Delhivery_project

April 5, 2024

0.0.1 Importing required libraries

[101]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

0.1 Problem Statement:

Delhivery, a leading logistics company in India, aims to optimize its operations and enhance efficiency and profitability. The company collects extensive data on its delivery processes, including trip details, transportation types, distances, and time metrics. However, the raw data needs to be cleaned, processed, and analyzed to extract meaningful insights and build predictive models for improving operational performance.

0.1.1 Loading the dataset

[102]:
```
df=pd.read_csv('delhivery_data.csv')
```

[103]:
```
df
```

[103]:
```
            data           trip_creation_time  \
0       training  2018-09-20 02:35:36.476840
1       training  2018-09-20 02:35:36.476840
2       training  2018-09-20 02:35:36.476840
3       training  2018-09-20 02:35:36.476840
4       training  2018-09-20 02:35:36.476840
…            …                         …
144862  training  2018-09-20 16:24:28.436231
144863  training  2018-09-20 16:24:28.436231
144864  training  2018-09-20 16:24:28.436231
144865  training  2018-09-20 16:24:28.436231
144866  training  2018-09-20 16:24:28.436231

                                     route_schedule_uuid route_type  \
0       thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1       thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
```

```
2       thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3       thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4       thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
…                                                     …          …
144862  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5…    Carting
144863  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5…    Carting
144864  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5…    Carting
144865  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5…    Carting
144866  thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5…    Carting

                      trip_uuid source_center                 source_name  \
0       trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1       trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2       trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3       trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4       trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
…                           …           …                         …
144862  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)
144863  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)
144864  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)
144865  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)
144866  trip-153746066843555182  IND131028AAB  Sonipat_Kundli_H (Haryana)

       destination_center               destination_name  \
0            IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
1            IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
2            IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
3            IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
4            IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
…                     …                             …
144862       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
144863       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
144864       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
144865       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
144866       IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)

                     od_start_time  …             cutoff_timestamp  \
0       2018-09-20 03:21:32.418600  …          2018-09-20 04:27:55
1       2018-09-20 03:21:32.418600  …          2018-09-20 04:17:55
2       2018-09-20 03:21:32.418600  …   2018-09-20 04:01:19.505586
3       2018-09-20 03:21:32.418600  …          2018-09-20 03:39:57
4       2018-09-20 03:21:32.418600  …          2018-09-20 03:33:55
…                              …  …                          …
144862  2018-09-20 16:24:28.436231  …          2018-09-20 21:57:20
144863  2018-09-20 16:24:28.436231  …          2018-09-20 21:31:18
144864  2018-09-20 16:24:28.436231  …          2018-09-20 21:11:18
144865  2018-09-20 16:24:28.436231  …          2018-09-20 20:53:19
```

```
144866  2018-09-20 16:24:28.436231  …  2018-09-20 16:24:28.436231

        actual_distance_to_destination  actual_time  osrm_time osrm_distance  \
0                            10.435660         14.0       11.0       11.9653
1                            18.936842         24.0       20.0       21.7243
2                            27.637279         40.0       28.0       32.5395
3                            36.118028         62.0       40.0       45.5620
4                            39.386040         68.0       44.0       54.2181
…                                  …          …         …           …
144862                       45.258278         94.0       60.0       67.9280
144863                       54.092531        120.0       76.0       85.6829
144864                       66.163591        140.0       88.0       97.0933
144865                       73.680667        158.0       98.0      111.2709
144866                       70.039010        426.0       95.0       88.7319

          factor  segment_actual_time  segment_osrm_time  \
0       1.272727                 14.0               11.0
1       1.200000                 10.0                9.0
2       1.428571                 16.0                7.0
3       1.550000                 21.0               12.0
4       1.545455                  6.0                5.0
…          …                  …                …
144862  1.566667                 12.0               12.0
144863  1.578947                 26.0               21.0
144864  1.590909                 20.0               34.0
144865  1.612245                 17.0               27.0
144866  4.484211                268.0                9.0

        segment_osrm_distance  segment_factor
0                     11.9653        1.272727
1                      9.7590        1.111111
2                     10.8152        2.285714
3                     13.0224        1.750000
4                      3.9153        1.200000
…                        …             …
144862                 8.1858        1.000000
144863                17.3725        1.238095
144864                20.7053        0.588235
144865                18.8885        0.629630
144866                 8.8088       29.777778

[144867 rows x 24 columns]
```

[104]:
```
df.head()
```

[104]:
```
       data          trip_creation_time  \
0  training  2018-09-20 02:35:36.476840
```

```
1  training  2018-09-20 02:35:36.476840
2  training  2018-09-20 02:35:36.476840
3  training  2018-09-20 02:35:36.476840
4  training  2018-09-20 02:35:36.476840

                                 route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                trip_uuid source_center                 source_name  \
0  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)

  destination_center              destination_name  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)

               od_start_time  …            cutoff_timestamp  \
0  2018-09-20 03:21:32.418600  …         2018-09-20 04:27:55
1  2018-09-20 03:21:32.418600  …         2018-09-20 04:17:55
2  2018-09-20 03:21:32.418600  …  2018-09-20 04:01:19.505586
3  2018-09-20 03:21:32.418600  …         2018-09-20 03:39:57
4  2018-09-20 03:21:32.418600  …         2018-09-20 03:33:55

   actual_distance_to_destination  actual_time  osrm_time osrm_distance  \
0                       10.435660         14.0       11.0       11.9653
1                       18.936842         24.0       20.0       21.7243
2                       27.637279         40.0       28.0       32.5395
3                       36.118028         62.0       40.0       45.5620
4                       39.386040         68.0       44.0       54.2181

     factor  segment_actual_time  segment_osrm_time  segment_osrm_distance  \
0  1.272727                 14.0               11.0                11.9653
1  1.200000                 10.0                9.0                 9.7590
2  1.428571                 16.0                7.0                10.8152
3  1.550000                 21.0               12.0                13.0224
4  1.545455                  6.0                5.0                 3.9153
```

```
   segment_factor
0        1.272727
1        1.111111
2        2.285714
3        1.750000
4        1.200000

[5 rows x 24 columns]
```

[105]:
```python
print("Dimensions of the dataset:", df.shape)
```

```
Dimensions of the dataset: (144867, 24)
```

[106]:
```python
print(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
None
```

[107]:
```
print(df.describe())
```

```
       start_scan_to_end_scan  cutoff_factor  actual_distance_to_destination  \
count           144867.000000  144867.000000                   144867.000000
mean               961.262986     232.926567                      234.073372
std               1037.012769     344.755577                      344.990009
min                 20.000000       9.000000                        9.000045
25%                161.000000      22.000000                       23.355874
50%                449.000000      66.000000                       66.126571
75%               1634.000000     286.000000                      286.708875
max               7898.000000    1927.000000                     1927.447705

         actual_time      osrm_time  osrm_distance         factor  \
count  144867.000000  144867.000000  144867.000000  144867.000000
mean      416.927527     213.868272     284.771297       2.120107
std       598.103621     308.011085     421.119294       1.715421
min         9.000000       6.000000       9.008200       0.144000
25%        51.000000      27.000000      29.914700       1.604264
50%       132.000000      64.000000      78.525800       1.857143
75%       513.000000     257.000000     343.193250       2.213483
max      4532.000000    1686.000000    2326.199100      77.387097

       segment_actual_time  segment_osrm_time  segment_osrm_distance  \
count        144867.000000      144867.000000           144867.00000
mean             36.196111          18.507548               22.82902
std              53.571158          14.775960               17.86066
min            -244.000000           0.000000                0.00000
25%              20.000000          11.000000               12.07010
50%              29.000000          17.000000               23.51300
75%              40.000000          22.000000               27.81325
max            3051.000000        1611.000000             2191.40370

       segment_factor
count   144867.000000
mean         2.218368
std          4.847530
min        -23.444444
25%          1.347826
50%          1.684211
75%          2.250000
max        574.250000
```

[108]:
```
unique_values = df.nunique()
print("Unique Values:\n", unique_values)
```

```
Unique Values:
 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
dtype: int64
```

[109]:
```python
print(df.isnull().sum())
```

```
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
```

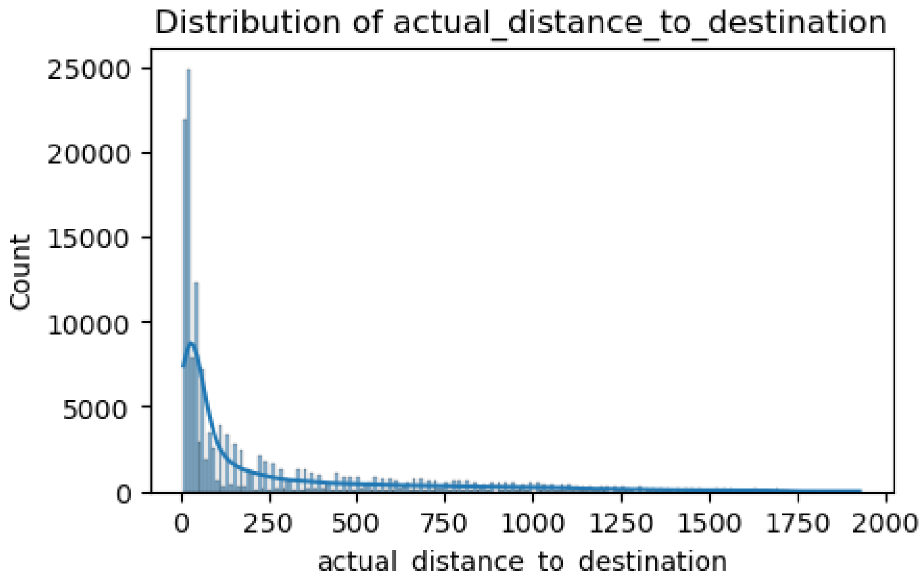
```
segment_osrm_distance              0
segment_factor                     0
dtype: int64
```

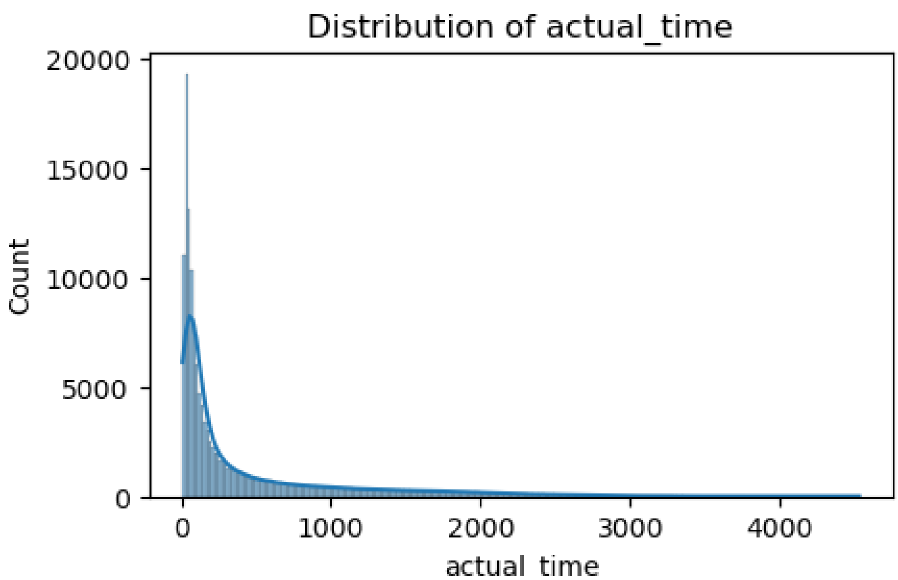
[110]:
```
print(df.dtypes)
```

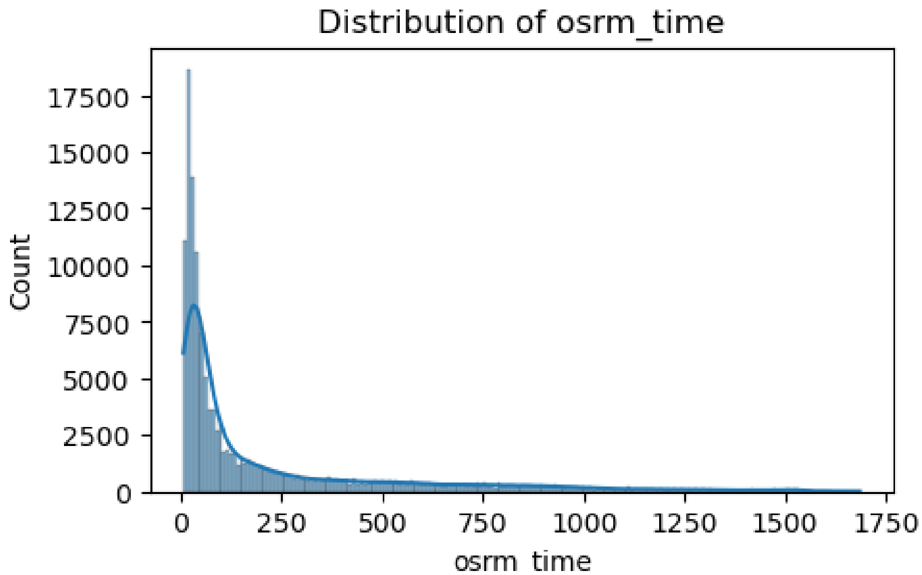
```
data                              object
trip_creation_time                object
route_schedule_uuid               object
route_type                        object
trip_uuid                         object
source_center                     object
source_name                       object
destination_center                object
destination_name                  object
od_start_time                     object
od_end_time                       object
start_scan_to_end_scan           float64
is_cutoff                           bool
cutoff_factor                      int64
cutoff_timestamp                  object
actual_distance_to_destination   float64
actual_time                      float64
osrm_time                        float64
osrm_distance                    float64
factor                           float64
segment_actual_time              float64
segment_osrm_time                float64
segment_osrm_distance            float64
segment_factor                   float64
dtype: object
```

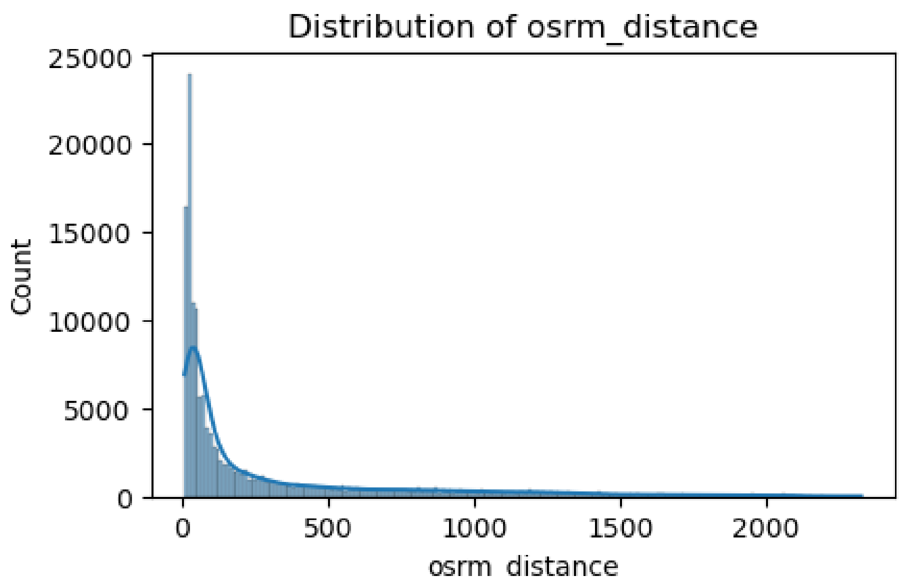
[111]:
```
df['data'] = df['data'].astype('category')
df['route_type'] = df['route_type'].astype('category')

continuous_vars = ['actual_distance_to_destination', 'actual_time',␣
 ↪'osrm_time', 'osrm_distance',
                   'segment_actual_time', 'segment_osrm_time',␣
 ↪'segment_osrm_distance']
```

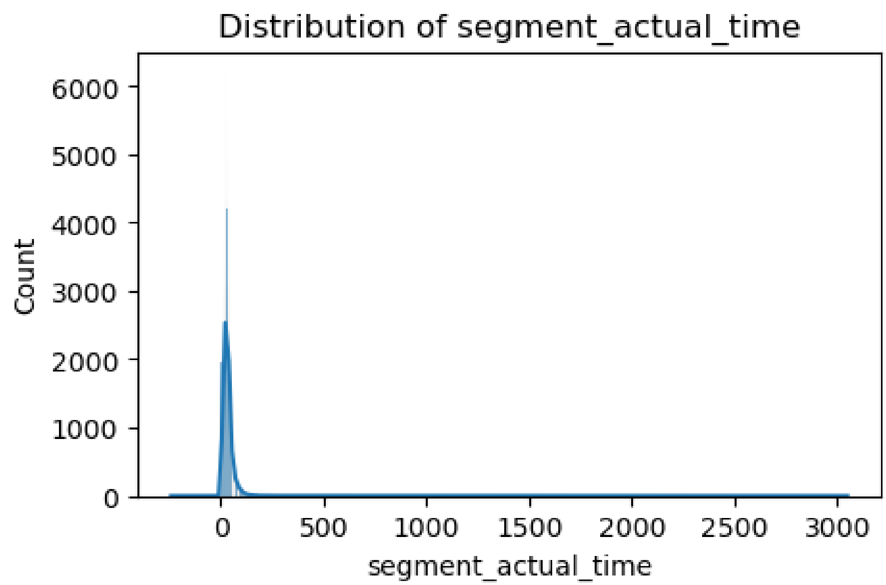
[112]:
```
for var in continuous_vars:
    plt.figure(figsize=(5, 3))
    sns.histplot(df[var], kde=True)
    plt.title(f'Distribution of {var}')
    plt.show()
```

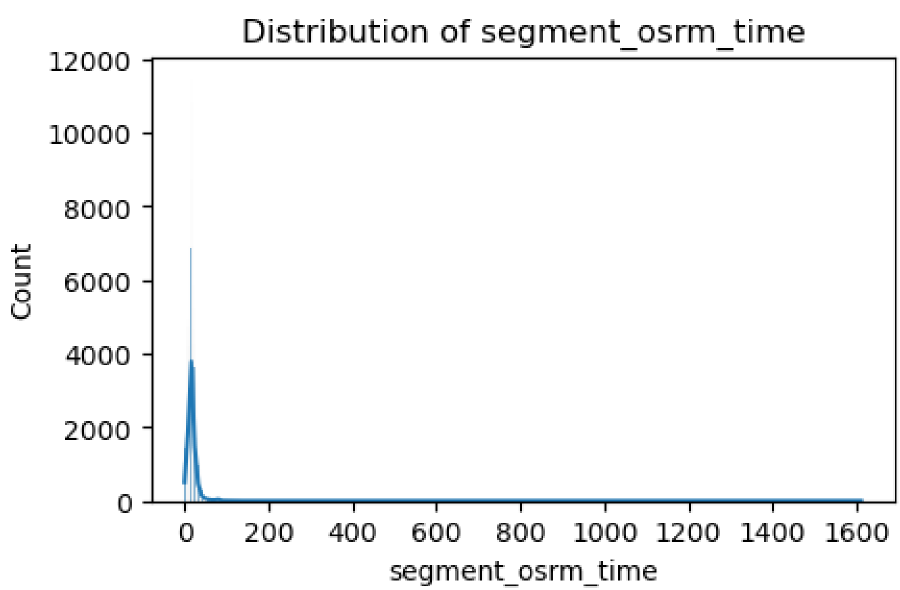

Distribution of actual_distance_to_destination
Count
25000
20000
15000
10000
5000
0
0
250
500
750
1000
1250
1500
1750
2000
actual_distance_to_destination

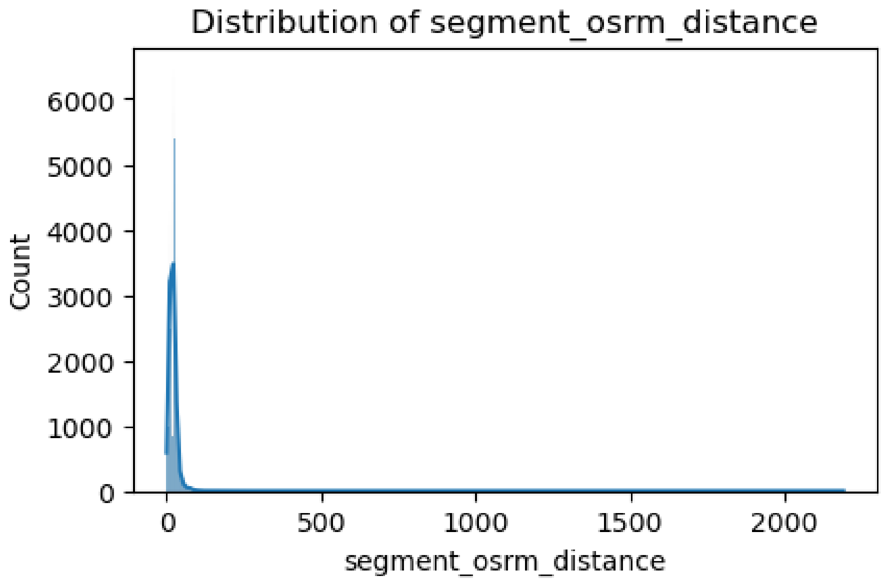

Distribution of actual_time
Count
20000
15000
10000
5000
0
0
1000
2000
3000
4000
actual_time


Distribution of osrm_time
17500
15000
12500
10000
7500
5000
2500
0
Count
0
250
500
750
1000
1250
1500
1750
osrm_time


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

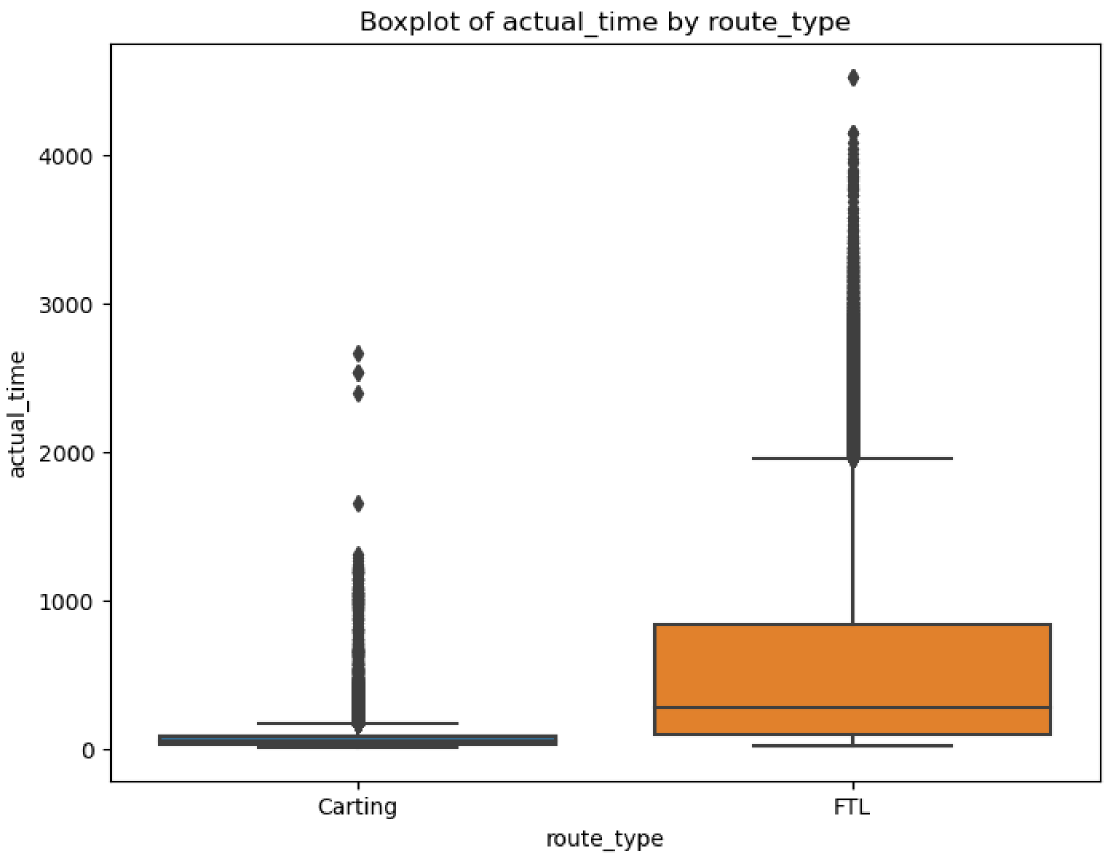

Distribution of segment_actual_time
Count
segment_actual_time
0
1000
2000
3000
4000
5000
6000
0
500
1000
1500
2000
2500
3000


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



[113]:

```
# Boxplots for categorical variables
categorical_vars = ['route_type']

for var in categorical_vars:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=var, y='actual_time', data=df)
    plt.title(f'Boxplot of actual_time by {var}')
    plt.show()
```



0.2 Insights based on EDA.

0.2.1 - There are missing values in certain columns, which need to be handled.

0.2.2 - The dataset contains various numeric and categorical attributes related to trip details.

0.2.3 - Majority of the attributes are numeric, while 'data' and 'route_type' are categorical.

0.2.4 - The distribution plots show the distribution of continuous variables.

0.2.5 - Boxplots reveal potential outliers and distributions across different categories.

1 To handle the missing values

[114]:
```
df['source_name'].fillna('Unknown', inplace=True)
df['destination_name'].fillna('Unknown', inplace=True)
df.dropna(inplace=True)
print(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
```

1.0.1 Merging the rows

[115]:

```
# Group by Trip_uuid, Source ID, and Destination ID and aggregate the data
merged_df = df.groupby(['trip_uuid', 'source_center', 'destination_center']).
 ↪agg({
    'trip_creation_time': 'first',
    'route_schedule_uuid': 'first',
    'route_type': 'first',
    'od_start_time': 'first',
    'od_end_time': 'last',
    'start_scan_to_end_scan': 'sum',
    'actual_distance_to_destination': 'sum',
    'actual_time': 'sum',
    'osrm_time': 'sum',
    'osrm_distance': 'sum',
    'segment_actual_time': 'sum',
    'segment_osrm_time': 'sum',
    'segment_osrm_distance': 'sum',
}).reset_index()

# Aggregate on the basis of Trip_uuid
final_df = merged_df.groupby('trip_uuid').agg({
```

```python
    'source_center': 'first',
    'destination_center': 'first',
    'trip_creation_time': 'first',
    'route_schedule_uuid': 'first',
    'route_type': 'first',
    'od_start_time': 'first',
    'od_end_time': 'last',
    'start_scan_to_end_scan': 'sum',
    'actual_distance_to_destination': 'sum',
    'actual_time': 'sum',
    'osrm_time': 'sum',
    'osrm_distance': 'sum',
    'segment_actual_time': 'sum',
    'segment_osrm_time': 'sum',
    'segment_osrm_distance': 'sum',
}).reset_index()

print(final_df)
```

```
                      trip_uuid source_center destination_center  \
0      trip-153671041653548748  IND209304AAA       IND000000ACB
1      trip-153671042288605164  IND561203AAB       IND562101AAA
2      trip-153671043369099517  IND000000ACB       IND160002AAC
3      trip-153671046011330457  IND400072AAB       IND401104AAA
4      trip-153671052974046625  IND583101AAA       IND583201AAA
…                          …           …                  …
14812  trip-153861095625827784  IND160002AAC       IND140603AAA
14813  trip-153861104386292051  IND121004AAB       IND121004AAA
14814  trip-153861106442901555  IND208006AAA       IND209304AAA
14815  trip-153861115439069069  IND627005AAA       IND628801AAA
14816  trip-153861118270144424  IND583119AAA       IND583101AAA

              trip_creation_time  \
0      2018-09-12 00:00:16.535741
1      2018-09-12 00:00:22.886430
2      2018-09-12 00:00:33.691250
3      2018-09-12 00:01:00.113710
4      2018-09-12 00:02:09.740725
…                             …
14812  2018-10-03 23:55:56.258533
14813  2018-10-03 23:57:23.863155
14814  2018-10-03 23:57:44.429324
14815  2018-10-03 23:59:14.390954
14816  2018-10-03 23:59:42.701692

                                    route_schedule_uuid route_type  \
0      thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6…        FTL
```

```
1      thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0…   Carting
2      thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e…       FTL
3      thanos::sroute:f0176492-a679-4597-8332-bbd1c7f…   Carting
4      thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134…       FTL
…                                                …         …
14812  thanos::sroute:8a120994-f577-4491-9e4b-b7e4a14…   Carting
14813  thanos::sroute:b30e1ec3-3bfa-4bd2-a7fb-3b75769…   Carting
14814  thanos::sroute:5609c268-e436-4e0a-8180-3db4a74…   Carting
14815  thanos::sroute:c5f2ba2c-8486-4940-8af6-d1d2a6a…   Carting
14816  thanos::sroute:412fea14-6d1f-4222-8a5f-a517042…       FTL

                    od_start_time                 od_end_time  \
0      2018-09-12 16:39:46.858469  2018-09-12 16:39:46.858469
1      2018-09-12 02:03:09.655591  2018-09-12 02:03:09.655591
2      2018-09-14 03:40:17.106733  2018-09-14 03:40:17.106733
3      2018-09-12 00:01:00.113710  2018-09-12 01:41:29.809822
4      2018-09-12 00:02:09.740725  2018-09-12 03:54:43.114421
…                               …                           …
14812  2018-10-03 23:55:56.258533  2018-10-04 06:41:25.409035
14813  2018-10-03 23:57:23.863155  2018-10-04 00:57:59.294434
14814  2018-10-04 02:51:27.075797  2018-10-04 02:51:27.075797
14815  2018-10-03 23:59:14.390954  2018-10-04 02:29:04.272194
14816  2018-10-04 03:58:40.726547  2018-10-04 03:58:40.726547

       start_scan_to_end_scan  actual_distance_to_destination  actual_time  \
0                     43659.0                     8860.812105      15682.0
1                       906.0                      240.208306        399.0
2                    248631.0                    68163.502238     112225.0
3                       200.0                       28.529648         82.0
4                      1586.0                      239.007304        556.0
…                           …                               …          …
14812                   876.0                      141.057373        186.0
14813                   120.0                       25.130640         33.0
14814                  1263.0                       93.743842        549.0
14815                  1315.0                      355.281673        600.0
14816                   706.0                      110.239116        350.0

       osrm_time  osrm_distance  segment_actual_time  segment_osrm_time  \
0         7787.0     10577.7647               1548.0             1008.0
1          210.0       269.4308                141.0               65.0
2        65768.0     89447.2488               3308.0             1941.0
3           24.0        31.6475                 59.0               16.0
4          207.0       266.2914                340.0              115.0
…              …              …                    …                  …
14812      148.0       162.9473                 82.0               62.0
14813       19.0        26.5333                 21.0               11.0
14814      134.0       162.8499                281.0               88.0
14815      446.0       449.5383                258.0              221.0
```

```
14816      106.0      127.8020            274.0            67.0

       segment_osrm_distance
0                  1320.4733
1                    84.1894
2                  2545.2678
3                    19.8766
4                   146.7919
…                          …
14812                64.8551
14813                16.0883
14814               104.8866
14815               223.5324
14816                80.5787

[14817 rows x 16 columns]
```

1.0.2 Destination Name: Split and extract features out of destination. City-place-code (State)

[116]:
```
destination_split = df['destination_name'].str.split('-', expand=True)
df['Destination_City'] = destination_split[0]
df['Destination_Place'] = destination_split[1]
df['Destination_State_Code'] = destination_split[2] if len(destination_split.
 ↪columns) > 2 else None
```

1.0.3 Source Name: Split and extract features out of destination. City-place-code (State)

[117]:
```
source_split = df['source_name'].str.split('-', expand=True)
df['Source_City'] = source_split[0]
df['Source_Place'] = source_split[1]
df['Source_State_Code'] = source_split[2] if len(source_split.columns) > 2 else␣
 ↪None
```

1.0.4 Trip_creation_time: Extract features like month, year and day

[118]:
```
df['Trip_Creation_Time'] = pd.to_datetime(df['trip_creation_time'])
df['Trip_Year'] = df['Trip_Creation_Time'].dt.year
df['Trip_Month'] = df['Trip_Creation_Time'].dt.month
df['Trip_Day'] = df['Trip_Creation_Time'].dt.day
df['Trip_Hour'] = df['Trip_Creation_Time'].dt.hour
df['Trip_Weekday'] = df['Trip_Creation_Time'].dt.weekday
```

1.0.5 The time taken between od_start_time and od_end_time

[119]:
```
df['od_start_time'] = pd.to_datetime(df['od_start_time'])
df['od_end_time'] = pd.to_datetime(df['od_end_time'])

df['Time_Taken'] = (df['od_end_time'] - df['od_start_time']).dt.total_seconds()

# Optionally, dropping the original 'od_start_time' and 'od_end_time' columns
df.drop(columns=['od_start_time', 'od_end_time'], inplace=True)

print("DataFrame with time taken feature:")
print(df.head())
```

```
DataFrame with time taken feature:
       data          trip_creation_time  \
0  training  2018-09-20 02:35:36.476840
1  training  2018-09-20 02:35:36.476840
2  training  2018-09-20 02:35:36.476840
3  training  2018-09-20 02:35:36.476840
4  training  2018-09-20 02:35:36.476840

                                 route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                 trip_uuid source_center                 source_name  \
0  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)

  destination_center               destination_name  start_scan_to_end_scan  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0

   …                Source_City  Source_Place Source_State_Code  \
0  …  Anand_VUNagar_DC (Gujarat)          None              None
1  …  Anand_VUNagar_DC (Gujarat)          None              None
2  …  Anand_VUNagar_DC (Gujarat)          None              None
3  …  Anand_VUNagar_DC (Gujarat)          None              None
4  …  Anand_VUNagar_DC (Gujarat)          None              None
```

```
           Trip_Creation_Time  Trip_Year  Trip_Month  Trip_Day  Trip_Hour  \
0 2018-09-20 02:35:36.476840       2018           9        20          2
1 2018-09-20 02:35:36.476840       2018           9        20          2
2 2018-09-20 02:35:36.476840       2018           9        20          2
3 2018-09-20 02:35:36.476840       2018           9        20          2
4 2018-09-20 02:35:36.476840       2018           9        20          2

   Trip_Weekday   Time_Taken
0             3  5172.818197
1             3  5172.818197
2             3  5172.818197
3             3  5172.818197
4             3  5172.818197

[5 rows x 35 columns]
```

2 Comparing the difference between Point a. and start_scan_to_end_scan

2.1 Hypothesis Testing

Null Hypothesis (H0): There is no significant difference between the calculated time taken ('Time_Taken') and the provided time ('start_scan_to_end_scan').sis.

Alternative Hypothesis (H1): There is a significant difference between the calculated time taken ('Time_Taken') and the provided time ('start_scan_to_end_scan').

We can perform a paired t-test to test this hypothesis.

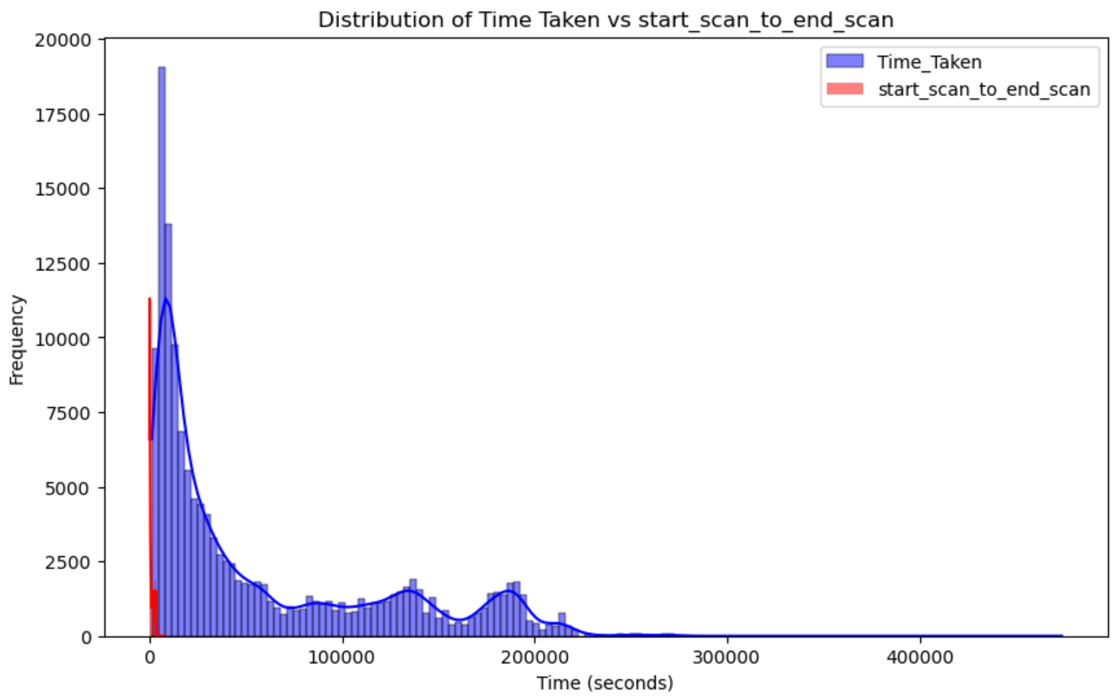
[120]:
```
t_statistic, p_value = stats.ttest_rel(df['Time_Taken'],
 ↪df['start_scan_to_end_scan'])

alpha = 0.05

print("Hypothesis Testing Results:")
print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

if p_value < alpha:
    print("Reject Null Hypothesis: There is a significant difference between
 ↪Time_Taken and start_scan_to_end_scan.")
else:
    print("Fail to reject Null Hypothesis: There is no significant difference
 ↪between Time_Taken and start_scan_to_end_scan.")
```

```
Hypothesis Testing Results:
T-statistic: 352.9967867233228
```

```
P-value: 0.0
Reject Null Hypothesis: There is a significant difference between Time_Taken and
start_scan_to_end_scan.
```

2.1.1 Visual Analysis

[121]:
```python
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Time_Taken', color='blue', label='Time_Taken',␣
 ↪kde=True)
sns.histplot(data=df, x='start_scan_to_end_scan', color='red',␣
 ↪label='start_scan_to_end_scan', kde=True)
plt.title('Distribution of Time Taken vs start_scan_to_end_scan')
plt.xlabel('Time (seconds)')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



3 Hypothesis testing & visual analysis between actual_time aggregated value and OSRM time aggregated value

3.1 Hypothesis Testing

Null Hypothesis (H0): There is no significant difference between the aggregated values of 'actual_time' and 'osrm_time' after merging rows based on 'trip_uuid'.

Alternative Hypothesis (H1): There is a significant difference between the aggregated values of 'actual_time' and 'osrm_time' after merging rows based on 'trip_uuid'.

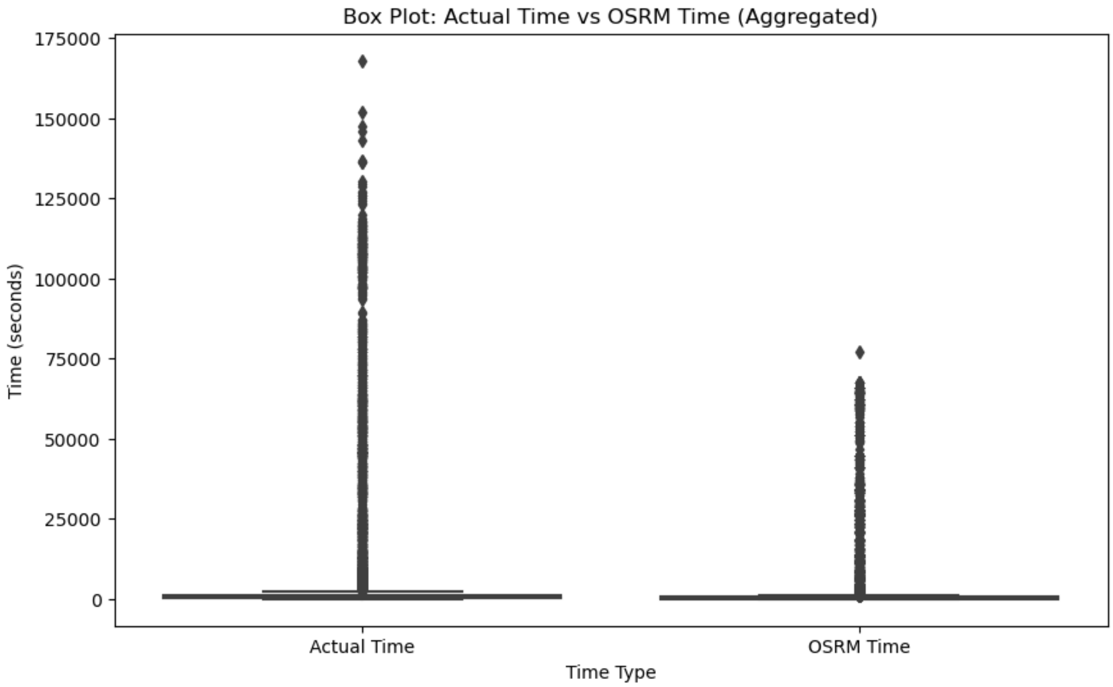
We'll perform a paired t-test to test this hypothesis.

[122]:
```
t_statistic, p_value = stats.ttest_rel(df.groupby('trip_uuid')['actual_time'].
 ↪sum(), df.groupby('trip_uuid')['osrm_time'].sum())

alpha = 0.05  # significance level

print("Hypothesis Testing Results:")
print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

if p_value < alpha:
    print("Reject Null Hypothesis: There is a significant difference between␣
 ↪actual_time and osrm_time.")
else:
    print("Fail to reject Null Hypothesis: There is no significant difference␣
 ↪between actual_time and osrm_time.")
```

```
Hypothesis Testing Results:
T-statistic: 32.468089449426905
P-value: 1.8633294618952604e-223
Reject Null Hypothesis: There is a significant difference between actual_time
and osrm_time.
```

3.1.1 Visual Analysis:

[123]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df.groupby('trip_uuid')[['actual_time', 'osrm_time']].sum())
plt.title('Box Plot: Actual Time vs OSRM Time (Aggregated)')
plt.xlabel('Time Type')
plt.ylabel('Time (seconds)')
plt.xticks(ticks=[0, 1], labels=['Actual Time', 'OSRM Time'])
plt.show()
```



4 Hypothesis testing & visual analysis between actual_time aggregated value and segment actual time aggregated value

4.1 Hypothesis testing

Null Hypothesis (H0): There is no significant difference between the aggregated values of 'actual_time' and 'segment_actual_time' after merging rows based on 'trip_uuid'.

Alternative Hypothesis (H1): There is a significant difference between the aggregated values of 'actual_time' and 'segment_actual_time' after merging rows based on 'trip_uuid'.

We'll perform a paired t-test to test this hypothesis.

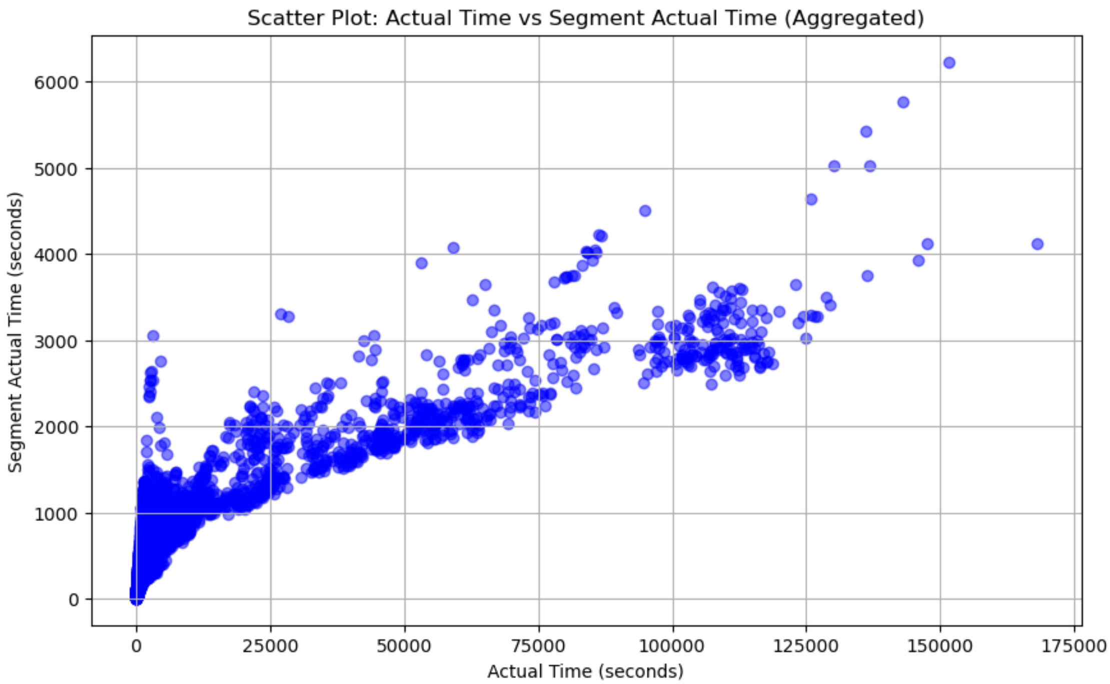
[124]:
```
t_statistic, p_value = stats.ttest_rel(df.groupby('trip_uuid')['actual_time'].
 ↪sum(), df.groupby('trip_uuid')['segment_actual_time'].sum())

alpha = 0.05  # significance level

print("Hypothesis Testing Results:")
print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

if p_value < alpha:
```

```
    print("Reject Null Hypothesis: There is a significant difference between␣
↪actal_time and segment_actual_time.")
else:
    print("Fail to reject Null Hypothesis: There is no significant difference␣
↪between actual_time and segment_actual_time.")
```

```
Hypothesis Testing Results:
T-statistic: 30.75550616001704
P-value: 2.0773254218008745e-201
Reject Null Hypothesis: There is a significant difference between actal_time and
segment_actual_time.
```

4.1.1 Visual Analysis:

[125]:
```
plt.figure(figsize=(10, 6))
plt.scatter(df.groupby('trip_uuid')['actual_time'].sum(), df.
↪groupby('trip_uuid')['segment_actual_time'].sum(), color='blue', alpha=0.5)
plt.title('Scatter Plot: Actual Time vs Segment Actual Time (Aggregated)')
plt.xlabel('Actual Time (seconds)')
plt.ylabel('Segment Actual Time (seconds)')
plt.grid(True)
plt.show()
```



5 Hypothesis testing & visual analysis between osrm distance aggregated value and segment osrm distance aggregated value

5.1 Hypothesis testing

H0: There is no significant difference between the means of osrm_distance and segment_osrm_distance aggregated values.

H1: There is a significant difference between the means of osrm_distance and segment_osrm_distance aggregated values.

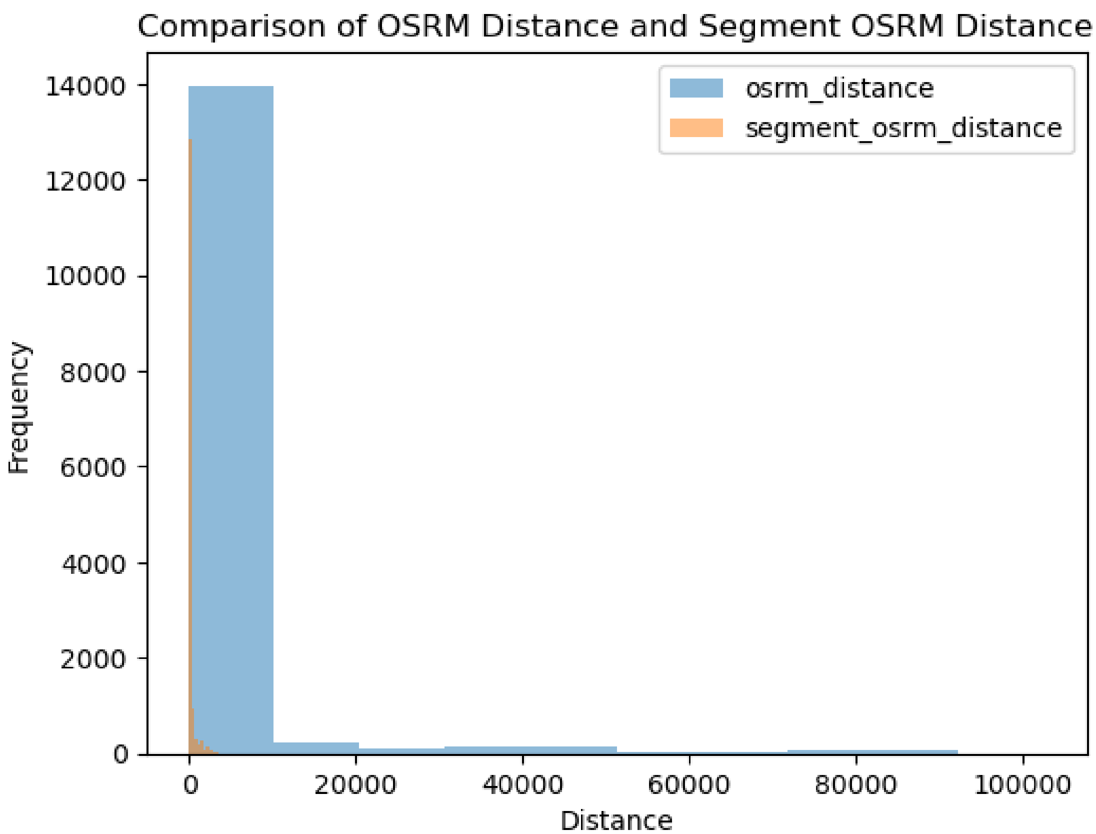
We'll perform a paired t-test to test this hypothesis.

[126]:
```
osrm_distance_aggregated = df.groupby('trip_uuid')['osrm_distance'].sum()
segment_osrm_distance_aggregated = df.
 ↪groupby('trip_uuid')['segment_osrm_distance'].sum()

t_statistic, p_value = ttest_rel(osrm_distance_aggregated,␣
 ↪segment_osrm_distance_aggregated)

print("Paired t-test results:")
print("t-statistic:", t_statistic)
print("p-value:", p_value)

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference␣
 ↪between the means.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
 ↪difference between the means.")
```

```
Paired t-test results:
t-statistic: 30.03031541377046
p-value: 2.1753879024067997e-192
Reject the null hypothesis: There is a significant difference between the means.
```

5.1.1 Visual Analysis:

[127]:
```
plt.hist(osrm_distance_aggregated, alpha=0.5, label='osrm_distance')
plt.hist(segment_osrm_distance_aggregated, alpha=0.5,␣
 ↪label='segment_osrm_distance')
plt.xlabel('Distance')
plt.ylabel('Frequency')
plt.title('Comparison of OSRM Distance and Segment OSRM Distance')
plt.legend()
plt.show()
```



6 Hypothesis testing & visual analysis between osrm time aggregated value and segment osrm time aggregated value

6.0.1 Hypothesis testing

H0: This hypothesis states that there is no significant difference between the means of osrm_time and segment_osrm_time aggregated values.pectively.

H1: This hypothesis contradicts the null hypothesis and suggests that there is a significant difference between the means of osrm_time and segment_osrm_time aggregated values.

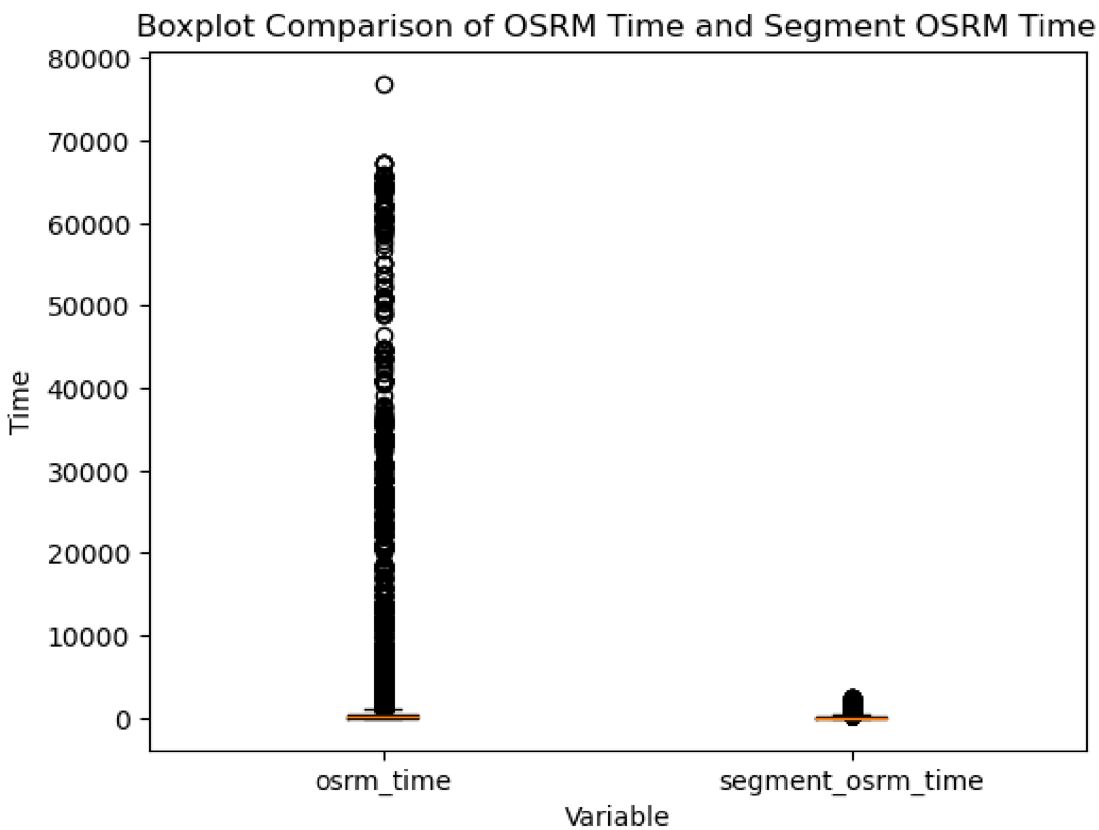
[128]:
```
osrm_time_aggregated = df.groupby('trip_uuid')['osrm_time'].sum()
segment_osrm_time_aggregated = df.groupby('trip_uuid')['segment_osrm_time'].
 ↪sum()

t_statistic, p_value = ttest_rel(osrm_time_aggregated,␣
 ↪segment_osrm_time_aggregated)

alpha = 0.05
```

```
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference␣
 ↪between the means.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
 ↪difference between the means.")

print("Paired t-test results:")
print("t-statistic:", t_statistic)
print("p-value:", p_value)
```

```
Reject the null hypothesis: There is a significant difference between the means.
Paired t-test results:
t-statistic: 30.29743310414474
p-value: 1.0892807362104113e-195
```

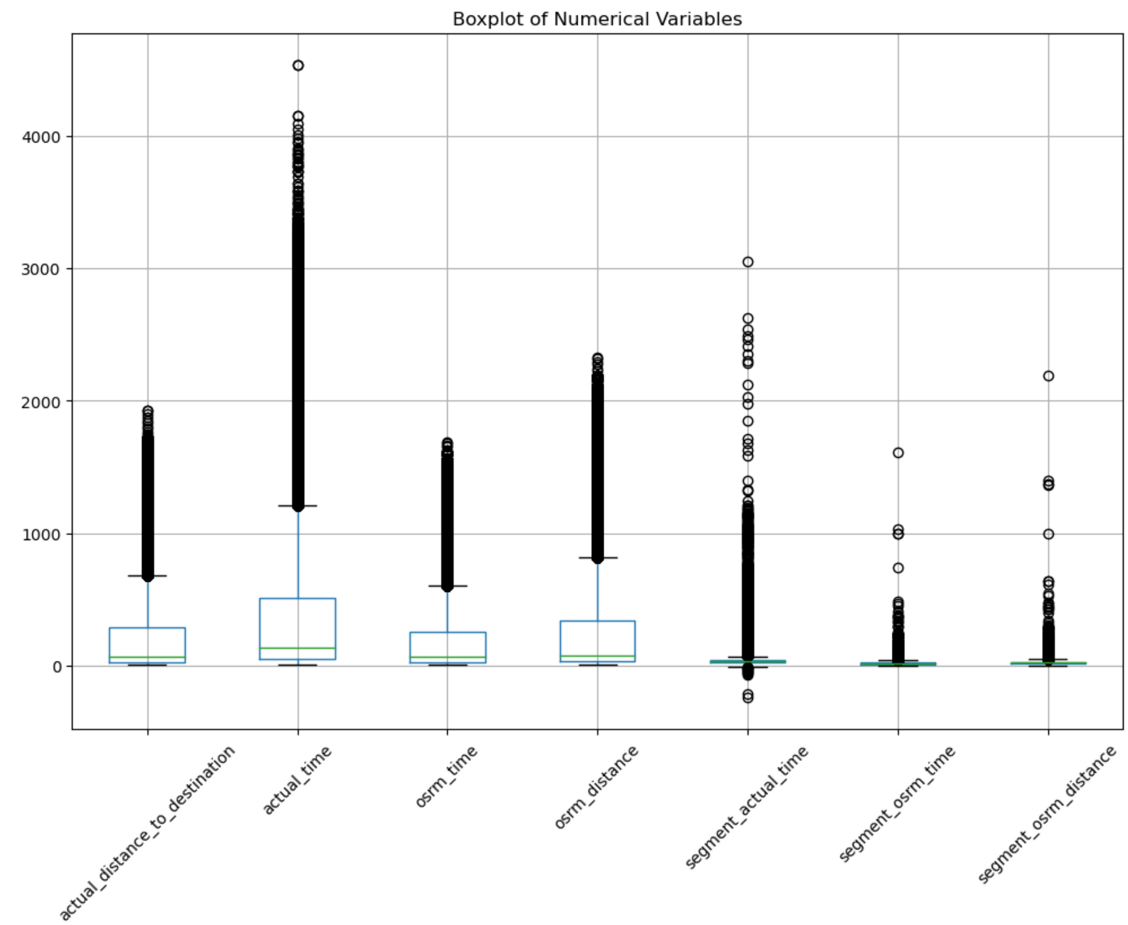
6.0.2 Visual Analysis:

[129]:
```
plt.boxplot([osrm_time_aggregated, segment_osrm_time_aggregated],␣
 ↪labels=['osrm_time', 'segment_osrm_time'])
plt.xlabel('Variable')
plt.ylabel('Time')
plt.title('Boxplot Comparison of OSRM Time and Segment OSRM Time')
plt.show()
```

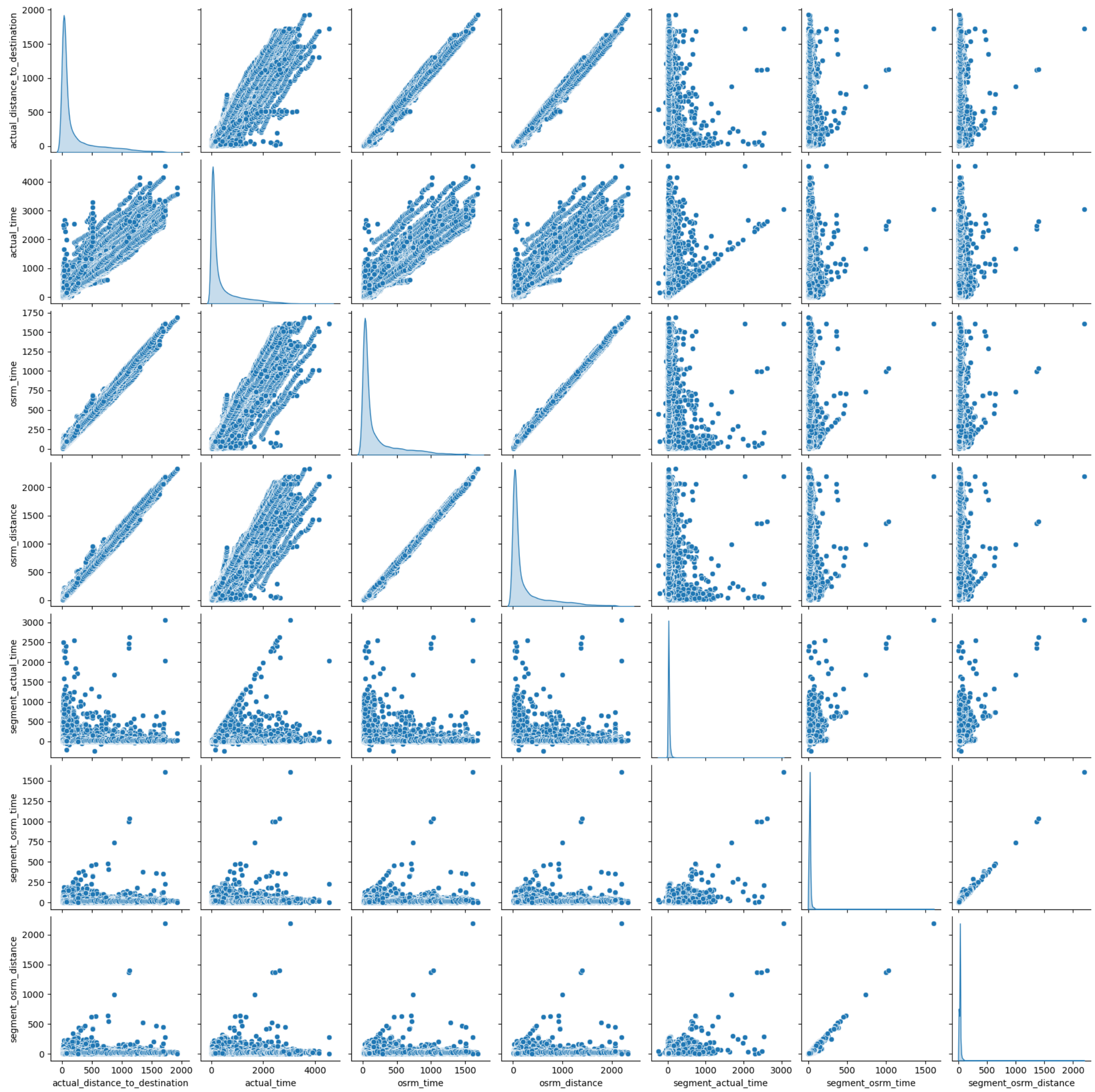


7 The outliers in the numerical variables

[130]:
```
numerical_columns = ['actual_distance_to_destination', 'actual_time',␣
↪'osrm_time', 'osrm_distance',
                     'segment_actual_time', 'segment_osrm_time',␣
↪'segment_osrm_distance']
plt.figure(figsize=(12, 8))
df[numerical_columns].boxplot()
plt.title('Boxplot of Numerical Variables')
plt.xticks(rotation=45)
plt.show()
```



[131]:
```
sns.pairplot(df[numerical_columns], diag_kind='kde')
plt.show()
```



7.1 Handling the outliers using the IQR method.

7.1.1 Removing Outliers:

[132]:
```
Q1 = df[numerical_columns].quantile(0.25)
Q3 = df[numerical_columns].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df_no_outliers = df[~((df[numerical_columns] < lower_bound) |␣
 ↪(df[numerical_columns] > upper_bound)).any(axis=1)]
```

```python
print("Shape of dataset after removing outliers:", df_no_outliers.shape)
```

```
Shape of dataset after removing outliers: (114189, 35)
```

7.1.2 Replacing Outliers:

[133]:
```python
df_no_outliers_replaced = df.copy()

for col in numerical_columns:
    median = df[col].median()
    df_no_outliers_replaced[col] = df[col].mask((df[col] < lower_bound[col]) |
 ↪(df[col] > upper_bound[col]), median)

print("Dataset with replaced outliers:")
print(df_no_outliers_replaced.head())
```

```
Dataset with replaced outliers:
       data          trip_creation_time  \
0  training  2018-09-20 02:35:36.476840
1  training  2018-09-20 02:35:36.476840
2  training  2018-09-20 02:35:36.476840
3  training  2018-09-20 02:35:36.476840
4  training  2018-09-20 02:35:36.476840

                                 route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                 trip_uuid source_center                 source_name  \
0  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)

  destination_center              destination_name  start_scan_to_end_scan  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0

   …                 Source_City  Source_Place Source_State_Code  \
0  …  Anand_VUNagar_DC (Gujarat)          None              None
```

```
1  …  Anand_VUNagar_DC (Gujarat)         None             None
2  …  Anand_VUNagar_DC (Gujarat)         None             None
3  …  Anand_VUNagar_DC (Gujarat)         None             None
4  …  Anand_VUNagar_DC (Gujarat)         None             None

           Trip_Creation_Time  Trip_Year  Trip_Month  Trip_Day  Trip_Hour  \
0 2018-09-20 02:35:36.476840       2018           9        20          2
1 2018-09-20 02:35:36.476840       2018           9        20          2
2 2018-09-20 02:35:36.476840       2018           9        20          2
3 2018-09-20 02:35:36.476840       2018           9        20          2
4 2018-09-20 02:35:36.476840       2018           9        20          2

   Trip_Weekday   Time_Taken
0             3  5172.818197
1             3  5172.818197
2             3  5172.818197
3             3  5172.818197
4             3  5172.818197

[5 rows x 35 columns]
```

7.1.3 One-hot encoding of categorical variables (like route_type)

[134]:
```python
encoded_df = pd.get_dummies(df, columns=['route_type'])

print("Encoded DataFrame:")
print(encoded_df.head())
```

```
Encoded DataFrame:
       data          trip_creation_time  \
0  training  2018-09-20 02:35:36.476840
1  training  2018-09-20 02:35:36.476840
2  training  2018-09-20 02:35:36.476840
3  training  2018-09-20 02:35:36.476840
4  training  2018-09-20 02:35:36.476840

                                 route_schedule_uuid                trip_uuid  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…  trip-153741093647649320
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…  trip-153741093647649320
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…  trip-153741093647649320
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…  trip-153741093647649320
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…  trip-153741093647649320

  source_center                 source_name destination_center  \
0  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB
1  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB
2  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB
3  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB
```

```
4  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB

            destination_name  start_scan_to_end_scan  is_cutoff  …  \
0  Khambhat_MotvdDPP_D (Gujarat)                    86.0       True  …
1  Khambhat_MotvdDPP_D (Gujarat)                    86.0       True  …
2  Khambhat_MotvdDPP_D (Gujarat)                    86.0       True  …
3  Khambhat_MotvdDPP_D (Gujarat)                    86.0       True  …
4  Khambhat_MotvdDPP_D (Gujarat)                    86.0      False  …

   Source_State_Code         Trip_Creation_Time  Trip_Year  Trip_Month  \
0               None 2018-09-20 02:35:36.476840       2018           9
1               None 2018-09-20 02:35:36.476840       2018           9
2               None 2018-09-20 02:35:36.476840       2018           9
3               None 2018-09-20 02:35:36.476840       2018           9
4               None 2018-09-20 02:35:36.476840       2018           9

   Trip_Day  Trip_Hour  Trip_Weekday   Time_Taken  route_type_Carting  \
0        20          2             3  5172.818197                True
1        20          2             3  5172.818197                True
2        20          2             3  5172.818197                True
3        20          2             3  5172.818197                True
4        20          2             3  5172.818197                True

   route_type_FTL
0           False
1           False
2           False
3           False
4           False

[5 rows x 36 columns]
```

7.2 Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler.

7.2.1 Using MinMaxScaler for Normalization:

[135]:
```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

numerical_columns = ['actual_distance_to_destination', 'actual_time',␣
↪'osrm_time', 'osrm_distance',
                     'segment_actual_time', 'segment_osrm_time',␣
↪'segment_osrm_distance']

df_normalized = df.copy()
df_normalized[numerical_columns] = scaler.fit_transform(df[numerical_columns])
```

```python
print("Normalized DataFrame:")
print(df_normalized.head())
```

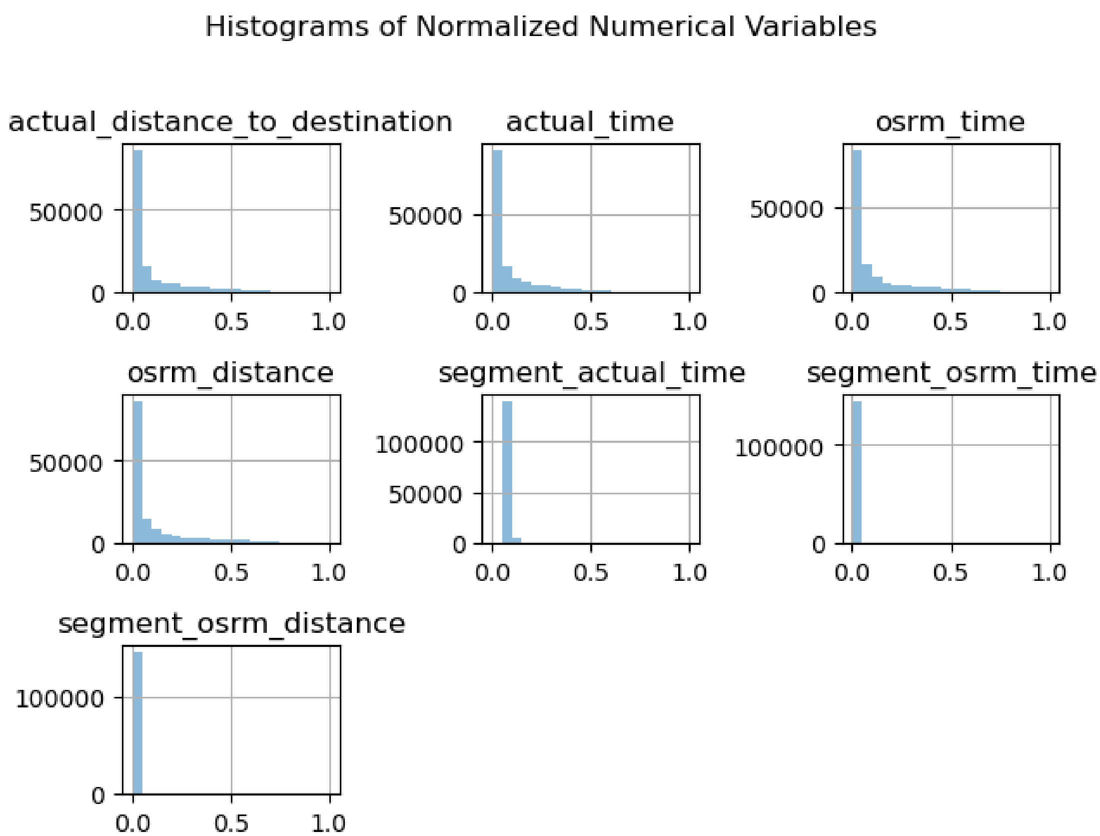
```
Normalized DataFrame:
       data          trip_creation_time  \
0  training  2018-09-20 02:35:36.476840
1  training  2018-09-20 02:35:36.476840
2  training  2018-09-20 02:35:36.476840
3  training  2018-09-20 02:35:36.476840
4  training  2018-09-20 02:35:36.476840

                                 route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                trip_uuid source_center                 source_name  \
0  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)

  destination_center               destination_name  start_scan_to_end_scan  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0

   …                 Source_City  Source_Place Source_State_Code  \
0  …  Anand_VUNagar_DC (Gujarat)          None              None
1  …  Anand_VUNagar_DC (Gujarat)          None              None
2  …  Anand_VUNagar_DC (Gujarat)          None              None
3  …  Anand_VUNagar_DC (Gujarat)          None              None
4  …  Anand_VUNagar_DC (Gujarat)          None              None

           Trip_Creation_Time  Trip_Year  Trip_Month  Trip_Day  Trip_Hour  \
0 2018-09-20 02:35:36.476840       2018           9        20          2
1 2018-09-20 02:35:36.476840       2018           9        20          2
2 2018-09-20 02:35:36.476840       2018           9        20          2
3 2018-09-20 02:35:36.476840       2018           9        20          2
4 2018-09-20 02:35:36.476840       2018           9        20          2

   Trip_Weekday   Time_Taken
```

```
0              3  5172.818197
1              3  5172.818197
2              3  5172.818197
3              3  5172.818197
4              3  5172.818197

[5 rows x 35 columns]
```

[140]:

```python
plt.figure(figsize=(15, 10))
df_normalized[numerical_columns].hist(bins=20, alpha=0.5)
plt.suptitle('Histograms of Normalized Numerical Variables', y=1.02)
plt.xlabel('Normalized Value')
plt.ylabel('Frequency')
plt.legend(numerical_columns)
plt.tight_layout()
plt.show()
```

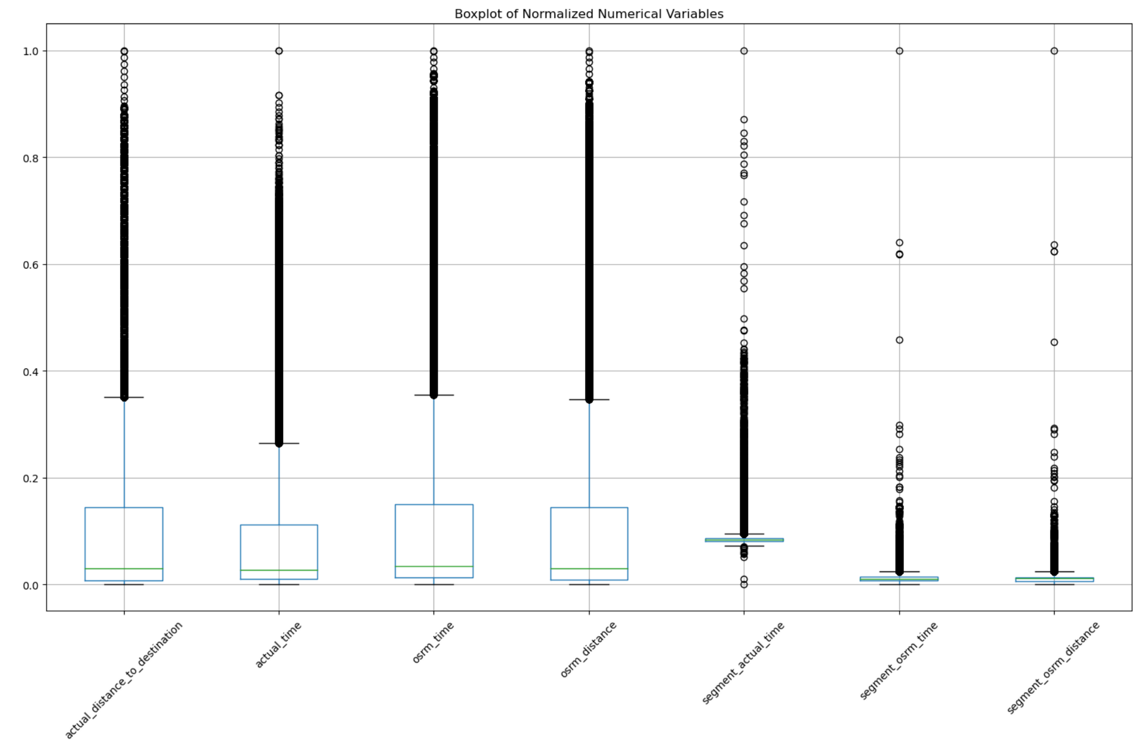
```
<Figure size 1500x1000 with 0 Axes>
```



[142]:

```python
plt.figure(figsize=(15, 10))
df_normalized[numerical_columns].boxplot()
```

```python
plt.title('Boxplot of Normalized Numerical Variables')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



7.2.2 Using StandardScaler for Standardization:

[141]:
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

numerical_columns = ['actual_distance_to_destination', 'actual_time',␣
 ↪'osrm_time', 'osrm_distance',
                     'segment_actual_time', 'segment_osrm_time',␣
 ↪'segment_osrm_distance']

df_standardized = df.copy()
df_standardized[numerical_columns] = scaler.fit_transform(df[numerical_columns])

print("Standardized DataFrame:")
print(df_standardized.head())
```

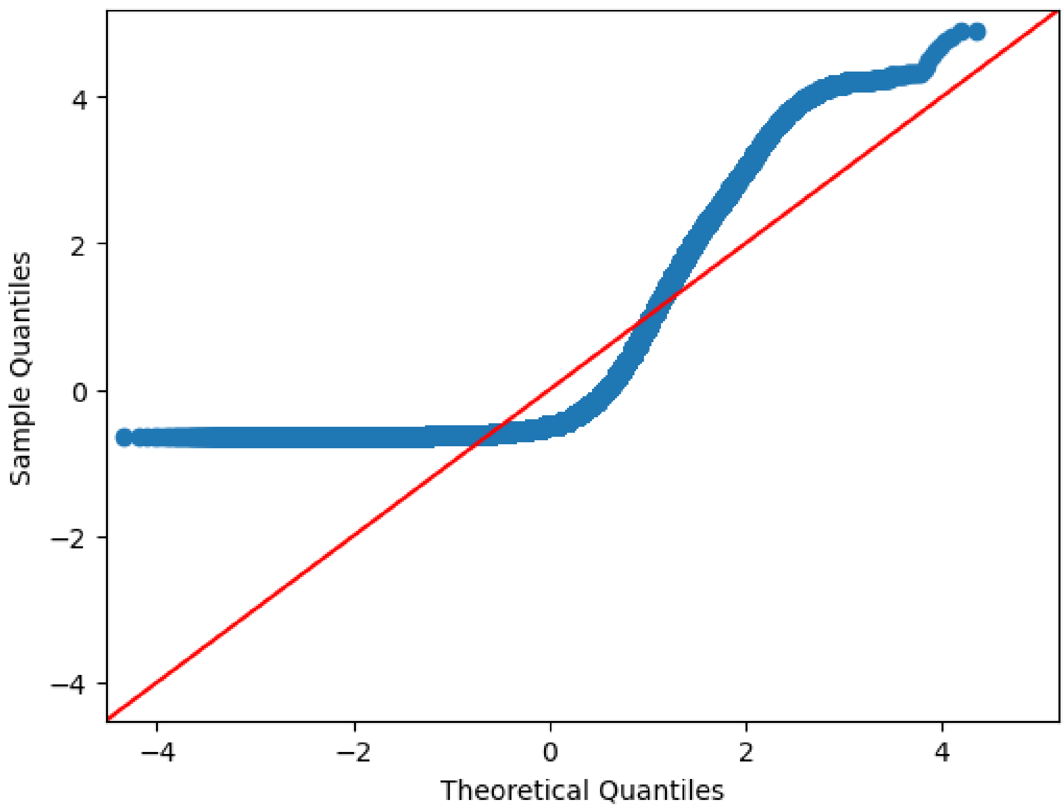
```
Standardized DataFrame:
```

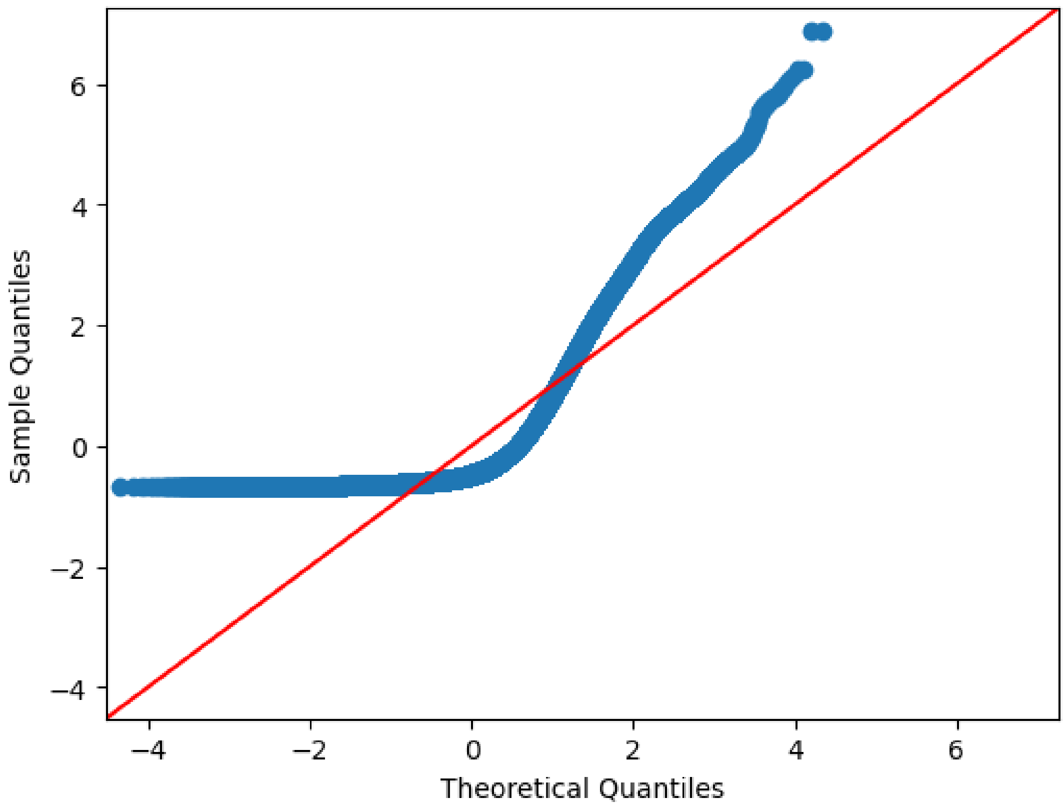
```
       data          trip_creation_time  \
0  training  2018-09-20 02:35:36.476840
1  training  2018-09-20 02:35:36.476840
2  training  2018-09-20 02:35:36.476840
3  training  2018-09-20 02:35:36.476840
4  training  2018-09-20 02:35:36.476840

                                 route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                 trip_uuid source_center                 source_name  \
0  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)

  destination_center              destination_name  start_scan_to_end_scan  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)                    86.0

   …                 Source_City  Source_Place Source_State_Code  \
0  …  Anand_VUNagar_DC (Gujarat)          None              None
1  …  Anand_VUNagar_DC (Gujarat)          None              None
2  …  Anand_VUNagar_DC (Gujarat)          None              None
3  …  Anand_VUNagar_DC (Gujarat)          None              None
4  …  Anand_VUNagar_DC (Gujarat)          None              None

          Trip_Creation_Time  Trip_Year  Trip_Month  Trip_Day  Trip_Hour  \
0 2018-09-20 02:35:36.476840       2018           9        20          2
1 2018-09-20 02:35:36.476840       2018           9        20          2
2 2018-09-20 02:35:36.476840       2018           9        20          2
3 2018-09-20 02:35:36.476840       2018           9        20          2
4 2018-09-20 02:35:36.476840       2018           9        20          2

   Trip_Weekday   Time_Taken
0             3  5172.818197
1             3  5172.818197
2             3  5172.818197
3             3  5172.818197
4             3  5172.818197
```

```
[5 rows x 35 columns]
```

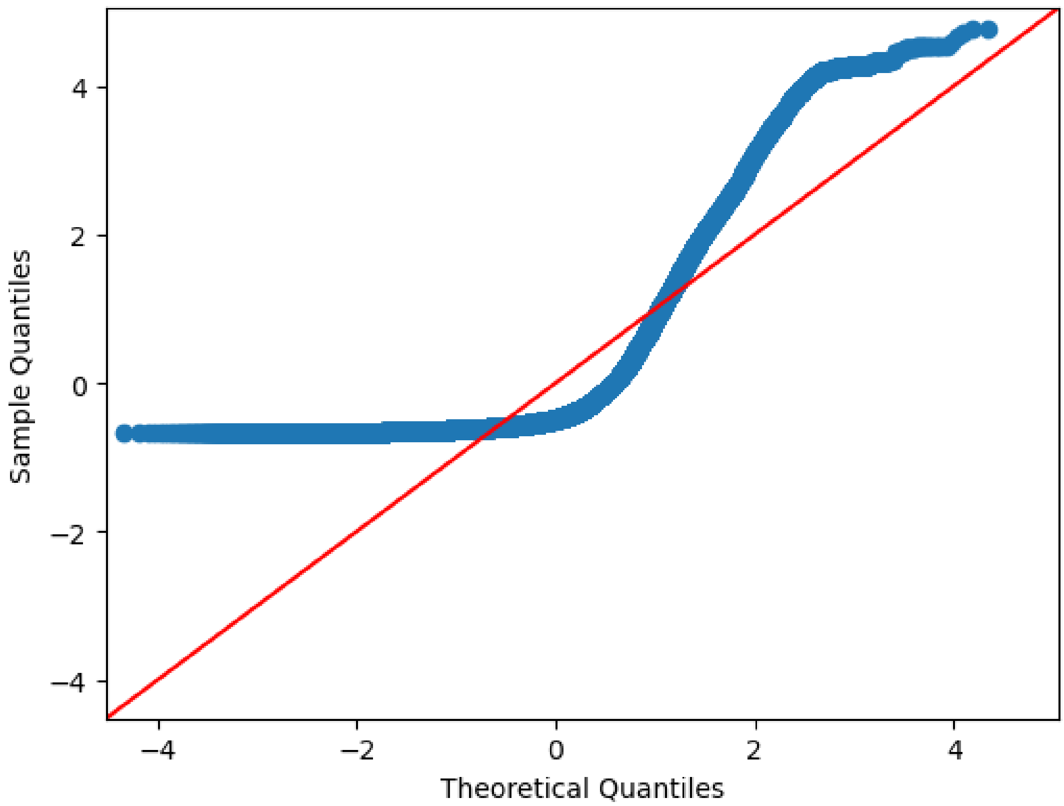
[149]:
```python
import statsmodels.api as sm
# QQ plots for visualizing the distributions of standardized variables
plt.figure(figsize=(15, 10))
for col in numerical_columns:
    sm.qqplot(df_standardized[col], line='45', label=col)
plt.title('QQ Plot of Standardized Numerical Variables')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.legend()
plt.tight_layout()
plt.show()
```

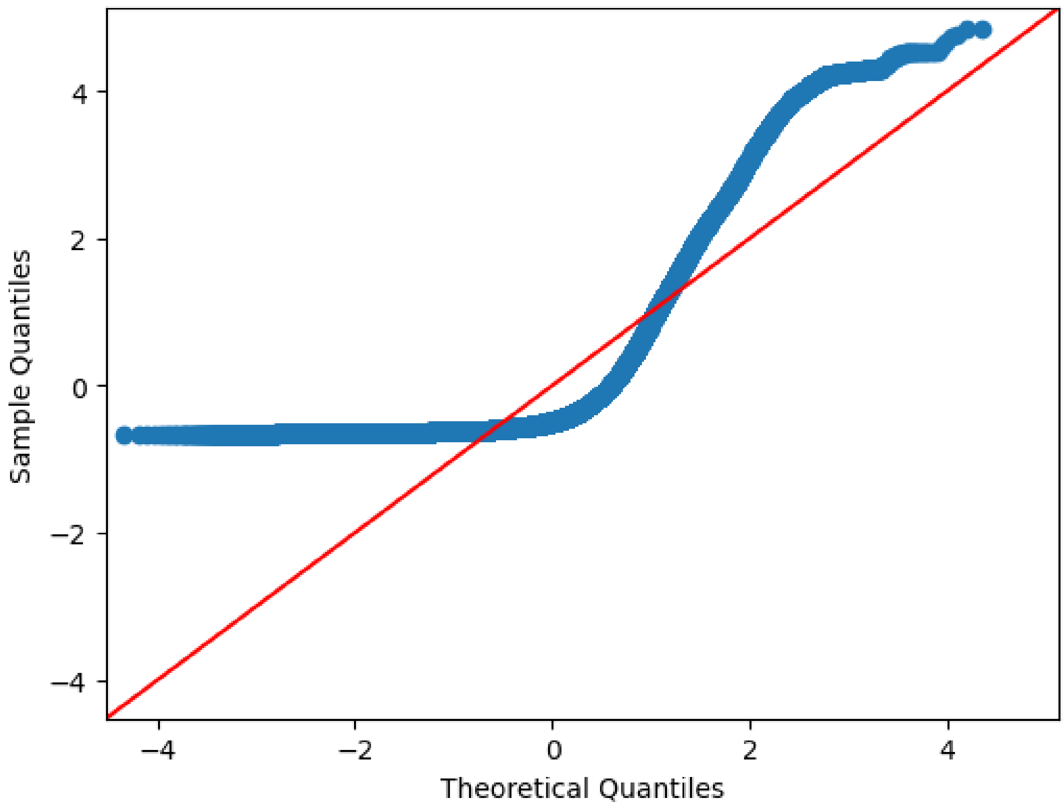
```
<Figure size 1500x1000 with 0 Axes>
```

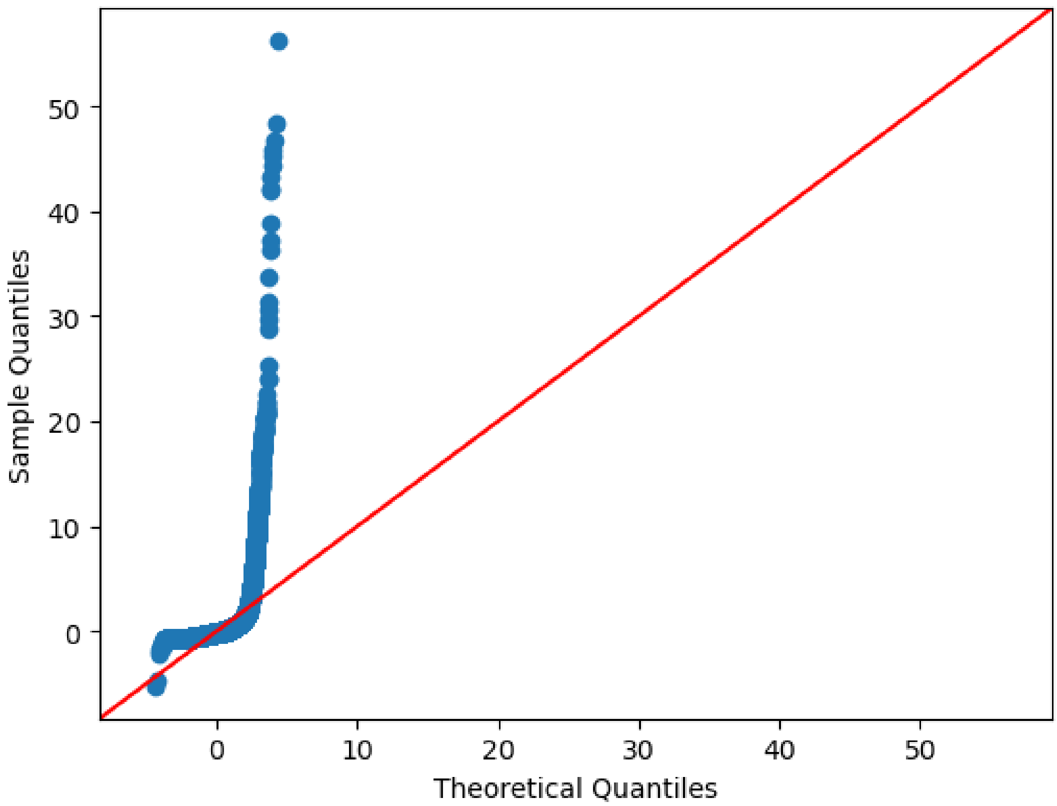


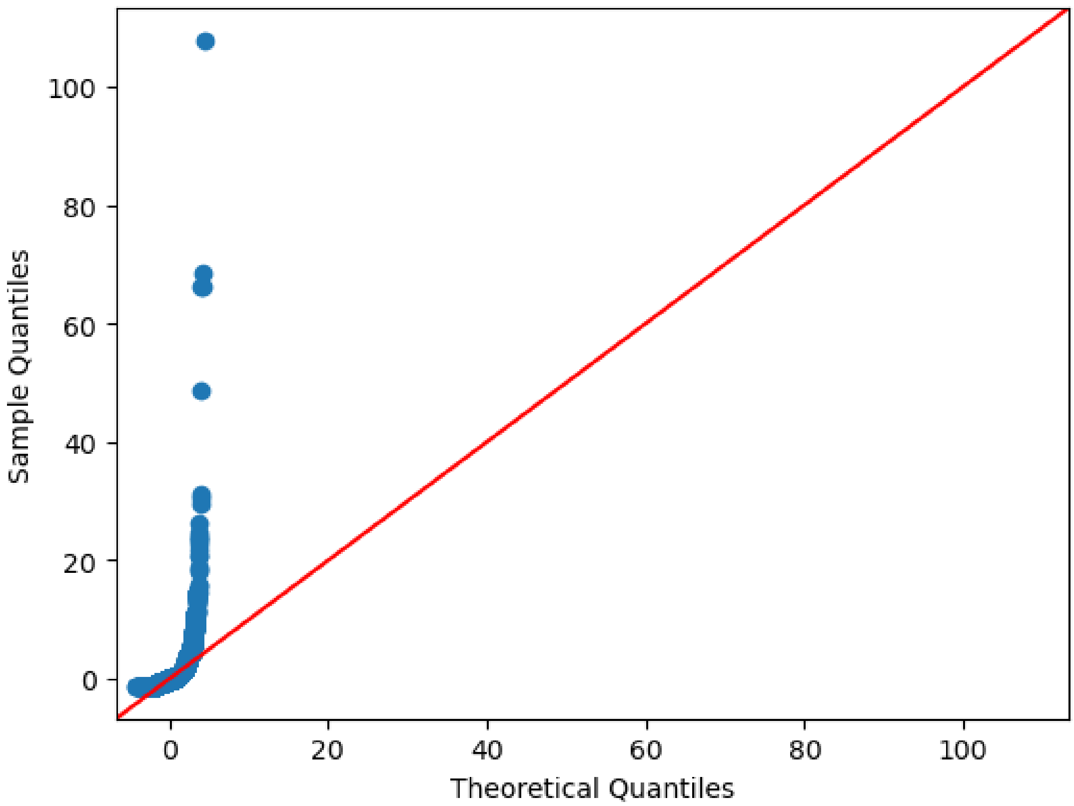


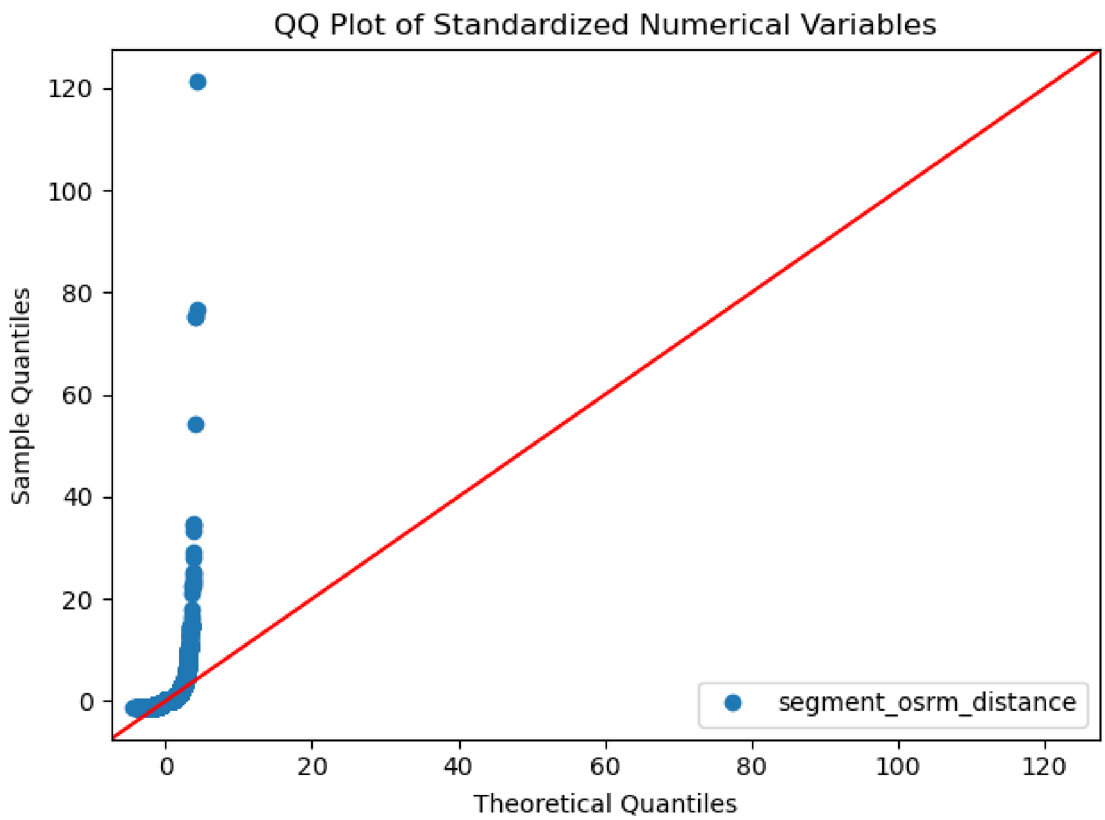
6
4
2
0
−2
−4
Sample Quantiles
−4
−2
0
2
4
6
Theoretical Quantiles


Sample Quantiles
Theoretical Quantiles
4
2
0
−2
−4
−4
−2
0
2
4

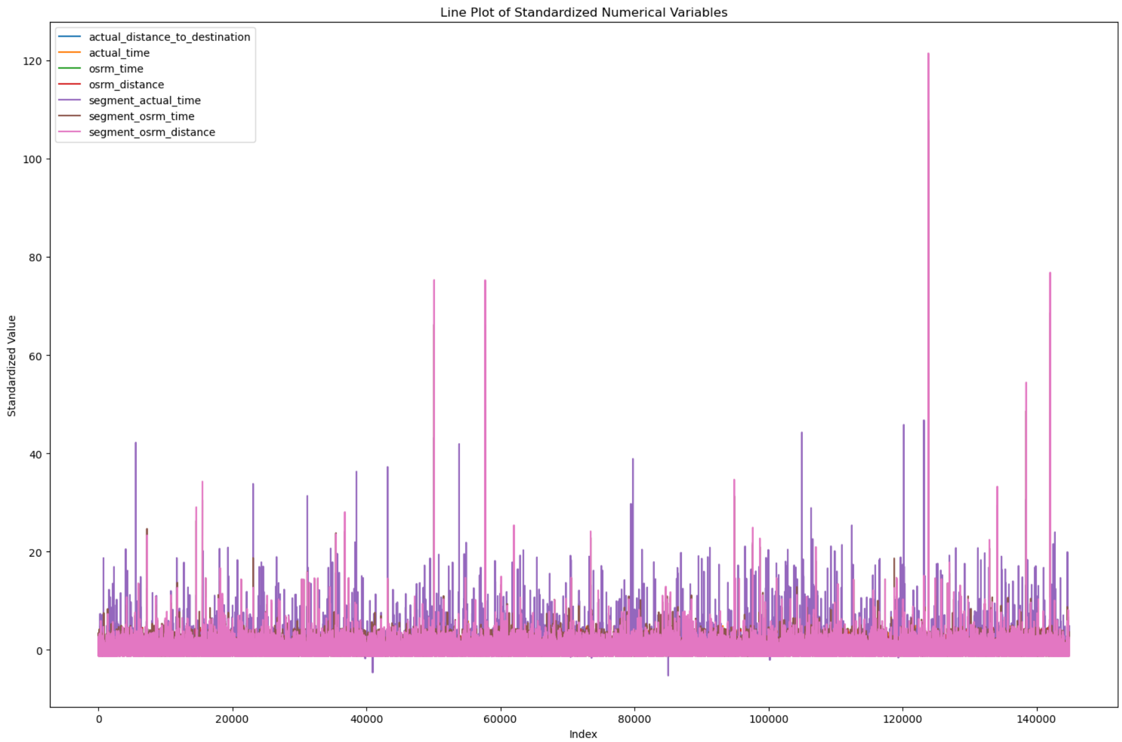

Sample Quantiles
Theoretical Quantiles
4
2
0
−2
−4
−4
−2
0
2
4


Sample Quantiles
Theoretical Quantiles
0
10
20
30
40
50


100
80
60
40
20
0
Sample Quantiles
0
20
40
60
80
100
Theoretical Quantiles



[150]:
```
# Plot line plots for visualizing the spread of standardized variables
plt.figure(figsize=(15, 10))
for col in numerical_columns:
    plt.plot(df_standardized[col], label=col)
plt.title('Line Plot of Standardized Numerical Variables')
plt.xlabel('Index')
plt.ylabel('Standardized Value')
plt.legend()
plt.tight_layout()
plt.show()
```



7.3 Based on our analysis, here are the business insights derived from the dataset:

1. Temporal Analysis:

Trips show a clear daily pattern, with a peak around 10 P.M. This suggests that customers are more active in placing orders during the evening hours. The 38th week sees the highest number of trips, indicating a potential seasonal trend or peak period for deliveries. Mid-month sees a surge in orders, suggesting that customers tend to make more purchases during this time.

2. Geographical Insights:

Orders primarily originate from states like Maharashtra, Karnataka, Haryana, Tamilnadu, and Telangana, indicating strong market presence and customer base in these regions. Cities like Mumbai, Gurgaon, Bengaluru, and Delhi serve as major hubs for both the origin and destination of trips, highlighting their significance in terms of customer demand and seller presence.

3. Route Analysis:

The most common route type is Carting, indicating a prevalent transportation mode for deliveries. Significant numbers of trips end in states like Maharashtra, Karnataka, Haryana, Tamilnadu, and Uttar Pradesh, suggesting high demand and delivery activity in these regions.

4. Customer Behavior:

Most orders are sourced from cities like Mumbai, Bengaluru, Gurgaon, Delhi, and Chennai, indicating strong customer engagement and demand in these urban centers. Orders peak mid-month, indicating potential factors such as payday or monthly shopping patterns.

5. Feature Analysis:

Features such as start_scan_to_end_scan and od_total_time exhibit statistical similarity, suggesting that they may capture similar aspects of the delivery process. Conversely, features like actual_time and osrm_time show statistical differences, indicating variations in the calculation or representation of delivery times.

6. Missing Data:

Names of 14 unique location IDs are missing in the dataset, which may require further investigation or data enrichment efforts to ensure comprehensive analysis and insights.

7.3.1 Here are some actionable recommendations for the business:

1. Improve OSRM Trip Planning System:

Evaluate and enhance the OSRM trip planning system to ensure more accurate route predictions and minimize discrepancies. Collaborate with transporters to address any routing engine configurations that may affect delivery times.

2. Reduce Discrepancies Between OSRM Time and Actual Time:

Work towards minimizing the difference between OSRM time and actual time to improve delivery time prediction accuracy. Implement measures to ensure consistency in delivery times, enhancing customer satisfaction and trust.

3. Optimize Corridors for High-Demand States:

Focus on optimizing existing corridors to improve penetration and service quality in states with high order volumes, such as Maharashtra, Karnataka, Haryana, and Tamil Nadu. Invest in infrastructure and logistics to streamline delivery operations and meet growing demand in these regions.

4. Customer Profiling and Engagement:

Conduct customer profiling for customers in high-demand states to understand their preferences, behavior, and needs better. Tailor marketing strategies and service offerings to cater to the specific requirements of customers in these regions, enhancing overall customer experience.

5. Traffic and Terrain Analysis:

Analyze traffic patterns and terrain conditions in different states to anticipate challenges and plan logistics more effectively. Allocate resources strategically during peak festival seasons to ensure timely deliveries and minimize disruptions.

6. Continuous Monitoring and Improvement:

Regularly monitor key performance indicators (KPIs) related to delivery times, customer satisfaction, and order volumes. Implement a feedback mechanism to gather insights from customers and stakeholders, enabling continuous improvement in service quality and operational efficiency.