Delhivery :Feature Engineering Business Case

In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import shapiro
from scipy.stats import boxcox
import statsmodels.api as sm
from scipy.stats import ttest_ind
```

In [7]:
```
df = pd.read_csv('delhivery_data.csv')
df.head()
```

Out[7]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_n
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu

5 rows × 24 columns

In [8]:
```
df.shape
```

Out[8]:
```
(144867, 24)
```

In [9]:
```
df.size
```

Out[9]:
```
3476808
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'is_cutoff', 'cutoff_factor',
       'cutoff_timestamp', 'actual_distance_to_destination', 'actual_time',
       'osrm_time', 'osrm_distance', 'factor', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'segment_factor'],
      dtype='object')
```

In [11]:
```
df.info
```

Out[11]:
```
<bound method DataFrame.info of             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
...          ...                         ...
144862  training  2018-09-20 16:24:28.436231
144863  training  2018-09-20 16:24:28.436231
144864  training  2018-09-20 16:24:28.436231
144865  training  2018-09-20 16:24:28.436231
144866  training  2018-09-20 16:24:28.436231

                                    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
...                                                   ...        ...
144862  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...    Carting
144863  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...    Carting
144864  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...    Carting
144865  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...    Carting
144866  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...    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)
...                         ...           ...                        ...
144862  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)
144863  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)
144864  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)
144865  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)
144866  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)

       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)
...                   ...                           ...
144862       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
144863       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
144864       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
144865       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
144866       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)

                  od_start_time  ...            cutoff_timestamp  \
0       2018-09-20 03:21:32.418600  ...         2018-09-20 04:27:55
1       2018-09-20 03:21:32.418600  ...         2018-09-20 04:17:55
2       2018-09-20 03:21:32.418600  ...  2018-09-20 04:01:19.505586
3       2018-09-20 03:21:32.418600  ...         2018-09-20 03:39:57
4       2018-09-20 03:21:32.418600  ...         2018-09-20 03:33:55
...                            ...  ...                         ...
144862  2018-09-20 16:24:28.436231  ...         2018-09-20 21:57:20
144863  2018-09-20 16:24:28.436231  ...         2018-09-20 21:31:18
144864  2018-09-20 16:24:28.436231  ...         2018-09-20 21:11:18
144865  2018-09-20 16:24:28.436231  ...         2018-09-20 20:53:19
144866  2018-09-20 16:24:28.436231  ...  2018-09-20 16:24:28.436231

        actual_distance_to_destination  actual_time  osrm_time osrm_distance  \
0                            10.435660         14.0       11.0       11.9653
1                            18.936842         24.0       20.0       21.7243
2                            27.637279         40.0       28.0       32.5395
3                            36.118028         62.0       40.0       45.5620
4                            39.386040         68.0       44.0       54.2181
...                                ...          ...        ...          ...
144862                       45.258278         94.0       60.0       67.9280
144863                       54.092531        120.0       76.0       85.6829
144864                       66.163591        140.0       88.0       97.0933
144865                       73.680667        158.0       98.0      111.2709
144866                       70.039010        426.0       95.0       88.7319

          factor  segment_actual_time  segment_osrm_time  \
0       1.272727                 14.0               11.0
1       1.200000                 10.0                9.0
2       1.428571                 16.0                7.0
3       1.550000                 21.0               12.0
4       1.545455                  6.0                5.0
...          ...                  ...                ...
144862  1.566667                 12.0               12.0
```

```
144863  1.578947                26.0              21.0
144864  1.590909                20.0              34.0
144865  1.612245                17.0              27.0
144866  4.484211               268.0               9.0

        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
...                       ...             ...
144862                 8.1858        1.000000
144863                17.3725        1.238095
144864                20.7053        0.588235
144865                18.8885        0.629630
144866                 8.8088       29.777778

[144867 rows x 24 columns]>
```

In [12]:
```
remove_col = ['segment_factor', 'factor','cutoff_timestamp','cutoff_factor','is_cutoff']
```

In [13]:
```
df.drop(columns=remove_col, inplace=True)
```

In [14]:
```
df.shape
```

Out[14]:
```
(144867, 19)
```

In [15]:
```
df["data"].nunique() , df["data"].unique()
```

Out[15]:
```
(2, array(['training', 'test'], dtype=object))
```

In [16]:
```
for i in df.columns:
    print(f"The unique elements in the columns are :: {i} ----->  {df[i].nunique()}")
```

```
The unique elements in the columns are :: data ----->  2
The unique elements in the columns are :: trip_creation_time ----->  14817
The unique elements in the columns are :: route_schedule_uuid ----->  1504
The unique elements in the columns are :: route_type ----->  2
The unique elements in the columns are :: trip_uuid ----->  14817
The unique elements in the columns are :: source_center ----->  1508
The unique elements in the columns are :: source_name ----->  1498
The unique elements in the columns are :: destination_center ----->  1481
The unique elements in the columns are :: destination_name ----->  1468
The unique elements in the columns are :: od_start_time ----->  26369
The unique elements in the columns are :: od_end_time ----->  26369
The unique elements in the columns are :: start_scan_to_end_scan ----->  1915
The unique elements in the columns are :: actual_distance_to_destination ----->  144515
The unique elements in the columns are :: actual_time ----->  3182
The unique elements in the columns are :: osrm_time ----->  1531
The unique elements in the columns are :: osrm_distance ----->  138046
The unique elements in the columns are :: segment_actual_time ----->  747
The unique elements in the columns are :: segment_osrm_time ----->  214
The unique elements in the columns are :: segment_osrm_distance ----->  113799
```

In [17]:
```
df.isna().sum()
```

Out[17]:
```
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
actual_distance_to_destination  0
actual_time                     0
osrm_time                       0
osrm_distance                   0
segment_actual_time             0
segment_osrm_time               0
segment_osrm_distance           0
dtype: int64
```

In [18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 19 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   data                            144867 non-null  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  actual_distance_to_destination  144867 non-null  float64
 13  actual_time                     144867 non-null  float64
 14  osrm_time                       144867 non-null  float64
 15  osrm_distance                   144867 non-null  float64
 16  segment_actual_time             144867 non-null  float64
 17  segment_osrm_time               144867 non-null  float64
 18  segment_osrm_distance           144867 non-null  float64
dtypes: float64(8), object(11)
memory usage: 21.0+ MB
```

In [19]:
```
df.describe()
```

Out[19]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time	segment_osrm_time	segment
count	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	
mean	961.262986	234.073372	416.927527	213.868272	284.771297	36.196111	18.507548	
std	1037.012769	344.990009	598.103621	308.011085	421.119294	53.571158	14.775960	
min	20.000000	9.000045	9.000000	6.000000	9.008200	-244.000000	0.000000	
25%	161.000000	23.355874	51.000000	27.000000	29.914700	20.000000	11.000000	
50%	449.000000	66.126571	132.000000	64.000000	78.525800	29.000000	17.000000	
75%	1634.000000	286.708875	513.000000	257.000000	343.193250	40.000000	22.000000	
max	7898.000000	1927.447705	4532.000000	1686.000000	2326.199100	3051.000000	1611.000000	

In [21]:
```
df.describe(include='object')
```

Out[21]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destinat
count	144867	144867	144867	144867	144867	144867	144574	144867	
unique	2	14817	1504	2	14817	1508	1498	1481	
top	training	2018-09-28 05:23:15.359220	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	trip-153811219535896559	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND000000ACB	Gurgaon_Bi
freq	104858	101	1812	99660	101	23347	23347	15192	

In [22]:
```
df['data'] = df['data'].astype('category')
df['route_type'] = df['route_type'].astype('category')
```

In [23]:
```
df[:1]
```

Out[23]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_n
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu

In [24]:
```
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'])
```

Let's check for the Data start time and end time

In [25]:
```
df['trip_creation_time'].min(), df['trip_creation_time'].max()
```

Out[25]:
```
(Timestamp('2018-09-12 00:00:16.535741'),
 Timestamp('2018-10-03 23:59:42.701692'))
```

In [26]:
```
df['od_start_time'].min() , df['od_end_time'].max()
```

Out[26]:
```
(Timestamp('2018-09-12 00:00:16.535741'),
 Timestamp('2018-10-08 03:00:24.353479'))
```

Let's check for the Data start time and end time

In [27]:
```
df[:2]
```

Out[27]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_r
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu

In [28]:
```
df.isna().sum()
```

Out[28]:
```
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
actual_distance_to_destination  0
actual_time                     0
osrm_time                       0
osrm_distance                   0
segment_actual_time             0
segment_osrm_time               0
segment_osrm_distance           0
dtype: int64
```

In [29]:
```
df[['source_center','source_name']].isna().sum()
```

Out[29]:
```
source_center      0
source_name      293
dtype: int64
```

In [30]:
```
df[df['source_name'].isna()][:5]
```

Out[30]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	c
112	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	Jaipur_Hub (Rajasthan)	06
113	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	Jaipur_Hub (Rajasthan)	06
114	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	Jaipur_Hub (Rajasthan)	06
115	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	Jaipur_Hub (Rajasthan)	06
116	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	Jaipur_Hub (Rajasthan)	06

Let's check for the Data start time and end time

In [31]:
```
df[(df["source_name"].notnull()) & (df["source_center"].isin(df[df["source_name"].isnull()]["source_center"]))]
```

Out[31]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end

In [32]:
```
df[['destination_name','destination_center']].isna().sum()
```

Out[32]:
```
destination_name      261
destination_center      0
dtype: int64
```

Let's check for the Data start time and end time

In [33]:
```
[(df["destination_name"].notnull()) & (df["destination_center"].isin(df[df["destination_name"].isnull()]["destination_center"]))]
```

Out[33]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	od_end

Let's check for the Data start time and end time

In [34]:
```
df[df['segment_actual_time'] < 0]
```

Out[34]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	d
1805	training	2018-09-14 18:24:50.326548	thanos::sroute:21a9cdbc-469e-40e7-8fda-f4c02a1...	Carting	trip-153694949570782725	IND000000ACT	Delhi_Gateway_HB (Delhi)	IND000000ACB	Gurg
3761	training	2018-09-24 02:51:40.385840	thanos::sroute:f01c8bbd-655d-42ea-9abf-60d5040...	FTL	trip-153775750038541173	IND821115AAB	Sasaram_Central_I_2 (Bihar)	IND209304AAA	Ka
4040	test	2018-10-01 15:23:08.328308	thanos::sroute:749c5c42-2826-40c9-b0d3-5102cee...	Carting	trip-153840738832805377	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND110030AAD	Delhi
6329	training	2018-09-24 06:45:30.168184	thanos::sroute:b43ac2fc-3ec6-4a37-adf2-00b2561...	Carting	trip-153777153016792753	IND160002AAC	Chandigarh_Mehmdpur_H (Punjab)	IND140603AAA	Zira
39825	training	2018-09-23 22:33:30.538708	thanos::sroute:54a7c356-361d-4e74-9ee7-d420c37...	FTL	trip-153774201053841850	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND302014AAA	Jaipu
40942	training	2018-09-24 01:49:00.396529	thanos::sroute:e3e7c92f-55a9-4ecb-a4a7-919bbb7...	Carting	trip-153775374039614677	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurg
56464	training	2018-09-24 04:05:35.796451	thanos::sroute:870bead8-6c8a-458f-b4d8-658de44...	FTL	trip-153776193579607660	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND834002AAB	Ranchi
58697	training	2018-09-24 19:41:53.611094	thanos::sroute:cf2e63a5-87d6-4e39-a2d6-5555ed6...	FTL	trip-153781811361072511	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND501359AAE	Hyderal
68205	test	2018-10-03 22:28:20.454881	thanos::sroute:870bead8-6c8a-458f-b4d8-658de44...	FTL	trip-153860570045461434	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND834002AAB	Ranchi
70479	training	2018-09-14 09:54:59.693367	thanos::sroute:3f5f6285-256d-48ea-9a07-ec85cfe...	FTL	trip-153691889969297161	IND328001AAA	Dholpur_GtRoad_D (Rajasthan)	IND474003AAA	(
73603	training	2018-09-14 09:26:48.853918	thanos::sroute:f9784448-ffd3-4a4c-957f-b7fa68c...	FTL	trip-153691720885367295	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND683511AAA	
85042	training	2018-09-13 01:48:43.812361	thanos::sroute:2994b7ae-bb9c-4d7c-a30f-7882553...	FTL	trip-153680332381212407	IND421302AAG	Bhiwandi_Mankoli_HB (Maharashtra)	IND562132AAA	Banga
95433	training	2018-09-24 23:50:47.637174	thanos::sroute:54a7c356-361d-4e74-9ee7-d420c37...	FTL	trip-153783304763693286	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND302014AAA	Jaipu
97566	training	2018-09-24 18:18:00.311575	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	trip-153781308031132891	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562132AAA	Banga
100205	training	2018-09-17 08:05:40.886155	thanos::sroute:06b37046-b91b-4f4d-9aac-e799aa0...	Carting	trip-153717154088593859	IND792001AAA	Tezu_Farmnala_D (Arunachal Pradesh)	IND786001AAB	Dit
116173	training	2018-09-22 04:36:54.081516	thanos::sroute:fe14dda1-e713-451e-9bd3-d27c920...	Carting	trip-153759101408126548	IND382421AAA	Gandhinagar_DC (Gujarat)	IND382430AAB	Ahme
119377	training	2018-09-12 08:51:23.771627	thanos::sroute:6a74dc96-d3ef-4697-9ae8-8eab11c...	Carting	trip-153674228377135882	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000000ACB	Gurg
120153	test	2018-09-27 01:18:15.793512	thanos::sroute:1f3b54e4-3374-43fd-8ed1-198bb9a...	Carting	trip-153801109579320959	IND421401AAC	Murbad_SnkunDPP_D (Maharashtra)	IND421601AAB	Asang
125821	training	2018-09-24 03:07:59.175026	thanos::sroute:16f438c4-c258-4955-8358-bdb1e25...	FTL	trip-153775847917477411	IND421302AAG	Bhiwandi_Mankoli_HB (Maharashtra)	IND712311AAA	Kol
131578	training	2018-09-22 22:41:23.003019	thanos::sroute:c89bcbd0-26ee-4dc1-b9e7-cea1609...	Carting	trip-153765608300279258	IND721212AAA	Ghatal_KrshDPP_D (West Bengal)	IND712310AAE	Kolkata
142409	training	2018-09-24 10:08:09.819893	thanos::sroute:18104c25-811c-4b93-8c92-8c6923c...	FTL	trip-153778368981951440	IND421302AAG	Bhiwandi_Mankoli_HB (Maharashtra)	IND562132AAA	Banga

In [35]:
```
df.drop(df[df['segment_actual_time'] < 0].index, inplace=True)
```

In [36]:
```
df.describe()
```

Out[36]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time	segment_osrm_time	segment
count	144846.000000	144846.000000	144846.000000	144846.000000	144846.000000	144846.000000	144846.000000	
mean	961.226537	234.057171	416.908724	213.853002	284.750969	36.207427	18.507304	
std	1036.993595	344.974984	598.085058	307.997702	421.101831	53.561259	14.775870	
min	20.000000	9.000045	9.000000	6.000000	9.008200	0.000000	0.000000	
25%	161.000000	23.354927	51.000000	27.000000	29.909925	20.000000	11.000000	
50%	449.000000	66.126234	132.000000	64.000000	78.524600	29.000000	17.000000	
75%	1634.000000	286.706673	513.000000	257.000000	343.062075	40.000000	22.000000	
max	7898.000000	1927.447705	4532.000000	1686.000000	2326.199100	3051.000000	1611.000000	

Let's check for the Data start time and end time

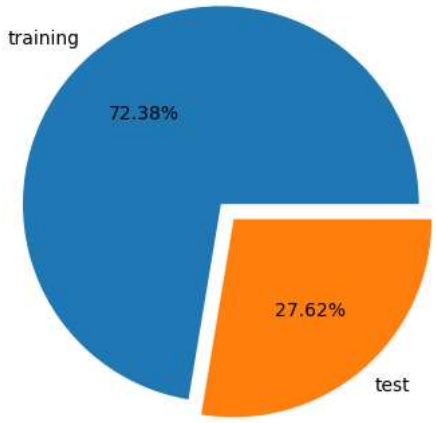
In [37]:
```
df['data'].unique()
```

Out[37]:
```
['training', 'test']
Categories (2, object): ['test', 'training']
```

In [38]:
```
value = df['data'].value_counts()
value
```

Out[38]:
```
training    104840
test         40006
Name: data, dtype: int64
```

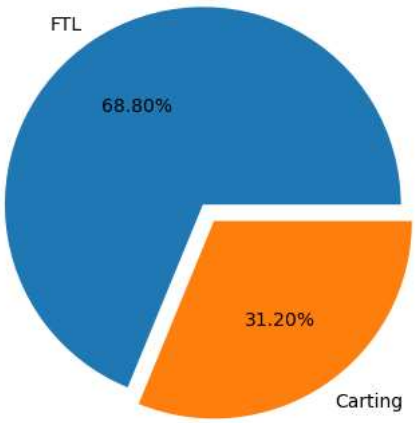
In [39]:
```
plt.pie(value, labels = ['training', 'test'],
        explode = [0, 0.1],
       autopct = '%.2f%%')
plt.show()
```



In [40]:
```
Route_type = df['route_type'].value_counts(normalize=False)
Route_type
```

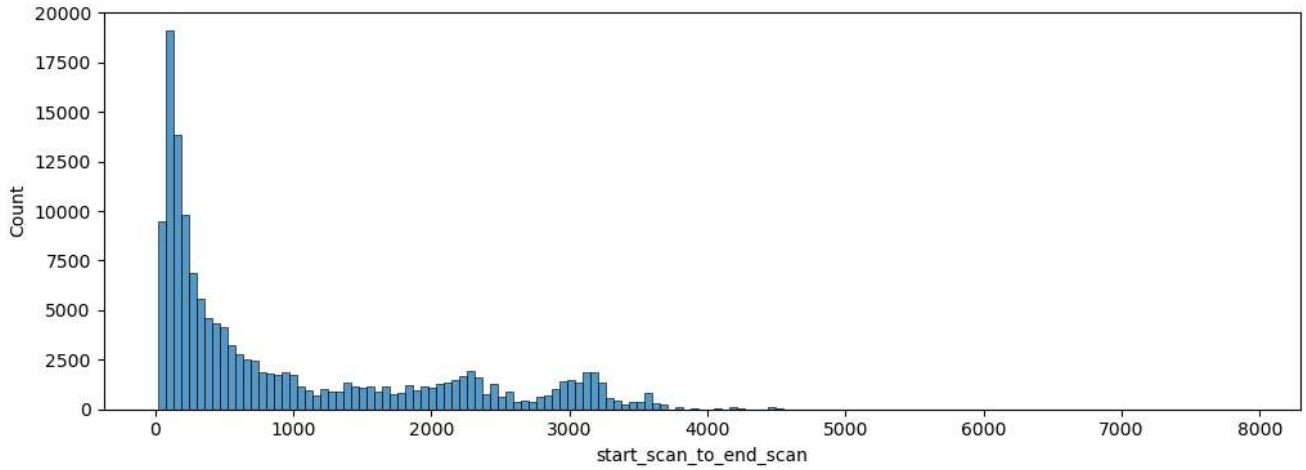
Out[40]:
```
FTL        99648
Carting    45198
Name: route_type, dtype: int64
```

In [41]:
```
plt.pie(Route_type, labels = ['FTL', 'Carting'],
        explode = [0, 0.1],
       autopct = '%.2f%%')
plt.show()
```

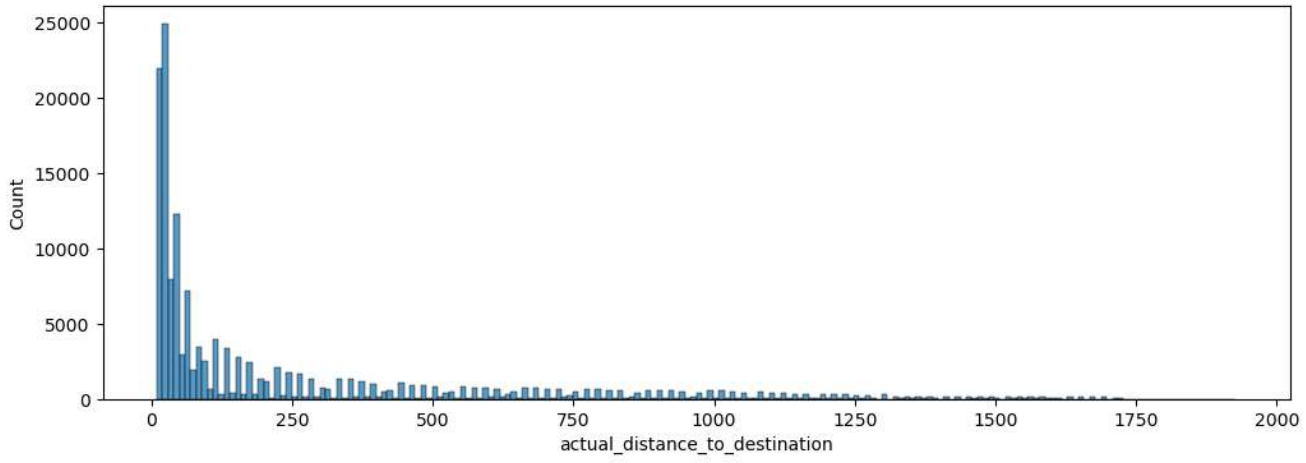


Let's check for the Data start time and end time

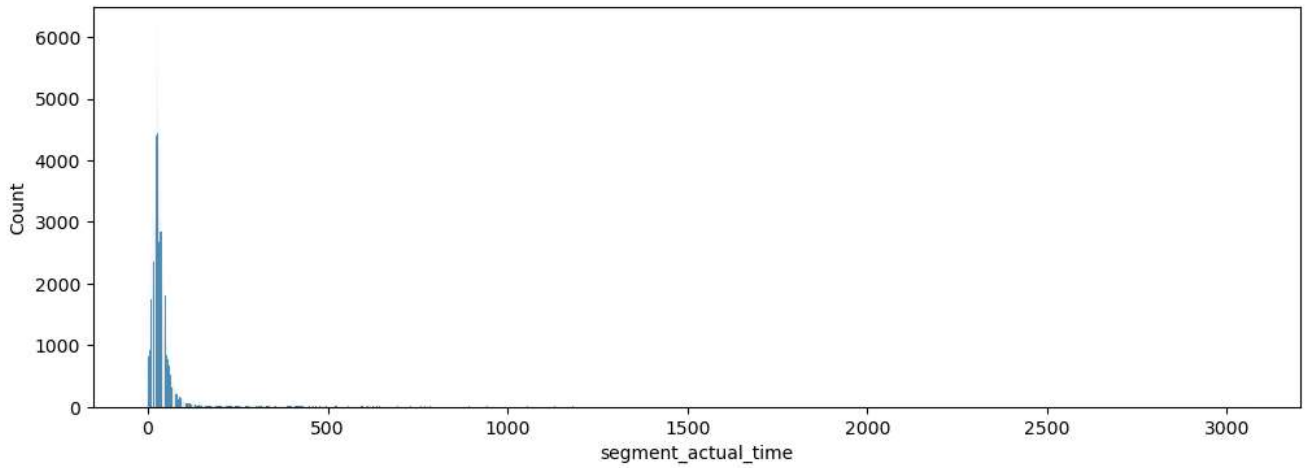
In [42]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['start_scan_to_end_scan'])
plt.show()
```



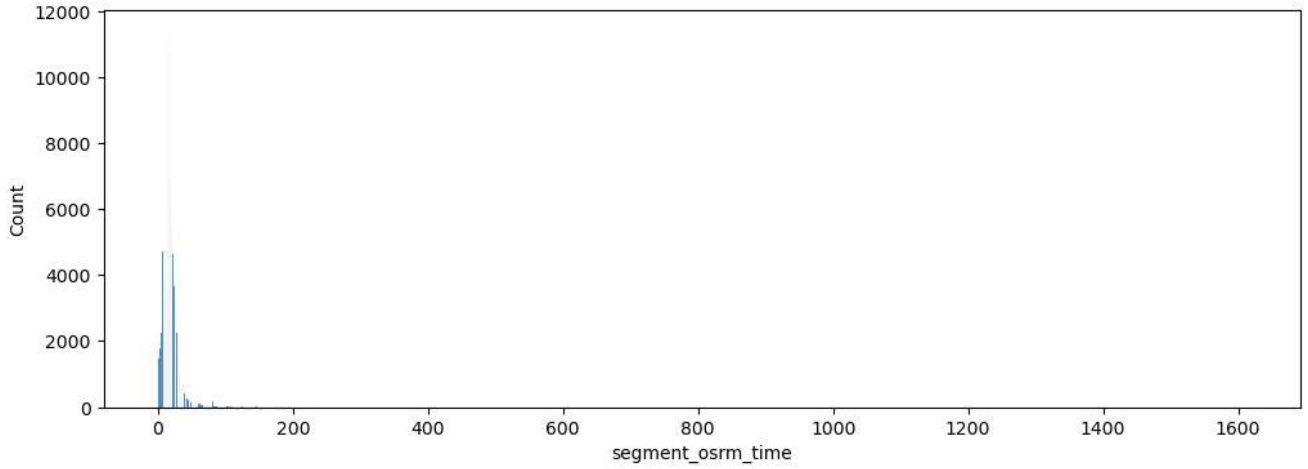
In [43]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['actual_distance_to_destination'])
plt.show()
```



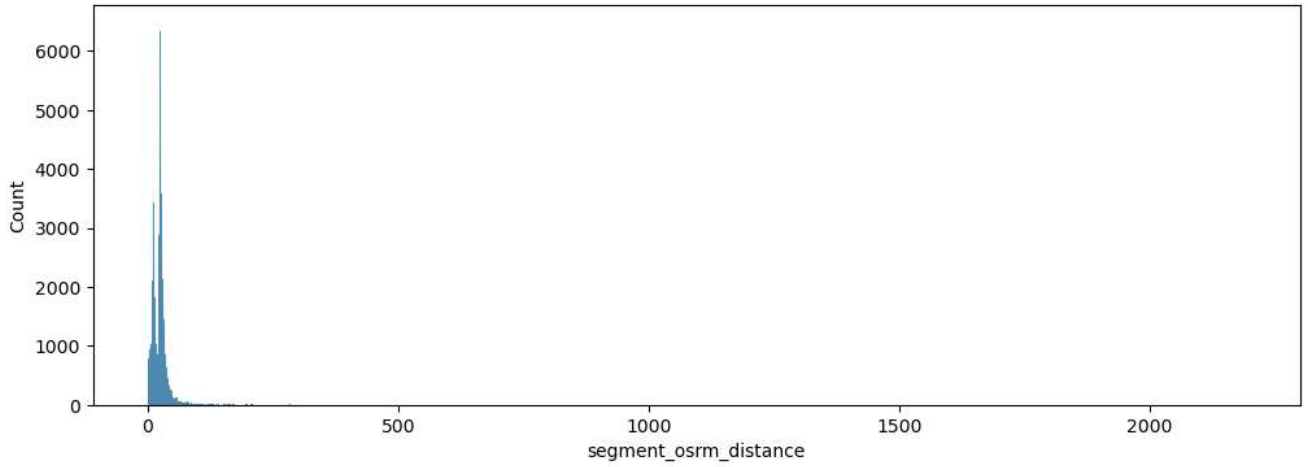
In [46]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['segment_actual_time'])
plt.show()
```



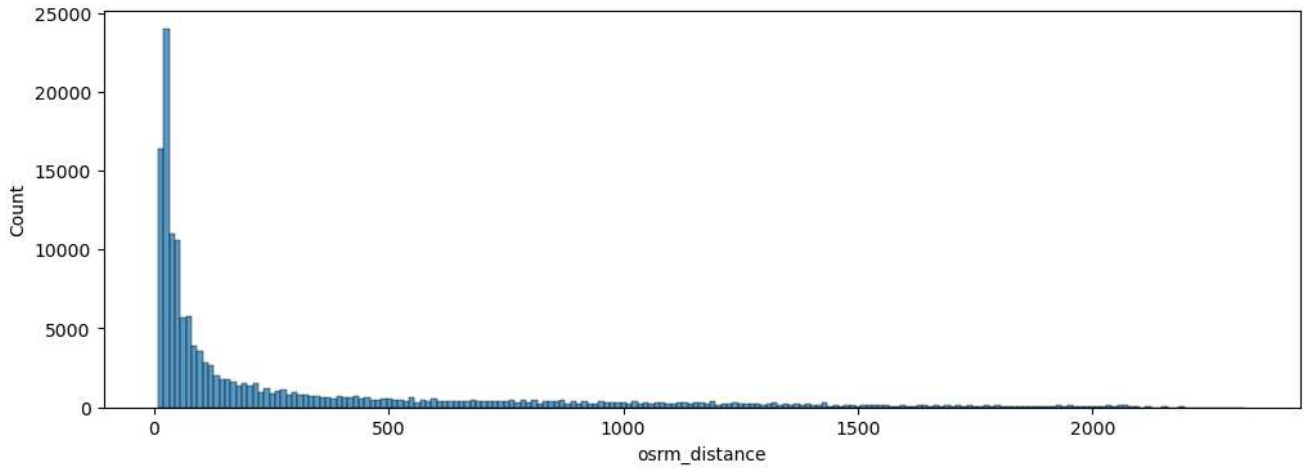
In [47]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['segment_osrm_time'])
plt.show()
```



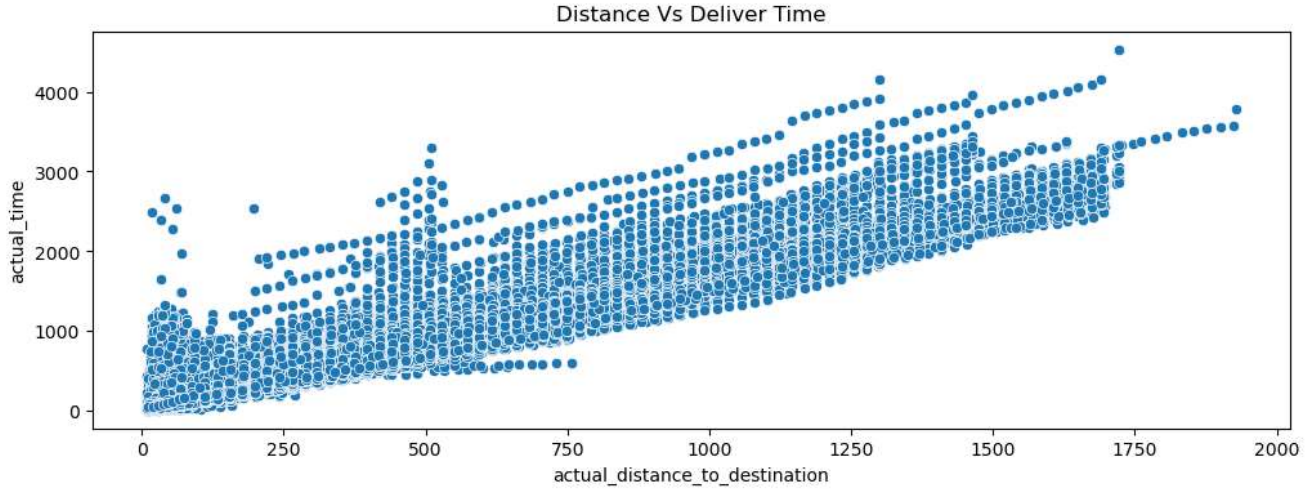
In [48]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['segment_osrm_distance'])
plt.show()
```



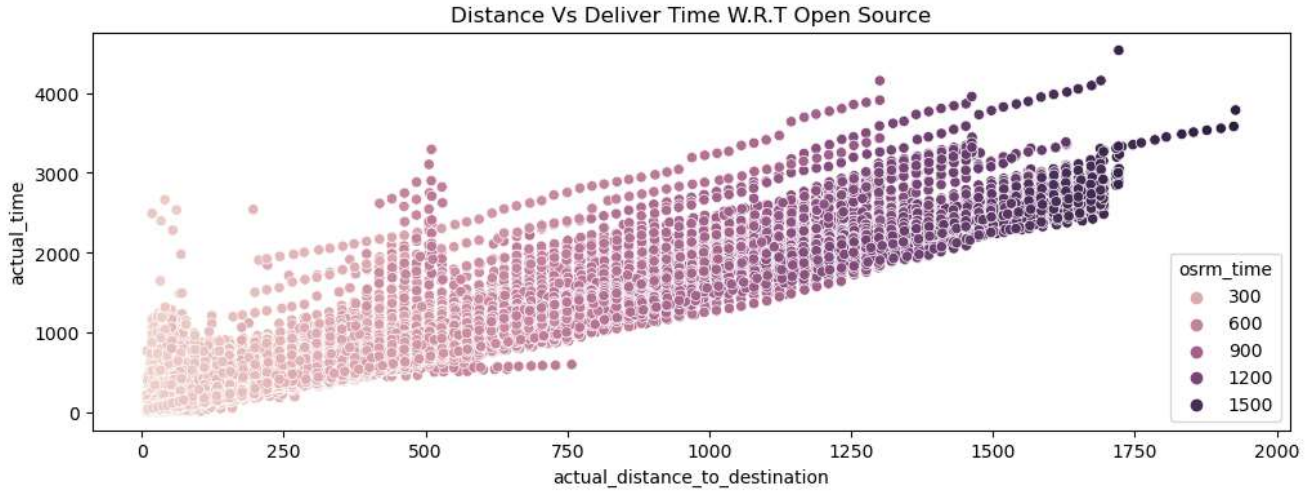
In [49]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['osrm_distance'])
plt.show()
```



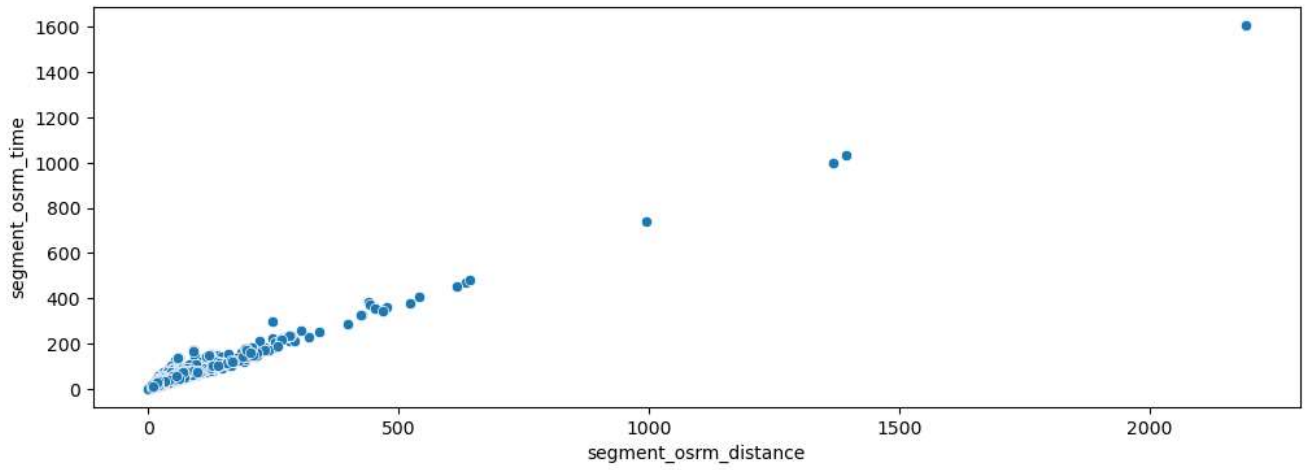
In [50]:
```
plt.figure(figsize=(12,4))
sns.scatterplot(data=df, x="actual_distance_to_destination", y="actual_time")
plt.title("Distance Vs Deliver Time")
plt.show()
```



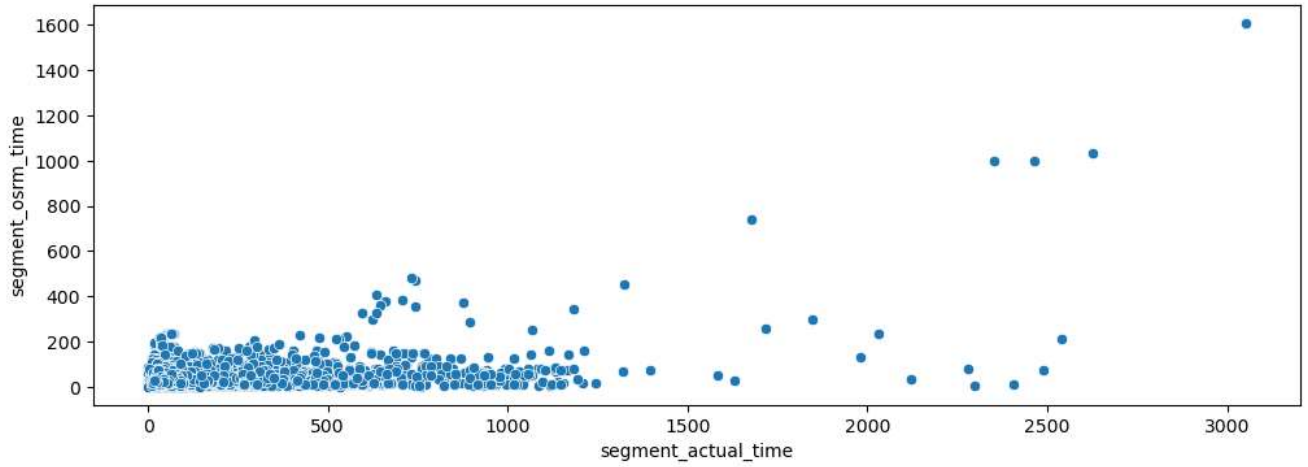
In [51]:
```
plt.figure(figsize=(12,4))
sns.scatterplot(data=df, x="actual_distance_to_destination", y="actual_time", hue="osrm_time")
plt.title("Distance Vs Deliver Time W.R.T Open Source")
plt.show()
```



In [52]:
```
plt.figure(figsize=(12,4))
sns.scatterplot(data=df, x="segment_osrm_distance", y="segment_osrm_time")
plt.show()
```



In [53]:
```
plt.figure(figsize=(12,4))
sns.scatterplot(data=df, x="segment_actual_time", y="segment_osrm_time")
plt.show()
```



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

```
C:\Users\siddh\AppData\Local\Temp\ipykernel_28396\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  df.corr()
```

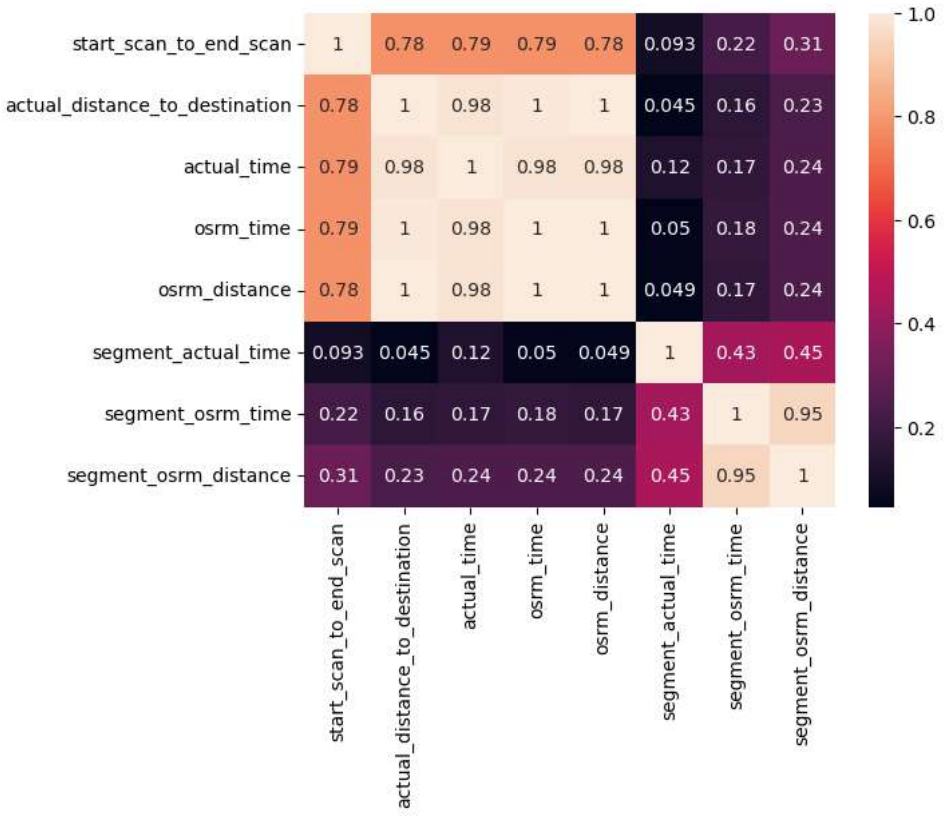
Out[54]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time	segment_os
start_scan_to_end_scan	1.000000	0.784988	0.785924	0.785283	0.784120	0.093372	
actual_distance_to_destination	0.784988	1.000000	0.978658	0.995872	0.997148	0.045320	
actual_time	0.785924	0.978658	1.000000	0.977996	0.979398	0.124483	
osrm_time	0.785283	0.995872	0.977996	1.000000	0.999119	0.049977	
osrm_distance	0.784120	0.997148	0.979398	0.999119	1.000000	0.048787	
segment_actual_time	0.093372	0.045320	0.124483	0.049977	0.048787	1.000000	
segment_osrm_time	0.219844	0.158836	0.171480	0.177074	0.169157	0.433604	
segment_osrm_distance	0.306972	0.232119	0.242296	0.242288	0.239672	0.449167	

In [55]:
```
sns.heatmap(df.corr(), annot=True)
```

```
C:\Users\siddh\AppData\Local\Temp\ipykernel_28396\621126171.py:1: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  sns.heatmap(df.corr(), annot=True)
```

Out[55]: <Axes: >



Feature Engineering

In [56]:
```
df['od_time_diff_hour'] = df['od_end_time'] - df['od_start_time']
df.drop(columns={'od_start_time','od_end_time'}, inplace=True)
```

In [57]:
```
df['od_time_diff_hour'] = df['od_time_diff_hour'].dt.total_seconds()/60
df['od_time_diff_hour']
```

Out[57]:
```
0          86.213637
1          86.213637
2          86.213637
3          86.213637
4          86.213637
             ...
144862    427.686364
144863    427.686364
144864    427.686364
144865    427.686364
144866    427.686364
Name: od_time_diff_hour, Length: 144846, dtype: float64
```

In [58]:
```
df['destination_city'] = df['destination_name'].str.split("_", expand= True)[0]
```

In [59]:
```
df['destination_city'].head()
```

Out[59]:
```
0    Khambhat
1    Khambhat
2    Khambhat
3    Khambhat
4    Khambhat
Name: destination_city, dtype: object
```

In [60]:
```
df['destination_place'] = df['destination_name'].str.split("_", expand=True)[1]
```

In [61]:
```
df['destination_place'].head()
```

Out[61]:
```
0    MotvdDPP
1    MotvdDPP
2    MotvdDPP
3    MotvdDPP
4    MotvdDPP
Name: destination_place, dtype: object
```

In [62]:
```
df['destination_state'] = df['destination_name'].str.split("_", expand=True)[2].str.split(" ", expand= True)[1]
```

In [63]:
```
df['destination_state'] = df['destination_state'].str.replace("(", "").str.replace(")", "")
```

```
C:\Users\siddh\AppData\Local\Temp\ipykernel_28396\2165789120.py:1: FutureWarning: The default value of regex will change from T
rue to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings wh
en regex=True.
  df['destination_state'] = df['destination_state'].str.replace("(", "").str.replace(")", "")
```

In [64]:
```
df['destination_state']
```

Out[64]:
```
0         Gujarat
1         Gujarat
2         Gujarat
3         Gujarat
4         Gujarat
           ...
144862    Haryana
144863    Haryana
144864    Haryana
144865    Haryana
144866    Haryana
Name: destination_state, Length: 144846, dtype: object
```

In [65]:
```
df['trip_year'] = df['trip_creation_time'].dt.year
df['trip_year']
```

Out[65]:
```
0         2018
1         2018
2         2018
3         2018
4         2018
          ...
144862    2018
144863    2018
144864    2018
144865    2018
144866    2018
Name: trip_year, Length: 144846, dtype: int64
```

In [66]:
```
df['trip_month'] = df['trip_creation_time'].dt.month
df['trip_month']
```

Out[66]:
```
0         9
1         9
2         9
3         9
4         9
         ..
144862    9
144863    9
144864    9
144865    9
144866    9
Name: trip_month, Length: 144846, dtype: int64
```

In [67]:
```
df['trip_day'] = df['trip_creation_time'].dt.day
df['trip_day']
```

Out[67]:
```
0         20
1         20
2         20
3         20
4         20
          ..
144862    20
144863    20
144864    20
144865    20
144866    20
Name: trip_day, Length: 144846, dtype: int64
```

In [68]:
```
state'] = df['source_name'].str.split("_", expand=True)[2].str.split(" ", expand=True)[1].str.replace("(","").str.replace(")","")
```

```
C:\Users\siddh\AppData\Local\Temp\ipykernel_28396\726971721.py:1: FutureWarning: The default value of regex will change from Tr
ue to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings whe
n regex=True.
  df['source_state'] = df['source_name'].str.split("_", expand=True)[2].str.split(" ", expand=True)[1].str.replace("(","").str.
replace(")","")
```

In [69]:
```
df['source_state']
```

Out[69]:
```
0         Gujarat
1         Gujarat
2         Gujarat
3         Gujarat
4         Gujarat
           ...
144862    Haryana
144863    Haryana
144864    Haryana
144865    Haryana
144866    Haryana
Name: source_state, Length: 144846, dtype: object
```

In [70]:
```
df['source_city'] =  df['source_name'].str.split("_", expand=True)[0]
```

In [71]:
```
df['source_city'].head()
```

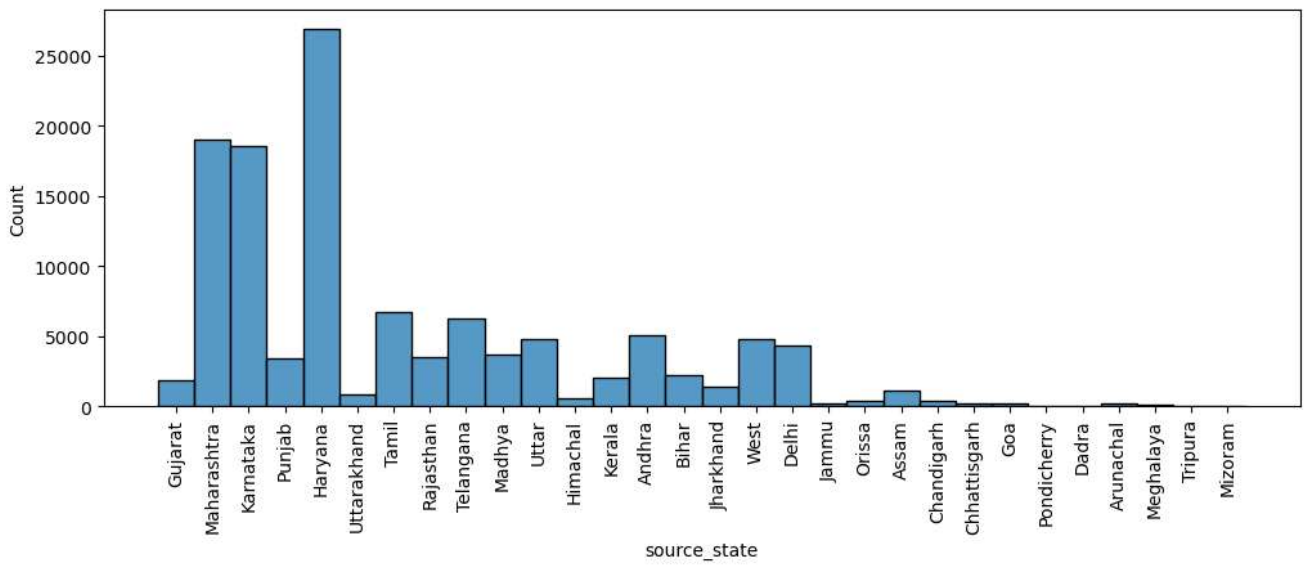
Out[71]:
```
0    Anand
1    Anand
2    Anand
3    Anand
4    Anand
Name: source_city, dtype: object
```

In [72]:
```
df['source_place'] = df['source_name'].str.split("_", expand =True)[1]
```

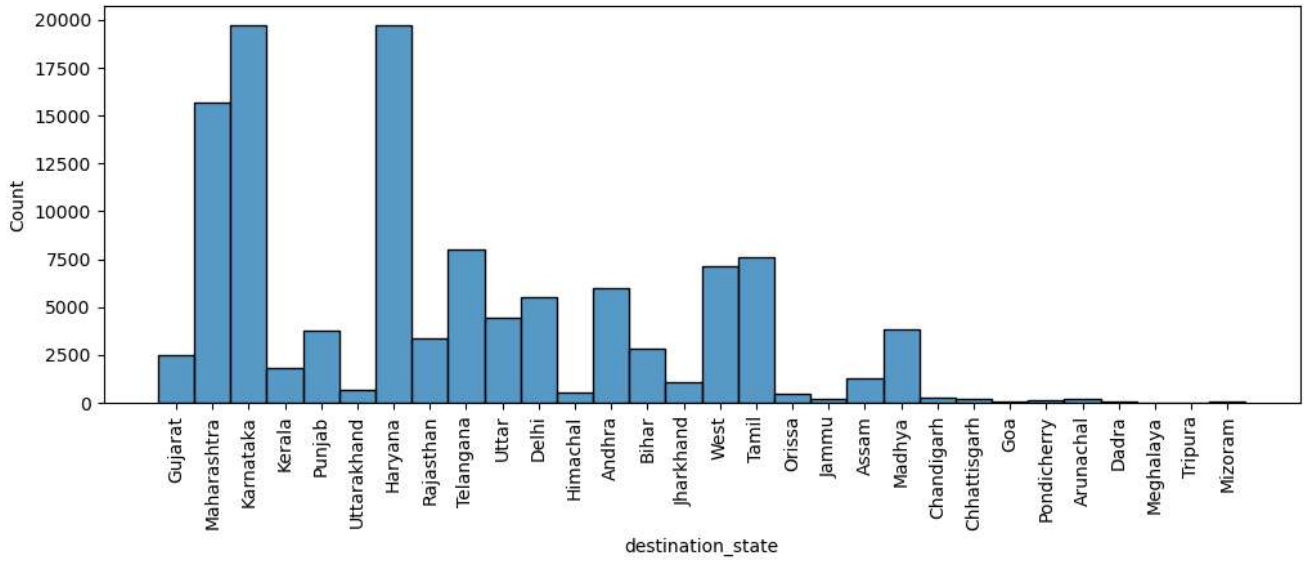
In [73]:
```
df['source_place'].head()
```

Out[73]:
```
0    VUNagar
1    VUNagar
2    VUNagar
3    VUNagar
4    VUNagar
Name: source_place, dtype: object
```

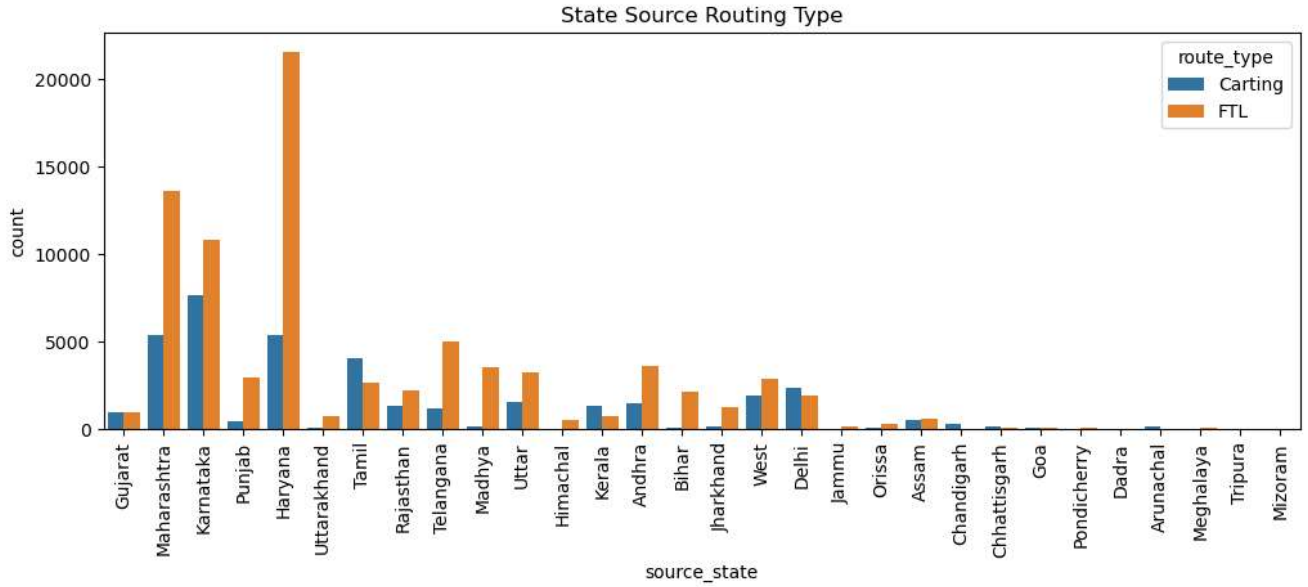
In [74]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['source_state'])
plt.xticks(rotation = 90)
plt.show()
```



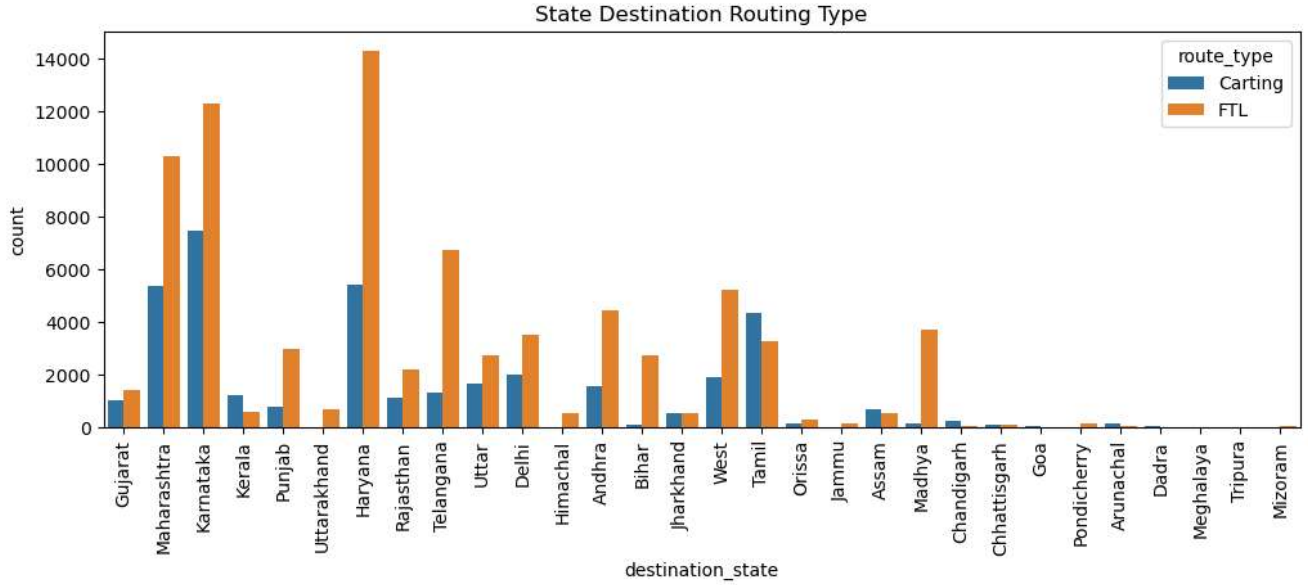
In [75]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['destination_state'])
plt.xticks(rotation = 90)
plt.show()
```



In [76]:
```
plt.figure(figsize=(12,4))
sns.countplot(data=df, x = df['source_state'], hue='route_type')
plt.xticks(rotation = 90)
plt.title("State Source Routing Type")
plt.show()
```



In [77]:
```
plt.figure(figsize=(12,4))
sns.countplot(data=df, x = df['destination_state'], hue='route_type')
plt.xticks(rotation = 90)
plt.title("State Destination Routing Type")
plt.show()
```



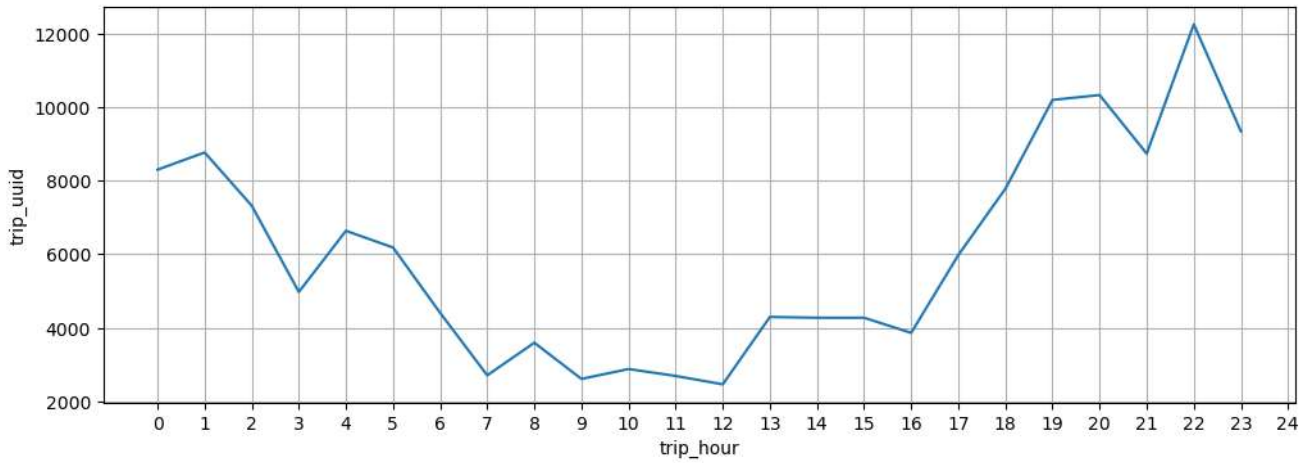
In [78]:
```
df['trip_hour'] = df['trip_creation_time'].dt.hour
```

In [79]:
```
hour_trips = df.groupby(by='trip_hour')['trip_uuid'].count().reset_index()
hour_trips.head()
```

Out[79]:

	trip_hour	trip_uuid
0	0	8299
1	1	8768
2	2	7320
3	3	4975
4	4	6637

In [80]:
```
plt.figure(figsize=(12,4))
sns.lineplot(data=hour_trips, x='trip_hour', y="trip_uuid")
plt.grid('both')
plt.xticks(np.arange(0,25))
plt.show()
```



In [81]:
```
df['route_schedule'] = df['route_schedule_uuid'].str.split(":", expand=True)[0]
```

In [82]:
```
df['route_schedule']
```

Out[82]:
```
0         thanos
1         thanos
2         thanos
3         thanos
4         thanos
           ...
144862    thanos
144863    thanos
144864    thanos
144865    thanos
144866    thanos
Name: route_schedule, Length: 144846, dtype: object
```

In [83]:
```
df[:2]
```

Out[83]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_r
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu

2 rows × 29 columns

In [84]:
```
df.nunique()
```

Out[84]:
```
data                                2
trip_creation_time              14817
route_schedule_uuid              1504
route_type                          2
trip_uuid                       14817
source_center                    1508
source_name                      1498
destination_center               1481
destination_name                 1468
start_scan_to_end_scan           1915
actual_distance_to_destination  144494
actual_time                      3182
osrm_time                        1531
osrm_distance                  138026
segment_actual_time               729
segment_osrm_time                 214
segment_osrm_distance          113782
od_time_diff_hour               26369
destination_city                 1258
destination_place                1154
destination_state                  30
trip_year                           1
trip_month                          2
trip_day                           22
source_state                       30
source_city                      1262
source_place                     1178
trip_hour                          24
route_schedule                      1
dtype: int64
```

In [85]:
```
df['source_state'].value_counts()
```

Out[85]:
```
Haryana          26935
Maharashtra      18993
Karnataka        18531
Tamil             6697
Telangana         6212
Andhra            5087
Uttar             4791
West              4742
Delhi             4317
Madhya            3687
Rajasthan         3529
Punjab            3380
Bihar             2206
Kerala            2038
Gujarat           1867
Jharkhand         1343
Assam             1126
Uttarakhand        827
Himachal           532
Orissa             404
Chandigarh         367
Chhattisgarh       229
Jammu              182
Goa                165
Arunachal          150
Meghalaya           77
Pondicherry         49
Dadra               30
Mizoram             26
Tripura              5
Name: source_state, dtype: int64
```

In [86]:
```
df['destination_state'].value_counts()
```

Out[86]:
```
Karnataka       19750
Haryana         19744
Maharashtra     15682
Telangana        8032
Tamil            7610
West             7110
Andhra           5984
Delhi            5490
Uttar            4413
Madhya           3852
Punjab           3751
Rajasthan        3335
Bihar            2809
Gujarat          2456
Kerala           1834
Assam            1249
Jharkhand        1076
Uttarakhand       666
Himachal          543
Orissa            453
Chandigarh        282
Chhattisgarh      221
Arunachal         185
Jammu             167
Pondicherry       154
Goa                74
Dadra              34
Mizoram            31
Tripura             9
Meghalaya           8
Name: destination_state, dtype: int64
```

In [87]:
```
df['source_name'].value_counts()[:5]
```

Out[87]:
```
Gurgaon_Bilaspur_HB (Haryana)        23342
Bangalore_Nelmngla_H (Karnataka)      9972
Bhiwandi_Mankoli_HB (Maharashtra)     9085
Pune_Tathawde_H (Maharashtra)         4061
Hyderabad_Shamshbd_H (Telangana)      3340
Name: source_name, dtype: int64
```

In [89]:
```
df['destination_name'].value_counts()[:5]
```

Out[89]:
```
Gurgaon_Bilaspur_HB (Haryana)        15189
Bangalore_Nelmngla_H (Karnataka)     11016
Bhiwandi_Mankoli_HB (Maharashtra)     5492
Hyderabad_Shamshbd_H (Telangana)      5141
Kolkata_Dankuni_HB (West Bengal)      4891
Name: destination_name, dtype: int64
```

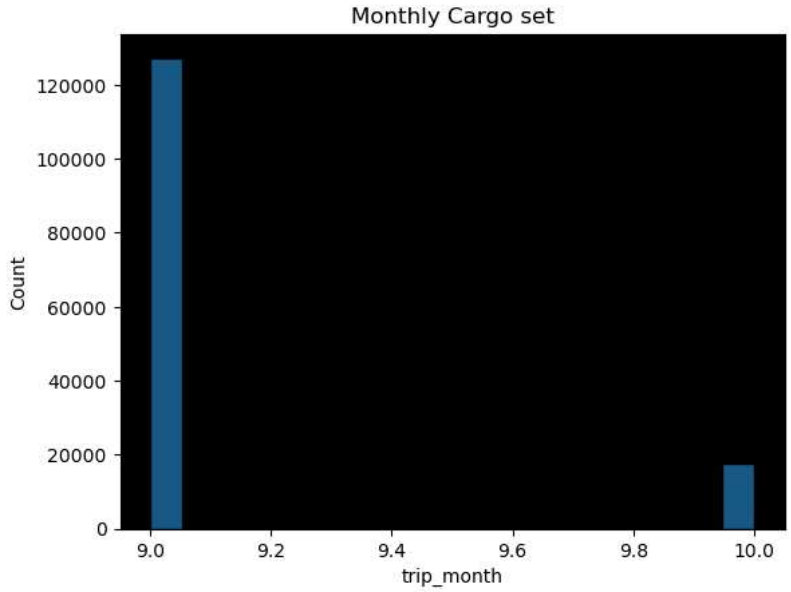
Source and Destination

In [90]:
```
df["source_destination"]=df["source_name"] + df["destination_name"]
```

In [91]:
```
df['source_destination'][:4]
```

Out[91]:
```
0    Anand_VUNagar_DC (Gujarat)Khambhat_MotvdDPP_D ...
1    Anand_VUNagar_DC (Gujarat)Khambhat_MotvdDPP_D ...
2    Anand_VUNagar_DC (Gujarat)Khambhat_MotvdDPP_D ...
3    Anand_VUNagar_DC (Gujarat)Khambhat_MotvdDPP_D ...
Name: source_destination, dtype: object
```

In [92]:
```
sns.histplot(df['trip_month'])
plt.title("Monthly Cargo set")
ax = plt.gca()
ax.set_facecolor("black")
plt.show()
```



Top 10 Most business Places from the source where the cart and FTL took places from the source to destination

In [93]:
```
df['source_place'].value_counts()[:10]
```

Out[93]:
```
Bilaspur         23459
Nelmngla         10050
Mankoli           9085
Central           8988
Tathawde          4061
Shamshbd          3340
Dankuni           2612
Mehmdpur          2449
HUB (Gujarat)     2189
Airport           2013
Name: source_place, dtype: int64
```

Bussiest Corridor

In [94]:
```
df['source_destination'].value_counts()[:5]
```

Out[94]:
```
Gurgaon_Bilaspur_HB (Haryana)Bangalore_Nelmngla_H (Karnataka)     4975
Bangalore_Nelmngla_H (Karnataka)Gurgaon_Bilaspur_HB (Haryana)     3316
Gurgaon_Bilaspur_HB (Haryana)Kolkata_Dankuni_HB (West Bengal)     2862
Gurgaon_Bilaspur_HB (Haryana)Hyderabad_Shamshbd_H (Telangana)     1638
Gurgaon_Bilaspur_HB (Haryana)Bhiwandi_Mankoli_HB (Maharashtra)    1617
Name: source_destination, dtype: int64
```

Least Bussiest Corridor

In [95]:
```
df['source_destination'].value_counts(ascending=True)[:5]
```

Out[95]:
```
Malerkotla_DC (Punjab)Dhuri_DMComDPP_D (Punjab)                          1
Daurala_Sardhnrd_D (Uttar Pradesh)Khatauli_TilakNgr_D (Uttar Pradesh)    1
Salem_Kadtmpty_H (Tamil Nadu)Salem_Kadtmpty_D (Tamil Nadu)               1
Kayamkulam_Bhrnikvu_D (Kerala)Manthuka_Central_D_1 (Kerala)              1
Mumbai_East_I_21 (Maharashtra)Mumbai_Kalyan (Maharashtra)                1
Name: source_destination, dtype: int64
```

Average Distance between Bussiest Corridor

In [96]:
```
source_destination'] == "Gurgaon_Bilaspur_HB (Haryana)Bangalore_Nelmngla_H (Karnataka)"]['actual_distance_to_destination'].mean()
```

Out[96]:
```
859.7926156621767
```

Average Distance between Least Bussiest Corridor

In [97]:
```
df[df['source_destination'] == "Malerkotla_DC (Punjab)Dhuri_DMComDPP_D (Punjab)"]["actual_distance_to_destination"].mean()
```

Out[97]:
```
17.100289404993177
```

Average Time Between the Bussiest Corridor

In [98]:
```
df[df['source_destination'] == "Gurgaon_Bilaspur_HB (Haryana)Bangalore_Nelmngla_H (Karnataka)"]['actual_time'].mean()
```

Out[98]:
```
1367.1736683417084
```

Average Time Between the Least Bussiest Corridor

In [99]:
```
df[df['source_destination'] == "Malerkotla_DC (Punjab)Dhuri_DMComDPP_D (Punjab)"]["actual_time"].mean()
```

Out[99]:
```
36.0
```

Data Wrangling

```
Grouping by segment || In- Depth Analysis
a. Create a unique identifier for different segments of a trip based on the combination of the trip_uuid, source_center,
and destination_center and name it as segment_key.
```

In [101]:
```
df.head()
```

Out[101]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_r
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDF (Gu

5 rows × 30 columns

In [102]:
```
df.columns
```

Out[102]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'start_scan_to_end_scan',
       'actual_distance_to_destination', 'actual_time', 'osrm_time',
       'osrm_distance', 'segment_actual_time', 'segment_osrm_time',
       'segment_osrm_distance', 'od_time_diff_hour', 'destination_city',
       'destination_place', 'destination_state', 'trip_year', 'trip_month',
       'trip_day', 'source_state', 'source_city', 'source_place', 'trip_hour',
       'route_schedule', 'source_destination'],
      dtype='object')
```

In [103]:
```
grouping_1 = ['trip_uuid', 'source_center', 'destination_center']
df1 = df.groupby(by = grouping_1, as_index = False).agg({'data' : 'first',
                                                         'route_type' : 'first',
                                                        'trip_creation_time' : 'first',
                                                        'source_name' : 'first',
                                                        'destination_name' : 'last',
                                                        'start_scan_to_end_scan' : 'first',
                                                        'actual_distance_to_destination' : 'last',
                                                        'actual_time' : 'last',
                                                        'osrm_time' : 'last',
                                                        'osrm_distance' : 'last',
                                                        'segment_actual_time' : 'sum',
                                                        'segment_osrm_time' : 'sum',
                                                        'segment_osrm_distance' : 'sum'
                                                         })
```

In [104]:
```
df2 = df1.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',
                                                           'start_scan_to_end_scan' : 'sum',
                                                           'actual_distance_to_destination' : 'sum',
                                                           'osrm_time' : 'sum',
                                                           'osrm_distance' : 'sum',
                                                           'segment_actual_time' : 'sum',
                                                           'segment_osrm_time' : 'sum',
                                                           'segment_osrm_distance' : 'sum'})
```

In [105]:
```
df2.shape
```

Out[105]:
```
(14817, 15)
```

In [106]:
```
df2.describe().T
```

Out[106]:

	count	mean	std	min	25%	50%	75%	max
start_scan_to_end_scan	14817.0	530.810016	658.705957	23.000000	149.000000	280.000000	637.000000	7898.000000
actual_distance_to_destination	14817.0	164.477838	305.388147	9.002461	22.837239	48.474072	164.583208	2186.531787
osrm_time	14817.0	161.384018	271.360995	6.000000	29.000000	60.000000	168.000000	2032.000000
osrm_distance	14817.0	204.344689	370.395573	9.072900	30.819200	65.618800	208.475000	2840.081000
segment_actual_time	14817.0	353.951610	556.320988	9.000000	66.000000	147.000000	367.000000	6230.000000
segment_osrm_time	14817.0	180.921172	314.485624	6.000000	31.000000	65.000000	185.000000	2564.000000
segment_osrm_distance	14817.0	223.164000	416.547252	9.072900	32.654500	70.154400	218.710200	3523.632400

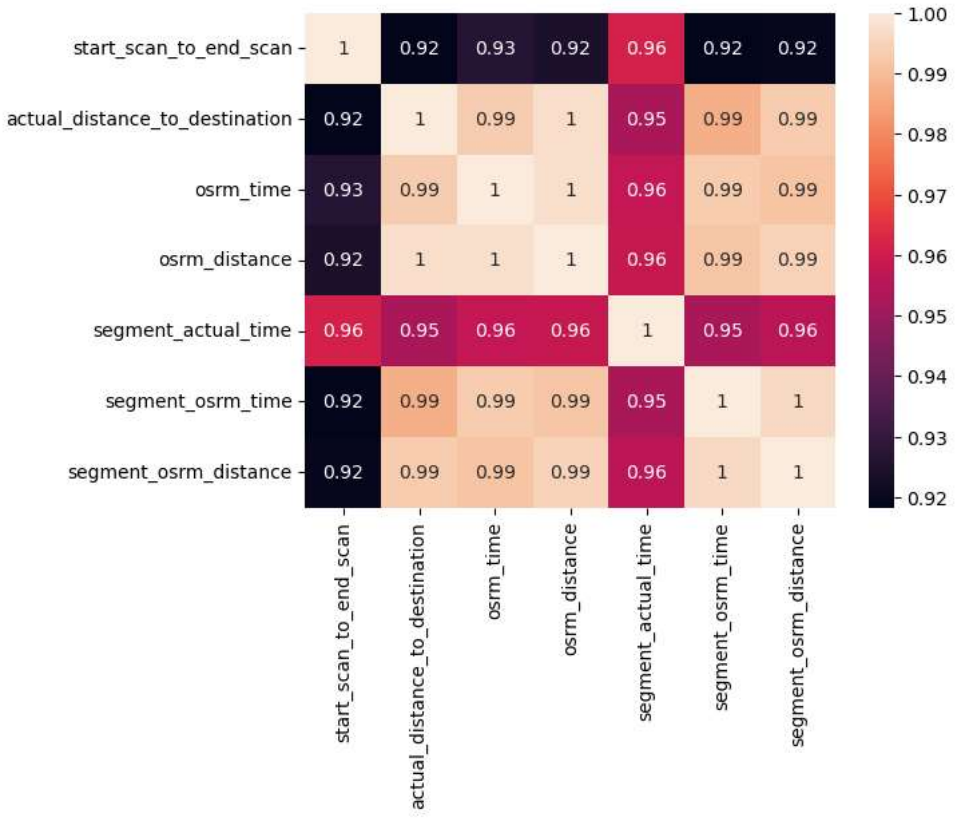
In [107]:
```
df2.isna().sum()
```

Out[107]:
```
trip_uuid                         0
source_center                     0
destination_center                0
data                              0
route_type                        0
trip_creation_time                0
source_name                      10
destination_name                  8
start_scan_to_end_scan            0
actual_distance_to_destination    0
osrm_time                         0
osrm_distance                     0
segment_actual_time               0
segment_osrm_time                 0
segment_osrm_distance             0
dtype: int64
```

In [108]:
```
sns.heatmap(df2.corr(), annot=True)
```

```
C:\Users\siddh\AppData\Local\Temp\ipykernel_28396\1738403036.py:1: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  sns.heatmap(df2.corr(), annot=True)
```

Out[108]: <Axes: >



In [109]:
```
df2.head()
```

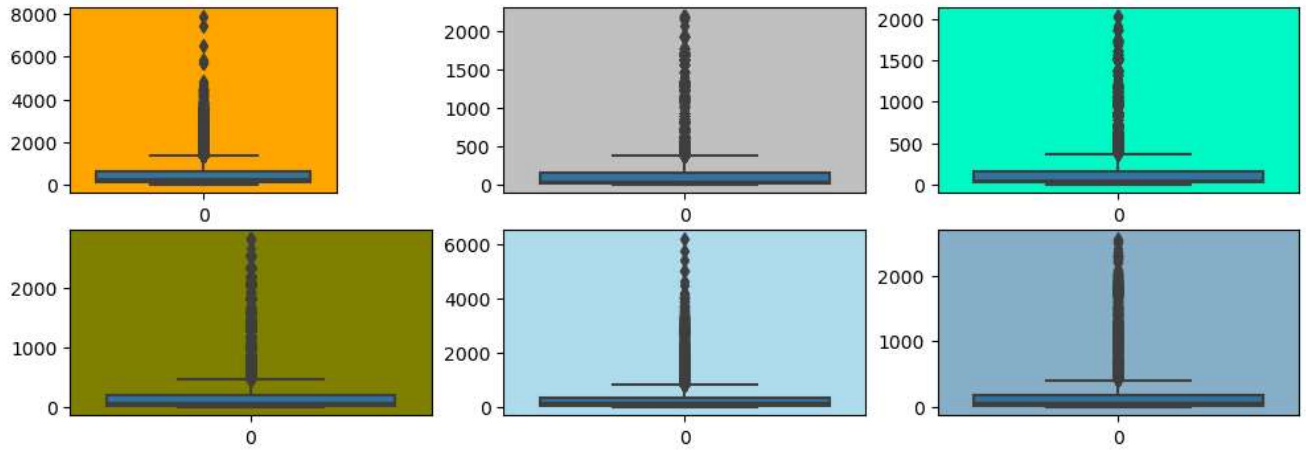
Out[109]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	DoddaBlpur_ChikaDPP_D (Karnataka)	DoddaBlpur_ChikaDPP_D (Karnataka)	
2	trip-153671043369099517	IND000000ACB	IND000000ACB	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur_HB (Haryana)	Gurgaon_Bilaspur_HB (Haryana)	
3	trip-153671046011330457	IND400072AAB	IND401104AAA	training	Carting	2018-09-12 00:01:00.113710	Mumbai Hub (Maharashtra)	Mumbai_MiraRd_IP (Maharashtra)	
4	trip-153671052974046625	IND583101AAA	IND583119AAA	training	FTL	2018-09-12 00:02:09.740725	Bellary_Dc (Karnataka)	Sandur_WrdN1DPP_D (Karnataka)	

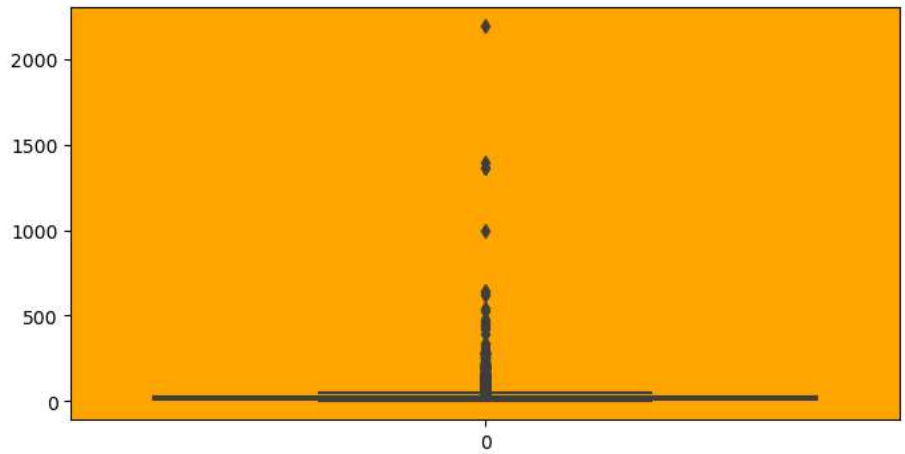
In [ ]:
```
```

In [110]:
```
numerical_columns = ['start_scan_to_end_scan', 'actual_distance_to_destination',
                     'osrm_time', 'osrm_distance', 'segment_actual_time',
                     'segment_osrm_time', 'segment_osrm_distance']
```

In [111]:
```
plt.figure(figsize=(12,4))
plt.subplot(2,4,1)
sns.boxplot(df2['start_scan_to_end_scan'])
ax=plt.gca()
ax.set_facecolor('orange')
plt.subplot(2,3,2)
sns.boxplot(df2['actual_distance_to_destination'])
ax=plt.gca()
ax.set_facecolor('silver')
plt.subplot(2,3,3)
sns.boxplot(df2['osrm_time'])
ax=plt.gca()
ax.set_facecolor('#01F9C6')
plt.subplot(2,3,4)
sns.boxplot(df2['osrm_distance'])
ax=plt.gca()
ax.set_facecolor('#808000')
plt.subplot(2,3,5)
sns.boxplot(df2['segment_actual_time'])
ax=plt.gca()
ax.set_facecolor('#AFDCEC')
plt.subplot(2,3,6)
sns.boxplot(df2['segment_osrm_time'])
ax=plt.gca()
ax.set_facecolor('#87AFC7')
plt.show()
```



In [112]:
```
plt.figure(figsize=(8,4))
sns.boxplot(df['segment_osrm_distance'])
ax=plt.gca()
ax.set_facecolor('#FFA500')
plt.show()
```



In [113]:
```
df2[numerical_columns].describe().T
```

Out[113]:

	count	mean	std	min	25%	50%	75%	max
start_scan_to_end_scan	14817.0	530.810016	658.705957	23.000000	149.000000	280.000000	637.000000	7898.000000
actual_distance_to_destination	14817.0	164.477838	305.388147	9.002461	22.837239	48.474072	164.583208	2186.531787
osrm_time	14817.0	161.384018	271.360995	6.000000	29.000000	60.000000	168.000000	2032.000000
osrm_distance	14817.0	204.344689	370.395573	9.072900	30.819200	65.618800	208.475000	2840.081000
segment_actual_time	14817.0	353.951610	556.320988	9.000000	66.000000	147.000000	367.000000	6230.000000
segment_osrm_time	14817.0	180.921172	314.485624	6.000000	31.000000	65.000000	185.000000	2564.000000
segment_osrm_distance	14817.0	223.164000	416.547252	9.072900	32.654500	70.154400	218.710200	3523.632400

In [ ]:
```
```

In [114]:
```
q1 = np.quantile(df2['start_scan_to_end_scan'], 0.25)
q3 = np.quantile(df2['start_scan_to_end_scan'], 0.75)
print(f"25th Percentile:-> {q1} , 75th Percentile:-> {q3}")
print("-"*50)
IQR = q3 - q1
print(f"The IQR is : --> {IQR}")
print("-"*50)
LW = q1-1.5*IQR
print(f"The Lower wick --> {LW}")
print("-"*50)
UW = q3 + 1.5*IQR
print(f"The Upper wick --> {UW}")
print("-"*50)
outliers = df2.loc[(df2['start_scan_to_end_scan'] < LW) | (df2['start_scan_to_end_scan'] > UW)]
```

```
25th Percentile:-> 149.0 , 75th Percentile:-> 637.0
--------------------------------------------------
The IQR is : --> 488.0
--------------------------------------------------
The Lower wick --> -583.0
--------------------------------------------------
The Upper wick --> 1369.0
--------------------------------------------------
```

In [ ]:
```
```

In [115]:
```
q1 = np.quantile(df2['actual_distance_to_destination'], 0.25)
q3 = np.quantile(df2['actual_distance_to_destination'], 0.75)
print(f"25th Percentile:-> {q1} , 75th Percentile:-> {q3}")
print("-"*50)
IQR = q3 - q1
print(f"The IQR is : --> {IQR}")
print("-"*50)
LW = q1-1.5*IQR
print(f"The Lower wick --> {LW}")
print("-"*50)
UW = q3 + 1.5*IQR
print(f"The Upper wick --> {UW}")
print("-"*50)
outliers = df2.loc[(df2['actual_distance_to_destination'] < LW) | (df2['actual_distance_to_destination'] > UW)]
print("The Outliers present are: -> ",len(outliers))
```

```
25th Percentile:-> 22.83723905859321 , 75th Percentile:-> 164.58320763841138
--------------------------------------------------
The IQR is : --> 141.74596857981817
--------------------------------------------------
The Lower wick --> -189.78171381113404
--------------------------------------------------
The Upper wick --> 377.2021605081386
--------------------------------------------------
The Outliers present are: ->  1449
```

In [ ]:
```
```

In [116]:
```
q1 = np.quantile(df2['osrm_time'], 0.25)
q3 = np.quantile(df2['osrm_time'], 0.75)
print(f"25th Percentile:-> {q1} , 75th Percentile:-> {q3}")
print("-"*50)
IQR = q3 - q1
print(f"The IQR is : --> {IQR}")
print("-"*50)
LW = q1-1.5*IQR
print(f"The Lower wick --> {LW}")
print("-"*50)
UW = q3 + 1.5*IQR
print(f"The Upper wick --> {UW}")
print("-"*50)
outliers = df2.loc[(df2['osrm_time'] < LW) | (df2['osrm_time'] > UW)]
print("The Outliers present are: -> ",len(outliers))
```

```
25th Percentile:-> 29.0 , 75th Percentile:-> 168.0
--------------------------------------------------
The IQR is : --> 139.0
--------------------------------------------------
The Lower wick --> -179.5
--------------------------------------------------
The Upper wick --> 376.5
--------------------------------------------------
The Outliers present are: ->  1517
```

In [ ]:
```
```

In [117]:
```
q1 = np.quantile(df2['osrm_distance'], 0.25)
q3 = np.quantile(df2['osrm_distance'], 0.75)
print(f"25th Percentile:-> {q1} , 75th Percentile:-> {q3}")
print("-"*50)
IQR = q3 - q1
print(f"The IQR is : --> {IQR}")
print("-"*50)
LW = q1-1.5*IQR
print(f"The Lower wick --> {LW}")
print("-"*50)
UW = q3 + 1.5*IQR
print(f"The Upper wick --> {UW}")
print("-"*50)
outliers = df2.loc[(df2['osrm_distance'] < LW) | (df2['osrm_distance'] > UW)]
print("The Outliers present are: -> ",len(outliers))
```

```
25th Percentile:-> 30.8192 , 75th Percentile:-> 208.475
--------------------------------------------------
The IQR is : --> 177.6558
--------------------------------------------------
The Lower wick --> -235.6645
--------------------------------------------------
The Upper wick --> 474.9587
--------------------------------------------------
The Outliers present are: ->  1524
```

In [ ]:
```
```

In [118]:
```
q1 = np.quantile(df2['segment_actual_time'], 0.25)
q3 = np.quantile(df2['segment_actual_time'], 0.75)
print(f"25th Percentile:-> {q1} , 75th Percentile:-> {q3}")
print("-"*50)
IQR = q3 - q1
print(f"The IQR is : --> {IQR}")
print("-"*50)
LW = q1-1.5*IQR
print(f"The Lower wick --> {LW}")
print("-"*50)
UW = q3 + 1.5*IQR
print(f"The Upper wick --> {UW}")
print("-"*50)
outliers = df2.loc[(df2['segment_actual_time'] < LW) | (df2['segment_actual_time'] > UW)]
print("The Outliers present are: -> ",len(outliers))
```

```
25th Percentile:-> 66.0 , 75th Percentile:-> 367.0
--------------------------------------------------
The IQR is : --> 301.0
--------------------------------------------------
The Lower wick --> -385.5
--------------------------------------------------
The Upper wick --> 818.5
--------------------------------------------------
The Outliers present are: ->  1643
```

In [ ]:

In [119]:
```
q1 = np.quantile(df2['segment_osrm_time'], 0.25)
q3 = np.quantile(df2['segment_osrm_time'], 0.75)
print(f"25th Percentile:-> {q1} , 75th Percentile:-> {q3}")
print("-"*50)
IQR = q3 - q1
print(f"The IQR is : --> {IQR}")
print("-"*50)
LW = q1-1.5*IQR
print(f"The Lower wick --> {LW}")
print("-"*50)
UW = q3 + 1.5*IQR
print(f"The Upper wick --> {UW}")
print("-"*50)
outliers = df2.loc[(df2['segment_osrm_time'] < LW) | (df2['segment_osrm_time'] > UW)]
print("The Outliers present are: -> ",len(outliers))
```

```
25th Percentile:-> 31.0 , 75th Percentile:-> 185.0
--------------------------------------------------
The IQR is : --> 154.0
--------------------------------------------------
The Lower wick --> -200.0
--------------------------------------------------
The Upper wick --> 416.0
--------------------------------------------------
The Outliers present are: ->  1492
```

In [ ]:
```
#OSRM DISTANCE
```

In [120]:
```
q1 = np.quantile(df2['segment_osrm_distance'], 0.25)
q3 = np.quantile(df2['segment_osrm_distance'], 0.75)
print(f"25th Percentile:-> {q1} , 75th Percentile:-> {q3}")
print("-"*50)
IQR = q3 - q1
print(f"The IQR is : --> {IQR}")
print("-"*50)
LW = q1-1.5*IQR
print(f"The Lower wick --> {LW}")
print("-"*50)
UW = q3 + 1.5*IQR
print(f"The Upper wick --> {UW}")
print("-"*50)
outliers = df2.loc[(df2['segment_osrm_distance'] < LW) | (df2['segment_osrm_distance'] > UW)]
print("The Outliers present are: -> ",len(outliers))
```

```
25th Percentile:-> 32.6545 , 75th Percentile:-> 218.7102
--------------------------------------------------
The IQR is : --> 186.0557
--------------------------------------------------
The Lower wick --> -246.42905000000002
--------------------------------------------------
The Upper wick --> 497.79375
--------------------------------------------------
The Outliers present are: ->  1549
```

In [ ]:

In [121]:
```
df2[:2]
```

Out[121]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	DoddablpurChikaDPP_D (Karnataka)	DoddablpurChikaDPP_D (Karnataka)	

In [ ]:

In [122]:
```
df2['data'].value_counts()
```

Out[122]:
```
training    10654
test         4163
Name: data, dtype: int64
```

In [123]:
```
from sklearn.preprocessing import LabelEncoder
```

In [124]:
```
le = LabelEncoder()
le
```

Out[124]:
```
LabelEncoder()
```

In [125]:
```
df2['data_label'] = le.fit_transform(df2['data'])
```

In [126]:
```
df2["data_label"]
```

Out[126]:
```
0        1
1        1
2        1
3        1
4        1
        ..
14812    0
14813    0
14814    0
14815    0
14816    0
Name: data_label, Length: 14817, dtype: int32
```

In [127]:
```
df2["data"].value_counts()
```

Out[127]:
```
training    10654
test         4163
Name: data, dtype: int64
```

In [128]:
```
df2.head()
```

Out[128]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	DoddaBlpur_ChikaDPP_D (Karnataka)	DoddaBlpur_ChikaDPP_D (Karnataka)	
2	trip-153671043369099517	IND000000ACB	IND000000ACB	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur_HB (Haryana)	Gurgaon_Bilaspur_HB (Haryana)	
3	trip-153671046011330457	IND400072AAB	IND401104AAA	training	Carting	2018-09-12 00:01:00.113710	Mumbai Hub (Maharashtra)	Mumbai_MiraRd_IP (Maharashtra)	
4	trip-153671052974046625	IND583101AAA	IND583119AAA	training	FTL	2018-09-12 00:02:09.740725	Bellary_Dc (Karnataka)	Sandur_WrdN1DPP_D (Karnataka)	

In [ ]:
```
```

In [129]:
```
df2['route_type'].value_counts()
```

Out[129]:
```
Carting    8908
FTL        5909
Name: route_type, dtype: int64
```

In [130]:
```
labelencoder = LabelEncoder()
labelencoder
```

Out[130]:
```
LabelEncoder()
```

In [131]:
```
df2['route_label'] =labelencoder.fit_transform(df2['route_type'])
df2['route_label']
```

Out[131]:
```
0        1
1        0
2        1
3        0
4        1
        ..
14812    0
14813    0
14814    0
14815    0
14816    1
Name: route_label, Length: 14817, dtype: int32
```

In [132]:
```
df2.head()
```

Out[132]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	DoddaBlpur_ChikaDPP_D (Karnataka)	DoddaBlpur_ChikaDPP_D (Karnataka)	
2	trip-153671043369099517	IND000000ACB	IND000000ACB	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur_HB (Haryana)	Gurgaon_Bilaspur_HB (Haryana)	
3	trip-153671046011330457	IND400072AAB	IND401104AAA	training	Carting	2018-09-12 00:01:00.113710	Mumbai Hub (Maharashtra)	Mumbai_MiraRd_IP (Maharashtra)	
4	trip-153671052974046625	IND583101AAA	IND583119AAA	training	FTL	2018-09-12 00:02:09.740725	Bellary_Dc (Karnataka)	Sandur_WrdN1DPP_D (Karnataka)	

In [133]:
```
#Normalize/ Standarize
```

In [ ]:
```
```

In [134]:
```
Data = df2.copy()
Data.head()
```

Out[134]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	DoddaBlpur_ChikaDPP_D (Karnataka)	DoddaBlpur_ChikaDPP_D (Karnataka)	
2	trip-153671043369099517	IND000000ACB	IND000000ACB	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur_HB (Haryana)	Gurgaon_Bilaspur_HB (Haryana)	
3	trip-153671046011330457	IND400072AAB	IND401104AAA	training	Carting	2018-09-12 00:01:00.113710	Mumbai Hub (Maharashtra)	Mumbai_MiraRd_IP (Maharashtra)	
4	trip-153671052974046625	IND583101AAA	IND583119AAA	training	FTL	2018-09-12 00:02:09.740725	Bellary_Dc (Karnataka)	Sandur_WrdN1DPP_D (Karnataka)	

In [135]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [136]:
```
mm = MinMaxScaler()
mm
```

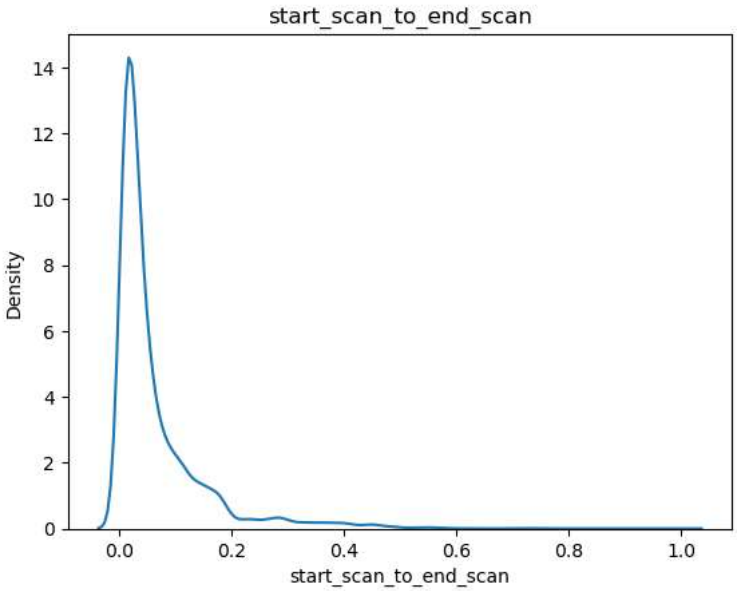
Out[136]:
```
MinMaxScaler()
```

In [137]:
```
Data['start_scan_to_end_scan'] = mm.fit_transform(Data['start_scan_to_end_scan'].to_numpy().reshape(-1,1))
Data['start_scan_to_end_scan'].head(2)
```

Out[137]:
```
0    0.283937
1    0.019937
Name: start_scan_to_end_scan, dtype: float64
```

In [139]:
```
```

In [140]:
```
sns.kdeplot(Data['start_scan_to_end_scan'])
plt.title("start_scan_to_end_scan")
plt.show()
```



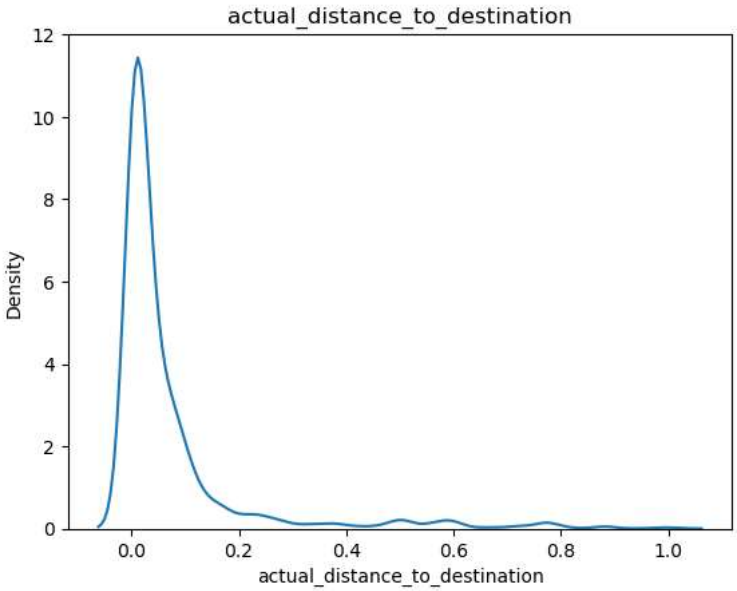
In [141]:
```
mm = MinMaxScaler()
mm
```

Out[141]:
```
MinMaxScaler()
```

In [142]:
```
Data['actual_distance_to_destination'] = mm.fit_transform(Data['actual_distance_to_destination'].to_numpy().reshape(-1,1))
Data['actual_distance_to_destination'][:3]
```

Out[142]:
```
0    0.374613
1    0.029476
2    0.880999
Name: actual_distance_to_destination, dtype: float64
```

In [143]:
```
sns.kdeplot(Data['actual_distance_to_destination'])
plt.title("actual_distance_to_destination")
plt.show()
```



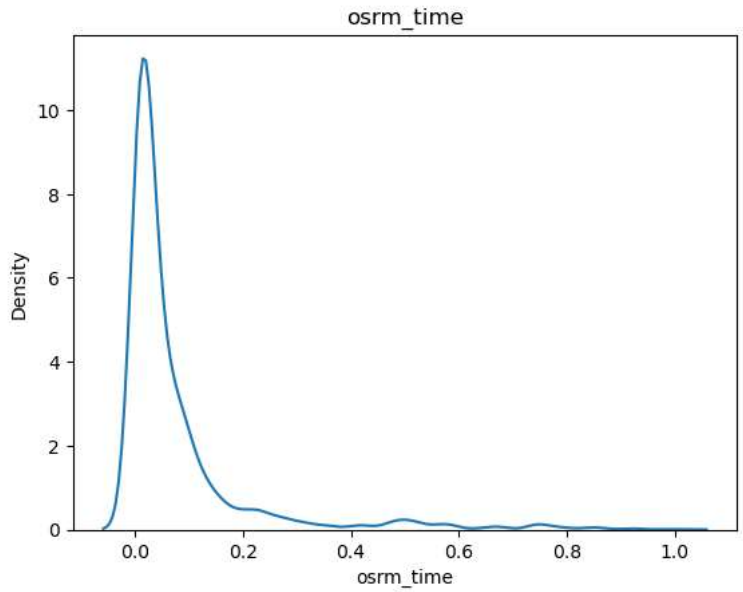
In [144]:
```
mm = MinMaxScaler()
mm
```

Out[144]:
```
MinMaxScaler()
```

In [145]:
```
Data['osrm_time'] = mm.fit_transform(Data['osrm_time'].to_numpy().reshape(-1,1))
Data['osrm_time'][:3]
```

Out[145]:
```
0    0.350938
1    0.030602
2    0.855874
Name: osrm_time, dtype: float64
```

In [146]:
```
sns.kdeplot(Data['osrm_time'])
plt.title("osrm_time")
plt.show()
```



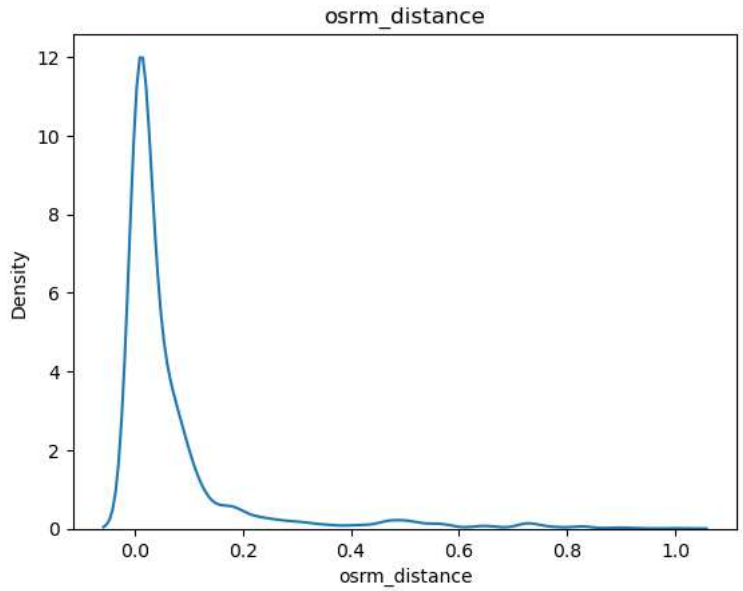
In [147]:
```
mm = MinMaxScaler()
mm
```

Out[147]:
```
MinMaxScaler()
```

In [148]:
```
Data['osrm_distance'] = mm.fit_transform(Data['osrm_distance'].to_numpy().reshape(-1,1))
Data['osrm_distance'][:3]
```

Out[148]:
```
0    0.346972
1    0.026859
2    0.828325
Name: osrm_distance, dtype: float64
```

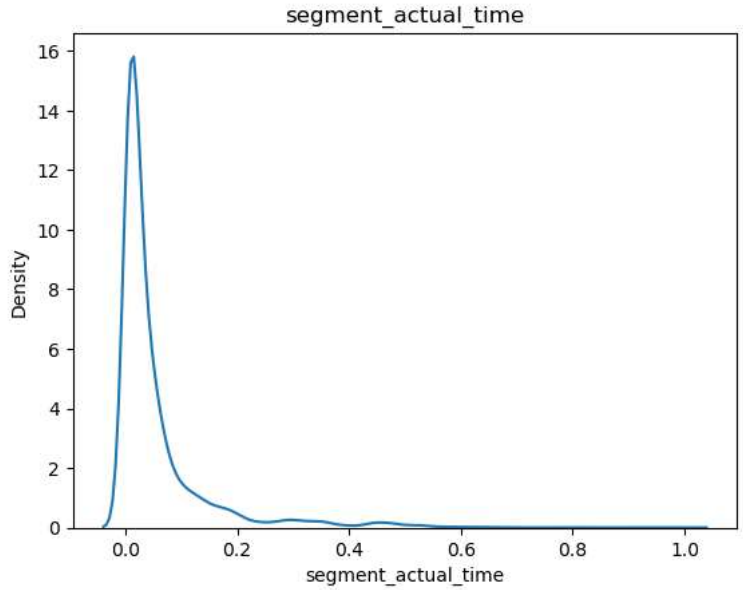
In [149]:
```
sns.kdeplot(Data['osrm_distance'])
plt.title("osrm_distance")
plt.show()
```



In [150]:
```
Data['segment_actual_time'] = mm.fit_transform(Data['segment_actual_time'].to_numpy().reshape(-1,1))
Data['segment_actual_time'][:3]
```

Out[150]:
```
0    0.247388
1    0.021218
2    0.530301
Name: segment_actual_time, dtype: float64
```

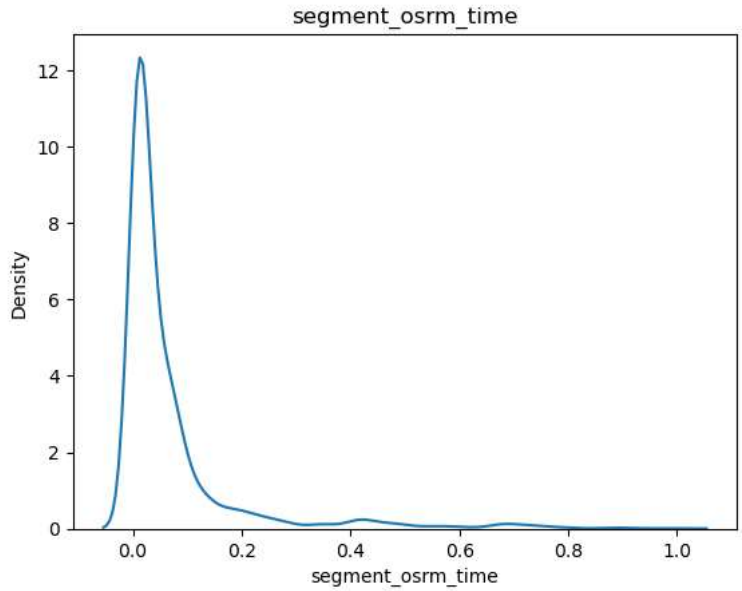
In [151]:
```
sns.kdeplot(Data['segment_actual_time'])
plt.title("segment_actual_time")
plt.show()
```



In [152]:
```
Data['segment_osrm_time'] = mm.fit_transform(Data['segment_osrm_time'].to_numpy().reshape(-1,1))
Data['segment_osrm_time'][:3]
```

Out[152]:
```
0    0.391712
1    0.023065
2    0.756450
Name: segment_osrm_time, dtype: float64
```

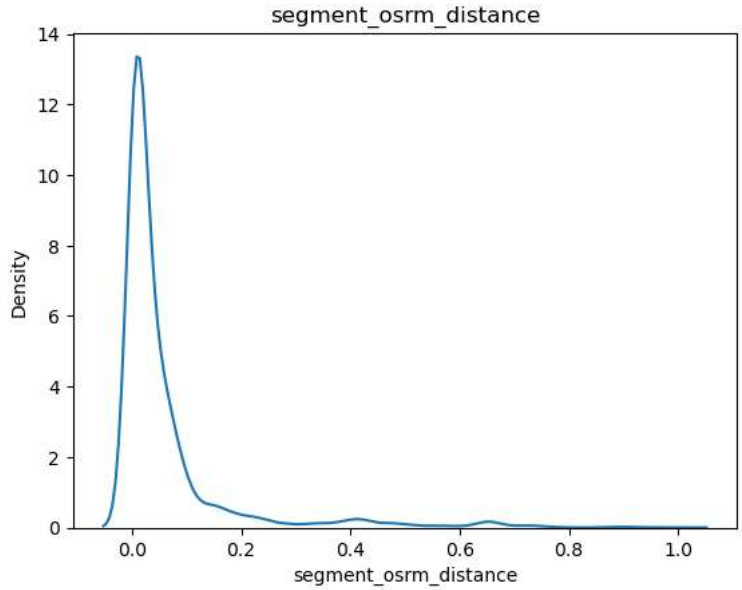
In [153]:
```
sns.kdeplot(Data['segment_osrm_time'])
plt.title("segment_osrm_time")
plt.show()
```



In [154]:
```
Data['segment_osrm_distance'] = mm.fit_transform(Data['segment_osrm_distance'].to_numpy().reshape(-1,1))
Data['segment_osrm_distance'][:3]
```

Out[154]:
```
0    0.373134
1    0.021373
2    0.721625
Name: segment_osrm_distance, dtype: float64
```

In [155]:
```
sns.kdeplot(Data['segment_osrm_distance'])
plt.title("segment_osrm_distance")
plt.show()
```



Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler.

```
### Let's Analyse the Data from the Standard Scaler Function to check the Data comes out to be in Standard Format for the rest of the Analysis
```

```
# This standardization also known as the z- score normalization, it aims to transform the data into the standard normal
distribution - with the mean 0 and std deviation 1
```

In [159]:
```
Sdata = df2.copy()
Sdata.head(2)
```

Out[159]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaDPP_D (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	

In [160]:
```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
ss
```

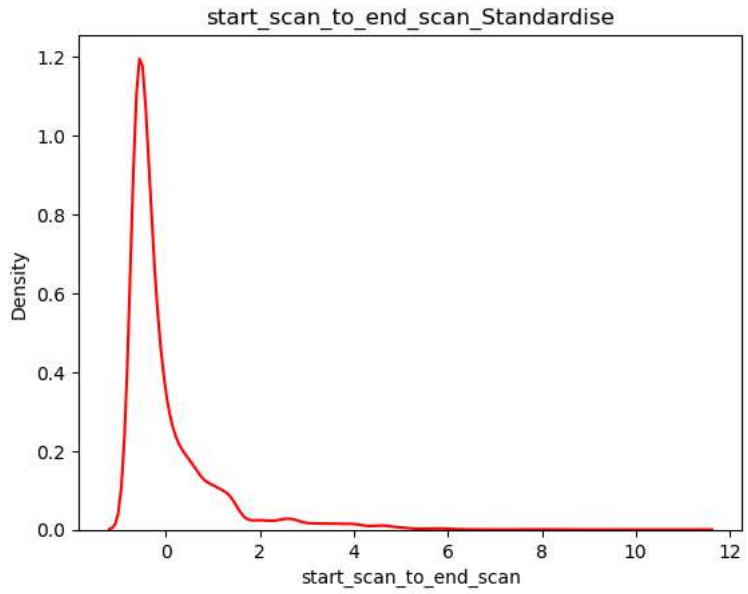
Out[160]:
```
StandardScaler
StandardScaler()
```

In [161]:
```
Sdata['start_scan_to_end_scan'] = ss.fit_transform(Sdata['start_scan_to_end_scan'].to_numpy().reshape(-1,1))
Sdata[:3]
```

Out[161]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaDPP_D (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	
2	trip-153671043369099517	IND000000ACB	IND000000ACB	training	FTL	2018-09-12 00:00:33.691250	Gurgaon_Bilaspur_HB (Haryana)	Gurgaon_Bilaspur_HB (Haryana)	

In [162]:
```
sns.kdeplot(Sdata['start_scan_to_end_scan'], color="red")
plt.title("start_scan_to_end_scan_Standardise")
plt.show()
```

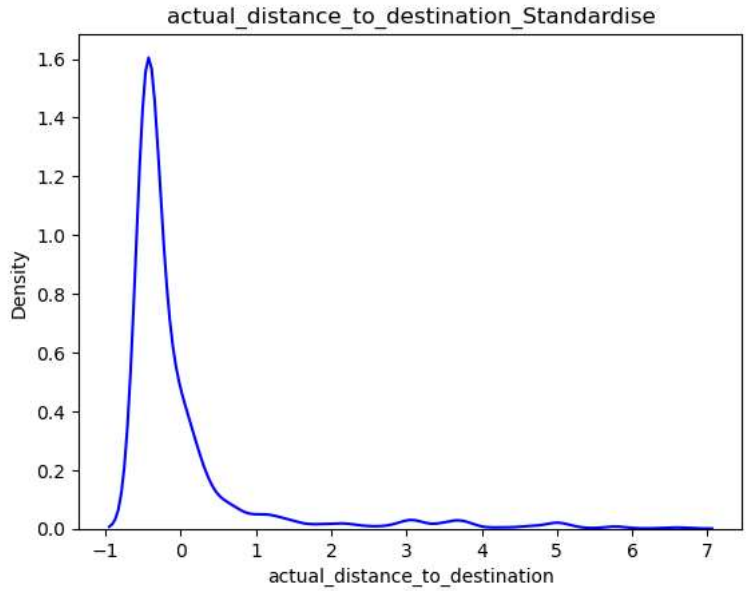


In [164]:
```
Sdata['actual_distance_to_destination'] = ss.fit_transform(Sdata['actual_distance_to_destination'].to_numpy().reshape(-1,1))
Sdata[:2]
```

Out[164]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaDPP_D (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	

In [165]:
```
sns.kdeplot(Sdata['actual_distance_to_destination'], color="blue")
plt.title("actual_distance_to_destination_Standardise")
plt.show()
```

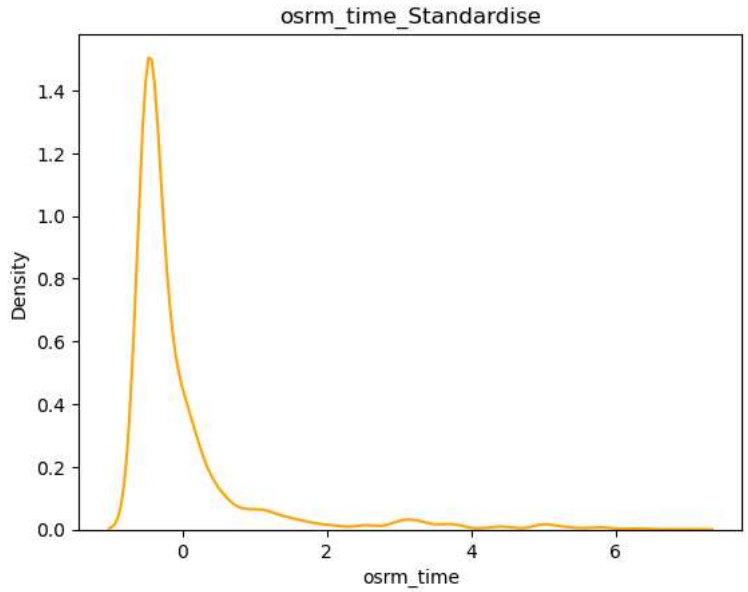


In [166]:
```
Sdata['osrm_time'] = ss.fit_transform(Sdata['osrm_time'].to_numpy().reshape(-1,1))
Sdata[:2]
```

Out[166]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_tc
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaDPP_D (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	

In [167]:
```
sns.kdeplot(Sdata['osrm_time'], color="orange")
plt.title("osrm_time_Standardise")
plt.show()
```

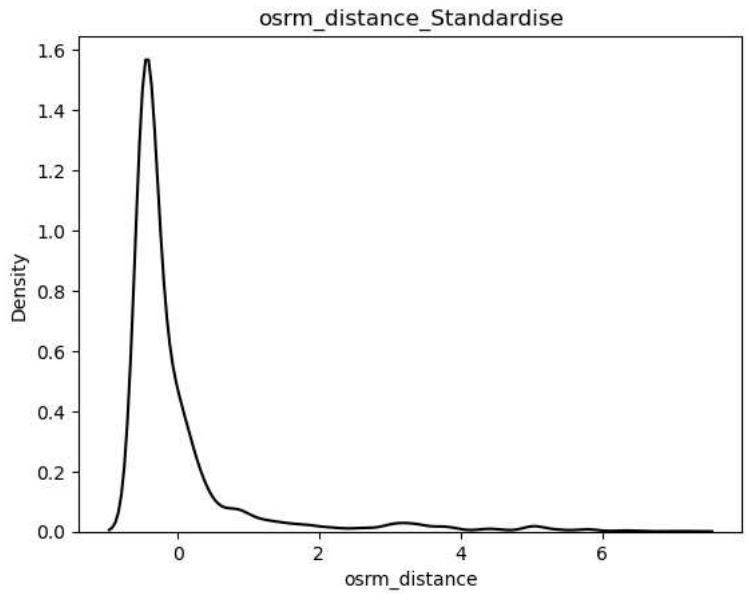


In [168]:
```
Sdata['osrm_distance'] = ss.fit_transform(Sdata['osrm_distance'].to_numpy().reshape(-1,1))
Sdata[:2]
```

Out[168]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_tc
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaDPP_D (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	

In [169]:
```
sns.kdeplot(Sdata['osrm_distance'], color="black")
plt.title("osrm_distance_Standardise")
plt.show()
```

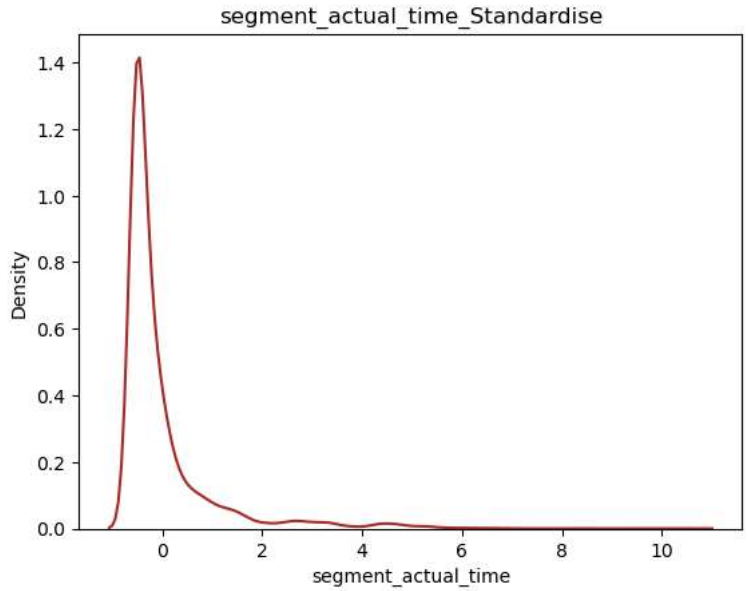


In [170]:
```
Sdata['segment_actual_time'] = ss.fit_transform(Sdata['segment_actual_time'].to_numpy().reshape(-1,1))
Sdata[:2]
```

Out[170]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_tc
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaDPP_D (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	

In [171]:
```
sns.kdeplot(Sdata['segment_actual_time'], color="brown")
plt.title("segment_actual_time_Standardise")
plt.show()
```

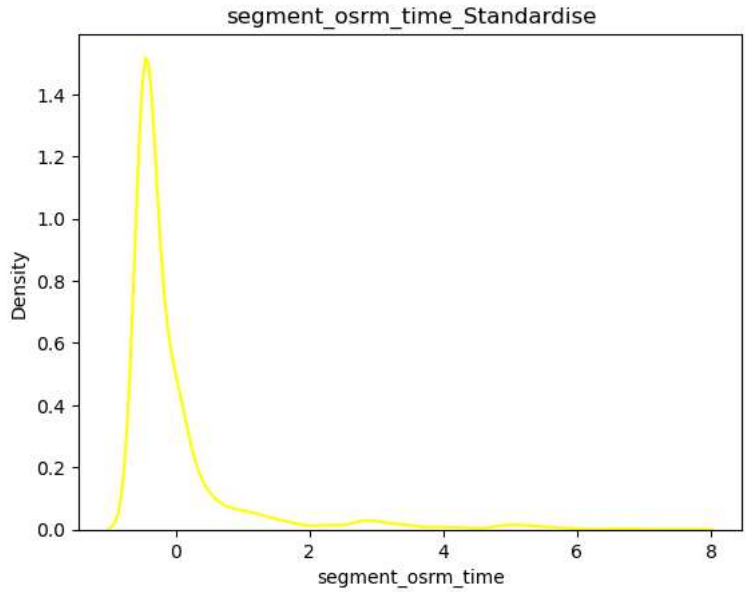


In [172]:
```
Sdata['segment_osrm_time'] = ss.fit_transform(Sdata['segment_osrm_time'].to_numpy().reshape(-1,1))
Sdata[:2]
```

Out[172]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_tc
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaDPP_D (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	

In [173]:
```
sns.kdeplot(Sdata['segment_osrm_time'], color="yellow")
plt.title("segment_osrm_time_Standardise")
plt.show()
```

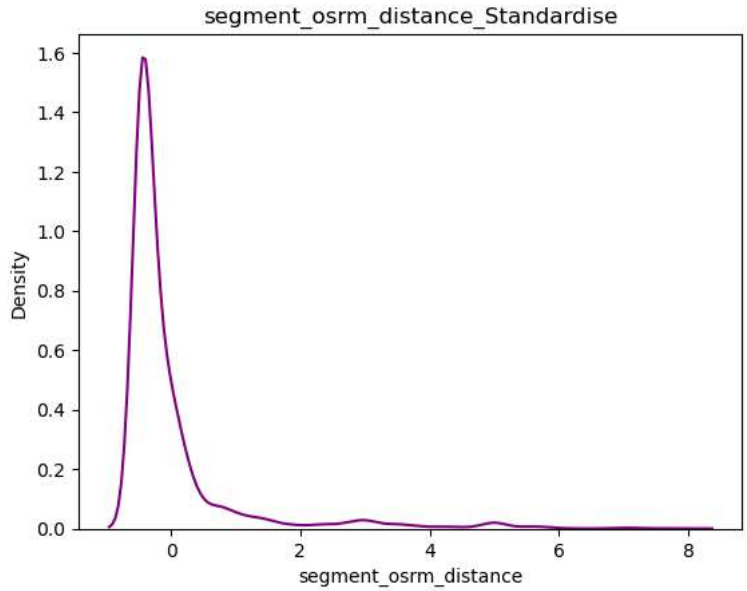


In [174]:
```
Sdata['segment_osrm_distance'] = ss.fit_transform(Sdata['segment_osrm_distance'].to_numpy().reshape(-1,1))
Sdata[:2]
```

Out[174]:

	trip_uuid	source_center	destination_center	data	route_type	trip_creation_time	source_name	destination_name	start_scan_to
0	trip-153671041653548748	IND209304AAA	IND209304AAA	training	FTL	2018-09-12 00:00:16.535741	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	
1	trip-153671042288605164	IND561203AAB	IND561203AAB	training	Carting	2018-09-12 00:00:22.886430	Doddablpur_ChikaDPP_D (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	

In [175]:
```
sns.kdeplot(Sdata['segment_osrm_distance'], color="purple")
plt.title("segment_osrm_distance_Standardise")
plt.show()
```



Hypothesis Testing

In [ ]:
```
Perform hypothesis testing / visual analysis between : actual_time aggregated value and OSRM time aggregated value.
```

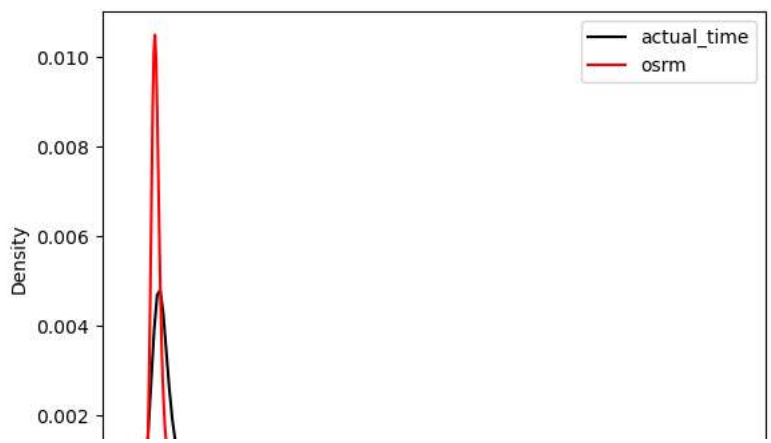
In [177]:
```
df1[['actual_time','osrm_time']]
```

Out[177]:

	actual_time	osrm_time
0	732.0	329.0
1	830.0	388.0
2	47.0	26.0
3	96.0	42.0
4	611.0	212.0
...	...	...
26363	51.0	41.0
26364	90.0	48.0
26365	30.0	14.0
26366	233.0	42.0
26367	42.0	26.0

26368 rows × 2 columns

In [179]:
```
sns.kdeplot(df1['actual_time'], color="black", label = "actual_time")
sns.kdeplot(df1['osrm_time'], color="red",label = "osrm")
plt.legend()
plt.show()
```



In [ ]:
```
Here, from the Data we can see the both comes into the Numerical Columns. so, the hypothesis testing for the both of the
columns are fall into the TTest and the actual_time - Actual time taken to complete the delivery (Cumulative) and osrm_time - An
shortest path between points in a given map (Includes usual traffic, distance through
major and minor roads) and gives the time (Cumulative) are performing the indivisual behaviour, so we use here the
Ttest_ind for the Hypothesis Analysis
```

```
The Above graph does not statisfy the Normal Distribution, lets check the Data to form a Normal Distribution with the Help of
the Shapiro wilk test or the Boxcox test
```

```
H0: The sample follows normal distribution
Ha: The sample does not follow normal distribution

alpha = 0.05

Test Statistics : Shapiro-Wilk test for normality
```

In [181]:
```
stats , pvalue = shapiro(df['actual_time'].sample(5000))
print(f"The stats value is: {stats} and the p_value comes to be {pvalue}")
alpha = 0.05
if pvalue < alpha:
    print("The sample does not follow normal distribution")
else:
    print("The sample follows normal distribution")
```

```
The stats value is: 0.6824350357055664 and the p_value comes to be 0.0
The sample does not follow normal distribution
```

In [ ]:
```
#Let's Analyse with the help of the Boxcox test
```

In [182]:
```
transform_actual_time = boxcox(df['actual_time'])[0]
stats, pvalue = shapiro(transform_actual_time)
print(f"The stats value is: {stats} and the p_value comes to be {pvalue}")
alpha = 0.05
if pvalue < alpha:
    print("The sample does not follow normal distribution")
else:
    print("The sample follows normal distribution")
```

```
The stats value is: 0.9715785980224609 and the p_value comes to be 0.0
The sample does not follow normal distribution

C:\ProgramData\anaconda3\lib\site-packages\scipy\stats\_morestats.py:1816: UserWarning: p-value may not be accurate for N > 500
0.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

In [183]:
```
#From the Both of the test, we find as, the data doesnot follow the normal Distribution
```

In [184]:
```
df1['actual_time'].mean() , df1['osrm_time'].mean()
```

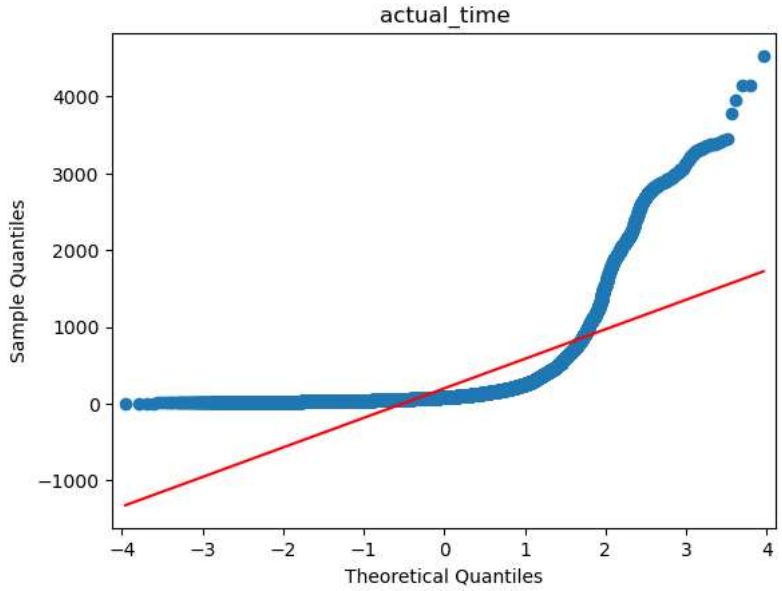
Out[184]:
```
(200.690192657767, 90.68670358009709)
```

```
Let's setup the framework for the hypothesis testing for analysing the actual time and osrm time, as we include the
mean of both of them.
```

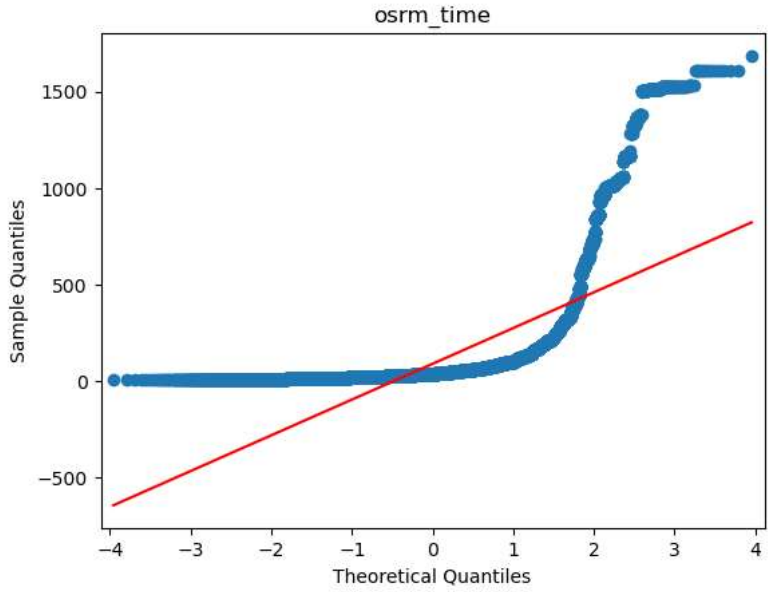
H0: The mean between the both of the values is same ie. (mu1 = mu2)
    Ha: The mean between the both of the values are different i.e (mu1 != mu2) or (mu1 > mu2)

Also, lets have the analysis for the confidence level at 5% and significance value at the 95%.
Where the alpha = 0.05

In [185]:
```
sm.qqplot(df1['actual_time'], line='s')
plt.title("actual_time")
plt.show()
```



In [186]:
```
sm.qqplot(df1['osrm_time'], line='s')
plt.title("osrm_time")
plt.show()
```



In [187]:
```
alpha  = 0.05
statistic, pvalue = ttest_ind(df1['actual_time'], df1['osrm_time'])
print(f"The statistic value is : {statistic} and the P_value is : {pvalue}")
if pvalue < alpha:
    print(f"Reject the Null hypothesis at : {pvalue}, the values for the Both of the columns are Not same")
else:
    print(f"Failed to Reject H0 at : {pvalue}, the values are Both of the time is same")
```

```
The statistic value is : 41.82845508363711 and the P_value is : 0.0
Reject the Null hypothesis at : 0.0, the values for the Both of the columns are Not same
```

Hypothesis testing Actual_time aggregated value and segment actual time aggregated value.

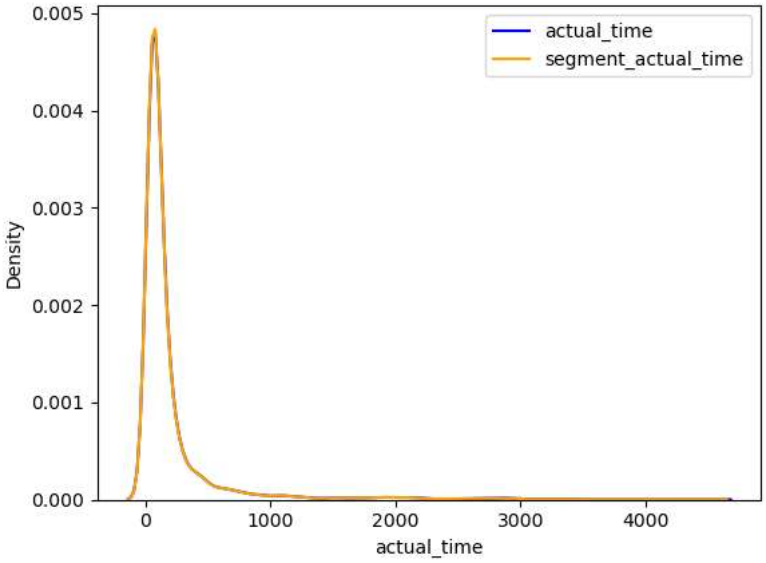
In [188]:
```
df1[['actual_time', 'segment_actual_time']]
```

Out[188]:

	actual_time	segment_actual_time
0	732.0	728.0
1	830.0	820.0
2	47.0	46.0
3	96.0	95.0
4	611.0	608.0
...	...	...
26363	51.0	49.0
26364	90.0	89.0
26365	30.0	29.0
26366	233.0	233.0
26367	42.0	41.0

26368 rows × 2 columns

In [189]:
```
sns.kdeplot(df1['actual_time'], color="blue", label = "actual_time")
sns.kdeplot(df1['segment_actual_time'], color="orange",label = "segment_actual_time")
plt.legend()
plt.show()
```



From the above graph we can depict that, the data from both of the columns are overlapping to each other as it says, the Data are lies in the same direction with the same point of values, where the both columns are indepedent to each other. so we use the Ttest_ind for the Hypotheis testing and predict the level of the homogenous between the columns which are actual time and Segment actual time.

The Above graph does not statisfy the Normal Distribution, lets check the Data to form a Normal Distribution with the Help of the Shapiro wilk test or the Boxcox test

H0: The sample follows normal distribution
Ha: The sample does not follow normal distribution

alpha = 0.05

Test Statistics : Shapiro-Wilk test for normality

In [191]:
```
stats , pvalue = shapiro(df['segment_actual_time'].sample(5000))
print(f"The stats value is: {stats} and the p_value comes to be {pvalue}")
alpha = 0.05
if pvalue < alpha:
    print("The sample does not follow normal distribution")
else:
    print("The sample follows normal distribution")
```

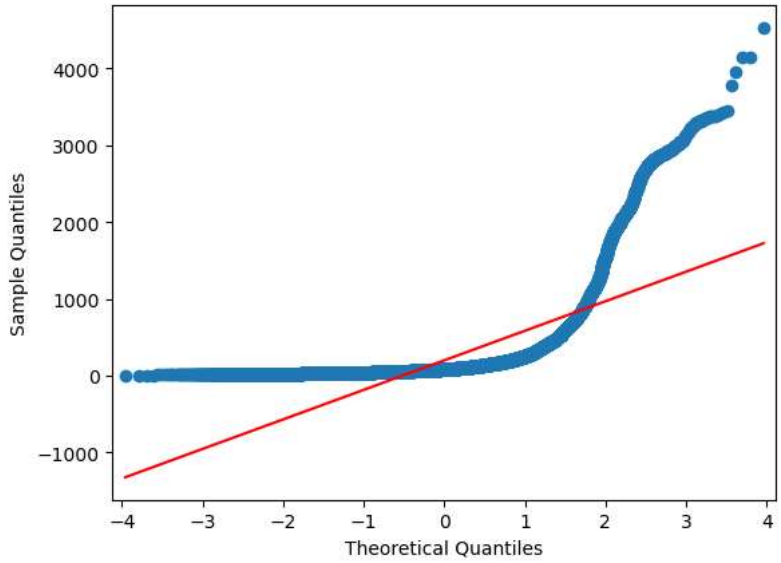
```
The stats value is: 0.36624062061309814 and the p_value comes to be 0.0
The sample does not follow normal distribution
```

In [192]:
```
df1['actual_time'].mean(), df1['segment_actual_time'].mean()
```

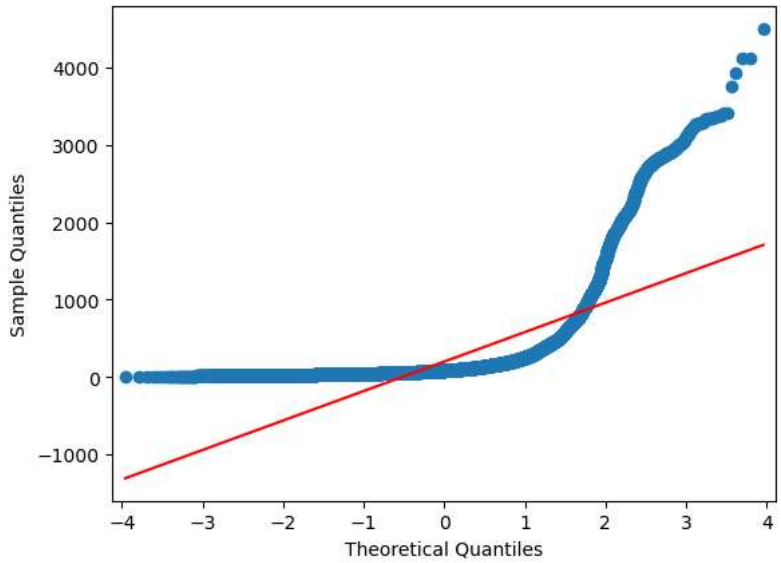
Out[192]:
```
(200.690192657767, 198.8964274878641)
```

```
From the above means says the, the values for the Graph almost lies on the same predicted values, the mean between
both of the columns are on the same page as the difference between them is too low.
Let's Analyse the Above columns in the Form QQ plot to analyse the Distribution is Gaussian or not.
```

In [193]:
```
sm.qqplot(df1['actual_time'], line='s')
plt.show()
```



In [194]:
```
sm.qqplot(df1['segment_actual_time'], line='s')
plt.show()
```




```
Let's perform the Ttest_ind for the osrm distance and segment distance for the hypotheis testing for numerical columns.
Let's suppose the significance value for the testing is 95% with the confidence level 5%.
which belongs to the alpha = 0.05%
Framework:
H0: The osrm distance and segment distance is same (mu1 = mu2)
```

Ha: The osrm distance and segment distance is not same (mu1 != mu2)

In [195]:
```
alpha  = 0.05
statistic, pvalue = ttest_ind(df1['actual_time'], df1['segment_actual_time'])
print(f"The statistic value is : {statistic} and the P_value is : {pvalue}")
if pvalue < alpha:
    print(f"Reject the Null hypothesis at : {pvalue}, the values for the Both of the columns are Not same")
else:
    print(f"Failed to Reject H0 at : {pvalue}, the values are Both of the time is same")
print(" ** This Includes the Actual time and Segmented Actual time are the on the same Distributed values **")
```

```
The statistic value is : 0.5376121149549357 and the P_value is : 0.5908471331390668
Failed to Reject H0 at : 0.5908471331390668, the values are Both of the time is same
 ** This Includes the Actual time and Segmented Actual time are the on the same Distributed values **
```

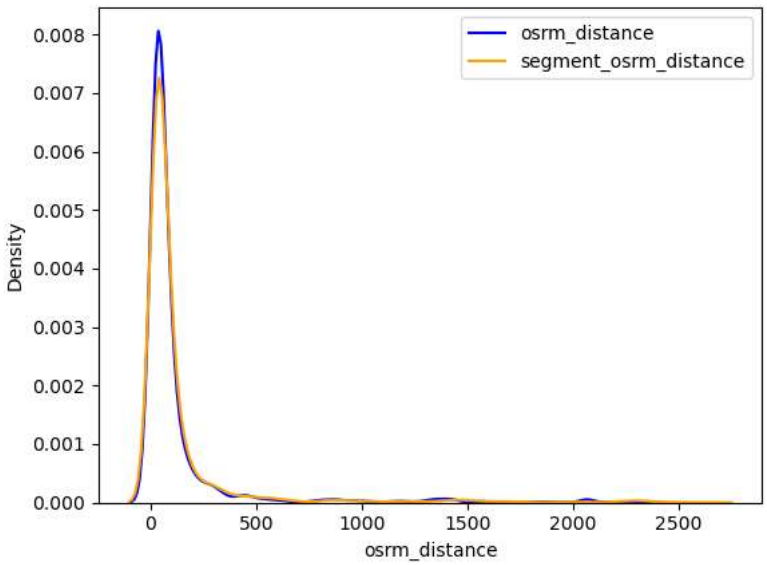
Hypothesis Testing between OSRM distance aggregated value and segment OSRM distance aggregated value.

In [197]:
```
df1[['osrm_distance', 'segment_osrm_distance']]
```

Out[197]:

	osrm_distance	segment_osrm_distance
0	446.5496	670.6205
1	544.8027	649.8528
2	28.1994	28.1995
3	56.9116	55.9899
4	281.2109	317.7408
...	...	...
26363	42.5213	42.1431
26364	40.6080	78.5869
26365	16.0185	16.0184
26366	52.5303	52.5303
26367	28.0484	28.0484

In [198]:
```
sns.kdeplot(df1['osrm_distance'], color="blue", label = "osrm_distance")
sns.kdeplot(df1['segment_osrm_distance'], color="orange",label = "segment_osrm_distance")
plt.legend()
plt.show()
```



From the above graph we can depict the slight difference between the osrm distance and segment distance where the mean falls on the same ground and but the density of the osrm distance comes more than the segment distance. The both of the column which is osrm distance and segment distance are the individual behaviour.
Here, we check fot the ttest for both of the indivisual numerical columns and classify the test to check the differences between.

The Above graph does not statisfy the Normal Distribution, lets check the Data to form a Normal Distribution with the Help of the Shapiro wilk test or the Boxcox test

In [ ]:
```
H0: The sample follows normal distribution
Ha: The sample does not follow normal distribution

alpha = 0.05

Test Statistics : Shapiro-Wilk test for normality
```

In [200]:
```
stats , pvalue = shapiro(df['osrm_time'].sample(5000))
print(f"The stats value is: {stats} and the p_value comes to be {pvalue}")
alpha = 0.05
if pvalue < alpha:
    print("The sample does not follow normal distribution")
else:
    print("The sample follows normal distribution")
```

```
The stats value is: 0.6898388266563416 and the p_value comes to be 0.0
The sample does not follow normal distribution
```

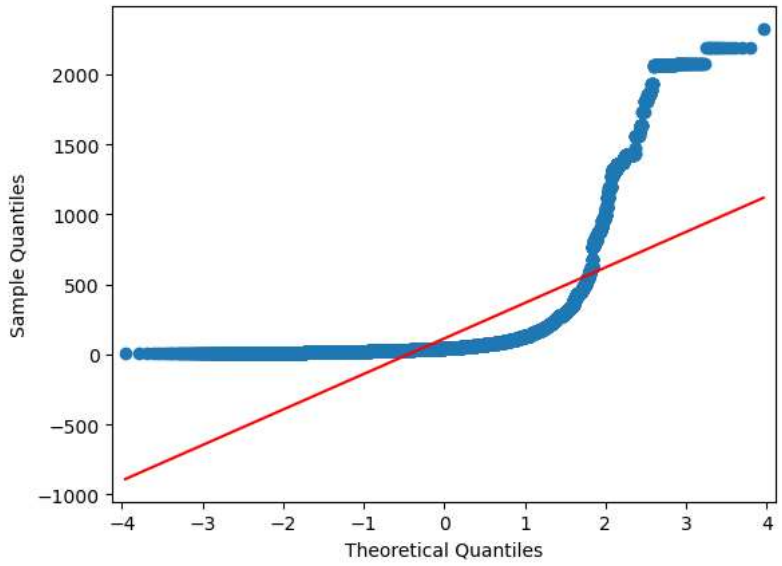
In [201]:
```
df1['osrm_distance'].mean(), df1['segment_osrm_distance'].mean()
```

Out[201]:
```
(114.82764181962986, 125.40279846404732)
```

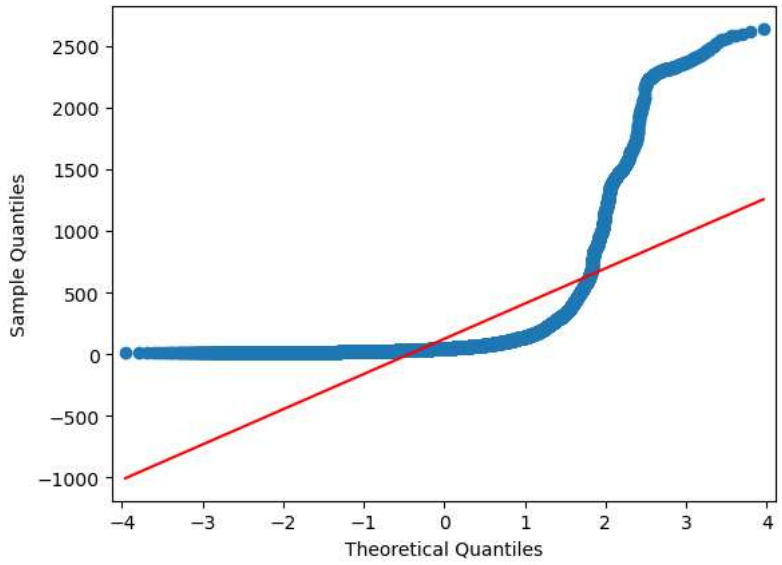
In [ ]:
```

```

In [202]:
```
sm.qqplot(df1['osrm_distance'], line='s')
plt.show()
```



In [204]:
```
sm.qqplot(df1['segment_osrm_distance'], line='s')
plt.show()
```



In [ ]:
```
#Let's perform the Ttest_ind for the osrm distance and segment distance for the hypotheis testing for numerical columns.
Let's suppose the significance value for the testing is 95% with the confidence level 5%.
which belongs to the alpha = 0.05%
Framework:
H0: The osrm distance and segment distance is same (mu1 = mu2)
    Ha: The osrm distance and segment distance is not same (mu1 !=  mu2)
```

In [206]:
```
alpha  = 0.05
statistic, pvalue = ttest_ind(df1['osrm_distance'], df1['segment_osrm_distance'])
print(f"The statistic value is : {statistic} and the P_value is : {pvalue}")
if pvalue < alpha:
    print(f"Reject the Null hypothesis at : {pvalue}, the mean values for the Both of the columns are Not same")
else:
    print(f"Failed to Reject H0 at : {pvalue}, the Mean values are Both of the time is same")
print(" ** This Includes the Osrm Distance and Segment Distance are the on the different Distributed values **")
```

```
The statistic value is : -4.4923128284182905 and the P_value is : 7.060322481862229e-06
Reject the Null hypothesis at : 7.060322481862229e-06, the mean values for the Both of the columns are Not same
 ** This Includes the Osrm Distance and Segment Distance are the on the different Distributed values **
```

Hypothesis Testing for OSRM time aggregated value and segment OSRM time aggregated value.

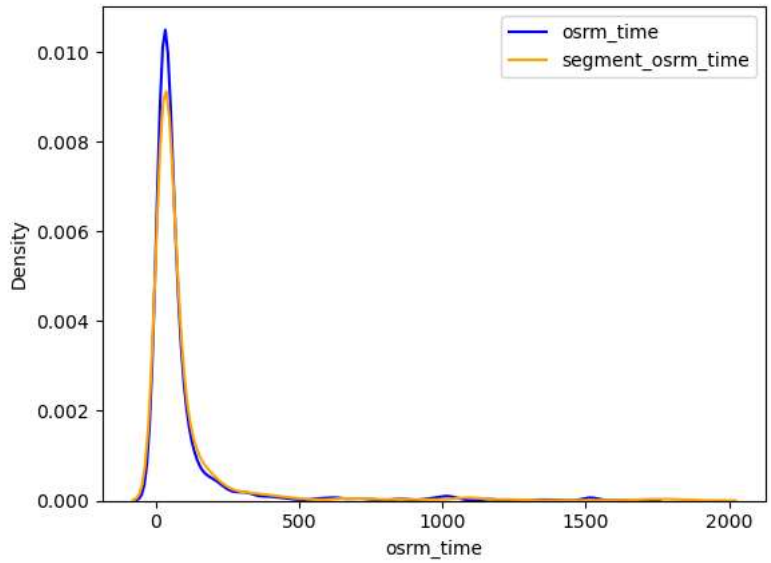
In [208]:
```
df1[['osrm_time','segment_osrm_time']]
```

Out[208]:

	osrm_time	segment_osrm_time
0	329.0	534.0
1	388.0	474.0
2	26.0	26.0
3	42.0	39.0
4	212.0	231.0
...	...	...
26363	41.0	42.0
26364	48.0	77.0
26365	14.0	14.0
26366	42.0	42.0
26367	26.0	25.0

26368 rows × 2 columns

In [209]:
```
sns.kdeplot(df1['osrm_time'], color="blue", label = "osrm_time")
sns.kdeplot(df1['segment_osrm_time'], color="orange",label = "segment_osrm_time")
plt.legend()
plt.show()
```



The graph indicates overlapping data in both columns, particularly highlighting higher density in the "osrm time" column. The data points align in the same direction with similar values, suggesting independence between the two columns. To assess the homogeneity level between "osrm_time" and "Segment osrm_time," we employ the Ttest_ind for hypothesis testing.

The Above graph does not statisfy the Normal Distribution, lets check the Data to form a Normal Distribution with the Help of the Shapiro wilk test or the Boxcox test

H0: The sample follows normal distribution
Ha: The sample does not follow normal distribution

alpha = 0.05

Test Statistics : Shapiro-Wilk test for normality

In [212]:
```
stats , pvalue = shapiro(df['segment_osrm_time'].sample(5000))
print(f"The stats value is: {stats} and the p_value comes to be {pvalue}")
alpha = 0.05
if pvalue < alpha:
    print("The sample does not follow normal distribution")
else:
    print("The sample follows normal distribution")
```

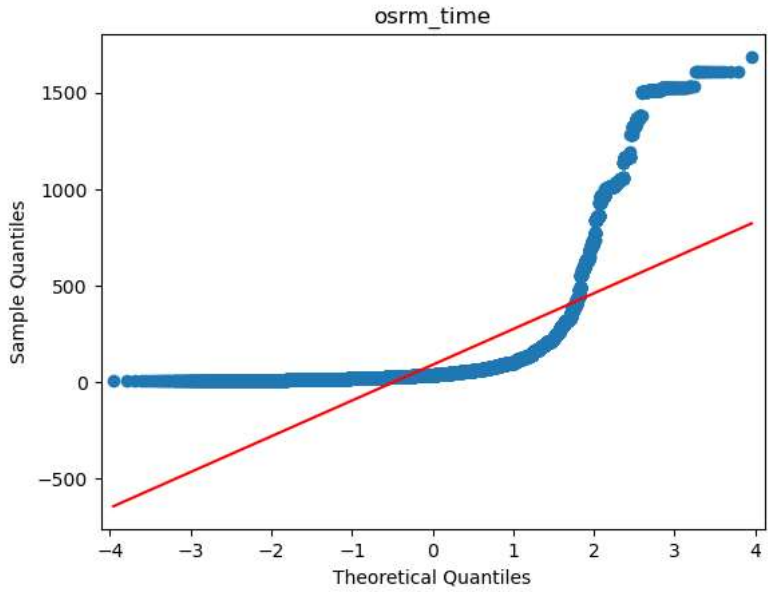
```
The stats value is: 0.631301999092102 and the p_value comes to be 0.0
The sample does not follow normal distribution
```

In [213]:
```
df1['osrm_time'].mean(), df1['segment_osrm_time'].mean()
```

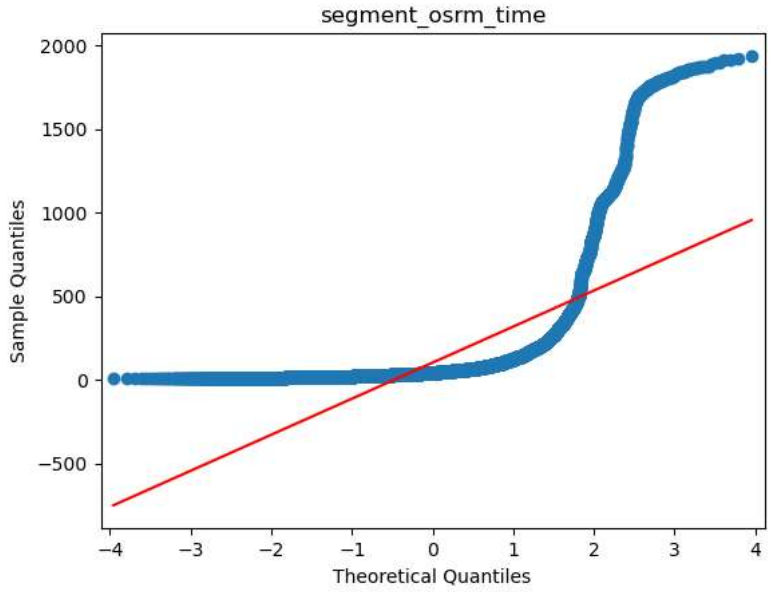
Out[213]:
```
(90.68670358009709, 101.66523816747574)
```

From the above means says the, the values for the Graph almost lies on the same predicted values, the mean between both of the columns are on the same page slighter more the segmented osrm time as the difference between them is too low.
Let's Analyse the Above columns in the Form QQ plot to analyse the Distribution is Gaussian or not.

In [215]:
```
sm.qqplot(df1['osrm_time'], line='s')
plt.title("osrm_time")
plt.show()
```



In [216]:
```
sm.qqplot(df1['segment_osrm_time'], line='s')
plt.title("segment_osrm_time")
plt.show()
```




```
Let's Analyse the columns with the Hypothesis testing by creating the Framework between them.
H0: The mean between the Columns (osrm time) and (segment osrm time) are same (mu1 = mu2)
Ha: The mean between the (osrm time) and (segment osrm time) are not same (mu1 != mu2)
    Also, lets have the analysis for the confidence level at 95% and significance value at the 5%.
    Where the alpha = 0.05
```

In [218]:
```
alpha  = 0.05
statistic, pvalue = ttest_ind(df1['osrm_time'], df1['segment_osrm_time'])
print(f"The statistic value is : {statistic} and the P_value is : {pvalue}")
if pvalue < alpha:
    print(f"Reject the Null hypothesis at : {pvalue}, the mean values for the Both of the columns are Not same")
else:
    print(f"Failed to Reject H0 at : {pvalue}, the Mean values are Both of the is same and lies on the same page")
print(" ** This Includes the Osrm time and Segment osrm time are the on the different Distributed values **")
```

```
The statistic value is : -6.273924701299134 and the P_value is : 3.547839442615796e-10
Reject the Null hypothesis at : 3.547839442615796e-10, the mean values for the Both of the columns are Not same
 ** This Includes the Osrm time and Segment osrm time are the on the different Distributed values **
```

Business Insights

In [ ]:
1. The data presented encompasses the dates "2018-09-12 00:00:16" to "2018-10-08 03:00:24."
2. There are roughly 1481 destination centres, 1508 source centres, 690 source cities, and 806 destination cities out of 14817 unique trip IDs.
3. Testing uses up the majority of the data, rather than training.
4. FTL routes are the most popular kind.
5. The data lacks the names for 14 distinct location ids. After midday, the number of journeys increases, reaches its peak around 10 oclock in the evening, and then begins to decline.
6. The majority of orders arrive in September. This indicates that consumers are more likely to place an order in the middle of September and less likely to do so in October.
7.States like Maharashtra, Karnataka, Haryana, Tamil Nadu, and Telangana supply the majority of orders.
8. Mumbai was the starting point for the greatest number of visits, followed by Delhi, Bengaluru, Bhiwandi, and Gurgaon. This indicates that these cities have a substantial seller base.
9. The state of Maharashtra had the greatest number of journeys cease, followed by Karnataka, Haryana, Tamil Nadu, and Uttar Pradesh. This indicates that there are a disproportionately large amount of orders placed in these states.
10. Mumbai was the destination of the greatest number of travels, followed by Bengaluru, Gurgaon, Delhi, and Chennai. This indicates that a notably large number of orders are placed in these cities.
11. The majority of orders, in terms of destination, originate from Delhi, Bangalore, Mumbai, Gurgaon, and Bengaluru.
12. The features osrm_time and actual_time differ statistically.
13. The features of segment_actual_time and actual time are statistically comparable.
14. The features segment_osrm_distance and osrm_distance differ from one another statistically.
15. There is a statistical difference between the osrm_time and segment_osrm_time.

Recommendations

In [ ]:
The OSRM trip planner need enhancement.
If the routing engine is set up to produce the best possible outcomes, then discrepancies must be accommodated for carriers.

Actual_time differs from osrm_time. The team must ensure that this discrepancy is minimised in order to improve delivery time estimation and enable customers to conveniently anticipate a precise arrival time.

The osrm distance and the actual distance travelled do not match, meaning that either the osrm devices are not accurately estimating the route based on factors like traffic, distance, and other considerations, or the delivery person is not adhering to the predetermined route, which could result in delayed deliveries. The team must investigate it.

The majority of orders originate in or go to states like Tamil Nadu, Maharashtra, Karnataka, and Haryana. Enhancements to the current corridors can be made to significantly boost penetration in these locations.

To enhance consumers purchasing and delivery experiences and understand why significant orders are coming from the states of Maharashtra, Karnataka, Haryana, Tamil Nadu, and Uttar Pradesh, customer profiles of these clients must be created.

From a state perspective, some states may have extremely high traffic volumes and unfavourable topography. This will serve as a useful guide for scheduling and meeting demand throughout the busiest festival seasons.