In [1]:
```
!jupyter nbconvert --to html /content/Delhivery.ipynb
```

```
[NbConvertApp] Converting notebook /content/Delhivery.ipynb to html
[NbConvertApp] WARNING | Alternative text is missing on 4 image(s).
[NbConvertApp] Writing 601911 bytes to /content/Delhivery.html
```

In [3]:
```
# impoting libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

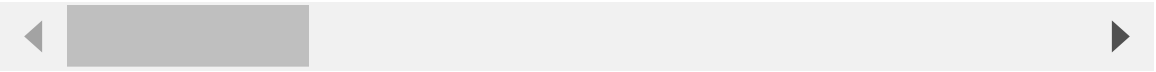
In [4]:
```
data = pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/0
```

In [ ]:
```
data
```

Out[ ]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355

144867 rows × 24 columns



In [6]:
```
# the number of coloumns in the dataframe is way too big to be displayed on the
# so below line tells pandas to display all columns.
pd.set_option('display.max_columns', None)
data.head()
```

Out[6]:

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

In [7]:
```
data.describe()
```

Out[7]:

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_ti
count	144867.000000	144867.000000	144867.000000	144867.0000
mean	961.262986	232.926567	234.073372	416.9275
std	1037.012769	344.755577	344.990009	598.1036
min	20.000000	9.000000	9.000045	9.0000
25%	161.000000	22.000000	23.355874	51.0000
50%	449.000000	66.000000	66.126571	132.0000
75%	1634.000000	286.000000	286.708875	513.0000
max	7898.000000	1927.000000	1927.447705	4532.0000

In [8]:
```
# check for null values
data.isnull().sum()
```

Out[8]:

	0
data	0
trip_creation_time	0
route_schedule_uuid	0
route_type	0
trip_uuid	0
source_center	0
source_name	293
destination_center	0
destination_name	261
od_start_time	0
od_end_time	0
start_scan_to_end_scan	0
is_cutoff	0
cutoff_factor	0
cutoff_timestamp	0
actual_distance_to_destination	0
actual_time	0
osrm_time	0
osrm_distance	0
factor	0
segment_actual_time	0
segment_osrm_time	0
segment_osrm_distance	0
segment_factor	0

dtype: int64

In [9]:
```
# there are some missing values
print(data[['source_name', 'destination_name']].isnull().sum())
```

```
source_name         293
destination_name    261
dtype: int64
```

In [10]:
```
print(data[['source_name', 'destination_name']].isnull().mean() * 100)
```

```
source_name         0.202254
destination_name    0.180165
dtype: float64
```

The missing values are less the 1% so we are ignoring them as dropping them is unlikely to affect the final results.

In [11]:
```
data.columns
```

Out[11]:
```
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 [12]:
```
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 [13]:
```
# dropping unkown fields
data = data.drop(columns =['cutoff_factor', 'cutoff_timestamp', 'factor', 'segme
```

In [14]:
```
data.shape
```

Out[14]:
```
(144867, 20)
```

In [15]:
```
data
```

Out[15]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109364764
...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	15374606684355

144867 rows × 20 columns

In [16]:
```
data.sample()
```

Out[16]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_
129340	training	2018-09-12 04:59:53.071019	thanos::sroute:396d96a3-e2f8-4c40-af0e-056e11f...	FTL	1536728393070

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

Out[17]:

	0
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
actual_distance_to_destination	144515
actual_time	3182
osrm_time	1531
osrm_distance	138046
segment_actual_time	747
segment_osrm_time	214
segment_osrm_distance	113799

dtype: int64

In [18]:
```
# converting the datatypes to category for columns which have only 2 values.
data['data'] = data['data'].astype('category')
data['route_type'] = data['route_type'].astype('category')


#converting time columns to datetime format
datetime_cols = ['trip_creation_time', 'od_start_time', 'od_end_time']
for _ in datetime_cols:
    data[_] = pd.to_datetime(data[_])
```

In [19]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 20 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144867 non-null  category
 1   trip_creation_time              144867 non-null  datetime64[ns]
 2   route_schedule_uuid             144867 non-null  object
 3   route_type                      144867 non-null  category
 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  actual_distance_to_destination  144867 non-null  float64
 14  actual_time                     144867 non-null  float64
 15  osrm_time                       144867 non-null  float64
 16  osrm_distance                   144867 non-null  float64
 17  segment_actual_time             144867 non-null  float64
 18  segment_osrm_time               144867 non-null  float64
 19  segment_osrm_distance           144867 non-null  float64
dtypes: bool(1), category(2), datetime64[ns](3), float64(8), object(6)
memory usage: 19.2+ MB
```

In [20]:
```
# as we are also handling with data and time,
# we must calculate the time period of this data being taken from
```

In [21]:
```
# span of the time between the first and last trip created
data['trip_creation_time'].max(), data['trip_creation_time'].min() , data['trip_
```

Out[21]:
```
(Timestamp('2018-10-03 23:59:42.701692'),
 Timestamp('2018-09-12 00:00:16.535741'),
 Timedelta('21 days 23:59:26.165951'))
```

In [22]:
```
# span of the time between the first and last trip ending
data['od_end_time'].max(), data['od_end_time'].min(), data['od_end_time'].max()
```

Out[22]:
```
(Timestamp('2018-10-08 03:00:24.353479'),
 Timestamp('2018-09-12 00:50:10.814399'),
 Timedelta('26 days 02:10:13.539080'))
```

In [23]:
```
# total time period of the datapoints
total_time = data['od_end_time'].max() - data['trip_creation_time'].min()
total_time
```

Out[23]:
```
Timedelta('26 days 03:00:07.817738')
```

In [24]:
```
# from the above sample data extract features like Month year and day from the t

data['Month'] = data['trip_creation_time'].dt.month
data['Year'] = data['trip_creation_time'].dt.year
data['Day'] = data['trip_creation_time'].dt.day

print(data[['trip_creation_time', 'Month', 'Year', 'Day']].head(23))
# data
```

	trip_creation_time	Month	Year	Day
0	2018-09-20 02:35:36.476840	9	2018	20
1	2018-09-20 02:35:36.476840	9	2018	20
2	2018-09-20 02:35:36.476840	9	2018	20
3	2018-09-20 02:35:36.476840	9	2018	20
4	2018-09-20 02:35:36.476840	9	2018	20
5	2018-09-20 02:35:36.476840	9	2018	20
6	2018-09-20 02:35:36.476840	9	2018	20
7	2018-09-20 02:35:36.476840	9	2018	20
8	2018-09-20 02:35:36.476840	9	2018	20
9	2018-09-20 02:35:36.476840	9	2018	20
10	2018-09-23 06:42:06.021680	9	2018	23
11	2018-09-23 06:42:06.021680	9	2018	23
12	2018-09-23 06:42:06.021680	9	2018	23
13	2018-09-23 06:42:06.021680	9	2018	23
14	2018-09-23 06:42:06.021680	9	2018	23
15	2018-09-14 15:42:46.437249	9	2018	14
16	2018-09-14 15:42:46.437249	9	2018	14
17	2018-09-13 20:44:19.424489	9	2018	13
18	2018-09-13 20:44:19.424489	9	2018	13
19	2018-09-13 20:44:19.424489	9	2018	13
20	2018-09-13 20:44:19.424489	9	2018	13
21	2018-09-13 20:44:19.424489	9	2018	13
22	2018-09-13 20:44:19.424489	9	2018	13

The given data set consists of 26 days of data points

Now let's find out how many cities, states are involved in this

In [25]:
```
import re

def extract_location(source_name):
  pattern = r"(.+?)\((.+?)\)"

  match = re.search(pattern, source_name)
  if match:
    state = match.group(1)
    city = match.group(2)
    return state, city
  else:
    return None, None
```

In [26]:
```
# spliting and extracting city and state from the source column
data[['source_state', 'source_city']] = data['source_name'].apply(lambda x: pd.S

print(data[['source_name', 'source_state', 'source_city']])
```

```
                        source_name       source_state source_city
0       Anand_VUNagar_DC (Gujarat)  Anand_VUNagar_DC      Gujarat
1       Anand_VUNagar_DC (Gujarat)  Anand_VUNagar_DC      Gujarat
2       Anand_VUNagar_DC (Gujarat)  Anand_VUNagar_DC      Gujarat
3       Anand_VUNagar_DC (Gujarat)  Anand_VUNagar_DC      Gujarat
4       Anand_VUNagar_DC (Gujarat)  Anand_VUNagar_DC      Gujarat
...                            ...                ...         ...
144862  Sonipat_Kundli_H (Haryana)  Sonipat_Kundli_H      Haryana
144863  Sonipat_Kundli_H (Haryana)  Sonipat_Kundli_H      Haryana
144864  Sonipat_Kundli_H (Haryana)  Sonipat_Kundli_H      Haryana
144865  Sonipat_Kundli_H (Haryana)  Sonipat_Kundli_H      Haryana
144866  Sonipat_Kundli_H (Haryana)  Sonipat_Kundli_H      Haryana

[144867 rows x 3 columns]
```

In [27]:
```
data[['destination_state', 'destination_city']] = data['destination_name'].apply

print(data[['destination_name', 'destination_state', 'destination_city']])
```

```
                   destination_name     destination_state destination_city
0       Khambhat_MotvdDPP_D (Gujarat)  Khambhat_MotvdDPP_D           Gujarat
1       Khambhat_MotvdDPP_D (Gujarat)  Khambhat_MotvdDPP_D           Gujarat
2       Khambhat_MotvdDPP_D (Gujarat)  Khambhat_MotvdDPP_D           Gujarat
3       Khambhat_MotvdDPP_D (Gujarat)  Khambhat_MotvdDPP_D           Gujarat
4       Khambhat_MotvdDPP_D (Gujarat)  Khambhat_MotvdDPP_D           Gujarat
...                               ...                ...             ...
144862  Gurgaon_Bilaspur_HB (Haryana)  Gurgaon_Bilaspur_HB           Haryana
144863  Gurgaon_Bilaspur_HB (Haryana)  Gurgaon_Bilaspur_HB           Haryana
144864  Gurgaon_Bilaspur_HB (Haryana)  Gurgaon_Bilaspur_HB           Haryana
144865  Gurgaon_Bilaspur_HB (Haryana)  Gurgaon_Bilaspur_HB           Haryana
144866  Gurgaon_Bilaspur_HB (Haryana)  Gurgaon_Bilaspur_HB           Haryana

[144867 rows x 3 columns]
```

In [33]:
```
# np.set_printoptions(threshold=np.inf)
unique_sources = data['source_name'].unique()
print(unique_sources[0:20])
```

```
['Anand_VUNagar_DC (Gujarat)' 'Khambhat_MotvdDPP_D (Gujarat)'
 'Bhiwandi_Mankoli_HB (Maharashtra)' 'LowerParel_CP (Maharashtra)'
 'Bangalore_Nelmngla_H (Karnataka)' 'Bengaluru_Bomsndra_HB (Karnataka)'
 'Ludhiana_GillChwk_DC (Punjab)' 'Jagraon_DC (Punjab)'
 'Raikot_DC (Punjab)' 'Junagadh_DPC (Gujarat)' 'Veraval_DC (Gujarat)'
 'Kodinar_NCplxDPP_D (Gujarat)' 'Una_Mamlatdr_DC (Gujarat)'
 'Talala_SsnRdDPP_D (Gujarat)' 'Sonipat_Kundli_H (Haryana)'
 'Roorkee_IOTCEncl_L (Uttarakhand)' 'Haridwar (Uttarakhand)'
 'MAA_Poonamallee_HB (Tamil Nadu)' 'Ludhiana_MilrGanj_HB (Punjab)'
 'Jalandhar_DPC (Punjab)']
```

In [34]:
```
data['source_name'].nunique()
```

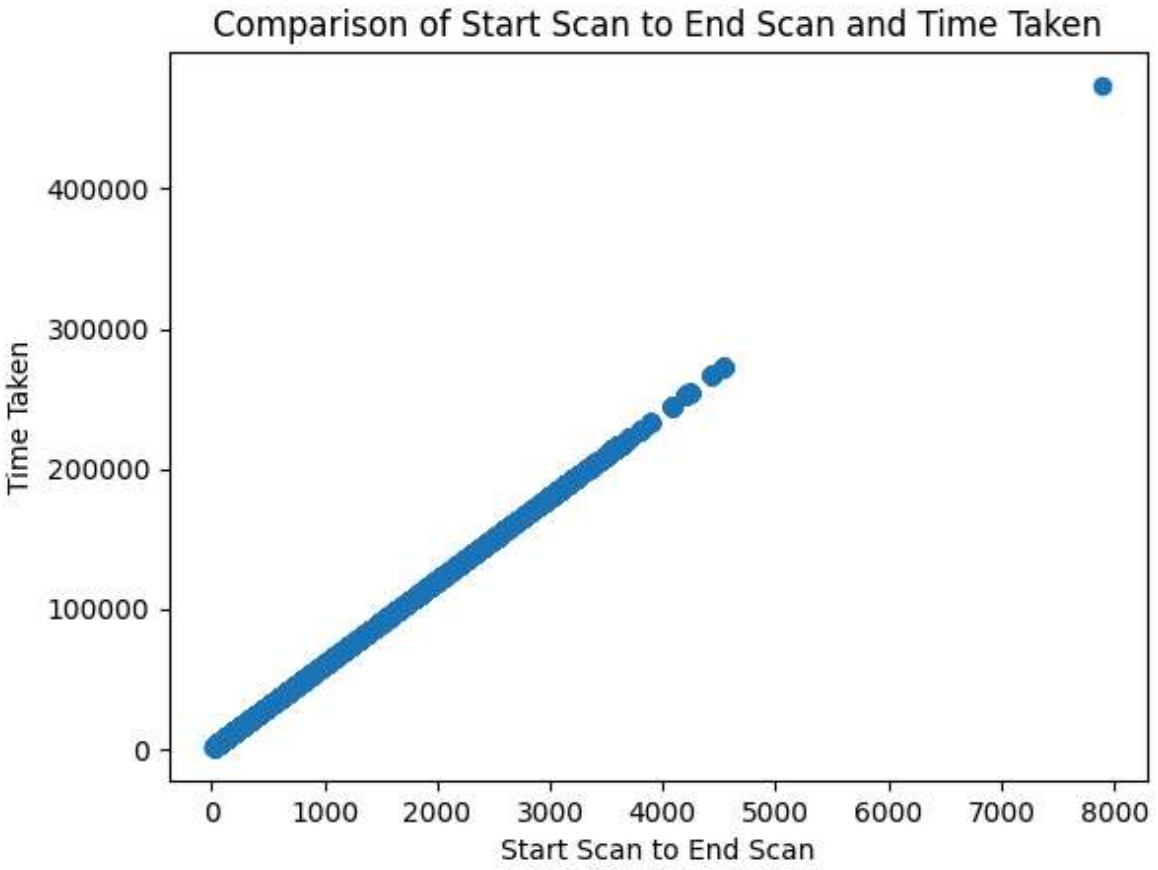
Out[34]: 1498

In [35]:
```
data['time_taken'] = (data['od_end_time'] - data['od_start_time']).dt.total_secc
```

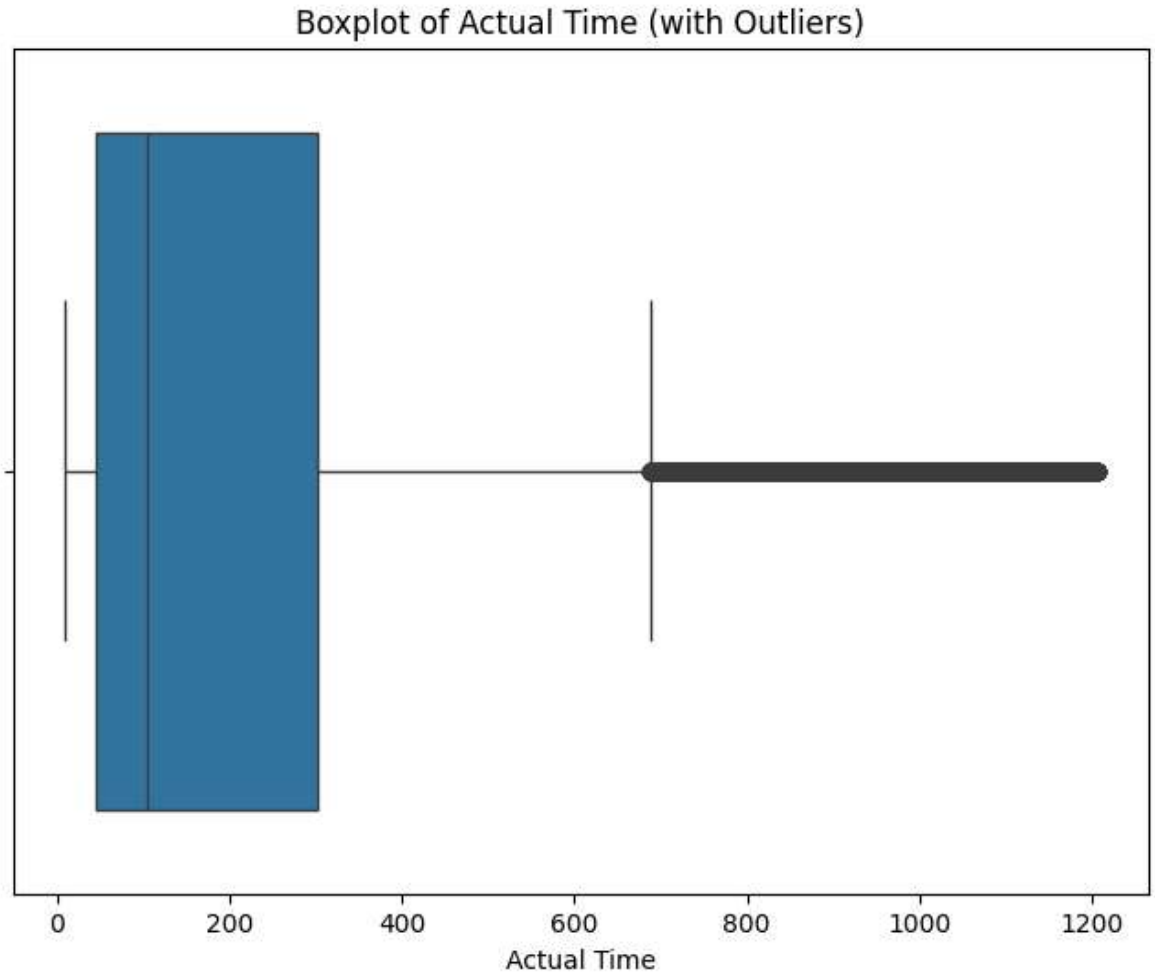
In [36]:
```
import scipy.stats as stats

# Hypothesis testing
t_stat, p_value = stats.ttest_rel(data['start_scan_to_end_scan'], data['time_tak
print(f'T-statistic: {t_stat}, P-value: {p_value}')
```

```
# Scatter plot for visual analysis
import matplotlib.pyplot as plt
plt.scatter(data['start_scan_to_end_scan'], data['time_taken'])
plt.xlabel('Start Scan to End Scan')
plt.ylabel('Time Taken')
plt.title('Comparison of Start Scan to End Scan and Time Taken')
plt.show()
```

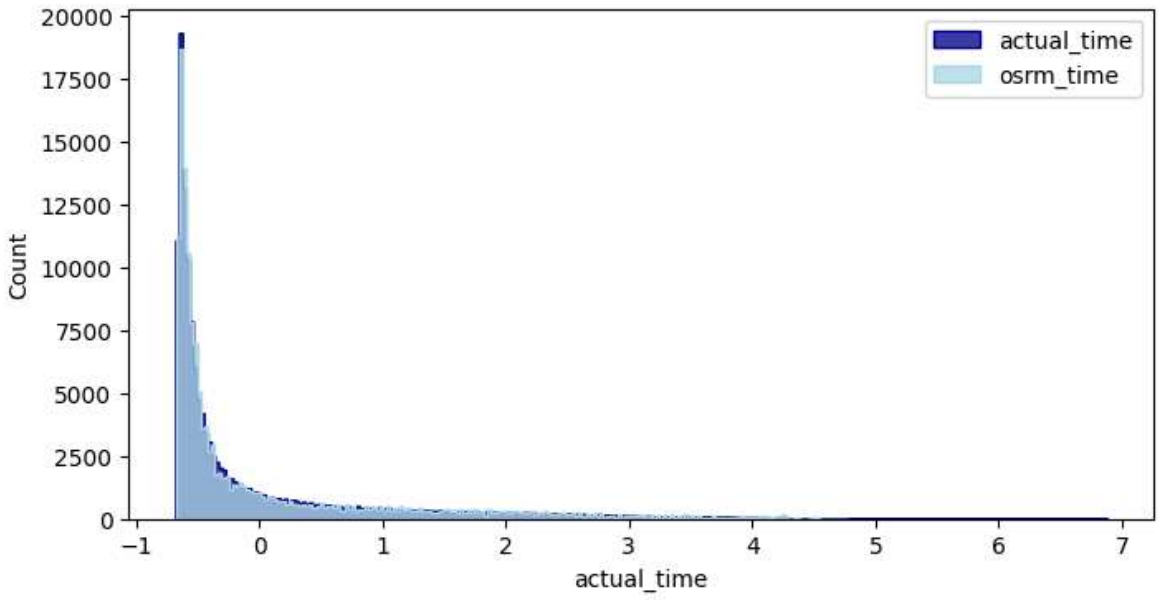
```
T-statistic: -352.9967867233228, P-value: 0.0
```



In [37]:
```
# Detecting outliers using IQR
Q1 = data['actual_time'].quantile(0.25)
Q3 = data['actual_time'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out the outliers
data_cleaned = data[(data['actual_time'] >= lower_bound) & (data['actual_time']

plt.figure(figsize=(8, 6))
sns.boxplot(data=data_cleaned, x='actual_time')
plt.title('Boxplot of Actual Time (with Outliers)')
plt.xlabel('Actual Time')
plt.show()
```

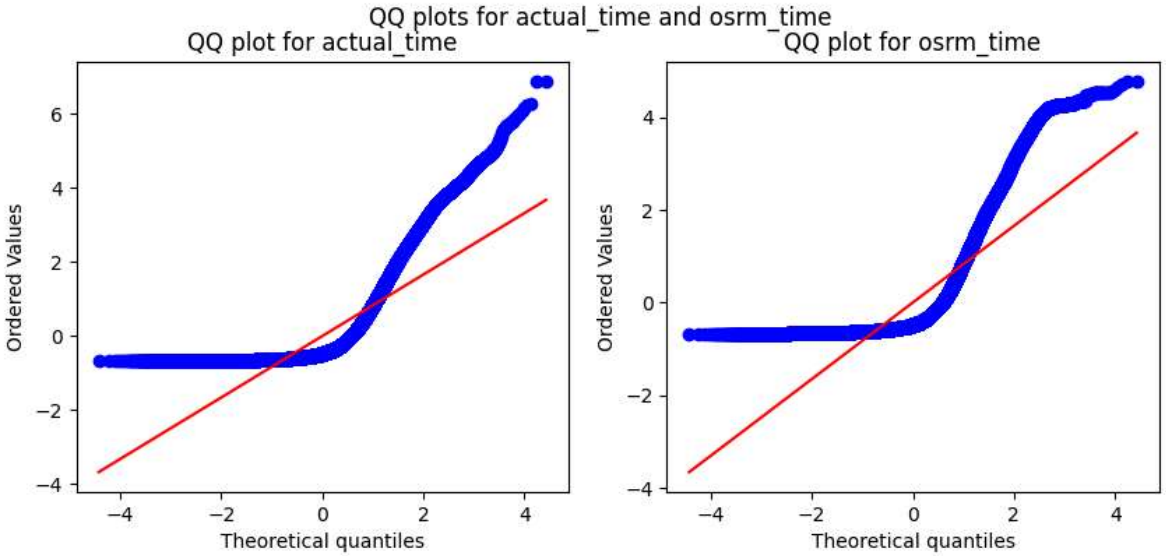


In [39]:
```
plt.figure(figsize = (8, 4))
sns.histplot(data['actual_time'], element = 'step', color = 'darkblue')
sns.histplot(data['osrm_time'], element = 'step', color = 'lightblue')
plt.legend(['actual_time', 'osrm_time'])
plt.show()
```



In [40]:
```
# using QQ plot

from scipy import stats
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.suptitle('QQ plots for actual_time and osrm_time')
stats.probplot(data['actual_time'], plot=plt.subplot(1, 2, 1), dist='norm', fit=

plt.title('QQ plot for actual_time')
plt.subplot(1, 2, 2)
stats.probplot(data['osrm_time'], plot=plt.subplot(1, 2, 2), dist='norm', fit=Tr
plt.title('QQ plot for osrm_time')
plt.show()
```



In [41]:
```
import scipy.stats as spy

test_stat, p_value = spy.shapiro(data['actual_time'].sample(3000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')


test_stat, p_value = spy.shapiro(data['osrm_time'].sample(5000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 2.5471659828998023e-59
The sample does not follow normal distribution
p-value 1.1972760110924658e-70
The sample does not follow normal distribution
```