```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as ss
import re
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy.stats import ttest_rel
import statistics
```

```
palette = sns.color_palette("deep")
sns.set_palette(palette)
```

```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181
To: /content/delhivery_data.csv?1642751181
100% 55.6M/55.6M [00:00<00:00, 76.8MB/s]
```

```
df = pd.read_csv('delhivery_data.csv?1642751181')
```

Basic Data Cleaning and exploration

```
df.columns
```

```
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')
```

```
df.shape
```

```
(144867, 24)
```

```
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
```

```
df.head()
```

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

5 rows × 24 columns

```
{i : df[i].nunique() for i in df.columns}
```

```
{'data': 2,
 'trip_creation_time': 14817,
 'route_schedule_uuid': 1504,
 'route_type': 2,
 'trip_uuid': 14817,
 'source_center': 1508,
 'source_name': 1498,
 'destination_center': 1481,
 'destination_name': 1468,
 'od_start_time': 26369,
 'od_end_time': 26369,
 'start_scan_to_end_scan': 1915,
 'is_cutoff': 2,
 'cutoff_factor': 501,
 'cutoff_timestamp': 93180,
 'actual_distance_to_destination': 144515,
 'actual_time': 3182,
 'osrm_time': 1531,
 'osrm_distance': 138046,
 'factor': 45641,
 'segment_actual_time': 747,
 'segment_osrm_time': 214,
 'segment_osrm_distance': 113799,
 'segment_factor': 5675}
```

```
df.describe()
```

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	factor	segment_ac
count	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144
mean	961.262986	232.926567	234.073372	416.927527	213.868272	284.771297	2.120107	
std	1037.012769	344.755577	344.990009	598.103621	308.011085	421.119294	1.715421	
min	20.000000	9.000000	9.000045	9.000000	6.000000	9.008200	0.144000	-2
25%	161.000000	22.000000	23.355874	51.000000	27.000000	29.914700	1.604264	
50%	449.000000	66.000000	66.126571	132.000000	64.000000	78.525800	1.857143	
75%	1634.000000	286.000000	286.708875	513.000000	257.000000	343.193250	2.213483	
max	7898.000000	1927.000000	1927.447705	4532.000000	1686.000000	2326.199100	77.387097	3(

```
df.describe(include = object)
```

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

Handling Null Values

```
a1 = df[pd.isna(df['source_name'])]['source_center'].unique()
```

```
b1 = pd.DataFrame((df[df['source_center'].isin(a1)][['source_center','source_name']]).groupby('source_center').agg(lambda x: list(set(x)))).sort_in
b1
```

	source_center	source_name
0	IND126116AAA	[nan]
1	IND282002AAD	[nan]
2	IND331022A1B	[nan]
3	IND342902A1B	[nan]
4	IND465333A1B	[nan]
5	IND505326AAB	[nan]
6	IND509103AAC	[nan]
7	IND577116AAA	[nan]
8	IND841301AAC	[nan]
9	IND852118A1B	[nan]

Next steps: Generate code with b1 View recommended plots

```
a2 = df[pd.isna(df['destination_name'])]['destination_center'].unique()
```

```
b2 = pd.DataFrame((df[df['destination_center'].isin(a2)][['destination_center','destination_name']]).groupby('destination_center').agg(lambda x: li
b2
```

	destination_center	destination_name
0	IND122015AAC	[nan]
1	IND126116AAA	[nan]
2	IND221005A1A	[nan]
3	IND250002AAC	[nan]
4	IND282002AAD	[nan]
5	IND331001A1C	[nan]
6	IND342902A1B	[nan]
7	IND465333A1B	[nan]
8	IND505326AAB	[nan]
9	IND509103AAC	[nan]
10	IND577116AAA	[nan]
11	IND841301AAC	[nan]
12	IND852118A1B	[nan]

Next steps: Generate code with b2 View recommended plots

```
destination_list = (df[df['destination_name'].notna()][['destination_center','destination_name']]).groupby(['destination_center']).agg({'destinatio
source_list = (df[df['source_name'].notna()][['source_center','source_name']]).groupby(['source_center']).agg({'source_name' : 'first'})
```

```
([b1.isin(destination_list.iloc[:,0])],[b2.isin(source_list.iloc[:,0])])
```

```
([   source_center  source_name
  0          False        False
  1          False        False
  2          False        False
  3          False        False
  4          False        False
  5          False        False
  6          False        False
  7          False        False
  8          False        False
  9          False        False],
 [    destination_center  destination_name
  0               False             False
  1               False             False
  2               False             False
  3               False             False
  4               False             False
  5               False             False
  6               False             False
  7               False             False
  8               False             False
  9               False             False
  10              False             False
  11              False             False
  12              False             False])
```

```
df[df['source_name'].isna() & df['destination_center'].isna()].shape
```

```
(0, 24)
```

As above analysis on the Null values in Source and Destination Centers and Name, it shows that there are 10 and 13 missing Names respectively and all are different in both the groups. In total for 23 Centers the Name values are missing. As there are no missing values for both Destination name and Source Name. So replacing all the missing values with the 'Unknown' values.

```
df.fillna('Unknown',inplace = True)
```

```
(df.isnull().sum())
```

```
data                            0
trip_creation_time              0
route_schedule_uuid             0
route_type                      0
trip_uuid                       0
source_center                   0
source_name                     0
destination_center              0
destination_name                0
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
```

```
#scan to whether convert the dtype of some columns to int or float. if the shape of output is (0,) then it is integer otherwise float attribute.
columns_to_check = ['start_scan_to_end_scan', 'actual_distance_to_destination', 'actual_time',
       'osrm_time', 'osrm_distance', 'factor', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'segment_factor']

colums_division = {i: (df[df[i] % 1 != 0][i].unique().shape) for i in columns_to_check}

print(colums_division)
#this proves that there are no float values in the attribute, the dtype can be changed to int.
```

```
{'start_scan_to_end_scan': (0,), 'actual_distance_to_destination': (144515,), 'actual_time': (0,), 'osrm_time': (0,), 'osrm_distance': (138031,),
```

10 clumns are assigned float as datatype but only 5 require float as datatype. Converting appropriate data to int and float(2 digit rounding).

```
#to convert data from given datatytpe to required datatype
# 1. Seggregation of columns 2. Assigning the required datatypes
datetime_columns = ['trip_creation_time','od_start_time', 'od_end_time','cutoff_timestamp']
float_columns = []
int_columns = []
for i,j in colums_division.items():
  if j == (0,):
    int_columns.append(i)
  else: float_columns.append(i)
```

```
#Converting Data in appropriate datatype

for i in datetime_columns:
  df[i] = pd.to_datetime(df[i],format = 'ISO8601')

for i in int_columns:
  df[i] = df[i].astype('int8')

for i in float_columns:
  df[i] = df[i].round(2)
```

```
pd.DataFrame(df['cutoff_factor'] < df['factor']).value_counts()
```

```
False    144616
True        251
Name: count, dtype: int64
```

```
pd.DataFrame(df['segment_factor'] < df['factor']).value_counts()
```

```
True     72568
False    72299
Name: count, dtype: int64
```

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

```
is_cutoff
True     118749
False     26118
Name: count, dtype: int64
```

As there is no apparant raltionship between any column to column name containing 'cutoff' as part, so the columns will be dropped from the database for analysis.

```
df.describe()
```

	trip_creation_time	od_start_time	od_end_time	start_scan_to_end_scan	cutoff_factor	cutoff_timestamp	actual_distance_to_desti
count	144867	144867	144867	144867.000000	144867.000000	144867	144867.0
mean	2018-09-22 13:34:23.659819264	2018-09-22 18:02:45.855230720	2018-09-23 10:04:31.395393024	4.180973	232.926567	2018-09-23 02:32:29.393999616	234.0
min	2018-09-12 00:00:16.535741	2018-09-12 00:00:16.535741	2018-09-12 00:50:10.814399	-128.000000	9.000000	2018-09-12 00:02:09.740725	9.0
25%	2018-09-17 03:20:51.775845888	2018-09-17 08:05:40.886155008	2018-09-18 01:48:06.410121984	-68.000000	22.000000	2018-09-17 18:59:37	23.3
50%	2018-09-22 04:24:27.932764928	2018-09-22 08:53:00.116656128	2018-09-23 03:13:03.520212992	8.000000	66.000000	2018-09-22 20:31:21	66.1
75%	2018-09-27 17:57:56.350054912	2018-09-27 22:41:50.285857024	2018-09-28 12:49:06.054018048	76.000000	286.000000	2018-09-28 05:14:48.500000	286.7
max	2018-10-03 23:59:42.701692	2018-10-06 04:27:23.392375	2018-10-08 03:00:24.353479	127.000000	1927.000000	2018-10-06 23:44:12	1927.4
std	NaN	NaN	NaN	78.408839	344.755577	NaN	344.9

```
((df['factor'] == (df['actual_time']/df['osrm_time'])).value_counts(),(df['segment_factor'] == (df['segment_actual_time']/df['segment_osrm_time'])).value_counts())
```

```
(False    131048
 True      13819
 Name: count, dtype: int64,
 False    104039
 True      40828
 Name: count, dtype: int64)
```

```
columns_to_drop = [col for col in df.columns if 'cutoff' in col]
df = df.drop(columns=columns_to_drop)
columns_to_drop = [col for col in df.columns if 'factor' in col]
df = df.drop(columns=columns_to_drop)
```

```
df_check1 = pd.DataFrame(df.groupby('trip_uuid').agg({'route_schedule_uuid' : 'nunique'}))
df_check1['route_schedule_uuid'].value_counts()
```

```
route_schedule_uuid
1    14817
Name: count, dtype: int64
```

```
df_check1 = pd.DataFrame(df.groupby('route_schedule_uuid').agg({'trip_uuid' : 'nunique'}))
df_check1['trip_uuid'].value_counts()
```

```
trip_uuid
1     163
2     111
4      91
3      90
12     82
6      76
8      73
7      67
10     64
9      62
16     62
15     59
13     58
14     55
11     54
5      53
17     51
18     48
19     44
21     40
20     37
22     30
23      5
```

```
24      5
25      4
32      3
30      2
33      2
26      2
39      1
37      1
40      1
46      1
53      1
35      1
27      1
36      1
43      1
41      1
34      1
Name: count, dtype: int64
```

Trip_uuid and route_schedule_uuid are different from each other and both are INdependent.

```
(df['osrm_distance'] == df['segment_osrm_distance']).value_counts()
```

```
False    118498
True      26369
Name: count, dtype: int64
```

osrm_distance and segment_osrm_distnce are both independent.

```
df.describe()
```

	trip_creation_time	od_start_time	od_end_time	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_tim
count	144867	144867	144867	144867.000000	144867.000000	144867.000000	144867.00000
mean	2018-09-22 13:34:23.659819264	2018-09-22 18:02:45.855230720	2018-09-23 10:04:31.395393024	4.180973	234.073367	18.994526	20.10687
min	2018-09-12 00:00:16.535741	2018-09-12 00:00:16.535741	2018-09-12 00:50:10.814399	-128.000000	9.000000	-128.000000	-128.00000
25%	2018-09-17 03:20:51.775845888	2018-09-17 08:05:40.886155008	2018-09-18 01:48:06.410121984	-68.000000	23.360000	-28.000000	9.00000
50%	2018-09-22 04:24:27.932764928	2018-09-22 08:53:00.116656128	2018-09-23 03:13:03.520212992	8.000000	66.130000	34.000000	26.00000
75%	2018-09-27 17:57:56.350054912	2018-09-27 22:41:50.285857024	2018-09-28 12:49:06.054018048	76.000000	286.710000	68.000000	55.00000
max	2018-10-03 23:59:42.701692	2018-10-06 04:27:23.392375	2018-10-08 03:00:24.353479	127.000000	1927.450000	127.000000	127.00000
std	NaN	NaN	NaN	78.408839	344.990002	68.223166	58.03479

```
df.describe(include = 'object')
```

	data	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name
count	144867	144867	144867	144867	144867	144867	144867	144867
unique	2	1504	2	14817	1508	1499	1481	1469
top	training	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	trip-153811219535896559	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)

```
df.columns
```

```
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'],
      dtype='object')
```

```
df['segment_key'] = df['trip_uuid'] + '_' + df['source_center'] + '_' + df['destination_center']
df['segment_actual_time_sum'] = df.groupby('segment_key')['segment_actual_time'].cumsum()
df['segment_osrm_distance_sum'] = df.groupby('segment_key')['segment_osrm_distance'].cumsum()
df['segment_osrm_time_sum'] = df.groupby('segment_key')['segment_osrm_time'].cumsum()
```

```
df.columns
```

```
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', 'segment_key',
       'segment_actual_time_sum', 'segment_osrm_distance_sum',
       'segment_osrm_time_sum'],
      dtype='object')
```

```
df['actual_time']
```

```
0          14
1          24
2          40
3          62
4          68
         ...
144862     94
144863    120
144864   -116
144865    -98
144866    -86
Name: actual_time, Length: 144867, dtype: int8
```

```
create_segment_dict = {'data' : 'first', 'trip_creation_time' : 'first','trip_uuid' : 'first', 'route_schedule_uuid' : 'first', 'route_type' : 'fir
       'source_name' : 'first', 'destination_name' : 'last', 'od_start_time' : 'first', 'od_end_time' : 'last',
       'start_scan_to_end_scan' : 'first', 'actual_distance_to_destination' : 'sum',
       'actual_time' : 'sum', 'osrm_time' : 'sum', 'osrm_distance' : 'sum', 'segment_actual_time_sum' : 'last', 'segment_osrm_distance_sum': 'last'
       'segment_osrm_time_sum' : 'last'}
```

```
df1 = df.groupby('segment_key').agg(create_segment_dict).reset_index().sort_values(['segment_key','od_end_time'])
```

Feature Engineering

```
#OD_Time_Difference Feature Creation
df1['od_time_diff'] = (df1['od_end_time']-df1['od_start_time']).apply(lambda x: (np.ceil(x.total_seconds() / 60)))
df1['od_time_diff'] = df1['od_time_diff'].astype('int')
```

```
#Using Regex extracting the text info from the Source and Destination Name.
import re
def center_split(text):
  pattern = r'^(\w+)_(\w+)_([\w\s]+)? \(([\w\s]+)\)$'
  match = re.match(pattern, text)

  if match:
    a = match.group(1)
    b = match.group(2)
    c = match.group(3)
    d = match.group(4)
  else:
    a = 'Unknown'
    b = 'Unknown'
    c = 'Unknown'
    d = 'Unknown'
  return a , b , c, d
```

```
df1['source_city'] = df1['source_name'].apply(center_split).apply(lambda x : x[0])
df1['source_place'] = df1['source_name'].apply(center_split).apply(lambda x : x[1])
df1['source_code'] = df1['source_name'].apply(center_split).apply(lambda x : x[2])
df1['source_state'] = df1['source_name'].apply(center_split).apply(lambda x : x[3])
```

```
df1['destination_city'] = df1['destination_name'].apply(center_split).apply(lambda x : x[0])
df1['destination_place'] = df1['destination_name'].apply(center_split).apply(lambda x : x[1])
df1['destination_code'] = df1['destination_name'].apply(center_split).apply(lambda x : x[2])
df1['destination_state'] = df1['destination_name'].apply(center_split).apply(lambda x : x[3])
```

```
df1.iloc[0:2,:]
```

	segment_key	data	trip_creation_time	trip_uuid	route_schedule_uuid	route_type	source_
0	trip-153671041653548748_IND209304AAA_IND000000ACB	training	2018-09-12 00:00:16.535741	trip-153671041653548748	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	Kanpur_Centra (Uttar Pra
1	trip-153671041653548748_IND462022AAA_IND209304AAA	training	2018-09-12 00:00:16.535741	trip-153671041653548748	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	Bhopal_Trnsp (Madhya Pra

2 rows × 27 columns

```
df1['month'] = df1['trip_creation_time'].dt.month
df1['year'] = df1['trip_creation_time'].dt.year
df1['day'] = df1['trip_creation_time'].dt.day
df1['hour'] = df1['trip_creation_time'].dt.hour
df1['day_name'] = df1['trip_creation_time'].dt.day_name()
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26368 entries, 0 to 26367
Data columns (total 32 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   segment_key                     26368 non-null  object
 1   data                            26368 non-null  object
 2   trip_creation_time              26368 non-null  datetime64[ns]
 3   trip_uuid                       26368 non-null  object
 4   route_schedule_uuid             26368 non-null  object
 5   route_type                      26368 non-null  object
 6   source_name                     26368 non-null  object
 7   destination_name                26368 non-null  object
 8   od_start_time                   26368 non-null  datetime64[ns]
 9   od_end_time                     26368 non-null  datetime64[ns]
 10  start_scan_to_end_scan          26368 non-null  int8
 11  actual_distance_to_destination  26368 non-null  float64
 12  actual_time                     26368 non-null  int64
 13  osrm_time                       26368 non-null  int64
 14  osrm_distance                   26368 non-null  float64
 15  segment_actual_time_sum         26368 non-null  int64
 16  segment_osrm_distance_sum       26368 non-null  float64
 17  segment_osrm_time_sum           26368 non-null  int64
 18  od_time_diff                    26368 non-null  int64
 19  source_city                     26368 non-null  object
 20  source_place                    26368 non-null  object
 21  source_code                     26368 non-null  object
 22  source_state                    26368 non-null  object
 23  destination_city                26368 non-null  object
 24  destination_place               26368 non-null  object
 25  destination_code                26368 non-null  object
 26  destination_state               26368 non-null  object
 27  month                           26368 non-null  int32
 28  year                            26368 non-null  int32
 29  day                             26368 non-null  int32
 30  hour                            26368 non-null  int32
 31  day_name                        26368 non-null  object
dtypes: datetime64[ns](3), float64(3), int32(4), int64(5), int8(1), object(16)
memory usage: 5.9+ MB
```

In-depth Analysis

```
df1.columns
```

```
Index(['segment_key', 'data', 'trip_creation_time', 'trip_uuid',
       'route_schedule_uuid', 'route_type', 'source_name', '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_sum', 'segment_osrm_distance_sum',
       'segment_osrm_time_sum', 'od_time_diff', 'source_city', 'source_place',
       'source_code', 'source_state', 'destination_city', 'destination_place',
       'destination_code', 'destination_state', 'month', 'year', 'day', 'hour',
       'day_name'],
      dtype='object')
```

```
create_trip_dict = {
      'data' : 'first','route_type' : 'first','start_scan_to_end_scan' : 'first',
      'actual_distance_to_destination' : 'first', 'actual_time' : 'first', 'osrm_time' : 'first',
      'osrm_distance' : 'first', 'segment_actual_time_sum' : 'first', 'segment_osrm_distance_sum' : 'first',
      'segment_osrm_time_sum': 'first', 'od_time_diff' : 'first', 'source_city' : 'first', 'source_state' : 'first', 'destination_city' : 'first',
      'destination_state' : 'first', 'month' : 'first', 'year' : 'first', 'day' : 'first', 'day_name' : 'first', 'hour' : 'first'
}
```

```
df2 = df1.groupby('trip_uuid').agg(create_trip_dict).reset_index()
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14817 entries, 0 to 14816
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   trip_uuid                       14817 non-null  object
 1   data                            14817 non-null  object
 2   route_type                      14817 non-null  object
 3   start_scan_to_end_scan          14817 non-null  int8
 4   actual_distance_to_destination  14817 non-null  float64
 5   actual_time                     14817 non-null  int64
 6   osrm_time                       14817 non-null  int64
 7   osrm_distance                   14817 non-null  float64
 8   segment_actual_time_sum         14817 non-null  int64
```

```
 9   segment_osrm_distance_sum       14817 non-null  float64
 10  segment_osrm_time_sum           14817 non-null  int64
 11  od_time_diff                    14817 non-null  int64
 12  source_city                     14817 non-null  object
 13  source_state                    14817 non-null  object
 14  destination_city                14817 non-null  object
 15  destination_state               14817 non-null  object
 16  month                           14817 non-null  int32
 17  year                            14817 non-null  int32
 18  day                             14817 non-null  int32
 19  day_name                        14817 non-null  object
 20  hour                            14817 non-null  int32
dtypes: float64(3), int32(4), int64(5), int8(1), object(8)
memory usage: 2.0+ MB
```

```
print(df2.head())
```

```
                 trip_uuid      data route_type  start_scan_to_end_scan  \
0  trip-153671041653548748  training        FTL                     -20
1  trip-153671042288605164  training    Carting                      58
2  trip-153671043369099517  training        FTL                      66
3  trip-153671046011330457  training    Carting                     100
4  trip-153671052974046625  training        FTL                    -104

   actual_distance_to_destination  actual_time  osrm_time  osrm_distance  \
0                         3778.76          340        136        4540.11
1                           53.31           96         55          60.31
2                         1725.60          297       -109        1975.73
3                           28.53           82         24          31.65
4                          126.28           21         97         135.04

   segment_actual_time_sum  segment_osrm_distance_sum  ...  od_time_diff  \
0                      472                     670.60  ...          1261
1                       46                      28.20  ...            59
2                      352                     317.74  ...           835
3                       59                      19.88  ...           101
4                      147                      63.64  ...           153

      source_city   source_state destination_city destination_state month  \
0  Kanpur_Central  Uttar Pradesh          Gurgaon           Haryana     9
1      Doddablpur      Karnataka       Chikblapur         Karnataka     9
2         Gurgaon        Haryana       Chandigarh            Punjab     9
3         Unknown        Unknown           Mumbai       Maharashtra     9
4         Unknown        Unknown          Unknown           Unknown     9

   year  day   day_name hour
0  2018   12  Wednesday    0
1  2018   12  Wednesday    0
2  2018   12  Wednesday    0
3  2018   12  Wednesday    0
4  2018   12  Wednesday    0

[5 rows x 21 columns]
```

```
df2.describe()
```

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time_sum	segment_osrm_dis
count	14817.000000	14817.000000	14817.000000	14817.000000	14817.000000	14817.000000	148
mean	8.798947	1777.927886	108.864682	110.653304	2157.885488	194.382129	1
std	81.196025	7894.574949	141.580812	147.564852	9614.572736	397.911979	3
min	-128.000000	9.000000	-888.000000	-989.000000	9.070000	-201.000000	
25%	-69.000000	43.680000	47.000000	34.000000	53.120000	47.000000	
50%	25.000000	84.970000	101.000000	73.000000	104.290000	79.000000	
75%	83.000000	213.370000	182.000000	159.000000	274.430000	149.000000	
max	127.000000	85110.850000	950.000000	1304.000000	102415.860000	3081.000000	26

```
df2.dtypes.unique()
```

```
array([dtype('O'), dtype('int8'), dtype('float64'), dtype('int64'),
       dtype('int32')], dtype=object)
```

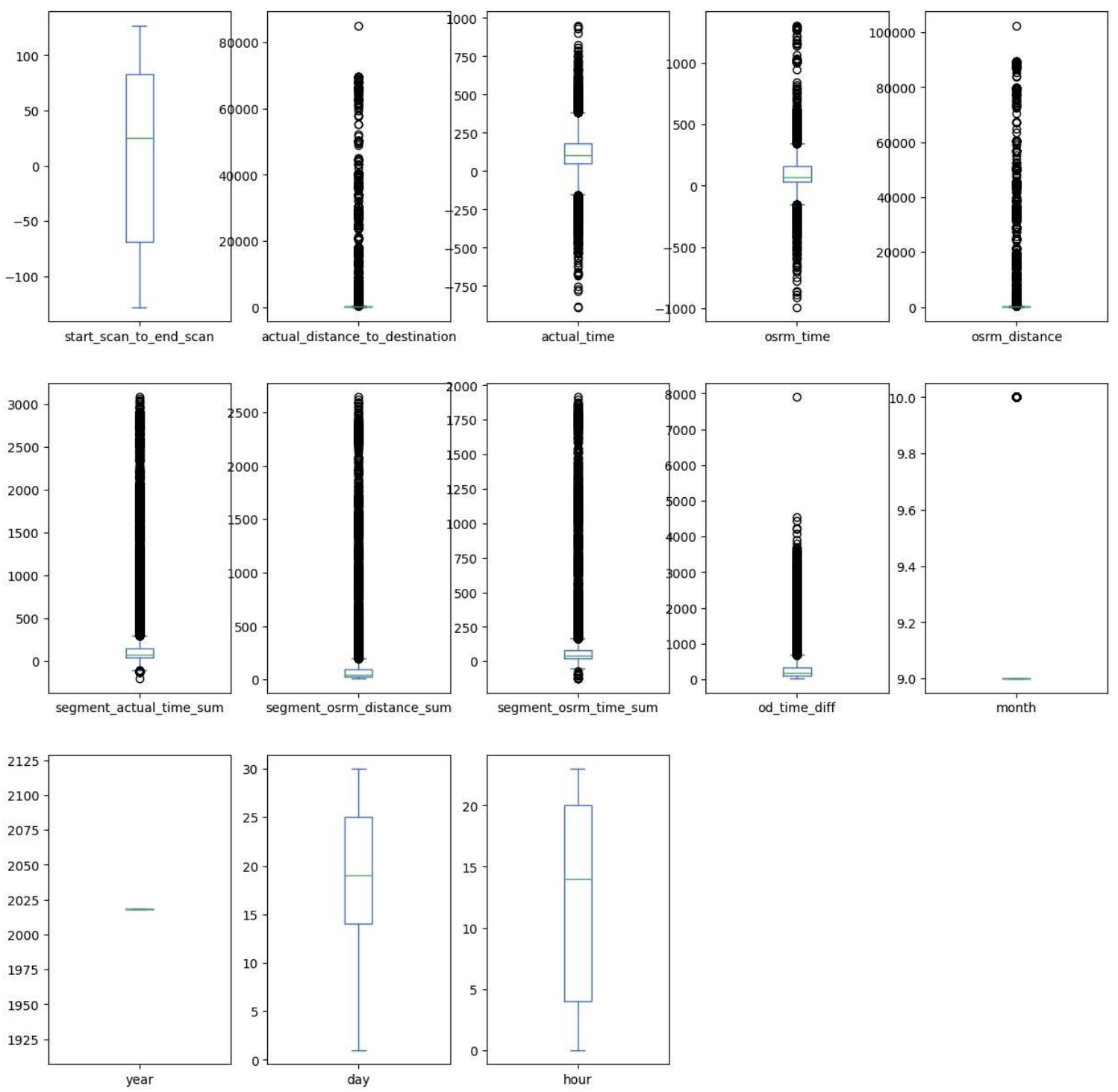
```
categorical_columns = df2.select_dtypes(include=['object', 'string']).columns.tolist()
categorical_columns
```

```
['trip_uuid',
 'data',
 'route_type',
 'source_city',
 'source_state',
 'destination_city',
 'destination_state',
 'day_name']
```

```
numerical_columns = df2.select_dtypes(include = ['int8', 'float64','int64','int32']).columns.tolist()
numerical_columns
```

```
['start_scan_to_end_scan',
 'actual_distance_to_destination',
 'actual_time',
 'osrm_time',
 'osrm_distance',
 'segment_actual_time_sum',
 'segment_osrm_distance_sum',
 'segment_osrm_time_sum',
 'od_time_diff',
 'month',
 'year',
 'day',
 'hour']
```

```
df2[numerical_columns].plot(kind='box', subplots = True, layout=(3 , int(np.ceil(len(numerical_columns)/3))), figsize=(15, 15))
plt.show()
```




```
iqr_values = df2[numerical_columns].quantile(0.75) - df2[numerical_columns].quantile(0.25)
print(iqr_values)
```

```
start_scan_to_end_scan            152.00
actual_distance_to_destination    169.69
actual_time                       135.00
osrm_time                         125.00
osrm_distance                     221.31
segment_actual_time_sum           102.00
segment_osrm_distance_sum          68.74
segment_osrm_time_sum              56.00
od_time_diff                      230.00
month                               0.00
year                                0.00
day                                11.00
hour                               16.00
dtype: float64
```

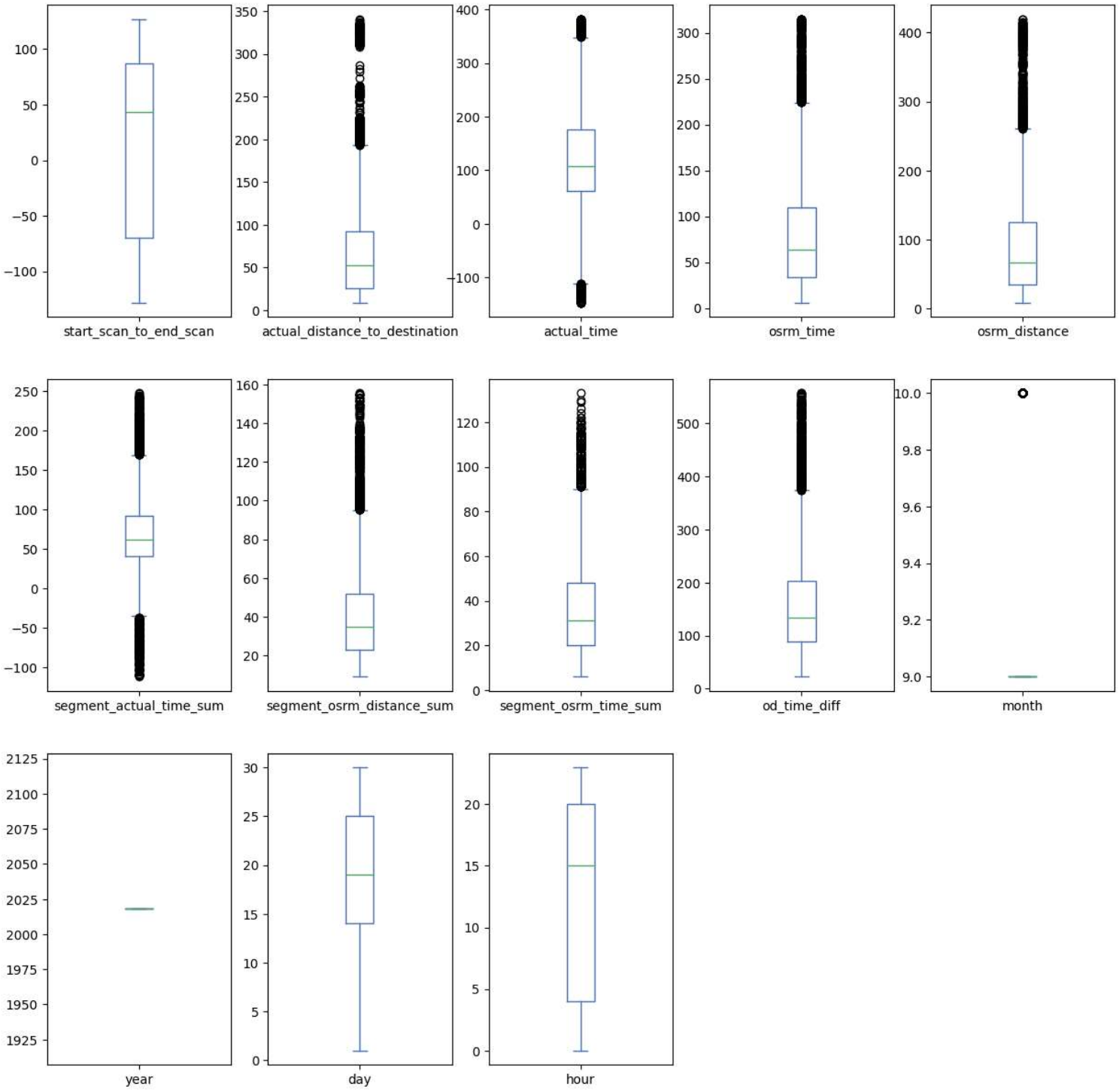
For Columns 'actual_distance_to_destination', 'actual_time', 'osrm_time' ,'osrm_distance', 'segment_actual_time_sum', 'segment_osrm_distance_sum', 'segment_osrm_time_sum', 'od_time_diff' , IQR method of 'Removal' to be used as the values are far more spread out from the mean and their effect cannot be added to the data for further use as the data points are more than 10 times away from the mean.

```
numerical_columns_subset1 = []
for i in numerical_columns:
  if (2 * np.average(df2[i])) < max(df2[i]):
    numerical_columns_subset1.append(i)
numerical_columns_subset1
```

```
['start_scan_to_end_scan',
 'actual_distance_to_destination',
 'actual_time',
 'osrm_time',
 'osrm_distance',
 'segment_actual_time_sum',
 'segment_osrm_distance_sum',
 'segment_osrm_time_sum',
 'od_time_diff']
```

```
for i in numerical_columns_subset1:
  lower_bound = df2[i].quantile(0.25) - 1.5 * iqr_values[i]
  upper_bound = df2[i].quantile(0.75) + 1.5 * iqr_values[i]
  df2 = df2[(df2[i] >= lower_bound) & (df2[i] <= upper_bound)]
```

```
df2[numerical_columns].plot(kind='box', subplots = True, layout=(3 , int(np.ceil(len(numerical_columns)/3))), figsize=(15, 15))
plt.show()
```




```
df2.reset_index(drop=True , inplace = True)
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10860 entries, 0 to 10859
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   trip_uuid                       10860 non-null  object
 1   data                            10860 non-null  object
 2   route_type                      10860 non-null  object
 3   start_scan_to_end_scan          10860 non-null  int8
 4   actual_distance_to_destination  10860 non-null  float64
 5   actual_time                     10860 non-null  int64
 6   osrm_time                       10860 non-null  int64
 7   osrm_distance                   10860 non-null  float64
 8   segment_actual_time_sum         10860 non-null  int64
 9   segment_osrm_distance_sum       10860 non-null  float64
 10  segment_osrm_time_sum           10860 non-null  int64
 11  od_time_diff                    10860 non-null  int64
 12  source_city                     10860 non-null  object
 13  source_state                    10860 non-null  object
 14  destination_city                10860 non-null  object
```

```
 15  destination_state              10860 non-null  object
 16  month                          10860 non-null  int32
 17  year                           10860 non-null  int32
 18  day                            10860 non-null  int32
 19  day_name                       10860 non-null  object
 20  hour                           10860 non-null  int32
dtypes: float64(3), int32(4), int64(5), int8(1), object(8)
memory usage: 1.5+ MB
```

```
df2.head()
```

	trip_uuid	data	route_type	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_a
0	trip-153671042288605164	training	Carting	58	53.31	96	55	60.31	
1	trip-153671046011330457	training	Carting	100	28.53	82	24	31.65	
2	trip-153671052974046625	training	FTL	-104	126.28	21	97	135.04	
3	trip-153671055416136166	training	Carting	60	25.14	71	19	26.35	
4	trip-153671066201138152	training	Carting	98	9.10	24	13	12.02	

5 rows × 21 columns

```
df2.describe()
```

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time_sum	segment_osrm_dis
count	10860.000000	10860.000000	10860.000000	10860.000000	10860.000000	10860.000000	108
mean	13.399908	74.252401	120.760405	83.738766	93.718844	69.743462	
std	83.137055	62.042412	92.977174	64.240687	75.569964	44.930642	
min	-128.000000	9.000000	-147.000000	6.000000	9.070000	-112.000000	
25%	-70.000000	25.530000	61.000000	34.000000	34.435000	41.000000	
50%	43.000000	52.075000	107.000000	64.000000	67.475000	62.000000	
75%	87.000000	92.752500	176.000000	110.000000	125.027500	92.000000	
max	127.000000	340.760000	382.000000	315.000000	419.390000	247.000000	1

Data Normalization

```
df2[numerical_columns_subset1]
```

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time_sum	segment_osrm_distanc
0	58	53.31	96	55	60.31	46	
1	100	28.53	82	24	31.65	59	
2	-104	126.28	21	97	135.04	147	
3	60	25.14	71	19	26.35	39	
4	98	9.10	24	13	12.02	24	
...	...	...	...	...	...	...	
10855	-104	54.12	70	51	58.26	33	
10856	60	25.13	33	19	26.54	21	
10857	-8	46.63	36	61	74.28	190	
10858	105	128.37	207	161	163.87	59	
10859	31	62.55	22	59	76.52	-23	

10860 rows × 9 columns

As multiple colums are having the measurements like time and distances. They are needed for comparision with each other , by using StandardScaler, the 'Mean' value is missed and comarision cannot be carried out using statistical methods. The columns needed to be MinMaxScaloed based on the minimum and maximum value across similar attributes.

```
df2.columns
```

```
Index(['trip_uuid', 'data', 'route_type', 'start_scan_to_end_scan',
       'actual_distance_to_destination', 'actual_time', 'osrm_time',
       'osrm_distance', 'segment_actual_time_sum', 'segment_osrm_distance_sum',
       'segment_osrm_time_sum', 'od_time_diff', 'source_city', 'source_state',
       'destination_city', 'destination_state', 'month', 'year', 'day',
       'day_name', 'hour'],
      dtype='object')
```

```
distance_columns = ['actual_distance_to_destination','osrm_distance', 'segment_osrm_distance_sum']
time_columns = ['actual_time', 'osrm_time','segment_actual_time_sum', 'segment_osrm_time_sum', 'od_time_diff']
```

```
concatenated_data = np.concatenate([df2[attr].values.reshape(-1, 1) for attr in distance_columns], axis=1)
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(concatenated_data)
scaled_data = scaled_data.reshape(-1, len(distance_columns))
df2[distance_columns] = pd.DataFrame(scaled_data, columns=distance_columns)
df2.head()
```

	trip_uuid	data	route_type	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_a
0	trip-153671042288605164	training	Carting	58	0.133560	96	55	0.124878	
1	trip-153671046011330457	training	Carting	100	0.058868	82	24	0.055030	
2	trip-153671052974046625	training	FTL	-104	0.353509	21	97	0.307004	
3	trip-153671055416136166	training	Carting	60	0.048650	71	19	0.042113	
4	trip-153671066201138152	training	Carting	98	0.000301	24	13	0.007190	

5 rows × 21 columns

```
concatenated_data = np.concatenate([df2[attr].values.reshape(-1, 1) for attr in time_columns], axis=1)
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(concatenated_data)
scaled_data = scaled_data.reshape(-1, len(time_columns))
df2[time_columns] = pd.DataFrame(scaled_data, columns=time_columns)
df2.head()
```

	trip_uuid	data	route_type	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_a
0	trip-153671042288605164	training	Carting	58	0.133560	0.459357	0.158576	0.124878	
1	trip-153671046011330457	training	Carting	100	0.058868	0.432892	0.058252	0.055030	
2	trip-153671052974046625	training	FTL	-104	0.353509	0.317580	0.294498	0.307004	
3	trip-153671055416136166	training	Carting	60	0.048650	0.412098	0.042071	0.042113	
4	trip-153671066201138152	training	Carting	98	0.000301	0.323251	0.022654	0.007190	

5 rows × 21 columns

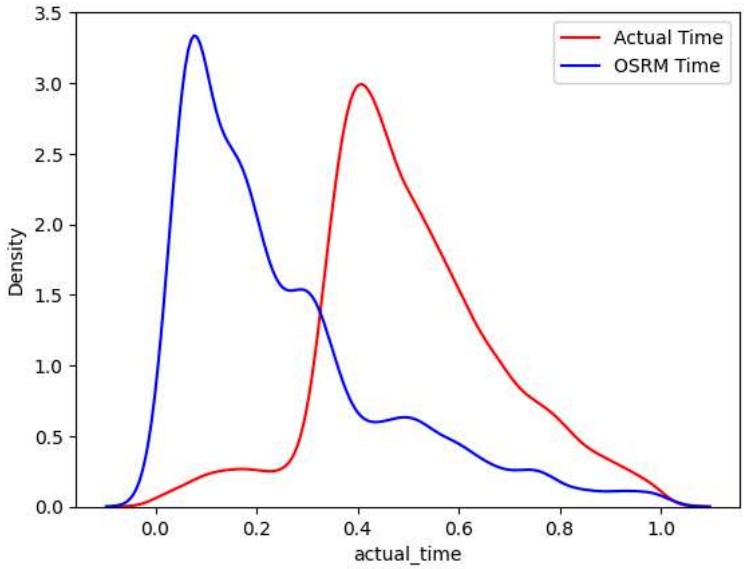
```
df2[categorical_columns].nunique()
```

```
trip_uuid            10860
data                     2
route_type               2
source_city            561
source_state            26
destination_city       668
destination_state       27
day_name                 7
dtype: int64
```

As number of unique entities are higher in all but 2, i.e. data and route_type. One Hot Encoding is preffered over those two attributes as they do not increase dimensionality of the data by much.

Hypothesis Testing

```
# 1. OSRM Time vs Actual Time
sns.kdeplot(df2['actual_time'], color= 'red', label = 'Actual Time')
sns.kdeplot(df2['osrm_time'], color = 'blue', label = 'OSRM Time')

plt.legend()
plt.show()
```



Based on the grahical nature of both the curves the left-handed Relative T-test is selected as the best statistical tool to analyse the nature of both values.

```
H0 = 'Failed to reject Null Hypothesis.  OSRM time and Actual Time for the delivery are same.With P-Value of'
Ha = 'Null Hypothesis is Rejected. Actual Time is greater than the OSRM time. With P-Value of'

stat, pvalue = ttest_rel(df2['actual_time'],df2['osrm_time'],alternative = 'greater')
alpha = 0.05

if pvalue < alpha:
  print(f"{Ha} {pvalue.round(2)}")
else :
  print(f"{H0} {pvalue.round(2)}")
```

```
Null Hypothesis is Rejected. Actual Time is greater than the OSRM time. With P-Value of 0.0
```

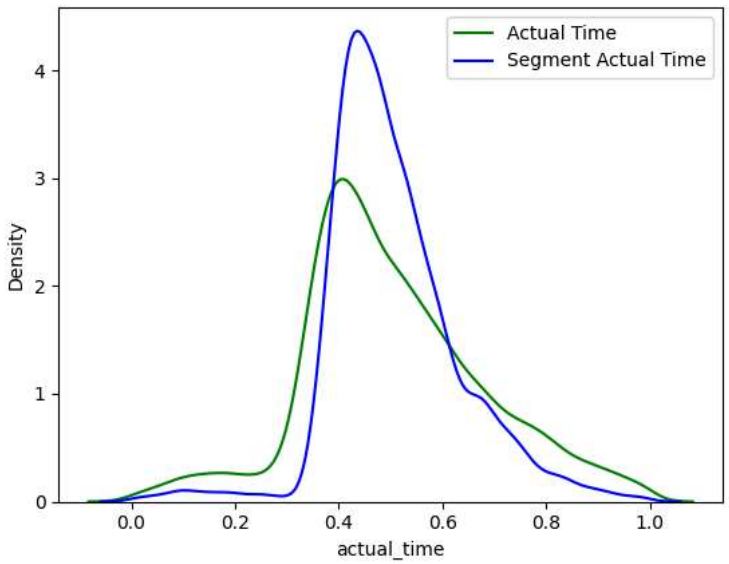
```
ttest_rel(df2['actual_time'],df2['osrm_time'],alternative = 'greater')
```

```
TtestResult(statistic=124.96971893911095, pvalue=0.0, df=10859)
```

Observatins:

1. The spread of both the data i.e. Actual Time and OSRM time are similar.
2. The mean of Actual time is higher than the OSRM time.
3. The OSRM time and Actual time are different from one another.
4. Actual Time is greater than the OSRM time for the deliveries.
5. The OSRM time does not denote the Actual delivery time accurately. The OSRM delivery time is not a good measure of the actual time to be used for planning deliveries.
6. The OSRM time is required to be corrected for more accurate delivery planning.

```
#2. Actual Time vs Segment Actual Time

sns.kdeplot(df2['actual_time'], color= 'green', label = 'Actual Time')
sns.kdeplot(df2['segment_actual_time_sum'], color = 'blue', label = 'Segment Actual Time')

plt.legend()
plt.show()
```



Based on the grahical nature of both the curves the two-sided Relative T-test is selected as the best statistical tool to analyse the nature of both values.

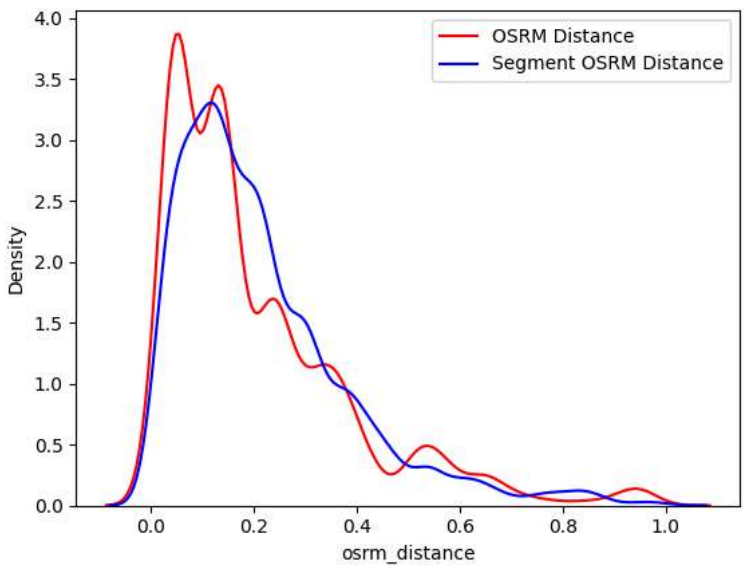
```
H0 = 'Failed to reject Null Hypothesis. Segment Actual Time and Actual Time for the delivery are similar. With P-Value of'
Ha = 'Null Hypothesis is Rejected. Actual Time and Segement Actual time are not same. With P-Value of'

stat, pvalue = ttest_rel(df2['actual_time'],df2['segment_actual_time_sum'],alternative = 'two-sided')
alpha = 0.05

if pvalue < alpha:
  print(f"{Ha} {pvalue.round(2)}")
else :
  print(f"{H0} {pvalue.round(2)}")
```

```
Failed to reject Null Hypothesis. Segment Actual Time and Actual Time for the delivery are similar. With P-Value of 0.96
```

Observatins:

1. The spread of both the data i.e. Actual Time and Actual Segment time are similar.
2. The mean of Actual time is in same range in that of segment Actual time.
3. The Segment Actual time and Actual time are denoting the same things.
4. Actual Time is similar to Actual Segment time for the deliveries.
5. The Actual Segment time does denote the Actual delivery time accurately. The Actual Segment delivery time is a good measure of the actual time to be used for planning deliveries.

```
# 3. OSRM Distance vs Segment OSRM Distance
sns.kdeplot(df2['osrm_distance'], color= 'red', label = 'OSRM Distance')
sns.kdeplot(df2['segment_osrm_distance_sum'], color = 'blue', label = 'Segment OSRM Distance')

plt.legend()
plt.show()
```



Based on the grahical nature of both the curves the left sided Relative T-test is selected as the best statistical tool to analyse the nature of both values.

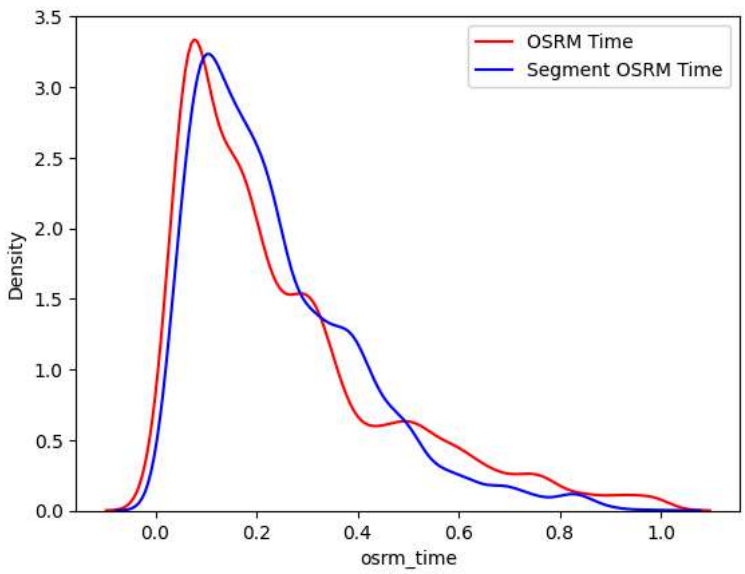
```
#OSRM Distance vs Segment OSRM Distance
H0 = 'Failed to reject Null Hypothesis. OSRM Distance is greater than Segment OSRM Distance for the delivery are similar. With P-Value of'
Ha = 'Null Hypothesis is Rejected. Segment OSRM Distance is not less than the OSRM Distance. With P-Value of'

stat, pvalue = ttest_rel(df2['osrm_distance'],df2['segment_osrm_distance_sum'],alternative = 'greater')
alpha = 0.05

if pvalue < alpha:
  print(f"{Ha} {pvalue.round(2)}")
else :
  print(f"{H0} {pvalue.round(2)}")
```

```
Failed to reject Null Hypothesis. OSRM Distance is greater than Segment OSRM Distance for the delivery are similar. With P-Value of 1.0
```

Observatins:

1. The spread of both the data i.e. Segment OSRM Distance and OSRM Distance are similar.
2. The mean of OSRM Distance is higher than the Segment OSRM Distance.
3. The OSRM Distance and Segment OSRM Distance are different from one another.
4. OSRM Distance is greater than the Segment OSRM Distance for the deliveries.
5. Use of OSRM distance data might lead to allocation of excessive resources. Leading to requirement of higher cost and rsource requirement.
6. The OSRM distance is required to be corrected for more accurate delivery planning.

```
#4. OSRM Time vs Segment OSRM Time
sns.kdeplot(df2['osrm_time'], color= 'red', label = 'OSRM Time')
sns.kdeplot(df2['segment_osrm_time_sum'], color = 'blue', label = 'Segment OSRM Time')

plt.legend()
plt.show()
```



Based on the grahical nature of both the curves the right sided Relative T-test is selected as the best statistical tool to analyse the nature of both values.

```
#4. OSRM Time vs Segment OSRM Time
H0 = 'Failed to reject Null Hypothesis. OSRM time is less than Segment OSRM Time for the delivery are similar. With P-Value of'
Ha = 'Null Hypothesis is Rejected. Segment OSRM Time is not less than the Segment OSRM Time. With P-Value of'

stat, pvalue = ttest_rel(df2['osrm_time'],df2['segment_osrm_time_sum'],alternative = 'less')
alpha = 0.05

if pvalue < alpha:
  print(f"{Ha} {pvalue.round(2)}")
else :
  print(f"{H0} {pvalue.round(2)}")
```

```
Failed to reject Null Hypothesis. OSRM time is less than Segment OSRM Time for the delivery are similar. With P-Value of 1.0
```

Observatins:

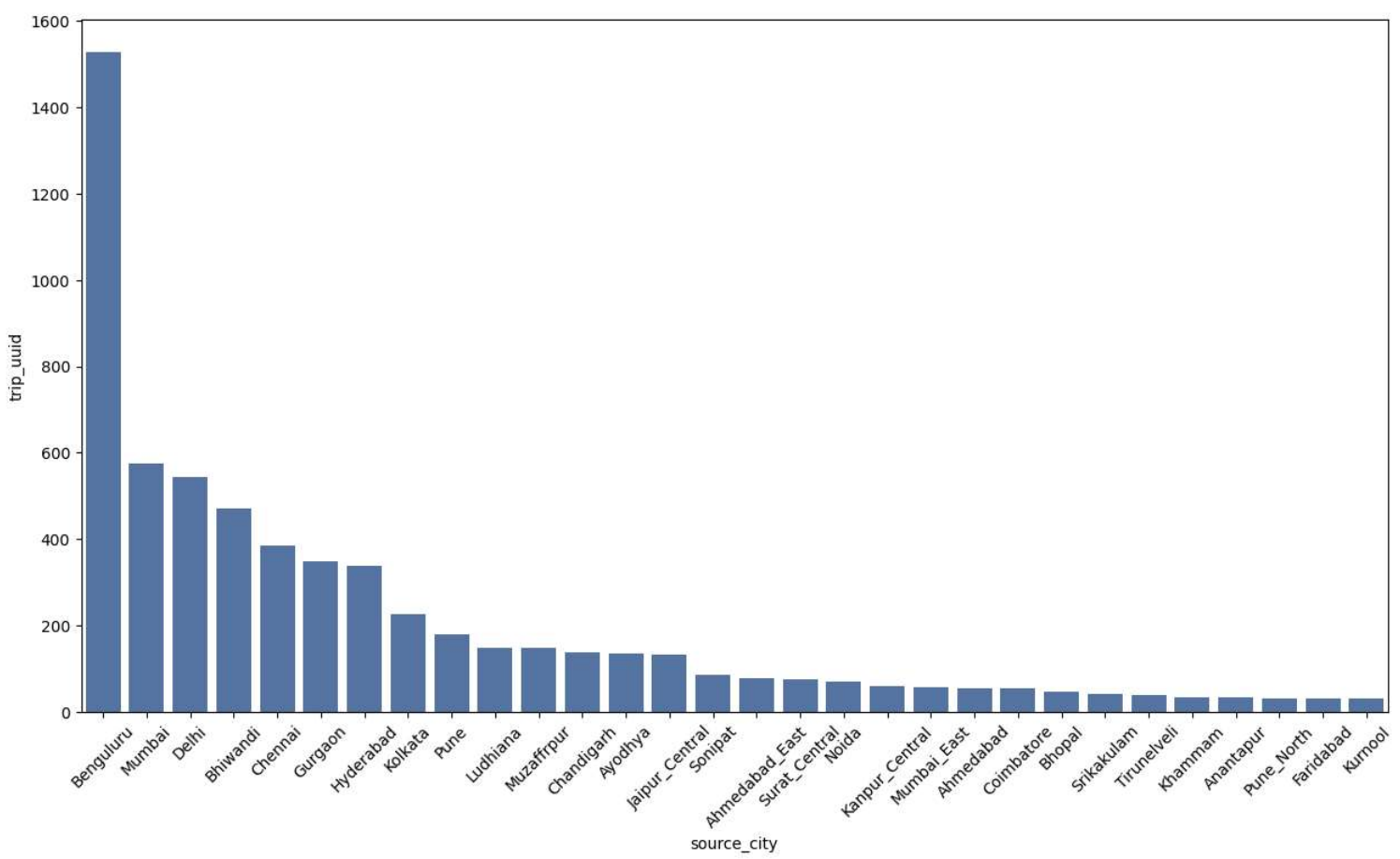
1. The spread of both the data i.e. Segment OSRM Time and OSRM Time are similar.
2. The mean of OSRM Time is less than the Segment OSRM Time.
3. The OSRM Time and Segment OSRM time are different from one another.
4. OSRM time is less than the Segment OSRM Distance for the deliveries.
5. Use of OSRM time data is better estimatior for planning for deliveries than the Segment OSRM Time.

```
[i for i in df2['source_city'].unique() if len(i) < 4]
```

```
['FBD',
 'CCU',
 'MAA',
 'BOM',
 'Pen',
 'BLR',
 'Del',
 'Cjb',
 'Amd',
 'Goa',
 'Hyd',
 'Mau']
```

```
city_name_dict = {'FBD' : 'Ayodhya', 'CCU' : 'Kolkata', 'MAA' : 'Chennai' , 'BOM' :'Mumbai', 'BLR' : 'Benguluru', 'Bangalore' : 'Benguluru',
                  'Del' : 'Delhi' , 'Cjb' : 'Coimbatore', 'AMD' : 'Ahmedabad', 'Hyd' : 'Hyderabad', 'Madras' : 'Chennai', 'Bengaluru' : 'Benguluru'
def city_dict(a):
  if a in city_name_dict:
    return city_name_dict[a]
  else:
    return a

df2['source_city'] = df2['source_city'].apply(lambda x: city_dict(x))
```

```
source_city_count = df2[df2['source_city'] != 'Unknown'].groupby('source_city').agg({'trip_uuid' : 'nunique'})
source_city_count = source_city_count.sort_values(by = 'trip_uuid',ascending = False).iloc[0:30,:]
plt.figure(figsize = (15,8))
sns.barplot(x='source_city', y='trip_uuid', data=source_city_count)
plt.xticks(rotation = 45)
plt.show()
```

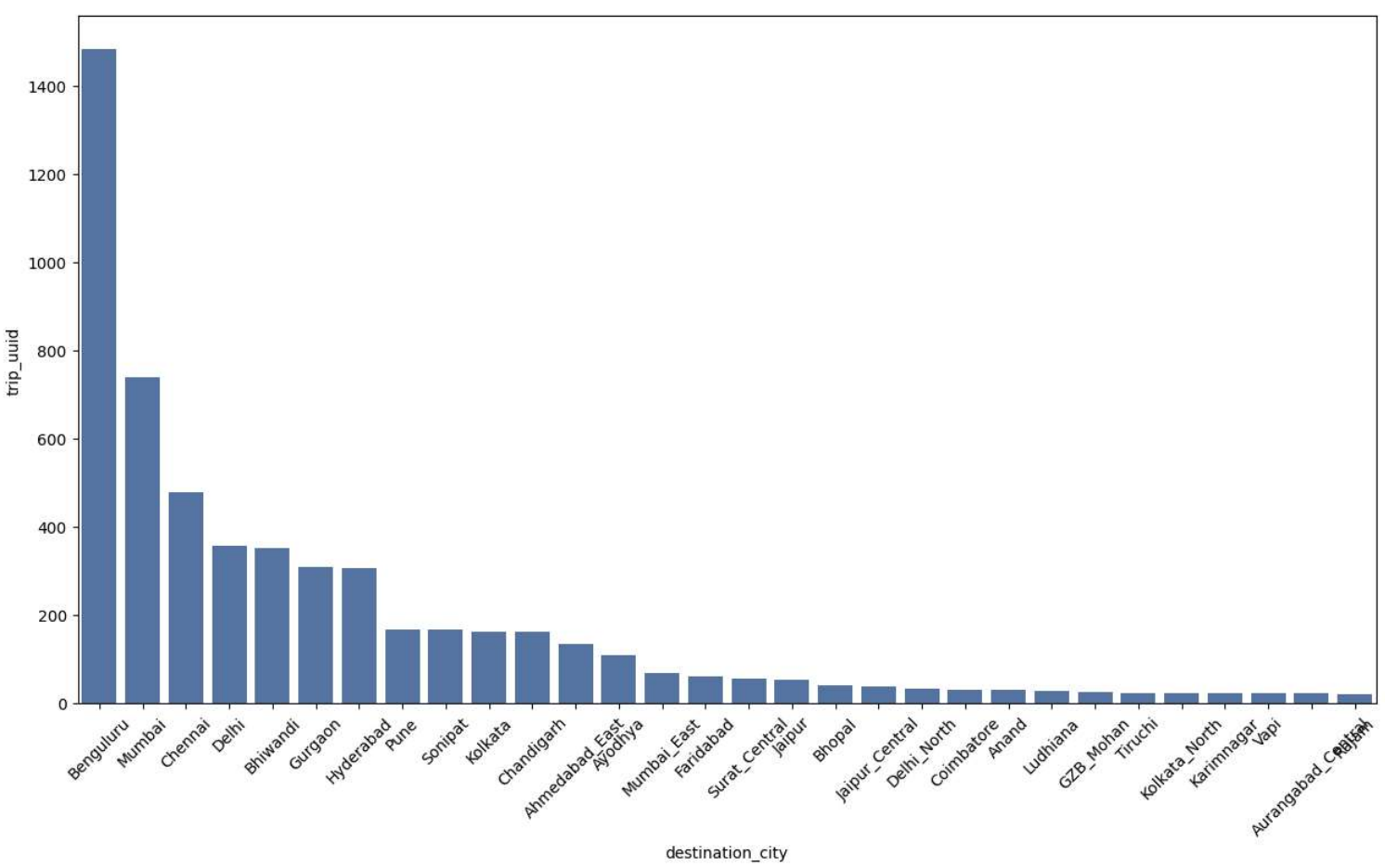



```
destination_abbr = [i for i in df2['destination_city'].unique() if len(i)<4]
destination_abbr
```

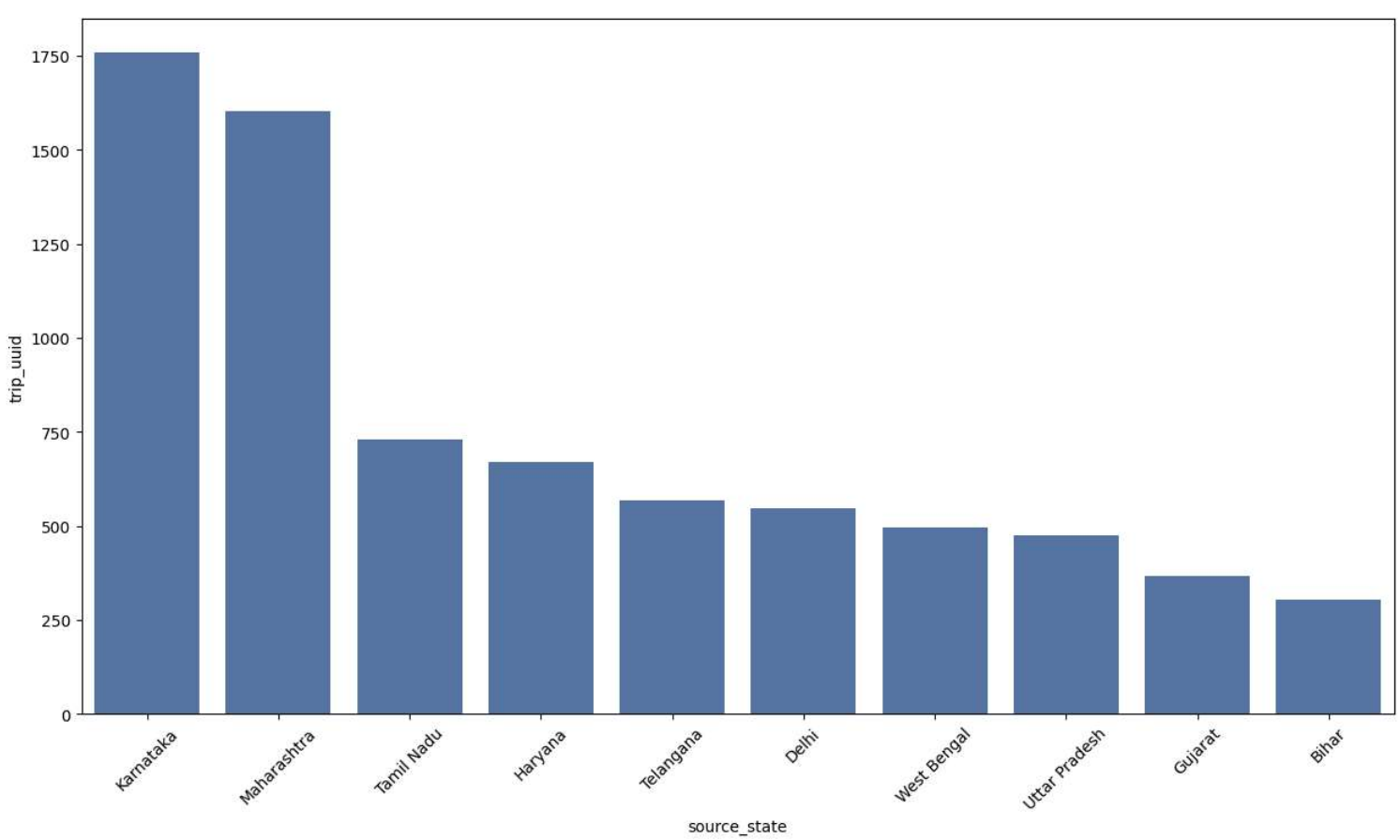
```
['MAA', 'BLR', 'FBD', 'CCU', 'Del', 'Goa', 'Hyd', 'Mau', 'Amd', 'Una', 'Cjb']
```

```
df2['destination_city'] = df2['destination_city'].apply(lambda x: city_dict(x))
```

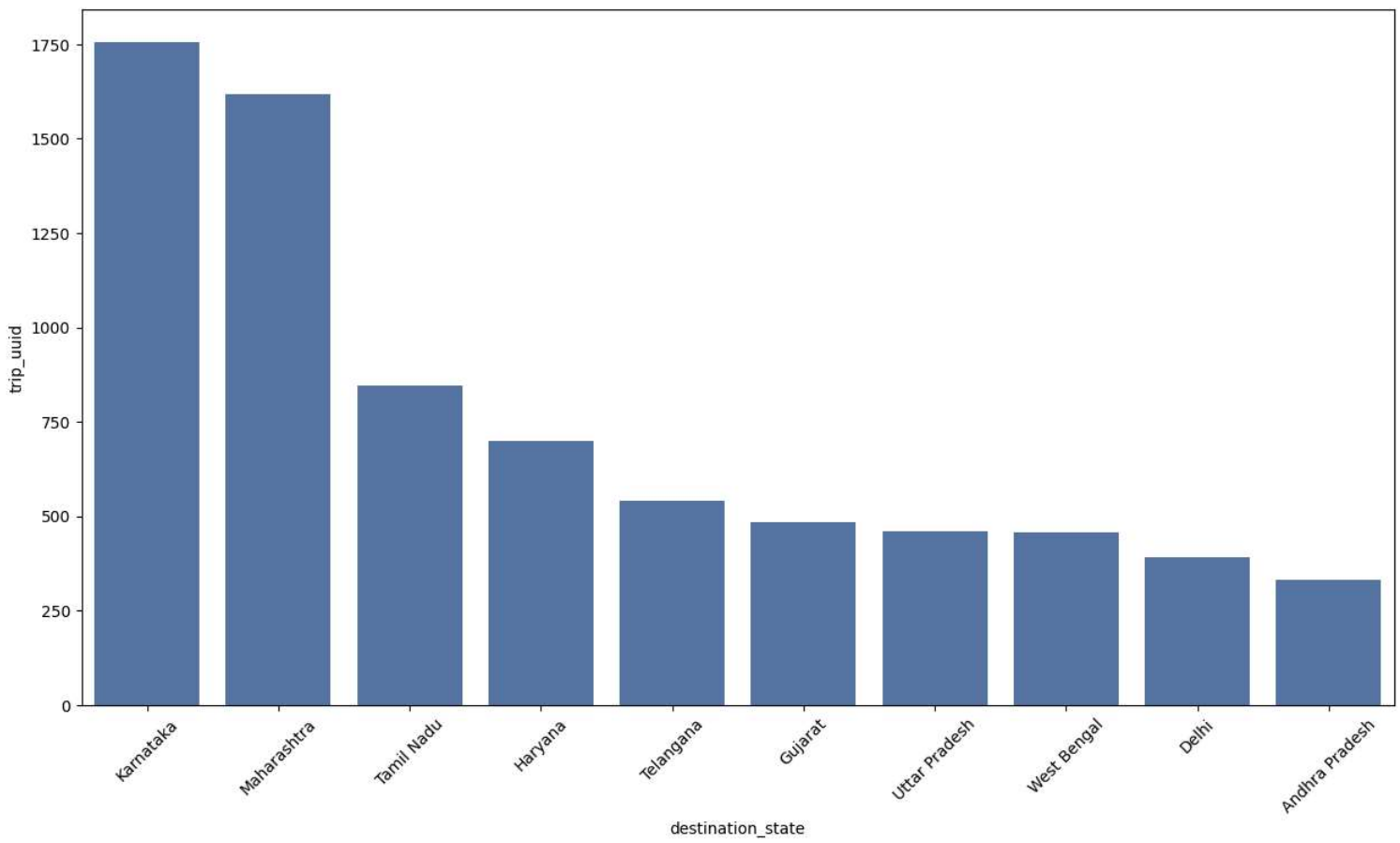
```
destination_city_count = df2[df2['destination_city'] != 'Unknown'].groupby('destination_city').agg({'trip_uuid' : 'nunique'})
destination_city_count = destination_city_count.sort_values(by = 'trip_uuid',ascending = False).iloc[0:30,:]
plt.figure(figsize = (15,8))
sns.barplot(x='destination_city', y='trip_uuid', data=destination_city_count)
plt.xticks(rotation = 45)
plt.show()
```



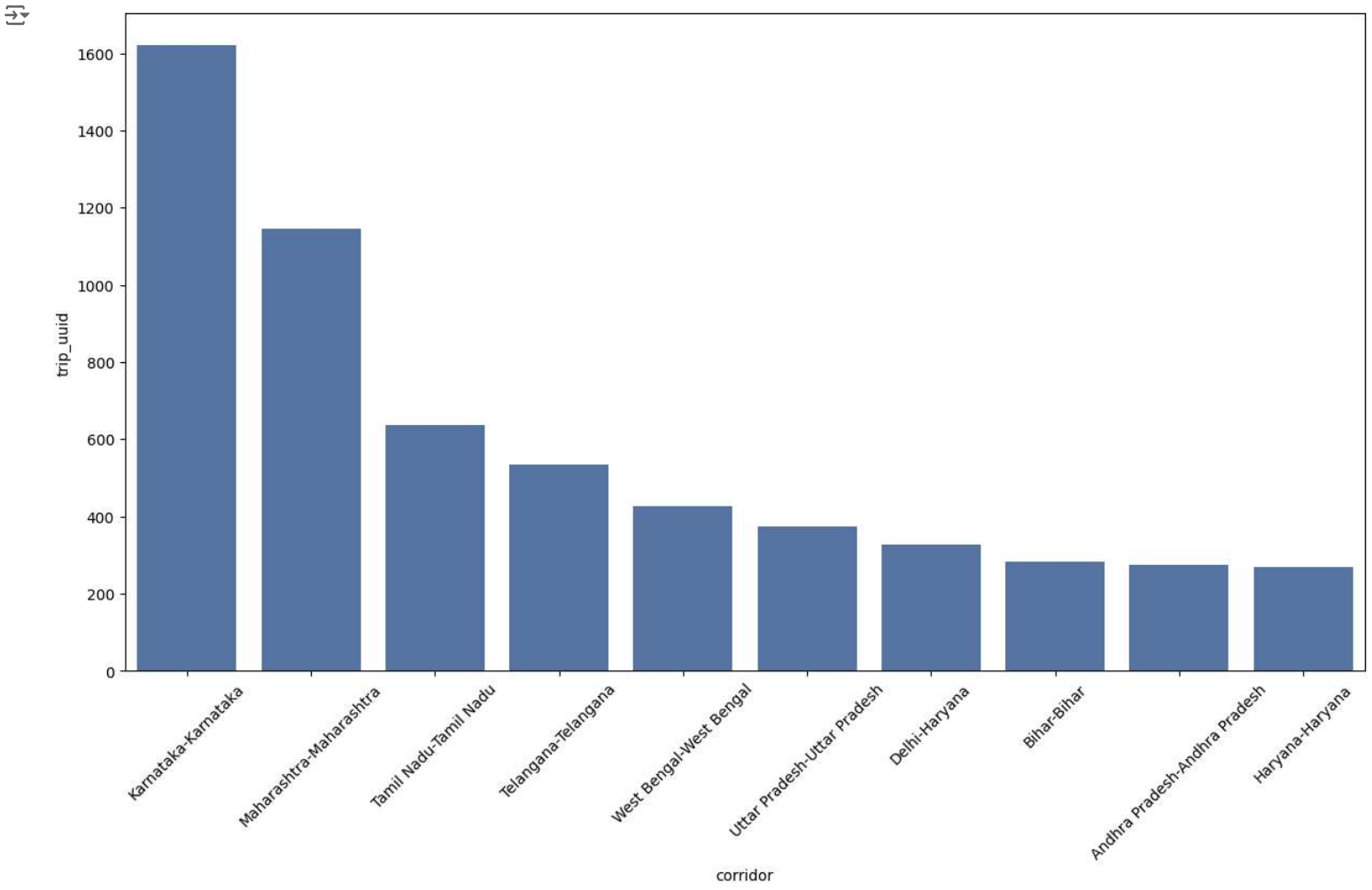

```
source_state_count = df2[df2['source_state'] != 'Unknown'].groupby('source_state').agg({'trip_uuid' : 'nunique'})
source_state_count = source_state_count.sort_values(by = 'trip_uuid',ascending = False).iloc[0:10,:]
plt.figure(figsize = (15,8))
sns.barplot(x='source_state', y='trip_uuid', data=source_state_count)
plt.xticks(rotation = 45)
plt.show()
```



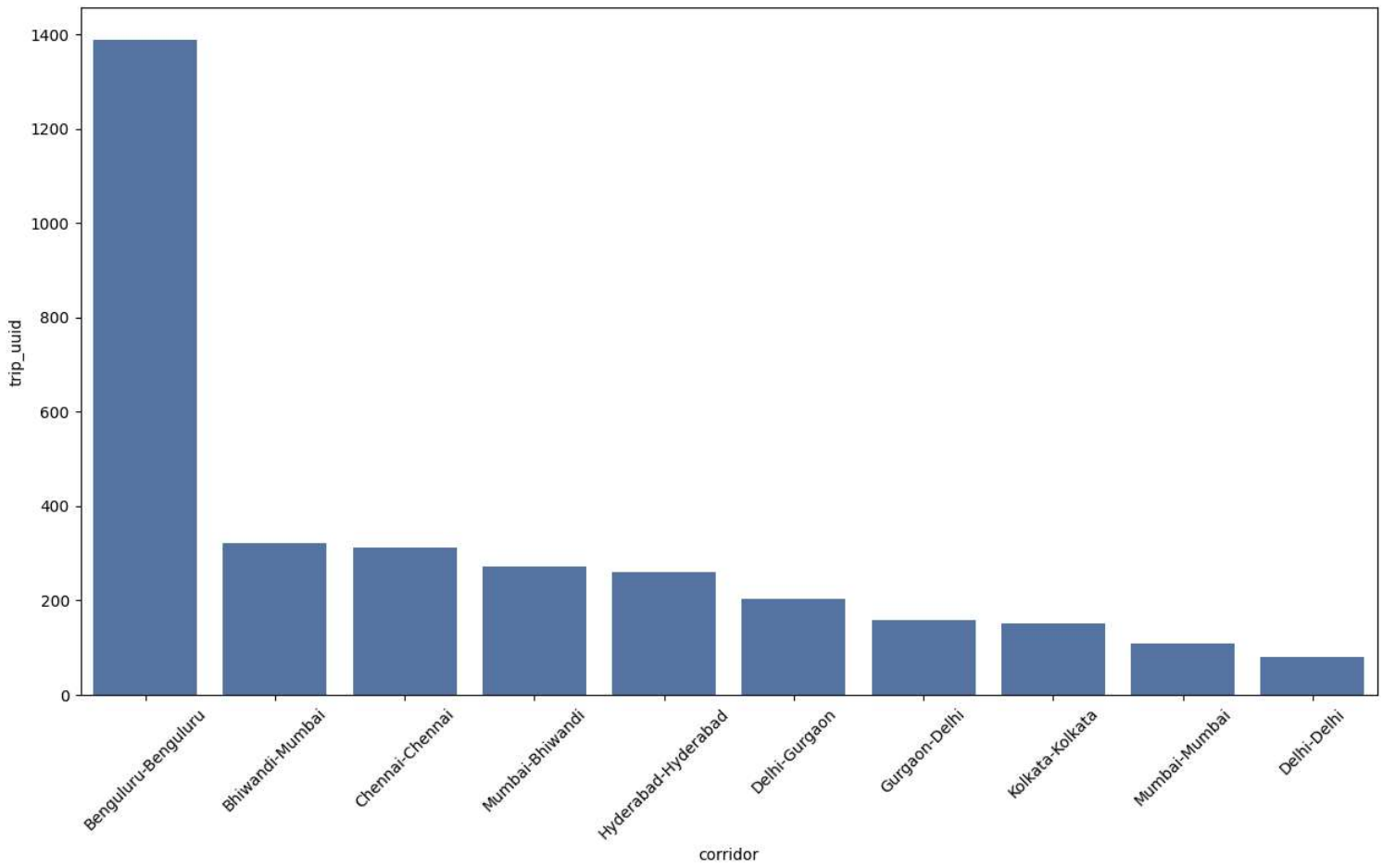

```
destination_state_count = df2[df2['destination_state'] != 'Unknown'].groupby('destination_state').agg({'trip_uuid' : 'nunique'})
destination_state_count = destination_state_count.sort_values(by = 'trip_uuid',ascending = False).iloc[0:10,:]
plt.figure(figsize = (15,8))
sns.barplot(x='destination_state', y='trip_uuid', data=destination_state_count)
plt.xticks(rotation = 45)
plt.show()
```



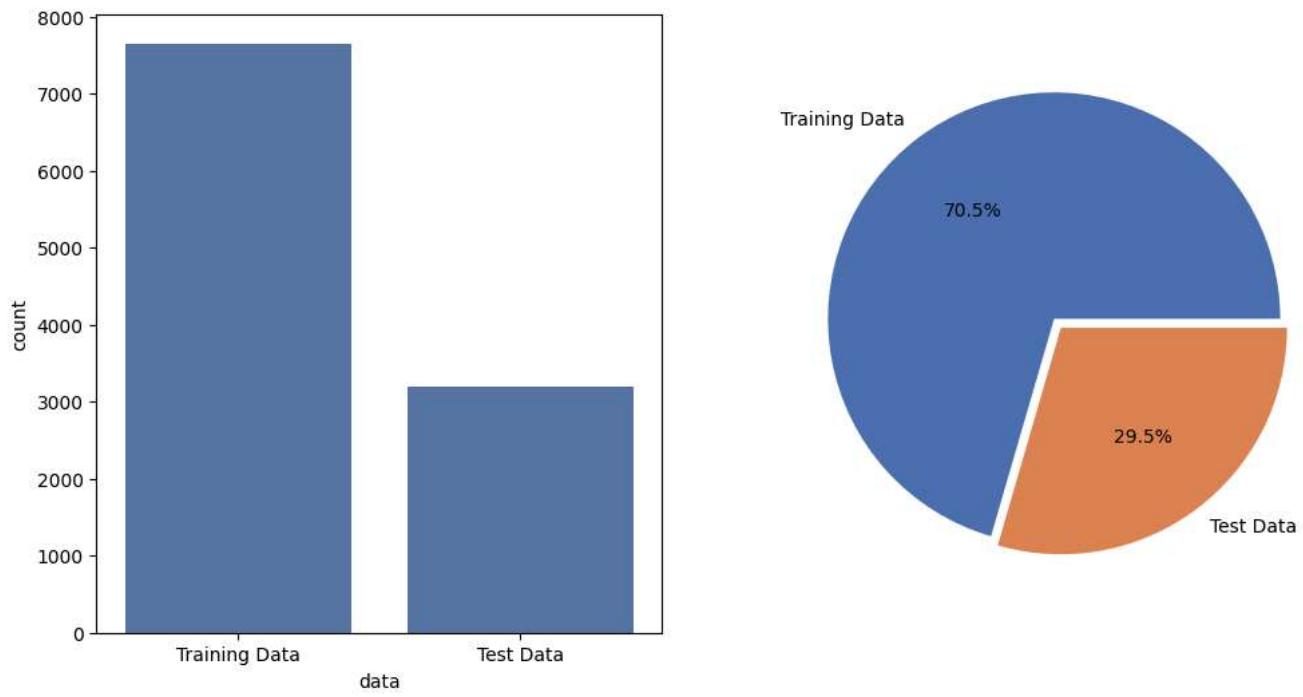
```
state_corridor = df2[(df2['destination_state'] != 'Unknown') & (df2['source_state'] != 'Unknown')].groupby(['source_state', 'destination_state']).a
state_corridor = state_corridor.sort_values(by = 'trip_uuid',ascending = False).iloc[0:10,:].reset_index()
state_corridor['corridor'] = state_corridor['source_state'] +  '-' +state_corridor['destination_state']
plt.figure(figsize = (15,8))
sns.barplot(x='corridor', y='trip_uuid', data=state_corridor)
plt.xticks(rotation = 45)
plt.show()
```



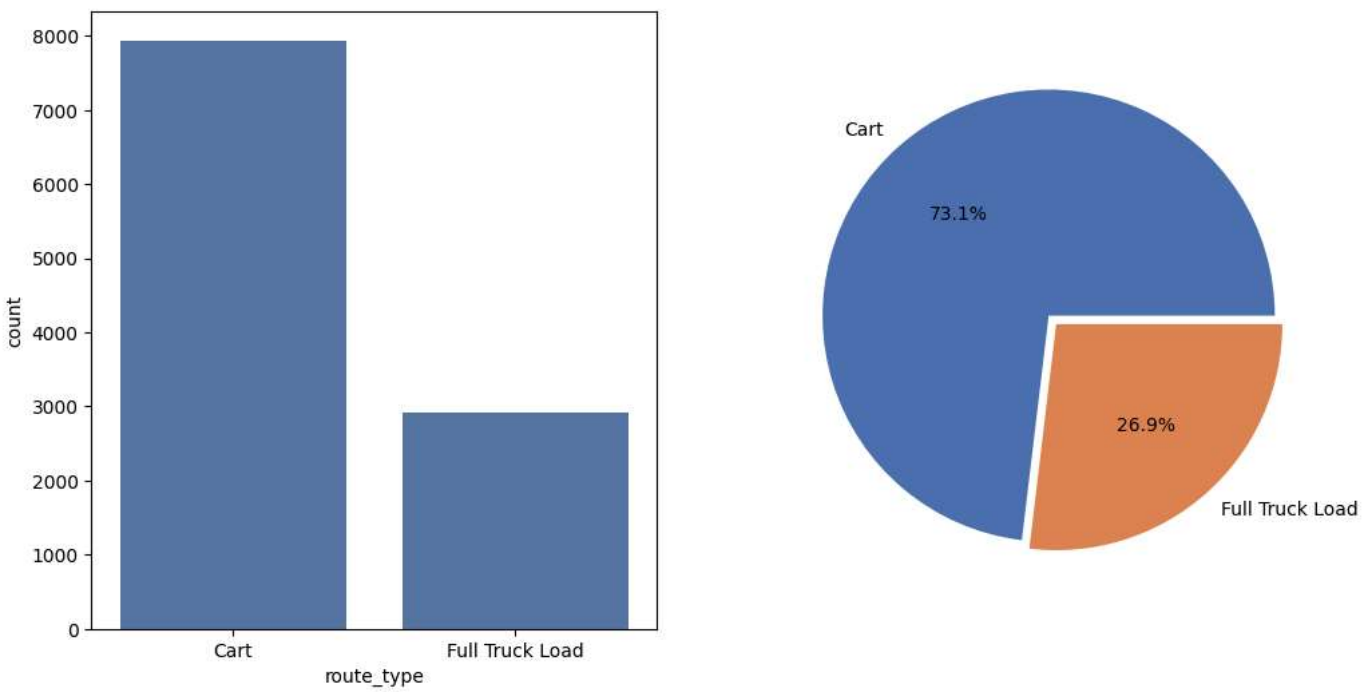

```
city_corridor = df2[(df2['destination_city'] != 'Unknown') & (df2['source_city'] != 'Unknown')].groupby(['source_city', 'destination_city']).agg({'
city_corridor = city_corridor.sort_values(by = 'trip_uuid',ascending = False).iloc[0:10,:].reset_index()
city_corridor['corridor'] = city_corridor['source_city'] +  '-' +city_corridor['destination_city']
plt.figure(figsize = (15,8))
sns.barplot(x='corridor', y='trip_uuid', data=city_corridor)
plt.xticks(rotation = 45)
plt.show()
```



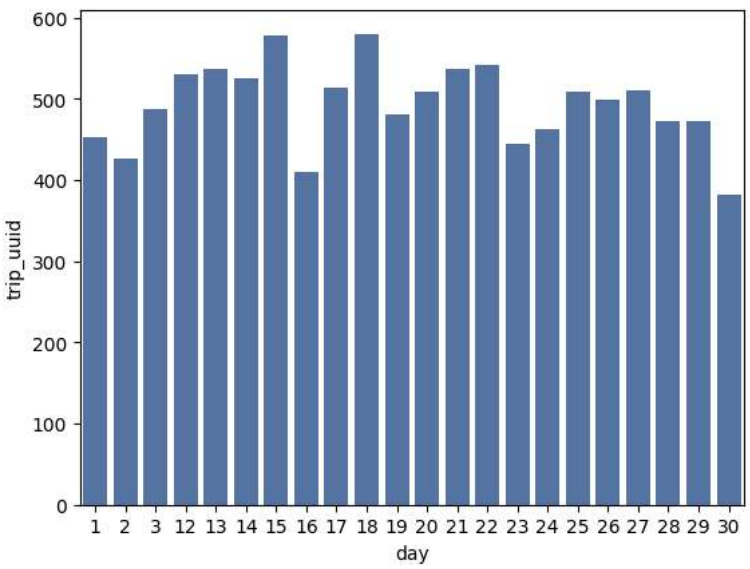

```
plt.figure(figsize=(12, 6))

# First subplot: Count plot
plt.subplot(121)
sns.countplot(data = df2, x='data')
plt.xticks(ticks=['training' , 'test'], labels=['Training Data', 'Test Data'])

# Second subplot: Pie chart
plt.subplot(122)
plt.pie(df2['data'].value_counts(), autopct='%1.1f%%', explode=[0.025, 0.025], labels=['Training Data', 'Test Data'])

plt.show()
```



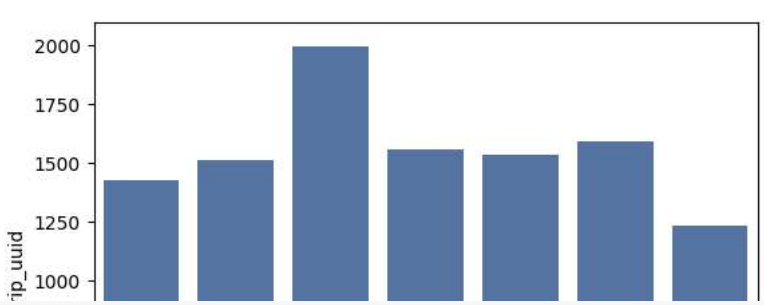
```
plt.figure(figsize=(12, 6))

# First subplot: Count plot
plt.subplot(121)
sns.countplot(data = df2, x='route_type')
plt.xticks(ticks=['Carting' , 'FTL'], labels=['Cart', 'Full Truck Load'])

# Second subplot: Pie chart
plt.subplot(122)
plt.pie(df2['route_type'].value_counts(), autopct='%1.1f%%', explode=[0.025, 0.025], labels=['Cart', 'Full Truck Load'])

plt.show()
```




```
df2_day = df2.groupby('day').agg({'trip_uuid' : 'nunique'}).reset_index().sort_values(by = 'day')
sns.barplot(data = df2_day, x = 'day', y= 'trip_uuid')
plt.show()
```




```
df2_dayname = df2.groupby('day_name').agg({'trip_uuid' : 'nunique'}).reset_index()
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
df2_dayname['day_name'] = pd.Categorical(df2_dayname['day_name'], categories=day_order, ordered=True)
sns.barplot(data = df2_dayname, x = 'day_name', y= 'trip_uuid')
plt.show()
```




```
df2_hour = df2.groupby('hour').agg({'trip_uuid' : 'nunique'}).reset_index().sort_values('hour')
sns.barplot(data = df2_hour, x = 'hour', y= 'trip_uuid')
plt.show()
```