🚚 DELHIVERY - Business CaseStudy 🚚

Feature Engineering

Analysed by : **Aakash**

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.

Byconducting 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

Feature	Description
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
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
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 mino
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

```
# 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')
```

```
# pd_reading the data
delhivery_data = pd.read_csv('delhivery_data.csv')
```

```
# setting the option of displaying all the columns
pd.set_option('display.max_columns', 50)
```

```
# making a deep copy for backup
dd = delhivery_data.copy()
dd.head()
```

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

🧐 Exploration of data :

```
1 dd.shape
```

```
(144867, 24)
```

```
1 dd.columns
```

```
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')
```

```
1 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

```
1 dd.describe().T
```

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

```
1 dd.describe(include=object).T
```

	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

```
1 dd[dd.duplicated()]
```

data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destinati

🏷️ Insights

- The dataset does not contain any duplicates.

🔸❓Null Detection

```
1 dd.isna().any()
```

```
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
```

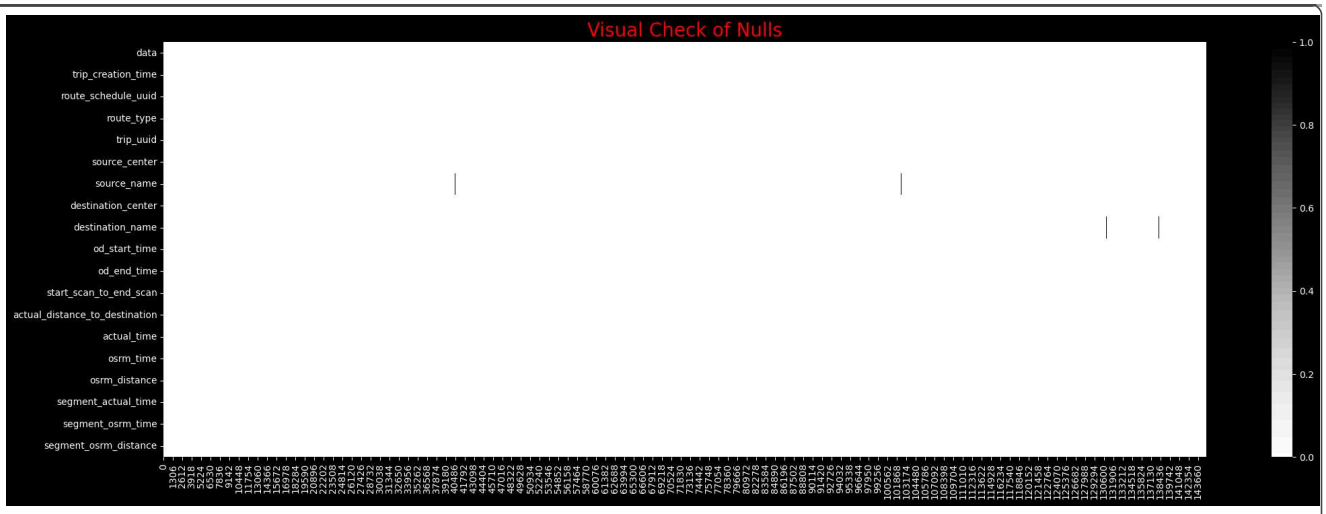
```
1 dd.isnull().sum()
```

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

```
1 def missing_data(df):
2     total_missing_df = df.isnull().sum().sort_values(ascending =False)
3     percent_missing_df = (df.isnull().sum()/df.isna().count()*100).sort_values(ascending=False) # ----> /len(dd)
4     missing_data_df = pd.concat([total_missing_df, percent_missing_df], axis=1, keys=['Total', 'Percent'])
5     return missing_data_df
6
7 missing_pct = missing_data(dd)
8 missing_pct[missing_pct['Total']>0]
```

	Total	Percent
source_name	293	0.202254
destination_name	261	0.180165

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




```
# Dropping unknown fields

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

```
dd.sample()
```

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

```
dd.shape
```

```
(144867, 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)
```

```
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

```
1 dd.sample()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	dest
124444	training	2018-09-23 02:44:13.665024	thanos::sroute:f01c8bbd-655d-42ea-9abf-60d5040...	FTL	trip-153767065366477819	IND821115AAB	Sasaram_Central_I_2 (Bihar)	

```
1 dd.dtypes
```

```
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
```

```
1 # Converting the datatypes to category for columns like data and route_type as they only have 2 values.
2 dd['data'] = dd['data'].astype('category')
3 dd['route_type'] = dd['route_type'].astype('category')
4
5 # Converting time columns to datetime format
6 datetime_cols = ['trip_creation_time', 'od_start_time', 'od_end_time']
7 for _ in datetime_cols:
8     dd[_] = pd.to_datetime(dd[_])
```

```
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

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

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

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

```
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.

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

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

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

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

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

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

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

```
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`....

```
dd[(dd.source_name.isna())&(dd.destination_name.isna())]
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination
68006	training	2018-09-26 22:21:56.619259	thanos::sroute:cfb575b8-df26-48f5-8427-6f48f9d...	FTL	trip-153800051661903546	IND331022A1B	NaN	IND331
68007	training	2018-09-26 22:21:56.619259	thanos::sroute:cfb575b8-df26-48f5-8427-6f48f9d...	FTL	trip-153800051661903546	IND331022A1B	NaN	IND331
68008	training	2018-09-26 22:21:56.619259	thanos::sroute:cfb575b8-df26-48f5-8427-6f48f9d...	FTL	trip-153800051661903546	IND331022A1B	NaN	IND331

```
dd[dd.source_name.isna()]
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destinatio
112	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND3
113	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND3
114	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND3
115	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND3
116	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND3
...	...	...	...	...	...	...	...	
144484	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND4
144485	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND4
144486	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND4
144487	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND4
144488	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND4

293 rows × 19 columns

```
1 dd[dd.destination_name.isna()]
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	d
110	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342601AAA	Piparcity_BsstdDPP_D (Rajasthan)	
111	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342601AAA	Piparcity_BsstdDPP_D (Rajasthan)	
982	test	2018-10-01 20:56:18.155260	thanos::sroute:d0ebdacd-e09b-47d3-be77-c9c4a05...	FTL	trip-153842737815495661	IND573103AAA	Arsikere_HsnRdDPP_D (Karnataka)	
983	test	2018-10-01 20:56:18.155260	thanos::sroute:d0ebdacd-e09b-47d3-be77-c9c4a05...	FTL	trip-153842737815495661	IND573103AAA	Arsikere_HsnRdDPP_D (Karnataka)	
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)	
...	...	...	...	...	...	...	...	
144478	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	
144479	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	
144480	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	
144481	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	
144482	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	

261 rows × 19 columns

```
ddd = dd.copy()
```

```
missing_source_name = ddd.loc[ddd['source_name'].isnull(), 'source_center'].unique()
missing_source_name
```

```
array(['IND342902A1B', 'IND577116AAA', 'IND282002AAD', 'IND465333A1B',
       'IND841301AAC', 'IND509103AAC', 'IND126116AAA', 'IND331022A1B',
       'IND505326AAB', 'IND852118A1B'], dtype=object)
```

```
missing_destination_name = ddd.loc[ddd['destination_name'].isnull(), 'destination_center'].unique()
missing_destination_name
```

```
array(['IND342902A1B', 'IND577116AAA', 'IND282002AAD', 'IND465333A1B',
       'IND841301AAC', 'IND505326AAB', 'IND852118A1B', 'IND126116AAA',
       'IND509103AAC', 'IND221005A1A', 'IND250002AAC', 'IND331001A1C',
       'IND122015AAC'], dtype=object)
```

```
# 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()
```

```
False
```

```
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
```

```
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
```

```
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
```

```
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
```

```
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, c
```

```
ddd.source_name.value_counts()
```

```
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
```

```
1 ddd.destination_name.value_counts()
```

```
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..

```
1 261+290
```

```
551
```

```
1 len(delhivery_data)
```

```
144867
```

```
1 144867 - 551
```

```
144316
```

```
1 df = dd.dropna()
```

```
1 df.isna().sum().any()
```

```
False
```

```
1 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.

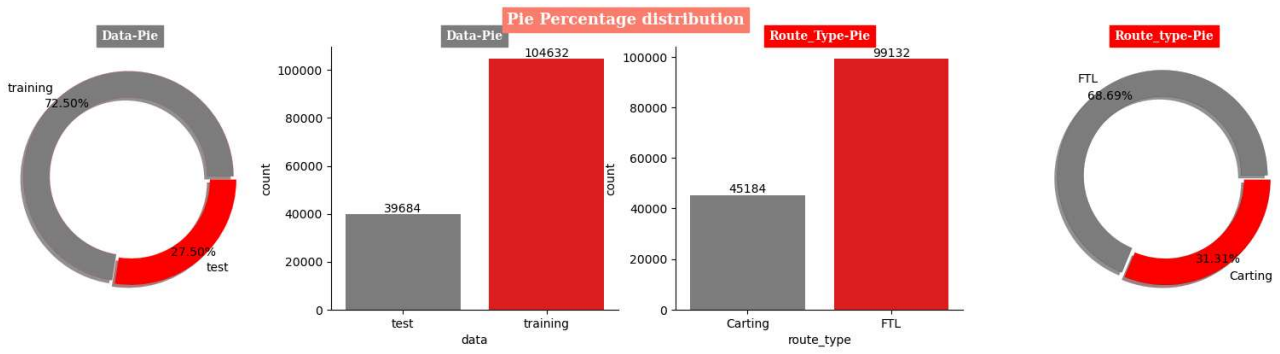
Exploratory Data Analysis

```
cp = ['gray','red','dimgrey','tomato','dimgray','orangered','k','salmon','gray','red','dimgrey','tomato','dimgray','oranger
```

```
df.sample()
```

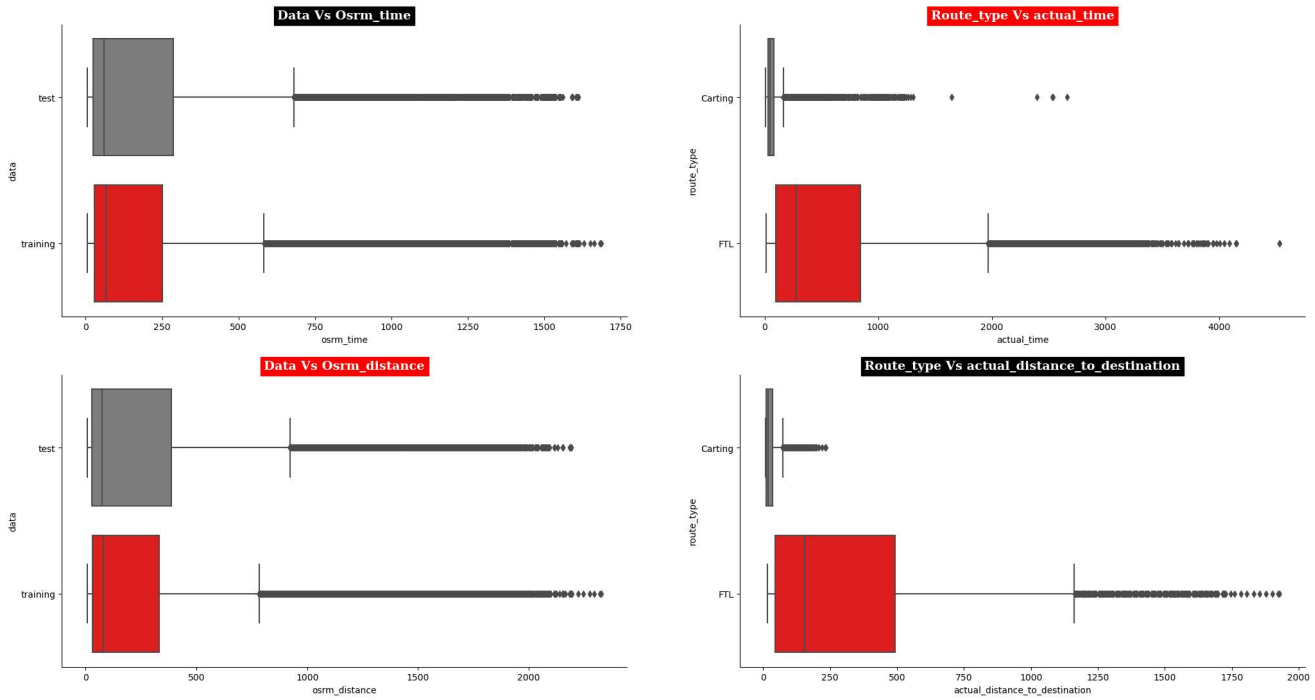
	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destina
92389	training	2018-09-14 23:36:50.771430	thanos::sroute:3f001d56-e933-4ba1-a7cf-26828f7...	Carting	trip-153696821077115152	IND515201AAA	Hindupur_Parigi_D (Andhra Pradesh)	IN

```
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 , explod
       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()
```




```
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')
```

```
12
13 plt.subplot(223)
14 sns.boxplot(data=df,y='data',x='osrm_distance',palette=cp)
15 plt.title('Data Vs Osrm_distance',fontsize=14,fontfamily='serif',fontweight='bold',backgroundcolor='r',color='w')
16
17 plt.subplot(224)
18 sns.boxplot(data=df,y='route_type',x='actual_distance_to_destination',palette=cp)
19 plt.title('Route_type Vs actual_distance_to_destination',fontsize=14,fontfamily='serif',fontweight='bold',backgroundcolor=
20 sns.despine()
```



🏷️ 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.

```
1 # Grouping by segment
2 # Creating a unique identifier for each segment of a trip
3
4 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()
```

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

144316 rows × 7 columns

```
# 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
```

	segment_key	trip_uuid	data	route_type	trip_creation_time	so
0	trip-153671041653548748+IND209304AAA+IND000000ACB	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Kanpur_C (Utta
1	trip-153671041653548748+IND462022AAA+IND209304AAA	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Bhopal_ (Madhy
2	trip-153671042288605164+IND561203AAB+IND562101AAA	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_Ch (
3	trip-153671042288605164+IND572101AAA+IND561203AAB	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Tumkur_ (
4	trip-153671043369099517+IND000000ACB+IND160002AAC	trip-153671043369099517	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bi
...	...	...	...	...	...	
26217	trip-153861115439069069+IND628204AAA+IND627657AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Tirchchndr_Shr (T
26218	trip-153861115439069069+IND628613AAA+IND627005AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Peikulam_Sri (T
26219	trip-153861115439069069+IND628801AAA+IND628204AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Eral_Busstan
26220	trip-153861118270144424+IND583119AAA+IND583101AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Sandur_Wrc (
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (

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

```
# 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
```

	segment_key	trip_uuid	data	route_type	trip_creation_time	so
0	trip-153671041653548748+IND209304AAA+IND000000ACB	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Kanpur_C (Utta
1	trip-153671041653548748+IND462022AAA+IND209304AAA	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Bhopal_ (Madhy
2	trip-153671042288605164+IND561203AAB+IND562101AAA	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_Ch (
3	trip-153671042288605164+IND572101AAA+IND561203AAB	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Tumkur_ (
4	trip-153671043369099517+IND000000ACB+IND160002AAC	trip-153671043369099517	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bi
...	...	...	...	...	...	
26217	trip-153861115439069069+IND628204AAA+IND627657AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Tirchchndr_Shr (T
26218	trip-153861115439069069+IND628613AAA+IND627005AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Peikulam_Sri (T
26219	trip-153861115439069069+IND628801AAA+IND628204AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Eral_Busstan
26220	trip-153861118270144424+IND583119AAA+IND583101AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Sandur_Wrc (
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (

26222 rows × 22 columns

```
segment_agg_data.sample()
```

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

```
# de = segment_agg_data.drop(columns=['source_city','source_state','source_place','destination_place','destination_city','d
de = segment_agg_data.copy()
```

```
de.sample()
```

	segment_key	trip_uuid	data	route_type	trip_creation_time	source_
4962	trip-153704666399260818+IND421302AAG+IND401104AAA	trip-153704666399260818	training	Carting	2018-09-15 21:24:23.992906	Bhiwandi_Manko (Mahara

```
# # 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]
```

```
# using regex pattern to seperate 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:
```

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

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

```
de
```

	segment_key	trip_uuid	data	route_type	trip_creation_time	sou
0	trip-153671041653548748+IND209304AAA+IND000000ACB	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Kanpur_C (Utta
1	trip-153671041653548748+IND462022AAA+IND209304AAA	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Bhopal_ (Madhya
2	trip-153671042288605164+IND561203AAB+IND562101AAA	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_Ch (
3	trip-153671042288605164+IND572101AAA+IND561203AAB	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Tumkur_ (
4	trip-153671043369099517+IND000000ACB+IND160002AAC	trip-153671043369099517	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bi
...	...	...	...	...	...	
26217	trip-153861115439069069+IND628204AAA+IND627657AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Tirchchndr_Shr (T
26218	trip-153861115439069069+IND628613AAA+IND627005AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Peikulam_Sri (T
26219	trip-153861115439069069+IND628801AAA+IND628204AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Eral_Busstan
26220	trip-153861118270144424+IND583119AAA+IND583101AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Sandur_Wrc (
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (

26222 rows × 28 columns

```
de[(de['source_place']=='') | (de['destination_place']=='')]
```

	segment_key	trip_uuid	data	route_type	trip_creation_time	sourc
7	trip-153671052974046625+IND583101AAA+IND583201AAA	trip-153671052974046625	training	FTL	2018-09-12 00:02:09.740725	Bellary_Dc (Kar
9	trip-153671052974046625+IND583201AAA+IND583119AAA	trip-153671052974046625	training	FTL	2018-09-12 00:02:09.740725	Hospet (Kar
19	trip-153671110078355292+IND121004AAB+IND121001AAA	trip-153671110078355292	training	Carting	2018-09-12 00:11:40.783923	FBD_Balabhgar (H
33	trip-153671173668736946+IND110043AAA+IND110078AAA	trip-153671173668736946	training	Carting	2018-09-12 00:22:16.687619	Delhi_Na
80	trip-153671320807895983+IND121004AAB+IND121102AAA	trip-153671320807895983	training	Carting	2018-09-12 00:46:48.079257	FBD_Balabhgar (H
...	...	...	...	...	...	
26118	trip-153860849934816308+IND110078AAA+IND110043AAA	trip-153860849934816308	test	Carting	2018-10-03 23:14:59.348414	Janakpur
26153	trip-153860958923357924+IND842003AAB+IND482002AAA	trip-153860958923357924	test	Carting	2018-10-03 23:33:09.233829	Jabalpur_Adh (Madhya P
26180	trip-153861007249500192+IND842001AAA+IND846004AAA	trip-153861007249500192	test	FTL	2018-10-03 23:41:12.495257	Muzaffrpur_B
26181	trip-153861007249500192+IND846004AAA+IND847103AAA	trip-153861007249500192	test	FTL	2018-10-03 23:41:12.495257	Darbhanga
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (Kar

782 rows × 28 columns

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

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

segment_key	trip_uuid	data	route_type	trip_creation_time	source_name	destination_name	od_start_time	od_end_time	start_

```
de.isna().sum()
```

```
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
```

```
#de = de.drop(columns=['source_city','source_state','source_place','destination_city', 'destination_place', 'destination_st
```

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

```
np.set_printoptions(threshold=np.inf)
```

```
de['source_city'].unique()
```

```
       'Pataudi', 'Nahariagun', 'Bhadohi', 'Cherchaia', 'Madhapalle',
       'Mokokchung', 'Pappadahandi', 'Lalitpur', 'Farrukhbad', 'Shahabad',
       'Jalalabad', 'Tilhar', 'Garhshnker', 'Hajipur', 'Paranpur',
       'Gauribidanur', 'Wankaner', 'Morbi', 'BariSadri', 'Neemuch',
       'Sangareddy', 'Islampure', 'Nakhatrana', 'Chandauli', 'Dhrmsthala',
       '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', 'Brajrajnagar', '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', 'Kendrpara', '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)
```

```
1 de['source_place'].unique()
```

```
array(['Central_H_6', 'Trnsport_H', 'ChikaDPP_D', 'Veersagr_I',
       'Bilaspur_HB', 'Nelmngla_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', 'Balabhgarh_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', 'Sriramp t_D', 'GlbrgaRD_D',
       'DBRCmplx_D', 'Mainroad_D', 'OnkarDPP_D', 'KaranNGR_D',
       'Panchot_IP', 'Sohagpur_D', 'ChainDPP_D', 'Chrompet_L',
       'Busstand_D', 'JirgeNgr_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', 'Swamylyt_D',
       'Narasipr_D', 'BrDbleRd_D', 'SulthnRd_D', 'NarsipuraRd_D',
       'Yadvgiri_IP', 'RatnaDPP_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', 'Thalthru_DC', 'Bypasrd_D',
       'Sttyapar_D', 'Poonamallee_HB', 'VUNagar_DC', 'MotvdDPP_D',
       'NlgaonRd_D', 'Srnwsngr_D', 'Dakor_DC', 'Vaghasi_IP', 'Bnnrghta_L',
       'Thirumtr_IP', 'SelamRd_D', 'EastmnRD_D', 'VkotRoad_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', 'Govndsgr_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', 'Knrpati_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',
```

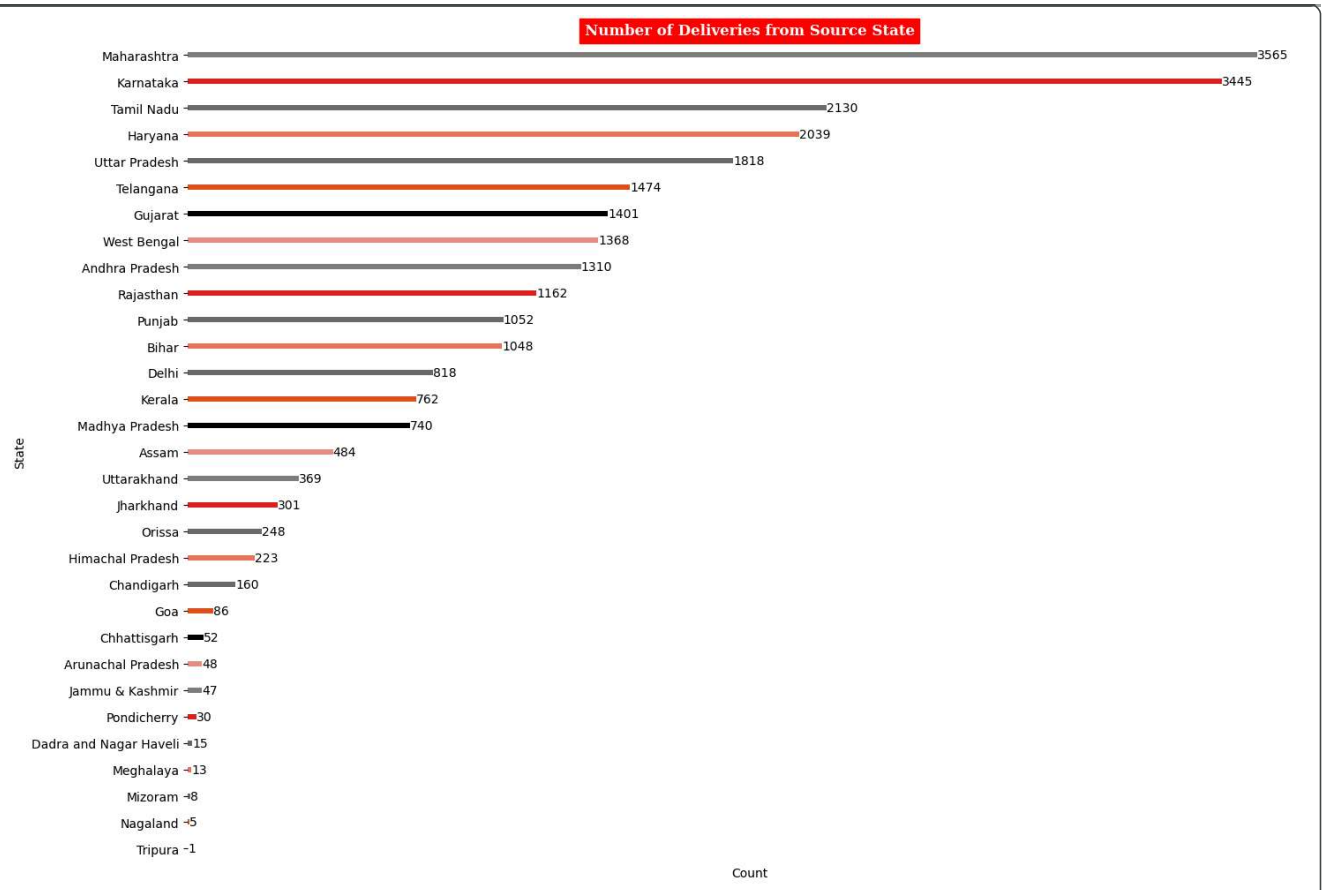
```
1 de['source_state'].unique()
```

```
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)
```

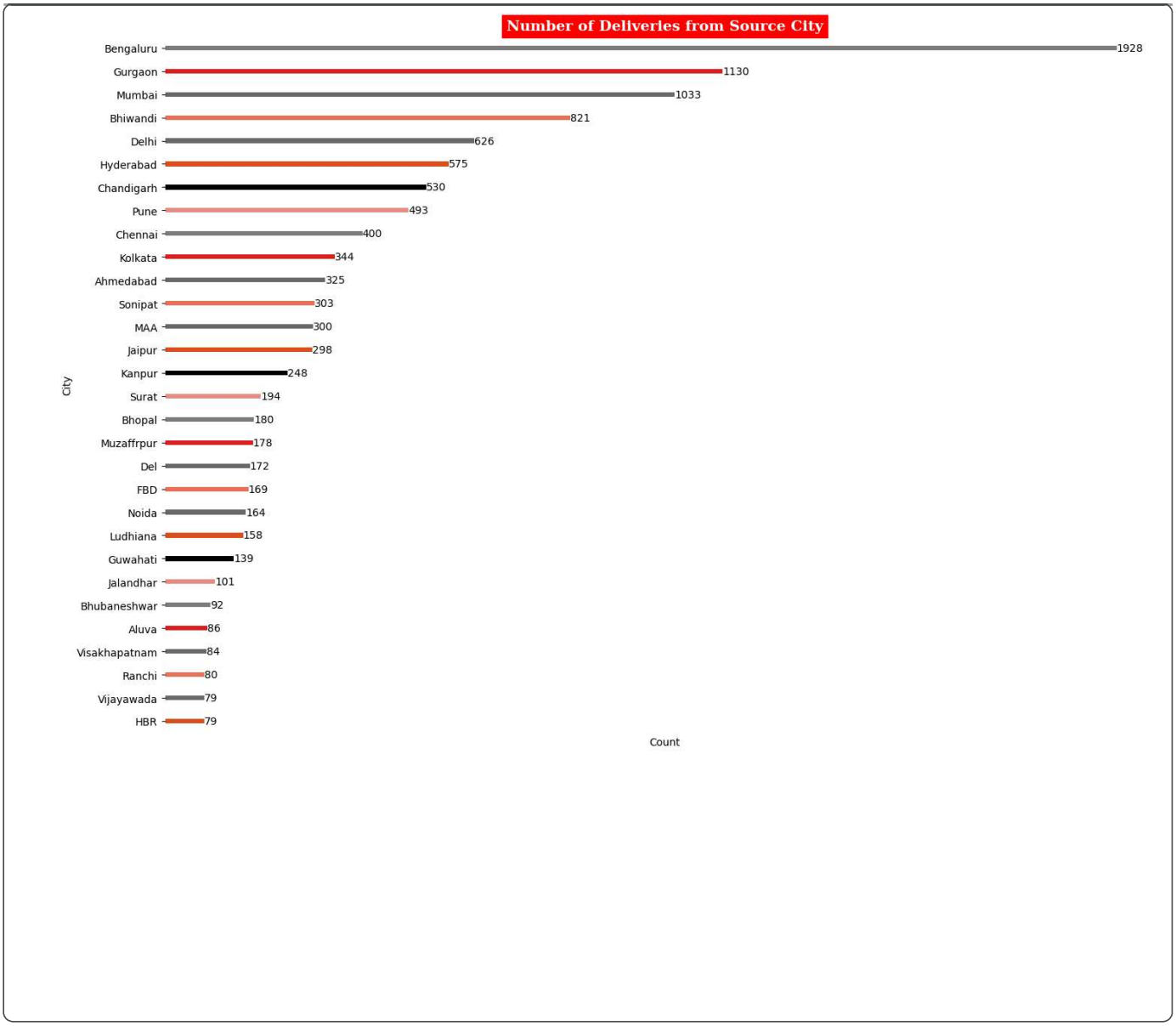
```
1 de['source_state'].value_counts().to_frame().style.background_gradient(cmap='Reds')
```

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
Goa	86
Chhattisgarh	52
Arunachal Pradesh	48
Jammu & Kashmir	47
Pondicherry	30
Dadra and Nagar Haveli	15
Meghalaya	13
Mizoram	8
Nagaland	5
Tripura	1

```
 1 state_counts = de['source_state'].value_counts().to_frame().reset_index()
 2 state_counts.columns = ['State', 'Count']
 3
 4 plt.figure(figsize=(15,10))
 5 a = sns.barplot(y='State', x='Count', data=state_counts,palette=cp,width=0.2)
 6 a.bar_label(a.containers[0], label_type='edge')
 7 plt.xticks([])
 8 plt.ylabel('State')
 9 plt.xlabel('Count')
10 plt.title('Number of Deliveries from Source State',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='r',col
11 plt.tight_layout()
12 sns.despine(bottom=True,left=True)
13 plt.show()
```




```
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',colc
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.

```
1 de.describe().T
```

	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
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
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
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
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
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
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
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
od_time_diff_hour	26222.0	4.985121	0.345047	1.516296	2.538946	5.121433

```
1 de.describe(include='object').T
```

	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

```
1 de['destination_city'].unique()
```

```
'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', 'Haveri', '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',
```

```
1 de['destination_place'].unique()
```

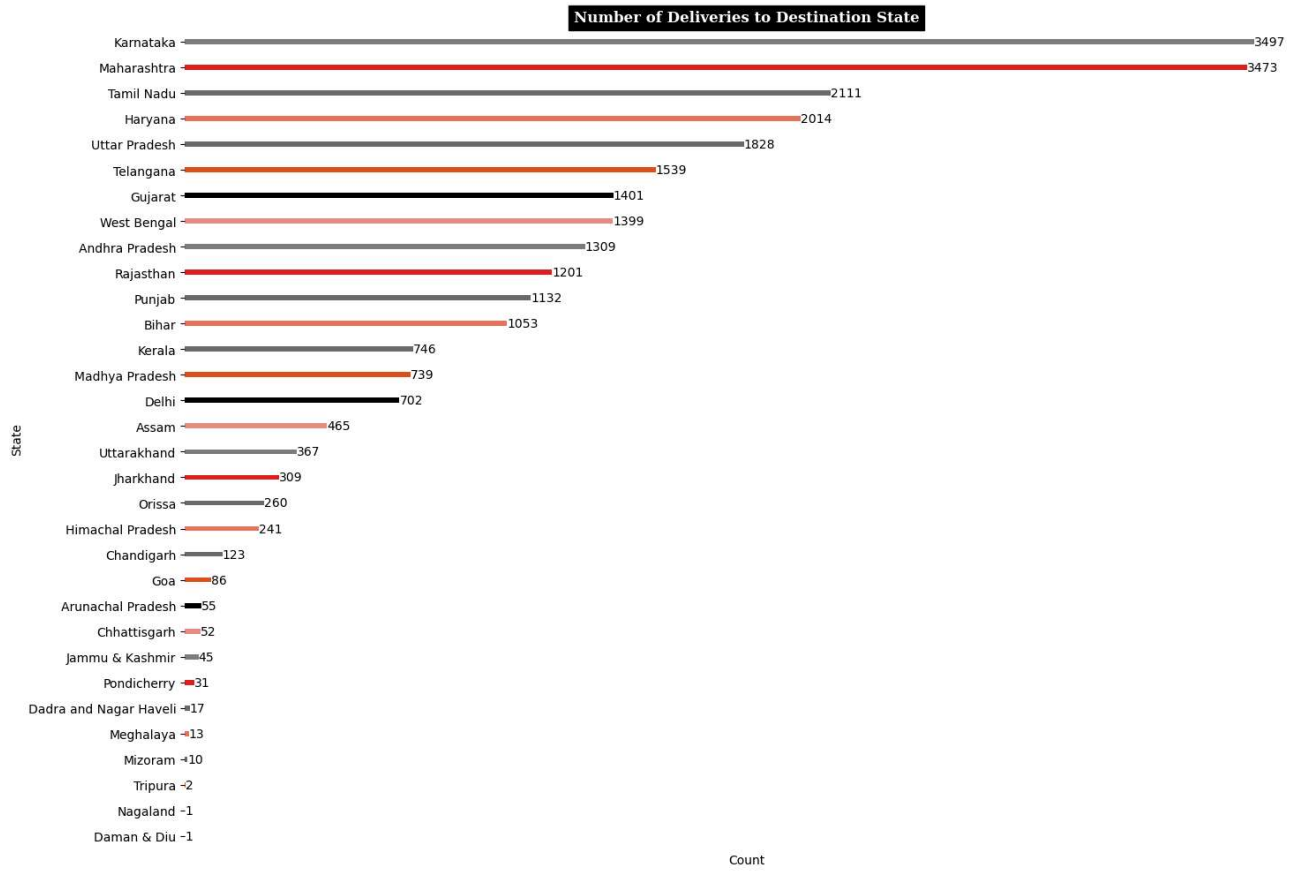
```
array(['Bilaspur_HB', 'Central_H_6', 'ShntiSgr_D', 'ChikaDPP_D',
       'Mehmdpur_H', 'MiraRd_IP', 'Hospet', 'Dc', 'WrdN1DPP_D',
       'Sriperumbudur_Dc', 'Poonamallee', 'Vandalur_Dc', 'NwYlhnka_DC',
       'Layout PC', 'Central_I_4', 'Central_D_3', 'Bhogal',
       'Rahatani DPC', 'Faridabad', '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', 'NkshtrPz_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', 'RTOofice_D', 'lalaNGR_D',
       'Athithnr_DC', 'RjndraRd_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', 'Balabhgarh_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',
       'JmnvadRd_DC', 'AdrshSt_DC', 'Nagaram_D', 'Krishna_D',
       'Rynapadu_H', 'HydRoad_DC', 'Ragvendr_D', 'NSTRoad_I',
       'BsstdDPP_D', 'HawaiPlr_DC', 'Panchot_IP', 'Bargawan_DC',
       'KGAirprt_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', 'NrdawDPP_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', 'Srnwsngr_D', 'Aswningr_I', 'Sec 02_DPC', 'Dakor_DC',
       'GandhiRd_D', 'EastmnRD_D', 'VkotRoad_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', 'Govndsgr_D', 'BljiMrkt_D', 'Bnnrghta_L',
       'Beliaghata_DPC', 'Airport_H', 'Lake Avenue_DPC', 'East',
       'Memnagar', 'Mumbra_DC', 'Satellite', 'Pakrela_D', 'Auliyapr_D',
       'Ulhasngr_DC', 'Pawane_L', 'Kalyan West _Dc', 'Bbganj_I',
```

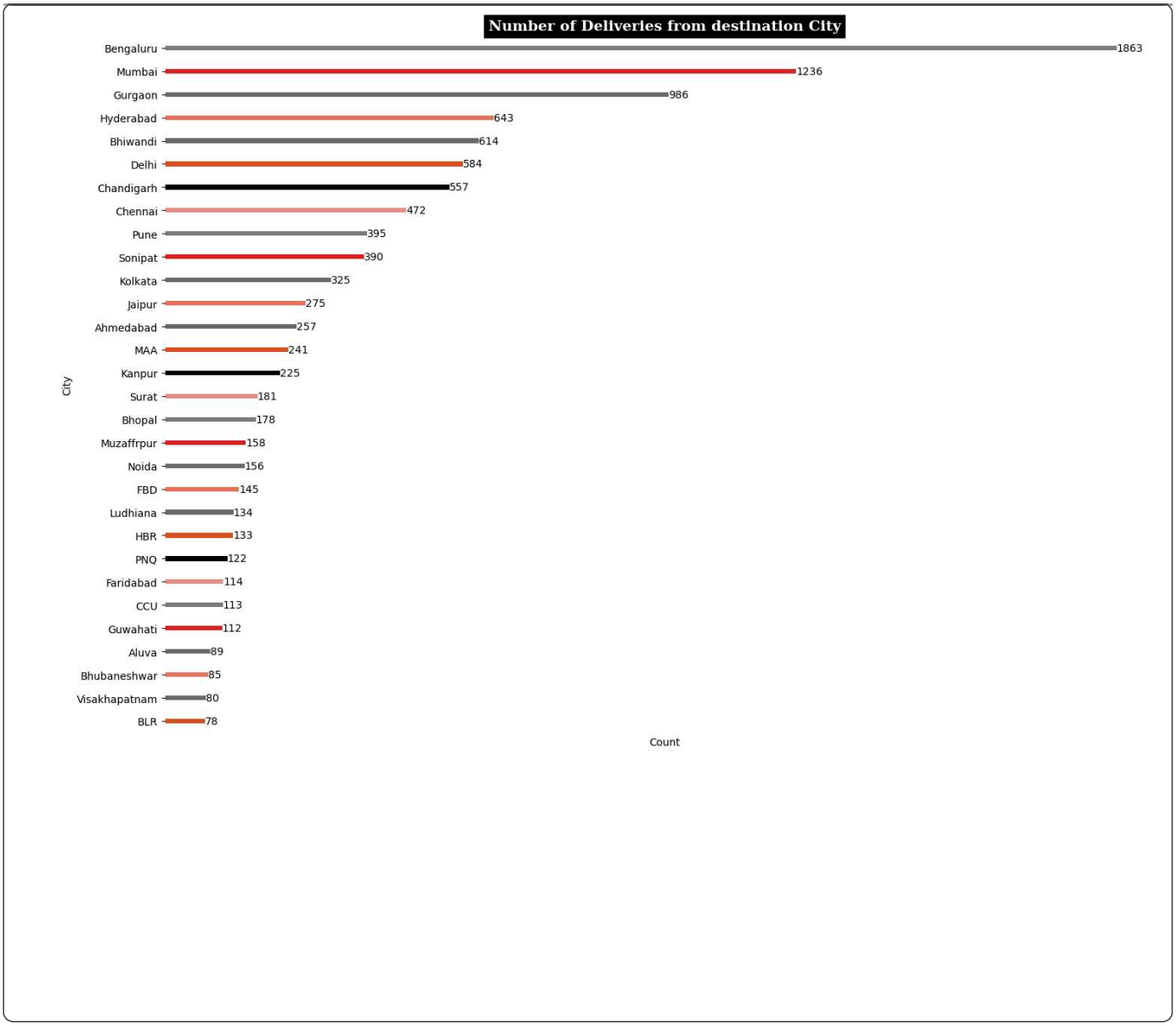
```
1 de['destination_state'].unique()
```

```
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)
```

```
1 state_counts = de['destination_state'].value_counts().to_frame().reset_index()
2 state_counts.columns = ['State', 'Count']
3
4 plt.figure(figsize=(15,10))
5 a = sns.barplot(y='State', x='Count', data=state_counts,palette=cp,width=0.2)
6 a.bar_label(a.containers[0], label_type='edge')
7 plt.xticks([])
8 plt.ylabel('State')
9 plt.xlabel('Count')
10 plt.title('Number of Deliveries to Destination State',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='k',
11 plt.tight_layout()
12 sns.despine(bottom=True,left=True)
13 plt.show()
```




```
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'
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.

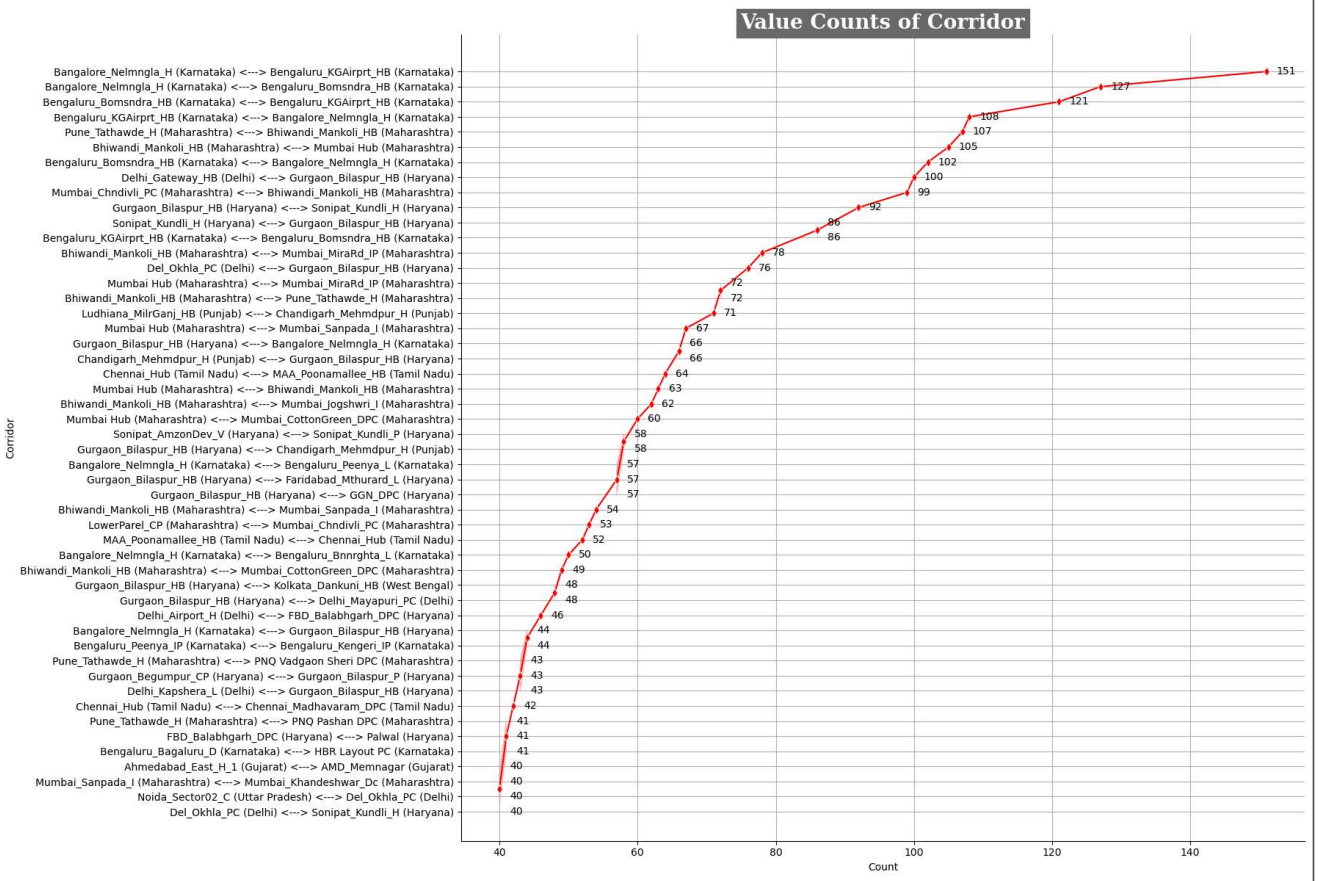
```
1 np.set_printoptions(threshold=np.inf)
```

```
1 de['corridor'] = de['source_name'] +' <---> '+ de['destination_name']
2 de['corridor'].value_counts()
```

```
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
```

```
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='dimgrey',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:

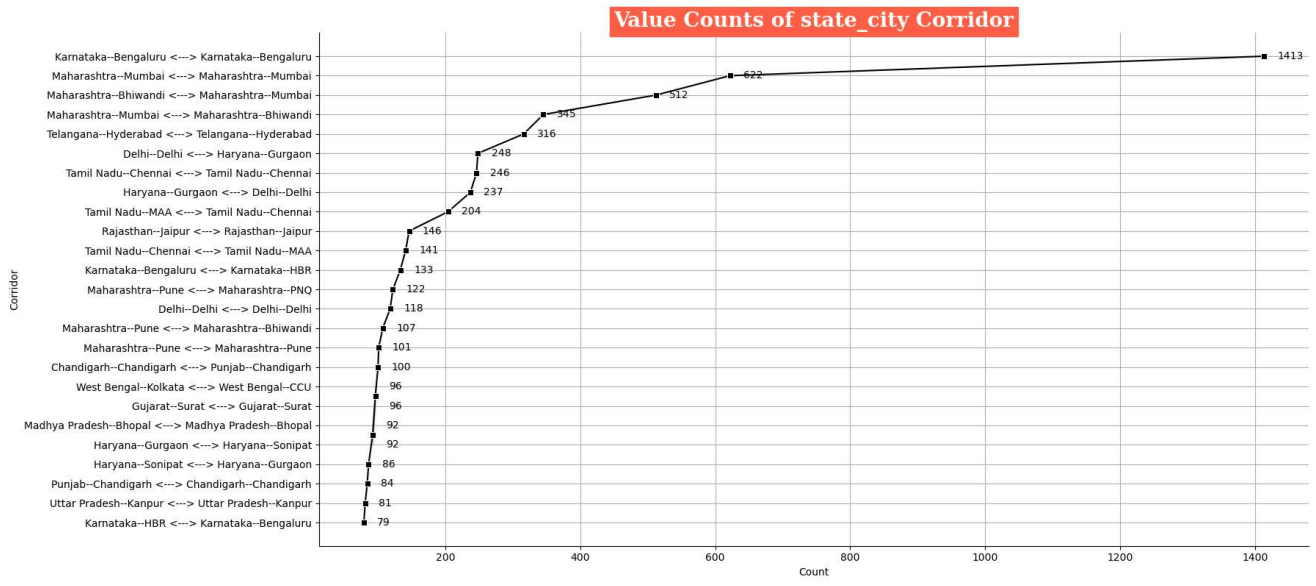
- The route between **Bangalore_Nelamangala_H** to **Bengaluru_KGAirport_HB**,**Bengaluru_Bomsndra_HB** sees the highest package volume, with 151 and 127 packages sent respectively.
- **Bengaluru_Bommasandra_HB** to **Bengaluru_KGAirport_HB** is also popular, with 121 packages sent.
- **Bengaluru_KGAirport_HB**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

```
de['state_corridor'] = de['source_state']+'--'+de['source_city'] +' <---> '+ de['destination_state']+'--'+de['destination_c
de['state_corridor'].value_counts()
```

```
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
```

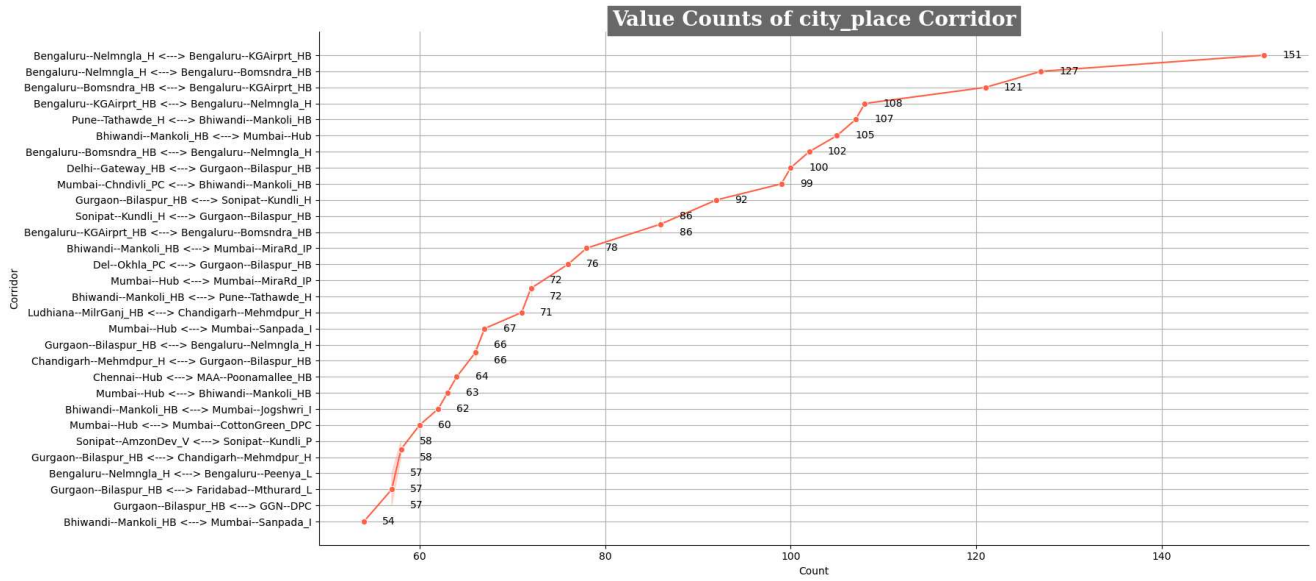
```
state_corridor_counts = de['state_corridor'].value_counts()[:25]

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',c
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()
```




```
de['city_corridor'] = de['source_city']+'--'+de['source_place'] +' <---> '+ de['destination_city']+'--'+de['destination_pla
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
```

```
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',
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.

```
# 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
```

```
7 de['trip_creation_week'] = de['trip_creation_time'].dt.isocalendar().week
```

	segment_key	trip_uuid	data	route_type	trip_creation_time	so
0	trip-153671041653548748+IND209304AAA+IND000000ACB	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Kanpur_C (Utta
1	trip-153671041653548748+IND462022AAA+IND209304AAA	trip-153671041653548748	training	FTL	2018-09-12 00:00:16.535741	Bhopal_ (Madhy
2	trip-153671042288605164+IND561203AAB+IND562101AAA	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_Ch (
3	trip-153671042288605164+IND572101AAA+IND561203AAB	trip-153671042288605164	training	Carting	2018-09-12 00:00:22.886430	Tumkur_ (
4	trip-153671043369099517+IND000000ACB+IND160002AAC	trip-153671043369099517	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bi
...	...	...	...	...	...	
26217	trip-153861115439069069+IND628204AAA+IND627657AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Tirchchndr_Shr (T
26218	trip-153861115439069069+IND628613AAA+IND627005AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Peikulam_Sri (T
26219	trip-153861115439069069+IND628801AAA+IND628204AAA	trip-153861115439069069	test	Carting	2018-10-03 23:59:14.390954	Eral_Busstan
26220	trip-153861118270144424+IND583119AAA+IND583101AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Sandur_Wrc (
26221	trip-153861118270144424+IND583201AAA+IND583119AAA	trip-153861118270144424	test	FTL	2018-10-03 23:59:42.701692	Hospet (

26222 rows × 37 columns

🔷 In-Depth Analysis

```
1 new_df = de.copy()
```

```
1 new_df.columns
```

```
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')
```

```
1 new_df.sample(2)
```

	segment_key	trip_uuid	data	route_type	trip_creation_time	sourc
3330	trip-153694740553026744+IND574104AAA+IND574216AAA	trip-153694740553026744	training	Carting	2018-09-14 17:50:05.530477	Karkala_Marke (Kar
3755	trip-153696604384005633+IND507117AAB+IND507303AAA	trip-153696604384005633	training	FTL	2018-09-14 23:00:43.840397	Manuguru_AskN (Tela

```
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
```

	trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creati
0	trip-153671041653548748	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	trip-153671042288605164	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	trip-153671043369099517	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	trip-153671046011330457	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	trip-153671052974046625	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	trip-153861095625827784	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	
14783	trip-153861104386292051	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	
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	
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	
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	

14787 rows × 35 columns

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

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

14787 rows × 12 columns

```
numerical_columns.describe().T
```

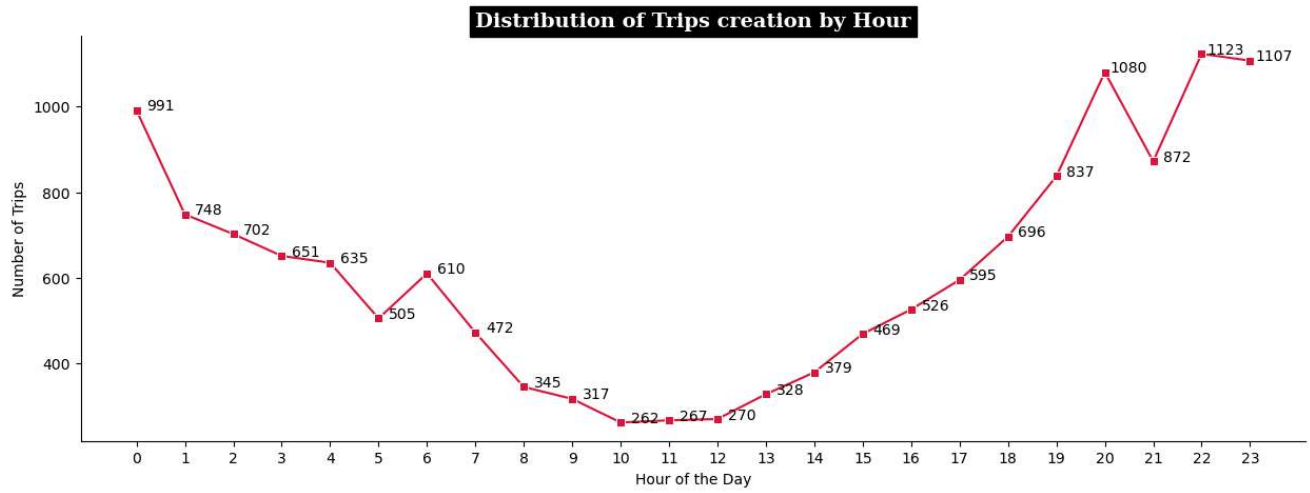
	count	mean	std	min	25%	50%	75%	max
od_time_diff_hour	14787.0	8.840187	10.978880	0.391024	2.494975	4.661846	10.558962	131.642533
start_scan_to_end_scan	14787.0	529.429016	658.254395	23.000000	149.000000	279.000000	632.000000	7898.000000
actual_distance_to_destination	14787.0	164.090195	305.502808	9.002461	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

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

	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) <---> Bengalu...	151
state_corridor	14787	1366	Karnataka--Bengaluru <---> Karnataka--Bengaluru	1333
city_corridor	14787	1737	Bengaluru--Nelmngla_H <---> Bengaluru--KGAirpr...	151

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

```
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',col
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Trips')
sns.despine()
plt.show()
```




```
1 trip_df.sample()
```

	trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creatio
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	

```
1 trip_df.trip_creation_year.value_counts()
```

```
trip_creation_year
2018    14787
Name: count, dtype: int64
```

```
1 trip_df.trip_creation_month.value_counts()
```

```
trip_creation_month
9     13011
10     1776
Name: count, dtype: int64
```

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

```
trip_creation_month
9     87.98945
10    12.01055
Name: proportion, dtype: float64
```

```
1 trip_df.trip_creation_week.value_counts()
```

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

```
1 trip_df.trip_creation_weekday.value_counts(ascending=True)
```

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

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

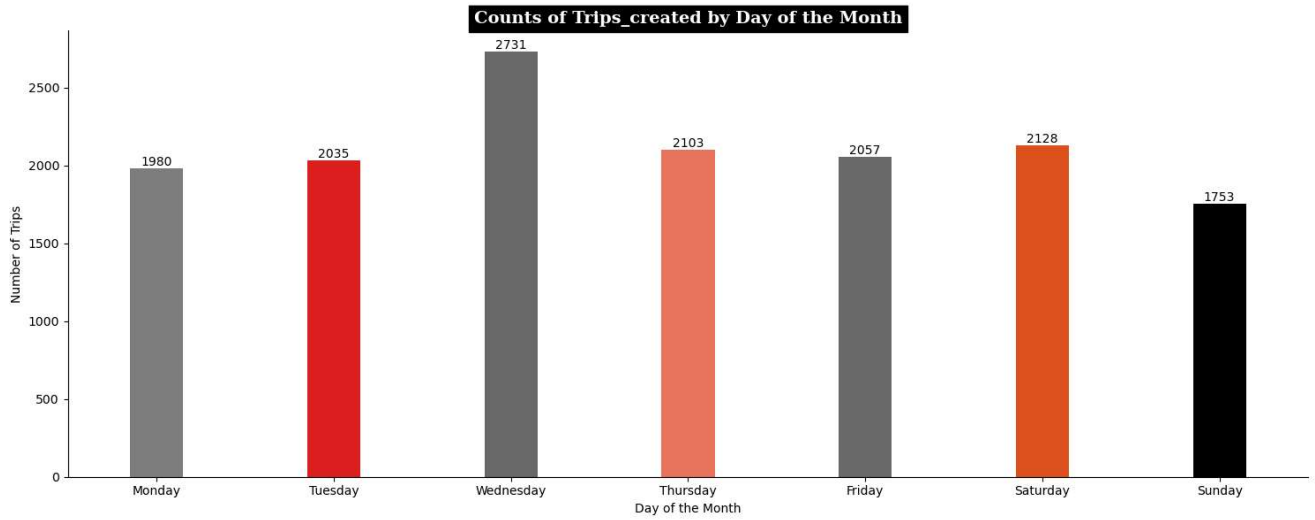
```
trip_df.trip_creation_day.value_counts()
```

```
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
```

```
trip_df.sample()
```

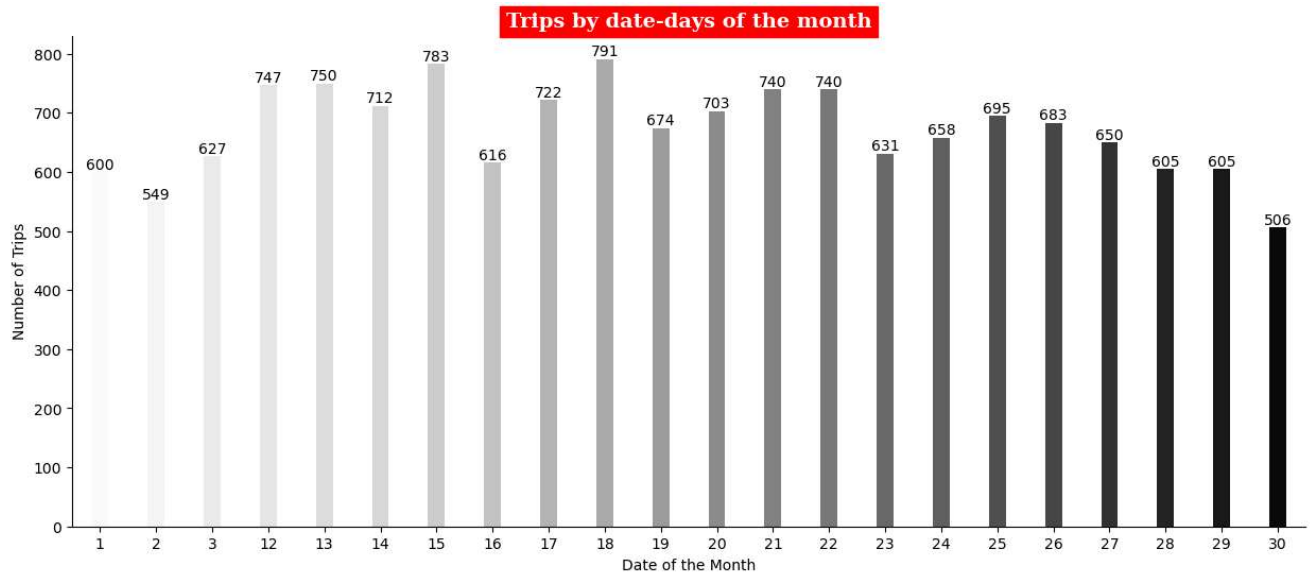
	trip_uuid	data	route_type	od_start_time	od_end_time	od_time_diff_hour	trip_creation_time	trip_creatio
8403	trip-153772987770593762	training	FTL	2018-09-24 00:14:22.226774	2018-09-24 02:42:00.976996	3.847563	2018-09-23 19:11:17.706293	

```
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
plt.xlabel('Day of the Month')
plt.ylabel('Number of Trips')
plt.tight_layout()
sns.despine()
plt.show()
```




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

```
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
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()
```



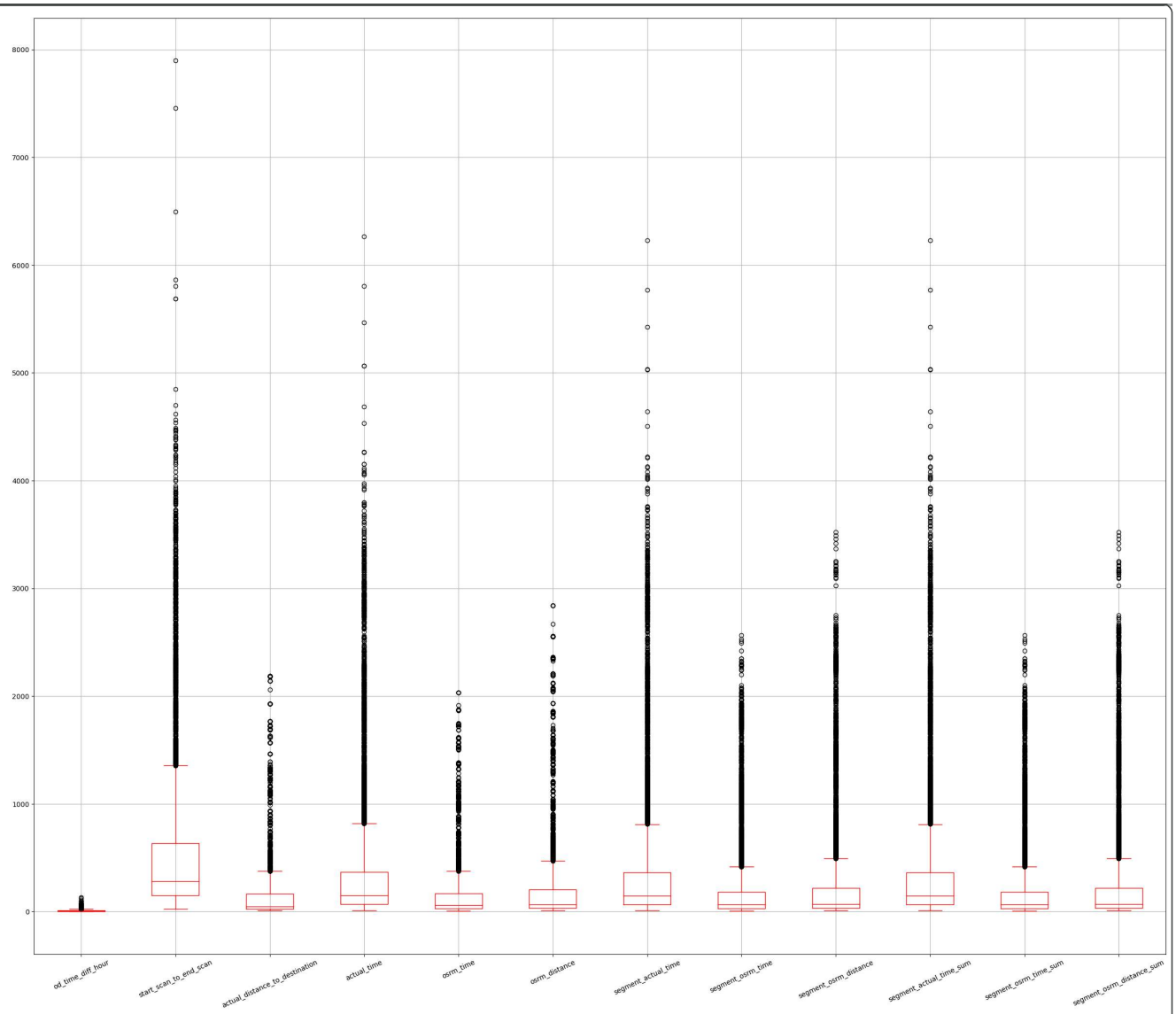
Outlier treatment

```
1 numerical_columns
```

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

14787 rows × 12 columns

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




```
1 numerical_columns.columns
```

```
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')
```

```
1 num_cols = numerical_columns.columns.tolist()
2 num_cols
```

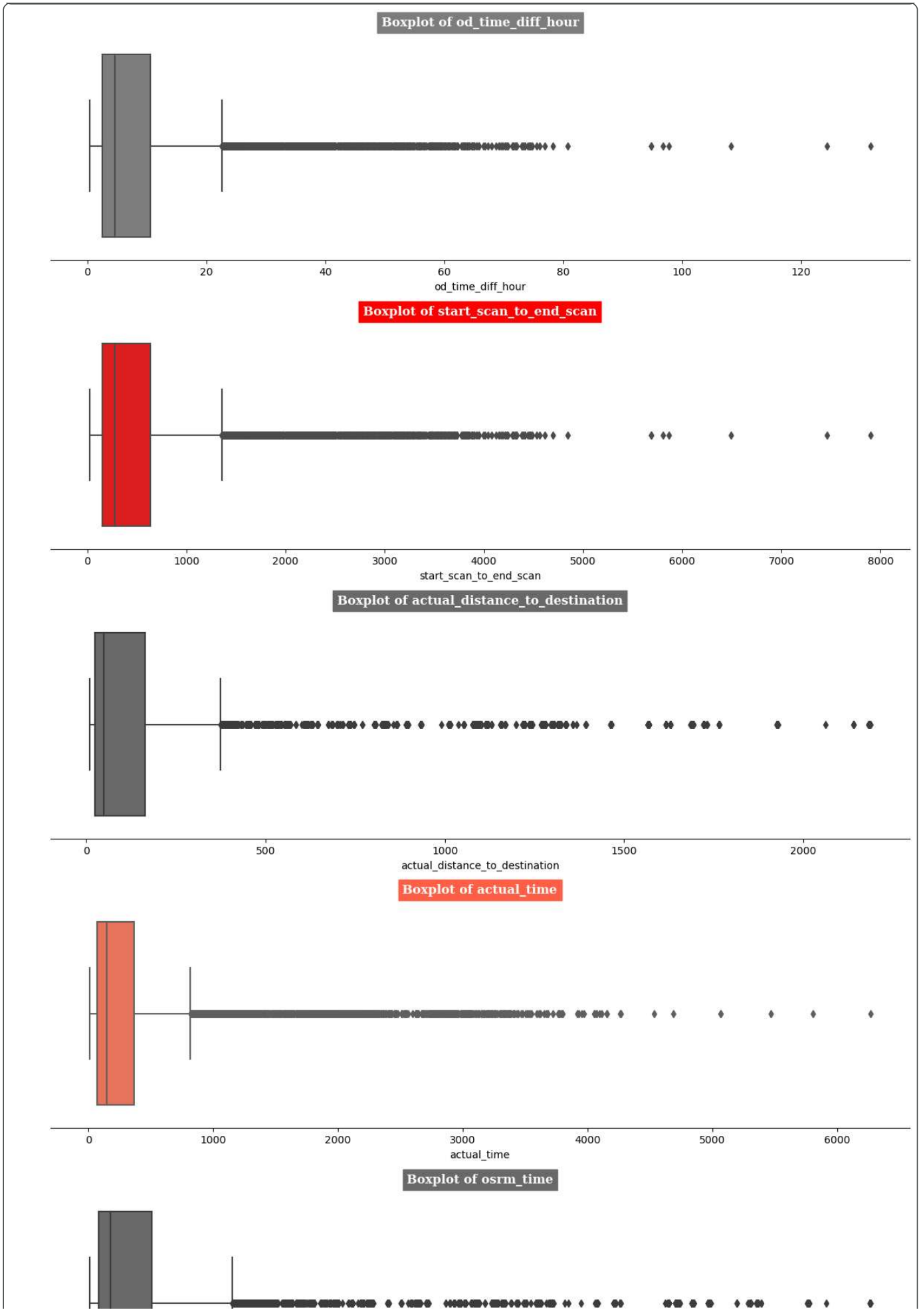
```
['od_time_diff_hour',
 'start_scan_to_end_scan',
 'actual_distance_to_destination',
 'actual_time',
```

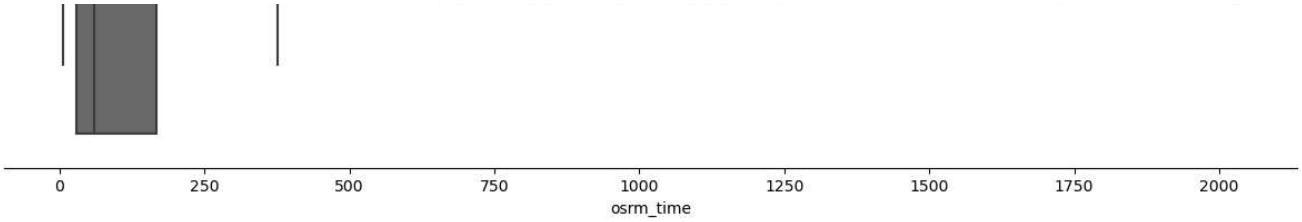
```
 'osrm_time',
 'osrm_distance',
 'segment_actual_time',
 'segment_osrm_time',
 'segment_osrm_distance',
 'segment_actual_time_sum',
 'segment_osrm_time_sum',
 'segment_osrm_distance_sum']
```

```
# 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)
```

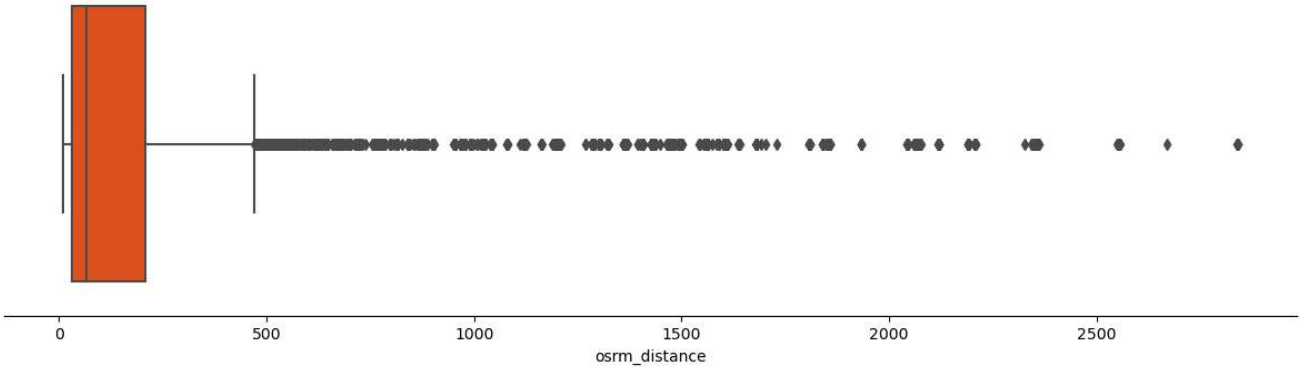
```
od_time_diff_hour                  8.063987
start_scan_to_end_scan           483.000000
actual_distance_to_destination   140.814157
actual_time                      300.000000
osrm_time                        139.000000
osrm_distance                    175.887303
segment_actual_time              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
```

```
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()
```

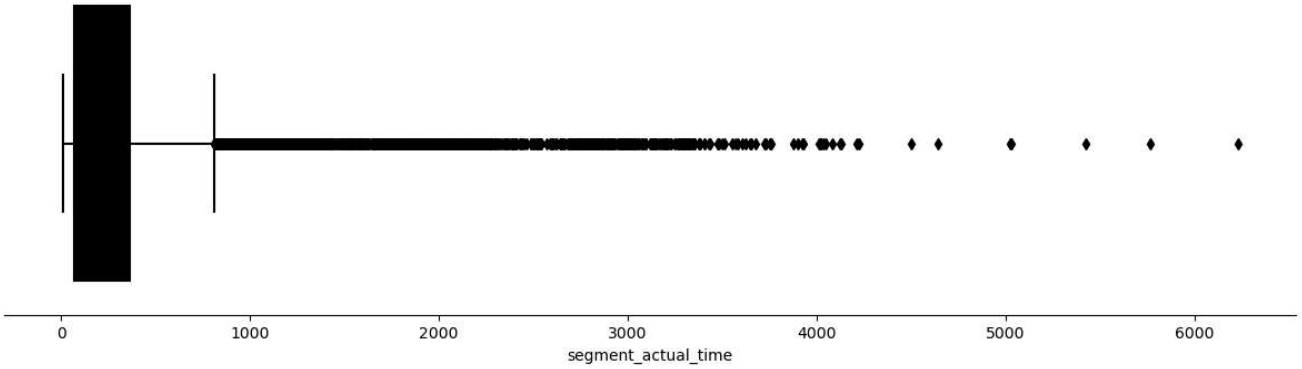

Boxplot of od_time_diff_hour
0
20
40
60
80
100
120
od_time_diff_hour
Boxplot of start_scan_to_end_scan
0
1000
2000
3000
4000
5000
6000
7000
8000
start_scan_to_end_scan
Boxplot of actual_distance_to_destination
0
500
1000
1500
2000
actual_distance_to_destination
Boxplot of actual_time
0
1000
2000
3000
4000
5000
6000
actual_time
Boxplot of osrm_time



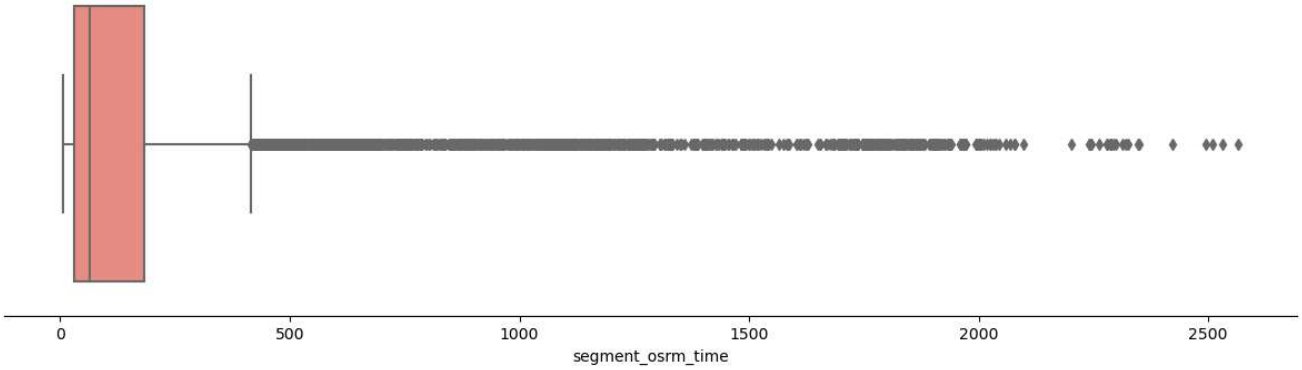
Boxplot of osrm_distance



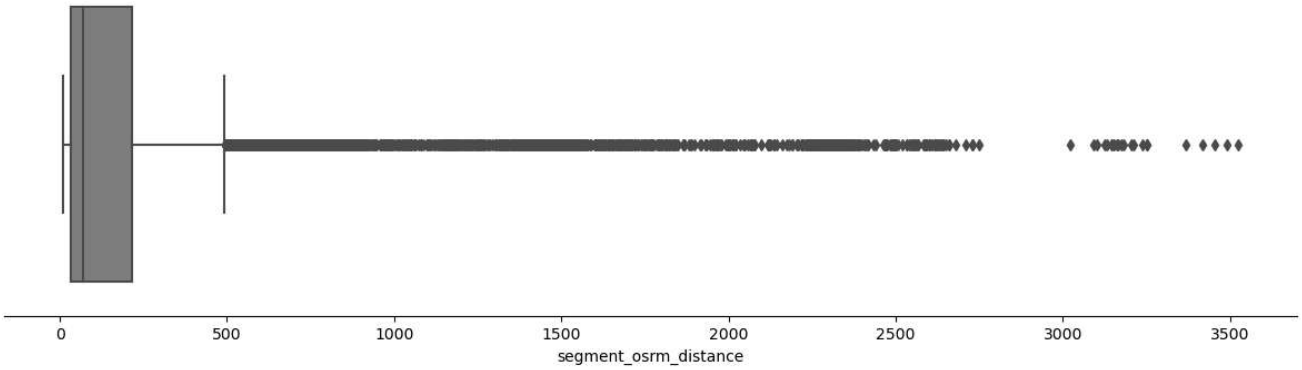
Boxplot of segment_actual_time

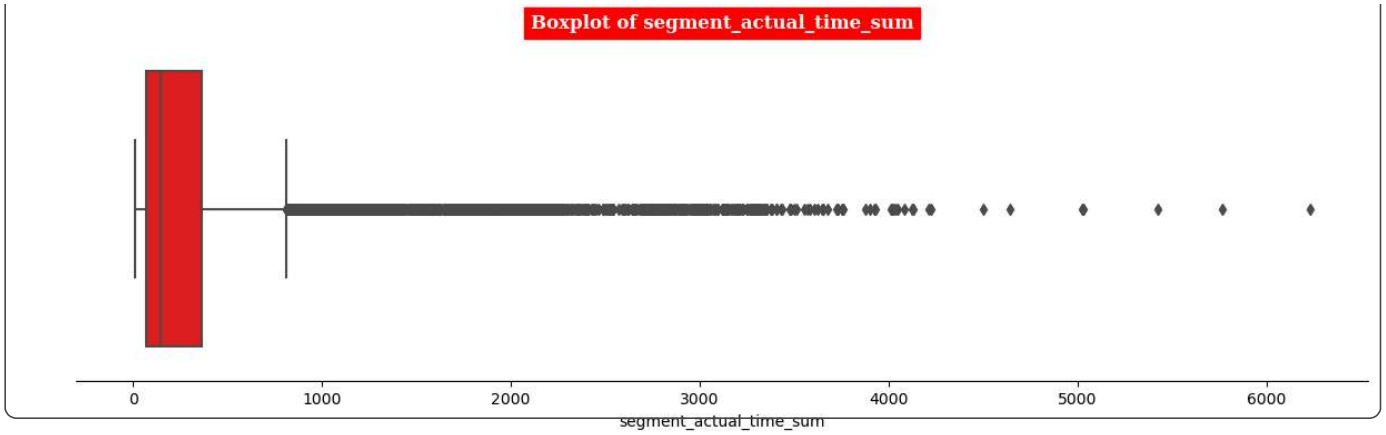


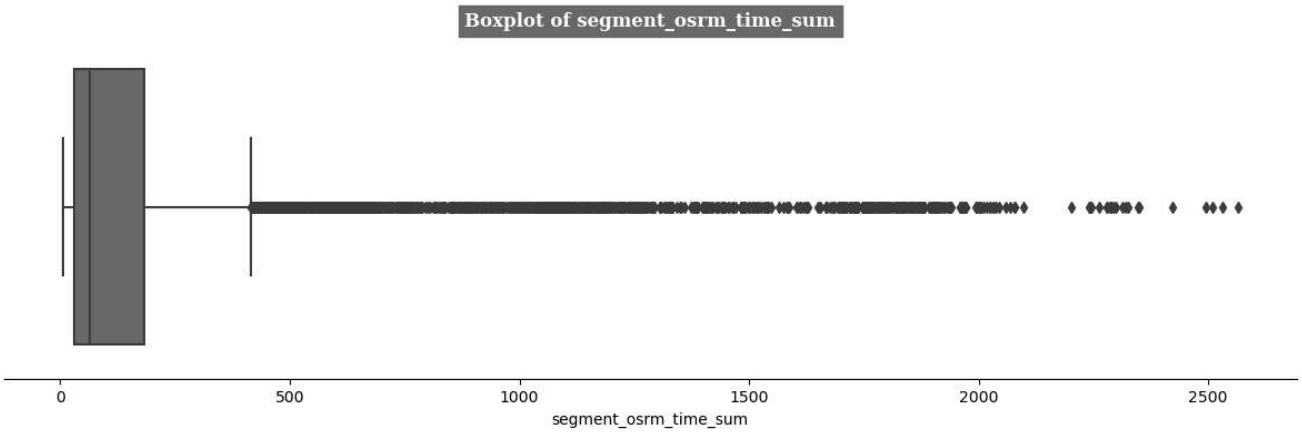
Boxplot of segment_osrm_time

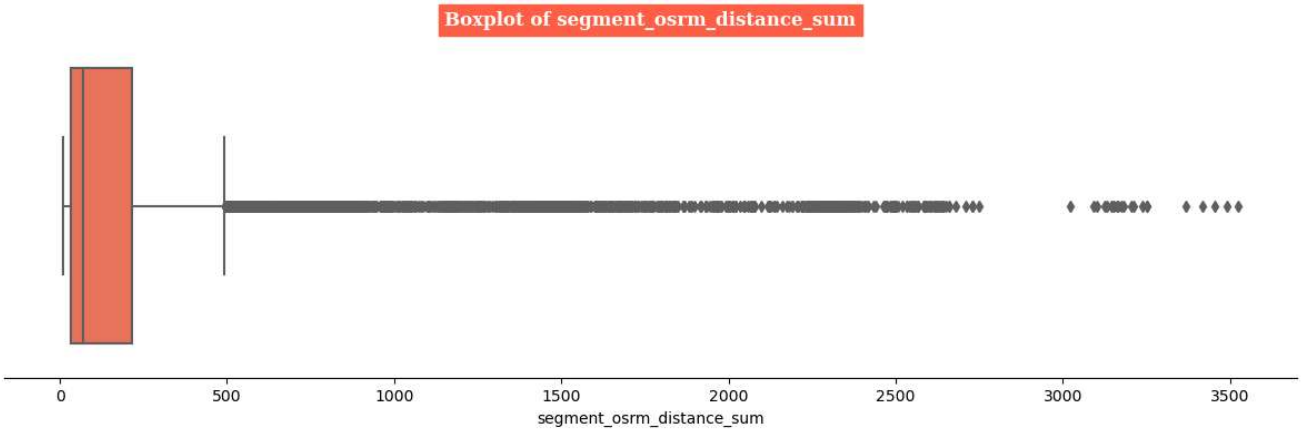


Boxplot of segment_osrm_distance



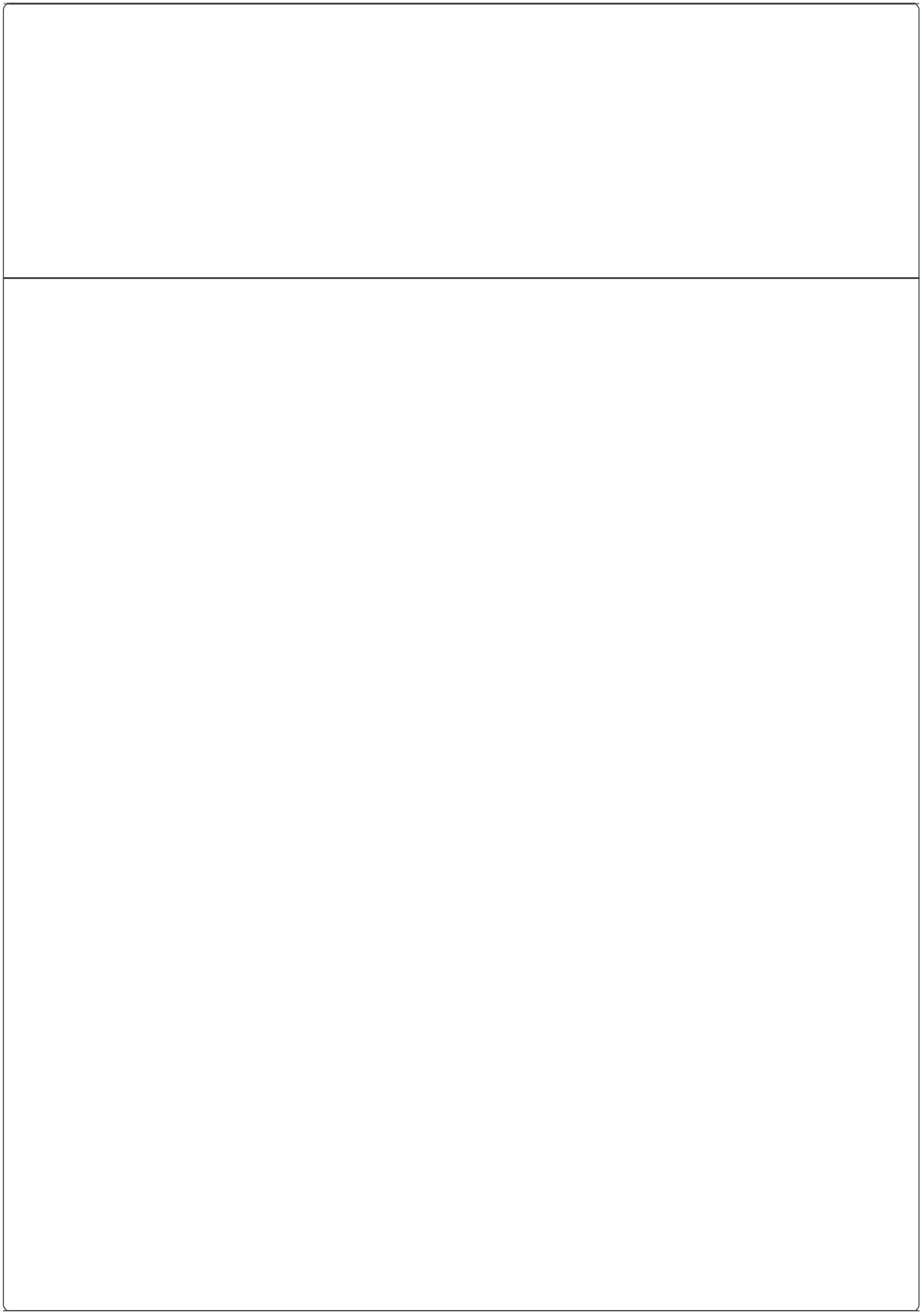

Boxplot of segment_actual_time_sum
0
1000
2000
3000
4000
5000
6000
segment_actual_time_sum


Boxplot of segment_osrm_time_sum
0
500
1000
1500
2000
2500
segment_osrm_time_sum


Boxplot of segment_osrm_distance_sum
0
500
1000
1500
2000
2500
3000
3500
segment_osrm_distance_sum

Outlier Removal

```
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

    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], colc

    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

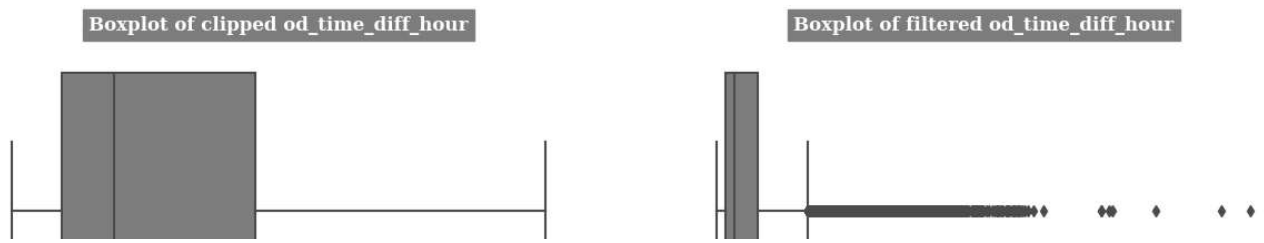
	od_time_diff_hour
0	22.654942
1	3.026865
2	22.654942
3	1.674916
4	11.972484
...	...
14782	4.300482
14783	1.009842
14784	7.035331
14785	5.808548
14786	5.906793

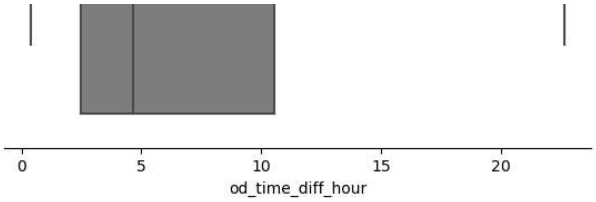
14787 rows × 1 columns

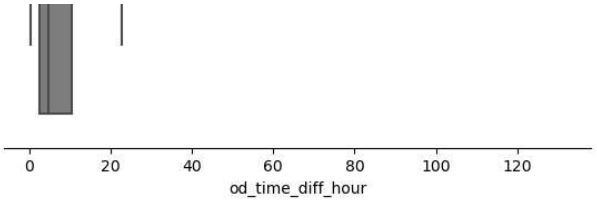
Filtered data of od_time_diff_hour

	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







	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

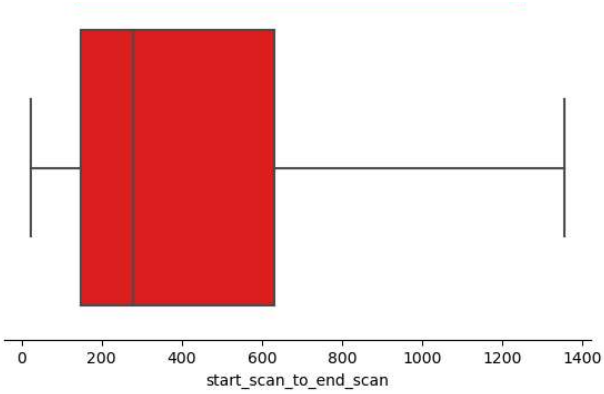
Filtered data of start_scan_to_end_scan

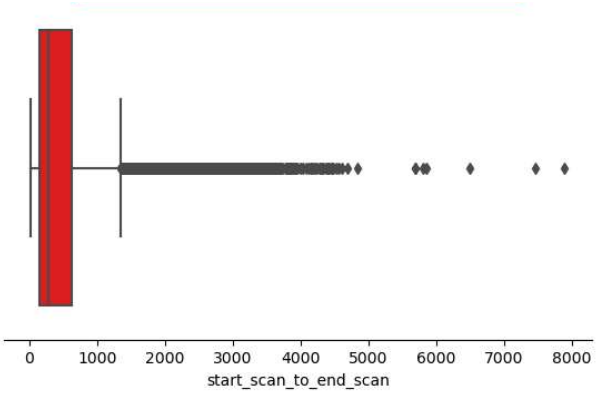
	start_scan_to_end_scan
0	2259.0
1	180.0
2	3933.0
3	100.0
4	717.0
...	...
14782	257.0
14783	60.0
14784	421.0
14785	347.0
14786	353.0

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

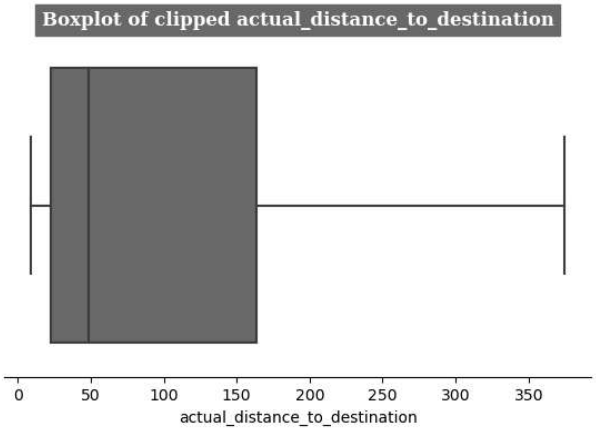
14787 rows × 1 columns

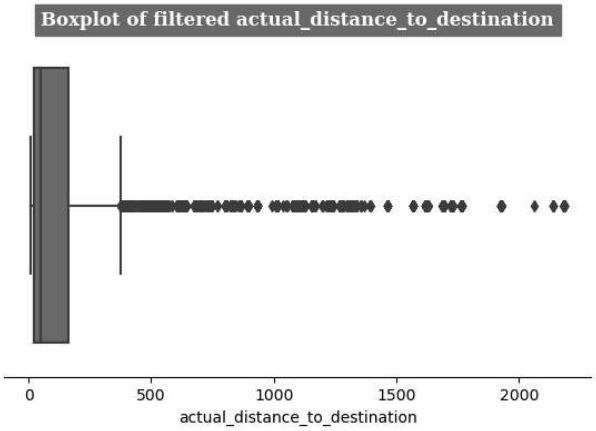
Filtered data of actual_distance_to_destination

	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





	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

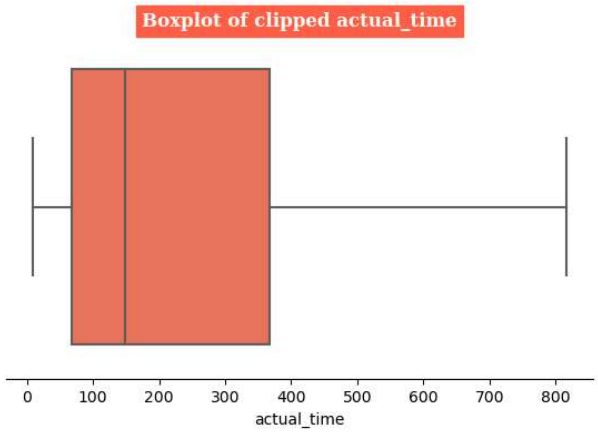
14787 rows × 1 columns

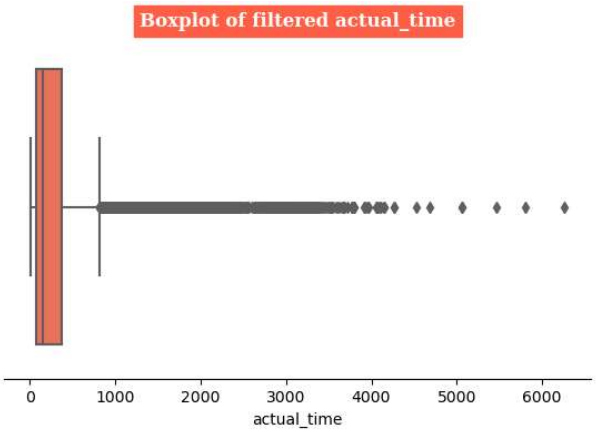
```
Filtered data of actual_time
```

	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





	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

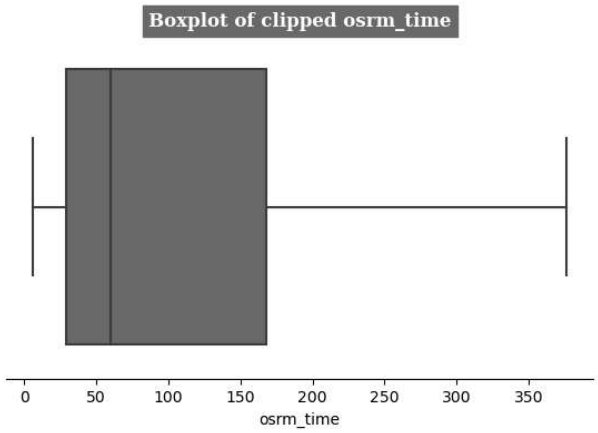
14787 rows × 1 columns

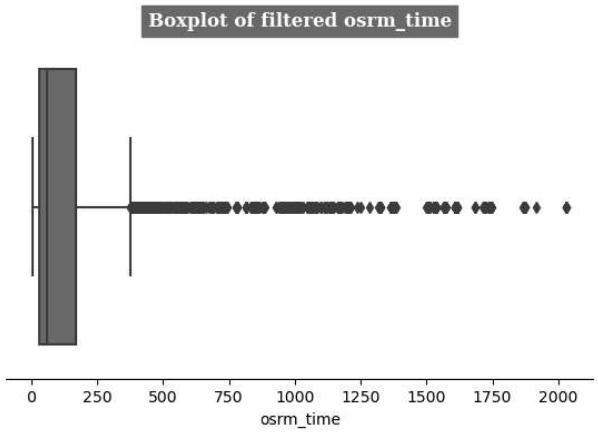
```
Filtered data of osrm_time
```

	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





	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

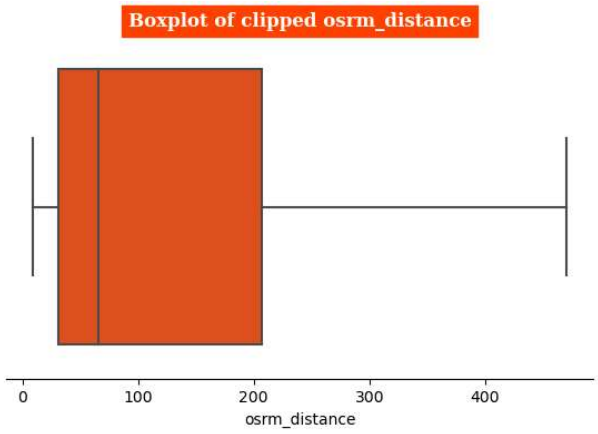
14785	171.110306
14786	80.578705

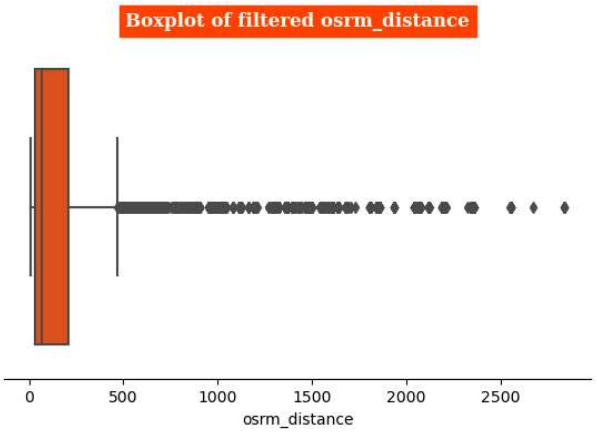
14787 rows × 1 columns

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





	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

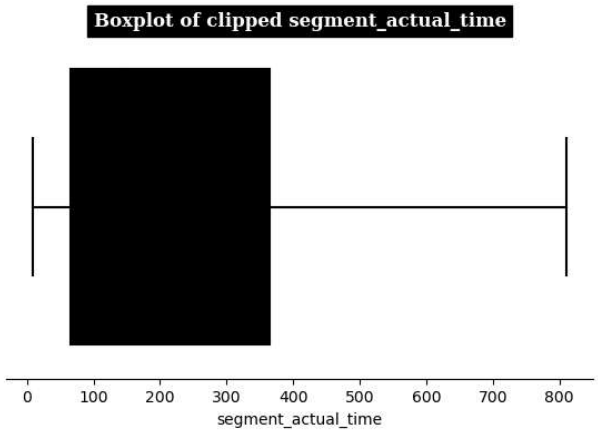
4	340.0
...	...
14782	82.0
14783	21.0
14784	281.0
14785	258.0
14786	274.0

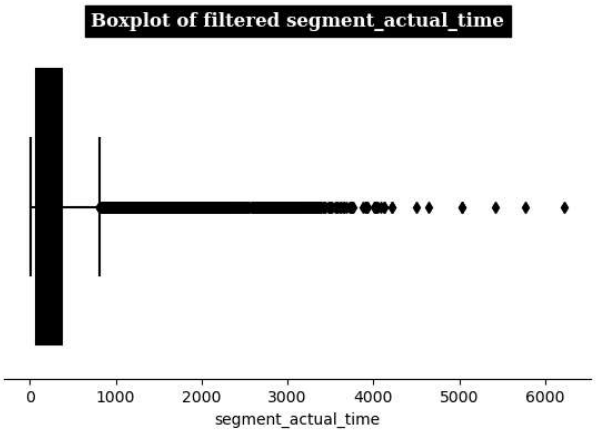
14787 rows × 1 columns

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





	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

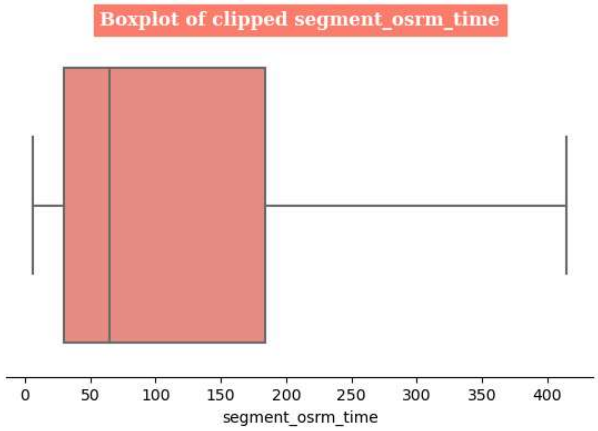
0	415.0
1	65.0
2	415.0
3	16.0
4	115.0
...	...
14782	62.0
14783	11.0
14784	88.0
14785	221.0
14786	67.0

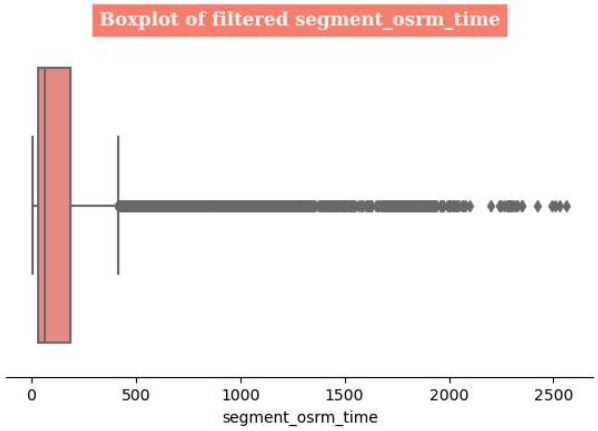
14787 rows × 1 columns

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





	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
```

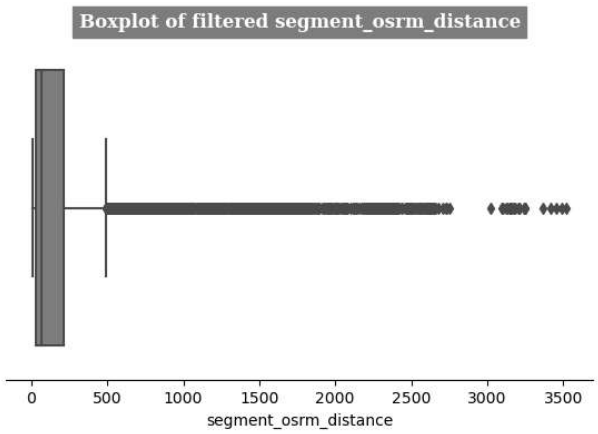
	segment_osrm_distance
0	492.533245
1	84.189400
2	492.533245
3	19.876600
4	146.791901
...	...
14782	64.855103
14783	16.088299
14784	104.886597
14785	223.532394
14786	80.578705

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



	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

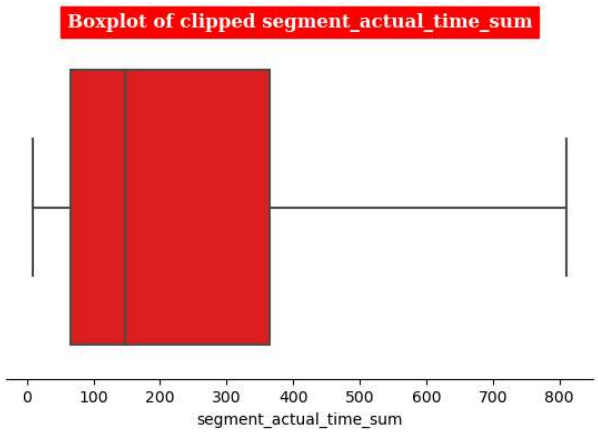
	segment_actual_time_sum
0	811.0
1	141.0
2	811.0
3	59.0
4	340.0
...	...
14782	82.0
14783	21.0
14784	281.0
14785	258.0
14786	274.0

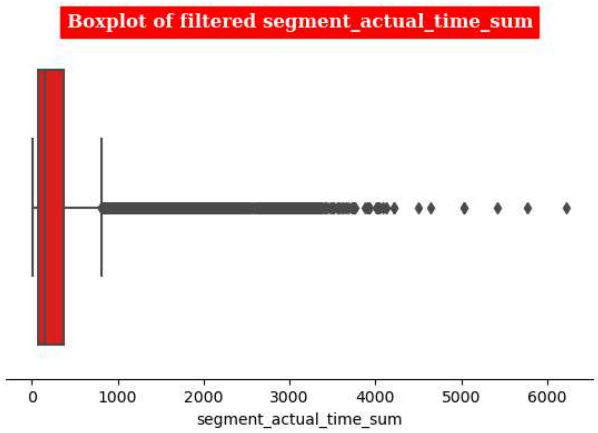
14787 rows × 1 columns

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





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

Clipped data of segment_osrm_time_sum

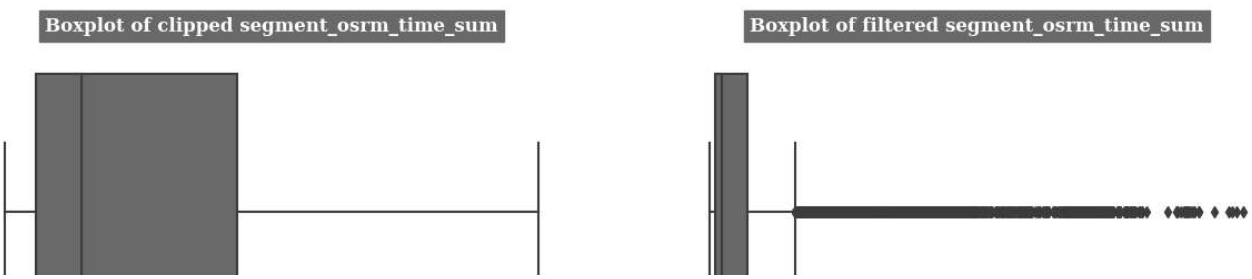
	segment_osrm_time_sum
0	415.0
1	65.0
2	415.0
3	16.0
4	115.0
...	...
14782	62.0
14783	11.0
14784	88.0
14785	221.0
14786	67.0

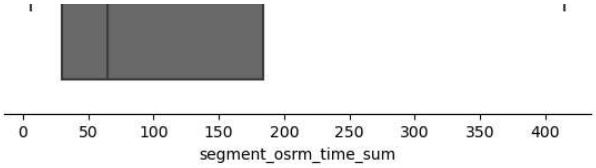
14787 rows × 1 columns

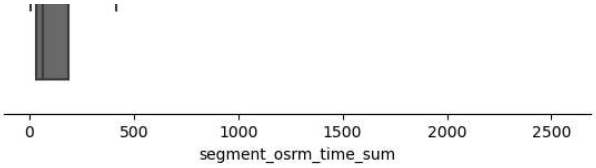
Filtered data of segment_osrm_time_sum

	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







	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

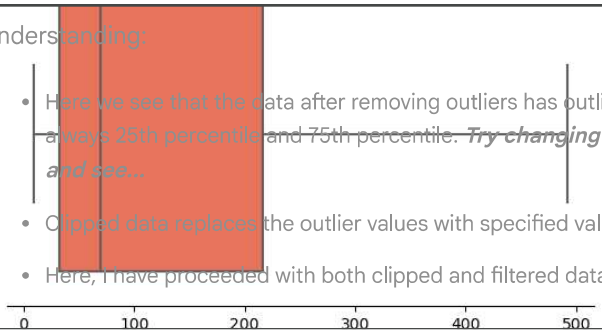
14787 rows × 1 columns

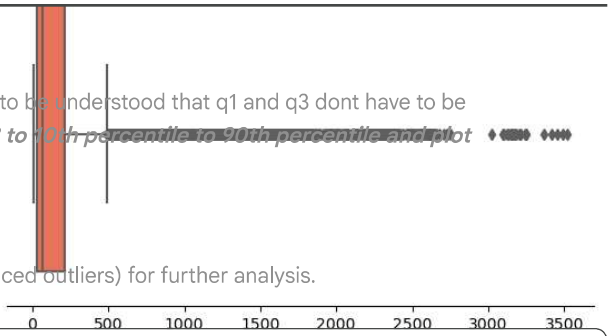
Boxplot of clipped segment_osrm_distance_sum

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.






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

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

14787 rows × 12 columns

```
num_cols
```

```
['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']
```

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

14787 rows × 12 columns

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

14787 rows × 12 columns

```
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()
```


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