"Delivering Excellence: A Logistics Case Study"

Avesh Raza Nagori

9082425683 aveshnagauri5@gmail.com (mailto:aveshnagauri5@gmail.com)

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

In [2]:
```
df = pd.read_csv("delhivery_data.txt")
```

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

Out[3]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388

5 rows × 24 columns

Converting time columns to datetime

In [4]:
```
df['od_start_time'] = pd.to_datetime(df['od_start_time'])
df['od_end_time'] = pd.to_datetime(df['od_end_time'])
```

In [5]:
```
df.head()
```

Out[5]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388

5 rows × 24 columns

In [6]:
```
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  datetime64[ns]
 10  od_end_time                     144867 non-null  datetime64[ns]
 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), datetime64[ns](2), float64(10), int64(1), object(10)
memory usage: 25.6+ MB
```

Grouping by sub-journey in the trip

In [7]:
```
df['segment_key'] = df['trip_uuid'] + df['source_center'] + df['destinatio

segment_cols = ['segment_actual_time', 'segment_osrm_time', 'segment_osrm_

for cols in segment_cols:
    df[cols + '_sum'] = df.groupby('segment_key')[cols].cumsum()

df[[cols + '_sum' for cols in segment_cols]]
```

Out[7]:

	segment_actual_time_sum	segment_osrm_time_sum	segment_osrm_distance_sum
0	14.0	11.0	11.9653
1	24.0	20.0	21.7243
2	40.0	27.0	32.5395
3	61.0	39.0	45.5619
4	67.0	44.0	49.4772
...	...	...	...
144862	92.0	94.0	65.3487
144863	118.0	115.0	82.7212
144864	138.0	149.0	103.4265
144865	155.0	176.0	122.3150
144866	423.0	185.0	131.1238

144867 rows × 3 columns

Aggregating at Sub-Journey Level

In [8]:
```
create_segment_dict = {

    'data' : 'first',
    'trip_creation_time': 'first',
    'route_schedule_uuid' : 'first',
    'route_type' : 'first',
    'trip_uuid' : 'first',
    'source_center' : 'first',
    'source_name' : 'first',

    'destination_center' : 'last',
    'destination_name' : 'last',

    'od_start_time' : 'first',
    'od_end_time' : 'first',
    'start_scan_to_end_scan' : 'first',


    'actual_distance_to_destination' : 'last',
    'actual_time' : 'last',

    'osrm_time' : 'last',
    'osrm_distance' : 'last',

    'segment_actual_time_sum' : 'last',
    'segment_osrm_distance_sum' : 'last',
    'segment_osrm_time_sum' : 'last',

    }
```

In [9]:
```
# Groupby mini-trips, sorting by time

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

In [10]:
```
segment.head()
```

Out[10]:

	index	segment_key	data	trip_creation_time	rou
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	thanos::
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	thanos::
2	2	trip-153671042288605164IND561203AABIND562101AAA	training	2018-09-12 00:00:22.886430	thanos::
3	3	trip-153671042288605164IND572101AAAIND561203AAB	training	2018-09-12 00:00:22.886430	thanos::
4	4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-12 00:00:33.691250	thanos::

5 rows × 21 columns

In [11]:
```
segment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26368 entries, 0 to 26367
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   index                           26368 non-null  int64
 1   segment_key                     26368 non-null  object
 2   data                            26368 non-null  object
 3   trip_creation_time              26368 non-null  object
 4   route_schedule_uuid             26368 non-null  object
 5   route_type                      26368 non-null  object
 6   trip_uuid                       26368 non-null  object
 7   source_center                   26368 non-null  object
 8   source_name                     26302 non-null  object
 9   destination_center              26368 non-null  object
 10  destination_name                26287 non-null  object
 11  od_start_time                   26368 non-null  datetime64[ns]
 12  od_end_time                     26368 non-null  datetime64[ns]
 13  start_scan_to_end_scan          26368 non-null  float64
 14  actual_distance_to_destination  26368 non-null  float64
 15  actual_time                     26368 non-null  float64
 16  osrm_time                       26368 non-null  float64
 17  osrm_distance                   26368 non-null  float64
 18  segment_actual_time_sum         26368 non-null  float64
 19  segment_osrm_distance_sum       26368 non-null  float64
 20  segment_osrm_time_sum           26368 non-null  float64
dtypes: datetime64[ns](2), float64(8), int64(1), object(10)
memory usage: 4.2+ MB
```

Calculating time taken between od_start_time and od_end_time and keep it as a feature

In [12]:
```
segment['od_time_diff_hour'] = (segment['od_end_time'] - segment['od_start
```

In [13]:
```
segment['od_time_diff_hour'].head()
```

Out[13]:
```
0    21.010074
1    16.658423
2     0.980540
3     2.046325
4    13.910649
Name: od_time_diff_hour, dtype: float64
```

In [14]:
```
segment.head()
```

Out[14]:

	index	segment_key	data	trip_creation_time	rout
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	thanos::
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	thanos::
2	2	trip-153671042288605164IND561203AABIND562101AAA	training	2018-09-12 00:00:22.886430	thanos::
3	3	trip-153671042288605164IND572101AAAIND561203AAB	training	2018-09-12 00:00:22.886430	thanos::
4	4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-12 00:00:33.691250	thanos::

5 rows × 22 columns

In [ ]:

In [15]:
```
segment.describe()
```

Out[15]:

	index	od_start_time	od_end_time	start_scan_to_end_scan	actual_
count	26368.000000	26368	26368	26368.000000	
mean	13183.500000	2018-09-22 18:35:33.012112128	2018-09-22 23:34:19.660814336	298.278671	
min	0.000000	2018-09-12 00:00:16.535741	2018-09-12 00:50:10.814399	20.000000	
25%	6591.750000	2018-09-17 08:36:26.495753472	2018-09-17 16:27:20.898079744	91.000000	
50%	13183.500000	2018-09-22 08:33:44.414494720	2018-09-22 16:37:58.917223936	152.000000	
75%	19775.250000	2018-09-28 00:13:59.749550848	2018-09-28 03:42:07.161700864	307.000000	
max	26367.000000	2018-10-06 04:27:23.392375	2018-10-08 03:00:24.353479	7898.000000	
std	7611.930285	NaN	NaN	440.561588	

In [16]:
```
segment.isnull().sum()
```

Out[16]:
```
index                              0
segment_key                        0
data                               0
trip_creation_time                 0
route_schedule_uuid                0
route_type                         0
trip_uuid                          0
source_center                      0
source_name                       66
destination_center                 0
destination_name                  81
od_start_time                      0
od_end_time                        0
start_scan_to_end_scan             0
actual_distance_to_destination     0
actual_time                        0
osrm_time                          0
osrm_distance                      0
segment_actual_time_sum            0
segment_osrm_distance_sum          0
segment_osrm_time_sum              0
od_time_diff_hour                  0
dtype: int64
```

Insight:

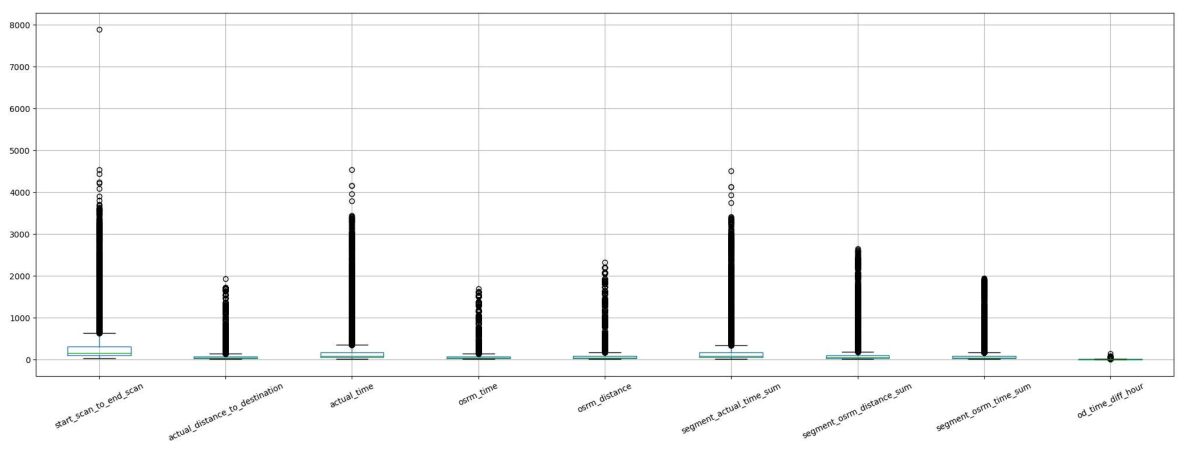
There are 66 null values in source_name and 81 null values in destination name.

In [17]:
```
segment.nunique()
```

Out[17]:
```
index                             26368
segment_key                       26368
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                     26368
od_end_time                       26368
start_scan_to_end_scan             1915
actual_distance_to_destination    26339
actual_time                        1658
osrm_time                           560
osrm_distance                     26015
segment_actual_time_sum            1676
segment_osrm_distance_sum         26093
segment_osrm_time_sum              1102
od_time_diff_hour                 26368
dtype: int64
```

In [18]:
```
# Boxplot for contionous variables

num_cols = ['start_scan_to_end_scan', 'actual_distance_to_destination', 'a
            'osrm_distance', 'segment_actual_time_sum', 'segment_osrm_dist
            'segment_osrm_time_sum', 'od_time_diff_hour']

segment[num_cols].boxplot(rot=25, figsize=(25,8))
```

Out[18]: <Axes: >



Insight:

Outliers can be seen in every columns

Using IQR method to solve the problem of Outliers

In [19]:
```
Q1 = segment[num_cols].quantile(0.25)
Q3 = segment[num_cols].quantile(0.75)

IQR = Q3-Q1
```

In [38]:
```
lower_bound = Q1 - 1.5* IQR
upper_bound = Q3 + 1.5* IQR

trip = segment[~((segment[num_cols]<lower_bound) | (segment[num_cols]<uppe
```

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

Out[39]:

	index	segment_key	data	trip_creation_time	ro
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	thanos
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	thanos
4	4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-12 00:00:33.691250	thanos
5	5	trip-153671043369099517IND562132AAAIND000000ACB	training	2018-09-12 00:00:33.691250	thanos
82	82	trip-153671321710455800IND421302AAGIND000000ACB	training	2018-09-12 00:46:57.104787	thanos

5 rows × 22 columns

Splitting destination_name and source_name to get state,city

In [40]:
```
trip['destination_name'].head()
```

Out[40]:
```
0           Gurgaon_Bilaspur_HB (Haryana)
1      Kanpur_Central_H_6 (Uttar Pradesh)
4         Chandigarh_Mehmdpur_H (Punjab)
5           Gurgaon_Bilaspur_HB (Haryana)
82          Gurgaon_Bilaspur_HB (Haryana)
Name: destination_name, dtype: object
```

In [41]:
```
trip['destination_name'] = trip['destination_name'].str.lower() #Lowering
trip['source_name'] = trip['source_name'].str.lower()
```

In [42]:
```
def place2state(x):
    # transform  "gurgaon_bilaspur_hb (haryana)" into "haryana)""
    state = x.split('(')[1]

    return state[:-1] #removing ')' from ending

def place2city(x):
    # We will remove state
    city = x.split(' (')[0]

    city = city.split('_')[0]

    #Now dealing with edge cases

    if city == 'pnq vadgaon sheri dpc':
      return 'vadgaonsheri'

    # ['PNQ Pashan DPC', 'Bhopal MP Nagar', 'HBR Layout PC',
    #  'PNQ Rahatani DPC', 'Pune Balaji Nagar', 'Mumbai Antop Hill']

    if city in ['pnq pashan dpc','pnq rahatani dpc', 'pune balaji nagar']:
        return 'pune'

    if city == 'hbr layout pc' : return 'bengaluru'
    if city == 'bhopal mp nagar' : return 'bhopal'
    if city == 'mumbai antop hill' : return 'mumbai'


    return city

def place2city_place(x):

    # We will remove state
    x = x.split(' (')[0]

    len_ = len(x.split('_'))

    if len_ >= 3:
        return x.split('_')[1]

    # Small cities have same city and place name
    if len_ == 2:
        return x.split('_')[0]


    # Now we need to deal with edge cases or imporper name convention

    #if len(x.split(' ')) == 2:
    #

    return x.split(' ')[0]


def place2code(x):
    # We will remove state
    x = x.split(' (')[0]
```

```
    if len(x.split('_')) >= 3 :
        return x.split('_')[-1]

    return 'none'
```

In [43]:
```
trip['destination_state'] = trip['destination_name'].apply(lambda x: place
trip['destination_city']  = trip['destination_name'].apply(lambda x: place
trip['destination_place'] = trip['destination_name'].apply(lambda x: place
trip['destination_code']  = trip['destination_name'].apply(lambda x: place
```

In [44]:
```
trip[['destination_state', 'destination_city', 'destination_place', 'desti
```

Out[44]:

	destination_state	destination_city	destination_place	destination_code
0	haryana	gurgaon	bilaspur	hb
1	uttar pradesh	kanpur	central	6
4	punjab	chandigarh	mehmdpur	h
5	haryana	gurgaon	bilaspur	hb
82	haryana	gurgaon	bilaspur	hb
...	...	...	...	...
26222	haryana	sonipat	kundli	h
26255	maharashtra	bhiwandi	mankoli	hb
26265	punjab	chandigarh	mehmdpur	h
26266	haryana	gurgaon	bilaspur	hb
26333	uttar pradesh	kanpur	central	6

1824 rows × 4 columns

In [45]:
```
trip['source_state'] = trip['source_name'].apply(lambda x: place2state(x))
trip['source_city']  = trip['source_name'].apply(lambda x: place2city(x))
trip['source_place'] = trip['source_name'].apply(lambda x: place2city_plac
trip['source_code']  = trip['source_name'].apply(lambda x: place2code(x))
```

In [46]:
```
trip[['source_state', 'source_city', 'source_place', 'source_code']]
```

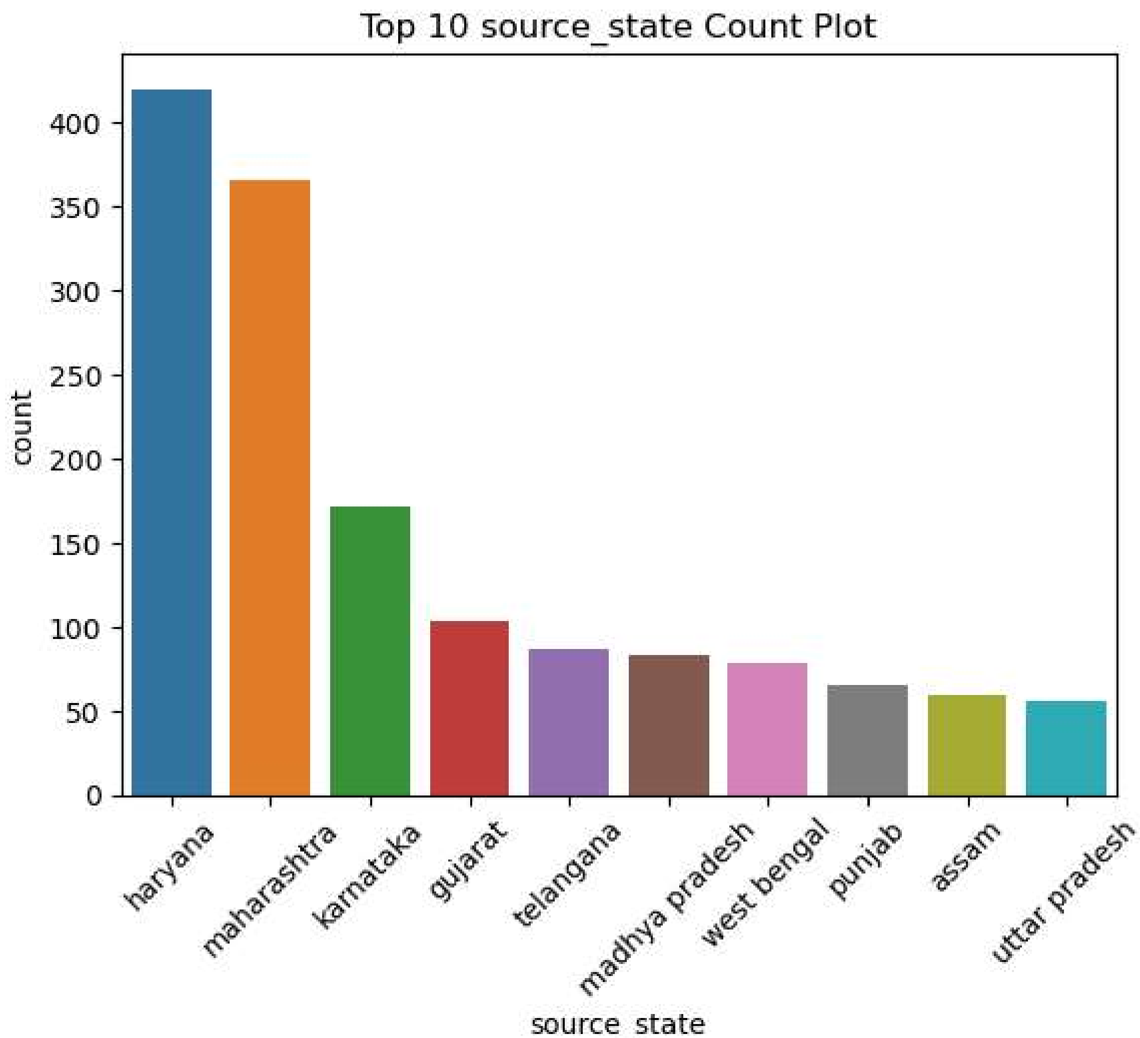
Out[46]:

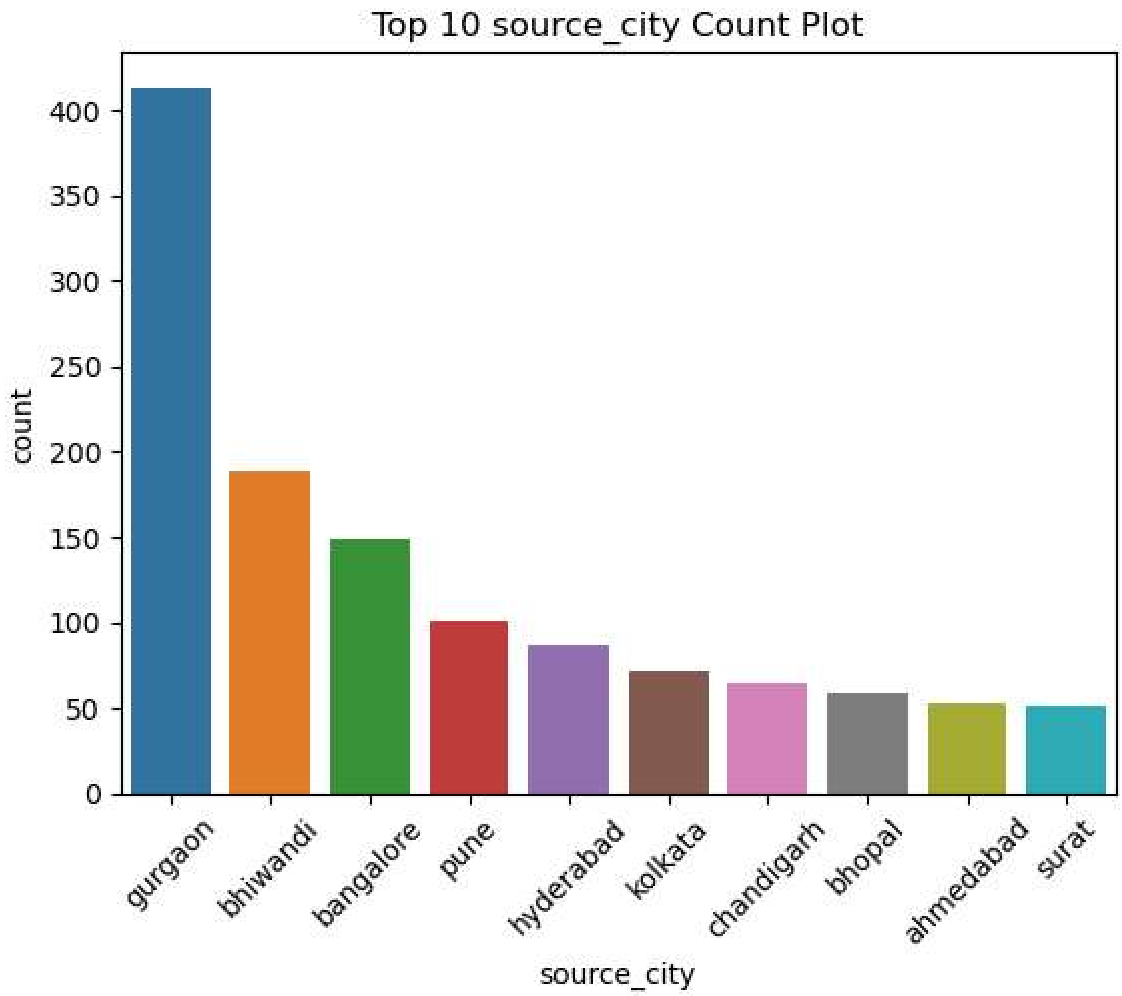
	source_state	source_city	source_place	source_code
0	uttar pradesh	kanpur	central	6
1	madhya pradesh	bhopal	trnsport	h
4	haryana	gurgaon	bilaspur	hb
5	karnataka	bangalore	nelmngla	h
82	maharashtra	bhiwandi	mankoli	hb
...	...	...	...	...
26222	maharashtra	bhiwandi	mankoli	hb
26255	maharashtra	akola	gaurkshn	i
26265	haryana	gurgaon	bilaspur	hb
26266	karnataka	bangalore	nelmngla	h
26333	madhya pradesh	bhopal	trnsport	h

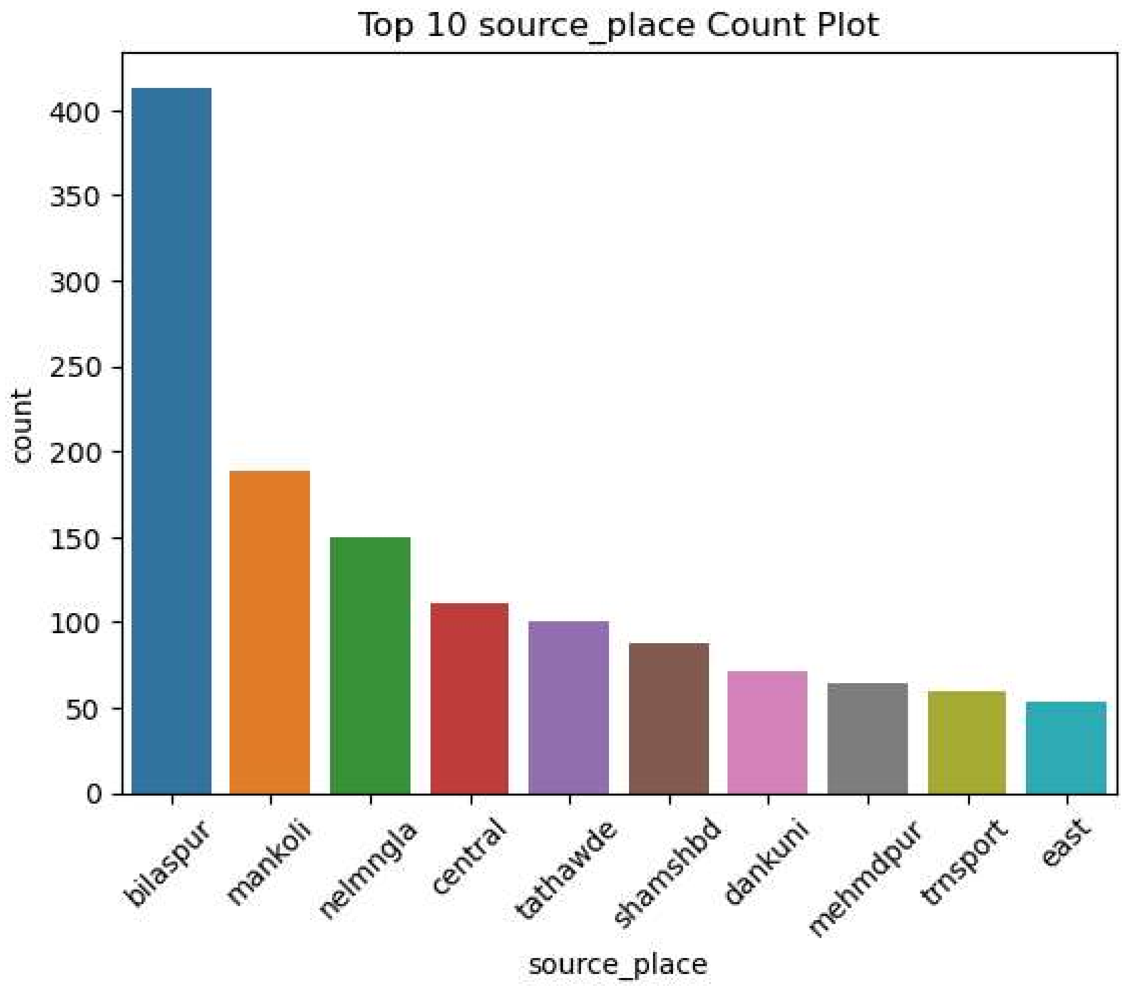
1824 rows × 4 columns

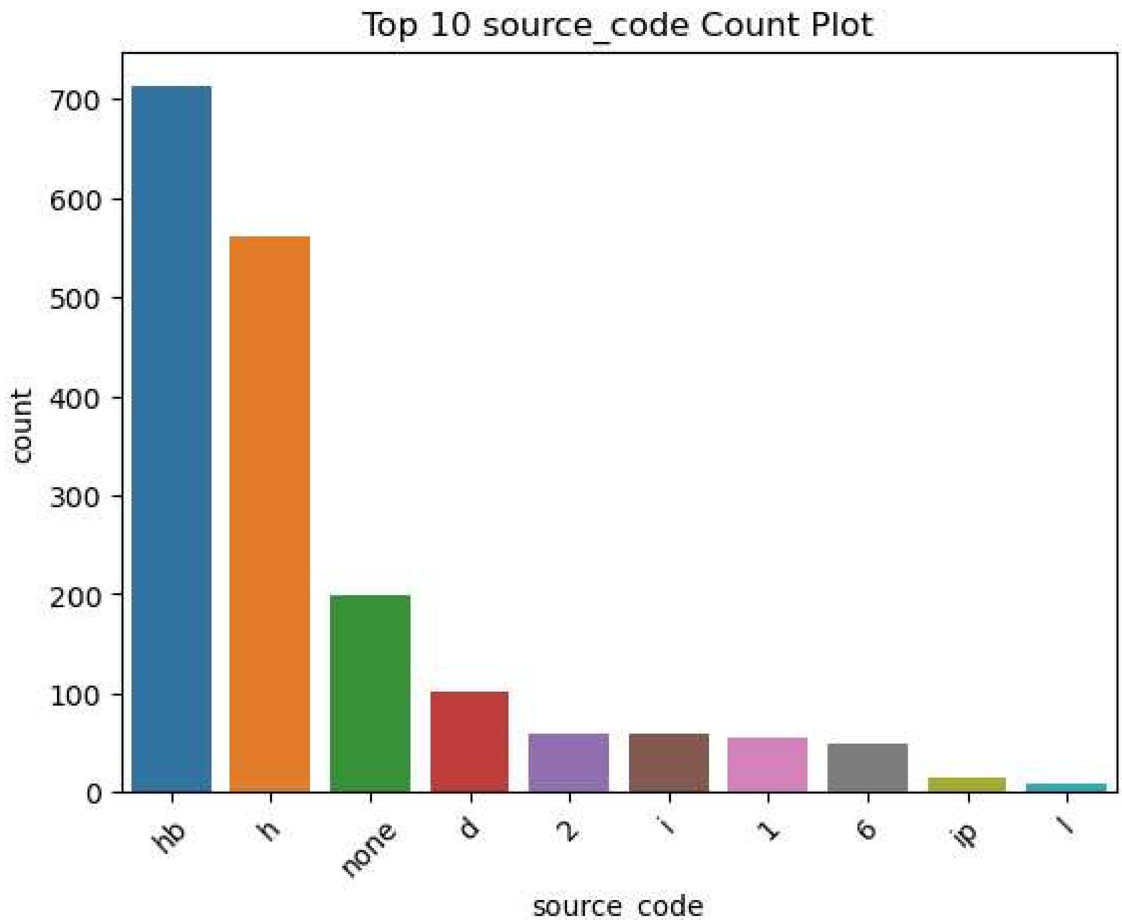
In [47]:

```
arr = ['source_state', 'source_city', 'source_place', 'source_code']

for col in arr:
    top_10_values = trip[col].value_counts().nlargest(10).index

    filtered_trip = trip[trip[col].isin(top_10_values)]

    sns.countplot(data=filtered_trip, x=col, order=top_10_values)

    # Display the plot
    plt.xticks(rotation=45)
    plt.title(f'Top 10 {col} Count Plot')
    plt.show()
```




Top 10 source_city Count Plot
count
0
50
100
150
200
250
300
350
400
gurgaon
bhiwandi
bangalore
pune
hyderabad
kolkata
chandigarh
bhopal
ahmedabad
surat
source_city


Top 10 source_place Count Plot
count
400
350
300
250
200
150
100
50
0
bilaspur
mankoli
nelmngla
central
tathawde
shamshbd
dankuni
mehmdpur
trnsport
east
source_place



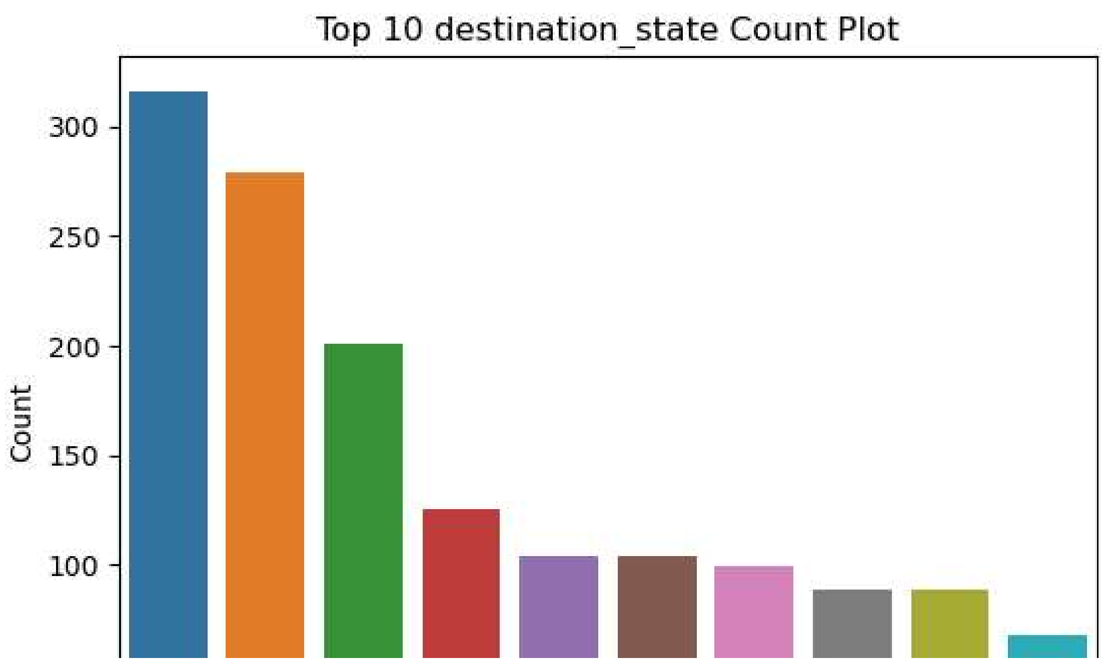
Insight:

1. The top states sending the most parcels are Haryana and Maharashtra, followed by Karnataka, Gujarat, and Madhya Pradesh. Haryana and Maharashtra significantly outpace the others.
2. Among the cities, Gurgaon has the highest number of parcels, followed by Bhiwandi. Pune, Bangalore, and Hyderabad are at similar levels.
3. Bilaspur stands out as the top place, with Mankoli in second place. Tathawde, Nelamangala, and Central are all at similar levels.
4. HB and H are the highest, with the rest trailing far behind.

Top destination state

In [49]:

```
arr = ['destination_state', 'destination_city', 'destination_place', 'dest

for col in arr:
    top_10_values = trip[col].value_counts().nlargest(10).index

    filtered_trip = trip[trip[col].isin(top_10_values)]

    sns.countplot(data=filtered_trip, x=col, order=top_10_values)

    plt.xticks(rotation=45)
    plt.title(f'Top 10 {col} Count Plot')
    plt.xlabel(col.replace('_', ' ').capitalize())
    plt.ylabel('Count')

    plt.show()
```



Insight:

1. Haryana and Maharashtra are the top states, with very little separating them. Karnataka is third, followed closely by Telangana and West Bengal. Surprisingly, Delhi, a metro city, is in the 10th position.
2. Gurgaon leads, with a 20% difference between Bangalore and the top. Hyderabad, Kolkata, and Pune are somewhat similar, followed by the rest.

3. Bilaspur is on top, slightly below Nelamangala. Central, Shamshabad, and Dankuni follow.
4. H and HB are the top two, with the rest trailing behind significantly.
5. A close insight is that the top two are doing significantly higher numbers, while the rest

In [51]:
```
create_trip_dict = {

    'data' : 'first',
    'trip_creation_time': 'first',
    'route_schedule_uuid' : 'first',
    'route_type' : 'first',
    'trip_uuid' : 'first',

    'source_center' : 'first',
    'source_name' : 'first',

    'destination_center' : 'last',
    'destination_name' : 'last',

    'start_scan_to_end_scan' : 'sum',
    'od_time_diff_hour' : 'sum',

    'actual_distance_to_destination' : 'sum',
    'actual_time' : 'sum',
    'osrm_time' : 'sum',
    'osrm_distance' : 'sum',

    'segment_actual_time_sum' : 'sum',
    'segment_osrm_distance_sum' : 'sum',
    'segment_osrm_time_sum' : 'sum',

    }
```

In [52]:
```
trip = segment.groupby('trip_uuid').agg(create_trip_dict).reset_index(drop
```

Performing paired sample t-test between actual_time and segment_actual_time_sum.

Null Hypothesis(H0) :-There is no difference in the time of actual_time and segment_actual_time_sum.

Alternate Hypothesis(H1):- There is Significant difference in the time of actual_time and segment_actual_time_sum.

In [53]:

```
t_statistic, p_value = stats.ttest_rel(trip['actual_time'], trip['segment_

print(f'T-statistic: {t_statistic}')
print(f'P-value: {p_value}')

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("We reject the null hypothesis. There is a significant differenc
else:
    print("We fail to reject the null hypothesis. There is no significant 
```

```
T-statistic: 58.993208505474314
P-value: 0.0
We reject the null hypothesis. There is a significant difference between
'actual_time' and 'segment_actual_time_sum'.
```

Performing paired sample t-test between actual_distance_to_destination and osrm_distance

Null Hypothesis(H0): There is no significant difference between actual_distance_to_destination and osrm_distance.

Alternate Hypothesis(H1): There is significant difference between actual_distance_to_destination and osrm_distance.

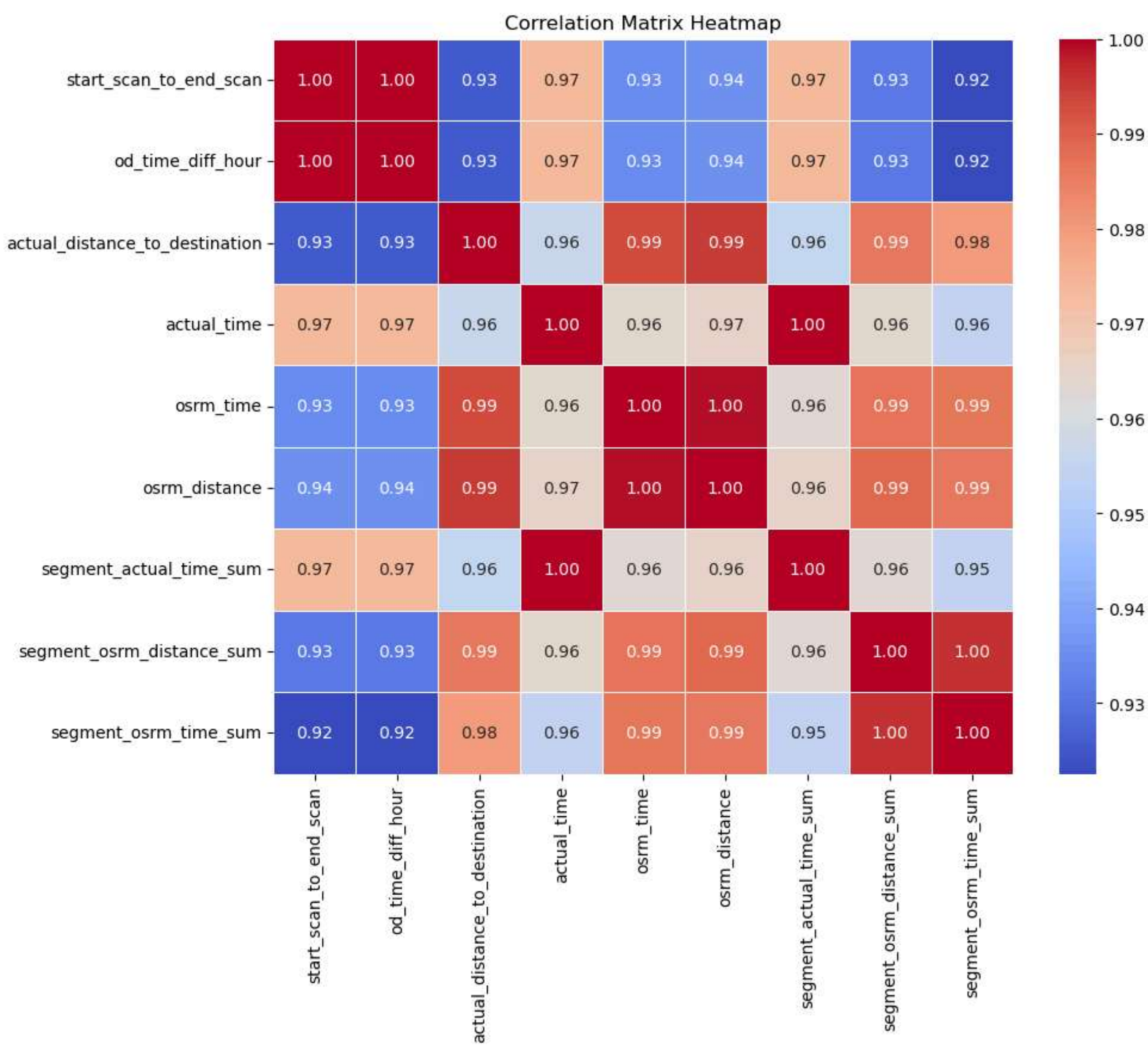
In [54]:

```
t_statistic, p_value = stats.ttest_rel(trip['actual_distance_to_destinatio

print(f'T-statistic: {t_statistic}')
print(f'P-value: {p_value}')

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("We reject the null hypothesis. There is a significant differenc
else:
    print("We fail to reject the null hypothesis. There is no significant 
```

```
T-statistic: -58.2845700210088
P-value: 0.0
We reject the null hypothesis. There is a significant difference between
'osrm_distance' and 'actual_distance_to_destination'.
```

Heatmap for numerical columns to detect relations

In [55]:
```
arr = trip.copy()
arr = arr.drop(columns=['data','trip_creation_time','route_schedule_uuid',
                        'trip_uuid','source_name','destination_name','sour
corr_matrix = arr.corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm', linewidth
plt.title('Correlation Matrix Heatmap')
plt.show()
```



Insights: All the columns in the heatmap exhibit a strong positive correlation.

One hot Encoding

In [58]:
```
trip['route_type'].value_counts()
```

Out[58]:
```
route_type
FTL        1547
Carting       7
Name: count, dtype: int64
```

In [61]:
```
trip['route_type'] = trip['route_type'].map({'FTL':0, 'Carting':1})
```

In [63]:
```
trip['route_type'].value_counts()
```

Out[63]:
```
route_type
0    1547
1       7
Name: count, dtype: int64
```

Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler

In [64]:
```
from sklearn.preprocessing import StandardScaler
```

In [65]:
```
scaler = StandardScaler()
scaler.fit(trip[num_cols])
```

Out[65]:
```
StandardScaler()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [66]:
```
trip[num_cols] = scaler.transform(trip[num_cols])
```

In [67]:
```
trip[num_cols]
```

Out[67]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_di
0	0.475565	0.019000	0.066208	-0.034561	-0.(
1	2.148394	2.125420	1.978349	2.205717	2.1
2	0.554510	0.503192	0.631816	0.515107	0.5
3	0.518535	0.571661	0.518266	0.578614	0.5
4	-0.764566	-0.573204	-0.760780	-0.553570	-0.5
...	...	...	...	...	
1549	0.267711	0.374262	0.133695	0.258887	0.3
1550	0.241729	0.638088	0.323303	0.666210	0.6
1551	-0.779556	-0.701278	-0.631161	-0.746282	-0.7
1552	2.625060	2.123323	1.935500	2.203528	2.1
1553	-0.567704	-0.712083	-0.539036	-0.755041	-0.7

1554 rows × 9 columns

In [68]:
```
trip[num_cols].describe()
```

Out[68]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	c
count	1.554000e+03	1.554000e+03	1.554000e+03	1.554000e+03	
mean	-1.188810e-16	2.194727e-16	1.051640e-16	9.144694e-18	
std	1.000322e+00	1.000322e+00	1.000322e+00	1.000322e+00	
min	-1.151295e+00	-1.296970e+00	-1.238547e+00	-1.282810e+00	-
25%	-8.602492e-01	-8.844447e-01	-8.529054e-01	-8.864360e-01	
50%	-2.549231e-01	-1.515042e-01	-2.755137e-01	-2.207032e-01	
75%	5.872370e-01	8.045721e-01	6.411894e-01	6.202224e-01	
max	6.110618e+00	2.530844e+00	5.104191e+00	2.845171e+00	

Recommendation:

1. There is a significant difference between Actual Time and ORSM Time Distance.
2. Adjustments are needed in these two areas to improve customer experience.
3. Revisit the information provided to the routing engine for trip planning. Verify with transporters to ensure the routing engine is configured for optimal results and check for any discrepancies.

In [ ]: