Copy_of_scaler_delhivery_case_study

December 14, 2024

[1]:
```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/
↪original/delhivery_data.csv
```

```
Downloading…
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/ori
ginal/delhivery_data.csv
To: /content/delhivery_data.csv
100% 55.6M/55.6M [00:00<00:00, 108MB/s]
```

[2]:
```
import pandas as pd
pd.set_option('display.max_columns', None)
df = pd.read_csv('delhivery_data.csv')
df.head()
```

[2]:
```
       data          trip_creation_time  \
0  training  2018-09-20 02:35:36.476840
1  training  2018-09-20 02:35:36.476840
2  training  2018-09-20 02:35:36.476840
3  training  2018-09-20 02:35:36.476840
4  training  2018-09-20 02:35:36.476840

                                 route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                 trip_uuid source_center                 source_name  \
0  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)

  destination_center              destination_name  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
```

```
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)

              od_start_time                  od_end_time  \
0  2018-09-20 03:21:32.418600  2018-09-20 04:47:45.236797
1  2018-09-20 03:21:32.418600  2018-09-20 04:47:45.236797
2  2018-09-20 03:21:32.418600  2018-09-20 04:47:45.236797
3  2018-09-20 03:21:32.418600  2018-09-20 04:47:45.236797
4  2018-09-20 03:21:32.418600  2018-09-20 04:47:45.236797

   start_scan_to_end_scan  is_cutoff  cutoff_factor  \
0                    86.0       True              9
1                    86.0       True             18
2                    86.0       True             27
3                    86.0       True             36
4                    86.0      False             39

             cutoff_timestamp  actual_distance_to_destination  actual_time  \
0         2018-09-20 04:27:55                       10.435660         14.0
1         2018-09-20 04:17:55                       18.936842         24.0
2  2018-09-20 04:01:19.505586                       27.637279         40.0
3         2018-09-20 03:39:57                       36.118028         62.0
4         2018-09-20 03:33:55                       39.386040         68.0

   osrm_time  osrm_distance    factor  segment_actual_time  segment_osrm_time  \
0       11.0        11.9653  1.272727                 14.0               11.0
1       20.0        21.7243  1.200000                 10.0                9.0
2       28.0        32.5395  1.428571                 16.0                7.0
3       40.0        45.5620  1.550000                 21.0               12.0
4       44.0        54.2181  1.545455                  6.0                5.0

   segment_osrm_distance  segment_factor
0                11.9653        1.272727
1                 9.7590        1.111111
2                10.8152        2.285714
3                13.0224        1.750000
4                 3.9153        1.200000
```

[3]:
```
df.shape
```

[3]: (144867, 24)

[4]:
```
df.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
```

[5]:
```
df.describe()
```

[5]:
```
       start_scan_to_end_scan  cutoff_factor  actual_distance_to_destination  \
count           144867.000000  144867.000000                   144867.000000
mean               961.262986     232.926567                      234.073372
std               1037.012769     344.755577                      344.990009
min                 20.000000       9.000000                        9.000045
25%                161.000000      22.000000                       23.355874
50%                449.000000      66.000000                       66.126571
75%               1634.000000     286.000000                      286.708875
max               7898.000000    1927.000000                     1927.447705

         actual_time      osrm_time  osrm_distance         factor  \
count  144867.000000  144867.000000  144867.000000  144867.000000
mean      416.927527     213.868272     284.771297       2.120107
std       598.103621     308.011085     421.119294       1.715421
min         9.000000       6.000000       9.008200       0.144000
```

25%	51.000000	27.000000	29.914700	1.604264
50%	132.000000	64.000000	78.525800	1.857143
75%	513.000000	257.000000	343.193250	2.213483
max	4532.000000	1686.000000	2326.199100	77.387097

	segment_actual_time	segment_osrm_time	segment_osrm_distance	\
count	144867.000000	144867.000000	144867.00000	
mean	36.196111	18.507548	22.82902	
std	53.571158	14.775960	17.86066	
min	-244.000000	0.000000	0.00000	
25%	20.000000	11.000000	12.07010	
50%	29.000000	17.000000	23.51300	
75%	40.000000	22.000000	27.81325	
max	3051.000000	1611.000000	2191.40370	

	segment_factor
count	144867.000000
mean	2.218368
std	4.847530
min	-23.444444
25%	1.347826
50%	1.684211
75%	2.250000
max	574.250000

[6]:
```
df.describe(include=object)
```

[6]:

	data	trip_creation_time	\
count	144867	144867	
unique	2	14817	
top	training	2018-09-28 05:23:15.359220	
freq	104858	101	

	route_schedule_uuid	route_type	\
count	144867	144867	
unique	1504	2	
top	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f…	FTL	
freq	1812	99660	

	trip_uuid	source_center	source_name	\
count	144867	144867	144574	
unique	14817	1508	1498	
top	trip-153811219535896559	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	
freq	101	23347	23347	

	destination_center	destination_name	\
count	144867	144606	

```
unique              1481                              1468
top          IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
freq               15192                             15192

                     od_start_time                 od_end_time  \
count                       144867                      144867
unique                       26369                       26369
top     2018-09-21 18:37:09.322207  2018-09-24 09:59:15.691618
freq                            81                          81

           cutoff_timestamp
count                144867
unique                93180
top     2018-09-24 05:19:20
freq                     40
```

[7]:
```python
df.isna().sum()
```

[7]:
```
data                                0
trip_creation_time                  0
route_schedule_uuid                 0
route_type                          0
trip_uuid                           0
source_center                       0
source_name                       293
destination_center                  0
destination_name                  261
od_start_time                       0
od_end_time                         0
start_scan_to_end_scan              0
is_cutoff                           0
cutoff_factor                       0
cutoff_timestamp                    0
actual_distance_to_destination      0
actual_time                         0
osrm_time                           0
osrm_distance                       0
factor                              0
segment_actual_time                 0
segment_osrm_time                   0
segment_osrm_distance               0
segment_factor                      0
dtype: int64
```

[8]:
```python
#  percentage of null values in each columns
(df.isna().sum() / len(df)) * 100
```

[8]:
```
data                              0.000000
trip_creation_time                0.000000
route_schedule_uuid               0.000000
route_type                        0.000000
trip_uuid                         0.000000
source_center                     0.000000
source_name                       0.202254
destination_center                0.000000
destination_name                  0.180165
od_start_time                     0.000000
od_end_time                       0.000000
start_scan_to_end_scan            0.000000
is_cutoff                         0.000000
cutoff_factor                     0.000000
cutoff_timestamp                  0.000000
actual_distance_to_destination    0.000000
actual_time                       0.000000
osrm_time                         0.000000
osrm_distance                     0.000000
factor                            0.000000
segment_actual_time               0.000000
segment_osrm_time                 0.000000
segment_osrm_distance             0.000000
segment_factor                    0.000000
dtype: float64
```

[9]:
```python
# To fill in source names,
# we find the pairs source_names, source_centers which are one to one

# so if we find one source center for a missing source_name we can fill that␣
↪value


# df[df['source_name'].notnull() & (df['source_name'].isnull() &␣
↪df['source_center']) ]

# All unique pairs

source_pairs = df.loc[df['source_name'].
↪notnull(),['source_name','source_center']].drop_duplicates()
source_pairs
missing_sources = df.loc[df['source_name'].isnull(), 'source_center'].unique()␣
↪# includes the source_centers for the missing source_names

# now we can match the source_centers in the source_pairs to get the␣
↪corresponding source_names
```

```python
for centers in missing_sources:
    matches = source_pairs[source_pairs['source_center'] == centers]
    if not matches.empty:
        print(f"Found match for {centers}  ==> {matches}")
    else:
        print(f"No matches found for {centers}")
```

```
No matches found for IND342902A1B
No matches found for IND577116AAA
No matches found for IND282002AAD
No matches found for IND465333A1B
No matches found for IND841301AAC
No matches found for IND509103AAC
No matches found for IND126116AAA
No matches found for IND331022A1B
No matches found for IND505326AAB
No matches found for IND852118A1B
```

[10]:

```python
destination_pairs = df.loc[df['destination_name'].
 ↪notnull(),['destination_name','destination_center']].drop_duplicates()
destination_pairs
missing_destinations = df.loc[df['destination_name'].isnull(),␣
 ↪'destination_center'].unique() # includes the destination_centers for the␣
 ↪missing destination_names

# now we can match the destination_centers in the destination_pairs to get the␣
 ↪corresponding destination_names

for centers in missing_destinations:
    matches = destination_pairs[destination_pairs['destination_center'] ==␣
 ↪centers]
    if not matches.empty:
        print(f"Found match for {centers}  ==> {matches}")
    else:
        print(f"No matches found for {centers}")
```

```
No matches found for IND342902A1B
No matches found for IND577116AAA
No matches found for IND282002AAD
No matches found for IND465333A1B
No matches found for IND841301AAC
No matches found for IND505326AAB
No matches found for IND852118A1B
No matches found for IND126116AAA
No matches found for IND509103AAC
No matches found for IND221005A1A
No matches found for IND250002AAC
```

```
No matches found for IND331001A1C
No matches found for IND122015AAC
```

[11]:
```python
# Convert to datetime
df["trip_creation_time"] = pd.to_datetime(df["trip_creation_time"])
df["od_start_time"] = pd.to_datetime(df["od_start_time"])
df["od_end_time"] = pd.to_datetime(df["od_end_time"])
```

[12]:
```python
trip_by_month = df['trip_creation_time'].dt.month_name().value_counts()
trip_by_month
```

[12]:
```
trip_creation_time
September    127349
October       17518
Name: count, dtype: int64
```

[13]:
```python
df["trip_creation_time"].dt.year.value_counts()
```

[13]:
```
trip_creation_time
2018    144867
Name: count, dtype: int64
```

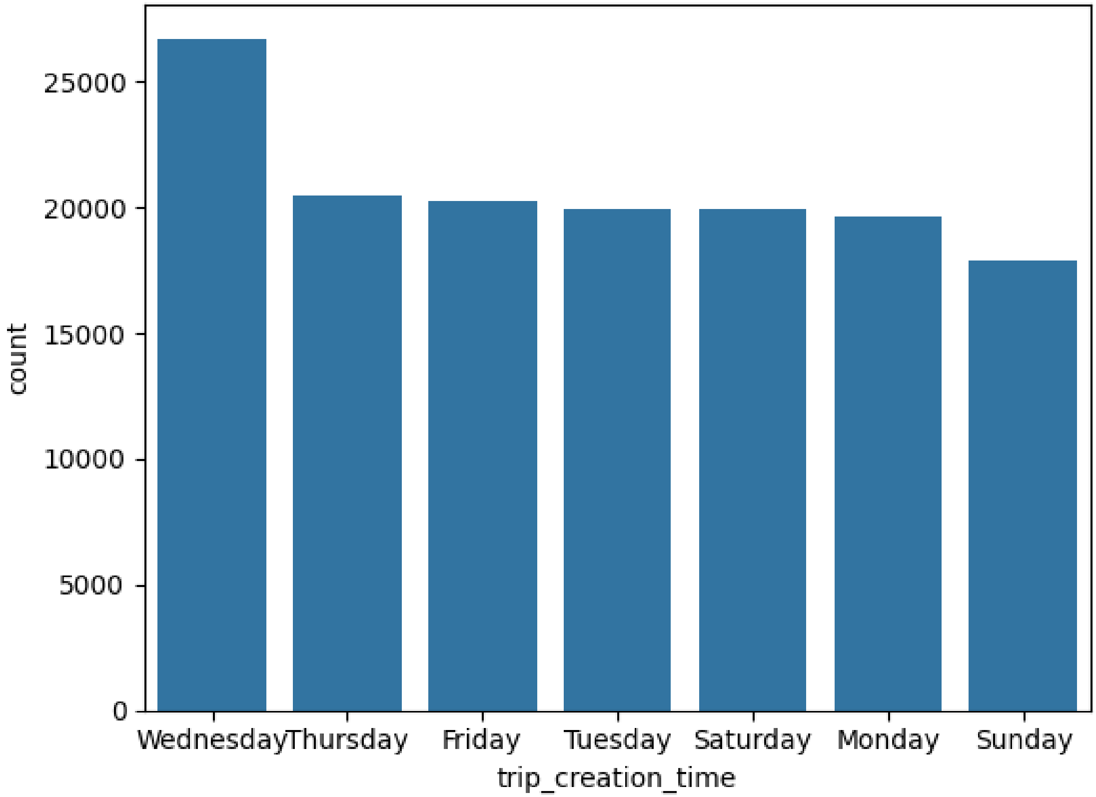
[14]:
```python
trip_day = df["trip_creation_time"].dt.day_name().value_counts()
trip_day
```

[14]:
```
trip_creation_time
Wednesday    26732
Thursday     20481
Friday       20242
Tuesday      19961
Saturday     19936
Monday       19645
Sunday       17870
Name: count, dtype: int64
```

[15]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tqdm
```

[16]:
```python
trip_day = trip_day.reset_index()
sns.barplot(data=trip_day, x='trip_creation_time', y='count')
```

[16]:
```
<Axes: xlabel='trip_creation_time', ylabel='count'>
```



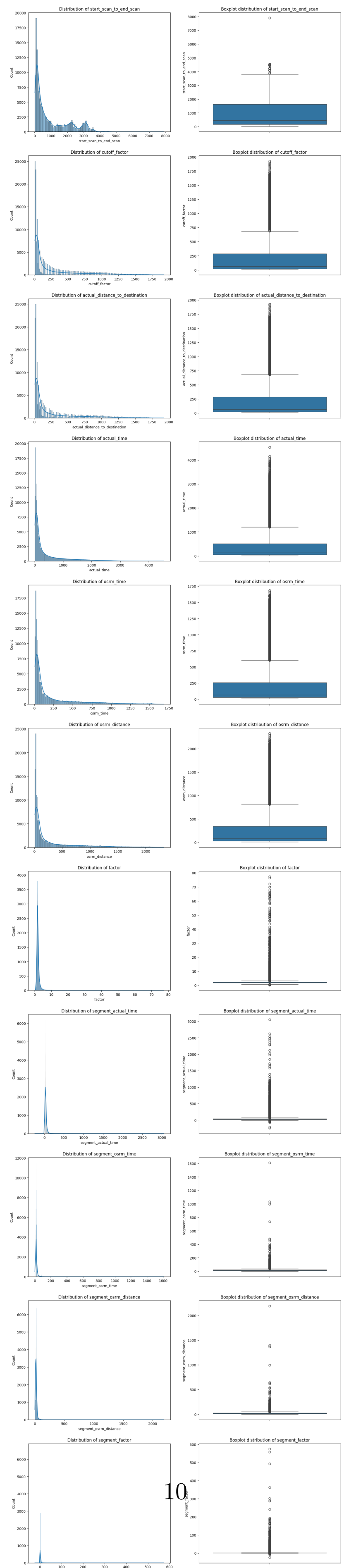
[16]:

```

```

[17]:

```
# univariate analysis
variables = df.select_dtypes(include=np.number).columns.tolist()

fig, ax = plt.subplots(11, 2, figsize=(16,80))


for i,v in tqdm.tqdm(enumerate(variables)):
  sns.histplot(data=df[v], kde=True, ax=ax[i,0])
  ax[i,0].set_title(f'Distribution of {v}')

  sns.boxplot(y=df[v], ax=ax[i,1], data=df)
  ax[i,1].set_title(f'Boxplot distribution of {v}')


plt.show()
```

```
11it [00:58,  5.32s/it]
```


Distribution of start_scan_to_end_scan
Boxplot distribution of start_scan_to_end_scan
Distribution of cutoff_factor
Boxplot distribution of cutoff_factor
Distribution of actual_distance_to_destination
Boxplot distribution of actual_distance_to_destination
Distribution of actual_time
Boxplot distribution of actual_time
Distribution of osrm_time
Boxplot distribution of osrm_time
Distribution of osrm_distance
Boxplot distribution of osrm_distance
Distribution of factor
Boxplot distribution of factor
Distribution of segment_actual_time
Boxplot distribution of segment_actual_time
Distribution of segment_osrm_time
Boxplot distribution of segment_osrm_time
Distribution of segment_osrm_distance
Boxplot distribution of segment_osrm_distance
Distribution of segment_factor
Boxplot distribution of segment_factor

[18]:
```
df['source_name'].unique().tolist()[:30],df['destination_name'].unique().
↪tolist()[:30]
```

[18]:
```
(['Anand_VUNagar_DC (Gujarat)',
  'Khambhat_MotvdDPP_D (Gujarat)',
  'Bhiwandi_Mankoli_HB (Maharashtra)',
  'LowerParel_CP (Maharashtra)',
  'Bangalore_Nelmngla_H (Karnataka)',
  'Bengaluru_Bomsndra_HB (Karnataka)',
  'Ludhiana_GillChwk_DC (Punjab)',
  'Jagraon_DC (Punjab)',
  'Raikot_DC (Punjab)',
  'Junagadh_DPC (Gujarat)',
  'Veraval_DC (Gujarat)',
  'Kodinar_NCplxDPP_D (Gujarat)',
  'Una_Mamlatdr_DC (Gujarat)',
  'Talala_SsnRdDPP_D (Gujarat)',
  'Sonipat_Kundli_H (Haryana)',
  'Roorkee_IOTCEncl_L (Uttarakhand)',
  'Haridwar (Uttarakhand)',
  'MAA_Poonamallee_HB (Tamil Nadu)',
  'Ludhiana_MilrGanj_HB (Punjab)',
  'Jalandhar_DPC (Punjab)',
  'Gurgaon_Begumpur_CP (Haryana)',
  'Jaipur_Hub (Rajasthan)',
  'Ajmer_FoySGRRD_I (Rajasthan)',
  'Pali_Nayagaon_I (Rajasthan)',
  'Jodhpur_Basni_I (Rajasthan)',
  'Piparcity_BsstdDPP_D (Rajasthan)',
  nan,
  'Hyderabad_Chikdply_C (Telangana)',
  'Bhopal_Trnsport_H (Madhya Pradesh)',
  'Kanpur_Central_H_6 (Uttar Pradesh)'],
 ['Khambhat_MotvdDPP_D (Gujarat)',
  'Anand_Vaghasi_IP (Gujarat)',
  'Pune_Tathawde_H (Maharashtra)',
  'Mumbai_Chndivli_PC (Maharashtra)',
  'Bengaluru_Bomsndra_HB (Karnataka)',
  'Aluva_Peedika_H (Kerala)',
  'Jagraon_DC (Punjab)',
  'Raikot_DC (Punjab)',
  'Ludhiana_MilrGanj_HB (Punjab)',
  'Bengaluru_Bnnrghta_L (Karnataka)',
  'Junagadh_keshod_DC (Gujarat)',
  'Kodinar_NCplxDPP_D (Gujarat)',
```

```
 'Una_Mamlatdr_DC (Gujarat)',
 'Talala_SsnRdDPP_D (Gujarat)',
 'Junagadh_DPC (Gujarat)',
 'Roorkee_IOTCEncl_L (Uttarakhand)',
 'Haridwar (Uttarakhand)',
 'Rishikesh_DC (Uttarakhand)',
 'Chennai_Hub (Tamil Nadu)',
 'Jalandhar_DPC (Punjab)',
 'Amritsar_DPC (Punjab)',
 'Gurgaon_Bilaspur_P (Haryana)',
 'Ajmer_FoySGRRD_I (Rajasthan)',
 'Pali_Nayagaon_I (Rajasthan)',
 'Jodhpur_Basni_I (Rajasthan)',
 'Piparcity_BsstdDPP_D (Rajasthan)',
 nan,
 'Jaipur_Hub (Rajasthan)',
 'Hyderabad_Shamshbd_P (Telangana)',
 'Kanpur_Central_H_6 (Uttar Pradesh)'])
```

[19]:
```
# We need to deal with the missing values
# From the looks of it, the source name follows the format  city_xyz (state)
# We can extract city and state from this


df['source_city'] = df['source_name'].str.split(" ", n=1,expand=True)[0].str.
↪split("_", n=1, expand=True)[0]


df['source_state'] = df['source_name'].str.split(" ", n=1, expand=True)[1].str.
↪replace("(","").str.replace(")","")



df['destination_city'] = df['destination_name'].str.split(" ",␣
↪n=1,expand=True)[0].str.split("_", n=1, expand=True)[0]


df['destination_state'] = df['destination_name'].str.split(" ", n=1,␣
↪expand=True)[1].str.replace("(","").str.replace(")","")

df.head(20)
```

[19]:
```
       data          trip_creation_time  \
0   training 2018-09-20 02:35:36.476840
1   training 2018-09-20 02:35:36.476840
2   training 2018-09-20 02:35:36.476840
3   training 2018-09-20 02:35:36.476840
```

```
4   training 2018-09-20 02:35:36.476840
5   training 2018-09-20 02:35:36.476840
6   training 2018-09-20 02:35:36.476840
7   training 2018-09-20 02:35:36.476840
8   training 2018-09-20 02:35:36.476840
9   training 2018-09-20 02:35:36.476840
10  training 2018-09-23 06:42:06.021680
11  training 2018-09-23 06:42:06.021680
12  training 2018-09-23 06:42:06.021680
13  training 2018-09-23 06:42:06.021680
14  training 2018-09-23 06:42:06.021680
15  training 2018-09-14 15:42:46.437249
16  training 2018-09-14 15:42:46.437249
17  training 2018-09-13 20:44:19.424489
18  training 2018-09-13 20:44:19.424489
19  training 2018-09-13 20:44:19.424489

                                 route_schedule_uuid route_type  \
0   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
5   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
6   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
7   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
8   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
9   thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
10  thanos::sroute:ff52ef7a-4d0d-4063-9bfe-cc21172…        FTL
11  thanos::sroute:ff52ef7a-4d0d-4063-9bfe-cc21172…        FTL
12  thanos::sroute:ff52ef7a-4d0d-4063-9bfe-cc21172…        FTL
13  thanos::sroute:ff52ef7a-4d0d-4063-9bfe-cc21172…        FTL
14  thanos::sroute:ff52ef7a-4d0d-4063-9bfe-cc21172…        FTL
15  thanos::sroute:a16bfa03-3462-4bce-9c82-5784c7d…    Carting
16  thanos::sroute:a16bfa03-3462-4bce-9c82-5784c7d…    Carting
17  thanos::sroute:76951383-1608-44e4-a284-46d92e8…        FTL
18  thanos::sroute:76951383-1608-44e4-a284-46d92e8…        FTL
19  thanos::sroute:76951383-1608-44e4-a284-46d92e8…        FTL

                trip_uuid source_center                     source_name  \
0   trip-153741093647649320  IND388121AAA      Anand_VUNagar_DC (Gujarat)
1   trip-153741093647649320  IND388121AAA      Anand_VUNagar_DC (Gujarat)
2   trip-153741093647649320  IND388121AAA      Anand_VUNagar_DC (Gujarat)
3   trip-153741093647649320  IND388121AAA      Anand_VUNagar_DC (Gujarat)
4   trip-153741093647649320  IND388121AAA      Anand_VUNagar_DC (Gujarat)
5   trip-153741093647649320  IND388620AAB   Khambhat_MotvdDPP_D (Gujarat)
6   trip-153741093647649320  IND388620AAB   Khambhat_MotvdDPP_D (Gujarat)
```

```
7   trip-153741093647649320  IND388620AAB      Khambhat_MotvdDPP_D (Gujarat)
8   trip-153741093647649320  IND388620AAB      Khambhat_MotvdDPP_D (Gujarat)
9   trip-153741093647649320  IND388620AAB      Khambhat_MotvdDPP_D (Gujarat)
10  trip-153768492602129387  IND421302AAG  Bhiwandi_Mankoli_HB (Maharashtra)
11  trip-153768492602129387  IND421302AAG  Bhiwandi_Mankoli_HB (Maharashtra)
12  trip-153768492602129387  IND421302AAG  Bhiwandi_Mankoli_HB (Maharashtra)
13  trip-153768492602129387  IND421302AAG  Bhiwandi_Mankoli_HB (Maharashtra)
14  trip-153768492602129387  IND421302AAG  Bhiwandi_Mankoli_HB (Maharashtra)
15  trip-153693976643699843  IND400011AAA        LowerParel_CP (Maharashtra)
16  trip-153693976643699843  IND400011AAA        LowerParel_CP (Maharashtra)
17  trip-153687145942424248  IND562132AAA   Bangalore_Nelmngla_H (Karnataka)
18  trip-153687145942424248  IND562132AAA   Bangalore_Nelmngla_H (Karnataka)
19  trip-153687145942424248  IND560099AAB  Bengaluru_Bomsndra_HB (Karnataka)

   destination_center                   destination_name  \
0        IND388620AAB      Khambhat_MotvdDPP_D (Gujarat)
1        IND388620AAB      Khambhat_MotvdDPP_D (Gujarat)
2        IND388620AAB      Khambhat_MotvdDPP_D (Gujarat)
3        IND388620AAB      Khambhat_MotvdDPP_D (Gujarat)
4        IND388620AAB      Khambhat_MotvdDPP_D (Gujarat)
5        IND388320AAA         Anand_Vaghasi_IP (Gujarat)
6        IND388320AAA         Anand_Vaghasi_IP (Gujarat)
7        IND388320AAA         Anand_Vaghasi_IP (Gujarat)
8        IND388320AAA         Anand_Vaghasi_IP (Gujarat)
9        IND388320AAA         Anand_Vaghasi_IP (Gujarat)
10       IND411033AAA      Pune_Tathawde_H (Maharashtra)
11       IND411033AAA      Pune_Tathawde_H (Maharashtra)
12       IND411033AAA      Pune_Tathawde_H (Maharashtra)
13       IND411033AAA      Pune_Tathawde_H (Maharashtra)
14       IND411033AAA      Pune_Tathawde_H (Maharashtra)
15       IND400072AAD   Mumbai_Chndivli_PC (Maharashtra)
16       IND400072AAD   Mumbai_Chndivli_PC (Maharashtra)
17       IND560099AAB  Bengaluru_Bomsndra_HB (Karnataka)
18       IND560099AAB  Bengaluru_Bomsndra_HB (Karnataka)
19       IND683511AAA           Aluva_Peedika_H (Kerala)

                 od_start_time                od_end_time  \
0   2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
1   2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
2   2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
3   2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
4   2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
5   2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
6   2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
7   2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
8   2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
9   2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
```

```
10 2018-09-23 06:42:06.021680 2018-09-23 11:44:28.365845
11 2018-09-23 06:42:06.021680 2018-09-23 11:44:28.365845
12 2018-09-23 06:42:06.021680 2018-09-23 11:44:28.365845
13 2018-09-23 06:42:06.021680 2018-09-23 11:44:28.365845
14 2018-09-23 06:42:06.021680 2018-09-23 11:44:28.365845
15 2018-09-14 15:42:46.437249 2018-09-14 17:31:45.368791
16 2018-09-14 15:42:46.437249 2018-09-14 17:31:45.368791
17 2018-09-13 20:44:19.424489 2018-09-13 23:59:56.061158
18 2018-09-13 20:44:19.424489 2018-09-13 23:59:56.061158
19 2018-09-13 23:59:56.061158 2018-09-14 13:55:58.765334

    start_scan_to_end_scan  is_cutoff  cutoff_factor  \
0                     86.0       True              9
1                     86.0       True             18
2                     86.0       True             27
3                     86.0       True             36
4                     86.0      False             39
5                    109.0       True              9
6                    109.0       True             18
7                    109.0       True             27
8                    109.0       True             36
9                    109.0      False             43
10                   302.0       True             22
11                   302.0       True             44
12                   302.0       True             66
13                   302.0       True             88
14                   302.0      False            100
15                   108.0       True              9
16                   108.0      False             16
17                   195.0       True             22
18                   195.0      False             39
19                   836.0       True             22

              cutoff_timestamp  actual_distance_to_destination  actual_time  \
0          2018-09-20 04:27:55                       10.435660         14.0
1          2018-09-20 04:17:55                       18.936842         24.0
2   2018-09-20 04:01:19.505586                       27.637279         40.0
3          2018-09-20 03:39:57                       36.118028         62.0
4          2018-09-20 03:33:55                       39.386040         68.0
5          2018-09-20 06:15:58                       10.403038         15.0
6          2018-09-20 05:47:29                       18.045481         44.0
7          2018-09-20 05:25:58                       28.061896         65.0
8          2018-09-20 05:15:56                       38.939167         76.0
9          2018-09-20 04:49:20                       43.595802        102.0
10         2018-09-23 11:05:19                       23.194334         38.0
11         2018-09-23 10:27:22                       44.045659         76.0
12         2018-09-23 09:45:25                       72.849327        117.0
```

13	2018-09-23 09:21:27	88.076599	141.0
14	2018-09-23 08:39:31	100.708423	183.0
15	2018-09-14 16:29:54	9.355852	46.0
16	2018-09-14 16:15:53	16.431273	60.0
17	2018-09-13 23:25:20	23.635811	30.0
18	2018-09-13 22:47:26	39.806036	67.0
19	2018-09-14 12:45:25	24.319864	50.0

	osrm_time	osrm_distance	factor	segment_actual_time \
0	11.0	11.9653	1.272727	14.0
1	20.0	21.7243	1.200000	10.0
2	28.0	32.5395	1.428571	16.0
3	40.0	45.5620	1.550000	21.0
4	44.0	54.2181	1.545455	6.0
5	11.0	12.1171	1.363636	15.0
6	17.0	21.2890	2.588235	28.0
7	29.0	35.8252	2.241379	21.0
8	39.0	47.1900	1.948718	10.0
9	45.0	53.2334	2.266667	26.0
10	24.0	26.8622	1.583333	38.0
11	41.0	54.4326	1.853659	37.0
12	68.0	89.6680	1.720588	41.0
13	80.0	108.3939	1.762500	23.0
14	95.0	129.3519	1.926316	41.0
15	11.0	11.4344	4.181818	46.0
16	16.0	18.7941	3.750000	14.0
17	30.0	28.9765	1.000000	30.0
18	53.0	52.1256	1.264151	37.0
19	24.0	29.7046	2.083333	50.0

	segment_osrm_time	segment_osrm_distance	segment_factor	source_city \
0	11.0	11.9653	1.272727	Anand
1	9.0	9.7590	1.111111	Anand
2	7.0	10.8152	2.285714	Anand
3	12.0	13.0224	1.750000	Anand
4	5.0	3.9153	1.200000	Anand
5	11.0	12.1171	1.363636	Khambhat
6	6.0	9.1719	4.666667	Khambhat
7	11.0	14.5362	1.909091	Khambhat
8	10.0	11.3648	1.000000	Khambhat
9	6.0	6.0434	4.333333	Khambhat
10	24.0	26.8622	1.583333	Bhiwandi
11	27.0	30.1058	1.370370	Bhiwandi
12	26.0	35.2353	1.576923	Bhiwandi
13	14.0	17.2476	1.642857	Bhiwandi
14	15.0	20.9580	2.733333	Bhiwandi
15	11.0	11.4344	4.181818	LowerParel

```
16              5.0                 7.3597        2.800000  LowerParel
17             30.0                28.9765        1.000000   Bangalore
18             26.0                24.9545        1.423077   Bangalore
19             24.0                29.7046        2.083333   Bengaluru

   source_state destination_city destination_state
0       Gujarat         Khambhat           Gujarat
1       Gujarat         Khambhat           Gujarat
2       Gujarat         Khambhat           Gujarat
3       Gujarat         Khambhat           Gujarat
4       Gujarat         Khambhat           Gujarat
5       Gujarat            Anand           Gujarat
6       Gujarat            Anand           Gujarat
7       Gujarat            Anand           Gujarat
8       Gujarat            Anand           Gujarat
9       Gujarat            Anand           Gujarat
10  Maharashtra             Pune       Maharashtra
11  Maharashtra             Pune       Maharashtra
12  Maharashtra             Pune       Maharashtra
13  Maharashtra             Pune       Maharashtra
14  Maharashtra             Pune       Maharashtra
15  Maharashtra           Mumbai       Maharashtra
16  Maharashtra           Mumbai       Maharashtra
17    Karnataka        Bengaluru         Karnataka
18    Karnataka        Bengaluru         Karnataka
19    Karnataka            Aluva            Kerala
```

[20]:

```
df['source_center'].unique().tolist()[:20], df['destination_center'].unique().
 ↪tolist()[:20]
# This looks like pin codes
# Foramt followed is this IND(6 DIGIT PIN)XXX


df['source_pincode'] = df['source_center'].apply(lambda x: x[3:9])
df['destination_pincode'] = df['destination_center'].apply(lambda x: x[3:9])

df.head(10)
```

[20]:

```
       data          trip_creation_time  \
0  training 2018-09-20 02:35:36.476840
1  training 2018-09-20 02:35:36.476840
2  training 2018-09-20 02:35:36.476840
3  training 2018-09-20 02:35:36.476840
4  training 2018-09-20 02:35:36.476840
5  training 2018-09-20 02:35:36.476840
6  training 2018-09-20 02:35:36.476840
7  training 2018-09-20 02:35:36.476840
```

```
8  training 2018-09-20 02:35:36.476840
9  training 2018-09-20 02:35:36.476840

                              route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
5  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
6  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
7  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
8  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
9  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                trip_uuid source_center                 source_name  \
0  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
5  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
6  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
7  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
8  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
9  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)

  destination_center              destination_name  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
5       IND388320AAA     Anand_Vaghasi_IP (Gujarat)
6       IND388320AAA     Anand_Vaghasi_IP (Gujarat)
7       IND388320AAA     Anand_Vaghasi_IP (Gujarat)
8       IND388320AAA     Anand_Vaghasi_IP (Gujarat)
9       IND388320AAA     Anand_Vaghasi_IP (Gujarat)

               od_start_time                od_end_time  \
0 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
1 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
2 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
3 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
4 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
5 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
6 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
```

```
7 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
8 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
9 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764

   start_scan_to_end_scan  is_cutoff  cutoff_factor  \
0                    86.0       True              9
1                    86.0       True             18
2                    86.0       True             27
3                    86.0       True             36
4                    86.0      False             39
5                   109.0       True              9
6                   109.0       True             18
7                   109.0       True             27
8                   109.0       True             36
9                   109.0      False             43

             cutoff_timestamp  actual_distance_to_destination  actual_time  \
0         2018-09-20 04:27:55                       10.435660         14.0
1         2018-09-20 04:17:55                       18.936842         24.0
2  2018-09-20 04:01:19.505586                       27.637279         40.0
3         2018-09-20 03:39:57                       36.118028         62.0
4         2018-09-20 03:33:55                       39.386040         68.0
5         2018-09-20 06:15:58                       10.403038         15.0
6         2018-09-20 05:47:29                       18.045481         44.0
7         2018-09-20 05:25:58                       28.061896         65.0
8         2018-09-20 05:15:56                       38.939167         76.0
9         2018-09-20 04:49:20                       43.595802        102.0

   osrm_time  osrm_distance    factor  segment_actual_time  segment_osrm_time  \
0       11.0        11.9653  1.272727                 14.0               11.0
1       20.0        21.7243  1.200000                 10.0                9.0
2       28.0        32.5395  1.428571                 16.0                7.0
3       40.0        45.5620  1.550000                 21.0               12.0
4       44.0        54.2181  1.545455                  6.0                5.0
5       11.0        12.1171  1.363636                 15.0               11.0
6       17.0        21.2890  2.588235                 28.0                6.0
7       29.0        35.8252  2.241379                 21.0               11.0
8       39.0        47.1900  1.948718                 10.0               10.0
9       45.0        53.2334  2.266667                 26.0                6.0

   segment_osrm_distance  segment_factor source_city source_state  \
0                11.9653        1.272727       Anand      Gujarat
1                 9.7590        1.111111       Anand      Gujarat
2                10.8152        2.285714       Anand      Gujarat
3                13.0224        1.750000       Anand      Gujarat
4                 3.9153        1.200000       Anand      Gujarat
5                12.1171        1.363636    Khambhat      Gujarat
```

```
6              9.1719        4.666667    Khambhat     Gujarat
7             14.5362        1.909091    Khambhat     Gujarat
8             11.3648        1.000000    Khambhat     Gujarat
9              6.0434        4.333333    Khambhat     Gujarat

  destination_city destination_state source_pincode destination_pincode
0         Khambhat           Gujarat         388121              388620
1         Khambhat           Gujarat         388121              388620
2         Khambhat           Gujarat         388121              388620
3         Khambhat           Gujarat         388121              388620
4         Khambhat           Gujarat         388121              388620
5            Anand           Gujarat         388620              388320
6            Anand           Gujarat         388620              388320
7            Anand           Gujarat         388620              388320
8            Anand           Gujarat         388620              388320
9            Anand           Gujarat         388620              388320
```

[21]:

```
df['source_state'].unique().tolist()
# we can see some weird names like  Nagar_DC Rajasthan,Alipore_DPC West Bengal
# lets replace them with the actual names
```

[21]:

```
['Gujarat',
 'Maharashtra',
 'Karnataka',
 'Punjab',
 'Haryana',
 'Uttarakhand',
 'Tamil Nadu',
 'Rajasthan',
 nan,
 'Telangana',
 'Madhya Pradesh',
 'Uttar Pradesh',
 'Himachal Pradesh',
 'Kerala',
 'Andhra Pradesh',
 'Bihar',
 'Jharkhand',
 'Hub Maharashtra',
 'Assam',
 'West Bengal',
 'Orissa',
 'Delhi',
 'Nagar_DC Rajasthan',
 'Jammu & Kashmir',
 'Alipore_DPC West Bengal',
 'Chandigarh',
```

```
'Chhattisgarh',
'Vadgaon Sheri DPC Maharashtra',
'Goa',
'02_DPC Uttar Pradesh',
'MP Nagar Madhya Pradesh',
'Road Punjab',
'Pondicherry',
'Layout PC Karnataka',
'Mandakni Madhya Pradesh',
'Dadra and Nagar Haveli',
'DC Maharashtra',
'Arunachal Pradesh',
'Antop Hill Maharashtra',
'City Madhya Pradesh',
'Pashan DPC Maharashtra',
'Nagaland',
'Meghalaya',
'DC Rajasthan',
'West _Dc Maharashtra',
'Nagar Uttar Pradesh',
'_NAD Andhra Pradesh',
'Avenue_DPC West Bengal',
'Tripura',
'Mizoram',
'Rahatani DPC Maharashtra',
'Balaji Nagar Maharashtra',
'Goa Goa',
'Kothanur_L Karnataka',
'Mahim Maharashtra']
```

[22]:

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

```
                                "Rahatani DPC Maharashtra":"Maharashtra",
                                "Mahim Maharashtra":"Maharashtra",
                                "DC Maharashtra":"Maharashtra",
                                "_NAD Andhra Pradesh":"Andhra Pradesh",
                                                           })

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

[23]:
```
df['source_state'].unique().tolist()
```

[23]:
```
['Gujarat',
 'Maharashtra',
 'Karnataka',
 'Punjab',
 'Haryana',
 'Uttarakhand',
 'Tamil Nadu',
 'Rajasthan',
 nan,
 'Telangana',
 'Madhya Pradesh',
 'Uttar Pradesh',
 'Himachal Pradesh',
```

```
 'Kerala',
 'Andhra Pradesh',
 'Bihar',
 'Jharkhand',
 'Hub Maharashtra',
 'Assam',
 'West Bengal',
 'Orissa',
 'Delhi',
 'Jammu & Kashmir',
 'Chandigarh',
 'Chhattisgarh',
 'Goa',
 'Road Punjab',
 'Pondicherry',
 'Dadra and Nagar Haveli',
 'Arunachal Pradesh',
 'Nagaland',
 'Meghalaya',
 'Tripura',
 'Mizoram']
```

[24]:
```
# Creating feature source_location => source_city + source_state
# Creating feature destination_location => source_city + source_state


df['source_location'] = df['source_city'] + ' ' + df['source_state']
df['destination_location'] = df['destination_city'] + ' ' +␣
 ↪df['destination_state']

df.head(10)
```

[24]:
```
       data          trip_creation_time  \
0  training 2018-09-20 02:35:36.476840
1  training 2018-09-20 02:35:36.476840
2  training 2018-09-20 02:35:36.476840
3  training 2018-09-20 02:35:36.476840
4  training 2018-09-20 02:35:36.476840
5  training 2018-09-20 02:35:36.476840
6  training 2018-09-20 02:35:36.476840
7  training 2018-09-20 02:35:36.476840
8  training 2018-09-20 02:35:36.476840
9  training 2018-09-20 02:35:36.476840

                                 route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
```

```
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
5  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
6  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
7  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
8  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
9  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                trip_uuid source_center                  source_name  \
0  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA     Anand_VUNagar_DC (Gujarat)
5  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
6  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
7  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
8  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
9  trip-153741093647649320  IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)

  destination_center               destination_name  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
5       IND388320AAA     Anand_Vaghasi_IP (Gujarat)
6       IND388320AAA     Anand_Vaghasi_IP (Gujarat)
7       IND388320AAA     Anand_Vaghasi_IP (Gujarat)
8       IND388320AAA     Anand_Vaghasi_IP (Gujarat)
9       IND388320AAA     Anand_Vaghasi_IP (Gujarat)

               od_start_time                od_end_time  \
0 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
1 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
2 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
3 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
4 2018-09-20 03:21:32.418600 2018-09-20 04:47:45.236797
5 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
6 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
7 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
8 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764
9 2018-09-20 04:47:45.236797 2018-09-20 06:36:55.627764

   start_scan_to_end_scan  is_cutoff  cutoff_factor  \
0                    86.0       True              9
```

```
1                   86.0       True            18
2                   86.0       True            27
3                   86.0       True            36
4                   86.0      False            39
5                  109.0       True             9
6                  109.0       True            18
7                  109.0       True            27
8                  109.0       True            36
9                  109.0      False            43

            cutoff_timestamp  actual_distance_to_destination  actual_time  \
0        2018-09-20 04:27:55                       10.435660         14.0
1        2018-09-20 04:17:55                       18.936842         24.0
2 2018-09-20 04:01:19.505586                       27.637279         40.0
3        2018-09-20 03:39:57                       36.118028         62.0
4        2018-09-20 03:33:55                       39.386040         68.0
5        2018-09-20 06:15:58                       10.403038         15.0
6        2018-09-20 05:47:29                       18.045481         44.0
7        2018-09-20 05:25:58                       28.061896         65.0
8        2018-09-20 05:15:56                       38.939167         76.0
9        2018-09-20 04:49:20                       43.595802        102.0

  osrm_time  osrm_distance    factor  segment_actual_time  segment_osrm_time  \
0      11.0        11.9653  1.272727                 14.0               11.0
1      20.0        21.7243  1.200000                 10.0                9.0
2      28.0        32.5395  1.428571                 16.0                7.0
3      40.0        45.5620  1.550000                 21.0               12.0
4      44.0        54.2181  1.545455                  6.0                5.0
5      11.0        12.1171  1.363636                 15.0               11.0
6      17.0        21.2890  2.588235                 28.0                6.0
7      29.0        35.8252  2.241379                 21.0               11.0
8      39.0        47.1900  1.948718                 10.0               10.0
9      45.0        53.2334  2.266667                 26.0                6.0

  segment_osrm_distance  segment_factor source_city source_state  \
0               11.9653        1.272727       Anand      Gujarat
1                9.7590        1.111111       Anand      Gujarat
2               10.8152        2.285714       Anand      Gujarat
3               13.0224        1.750000       Anand      Gujarat
4                3.9153        1.200000       Anand      Gujarat
5               12.1171        1.363636    Khambhat      Gujarat
6                9.1719        4.666667    Khambhat      Gujarat
7               14.5362        1.909091    Khambhat      Gujarat
8               11.3648        1.000000    Khambhat      Gujarat
9                6.0434        4.333333    Khambhat      Gujarat

 destination_city destination_state source_pincode destination_pincode  \
```

```
0         Khambhat           Gujarat        388121              388620
1         Khambhat           Gujarat        388121              388620
2         Khambhat           Gujarat        388121              388620
3         Khambhat           Gujarat        388121              388620
4         Khambhat           Gujarat        388121              388620
5            Anand           Gujarat        388620              388320
6            Anand           Gujarat        388620              388320
7            Anand           Gujarat        388620              388320
8            Anand           Gujarat        388620              388320
9            Anand           Gujarat        388620              388320

  source_location destination_location
0   Anand Gujarat     Khambhat Gujarat
1   Anand Gujarat     Khambhat Gujarat
2   Anand Gujarat     Khambhat Gujarat
3   Anand Gujarat     Khambhat Gujarat
4   Anand Gujarat     Khambhat Gujarat
5 Khambhat Gujarat       Anand Gujarat
6 Khambhat Gujarat       Anand Gujarat
7 Khambhat Gujarat       Anand Gujarat
8 Khambhat Gujarat       Anand Gujarat
9 Khambhat Gujarat       Anand Gujarat
```

[25]:
```python
# We can drop off the centers and the name columns for source and destination

df2 = df.copy()

df2.drop(
    ['source_center',"source_name","destination_center","destination_name"],
    axis = 1,
    inplace=True
)

df2.head(10)
```

[25]:
```
       data          trip_creation_time  \
0  training 2018-09-20 02:35:36.476840
1  training 2018-09-20 02:35:36.476840
2  training 2018-09-20 02:35:36.476840
3  training 2018-09-20 02:35:36.476840
4  training 2018-09-20 02:35:36.476840
5  training 2018-09-20 02:35:36.476840
6  training 2018-09-20 02:35:36.476840
7  training 2018-09-20 02:35:36.476840
8  training 2018-09-20 02:35:36.476840
9  training 2018-09-20 02:35:36.476840
```

```
                       route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
5  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
6  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
7  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
8  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
9  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                trip_uuid              od_start_time  \
0  trip-153741093647649320 2018-09-20 03:21:32.418600
1  trip-153741093647649320 2018-09-20 03:21:32.418600
2  trip-153741093647649320 2018-09-20 03:21:32.418600
3  trip-153741093647649320 2018-09-20 03:21:32.418600
4  trip-153741093647649320 2018-09-20 03:21:32.418600
5  trip-153741093647649320 2018-09-20 04:47:45.236797
6  trip-153741093647649320 2018-09-20 04:47:45.236797
7  trip-153741093647649320 2018-09-20 04:47:45.236797
8  trip-153741093647649320 2018-09-20 04:47:45.236797
9  trip-153741093647649320 2018-09-20 04:47:45.236797

                od_end_time  start_scan_to_end_scan  is_cutoff  \
0 2018-09-20 04:47:45.236797                    86.0       True
1 2018-09-20 04:47:45.236797                    86.0       True
2 2018-09-20 04:47:45.236797                    86.0       True
3 2018-09-20 04:47:45.236797                    86.0       True
4 2018-09-20 04:47:45.236797                    86.0      False
5 2018-09-20 06:36:55.627764                   109.0       True
6 2018-09-20 06:36:55.627764                   109.0       True
7 2018-09-20 06:36:55.627764                   109.0       True
8 2018-09-20 06:36:55.627764                   109.0       True
9 2018-09-20 06:36:55.627764                   109.0      False

   cutoff_factor            cutoff_timestamp  actual_distance_to_destination  \
0              9         2018-09-20 04:27:55                       10.435660
1             18         2018-09-20 04:17:55                       18.936842
2             27  2018-09-20 04:01:19.505586                       27.637279
3             36         2018-09-20 03:39:57                       36.118028
4             39         2018-09-20 03:33:55                       39.386040
5              9         2018-09-20 06:15:58                       10.403038
6             18         2018-09-20 05:47:29                       18.045481
7             27         2018-09-20 05:25:58                       28.061896
8             36         2018-09-20 05:15:56                       38.939167
9             43         2018-09-20 04:49:20                       43.595802
```

```
   actual_time  osrm_time  osrm_distance    factor  segment_actual_time  \
0         14.0       11.0        11.9653  1.272727                 14.0
1         24.0       20.0        21.7243  1.200000                 10.0
2         40.0       28.0        32.5395  1.428571                 16.0
3         62.0       40.0        45.5620  1.550000                 21.0
4         68.0       44.0        54.2181  1.545455                  6.0
5         15.0       11.0        12.1171  1.363636                 15.0
6         44.0       17.0        21.2890  2.588235                 28.0
7         65.0       29.0        35.8252  2.241379                 21.0
8         76.0       39.0        47.1900  1.948718                 10.0
9        102.0       45.0        53.2334  2.266667                 26.0

   segment_osrm_time  segment_osrm_distance  segment_factor source_city  \
0               11.0                11.9653        1.272727       Anand
1                9.0                 9.7590        1.111111       Anand
2                7.0                10.8152        2.285714       Anand
3               12.0                13.0224        1.750000       Anand
4                5.0                 3.9153        1.200000       Anand
5               11.0                12.1171        1.363636    Khambhat
6                6.0                 9.1719        4.666667    Khambhat
7               11.0                14.5362        1.909091    Khambhat
8               10.0                11.3648        1.000000    Khambhat
9                6.0                 6.0434        4.333333    Khambhat

  source_state destination_city destination_state source_pincode  \
0      Gujarat         Khambhat           Gujarat         388121
1      Gujarat         Khambhat           Gujarat         388121
2      Gujarat         Khambhat           Gujarat         388121
3      Gujarat         Khambhat           Gujarat         388121
4      Gujarat         Khambhat           Gujarat         388121
5      Gujarat            Anand           Gujarat         388620
6      Gujarat            Anand           Gujarat         388620
7      Gujarat            Anand           Gujarat         388620
8      Gujarat            Anand           Gujarat         388620
9      Gujarat            Anand           Gujarat         388620

  destination_pincode   source_location destination_location
0              388620     Anand Gujarat     Khambhat Gujarat
1              388620     Anand Gujarat     Khambhat Gujarat
2              388620     Anand Gujarat     Khambhat Gujarat
3              388620     Anand Gujarat     Khambhat Gujarat
4              388620     Anand Gujarat     Khambhat Gujarat
5              388320  Khambhat Gujarat        Anand Gujarat
6              388320  Khambhat Gujarat        Anand Gujarat
7              388320  Khambhat Gujarat        Anand Gujarat
8              388320  Khambhat Gujarat        Anand Gujarat
```

```
9               388320   Khambhat Gujarat        Anand Gujarat
```

[26]:
```
# Converting time fields to hour format for smooth and better understanding
```

[27]:
```
df2["start_scan_to_end_scan"]/60
```

[27]:
```
0         1.433333
1         1.433333
2         1.433333
3         1.433333
4         1.433333
            …
144862    7.116667
144863    7.116667
144864    7.116667
144865    7.116667
144866    7.116667
Name: start_scan_to_end_scan, Length: 144867, dtype: float64
```

[28]:
```
df2["start_scan_to_end_scan"] = df2["start_scan_to_end_scan"]/60
df2["actual_time"] = df2["actual_time"]/60
df2["osrm_time"] = df2["osrm_time"]/60
df2["segment_actual_time"] = df2["segment_actual_time"]/60
df2["segment_osrm_time"] = df2["segment_osrm_time"]/60



df2.head(10)
```

[28]:
```
       data         trip_creation_time  \
0  training 2018-09-20 02:35:36.476840
1  training 2018-09-20 02:35:36.476840
2  training 2018-09-20 02:35:36.476840
3  training 2018-09-20 02:35:36.476840
4  training 2018-09-20 02:35:36.476840
5  training 2018-09-20 02:35:36.476840
6  training 2018-09-20 02:35:36.476840
7  training 2018-09-20 02:35:36.476840
8  training 2018-09-20 02:35:36.476840
9  training 2018-09-20 02:35:36.476840

                                 route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
```

```
5  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
6  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
7  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
8  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
9  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                trip_uuid              od_start_time  \
0  trip-153741093647649320 2018-09-20 03:21:32.418600
1  trip-153741093647649320 2018-09-20 03:21:32.418600
2  trip-153741093647649320 2018-09-20 03:21:32.418600
3  trip-153741093647649320 2018-09-20 03:21:32.418600
4  trip-153741093647649320 2018-09-20 03:21:32.418600
5  trip-153741093647649320 2018-09-20 04:47:45.236797
6  trip-153741093647649320 2018-09-20 04:47:45.236797
7  trip-153741093647649320 2018-09-20 04:47:45.236797
8  trip-153741093647649320 2018-09-20 04:47:45.236797
9  trip-153741093647649320 2018-09-20 04:47:45.236797

                od_end_time  start_scan_to_end_scan  is_cutoff  \
0 2018-09-20 04:47:45.236797                1.433333       True
1 2018-09-20 04:47:45.236797                1.433333       True
2 2018-09-20 04:47:45.236797                1.433333       True
3 2018-09-20 04:47:45.236797                1.433333       True
4 2018-09-20 04:47:45.236797                1.433333      False
5 2018-09-20 06:36:55.627764                1.816667       True
6 2018-09-20 06:36:55.627764                1.816667       True
7 2018-09-20 06:36:55.627764                1.816667       True
8 2018-09-20 06:36:55.627764                1.816667       True
9 2018-09-20 06:36:55.627764                1.816667      False

   cutoff_factor            cutoff_timestamp  actual_distance_to_destination  \
0              9         2018-09-20 04:27:55                       10.435660
1             18         2018-09-20 04:17:55                       18.936842
2             27  2018-09-20 04:01:19.505586                       27.637279
3             36         2018-09-20 03:39:57                       36.118028
4             39         2018-09-20 03:33:55                       39.386040
5              9         2018-09-20 06:15:58                       10.403038
6             18         2018-09-20 05:47:29                       18.045481
7             27         2018-09-20 05:25:58                       28.061896
8             36         2018-09-20 05:15:56                       38.939167
9             43         2018-09-20 04:49:20                       43.595802

   actual_time  osrm_time  osrm_distance    factor  segment_actual_time  \
0     0.233333   0.183333        11.9653  1.272727             0.233333
1     0.400000   0.333333        21.7243  1.200000             0.166667
2     0.666667   0.466667        32.5395  1.428571             0.266667
3     1.033333   0.666667        45.5620  1.550000             0.350000
```

```
4     1.133333   0.733333         54.2181  1.545455             0.100000
5     0.250000   0.183333         12.1171  1.363636             0.250000
6     0.733333   0.283333         21.2890  2.588235             0.466667
7     1.083333   0.483333         35.8252  2.241379             0.350000
8     1.266667   0.650000         47.1900  1.948718             0.166667
9     1.700000   0.750000         53.2334  2.266667             0.433333

   segment_osrm_time  segment_osrm_distance  segment_factor source_city  \
0           0.183333                11.9653        1.272727       Anand
1           0.150000                 9.7590        1.111111       Anand
2           0.116667                10.8152        2.285714       Anand
3           0.200000                13.0224        1.750000       Anand
4           0.083333                 3.9153        1.200000       Anand
5           0.183333                12.1171        1.363636    Khambhat
6           0.100000                 9.1719        4.666667    Khambhat
7           0.183333                14.5362        1.909091    Khambhat
8           0.166667                11.3648        1.000000    Khambhat
9           0.100000                 6.0434        4.333333    Khambhat

  source_state destination_city destination_state source_pincode  \
0      Gujarat         Khambhat           Gujarat         388121
1      Gujarat         Khambhat           Gujarat         388121
2      Gujarat         Khambhat           Gujarat         388121
3      Gujarat         Khambhat           Gujarat         388121
4      Gujarat         Khambhat           Gujarat         388121
5      Gujarat            Anand           Gujarat         388620
6      Gujarat            Anand           Gujarat         388620
7      Gujarat            Anand           Gujarat         388620
8      Gujarat            Anand           Gujarat         388620
9      Gujarat            Anand           Gujarat         388620

  destination_pincode   source_location destination_location
0              388620     Anand Gujarat     Khambhat Gujarat
1              388620     Anand Gujarat     Khambhat Gujarat
2              388620     Anand Gujarat     Khambhat Gujarat
3              388620     Anand Gujarat     Khambhat Gujarat
4              388620     Anand Gujarat     Khambhat Gujarat
5              388320  Khambhat Gujarat        Anand Gujarat
6              388320  Khambhat Gujarat        Anand Gujarat
7              388320  Khambhat Gujarat        Anand Gujarat
8              388320  Khambhat Gujarat        Anand Gujarat
9              388320  Khambhat Gujarat        Anand Gujarat
```

[29]:
```
actual_time_by_trip = df2.groupby('trip_uuid')['actual_time'].sum().
 ↪reset_index()
actual_time_by_trip
```

[29]:

```
                      trip_uuid  actual_time
0      trip-153671041653548748   261.366667
1      trip-153671042288605164     6.650000
2      trip-153671043369099517  1870.416667
3      trip-153671046011330457     1.366667
4      trip-153671052974046625     9.266667
…                          …          …
14812  trip-153861095625827784     3.100000
14813  trip-153861104386292051     0.550000
14814  trip-153861106442901555     9.150000
14815  trip-153861115439069069    10.000000
14816  trip-153861118270144424     5.833333

[14817 rows x 2 columns]
```

[30]:

```python
trip_source_destination = df2.groupby(['trip_uuid', 'source_pincode',␣
↪'destination_pincode'])['actual_time'].sum().reset_index()
trip_source_destination
```

[30]:

```
                      trip_uuid source_pincode destination_pincode  actual_time
0      trip-153671041653548748         209304              000000   108.066667
1      trip-153671041653548748         462022              209304   153.300000
2      trip-153671042288605164         561203              562101     1.600000
3      trip-153671042288605164         572101              561203     5.050000
4      trip-153671043369099517         000000              160002    43.350000
…                          …            …                   …          …
26347  trip-153861115439069069         628204              627657     1.983333
26348  trip-153861115439069069         628613              627005     2.883333
26349  trip-153861115439069069         628801              628204     0.850000
26350  trip-153861118270144424         583119              583101     4.633333
26351  trip-153861118270144424         583201              583119     1.200000

[26352 rows x 4 columns]
```

[31]:

```python
segment_actual_time = df2.groupby("trip_uuid")["segment_actual_time"].sum().
↪reset_index()
segment_actual_time
```

[31]:

```
                      trip_uuid  segment_actual_time
0      trip-153671041653548748            25.800000
1      trip-153671042288605164             2.350000
2      trip-153671043369099517            55.133333
3      trip-153671046011330457             0.983333
4      trip-153671052974046625             5.666667
…                          …                  …
14812  trip-153861095625827784             1.366667
14813  trip-153861104386292051             0.350000
```

```
14814  trip-153861106442901555            4.683333
14815  trip-153861115439069069            4.300000
14816  trip-153861118270144424            4.566667

[14817 rows x 2 columns]
```

[32]:
```python
osrm_time = df2.groupby(["trip_uuid",
              "start_scan_to_end_scan"])["osrm_time"].max().reset_index().
 ↪groupby("trip_uuid")["osrm_time"].sum().reset_index()
osrm_time
```

[32]:
```
                     trip_uuid  osrm_time
0      trip-153671041653548748  12.383333
1      trip-153671042288605164   1.133333
2      trip-153671043369099517  29.016667
3      trip-153671046011330457   0.250000
4      trip-153671052974046625   1.950000
…                          …        …
14812  trip-153861095625827784   1.033333
14813  trip-153861104386292051   0.200000
14814  trip-153861106442901555   0.900000
14815  trip-153861115439069069   3.066667
14816  trip-153861118270144424   1.133333

[14817 rows x 2 columns]
```

[33]:
```python
df2["time_between_od_start_od_end"] = ((df["od_end_time"]-df["od_start_time"])/
 ↪pd.Timedelta(1,unit="hour"))

df["time_between_od_start_od_end"] = ((df["od_end_time"]-df["od_start_time"])/
 ↪pd.Timedelta(1,unit="hour"))
```

[34]:
```python
time_between_od_start_od_end = df2.
 ↪groupby("trip_uuid")["time_between_od_start_od_end"].unique().reset_index()
time_between_od_start_od_end
```

[34]:
```
                     trip_uuid  \
0      trip-153671041653548748
1      trip-153671042288605164
2      trip-153671043369099517
3      trip-153671046011330457
4      trip-153671052974046625
…                          …
14812  trip-153861095625827784
14813  trip-153861104386292051
14814  trip-153861106442901555
14815  trip-153861115439069069
```

```
14816  trip-153861118270144424

                            time_between_od_start_od_end
0                             [16.65842298, 21.0100736875]
1                [2.0463247669444447, 0.9805397955555556]
2                [51.662059856388886, 13.910648811388889]
3                                     [1.6749155866666667]
4       [2.5335485744444446, 1.3423885633333332, 8.096…
…                                                      …
14812                   [2.546464057777778, 1.7540180775]
14813                                 [1.0098420219444444]
14814                   [2.895179575833333, 4.1401515375]
14815   [1.7609491794444445, 0.7362400538888889, 1.035…
14816                   [1.1155594141666667, 4.7912334425]

[14817 rows x 2 columns]
```

[35]:

```python
time_between_od_start_od_end["time_between_od_start_od_end"] =␣
↪time_between_od_start_od_end["time_between_od_start_od_end"].apply(sum)
time_between_od_start_od_end["time_between_od_start_od_end"]
```

[35]:

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

[36]:

```python
start_scan_to_end_scan = ((df2.groupby("trip_uuid")["start_scan_to_end_scan"].
↪unique())).reset_index()
start_scan_to_end_scan
```

[36]:

```
                     trip_uuid  \
0      trip-153671041653548748
1      trip-153671042288605164
2      trip-153671043369099517
3      trip-153671046011330457
4      trip-153671052974046625
…                          …
14812  trip-153861095625827784
14813  trip-153861104386292051
```

```
14814  trip-153861106442901555
14815  trip-153861115439069069
14816  trip-153861118270144424

                                  start_scan_to_end_scan
0                                           [16.65, 21.0]
1                  [2.033333333333333, 0.9666666666666667]
2                                           [51.65, 13.9]
3                                    [1.6666666666666667]
4       [2.533333333333333, 1.3333333333333333, 8.0833…
…                                                      …
14812                          [2.533333333333333, 1.75]
14813                                               [1.0]
14814              [2.8833333333333333, 4.133333333333334]
14815  [1.75, 0.7333333333333333, 1.0333333333333334,…
14816                         [1.1, 4.783333333333333]

[14817 rows x 2 columns]
```

[37]:
```python
start_scan_to_end_scan["start_scan_to_end_scan"] = start_scan_to_end_scan["start_scan_to_end_scan"].apply(sum)
start_scan_to_end_scan["start_scan_to_end_scan"]
```

[37]:
```
0        37.650000
1         3.000000
2        65.550000
3         1.666667
4        11.950000
           …
14812     4.283333
14813     1.000000
14814     7.016667
14815     5.783333
14816     5.883333
Name: start_scan_to_end_scan, Length: 14817, dtype: float64
```

[38]:
```python
time_between_od_start_od_end["time_between_od_start_od_end"] - start_scan_to_end_scan["start_scan_to_end_scan"]
```

[38]:
```
0        0.018497
1        0.026865
2        0.022709
3        0.008249
4        0.022484
           …
14812    0.017149
14813    0.009842
```

```
14814    0.018664
14815    0.025214
14816    0.023460
Length: 14817, dtype: float64
```

[39]:

```python
actual_distance_to_destination = df2.
 ↪groupby(["trip_uuid","start_scan_to_end_scan"])["actual_distance_to_destination"].
 ↪max().reset_index().groupby("trip_uuid")["actual_distance_to_destination"].
 ↪sum().reset_index()

actual_distance_to_destination
```

[39]:

```
                      trip_uuid  actual_distance_to_destination
0      trip-153671041653548748                      824.732854
1      trip-153671042288605164                       73.186911
2      trip-153671043369099517                     1932.273969
3      trip-153671046011330457                       17.175274
4      trip-153671052974046625                      127.448500
…                           …                              …
14812  trip-153861095625827784                       57.762332
14813  trip-153861104386292051                       15.513784
14814  trip-153861106442901555                       38.684839
14815  trip-153861115439069069                      134.723836
14816  trip-153861118270144424                       66.081533

[14817 rows x 2 columns]
```

[40]:

```python
segment_osrm_distance = df2[["trip_uuid","segment_osrm_distance"]].
 ↪groupby("trip_uuid")["segment_osrm_distance"].sum().reset_index()

segment_osrm_distance
```

[40]:

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

[14817 rows x 2 columns]
```

[41]:
```
segment_osrm_distance['segment_osrm_distance'] -␣
↪actual_distance_to_destination['actual_distance_to_destination']
```

[41]:
```
0         495.740446
1          11.002489
2         612.993831
3           2.701326
4          19.343400
             …
14812       7.092768
14813       0.574516
14814      66.201761
14815      88.808564
14816      14.497167
Length: 14817, dtype: float64
```

[42]:
```
# splitting the creation time into date time (hour)
df2['trip_creation_date'] = pd.to_datetime(df2['trip_creation_time'].dt.date)
df2['trip_creation_date']
```

[42]:
```
0        2018-09-20
1        2018-09-20
2        2018-09-20
3        2018-09-20
4        2018-09-20
             …
144862   2018-09-20
144863   2018-09-20
144864   2018-09-20
144865   2018-09-20
144866   2018-09-20
Name: trip_creation_date, Length: 144867, dtype: datetime64[ns]
```

[43]:
```
df2['trip_creation_day'] = df2['trip_creation_time'].dt.day
df2['trip_creation_day'] = df2['trip_creation_day'].astype('int8')
df2['trip_creation_day'].head()
```

[43]:
```
0    20
1    20
2    20
3    20
4    20
Name: trip_creation_day, dtype: int8
```

[44]:
```
df2['trip_creation_month'] = df2['trip_creation_time'].dt.month
df2['trip_creation_month'] = df2['trip_creation_month'].astype('int8')
df2['trip_creation_month'].head()
```

[44]:
```
0    9
1    9
2    9
3    9
4    9
Name: trip_creation_month, dtype: int8
```

[45]:
```python
df2['trip_creation_year'] = df2['trip_creation_time'].dt.year
df2['trip_creation_year'] = df2['trip_creation_year'].astype('int16')
df2['trip_creation_year'].head()
```

[45]:
```
0    2018
1    2018
2    2018
3    2018
4    2018
Name: trip_creation_year, dtype: int16
```

[46]:
```python
df2['trip_creation_week'] = df2['trip_creation_time'].dt.isocalendar().week
df2['trip_creation_week'] = df2['trip_creation_week'].astype('int8')
df2['trip_creation_week'].head()
```

[46]:
```
0    38
1    38
2    38
3    38
4    38
Name: trip_creation_week, dtype: int8
```

[47]:
```python
df2['trip_creation_hour'] = df2['trip_creation_time'].dt.hour
df2['trip_creation_hour'] = df2['trip_creation_hour'].astype('int8')
df2['trip_creation_hour'].head()
```

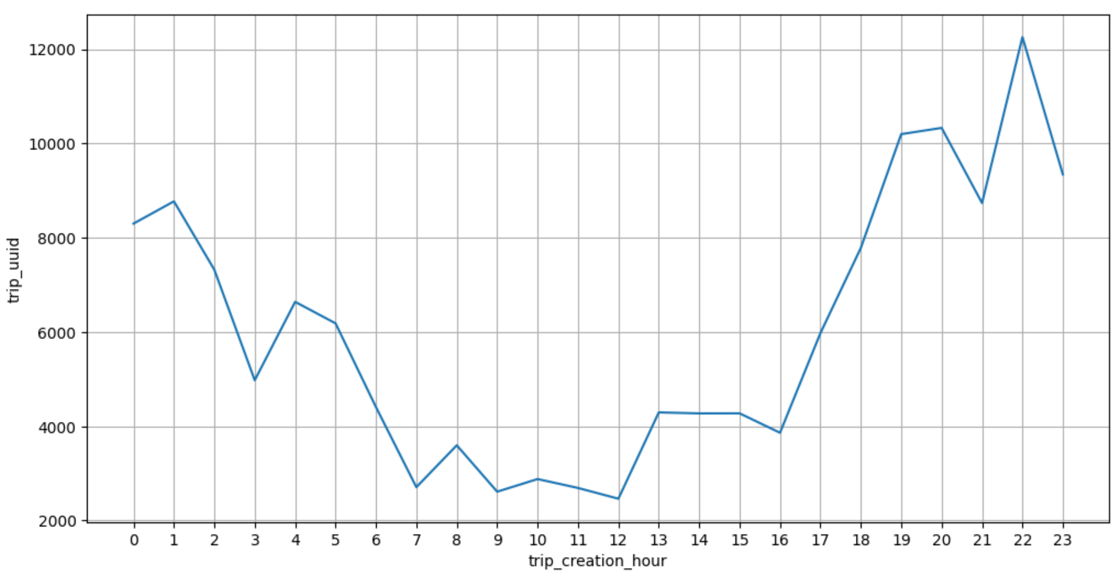
[47]:
```
0    2
1    2
2    2
3    2
4    2
Name: trip_creation_hour, dtype: int8
```

[48]:
```python
df2['od_total_time'] = df2['od_end_time'] - df2['od_start_time']
df2.drop(columns = ['od_end_time', 'od_start_time'], inplace = True)
df2['od_total_time'] = df2['od_total_time'].apply(lambda x : round(x.
 ↪total_seconds() / 60.0, 2))
df2['od_total_time'].head()
```

[48]:
```
0    86.21
1    86.21
2    86.21
3    86.21
4    86.21
Name: od_total_time, dtype: float64
```

[49]:
```
df_hour = df2.groupby(by = 'trip_creation_hour')['trip_uuid'].count().
    ↪to_frame().reset_index()
plt.figure(figsize = (12, 6))
sns.lineplot(data = df_hour,
             x = df_hour['trip_creation_hour'],
             y = df_hour['trip_uuid'],
             markers = '*')
plt.xticks(np.arange(0,24))
plt.grid('both')
plt.plot()
```

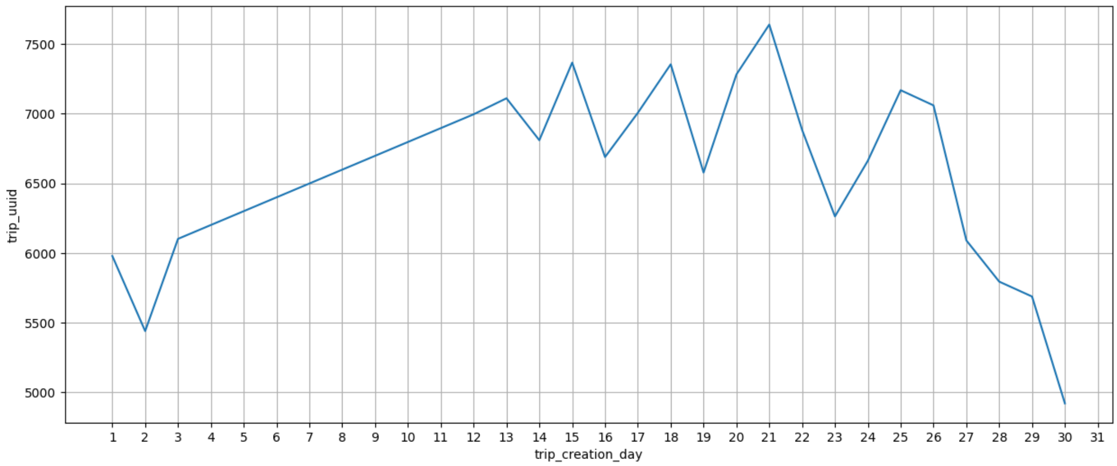
[49]: []



[50]:
```
df_day = df2.groupby(by = 'trip_creation_day')['trip_uuid'].count().to_frame().
    ↪reset_index()
plt.figure(figsize = (15, 6))
sns.lineplot(data = df_day,
             x = df_day['trip_creation_day'],
             y = df_day['trip_uuid'],
             markers = 'o')
plt.xticks(np.arange(1, 32))
```

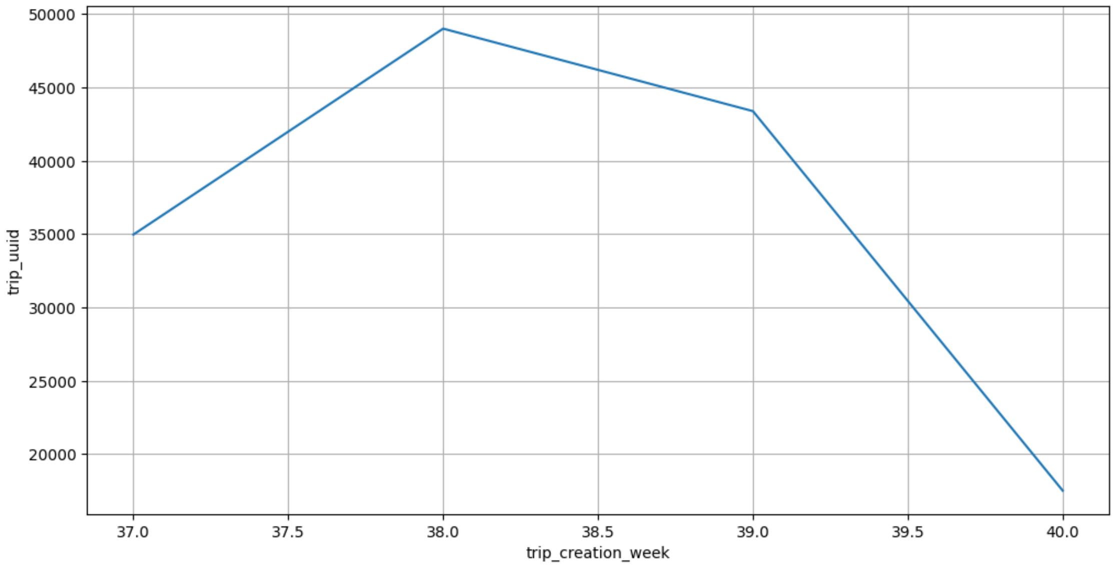
```
plt.grid('both')
plt.plot()
```

[50]: []



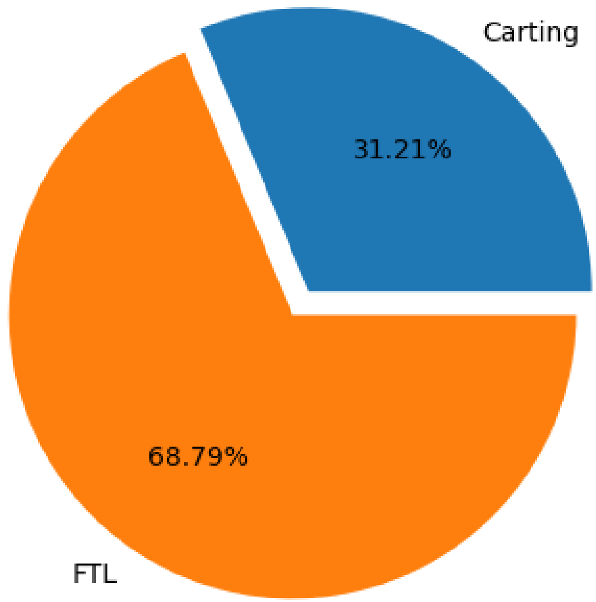
[51]:
```
df_week = df2.groupby(by = 'trip_creation_week')['trip_uuid'].count().
  ↪to_frame().reset_index()
plt.figure(figsize = (12, 6))
sns.lineplot(data = df_week,
             x = df_week['trip_creation_week'],
             y = df_week['trip_uuid'],
             markers = 'o')
plt.grid('both')
plt.plot()
```

[51]: []



[52]:
```
df_route = df2.groupby(by = 'route_type')['trip_uuid'].count().to_frame().
  ↪reset_index()
df_route['perc'] = np.round(df_route['trip_uuid'] * 100/ df_route['trip_uuid'].
  ↪sum(), 2)
plt.pie(x = df_route['trip_uuid'],
        labels = ['Carting', 'FTL'],
        explode = [0, 0.1],
        autopct = '%.2f%%')
plt.plot()
```

[52]:
```
[]
```



[53]:
```
df_source_state = df2.groupby(by = 'source_state')['trip_uuid'].count().
    ↪to_frame().reset_index()
df_source_state['perc'] = np.round(df_source_state['trip_uuid'] * 100/␣
    ↪df_source_state['trip_uuid'].sum(), 2)
df_source_state = df_source_state.sort_values(by = 'trip_uuid', ascending =␣
    ↪False)
df_source_state
```

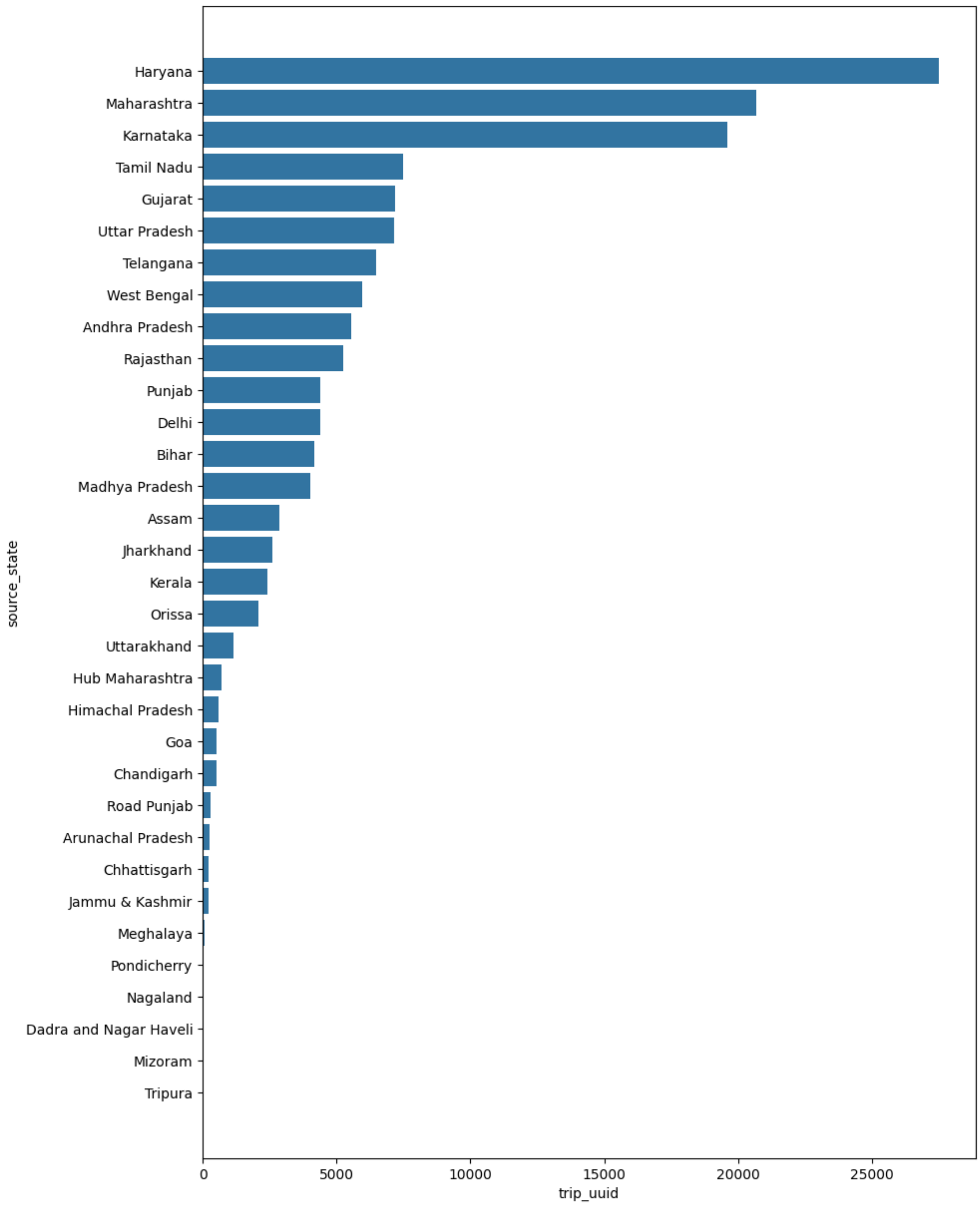
[53]:

	source_state	trip_uuid	perc
10	Haryana	27499	19.02
18	Maharashtra	20692	14.31
15	Karnataka	19578	13.54
27	Tamil Nadu	7494	5.18
9	Gujarat	7202	4.98
30	Uttar Pradesh	7137	4.94
28	Telangana	6496	4.49
32	West Bengal	5963	4.12
0	Andhra Pradesh	5539	3.83
25	Rajasthan	5267	3.64
24	Punjab	4410	3.05
7	Delhi	4398	3.04
3	Bihar	4190	2.90
17	Madhya Pradesh	4021	2.78

```
2                     Assam     2875   1.99
14                Jharkhand     2597   1.80
16                   Kerala     2413   1.67
22                   Orissa     2094   1.45
31              Uttarakhand     1162   0.80
12         Hub Maharashtra       709   0.49
11         Himachal Pradesh      587   0.41
8                       Goa      514   0.36
4                Chandigarh      507   0.35
26              Road Punjab      294   0.20
1         Arunachal Pradesh      245   0.17
5             Chhattisgarh       229   0.16
13          Jammu & Kashmir      226   0.16
19                Meghalaya       86   0.06
23              Pondicherry       49   0.03
21                 Nagaland       40   0.03
6    Dadra and Nagar Haveli       30   0.02
20                  Mizoram       26   0.02
29                  Tripura        5   0.00
```

[54]:
```
plt.figure(figsize = (10, 15))
sns.barplot(data = df_source_state,
            x = df_source_state['trip_uuid'],
            y = df_source_state['source_state'])
plt.plot()
```

[54]: []



[55]:
```
df_source_city = df2.groupby(by = 'source_city')['trip_uuid'].count().
    ↪to_frame().reset_index()
df_source_city['perc'] = np.round(df_source_city['trip_uuid'] * 100/␣
    ↪df_source_city['trip_uuid'].sum(), 2)
df_source_city = df_source_city.sort_values(by = 'trip_uuid', ascending =␣
    ↪False)[:30]
```

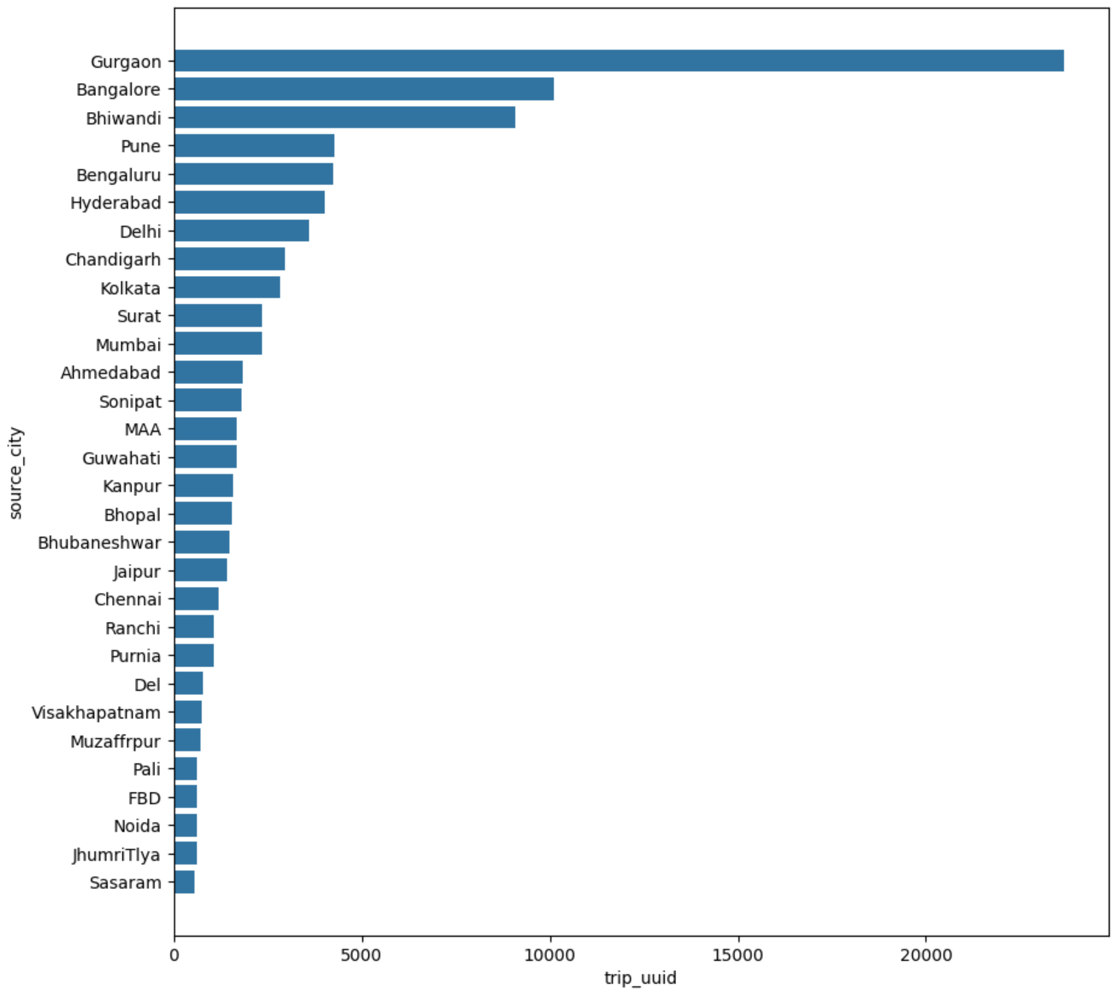
```
df_source_city
```

[55]:

	source_city	trip_uuid	perc
420	Gurgaon	23665	16.37
102	Bangalore	10104	6.99
171	Bhiwandi	9088	6.29
946	Pune	4275	2.96
139	Bengaluru	4237	2.93
465	Hyderabad	4023	2.78
299	Delhi	3587	2.48
233	Chandigarh	2957	2.05
626	Kolkata	2844	1.97
1134	Surat	2362	1.63
777	Mumbai	2343	1.62
7	Ahmedabad	1850	1.28
1115	Sonipat	1795	1.24
678	MAA	1678	1.16
423	Guwahati	1678	1.16
558	Kanpur	1581	1.09
173	Bhopal	1562	1.08
174	Bhubaneshwar	1501	1.04
479	Jaipur	1412	0.98
239	Chennai	1188	0.82
983	Ranchi	1074	0.74
949	Purnia	1053	0.73
298	Del	766	0.53
1226	Visakhapatnam	748	0.52
788	Muzaffrpur	709	0.49
875	Pali	622	0.43
352	FBD	615	0.43
851	Noida	614	0.42
515	JhumriTlya	614	0.42
1046	Sasaram	541	0.37

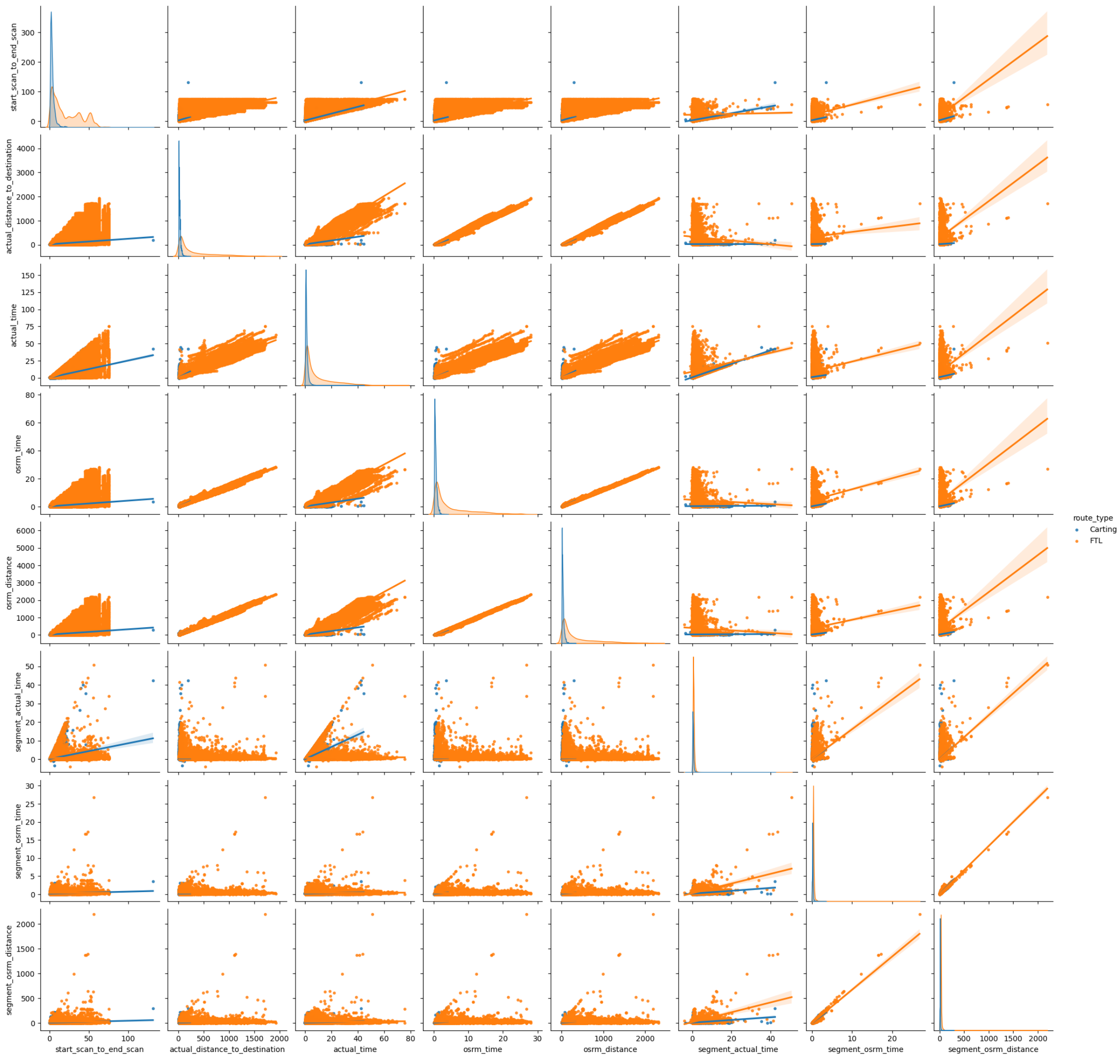
[56]:
```
plt.figure(figsize = (10, 10))
sns.barplot(data = df_source_city,
            x = df_source_city['trip_uuid'],
            y = df_source_city['source_city'])
plt.plot()
```

[56]: []



[57]:
```
numerical_columns = [ 'start_scan_to_end_scan',␣
↪'actual_distance_to_destination',
                     'actual_time', 'osrm_time', 'osrm_distance',␣
↪'segment_actual_time',
                     'segment_osrm_time', 'segment_osrm_distance']
sns.pairplot(data = df2,
             vars = numerical_columns,
             kind = 'reg',
             hue = 'route_type',
             markers = '.')
plt.plot()
```

[57]: []



[58]:
```python
df_corr = df2[numerical_columns].corr()
df_corr
```

[58]:
```
                                start_scan_to_end_scan  \
start_scan_to_end_scan                        1.000000
actual_distance_to_destination                0.785006
actual_time                                   0.785937
osrm_time                                     0.785298
osrm_distance                                 0.784138
segment_actual_time                           0.093301
segment_osrm_time                             0.219848
segment_osrm_distance                         0.306983

                                actual_distance_to_destination  actual_time  \
start_scan_to_end_scan                                0.785006     0.785937
```

```
actual_distance_to_destination                        1.000000     0.978659
actual_time                                           0.978659     1.000000
osrm_time                                             0.995872     0.977998
osrm_distance                                         0.997149     0.979399
segment_actual_time                                   0.045241     0.124411
segment_osrm_time                                     0.158832     0.171465
segment_osrm_distance                                 0.232119     0.242282

                                osrm_time  osrm_distance  segment_actual_time  \
start_scan_to_end_scan           0.785298       0.784138             0.093301
actual_distance_to_destination   0.995872       0.997149             0.045241
actual_time                      0.977998       0.979399             0.124411
osrm_time                        1.000000       0.999119             0.049892
osrm_distance                    0.999119       1.000000             0.048705
segment_actual_time              0.049892       0.048705             1.000000
segment_osrm_time                0.177066       0.169151             0.433422
segment_osrm_distance            0.242282       0.239669             0.448959

                                segment_osrm_time  segment_osrm_distance
start_scan_to_end_scan                   0.219848               0.306983
actual_distance_to_destination           0.158832               0.232119
actual_time                              0.171465               0.242282
osrm_time                                0.177066               0.242282
osrm_distance                            0.169151               0.239669
segment_actual_time                      0.433422               0.448959
segment_osrm_time                        1.000000               0.948523
segment_osrm_distance                    0.948523               1.000000
```

[59]:

```python
plt.figure(figsize = (15, 10))
sns.heatmap(data = df_corr, vmin = -1, vmax = 1, annot = True)
plt.plot()
```

[59]: []

[60]:
```
df['od_total_time'] = df['od_end_time'] - df['od_start_time']
df.drop(columns = ['od_end_time', 'od_start_time'], inplace = True)
df['od_total_time'] = df['od_total_time'].apply(lambda x : round(x.
 ↪total_seconds() / 60.0, 2))
df['od_total_time'].head()
```

[60]:
```
0    86.21
1    86.21
2    86.21
3    86.21
4    86.21
Name: od_total_time, dtype: float64
```

[61]:
```
df3 = df.groupby(by = 'trip_uuid', as_index = False).agg({'source_center' :␣
 ↪'first',
                                                            'destination_center'␣
 ↪: 'last',
                                                            'data' : 'first',
                                                            'route_type' :␣
 ↪'first',
```

```
                                                                   'trip_creation_time'␣
↪: 'first',
                                                                   'source_name' :␣
↪'first',
                                                                   'destination_name' :␣
↪'last',
                                                                   'od_total_time' :␣
↪'sum',
                                                                  ␣
↪'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'})
df3
```

[61]:

```
                     trip_uuid source_center destination_center      data  \
0      trip-153671041653548748  IND462022AAA       IND000000ACB  training
1      trip-153671042288605164  IND572101AAA       IND562101AAA  training
2      trip-153671043369099517  IND562132AAA       IND160002AAC  training
3      trip-153671046011330457  IND400072AAB       IND401104AAA  training
4      trip-153671052974046625  IND583101AAA       IND583101AAA  training
…                          …           …                  …         …
14812  trip-153861095625827784  IND160002AAC       IND160002AAC      test
14813  trip-153861104386292051  IND121004AAB       IND121004AAA      test
14814  trip-153861106442901555  IND209304AAA       IND209304AAA      test
14815  trip-153861115439069069  IND627005AAA       IND627005AAA      test
14816  trip-153861118270144424  IND583201AAA       IND583101AAA      test

      route_type         trip_creation_time  \
0            FTL 2018-09-12 00:00:16.535741
1        Carting 2018-09-12 00:00:22.886430
2            FTL 2018-09-12 00:00:33.691250
3        Carting 2018-09-12 00:01:00.113710
4            FTL 2018-09-12 00:02:09.740725
…          …                          …
14812    Carting 2018-10-03 23:55:56.258533
```

```
14813    Carting 2018-10-03 23:57:23.863155
14814    Carting 2018-10-03 23:57:44.429324
14815    Carting 2018-10-03 23:59:14.390954
14816        FTL 2018-10-03 23:59:42.701692

                               source_name  \
0        Bhopal_Trnsport_H (Madhya Pradesh)
1             Tumkur_Veersagr_I (Karnataka)
2          Bangalore_Nelmngla_H (Karnataka)
3                 Mumbai Hub (Maharashtra)
4                   Bellary_Dc (Karnataka)
…                                        …
14812        Chandigarh_Mehmdpur_H (Punjab)
14813          FBD_Balabhgarh_DPC (Haryana)
14814    Kanpur_Central_H_6 (Uttar Pradesh)
14815  Tirunelveli_VdkkuSrt_I (Tamil Nadu)
14816                   Hospet (Karnataka)

                          destination_name  od_total_time  \
0             Gurgaon_Bilaspur_HB (Haryana)       43680.51
1          Chikblapur_ShntiSgr_D (Karnataka)         913.17
2            Chandigarh_Mehmdpur_H (Punjab)      248694.12
3            Mumbai_MiraRd_IP (Maharashtra)         200.98
4                   Bellary_Dc (Karnataka)         1588.69
…                                        …            …
14812        Chandigarh_Mehmdpur_H (Punjab)         879.33
14813        Faridabad_Blbgarh_DC (Haryana)         121.18
14814    Kanpur_Central_H_6 (Uttar Pradesh)        1266.36
14815  Tirunelveli_VdkkuSrt_I (Tamil Nadu)        1320.44
14816                   Bellary_Dc (Karnataka)         708.80

       start_scan_to_end_scan  actual_distance_to_destination  actual_time  \
0                     43659.0                     8860.812105      15682.0
1                       906.0                      240.208306        399.0
2                    248631.0                    68163.502238     112225.0
3                       200.0                       28.529648         82.0
4                      1586.0                      239.007304        556.0
…                           …                               …            …
14812                   876.0                      141.057373        186.0
14813                   120.0                       25.130640         33.0
14814                  1263.0                       93.743842        549.0
14815                  1315.0                      355.281673        600.0
14816                   706.0                      110.239116        350.0

       osrm_time  osrm_distance  segment_actual_time  segment_osrm_time  \
0         7787.0     10577.7647               1548.0             1008.0
1          210.0       269.4308                141.0               65.0
```

```
2        65768.0     89447.2488              3308.0             1941.0
3           24.0        31.6475                59.0               16.0
4          207.0       266.2914               340.0              115.0
…            …           …                …                 …
14812      148.0       162.9473                82.0               62.0
14813       19.0        26.5333                21.0               11.0
14814      134.0       162.8499               281.0               88.0
14815      446.0       449.5383               258.0              221.0
14816      106.0       127.8020               274.0               67.0

       segment_osrm_distance
0                  1320.4733
1                    84.1894
2                  2545.2678
3                    19.8766
4                   146.7919
…                        …
14812                64.8551
14813                16.0883
14814               104.8866
14815               223.5324
14816                80.5787

[14817 rows x 17 columns]
```

[62]:

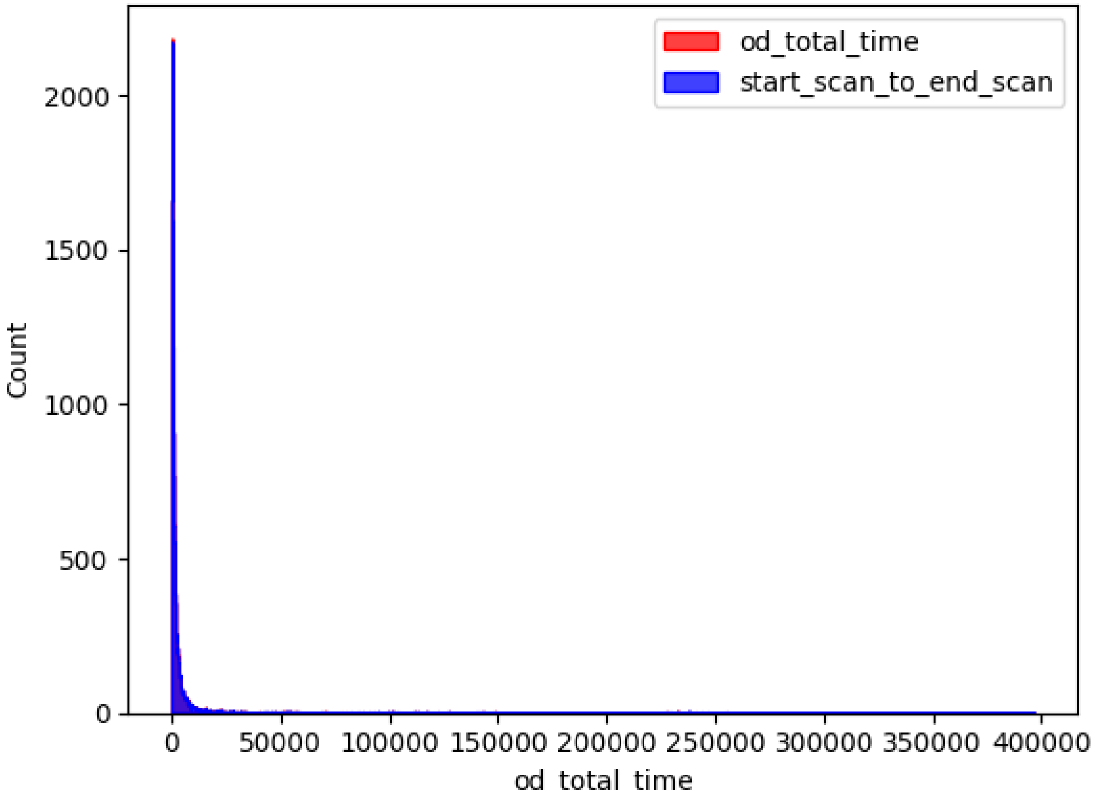
```python
from scipy import stats

# Compare the difference between od_total_time and start_scan_to_end_scan. Do
 ↪hypothesis testing/ Visual analysis to check.

# Null Hypothesis ( H0 ) - od_total_time (Total Trip Time) and
 ↪start_scan_to_end_scan (Expected total trip time) are same.

# Alternate Hypothesis ( HA ) - od_total_time (Total Trip Time) and
 ↪start_scan_to_end_scan (Expected total trip time) are different.

df3[['od_total_time', 'start_scan_to_end_scan']].describe()
```

[62]:

	od_total_time	start_scan_to_end_scan
count	14817.000000	14817.000000
mean	9403.194234	9398.345482
std	33707.727320	33701.706672
min	26.500000	26.000000
25%	409.580000	408.000000
50%	987.520000	985.000000
75%	2832.710000	2826.000000
max	396834.500000	396800.000000

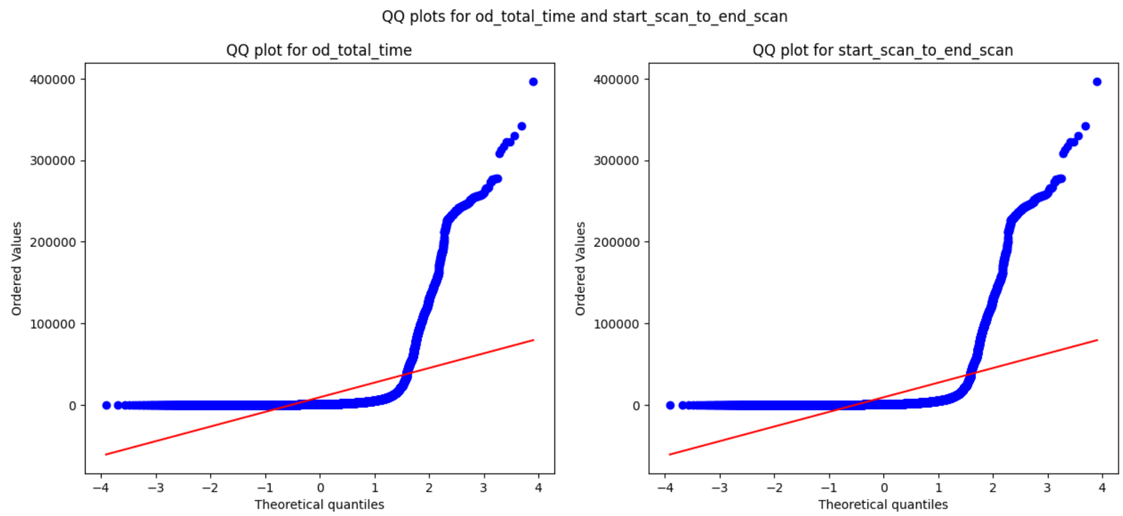
[63]:
```python
# Check for normality
sns.histplot(df3['od_total_time'], element = 'step', color = 'red')
sns.histplot(df3['start_scan_to_end_scan'], element = 'step', color = 'blue')
plt.legend(['od_total_time', 'start_scan_to_end_scan'])
plt.plot()
```

[63]: []



[64]:
```python
# Use qq plot
plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.suptitle('QQ plots for od_total_time and start_scan_to_end_scan')
stats.probplot(df3['od_total_time'], plot = plt, dist = 'norm')
plt.title('QQ plot for od_total_time')
plt.subplot(1, 2, 2)
stats.probplot(df3['start_scan_to_end_scan'], plot = plt, dist = 'norm')
plt.title('QQ plot for start_scan_to_end_scan')
plt.plot()
```

[64]: []



[65]:
```
# using shapiro test
test_stat, p_value = stats.shapiro(df3['od_total_time'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 4.142430266527794e-88
The sample does not follow normal distribution
```

[66]:
```
test_stat, p_value = stats.shapiro(df3['start_scan_to_end_scan'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 5.737245348389836e-88
The sample does not follow normal distribution
```

[67]:
```
# df['od_total_time'] = df['od_end_time'] - df['od_start_time']
# df.drop(columns = ['od_end_time', 'od_start_time'], inplace = True)
# df['od_total_time'] = df['od_total_time'].apply(lambda x : round(x.
↪total_seconds() / 60.0, 2))
# df['od_total_time'].head()
```

[68]:
```
# Null Hypothesis(H0) - Homogenous Variance

# Alternate Hypothesis(HA) - Non Homogenous Variance
```

```
test_stat, p_value = stats.levene(df3['od_total_time'],␣
 ↪df3['start_scan_to_end_scan'])
print('p-value', p_value)
if p_value < 0.05:
    print('The samples do not have  Homogenous Variance')
else:
    print('The samples have Homogenous Variance ')
```

```
p-value 0.99354015976249
The samples have Homogenous Variance
```

[69]:
```
# Since the samples are not normally distributed, T-Test cannot be applied here,
#  we can perform its non parametric equivalent test i.e., Mann-Whitney U rank␣
 ↪test for two independent samples.

test_stat, p_value = stats.mannwhitneyu(df3['od_total_time'],␣
 ↪df3['start_scan_to_end_scan'])
print('P-value :',p_value)
```

```
P-value : 0.8411023369352699
```

[70]:
```
# Since p-value > alpha therfore it can be concluded that od_total_time and␣
 ↪start_scan_to_end_scan are similar.
```

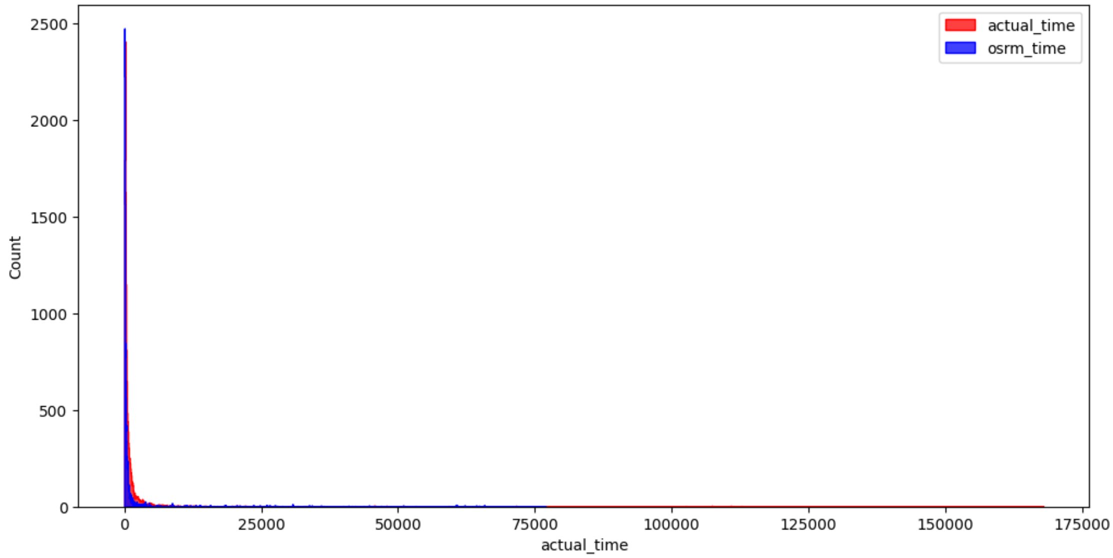
[71]:
```
# Do hypothesis testing / visual analysis between actual_time aggregated value␣
 ↪and OSRM time aggregated value (aggregated values are the values you'll get␣
 ↪after merging the rows on the basis of trip_uuid)

df3[['actual_time', 'osrm_time']].describe()
```

[71]:

	actual_time	osrm_time
count	14817.000000	14817.000000
mean	4076.333941	2091.007289
std	15216.870041	7956.882351
min	9.000000	6.000000
25%	142.000000	62.000000
50%	348.000000	167.000000
75%	1063.000000	516.000000
max	167920.000000	76953.000000

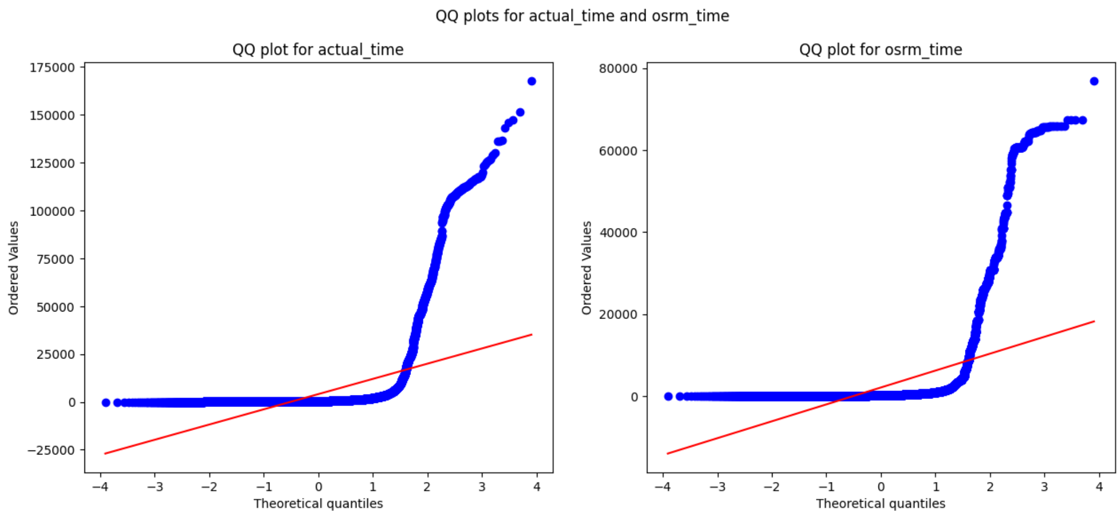
[72]:
```
plt.figure(figsize = (12, 6))
sns.histplot(df3['actual_time'], element = 'step', color = 'red')
sns.histplot(df3['osrm_time'], element = 'step', color = 'blue')
plt.legend(['actual_time', 'osrm_time'])
plt.plot()
```

[72]: []



[73]:
```python
# check normality
# qq plot
plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.suptitle('QQ plots for actual_time and osrm_time')
stats.probplot(df3['actual_time'], plot = plt, dist = 'norm')
plt.title('QQ plot for actual_time')
plt.subplot(1, 2, 2)
stats.probplot(df3['osrm_time'], plot = plt, dist = 'norm')
plt.title('QQ plot for osrm_time')
plt.plot()
```

[73]: []



[74]:
```
# It can be seen from the above plots that the samples do not come from normal␣
↪distribution.
# H0: The sample follows normal distribution
# H1: The sample does not follow normal distribution

# shapiro test for normality
test_stat, p_value = stats.shapiro(df3['actual_time'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 1.506608612852733e-88
The sample does not follow normal distribution
```

[75]:
```
test_stat, p_value = stats.shapiro(df3['osrm_time'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 8.836808629246943e-89
The sample does not follow normal distribution
```

[76]:
```
# levens test to check for variance
# Null Hypothesis(H0) - Homogenous Variance

# Alternate Hypothesis(HA) - Non Homogenous Variance
```

```
test_stat, p_value = stats.levene(df3['actual_time'], df3['osrm_time'])
print('p-value', p_value)
if p_value < 0.05:
    print('The samples do not have  Homogenous Variance')
else:
    print('The samples have Homogenous Variance ')
```

```
p-value 1.8781986379569502e-41
The samples do not have  Homogenous Variance
```

[77]:
```
# Since the samples do not follow any of the assumptions T-Test cannot be␣
 ↪applied here, we can perform its non parametric equivalent test i.e.,␣
 ↪Mann-Whitney U rank test for two independent samples.

test_stat, p_value = stats.mannwhitneyu(df3['actual_time'], df3['osrm_time'])
print('p-value', p_value)
if p_value < 0.05:
    print('The samples are not similar')
else:
    print('The samples are similar ')
```

```
p-value 0.0
The samples are not similar
```

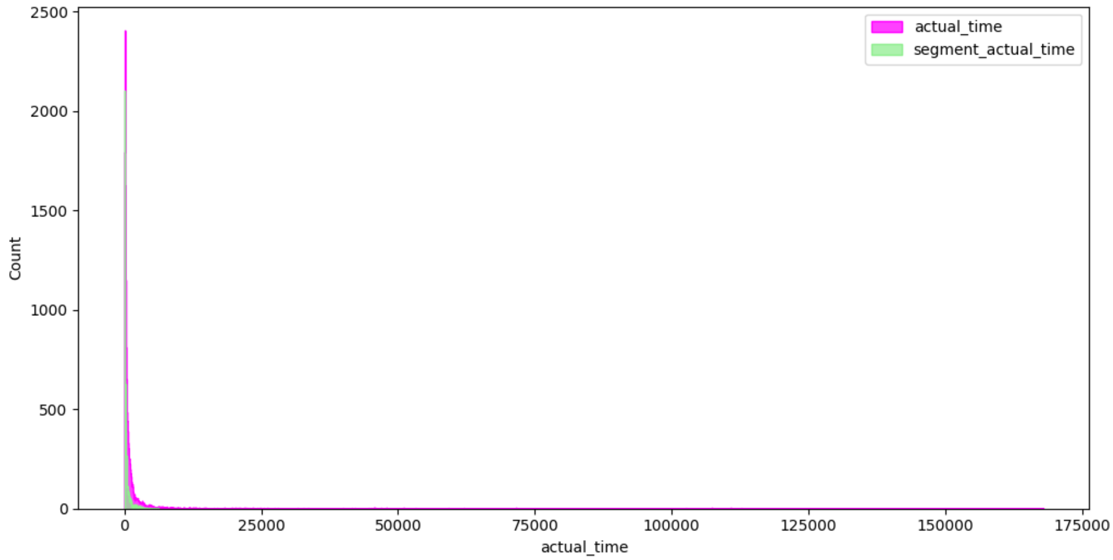
[78]:
```
# Do hypothesis testing/ visual analysis between actual_time aggregated value␣
 ↪and segment actual time aggregated value (aggregated values are the values␣
 ↪you'll get after merging the rows on the basis of trip_uuid)
df3[['actual_time', 'segment_actual_time']].describe()
```

[78]:

	actual_time	segment_actual_time
count	14817.000000	14817.000000
mean	4076.333941	353.892286
std	15216.870041	556.247965
min	9.000000	9.000000
25%	142.000000	66.000000
50%	348.000000	147.000000
75%	1063.000000	367.000000
max	167920.000000	6230.000000

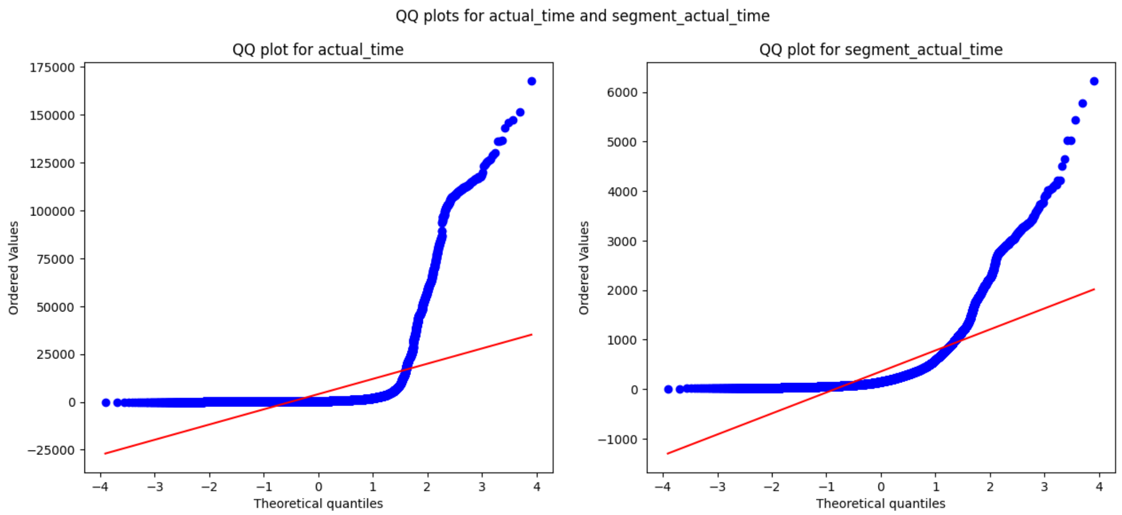
[79]:
```
plt.figure(figsize = (12, 6))
sns.histplot(df3['actual_time'], element = 'step', color = 'magenta')
sns.histplot(df3['segment_actual_time'], element = 'step', color = 'lightgreen')
plt.legend(['actual_time', 'segment_actual_time'])
plt.plot()
```

[79]: []



[80]:
```
# check for normality
# qq plot

plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.suptitle('QQ plots for actual_time and segment_actual_time')
stats.probplot(df3['actual_time'], plot = plt, dist = 'norm')
plt.title('QQ plot for actual_time')
plt.subplot(1, 2, 2)
stats.probplot(df3['segment_actual_time'], plot = plt, dist = 'norm')
plt.title('QQ plot for segment_actual_time')
plt.plot()
```

[80]: []



[81]:
```
# Shapiro test
# H0 : The sample follows normal distribution
# H1 : The sample does not follow normal distribution

test_stat, p_value = stats.shapiro(df3['actual_time'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 3.104588753102998e-88
The sample does not follow normal distribution
```

[82]:
```
test_stat, p_value = stats.shapiro(df3['segment_actual_time'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 4.784775493921149e-76
The sample does not follow normal distribution
```

[83]:
```
# Variance check using levens test
# Null Hypothesis(H0) - Homogenous Variance

# Alternate Hypothesis(HA) - Non Homogenous Variance
```

```
test_stat, p_value = stats.levene(df3['actual_time'],␣
 ↪df3['segment_actual_time'])
print('p-value', p_value)

if p_value < 0.05:
    print('The samples do not have Homogenous Variance')
else:
    print('The samples have Homogenous Variance ')
```

```
p-value 1.4988439075759175e-184
The samples do not have Homogenous Variance
```

[84]:
```
# Since the samples do not come from normal distribution T-Test cannot be␣
 ↪applied here, we can perform its non parametric equivalent test i.e.,␣
 ↪Mann-Whitney U rank test for two independent samples.

test_stat, p_value = stats.mannwhitneyu(df3['actual_time'],␣
 ↪df3['segment_actual_time'])
print('p-value', p_value)
if p_value < 0.05:
    print('The samples are not similar')
else:
    print('The samples are similar ')
```

```
p-value 0.0
The samples are not similar
```

[85]:
```
# Do hypothesis testing/ visual analysis between osrm distance aggregated value␣
 ↪and segment osrm distance aggregated value (aggregated values are the values␣
 ↪you'll get after merging the rows on the basis of trip_uuid)


df3[['osrm_distance', 'segment_osrm_distance']].describe()
```

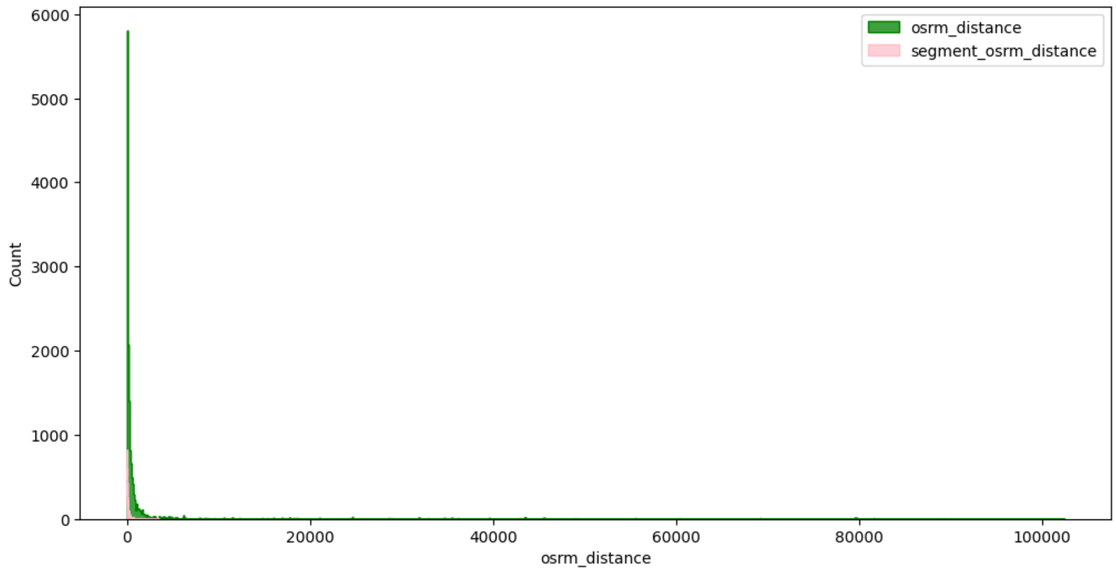
[85]:

	osrm_distance	segment_osrm_distance
count	14817.000000	14817.000000
mean	2784.231856	223.201161
std	10759.101819	416.628374
min	9.072900	9.072900
25%	65.738600	32.654500
50%	173.593600	70.154400
75%	607.677400	218.802400
max	102415.868000	3523.632400

[86]:
```
plt.figure(figsize = (12, 6))
sns.histplot(df3['osrm_distance'], element = 'step', color = 'green', bins =␣
 ↪1000)
```

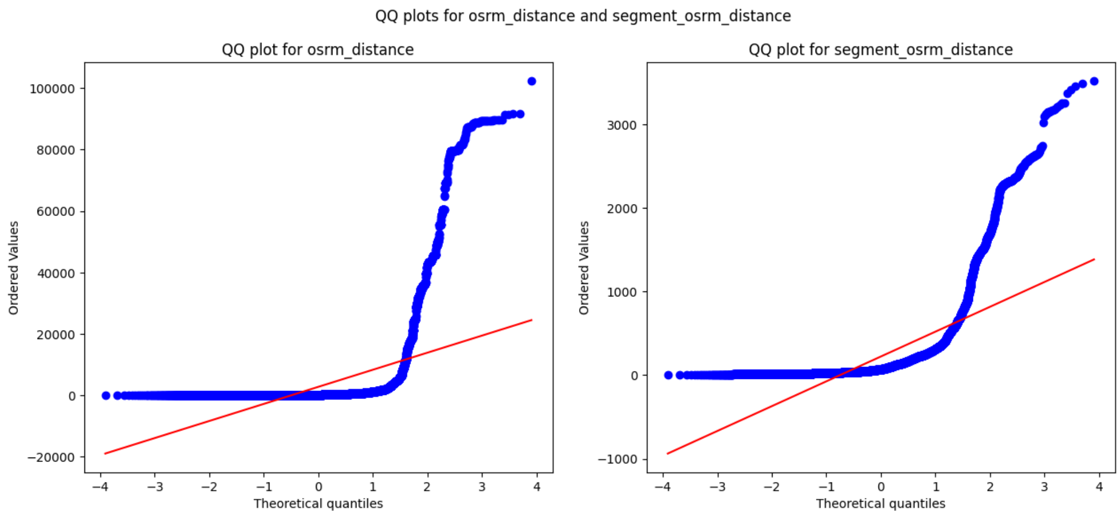
```
sns.histplot(df3['segment_osrm_distance'], element = 'step', color = 'pink',
    bins = 1000)
plt.legend(['osrm_distance', 'segment_osrm_distance'])
plt.plot()
```

[86]: []



[87]:
```
# normality check
# qq plot
plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.suptitle('QQ plots for osrm_distance and segment_osrm_distance')
stats.probplot(df3['osrm_distance'], plot = plt, dist = 'norm')
plt.title('QQ plot for osrm_distance')
plt.subplot(1, 2, 2)
stats.probplot(df3['segment_osrm_distance'], plot = plt, dist = 'norm')
plt.title('QQ plot for segment_osrm_distance')
plt.plot()
```

[87]: []



[88]:
```
# Shapiro test
# H0 : The sample follows normal distribution
# H1 : The sample does not follow normal distribution

test_stat, p_value = stats.shapiro(df3['osrm_distance'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 3.331106628196125e-88
The sample does not follow normal distribution
```

[89]:
```
test_stat, p_value = stats.shapiro(df3['segment_osrm_distance'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 2.0535644446456572e-80
The sample does not follow normal distribution
```

[90]:
```
# Varince check using levens test
  # Null Hypothesis(H0) - Homogenous Variance

# Alternate Hypothesis(HA) - Non Homogenous Variance
```

```
test_stat, p_value = stats.levene(df3['osrm_distance'],␣
 ↪df3['segment_osrm_distance'])
print('p-value', p_value)

if p_value < 0.05:
    print('The samples do not have Homogenous Variance')
else:
    print('The samples have Homogenous Variance ')
```

```
p-value 4.359664959167002e-177
The samples do not have Homogenous Variance
```

[91]:
```
# Since the samples do not follow any of the assumptions, T-Test cannot be␣
 ↪applied here. We can perform its non parametric equivalent test i.e.,␣
 ↪Mann-Whitney U rank test for two independent samples.
test_stat, p_value = stats.mannwhitneyu(df3['osrm_distance'],␣
 ↪df3['segment_osrm_distance'])
print('p-value', p_value)
if p_value < 0.05:
    print('The samples are not similar')
else:
    print('The samples are similar ')
```

```
p-value 0.0
The samples are not similar
```

[92]:
```
# Since p-value < alpha therfore it can be concluded that osrm_distance and␣
 ↪segment_osrm_distance are not similar.
```

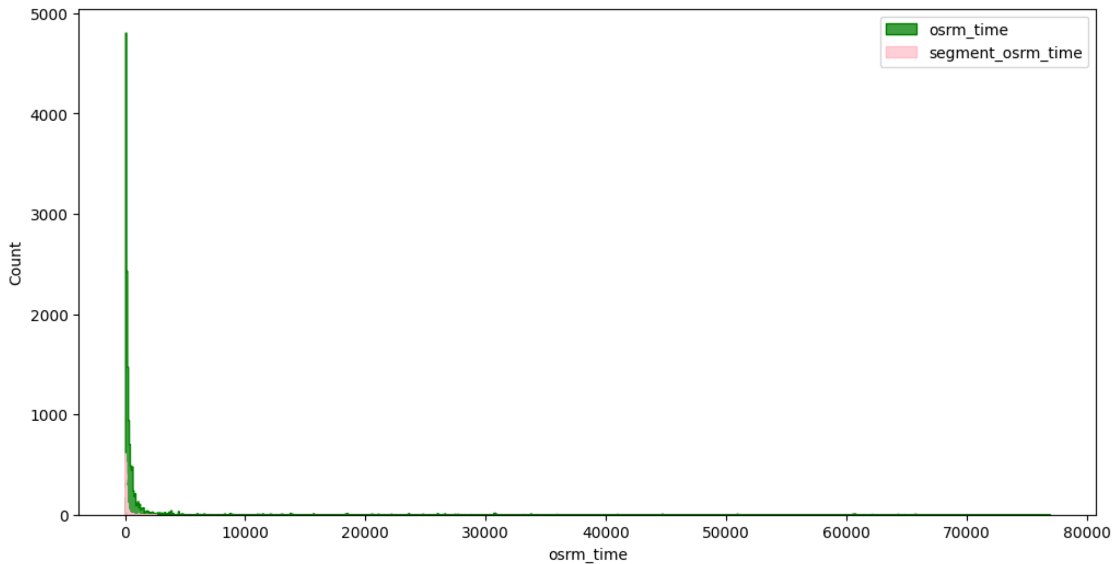
[93]:
```
# Do hypothesis testing/ visual analysis between osrm time aggregated value and␣
 ↪segment osrm time aggregated value (aggregated values are the values you'll␣
 ↪get after merging the rows on the basis of trip_uuid)
df2[['osrm_time', 'segment_osrm_time']].describe().T
```

[93]:
```
                      count      mean       std  min       25%       50%  \
osrm_time          144867.0  3.564471  5.133518  0.1  0.450000  1.066667
segment_osrm_time  144867.0  0.308459  0.246266  0.0  0.183333  0.283333

                        75%    max
osrm_time          4.283333  28.10
segment_osrm_time  0.366667  26.85
```

[94]:
```
plt.figure(figsize = (12, 6))
sns.histplot(df3['osrm_time'], element = 'step', color = 'green', bins = 1000)
sns.histplot(df3['segment_osrm_time'], element = 'step', color = 'pink', bins =␣
 ↪1000)
plt.legend(['osrm_time', 'segment_osrm_time'])
```

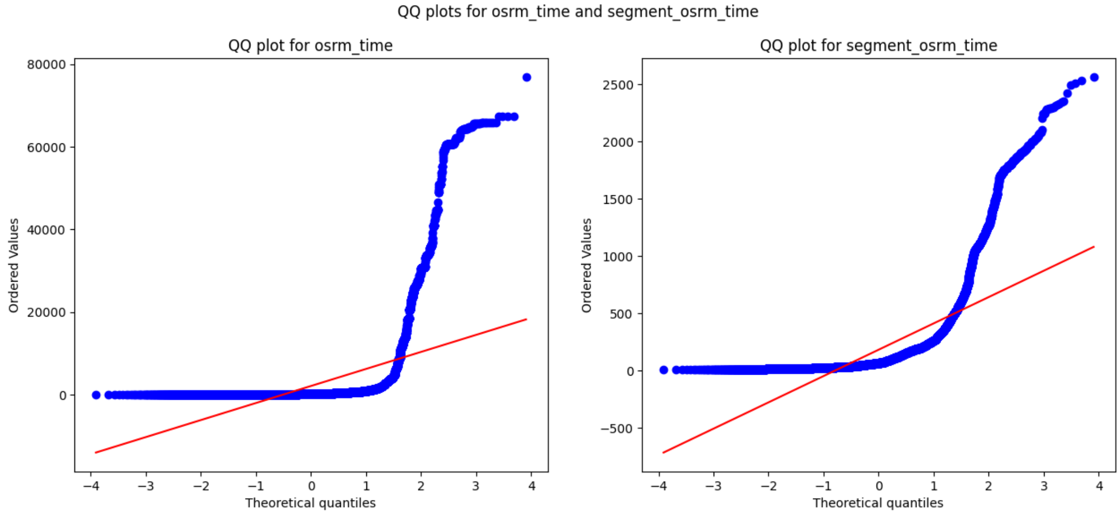
```
plt.plot()
```

[94]: []



[95]:
```
# Normality check using qq plot
plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.suptitle('QQ plots for osrm_time and segment_osrm_time')
stats.probplot(df3['osrm_time'], plot = plt, dist = 'norm')
plt.title('QQ plot for osrm_time')
plt.subplot(1, 2, 2)
stats.probplot(df3['segment_osrm_time'], plot = plt, dist = 'norm')
plt.title('QQ plot for segment_osrm_time')
plt.plot()
```

[95]: []



[96]:
```
# Shapiro test
# H1 : The sample follows normal distribution
# H2 : The sample does not follow normal distribution


test_stat, p_value = stats.shapiro(df3['osrm_time'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 2.0190093052337492e-88
The sample does not follow normal distribution
```

[97]:
```
test_stat, p_value = stats.shapiro(df3['segment_osrm_time'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 4.250788344493806e-79
The sample does not follow normal distribution
```

[98]:
```
# Variance test using levens test
# Null Hypothesis(H0) - Homogenous Variance

# Alternate Hypothesis(HA) - Non Homogenous Variance

```

```
test_stat, p_value = stats.levene(df3['osrm_time'], df3['segment_osrm_time'])
print('p-value', p_value)

if p_value < 0.05:
    print('The samples do not have Homogenous Variance')
else:
    print('The samples have Homogenous Variance ')
```

```
p-value 6.44468592657146e-179
The samples do not have Homogenous Variance
```

[99]:
```
# Since the samples do not follow any of the assumptions, T-Test cannot be␣
↪applied here. We can perform its non parametric equivalent test i.e.,␣
↪Mann-Whitney U rank test for two independent samples.

test_stat, p_value = stats.mannwhitneyu(df3['osrm_time'],␣
↪df3['segment_osrm_time'])
print('p-value', p_value)
if p_value < 0.05:
    print('The samples are not similar')
else:
    print('The samples are similar ')
```

```
p-value 0.0
The samples are not similar
```

[100]:
```
# Since p-value < alpha therfore it can be concluded that osrm_time and␣
↪segment_osrm_time are not similar.
```

[101]:
```
# Find outliers in the numerical variables (you might find outliers in almost␣
↪all the variables), and check it using visual analysis
numerical_columns = ['od_total_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']
df3[numerical_columns].describe().T
```

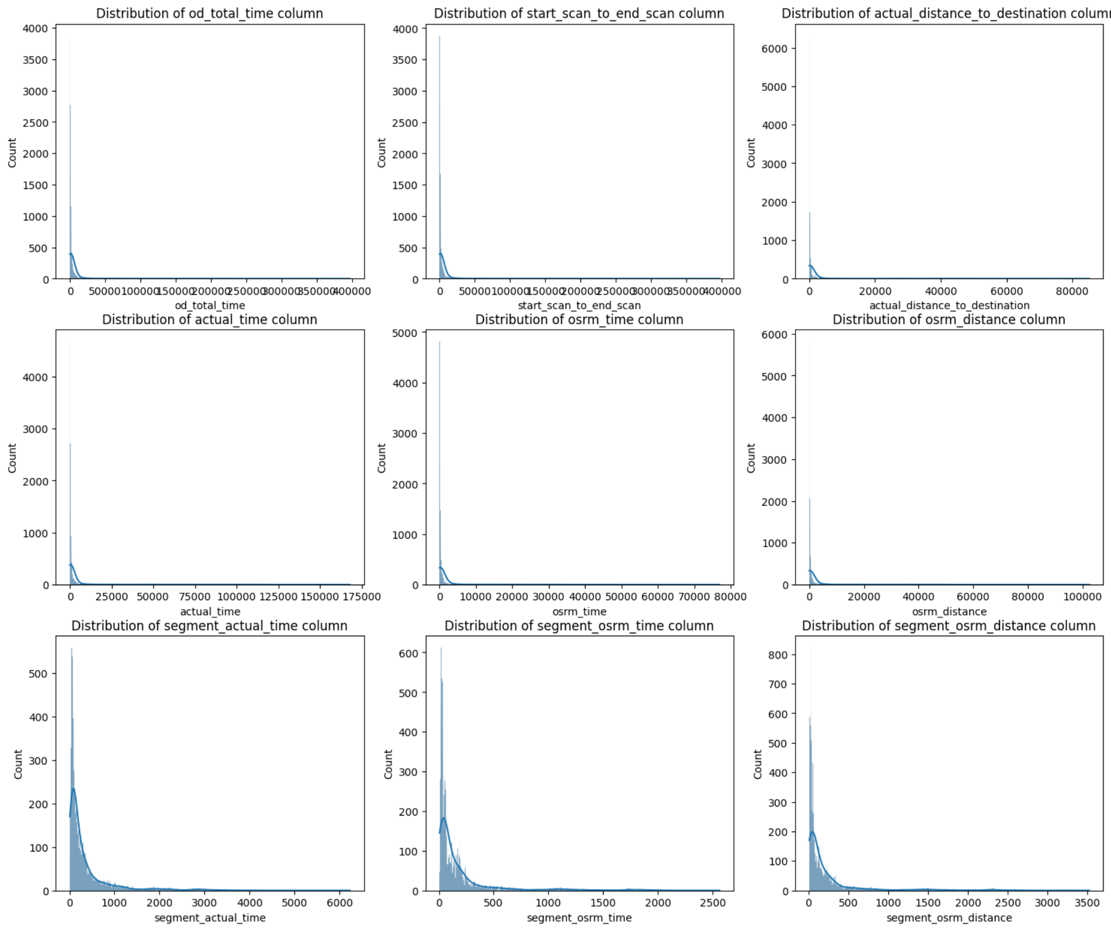
[101]:

	count	mean	std	min \
od_total_time	14817.0	9403.194234	33707.727320	26.500000
start_scan_to_end_scan	14817.0	9398.345482	33701.706672	26.000000
actual_distance_to_destination	14817.0	2288.554169	8798.110164	9.002461
actual_time	14817.0	4076.333941	15216.870041	9.000000
osrm_time	14817.0	2091.007289	7956.882351	6.000000
osrm_distance	14817.0	2784.231856	10759.101819	9.072900
segment_actual_time	14817.0	353.892286	556.247965	9.000000
segment_osrm_time	14817.0	180.949787	314.542047	6.000000

```
segment_osrm_distance           14817.0   223.201161    416.628374   9.072900

                                      25%          50%           75%  \
od_total_time                   409.580000  987.520000  2832.710000
start_scan_to_end_scan          408.000000  985.000000  2826.000000
actual_distance_to_destination   49.597866  134.059655   463.956888
actual_time                     142.000000  348.000000  1063.000000
osrm_time                        62.000000  167.000000   516.000000
osrm_distance                    65.738600  173.593600   607.677400
segment_actual_time              66.000000  147.000000   367.000000
segment_osrm_time                31.000000   65.000000   185.000000
segment_osrm_distance            32.654500   70.154400   218.802400

                                         max
od_total_time                   396834.500000
start_scan_to_end_scan          396800.000000
actual_distance_to_destination   85110.885093
actual_time                     167920.000000
osrm_time                        76953.000000
osrm_distance                   102415.868000
segment_actual_time               6230.000000
segment_osrm_time                 2564.000000
segment_osrm_distance             3523.632400
```

[102]:

```
plt.figure(figsize = (18, 15))
for i in range(len(numerical_columns)):
    plt.subplot(3, 3, i + 1)
    sns.histplot(df3[numerical_columns[i]], bins = 1000, kde = True)
    plt.title(f"Distribution of {numerical_columns[i]} column")
    plt.plot()
```



[103]:
```python
# Checking for outliers

for i in numerical_columns:
    Q1 = np.quantile(df2[i], 0.25)
    Q3 = np.quantile(df2[i], 0.75)
    IQR = Q3 - Q1
    LB = Q1 - 1.5 * IQR
    UB = Q3 + 1.5 * IQR
    outliers = df3.loc[(df2[i] < LB) | (df2[i] > UB)]
    print('Column :', i)
    print(f'Q1 : {Q1}')
    print(f'Q3 : {Q3}')
    print(f'IQR : {IQR}')
    print(f'LB : {LB}')
    print(f'UB : {UB}')
    print(f'Number of outliers : {outliers.shape[0]}')
    print('<<<<<<<<<<<<<>>>>>>>>>>>>>')
```

```
Column : od_total_time
```

```
Q1 : 161.5
Q3 : 1634.95
IQR : 1473.45
LB : -2048.675
UB : 3845.125
Number of outliers : 0
<<<<<<<<<<<<<>>>>>>>>>>>>>
Column : start_scan_to_end_scan
Q1 : 2.683333333333333
Q3 : 27.233333333333334
IQR : 24.55
LB : -34.14166666666667
UB : 64.05833333333334
Number of outliers : 0
<<<<<<<<<<<<<>>>>>>>>>>>>>
Column : actual_distance_to_destination
Q1 : 23.355874361432974
Q3 : 286.7088745976663
IQR : 263.3530002362333
LB : -371.6736259929169
UB : 681.7383749520162
Number of outliers : 1469
<<<<<<<<<<<<<>>>>>>>>>>>>>
Column : actual_time
Q1 : 0.85
Q3 : 8.55
IQR : 7.700000000000001
LB : -10.700000000000001
UB : 20.1
Number of outliers : 1318
<<<<<<<<<<<<<>>>>>>>>>>>>>
Column : osrm_time
Q1 : 0.45
Q3 : 4.283333333333333
IQR : 3.833333333333333
LB : -5.3
UB : 10.033333333333333
Number of outliers : 1439
<<<<<<<<<<<<<>>>>>>>>>>>>>
Column : osrm_distance
Q1 : 29.9147
Q3 : 343.19325000000003
IQR : 313.27855000000005
LB : -440.0031250000001
UB : 813.1110750000001
Number of outliers : 1449
<<<<<<<<<<<<<>>>>>>>>>>>>>
Column : segment_actual_time
```

```
Q1 : 0.3333333333333333
Q3 : 0.6666666666666666
IQR : 0.3333333333333333
LB : -0.16666666666666669
UB : 1.1666666666666665
Number of outliers : 932
<<<<<<<<<<<<<>>>>>>>>>>>>>
Column : segment_osrm_time
Q1 : 0.18333333333333332
Q3 : 0.36666666666666664
IQR : 0.18333333333333332
LB : -0.09166666666666665
UB : 0.6416666666666666
Number of outliers : 629
<<<<<<<<<<<<<>>>>>>>>>>>>>
Column : segment_osrm_distance
Q1 : 12.0701
Q3 : 27.81325
IQR : 15.74315
LB : -11.544625
UB : 51.427975
Number of outliers : 417
<<<<<<<<<<<<<>>>>>>>>>>>>>
```

[104]:
```python
# Do one-hot encoding of categorical variables (like route_type)
df3['route_type'].value_counts()
```

[104]:
```
route_type
Carting    8908
FTL        5909
Name: count, dtype: int64
```

[105]:
```python
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df2['route_type'] = label_encoder.fit_transform(df2['route_type'])
```

[106]:
```python
df2['route_type'].value_counts()
```

[106]:
```
route_type
1    99660
0    45207
Name: count, dtype: int64
```

[107]:
```python
df2['data'].value_counts()
```

[107]:
```
data
training    104858
```

```
test         40009
Name: count, dtype: int64
```

[108]:
```
label_encoder = LabelEncoder()
df2['data'] = label_encoder.fit_transform(df2['data'])
```

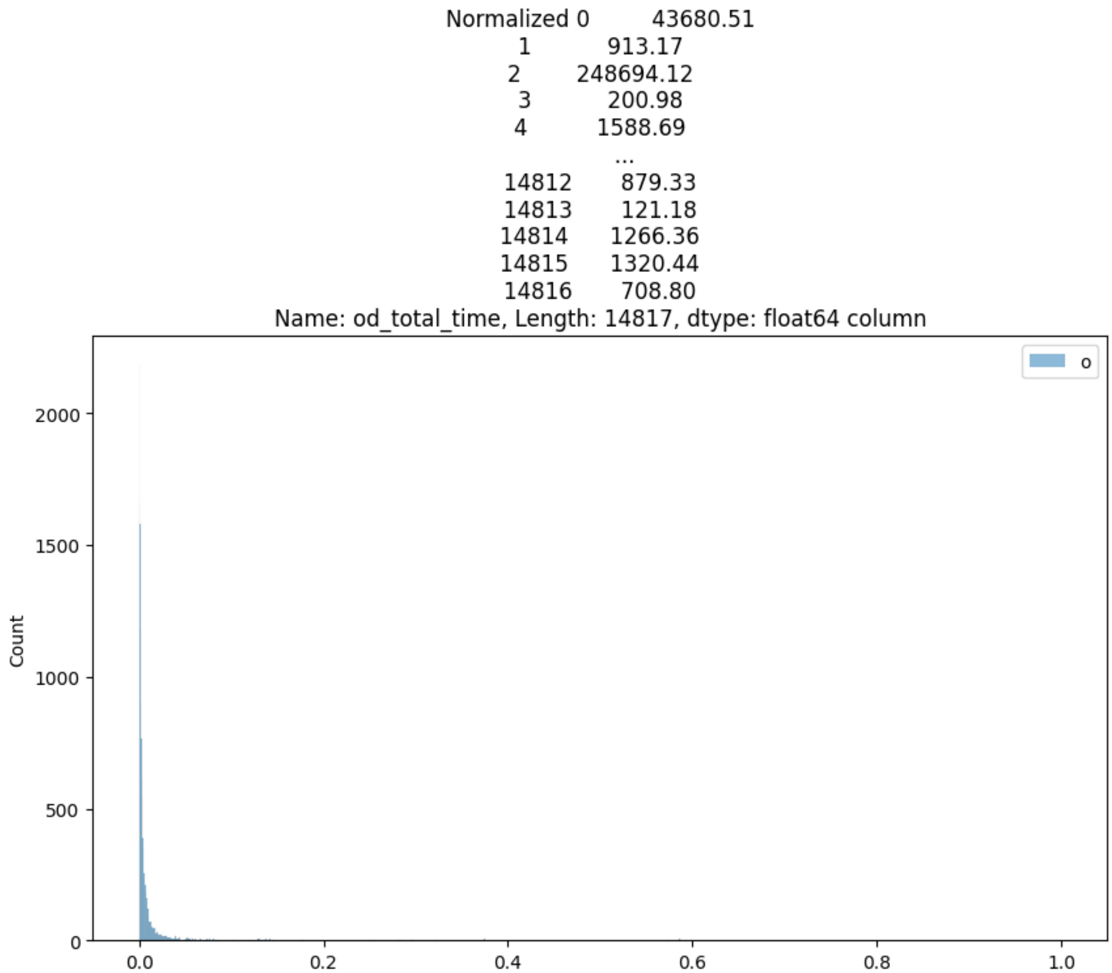
[109]:
```
df2['data'].value_counts()
```

[109]:
```
data
1    104858
0     40009
Name: count, dtype: int64
```

[110]:
```
# Normalize/ Standardize the numerical features using MinMaxScaler or␣
↪StandardScaler.
from sklearn.preprocessing import MinMaxScaler
```

[111]:
```
plt.figure(figsize = (10, 6))
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df3['od_total_time'].to_numpy().reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Normalized {df3['od_total_time']} column")
plt.legend('od_total_time')
plt.plot()
```

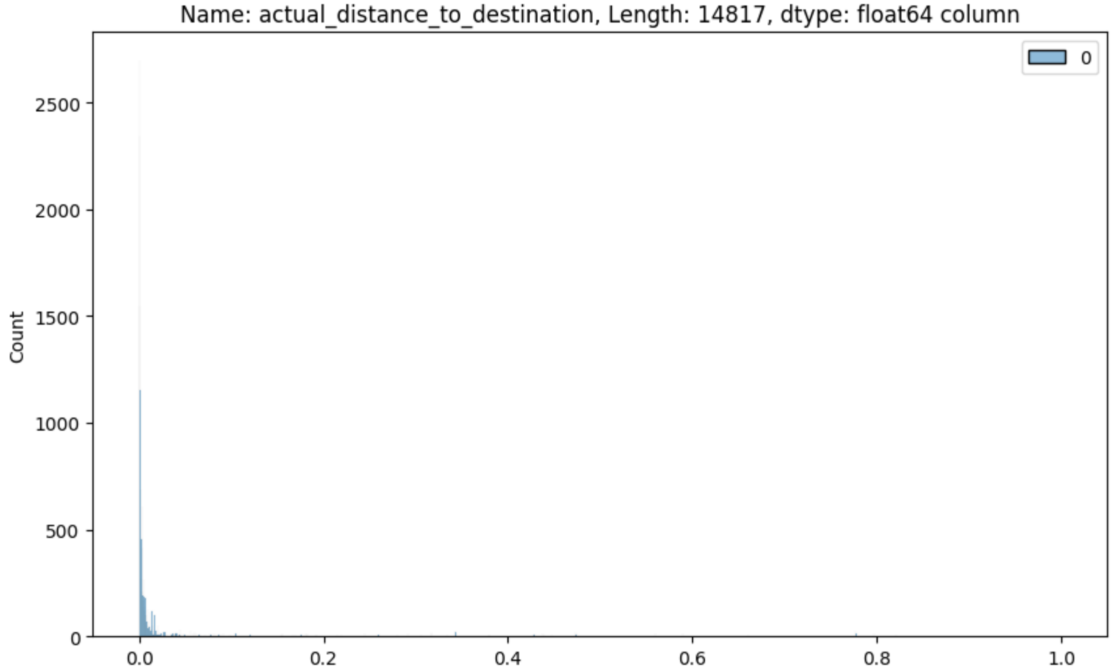
[111]:
```
[]
```



[112]:
```
plt.figure(figsize = (10, 6))
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df3['actual_distance_to_destination'].to_numpy().
 ↪reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Normalized {df3['actual_distance_to_destination']} column")
plt.plot()
```

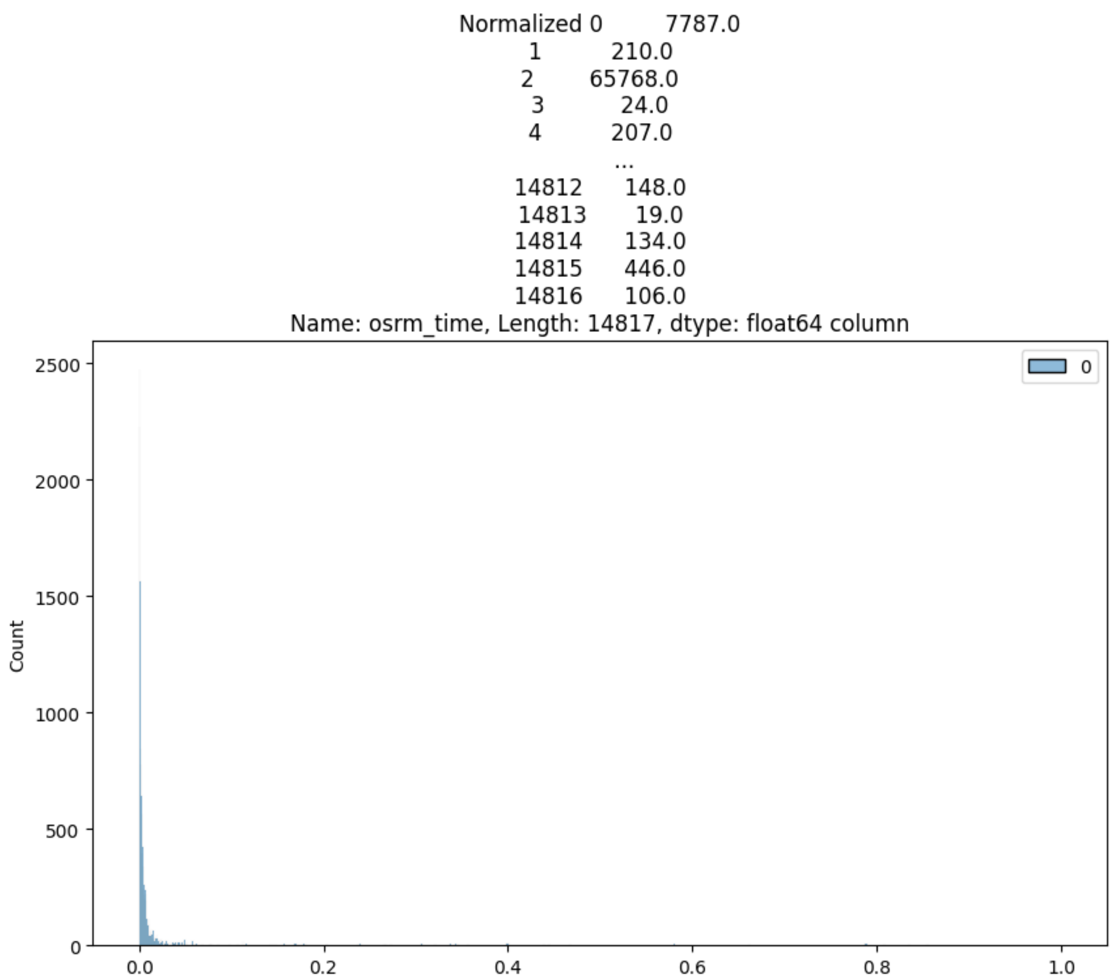
[112]: []

Normalized 0   8860.812105
1   240.208306
2   68163.502238
3   28.529648
4   239.007304
...
14812   141.057373
14813   25.130640
14814   93.743842
14815   355.281673
14816   110.239116
Name: actual_distance_to_destination, Length: 14817, dtype: float64 column



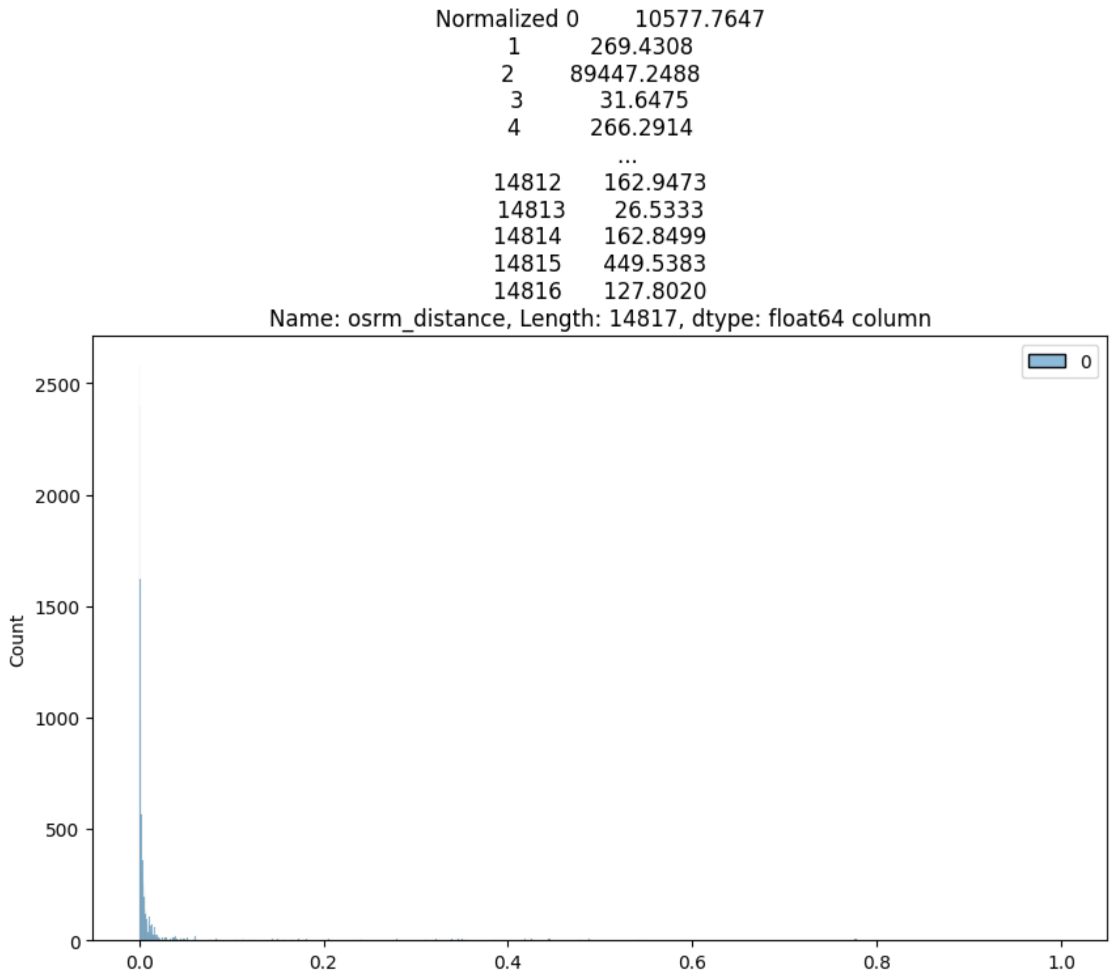
[113]:
```python
plt.figure(figsize = (10, 6))
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df3['osrm_time'].to_numpy().reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Normalized {df3['osrm_time']} column")
plt.plot()
```

[113]: []



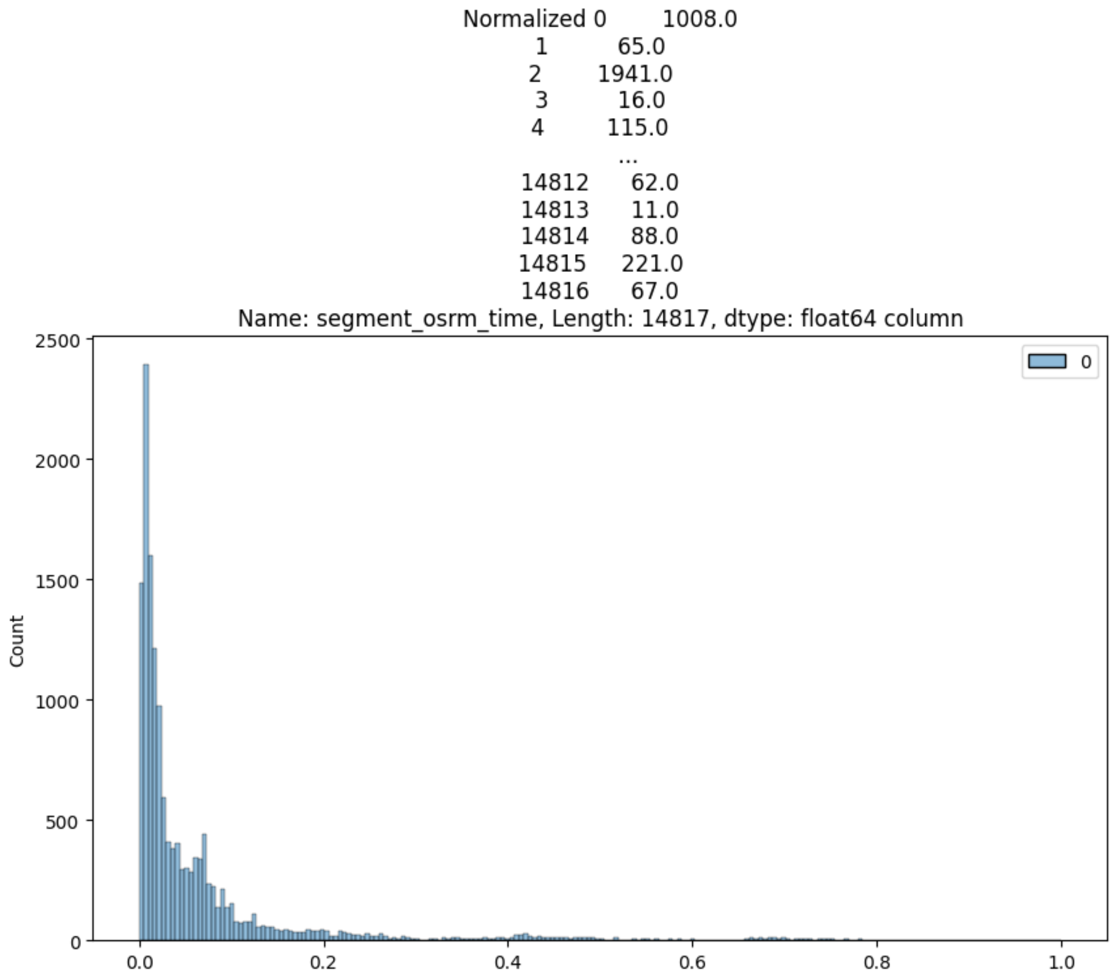
[114]:
```
plt.figure(figsize = (10, 6))
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df3['osrm_distance'].to_numpy().reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Normalized {df3['osrm_distance']} column")
plt.plot()
```

[114]: []



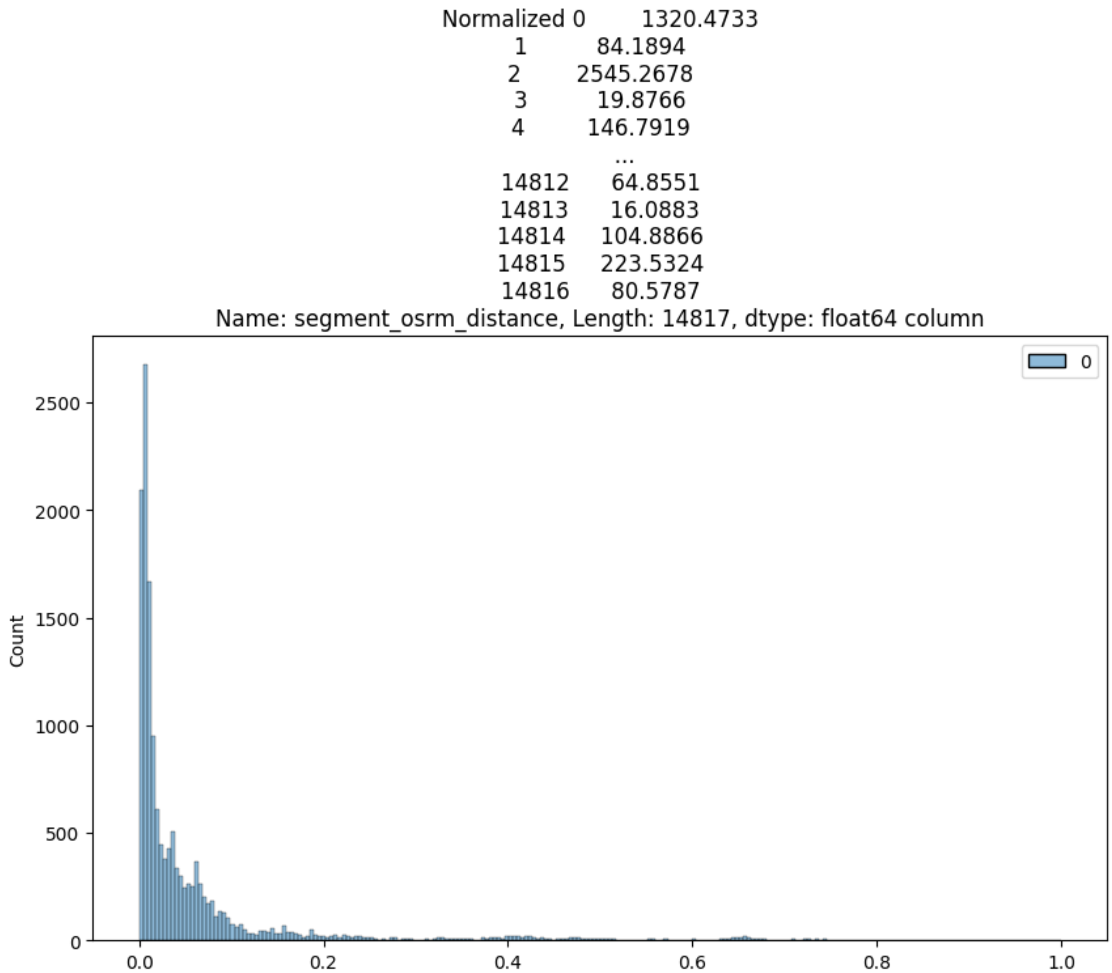
[115]:
```
plt.figure(figsize = (10, 6))
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df3['segment_osrm_time'].to_numpy().reshape(-1,␣
↪1))
sns.histplot(scaled)
plt.title(f"Normalized {df3['segment_osrm_time']} column")
plt.plot()
```

[115]: []



[116]:
```
plt.figure(figsize = (10, 6))
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df3['segment_osrm_distance'].to_numpy().
  ↪reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Normalized {df3['segment_osrm_distance']} column")
plt.plot()
```

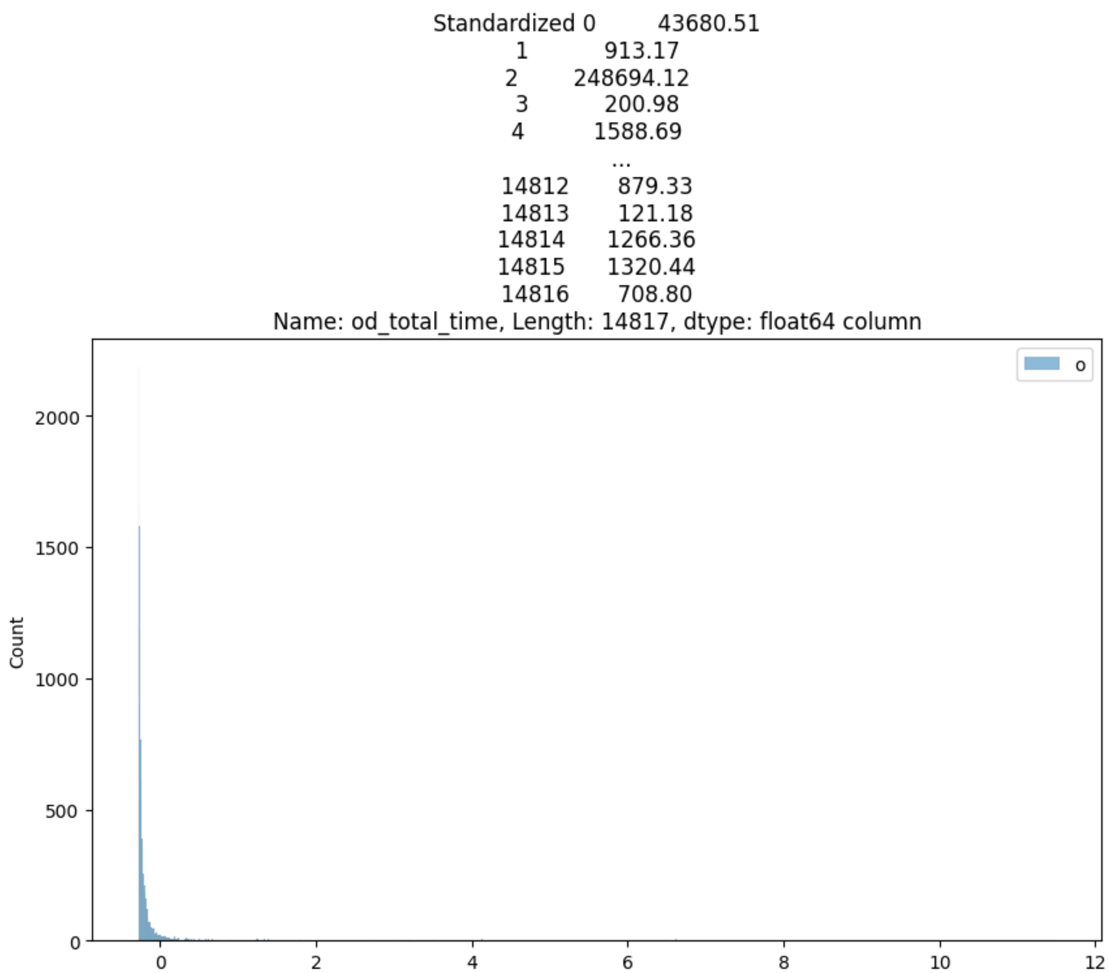
[116]: []



[117]:
```
from sklearn.preprocessing import StandardScaler
```

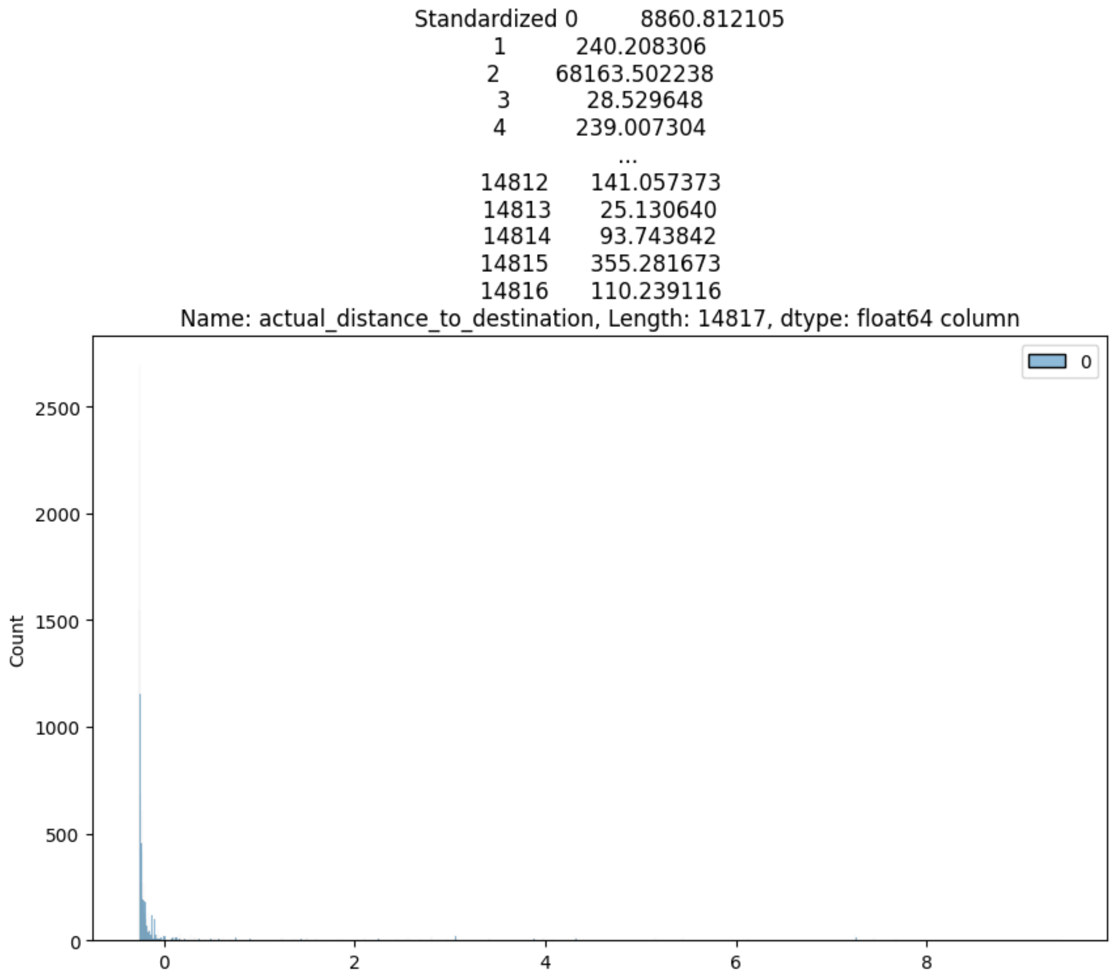
[118]:
```
plt.figure(figsize = (10, 6))
# define standard scaler
scaler = StandardScaler()
# transform data
scaled = scaler.fit_transform(df3['od_total_time'].to_numpy().reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Standardized {df3['od_total_time']} column")
plt.legend('od_total_time')
plt.plot()
```

[118]: []



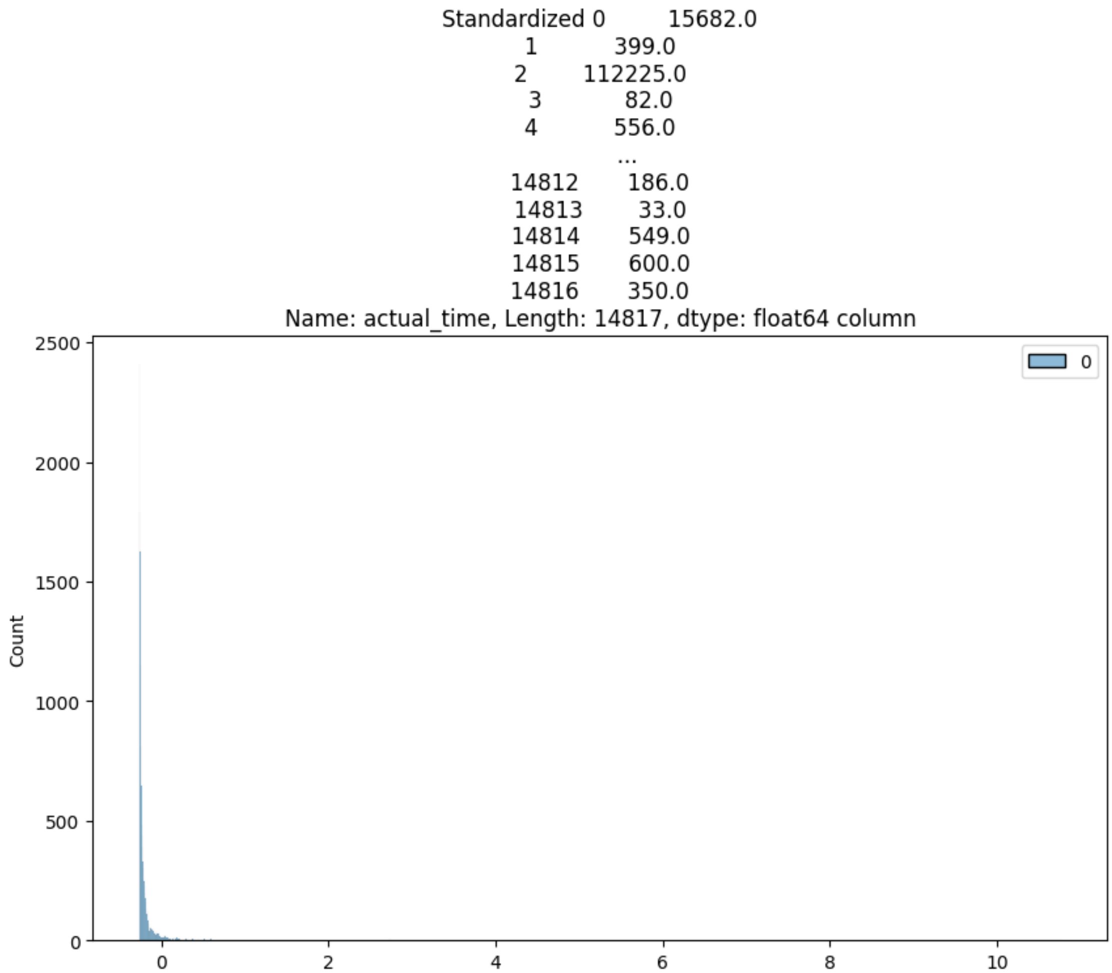
[119]:
```
plt.figure(figsize = (10, 6))
scaler = StandardScaler()
scaled = scaler.fit_transform(df3['actual_distance_to_destination'].to_numpy().
 ↪reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Standardized {df3['actual_distance_to_destination']} column")
plt.plot()
```

[119]: []




```
[120]: plt.figure(figsize = (10, 6))
       scaler = StandardScaler()
       scaled = scaler.fit_transform(df3['actual_time'].to_numpy().reshape(-1, 1))
       sns.histplot(scaled)
       plt.title(f"Standardized {df3['actual_time']} column")
       plt.plot()
```

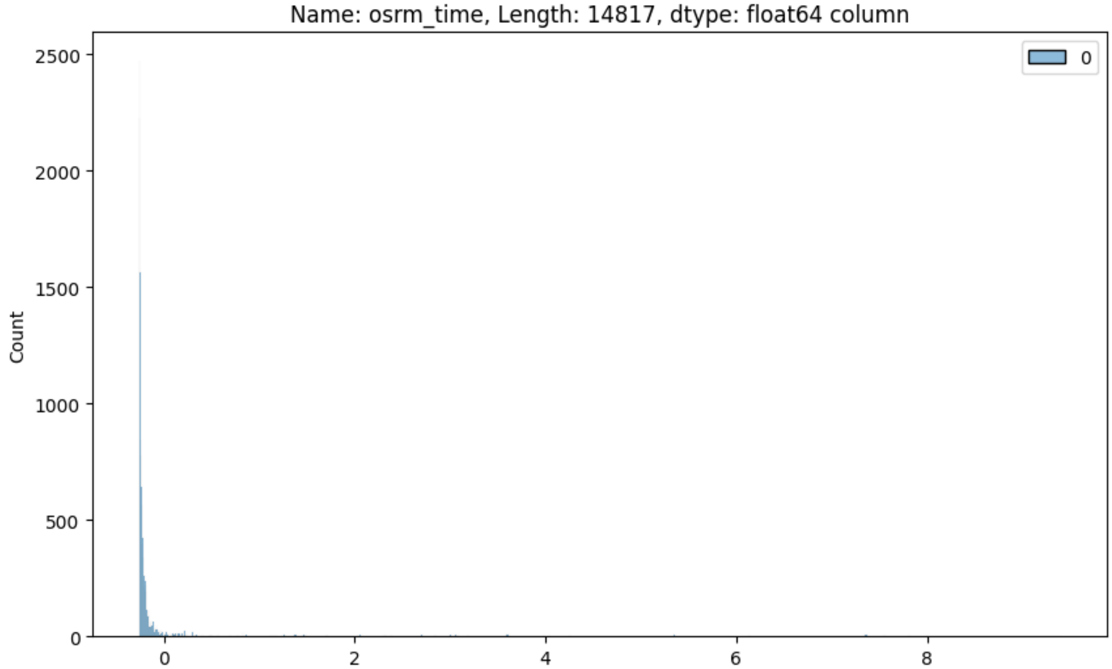
```
[120]: []
```




```
[121]: plt.figure(figsize = (10, 6))
       scaler = StandardScaler()
       scaled = scaler.fit_transform(df3['osrm_time'].to_numpy().reshape(-1, 1))
       sns.histplot(scaled)
       plt.title(f"Standardized {df3['osrm_time']} column")
       plt.plot()
```

```
[121]: []
```

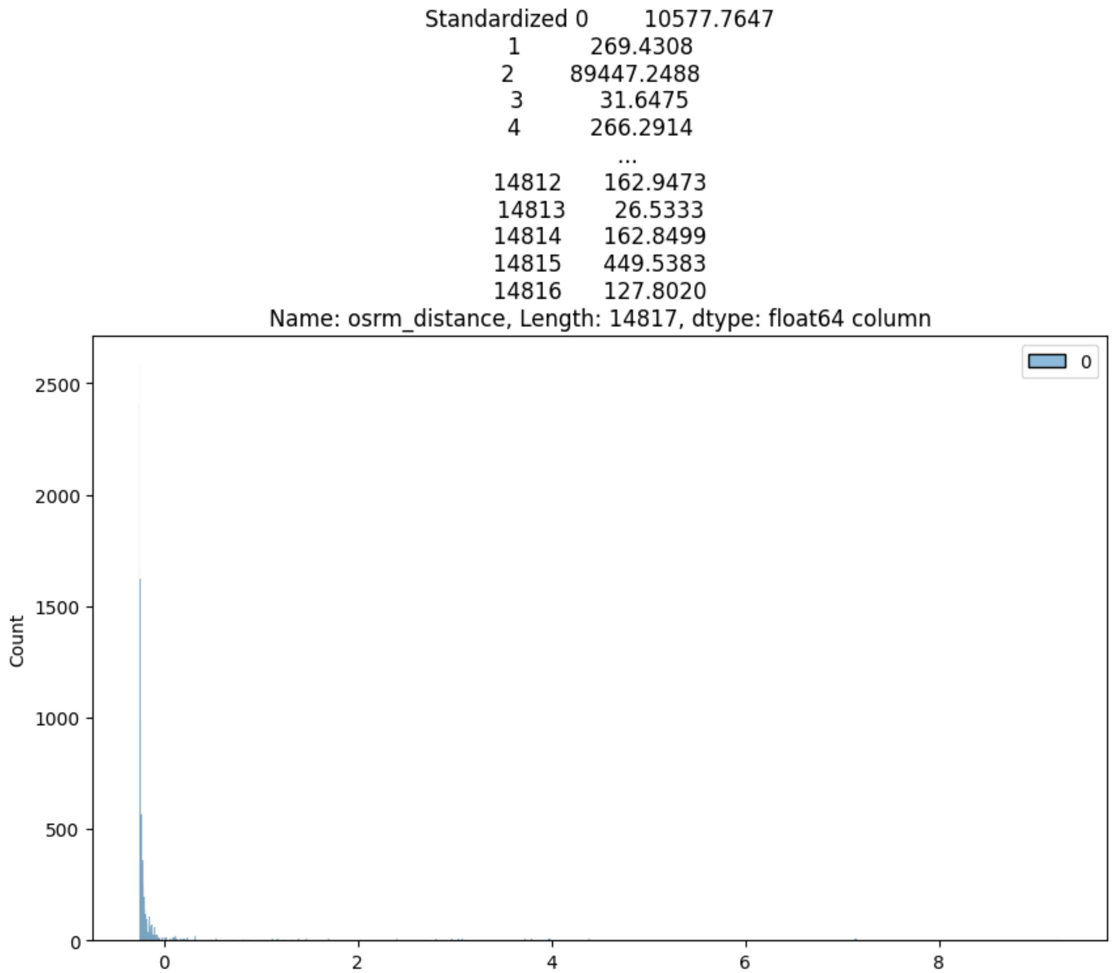
Standardized 0   7787.0
1   210.0
2   65768.0
3   24.0
4   207.0
...
14812   148.0
14813   19.0
14814   134.0
14815   446.0
14816   106.0
Name: osrm_time, Length: 14817, dtype: float64 column



[122]:

```
plt.figure(figsize = (10, 6))
scaler = StandardScaler()
scaled = scaler.fit_transform(df3['osrm_distance'].to_numpy().reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Standardized {df3['osrm_distance']} column")
plt.plot()
```

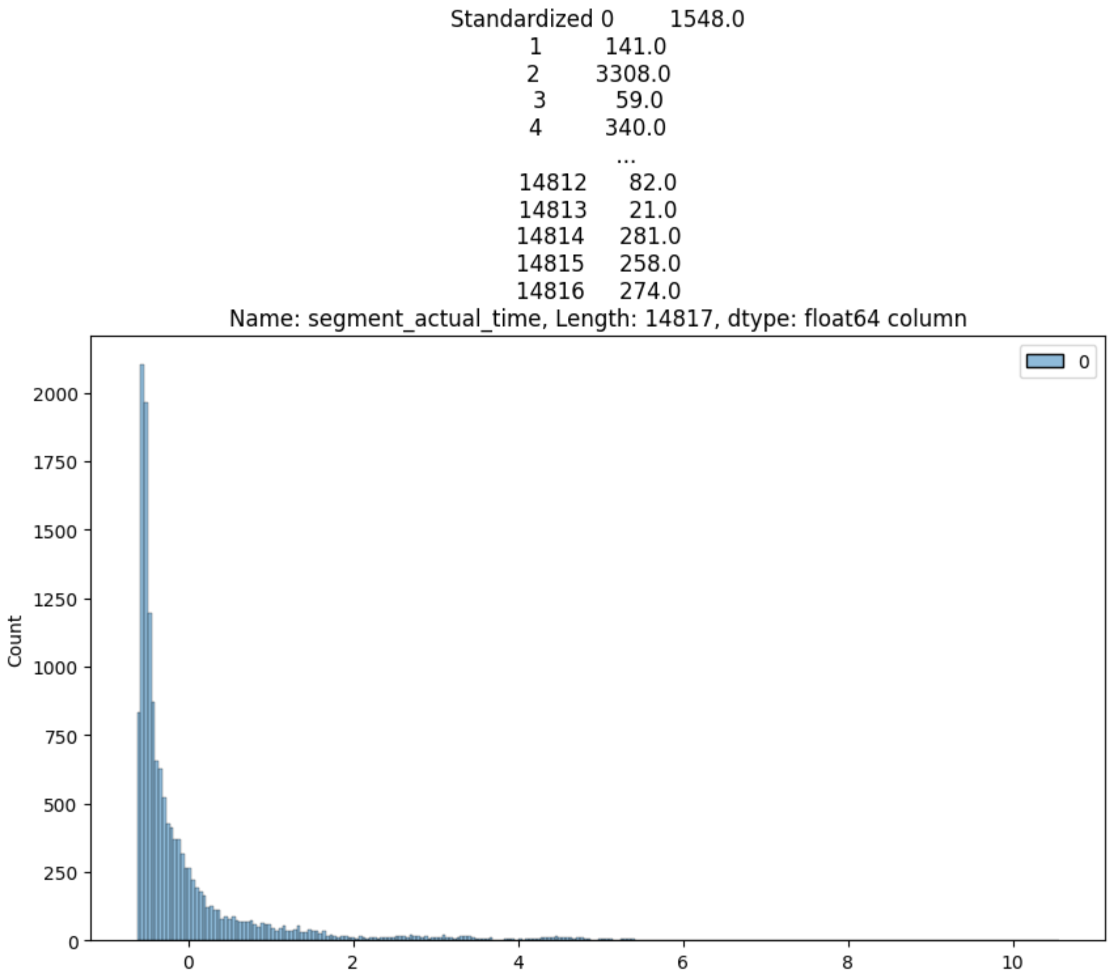
[122]: []



[123]:

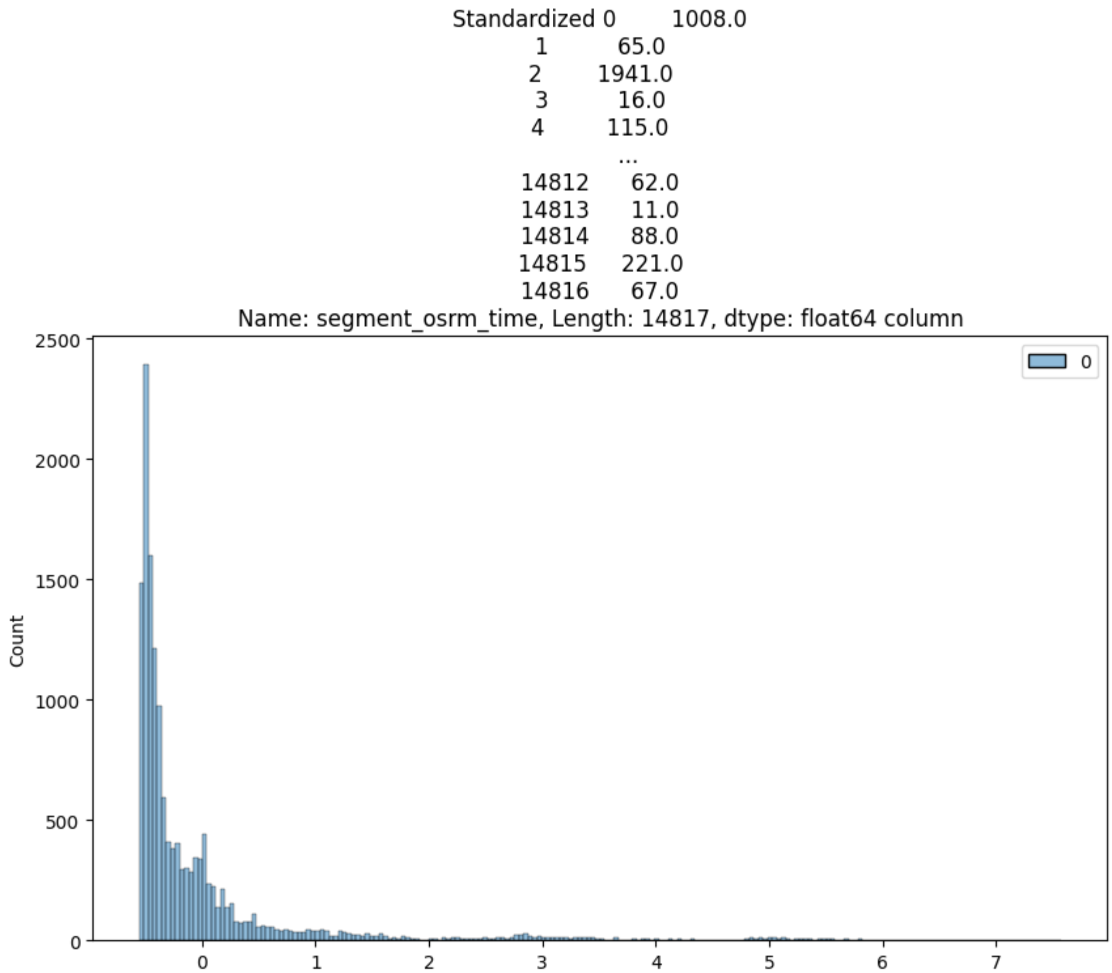
```
plt.figure(figsize = (10, 6))
scaler = StandardScaler()
scaled = scaler.fit_transform(df3['segment_actual_time'].to_numpy().reshape(-1,
 ↪1))
sns.histplot(scaled)
plt.title(f"Standardized {df3['segment_actual_time']} column")
plt.plot()
```

[123]: []



[124]:
```
plt.figure(figsize = (10, 6))
scaler = StandardScaler()
scaled = scaler.fit_transform(df3['segment_osrm_time'].to_numpy().reshape(-1,␣
 ↪1))
sns.histplot(scaled)
plt.title(f"Standardized {df3['segment_osrm_time']} column")
plt.plot()
```

[124]: []

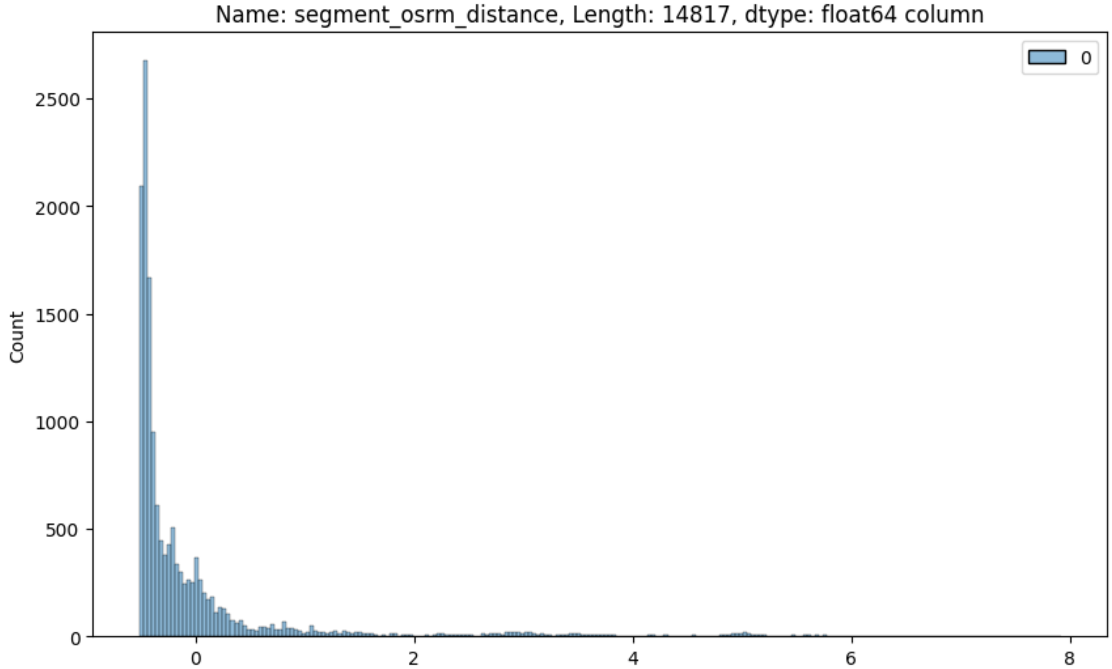


[125]:

```
plt.figure(figsize = (10, 6))
scaler = StandardScaler()
scaled = scaler.fit_transform(df3['segment_osrm_distance'].to_numpy().
    ↪reshape(-1, 1))
sns.histplot(scaled)
plt.title(f"Standardized {df3['segment_osrm_distance']} column")
plt.plot()
```

[125]: []

```
Standardized 0        1320.4733
1          84.1894
2        2545.2678
3          19.8766
4         146.7919
            ...
14812      64.8551
14813      16.0883
14814     104.8866
14815     223.5324
14816      80.5787
Name: segment_osrm_distance, Length: 14817, dtype: float64 column
```



[125]: