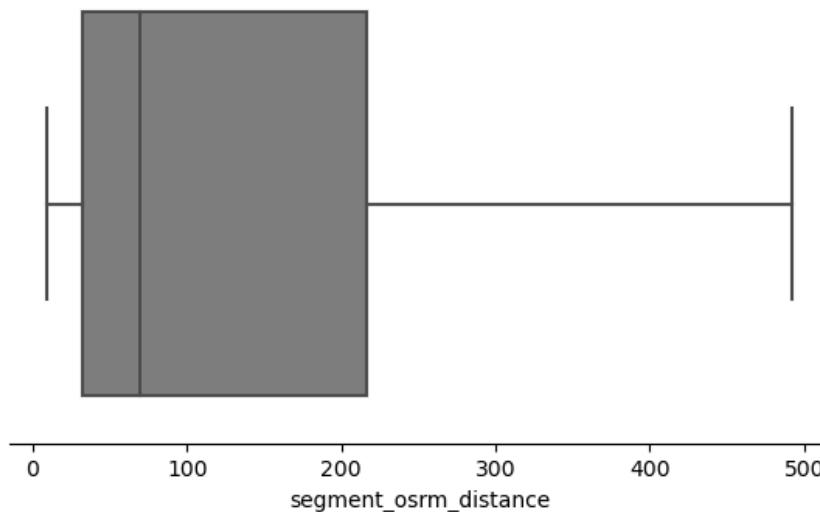
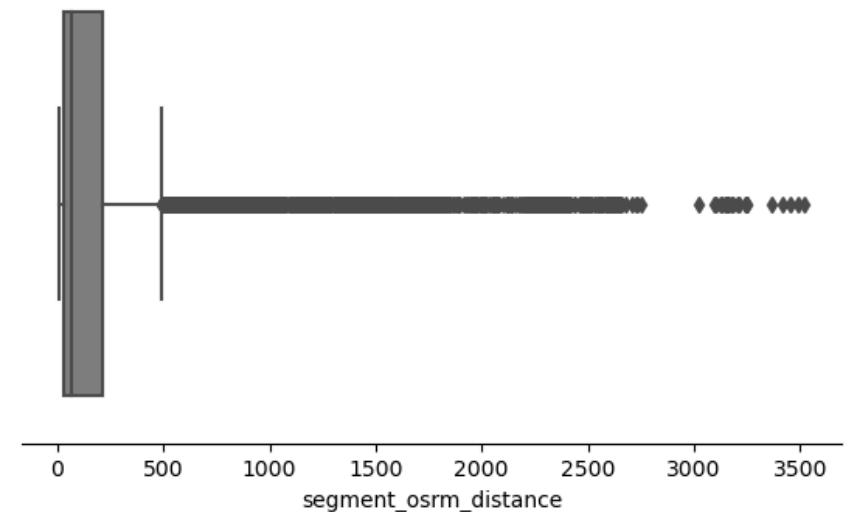
segment_osrm_distance
0 492.533245
1 84.189400
2 492.533245
3 19.876600
4 146.791901
... ...
14782 64.855103
14783 16.088299
14784 104.886597
14785 223.532394
14786 80.578705

14787 rows × 1 columns

Filtered data of segment_osrm_distance

segment_osrm_distance
0 1320.473267
1 84.189400
2 2545.267822
3 19.876600
4 146.791901
... ...
14782 64.855103
14783 16.088299
14784 104.886597
14785 223.532394
14786 80.578705

14787 rows × 1 columns

Boxplot of clipped segment_osrm_distance**Boxplot of filtered segment_osrm_distance**

segment_actual_time_sum
0 1548.0
1 141.0
2 3308.0
3 59.0
4 340.0
... ...
14782 82.0
14783 21.0
14784 281.0
14785 258.0
14786 274.0

14787 rows × 1 columns

Clipped data of segment_actual_time_sum

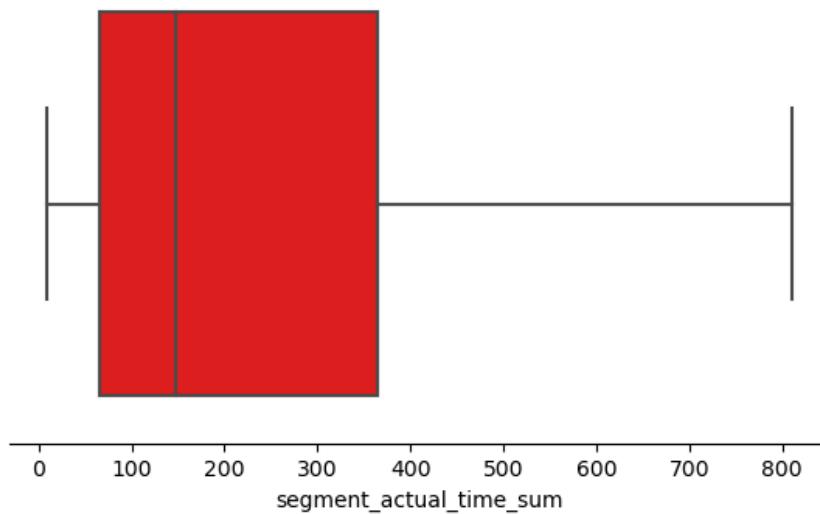
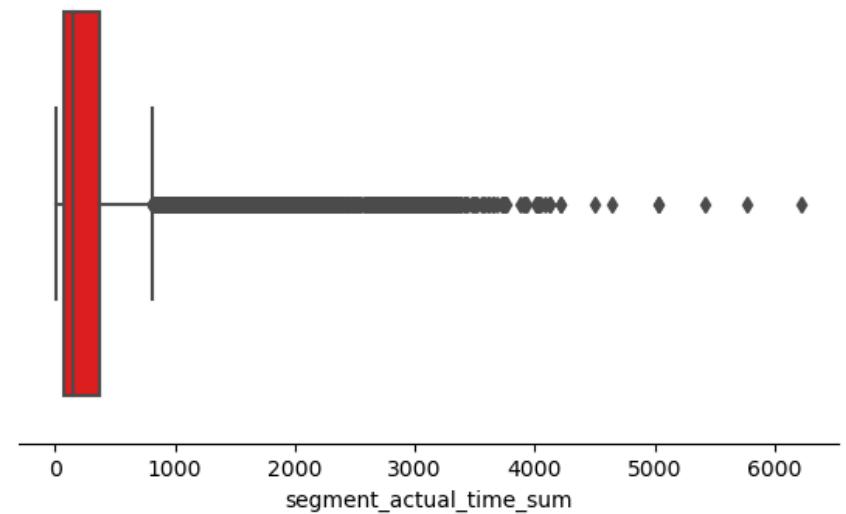
segment_actual_time_sum
0 811.0
1 141.0
2 811.0
3 59.0
4 340.0
... ...
14782 82.0
14783 21.0
14784 281.0
14785 258.0
14786 274.0

14787 rows × 1 columns

Filtered data of segment_actual_time_sum

segment_actual_time_sum
0 1548.0
1 141.0
2 3308.0
3 59.0
4 340.0
... ...
14782 82.0
14783 21.0
14784 281.0
14785 258.0
14786 274.0

14787 rows × 1 columns

Boxplot of clipped segment_actual_time_sum**Boxplot of filtered segment_actual_time_sum**

segment_osrm_time_sum	
0	1008.0
1	65.0
2	1941.0
3	16.0
4	115.0
...	...
14782	62.0
14783	11.0
14784	88.0
14785	221.0
14786	67.0

14787 rows × 1 columns

Clipped data of segment_osrm_time_sum

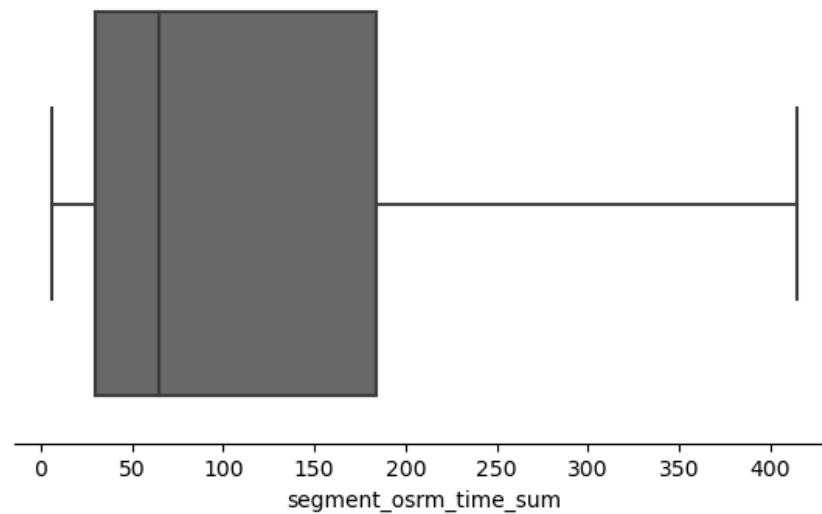
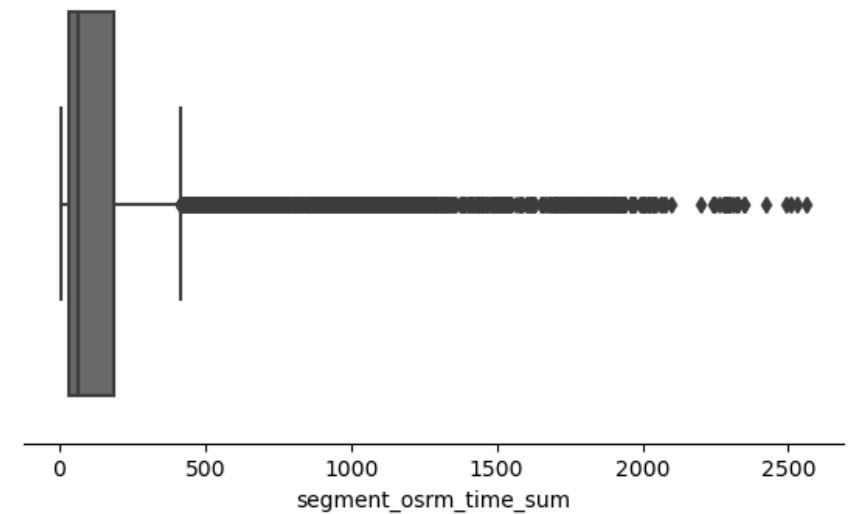
segment_osrm_time_sum	
0	415.0
1	65.0
2	415.0
3	16.0
4	115.0
...	...
14782	62.0
14783	11.0
14784	88.0
14785	221.0
14786	67.0

14787 rows × 1 columns

Filtered data of segment_osrm_time_sum

segment_osrm_time_sum
0 1008.0
1 65.0
2 1941.0
3 16.0
4 115.0
... ...
14782 62.0
14783 11.0
14784 88.0
14785 221.0
14786 67.0

14787 rows × 1 columns

Boxplot of clipped segment_osrm_time_sum**Boxplot of filtered segment_osrm_time_sum**

segment_osrm_distance_sum

0	1320.473267
1	84.189400
2	2545.267822
3	19.876600
4	146.791901
...	...
14782	64.855103
14783	16.088299
14784	104.886597
14785	223.532394
14786	80.578705

14787 rows × 1 columns

Clipped data of segment_osrm_distance_sum

segment_osrm_distance_sum	
0	492.533245
1	84.189400
2	492.533245
3	19.876600
4	146.791901
...	...
14782	64.855103
14783	16.088299
14784	104.886597
14785	223.532394
14786	80.578705

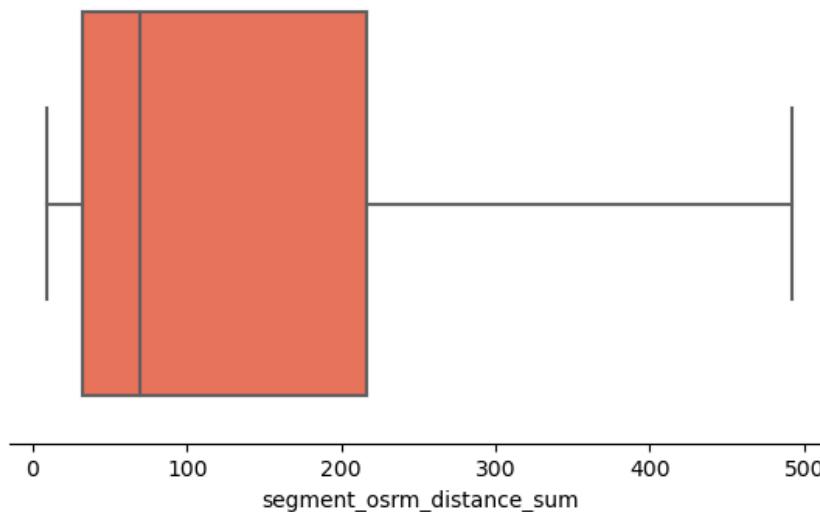
14787 rows × 1 columns

Filtered data of segment_osrm_distance_sum

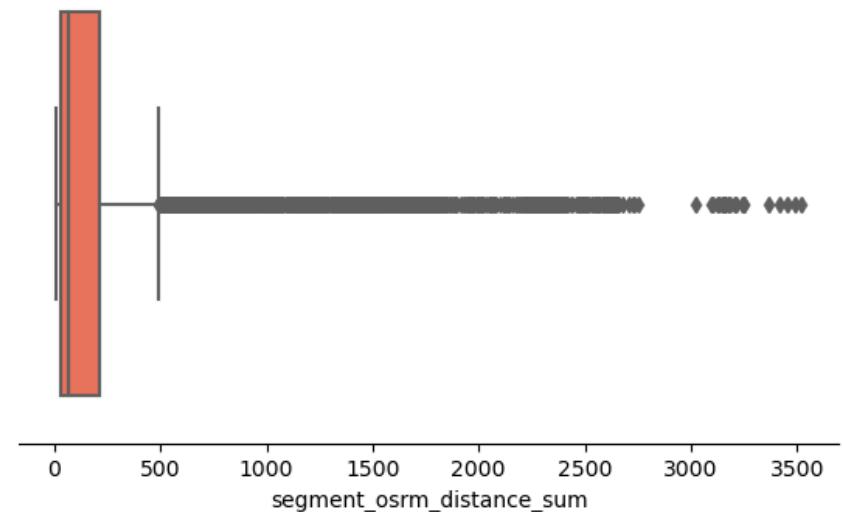
segment_osrm_distance_sum
0 1320.473267
1 84.189400
2 2545.267822
3 19.876600
4 146.791901
... ...
14782 64.855103
14783 16.088299
14784 104.886597
14785 223.532394
14786 80.578705

14787 rows × 1 columns

Boxplot of clipped segment_osrm_distance_sum



Boxplot of filtered segment_osrm_distance_sum



💡 Understanding:

- Here we see that the data after removing outliers has outliers. It has to be understood that q1 and q3 dont have to be always 25th percentile and 75th percentile. ***Try changing q1 and q3 to 10th percentile to 90th percentile and plot and see...***
- Clipped data replaces the outlier values with specified values.
- Here, I have proceeded with both clipped and filtered data(with reduced outliers) for further analysis.

In [135...]

```
num_df = numerical_columns.copy()
num_df
```

Out[135...]

	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time
0	37.668497	2259.0	824.732849	1562.0	717.0	991.352295	1548
1	3.026865	180.0	73.186905	143.0	68.0	85.111000	141
2	65.572709	3933.0	1927.404297	3347.0	1740.0	2354.066650	3308
3	1.674916	100.0	17.175274	59.0	15.0	19.680000	59
4	11.972484	717.0	127.448502	341.0	117.0	146.791794	340
...
14782	4.300482	257.0	57.762333	83.0	62.0	73.462997	82
14783	1.009842	60.0	15.513784	21.0	12.0	16.088200	21
14784	7.035331	421.0	38.684837	282.0	48.0	58.903702	28
14785	5.808548	347.0	134.723831	264.0	179.0	171.110306	258
14786	5.906793	353.0	66.081528	275.0	68.0	80.578705	274

14787 rows × 12 columns



In [136...]

num_cols

Out[136...]

```
['od_time_diff_hour',
 'start_scan_to_end_scan',
 'actual_distance_to_destination',
 'actual_time',
 'osrm_time',
 'osrm_distance',
 'segment_actual_time',
 'segment_osrm_time',
 'segment_osrm_distance',
 'segment_actual_time_sum',
 'segment_osrm_time_sum',
 'segment_osrm_distance_sum']
```

In [137...]

```

Q1 = np.percentile(num_df[num_cols], 25)
Q3 = np.percentile(num_df[num_cols], 75)
IQR = Q3 - Q1

lower_bound = Q1 - (1.5 * IQR)
upper_bound = Q3 + (1.5 * IQR)

clipped_num_df = np.clip(num_df[num_cols], lower_bound, upper_bound)
display(clipped_num_df)

filtered_num_df = num_df[num_cols][(num_df[num_cols] >= lower_bound) | (num_df[num_cols] <= upper_bound)]
display(filtered_num_df)

```

	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time
0	37.668497	543.285302	543.285302	543.285302	543.285302	543.285302	543.285302
1	3.026865	180.000000	73.186905	143.000000	68.000000	85.111000	141.000000
2	65.572709	543.285302	543.285302	543.285302	543.285302	543.285302	543.285302
3	1.674916	100.000000	17.175274	59.000000	15.000000	19.680000	59.000000
4	11.972484	543.285302	127.448502	341.000000	117.000000	146.791794	340.000000
...
14782	4.300482	257.000000	57.762333	83.000000	62.000000	73.462997	82.000000
14783	1.009842	60.000000	15.513784	21.000000	12.000000	16.088200	21.000000
14784	7.035331	421.000000	38.684837	282.000000	48.000000	58.903702	281.000000
14785	5.808548	347.000000	134.723831	264.000000	179.000000	171.110306	258.000000
14786	5.906793	353.000000	66.081528	275.000000	68.000000	80.578705	274.000000

14787 rows × 12 columns



	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time
0	37.668497	2259.0	824.732849	1562.0	717.0	991.352295	1548.0
1	3.026865	180.0	73.186905	143.0	68.0	85.111000	141.0
2	65.572709	3933.0	1927.404297	3347.0	1740.0	2354.066650	3308.0
3	1.674916	100.0	17.175274	59.0	15.0	19.680000	59.0
4	11.972484	717.0	127.448502	341.0	117.0	146.791794	340.0
...
14782	4.300482	257.0	57.762333	83.0	62.0	73.462997	82.0
14783	1.009842	60.0	15.513784	21.0	12.0	16.088200	21.0
14784	7.035331	421.0	38.684837	282.0	48.0	58.903702	281.0
14785	5.808548	347.0	134.723831	264.0	179.0	171.110306	258.0
14786	5.906793	353.0	66.081528	275.0	68.0	80.578705	274.0

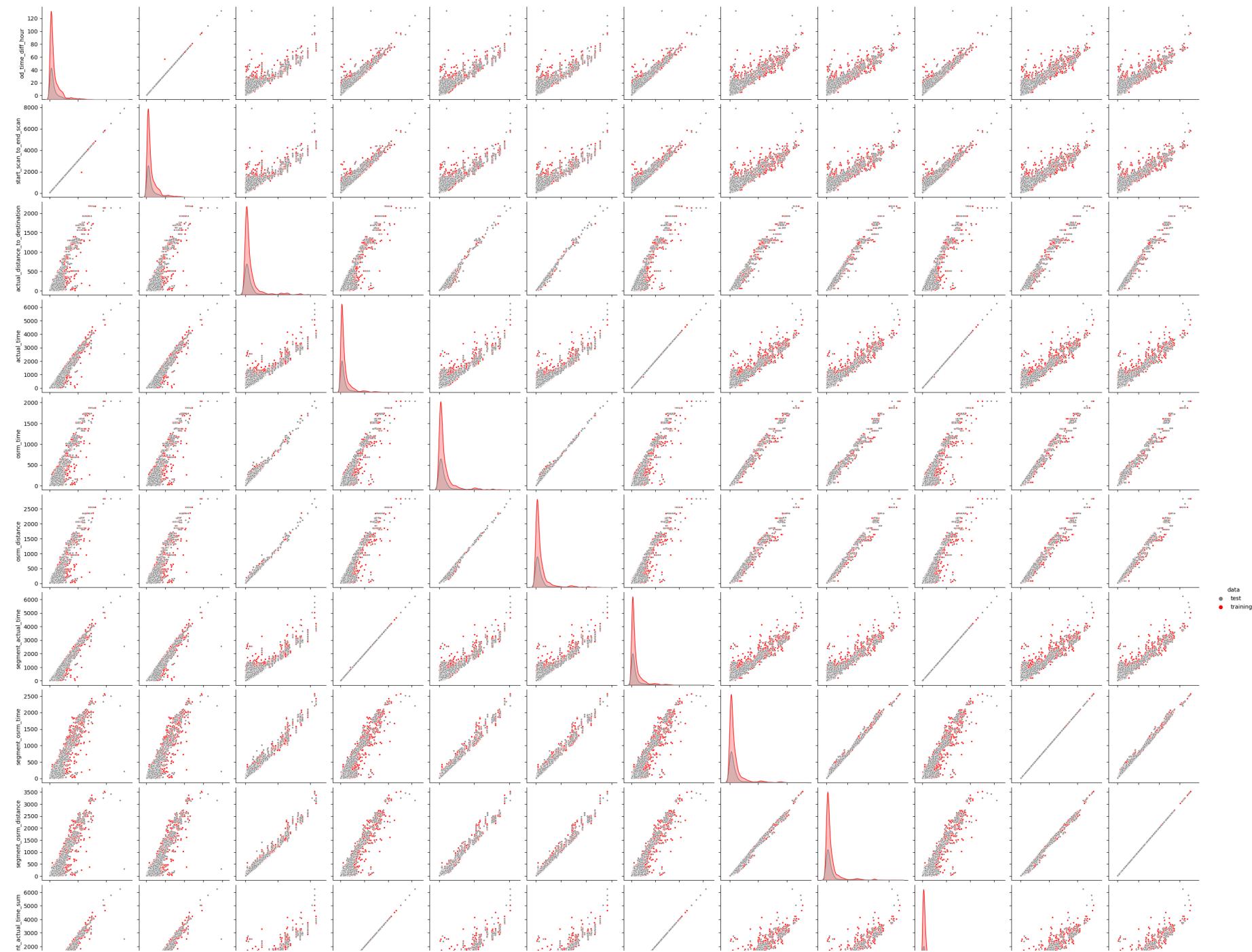
14787 rows × 12 columns

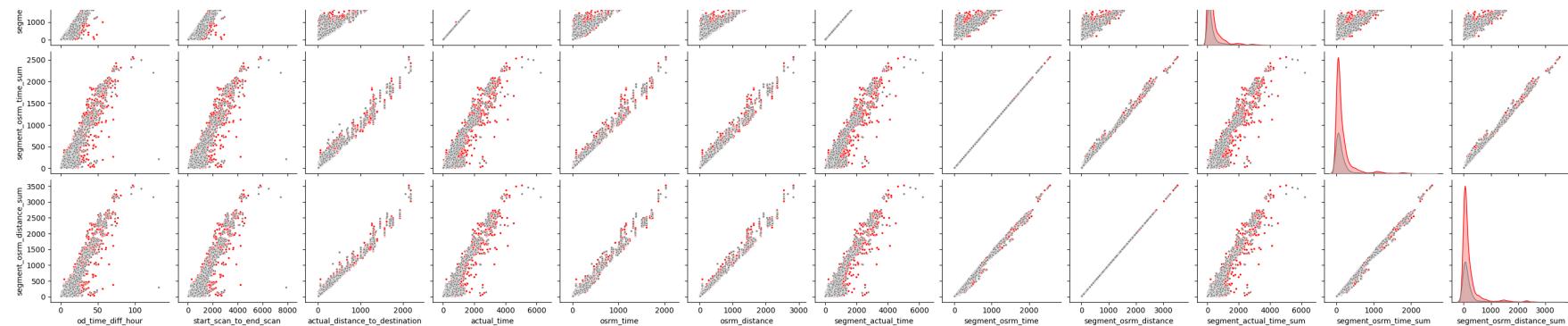


In [138...]

```
plt.figure(figsize=(14,0.05))
plt.axis('off')
plt.title(f' Pairplot Analysis',fontfamily='serif',fontweight='bold',fontsize=15,backgroundcolor='crimson',color='w')
sns.pairplot(data = trip_df,vars = num_cols,hue='data',markers = '.',palette=cp)
plt.show()
```

Pairplot Analysis

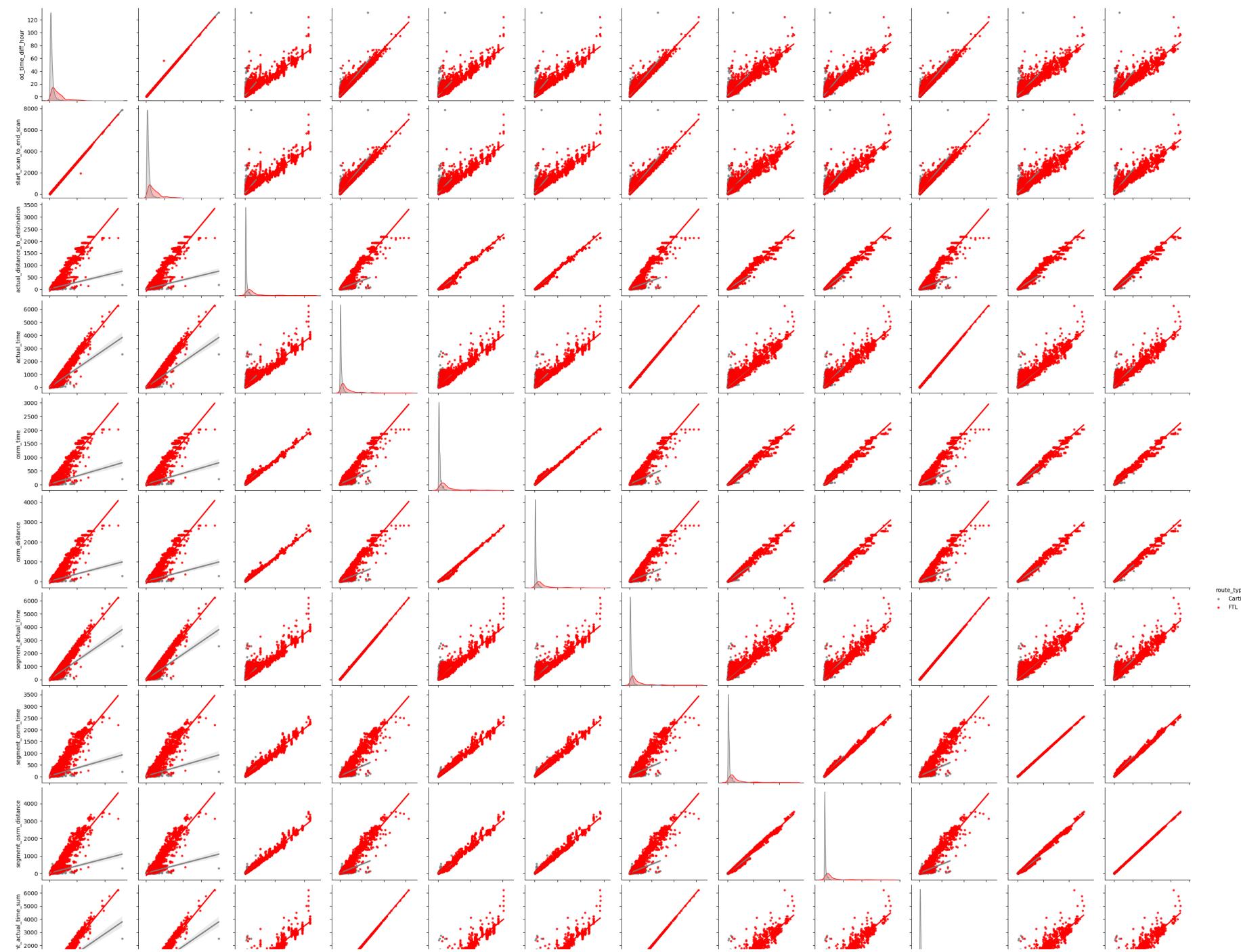


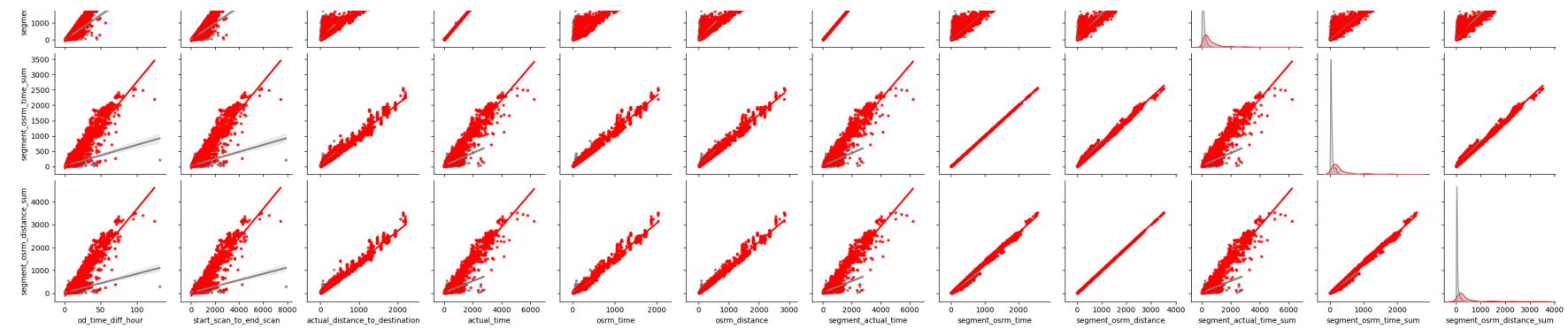


In [139...]

```
plt.figure(figsize=(14,0.05))
plt.axis('off')
plt.title(f' Pairplot Analysis',fontfamily='serif',fontweight='bold',fontsize=15,backgroundcolor='dimgray',color='w')
sns.pairplot(data = trip_df,vars=num_cols,kind = 'reg',hue='route_type',markers = '.',palette=cp)
plt.show()
```

Pairplot Analysis





In [140...]

```
clipped_df_corr = clipped_num_df.corr()  
clipped_df_corr  
  
filtered_df_corr = filtered_num_df.corr()  
filtered_df_corr
```

Out[140...]

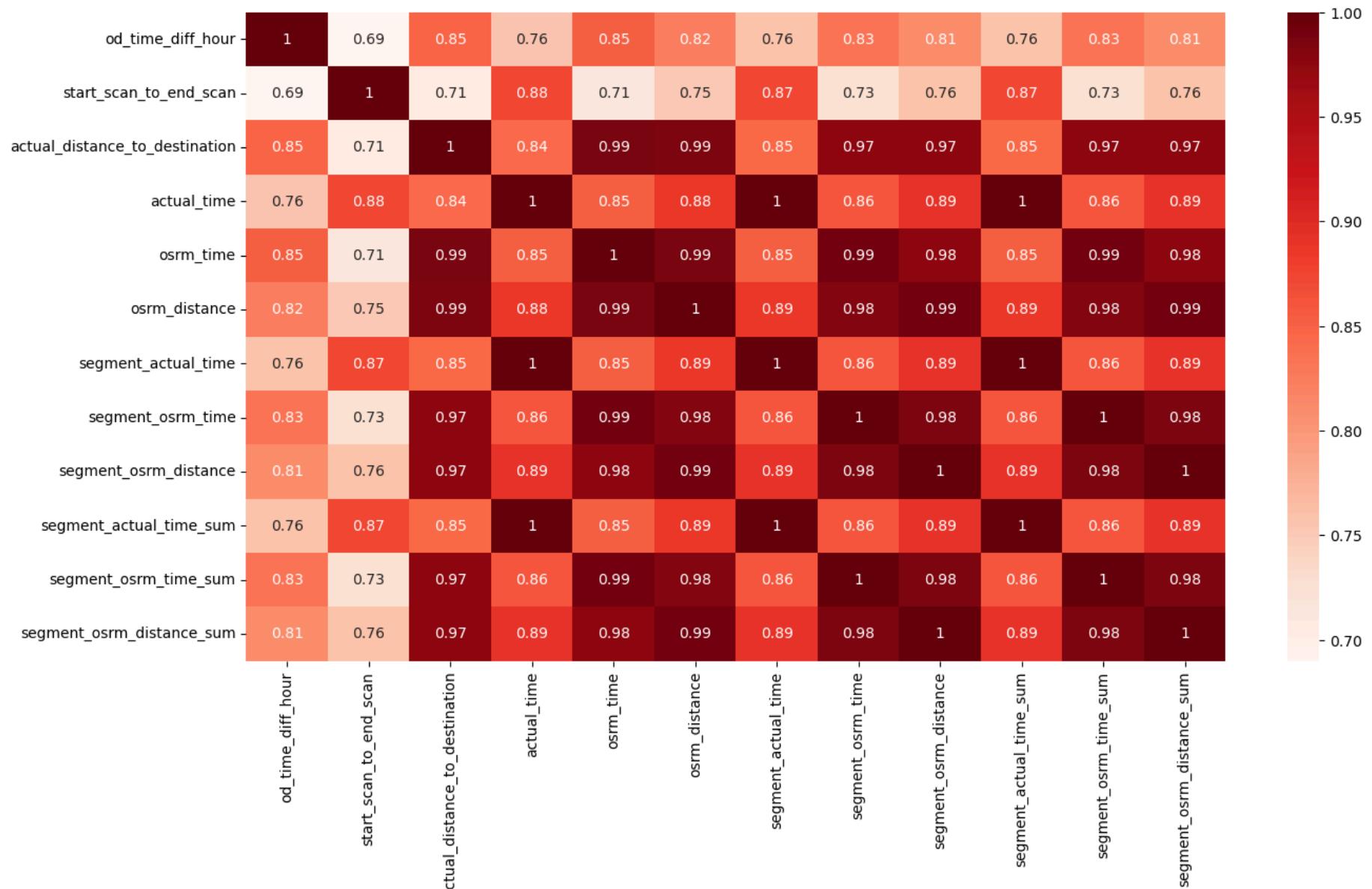
	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_dist
od_time_diff_hour	1.000000	0.999837		0.918644	0.961223	0.926973
start_scan_to_end_scan	0.999837	1.000000		0.919159	0.961612	0.927471
actual_distance_to_destination	0.918644	0.919159		1.000000	0.953920	0.993568
actual_time	0.961223	0.961612		0.953920	1.000000	0.958781
osrm_time	0.926973	0.927471		0.993568	0.958781	1.000000
osrm_distance	0.924683	0.925205		0.997268	0.959398	0.997588
segment_actual_time	0.961288	0.961634		0.952987	0.999989	0.957955
segment_osrm_time	0.918921	0.919429		0.987542	0.954044	0.993263
segment_osrm_distance	0.919665	0.920191		0.993068	0.957151	0.991624
segment_actual_time_sum	0.961288	0.961634		0.952987	0.999989	0.957955
segment_osrm_time_sum	0.918921	0.919429		0.987542	0.954044	0.993263
segment_osrm_distance_sum	0.919665	0.920191		0.993068	0.957151	0.991624



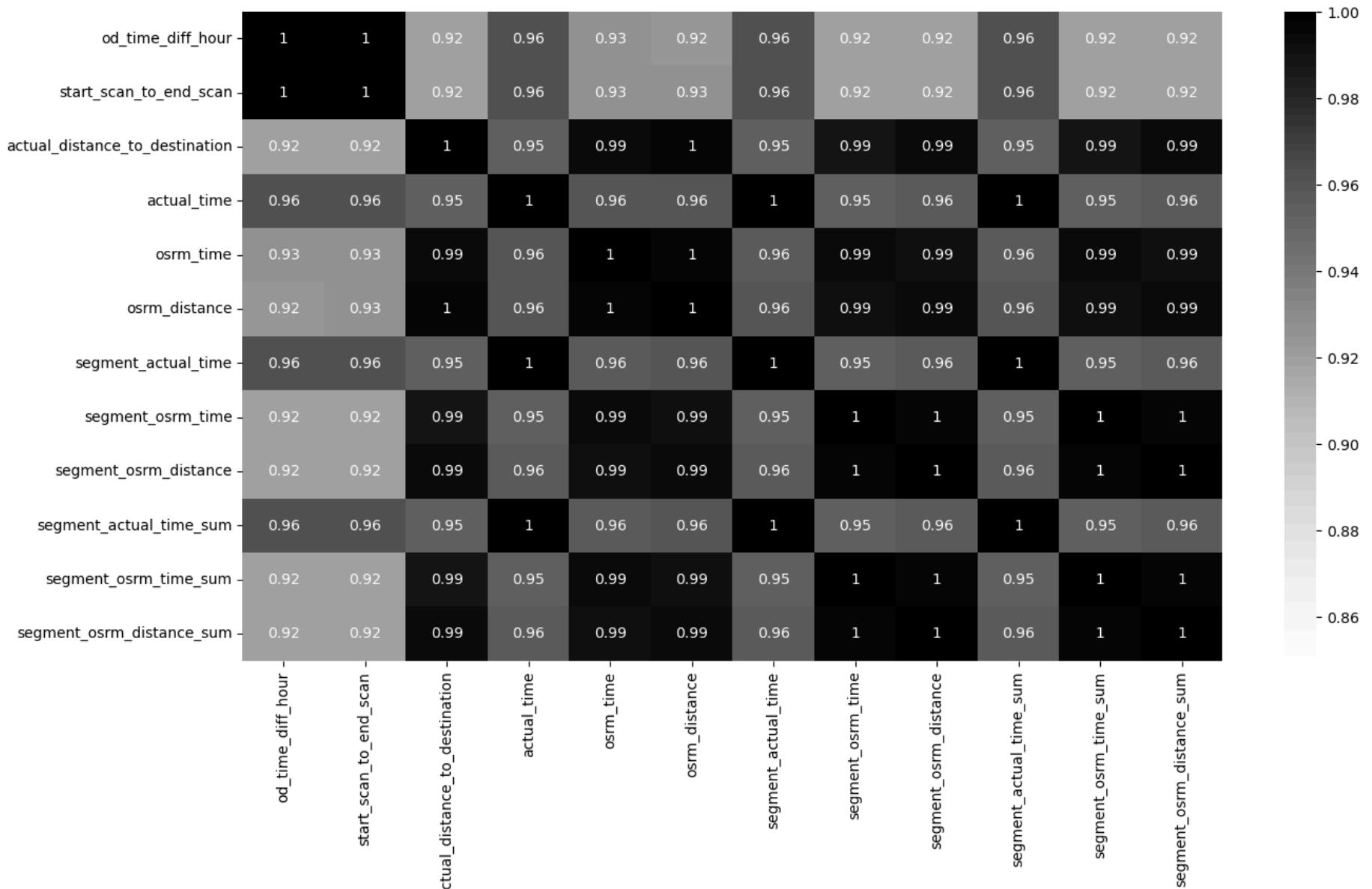
In [145...]

```
plt.figure(figsize = (15,8))
plt.suptitle(f'Correlation Analysis- clipped_df',fontfamily='serif',fontweight='bold',fontsize=15,backgroundcolor='k',color='w')
sns.heatmap(data = clipped_df_corr,vmin=0.69, annot = True, cmap='Reds')
plt.show()

plt.figure(figsize = (15,8))
plt.suptitle(f'Correlation Analysis - filtered_df',fontfamily='serif',fontweight='bold',fontsize=15,backgroundcolor='r',color='w')
sns.heatmap(data = filtered_df_corr,vmin=0.85, annot = True, cmap='Greys')
plt.show()
```

Correlation Analysis- clipped_df

Correlation Analysis - filtered_df



💡 Insights:

- Very High Correlation exists between all the numerical columns.

In [254...]

```
trip_df.skew(numeric_only = True)
```

Out[254...]

```
od_time_diff_hour           2.893550
trip_creation_month          2.337439
trip_creation_year           0.000000
trip_creation_day            -0.695241
trip_creation_hour           -0.206092
trip_creation_weekday        0.065904
trip_creation_week           0.181308
start_scan_to_end_scan       2.895337
actual_distance_to_destination 3.562931
actual_time                  3.375178
osrm_time                    3.455256
osrm_distance                3.553619
segment_actual_time          3.372043
segment_osrm_time             3.602915
segment_osrm_distance         3.714016
segment_actual_time_sum      3.372043
segment_osrm_time_sum         3.602915
segment_osrm_distance_sum    3.714016
trip_creation_dayofdate      -0.695241
dtype: float64
```

💡 Insights:

- We can see that Many of the data is ***Right-Skewed***.

1

⭐ one-hot encoding

In [146...]

```
trip_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14787 entries, 0 to 14786
Data columns (total 37 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   trip_uuid        14787 non-null   object  
 1   data              14787 non-null   category
 2   route_type       14787 non-null   category
 3   od_start_time    14787 non-null   datetime64[ns]
 4   od_end_time      14787 non-null   datetime64[ns]
 5   od_time_diff_hour 14787 non-null   float64 
 6   trip_creation_time 14787 non-null   datetime64[ns]
 7   trip_creation_month 14787 non-null   int32   
 8   trip_creation_year 14787 non-null   int32   
 9   trip_creation_day 14787 non-null   int32   
 10  trip_creation_hour 14787 non-null   int32   
 11  trip_creation_weekday 14787 non-null   int32   
 12  trip_creation_week 14787 non-null   UInt32  
 13  start_scan_to_end_scan 14787 non-null   float32 
 14  actual_distance_to_destination 14787 non-null   float32 
 15  actual_time       14787 non-null   float32 
 16  osrm_time         14787 non-null   float32 
 17  osrm_distance     14787 non-null   float32 
 18  segment_actual_time 14787 non-null   float32 
 19  segment_osrm_time 14787 non-null   float32 
 20  segment_osrm_distance 14787 non-null   float32 
 21  segment_actual_time_sum 14787 non-null   float32 
 22  segment_osrm_time_sum 14787 non-null   float32 
 23  segment_osrm_distance_sum 14787 non-null   float32 
 24  source_name        14787 non-null   object  
 25  source_city         14787 non-null   object  
 26  source_state        14787 non-null   object  
 27  source_place        14787 non-null   object  
 28  destination_name    14787 non-null   object  
 29  destination_city    14787 non-null   object  
 30  destination_state    14787 non-null   object  
 31  destination_place    14787 non-null   object  
 32  corridor            14787 non-null   object  
 33  state_corridor       14787 non-null   object  
 34  city_corridor        14787 non-null   object  
 35  trip_creation_day_week 14787 non-null   object
```

```
36 trip_creation_dayofdate      14787 non-null int32
dtypes: UInt32(1), category(2), datetime64[ns](3), float32(11), float64(1), int32(6), object(13)
memory usage: 3.0+ MB
```

In [147...]: categorical_cols = ['data', 'route_type']

In [149...]: # one hot encoding the categorical features
ohe = OneHotEncoder(sparse=False)
encoded_cat_cols = ohe.fit_transform(trip_df[categorical_cols])
categorical_encoded_df = pd.DataFrame(encoded_cat_cols, columns=ohe.get_feature_names_out(categorical_cols))
display(categorical_encoded_df)
encoded_df = pd.concat([trip_df, categorical_encoded_df], axis=1)
encoded_df

	data_test	data_training	route_type_Carting	route_type_FTL
0	0.0	1.0	0.0	1.0
1	0.0	1.0	1.0	0.0
2	0.0	1.0	0.0	1.0
3	0.0	1.0	1.0	0.0
4	0.0	1.0	0.0	1.0
...
14782	1.0	0.0	1.0	0.0
14783	1.0	0.0	1.0	0.0
14784	1.0	0.0	1.0	0.0
14785	1.0	0.0	1.0	0.0
14786	1.0	0.0	0.0	1.0

14787 rows × 4 columns

Out[149...]

		trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creation_mont
0	153671041653548748	trip-training	training	FTL	2018-09-12 16:39:46.858469	2018-09-12 16:39:46.858469	37.668497	2018-09-12 00:00:16.535741	
1	153671042288605164	trip-training	training	Carting	2018-09-12 02:03:09.655591	2018-09-12 02:03:09.655591	3.026865	2018-09-12 00:00:22.886430	
2	153671043369099517	trip-training	training	FTL	2018-09-14 03:40:17.106733	2018-09-14 03:40:17.106733	65.572709	2018-09-12 00:00:33.691250	
3	153671046011330457	trip-training	training	Carting	2018-09-12 00:01:00.113710	2018-09-12 01:41:29.809822	1.674916	2018-09-12 00:01:00.113710	
4	153671052974046625	trip-training	training	FTL	2018-09-12 00:02:09.740725	2018-09-12 03:54:43.114421	11.972484	2018-09-12 00:02:09.740725	
...
14782	153861095625827784	trip-test	test	Carting	2018-10-03 23:55:56.258533	2018-10-04 06:41:25.409035	4.300482	2018-10-03 23:55:56.258533	1
14783	153861104386292051	trip-test	test	Carting	2018-10-03 23:57:23.863155	2018-10-04 00:57:59.294434	1.009842	2018-10-03 23:57:23.863155	1

	trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creation_mont
14784	trip-153861106442901555	test	Carting	2018-10-04 02:51:27.075797	2018-10-04 02:51:27.075797	7.035331	2018-10-03 23:57:44.429324	1
14785	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	2018-10-04 02:29:04.272194	5.808548	2018-10-03 23:59:14.390954	1
14786	trip-153861118270144424	test	FTL	2018-10-04 03:58:40.726547	2018-10-04 03:58:40.726547	5.906793	2018-10-03 23:59:42.701692	1

14787 rows × 41 columns

0 - 1 Minmax scaler

- ◆ Most appropriate since the data is not gaussian

In [150...]

```
# Normalizing/Standardizing the numerical features using MinMaxScaler
min_max_scaler = MinMaxScaler()
min_max_scaled_numerical = min_max_scaler.fit_transform(trip_df[num_cols])

# Converting the scaled features back to a dataframe
min_max_scaled_df = pd.DataFrame(min_max_scaled_numerical, columns=num_cols)
min_max_scaled_df
```

Out[150...]

	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_tir
0	0.284016	0.283937	0.374613	0.248242	0.350938	0.346972	0.2473
1	0.020082	0.019937	0.029476	0.021419	0.030602	0.026859	0.0212
2	0.496617	0.496508	0.880999	0.533568	0.855874	0.828325	0.5303
3	0.009782	0.009778	0.003753	0.007992	0.004442	0.003747	0.0080
4	0.088239	0.088127	0.054395	0.053069	0.054788	0.048647	0.0532
...
14782	0.029786	0.029714	0.022392	0.011829	0.027641	0.022745	0.0117
14783	0.004715	0.004698	0.002990	0.001918	0.002962	0.002478	0.0019
14784	0.050623	0.050540	0.013631	0.043638	0.020731	0.017602	0.0437
14785	0.041276	0.041143	0.057736	0.040761	0.085390	0.057237	0.0400
14786	0.042024	0.041905	0.026213	0.042519	0.030602	0.025258	0.0425

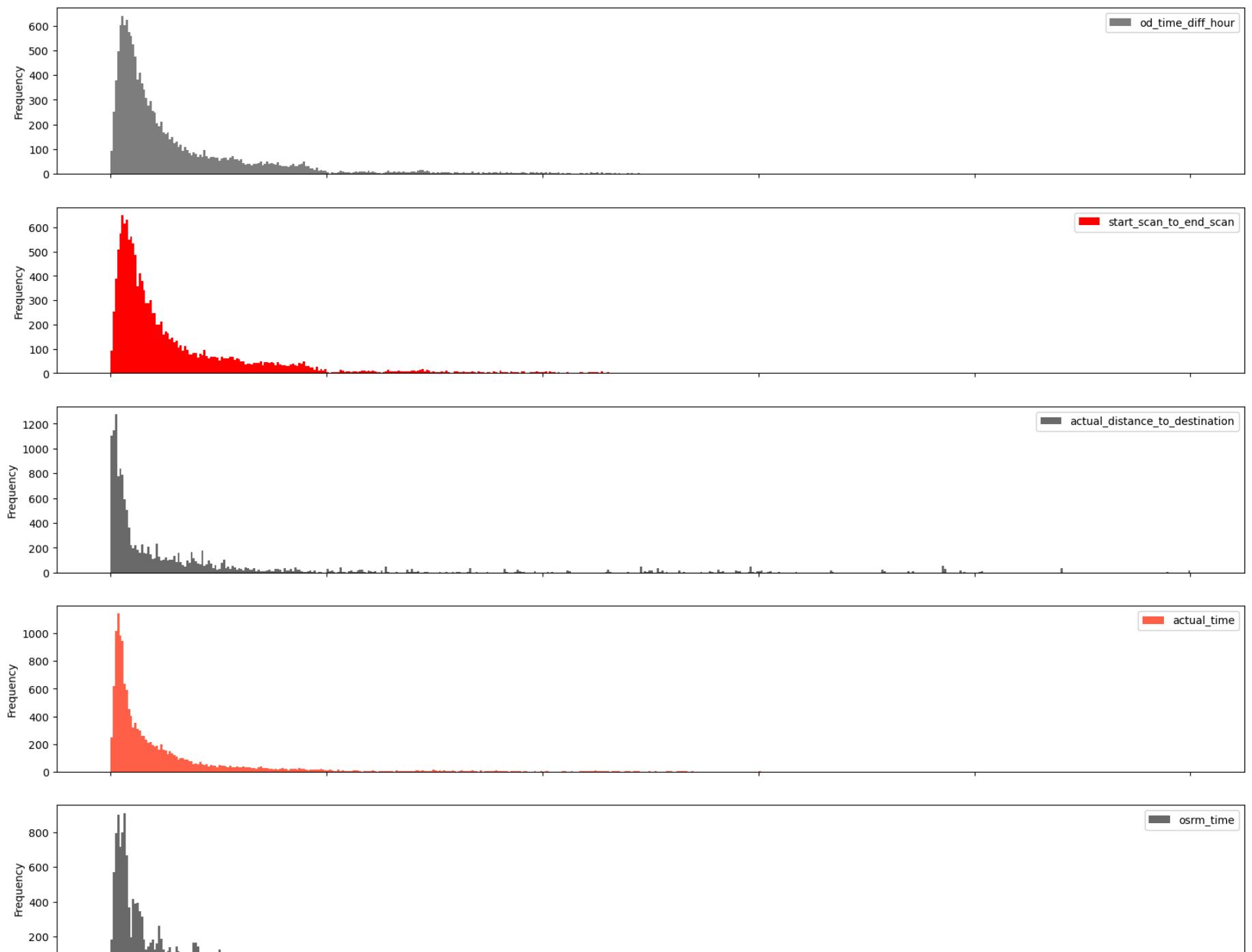
14787 rows × 12 columns

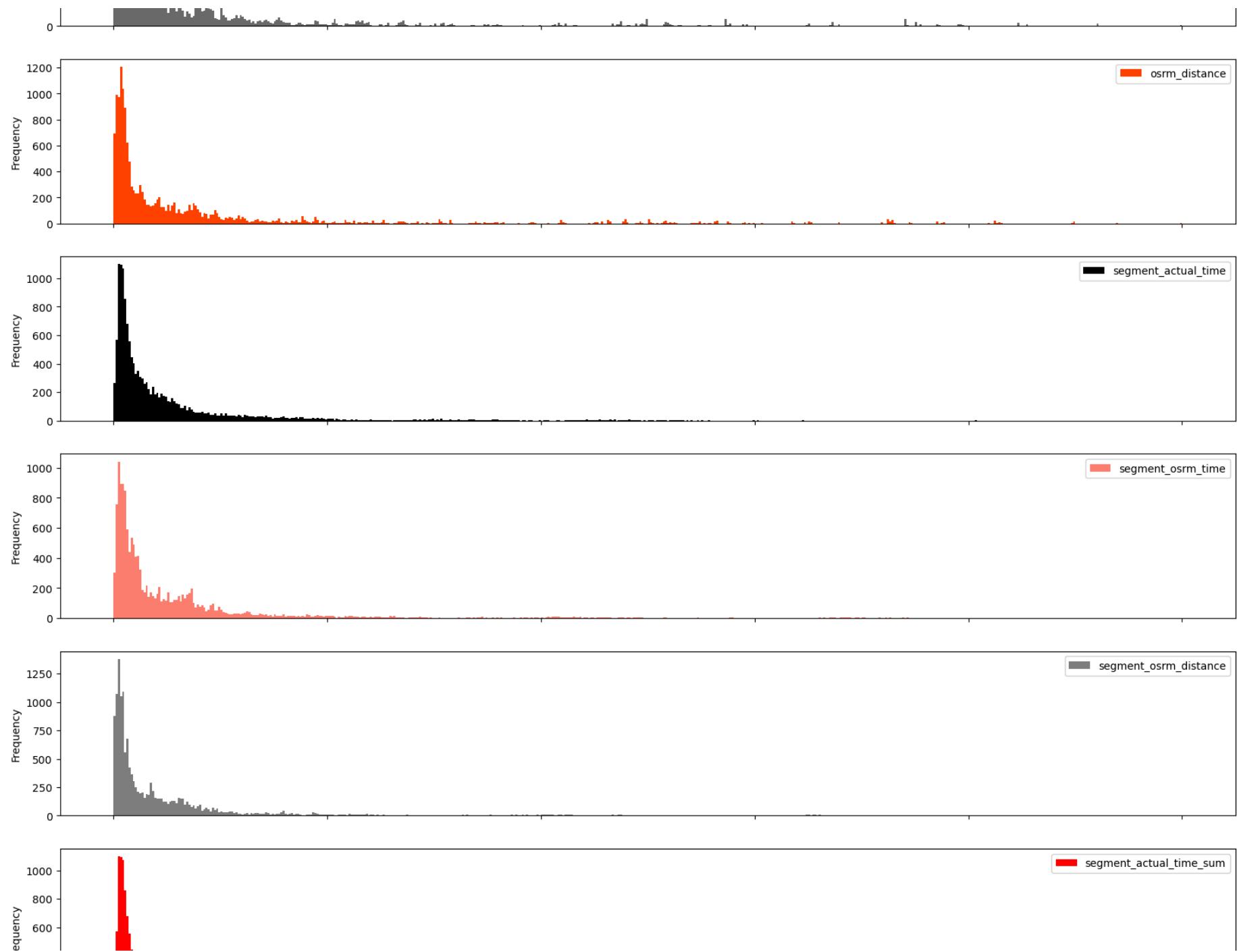


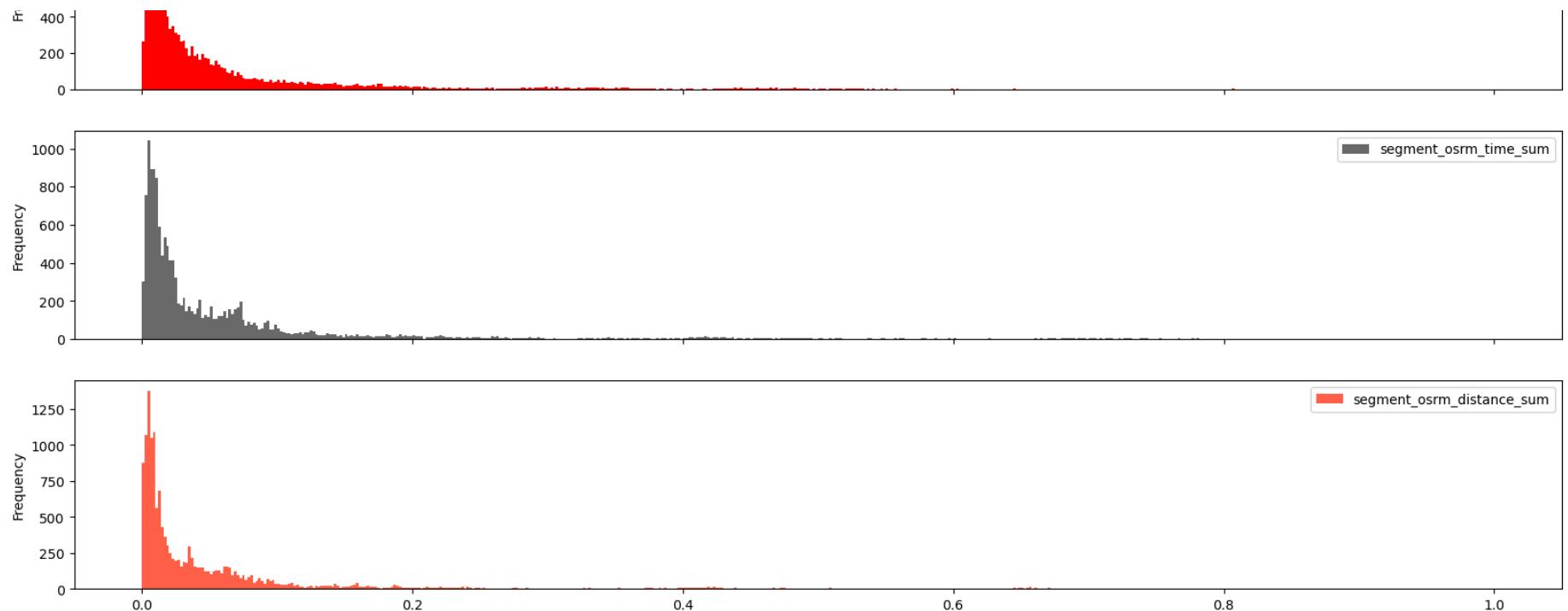
In [155...]

```
plt.figure(figsize=(14, 0.05))
plt.axis('off')
plt.suptitle(f'Min-Max scaled visualization of num_cols', fontfamily='serif', fontweight='bold', fontsize=15, backgroundcolor='k',
min_max_scaled_df.plot(kind='hist', figsize=(20,40), subplots=True, color=cp, bins=500)
plt.show()
```

Min-Max scaled visualization of num_cols







In [156...]

```
# Just so to know ... cant do this as `data is not gaussian`
# Standardization works only with data which follows normal distribution
# Standardizing the numerical features using StandardScaler
std_scaler = StandardScaler()
std_scaled = std_scaler.fit_transform(trip_df[num_cols])

# Converting the scaled features back to a dataframe
std_scaled_df = pd.DataFrame(std_scaled, columns=num_cols)
std_scaled_df
```

Out[156...]

	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_tir
0	2.625886	2.627598	2.162548	2.147277	2.048290	2.125107	2.1478
1	-0.529518	-0.530859	-0.297563	-0.379887	-0.342571	-0.320538	-0.3811
2	5.167598	5.170772	5.772034	5.326268	5.816936	5.802622	5.3113
3	-0.652664	-0.652397	-0.480911	-0.529486	-0.537818	-0.497115	-0.5285
4	0.285312	0.284962	-0.119943	-0.027259	-0.162059	-0.154082	-0.0234
...
14782	-0.413508	-0.413880	-0.348054	-0.486744	-0.364674	-0.351972	-0.4872
14783	-0.713243	-0.713166	-0.486350	-0.597162	-0.548870	-0.506808	-0.5968
14784	-0.164399	-0.164728	-0.410502	-0.132335	-0.416249	-0.391263	-0.1295
14785	-0.276143	-0.277150	-0.096128	-0.164392	0.066344	-0.088455	-0.1708
14786	-0.267194	-0.268034	-0.320822	-0.144802	-0.342571	-0.332769	-0.1421

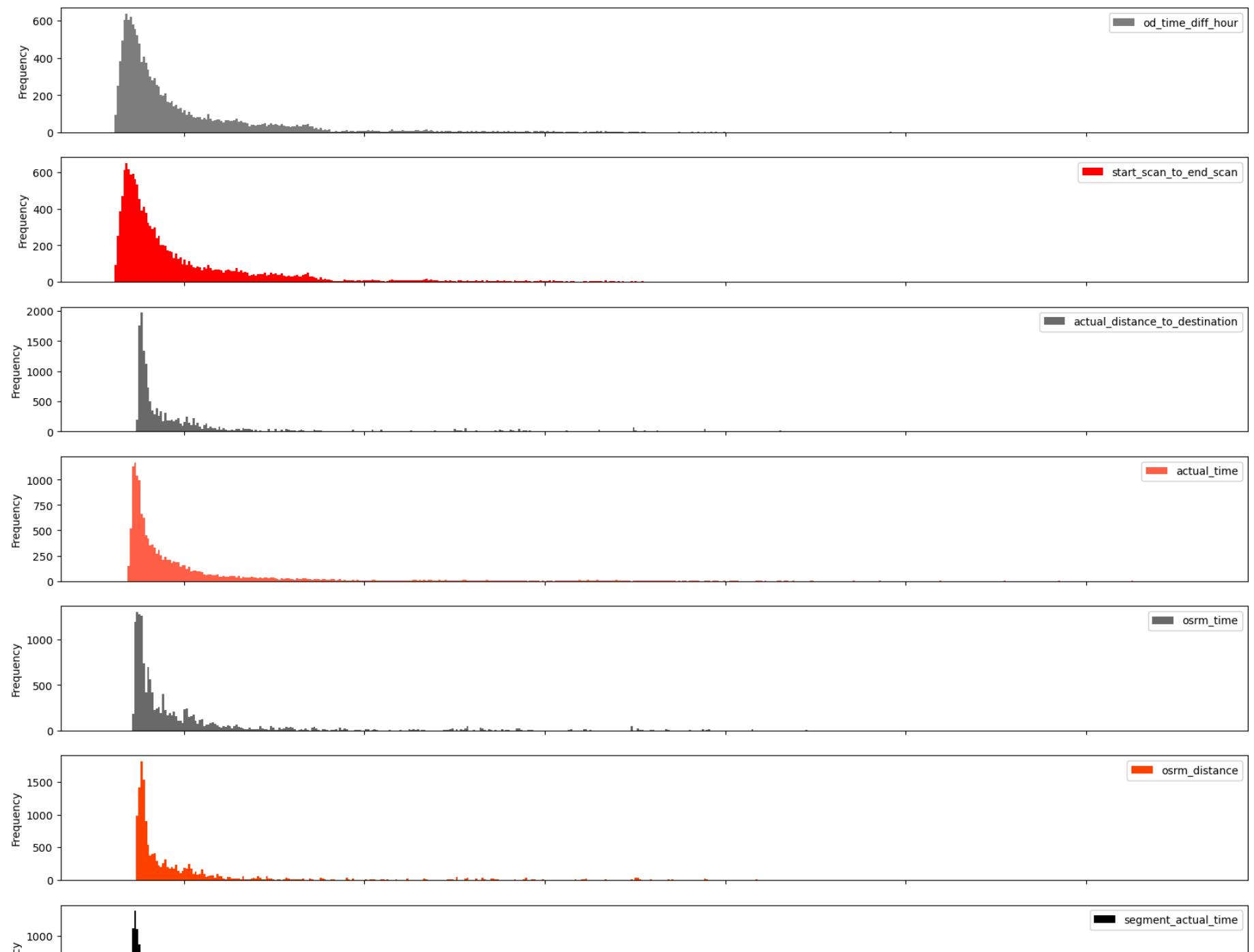
14787 rows × 12 columns

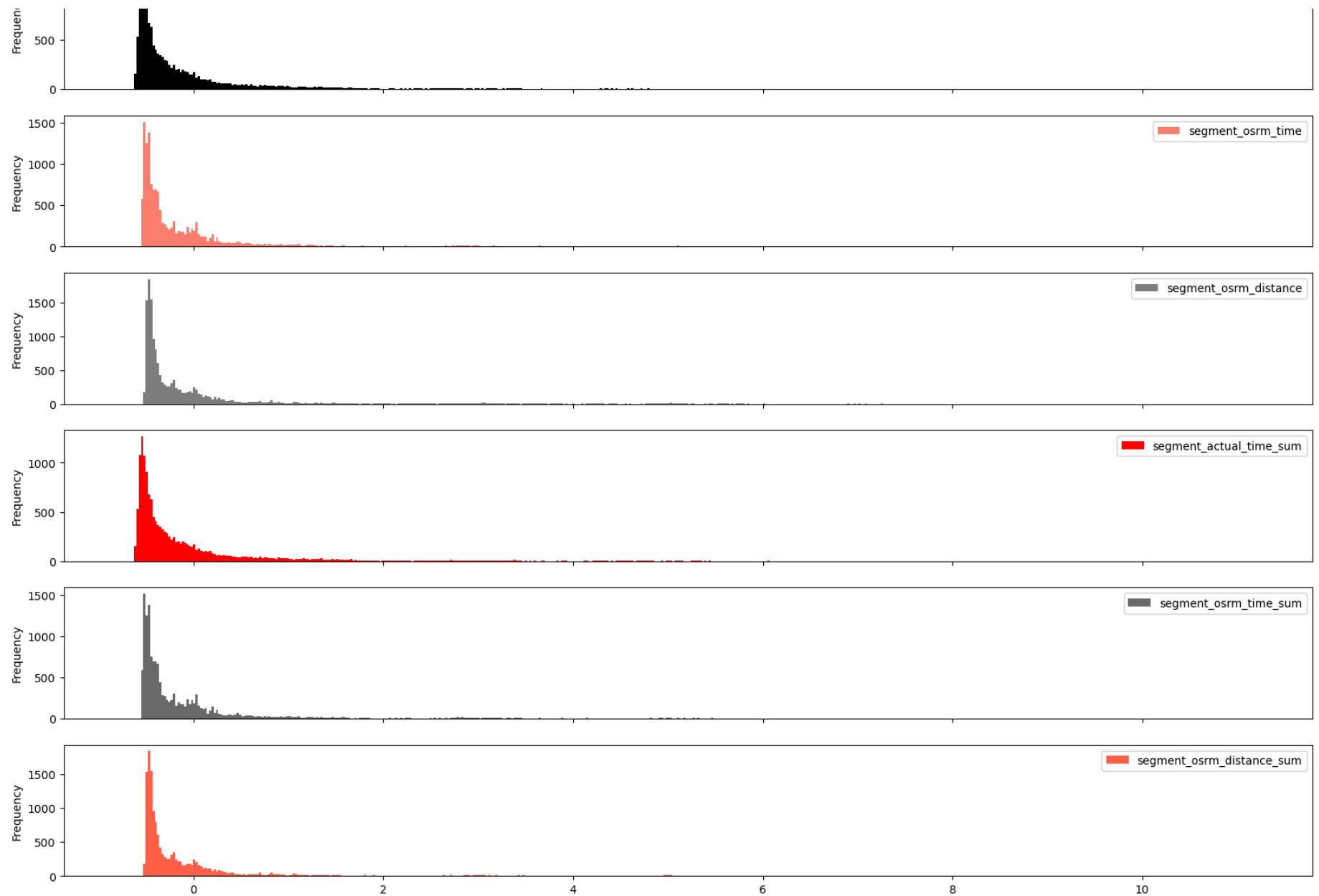


In [157...]

```
plt.figure(figsize=(14, 0.05))
plt.axis('off')
plt.suptitle(f'Standardized Num_cols scaled visualization', fontfamily='serif', fontweight='bold', fontsize=15, backgroundcolor='red')
std_scaled_df.plot(kind='hist', figsize=(20, 30), subplots=True, color=cp, bins=500)
plt.show()
```

Standardized Num_cols scaled visualization





👉 Hypothesis Testing:

- Perform hypothesis testing / visual analysis between :
 - ⚡ actual_time aggregated value and OSRM time aggregated value.
 - ⚡ actual_time aggregated value and segment actual time aggregated value.
 - ⚡ OSRM distance aggregated value and segment OSRM distance aggregated value.
 - ⚡ OSRM time aggregated value and segment OSRM time aggregated value.
- Note: Aggregated values are the values you'll get after merging the rows on the basis of trip_uuid.

Assumptions of T-Test

1. The sample size should be less than 30.
2. The population variance is unknown.
3. The population mean and standard deviation are finite.
4. The means of the two populations being compared should follow normal distributions.
5. If using Student's original definition of the t-test, the two populations being compared should have the same variance. If the sample sizes in the two groups being compared are equal, Student's original t-test is highly robust to the presence of unequal variances.

⚡ *STEP-1* : Set up Null Hypothesis

Null Hypothesis (H₀) - There is no significant difference in the mean values between column1 and column2

\$H_{0}\$\$: \mu_{\text{col1}} = \mu_{\text{col2}}

Alternate Hypothesis (H_a) - There is a significant difference in the mean values between column1 and column2

\$H_{a}\$\$: \mu_{\text{col1}} \neq \mu_{\text{col2}}

⚡ *STEP-2* : Checking for basic assumptions for the hypothesis

Normality checks

- Distribution check using **QQ Plot & prob Plot**
 - Confirmation by **Shapiro-wilks Test**
 - Confirmation by **Anderson-darling Test**
 - Homogeneity of Variances using **Levene's test**
-

⚡ *STEP-3*: Define Test statistics; Distribution of T under H_0 .

- We know that the test statistic while performing a T-Test follows T-distribution.

for independent variables:

```
>> If data follows normal distribution we go with **ttest_ind**  
>> Else we will go with **Mannwhitney_u test** (Non - Parametric test)
```

for dependent variables: (paired T-test)

```
>> If data follows normal distribution we go with **ttest_rel**  
>> Else we will go with **Wilcoxon signed rank test** (Non - Parametric test)
```

⚡ *STEP-4*: Decide the kind of test.

- We will be performing **Two tailed t-test**
-

⚡ *STEP-5*: Compute the p-value and fix value of alpha.

- we will be computing the t-stat value using the ttest function using scipy.stats.
 - We set our ***alpha to be 0.05 (i.e) confidence level = 95%***
-

⚡ *STEP-6*: Compare p-value and alpha.

- Based on p-value, we will accept or reject H0.

 | **p-val < alpha** : Reject H0

 | **p-val > alpha** : Accept H0

In [198...]

```
# def shapiro_and_anderson(name,col):

#     print(f"Performing SHAPIRO & ANDERSON DARLING TEST for {name} column")
#     print()
#     print('Shapiro Wilks Test')
#     shapiro_stat , p_val = shapiro(col)
#     if p_val < 0.05:
#         print(f'{name} - Data is not Gaussian')
#     else:
#         print(f'{name} - Data is Gaussian')
#     print()

#     print("As shapiro is sensitive, we go with ANDERSON DARLING TEST")
#     result = anderson(col)
#     if result.statistic > result.critical_values[2]:
#         print(f"{name} - Data does not follow normal distribution.")
#     else:
#         print(f"{name} - Data follows normal distribution.")
#     print()
#     print('-'*50)

#     def boxcox_transformation(name,col):

#         print(f'Performing BOXCOX transformation on {name} column')
```

```
#         transformed_data,best_lambda = boxcox(col)

#         return " "

# class NormalCheck:
#     def __init__(self, name, col):
#         self.name = name
#         self.col = col

#     def perform_checks(self):
#         boxcox_transformation(self.name, self.col)
#         shapiro_and_anderson(self.name, self.col)
```

In [207...]

```
class Normality_check:
    def __init__(self, name, col):
        self.name = name
        self.col = col

    def shapiro_and_anderson(self):
        print(f"Performing SHAPIRO & ANDERSON-DARLING TEST for {self.name} column")
        print()

        # Shapiro-Wilk Test
        print('Shapiro-Wilk Test')
        shapiro_stat, p_val = shapiro(self.col)
        if p_val < 0.05:
            print(f'{self.name} - Data is not Gaussian')
        else:
            print(f'{self.name} - Data is Gaussian')
        print()

        # Using Anderson-Darling Test
        print("Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test")
        result = anderson(self.col)
        if result.statistic > result.critical_values[2]:
            print(f'{self.name} - Data does not follow a normal distribution.')
        else:
            print(f'{self.name} - Data follows a normal distribution.')
        print()
```

```
print('-'*50)

def boxcox_transformation(self):
    print(f'Performing BOXCOX transformation on {self.name} column')
    transformed_data, best_lambda = boxcox(self.col)
    self.col = transformed_data # Update column data with transformed data
    self.shapiro_and_anderson() # Calling shapiro_and_anderson method after transformation

# normality_check = NormalCheck(name, col)
# normality_check.boxcox_transformation()
```

In [203...]

```
def levene_test(name1,name2,col1,col2):

    levene_stat, p_value = levene(col1,col2)

    print(f'Performing Levene Test for {name1} & {name2}')

    if p_value < 0.05:
        print('Does not have Homogenous (different) Variance')
    else:
        print('Have Homogenous (similar) variance')
    print()
    print('-'*50)
    print()
    return ""
```

In [216...]

```
## MannWhitney u Rank test
### Test statistics : Mann-Whitney U rank test for two independent samples

def mannwhitneyu_test(name1,name2,col1,col2):

    print(f'Performing Non-parametric Test - MannWhitneyU for {name1} & {name2}')
    test_stat, p_value = mannwhitneyu(col1,col2)

    if p_value < 0.05:
        print("Reject Null Hypothesis")
        print(f'There is a significant difference in the Mean values of {name1} and {name2}')
    else:
        print("Failed to Reject Null Hypothesis - Accept Ho")
```

```
    print(f'There is NO significant difference in the Mean values of {name1} and {name2}')\n\nprint()\nprint('-'*50)\nprint()\n\nreturn ""
```

In [227...]

```
def normality_plots(name1,name2,name3,name4,col1,col2,col3,col4):\n\n    plt.figure(figsize = (20,10))\n    plt.suptitle("Normality check - Histplot & QQ(prob)plot",fontsize=16,fontweight="bold",backgroundcolor=cp[5],color='w')\n\n    plt.subplot(241)\n    sns.histplot(col1, element = 'step', color = cp[1], kde = True, label = name1)\n    plt.title(f'Histplot - {name1}',fontsize=10,fontweight="bold",backgroundcolor=cp[1],color='w')\n    plt.legend()\n\n    plt.subplot(242)\n    sns.histplot(col3,element = 'step', color = cp[2], kde = True, label = name3 )\n    plt.title(f'Histplot - {name3}',fontsize=10,fontweight="bold",backgroundcolor=cp[2],color='w')\n    plt.legend()\n\n    plt.subplot(243)\n    probplot(col1, plot = plt, dist = 'norm')\n    plt.title(f'Probplot - {name1}',fontsize=10,fontweight="bold",backgroundcolor=cp[1],color='w')\n\n    plt.subplot(244)\n    probplot(col3, plot = plt, dist = 'norm')\n    plt.title(f'Probplot - {name3}',fontsize=10,fontweight="bold",backgroundcolor=cp[2],color='w')\n\n    plt.subplot(245)\n    sns.histplot(col2, element = 'step', color = cp[1], kde = True, label = name2 )\n    plt.title(f'Histplot - {name2}',fontsize=10,fontweight="bold",backgroundcolor=cp[1],color='w')\n    plt.legend()\n\n    plt.subplot(246)\n    sns.histplot(col4,element = 'step', color = cp[2], kde = True, label = name4)\n    plt.title(f'Histplot - {name4}',fontsize=10,fontweight="bold",backgroundcolor=cp[2],color='w')\n    plt.legend()
```

```

plt.subplot(247)
probplot(col2, plot = plt, dist = 'norm')
plt.title(f'Probplot - {name2}', fontsize=10, fontweight="bold", backgroundcolor=cp[1], color='w')

plt.subplot(248)
probplot(col4, plot = plt, dist = 'norm')
plt.title(f'Probplot - {name4}', fontsize=10, fontweight="bold", backgroundcolor=cp[2], color='w')

sns.despine()
plt.show()

```

⚡ Hypothesis testing - actual_time aggregated value and OSRM time aggregated value.

In [158...]: clipped_num_df.sample(3)

	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time
4578	10.118006	543.285302		220.068207	353.0	221.0	249.271805
176	1.358135	81.000000		9.083277	28.0	11.0	14.245600
505	6.646569	398.000000		101.146736	259.0	98.0	130.815598

In [219...]: clipped_num_df[['actual_time', 'osrm_time']].describe().T

	count	mean	std	min	25%	50%	75%	max
actual_time	14787.0	224.212577	185.922840	9.0	67.0	148.0	367.0	543.285302
osrm_time	14787.0	128.190912	150.301267	6.0	29.0	60.0	168.0	543.285302

In [167...]: filtered_num_df[['actual_time', 'osrm_time']].describe().T

Out[167...]

	count	mean	std	min	25%	50%	75%	max
actual_time	14787.0	356.306000	561.517761	9.0	67.0	148.0	367.0	6265.0
osrm_time	14787.0	160.990936	271.459229	6.0	29.0	60.0	168.0	2032.0

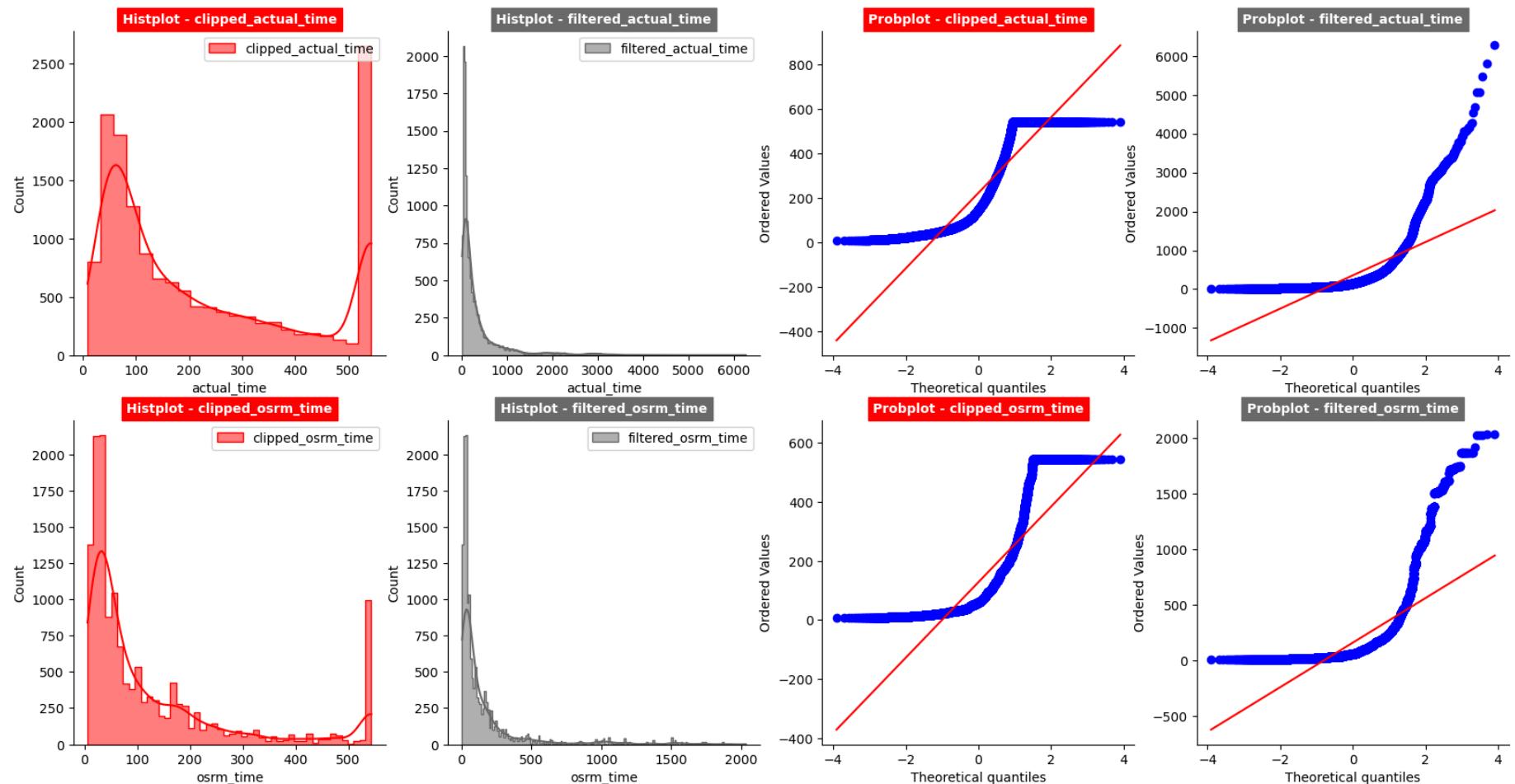
In [169...]

```
actual_time = clipped_num_df['actual_time']
osrm_time = clipped_num_df['osrm_time']
fil_actual_time = filtered_num_df['actual_time']
fil_osrm_time = filtered_num_df['osrm_time']
```

In [228...]

```
normality_plots('clipped_actual_time','clipped_osrm_time','filtered_actual_time','filtered_osrm_time',actual_time,osrm_time,fi
```

Normality check - Histplot & QQ(prob)plot



In [213...]

```
col_names= ['clipped_actual_time','clipped_osrm_time','filtered_actual_time','filtered_osrm_time']
cols = [actual_time,osrm_time,fil_actual_time,fil_osrm_time]

for _ in zip(col_names,cols):
    normality = Normality_check(_[0],_[1])
    normality.shapiro_and_anderson()
    normality.boxcox_transformation()
```

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_actual_time column

Shapiro-Wilk Test

clipped_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_actual_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on clipped_actual_time column

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_actual_time column

Shapiro-Wilk Test

clipped_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_actual_time - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_osrm_time column

Shapiro-Wilk Test

clipped_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_osrm_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on clipped_osrm_time column

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_osrm_time column

Shapiro-Wilk Test

clipped_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_osrm_time - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_actual_time column

Shapiro-Wilk Test

```
filtered_actual_time - Data is not Gaussian
```

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_actual_time - Data does not follow a normal distribution.

```
-----  
Performing BOXCOX transformation on filtered_actual_time column
```

```
Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_actual_time column
```

```
Shapiro-Wilk Test
```

```
filtered_actual_time - Data is not Gaussian
```

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_actual_time - Data does not follow a normal distribution.

```
-----  
Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_osrm_time column
```

```
Shapiro-Wilk Test
```

```
filtered_osrm_time - Data is not Gaussian
```

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_osrm_time - Data does not follow a normal distribution.

```
-----  
Performing BOXCOX transformation on filtered_osrm_time column
```

```
Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_osrm_time column
```

```
Shapiro-Wilk Test
```

```
filtered_osrm_time - Data is not Gaussian
```

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_osrm_time - Data does not follow a normal distribution.

```
-----  
In [204...]
```

```
levene_test('clipped_actual_time','clipped_osrm_time',actual_time,osrm_time),  
levene_test('filtered_actual_time','filtered_osrm_time',fil_actual_time,fil_osrm_time)
```

Performing Levene Test for clipped_actual_time & clipped_osrm_time
Does not have Homogenous (different) Variance

Performing Levene Test for filtered_actual_time & filtered_osrm_time
Does not have Homogenous (different) Variance

Out[204...]: ''

Wilcoxon signed rank test:

In [214...]

```
## with clipped data

# H0 : aggregated actual time is same as aggregated osrm time
# Ha : aggregated actual time is more than the aggregated osrm time

alpha = 0.05 #testing at 95% confidence

test_stat , p_value = wilcoxon(actual_time,osrm_time,alternative='greater')

if p_value < alpha:
    print("Reject Null Hypothesis - The Aggregated Actual_time is More than the Aggregated OSRM_time")
else:
    print("Fail to Reject Null Hypothesis - The Aggregated Actual_time is same as the Aggregated OSRM_time")
```

Reject Null Hypothesis - The Aggregated Actual_time is More than the Aggregated OSRM_time

In [215...]

```
## with filtered data

alpha = 0.05 #testing at 95% confidence

test_stat , p_value = wilcoxon(fil_actual_time,fil_osrm_time,alternative='greater')

if p_value < alpha:
    print("Reject Null Hypothesis - The Aggregated Actual_time is More than the Aggregated OSRM_time")
```

```
else:
    print("Fail to Reject Null Hypothesis - The Aggregated Actual_time is same as the Aggregated OSRM_time")
```

Reject Null Hypothesis - The Aggregated Actual_time is More than the Aggregated OSRM_time

In [218]:

```
### MannWhitney u Rank test

test_cols = [('clipped_actual_time','clipped_osrm_time',actual_time,osrm_time),
             ('filtered_actual_time','filtered_osrm_time',fil_actual_time,fil_osrm_time)]

for _ in test_cols:
    mannwhitneyu_test(_[0],_[1],_[2],_[3])
```

Performing Non-parametric Test - MannWhitneyU for clipped_actual_time & clipped_osrm_time

Reject Null Hypothesis

There is a significant difference in the Mean values of clipped_actual_time and clipped_osrm_time

Performing Non-parametric Test - MannWhitneyU for filtered_actual_time & filtered_osrm_time

Reject Null Hypothesis

There is a significant difference in the Mean values of filtered_actual_time and filtered_osrm_time

💡 Insights:

It is confirmed that There is a significant difference in the Mean values of Aggregated actual_time and Aggregated osrm_time through *MannwhitneyU* test.

- $H_0: \mu_{\text{Aggregated-actual-time}} \neq \mu_{\text{Aggregated-osrm-time}}$

Further, it is found that The Aggregated Actual_time is More than the Aggregated OSRM_time through *Wilcoxon signed Rank* test.

- $H_0: \mu_{\text{Aggregated-actual-time}} > \mu_{\text{Aggregated-osrm-time}}$

⚡ Actual_time aggregated value and Segment actual time aggregated value.

In [220...]: `clipped_num_df.sample()`

	<code>od_time_diff_hour</code>	<code>start_scan_to_end_scan</code>	<code>actual_distance_to_destination</code>	<code>actual_time</code>	<code>osrm_time</code>	<code>osrm_distance</code>	<code>segment_actual_time</code>
11839	4.644768	278.0		46.165558	104.0	63.0	75.968498



In [221...]: `clipped_num_df[['actual_time', 'segment_actual_time']].describe().T`

	<code>count</code>	<code>mean</code>	<code>std</code>	<code>min</code>	<code>25%</code>	<code>50%</code>	<code>75%</code>	<code>max</code>
<code>actual_time</code>	14787.0	224.212577	185.922840	9.0	67.0	148.0	367.0	543.285302
<code>segment_actual_time</code>	14787.0	222.904643	185.797003	9.0	66.0	147.0	364.0	543.285302

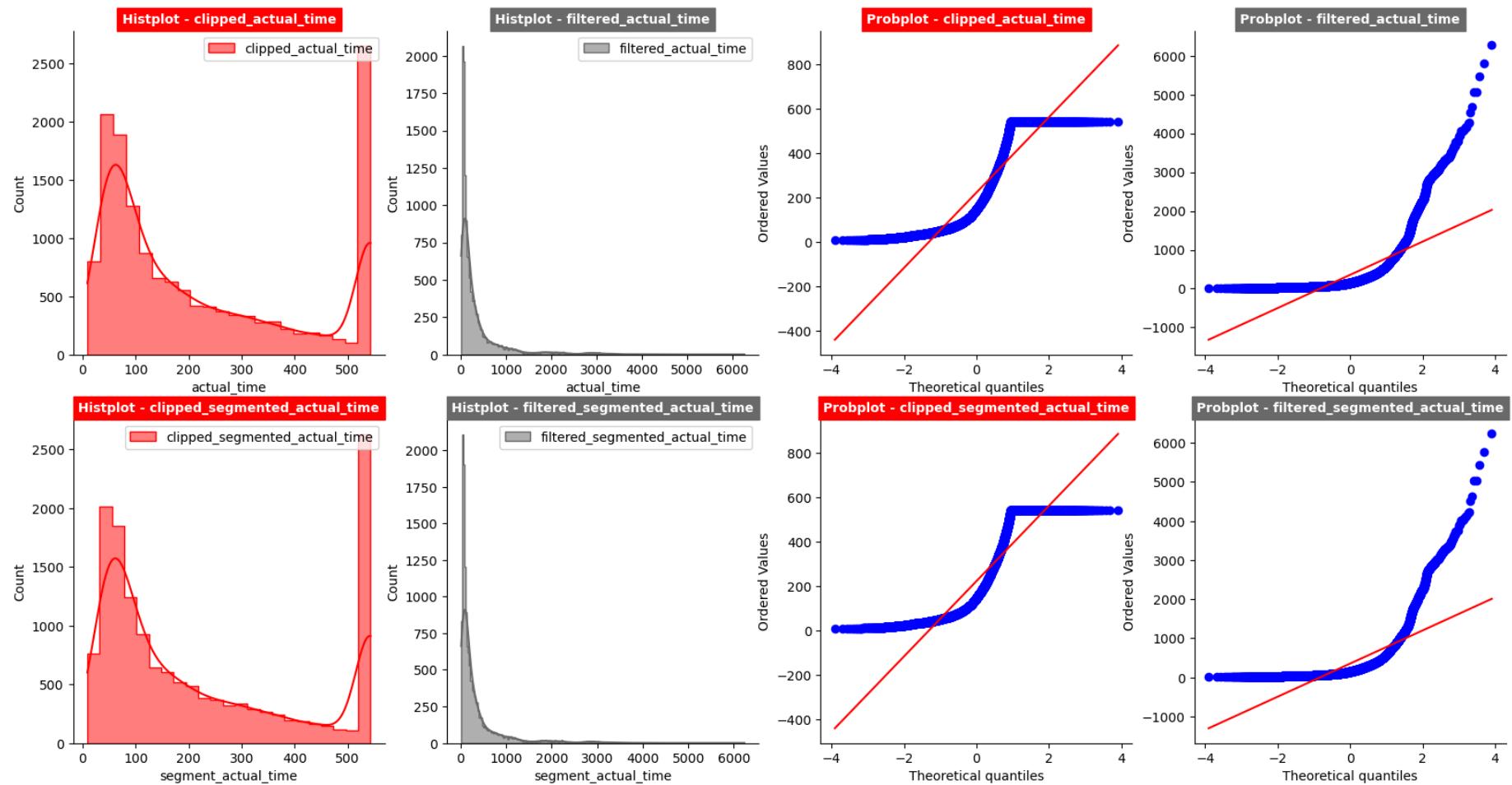
In [222...]: `filtered_num_df[['actual_time', 'segment_actual_time']].describe().T`

	<code>count</code>	<code>mean</code>	<code>std</code>	<code>min</code>	<code>25%</code>	<code>50%</code>	<code>75%</code>	<code>max</code>
<code>actual_time</code>	14787.0	356.306000	561.517761	9.0	67.0	148.0	367.0	6265.0
<code>segment_actual_time</code>	14787.0	353.059174	556.364441	9.0	66.0	147.0	364.0	6230.0

In [223...]:
`clipped_actual_time = clipped_num_df['actual_time']`
`clipped_segmented_actual_time = clipped_num_df['segment_actual_time']`
`filtered_actual_time = filtered_num_df['actual_time']`
`filtered_segmented_actual_time = filtered_num_df['segment_actual_time']`

In [229...]: `normality_plots("clipped_actual_time", "clipped_segmented_actual_time", "filtered_actual_time", "filtered_segmented_actual_time",
 clipped_actual_time, clipped_segmented_actual_time, filtered_actual_time, filtered_segmented_actual_time)`

Normality check - Histplot & QQ(prob)plot



In [230...]

```
col_names= ["clipped_actual_time","clipped_segmented_actual_time","filtered_actual_time","filtered_segmented_actual_time"]
cols = [clipped_actual_time,clipped_segmented_actual_time,filtered_actual_time,filtered_segmented_actual_time]

for _ in zip(col_names,cols):
    normality = Normality_check(_[0],_[1])
    normality.shapiro_and_anderson()
    normality.boxcox_transformation()
```

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_actual_time column

Shapiro-Wilk Test

clipped_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_actual_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on clipped_actual_time column

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_actual_time column

Shapiro-Wilk Test

clipped_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_actual_time - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_segmented_actual_time column

Shapiro-Wilk Test

clipped_segmented_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_segmented_actual_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on clipped_segmented_actual_time column

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_segmented_actual_time column

Shapiro-Wilk Test

clipped_segmented_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_segmented_actual_time - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_actual_time column

Shapiro-Wilk Test

filtered_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_actual_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on filtered_actual_time column

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_actual_time column

Shapiro-Wilk Test

filtered_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_actual_time - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_segmented_actual_time column

Shapiro-Wilk Test

filtered_segmented_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_segmented_actual_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on filtered_segmented_actual_time column

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_segmented_actual_time column

Shapiro-Wilk Test

filtered_segmented_actual_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_segmented_actual_time - Data does not follow a normal distribution.

In [232...]

```
levene_test("clipped_actual_time","clipped_segmented_actual_time",clipped_actual_time,clipped_segmented_actual_time),  
levene_test("filtered_actual_time","filtered_segmented_actual_time",filtered_actual_time,filtered_segmented_actual_time)
```

Performing Levene Test for clipped_actual_time & clipped_segmented_actual_time
Have Homogenous (similar) variance

Performing Levene Test for filtered_actual_time & filtered_segmented_actual_time
Have Homogenous (similar) variance

Out[232...]

In [231...]

```
### MannWhitney u Rank test

test_cols = [("clipped_actual_time", "clipped_segmented_actual_time", clipped_actual_time, clipped_segmented_actual_time),
             ("filtered_actual_time", "filtered_segmented_actual_time", filtered_actual_time, filtered_segmented_actual_time)]

for _ in test_cols:
    mannwhitneyu_test(_[0],_[1],_[2],_[3])
```

Performing Non-parametric Test - MannWhitneyU for clipped_actual_time & clipped_segmented_actual_time
Failed to Reject Null Hypothesis - Accept Ho
There is NO significant difference in the Mean values of clipped_actual_time and clipped_segmented_actual_time

Performing Non-parametric Test - MannWhitneyU for filtered_actual_time & filtered_segmented_actual_time
Failed to Reject Null Hypothesis - Accept Ho
There is NO significant difference in the Mean values of filtered_actual_time and filtered_segmented_actual_time

💡 Insights:

Even though data is not gaussian, though it has similar variance (confirmed by *Levene's* test)

It is confirmed that There is NO significant difference in the Mean values of Aggregated actual_time and segmented_actual_time through *MannWhitneyU* test.

- $H_0: \mu_{\text{Aggregated-actual-time}} = \mu_{\text{Segmented-actual-time}}$

⚡ OSRM distance aggregated value and segment OSRM distance aggregated value.

In [245...]
filtered_num_df.sample()

Out[245...]

	od_time_diff_hour	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time
3944	5.694592	341.0		88.352287	156.0	71.0	101.823898



In [246...]
clipped_num_df[['osrm_distance', 'segment_osrm_distance']].describe().T

Out[246...]

	count	mean	std	min	25%	50%	75%	max
osrm_distance	14787.0	144.551531	162.880435	9.0729	30.75690	65.302795	206.644203	543.285302
segment_osrm_distance	14787.0	150.959153	165.473080	9.0729	32.57885	69.784203	216.560608	543.285302

In [247...]
filtered_num_df[['osrm_distance', 'segment_osrm_distance']].describe().T

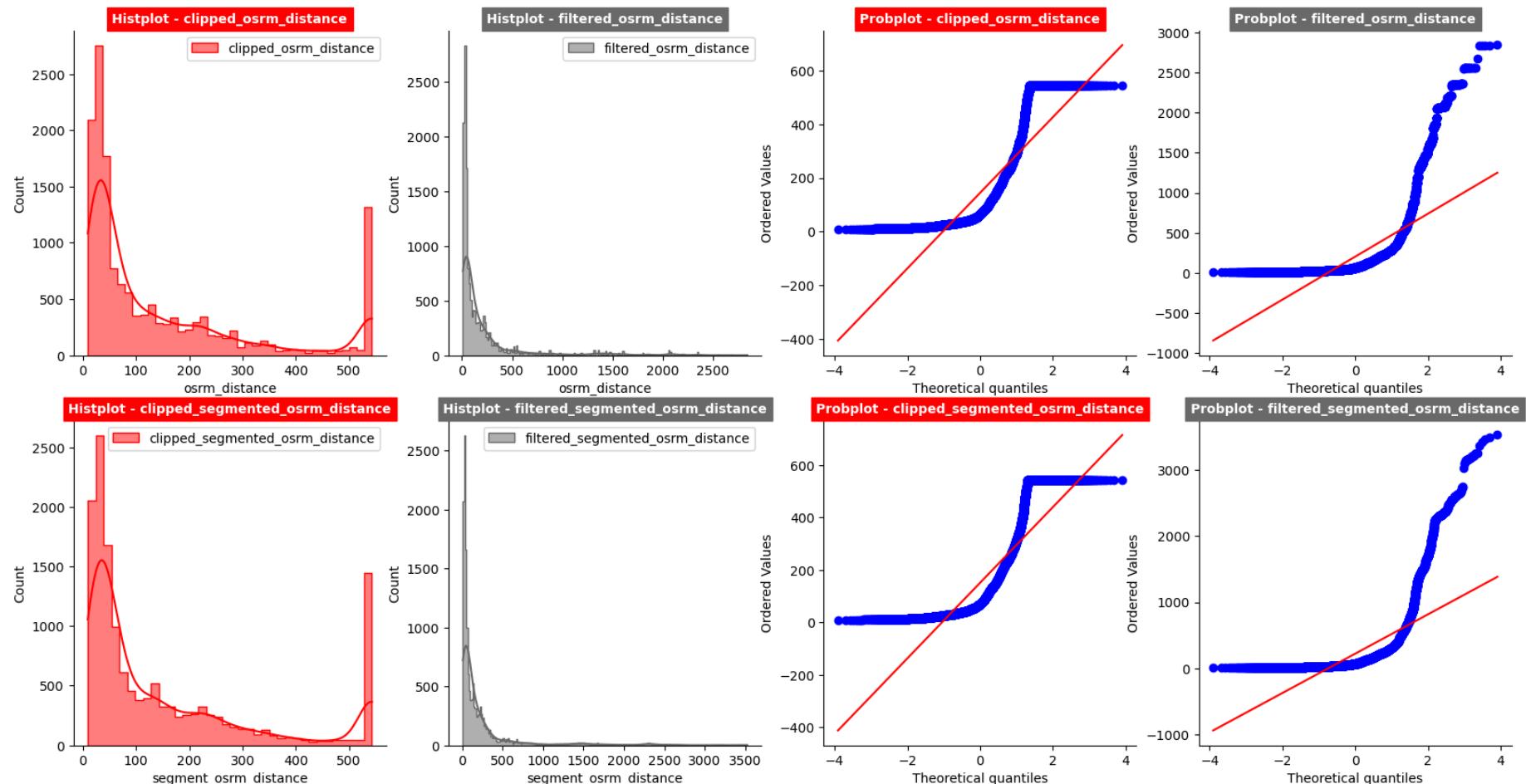
Out[247...]

	count	mean	std	min	25%	50%	75%	max
osrm_distance	14787.0	203.887405	370.565460	9.0729	30.75690	65.302795	206.644203	2840.081055
segment_osrm_distance	14787.0	222.705444	416.845642	9.0729	32.57885	69.784203	216.560608	3523.632324

In [248...]
clipped_osrm_distance = clipped_num_df['osrm_distance']
clipped_segmented_osrm_distance = clipped_num_df['segment_osrm_distance']
filtered_osrm_distance = filtered_num_df['osrm_distance']
filtered_segmented_osrm_distance = filtered_num_df['segment_osrm_distance']

In [249...]
normality_plots("clipped_osrm_distance", "clipped_segmented_osrm_distance", "filtered_osrm_distance", "filtered_segmented_osrm_distance")
clipped_osrm_distance, clipped_segmented_osrm_distance, filtered_osrm_distance, filtered_segmented_osrm_distance)

Normality check - Histplot & QQ(prob)plot



In [250...]

```

col_names= ["clipped_osrm_distance","clipped_segmented_osrm_distance","filtered_osrm_distance","filtered_segmented_osrm_distan
cols = [clipped_osrm_distance,clipped_segmented_osrm_distance,filtered_osrm_distance,filtered_segmented_osrm_distance]

for _ in zip(col_names,cols):
    normality = Normality_check(_[0],_[1])
    normality.shapiro_and_anderson()
    normality.boxcox_transformation()

```

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_osrm_distance column

Shapiro-Wilk Test

clipped_osrm_distance - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_osrm_distance - Data does not follow a normal distribution.

Performing BOXCOX transformation on clipped_osrm_distance column

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_osrm_distance column

Shapiro-Wilk Test

clipped_osrm_distance - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_osrm_distance - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_segmented_osrm_distance column

Shapiro-Wilk Test

clipped_segmented_osrm_distance - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_segmented_osrm_distance - Data does not follow a normal distribution.

Performing BOXCOX transformation on clipped_segmented_osrm_distance column

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_segmented_osrm_distance column

Shapiro-Wilk Test

clipped_segmented_osrm_distance - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_segmented_osrm_distance - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_osrm_distance column

Shapiro-Wilk Test

filtered_osrm_distance - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_osrm_distance - Data does not follow a normal distribution.

Performing BOXCOX transformation on filtered_osrm_distance column

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_osrm_distance column

Shapiro-Wilk Test

filtered_osrm_distance - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_osrm_distance - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_segmented_osrm_distance column

Shapiro-Wilk Test

filtered_segmented_osrm_distance - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_segmented_osrm_distance - Data does not follow a normal distribution.

Performing BOXCOX transformation on filtered_segmented_osrm_distance column

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_segmented_osrm_distance column

Shapiro-Wilk Test

filtered_segmented_osrm_distance - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_segmented_osrm_distance - Data does not follow a normal distribution.

In [251...]

```
levene_test("clipped_osrm_distance","clipped_segmented_osrm_distance",clipped_osrm_distance,clipped_segmented_osrm_distance),  
levene_test("filtered_osrm_distance","filtered_segmented_osrm_distance",filtered_osrm_distance,filtered_segmented_osrm_distance)
```

Performing Levene Test for clipped_osrm_distance & clipped_segmented_osrm_distance
Does not have Homogenous (different) Variance

Performing Levene Test for filtered_osrm_distance & filtered_segmented_osrm_distance
Does not have Homogenous (different) Variance

Out[251...]

In [252...]

```
### MannWhitney u Rank test

test_cols = [("clipped_osrm_distance", "clipped_segmented_osrm_distance", clipped_osrm_distance, clipped_segmented_osrm_distance),
             ("filtered_osrm_distance", "filtered_segmented_osrm_distance", filtered_osrm_distance, filtered_segmented_osrm_distance)]

for _ in test_cols:
    mannwhitneyu_test(_[0],_[1],_[2],_[3])
```

Performing Non-parametric Test - MannWhitneyU for clipped_osrm_distance & clipped_segmented_osrm_distance
Reject Null Hypothesis
There is a significant difference in the Mean values of clipped_osrm_distance and clipped_segmented_osrm_distance

Performing Non-parametric Test - MannWhitneyU for filtered_osrm_distance & filtered_segmented_osrm_distance
Reject Null Hypothesis
There is a significant difference in the Mean values of filtered_osrm_distance and filtered_segmented_osrm_distance

💡 Insights:

It is confirmed that There is a significant difference in the Mean values of osrm_distance and segmented_osrm_distance aggregated through *MannwhitneyU* test.

- $H_0: \mu_{\text{Aggregated-osrm-distance}} \neq \mu_{\text{Segmented-osrm-distance-aggregated}}$

⚡ OSRM time aggregated value and segment OSRM time aggregated value.

In [233...]: `filtered_num_df.sample()`

	<code>od_time_diff_hour</code>	<code>start_scan_to_end_scan</code>	<code>actual_distance_to_destination</code>	<code>actual_time</code>	<code>osrm_time</code>	<code>osrm_distance</code>	<code>segment_actual_time</code>
113	3.763914		225.0		45.959438	140.0	57.0

In [234...]: `clipped_num_df[['osrm_time', 'segment_osrm_time']].describe().T`

	<code>count</code>	<code>mean</code>	<code>std</code>	<code>min</code>	<code>25%</code>	<code>50%</code>	<code>75%</code>	<code>max</code>
<code>osrm_time</code>	14787.0	128.190912	150.301267	6.0	29.0	60.0	168.0	543.285302
<code>segment_osrm_time</code>	14787.0	136.843818	155.422255	6.0	30.0	65.0	184.0	543.285302

In [235...]: `filtered_num_df[['osrm_time', 'segment_osrm_time']].describe().T`

	<code>count</code>	<code>mean</code>	<code>std</code>	<code>min</code>	<code>25%</code>	<code>50%</code>	<code>75%</code>	<code>max</code>
<code>osrm_time</code>	14787.0	160.990936	271.459229	6.0	29.0	60.0	168.0	2032.0
<code>segment_osrm_time</code>	14787.0	180.511597	314.678741	6.0	30.0	65.0	184.0	2564.0

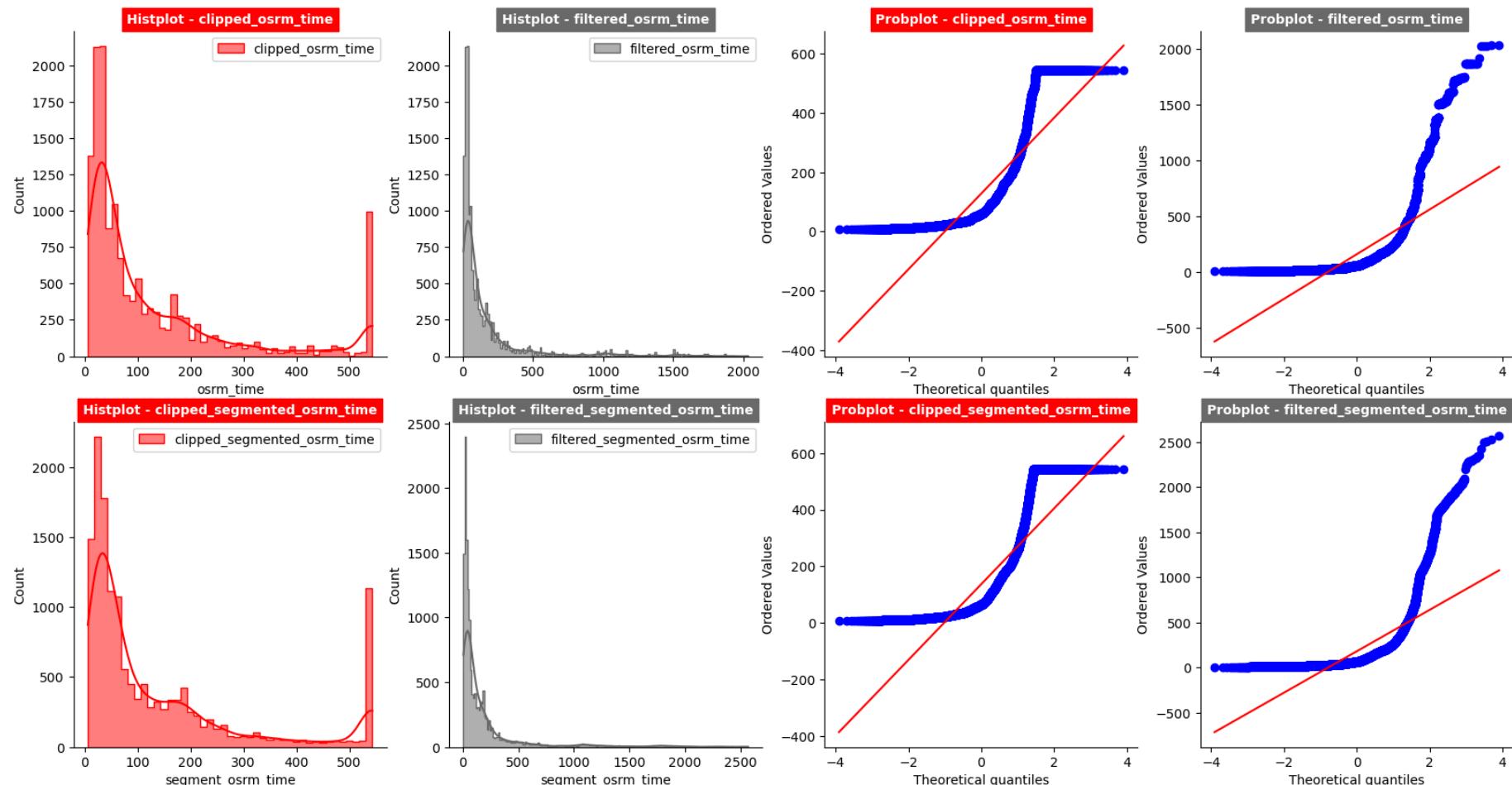
In [240...]

```
clipped_osrm_time = clipped_num_df['osrm_time']
clipped_segmented_osrm_time = clipped_num_df['segment_osrm_time']
filtered_osrm_time = filtered_num_df['osrm_time']
filtered_segmented_osrm_time = filtered_num_df['segment_osrm_time']
```

In [241...]

```
normality_plots("clipped_osrm_time","clipped_segmented_osrm_time","filtered_osrm_time","filtered_segmented_osrm_time",
                clipped_osrm_time,clipped_segmented_osrm_time,filtered_osrm_time,filtered_segmented_osrm_time)
```

Normality check - Histplot & QQ(prob)plot



```
In [242...]:  
col_names = ["clipped_osrm_time", "clipped_segmented_osrm_time", "filtered_osrm_time", "filtered_segmented_osrm_time"]  
cols = [clipped_osrm_time, clipped_segmented_osrm_time, filtered_osrm_time, filtered_segmented_osrm_time]  
  
for _ in zip(col_names, cols):  
    normality = Normality_check(_[0],_[1])  
    normality.shapiro_and_anderson()  
    normality.boxcox_transformation()
```

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_osrm_time column

Shapiro-Wilk Test

clipped_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_osrm_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on clipped_osrm_time column

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_osrm_time column

Shapiro-Wilk Test

clipped_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_osrm_time - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_segmented_osrm_time column

Shapiro-Wilk Test

clipped_segmented_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_segmented_osrm_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on clipped_segmented_osrm_time column

Performing SHAPIRO & ANDERSON-DARLING TEST for clipped_segmented_osrm_time column

Shapiro-Wilk Test

clipped_segmented_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test

clipped_segmented_osrm_time - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_osrm_time column

Shapiro-Wilk Test

filtered_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_osrm_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on filtered_osrm_time column
Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_osrm_time column

Shapiro-Wilk Test

filtered_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_osrm_time - Data does not follow a normal distribution.

Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_segmented_osrm_time column

Shapiro-Wilk Test

filtered_segmented_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_segmented_osrm_time - Data does not follow a normal distribution.

Performing BOXCOX transformation on filtered_segmented_osrm_time column
Performing SHAPIRO & ANDERSON-DARLING TEST for filtered_segmented_osrm_time column

Shapiro-Wilk Test

filtered_segmented_osrm_time - Data is not Gaussian

Since Shapiro-Wilk test is sensitive, we go with Anderson-Darling Test
filtered_segmented_osrm_time - Data does not follow a normal distribution.

In [243...]

```
levene_test("clipped_osrm_time","clipped_segmented_osrm_time",clipped_osrm_time,clipped_segmented_osrm_time),  
levene_test("filtered_osrm_time","filtered_segmented_osrm_time",filtered_osrm_time,filtered_segmented_osrm_time)
```

Performing Levene Test for clipped_osrm_time & clipped_segmented_osrm_time
 Does not have Homogenous (different) Variance

Performing Levene Test for filtered_osrm_time & filtered_segmented_osrm_time
 Does not have Homogenous (different) Variance

Out[243...]

In [244...]

```
### MannWhitney u Rank test

test_cols = [("clipped_osrm_time", "clipped_segmented_osrm_time", clipped_osrm_time, clipped_segmented_osrm_time),
             ("filtered_osrm_time", "filtered_segmented_osrm_time", filtered_osrm_time, filtered_segmented_osrm_time)]

for _ in test_cols:
    mannwhitneyu_test(_[0],_[1],_[2],_[3])
```

Performing Non-parametric Test - MannWhitneyU for clipped_osrm_time & clipped_segmented_osrm_time
 Reject Null Hypothesis
 There is a significant difference in the Mean values of clipped_osrm_time and clipped_segmented_osrm_time

Performing Non-parametric Test - MannWhitneyU for filtered_osrm_time & filtered_segmented_osrm_time
 Reject Null Hypothesis
 There is a significant difference in the Mean values of filtered_osrm_time and filtered_segmented_osrm_time

💡 Insights:

It is confirmed that There is a significant difference in the Mean values of Aggregated osrm_time and segmented_osrm_time aggregated through *MannwhitneyU* test.

- $H_0: \mu_{\text{Aggregated-osrm-time}} \neq \mu_{\text{Segmented-osrm-time-aggregated}}$

In [292... `trip_df.sample()`

	trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creation_month
8533	153774183943645102	trip-training	Carting	2018-09-24 00:46:07.807847	2018-09-24 07:32:00.833381	9.02261	2018-09-23 22:30:39.436721	9

In [295... `# To find the busiest corridor, we'll look at the most common combinations of source and destination states`
`corridor_counts = trip_df.groupby(['source_state', 'destination_state']).size().reset_index(name='count')`
`busiest_corridor = corridor_counts.sort_values(by='count', ascending=False).head(1)`

`# Average distance and time taken for the busiest corridor`
`busiest_corridor_details = busiest_corridor.merge(trip_df, on=['source_state', 'destination_state'])`
`average_distance = busiest_corridor_details['actual_distance_to_destination'].mean()`
`average_time = busiest_corridor_details['od_time_diff_hour'].mean()`

`print("Busiest corridor: ")`
`display(busiest_corridor)`
`print("Average distance: ", average_distance)`
`print("Average time (in hours): ", average_time)`

Busiest corridor:

	source_state	destination_state	count
85	Maharashtra	Maharashtra	2458

Average distance: 74.852844
Average time (in hours): 5.346577921457034

💡 🌱 Buisness Insights 🌱 💡

Based on EDA:

- The Timeframe of the data is '2018-09-12' to '2018-10-08' i.e(26 days).

- 88% of the trips are from October Month & remaining are from November
- The entire data is heavily right skewed
- Almost all the features are heavily positively correlated with each other & which is intuitive as well.
- Start & End dates of the months have less percent of trips compared to mid of the month. Though the difference is not huge
- That's very strange to see that there is absolutely no trip from 4th- 11th day of the month
- Most orders come mid-month. That means customers usually make more orders in the middle of the month.

Route type:

- The analysis reveals that a higher proportion of shipments are routed through Full Truck Load (FTL) as opposed to carting. This has important implications for the efficiency and speed of the delivery process.

Geographical Focus:

Understanding the busiest routes and distances can help in optimizing logistics operations, improving transportation efficiency, and potentially reducing costs.

- **State:** The states of Haryana, Maharashtra, and Karnataka are not only busy source states but also emerge as the busiest source states, indicating a high demand or significant business activities originating from these regions.
- **source city:** Gurgaon, Bangalore, and Bhiwandi are identified as the busiest source cities, suggesting that these cities play a crucial role in contributing to the overall business operations or transportation activities.
- **Destination city:** Gurgaon, Bangalore, and Hyderabad are identified as the busiest destination cities, underscoring their significance in terms of business activities or population movement.
- **Busiest corridor:** Overall, the busiest corridor is Mumbai_Maharashtra and Bangalore_Karnataka which has the maximum trips.
 - Average distance: 74.852844 kms
 - Average time (in hours): 5.346577921457034

Delivery Time & Distance Accuracy:

OSRM Time vs. Actual Time:

- The difference between the mean values of estimated delivery time and actual delivery time suggests that there may be variations or delays in the actual delivery process compared to the initial estimates.
- The fact that the mean of OSRM time is less than the mean of actual delivery time indicates that the estimated times provided by the OSRM (Open Source Routing Machine) service tend to be optimistic.

OSRM Distance vs. Actual Distance:

- The mean of OSRM distance being greater than the mean of actual distance to the destination suggests that the OSRM might overestimate the distances. This could impact route planning and fuel efficiency calculations.

Segment-wise time Analysis:

- The equality in the mean values of actual time and segment actual time suggests that the time measurements are consistent across different segments of the delivery process

Segment-wise distance Analysis:

- The mean of segment OSRM distance being greater than the mean of OSRM distance implies that the OSRM might provide more conservative estimates for distance within individual segments.

Further look into :

- As its depicted from the analysis that there is absolutely no trip from 4th- 11th day of the month, The reason for that can be figured out and catered to receive the orders in the these dates as well.
- More ways to promote FTL route handling system can be implemented to increase this percentage

Business Recommendations

Route Optimization:

- Given that the busiest state route is within Karnataka, it might be beneficial to optimize the transportation network within Karnataka to improve efficiency and reduce congestion. Consider implementing route optimization algorithms and real-time traffic monitoring to

enhance the transportation system.

- Since Gurgaon and Bangalore are identified as the busiest source and destination cities, respectively, focus on city-specific strategies to manage the high traffic volume.

Operational Efficiency:

- Since mean of OSRM time is less than the mean of actual delivery time, Businesses could use this insight to set more realistic delivery time expectations for customers.
- Since the mean of OSRM distance greater than the mean of actual distance, Businesses should consider adjusting their distance estimations for more accurate logistics planning.
- Since the mean of segment OSRM distance greater than the mean of OSRM distance, along with this, we have the actual distance travelled, Businesses can use this information to fine-tune their route planning and optimize segment-specific logistics.
- Implement advanced demand forecasting techniques to anticipate peak travel times and adjust transportation services accordingly. This proactive approach can help in better resource allocation and minimize the impact of congestion during peak hours.
- Overall, the analysis hints at potential areas for operational improvement. Businesses could focus on refining their route planning algorithms, addressing discrepancies in estimated times and distances, and streamlining processes between different stages of delivery to enhance overall operational efficiency.

Customer Satisfaction:

- Improving accuracy in estimated delivery times and distances can contribute to increased customer satisfaction.
- FTL shipments: Faster delivery times, facilitated by a higher proportion of FTL shipments, can directly impact customer satisfaction. Customers typically value timely deliveries, and this strategic choice aligns with meeting or exceeding customer expectations in terms of shipment speed.

Customer profiling:

- Customer profiling of the customers belonging to the states Maharashtra, Karnataka, Haryana, Tamil Nadu and Uttar Pradesh has to be done to get to know why major orders are coming from these states and to improve customers' buying and delivery experience.

Cost Optimization:

- Understanding the differences in estimated and actual times and distances can aid in cost optimization efforts.

- Fine-tuning logistics planning based on more accurate measurements can lead to better resource allocation and potentially reduce operational costs.

Strategic Decision-making:

- The preference for FTL over carting reflects a strategic decision by the logistics management.
- Understanding the reasons behind this choice and continuously evaluating its impact can guide future decision-making processes and help adapt to evolving business needs.

Collaboration with Stakeholders:

- Collaborate with relevant stakeholders, including government authorities, transportation companies, and local communities, to develop and implement comprehensive strategies for managing and optimizing transportation in the identified busy corridors and cities.
-