In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
```

In [2]:
```
delhivery_df = pd.read_csv('delhivery_data.txt',sep=',')
```

In [3]:
```
delhivery_df.head()
```

Out[3]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_cent
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA

5 rows × 24 columns

Defining Problem Statement and Analysing basic metrics

We have to create new features from the existing once, finding is their any similarity between 2 different features using hypothesis testing, applying standerdizing technique like StandardScaler, outlyer detuction, converting categorical variables to category using one hot encoding.

In [4]:
```
delhivery_df.shape
```

Out[4]: (144867, 24)

their are total 24 features and 144867 rows in the given data set.

In [5]:
```
delhivery_df.info()
```

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

we can observe that the data is a mixed of both object data types and float and int data types mostly

1 bool type 10 float type 1 int type 12 object type

In [6]:
```
delhivery_df.describe()
```

Out[6]:

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	os
count	144867.000000	144867.000000	144867.000000	144867.000000	14486
mean	961.262986	232.926567	234.073372	416.927527	21
std	1037.012769	344.755577	344.990009	598.103621	30
min	20.000000	9.000000	9.000045	9.000000	
25%	161.000000	22.000000	23.355874	51.000000	2
50%	449.000000	66.000000	66.126571	132.000000	6
75%	1634.000000	286.000000	286.708875	513.000000	25
max	7898.000000	1927.000000	1927.447705	4532.000000	168

we can observe that all the columns are effected by outlyers as the gap between 75% value and the max value is large in all the columns

Also we can observe that the mean and the std are close in almost all the columns from this we can conclude that their are more outlyers present in the data which is even effecting the std too.

In [7]:
```
delhivery_df.describe(include='object')
```

Out[7]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	sourc
count	144867	144867	144867	144867	144867	
unique	2	14817	1504	2	14817	
top	training	2018-10-01 05:04:55.268931	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	trip-153784927255069118	IND00
freq	104858	101	1812	99660	101	

the trip_creation_time column is having 14817 unique columns the route_type is having 2 unique values trip_uuid is having 14817 unique values so from the above data it is clear that their are no unique columns so we may need to do group by inorder to get the unique trip_UUId

In [8]:
```
delhivery_df.isna().sum()
```

Out[8]:
```
data                                0
trip_creation_time                  0
route_schedule_uuid                 0
route_type                          0
trip_uuid                           0
source_center                       0
source_name                       293
destination_center                  0
destination_name                  261
od_start_time                       0
od_end_time                         0
start_scan_to_end_scan              0
is_cutoff                           0
cutoff_factor                       0
cutoff_timestamp                    0
actual_distance_to_destination      0
actual_time                         0
osrm_time                           0
osrm_distance                       0
factor                              0
segment_actual_time                 0
segment_osrm_time                   0
segment_osrm_distance               0
segment_factor                      0
dtype: int64
```

except the source_name,destination_name all the other coulmns are not having any missing values

In [9]:
```
delhivery_df.head()
```

Out[9]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_cent
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA

5 rows × 24 columns

In [10]:
```
delhivery_df = delhivery_df.drop(['is_cutoff','cutoff_factor','cutoff_timestamp',
```

In [11]:
```
delhivery_df.shape
```

Out[11]:
```
(144867, 19)
```

after removing the unknown fields in the data we are left with 19 coulmns only

In [12]:
```
delhivery_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 [13]:
```
delhivery_df.describe()
```

Out[13]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_
count	144867.000000	144867.000000	144867.000000	144867.000000	14486
mean	961.262986	234.073372	416.927527	213.868272	28
std	1037.012769	344.990009	598.103621	308.011085	42
min	20.000000	9.000045	9.000000	6.000000	
25%	161.000000	23.355874	51.000000	27.000000	2
50%	449.000000	66.126571	132.000000	64.000000	7
75%	1634.000000	286.708875	513.000000	257.000000	34
max	7898.000000	1927.447705	4532.000000	1686.000000	232

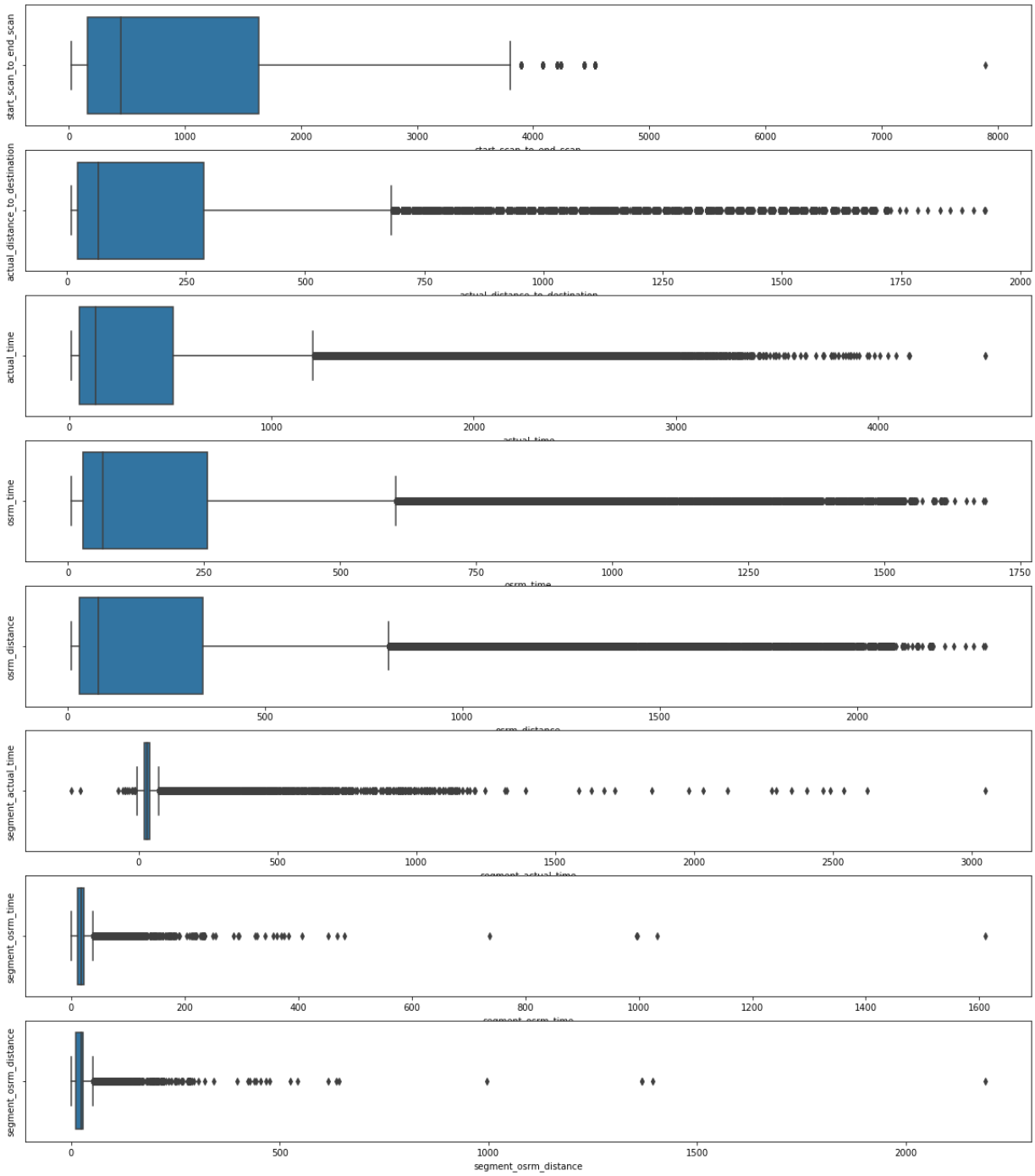
In [14]:
```
delhivery_df.describe().info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8 entries, count to max
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   start_scan_to_end_scan          8 non-null      float64
 1   actual_distance_to_destination  8 non-null      float64
 2   actual_time                     8 non-null      float64
 3   osrm_time                       8 non-null      float64
 4   osrm_distance                   8 non-null      float64
 5   segment_actual_time             8 non-null      float64
 6   segment_osrm_time               8 non-null      float64
 7   segment_osrm_distance           8 non-null      float64
dtypes: float64(8)
memory usage: 576.0+ bytes
```

In [15]:
```
def all_plots(df):
    plt.figure(figsize=(20,35))
    for i in range(len(df.describe().columns)):
        plt.subplot(12,1,i+1)
        sns.boxplot(data=df,x=df.describe().columns[i])
        plt.ylabel(df.describe().columns[i])
    plt.show()
```

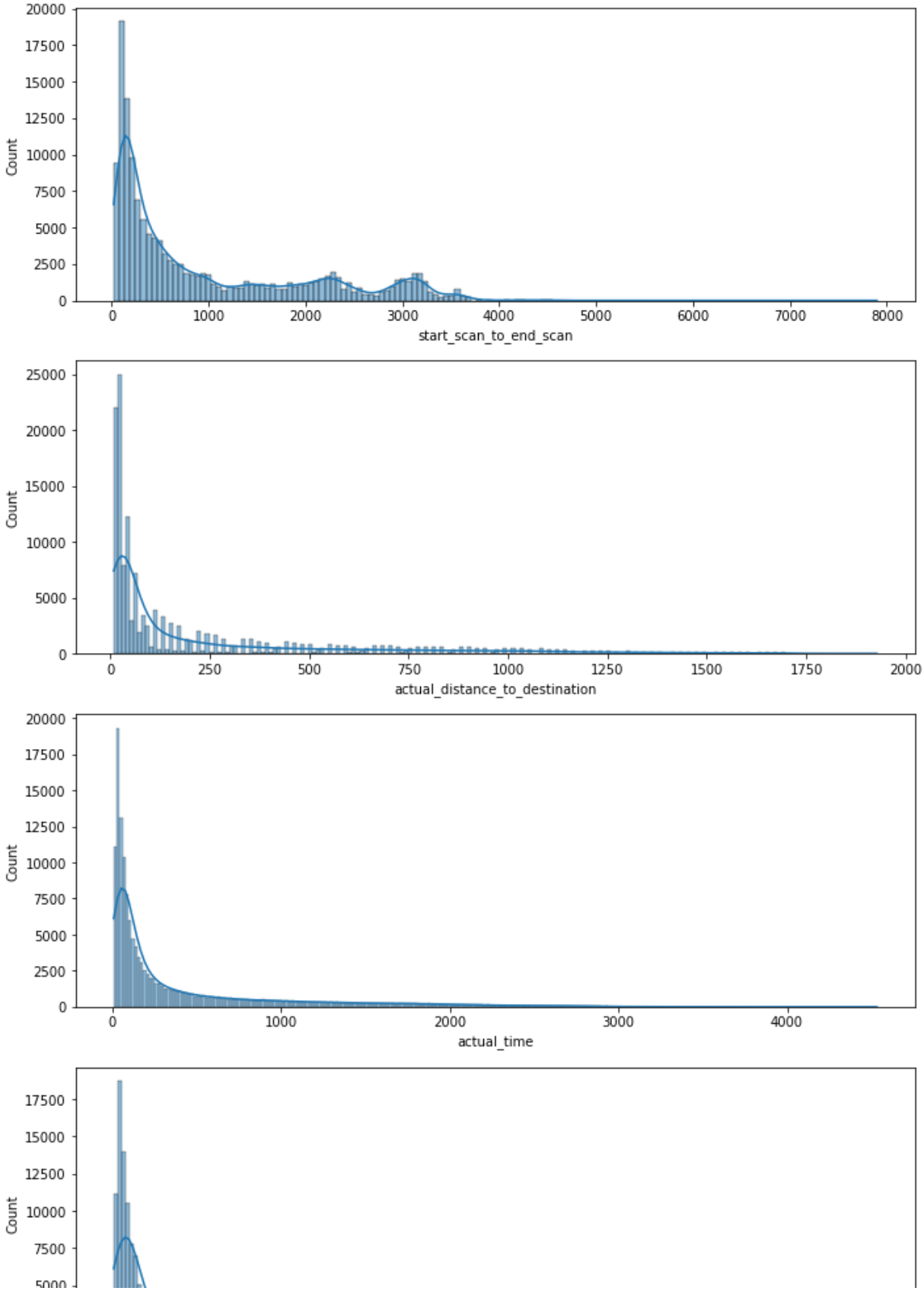
function used to plot the box plot of all categorical varibles

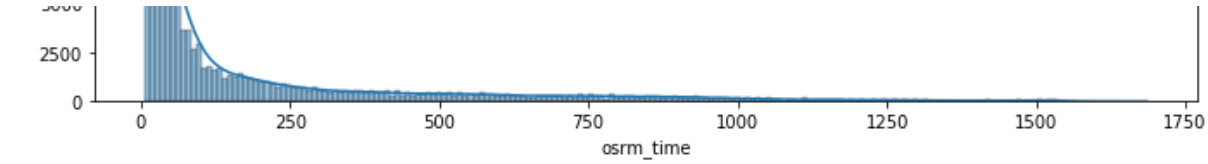
In [16]:
```
all_plots(delhivery_df)
```

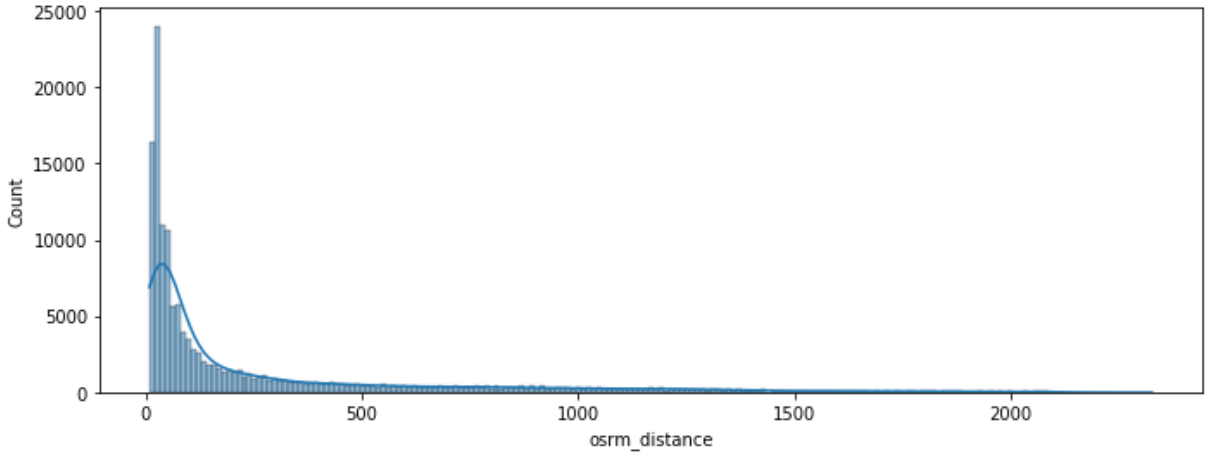


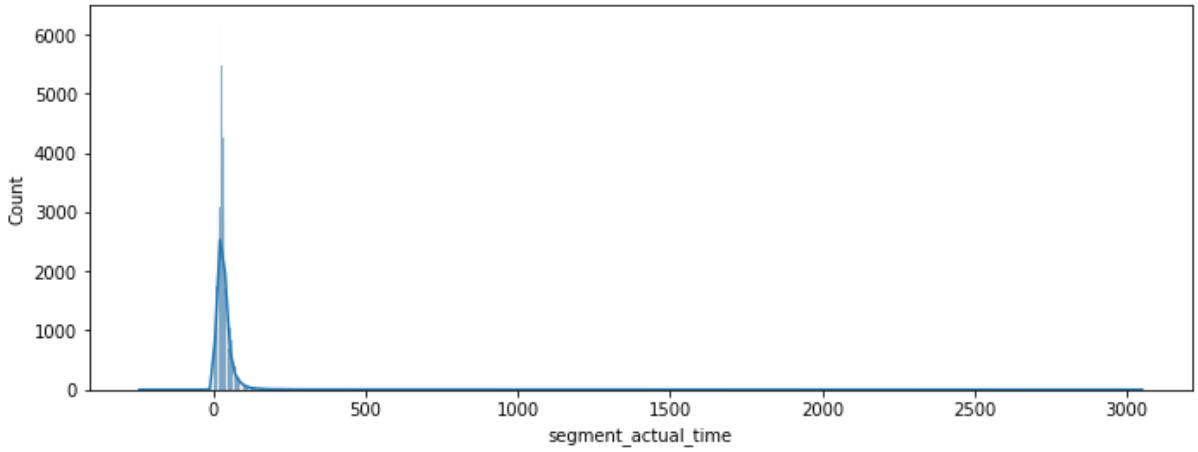
from teh above we can observe that the date is having a lot of outlyers

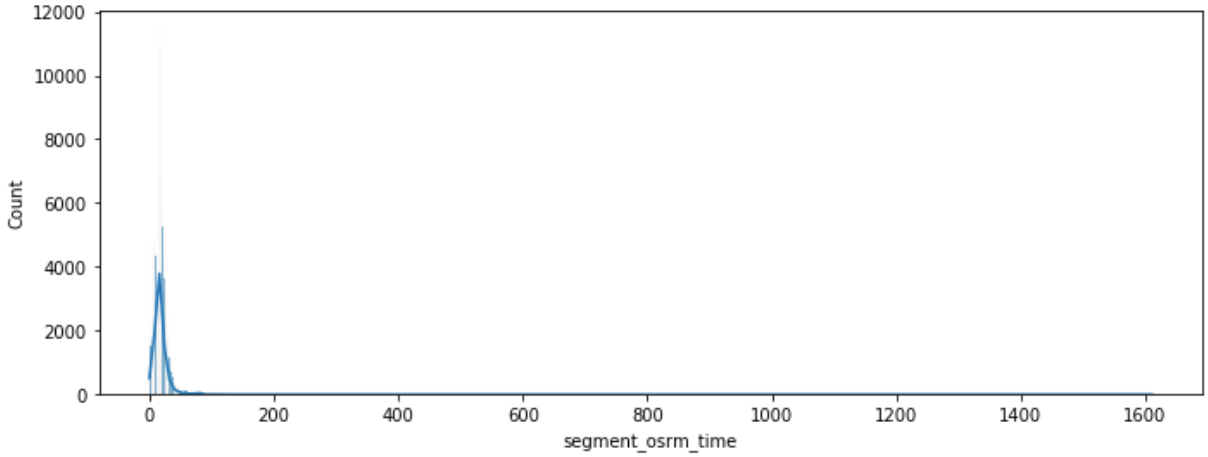
In [17]:
```
fig, axes = plt.subplots(8,1,figsize=(10,30))
for col, ax in zip(delhivery_df.describe().columns,axes.ravel()):
    sns.histplot(data=delhivery_df,x=col,ax=ax,kde=True)
    plt.xlabel(col)
fig.tight_layout()
plt.show()
```

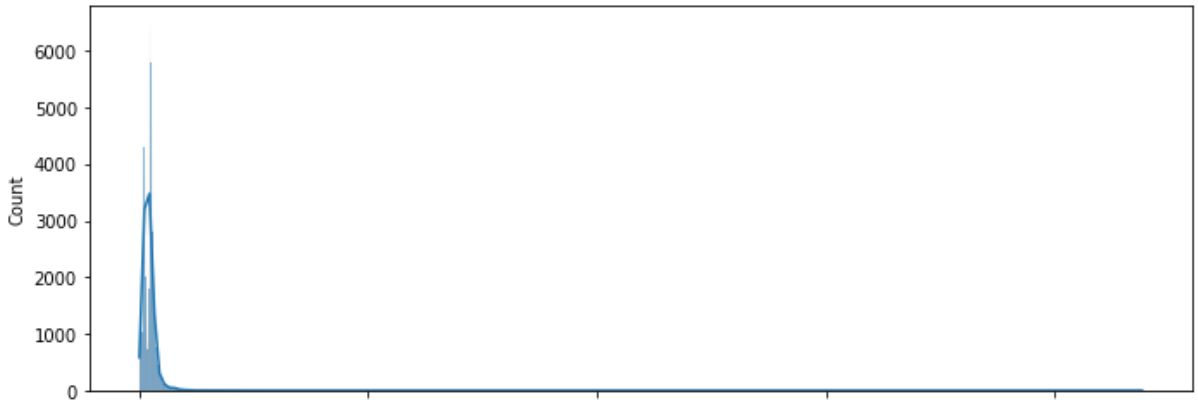


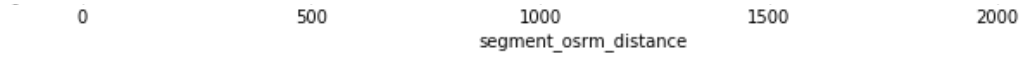

2500
0
0
250
500
750
1000
1250
1500
1750
osrm_time


25000
20000
15000
10000
5000
0
Count
0
500
1000
1500
2000
osrm_distance


6000
5000
4000
3000
2000
1000
0
Count
0
500
1000
1500
2000
2500
3000
segment_actual_time

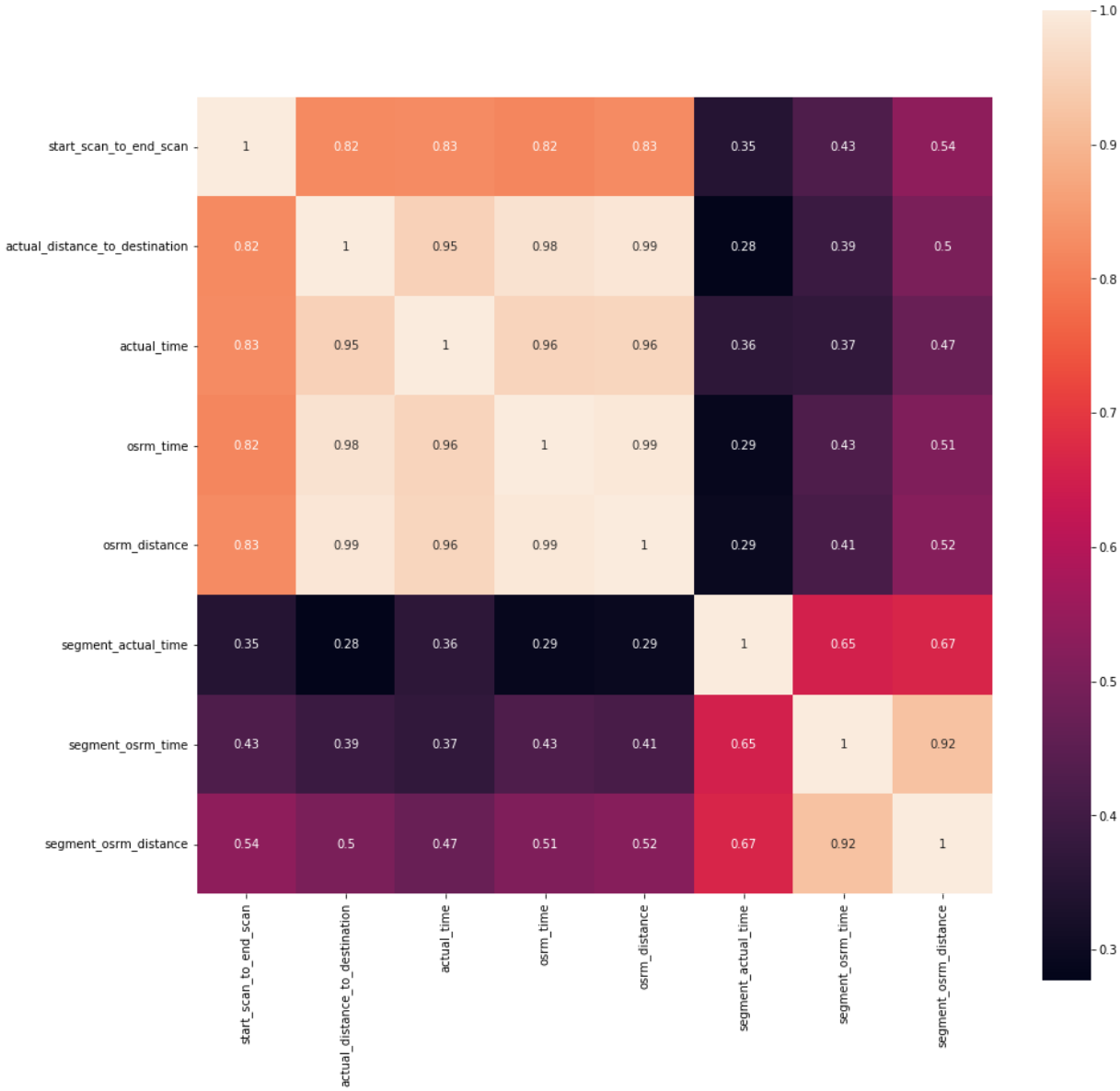

12000
10000
8000
6000
4000
2000
0
Count
0
200
400
600
800
1000
1200
1400
1600
segment_osrm_time


6000
5000
4000
3000
2000
1000
0
Count



from the above kde plot also it is confirmed that the data is consisting of a lot of outlyers

In [18]:
```
plt.figure(figsize=(15, 15))
sns.heatmap(delhivery_df.corr(method='spearman'), square=True,annot=True)
plt.show()
```



we can observe that the osrm_time and osrm_distancev is having highest corelation along with osrm_distance and osrm_time followed by osrm_distance and actual_distance_to_destination

In [19]:
```
delhivery_df.isna().sum()
```

Out[19]:
```
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 [20]:
```
delhivery_df.groupby(by=['trip_uuid','source_center','destination_center']).sum()
```

Out[20]:

			start_scan_to_end_scan	actual_distance_to_
trip_uuid	**source_center**	**destination_center**		
trip-153671041653548748	**IND209304AAA**	**IND000000ACB**	22680.0	
	IND462022AAA	**IND209304AAA**	20979.0	
trip-153671042288605164	**IND561203AAB**	**IND562101AAA**	174.0	
	IND572101AAA	**IND561203AAB**	732.0	
trip-153671043369099517	**IND000000ACB**	**IND160002AAC**	10008.0	
...	**...**	**...**	...	
trip-153861115439069069	**IND628204AAA**	**IND627657AAA**	248.0	
	IND628613AAA	**IND627005AAA**	364.0	
	IND628801AAA	**IND628204AAA**	88.0	
trip-153861118270144424	**IND583119AAA**	**IND583101AAA**	574.0	
	IND583201AAA	**IND583119AAA**	132.0	

26368 rows × 8 columns

In [21]:
```
delhivery_df.groupby(by='trip_uuid').sum()
```

Out[21]:

trip_uuid	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time
trip-153671041653548748	43659.0	8860.812105	15682.0	7787.(
trip-153671042288605164	906.0	240.208306	399.0	210.(
trip-153671043369099517	248631.0	68163.502238	112225.0	65768.(
trip-153671046011330457	200.0	28.529648	82.0	24.(
trip-153671052974046625	1586.0	239.007304	556.0	207.(
...	...	...	...	..
trip-153861095625827784	876.0	141.057373	186.0	148.(
trip-153861104386292051	120.0	25.130640	33.0	19.(
trip-153861106442901555	1263.0	93.743842	549.0	134.(
trip-153861115439069069	1315.0	355.281673	600.0	446.(
trip-153861118270144424	706.0	110.239116	350.0	106.(

14817 rows × 8 columns

grouping on trip_uuid so that we can have the actual time and actual distance etc

In [22]:
```
delhivery_df['trip_uuid'].nunique()
```

Out[22]: 14817

In [23]:
```
def source_splitting(data_df):
    data_df[['temp','source_state']] = data_df['source_name'].str.split('(',expar
    data_df['source_state'] = data_df['source_state'].str.strip(')')
    data_df[['source_city','source_place','source_code']] = data_df['temp'].str.s
    data_df.drop('temp',axis=1,inplace=True)
    return data_df
```

function used to extract the source_city,source_place,source_code,source_state from the source_name column

In [24]:
```
delhivery_df = source_splitting(delhivery_df)
```

In [25]:
```
delhivery_df.head()
```

Out[25]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_cent
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA

5 rows × 23 columns

In [26]:
```
def destination_splitting(data_df):
    data_df[['temp','destination_state']] = data_df['destination_name'].str.split
    data_df['destination_state'] = data_df['destination_state'].str.strip(')')
    data_df[['destination_city','destination_place','destination_code']] = data_d
    data_df.drop('temp',axis=1,inplace=True)
    return data_df
```

function used to extract the destination_city,destination_place,destination_code,destination_state from the destination_name column

In [27]:
```
delhivery_df = destination_splitting(delhivery_df)
```

In [28]:
```
delhivery_df.head()
```

Out[28]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_cent
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA

5 rows × 27 columns

In [29]:
```
delhivery_df['trip_creation_time'] = pd.to_datetime(delhivery_df['trip_creation_t
```

In [30]:
```
delhivery_df['trip_creation_year'] = delhivery_df['trip_creation_time'].dt.year
delhivery_df['trip_creation_month'] = delhivery_df['trip_creation_time'].dt.month
delhivery_df['trip_creation_day'] = delhivery_df['trip_creation_time'].dt.day
```

extracting the year month and day from trip creation time column

In [31]:
```
delhivery_df.head()
```

Out[31]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_cent
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA

5 rows × 30 columns

In [32]:
```
delhivery_df.columns
```

Out[32]:
```
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', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'source_state',
       'source_city', 'source_place', 'source_code', 'destination_state',
       'destination_city', 'destination_place', 'destination_code',
       'trip_creation_year', 'trip_creation_month', 'trip_creation_day'],
      dtype='object')
```

In [33]:
```
delhivery_df['time_taken'] = (pd.to_datetime(delhivery_df['od_end_time'])-pd.to_c
```

calculating the minutes difference between od_end_time and od_start_time

In [34]:
```
# delhivery_df['time_taken'] = delhivery_df.apply(lambda x:pd.to_datetime(x['od_e
```

In [35]:
```
delhivery_df.head()
```

Out[35]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_cent
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA

5 rows × 31 columns

In [36]:
```
delhivery_df.columns
```

Out[36]:
```
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', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'source_state',
       'source_city', 'source_place', 'source_code', 'destination_state',
       'destination_city', 'destination_place', 'destination_code',
       'trip_creation_year', 'trip_creation_month', 'trip_creation_day',
       'time_taken'],
      dtype='object')
```

In [37]:
```
delhivery_df['start_scan_to_end_scan']
```

Out[37]:
```
0          86.0
1          86.0
2          86.0
3          86.0
4          86.0
          ...
144862    427.0
144863    427.0
144864    427.0
144865    427.0
144866    427.0
Name: start_scan_to_end_scan, Length: 144867, dtype: float64
```

In [38]:
```
delhivery_aggre = delhivery_df.groupby(by='trip_uuid').sum()
```

In [39]:
```
delhivery_aggre.head()
```

Out[39]:

trip_uuid	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time
trip-153671041653548748	43659.0	8860.812105	15682.0	7787.0
trip-153671042288605164	906.0	240.208306	399.0	210.0
trip-153671043369099517	248631.0	68163.502238	112225.0	65768.0
trip-153671046011330457	200.0	28.529648	82.0	24.0
trip-153671052974046625	1586.0	239.007304	556.0	207.0

Hypothesis Testing

ks test

its used to check wheather 2 series are following similar distribution or not

Assumptions of ks test

As ks test is non-parameter test(i.e., it did not have any assumptions).

H0(Null hypothesis)

distribution of start_scan_to_end_scan and time_taken are same

Ha(Alternate hypothesis)

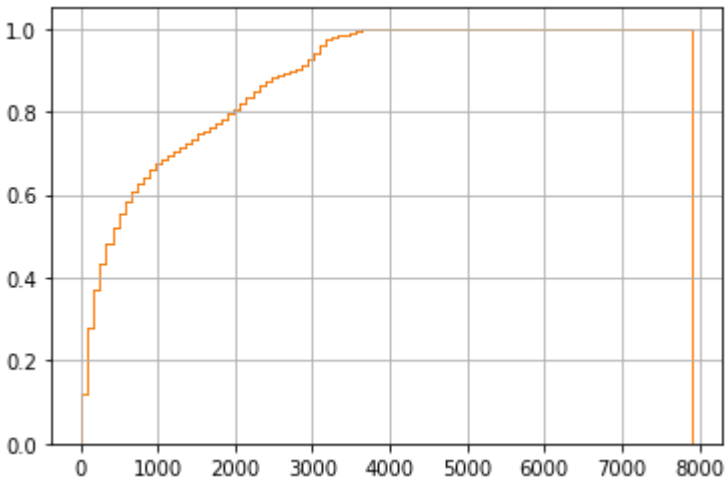
distribution of start_scan_to_end_scan and time_taken are not same

In [40]:
```
stats.ks_2samp(delhivery_df['start_scan_to_end_scan'],delhivery_df['time_taken'])
```

Out[40]:
```
Ks_2sampResult(statistic=0.0031684234504752717, pvalue=0.4611770659109279)
```

here we are using ks test in order to compare the distribution between start_scan_to_end_scan and time_taken

In [41]:
```
plt.grid()
a = plt.hist(delhivery_df['start_scan_to_end_scan'], bins=100, cumulative=True, l
b = plt.hist(delhivery_df['time_taken'], bins=100, cumulative=True, label='CDF',
plt.show()
```



Here we can see that the pvalue=0.4611 which is > 0.05 so both the distribution are same so we fail to reject H0

ks test

its used to check wheather 2 series are following similar distribution or not

Assumptions of ks test

As ks test is non-parameter test(i.e., it did not have any assumptions).

H0(Null hypothesis)

distribution of actual_time and osrm_time are same

Ha(Alternate hypothesis)

distribution of actual_time and osrm_time are not same

In [42]:
```
stats.ks_2samp(delhivery_aggre['actual_time'],delhivery_aggre['osrm_time'])
```

Out[42]:
```
Ks_2sampResult(statistic=0.23115340487278124, pvalue=0.0)
```

In [43]:
```
from statsmodels.stats.weightstats import ztest
```

2-sample z Test

AS the number of samples are large(>30) will be using Z-test insted of T-test and also the sample mean and the varience are known.

Assumptions of Z-test

1) The population mean and standerd deviation are finite. 2) Population standerd deviation are known.

H0(Null hypothesis):-

(U1)Mean of actual time is equal (U2)Mean of osrm time

U1 = U2

Ha(Alternate Hypothesis):-

(U1)Mean of actual time is not equal (U2)Mean of osrm time

U1 != U2

alpha(significance level or type I error):-

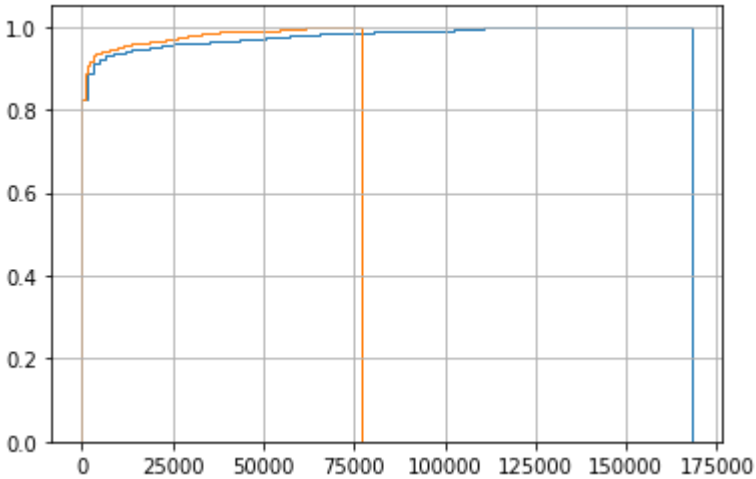
considering 5% significance level

In [44]:
```
ztest(delhivery_aggre['actual_time'],delhivery_aggre['osrm_time'])
```

Out[44]:
```
(14.073444960610715, 5.5308133576654005e-45)
```

performing the z test between actual time and osrm time as pvalue=5.53*10^-64 which is < 0.05 so their means are not same we reject the H0 hypothesis

In [45]:
```
plt.grid()
a = plt.hist(delhivery_aggre['actual_time'], bins=100, cumulative=True, label='CD
b = plt.hist(delhivery_aggre['osrm_time'], bins=100, cumulative=True, label='CDF'
plt.show()
```



2-sample z Test

AS the number of samples are large(>30) will be using Z-test insted of T-test and also the sample mean and the varience are known.

Assumptions of Z-test

1) The population mean and standerd deviation are finite. 2) Population standerd deviation are known.

H0(Null hypothesis):-

(U1)Mean of actual time is equal (U2)segment actual time

U1 = U2

Ha(Alternate Hypothesis):-

(U1)Mean of actual time is not equal (U2)segment actual time

U1 != U2

alpha(significance level or type I error):-

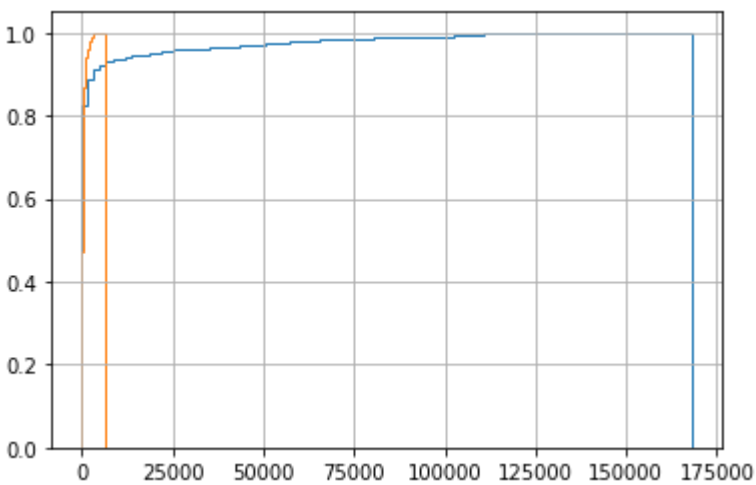
considering 5% significance level

In [46]:
```
ztest(delhivery_aggre['actual_time'],delhivery_aggre['segment_actual_time'])
```

Out[46]:
```
(29.75724632324628, 1.3974700546955178e-194)
```

performing the z test between actual time and segment actual time as pvalue=1.3974700546955178e-194 which is < 0.05 so their means are not same we reject the H0 hypothesis

In [47]:
```
plt.grid()
a = plt.hist(delhivery_aggre['actual_time'], bins=100, cumulative=True, label='CD
b = plt.hist(delhivery_aggre['segment_actual_time'], bins=100, cumulative=True, l
plt.show()
```



2-sample z Test

AS the number of samples are large(>30) will be using Z-test insted of T-test and also the sample mean and the varience are known.

Assumptions of Z-test

1) The population mean and standerd deviation are finite. 2) Population standerd deviation are known.

H0(Null hypothesis):-

(U1)Mean of osrm distance is equal (U2)segment osrm distance

U1 = U2

Ha(Alternate Hypothesis):-

(U1)Mean of osrm distance is not equal (U2)segment osrm distance

U1 != U2

alpha(significance level or type I error):-

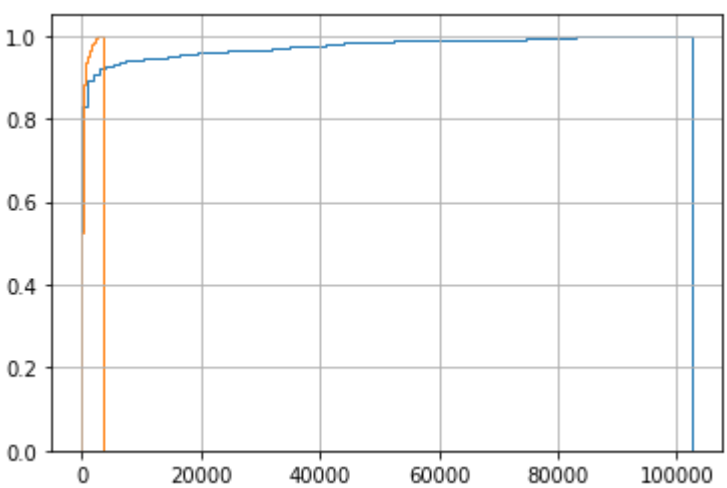
considering 5% significance level

In [48]:
```
ztest(delhivery_aggre['osrm_distance'],delhivery_aggre['segment_osrm_distance'])
```

Out[48]:
```
(28.952997899197353, 2.572707746884534e-184)
```

performing the z test between osrm distance and segment osrm distance as pvalue=2.572707746884534e-184 which is < 0.05 so their means are not same we reject the H0 hypothesis

In [49]:
```
plt.grid()
a = plt.hist(delhivery_aggre['osrm_distance'], bins=100, cumulative=True, label='
b = plt.hist(delhivery_aggre['segment_osrm_distance'], bins=100, cumulative=True,
plt.show()
```



2-sample z Test

AS the number of samples are large(>30) will be using Z-test insted of T-test and also the sample mean and the varience are known.

Assumptions of Z-test

1) The population mean and standerd deviation are finite. 2) Population standerd deviation are known.

H0(Null hypothesis):-

(U1)Mean of osrm time is equal (U2)segment osrm time

U1 = U2

Ha(Alternate Hypothesis):-

(U1)Mean of osrm time is not equal (U2)segment osrm time

U1 != U2

alpha(significance level or type I error):-

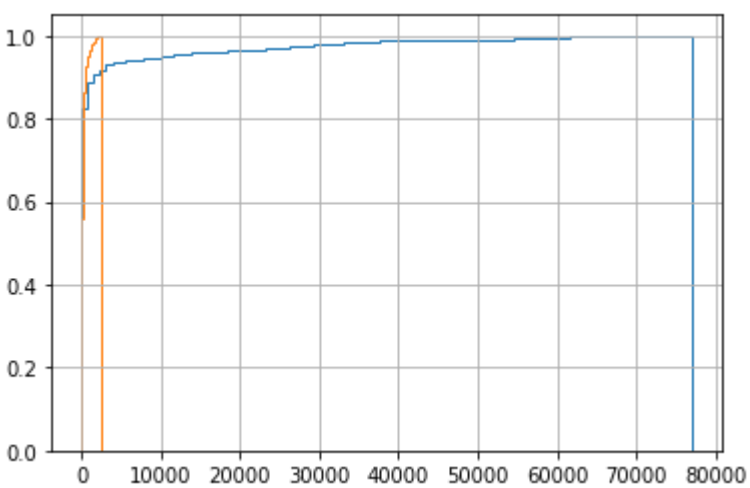
considering 5% significance level

In [50]:
```
ztest(delhivery_aggre['osrm_time'],delhivery_aggre['segment_osrm_time'])
```

Out[50]:
```
(29.19742674380395, 2.090716045464437e-187)
```

performing the z test between osrm time and segment osrm time as pvalue=2.090716045464437e-187 which is < 0.05 so their means are not same we reject the H0 hypothesis

In [51]:
```
plt.grid()
a = plt.hist(delhivery_aggre['osrm_time'], bins=100, cumulative=True, label='CDF'
b = plt.hist(delhivery_aggre['segment_osrm_time'], bins=100, cumulative=True, lab
plt.show()
```



ks test

its used to check wheather 2 series are following similar distribution or not

Assumptions of ks test

As ks test is non-parameter test(i.e., it did not have any assumptions).

H0(Null hypothesis)

distribution of osrm time and segment osrm time are same

Ha(Alternate hypothesis)

distribution of osrm time and segment osrm time are not same

In [52]:
```
stats.ks_2samp(delhivery_aggre['osrm_time'],delhivery_aggre['segment_osrm_time'])
```

Out[52]:
```
Ks_2sampResult(statistic=0.2400620908416008, pvalue=0.0)
```

Here we can see that the pvalue=0.0 which is < 0.05 so both the distribution are same so we reject H0

In [53]:
```
delhivery_df.describe()
```

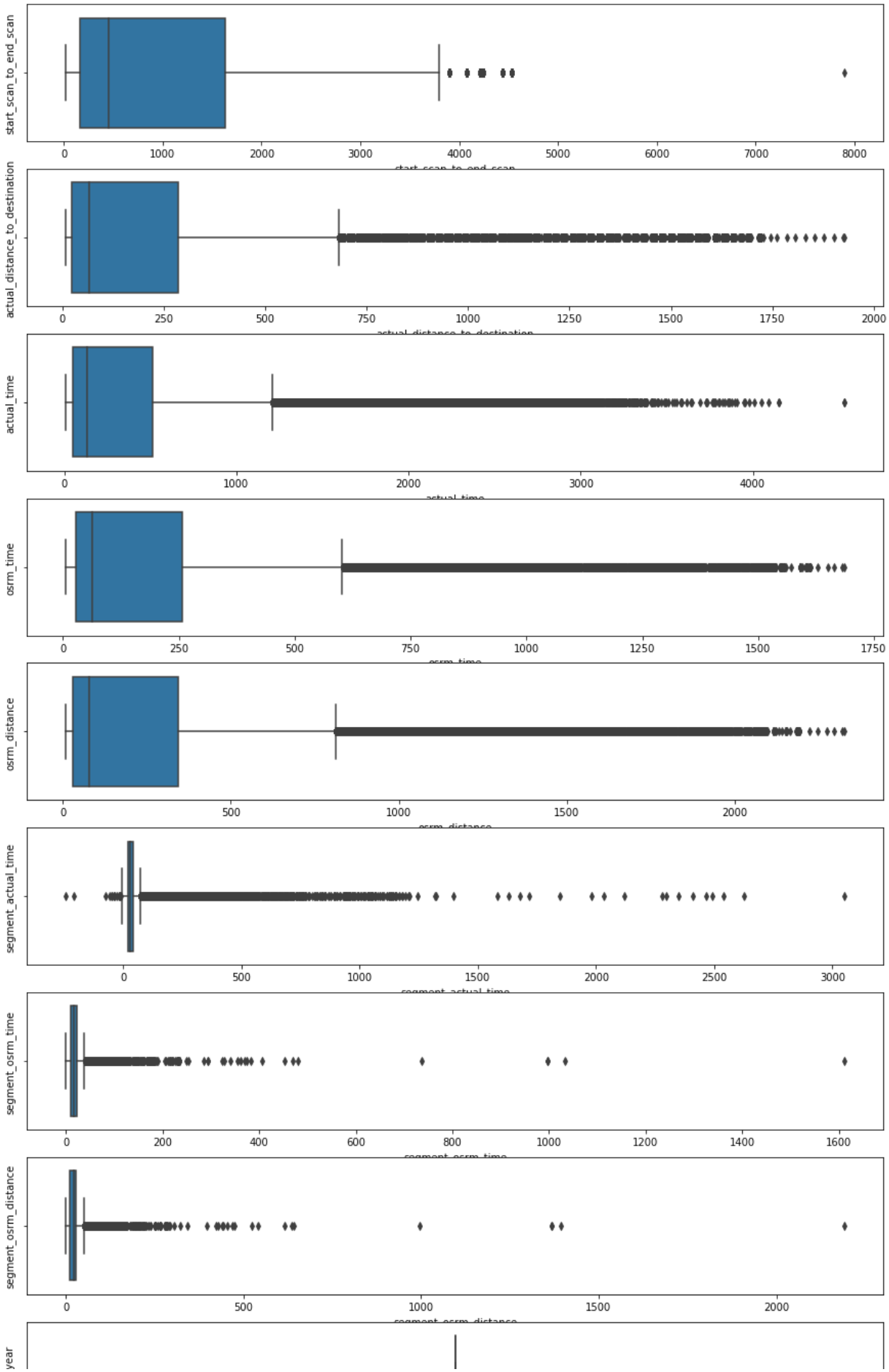
Out[53]:

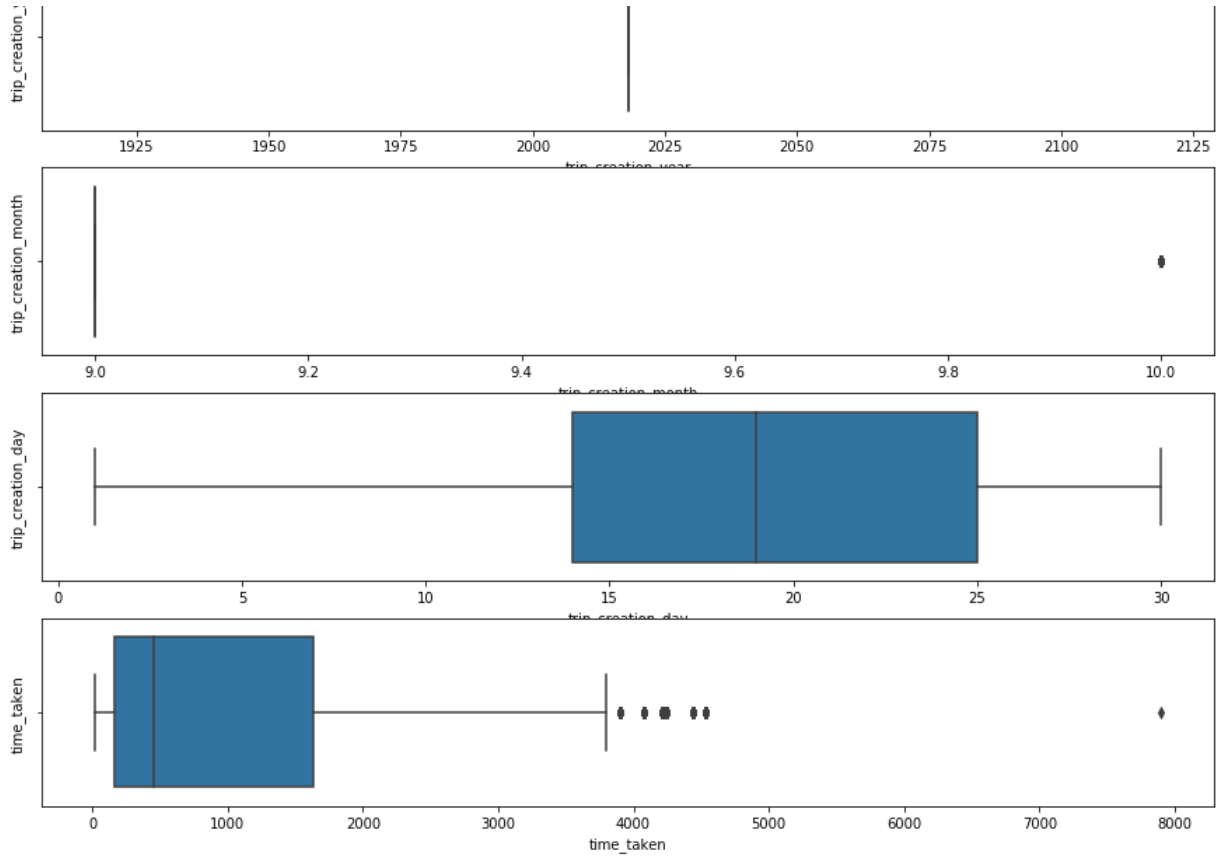
	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_
count	144867.000000	144867.000000	144867.000000	144867.000000	14486
mean	961.262986	234.073372	416.927527	213.868272	28
std	1037.012769	344.990009	598.103621	308.011085	42
min	20.000000	9.000045	9.000000	6.000000	
25%	161.000000	23.355874	51.000000	27.000000	2
50%	449.000000	66.126571	132.000000	64.000000	7
75%	1634.000000	286.708875	513.000000	257.000000	34
max	7898.000000	1927.447705	4532.000000	1686.000000	232

In [54]:
```
def all_plots(df):
    plt.figure(figsize=(15,35))
    for i in range(len(df.describe().columns)):
        plt.subplot(12,1,i+1)
        sns.boxplot(data=df,x=df.describe().columns[i])
        plt.ylabel(df.describe().columns[i])
    plt.show()
```

plotting the box plot of all the numerical variables

In [55]:
```
all_plots(delhivery_df)
```





In [56]:
```
q1, q3 = np.percentile(delhivery_df['start_scan_to_end_scan'],[25,75])
```

In [57]:
```
q3,q1,q3-q1
```

Out[57]:
```
(1634.0, 161.0, 1473.0)
```

In [58]:
```
delhivery_df.columns
```

Out[58]:
```
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', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'source_state',
       'source_city', 'source_place', 'source_code', 'destination_state',
       'destination_city', 'destination_place', 'destination_code',
       'trip_creation_year', 'trip_creation_month', 'trip_creation_day',
       'time_taken'],
      dtype='object')
```

In [59]:
```
delhivery_df['trip_creation_time'].nunique()
```

Out[59]:
```
14817
```

In [60]:
```
delhivery_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 31 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  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
 19  source_state                    144574 non-null  object
 20  source_city                     144574 non-null  object
 21  source_place                    142467 non-null  object
 22  source_code                     129924 non-null  object
 23  destination_state               144606 non-null  object
 24  destination_city                144606 non-null  object
 25  destination_place               142165 non-null  object
 26  destination_code                129038 non-null  object
 27  trip_creation_year              144867 non-null  int64
 28  trip_creation_month             144867 non-null  int64
 29  trip_creation_day               144867 non-null  int64
 30  time_taken                      144867 non-null  float64
dtypes: datetime64[ns](1), float64(9), int64(3), object(18)
memory usage: 34.3+ MB
```

In [61]:
```
for i in delhivery_df.columns:
    print('The unique values of the column:-  '+str(i))
    print(delhivery_df[i].unique())
    print(delhivery_df[i].nunique())
    print('_'*100)
```

```
The unique values of the column:-  data
['training' 'test']
2
____________________________________________________________________________________________________
The unique values of the column:-  trip_creation_time
['2018-09-20T02:35:36.476840000' '2018-09-23T06:42:06.021680000'
 '2018-09-14T15:42:46.437249000' ... '2018-09-22T11:30:41.399439000'
 '2018-09-17T11:35:28.838714000' '2018-09-20T16:24:28.436231000']
14817
____________________________________________________________________________________________________
The unique values of the column:-  route_schedule_uuid
['thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3297ef'
 'thanos::sroute:ff52ef7a-4d0d-4063-9bfe-cc211728881b'
 'thanos::sroute:a16bfa03-3462-4bce-9c82-5784c7d315e6' ...
 'thanos::sroute:72cf9feb-f4e3-4a55-b92a-0b686ee8fabc'
 'thanos::sroute:5e08be79-8a4c-4a91-a514-5350403c0e31'
 'thanos::sroute:a3c30562-87e5-471c-9646-0ed49c150996']
1504
____________________________________________________________________________________________________
The unique values of the column:-  route_type
['Carting' 'FTL']
2
____________________________________________________________________________________________________
The unique values of the column:-  trip_uuid
['trip-153741093647649320' 'trip-153768492602129387'
 'trip-153693976643699843' ... 'trip-153761584139918815'
 'trip-153718412883843340' 'trip-153746066843555182']
14817
____________________________________________________________________________________________________
The unique values of the column:-  source_center
['IND388121AAA' 'IND388620AAB' 'IND421302AAG' ... 'IND361335AAA'
 'IND562132AAC' 'IND639104AAB']
1508
____________________________________________________________________________________________________
The unique values of the column:-  source_name
['Anand_VUNagar_DC (Gujarat)' 'Khambhat_MotvdDPP_D (Gujarat)'
 'Bhiwandi_Mankoli_HB (Maharashtra)' ... 'Dwarka_StnRoad_DC (Gujarat)'
 'Bengaluru_Nelmngla_L (Karnataka)' 'Kulithalai_AnnaNGR_D (Tamil Nadu)']
1498
____________________________________________________________________________________________________
The unique values of the column:-  destination_center
['IND388620AAB' 'IND388320AAA' 'IND411033AAA' ... 'IND600004AAA'
 'IND134203AAA' 'IND400701AAA']
1481
```

```
_____________________________________________________________________________________________
The unique values of the column:-  destination_name
['Khambhat_MotvdDPP_D (Gujarat)' 'Anand_Vaghasi_IP (Gujarat)'
 'Pune_Tathawde_H (Maharashtra)' ... 'Chennai_Mylapore (Tamil Nadu)'
 'Naraingarh_Ward2DPP_D (Haryana)' 'Mumbai_Ghansoli_DC (Maharashtra)']
1468
_____________________________________________________________________________________________
The unique values of the column:-  od_start_time
['2018-09-20 03:21:32.418600' '2018-09-20 04:47:45.236797'
 '2018-09-23 06:42:06.021680' ... '2018-09-22 11:30:41.399439'
 '2018-09-17 11:35:28.838714' '2018-09-20 16:24:28.436231']
26369
_____________________________________________________________________________________________
The unique values of the column:-  od_end_time
['2018-09-20 04:47:45.236797' '2018-09-20 06:36:55.627764'
 '2018-09-23 11:44:28.365845' ... '2018-09-22 21:45:05.128533'
 '2018-09-17 13:32:21.128357' '2018-09-20 23:32:09.618069']
26369
_____________________________________________________________________________________________
The unique values of the column:-  start_scan_to_end_scan
[  86.  109.  302. ... 2476. 1161. 2949.]
1915
_____________________________________________________________________________________________
The unique values of the column:-  actual_distance_to_destination
[10.43566024 18.9368423  27.63727904 ... 66.16359134 73.68066734
 70.03901016]
144515
_____________________________________________________________________________________________
The unique values of the column:-  actual_time
[  14.   24.   40. ... 3169. 3318. 2980.]
3182
_____________________________________________________________________________________________
The unique values of the column:-  osrm_time
[  11.   20.   28. ... 1340. 1439. 1312.]
1531
_____________________________________________________________________________________________
The unique values of the column:-  osrm_distance
[ 11.9653  21.7243  32.5395 ...  97.0933 111.2709  88.7319]
138046
_____________________________________________________________________________________________
The unique values of the column:-  segment_actual_time
[ 1.400e+01  1.000e+01  1.600e+01  2.100e+01  6.000e+00  1.500e+01
  2.800e+01  2.600e+01  3.800e+01  3.700e+01  4.100e+01  2.300e+01
  4.600e+01  3.000e+01  5.000e+01  9.300e+01  6.200e+01  4.900e+01
  2.700e+01  3.500e+01  6.700e+01  2.000e+01  5.100e+01  9.400e+01
  1.900e+01  1.200e+01  1.800e+01  1.100e+01  2.000e+00  1.300e+01
  2.400e+01  5.700e+01  1.000e+00  0.000e+00  2.200e+01  2.500e+01
  4.500e+01  8.000e+00  3.600e+01  4.200e+01  4.400e+01  7.500e+01
```

7.800e+01	2.900e+01	3.400e+01	3.200e+01	6.000e+01	4.300e+01
7.900e+01	4.000e+01	6.900e+01	5.800e+01	5.200e+01	4.800e+01
5.500e+01	5.400e+01	4.700e+01	9.000e+00	8.700e+01	1.700e+01
6.800e+01	5.600e+01	3.300e+01	4.000e+00	5.300e+01	2.000e+02
6.500e+01	3.100e+01	8.800e+01	7.000e+00	8.400e+01	8.200e+01
3.900e+01	1.010e+02	1.900e+02	2.020e+02	9.700e+01	8.600e+01
2.910e+02	1.620e+02	4.310e+02	1.060e+02	1.530e+02	8.900e+01
7.300e+01	3.000e+00	1.360e+02	6.600e+01	6.300e+01	5.000e+00
4.220e+02	7.600e+01	8.300e+01	1.230e+02	7.200e+01	1.320e+02
6.850e+02	1.038e+03	6.100e+01	5.900e+01	1.710e+02	1.410e+02
7.000e+01	7.700e+01	1.250e+02	9.200e+01	7.100e+01	6.400e+01
1.040e+02	1.120e+02	9.000e+01	9.800e+01	3.030e+02	1.240e+02
8.100e+01	1.730e+02	9.100e+01	2.200e+02	1.750e+02	2.920e+02
1.170e+02	4.680e+02	6.940e+02	1.090e+02	1.300e+02	3.710e+02
6.110e+02	-2.600e+01	1.480e+02	1.070e+02	5.040e+02	1.150e+02
8.000e+01	1.790e+02	1.080e+02	9.600e+01	1.000e+02	8.500e+01
4.930e+02	4.440e+02	4.240e+02	7.600e+02	1.030e+02	2.320e+02
1.490e+02	2.050e+02	9.420e+02	1.270e+02	7.400e+01	1.660e+02
9.500e+01	1.440e+02	2.220e+02	1.540e+02	1.210e+02	1.840e+02
3.250e+02	1.020e+02	5.270e+02	1.110e+02	5.390e+02	1.590e+02
5.860e+02	3.460e+02	1.180e+02	3.190e+02	2.690e+02	2.950e+02
6.580e+02	2.410e+02	2.960e+02	9.900e+01	1.160e+02	1.140e+02
1.520e+02	-2.100e+01	2.130e+02	1.050e+02	1.220e+02	1.670e+02
-5.000e+00	1.780e+02	1.136e+03	1.190e+02	1.820e+02	2.120e+02
2.930e+02	1.870e+02	1.350e+02	9.010e+02	1.600e+02	2.240e+02
2.790e+02	1.280e+02	6.370e+02	1.310e+02	1.340e+02	5.590e+02
1.580e+02	1.200e+02	5.580e+02	3.940e+02	2.280e+02	2.770e+02
2.040e+02	2.297e+03	1.630e+02	1.130e+02	5.700e+02	2.720e+02
1.510e+02	7.080e+02	1.380e+02	2.810e+02	8.330e+02	-1.000e+00
5.020e+02	1.100e+02	1.570e+02	1.650e+02	2.080e+02	1.910e+02
3.480e+02	1.500e+02	1.430e+02	2.430e+02	2.330e+02	1.470e+02
3.550e+02	1.370e+02	6.590e+02	2.620e+02	2.440e+02	6.200e+02
1.810e+02	1.560e+02	2.700e+02	1.420e+02	6.270e+02	2.480e+02
2.510e+02	1.770e+02	1.860e+02	1.390e+02	2.880e+02	2.530e+02
2.230e+02	1.400e+02	1.330e+02	2.150e+02	3.470e+02	3.560e+02
2.670e+02	1.720e+02	1.290e+02	6.800e+02	3.450e+02	1.450e+02
4.760e+02	3.300e+02	1.880e+02	3.950e+02	3.850e+02	7.190e+02
1.039e+03	5.510e+02	4.590e+02	4.890e+02	2.100e+02	4.740e+02
1.700e+02	9.900e+02	1.550e+02	2.170e+02	3.230e+02	4.850e+02
3.180e+02	1.690e+02	4.190e+02	6.360e+02	1.850e+02	1.260e+02
3.020e+02	6.350e+02	2.030e+02	1.980e+02	3.600e+02	2.090e+02
2.290e+02	7.430e+02	1.117e+03	3.790e+02	3.090e+02	2.500e+02
1.460e+02	2.780e+02	3.140e+02	1.760e+02	2.340e+02	6.300e+02
3.930e+02	1.890e+02	4.160e+02	4.610e+02	2.630e+02	2.380e+02
1.140e+03	4.210e+02	2.260e+02	2.760e+02	1.610e+02	2.060e+02
1.153e+03	4.270e+02	3.390e+02	8.410e+02	5.220e+02	2.140e+02
1.830e+02	4.490e+02	6.120e+02	1.017e+03	1.640e+02	2.160e+02
3.520e+02	4.140e+02	4.470e+02	6.710e+02	2.210e+02	3.800e+02
2.940e+02	2.070e+02	2.370e+02	3.900e+02	3.750e+02	2.010e+02
1.800e+02	1.847e+03	3.880e+02	9.430e+02	5.160e+02	9.930e+02
2.640e+02	5.150e+02	9.470e+02	6.700e+02	1.680e+02	4.280e+02
4.040e+02	3.980e+02	2.270e+02	2.890e+02	3.990e+02	7.320e+02
5.560e+02	5.980e+02	1.050e+03	2.350e+02	7.660e+02	4.920e+02
6.460e+02	3.120e+02	6.950e+02	2.580e+02	3.590e+02	2.360e+02
3.540e+02	6.010e+02	6.400e+02	6.410e+02	5.070e+02	3.720e+02
2.750e+02	5.180e+02	4.400e+02	2.390e+02	5.100e+02	1.716e+03
7.410e+02	9.370e+02	3.860e+02	2.180e+02	2.650e+02	2.820e+02

2.990e+02	1.990e+02	6.890e+02	6.870e+02	3.220e+02	4.460e+02
3.810e+02	4.950e+02	5.930e+02	2.300e+02	9.340e+02	1.143e+03
3.840e+02	9.940e+02	3.160e+02	8.770e+02	4.150e+02	1.086e+03
1.960e+02	8.790e+02	8.960e+02	6.600e+02	5.240e+02	3.360e+02
5.440e+02	3.380e+02	6.620e+02	7.050e+02	3.110e+02	2.680e+02
1.211e+03	5.630e+02	1.981e+03	6.670e+02	3.320e+02	2.310e+02
8.430e+02	8.220e+02	-5.800e+01	5.730e+02	3.000e+02	-2.110e+02
1.740e+02	6.440e+02	3.570e+02	2.190e+02	2.600e+02	1.940e+02
2.110e+02	2.450e+02	2.032e+03	2.860e+02	2.710e+02	4.200e+02
5.430e+02	3.200e+02	5.570e+02	4.320e+02	2.460e+02	5.010e+02
6.480e+02	3.690e+02	5.250e+02	3.920e+02	6.430e+02	7.570e+02
4.780e+02	7.670e+02	2.850e+02	2.250e+02	7.170e+02	1.970e+02
9.580e+02	5.890e+02	4.620e+02	4.050e+02	1.036e+03	6.560e+02
2.351e+03	4.520e+02	6.530e+02	2.740e+02	1.077e+03	5.210e+02
4.830e+02	3.370e+02	4.500e+02	2.610e+02	9.500e+02	6.830e+02
3.500e+02	2.400e+02	3.910e+02	9.910e+02	2.590e+02	2.281e+03
5.910e+02	1.083e+03	3.280e+02	1.124e+03	1.207e+03	3.640e+02
5.360e+02	4.330e+02	5.870e+02	4.510e+02	7.110e+02	2.550e+02
2.980e+02	-1.200e+01	8.500e+02	9.350e+02	2.660e+02	5.480e+02
4.900e+02	2.464e+03	3.650e+02	5.750e+02	6.420e+02	-3.600e+01
1.008e+03	7.800e+02	2.540e+02	1.950e+02	3.820e+02	6.250e+02
6.510e+02	4.770e+02	5.060e+02	6.180e+02	4.350e+02	6.690e+02
9.920e+02	6.470e+02	3.270e+02	9.180e+02	1.067e+03	4.720e+02
1.125e+03	7.370e+02	4.130e+02	1.046e+03	6.970e+02	6.390e+02
4.290e+02	5.850e+02	3.080e+02	3.780e+02	5.400e+02	4.670e+02
5.170e+02	8.690e+02	5.660e+02	2.520e+02	4.340e+02	3.510e+02
6.230e+02	3.060e+02	7.680e+02	4.110e+02	1.065e+03	-4.200e+01
5.110e+02	3.490e+02	5.340e+02	5.090e+02	4.560e+02	1.055e+03
2.490e+02	4.410e+02	-5.100e+01	9.020e+02	3.010e+02	3.350e+02
3.050e+02	9.530e+02	9.050e+02	6.650e+02	4.090e+02	4.980e+02
5.690e+02	5.670e+02	5.950e+02	5.740e+02	7.990e+02	3.130e+02
6.280e+02	4.600e+02	4.870e+02	1.630e+03	7.700e+02	2.120e+03
2.800e+02	4.750e+02	4.800e+02	6.020e+02	4.660e+02	3.530e+02
6.980e+02	1.131e+03	7.620e+02	7.180e+02	9.950e+02	8.090e+02
2.730e+02	3.340e+02	2.970e+02	4.640e+02	4.170e+02	-2.440e+02
6.000e+02	4.970e+02	8.140e+02	9.320e+02	8.700e+02	3.580e+02
1.097e+03	6.260e+02	3.680e+02	7.550e+02	1.061e+03	1.032e+03
4.230e+02	8.910e+02	3.240e+02	3.420e+02	6.820e+02	1.152e+03
5.880e+02	4.100e+02	9.680e+02	7.710e+02	3.290e+02	7.310e+02
-3.000e+00	9.590e+02	4.020e+02	1.166e+03	8.150e+02	4.960e+02
4.390e+02	9.160e+02	1.182e+03	8.490e+02	2.830e+02	8.940e+02
5.940e+02	5.280e+02	2.470e+02	2.900e+02	7.120e+02	1.041e+03
2.420e+02	1.128e+03	-7.400e+01	7.070e+02	6.290e+02	3.330e+02
5.600e+02	6.770e+02	6.100e+02	9.820e+02	1.920e+02	8.550e+02
3.610e+02	6.210e+02	6.740e+02	1.001e+03	9.610e+02	7.330e+02
6.880e+02	7.200e+02	3.310e+02	2.570e+02	2.408e+03	5.310e+02
1.025e+03	8.400e+02	1.584e+03	1.246e+03	7.470e+02	8.930e+02
6.310e+02	6.780e+02	4.010e+02	5.970e+02	9.270e+02	1.167e+03
1.115e+03	3.260e+02	7.150e+02	1.930e+02	1.395e+03	9.670e+02
5.370e+02	6.040e+02	3.210e+02	3.770e+02	8.510e+02	1.016e+03
1.031e+03	4.550e+02	6.220e+02	3.890e+02	5.830e+02	-4.800e+01
4.180e+02	1.047e+03	6.930e+02	9.700e+02	7.040e+02	7.850e+02
-2.000e+00	2.491e+03	9.460e+02	4.650e+02	2.541e+03	1.122e+03
3.051e+03	9.740e+02	9.780e+02	9.040e+02	-1.600e+01	8.530e+02
4.790e+02	1.148e+03	5.720e+02	4.250e+02	5.530e+02	4.060e+02
-7.000e+00	3.070e+02	1.093e+03	4.630e+02	9.390e+02	5.450e+02
1.325e+03	9.150e+02	5.460e+02	7.530e+02	5.290e+02	4.370e+02

```
  5.200e+02  8.300e+02  1.677e+03  1.020e+03  7.480e+02  4.880e+02
  6.130e+02  9.510e+02  3.740e+02  7.360e+02  9.330e+02  5.790e+02
  2.625e+03  7.520e+02 -1.500e+01  1.192e+03  6.640e+02  1.320e+03
  2.870e+02  3.700e+02  1.104e+03]
747
_____________________________________________________________________________________________
The unique values of the column:-  segment_osrm_time
[1.100e+01 9.000e+00 7.000e+00 1.200e+01 5.000e+00 6.000e+00 1.000e+01
 2.400e+01 2.700e+01 2.600e+01 1.400e+01 1.500e+01 3.000e+01 1.800e+01
 3.800e+01 3.700e+01 2.500e+01 1.700e+01 2.200e+01 3.600e+01 3.200e+01
 1.600e+01 7.000e+01 3.500e+01 4.500e+01 1.300e+01 0.000e+00 8.000e+00
 2.000e+00 1.900e+01 2.300e+01 2.800e+01 2.000e+01 2.100e+01 3.300e+01
 3.400e+01 8.100e+01 3.000e+00 4.400e+01 1.000e+00 4.000e+00 3.900e+01
 4.000e+01 2.900e+01 5.300e+01 3.100e+01 7.500e+01 7.900e+01 9.700e+01
 4.800e+01 4.100e+01 4.300e+01 5.000e+01 5.400e+01 6.800e+01 4.200e+01
 4.600e+01 6.000e+01 5.800e+01 7.600e+01 4.900e+01 1.300e+02 7.800e+01
 6.700e+01 6.400e+01 5.500e+01 5.100e+01 1.420e+02 7.700e+01 7.100e+01
 5.600e+01 6.600e+01 5.900e+01 9.200e+01 6.200e+01 5.200e+01 5.700e+01
 8.000e+01 4.700e+01 7.400e+01 6.500e+01 8.800e+01 7.200e+01 6.900e+01
 7.300e+01 8.400e+01 8.200e+01 8.300e+01 1.540e+02 9.100e+01 6.100e+01
 9.400e+01 1.220e+02 6.300e+01 2.180e+02 9.800e+01 8.700e+01 3.830e+02
 9.000e+01 1.080e+02 9.300e+01 8.600e+01 8.900e+01 1.020e+02 1.600e+02
 2.210e+02 1.050e+02 1.330e+02 1.000e+02 1.060e+02 4.070e+02 8.500e+01
 1.340e+02 4.690e+02 1.800e+02 2.340e+02 1.490e+02 1.010e+02 1.450e+02
 1.140e+02 1.840e+02 2.270e+02 1.740e+02 1.320e+02 9.900e+01 9.600e+01
 1.310e+02 1.110e+02 1.040e+02 1.750e+02 2.300e+02 9.500e+01 1.250e+02
 2.950e+02 1.560e+02 1.160e+02 1.460e+02 1.410e+02 1.030e+02 1.170e+02
 2.310e+02 2.540e+02 2.200e+02 2.330e+02 1.810e+02 1.210e+02 1.270e+02
 3.700e+02 3.750e+02 1.500e+02 1.070e+02 1.610e+02 2.320e+02 1.090e+02
 1.200e+02 1.100e+02 9.970e+02 1.790e+02 1.130e+02 1.660e+02 9.960e+02
 1.240e+02 2.150e+02 1.570e+02 3.620e+02 1.430e+02 1.150e+02 1.280e+02
 1.700e+02 1.440e+02 2.350e+02 1.510e+02 3.560e+02 1.180e+02 1.390e+02
 1.710e+02 1.290e+02 1.190e+02 1.690e+02 1.630e+02 2.040e+02 1.480e+02
 1.830e+02 4.810e+02 3.410e+02 3.280e+02 2.130e+02 1.890e+02 1.910e+02
 1.400e+02 1.470e+02 2.080e+02 2.860e+02 2.160e+02 1.720e+02 1.380e+02
 1.670e+02 2.940e+02 1.230e+02 1.260e+02 2.110e+02 1.611e+03 2.190e+02
 2.490e+02 1.850e+02 1.580e+02 3.240e+02 1.770e+02 4.530e+02 1.520e+02
 1.760e+02 7.370e+02 1.730e+02 1.032e+03]
214
_____________________________________________________________________________________________
The unique values of the column:-  segment_osrm_distance
[11.9653  9.759  10.8152 ... 20.7053 18.8885  8.8088]
113799
_____________________________________________________________________________________________
The unique values of the column:-  source_state
['Gujarat' 'Maharashtra' 'Karnataka' 'Punjab' 'Haryana' 'Uttarakhand'
 'Tamil Nadu' 'Rajasthan' nan 'Telangana' 'Madhya Pradesh' 'Uttar Pradesh'
 'Himachal Pradesh' 'Kerala' 'Andhra Pradesh' 'Bihar' 'Jharkhand' 'Assam'
 'West Bengal' 'Orissa' 'Delhi' 'Jammu & Kashmir' 'Chandigarh'
 'Chhattisgarh' 'Goa' 'Pondicherry' 'Dadra and Nagar Haveli'
 'Arunachal Pradesh' 'Nagaland' 'Meghalaya' 'Tripura' 'Mizoram']
31
_____________________________________________________________________________________________
```

```
The unique values of the column:-  source_city
['Anand' 'Khambhat' 'Bhiwandi' ... 'Nohar' 'Dwarka' 'Kulithalai']
1262
_____________________________________________________________________________________________
The unique values of the column:-  source_place
['VUNagar' 'MotvdDPP' 'Mankoli' ... 'MnbzrDPP' 'StnRoad' 'AnnaNGR']
1155
_____________________________________________________________________________________________
The unique values of the column:-  source_code
['DC ' 'D ' 'HB ' None 'H ' 'L ' 'CP ' 'I ' nan 'C ' 'H_6 ' 'H_1 ' 'IP '
 'D_2 ' 'I_2 ' 'H_2 ' 'D_1 ' 'DPP_2 ' 'DPP_1 ' 'I_7 ' 'Pc ' 'I_1 ' 'I_4 '
 'R_8 ' 'PC ' 'Nagar_DPC ' 'I_21 ' 'D_15 ' 'DPC ' 'DPP_4 ' 'Dc ' 'DPP_3 '
 'H_4 ' 'V ' 'D_4 ' 'D_7 ' 'I_3 ' 'P ' 'M ' 'RP ' 'D_3 ' 'L_8 ' 'I_20 '

 'D_9 ' 'R_11 ' 'D_5 ' 'D_12 ' 'D_8 ' 'D_20 ' 'D_10 ']
49
_____________________________________________________________________________________________
The unique values of the column:-  destination_state
['Gujarat' 'Maharashtra' 'Karnataka' 'Kerala' 'Punjab' 'Uttarakhand'
 'Tamil Nadu' 'Haryana' 'Rajasthan' nan 'Telangana' 'Uttar Pradesh'
 'Delhi' 'Himachal Pradesh' 'Andhra Pradesh' 'Bihar' 'Jharkhand' 'Assam'
 'Orissa' 'West Bengal' 'Jammu & Kashmir' 'Madhya Pradesh' 'Chandigarh'
 'Chhattisgarh' 'Goa' 'Pondicherry' 'Arunachal Pradesh'
 'Dadra and Nagar Haveli' 'Meghalaya' 'Tripura' 'Mizoram' 'Daman & Diu'
 'Nagaland']
32
_____________________________________________________________________________________________
The unique values of the column:-  destination_city
['Khambhat' 'Anand' 'Pune' ... 'Kerala' 'AmaDubi' 'Naraingarh']
1258
_____________________________________________________________________________________________
The unique values of the column:-  destination_place
['MotvdDPP' 'Vaghasi' 'Tathawde' ... 'Mylapore ' 'Ward2DPP' 'Ghansoli']
1131
_____________________________________________________________________________________________
The unique values of the column:-  destination_code
['D ' 'IP ' 'H ' 'PC ' 'HB ' None 'L ' 'DC ' 'P ' 'I ' nan 'H_6 '
 'Nagar_DPC ' 'D_2 ' 'H_1 ' 'I_2 ' 'D_1 ' 'H_2 ' 'DPP_2 ' 'I_1 ' 'D_3 '
 'R_8 ' 'DPC ' 'I_4 ' 'GW ' 'Pc ' 'DPP_1 ' 'I_7 ' 'DPP_4 ' 'I_21 ' 'D_9 '
 'DPP_3 ' 'Dc ' 'H_4 ' 'C ' 'D_5 ' 'D_12 ' 'M ' 'L_8 ' 'D_10 '
 'Layout_PC ' 'D_7 ' 'I_3 ' 'I_20 ' 'INT ' 'RPC ' 'L_23 ' 'D_8 '
 'Gateway ' 'CP ']
49
_____________________________________________________________________________________________
The unique values of the column:-  trip_creation_year
[2018]
1
_____________________________________________________________________________________________
The unique values of the column:-  trip_creation_month
```

```
[ 9 10]
2
_____________________________________________________________________________
_____________________
The unique values of the column:-  trip_creation_day
[20 23 14 13 29 17 12  1 27 28 25 15 18 24  3 19 26 22 21  2 16 30]
22
_____________________________________________________________________________
_____________________
The unique values of the column:-  time_taken
[ 86.21363662 109.17318278 302.37240275 ... 614.3954849  116.87149405
 427.68636397]
26369
_____________________________________________________________________________
_____________________
```

finding the uniques values in all the columns

In [62]:
```
new_columns = []
for i in delhivery_df.columns[1:]:
    if delhivery_df[i].nunique()<10:
        new_columns.append(i)
```

finding the columns which have < 10 unique values

In [63]:
```
new_columns
```

Out[63]:
```
['route_type', 'trip_creation_year', 'trip_creation_month']
```

In [64]:
```
one_hot_encoded_data = pd.get_dummies(delhivery_df,columns=new_columns)
```

In [65]:
```
one_hot_encoded_data
```

Out[65]:

_place	destination_code	trip_creation_day	time_taken	route_type_Carting	route_type_FTL	trip_crea
vdDPP	D	20	86.213637	1	0	
vdDPP	D	20	86.213637	1	0	
vdDPP	D	20	86.213637	1	0	
vdDPP	D	20	86.213637	1	0	
vdDPP	D	20	86.213637	1	0	
...	...	...	...	...	...	
Bilaspur	HB	20	427.686364	1	0	
Bilaspur	HB	20	427.686364	1	0	
Bilaspur	HB	20	427.686364	1	0	
Bilaspur	HB	20	427.686364	1	0	
Bilaspur	HB	20	427.686364	1	0	

In [66]:
```
one_hot_encoded_data.columns
```

Out[66]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'trip_uuid',
       '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', 'source_state',
       'source_city', 'source_place', 'source_code', 'destination_state',
       'destination_city', 'destination_place', 'destination_code',
       'trip_creation_day', 'time_taken', 'route_type_Carting',
       'route_type_FTL', 'trip_creation_year_2018', 'trip_creation_month_9',
       'trip_creation_month_10'],
      dtype='object')
```

In [67]:
```
from sklearn.preprocessing import StandardScaler
```

In [68]:
```
scaler = StandardScaler()
std_data = scaler.fit_transform(delhivery_df[delhivery_df.describe().columns])
std_data = pd.DataFrame(std_data, columns=delhivery_df.describe().columns)
std_data.head()
```

Out[68]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance
0	-0.844026	-0.648246	-0.673677	-0.658642	-0.647814
1	-0.844026	-0.623604	-0.656958	-0.629422	-0.624640
2	-0.844026	-0.598385	-0.630207	-0.603449	-0.598958
3	-0.844026	-0.573802	-0.593424	-0.564489	-0.568034
4	-0.844026	-0.564329	-0.583392	-0.551502	-0.547479

applying the Standardize on the columns

In [72]:
```
delhivery_df['destination_state'].value_counts()
```

Out[72]:
```
Karnataka                 21065
Haryana                   20622
Maharashtra               18196
West Bengal                8499
Telangana                  8205
Tamil Nadu                 8058
Uttar Pradesh              7834
Gujarat                    6714
Rajasthan                  6361
Andhra Pradesh             6265
Delhi                      5754
Punjab                     5105
Madhya Pradesh             4345
Bihar                      4238
Orissa                     3234
Jharkhand                  2552
Kerala                     2230
Assam                      2000
Uttarakhand                 893
Goa                         580
Himachal Pradesh            553
Chandigarh                  389
Chhattisgarh                229
Arunachal Pradesh           211
Jammu & Kashmir             201
Pondicherry                 154
Meghalaya                    37
Dadra and Nagar Haveli       34
Mizoram                      31
Tripura                       9
Nagaland                      7
Daman & Diu                   1
Name: destination_state, dtype: int64
```

In [74]:
```
delhivery_df['destination_city'].value_counts()
```

Out[74]:
```
Gurgaon      15393
Bangalore    11087
Hyderabad     5838
Bhiwandi      5586
Delhi         5429
             ...
Baghpat          1
Vaikom           1
Basta            1
Manthuka         1
Hanskhali        1
Name: destination_city, Length: 1258, dtype: int64
```

In [73]:
```
delhivery_df['destination_place'].value_counts()
```

Out[73]:
```
Bilaspur    15363
Nelmngla    11019
Central      9373
Hub          6939
Mankoli      5586
            ...
Rajpura         1
Barout          1
Sangetha        1
JmnvadRd        1
DivrsnRd        1
Name: destination_place, Length: 1130, dtype: int64
```

In [79]:
```
delhivery_df.groupby(by=['source_place','destination_place']).count().columns
```

Out[79]:
```
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', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'source_state',
       'source_city', 'source_code', 'destination_state', 'destination_city',
       'destination_code', 'trip_creation_year', 'trip_creation_month',
       'trip_creation_day', 'time_taken'],
      dtype='object')
```

In [89]:
```
delhivery_df.groupby(by=['source_city','destination_city']).count().sort_values(b
```

Out[89]:

		data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid
source_city	**destination_city**					
Gurgaon	**Bangalore**	4976	4976	4976	4976	4976
Bangalore	**Gurgaon**	3316	3316	3316	3316	3316
Gurgaon	**Kolkata**	2862	2862	2862	2862	2862
Bengaluru	**Bengaluru**	2062	2062	2062	2062	2062
Bangalore	**Bengaluru**	1741	1741	1741	1741	1741
...	**...**	...	...	...	...	...
Tiruchi	**Tiruchi**	1	1	1	1	1
Hoshangabad	**Itarsi**	1	1	1	1	1
Hajipur	**Dighwara**	1	1	1	1	1
Hailakandi	**Badarpur**	1	1	1	1	1
Kottayam	**Kothanalloor**	1	1	1	1	1

2355 rows × 29 columns

In [100]:
```
delhivery_df[(delhivery_df['source_city']=='Gurgaon') & (delhivery_df['destinatic
```

Out[100]: 1367.2122186495176

In [102]:
```
# delhivery_df.groupby(by=['source_city','destination_city']).mean().sort_values(
delhivery_df[(delhivery_df['source_city']=='Gurgaon') & (delhivery_df['destinatic
```

Out[102]: 859.8276661545312

Business Insights

The most pf the orders are coming from the state karnataka(21065 no of orders are placed from this state)

The most of the orders are coming from Gurgaon city(15393 no of orders are placed from this city)

The most of the orders are coming from the place Bilaspur area(15363 no of orders are placed from this area)

The busiest corridor is Gurgaon Bangalore(4976 orders coming from this location)

The average time taken to reach the order from Gurgaon Bangalore is 1367.2122186495176 minutes

The average distance between the Gurgaon Bangalore corridor is 859.8276661545312

Recommendations

As we can see that most of the popular orders are from the metro cities we have to arrange more means of transportation to this areas which can reduce the time to reach the destination.

As we can se that their is a lot of difference between the calculated time and actual time so inorder to reduce the time between them we have to prefer the shortest routs

As we se that their is a lot of gap between the calculated distance and actual distance the orders are routed through multiple locations so if posible prefering the shortest posible rout will decrease the distance

In [ ]: