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

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

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

Along with that lets check from which cities most of the deliveries happen and their avg delivery time to the nearest hub/ delivery location.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stat
import datetime as dt
```

In [2]:

```
df = pd.read_csv('delhivery_data.csv')
```

In [3]:

```
df.shape
```

Out[3]:

```
(144867, 24)
```

In [4]:

```
df.head()
```

Out[4]:

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

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   data                            144867 non-null  object 
 1   trip_creation_time              144867 non-null  object 
 2   route_schedule_uuid             144867 non-null  object 
 3   route_type                      144867 non-null  object 
 4   trip_uuid                       144867 non-null  object 
 5   source_center                   144867 non-null  object 
 6   source_name                     144574 non-null  object 
 7   destination_center              144867 non-null  object 
 8   destination_name                144606 non-null  object 
 9   od_start_time                   144867 non-null  object 
 10  od_end_time                     144867 non-null  object 
 11  start_scan_to_end_scan          144867 non-null  float64
 12  is_cutoff                       144867 non-null  bool   
 13  cutoff_factor                   144867 non-null  int64  
```

In [6]:

```
df.drop(['is_cutoff','cutoff_factor','cutoff_timestamp','factor','segment_factor'],axis =1,
```

The above columns are not needed in the analysis as they dont add any value for now.

In [7]:

```
df.head()
```

Out[7]:

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

In [8]:

```
df['trip_creation_time'] = df['trip_creation_time'].astype('datetime64[ns]')
df['od_start_time'] = df['od_start_time'].astype('datetime64[ns]')
df['od_end_time'] = df['od_end_time'].astype('datetime64[ns]')
#df['data'] = df['data'].astype('category')
#df['route_type'] = df['route_type'].astype('category')
#df['source_center'] = df['source_center'].astype('category')
#df['source_name'] = df['source_name'].astype('category')
#df['destination_center'] = df['destination_center'].astype('category')
#df['destination_name'] = df['destination_name'].astype('category')
#df = df.astype({'start_scan_to_end_scan':'float32','actual_distance_to_destination':'float
```

In [9]:

```
# As route type is a categorical column we can perform one hot encoding to the column
df['route_type']=pd.get_dummies(df['route_type'],drop_first=True)
```

In [10]:

```
df['route_type'].unique()
```

Out[10]:

```
array([0, 1], dtype=uint8)
```

```
Route_type:
    0 : Carting
    1 : FTL
```

In [11]:

```
df.head()
```

Out[11]:

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

In [12]:

```
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  datetime64[ns]
 2   route_schedule_uuid             144867 non-null  object
 3   route_type                      144867 non-null  uint8
 4   trip_uuid                       144867 non-null  object
 5   source_center                   144867 non-null  object
 6   source_name                     144574 non-null  object
 7   destination_center              144867 non-null  object
 8   destination_name                144606 non-null  object
 9   od_start_time                   144867 non-null  datetime64[ns]
 10  od_end_time                     144867 non-null  datetime64[ns]
 11  start_scan_to_end_scan          144867 non-null  float64
 12  actual_distance_to_destination  144867 non-null  float64
 13  actual_time                     144867 non-null  float64
 14  osrm_time                       144867 non-null  float64
 15  osrm_distance                   144867 non-null  float64
 16  segment_actual_time             144867 non-null  float64
 17  segment_osrm_time               144867 non-null  float64
 18  segment_osrm_distance           144867 non-null  float64
dtypes: datetime64[ns](3), float64(8), object(7), uint8(1)
memory usage: 20.0+ MB
```

In [13]:

```
df.isnull().sum()/df.shape[0]
```

Out[13]:

```
data                              0.000000
trip_creation_time                0.000000
route_schedule_uuid               0.000000
route_type                        0.000000
trip_uuid                         0.000000
source_center                     0.000000
source_name                       0.002023
destination_center                0.000000
destination_name                  0.001802
od_start_time                     0.000000
od_end_time                       0.000000
start_scan_to_end_scan            0.000000
actual_distance_to_destination    0.000000
actual_time                       0.000000
osrm_time                         0.000000
osrm_distance                     0.000000
segment_actual_time               0.000000
segment_osrm_time                 0.000000
segment_osrm_distance             0.000000
dtype: float64
```

In [14]:

```
df[df.isna().any(axis=1)]
```

Out[14]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	so
110	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	1	trip-153786558437756691	IN
111	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	1	trip-153786558437756691	IN
112	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	1	trip-153786558437756691	IN
113	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	1	trip-153786558437756691	IN
114	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	1	trip-153786558437756691	IN
...	...	...	...	...	...	
144484	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	1	trip-153855756668984584	IN
144485	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	1	trip-153855756668984584	IN
144486	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	1	trip-153855756668984584	IN
144487	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	1	trip-153855756668984584	IN
144488	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	1	trip-153855756668984584	IN

551 rows × 19 columns

In [15]:

```
df.loc[ (df['source_center'] == 'IND342902A1B')]['source_name'].nunique()
```

Out[15]:

```
0
```

we see there is no values for source centers we can remove the null values as the missing value count is very low compared to data.

In [16]:

```
df.dropna(inplace = True)
```

In [17]:

```
df.shape
```

Out[17]:

```
(144316, 19)
```

In [18]:

```
df['data'].unique()
```

Out[18]:

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

In [19]:

```
df['data'].value_counts()
```

Out[19]:

```
training    104632
test         39684
Name: data, dtype: int64
```

In [20]:

```
df_train = df.loc[df['data'] == 'training']
```

In [21]:

```
df_test = df.loc[df['data'] == 'test']
```

In [22]:

```
df_train.drop(['data'],axis=1,inplace=True)
```

```
C:\Users\Ashok kumar\AppData\Local\Temp\ipykernel_5752\739344043.py:1: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df_train.drop(['data'],axis=1,inplace=True)
```

In [23]:

```
df_test.drop(['data'],axis=1,inplace=True)
```

```
C:\Users\Ashok kumar\AppData\Local\Temp\ipykernel_5752\618541614.py:1: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df_test.drop(['data'],axis=1,inplace=True)
```

As there is both testing and training data in the sample data we have seperated them as df_test and df_train and dropped the data column from them as it represents the type of data for test or train

In [24]:

```
#df_train.groupby([df_train['trip_uuid'],df_train['source_center'],df_train['source_name'],
```

In [25]:

```
df_train.columns
```

Out[25]:

```
Index(['trip_creation_time', 'route_schedule_uuid', 'route_type', 'trip_uui
d',
       'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance'],
      dtype='object')
```

In [26]:

```
df_grouped = df_train.groupby(['trip_uuid','source_center','source_name','destination_cente
                       tot_segment_osrm_time = ('segment_osrm_time','sum'),
                       tot_segment_osrm_distance = ('segment_osrm_distance','sum')
                       ).reset_index()
```

In [27]:

```
df_grouped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18893 entries, 0 to 18892
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   trip_uuid                       18893 non-null  object
 1   source_center                   18893 non-null  object
 2   source_name                     18893 non-null  object
 3   destination_center              18893 non-null  object
 4   destination_name                18893 non-null  object
 5   trip_creation_time              18893 non-null  datetime64[ns]
 6   route_schedule_uuid             18893 non-null  object
 7   route_type                      18893 non-null  uint8
 8   od_start_time                   18893 non-null  datetime64[ns]
 9   od_end_time                     18893 non-null  datetime64[ns]
 10  start_scan_to_end_scan          18893 non-null  float64
 11  actual_distance_to_destination  18893 non-null  float64
 12  actual_time                     18893 non-null  float64
 13  osrm_time                       18893 non-null  float64
 14  osrm_distance                   18893 non-null  float64
 15  tot_segment_actual_time         18893 non-null  float64
 16  tot_segment_osrm_time           18893 non-null  float64
 17  tot_segment_osrm_distance       18893 non-null  float64
dtypes: datetime64[ns](3), float64(8), object(6), uint8(1)
memory usage: 2.5+ MB
```

In [28]:

```
df_grouped.head()
```

Out[28]:

	trip_uuid	source_center	source_name	destination_center	destii
0	trip-153671041653548748	IND209304AAA	Kanpur_Central_H_6 (Uttar Pradesh)	IND000000ACB	Gurgaon_
1	trip-153671041653548748	IND462022AAA	Bhopal_Trnsport_H (Madhya Pradesh)	IND209304AAA	Kanpur_ (U
2	trip-153671042288605164	IND561203AAB	Doddablpur_ChikaDPP_D (Karnataka)	IND562101AAA	Chikblapu
3	trip-153671042288605164	IND572101AAA	Tumkur_Veersagr_I (Karnataka)	IND561203AAB	Doddablpur_
4	trip-153671043369099517	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND160002AAC	Chandigarh_I

In [29]:

```
df_grouped = df_grouped.sort_values(by = ['trip_uuid','od_start_time'])
```

As the trips might be shuffled we are ordering them by trip_uuid and od_start_time so that we can maintain the source an destination of an order in further process

In [30]:

```
df_grouped.head()
```

Out[30]:

	trip_uuid	source_center	source_name	destination_center	destin
1	trip-153671041653548748	IND462022AAA	Bhopal_Trnsport_H (Madhya Pradesh)	IND209304AAA	Kanpur_ (U
0	trip-153671041653548748	IND209304AAA	Kanpur_Central_H_6 (Uttar Pradesh)	IND000000ACB	Gurgaon_
3	trip-153671042288605164	IND572101AAA	Tumkur_Veersagr_I (Karnataka)	IND561203AAB	Doddablpur_(
2	trip-153671042288605164	IND561203AAB	Doddablpur_ChikaDPP_D (Karnataka)	IND562101AAA	Chikblapur
5	trip-153671043369099517	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND000000ACB	Gurgaon_

In [31]:

```
df_grouped.shape
```

Out[31]:

```
(18893, 18)
```

In [32]:

```
df_final=df_grouped.groupby(['trip_uuid']).agg(source_center = ('source_center','first'),
                                      source_name = ('source_name','first'),
                                      destination_center = ('destination_center','last'),
                                      destination_name = ('destination_name','last'),
                                      trip_creation_time = ('trip_creation_time','first'),
                                      route_schedule_uuid = ('route_schedule_uuid','first')
                                      route_type = ('route_type','first'),
                                      od_start_time = ('od_start_time','first'),
                                      od_end_time = ('od_end_time','last'),
                                      start_scan_to_end_scan = ('start_scan_to_end_scan','s
                                      actual_distance_to_destination = ('actual_distance_to
                                      actual_time =('actual_time','sum'),
                                      osrm_time = ('osrm_time','sum'),
                                      osrm_distance = ('osrm_distance','sum'),
                                      tot_segment_actual_time =('tot_segment_actual_time','
                                      tot_segment_osrm_time = ('tot_segment_osrm_time','sum
                                      tot_segment_osrm_distance = ('tot_segment_osrm_distan
```

In [33]:

```
df_final = df_final.reset_index()
```

In [34]:

```
df_final.columns
```

Out[34]:

```
Index(['trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'trip_creation_time', 'route_schedule_uuid',
       'route_type', 'od_start_time', 'od_end_time', 'start_scan_to_end_sca
n',
       'actual_distance_to_destination', 'actual_time', 'osrm_time',
       'osrm_distance', 'tot_segment_actual_time', 'tot_segment_osrm_time',
       'tot_segment_osrm_distance'],
      dtype='object')
```

In [35]:

```
df_final.head()
```

Out[35]:

	trip_uuid	source_center	source_name	destination_center	destinat
0	trip-153671041653548748	IND462022AAA	Bhopal_Trnsport_H (Madhya Pradesh)	IND000000ACB	Gurgaon_B
1	trip-153671042288605164	IND572101AAA	Tumkur_Veersagr_I (Karnataka)	IND562101AAA	Chikblapur_S (
2	trip-153671043369099517	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND160002AAC	Chandigarh_Me
3	trip-153671046011330457	IND400072AAB	Mumbai Hub (Maharashtra)	IND401104AAA	Mumbai_ (Ma
4	trip-153671052974046625	IND583101AAA	Bellary_Dc (Karnataka)	IND583101AAA	Bellary_Dc (

Now we have data for each order and their trip timings along with the source and destination and the type of trip

Lets start our analysis on this final dataset

In [36]:

```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10645 entries, 0 to 10644
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   trip_uuid                       10645 non-null  object
 1   source_center                   10645 non-null  object
 2   source_name                     10645 non-null  object
 3   destination_center              10645 non-null  object
 4   destination_name                10645 non-null  object
 5   trip_creation_time              10645 non-null  datetime64[ns]
 6   route_schedule_uuid             10645 non-null  object
 7   route_type                      10645 non-null  uint8
 8   od_start_time                   10645 non-null  datetime64[ns]
 9   od_end_time                     10645 non-null  datetime64[ns]
 10  start_scan_to_end_scan          10645 non-null  float64
 11  actual_distance_to_destination  10645 non-null  float64
 12  actual_time                     10645 non-null  float64
 13  osrm_time                       10645 non-null  float64
 14  osrm_distance                   10645 non-null  float64
 15  tot_segment_actual_time         10645 non-null  float64
 16  tot_segment_osrm_time           10645 non-null  float64
 17  tot_segment_osrm_distance       10645 non-null  float64
dtypes: datetime64[ns](3), float64(8), object(6), uint8(1)
memory usage: 1.4+ MB
```

Feature Extraction

In [37]:

```
df_final['od_year'] = df_final['trip_creation_time'].dt.year
df_final['od_month'] = df_final['trip_creation_time'].dt.month
df_final['od_date'] = df_final['trip_creation_time'].dt.date
df_final['od_hour'] = df_final['trip_creation_time'].dt.hour
```

In [38]:

```
df_final['source_city'] = df_final['source_name'].apply(lambda s:s.split('_')[0])
df_final['source_state'] = df_final['source_name'].apply(lambda s:s.split('(')[1].replace('
```

In [39]:

```
df_final['destination'] = df_final['destination_name'].apply(lambda s:s.split('_')[0])
df_final['destination_state'] = df_final['destination_name'].apply(lambda s:s.split('(')[1]
```

In [40]:

```
df_final['od_duration'] = (df_final['od_end_time']-df_final['od_start_time']).dt.total_seco
```

In [41]:

```
df_final.head()
```

	trip_uuid	source_center	source_name	destination_center	destination_name	tri
0	trip-153671041653548748	IND462022AAA	Bhopal_Trnsport_H (Madhya Pradesh)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	
1	trip-153671042288605164	IND572101AAA	Tumkur_Veersagr_I (Karnataka)	IND562101AAA	Chikblapur_ShntiSgr_D (Karnataka)	
2	trip-153671043369099517	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND160002AAC	Chandigarh_Mehmdpur_H (Punjab)	
3	trip-153671046011330457	IND400072AAB	Mumbai Hub (Maharashtra)	IND401104AAA	Mumbai_MiraRd_IP (Maharashtra)	
4	trip-153671052974046625	IND583101AAA	Bellary_Dc (Karnataka)	IND583101AAA	Bellary_Dc (Karnataka)	

EDA

In [42]:

```
df_final.shape
```

Out[42]:

```
(10645, 27)
```

In [43]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 144316 entries, 0 to 144866
Data columns (total 19 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144316 non-null  object
 1   trip_creation_time              144316 non-null  datetime64[ns]
 2   route_schedule_uuid             144316 non-null  object
 3   route_type                      144316 non-null  uint8
 4   trip_uuid                       144316 non-null  object
 5   source_center                   144316 non-null  object
 6   source_name                     144316 non-null  object
 7   destination_center              144316 non-null  object
 8   destination_name                144316 non-null  object
 9   od_start_time                   144316 non-null  datetime64[ns]
 10  od_end_time                     144316 non-null  datetime64[ns]
 11  start_scan_to_end_scan          144316 non-null  float64
 12  actual_distance_to_destination  144316 non-null  float64
 13  actual_time                     144316 non-null  float64
 14  osrm_time                       144316 non-null  float64
 15  osrm_distance                   144316 non-null  float64
 16  segment_actual_time             144316 non-null  float64
 17  segment_osrm_time               144316 non-null  float64
 18  segment_osrm_distance           144316 non-null  float64
dtypes: datetime64[ns](3), float64(8), object(7), uint8(1)
memory usage: 21.1+ MB
```

In [44]:

```
for i in df.columns:
    print(i +':'+ str(df[i].nunique()))
```

```
data:2
trip_creation_time:14787
route_schedule_uuid:1497
route_type:2
trip_uuid:14787
source_center:1496
source_name:1496
destination_center:1466
destination_name:1466
od_start_time:26223
od_end_time:26223
start_scan_to_end_scan:1914
actual_distance_to_destination:143965
actual_time:3182
osrm_time:1531
osrm_distance:137544
segment_actual_time:746
segment_osrm_time:214
segment_osrm_distance:113497
```

we can make data and route_type as category data types

In [45]:

```
for i in df.columns:
    if df[i].nunique() < 10:
        df[i] = df[i].astype('category')
```

In [46]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 144316 entries, 0 to 144866
Data columns (total 19 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144316 non-null  category
 1   trip_creation_time              144316 non-null  datetime64[ns]
 2   route_schedule_uuid             144316 non-null  object
 3   route_type                      144316 non-null  category
 4   trip_uuid                       144316 non-null  object
 5   source_center                   144316 non-null  object
 6   source_name                     144316 non-null  object
 7   destination_center              144316 non-null  object
 8   destination_name                144316 non-null  object
 9   od_start_time                   144316 non-null  datetime64[ns]
 10  od_end_time                     144316 non-null  datetime64[ns]
 11  start_scan_to_end_scan          144316 non-null  float64
 12  actual_distance_to_destination  144316 non-null  float64
 13  actual_time                     144316 non-null  float64
 14  osrm_time                       144316 non-null  float64
 15  osrm_distance                   144316 non-null  float64
 16  segment_actual_time             144316 non-null  float64
 17  segment_osrm_time               144316 non-null  float64
 18  segment_osrm_distance           144316 non-null  float64
dtypes: category(2), datetime64[ns](3), float64(8), object(6)
memory usage: 20.1+ MB
```

In [47]:

```
df['route_type'].value_counts()
```

Out[47]:

```
1    99132
0    45184
Name: route_type, dtype: int64
```

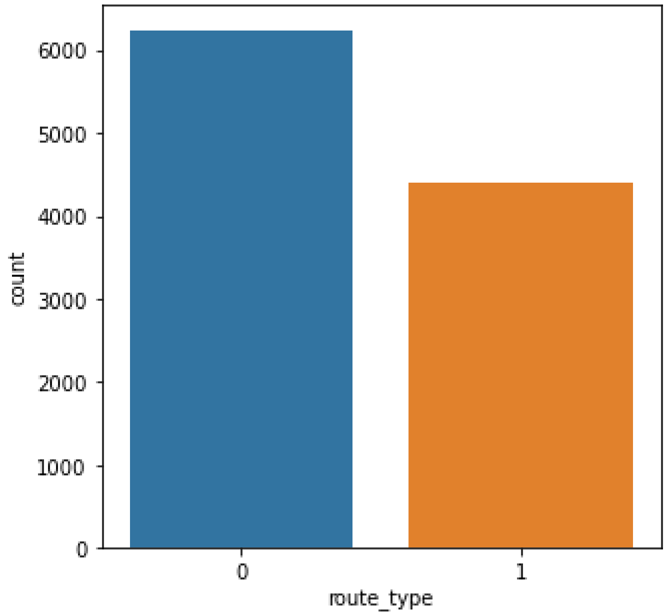
```
0:Carting
1:FTL
```

In [48]:

```
plt.figure(figsize=(5,5))
sns.countplot(x=df_final['route_type'])
```

Out[48]:

```
<AxesSubplot:xlabel='route_type', ylabel='count'>
```

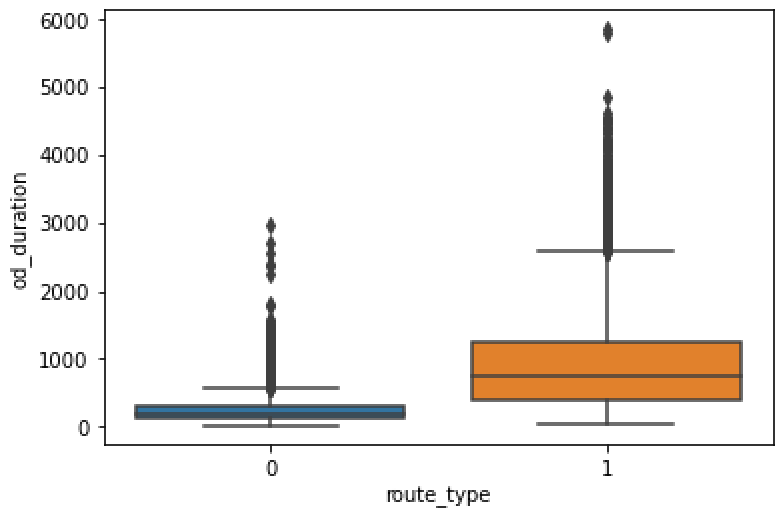


In [49]:

```
sns.boxplot(y=df_final['od_duration'],x=df_final['route_type'])
```

Out[49]:

```
<AxesSubplot:xlabel='route_type', ylabel='od_duration'>
```



we can see that most of the orders use FTL route type and we can infer that the distance between the source and destinations are large

In [50]:

```
df_final['source_city'].value_counts().sort_values(ascending=False).reset_index().head(5)
```

Out[50]:

	index	source_city
0	Gurgaon	749
1	Bengaluru	719
2	Bangalore	563
3	Bhiwandi	547
4	Delhi	467

We see that Bangalore and Bengaluru are same places but with different names we can replace one name with the other.

In [51]:

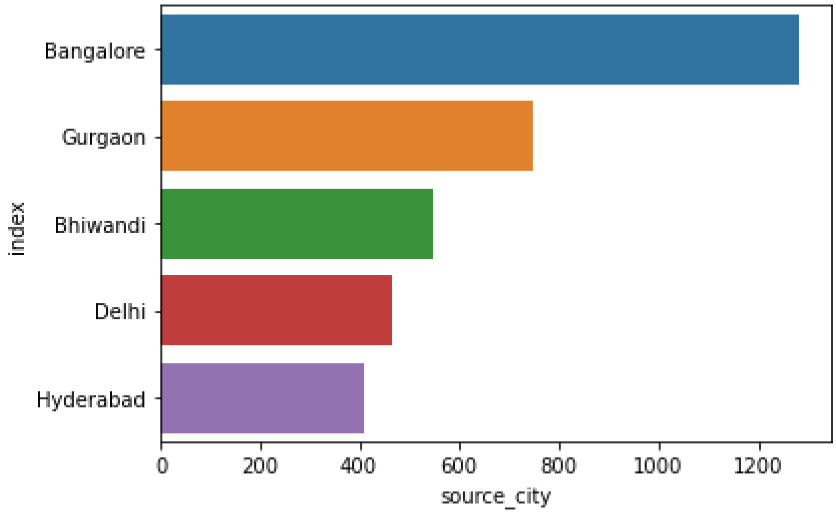
```
df_final['source_city'].replace('Bengaluru','Bangalore',inplace=True)
```

In [52]:

```
x=df_final['source_city'].value_counts().sort_values(ascending=False).reset_index().head()
sns.barplot(y=x['index'],x=x['source_city'])
```

Out[52]:

```
<AxesSubplot:xlabel='source_city', ylabel='index'>
```



Bangalore,Gurgaon and Bhiwandi are top three places from where most of the trips starts from.

We cant replace it in source name and centre , destination name and centre as they are different centers

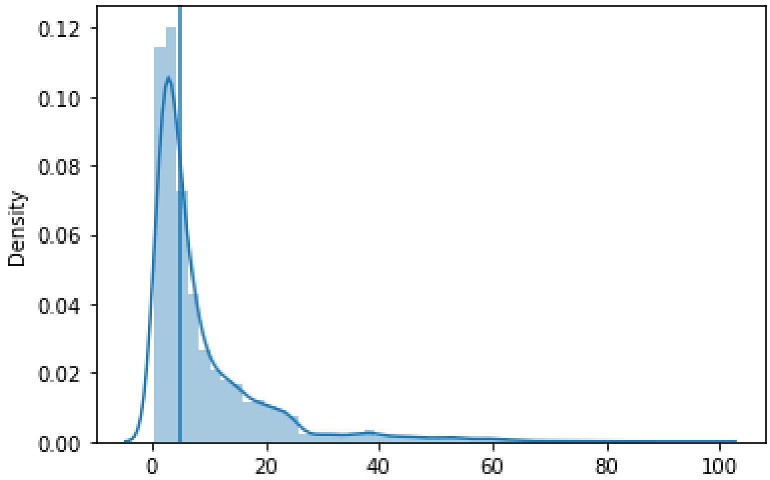
In [53]:

```
sns.distplot(x=df_final['start_scan_to_end_scan']/60)
plt.axvline(df_final['start_scan_to_end_scan'].median()/60)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[53]:

```
<matplotlib.lines.Line2D at 0x1d5d5116550>
```

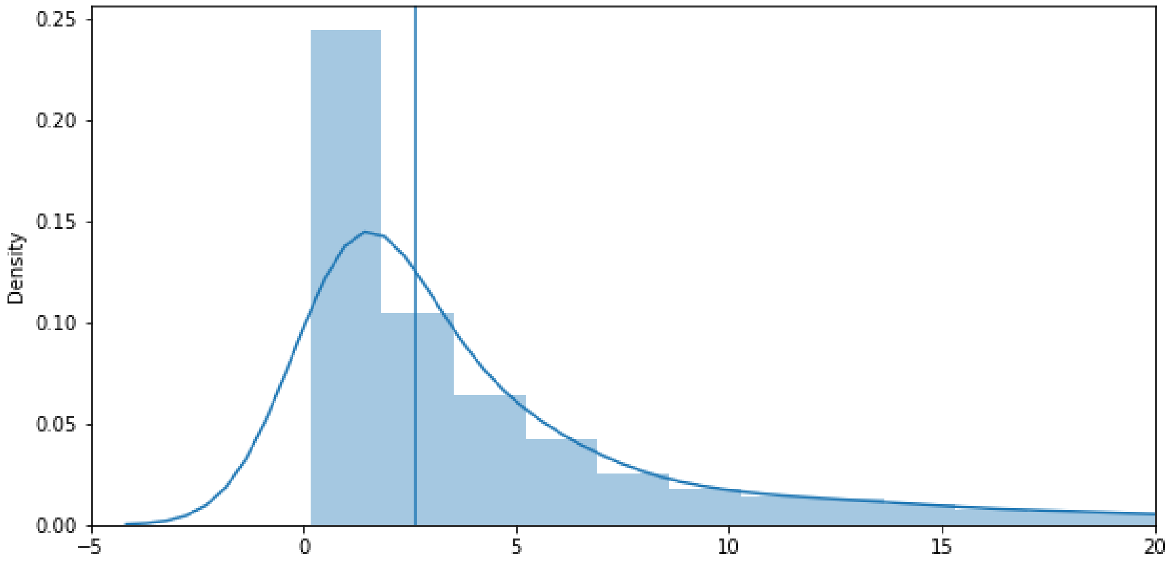


Most of the orders are delivered in 8 hours to the destination

In [54]:

```
plt.figure(figsize=[10,5])
sns.distplot(x=df_final['actual_time']/60)
plt.axvline(df_final['actual_time'].median()/60)
plt.xlim([-5,20])
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



From above we can see that most of the orders are delivered by 2-3 hours to the destination

In [55]:

```
df_final['od_month'].value_counts()
```

Out[55]:

```
9    10645
Name: od_month, dtype: int64
```

In [56]:

```
df_final['od_year'].value_counts()
```

Out[56]:

```
2018    10645
Name: od_year, dtype: int64
```

In [57]:

```
df_final['od_date'].value_counts().sort_values(ascending=False).head(10)
```

Out[57]:

```
2018-09-18    791
2018-09-15    783
2018-09-13    750
2018-09-12    747
2018-09-21    740
2018-09-22    740
2018-09-17    722
2018-09-14    712
2018-09-20    703
2018-09-25    695
Name: od_date, dtype: int64
```

These are the dates on which there are most number of orders took place, we can expect that there might be some offers occuring on the e-commerce platforms

In [58]:

```
df_final['od_hour'].value_counts().sort_values(ascending=False).head(3)
```

Out[58]:

```
22    826
20    784
23    731
Name: od_hour, dtype: int64
```

Most number of orders happens on 22,20 and 23rd hours of the day

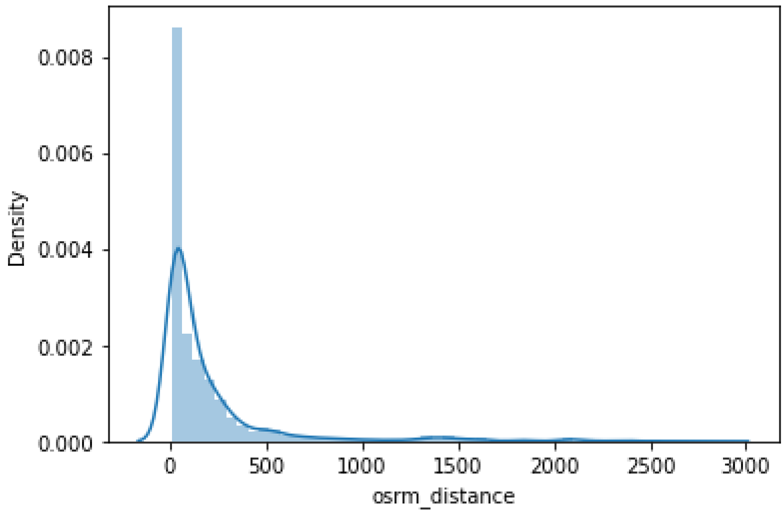
In [104]:

```
sns.distplot(df_final['osrm_distance'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[104]:

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



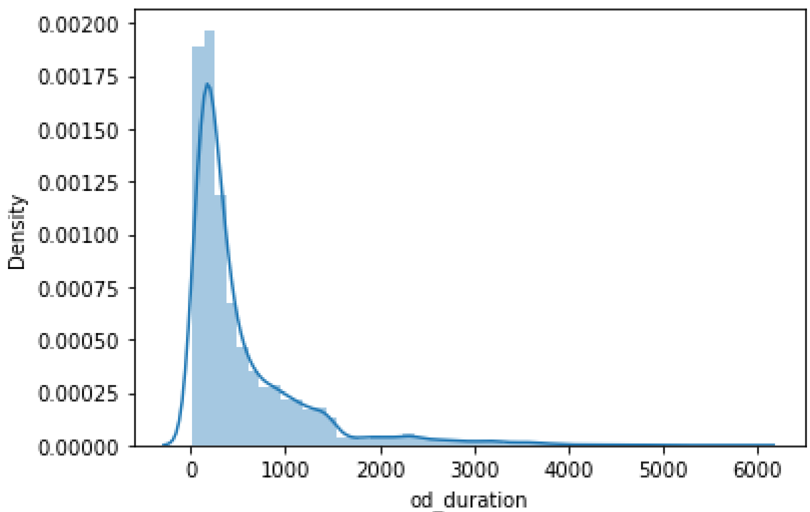
In [105]:

```
sns.distplot(df_final['od_duration'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[105]:

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



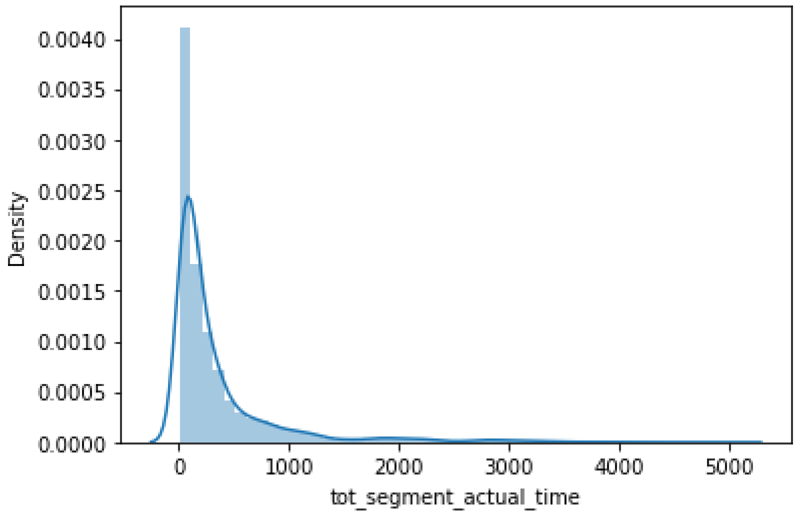
In [106]:

```
sns.distplot(df_final['tot_segment_actual_time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[106]:

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



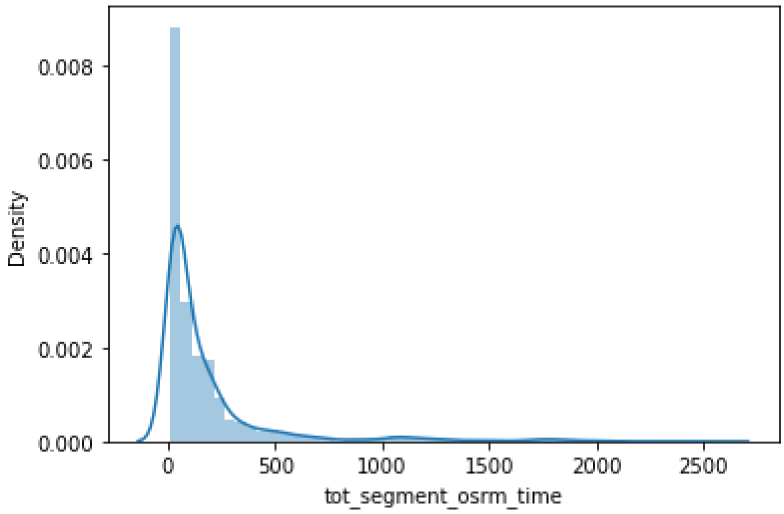
In [107]:

```
sns.distplot(df_final['tot_segment_osrm_time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[107]:

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



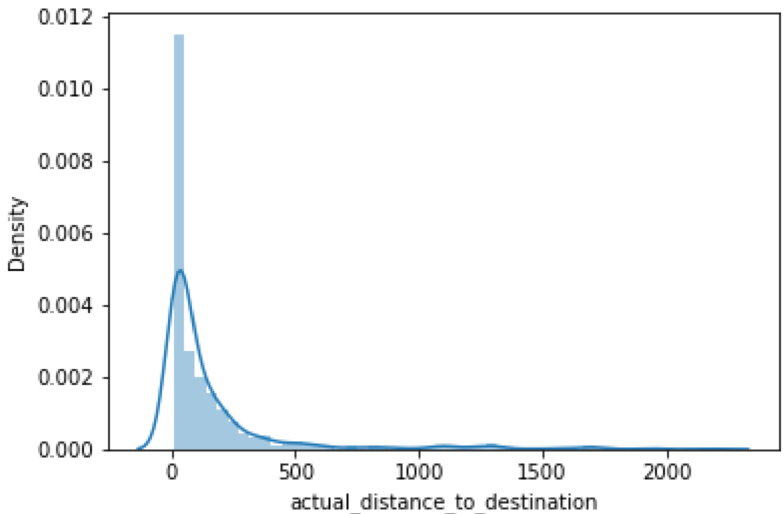
In [108]:

```
sns.distplot(df_final['actual_distance_to_destination'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[108]:

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



We have many outliers we can remove them using the IQR method

In [64]:

```
df_final.columns
```

Out[64]:

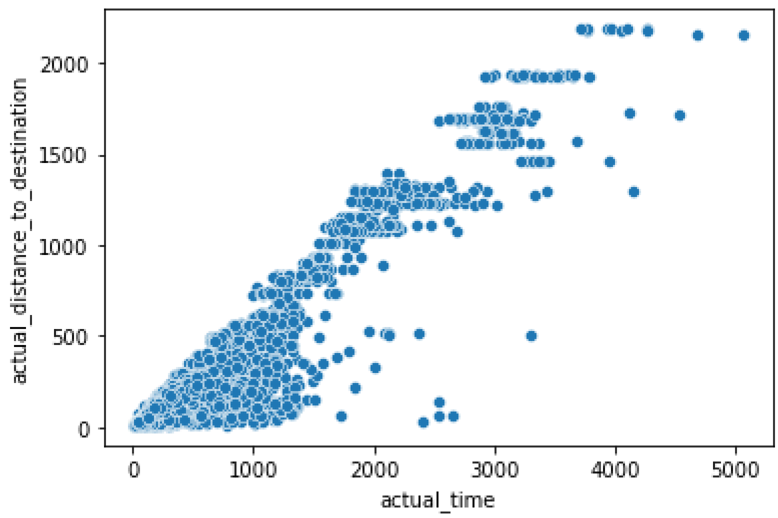
```
Index(['trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'trip_creation_time', 'route_schedule_uuid',
       'route_type', 'od_start_time', 'od_end_time', 'start_scan_to_end_sca
n',
       'actual_distance_to_destination', 'actual_time', 'osrm_time',
       'osrm_distance', 'tot_segment_actual_time', 'tot_segment_osrm_time',
       'tot_segment_osrm_distance', 'od_year', 'od_month', 'od_date',
       'od_hour', 'source_city', 'source_state', 'destination',
       'destination_state', 'od_duration'],
      dtype='object')
```

In [65]:

```
sns.scatterplot(x = df_final['actual_time'],y = df_final['actual_distance_to_destination'])
```

Out[65]:

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

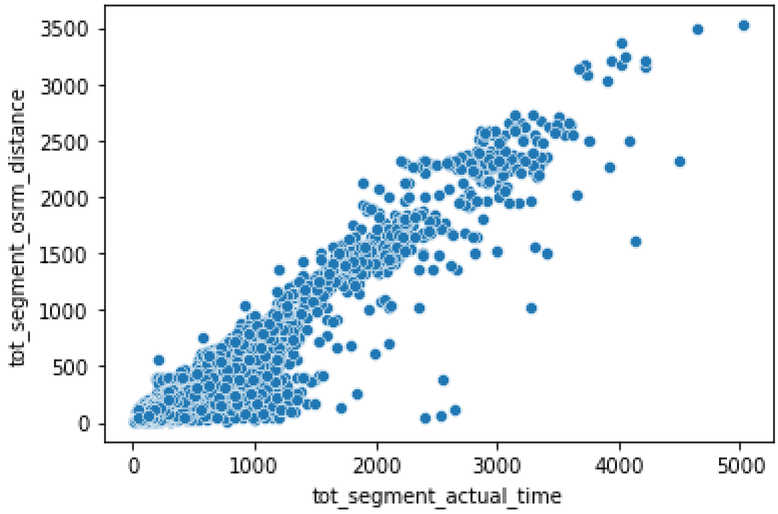


In [66]:

```
sns.scatterplot(x = df_final['tot_segment_actual_time'],y = df_final['tot_segment_osrm_dist
```

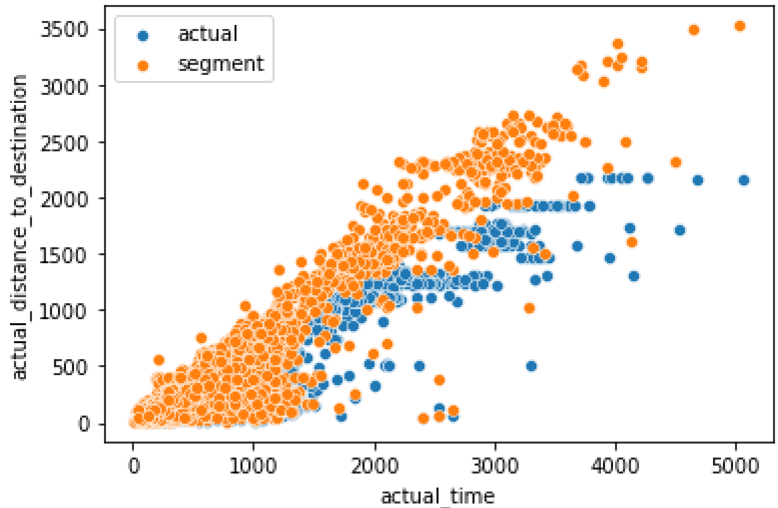
Out[66]:

```
<AxesSubplot:xlabel='tot_segment_actual_time', ylabel='tot_segment_osrm_dist
ance'>
```



In [67]:

```
sns.scatterplot(x = df_final['actual_time'],y = df_final['actual_distance_to_destination'])
sns.scatterplot(x = df_final['tot_segment_actual_time'],y = df_final['tot_segment_osrm_dist
plt.legend(['actual','segment'])
plt.show()
```



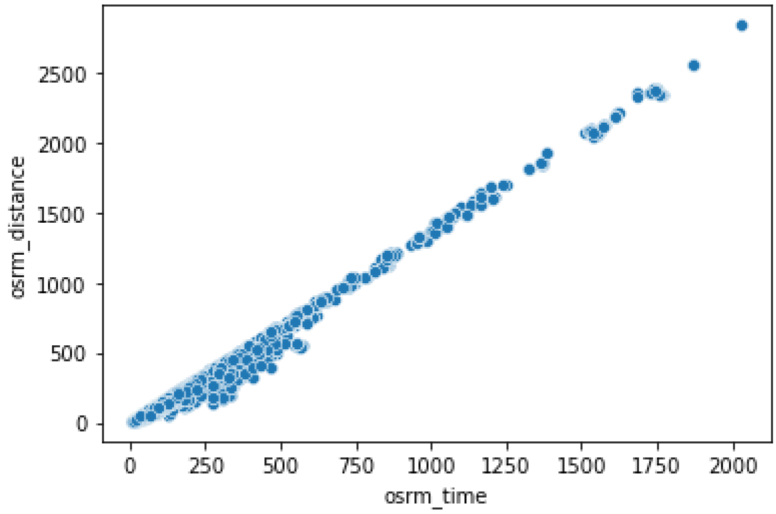
Segment time and distance is higher compared to actual time and distance when the distance is more

In [68]:

```
sns.scatterplot(x = df_final['osrm_time'],y = df_final['osrm_distance'])
```

Out[68]:

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



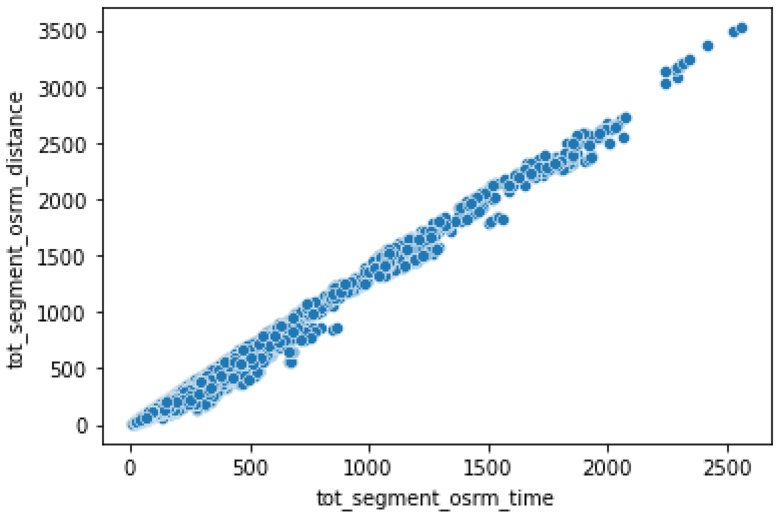
osrm time and distance turned out to be a linear plot, we can infer that if the distance increases then the time taken will also increase but there is a small variation in the small distance deliveries they might some extra time

In [69]:

```
sns.scatterplot(x = df_final['tot_segment_osrm_time'],y = df_final['tot_segment_osrm_distan
```

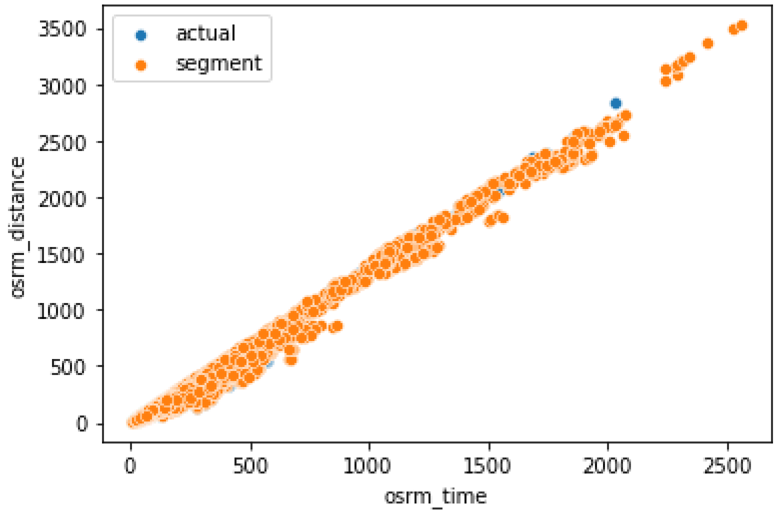
Out[69]:

```
<AxesSubplot:xlabel='tot_segment_osrm_time', ylabel='tot_segment_osrm_distan
ce'>
```



In [70]:

```
sns.scatterplot(x = df_final['osrm_time'],y = df_final['osrm_distance'])
sns.scatterplot(x = df_final['tot_segment_osrm_time'],y = df_final['tot_segment_osrm_distan
plt.legend(['actual','segment'])
plt.show()
```



From above plot it is evident that actual osrm and segment osrm distance and time are equal

In [94]:

```
inal.groupby([df_final['destination_name'],df_final['source_name']]).agg(count = ('destinati
                                                          dist = ('od_duration','median')).reset_index
```

Out[94]:

	destination_name	source_name	count	dist
390	Chandigarh_Mehmdpur_H (Punjab)	Chandigarh_Mehmdpur_H (Punjab)	138	934.154999
218	Bengaluru_KGAirprt_HB (Karnataka)	Bangalore_Nelmngla_H (Karnataka)	100	181.276543
1307	Muzaffrpur_Bbganj_I (Bihar)	Muzaffrpur_Bbganj_I (Bihar)	88	1073.862303
219	Bengaluru_KGAirprt_HB (Karnataka)	Bengaluru_Bomsndra_HB (Karnataka)	81	208.515774
1647	Sonipat_Kundli_H (Haryana)	Sonipat_Kundli_H (Haryana)	76	1318.409778

Chandigarh_Mehmdpur_H (Punjab) to Chandigarh_Mehmdpur_H (Punjab) and from Bengaluru_KGAirprt_HB (Karnataka) to Bangalore_Nelmngla_H (Karnataka) has more number of orders and large delivery time

In [79]:

```
df_final.groupby([df_final['source_state'],df_final['destination_state']]).
                    agg(Avg_time = ('od_duration','median')).reset_index().sort_values(by='
```

Out[79]:

	source_state	destination_state	Avg_time
136	Uttar Pradesh	Rajasthan	67.255231
19	Dadra and Nagar Haveli	Gujarat	69.595762
31	Gujarat	Dadra and Nagar Haveli	72.974409
90	Maharashtra	Madhya Pradesh	100.735564
101	Pondicherry	Tamil Nadu	155.109176
20	Delhi	Delhi	161.146246
112	Rajasthan	Madhya Pradesh	175.318115
17	Chandigarh	Punjab	181.086133
67	Karnataka	Karnataka	187.084762
22	Delhi	Haryana	192.198034

```
Uttar Pradesh to Rajasthan and Dadra and Nagar Haveli to Gujarat are the fastest delivery
trips
```

In [90]:

```
x = df_final.groupby([df_final['source_name']]).agg(count = ('source_state','count')).reset
y = df_final.groupby([df_final['destination_name']]).agg(count = ('destination_state','coun
x=pd.merge(x,y,how='inner',left_on='source_name',right_on='destination_name')
x['count'] = x['count_x']+x['count_y']
x.drop(columns=['destination_name','count_x','count_y'],inplace=True)
x.sort_values(by='count',ascending=False).head()
```

Out[90]:

	source_name	count
153	Gurgaon_Bilaspur_HB (Haryana)	1294
33	Bangalore_Nelmngla_H (Karnataka)	1032
55	Bhiwandi_Mankoli_HB (Maharashtra)	954
74	Chandigarh_Mehmdpur_H (Punjab)	618
180	Hyderabad_Shamshbd_H (Telangana)	552

```
Gurgaon_Bilaspur_HB (Haryana),Bangalore_Nelmngla_H (Karnataka),Bhiwandi_Mankoli_HB
(Maharashtra) are the top 3 busiest centers
```

In [138]:

```
by(df_final['destination_name']).agg(count = ('destination_center','count')).reset_index().
```

Out[138]:

	destination_name	count
316	Gurgaon_Bilaspur_HB (Haryana)	608
65	Bangalore_Nelmngla_H (Karnataka)	485
118	Bhiwandi_Mankoli_HB (Maharashtra)	407
167	Chandigarh_Mehmdpur_H (Punjab)	322
356	Hyderabad_Shamshbd_H (Telangana)	304
...	...	...
195	Chinnur_AsnsdhRD_D (Telangana)	1
84	Bellmpalli_BasthDPP_D (Telangana)	1
571	Mumbai_Skynet_INT (Maharashtra)	1
568	Mumbai_Panvel_D (Maharashtra)	1
329	Haldwani_PiliKoti_D (Uttarakhand)	1

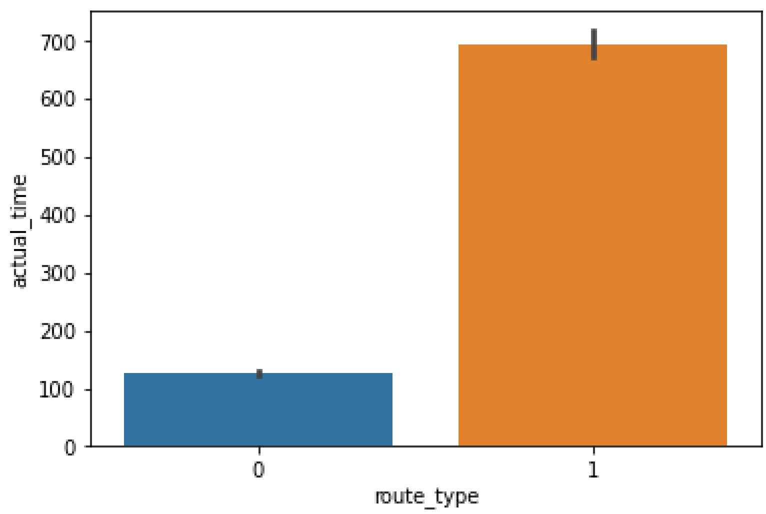
851 rows × 2 columns

In [111]:

```
sns.barplot(x=df_final['route_type'],y=df_final['actual_time'])
```

Out[111]:

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

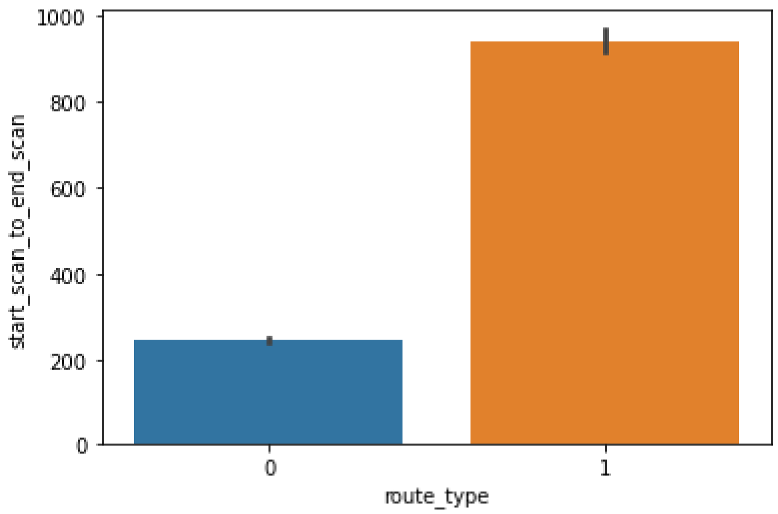


In [116]:

```
sns.barplot(x=df_final['route_type'],y=df_final['start_scan_to_end_scan'])
```

Out[116]:

<AxesSubplot:xlabel='route_type', ylabel='start_scan_to_end_scan'>

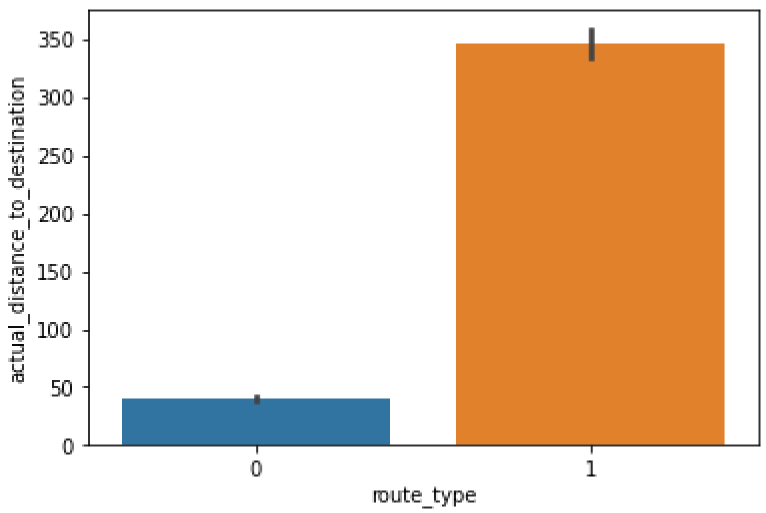


In [117]:

```
sns.barplot(x=df_final['route_type'],y=df_final['actual_distance_to_destination'])
```

Out[117]:

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

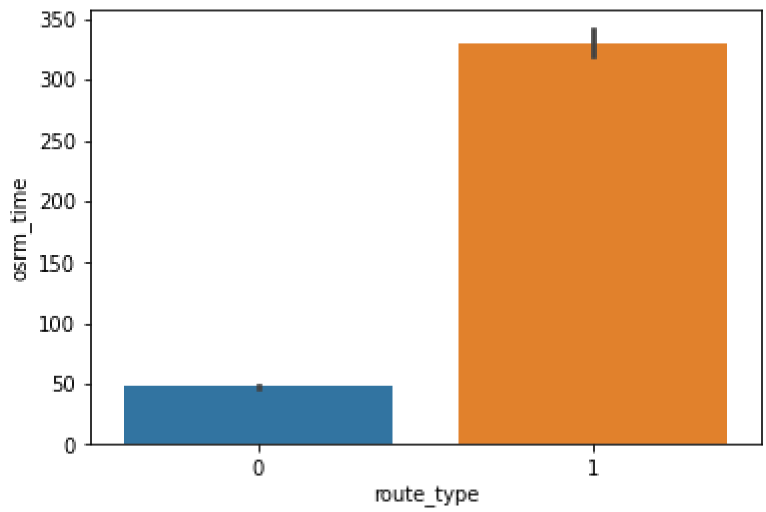


In [118]:

```
sns.barplot(x=df_final['route_type'],y=df_final['osrm_time'])
```

Out[118]:

<AxesSubplot:xlabel='route_type', ylabel='osrm_time'>

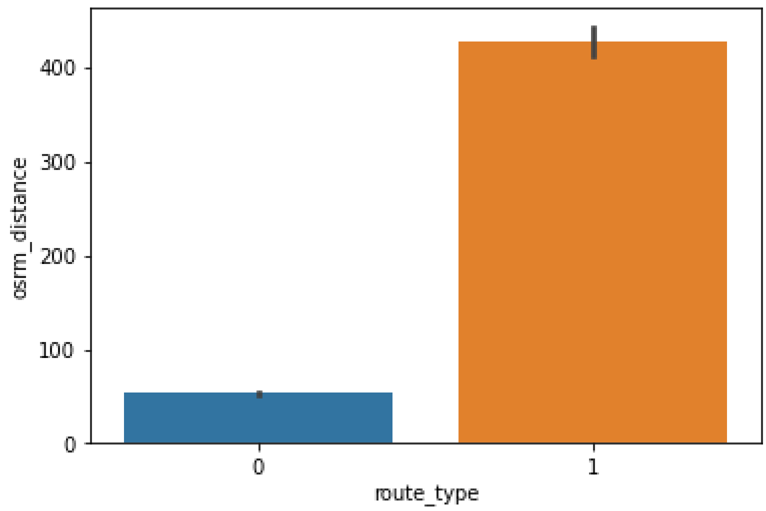


In [119]:

```
sns.barplot(x=df_final['route_type'],y=df_final['osrm_distance'])
```

Out[119]:

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

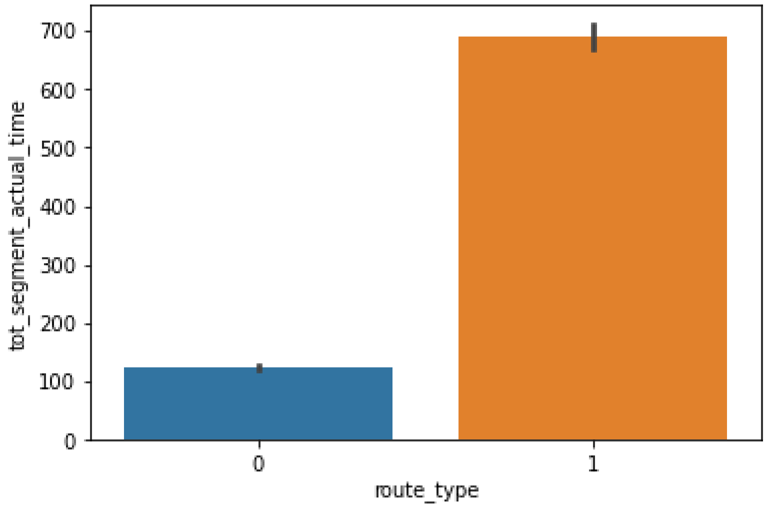


In [120]:

```
sns.barplot(x=df_final['route_type'],y=df_final['tot_segment_actual_time'])
```

Out[120]:

```
<AxesSubplot:xlabel='route_type', ylabel='tot_segment_actual_time'>
```

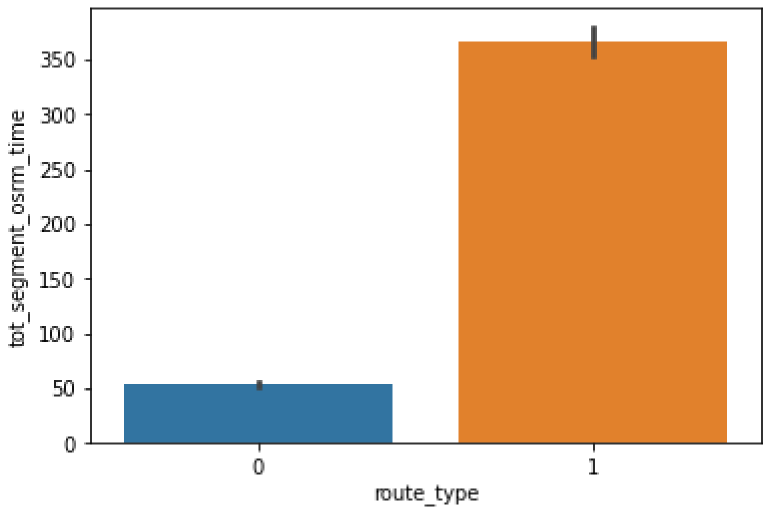


In [121]:

```
sns.barplot(x=df_final['route_type'],y=df_final['tot_segment_osrm_time'])
```

Out[121]:

```
<AxesSubplot:xlabel='route_type', ylabel='tot_segment_osrm_time'>
```

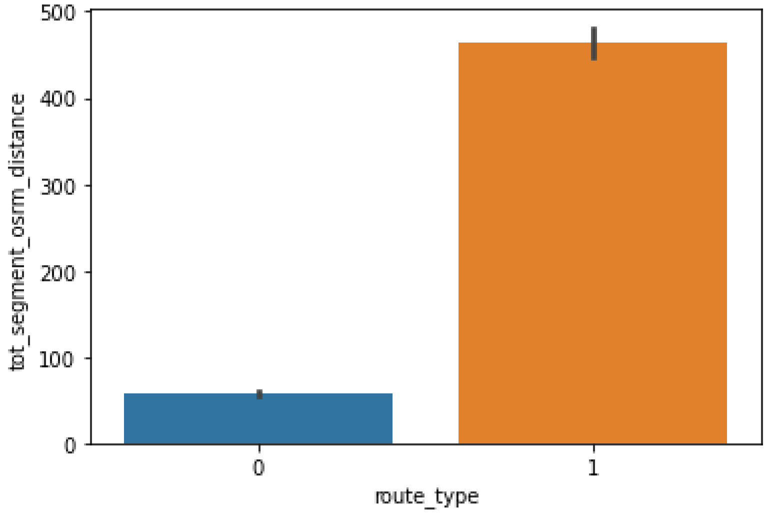


In [122]:

```
sns.barplot(x=df_final['route_type'],y=df_final['tot_segment_osrm_distance'])
```

Out[122]:

```
<AxesSubplot:xlabel='route_type', ylabel='tot_segment_osrm_distance'>
```

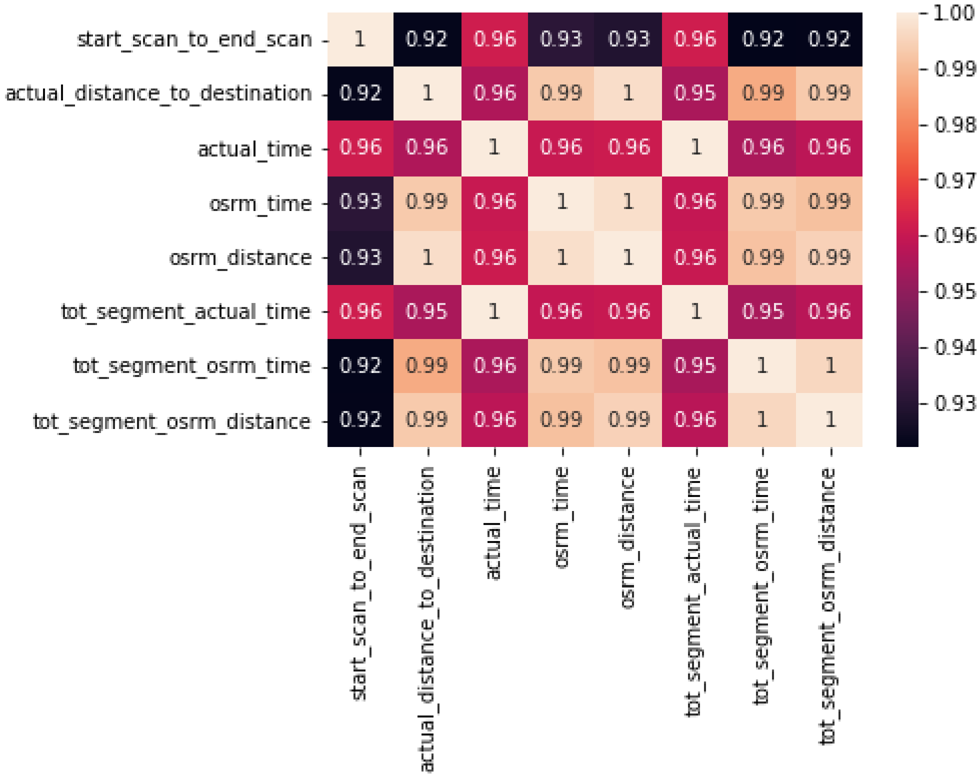


In [126]:

```
sns.heatmap(df_final[['start_scan_to_end_scan','actual_distance_to_destination', 'actual_ti
                      'tot_segment_actual_time', 'tot_segment_osrm_time','tot_segment_osrm_d
```

Out[126]:

<AxesSubplot:>



From above heatmap we can infer that if distance increases then the time taken is also increases

In [ ]:

```

```

Outliers

```
From above plots we see that there are many outliers in the sample.
Lets take a copy of this sample data and perform some IQR methods on the data
```

In [72]:

```
df_final_copy = df_final.copy()
```

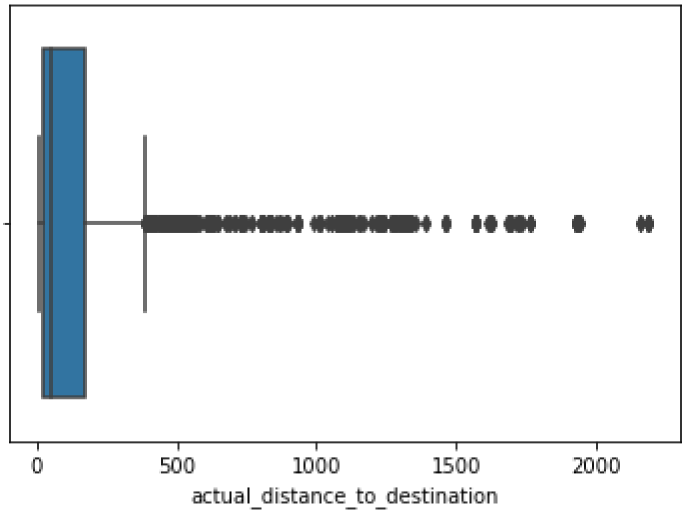
```
Now, we copied our final sample to final_copy and lets perform the further analysis on
final dataframe leaving final_copy untouched
```

In [73]:

```
sns.boxplot(x=df_final['actual_distance_to_destination'])
```

Out[73]:

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




```
There are more outliers in this feature, by performing IQR Method we can remove them.
```

In [74]:

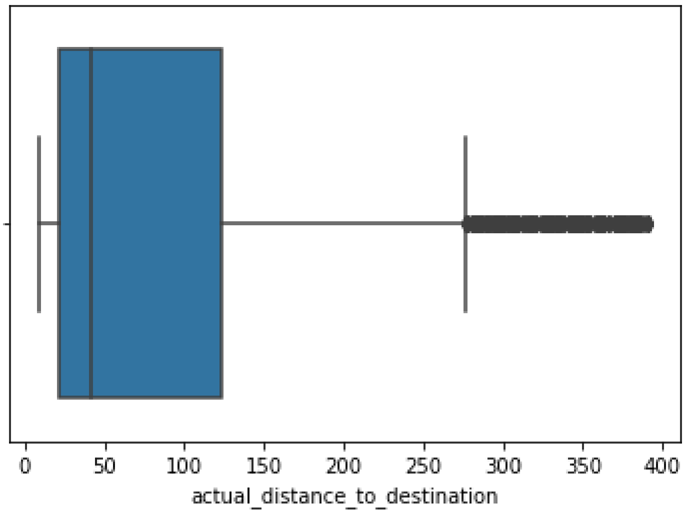
```
## Finding Quartiles
q1=df_final['actual_distance_to_destination'].quantile(0.25)
q3=df_final['actual_distance_to_destination'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['actual_distance_to_destination'] > q1 -1.5* iqr) & (df_final

## Plotting
sns.boxplot(x=df_final['actual_distance_to_destination'])
```

Out[74]:

<AxesSubplot:xlabel='actual_distance_to_destination'>



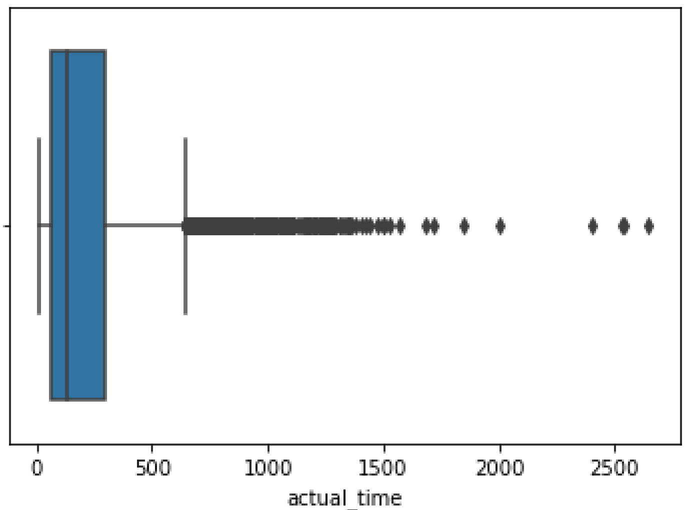
There are still some outliers but we can get some insights from these, Lets perform the same for each feature

In [75]:

```
sns.boxplot(x=df_final['actual_time'])
```

Out[75]:

<AxesSubplot:xlabel='actual_time'>



In [76]:

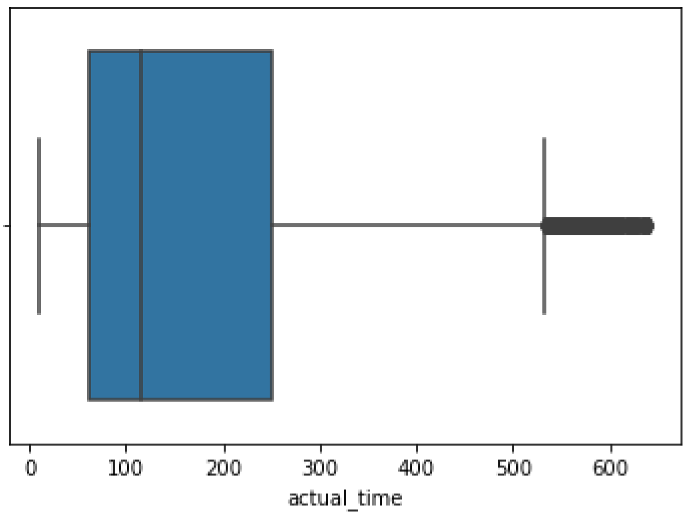
```
## Finding Quartiles
q1=df_final['actual_time'].quantile(0.25)
q3=df_final['actual_time'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['actual_time'] > q1 -1.5* iqr) & (df_final['actual_time'] < q

## Plotting
sns.boxplot(x=df_final['actual_time'])
```

Out[76]:

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

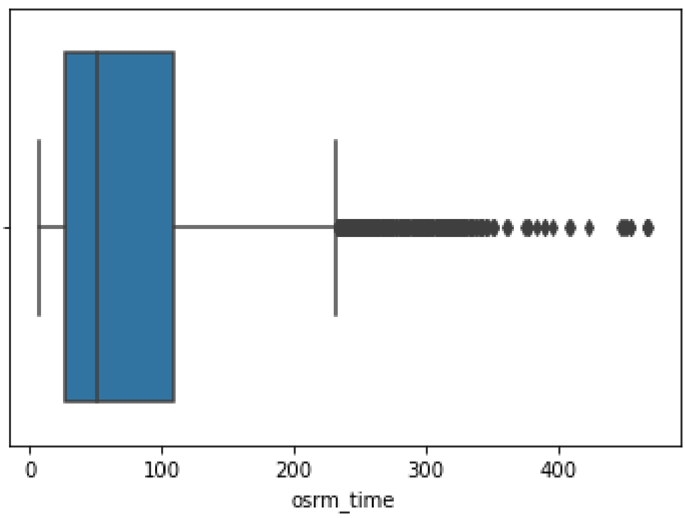


In [77]:

```
sns.boxplot(x=df_final['osrm_time'])
```

Out[77]:

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



In [78]:

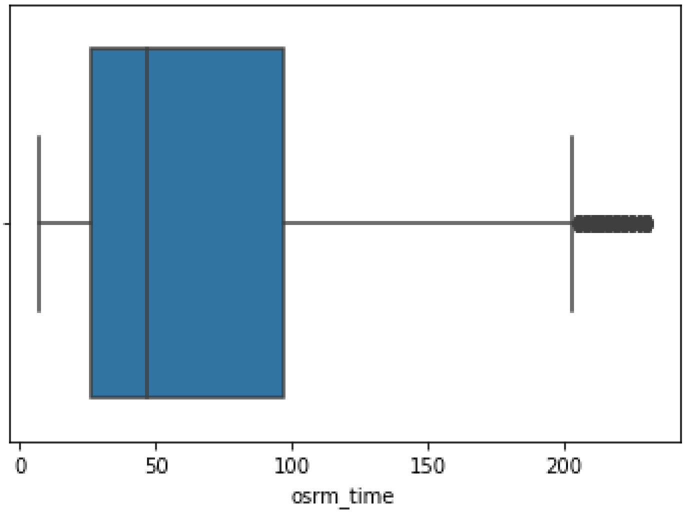
```
## Finding Quartiles
q1=df_final['osrm_time'].quantile(0.25)
q3=df_final['osrm_time'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['osrm_time'] > q1 -1.5* iqr) & (df_final['osrm_time'] < q3 +1

## Plotting
sns.boxplot(x=df_final['osrm_time'])
```

Out[78]:

<AxesSubplot:xlabel='osrm_time'>

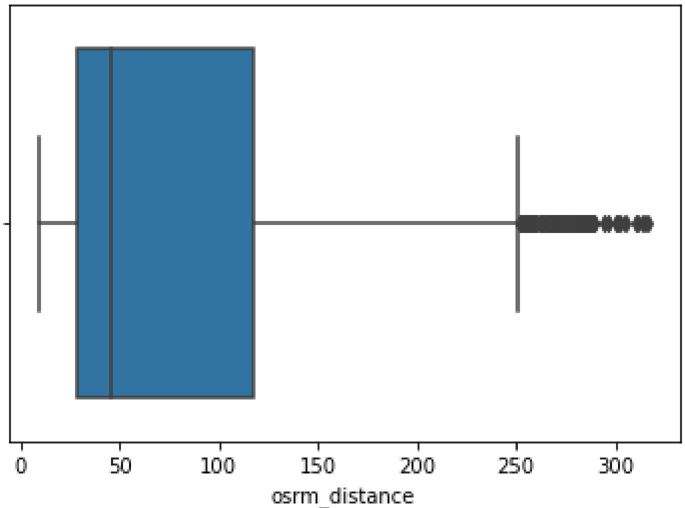


In [79]:

```
sns.boxplot(x=df_final['osrm_distance'])
```

Out[79]:

<AxesSubplot:xlabel='osrm_distance'>



In [80]:

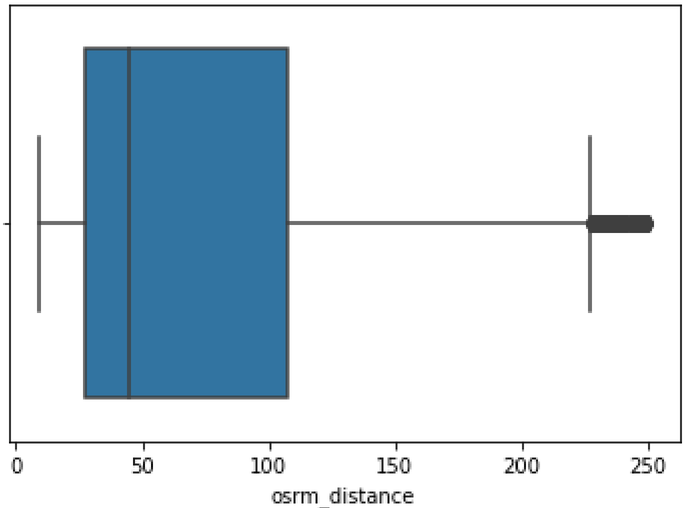
```
## Finding Quartiles
q1=df_final['osrm_distance'].quantile(0.25)
q3=df_final['osrm_distance'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['osrm_distance'] > q1 -1.5* iqr) & (df_final['osrm_distance']

## Plotting
sns.boxplot(x=df_final['osrm_distance'])
```

Out[80]:

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

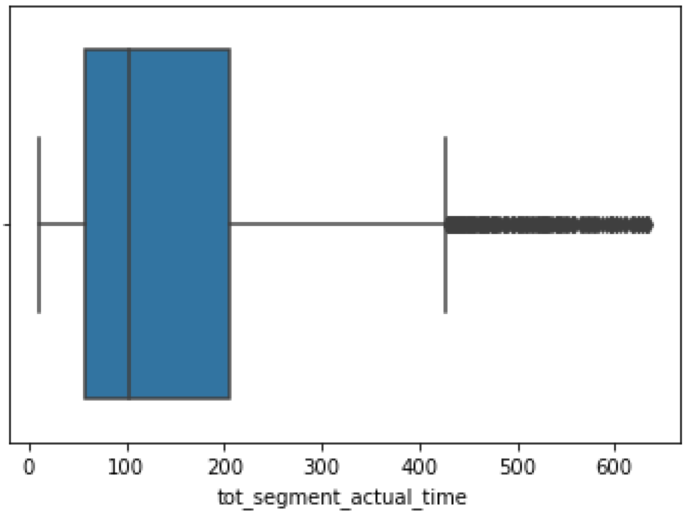


In [81]:

```
sns.boxplot(x=df_final['tot_segment_actual_time'])
```

Out[81]:

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



In [82]:

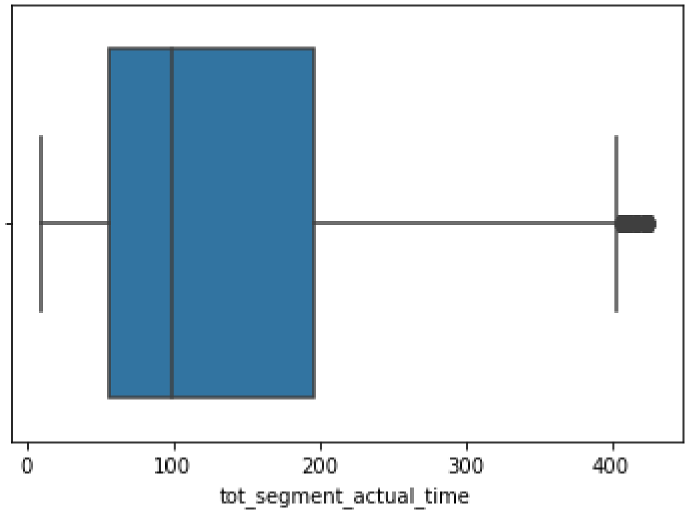
```
## Finding Quartiles
q1=df_final['tot_segment_actual_time'].quantile(0.25)
q3=df_final['tot_segment_actual_time'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['tot_segment_actual_time'] > q1 -1.5* iqr) & (df_final['tot_s

## Plotting
sns.boxplot(x=df_final['tot_segment_actual_time'])
```

Out[82]:

<AxesSubplot:xlabel='tot_segment_actual_time'>

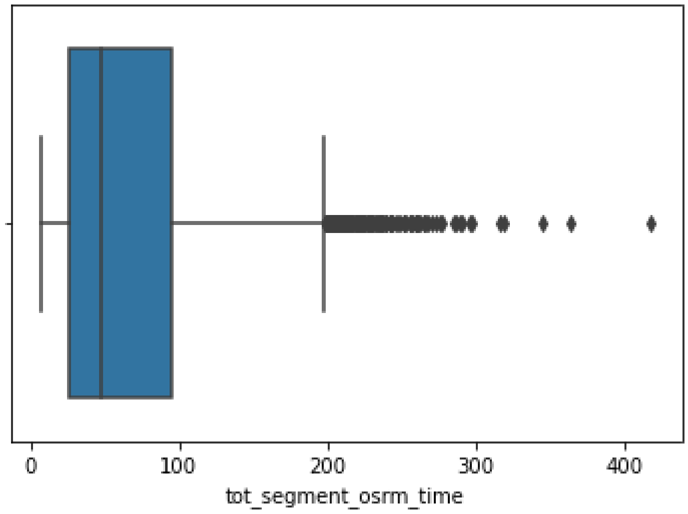


In [83]:

```
sns.boxplot(x=df_final['tot_segment_osrm_time'])
```

Out[83]:

<AxesSubplot:xlabel='tot_segment_osrm_time'>



In [84]:

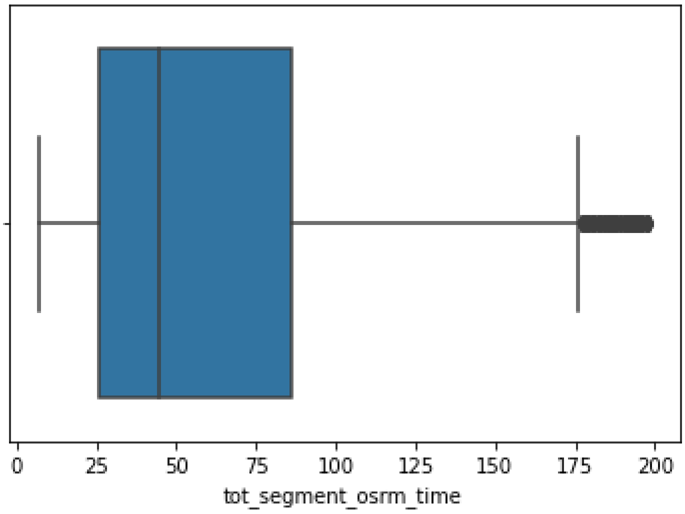
```
## Finding Quartiles
q1=df_final['tot_segment_osrm_time'].quantile(0.25)
q3=df_final['tot_segment_osrm_time'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['tot_segment_osrm_time'] > q1 -1.5* iqr) & (df_final['tot_seg

## Plotting
sns.boxplot(x=df_final['tot_segment_osrm_time'])
```

Out[84]:

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

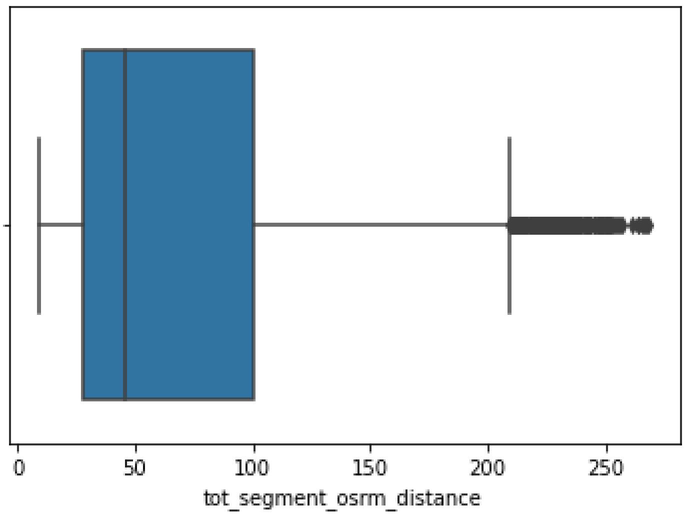


In [85]:

```
sns.boxplot(x=df_final['tot_segment_osrm_distance'])
```

Out[85]:

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



In [86]:

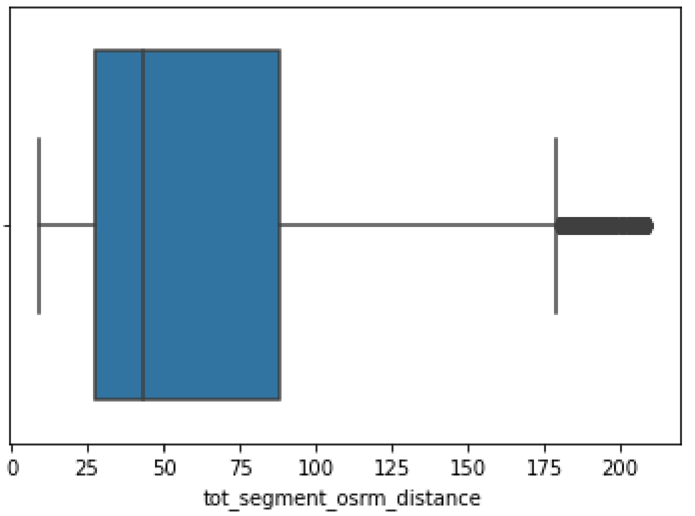
```
## Finding Quartiles
q1=df_final['tot_segment_osrm_distance'].quantile(0.25)
q3=df_final['tot_segment_osrm_distance'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['tot_segment_osrm_distance'] > q1 -1.5* iqr) & (df_final['tot

## Plotting
sns.boxplot(x=df_final['tot_segment_osrm_distance'])
```

Out[86]:

<AxesSubplot:xlabel='tot_segment_osrm_distance'>

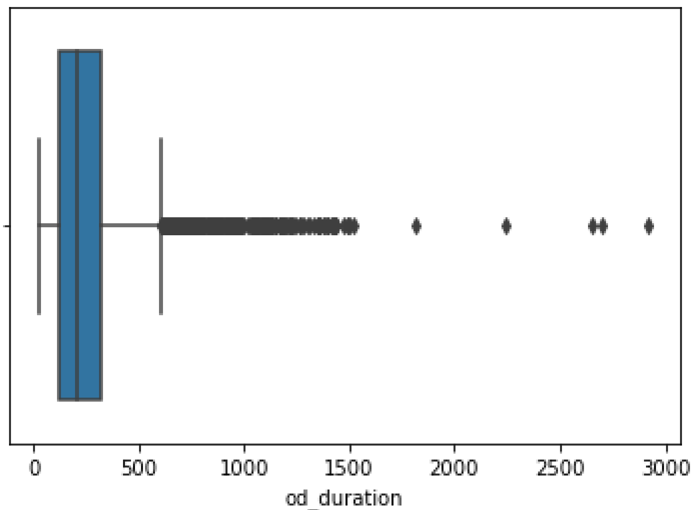


In [87]:

```
sns.boxplot(x=df_final['od_duration'])
```

Out[87]:

<AxesSubplot:xlabel='od_duration'>



In [88]:

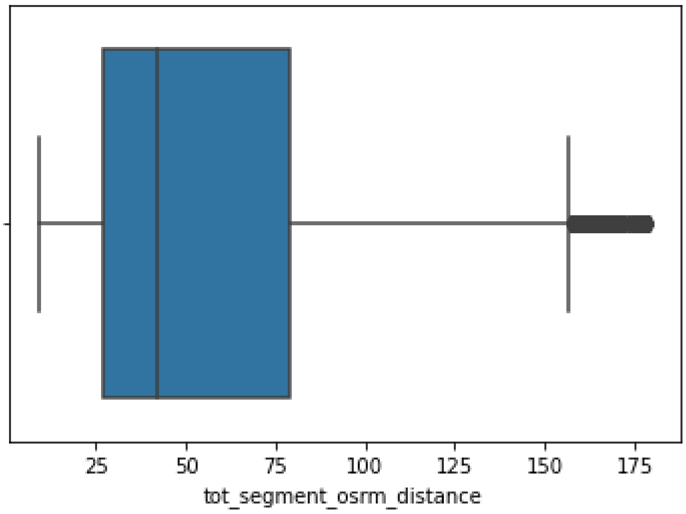
```
## Finding Quartiles
q1=df_final['tot_segment_osrm_distance'].quantile(0.25)
q3=df_final['tot_segment_osrm_distance'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['tot_segment_osrm_distance'] > q1 -1.5* iqr) & (df_final['tot

## Plotting
sns.boxplot(x=df_final['tot_segment_osrm_distance'])
```

Out[88]:

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

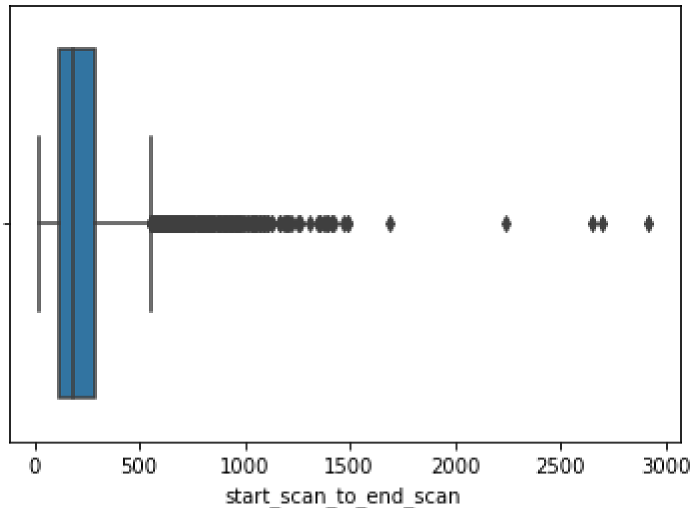


In [89]:

```
sns.boxplot(x=df_final['start_scan_to_end_scan'])
```

Out[89]:

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



In [90]:

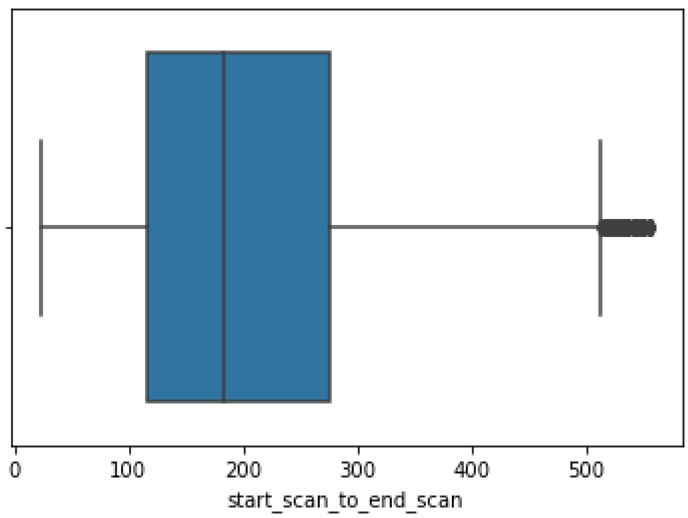
```
## Finding Quartiles
q1=df_final['start_scan_to_end_scan'].quantile(0.25)
q3=df_final['start_scan_to_end_scan'].quantile(0.75)
iqr=q3-q1

## Removing the outliers
df_final = df_final[(df_final['start_scan_to_end_scan'] > q1 -1.5* iqr) & (df_final['start_

## Plotting
sns.boxplot(x=df_final['start_scan_to_end_scan'])
```

Out[90]:

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

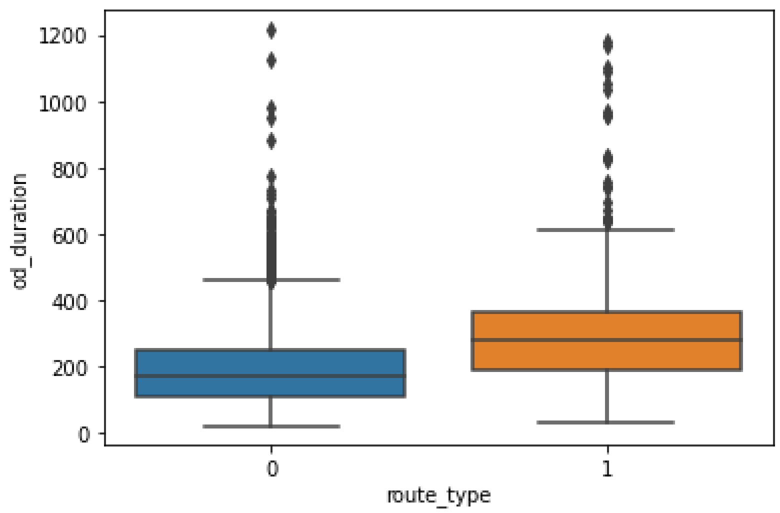


In [91]:

```
sns.boxplot(y=df_final['od_duration'],x=df_final['route_type'])
```

Out[91]:

```
<AxesSubplot:xlabel='route_type', ylabel='od_duration'>
```



In [92]:

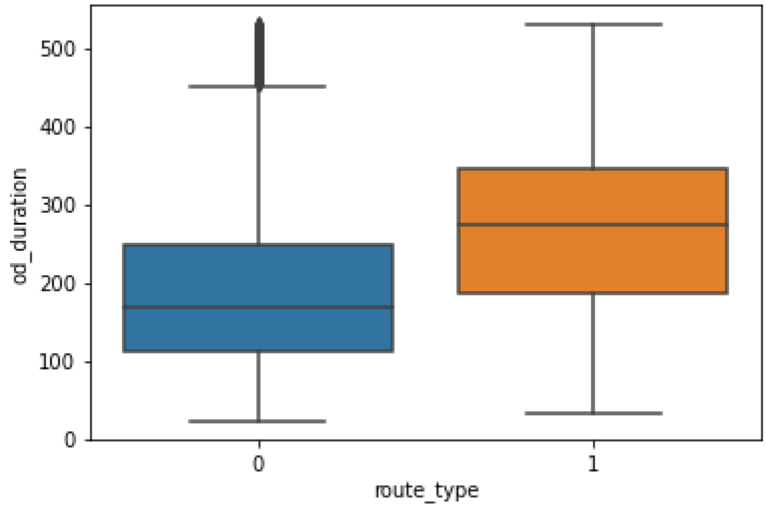
```
q1=df_final['od_duration'].quantile(0.25)
q3=df_final['od_duration'].quantile(0.75)
iqr=q3-q1

df_final = df_final[(df_final['od_duration'] > q1 -1.5* iqr) & (df_final['od_duration'] < q

sns.boxplot(y=df_final['od_duration'],x=df_final_copy['route_type'])
```

Out[92]:

```
<AxesSubplot:xlabel='route_type', ylabel='od_duration'>
```



The time to deliver the products for each type of route type has significant difference

In [93]:

```
df_final.shape
```

Out[93]:

```
(6737, 27)
```

Hypothesis Testing

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

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

We will use kstest to check the difference between the two features.

p-value - 0.05

Null Hypothesis : Both distributions are similar
Alternate Hypothesis : Both distributions are different
```

In [94]:

```
stat.ks_2samp(df_final['start_scan_to_end_scan'],df_final['od_duration'])
```

Out[94]:

```
KstestResult(statistic=0.019147988719014398, pvalue=0.16905771224625468)
```

As we see that p-value = 0.17 i.e., p-value > 0.05

we are failed to reject the null hypothesis and conclude that both the distributions are similar

Do hypothesis testing/ visual analysis between actual_time aggregated value and OSRM time aggregated value

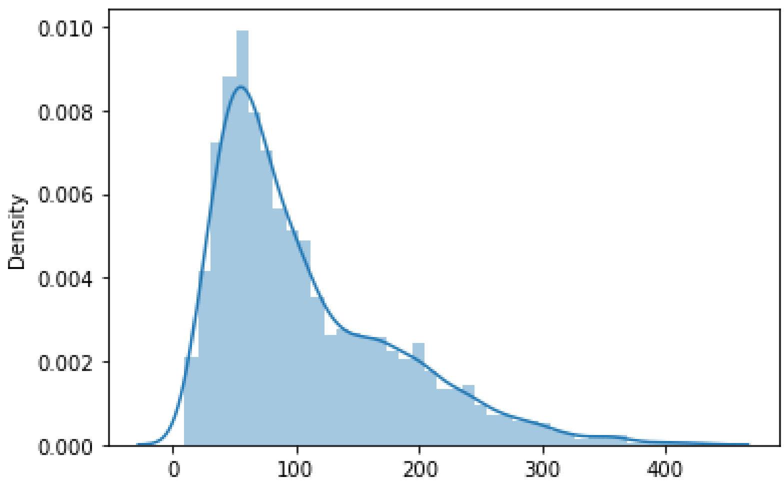
In [95]:

```
sns.distplot(x=df_final['actual_time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[95]:

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



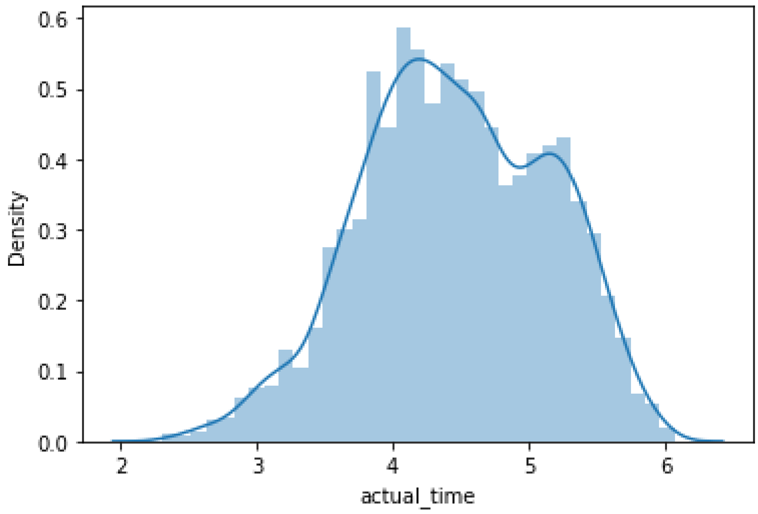
In [96]:

```
sns.distplot(np.log(df_final['actual_time']))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[96]:

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



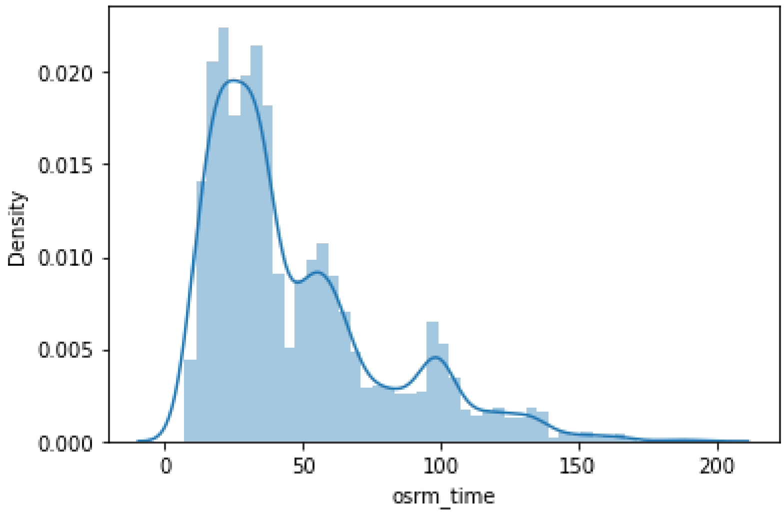
In [97]:

```
sns.distplot(df_final['osrm_time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[97]:

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



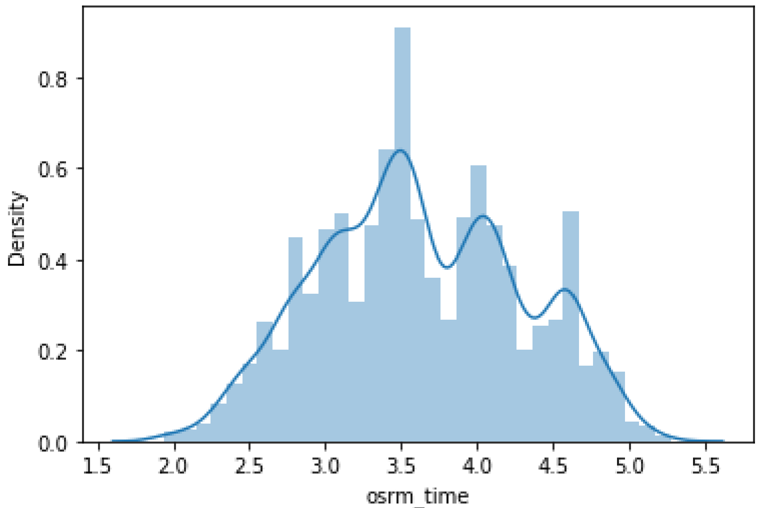
In [98]:

```
sns.distplot(np.log(df_final['osrm_time']))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[98]:

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



In [99]:

```
stat.shapiro(np.log(df_final['actual_time']).sample(4999))
```

Out[99]:

```
ShapiroResult(statistic=0.9909916520118713, pvalue=2.589737723525509e-17)
```

In [100]:

```
stat.shapiro(np.log(df_final['osrm_time']).sample(4999))
```

Out[100]:

```
ShapiroResult(statistic=0.9879415035247803, pvalue=2.9077392165762146e-20)
```

In [101]:

```
stat.levene(np.log(df_final['osrm_time']),np.log(df_final['actual_time']))
```

Out[101]:

```
LeveneResult(statistic=6.4884424804779695, pvalue=0.010868823316476829)
```

```
Assumptions:

As we have just sample data of a single month we can assume that the population data forms
the normal distribution
From levenes test we can cofirm that they dont have equal variances as p value < 0.05

p-value: 0.05
```

```
Ho : Both osrm_time and actual_time are similar
Ha : osrm_time and actual_time are different
```

In [102]:

```
stat.ttest_ind(np.log(df_final['osrm_time']).sample(30),np.log(df_final['actual_time']).sam
```

Out[102]:

```
Ttest_indResult(statistic=-3.244742650072782, pvalue=0.0019551427756188547)
```

```
p value < 0.05

We reject the null hypothesis and the times are different
```

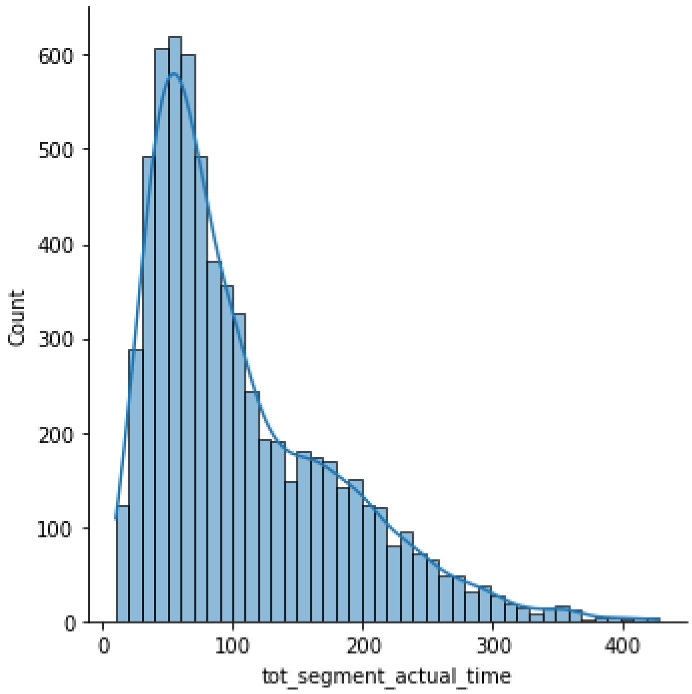
Do hypothesis testing/ visual analysis between actual_time aggregated value and segment actual time aggregated value

In [103]:

```
sns.displot(df_final['tot_segment_actual_time'],kde=True)
```

Out[103]:

```
<seaborn.axisgrid.FacetGrid at 0x19ac2ef0040>
```

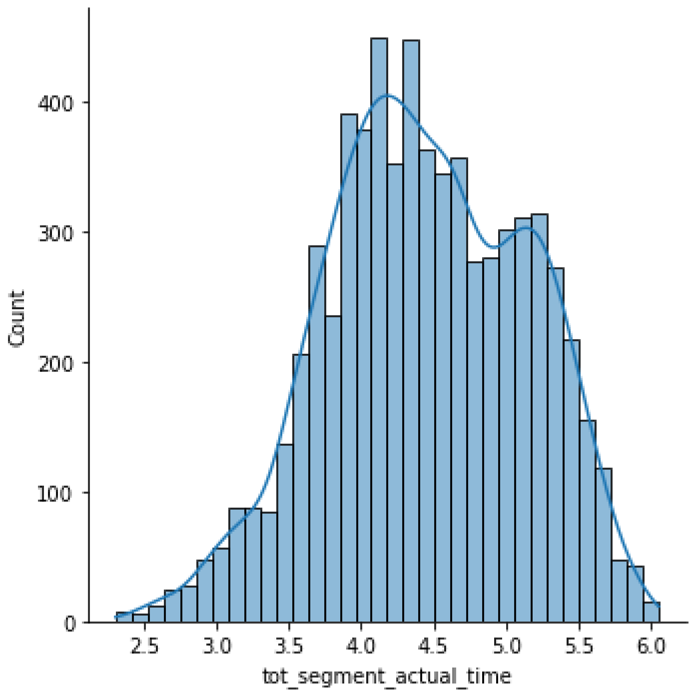


In [104]:

```
sns.displot(np.log(df_final['tot_segment_actual_time']),kde=True)
```

Out[104]:

```
<seaborn.axisgrid.FacetGrid at 0x19ac301d2e0>
```

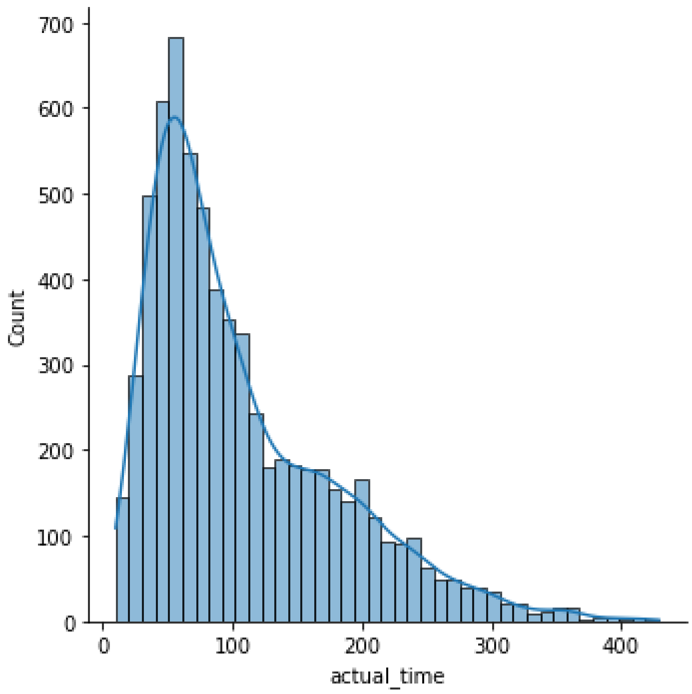


In [105]:

```
sns.displot(df_final['actual_time'],kde=True)
```

Out[105]:

<seaborn.axisgrid.FacetGrid at 0x19ac2fb4af0>

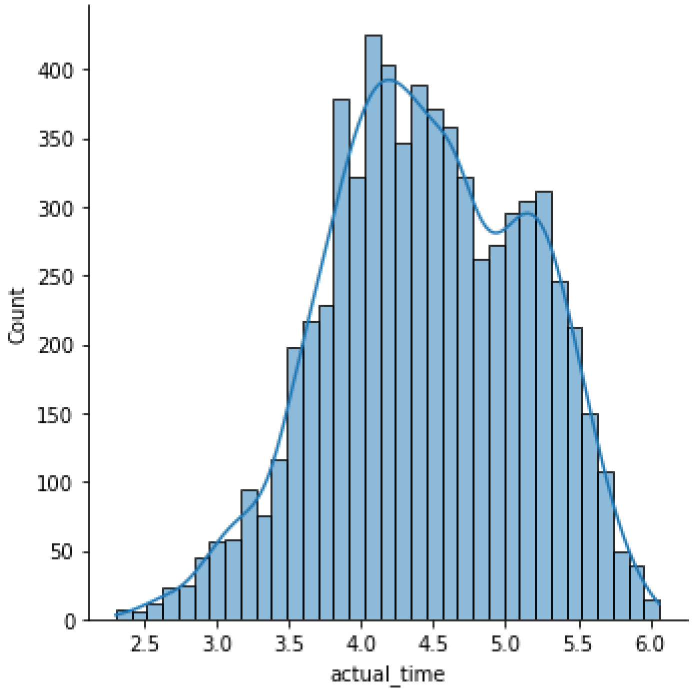


In [106]:

```
sns.displot(np.log(df_final['actual_time']),kde=True)
```

Out[106]:

```
<seaborn.axisgrid.FacetGrid at 0x19ac2753fa0>
```



In [107]:

```
stat.shapiro(np.log(df_final['tot_segment_actual_time']).sample(4999))
```

Out[107]:

```
ShapiroResult(statistic=0.990931510925293, pvalue=2.2322736070441883e-17)
```

In [108]:

```
stat.levene(np.log(df_final['tot_segment_actual_time']),np.log(df_final['actual_time']))
```

Out[108]:

```
LeveneResult(statistic=0.026364703352395363, pvalue=0.8710152645322702)
```

```
Assumptions:
```

```
As we have just sample data of a single month we can assume that the population data forms
the normal distribution
From levenes test we can cofirm that they have equal variances as p value > 0.05

p-value: 0.05

Ho : Both tot_segment_actual_time and actual_time are similar
Ha : tot_segment_actual_time and actual_time are different
```

In [109]:

```
stat.ttest_ind(np.log(df_final['tot_segment_actual_time']),np.log(df_final['actual_time']))
```

Out[109]:

```
Ttest_indResult(statistic=-0.9760413172396049, pvalue=0.32906151520940685)
```

```
p value > 0.05

We failed to reject the null hypothesis and the times are similar
```

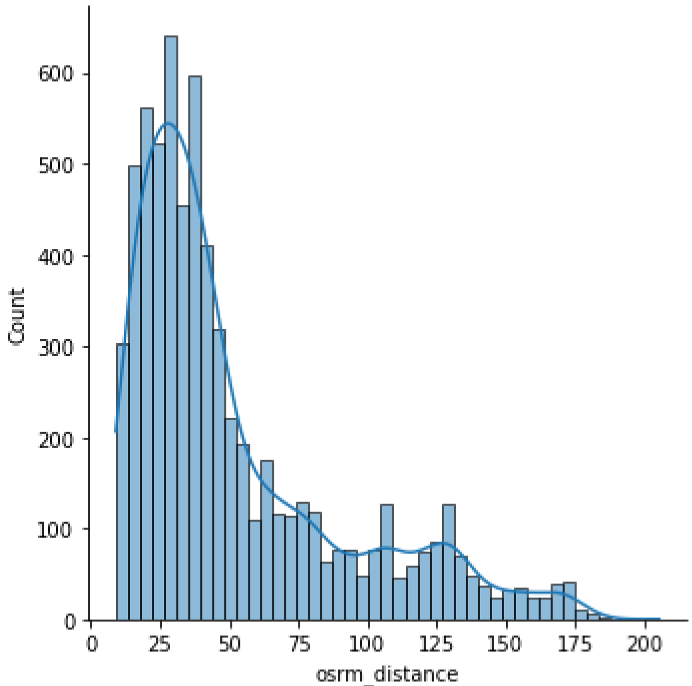
Do hypothesis testing/ visual analysis between osrm distance aggregated value and segment osrm distance aggregated value

In [110]:

```
sns.displot(df_final['osrm_distance'],kde=True)
```

Out[110]:

```
<seaborn.axisgrid.FacetGrid at 0x19ac3f5b4c0>
```

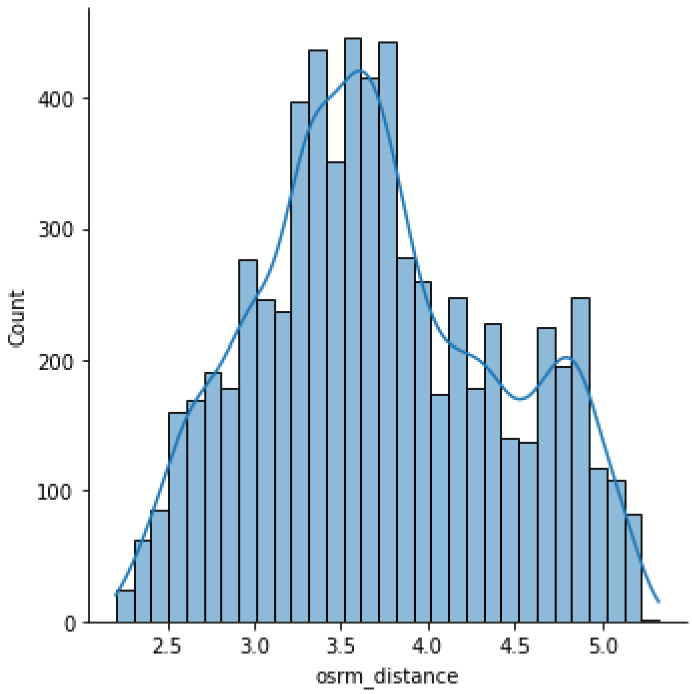


In [111]:

```
sns.displot(np.log(df_final['osrm_distance']),kde=True)
```

Out[111]:

<seaborn.axisgrid.FacetGrid at 0x19ac32b73d0>

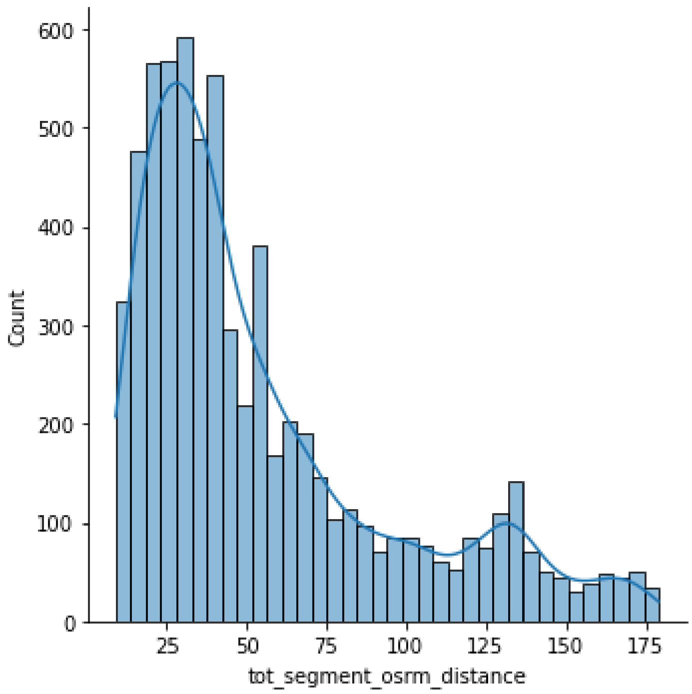


In [112]:

```
sns.displot(df_final['tot_segment_osrm_distance'],kde=True)
```

Out[112]:

<seaborn.axisgrid.FacetGrid at 0x19ac408fdf0>

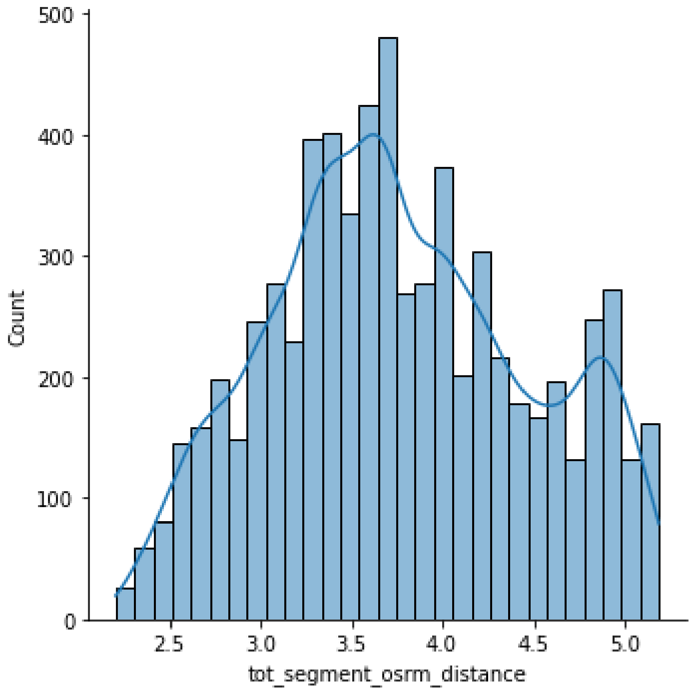


In [113]:

```
sns.displot(np.log(df_final['tot_segment_osrm_distance']),kde=True)
```

Out[113]:

<seaborn.axisgrid.FacetGrid at 0x19ac4025190>



In [114]:

```
stat.shapiro(np.log(df_final['tot_segment_osrm_distance']).sample(4999))
```

Out[114]:

```
ShapiroResult(statistic=0.9792553782463074, pvalue=2.0266776508940654e-26)
```

In [115]:

```
stat.shapiro(np.log(df_final['osrm_distance']).sample(4999))
```

Out[115]:

```
ShapiroResult(statistic=0.9776108860969543, pvalue=2.3501932444292945e-27)
```

In [116]:

```
stat.levene(np.log(df_final['osrm_distance']),np.log(df_final['tot_segment_osrm_distance'])
```

Out[116]:

```
LeveneResult(statistic=4.383476947299729, pvalue=0.03630754599786417)
```

```
Assumptions:

As we have just sample data of a single month we can assume that the population data forms
the normal distribution
From levenes test we can cofirm that they have equal variances as p value < 0.05

p-value: 0.05

Ho : Both osrm_distance and tot_segment_osrm_distance are similar
Ha : osrm_distance and tot_segment_osrm_distance are different
```

In [117]:

```
for i in range(5):
    print(stat.ttest_ind(np.log(df_final['osrm_distance']).sample(30),np.log(df_final['tot_
```

```
Ttest_indResult(statistic=-0.7541937780003664, pvalue=0.4538339230490047)
Ttest_indResult(statistic=0.535593311996646, pvalue=0.5942875847258196)
Ttest_indResult(statistic=-1.2514795237800307, pvalue=0.21587246822212136)
Ttest_indResult(statistic=1.3855475932353403, pvalue=0.1712623408881227)
Ttest_indResult(statistic=-1.7319078846965779, pvalue=0.08866186805757442)
```

```
We have tested to 5 different samples from the dataset

p value > 0.05

We failed to reject the null hypothesis and the distances are similar
```

Do hypothesis testing/ visual analysis between osrm time aggregated value and segment osrm time aggregated value

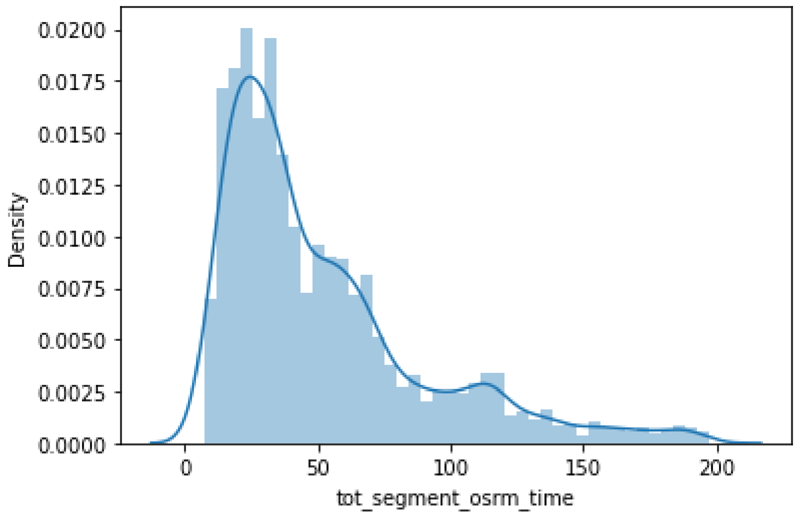
In [118]:

```
sns.distplot(df_final['tot_segment_osrm_time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[118]:

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



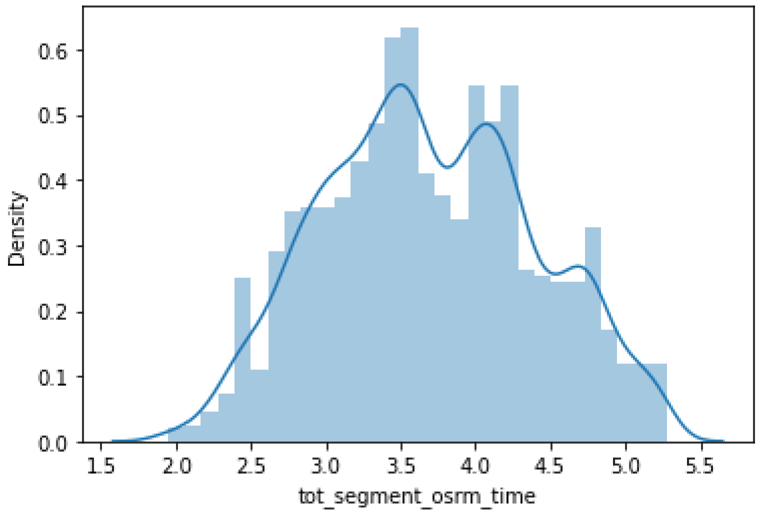
In [119]:

```
sns.distplot(np.log(df_final['tot_segment_osrm_time']))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[119]:

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



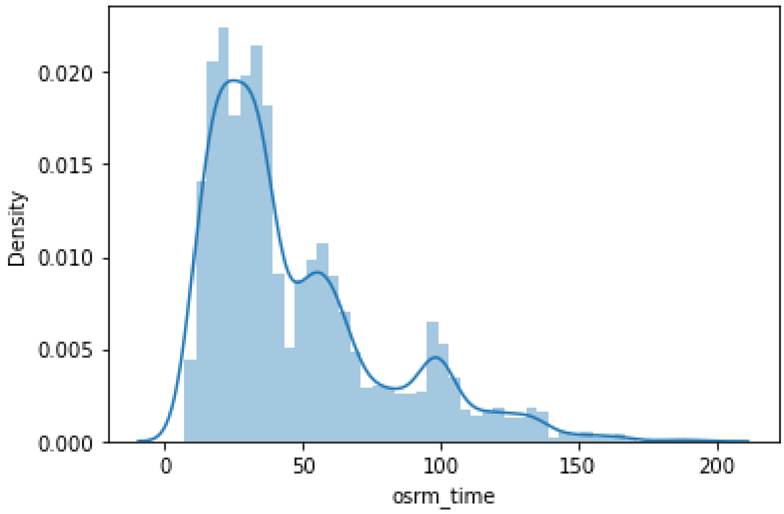
In [120]:

```
sns.distplot(df_final['osrm_time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[120]:

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



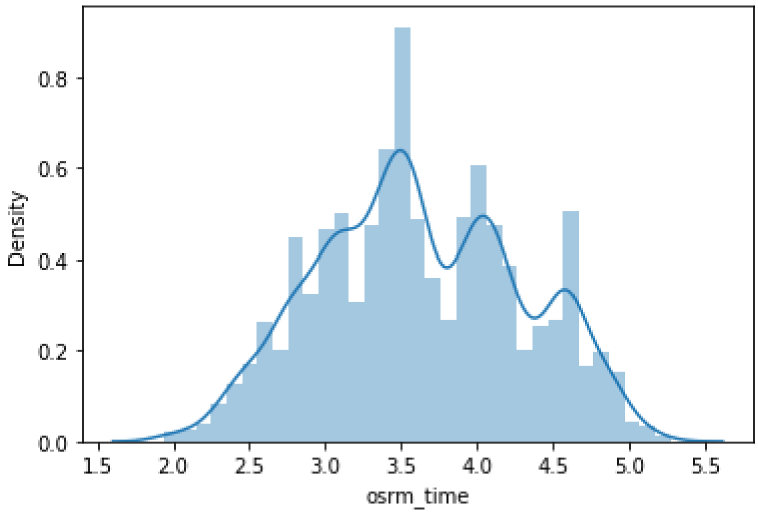
In [121]:

```
sns.distplot(np.log(df_final['osrm_time']))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[121]:

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



In [122]:

```
stat.shapiro(np.log(df_final['tot_segment_osrm_time']).sample(4999))
```

Out[122]:

```
ShapiroResult(statistic=0.989574134349823, pvalue=9.257258442513435e-19)
```

In [123]:

```
stat.shapiro(np.log(df_final['osrm_time']).sample(4999))
```

Out[123]:

```
ShapiroResult(statistic=0.9874498248100281, pvalue=1.0958402321190847e-20)
```

In [124]:

```
stat.levene(np.log(df_final['osrm_time']),np.log(df_final['tot_segment_osrm_time']))
```

Out[124]:

```
LeveneResult(statistic=29.346420986540565, pvalue=6.156793575167786e-08)
```

```
Assumptions:

As we have just sample data of a single month we can assume that the population data forms
the normal distribution
From levenes test we can cofirm that they dont have equal variances as p value < 0.05

p-value: 0.05

Ho : Both osrm_time and tot_segment_osrm_time are similar
Ha : osrm_time and tot_segment_osrm_time are different
```

In [125]:

```
stat.ttest_ind(np.log(df_final['osrm_time']).sample(50),np.log(df_final['tot_segment_osrm_t
```

Out[125]:

```
Ttest_indResult(statistic=0.2285668612239816, pvalue=0.8196835042689277)
```

```
p value > 0.05

We Failed to reject the null hypothesis and the times are similar
```

Do hypothesis testing/ visual analysis between osrm time aggregated value and tot_segment_actual_time

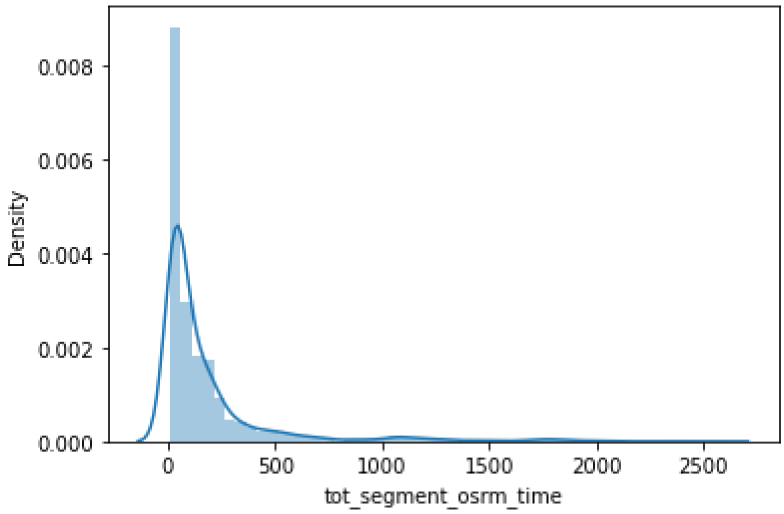
In [127]:

```
sns.distplot(df_final['tot_segment_osrm_time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[127]:

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



As the distribution is right skewed we will apply the log normal to it to make it normal

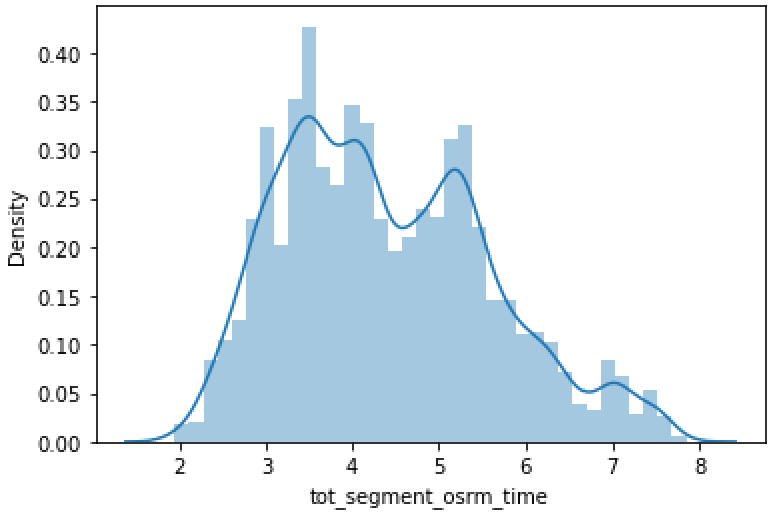
In [129]:

```
sns.distplot(np.log(df_final['tot_segment_osrm_time']))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[129]:

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



In [130]:

```
stat.shapiro(np.log(df_final['tot_segment_osrm_time']).sample(4999))
```

Out[130]:

```
ShapiroResult(statistic=0.9694132804870605, pvalue=2.316354876487428e-31)
```

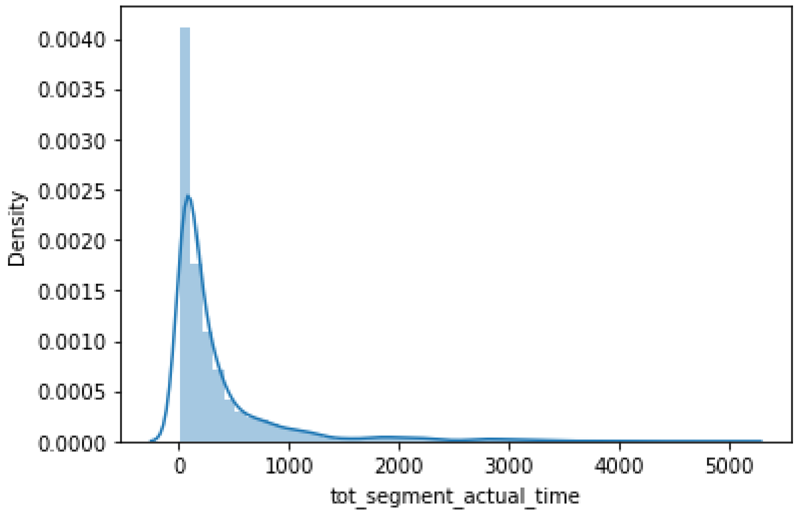
In [131]:

```
sns.distplot(df_final['tot_segment_actual_time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[131]:

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



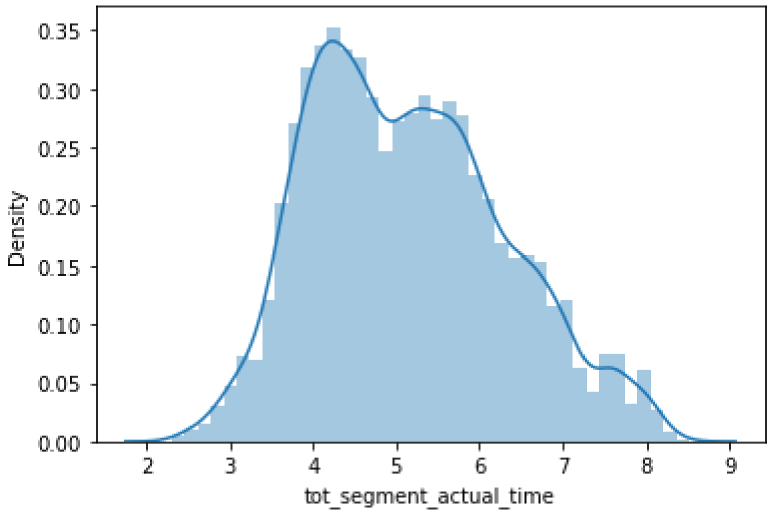
In [132]:

```
sns.distplot(np.log(df_final['tot_segment_actual_time']))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[132]:

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



In [133]:

```
stat.shapiro(np.log(df_final['tot_segment_actual_time']).sample(4999))
```

Out[133]:

```
ShapiroResult(statistic=0.9801135063171387, pvalue=6.569398193004184e-26)
```

In [134]:

```
stat.levene(np.log(df_final['tot_segment_actual_time']),np.log(df_final['tot_segment_osrm_t
```

Out[134]:

```
LeveneResult(statistic=10.09894256639767, pvalue=0.0014856645070554489)
```

```
Assumptions:

As we have just sample data of a single month we can assume that the population data forms
the normal distribution
From levenes test we can cofirm that they dont have equal variances as p value < 0.05

p-value: 0.05
```

```
Ho : Both tot_segment_osrm_time and tot_segment_actual_time are similar
Ha : tot_segment_osrm_time and tot_segment_actual_time are different
```

In [135]:

```
stat.ttest_ind(np.log(df_final['tot_segment_osrm_time']).sample(50),np.log(df_final['tot_se
```

Out[135]:

```
Ttest_indResult(statistic=-3.4093854986360417, pvalue=0.0009489499843708244)
```

```
p value < 0.05

We reject the null hypothesis and the times are different
```

In [ ]:

Normalize/ Standardize

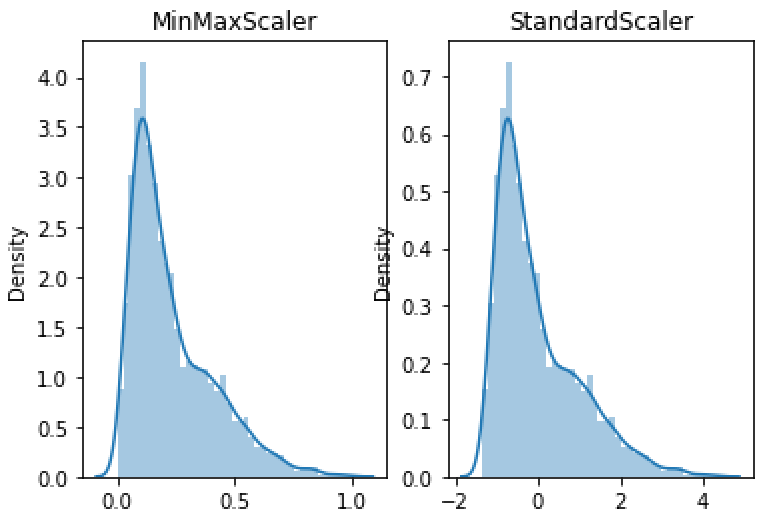
In [149]:

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
std_scale = StandardScaler()
MinMaxScaler = MinMaxScaler()
plt.subplot(121)
plt.title('MinMaxScaler')
sns.distplot(MinMaxScaler.fit_transform(df_final[["actual_time"]]))
plt.subplot(122)
plt.title('StandardScaler')
sns.distplot(std_scale.fit_transform(df_final[["actual_time"]]))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[149]:

```
<AxesSubplot:title={'center':'StandardScaler'}, ylabel='Density'>
```



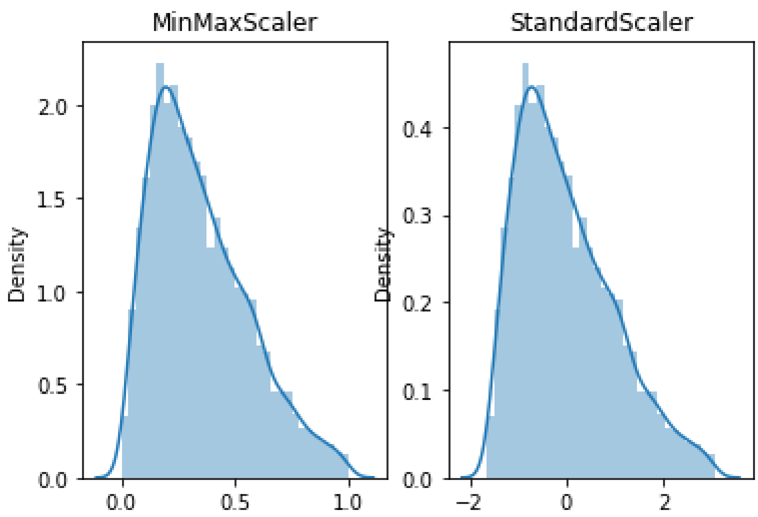
In [150]:

```
plt.subplot(121)
plt.title('MinMaxScaler')
sns.distplot(MinMaxScaler.fit_transform(df_final[["start_scan_to_end_scan"]]))
plt.subplot(122)
plt.title('StandardScaler')
sns.distplot(std_scale.fit_transform(df_final[["start_scan_to_end_scan"]]))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[150]:

```
<AxesSubplot:title={'center':'StandardScaler'}, ylabel='Density'>
```



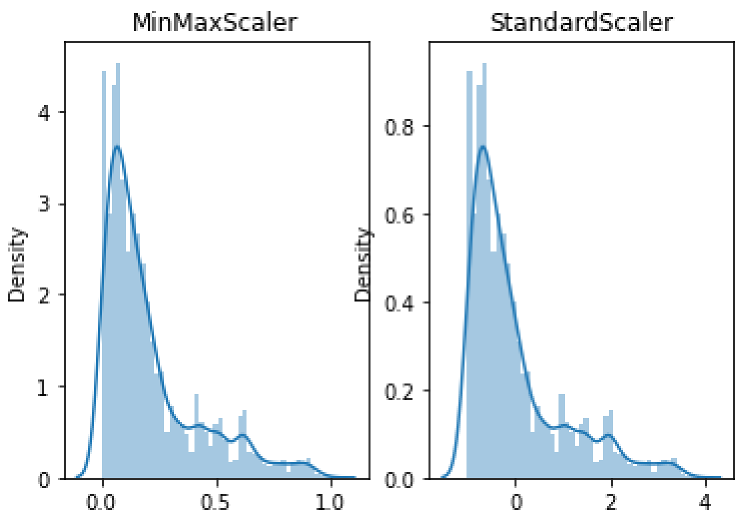
In [151]:

```
plt.subplot(121)
plt.title('MinMaxScaler')
sns.distplot(MinMaxScaler.fit_transform(df_final[["actual_distance_to_destination"]]))
plt.subplot(122)
plt.title('StandardScaler')
sns.distplot(std_scale.fit_transform(df_final[["actual_distance_to_destination"]]))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[151]:

```
<AxesSubplot:title={'center':'StandardScaler'}, ylabel='Density'>
```



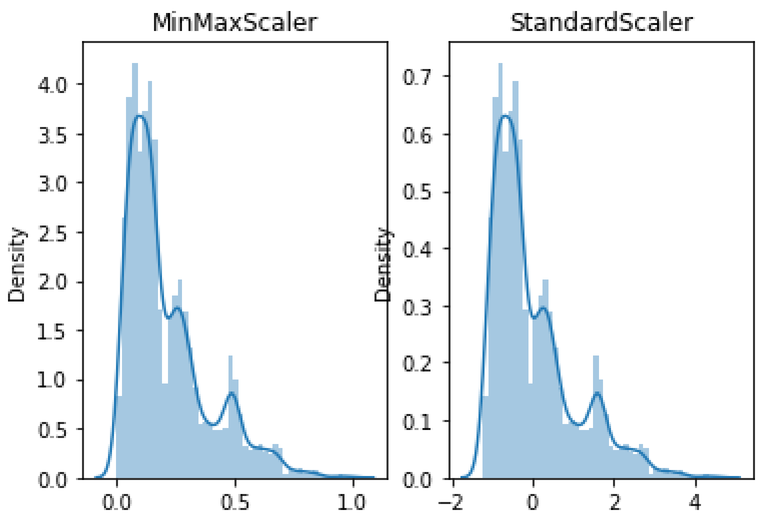
In [152]:

```
plt.subplot(121)
plt.title('MinMaxScaler')
sns.distplot(MinMaxScaler.fit_transform(df_final[["osrm_time"]]))
plt.subplot(122)
plt.title('StandardScaler')
sns.distplot(std_scale.fit_transform(df_final[["osrm_time"]]))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[152]:

```
<AxesSubplot:title={'center':'StandardScaler'}, ylabel='Density'>
```



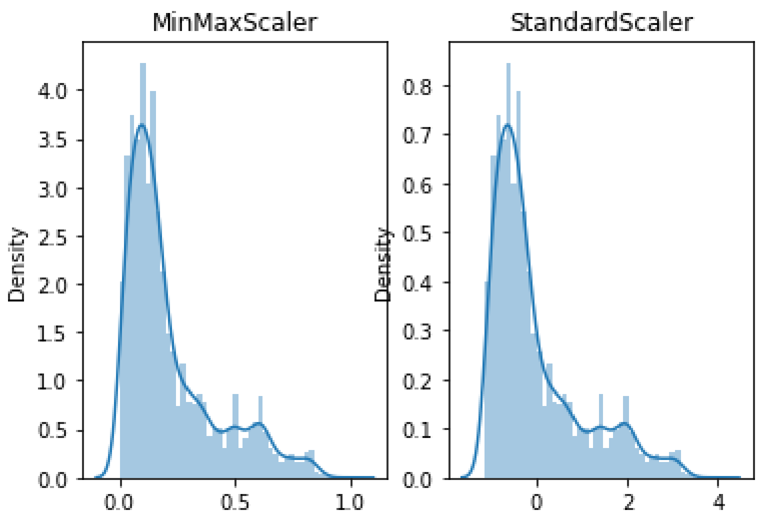
In [153]:

```
plt.subplot(121)
plt.title('MinMaxScaler')
sns.distplot(MinMaxScaler.fit_transform(df_final[["osrm_distance"]]))
plt.subplot(122)
plt.title('StandardScaler')
sns.distplot(std_scale.fit_transform(df_final[["osrm_distance"]]))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[153]:

```
<AxesSubplot:title={'center':'StandardScaler'}, ylabel='Density'>
```



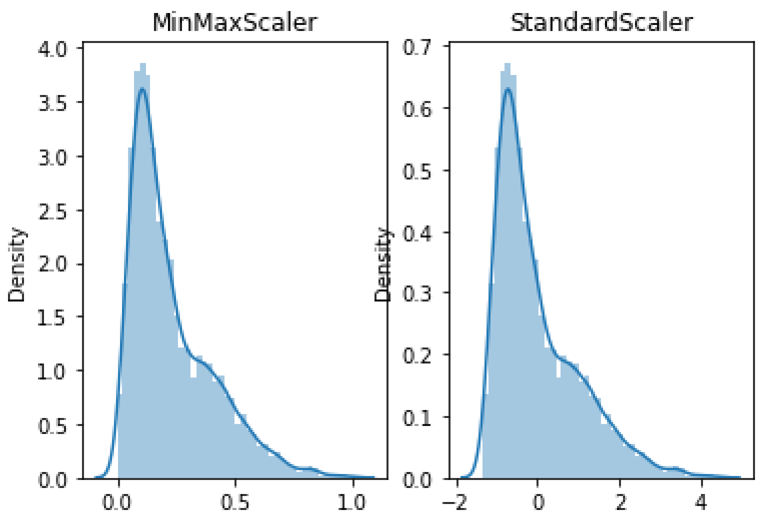
In [154]:

```
plt.subplot(121)
plt.title('MinMaxScaler')
sns.distplot(MinMaxScaler.fit_transform(df_final[["tot_segment_actual_time"]]))
plt.subplot(122)
plt.title('StandardScaler')
sns.distplot(std_scale.fit_transform(df_final[["tot_segment_actual_time"]]))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[154]:

```
<AxesSubplot:title={'center':'StandardScaler'}, ylabel='Density'>
```



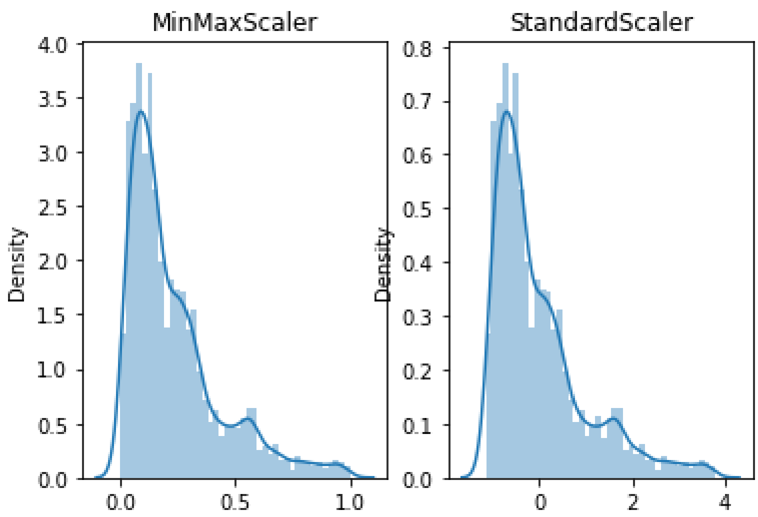
In [155]:

```
plt.subplot(121)
plt.title('MinMaxScaler')
sns.distplot(MinMaxScaler.fit_transform(df_final[["tot_segment_osrm_time"]]))
plt.subplot(122)
plt.title('StandardScaler')
sns.distplot(std_scale.fit_transform(df_final[["tot_segment_osrm_time"]]))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[155]:

<AxesSubplot:title={'center':'StandardScaler'}, ylabel='Density'>



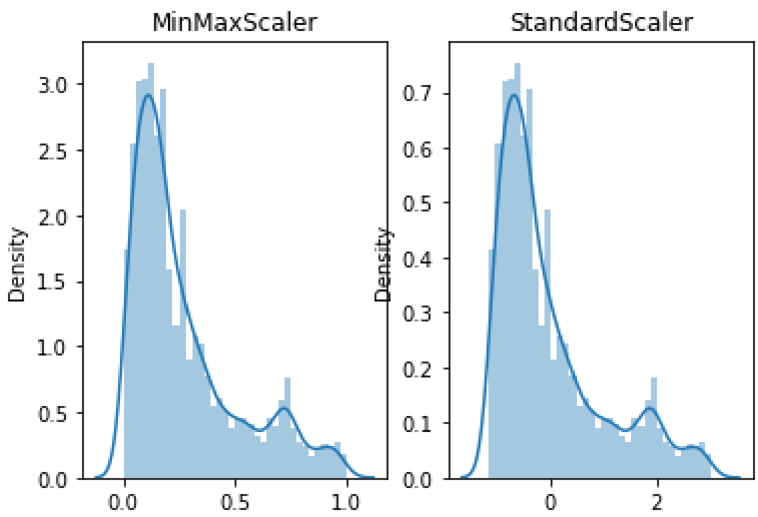
In [156]:

```
plt.subplot(121)
plt.title('MinMaxScaler')
sns.distplot(MinMaxScaler.fit_transform(df_final[["tot_segment_osrm_distance"]]))
plt.subplot(122)
plt.title('StandardScaler')
sns.distplot(std_scale.fit_transform(df_final[["tot_segment_osrm_distance"]]))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[156]:

```
<AxesSubplot:title={'center':'StandardScaler'}, ylabel='Density'>
```



In [ ]:

```

```

Insights

```
1)  Most of the orders are happening in the hours 22,20 and 23, we can infer as night
times.
2)  When compared between route types cartings are more than FTL
3)  FTL is used for large distance orders
4)  Bangalore,Gurgaon and Bhiwandi are top three places from where most of the trips
starts from.
5)  Most of the orders are delivered in 8 hours to the destination
6)  Time is directly proportional to distance but in somecases even if the distance is
small it took more time.
7)  OSRM time and OSRM distance are linearly proportional
8)  Actual time and segment times vary if the distance increases
```

```
9)  OSRM time and Segment OSRM times are highly correlated
10) Chandigarh_Mehmdpur_H (Punjab) to Chandigarh_Mehmdpur_H (Punjab) and from
Bengaluru_KGAirprt_HB (Karnataka) to Bangalore_Nelmngla_H (Karnataka) has more number of
orders and large delivery time
11) Uttar Pradesh to Rajasthan and Dadra and Nagar Haveli to Gujarat are the fastest
delivery trips
12) Gurgaon_Bilaspur_HB (Haryana),Bangalore_Nelmngla_H (Karnataka),Bhiwandi_Mankoli_HB
(Maharashtra) are the top 3 busiest centers
13) Most of the orders are delivered to Gurgaon_Bilaspur_HB (Haryana),Bangalore_Nelmngla_H
(Karnataka),Bhiwandi_Mankoli_HB (Maharashtra)
14) Carting orders are deliverd much faster to the destination compared to FTL
15) FTL covers more distance compared than Carting
16) Trips are created in the same number everyday
17) From hypothesis testing we are clear that we cannot rely on osrm times
```

Recommendation

```
-- We can establish more warehouses in other popuplar states like delhi and hyderabad too
as most of the orders are taking place.
-- If we add above source warehouses it will be helpful during the sale seasons when there
are many orders coming up so that we can reduce the delivery time as our main focus is on
delivering the products in less time.
```