In [34]:
```
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
# this allows plots to appear directly in the notebook
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
import random
random.seed = 42
import warnings
warnings.filterwarnings("ignore")

from scipy.stats import ttest_ind
from scipy.stats import f_oneway
from scipy.stats import chi2_contingency
from scipy.stats import ttest_1samp
from statsmodels.stats.multicomp import pairwise_tukeyhsd
import statsmodels.api as sm
```

In [35]:
```
dehivery_data = pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/as
dehivery_data.head(5).T
```

Out[35]:

	0	1	
data	training	training	trainin
trip_creation_time	2018-09-20 02:35:36.476840	2018-09-20 02:35:36.476840	2018-09-2 02:35:36.47684
route_schedule_uuid	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	thanos::sroute:eb7bfc7 b351-4c0e-a95 fa3d5c3
route_type	Carting	Carting	Cartin
trip_uuid	trip-153741093647649320	trip-153741093647649320	trip 15374109364764932
source_center	IND388121AAA	IND388121AAA	IND388121AA
source_name	Anand_VUNagar_DC (Gujarat)	Anand_VUNagar_DC (Gujarat)	Anand_VUNagar_D (Gujara
destination_center	IND388620AAB	IND388620AAB	IND388620AA
destination_name	Khambhat_MotvdDPP_D (Gujarat)	Khambhat_MotvdDPP_D (Gujarat)	Khambhat_MotvdDPP_ (Gujara
od_start_time	2018-09-20 03:21:32.418600	2018-09-20 03:21:32.418600	2018-09-2 03:21:32.41860
od_end_time	2018-09-20 04:47:45.236797	2018-09-20 04:47:45.236797	2018-09-2 04:47:45.23679
start_scan_to_end_scan	86.0	86.0	86
is_cutoff	True	True	Tru
cutoff_factor	9	18	2
cutoff_timestamp	2018-09-20 04:27:55	2018-09-20 04:17:55	2018-09-2 04:01:19.50558
actual_distance_to_destination	10.43566	18.936842	27.63727
actual_time	14.0	24.0	40
osrm_time	11.0	20.0	28
osrm_distance	11.9653	21.7243	32.539
factor	1.272727	1.2	1.42857
segment_actual_time	14.0	10.0	16
segment_osrm_time	11.0	9.0	7
segment_osrm_distance	11.9653	9.759	10.815
segment_factor	1.272727	1.111111	2.28571

Column Profiling:

data - tells whether the data is testing or training data

trip_creation_time – Timestamp of trip creation

route_schedule_uuid – Unique Id for a particular route schedule

route_type – Transportation type

FTL – Full Truck Load: FTL shipments get to the destination sooner, as the truck is making no other pickups or drop-offs along the way

Carting: Handling system consisting of small vehicles (carts)

trip_uuid - Unique ID given to a particular trip (A trip may include different source and destination centers)

source_center - Source ID of trip origin

source_name - Source Name of trip origin

destination_cente – Destination ID

destination_name – Destination Name

od_start_time – Trip start time

od_end_time – Trip end time

start_scan_to_end_scan – Time taken to deliver from source to destination

is_cutoff – Unknown field

cutoff_factor – Unknown field

cutoff_timestamp – Unknown field

actual_distance_to_destination – Distance in Kms between source and destination warehouse

actual_time – Actual time taken to complete the delivery (Cumulative)

osrm_time – An open-source routing engine time calculator which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) and gives the time (Cumulative)

osrm_distance – An open-source routing engine which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) (Cumulative)

factor – Unknown field

segment_actual_time – This is a segment time. Time taken by the subset of the package delivery

segment_osrm_time – This is the OSRM segment time. Time taken by the

subset of the package delivery

segment_osrm_distance – This is the OSRM distance. Distance covered by

subset of the package delivery

segment_factor – Unknown field

In [37]:
```
dehivery_data.shape
```

Out[37]:
```
(144867, 24)
```

In [28]:
```
dehivery_data.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
```

In [ ]:
```

```

Missing Values

In [29]:
```
def missingValue(df):
    #Identifying Missing data.
    total_null = df.isnull().sum().sort_values(ascending = False)
    percent = ((df.isnull().sum()/df.isnull().count())*100).sort_values(ascending =
    print("Total records = ", df.shape[0])

    md = pd.concat([total_null,percent.round(2)],axis=1,keys=['Total Missing','In P
    return md
missingValue(dehivery_data)
```

```
Total records =  144867
```

Out[29]:

	Total Missing	In Percent
source_name	293	0.20
destination_name	261	0.18
data	0	0.00
cutoff_factor	0	0.00
segment_osrm_distance	0	0.00
segment_osrm_time	0	0.00
segment_actual_time	0	0.00
factor	0	0.00
osrm_distance	0	0.00
osrm_time	0	0.00
actual_time	0	0.00
actual_distance_to_destination	0	0.00
cutoff_timestamp	0	0.00
is_cutoff	0	0.00
trip_creation_time	0	0.00
start_scan_to_end_scan	0	0.00
od_end_time	0	0.00
od_start_time	0	0.00
destination_center	0	0.00
source_center	0	0.00
trip_uuid	0	0.00
route_type	0	0.00
route_schedule_uuid	0	0.00
segment_factor	0	0.00

- It is not good to remove missing values here because here we are misssing source and destination name...but we have source and destination IDs ...we can connect with business team and fill those values but for our analysis purpose we will remove it for now because we are doing some prepracessing on this columns

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

Out[30]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	sou
112	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND
113	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND
114	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND
115	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND
116	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND
...	...	...	...	...	...	
144484	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND2
144485	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND2
144486	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND2
144487	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND2
144488	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND2

293 rows × 24 columns

In [31]:
```
dehivery_data.dropna(subset=["source_name","destination_name"],inplace=True)
```

In [32]:
```
dehivery_data[dehivery_data['source_name'].isna()]
#dehivery_data.groupby('destination_center').get_group('IND342902A1B')
```

Out[32]:

data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name

0 rows × 24 columns

In [33]:
```
missingValue(dehivery_data)
```

```
Total records =  144316
```

Out[33]:

	Total Missing	In Percent
data	0	0.0
trip_creation_time	0	0.0
segment_osrm_distance	0	0.0
segment_osrm_time	0	0.0
segment_actual_time	0	0.0
factor	0	0.0
osrm_distance	0	0.0
osrm_time	0	0.0
actual_time	0	0.0
actual_distance_to_destination	0	0.0
cutoff_timestamp	0	0.0
cutoff_factor	0	0.0
is_cutoff	0	0.0
start_scan_to_end_scan	0	0.0
od_end_time	0	0.0
od_start_time	0	0.0
destination_name	0	0.0
destination_center	0	0.0
source_name	0	0.0
source_center	0	0.0
trip_uuid	0	0.0
route_type	0	0.0
route_schedule_uuid	0	0.0
segment_factor	0	0.0

In [6]:
```
dehivery_data.columns
```

Out[6]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'is_cutoff', 'cutoff_factor',
       'cutoff_timestamp', 'actual_distance_to_destination', 'actual_time',
       'osrm_time', 'osrm_distance', 'factor', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'segment_factor'],
      dtype='object')
```

In [7]:
```
#pd.to_datetime(dehivery_data['trip_creation_time'].astype("string"), format=format
```

In [7]:
```

```

Data Preparation and Feature Engineering

In [8]:
```
#splitting destination name
dehivery_data['destination_state']=dehivery_data.destination_name.str.extract('.*\(
dehivery_data['destination_name_Nonstate']=dehivery_data.destination_name.str.repla
dehivery_data['destination_city'],dehivery_data['destination_place'],dehivery_data[

#splitting source name
dehivery_data['source_state']=dehivery_data.source_name.str.extract('.*\((.*)\).*')
dehivery_data['source_name_Nonstate']=dehivery_data.source_name.str.replace(r"\(.*\
dehivery_data['source_city'],dehivery_data['source_place'],dehivery_data['source_cc

dehivery_data.drop(["destination_name_Nonstate",'source_name_Nonstate'],axis=1,inpl
```

In [9]:
```
###set datetime column as index
format = '%Y-%m-%d %H:%M:%S'
dehivery_data['trip_creation_time'] = pd.to_datetime(dehivery_data['trip_creation_t
dehivery_data['od_start_time'] = pd.to_datetime(dehivery_data['od_start_time'].asty
dehivery_data['od_end_time'] = pd.to_datetime(dehivery_data['od_end_time'].astype('

# parse datetime colum & add new time related columns
dehivery_data.set_index('trip_creation_time',inplace=True)
```

In [10]:
```
dehivery_data['date_trip_creation'] = dehivery_data.index.date
dehivery_data['day_trip_creation'] = dehivery_data.index.day
dehivery_data['month_trip_creation'] = dehivery_data.index.month
dehivery_data['year_trip_creation'] = dehivery_data.index.year
dehivery_data['hour_trip_creation'] = dehivery_data.index.hour
dehivery_data['dow_trip_creation'] = dehivery_data.index.dayofweek
day_map = {0:'Monday', 1:'Tuesday', 2:'Wednesday', 3:'Thursday', 4:'Friday', 5:'Sat
dehivery_data.dow_trip_creation=dehivery_data.dow_trip_creation.map(day_map)

dehivery_data['woy_trip_creation'] = dehivery_data.index.weekofyear
dehivery_data=dehivery_data.reset_index()
```

In [11]:
```
#Calculate the time taken between od_start_time and od_end_time and keep it as a fe
dehivery_data['handling_time'] = dehivery_data['od_end_time']-dehivery_data['od_sta

dehivery_data['handling_time_seconds'] = (dehivery_data['od_end_time']-dehivery_dat

dehivery_data['handling_time_minutes'] = (dehivery_data['od_end_time']-dehivery_dat

dehivery_data['handling_time_hours'] = (dehivery_data['od_end_time']-dehivery_data[

dehivery_data['handling_time_days'] = (dehivery_data['od_end_time']-dehivery_data['
dehivery_data['handling_time_full_days'] = (dehivery_data['od_end_time']-dehivery_d
```

In [12]:
```
#Compare the difference between Point a. and start_scan_to_end_scan. Do hypothesis
```

In [47]:
```
```

In [13]:
```
grp=dehivery_data.groupby(['trip_uuid',"source_center","destination_center"])
grp.get_group(('trip-153860680368478736', 'IND209304AAA', 'IND209206AAB')).T
```

Out[13]:

	25602	25603	2560
trip_creation_time	2018-10-03 22:46:43.685043	2018-10-03 22:46:43.685043	2018-10-(22:46:43.68504
data	test	test	te
route_schedule_uuid	thanos::sroute:bf427bc4-6fe3-4868-bc71-5c55d4c...	thanos::sroute:bf427bc4-6fe3-4868-bc71-5c55d4c...	thanos::sroute:bf427bc 6fe3-4868-bc7 5c55d4c
route_type	Carting	Carting	Cartir
trip_uuid	trip-153860680368478736	trip-153860680368478736	trip 1538606803684787
source_center	IND209304AAA	IND209304AAA	IND209304AA
source_name	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	Kanpur_Central_H_ (Uttar Prades
destination_center	IND209206AAB	IND209206AAB	IND209206AA
destination_name	Ghatampur_StatinRD_D (Uttar Pradesh)	Ghatampur_StatinRD_D (Uttar Pradesh)	Ghatampur_StatinRD_ (Uttar Prades
od_start_time	2018-10-03 22:46:43.685043	2018-10-03 22:46:43.685043	2018-10-(22:46:43.68504
od_end_time	2018-10-04 01:59:59.306382	2018-10-04 01:59:59.306382	2018-10-(01:59:59.30638
start_scan_to_end_scan	193.0	193.0	193
is_cutoff	True	True	Tru
cutoff_factor	9	18	2
cutoff_timestamp	2018-10-04 00:59:31	2018-10-04 00:44:33	2018-10-04 00:08:3
actual_distance_to_destination	9.453186	19.305544	27.61695
actual_time	9.0	23.0	59
osrm_time	6.0	14.0	58
osrm_distance	9.6982	19.935	58.398
factor	1.5	1.642857	1.01724
segment_actual_time	9.0	14.0	36
segment_osrm_time	6.0	7.0	72
segment_osrm_distance	9.6982	10.2368	78.333
segment_factor	1.5	2.0	0
destination_state	Uttar Pradesh	Uttar Pradesh	Uttar Prades
destination_city	Ghatampur	Ghatampur	Ghatampu
destination_place	StatinRD	StatinRD	StatinR
destination_code	D	D	
source_state	Uttar Pradesh	Uttar Pradesh	Uttar Prades
source_city	Kanpur	Kanpur	Kanpu
source_place	Central	Central	Centr

	25602	25603	2560
source_code	H_6	H_6	H_
date_trip_creation	2018-10-03	2018-10-03	2018-10-(
day_trip_creation	3	3	
month_trip_creation	10	10	1
year_trip_creation	2018	2018	201
hour_trip_creation	22	22	2
dow_trip_creation	Wednesday	Wednesday	Wednesda
woy_trip_creation	40	40	4
handling_time	0 days 03:13:15.621339	0 days 03:13:15.621339	0 days 03:13:15.62133
handling_time_seconds	11595.621339	11595.621339	11595.62133
handling_time_minutes	193.260356	193.260356	193.26035
handling_time_hours	3.221006	3.221006	3.22100
handling_time_days	0.134209	0.134209	0.13420
handling_time_full_days	0	0	

In [14]:
```
trip_uuid_Group=dehivery_data.groupby(['trip_uuid',"source_center","destination_cer
trip_uuid_Group.columns=['_' .join(str(i) for i in col)for col in trip_uuid_Group.c
#trip_uuid_Group=dehivery_data.groupby(['trip_uuid'])

#trip_uuid_Group.apply(display)
#trip_uuid_Group.get_group(('trip-153860680368478736', 'IND209304AAA', 'IND209206AA
#trip_uuid_Group.get_group(('trip-153860680368478736')).T
trip_uuid_Group.head()
```

Out[14]:

	trip_uuid_	source_center_	destination_center_	actual_time_max	osrm_time_max	segr
0	trip-153671041653548748	IND209304AAA	IND000000ACB	732.0	349.0	
1	trip-153671041653548748	IND462022AAA	IND209304AAA	830.0	394.0	
2	trip-153671042288605164	IND561203AAB	IND562101AAA	47.0	26.0	
3	trip-153671042288605164	IND572101AAA	IND561203AAB	96.0	42.0	
4	trip-153671043369099517	IND000000ACB	IND160002AAC	611.0	212.0	

In [15]:
```
trip_uuid_Group.columns
```

Out[15]:
```
Index(['trip_uuid_', 'source_center_', 'destination_center_',
       'actual_time_max', 'osrm_time_max', 'segment_actual_time_sum',
       'segment_osrm_time_sum', 'actual_distance_to_destination_sum',
       'osrm_distance_max', 'segment_osrm_distance_sum', 'route_type_max'],
      dtype='object')
```

In [ ]:

In [17]:
```
trip_uuid_Group.iloc[:,3:].describe()
```

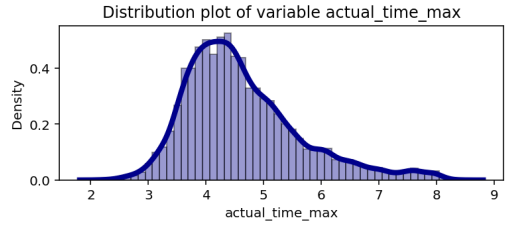
Out[17]:

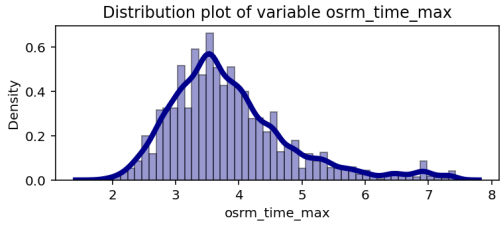
	actual_time_max	osrm_time_max	segment_actual_time_sum	segment_osrm_time_sum	actua
count	26368.000000	26368.000000	26368.000000	26368.000000	
mean	200.690193	91.072853	198.863092	101.681318	
std	384.853640	185.790830	381.283224	215.650948	
min	9.000000	6.000000	9.000000	6.000000	
25%	51.000000	25.000000	50.000000	25.000000	
50%	84.000000	39.000000	83.000000	42.000000	
75%	168.000000	73.000000	166.000000	79.000000	
max	4532.000000	1686.000000	4504.000000	1938.000000	

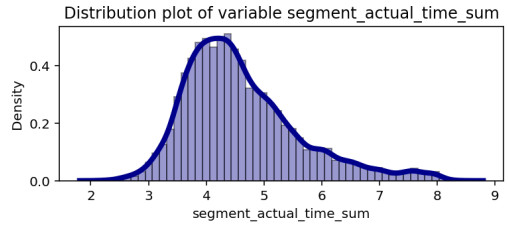
Normality tests

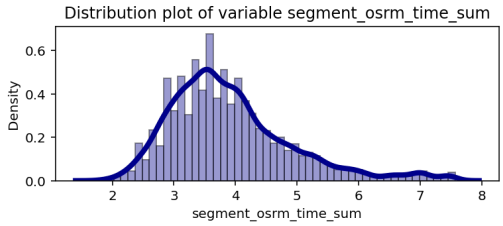
In [1]:
```
def dist_box_violin(data):
 # function plots a combined graph for univariate analysis of continous variable
 #to check spread, central tendency , dispersion and outliers
    Name=data.name.upper()
    fig, axes =plt.subplots(1,3,figsize=(17, 7))
    fig.suptitle("SPREAD OF DATA FOR "+ Name  , fontsize=18, fontweight='bold')
    sns.distplot(data,kde=False,color='Blue',ax=axes[0])
    axes[0].axvline(data.mean(), color='y', linestyle='--',linewidth=2)
    axes[0].axvline(data.median(), color='r', linestyle='dashed', linewidth=2)
    axes[0].axvline(data.mode()[0],color='g',linestyle='solid',linewidth=2)
    axes[0].legend({'Mean':data.mean(),'Median':data.median(),'Mode':data.mode()})
    sns.boxplot(x=data,showmeans=True, orient='h',color="purple",ax=axes[1])
    #just exploring violin plot
    sns.violinplot(data,ax=axes[2],showmeans=True)
    plt.show()
```

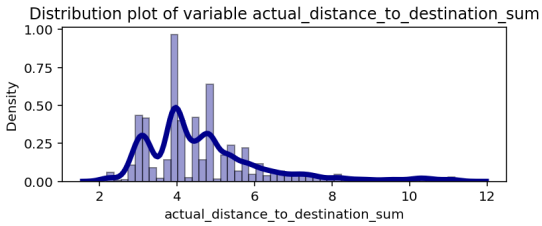
In [18]:
```
fig, axs = plt.subplots(4, 2,figsize=(15, 15))
plt.subplots_adjust(wspace = 0.5,hspace=1)

k=0
j=0
for i in trip_uuid_Group.iloc[:,3:-1].columns:
  #print(j,k)
  sns.distplot(np.log(trip_uuid_Group[i]),ax=axs[j,k], hist=True, kde=True, bins=ir
  #sns.histplot(np.log(trip_uuid_Group[i]),ax=axs[j,k],kde=True,bins=int(50), color
  axs[j,k].set_title(f'Distribution plot of variable {i}')
  if k!=1:
    k=k+1
  else:
    k=0
    j=j+1
```

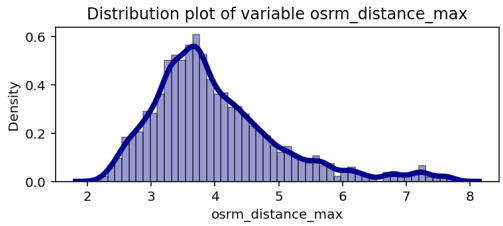


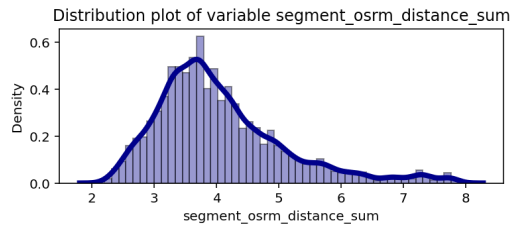


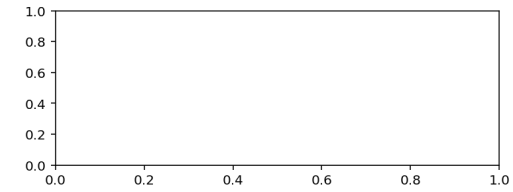












Stastical test to check Normality of data

shapiro

In [19]:
```
from scipy.stats import shapiro
for i in trip_uuid_Group.iloc[:,3:-1].columns:
  actualtime=(np.log(trip_uuid_Group[i])-np.mean(np.log(trip_uuid_Group[i])))/np.st
  tstat,pvalue=shapiro(actualtime)
  print(f"test-statistic {tstat}, pvalue : {pvalue}")
  if  pvalue<0.05:
    print(f"the distribution of {i} does not follows Gaussian distribution so it is
  else:
    print(f"the distribution of {i}  follows  Gaussian distribution  so it is log n
  print('-'*100)
```

```
test-statistic 0.9382826685905457, pvalue : 0.0
the distribution of actual_time_max does not follows Gaussian distribution so it i
s not log normal distribution
--------------------------------------------------------------------------------
------------------
test-statistic 0.9269282221794128, pvalue : 0.0
the distribution of osrm_time_max does not follows Gaussian distribution so it is
not log normal distribution
--------------------------------------------------------------------------------
------------------
test-statistic 0.9383612871170044, pvalue : 0.0
the distribution of segment_actual_time_sum does not follows Gaussian distribution
so it is not log normal distribution
--------------------------------------------------------------------------------
------------------
test-statistic 0.9313375949859619, pvalue : 0.0
the distribution of segment_osrm_time_sum does not follows Gaussian distribution s
o it is not log normal distribution
--------------------------------------------------------------------------------
------------------
test-statistic 0.8663156628608704, pvalue : 0.0
the distribution of actual_distance_to_destination_sum does not follows Gaussian d
istribution so it is not log normal distribution
--------------------------------------------------------------------------------
------------------
test-statistic 0.917636513710022, pvalue : 0.0
the distribution of osrm_distance_max does not follows Gaussian distribution so it
is not log normal distribution
--------------------------------------------------------------------------------
------------------
test-statistic 0.9186626672744751, pvalue : 0.0
the distribution of segment_osrm_distance_sum does not follows Gaussian distributi
on so it is not log normal distribution
--------------------------------------------------------------------------------
------------------
```

KS-test

In [20]:

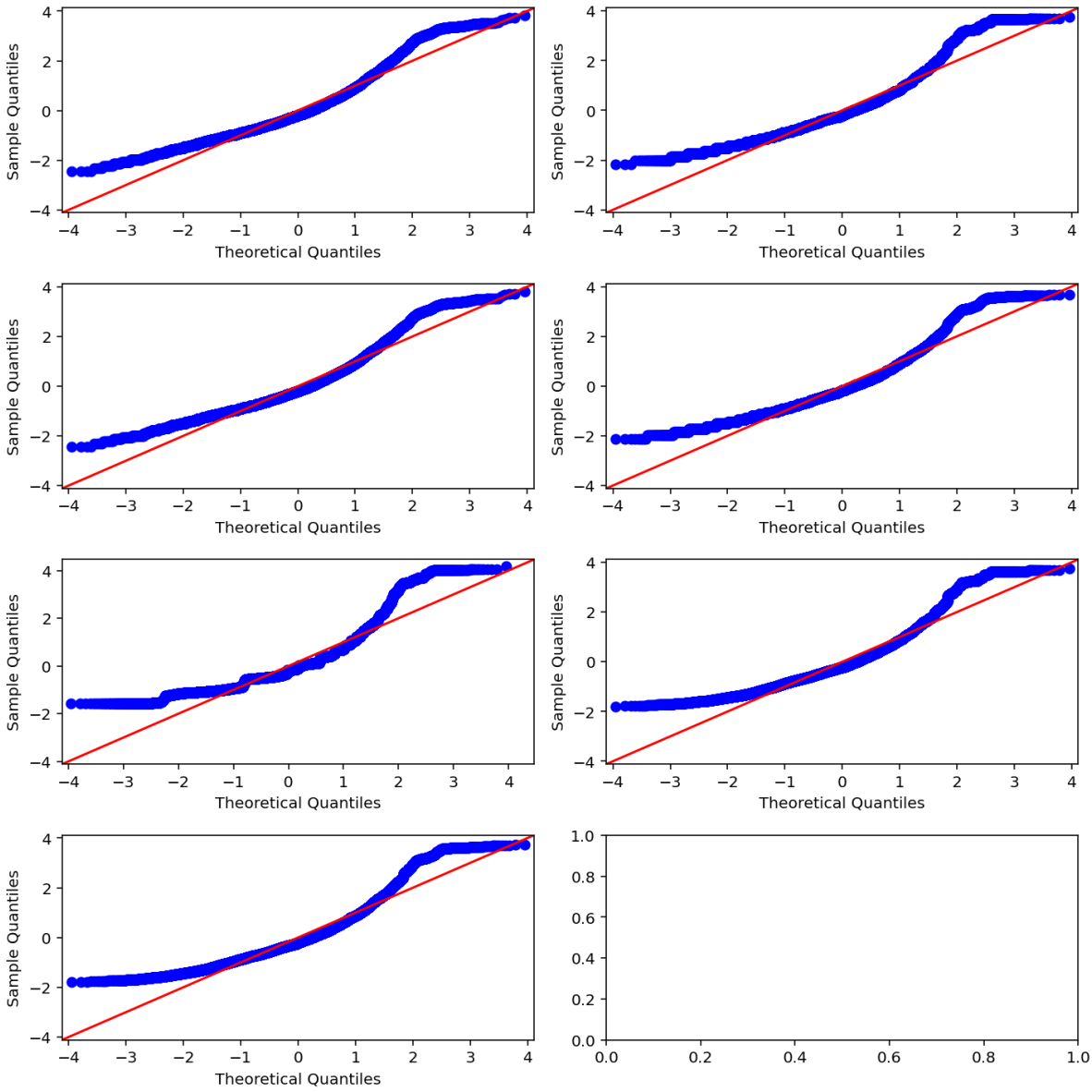
```
from scipy.stats import ks_1samp,norm
for i in trip_uuid_Group.iloc[:,3:-1].columns:
  actualtime=(np.log(trip_uuid_Group[i])-np.mean(np.log(trip_uuid_Group[i])))/np.st
  print(f"Normality test for {i}")
  tstat,pvalue=ks_1samp(actualtime,norm.cdf)
  if pvalue<=0.05:
      print("Reject null Hypothesis")
      print("Accept alternative Hypothesis(Given distribution does not follow Gauss
  else:
      print("Accept null hypothesis...(Given distribution  follow Gaussian so it is
  print('-'*100)
```

```
Normality test for actual_time_max
Reject null Hypothesis
Accept alternative Hypothesis(Given distribution does not follow Gaussian ...so it
is not log normal distribution)
--------------------------------------------------------------------------------
------------------
Normality test for osrm_time_max
Reject null Hypothesis
Accept alternative Hypothesis(Given distribution does not follow Gaussian ...so it
is not log normal distribution)
--------------------------------------------------------------------------------
------------------
Normality test for segment_actual_time_sum
Reject null Hypothesis
Accept alternative Hypothesis(Given distribution does not follow Gaussian ...so it
is not log normal distribution)
--------------------------------------------------------------------------------
------------------
Normality test for segment_osrm_time_sum
Reject null Hypothesis
Accept alternative Hypothesis(Given distribution does not follow Gaussian ...so it
is not log normal distribution)
--------------------------------------------------------------------------------
------------------
Normality test for actual_distance_to_destination_sum
Reject null Hypothesis
Accept alternative Hypothesis(Given distribution does not follow Gaussian ...so it
is not log normal distribution)
--------------------------------------------------------------------------------
------------------
Normality test for osrm_distance_max
Reject null Hypothesis
Accept alternative Hypothesis(Given distribution does not follow Gaussian ...so it
is not log normal distribution)
--------------------------------------------------------------------------------
------------------
Normality test for segment_osrm_distance_sum
Reject null Hypothesis
Accept alternative Hypothesis(Given distribution does not follow Gaussian ...so it
is not log normal distribution)
--------------------------------------------------------------------------------
------------------
```

QQ-Plot

In [21]:

```
import statsmodels.api as sm
fig, axs = plt.subplots(4, 2,figsize=(10, 10))

k=0
j=0
for i in trip_uuid_Group.iloc[:,3:-1].columns:
  actualtime=(np.log(trip_uuid_Group[i])-np.mean(np.log(trip_uuid_Group[i])))/np.st
  sm.qqplot(actualtime,line="45",fit=True,ax=axs[j,k])
  if k!=1:
    k=k+1
  else:
    k=0
    j=j+1
fig.tight_layout()
plt.show()
```



None of the above columns such as 'actual_time_max', 'osrm_time_max', 'segment_actual_time_sum',
'segment_osrm_time_sum', 'actual_distance_to_destination_sum',
'osrm_distance_max', 'segment_osrm_distance_sum'

In [21]:

In [21]:

In [21]:

In [21]:

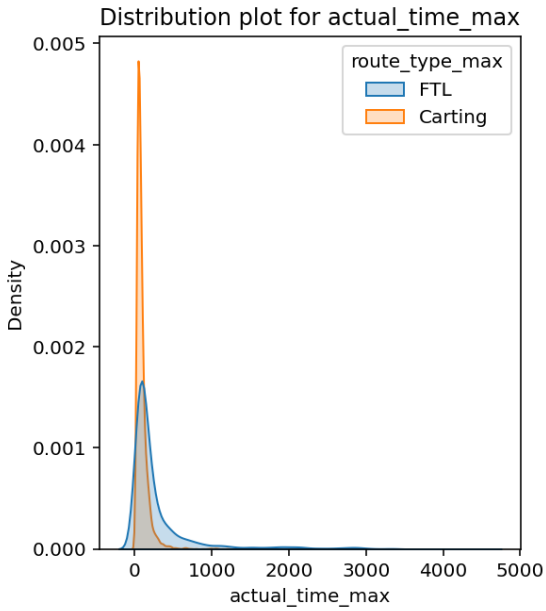
outliers analysis

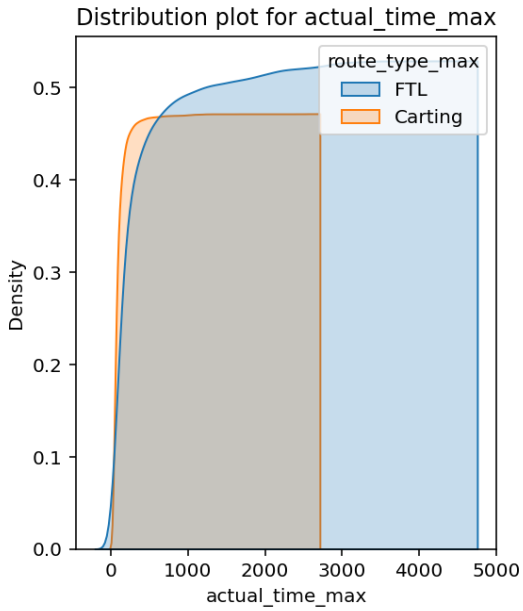
PDF and CDF

In [22]:
```
for i in trip_uuid_Group.columns[3:-1]:
  fig = plt.figure(figsize=(10,5))
  plt.subplots_adjust(wspace = 0.5,hspace=1)
```

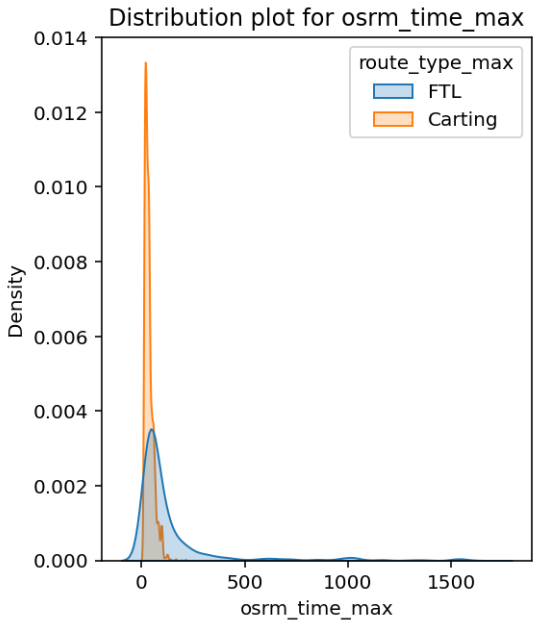
```
    ax1 = plt.subplot(121)
    sns.kdeplot(data=trip_uuid_Group,x=i, shade=True, ax=ax1,hue="route_type_max")
    plt.xlabel(f'{i}')
    plt.title(f"Distribution plot for {i}")

    ax2 = plt.subplot(122)
    sns.kdeplot(data=trip_uuid_Group,x=i, shade=True,hue="route_type_max", cumulative
    plt.xlabel(f'{i}')
    plt.title(f"Distribution plot for {i}")


    print('-'*50+i+'-'*50)
    plt.show()
```

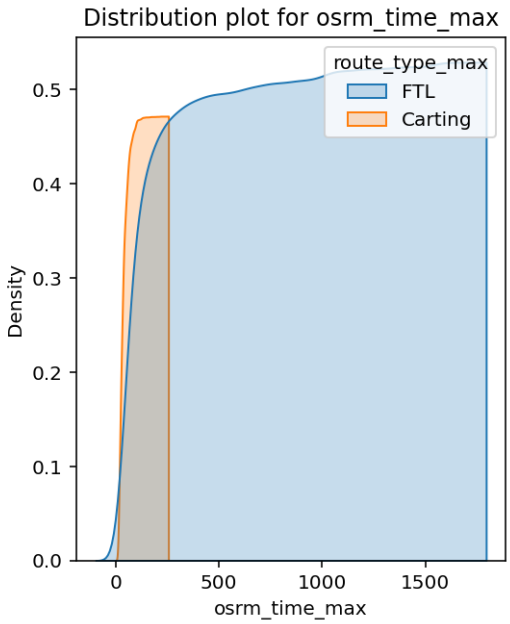
----------actual_time_max----------



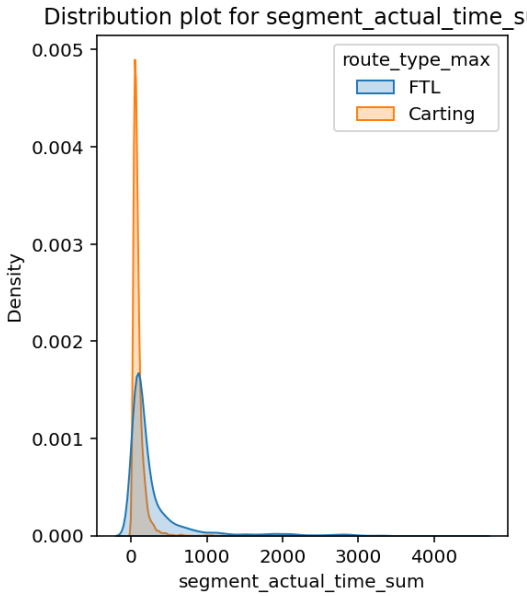


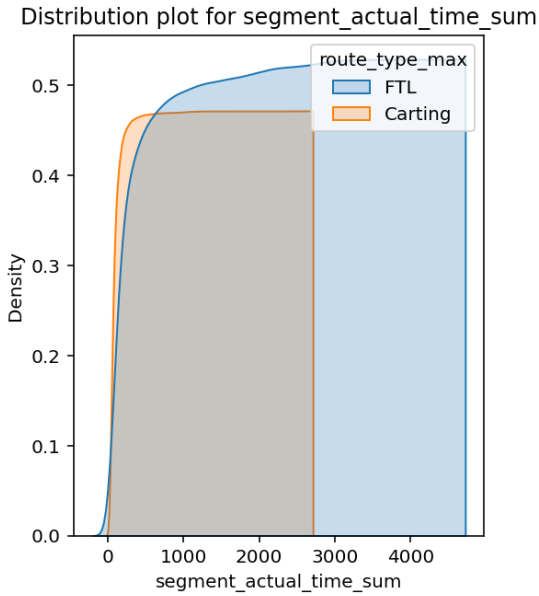
----------osrm_time_max----------



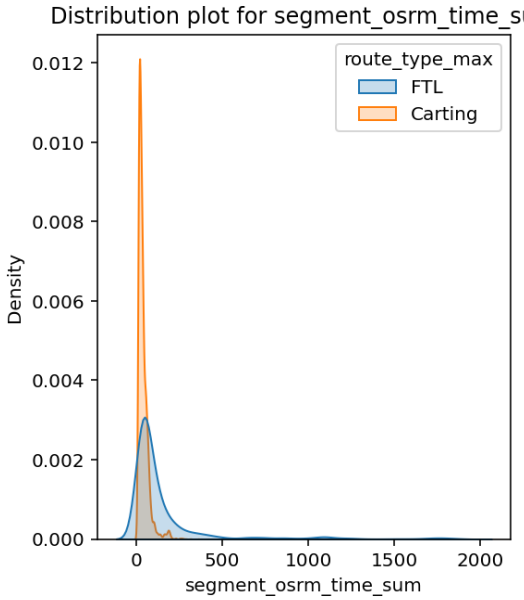


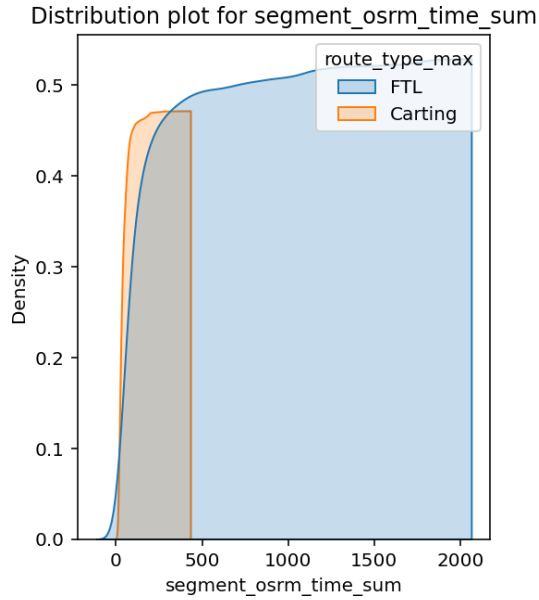
----------segment_actual_time_sum----------



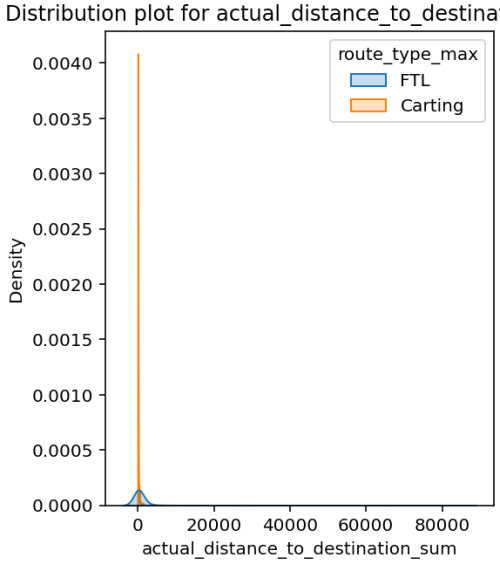


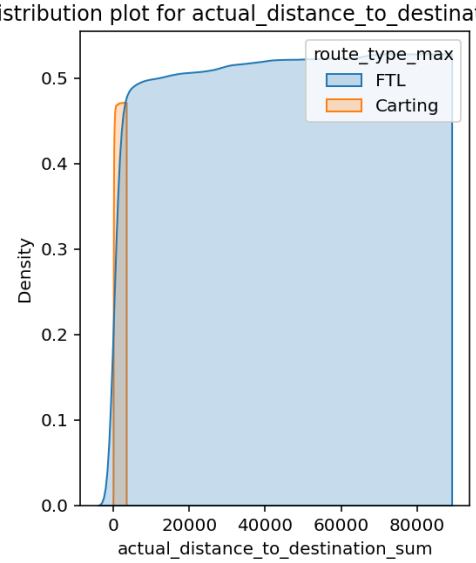
----------segment_osrm_time_sum----------



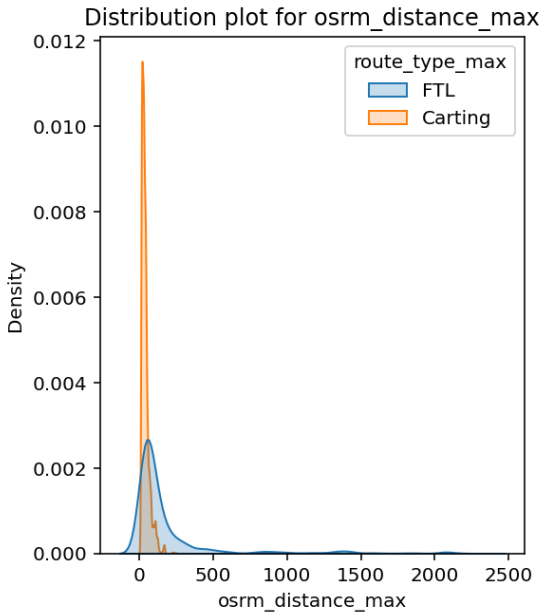


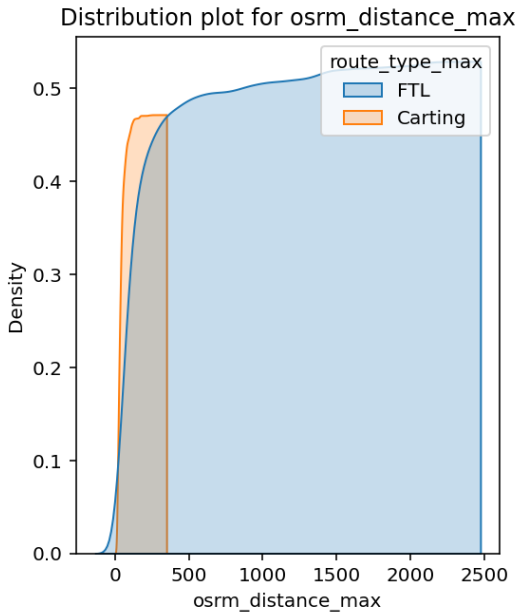
----------actual_distance_to_destination_sum----------



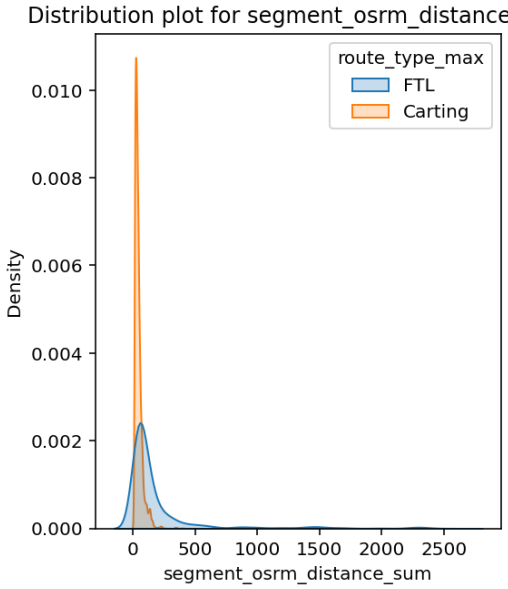


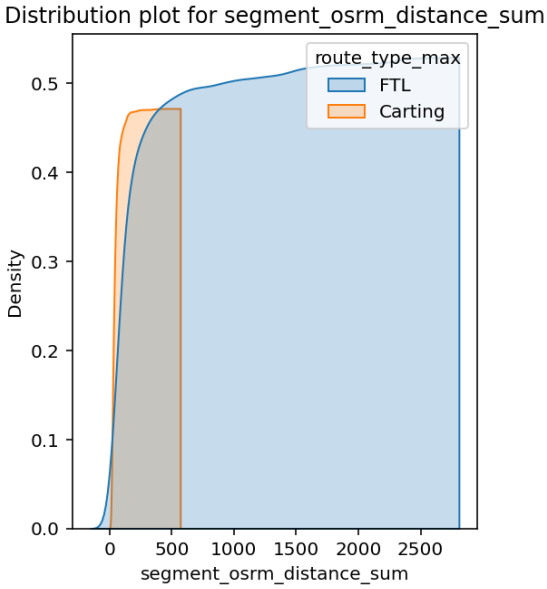
----------osrm_distance_max----------





----------segment_osrm_distance_sum----------

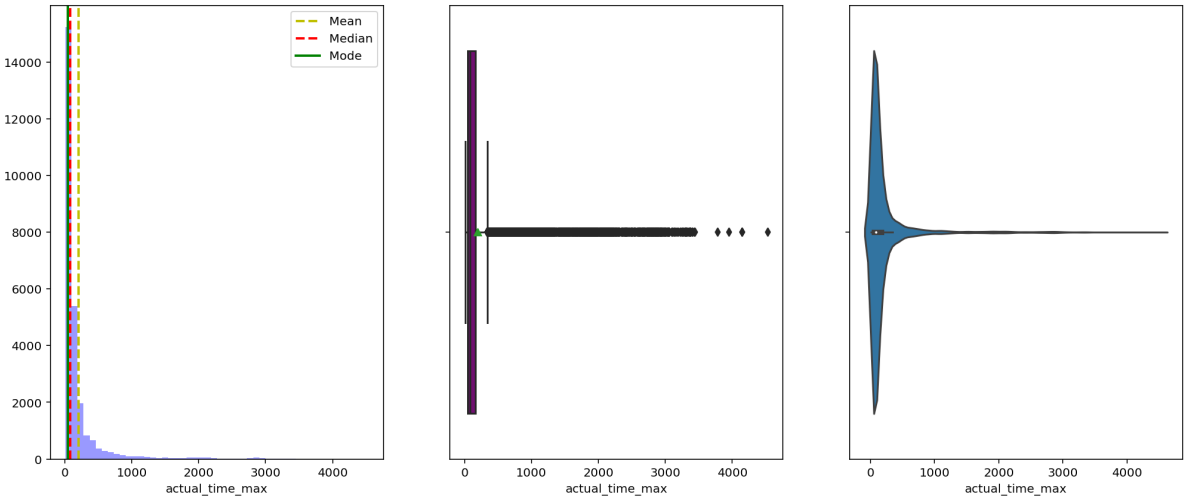




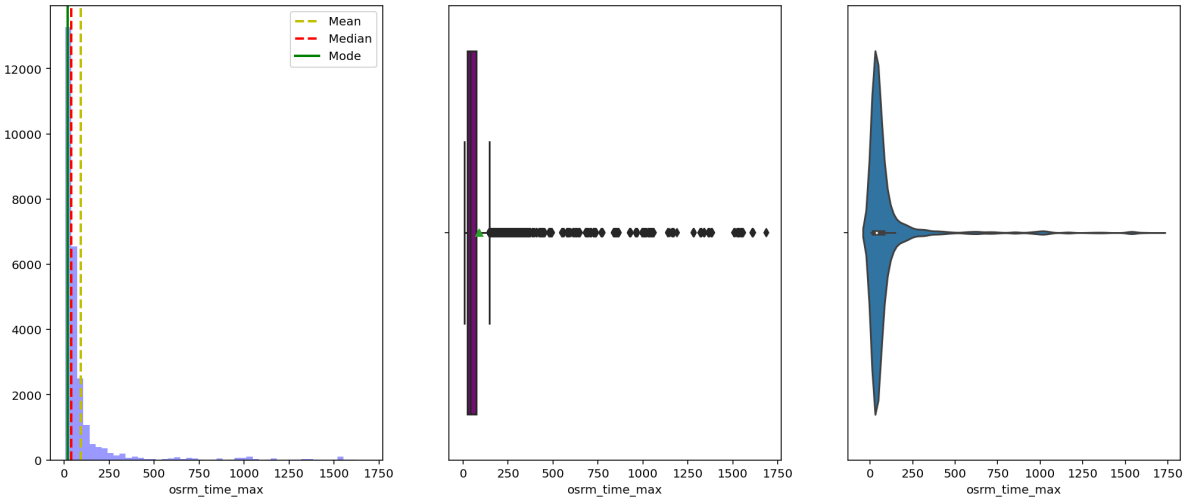
distribution and box and violin plot

In [23]:
```
for i in trip_uuid_Group.columns[3:-1]:
  dist_box_violin(trip_uuid_Group[i])
```

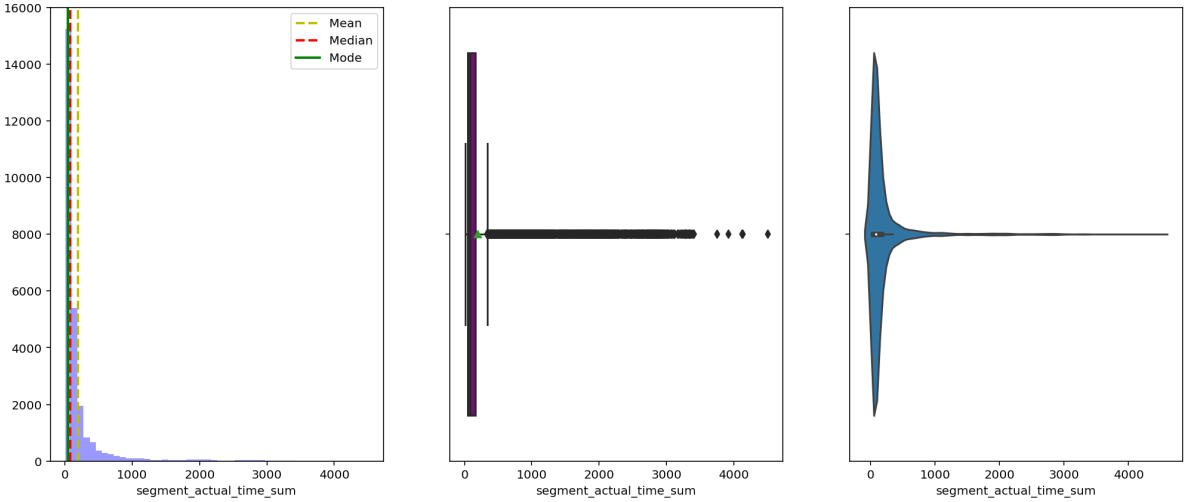
SPREAD OF DATA FOR ACTUAL_TIME_MAX



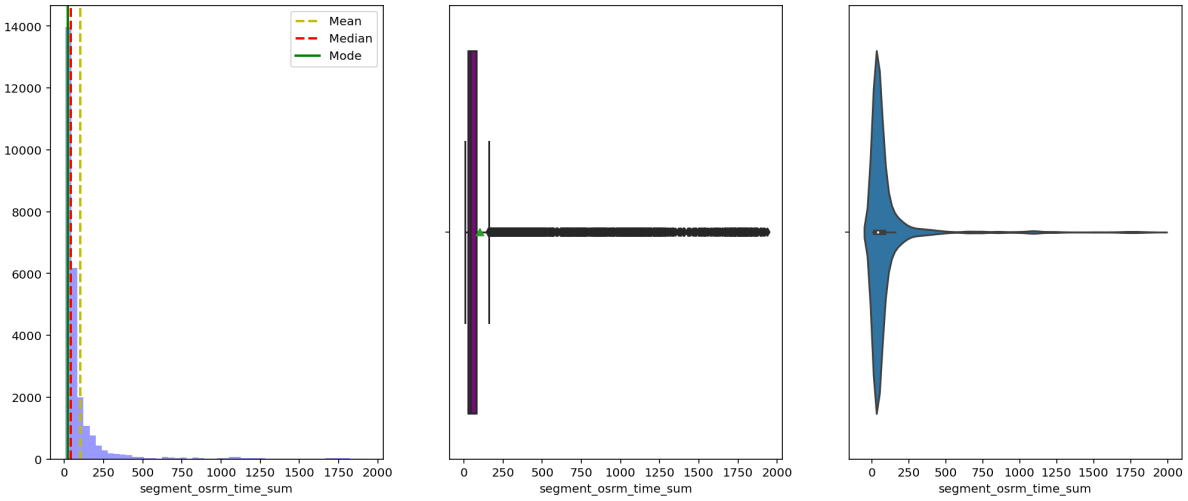
SPREAD OF DATA FOR OSRM_TIME_MAX



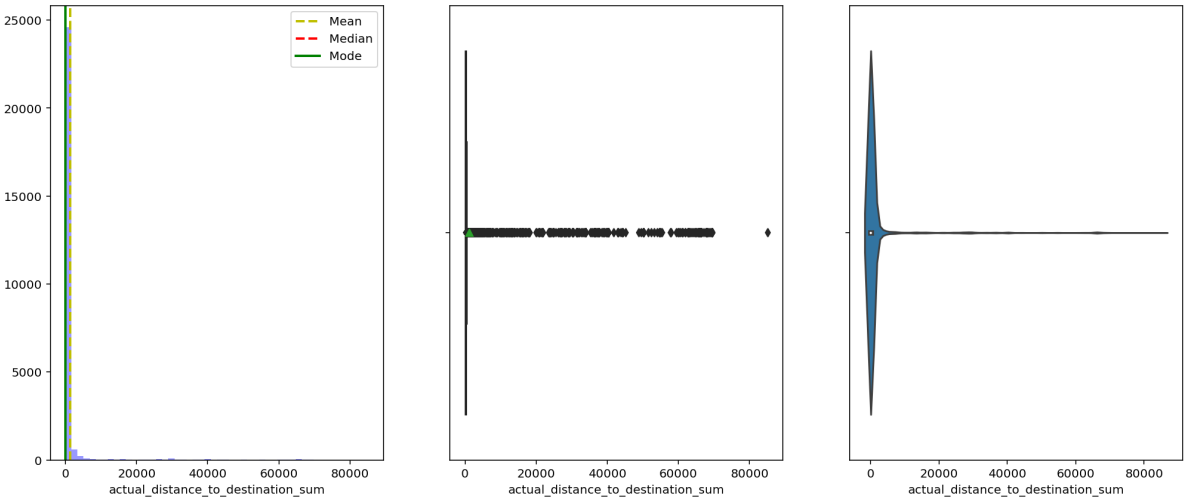
SPREAD OF DATA FOR SEGMENT_ACTUAL_TIME_SUM



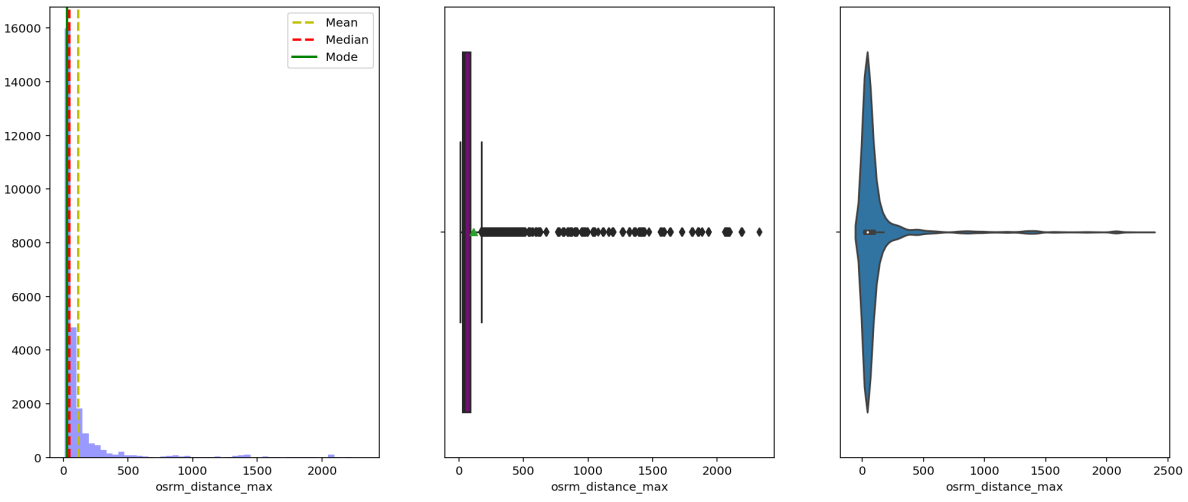
SPREAD OF DATA FOR SEGMENT_OSRM_TIME_SUM



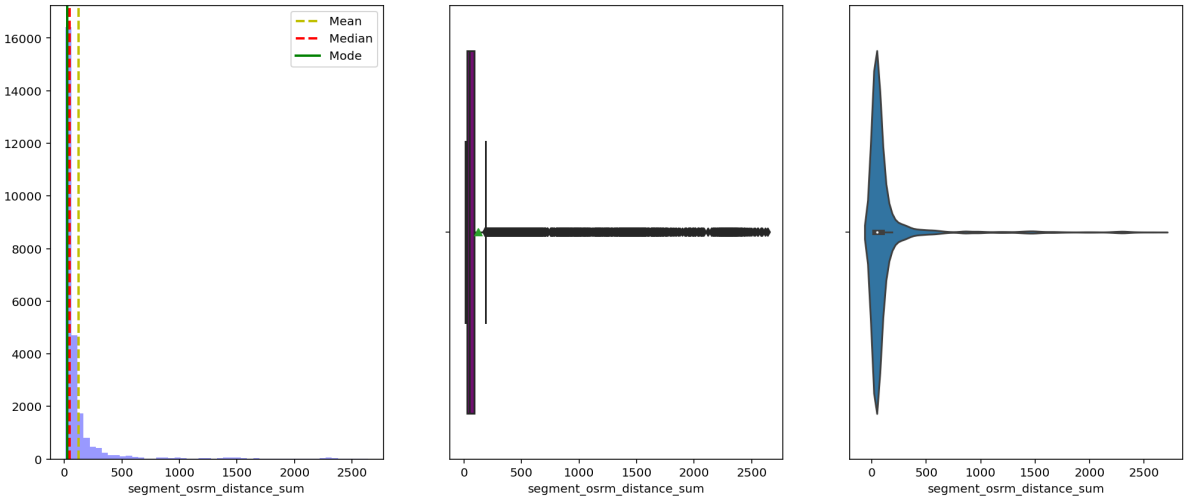
SPREAD OF DATA FOR ACTUAL_DISTANCE_TO_DESTINATION_SUM



SPREAD OF DATA FOR OSRM_DISTANCE_MAX

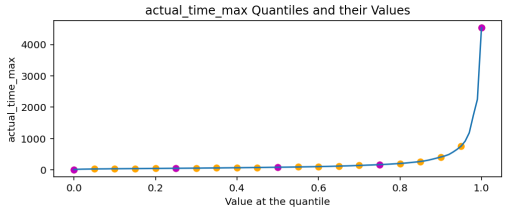


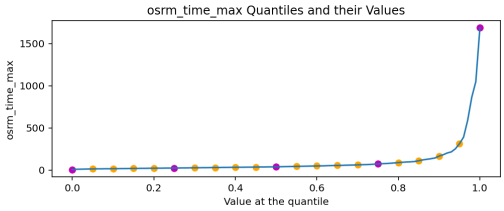
SPREAD OF DATA FOR SEGMENT_OSRM_DISTANCE_SUM

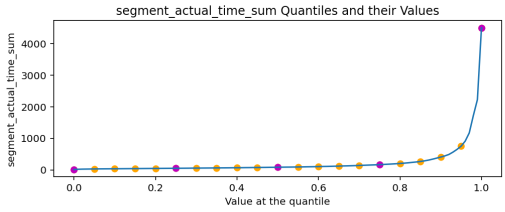


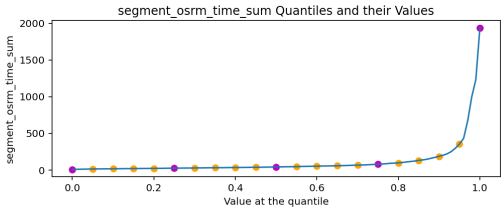
Quantile plot

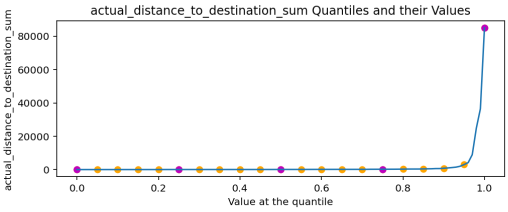
In [24]:
```
#fig = plt.figure(figsize=(10,5))
fig, axs = plt.subplots(4, 2,figsize=(20, 20))
k=0
j=0
plt.subplots_adjust(wspace = 0.5,hspace=1)
for ind,i in enumerate(trip_uuid_Group.columns[3:-1]):
  quantiles = trip_uuid_Group[i].quantile(np.arange(0,1.01,0.01), interpolation='hi

  axs[j,k].set_title(f"{i} Quantiles and their Values")
  #quantiles.plot()
  axs[j,k].plot(quantiles)
  if not ind:
    # quantiles with 0.05 difference
    axs[j,k].scatter(x=quantiles.index[::5], y=quantiles.values[::5], c='orange', l
    # quantiles with 0.25 difference
    axs[j,k].scatter(x=quantiles.index[::25], y=quantiles.values[::25], c='m', labe
  else:
    # quantiles with 0.05 difference
    axs[j,k].scatter(x=quantiles.index[::5], y=quantiles.values[::5], c='orange')
    # quantiles with 0.25 difference
    axs[j,k].scatter(x=quantiles.index[::25], y=quantiles.values[::25], c='m')

  axs[j,k].set_ylabel(f'{i}')
  axs[j,k].set_xlabel('Value at the quantile')
  #axs[j,k].legend(loc='best')

  if k!=1:
    k=k+1
  else:
    k=0
    j=j+1
plt.show()
```

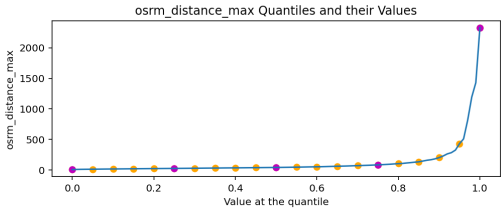


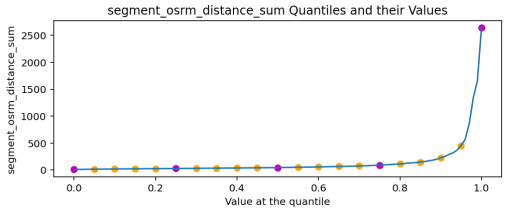


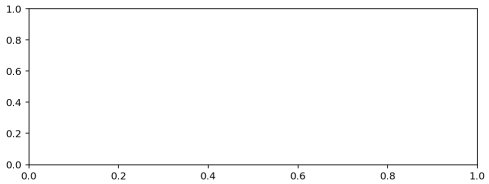












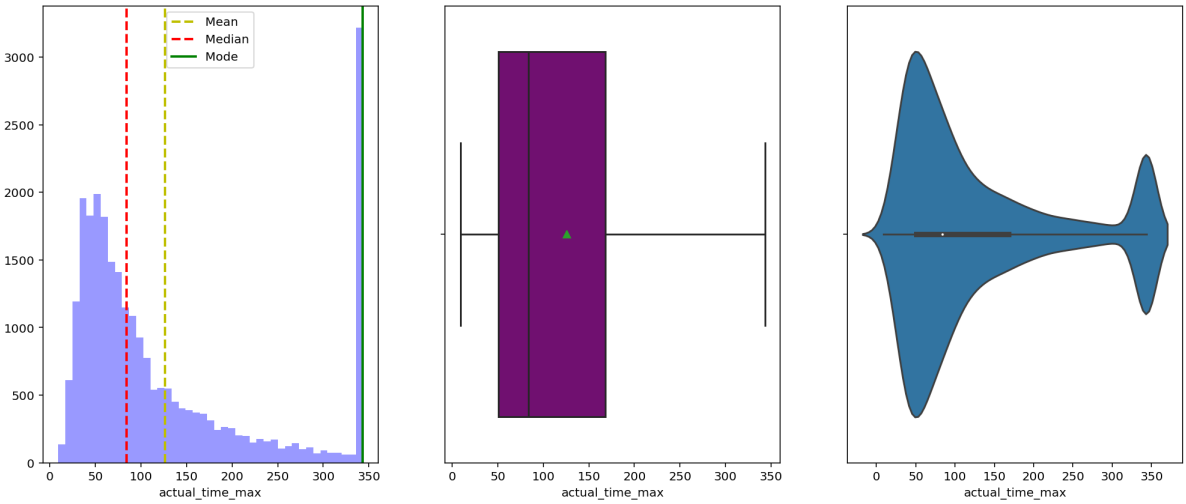
Outlier removal using IQR

In [25]:
```
for i in trip_uuid_Group.columns[3:-1]:
  Q1=np.quantile(trip_uuid_Group[i],0.25)
  Q3=np.quantile(trip_uuid_Group[i],0.75)
  IQR=Q3-Q1
  upperbound=Q3+1.5*IQR
  #print(upperbound)
  trip_uuid_Group[i]=trip_uuid_Group[i].clip(upper=upperbound)
```

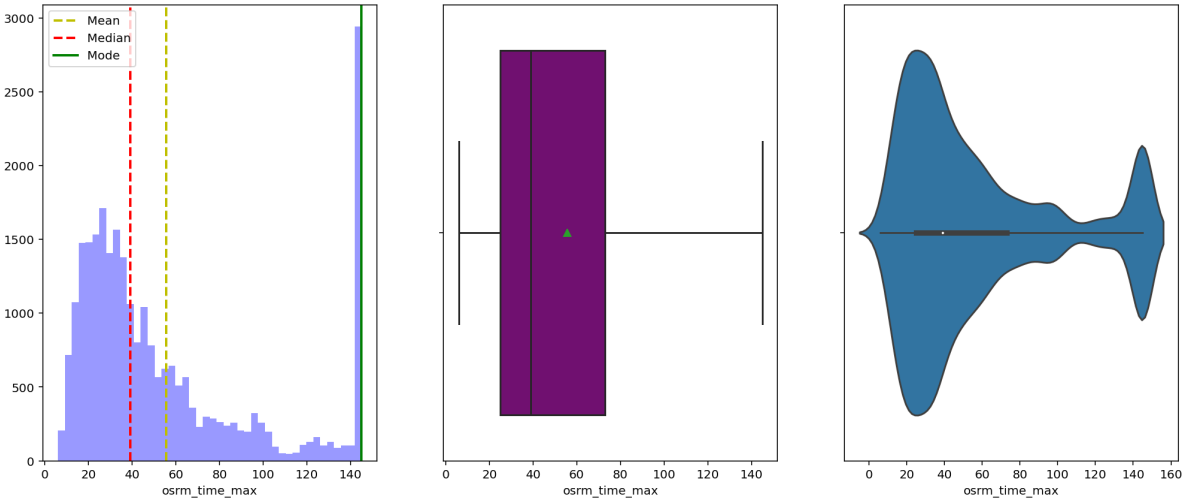
Visualisation after clipping to validate clipping process.

In [26]:
```
for i in trip_uuid_Group.columns[3:-1]:
  dist_box_violin(trip_uuid_Group[i])
```

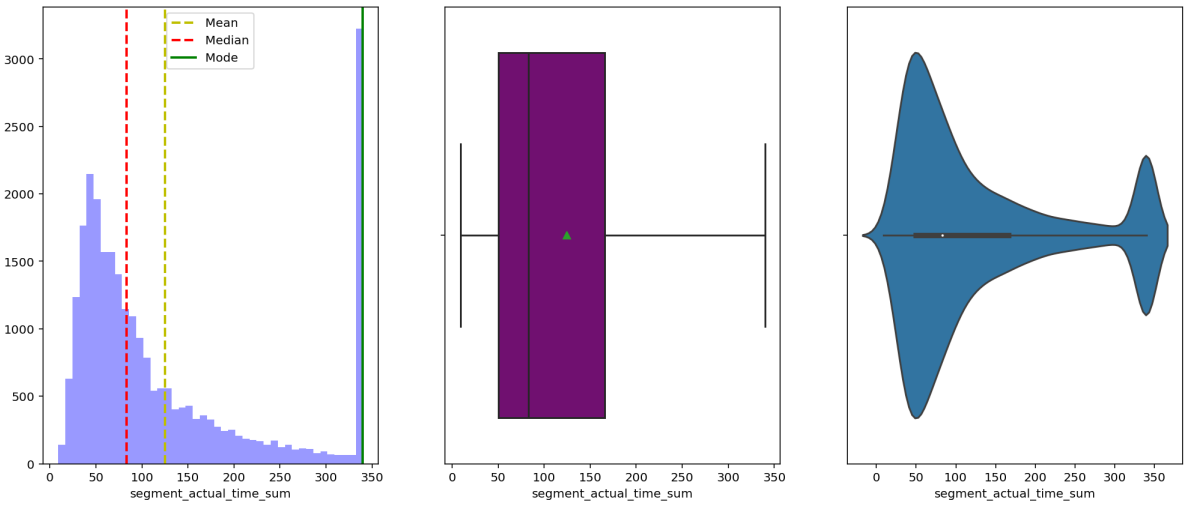
SPREAD OF DATA FOR ACTUAL_TIME_MAX



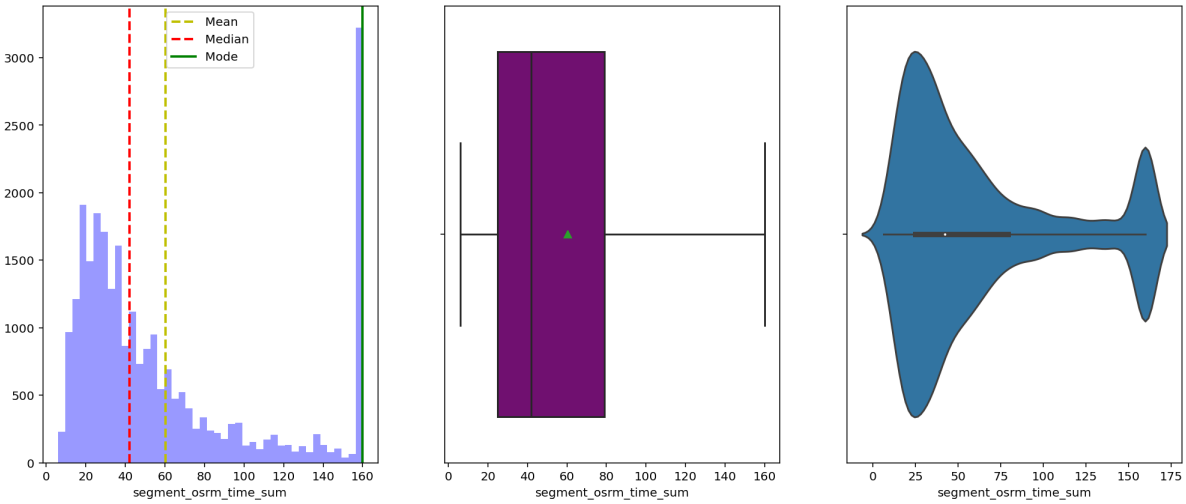
SPREAD OF DATA FOR OSRM_TIME_MAX



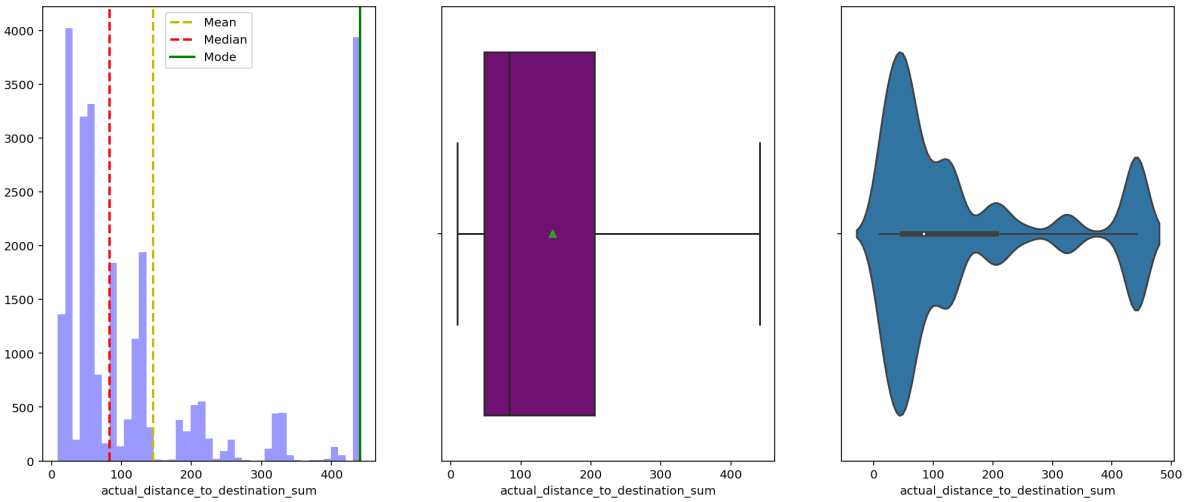
SPREAD OF DATA FOR SEGMENT_ACTUAL_TIME_SUM



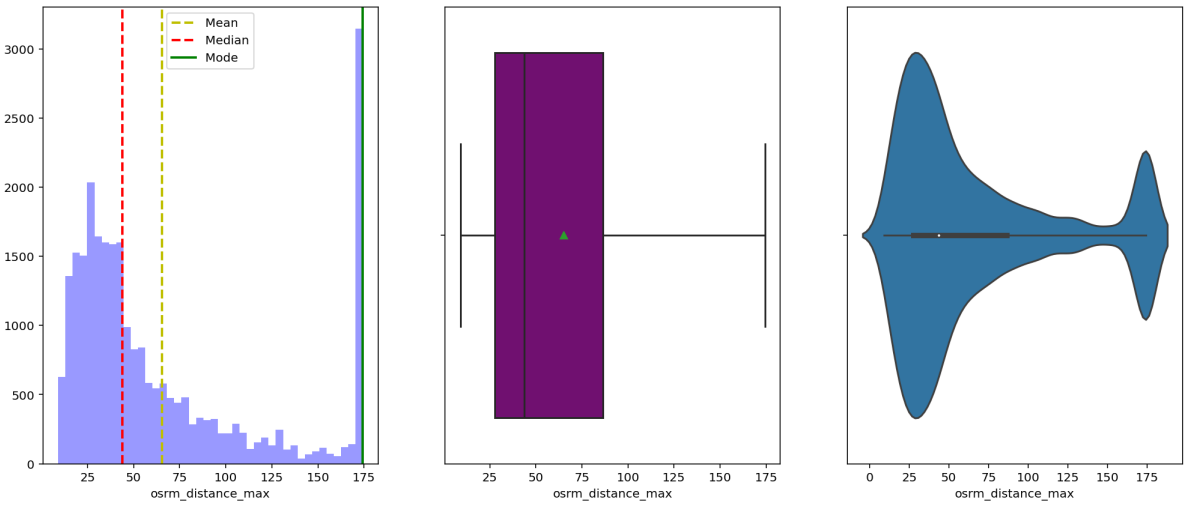
SPREAD OF DATA FOR SEGMENT_OSRM_TIME_SUM



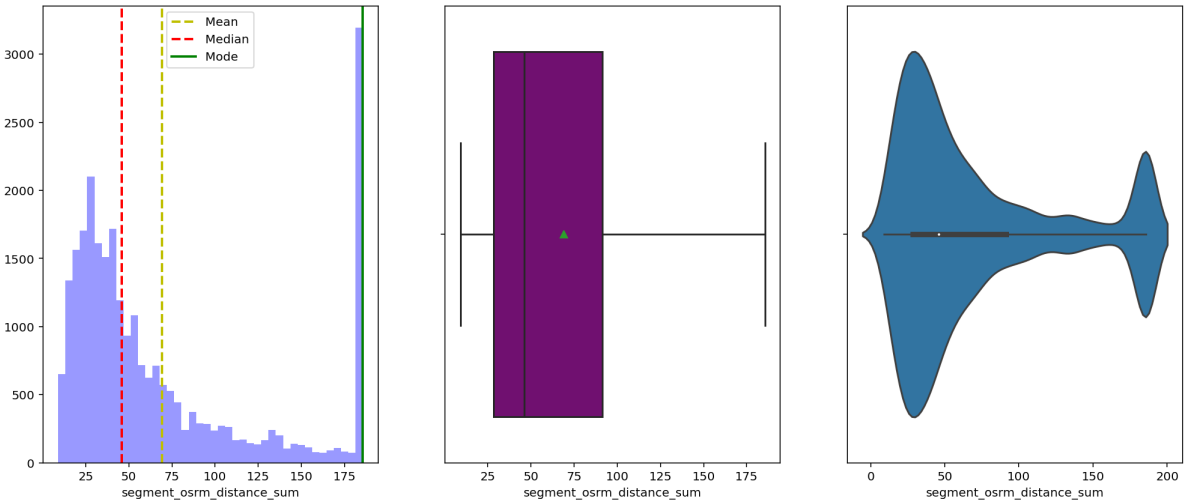
SPREAD OF DATA FOR ACTUAL_DISTANCE_TO_DESTINATION_SUM



SPREAD OF DATA FOR OSRM_DISTANCE_MAX



SPREAD OF DATA FOR SEGMENT_OSRM_DISTANCE_SUM



Encoding- one hot

In [27]:
```
dehivery_data["route_type"]=dehivery_data["route_type"].astype("category")
dehivery_data[pd.get_dummies(dehivery_data[["route_type"]],drop_first=True).columns
```

In [28]:
```
dehivery_data.head().T
```

Out[28]:

	0	1	
trip_creation_time	2018-09-20 02:35:36.476840	2018-09-20 02:35:36.476840	2018-09-2 02:35:36.47684
data	training	training	trainin
route_schedule_uuid	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	thanos::sroute:eb7bfc78 b351-4c0e-a95 fa3d5c3
route_type	Carting	Carting	Cartin
trip_uuid	trip-153741093647649320	trip-153741093647649320	trip 15374109364764932
source_center	IND388121AAA	IND388121AAA	IND388121AA
source_name	Anand_VUNagar_DC (Gujarat)	Anand_VUNagar_DC (Gujarat)	Anand_VUNagar_D (Gujara
destination_center	IND388620AAB	IND388620AAB	IND388620AA
destination_name	Khambhat_MotvdDPP_D (Gujarat)	Khambhat_MotvdDPP_D (Gujarat)	Khambhat_MotvdDPP_ (Gujara
od_start_time	2018-09-20 03:21:32.418600	2018-09-20 03:21:32.418600	2018-09-2 03:21:32.41860
od_end_time	2018-09-20 04:47:45.236797	2018-09-20 04:47:45.236797	2018-09-2 04:47:45.23679
start_scan_to_end_scan	86.0	86.0	86
is_cutoff	True	True	Tru
cutoff_factor	9	18	2
cutoff_timestamp	2018-09-20 04:27:55	2018-09-20 04:17:55	2018-09-2 04:01:19.50558
actual_distance_to_destination	10.43566	18.936842	27.63727
actual_time	14.0	24.0	40
osrm_time	11.0	20.0	28
osrm_distance	11.9653	21.7243	32.539
factor	1.272727	1.2	1.42857
segment_actual_time	14.0	10.0	16
segment_osrm_time	11.0	9.0	7
segment_osrm_distance	11.9653	9.759	10.815
segment_factor	1.272727	1.111111	2.28571
destination_state	Gujarat	Gujarat	Gujara
destination_city	Khambhat	Khambhat	Khambha
destination_place	MotvdDPP	MotvdDPP	MotvdDP
destination_code	D	D	
source_state	Gujarat	Gujarat	Gujara
source_city	Anand	Anand	Anan
source_place	VUNagar	VUNagar	VUNaga

	0	1	
source_code	DC	DC	D
date_trip_creation	2018-09-20	2018-09-20	2018-09-2
day_trip_creation	20	20	2
month_trip_creation	9	9	
year_trip_creation	2018	2018	201
hour_trip_creation	2	2	
dow_trip_creation	Thursday	Thursday	Thursda
woy_trip_creation	38	38	3
handling_time	0 days 01:26:12.818197	0 days 01:26:12.818197	0 days 01:26:12.81819
handling_time_seconds	5172.818197	5172.818197	5172.81819
handling_time_minutes	86.213637	86.213637	86.21363
handling_time_hours	1.436894	1.436894	1.43689
handling_time_days	0.059871	0.059871	0.05987
handling_time_full_days	0	0	
route_type_FTL	0	0	

In [29]:
```
from sklearn.preprocessing import StandardScaler
num_data=dehivery_data[dehivery_data.select_dtypes(["float64","int64"]).columns]
scaler=StandardScaler()
train_data=scaler.fit(num_data[dehivery_data.data=="training"])
transformed_train_data=train_data.transform(num_data[dehivery_data.data=="training"
transformed_test_data=train_data.transform(num_data[dehivery_data.data!="training"]

transformed_train_data=pd.DataFrame(transformed_train_data,columns=num_data.columns
transformed_test_data=pd.DataFrame(transformed_test_data,columns=num_data.columns)
```

In [30]:
```
transformed_test_data.head()
```

Out[30]:

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	osrm_time	osr
0	-0.850006	-0.610287	-0.608849	-0.664297	-0.641837	
1	-0.850006	-0.569019	-0.570439	-0.630258	-0.592400	
2	-0.841161	-0.616182	-0.618058	-0.611537	-0.579217	
3	-0.838213	-0.610287	-0.612811	-0.628556	-0.546259	
4	-0.838213	-0.563124	-0.565180	-0.594517	-0.506709	

5 rows × 21 columns

NOTE: here we are not ussing this trasformed data for further analysis we will use this transformed data when we are building machine learning and deep learning models

In [ ]:

HYPOTHESIS ANALYSIS

references:**

(https://gist.github.com/mblondel/1761714

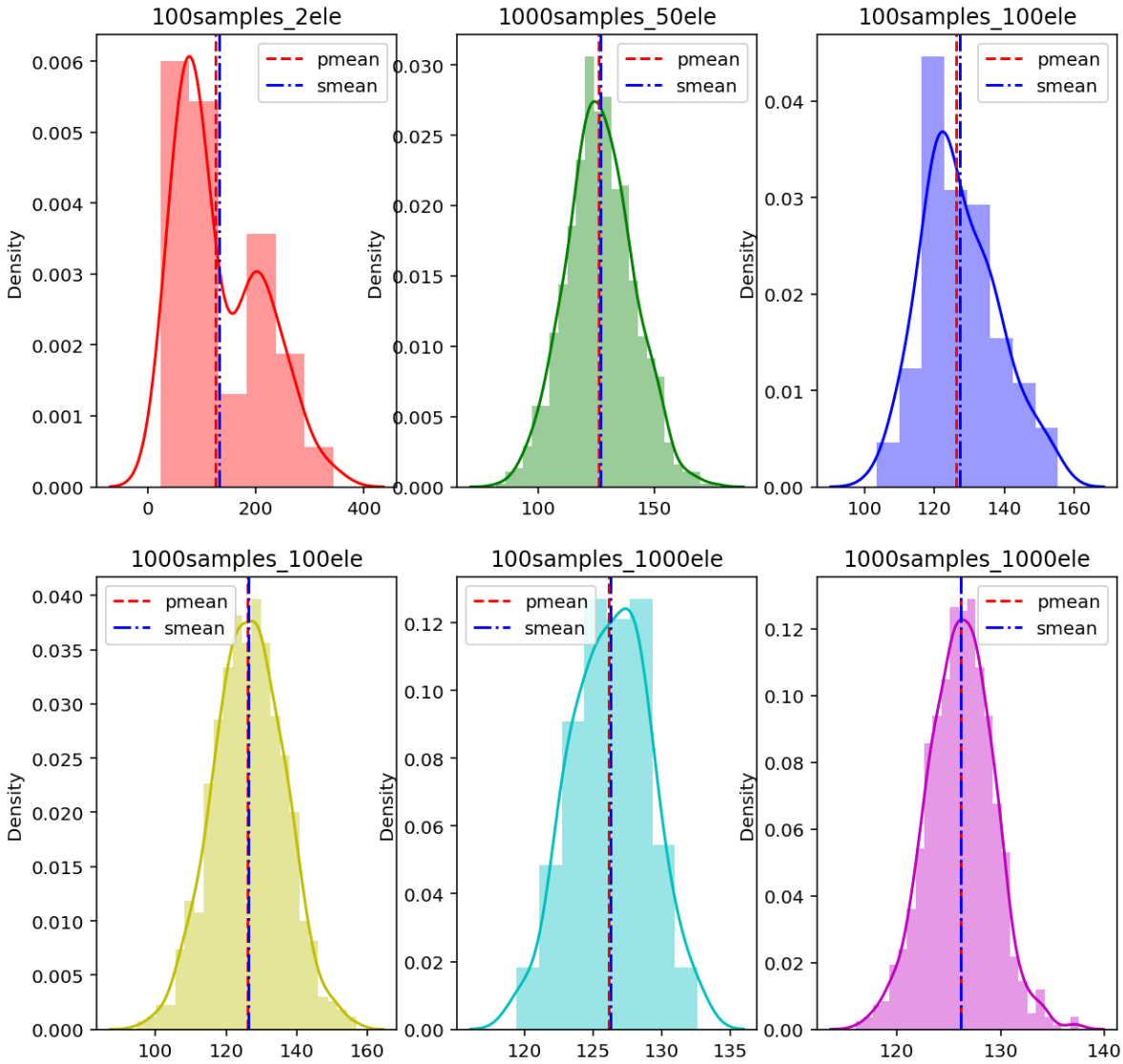
https://jmyao17.github.io/Statistics/Nonparametric_Statistical_Significance_Tests.html//)**

In [43]:
```
def get_means_of_n_samples_with_m_size(data, n, m):
    sample_mean_m_samples_n_element = []
    for i in range(0,n):
        sample = random.sample(range(0, data.shape[0]), m)
        #print(samples)
        sample_mean_m_samples_n_element.append(data[sample].mean())
    return sample_mean_m_samples_n_element
def central_limit_theorem(data, population_mean , i, j, color, key):
    sns.distplot(np.array(data), color=color, ax=axs[i, j])
    axs[i, j].axvline(population_mean, linestyle="--", color='r', label="pmean")
    axs[i, j].axvline(np.array(data).mean(), linestyle="-.", color='b', label="smea
    axs[i, j].set_title(key)
    axs[i, j].legend()
def diff_in_samples(dist1, dist2, gender1, gender2):
    print(f"The average {gender1} "+str(len(dist1))+" "+gender1+" =",dist1.mean())
    print(f"The average {gender2} "+str(len(dist2))+" "+gender2+"=",dist2.mean())
    diff_in_mean = dist1.mean()-dist2.mean()
    print("The difference between mean of "+gender1+" spending and "+gender2+" sper
    return diff_in_mean
def calculate_p_value(sample1, sample2, diff, alpha):
    #Step 2- Create list to store the average values of both the samples and the di
    difference=[]
    #Sampling the data for 1000 times
    total_sample = list(sample1)
    total_sample.extend(sample2)
    total_sample = np.array(total_sample)
    for i in range(0,1000):
        #Picking 100 random numbers
        samples = random.sample(range(0, len(total_sample)), 100)
        #First 50 random numbers are taken as set 1
        set1 = total_sample[samples[:50]].mean()
        #Next 50 random numbers are taken as set 2
        set2 = total_sample[samples[50:]].mean()
        #Taking the differnce between the two sets
        difference.append(set1 - set2)

    #Step3- Sorting the values and counting the number of values greater than the t
    difference.sort()
    count = sum(((i > diff) and (i>0)) for i in difference)
    pValue = count/len(difference)
    print("Percentage of values greater than the difference",diff," =",pValue*100,'
    print("The pValue = ",pValue, "and the significance P(Reject H0 when H0 is true
    if pValue>alpha:
        print("We fail to reject the null hypothesis")
    else:
        print("We can reject the null hypothesis")

    print('_'*50)
    return difference
def plt_cdfplot_withthreshold(j,c,difference,threshold,sample):
```

```
    sns.kdeplot(difference,cumulative=True,color=c, ax=axs[j])
    axs[j].axvline(threshold, linestyle="--", color='r', label=int(threshold))
    axs[j].set_title("CDF of differences for " + str(sample)+" samples")
    axs[j].legend()
    axs[j].grid()
```

In [ ]:

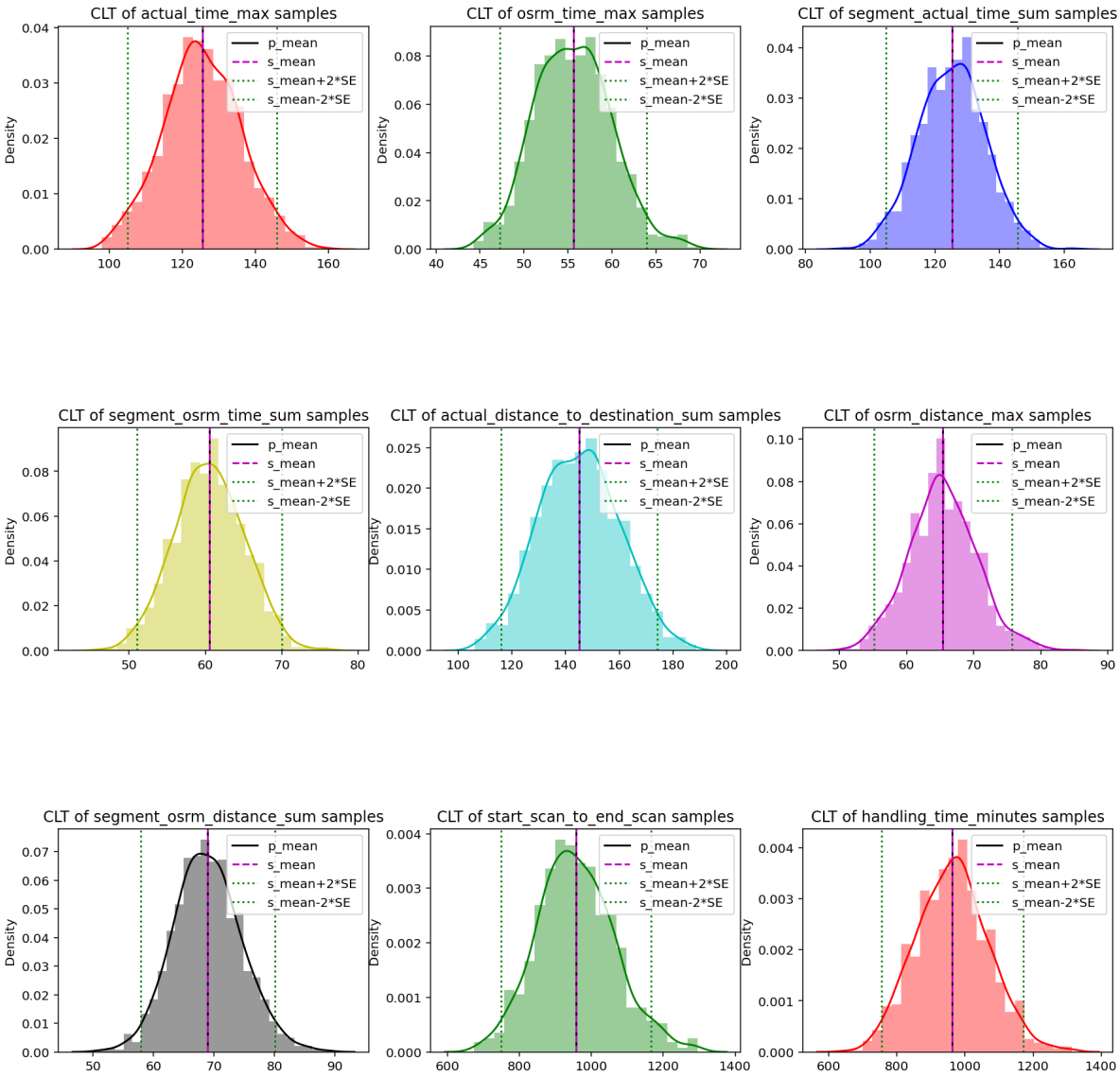
```
data=trip_uuid_Group.actual_time_max
# population mean
population_mean = data.mean().round(3)
# population std
population_std = data.std().round(3)
sample_means = dict()
sample_means['100samples_2ele'] = get_means_of_n_samples_with_m_size(data,100, 2)
sample_means['1000samples_50ele'] = get_means_of_n_samples_with_m_size(data,1000, 5

sample_means['100samples_100ele'] = get_means_of_n_samples_with_m_size(data,100, 10
sample_means['1000samples_100ele'] = get_means_of_n_samples_with_m_size(data,1000,

sample_means['100samples_1000ele'] = get_means_of_n_samples_with_m_size(data,100, 1
sample_means['1000samples_1000ele'] = get_means_of_n_samples_with_m_size(data,1000,
#red, green, blue, yellow, etc
colrs = ['r','g','b','y', 'c', 'm', 'k']
plt_grid  = [(0,0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2)]
sample_sizes = [(100,50), (1000, 50), (100, 100), (1000, 100), (100, 1000), (1000,

fig, axs = plt.subplots(2, 3,  figsize=(10, 10))
for i, key in enumerate(sample_means.keys()):
    central_limit_theorem(sample_means[key], population_mean , plt_grid[i][0], plt_
plt.show()
```



In [ ]:

```
def plt_confidence_interval(data, sample_mean, population_std, i, j, color,Category
    sns.distplot(data, color=color, ax=axs[i, j])
    plt.subplots_adjust(hspace = 0.8)
    p_mean=data.mean()
    right=sample_mean+2*(population_std/np.sqrt(sample_size))
    left=sample_mean-2*(population_std/np.sqrt(sample_size))
    axs[i, j].axvline(p_mean, linestyle="-", color='k', label="p_mean")
    axs[i, j].axvline(sample_mean, linestyle="--", color='m', label="s_mean")
    axs[i, j].axvline(sample_mean+2*(population_std/np.sqrt(sample_size)), linestyl
    axs[i, j].axvline(sample_mean-2*(population_std/np.sqrt(sample_size)), linestyl
    axs[i, j].legend()
    if Category_Name:
      axs[i, j].set_title("CLT of " + Category_Name +" samples")
    return p_mean,sample_mean,left,right
CLT_categories_MASK=[trip_uuid_Group[i] for i in trip_uuid_Group.columns[3:-1]]+[de
#print(len(CLT_categories_MASK))
Cetgories=[*trip_uuid_Group.columns[3:-1]]+["start_scan_to_end_scan","handling_time
sample_means_For_category = dict()
sample_size=100
sample_iterations=1000
for cetegoryName,eachCategory in  zip(Cetgories,CLT_categories_MASK):
  #print(data_category.shape)
  sample_means_For_category[str(sample_iterations)+'samples_'+str(sample_size)+'ele



colrs = ['r','g','b','y', 'c', 'm', 'k','g','r','g','b']
plt_grid  = [(0,0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2),(2,0), (2,1) , (2,2)]
```

```
fig, axs = plt.subplots(3, 3,  figsize=(15, 15))
#plt.rcParams['figure.constrained_layout.use'] = True

for i, key in enumerate(sample_means_For_category.keys()):
    #central_limit_theorem(sample_means_male[key], population_mean , plt_grid[i][0]
    #data_category=np.array(CLT_categories_MASK[i]['Purchase'].values)

    plt_confidence_interval(np.array(sample_means_For_category[key]), np.array(samp
plt.show()
```



ACTUAL Time vs osm time

overall mean time

- Null Hypothesis: the mean delevery time of actually time and osm time are same
- Alternative Hypothesis: the mean delevery time of actually time and osm time are Not same

How actual actual time to google time

- Null Hypothesis: the delevery time of actually time and osm time are same

- Alternative Hypothesis: the delevery time of actually time and osm time are Not same

In [ ]:

PARAMETRIC TEST-ttest

In [ ]:
```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05

actual_time=trip_uuid_Group.actual_time_max
osrm_time=trip_uuid_Group.osrm_time_max
bs_actual_time=np.random.choice(actual_time,size=1000)
bs_osrm_time=np.random.choice(osrm_time,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery time of actually time and osm time are same.
alternative_Hypothesis="HA: the mean delevery time of actually time and osm time ar

ttest_statistic,p_value,_=ttest_ind(bs_actual_time,bs_osrm_time,alternative="two-si
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : 19.148857318081962 | p-value : 3.561806290294747e-75
Reject ---> H0: the mean delevery time of actually time and osm time are same.
Accept ---> HA: the mean delevery time of actually time and osm time are Not same.
```

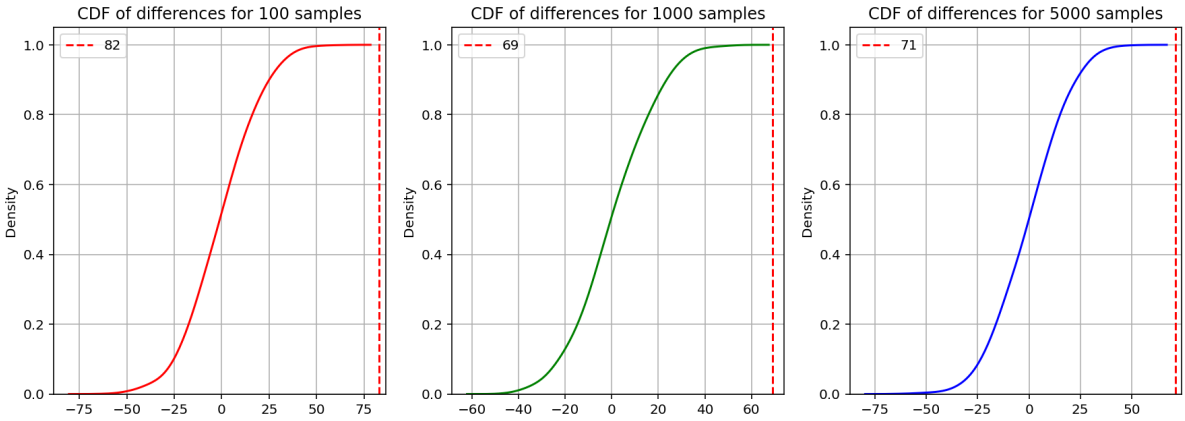
NON-PARAMETRIC(Permutation resampling)

In [ ]:
```
actual_time=trip_uuid_Group.actual_time_max
osrm_time=trip_uuid_Group.osrm_time_max
sample_sizes = [100,1000,5000]
alpha = 0.05
fig, axs = plt.subplots(1, 3,figsize=(15, 5))
for j, i in enumerate(sample_sizes):
    print("For Sample Size: ", 2*i)

    sample_actualtime=actual_time[random.sample(range(0, actual_time.shape[0]), i)]
    sample_osrmtime=osrm_time[random.sample(range(0, osrm_time.shape[0]), i)]

    diff_in_mean = diff_in_samples(sample_actualtime,sample_osrmtime, "actual_time"

    #Step 1- Combine b
    #Step 1- Combine both samples of size 50 each to a large sample of size 100 to
    differences = calculate_p_value(sample_actualtime,sample_osrmtime,diff_in_mean,
    plt_cdfplot_withthreshold(j, colrs[j],differences,threshold=diff_in_mean,sample
```

```
For Sample Size:  200
The average actual_time 100 actual_time = 142.18
The average osrm_time 100 osrm_time= 59.24
The difference between mean of actual_time spending and osrm_time spendings (diff_
100)= 82.94
Percentage of values greater than the difference 82.94  = 0.0 %
The pValue =  0.0 and the significance P(Reject H0 when H0 is true)= 0.05
We can reject the null hypothesis
```

```
For Sample Size:  2000
The average actual_time 1000 actual_time = 123.16
The average osrm_time 1000 osrm_time= 54.066
The difference between mean of actual_time spending and osrm_time spendings (diff_
1000)= 69.094
Percentage of values greater than the difference 69.094  = 0.0 %
The pValue =  0.0 and the significance P(Reject H0 when H0 is true)= 0.05
We can reject the null hypothesis
```

```
For Sample Size:  10000
The average actual_time 5000 actual_time = 126.3735
The average osrm_time 5000 osrm_time= 55.0972
The difference between mean of actual_time spending and osrm_time spendings (diff_
5000)= 71.2763
Percentage of values greater than the difference 71.2763  = 0.0 %
The pValue =  0.0 and the significance P(Reject H0 when H0 is true)= 0.05
We can reject the null hypothesis
```



NON-PARAMETRIC (mannwhitneyu)

In [ ]:

```python
# two-sample wilcoxon test
# a.k.a Mann Whitney U
from scipy.stats import mannwhitneyu
#dehivery_data.head().T#start_scan_to_end_scan,handling_time_minutes
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm

significance_value=0.05

actual_time=trip_uuid_Group.actual_time_max
osrm_time=trip_uuid_Group.osrm_time_max
bs_actual_time=np.random.choice(actual_time,size=1000)
bs_osrm_time=np.random.choice(osrm_time,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery time of actually time and osm time are same.
alternative_Hypothesis="HA: the mean delevery time of actually time and osm time ar

ttest_statistic,p_value=mannwhitneyu(bs_actual_time,bs_osrm_time,alternative="two-s
```

```
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : 766468.5 | p-value : 1.1866508840574162e-94
Reject ---> H0: the mean delevery time of actually time and osm time are same.
Accept ---> HA: the mean delevery time of actually time and osm time are Not same.
```

Diffrence od_start_time and od_end_time vs start_scan_to_end_scan

Null hypothesis: H0: the mean delevery time of "diffrence between start and end time" and "start_scan_to_end_scan" are same. **alternative_Hypothesis :** HA: the mean delevery time of "diffrence between start and end time" and "start_scan_to_end_scan" are Not same.

In [ ]:

PARAMETRIC TEST-ttest

In [ ]:
```
#dehivery_data.head().T#start_scan_to_end_scan,handling_time_minutes
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05

start_scan_to_end_scan=dehivery_data.start_scan_to_end_scan
handling_time_minutes=dehivery_data.handling_time_minutes
bs_start_scan_to_end_scan=np.random.choice(start_scan_to_end_scan,size=1000)
bs_handling_time_minutes=np.random.choice(handling_time_minutes,size=1000)
print("**********Two taield test**************")

null_Hypothesis="""H0: the mean delevery time of "diffrence between start and end t
alternative_Hypothesis="""HA: the mean delevery time of "diffrence between start ar

ttest_statistic,p_value,_=ttest_ind(bs_start_scan_to_end_scan,bs_handling_time_minu
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

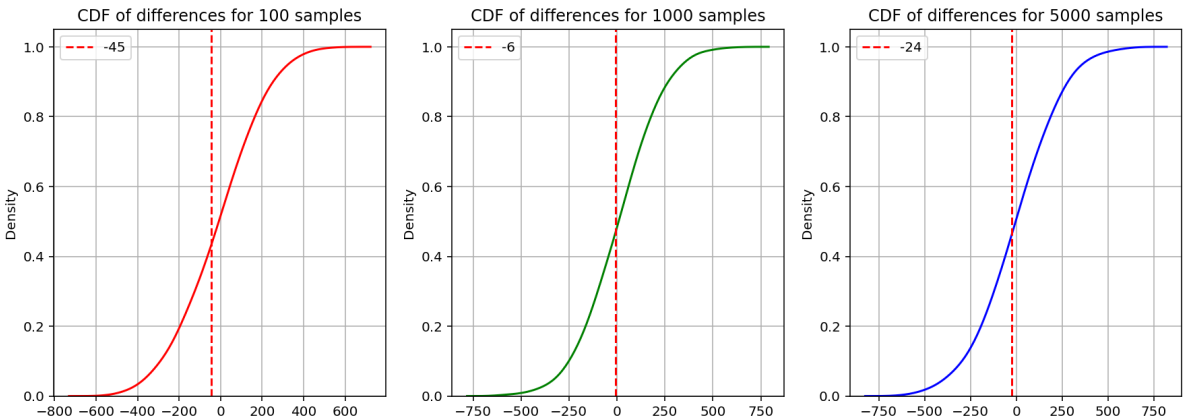
```
**********Two taield test**************
ttest statistic : 0.21160637250397563 | p-value : 0.8324357190771419
Failed to Reject ---> H0: the mean delevery time of "diffrence between start and e
nd time" and "start_scan_to_end_scan" are same.
```

NON-PARAMETRIC(Permutation resampling)

In [ ]:
```
start_scan_to_end_scan=dehivery_data.start_scan_to_end_scan
handling_time_minutes=dehivery_data.handling_time_minutes
sample_sizes = [100,1000,5000]
alpha = 0.05
fig, axs = plt.subplots(1, 3,figsize=(15, 5))
for j, i in enumerate(sample_sizes):
    print("For Sample Size: ", 2*i)
```

```
    sample_startscan=start_scan_to_end_scan[random.sample(range(0, start_scan_to_er
    sample_handlingtime=handling_time_minutes[random.sample(range(0, handling_time_

    diff_in_mean = diff_in_samples(sample_startscan,sample_handlingtime, "starttoer

    #Step 1- Combine b
    #Step 1- Combine both samples of size 50 each to a large sample of size 100 to
    differences = calculate_p_value(sample_startscan,sample_handlingtime,diff_in_me
    plt_cdfplot_withthreshold(j, colrs[j],differences,threshold=diff_in_mean,sample
```

```
For Sample Size:  200
The average starttoendscan 100 starttoendscan = 943.03
The average handllingtime 100 handllingtime= 988.3704937385
The difference between mean of starttoendscan spending and handllingtime spendings
(diff_100)= -45.34049373850007
Percentage of values greater than the difference -45.34049373850007  = 48.9 %
The pValue =  0.489 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis

_____________________________________________________
For Sample Size:  2000
The average starttoendscan 1000 starttoendscan = 955.912
The average handllingtime 1000 handllingtime= 962.10163228195
The difference between mean of starttoendscan spending and handllingtime spendings
(diff_1000)= -6.189632281949912
Percentage of values greater than the difference -6.189632281949912  = 52.1 %
The pValue =  0.521 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis

_____________________________________________________
For Sample Size:  10000
The average starttoendscan 5000 starttoendscan = 961.9574
The average handllingtime 5000 handllingtime= 985.99818417539
The difference between mean of starttoendscan spending and handllingtime spendings
(diff_5000)= -24.040784175389945
Percentage of values greater than the difference -24.040784175389945  = 49.0 %
The pValue =  0.49 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis

_____________________________________________________
```



NON-PARAMETRIC (mannwhitneyu)

In [ ]:
```
#dehivery_data.head().T#start_scan_to_end_scan,handling_time_minutes
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05

start_scan_to_end_scan=dehivery_data.start_scan_to_end_scan
handling_time_minutes=dehivery_data.handling_time_minutes
bs_start_scan_to_end_scan=np.random.choice(start_scan_to_end_scan,size=1000)
bs_handling_time_minutes=np.random.choice(handling_time_minutes,size=1000)
```

```
print("**********Two taield test**************")

null_Hypothesis="""H0: the mean delevery time of "diffrence between start and end t
alternative_Hypothesis="""H0: the mean delevery time of "diffrence between start ar

ttest_statistic,p_value=mannwhitneyu(bs_start_scan_to_end_scan,bs_handling_time_mir
#ttest_statistic,p_value,_=ttest_ind(bs_start_scan_to_end_scan,bs_handling_time_min
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : 519220.0 | p-value : 0.13665481219596842
Failed to Reject ---> H0: the mean delevery time of "diffrence between start and e
nd time" and "start_scan_to_end_scan" are same.
```

diffrence **actual time** vs **segment_actual_time**

In [ ]:
```
#trip_uuid_Group.columns
```

PARAMETRIC TEST-ttest

In [ ]:
```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05

actual_time=trip_uuid_Group.actual_time_max
segment_actual_time=trip_uuid_Group.segment_actual_time_sum
bs_actual_time=np.random.choice(actual_time,size=1000)
bs_segment_actual_time=np.random.choice(segment_actual_time,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery time of actually time and segment_actual_tim
alternative_Hypothesis="HA: the mean delevery time of actually time and segment_act

ttest_statistic,p_value,_=ttest_ind(bs_actual_time,bs_segment_actual_time,alternati
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : -0.18708439866635387 | p-value : 0.8516134550486583
Failed to Reject ---> H0: the mean delevery time of actually time and segment_actu
al_time are same.
```

NON-PARAMETRIC(Permutation resampling)

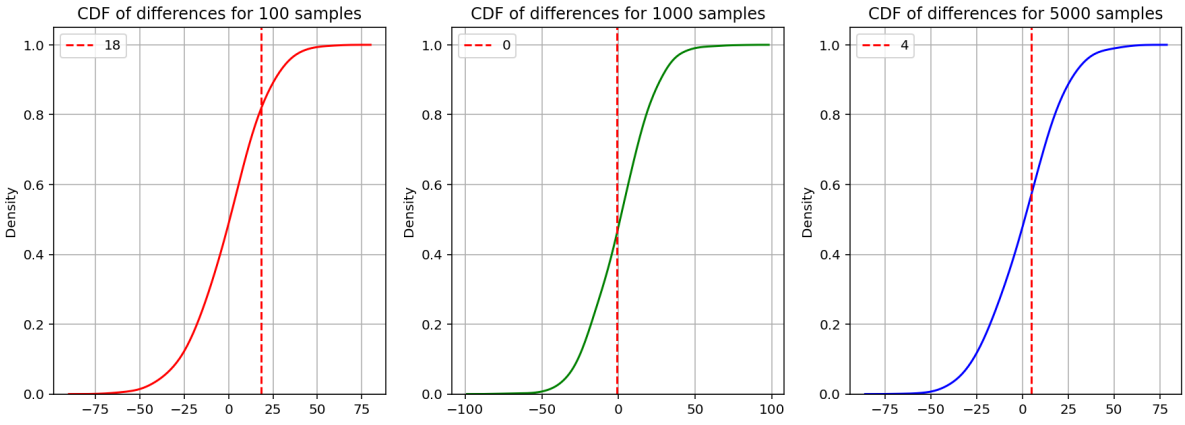
In [ ]:
```
actual_time=trip_uuid_Group.actual_time_max
segment_actual_time=trip_uuid_Group.segment_actual_time_sum

sample_sizes = [100,1000,5000]
alpha = 0.05
fig, axs = plt.subplots(1, 3,figsize=(15, 5))
for j, i in enumerate(sample_sizes):
```

```
    print("For Sample Size: ", 2*i)

    actual_timetaken=actual_time[random.sample(range(0, actual_time.shape[0]), i)]
    segment_actual_timetaken=segment_actual_time[random.sample(range(0, segment_act

    diff_in_mean = diff_in_samples(actual_timetaken,segment_actual_timetaken, "actu

    #Step 1- Combine b
    #Step 1- Combine both samples of size 50 each to a large sample of size 100 to
    differences = calculate_p_value(actual_timetaken,segment_actual_timetaken,diff_
    plt_cdfplot_withthreshold(j, colrs[j],differences,threshold=diff_in_mean,sample
```

```
For Sample Size:  200
The average actual time  100 actual time  = 144.57
The average segmented actaul time 100 segmented actaul time= 126.38
The difference between mean of actual time  spending and segmented actaul time spe
ndings (diff_100)= 18.189999999999998
Percentage of values greater than the difference 18.189999999999998  = 17.7 %
The pValue =  0.177 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
```

```
For Sample Size:  2000
The average actual time  1000 actual time  = 122.285
The average segmented actaul time 1000 segmented actaul time= 123.237
The difference between mean of actual time  spending and segmented actaul time spe
ndings (diff_1000)= -0.9519999999999982
Percentage of values greater than the difference -0.9519999999999982  = 52.1 %
The pValue =  0.521 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
```

```
For Sample Size:  10000
The average actual time  5000 actual time  = 128.4774
The average segmented actaul time 5000 segmented actaul time= 123.6528
The difference between mean of actual time  spending and segmented actaul time spe
ndings (diff_5000)= 4.82459999999999
Percentage of values greater than the difference 4.82459999999999  = 42.6 %
The pValue =  0.426 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
```



NON-PARAMETRIC (mannwhitneyu)

In [ ]:

```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05

actual_time=trip_uuid_Group.actual_time_max
segment_actual_time=trip_uuid_Group.segment_actual_time_sum
bs_actual_time=np.random.choice(actual_time,size=1000)
bs_segment_actual_time=np.random.choice(segment_actual_time,size=1000)
```

```
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery time of actually time and segment_actual_tim
alternative_Hypothesis="HA: the mean delevery time of actually time and segment_act
ttest_statistic,p_value=mannwhitneyu(bs_actual_time,bs_segment_actual_time,alternat

#ttest_statistic,p_value,_=ttest_ind(bs_actual_time,bs_segment_actual_time,alternat
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : 517054.0 | p-value : 0.1864676530409899
Failed to Reject ---> H0: the mean delevery time of actually time and segment_actu
al_time are same.
```

diffrence **osrm_distance** vs **segment_osrm_distance**

In [ ]:
```
trip_uuid_Group.columns
```

Out[ ]:
```
Index(['trip_uuid_', 'source_center_', 'destination_center_',
       'actual_time_max', 'osrm_time_max', 'segment_actual_time_sum',
       'segment_osrm_time_sum', 'actual_distance_to_destination_sum',
       'osrm_distance_max', 'segment_osrm_distance_sum'],
      dtype='object')
```

PARAMETRIC TEST-ttest

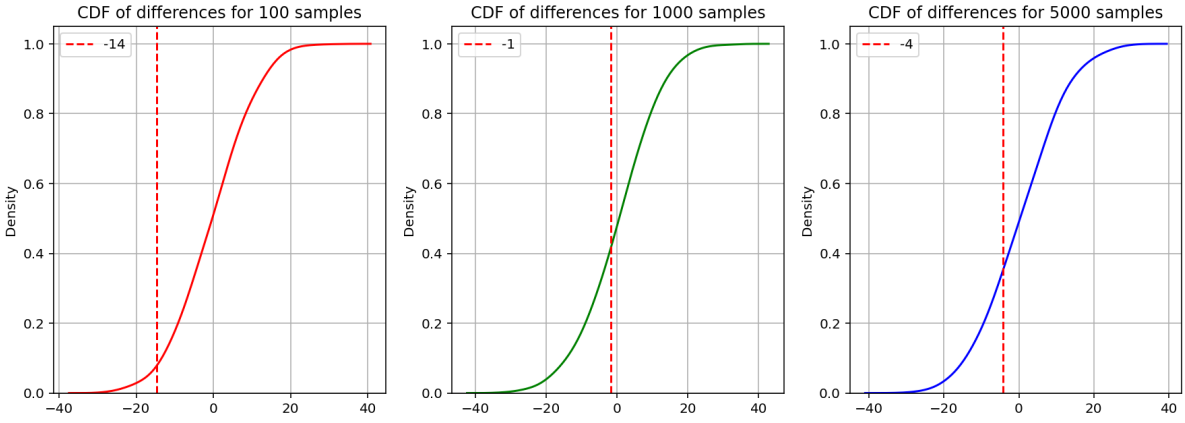
In [ ]:
```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05

osrm_distance=trip_uuid_Group.osrm_distance_max
segment_osrm_distance=trip_uuid_Group.segment_osrm_distance_sum
bs_osrm_distance=np.random.choice(osrm_distance,size=1000)
bs_segment_osrm_distance=np.random.choice(segment_osrm_distance,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery distance of osrm and segment_osrm are same."
alternative_Hypothesis="HA: the mean delevery distance of osrm and segment_osrm are

ttest_statistic,p_value,_=ttest_ind(bs_osrm_distance,bs_segment_osrm_distance,alter
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : -0.9688124685764322 | p-value : 0.33275602304121765
Failed to Reject ---> H0: the mean delevery distance of osrm and segment_osrm are
same.
```

NON-PARAMETRIC(Permutation resampling)

In [ ]:
```
osrm_distance=trip_uuid_Group.osrm_distance_max
segment_osrm_distance=trip_uuid_Group.segment_osrm_distance_sum

sample_sizes = [100,1000,5000]
alpha = 0.05
fig, axs = plt.subplots(1, 3,figsize=(15, 5))
for j, i in enumerate(sample_sizes):
    print("For Sample Size: ", 2*i)

    osrmdist=osrm_distance[random.sample(range(0, osrm_distance.shape[0]), i)]
    segment_osmdist=segment_osrm_distance[random.sample(range(0, segment_osrm_dista

    diff_in_mean = diff_in_samples(osrmdist,segment_osmdist, "osrm_distance ", "seg

    #Step 1- Combine b
    #Step 1- Combine both samples of size 50 each to a large sample of size 100 to
    differences = calculate_p_value(osrmdist,segment_osmdist,diff_in_mean, alpha)
    plt_cdfplot_withthreshold(j, colrs[j],differences,threshold=diff_in_mean,sample
```

```
For Sample Size:  200
The average osrm_distance  100 osrm_distance  = 52.59476250000001
The average segment_osrm_distance 100 segment_osrm_distance= 67.25085874999999
The difference between mean of osrm_distance  spending and segment_osrm_distance s
pendings (diff_100)= -14.656096249999983
Percentage of values greater than the difference -14.656096249999983  = 49.8 %
The pValue =  0.498 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
_____________________________________________________
For Sample Size:  2000
The average osrm_distance  1000 osrm_distance  = 65.906309825
The average segment_osrm_distance 1000 segment_osrm_distance= 67.676357075
The difference between mean of osrm_distance  spending and segment_osrm_distance s
pendings (diff_1000)= -1.7700472500000046
Percentage of values greater than the difference -1.7700472500000046  = 52.3000000
00000004 %
The pValue =  0.523 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
_____________________________________________________
For Sample Size:  10000
The average osrm_distance  5000 osrm_distance  = 64.75618959
The average segment_osrm_distance 5000 segment_osrm_distance= 69.02574505749999
The difference between mean of osrm_distance  spending and segment_osrm_distance s
pendings (diff_5000)= -4.269555467499984
Percentage of values greater than the difference -4.269555467499984  = 50.4 %
The pValue =  0.504 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
_____________________________________________________
```



NON-PARAMETRIC (mannwhitneyu)

In [ ]:
```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05

osrm_distance=trip_uuid_Group.osrm_distance_max
segment_osrm_distance=trip_uuid_Group.segment_osrm_distance_sum
bs_osrm_distance=np.random.choice(osrm_distance,size=1000)
bs_segment_osrm_distance=np.random.choice(segment_osrm_distance,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery distance of osrm and segment_osrm are same."
alternative_Hypothesis="HA: the mean delevery distance of osrm and segment_osrm are

#ttest_statistic,p_value,_=ttest_ind(bs_osrm_distance,bs_segment_osrm_distance,alte
ttest_statistic,p_value=mannwhitneyu(bs_osrm_distance,bs_segment_osrm_distance,alte

print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : 498469.0 | p-value : 0.905639073664004
Failed to Reject ---> H0: the mean delevery distance of osrm and segment_osrm are
same.
```

In [ ]:
```

```

osrm time aggregated value and segment osrm time aggregated value

PARAMETRIC TEST-ttest

In [ ]:
```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05
#osrm_time_max', 'segment_actual_time_sum',
osrm_time=trip_uuid_Group.osrm_time_max
segment_osrm_time=trip_uuid_Group.segment_osrm_time_sum
bs_osrm_time=np.random.choice(osrm_time,size=1000)
bs_segment_osrm_time=np.random.choice(segment_osrm_time,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery time of osrm_time and segment_osrm_time are
alternative_Hypothesis="HA: the mean delevery time of osrm_time and segment_osrm_ti

ttest_statistic,p_value,_=ttest_ind(bs_osrm_time,bs_segment_osrm_time,alternative='
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : -0.8965034332740817 | p-value : 0.3700919489967347
Failed to Reject ---> H0: the mean delevery time of osrm_time and segment_osrm_tim
e are same.
```

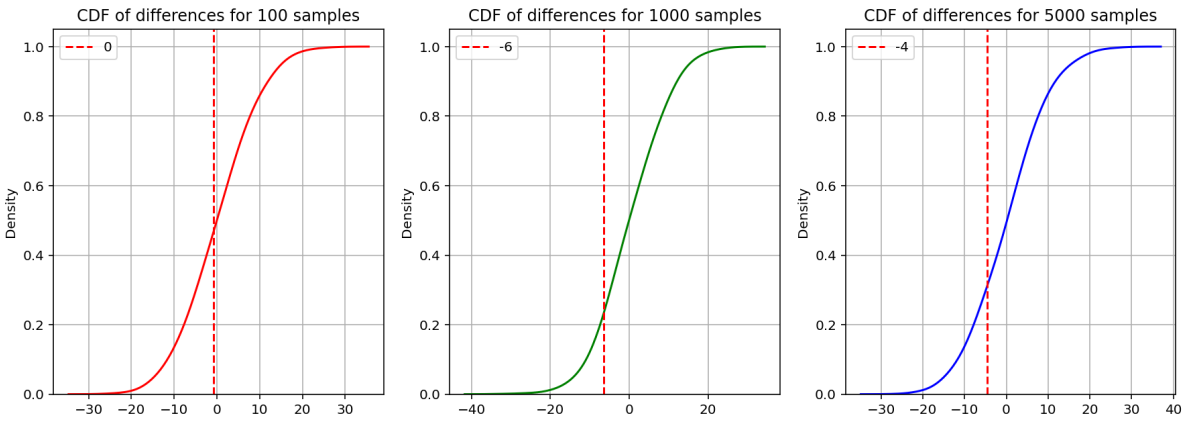
NON-PARAMETRIC(Permutation resampling)

In [ ]:
```
osrm_time=trip_uuid_Group.osrm_time_max
segment_osrm_time=trip_uuid_Group.segment_osrm_time_sum

sample_sizes = [100,1000,5000]
alpha = 0.05
fig, axs = plt.subplots(1, 3,figsize=(15, 5))
for j, i in enumerate(sample_sizes):
    print("For Sample Size: ", 2*i)

    sample_osrmTime=osrm_time[random.sample(range(0, osrm_time.shape[0]), i)]
    sample_segment_osrmTime=segment_osrm_time[random.sample(range(0, segment_osrm_t

    diff_in_mean = diff_in_samples(sample_osrmTime,sample_segment_osrmTime, "osrm_t

    #Step 1- Combine b
    #Step 1- Combine both samples of size 50 each to a large sample of size 100 to
    differences = calculate_p_value(sample_osrmTime,sample_segment_osrmTime,diff_ir
    plt_cdfplot_withthreshold(j, colrs[j],differences,threshold=diff_in_mean,sample
```

```
For Sample Size:  200
The average osrm_time  100 osrm_time  = 61.95
The average segment_osrm_time 100 segment_osrm_time= 62.68
The difference between mean of osrm_time  spending and segment_osrm_time spendings
(diff_100)= -0.7299999999999969
Percentage of values greater than the difference -0.7299999999999969  = 49.9 %
The pValue =  0.499 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
```

```
For Sample Size:  2000
The average osrm_time  1000 osrm_time  = 54.095
The average segment_osrm_time 1000 segment_osrm_time= 60.653
The difference between mean of osrm_time  spending and segment_osrm_time spendings
(diff_1000)= -6.558
Percentage of values greater than the difference -6.558  = 49.7 %
The pValue =  0.497 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
```

```
For Sample Size:  10000
The average osrm_time  5000 osrm_time  = 55.3872
The average segment_osrm_time 5000 segment_osrm_time= 59.9334
The difference between mean of osrm_time  spending and segment_osrm_time spendings
(diff_5000)= -4.546199999999999
Percentage of values greater than the difference -4.546199999999999  = 50.9 %
The pValue =  0.509 and the significance P(Reject H0 when H0 is true)= 0.05
We fail to reject the null hypothesis
```



NON-PARAMETRIC (mannwhitneyu)

In [ ]:
```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05
#osrm_time_max', 'segment_actual_time_sum',
osrm_time=trip_uuid_Group.osrm_time_max
segment_osrm_time=trip_uuid_Group.segment_osrm_time_sum
bs_osrm_time=np.random.choice(osrm_time,size=1000)
bs_segment_osrm_time=np.random.choice(segment_osrm_time,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery time of osrm_time and segment_osrm_time are
alternative_Hypothesis="HA: the mean delevery time of osrm_time and segment_osrm_ti

#ttest_statistic,p_value,_=ttest_ind(bs_osrm_time,bs_segment_osrm_time,alternative=
ttest_statistic,p_value=mannwhitneyu(bs_osrm_time,bs_segment_osrm_time,alternative=

print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : 480427.5 | p-value : 0.12951936394761165
Failed to Reject ---> H0: the mean delevery time of osrm_time and segment_osrm_tim
e are same.
```

osm distance vs actual distance

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

Out[36]:
```
Index(['trip_uuid_', 'source_center_', 'destination_center_',
       'actual_time_max', 'osrm_time_max', 'segment_actual_time_sum',
       'segment_osrm_time_sum', 'actual_distance_to_destination_sum',
       'osrm_distance_max', 'segment_osrm_distance_sum', 'route_type_max'],
      dtype='object')
```

PARAMETRIC TEST-ttest

In [41]:
```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import norm
significance_value=0.05

actual_distance=trip_uuid_Group.actual_distance_to_destination_sum
osrm_distance=trip_uuid_Group.osrm_distance_max
bs_actual_distance=np.random.choice(actual_distance,size=1000)
bs_osrm_distance=np.random.choice(osrm_distance,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery time of actual  distance  and  osrm distance
alternative_Hypothesis="HA: the mean delevery time of actual  distance  and  osrm d

ttest_statistic,p_value,_=ttest_ind(bs_actual_distance,bs_osrm_distance,alternative
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
```

```
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

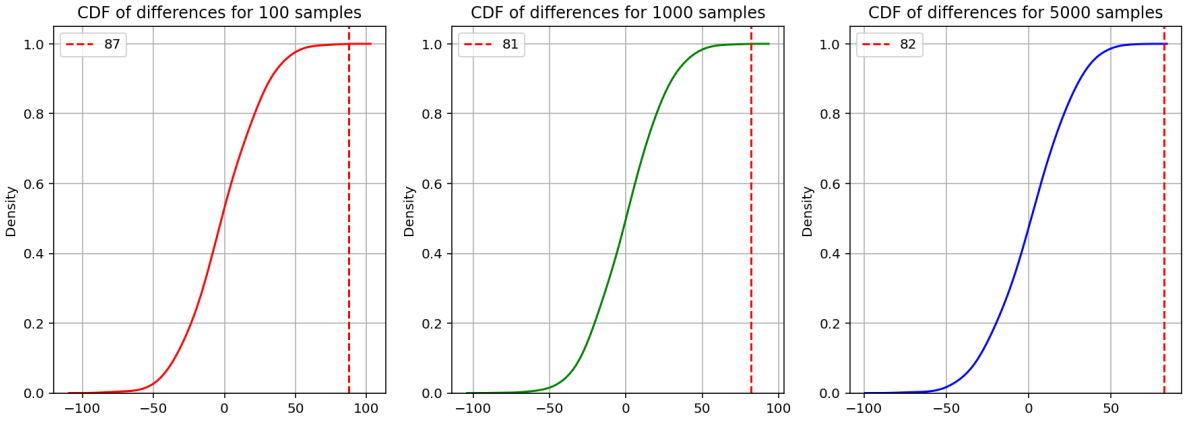
```
**********Two taield test**************
ttest statistic : 16.42992471789107 | p-value : 5.378619248707328e-57
Reject ---> H0: the mean delevery time of actual  distance  and  osrm distance are
same.
Accept ---> HA: the mean delevery time of actual  distance  and  osrm distance are
Not same.
```

NON-PARAMETRIC(Permutation resampling)

In [45]:

```
actual_distance=trip_uuid_Group.actual_distance_to_destination_sum
osrm_distance=trip_uuid_Group.osrm_distance_max
colrs = ['r','g','b','y', 'c', 'm', 'k']
sample_sizes = [100,1000,5000]
alpha = 0.05
fig, axs = plt.subplots(1, 3,figsize=(15, 5))
for j, i in enumerate(sample_sizes):
    print("For Sample Size: ", 2*i)

    sample_actual_distance=actual_distance[random.sample(range(0, actual_distance.s
    sample_osrm_distance=osrm_distance[random.sample(range(0, osrm_distance.shape[0

    diff_in_mean = diff_in_samples(sample_actual_distance,sample_osrm_distance, "ac

    #Step 1- Combine b
    #Step 1- Combine both samples of size 50 each to a large sample of size 100 to
    differences = calculate_p_value(sample_actual_distance,sample_osrm_distance,dif
    plt_cdfplot_withthreshold(j, colrs[j],differences,threshold=diff_in_mean,sample
```

```
For Sample Size:  200
The average actual_distance  100 actual_distance  = 166.14975039386886
The average osrm_distance 100 osrm_distance= 78.4269275
The difference between mean of actual_distance  spending and osrm_distance spendin
gs (diff_100)= 87.72282289386885
Percentage of values greater than the difference 87.72282289386885  = 0.0 %
The pValue =  0.0 and the significance P(Reject H0 when H0 is true)= 0.05
We can reject the null hypothesis
_________________________________________________
For Sample Size:  2000
The average actual_distance  1000 actual_distance  = 147.14342030518205
The average osrm_distance 1000 osrm_distance= 65.516221975
The difference between mean of actual_distance  spending and osrm_distance spendin
gs (diff_1000)= 81.62719833018205
Percentage of values greater than the difference 81.62719833018205  = 0.0 %
The pValue =  0.0 and the significance P(Reject H0 when H0 is true)= 0.05
We can reject the null hypothesis
_________________________________________________
For Sample Size:  10000
The average actual_distance  5000 actual_distance  = 147.6731690251908
The average osrm_distance 5000 osrm_distance= 65.56547343999999
The difference between mean of actual_distance  spending and osrm_distance spendin
gs (diff_5000)= 82.1076955851908
Percentage of values greater than the difference 82.1076955851908  = 0.0 %
The pValue =  0.0 and the significance P(Reject H0 when H0 is true)= 0.05
We can reject the null hypothesis
_________________________________________________
```



NON-PARAMETRIC (mannwhitneyu)

In [40]:
```
from statsmodels.stats.weightstats import ztest,ttest_ind
from scipy.stats import mannwhitneyu
from scipy.stats import norm
significance_value=0.05

actual_distance=trip_uuid_Group.actual_distance_to_destination_sum
osrm_distance=trip_uuid_Group.osrm_distance_max
bs_actual_distance=np.random.choice(actual_distance,size=1000)
bs_osrm_distance=np.random.choice(osrm_distance,size=1000)
print("**********Two taield test**************")

null_Hypothesis="H0: the mean delevery time of actual  distance  and  osrm distance
alternative_Hypothesis="HA: the mean delevery time of actual  distance  and  osrm d

#ttest_statistic,p_value,_=ttest_ind(bs_actual_distance,bs_osrm_distance,alternativ
ttest_statistic,p_value=mannwhitneyu(bs_actual_distance,bs_osrm_distance,alternativ
print(f"ttest statistic : {ttest_statistic} | p-value : {p_value}")
if p_value>significance_value:
  print(f"Failed to Reject ---> {null_Hypothesis}")
else:
  print(f"Reject ---> {null_Hypothesis}")
  print(f"Accept ---> {alternative_Hypothesis}")
```

```
**********Two taield test**************
ttest statistic : 637693.0 | p-value : 1.4569661826314825e-26
Reject ---> H0: the mean delevery time of actual  distance  and  osrm distance are
same.
Accept ---> HA: the mean delevery time of actual  distance  and  osrm distance are
Not same.
```

In [30]:
```
```

Insite vizualizations

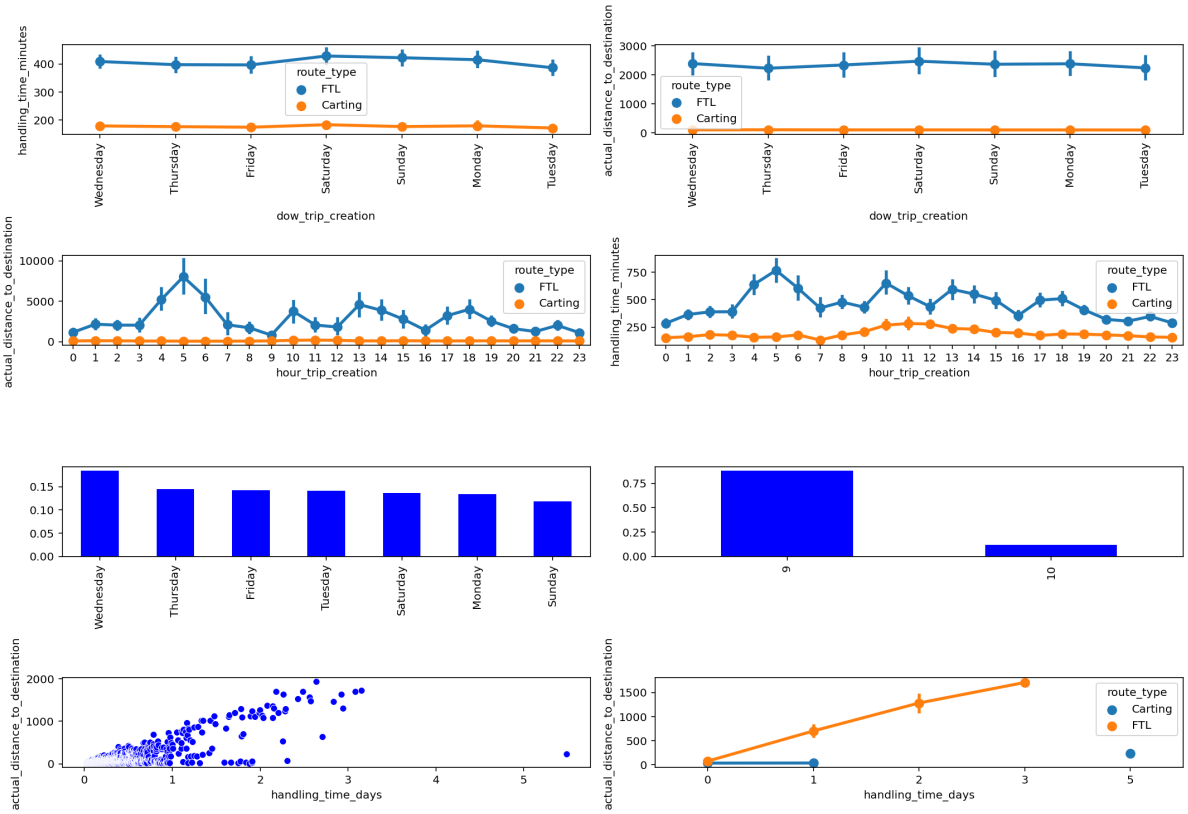
In [31]:
```
dehivery_data["route_type"]=dehivery_data["route_type"].astype("object")
sampleColumns=dehivery_data[["hour_trip_creation","month_trip_creation","dow_trip_c
Tripdata=sampleColumns.groupby(['trip_uuid', 'source_center',"destination_center"])
```

In [31]:
```
```

In [31]:
```
```

In [32]:
```
source_dest=dehivery_data[["source_city","destination_city","handling_time_days","a
source_dest["sourcedest"]=source_dest["source_city"]+" - "+source_dest["destinatior
```

```
fig, axs = plt.subplots(4, 2,figsize=(15, 10))
sns.pointplot(x="dow_trip_creation",y="handling_time_minutes",data=Tripdata,hue="ro
axs[0,0].tick_params(axis='x', rotation=90)

sns.pointplot(x="dow_trip_creation",y="actual_distance_to_destination",data=Tripdat
axs[0,1].tick_params(axis='x', rotation=90)

sns.pointplot(x="hour_trip_creation",y="actual_distance_to_destination",data=Tripda
sns.pointplot(x="hour_trip_creation",y="handling_time_minutes",data=Tripdata,hue="r
Tripdata["dow_trip_creation"].value_counts(normalize=True).plot(kind="bar",color="b
Tripdata["month_trip_creation"].value_counts(normalize=True).plot(kind="bar",color=
sns.scatterplot(x="handling_time_days",y="actual_distance_to_destination",data=sour
source_dest["handling_time_days"]=source_dest["handling_time_days"].map(int)
sns.pointplot(x="handling_time_days",y="actual_distance_to_destination",data=source
fig.tight_layout()
plt.show()
```



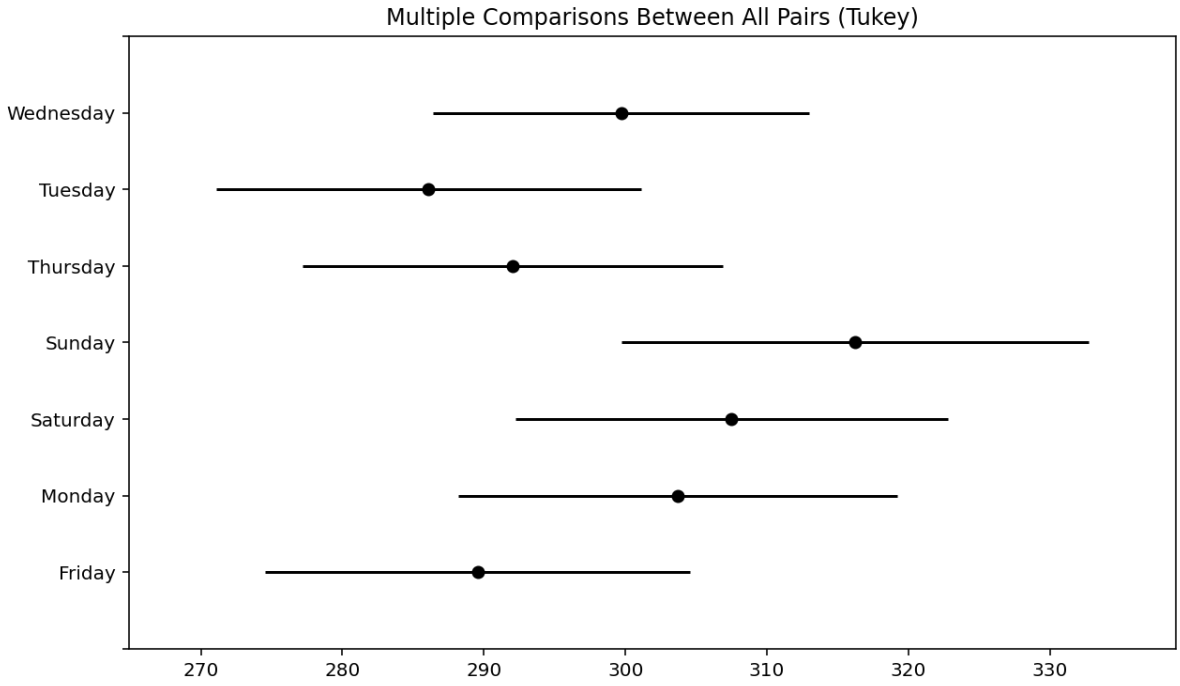
- total distance covered by FTL trucks are larger than the Carting.
- handling time on each day in a week are same
- Longer distance delhiveries are stating at 3-6 AM might be because of less traffic
- This tell us that FTL truck trips cover a lot of distance and they can deliver higher no of shipments

- This also tells us that FTL trucks takes more time to deliver than Carting and this is because Carting is used for shorter distance shipping.

Pairwise test to check if handling time on each day in a week or same or not

In [33]:
```
tukey_result = pairwise_tukeyhsd(Tripdata["handling_time_minutes"], Tripdata["dow_t
print(f"tukey_result -- {tukey_result}")
print("*"*100)
tukey_plot = tukey_result.plot_simultaneous()
```

```
tukey_result --    Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
 group1    group2  meandiff p-adj   lower    upper  reject
-------------------------------------------------------------
  Friday    Monday  14.1382 0.7961 -16.3729 44.6494  False
  Friday  Saturday  17.9498 0.5731 -12.3714 48.2709  False
  Friday    Sunday  26.6509 0.1617  -4.8676 58.1695  False
  Friday  Thursday   2.4701    0.9 -27.4353 32.3756  False
  Friday   Tuesday  -3.4683    0.9  -33.526 26.5894  False
  Friday Wednesday  10.1355    0.9 -18.1439 38.4148  False
  Monday  Saturday   3.8116    0.9 -26.9612 34.5843  False
  Monday    Sunday  12.5127    0.9 -19.4405 44.4659  False
  Monday  Thursday -11.6681    0.9 -42.0313 18.6951  False
  Monday   Tuesday -17.6065 0.5991 -48.1197 12.9066  False
  Monday Wednesday  -4.0028    0.9 -32.7658 24.7602  False
Saturday    Sunday   8.7011    0.9 -23.0706 40.4729  False
Saturday  Thursday -15.4796   0.71 -45.6519 14.6926  False
Saturday   Tuesday -21.4181 0.3639 -51.7412  8.9051  False
Saturday Wednesday  -7.8143    0.9 -36.3757  20.747  False
  Sunday  Thursday -24.1808 0.2574 -55.5561  7.1945  False
  Sunday   Tuesday -30.1192 0.0724 -61.6397  1.4012  False
  Sunday Wednesday -16.5155 0.6396 -46.3449  13.314  False
Thursday   Tuesday  -5.9385    0.9 -35.8459  23.969  False
Thursday Wednesday   7.6653    0.9 -20.4543  35.785  False
 Tuesday Wednesday  13.6038 0.7655 -14.6777 41.8853  False
-------------------------------------------------------------
****************************************************************************************
******************
```



Mean amount of time takes to complete a trip on each day in a week is same.

In [ ]:
```
```

Insights

Time:

- There is difference between **OSRM time and actual trip time.**
- Difference between **start and end scan times and Handling time** means are same which implies that there is no difference between these two times.
- There is no difference between **actual time(Cumulative) and segment actual time**.
- There is no difference between **OSRM time and segment OSRM time.**

Distance:

- There is difference between **OSRM distance and actual distance**.
- There is no difference between **actual distance(Cumulative) and segment actual distance**.

Others:

- The mean time to **complete a trip on any day in a week are same**.
- From above plot it is visible that as the **distance increases the no of days/hours to complete a trip** will also increase and this is true with both carting and FTL mode of shipments.
- The amount of distance covered by FTL vehicles are much higher than the Carting.
- The time taken to complete a trip by the **Carting vehicle** is much lower than the FTL vehicle and this is mainly because FTL trucks are used for long distance trips and **Carting is used for shorter distance trips**.
- The Longer distance deliveries are planing to covered by **the FTL/carting vehicles are much higher between 2am to 7pm** than any other slot in a day but the vehicles can cover larger distance post noon because of less traffic i.e 1am to 5pm

Recommednations

- The mean amount of **actual distance** and **OSRM distance** for a trip are not same which means that OSRM engine is not an Good map engine in predicting overall distance to the destination. Hence **Delihvery** should try to see if they can **improve/replace** OSRM engine to predict the distance accurately like google map,openstreetmap....etc
  - google-maps-vs-osrm
  - choosing-a-map-amapbox-google-maps-openstreetmap
- The mean amount of **actual time** and **osrm time** for a trip are not same which means that the prediction made by the osrm engine for delivery time are not reliable always and delihvery should focus on improving the accuracy of this machine or identifying if any other engine can improve their prediction times accuracy so that user experience can be improved.

- From above plots it is visible that Delihvery is using the **full truck loads** for longer distances and **carting** for shorter distances.it is also understood from data that the no of deliveries done by FTL are more than carting. In this regard, **Delihvery** is already doing good and to **increase no of deliveries in shorter time Delihvery should start adding more FTL trucks for shipment.**
- The Longer distance deliveries are planing to covered by the FTL/carting vehicles are much higher between **2am to 7pm** than any other slot in a day then they can deliver faster and traffic alo would be lesss on this time
- The mean amount of time taken to **complete a trip on any day in a week** is almost same(Here we just comparing handling time vs day on which trip is created) but slight saturday and sunday trip creation are bit faster
- We have **only 2 months data and taking actions on studying** the small subset of the entire business unit is bit risky and this may lead to wrong conclusions as well. Hence we may need data that will spread across various seasons in a year to study and take actions.

In [ ]: