In [15]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import scipy.stats as stats
```

In [16]:
```
#reading the file for analysis
data = pd.read_csv('delhivery_data.csv')
```

```
<ipython-input-16-bcc7edab74c4>:2: DtypeWarning: Columns (12) have mixed types. Spec
ify dtype option on import or set low_memory=False.
  data = pd.read_csv('delhivery_data.csv')
```

In [17]:
```
#Analyse the data types of the features present.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114711 entries, 0 to 114710
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   data                            114711 non-null  object 
 1   trip_creation_time              114711 non-null  object 
 2   route_schedule_uuid             114711 non-null  object 
 3   route_type                      114710 non-null  object 
 4   trip_uuid                       114710 non-null  object 
 5   source_center                   114710 non-null  object 
 6   source_name                     114498 non-null  object 
 7   destination_center              114710 non-null  object 
 8   destination_name                114544 non-null  object 
 9   od_start_time                   114710 non-null  object 
 10  od_end_time                     114710 non-null  object 
 11  start_scan_to_end_scan          114710 non-null  float64
 12  is_cutoff                       114710 non-null  object 
 13  cutoff_factor                   114710 non-null  float64
 14  cutoff_timestamp                114710 non-null  object 
 15  actual_distance_to_destination  114710 non-null  float64
 16  actual_time                     114710 non-null  float64
 17  osrm_time                       114710 non-null  float64
 18  osrm_distance                   114710 non-null  float64
 19  factor                          114710 non-null  float64
 20  segment_actual_time             114710 non-null  float64
 21  segment_osrm_time               114710 non-null  float64
 22  segment_osrm_distance           114710 non-null  float64
 23  segment_factor                  114710 non-null  float64
dtypes: float64(11), object(13)
memory usage: 21.0+ MB
```

In [18]:
```
data.head()
```

Out[18]:

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

5 rows × 24 columns

In [19]:

```
# check for missing values
data.isnull().sum()
```

Out[19]:

	0
data	0
trip_creation_time	0
route_schedule_uuid	0
route_type	1
trip_uuid	1
source_center	1
source_name	213
destination_center	1
destination_name	167
od_start_time	1
od_end_time	1
start_scan_to_end_scan	1
is_cutoff	1
cutoff_factor	1
cutoff_timestamp	1
actual_distance_to_destination	1
actual_time	1
osrm_time	1
osrm_distance	1
factor	1
segment_actual_time	1
segment_osrm_time	1
segment_osrm_distance	1
segment_factor	1

dtype: int64

In [20]:

```
# find the missing values percentage

missing_value = pd.DataFrame({
    'Missing Value': data.isnull().sum(),
    'Percentage': (data.isnull().sum() / len(data))*100
})
missing_value.sort_values(by='Percentage', ascending=False)
```

Out[20]:

	Missing Value	Percentage
source_name	213	0.185684
destination_name	167	0.145583
is_cutoff	1	0.000872
cutoff_factor	1	0.000872
segment_osrm_distance	1	0.000872
segment_osrm_time	1	0.000872
segment_actual_time	1	0.000872
factor	1	0.000872
osrm_distance	1	0.000872
osrm_time	1	0.000872
actual_time	1	0.000872
actual_distance_to_destination	1	0.000872
cutoff_timestamp	1	0.000872
segment_factor	1	0.000872
start_scan_to_end_scan	1	0.000872
od_end_time	1	0.000872
od_start_time	1	0.000872
destination_center	1	0.000872
source_center	1	0.000872
trip_uuid	1	0.000872
route_type	1	0.000872
trip_creation_time	0	0.000000
route_schedule_uuid	0	0.000000
data	0	0.000000

In [21]:

```
#getting the statistical values for numerical values
df = data.select_dtypes(include=['float64'])
df.drop(['cutoff_factor','factor','segment_factor'],axis=1,inplace=True) # dropping
df.describe()
```

Out[21]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time
count	114710.000000	114710.000000	114710.000000	114710.000000
mean	964.170700	234.146989	417.871145	213.941269
std	1043.348633	345.092034	600.806756	308.262358
min	20.000000	9.000055	9.000000	6.000000
25%	161.000000	23.353696	51.000000	27.000000
50%	447.000000	66.100107	132.000000	64.000000
75%	1625.000000	286.829623	514.000000	257.000000
max	4535.000000	1927.447705	4532.000000	1686.000000

In [22]:

```
#Updating the columns with most na with most-frequent value as they are categorical
cat_missing = ['source_name','destination_name']

freq_imputer = SimpleImputer(strategy = 'most_frequent') # mode
for col in cat_missing:
    data[col] = pd.DataFrame(freq_imputer.fit_transform(pd.DataFrame(data[col])))
```

In [23]:

```
#since the rest of the columns with na has only 1 entry, their impact on the outcom
data.dropna(inplace=True)
```

In [24]:

```
#updating the date related columns to panda datatime format.
cat_missing = ['trip_creation_time', 'od_start_time', 'od_end_time']

for col in cat_missing:
    data[col] = pd.to_datetime(data[col], format='mixed')
```

In [25]:

```
#Updated data types
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 114710 entries, 0 to 114709
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            114710 non-null  object
 1   trip_creation_time              114710 non-null  datetime64[ns]
 2   route_schedule_uuid             114710 non-null  object
 3   route_type                      114710 non-null  object
 4   trip_uuid                       114710 non-null  object
 5   source_center                   114710 non-null  object
 6   source_name                     114710 non-null  object
 7   destination_center              114710 non-null  object
 8   destination_name                114710 non-null  object
 9   od_start_time                   114710 non-null  datetime64[ns]
 10  od_end_time                     114710 non-null  datetime64[ns]
 11  start_scan_to_end_scan          114710 non-null  float64
 12  is_cutoff                       114710 non-null  object
 13  cutoff_factor                   114710 non-null  float64
 14  cutoff_timestamp                114710 non-null  object
 15  actual_distance_to_destination  114710 non-null  float64
 16  actual_time                     114710 non-null  float64
 17  osrm_time                       114710 non-null  float64
 18  osrm_distance                   114710 non-null  float64
 19  factor                          114710 non-null  float64
 20  segment_actual_time             114710 non-null  float64
 21  segment_osrm_time               114710 non-null  float64
 22  segment_osrm_distance           114710 non-null  float64
 23  segment_factor                  114710 non-null  float64
dtypes: datetime64[ns](3), float64(11), object(10)
memory usage: 21.9+ MB
```

In [26]:
```python
#creating new column segment key as to group the
data['segment_key'] = data['trip_uuid'] +' - ' + data ['source_center'] +' - ' +  d
```

In [27]:
```python
# aggregation dictionary
create_segment_dict = {
    'segment_osrm_distance': 'sum',
    # For categorical columns keeping first value.
    'trip_uuid': 'first',
    'source_center': 'first',
    'destination_center': 'first',
    # For datetime columns : keeping first and last values.
    'od_start_time': ['first', 'last'],
    'od_end_time': ['first', 'last'],
    'segment_actual_time':  ['first', 'last'],
        'segment_osrm_time':  ['first', 'last']
}

# Further group by segment_key for detailed aggregation
final_aggregated = data.groupby('segment_key').agg(create_segment_dict)

# Step 4: Sort the resulting DataFrame
final_aggregated.sort_values(by=['segment_key', ('od_end_time','last')], ascending=
```

```
# Display the final aggregated DataFrame
print(final_aggregated)
```

```
                                                  segment_osrm_distance  \
                                                                    sum
segment_key
trip-153671042288605164 - IND561203AAB - IND562...              28.1995
trip-153671042288605164 - IND572101AAA - IND561...              55.9899
trip-153671046011330457 - IND400072AAB - IND401...              19.8766
trip-153671052974046625 - IND583101AAA - IND583...              63.6461
trip-153671052974046625 - IND583119AAA - IND583...              53.5761
...                                                                 ...
trip-153861115439069069 - IND628204AAA - IND627...              42.1431
trip-153861115439069069 - IND628613AAA - IND627...              78.5869
trip-153861115439069069 - IND628801AAA - IND628...              16.0184
trip-153861118270144424 - IND583119AAA - IND583...              52.5303
trip-153861118270144424 - IND583201AAA - IND583...              28.0484

                                                                trip_uuid  \
                                                                    first
segment_key
trip-153671042288605164 - IND561203AAB - IND562...  trip-153671042288605164
trip-153671042288605164 - IND572101AAA - IND561...  trip-153671042288605164
trip-153671046011330457 - IND400072AAB - IND401...  trip-153671046011330457
trip-153671052974046625 - IND583101AAA - IND583...  trip-153671052974046625
trip-153671052974046625 - IND583119AAA - IND583...  trip-153671052974046625
...                                                                     ...
trip-153861115439069069 - IND628204AAA - IND627...  trip-153861115439069069
trip-153861115439069069 - IND628613AAA - IND627...  trip-153861115439069069
trip-153861115439069069 - IND628801AAA - IND628...  trip-153861115439069069
trip-153861118270144424 - IND583119AAA - IND583...  trip-153861118270144424
trip-153861118270144424 - IND583201AAA - IND583...  trip-153861118270144424

                                                  source_center  \
                                                          first
segment_key
trip-153671042288605164 - IND561203AAB - IND562...  IND561203AAB
trip-153671042288605164 - IND572101AAA - IND561...  IND572101AAA
trip-153671046011330457 - IND400072AAB - IND401...  IND400072AAB
trip-153671052974046625 - IND583101AAA - IND583...  IND583101AAA
trip-153671052974046625 - IND583119AAA - IND583...  IND583119AAA
...                                                          ...
trip-153861115439069069 - IND628204AAA - IND627...  IND628204AAA
trip-153861115439069069 - IND628613AAA - IND627...  IND628613AAA
trip-153861115439069069 - IND628801AAA - IND628...  IND628801AAA
trip-153861118270144424 - IND583119AAA - IND583...  IND583119AAA
trip-153861118270144424 - IND583201AAA - IND583...  IND583201AAA

                                                  destination_center  \
                                                               first
segment_key
trip-153671042288605164 - IND561203AAB - IND562...       IND562101AAA
trip-153671042288605164 - IND572101AAA - IND561...       IND561203AAB
trip-153671046011330457 - IND400072AAB - IND401...       IND401104AAA
trip-153671052974046625 - IND583101AAA - IND583...       IND583201AAA
trip-153671052974046625 - IND583119AAA - IND583...       IND583101AAA
...                                                               ...
trip-153861115439069069 - IND628204AAA - IND627...       IND627657AAA
trip-153861115439069069 - IND628613AAA - IND627...       IND627005AAA
```

```
trip-153861115439069069 - IND628801AAA - IND628...       IND628204AAA
trip-153861118270144424 - IND583119AAA - IND583...       IND583101AAA
trip-153861118270144424 - IND583201AAA - IND583...       IND583119AAA

                                                            od_start_time  \
                                                                    first
segment_key
trip-153671042288605164 - IND561203AAB - IND562... 2018-09-12 02:03:09.655591
trip-153671042288605164 - IND572101AAA - IND561... 2018-09-12 00:00:22.886430
trip-153671046011330457 - IND400072AAB - IND401... 2018-09-12 00:01:00.113710
trip-153671052974046625 - IND583101AAA - IND583... 2018-09-12 00:02:09.740725
trip-153671052974046625 - IND583119AAA - IND583... 2018-09-12 03:54:43.114421
...                                                                       ...
trip-153861115439069069 - IND628204AAA - IND627... 2018-10-04 02:29:04.272194
trip-153861115439069069 - IND628613AAA - IND627... 2018-10-04 04:16:39.894872
trip-153861115439069069 - IND628801AAA - IND628... 2018-10-04 01:44:53.808000
trip-153861118270144424 - IND583119AAA - IND583... 2018-10-04 03:58:40.726547
trip-153861118270144424 - IND583201AAA - IND583... 2018-10-04 02:51:44.712656

                                                                              \
                                                                     last
segment_key
trip-153671042288605164 - IND561203AAB - IND562... 2018-09-12 02:03:09.655591
trip-153671042288605164 - IND572101AAA - IND561... 2018-09-12 00:00:22.886430
trip-153671046011330457 - IND400072AAB - IND401... 2018-09-12 00:01:00.113710
trip-153671052974046625 - IND583101AAA - IND583... 2018-09-12 00:02:09.740725
trip-153671052974046625 - IND583119AAA - IND583... 2018-09-12 03:54:43.114421
...                                                                       ...
trip-153861115439069069 - IND628204AAA - IND627... 2018-10-04 02:29:04.272194
trip-153861115439069069 - IND628613AAA - IND627... 2018-10-04 04:16:39.894872
trip-153861115439069069 - IND628801AAA - IND628... 2018-10-04 01:44:53.808000
trip-153861118270144424 - IND583119AAA - IND583... 2018-10-04 03:58:40.726547
trip-153861118270144424 - IND583201AAA - IND583... 2018-10-04 02:51:44.712656

                                                              od_end_time  \
                                                                    first
segment_key
trip-153671042288605164 - IND561203AAB - IND562... 2018-09-12 03:01:59.598855
trip-153671042288605164 - IND572101AAA - IND561... 2018-09-12 02:03:09.655591
trip-153671046011330457 - IND400072AAB - IND401... 2018-09-12 01:41:29.809822
trip-153671052974046625 - IND583101AAA - IND583... 2018-09-12 02:34:10.515593
trip-153671052974046625 - IND583119AAA - IND583... 2018-09-12 12:00:30.683231
...                                                                       ...
trip-153861115439069069 - IND628204AAA - IND627... 2018-10-04 03:31:11.183797
trip-153861115439069069 - IND628613AAA - IND627... 2018-10-04 05:47:45.162682
trip-153861115439069069 - IND628801AAA - IND628... 2018-10-04 02:29:04.272194
trip-153861118270144424 - IND583119AAA - IND583... 2018-10-04 08:46:09.166940
trip-153861118270144424 - IND583201AAA - IND583... 2018-10-04 03:58:40.726547

                                                                              \
                                                                     last
segment_key
trip-153671042288605164 - IND561203AAB - IND562... 2018-09-12 03:01:59.598855
trip-153671042288605164 - IND572101AAA - IND561... 2018-09-12 02:03:09.655591
trip-153671046011330457 - IND400072AAB - IND401... 2018-09-12 01:41:29.809822
trip-153671052974046625 - IND583101AAA - IND583... 2018-09-12 02:34:10.515593
```

```
trip-153671052974046625 - IND583119AAA - IND583... 2018-09-12 12:00:30.683231
...                                                                         ...
trip-153861115439069069 - IND628204AAA - IND627... 2018-10-04 03:31:11.183797
trip-153861115439069069 - IND628613AAA - IND627... 2018-10-04 05:47:45.162682
trip-153861115439069069 - IND628801AAA - IND628... 2018-10-04 02:29:04.272194
trip-153861118270144424 - IND583119AAA - IND583... 2018-10-04 08:46:09.166940
trip-153861118270144424 - IND583201AAA - IND583... 2018-10-04 03:58:40.726547

                                                   segment_actual_time         \
                                                                 first   last
segment_key
trip-153671042288605164 - IND561203AAB - IND562...                18.0   15.0
trip-153671042288605164 - IND572101AAA - IND561...                14.0   20.0
trip-153671046011330457 - IND400072AAB - IND401...                23.0   36.0
trip-153671052974046625 - IND583101AAA - IND583...                42.0   59.0
trip-153671052974046625 - IND583119AAA - IND583...                51.0   79.0
...                                                                ...    ...
trip-153861115439069069 - IND628204AAA - IND627...                 9.0   11.0
trip-153861115439069069 - IND628613AAA - IND627...                15.0   51.0
trip-153861115439069069 - IND628801AAA - IND628...                21.0    8.0
trip-153861118270144424 - IND583119AAA - IND583...                45.0  188.0
trip-153861118270144424 - IND583201AAA - IND583...                30.0   11.0

                                                   segment_osrm_time
                                                               first  last
segment_key
trip-153671042288605164 - IND561203AAB - IND562...              10.0   7.0
trip-153671042288605164 - IND572101AAA - IND561...               8.0   3.0
trip-153671046011330457 - IND400072AAB - IND401...               9.0   7.0
trip-153671052974046625 - IND583101AAA - IND583...              17.0  12.0
trip-153671052974046625 - IND583119AAA - IND583...              18.0  26.0
...                                                              ...   ...
trip-153861115439069069 - IND628204AAA - IND627...              10.0   8.0
trip-153861115439069069 - IND628613AAA - IND627...              13.0  47.0
trip-153861115439069069 - IND628801AAA - IND628...               8.0   6.0
trip-153861118270144424 - IND583119AAA - IND583...              17.0  25.0
trip-153861118270144424 - IND583201AAA - IND583...              21.0   4.0

[20885 rows x 12 columns]
```

In [28]:
```
data.head()
```

Out[28]:

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

5 rows × 25 columns

In [29]:

```
# function to split city, place and area code from the given string.
def split_city_place_area_state(string_to_split):
  #initialise variable that would be return value holders
  city, place, area = '','',''
  split_string_arr = string_to_split.split('(')
  city_place_area = split_string_arr[0] # as to avoid taking state related values.
  state = split_string_arr[1].split(')')[0]
  if '_' in city_place_area:
    temp_arr = city_place_area.split('_')
  else:
    temp_arr = city_place_area.split(' ')
  match len(temp_arr): #case statement is added as the length of the string varies
    case 1:
      city = temp_arr[0]
    case 2:
      city = temp_arr[0]
      place = temp_arr[1]
    case 3:
      city = temp_arr[0]
      place = temp_arr[1]
      area = temp_arr[2]
    case 4:
      city = temp_arr[0]
      place = temp_arr[1]
```

```
    area = temp_arr[2] + ' ' + temp_arr[3]
  return city, place, area, state
```

In [30]:
```
data[['Source_City', 'Source_Place', 'Source_Code', 'Source_State']] = data['source
data[['Dest_City', 'Dest_Place', 'Dest_Code', 'Dest_State']] = data['destination_na
data
```

Out[30]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uui
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
...	...	...	...	...	
114705	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403
114706	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403
114707	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403
114708	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403
114709	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403

114710 rows × 33 columns

In [31]:
```
#extracting features from trip creation time
data['trip_creation_month'] = data['trip_creation_time'].dt.month
data['trip_creation_year'] = data['trip_creation_time'].dt.year
data['trip_creation_day'] = data['trip_creation_time'].dt.day
data['trip_creation_hour'] = data['trip_creation_time'].dt.hour
data['trip_creation_minute'] = data['trip_creation_time'].dt.minute
```

In [32]:
```
#Creating feature od_time_diff_hour
data['od_time_diff_hour'] = (data['od_end_time'] - data['od_start_time'])
data
```

Out[32]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uui
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip 15374109364764932
...	...	...	...	...	
114705	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403
114706	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403
114707	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403
114708	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403
114709	test	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	Carting	trip 15380202135995403

114710 rows × 39 columns

In [33]:
```
final_aggregated
```

Out[33]:

	segment_osrm_distance	trip_uuid	source_center	destina
	sum	first	first	
segment_key				
trip-153671042288605164 - IND561203AAB - IND562101AAA	28.1995	trip-153671042288605164	IND561203AAB	IND
trip-153671042288605164 - IND572101AAA - IND561203AAB	55.9899	trip-153671042288605164	IND572101AAA	IND
trip-153671046011330457 - IND400072AAB - IND401104AAA	19.8766	trip-153671046011330457	IND400072AAB	IND
trip-153671052974046625 - IND583101AAA - IND583201AAA	63.6461	trip-153671052974046625	IND583101AAA	IND
trip-153671052974046625 - IND583119AAA - IND583101AAA	53.5761	trip-153671052974046625	IND583119AAA	IND
...	...	...	...	
trip-153861115439069069 - IND628204AAA - IND627657AAA	42.1431	trip-153861115439069069	IND628204AAA	IND
trip-153861115439069069 - IND628613AAA - IND627005AAA	78.5869	trip-153861115439069069	IND628613AAA	IND
trip-153861115439069069 - IND628801AAA - IND628204AAA	16.0184	trip-153861115439069069	IND628801AAA	IND
trip-153861118270144424 - IND583119AAA - IND583101AAA	52.5303	trip-153861118270144424	IND583119AAA	IND
trip-153861118270144424 - IND583201AAA - IND583119AAA	28.0484	trip-153861118270144424	IND583201AAA	IND

20885 rows × 12 columns

In [34]:

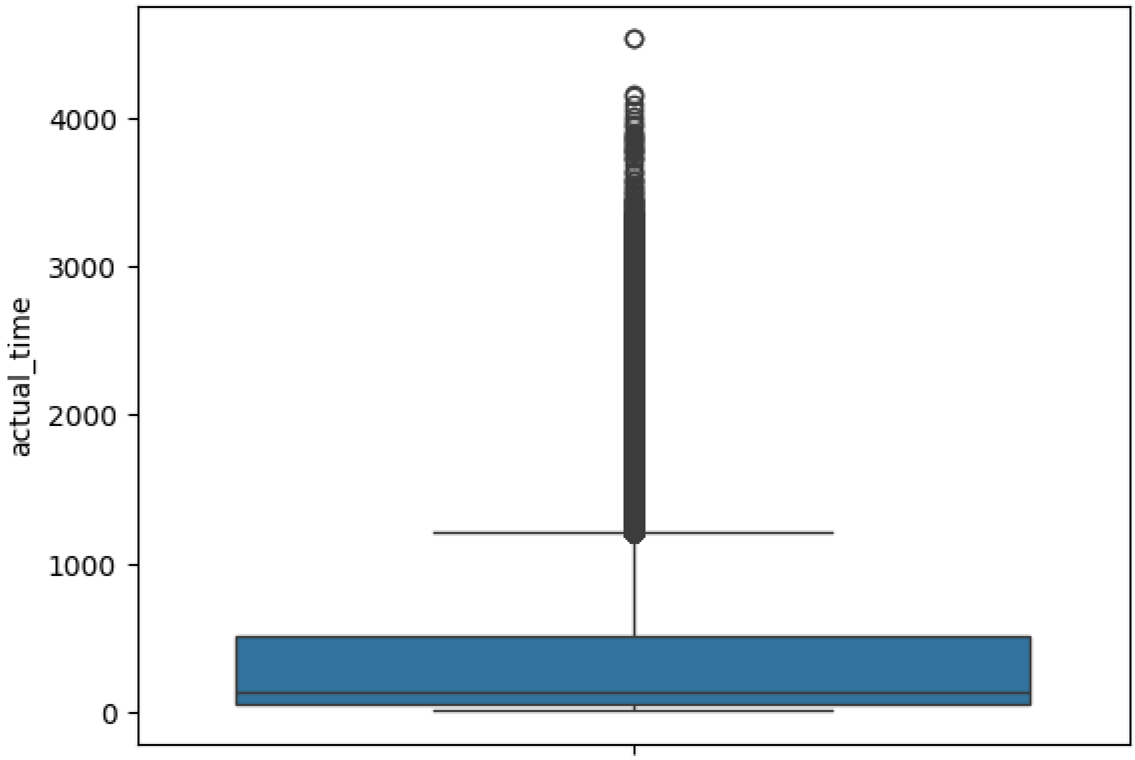
```
# Step 1: Define the aggregation dictionary
create_trip_dict = {
    'source_center': 'first',                            # Keep first source cen
    'destination_center': 'first',                       # Keep first destinatio
    'segment_actual_time':  'sum',                       #'sum' of segment actua
    'segment_osrm_time':  'sum',                         # 'sum' of OSRM time
    'segment_osrm_distance': 'sum',                      # Sum of OSRM distance
    'actual_distance_to_destination': 'first',           # Keep first actual dis
    'actual_time': 'sum',                                # 'sum' of actual time
    'osrm_time' :  'sum',                                # Sum of OSRM time
    'osrm_distance' : 'sum'                              # Sum of OSRM distance
}

# Step 2: Group by trip_uuid and apply aggregation
aggregated_trip_data = data.groupby('trip_uuid').agg(create_trip_dict)

# Reset index to flatten the DataFrame
aggregated_trip_data.reset_index(inplace=True)

# Step 3: Display the aggregated DataFrame
print(aggregated_trip_data)
```

```
                         trip_uuid source_center destination_center  \
0      trip-153671042288605164  IND572101AAA       IND561203AAB
1      trip-153671046011330457  IND400072AAB       IND401104AAA
2      trip-153671052974046625  IND583101AAA       IND583201AAA
3      trip-153671055416136166  IND600116AAB       IND600056AAA
4      trip-153671066201138152  IND600044AAD       IND600048AAA
...                        ...           ...                ...
11741  trip-153861095625827784  IND160002AAC       IND140603AAA
11742  trip-153861104386292051  IND121004AAB       IND121004AAA
11743  trip-153861106442901555  IND209304AAA       IND208006AAA
11744  trip-153861115439069069  IND627005AAA       IND628801AAA
11745  trip-153861118270144424  IND583201AAA       IND583119AAA

       segment_actual_time  segment_osrm_time  segment_osrm_distance  \
0                    141.0               65.0                84.1894
1                     59.0               16.0                19.8766
2                    340.0              115.0               146.7919
3                     60.0               23.0                28.0647
4                     24.0               13.0                12.0184
...                    ...                ...                    ...
11741                 82.0               62.0                64.8551
11742                 21.0               11.0                16.0883
11743                281.0               88.0               104.8866
11744                258.0              221.0               223.5324
11745                274.0               67.0                80.5787

       actual_distance_to_destination  actual_time  osrm_time  osrm_distance
0                            9.832310        399.0      210.0       269.4308
1                           11.354374         82.0       24.0        31.6475
2                           22.342846        556.0      207.0       266.2914
3                            9.271519         92.0       30.0        38.1953
4                            9.100510         24.0       13.0        12.0184
...                               ...          ...        ...            ...
11741                        9.226182        186.0      148.0       162.9473
11742                        9.616856         33.0       19.0        26.5333
11743                        9.336026        549.0      134.0       162.8499
11744                        9.107838        600.0      446.0       449.5383
11745                       22.156817        350.0      106.0       127.8020

[11746 rows x 10 columns]
```

In [35]:

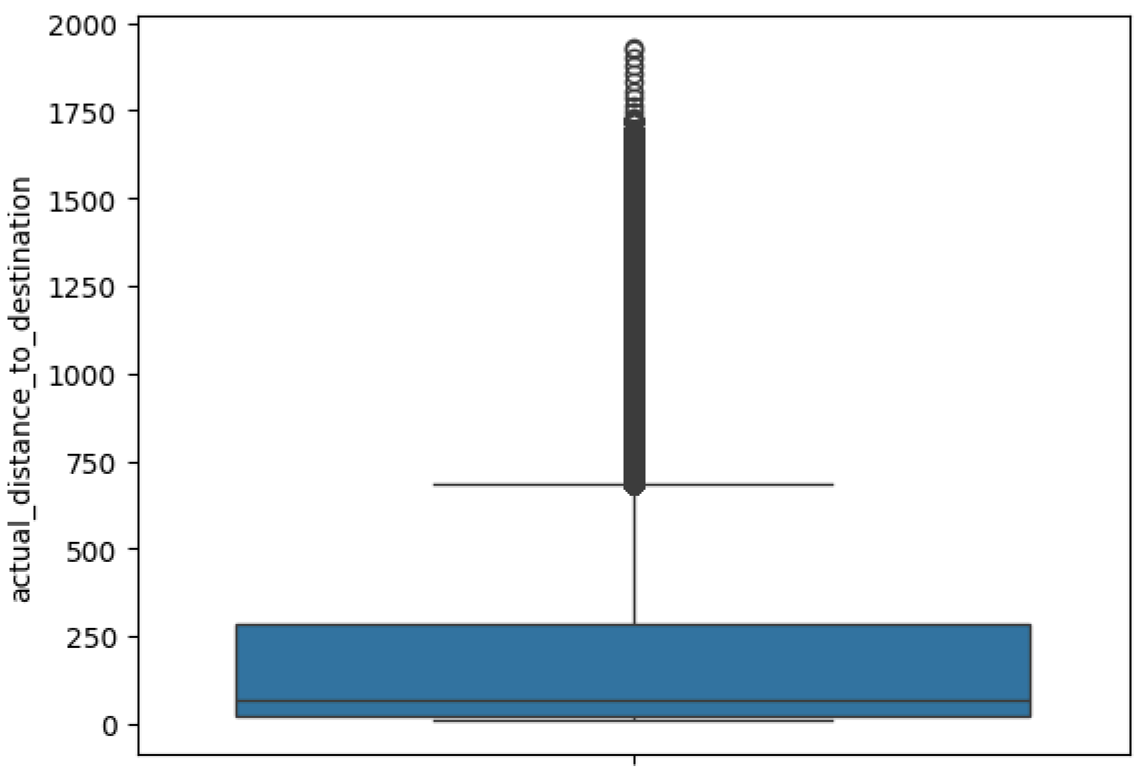
```
# Checking for outliers

sns.boxplot(data=data['actual_time'])
```

Out[35]: <Axes: ylabel='actual_time'>



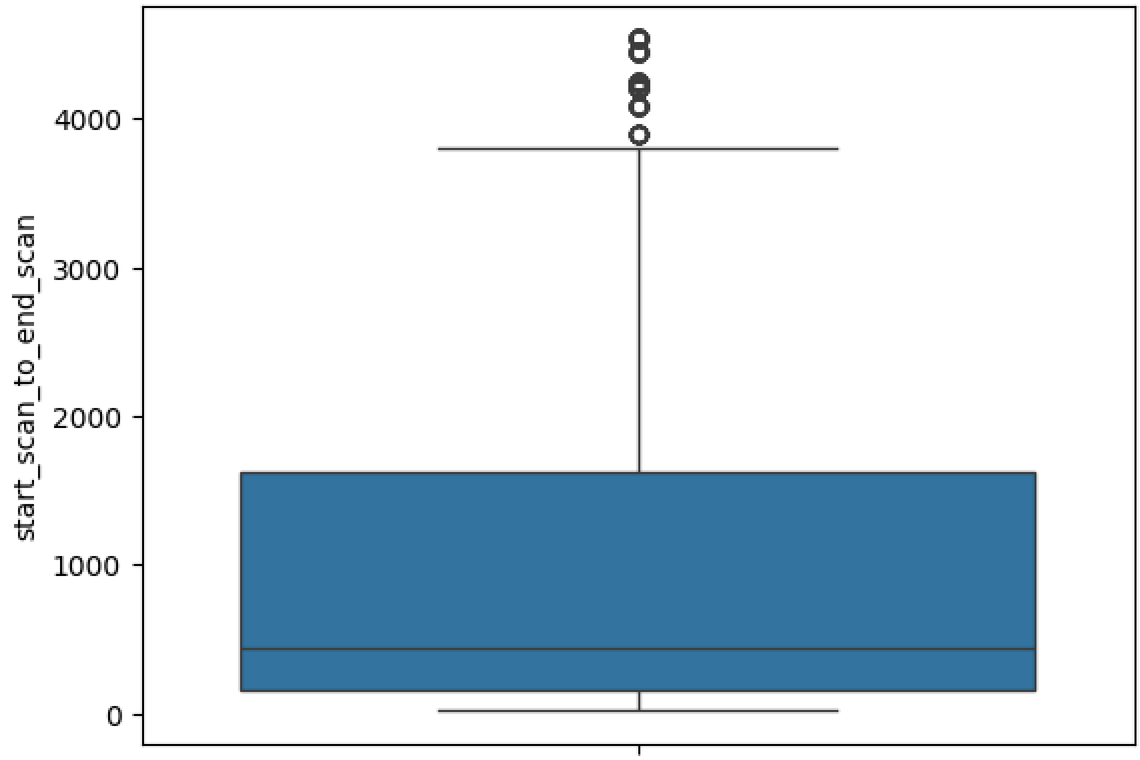
In [36]:
```
sns.boxplot(data=data['actual_distance_to_destination'])
```

Out[36]: <Axes: ylabel='actual_distance_to_destination'>



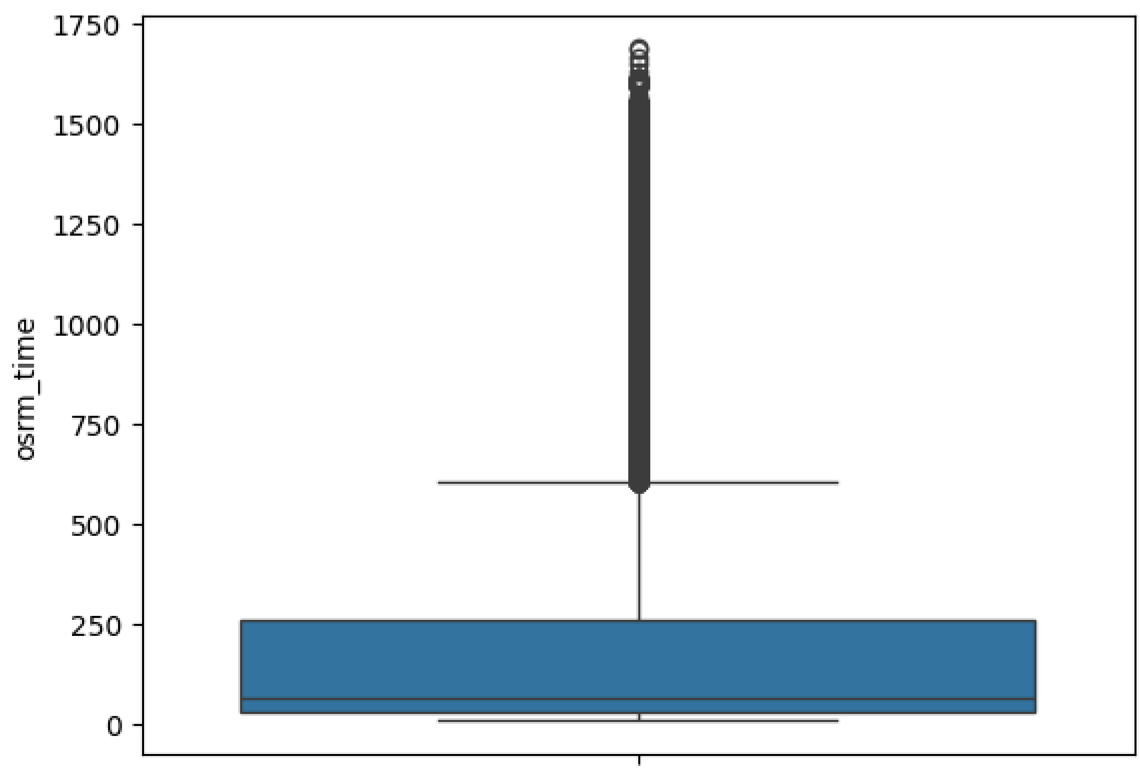
In [37]:
```
sns.boxplot(data=data['start_scan_to_end_scan'])
```

Out[37]: <Axes: ylabel='start_scan_to_end_scan'>



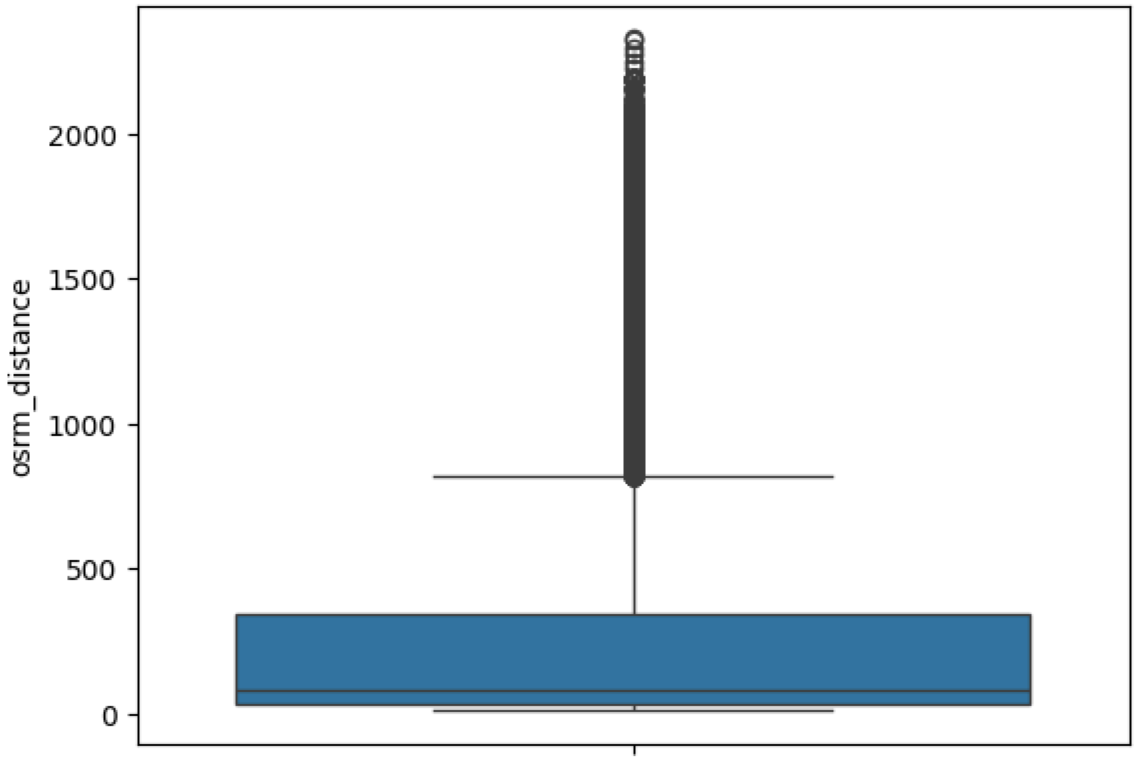
In [38]:
```
sns.boxplot(data=data['osrm_time'])
```

Out[38]: <Axes: ylabel='osrm_time'>



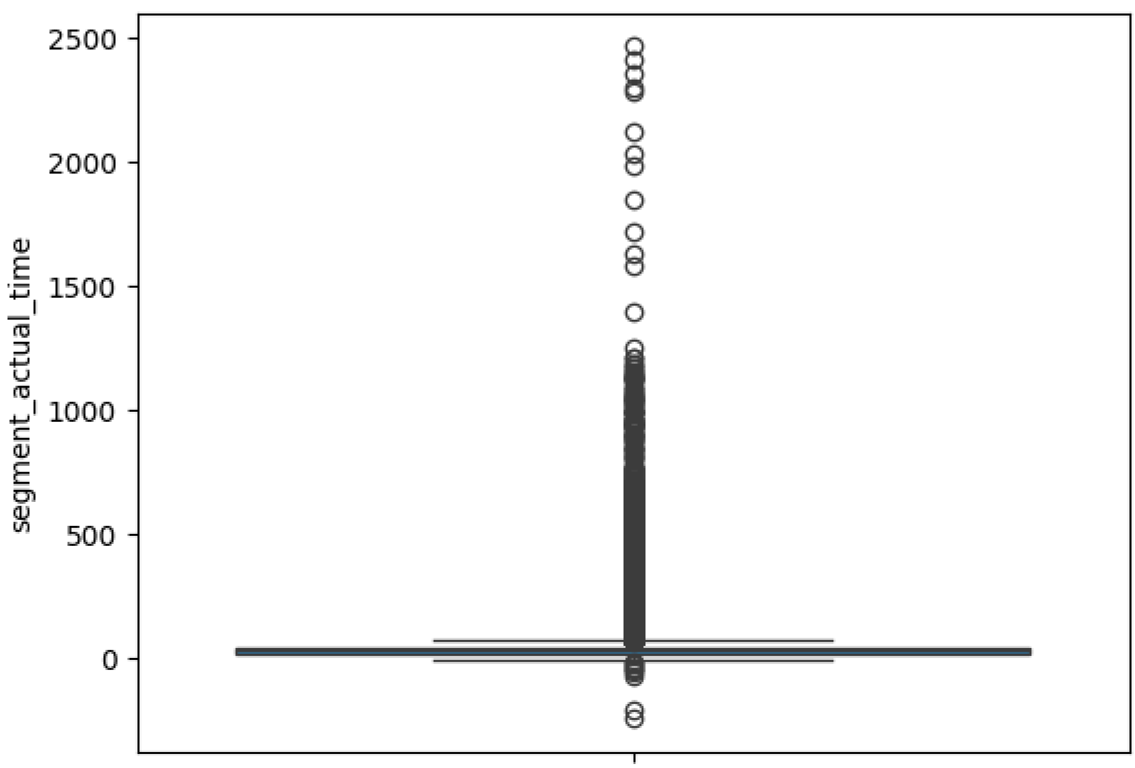
In [39]:
```
sns.boxplot(data=data['osrm_distance'])
```

Out[39]: <Axes: ylabel='osrm_distance'>



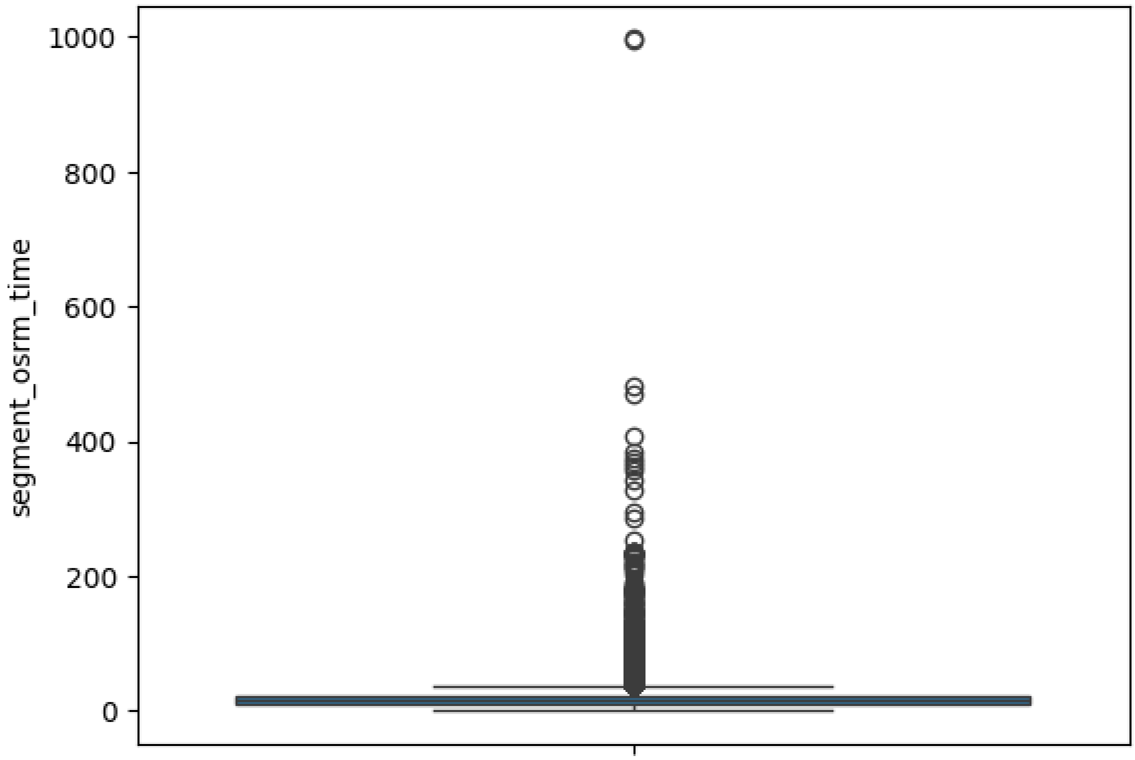
In [40]:
```python
sns.boxplot(data['segment_actual_time'])
```

Out[40]: <Axes: ylabel='segment_actual_time'>



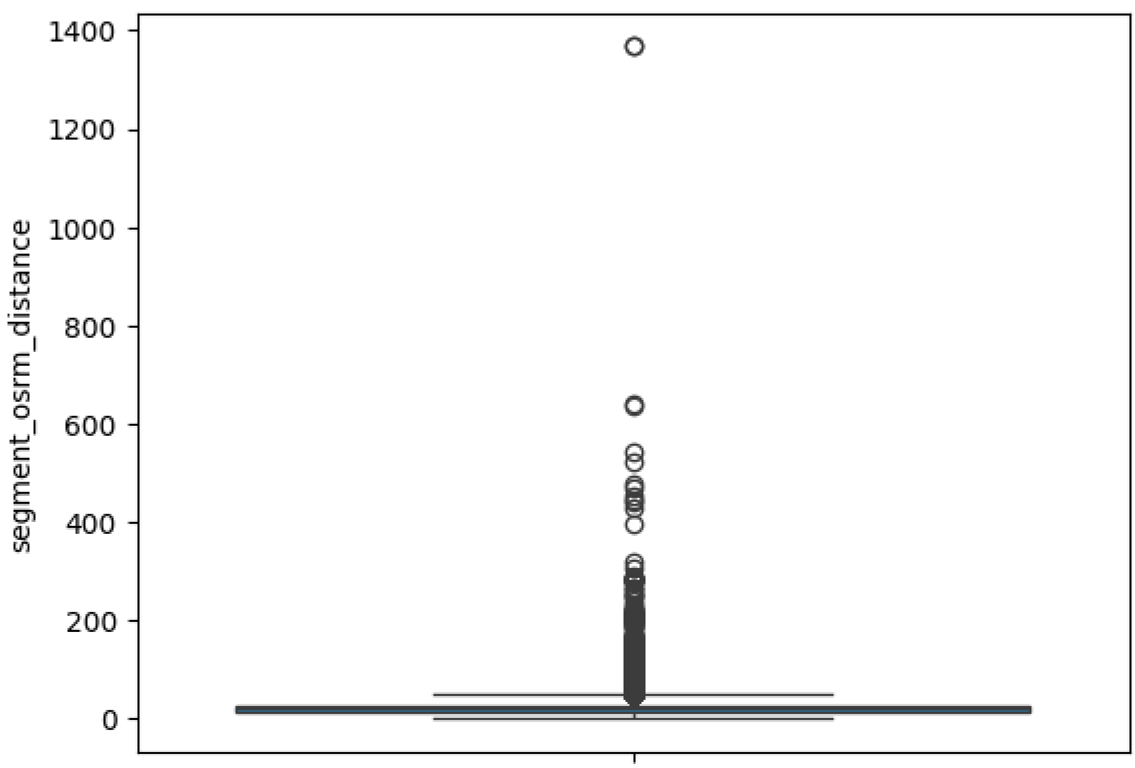
In [41]:
```python
sns.boxplot(data['segment_osrm_time'])
```

Out[41]: <Axes: ylabel='segment_osrm_time'>



In [42]:
```
sns.boxplot(data['segment_osrm_distance'])
```

Out[42]: <Axes: ylabel='segment_osrm_distance'>



In [43]:
```
#Checking whether the features that have outliers is gaussian in nature or not.
start_scan_to_end_scan = data['start_scan_to_end_scan']
osrm_time = data['osrm_time']
osrm_distance = data['osrm_distance']
```

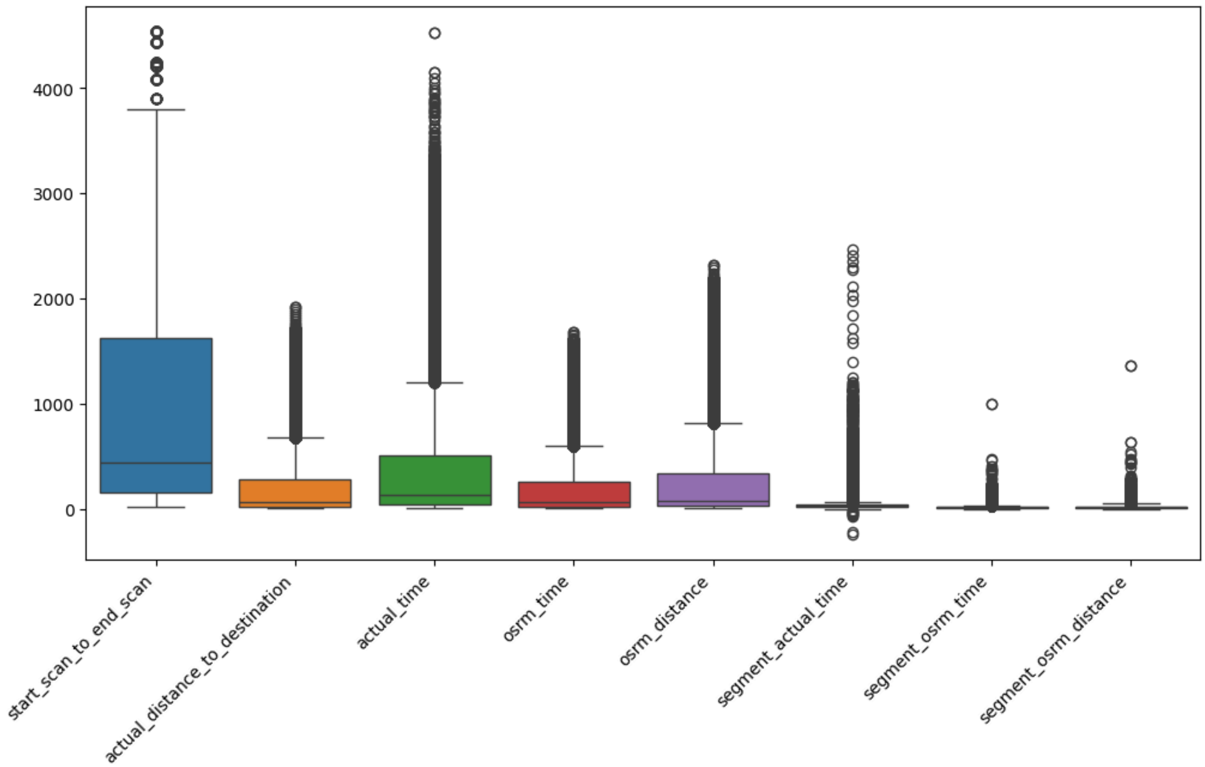
```
actual_distance_to_destination = data['actual_distance_to_destination']
actual_time = data['actual_time']

stat, p_value = stats.shapiro(start_scan_to_end_scan)
print('p-value -- start scan to end scan :', p_value)
if p_value < 0.05:
    print('start scan to end scan is not normal')
else:
    print('start scan to end scan is normal')

stat, p_value = stats.shapiro(osrm_time)
print('p-value osrm_time :', p_value)
if p_value < 0.05:
    print('osrm_time is not normal')
else:
    print('osrm_time is normal')

stat, p_value = stats.shapiro(osrm_distance)
print('p-value osrm_distance :', p_value)
if p_value < 0.05:
    print('osrm_distance is not normal')
else:
    print('osrm_distance is normal')

stat, p_value = stats.shapiro(actual_distance_to_destination)
print('p-value actual_distance_to_destination :', p_value)
if p_value < 0.05:
    print('actual_distance_to_destination is not normal')
else:
    print('actual_distance_to_destination is normal')

stat, p_value = stats.shapiro(actual_time)
print('p-value actual_time :', p_value)
if p_value < 0.05:
    print('actual_time is not normal')
else:
    print('actual_time is normal')
```

```
p-value -- start scan to end scan : 9.296664285919736e-135
start scan to end scan is not normal
p-value osrm_time : 1.225891382824286e-149
osrm_time is not normal
p-value osrm_distance : 3.5374504002939513e-150
osrm_distance is not normal
p-value actual_distance_to_destination : 7.871093328788817e-150
actual_distance_to_destination is not normal
p-value actual_time : 2.6214021289470624e-149
actual_time is not normal
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_axis_nan_policy.py:531: UserWar
ning: scipy.stats.shapiro: For N > 5000, computed p-value may not be accurate. Curre
nt N is 114710.
  res = hypotest_fun_out(*samples, **kwds)
```

In [44]:
```
#Box plot representation for the numerical variables.
df_num = data.select_dtypes(include=['float64', 'int64'])
df_num.drop(['cutoff_factor','factor','segment_factor'],axis=1,inplace=True) # drop
```

```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_num)
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [45]:
```
#Since the above distributions are not normal, IQR method of outlier treatment can
# obtain the first quartile
Q1 = df_num.quantile(0.25)
# obtain the third quartile
Q3 = df_num.quantile(0.75)
# obtain the IQR
IQR = Q3 - Q1
print(IQR)
print(data.shape)
```

```
start_scan_to_end_scan            1464.000000
actual_distance_to_destination     263.475927
actual_time                        463.000000
osrm_time                          230.000000
osrm_distance                      315.249850
segment_actual_time                 20.000000
segment_osrm_time                   11.000000
segment_osrm_distance               15.721050
dtype: float64
(114710, 39)
```

In [46]:
```
# Removal of all values above Q3 and below Q1
df_iqr=data[~((df_num < (Q1-1.5*IQR))|(df_num > (Q3 + 1.5*IQR))).any(axis=1)]
print(df_iqr.shape)
df_iqr.head()
```

```
(90289, 39)
```

Out[46]:

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

5 rows × 39 columns

In [47]:

```
# as distribution is not normal, we will have to do nromalisation.
num=df_iqr.select_dtypes(include=np.number)
num.drop(['cutoff_factor','factor','segment_factor'],axis=1,inplace=True) # droppin
num.head()
```

Out[47]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_dist
0	86.0	10.435660	14.0	11.0	11.!
1	86.0	18.936842	24.0	20.0	21.:
2	86.0	27.637279	40.0	28.0	32..
3	86.0	36.118028	62.0	40.0	45..
4	86.0	39.386040	68.0	44.0	54.:

In [48]:

```
#normalisation done for the numerical variable
# import MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
# instantiate the MinMaxScaler
min_max = MinMaxScaler()
# fit the MinMaxScaler
```

```
num['minmax_start_scan_to_end_scan'] = min_max.fit_transform(num[['start_scan_to_en
num['minmax_actual_distance_to_destination'] = min_max.fit_transform(num[['actual_d
num['minmax_osrm_time'] = min_max.fit_transform(num[['osrm_time']])
num['minmax_osrm_distance'] = min_max.fit_transform(num[['osrm_distance']])
num['minmax_actual_time'] = min_max.fit_transform(num[['actual_time']])
num['minmax_segment_actual_time'] = min_max.fit_transform(num[['segment_actual_time
num['minmax_segment_osrm_time'] = min_max.fit_transform(num[['segment_osrm_time']])
num['minmax_segment_osrm_distance'] = min_max.fit_transform(num[['segment_osrm_dist

# minimum and maximum value of the normalized variable
print(num['minmax_start_scan_to_end_scan'].min(), num['minmax_start_scan_to_end_sca
print(num['minmax_actual_distance_to_destination'].min(), num['minmax_actual_distan
print(num['minmax_osrm_time'].min(), num['minmax_osrm_time'].max())
print(num['minmax_osrm_distance'].min(), num['minmax_osrm_distance'].max())
print(num['minmax_actual_time'].min(), num['minmax_actual_time'].max())
print(num['minmax_segment_actual_time'].min(), num['minmax_segment_actual_time'].ma
print(num['minmax_segment_osrm_time'].min(), num['minmax_segment_osrm_time'].max())
print(num['minmax_segment_osrm_distance'].min(), num['minmax_segment_osrm_distance'
```

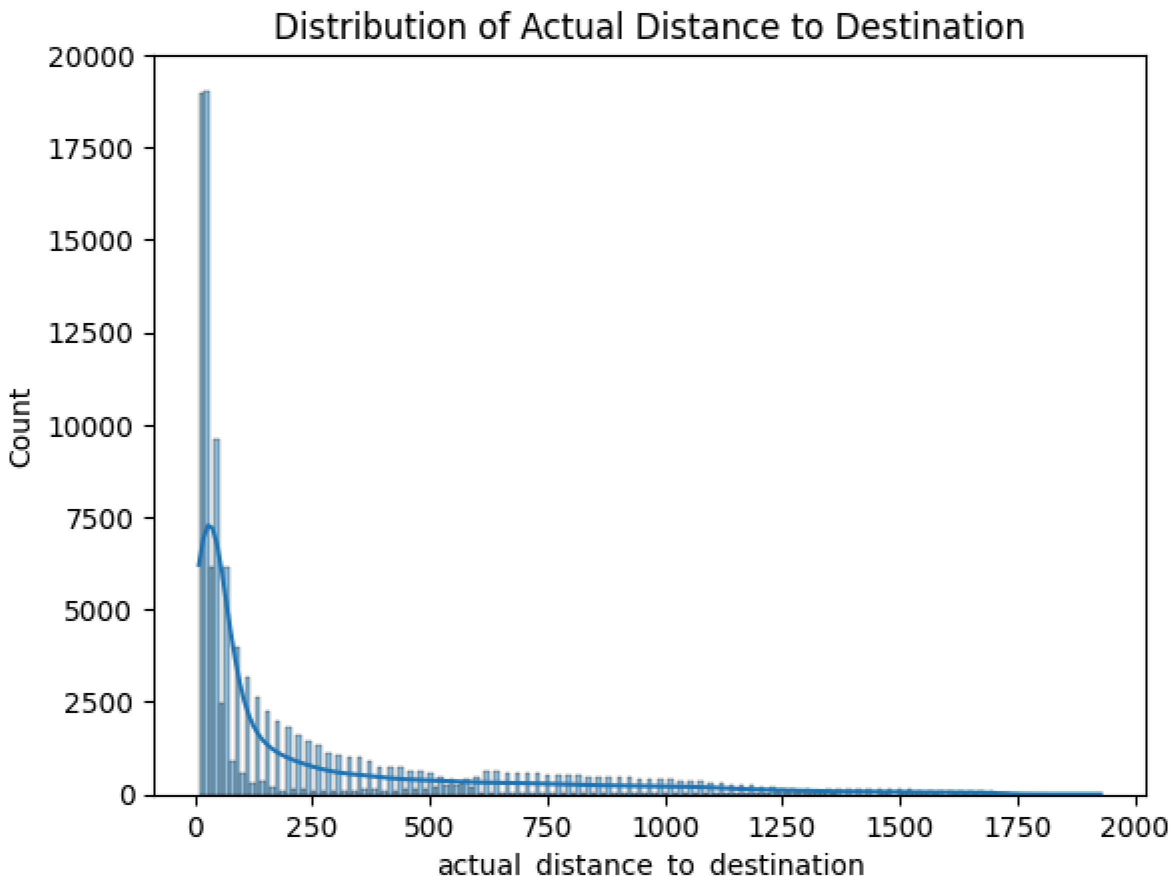
```
0.0 1.0
0.0 1.0
0.0 0.9999999999999999
0.0 1.0
0.0 1.0
0.0 1.0
0.0 1.0
0.0 1.0
```

In [48]:

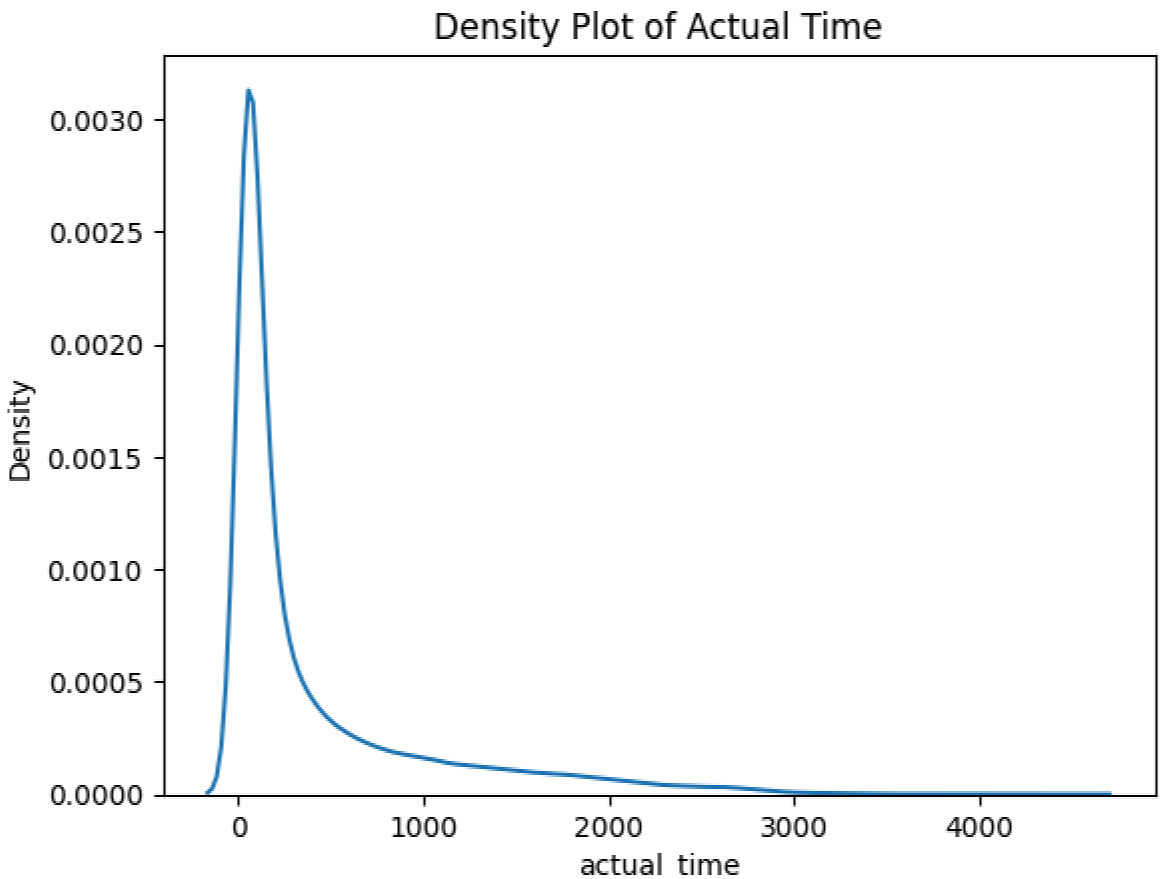
```
```

In [49]:

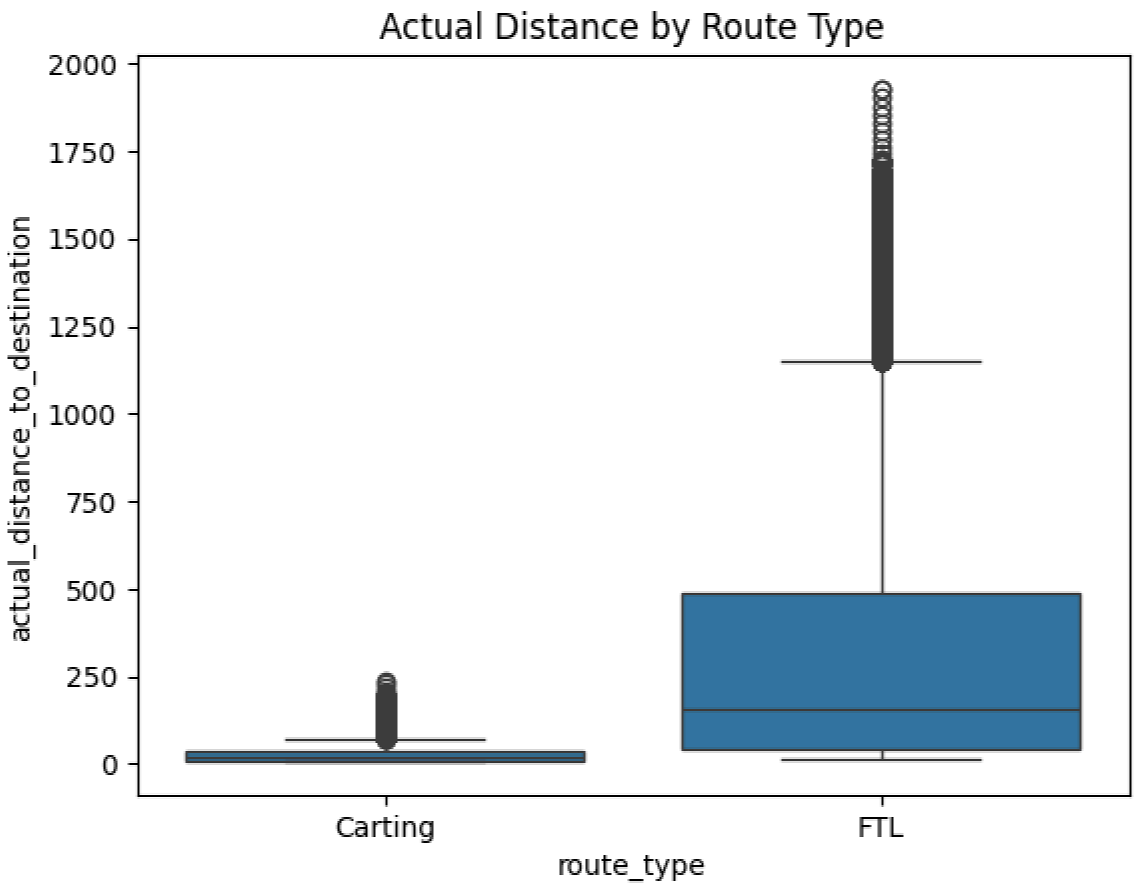
```
#visualisation of continuous variable
sns.histplot(data['actual_distance_to_destination'], kde=True)
plt.title('Distribution of Actual Distance to Destination')
plt.show()
```



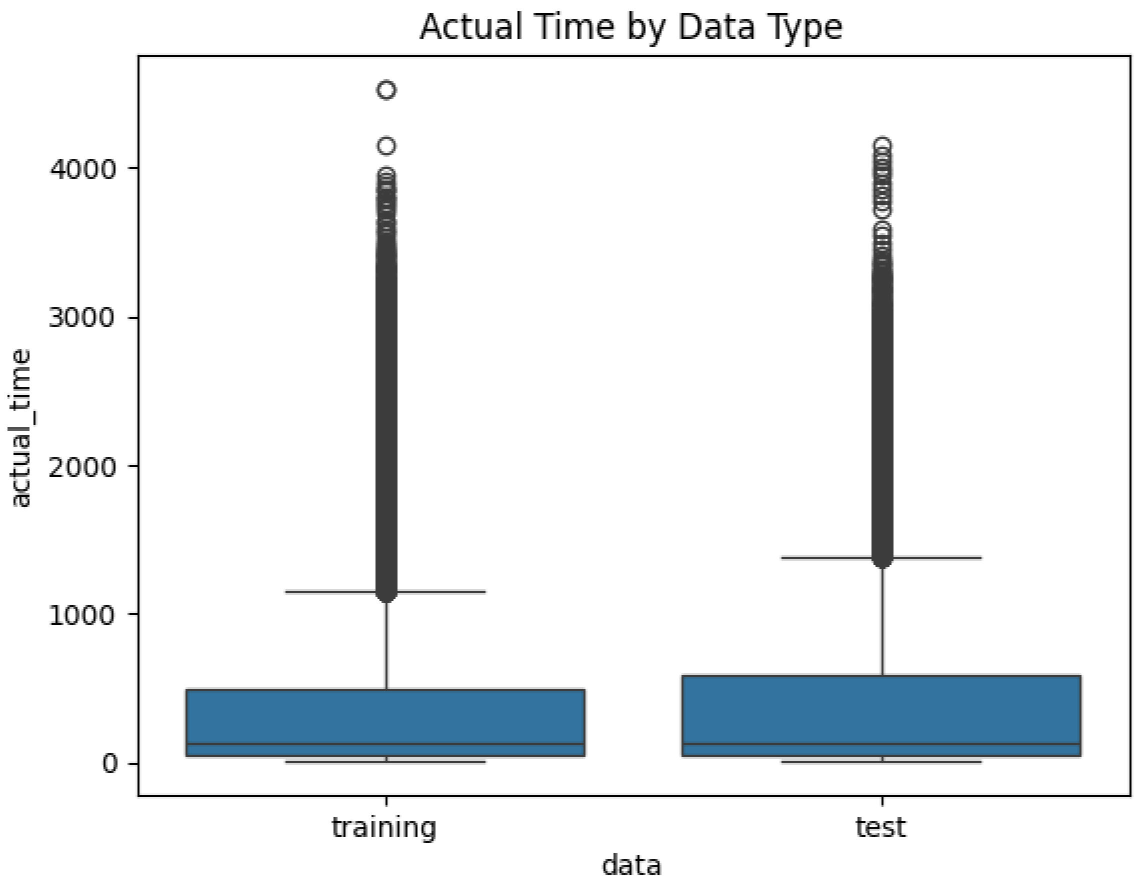
In [50]:
```
# Density Plot
sns.kdeplot(data['actual_time'])
plt.title('Density Plot of Actual Time')
plt.show()
```



In [51]:
```
# Boxplot of actual distance by route type
sns.boxplot(x='route_type', y='actual_distance_to_destination', data=data)
plt.title('Actual Distance by Route Type')
plt.show()
```



In [52]:
```
# Boxplot of actual time by data type
sns.boxplot(x='data', y='actual_time', data=data)
plt.title('Actual Time by Data Type')
plt.show()
```



In [53]:
```
#sns.countplot(data['Source_City'])
data['Source_City'].value_counts()
```

Out[53]:

	count
Source_City	
Gurgaon	18933
Bangalore	8403
Bhiwandi	7279
Pune	3412
Bengaluru	3229
...	...
Sumerpur	1
Kothanalloor	1
Hoshangabad	1
Kanhangad	1
Berhampur	1

1227 rows × 1 columns

dtype: int64

In [54]:

```
data['Dest_City'].value_counts()
```

Out[54]:

Dest_City	count
Gurgaon	12411
Bangalore	8461
Hyderabad	4644
Bhiwandi	4383
Delhi	4366
...	...
Perundurai	1
Khatauli	1
Dhuri	1
Kumta	1
Sidhmukh	1

1223 rows × 1 columns

dtype: int64

In [55]:
```
#One hot enconding for categorical variables.
data_ohe = pd.get_dummies(data, columns=['data','route_type'])
data_ohe
```

Out[55]:

	trip_creation_time	route_schedule_uuid	trip_uuid	source_center	
0	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	trip-153741093647649320	IND388121AAA	Ar
1	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	trip-153741093647649320	IND388121AAA	Ar
2	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	trip-153741093647649320	IND388121AAA	Ar
3	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	trip-153741093647649320	IND388121AAA	Ar
4	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	trip-153741093647649320	IND388121AAA	Ar
...	...	...	...	...	
114705	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	trip-153802021359954030	IND516115AAA	Ra
114706	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	trip-153802021359954030	IND516115AAA	Ra
114707	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	trip-153802021359954030	IND516115AAA	Ra
114708	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	trip-153802021359954030	IND516115AAA	Ra
114709	2018-09-27 03:50:13.599818	thanos::sroute:cce26bb2-2365-4c9c-88f4-74322d1...	trip-153802021359954030	IND516115AAA	Ra

114710 rows × 41 columns

In [56]:

```
#Do hypothesis testing/ visual analysis between actual_time aggregated value and OS
#H0 - actual time and OSRM time are not significantly different. H1 - actual time a
# Perform a KS test since the distribution is not normal

from scipy.stats import kstest

t_ks, p_value = kstest(aggregated_trip_data['actual_time'], aggregated_trip_data['o
t_ks, p_value
```

```
# Set significance level
alpha = 0.05

print("p-value is ",p_value)

if p_value < alpha:
    print("Reject Null Hypothesis.")
else:
    print("Fail to reject Null Hypothesis.")
```

```
p-value is  9.077739347778679e-277
Reject Null Hypothesis.
```

In [57]:
```
#Do hypothesis testing/ visual analysis between actual_time aggregated value and se
#H0 - actual time and segment actual time are not significantly different. H1 - act
# Perform a KS test since the distribution is not normal

from scipy.stats import kstest

t_ks, p_value = kstest(aggregated_trip_data['actual_time'], aggregated_trip_data['s
t_ks, p_value

# Set significance level
alpha = 0.05

print("p-value is ",p_value)

if p_value < alpha:
    print("Reject Null Hypothesis.")
else:
    print("Fail to reject Null Hypothesis.")
```

```
p-value is  3.546151336439e-311
Reject Null Hypothesis.
```

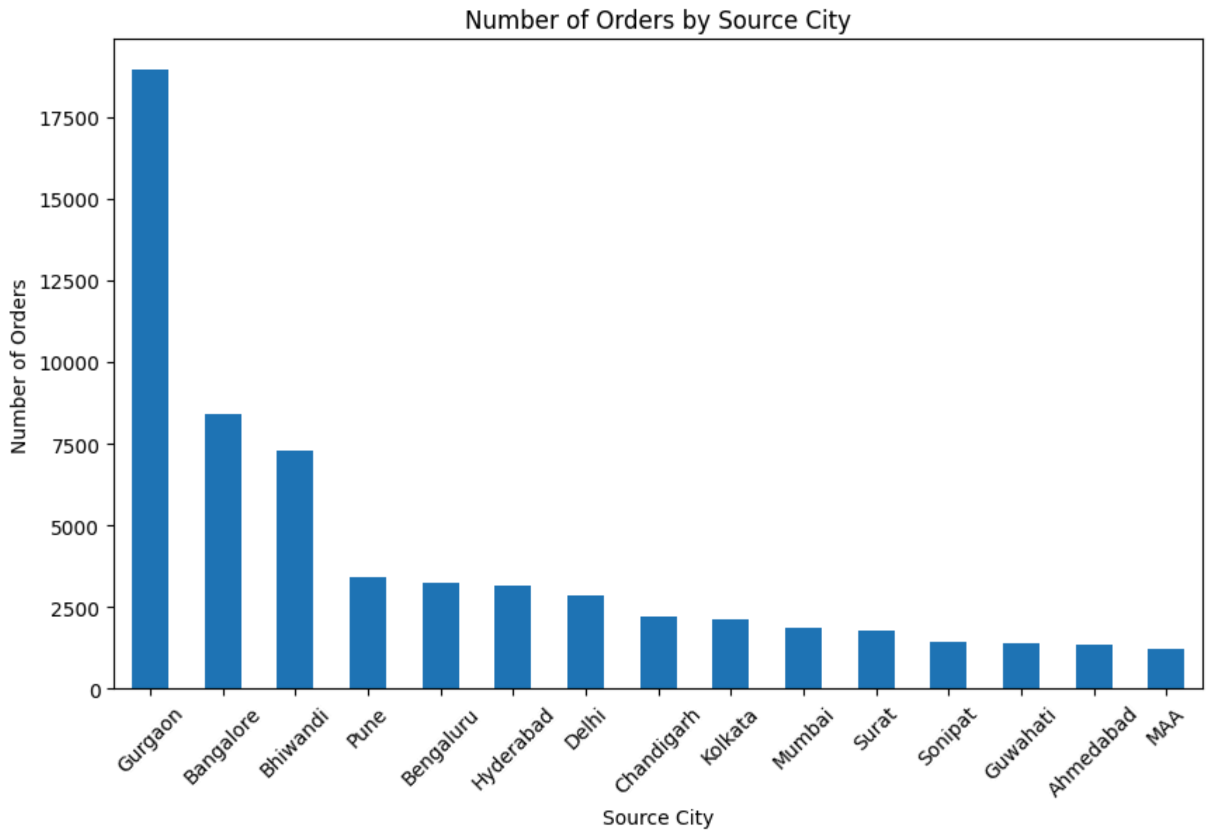
In [58]:
```
# Do hyponthesis testing/ visual analysis between osrm distance aggregated value an
#H0 - osrm distance and segment osrm distance are not significantly different. H1 -
# Perform a KS test since the distribution is not normal

from scipy.stats import kstest

t_ks, p_value = kstest(aggregated_trip_data['osrm_distance'], aggregated_trip_data[
t_ks, p_value

# Set significance level
alpha = 0.05

print("p-value is ",p_value)

if p_value < alpha:
    print("Reject Null Hypothesis.")
else:
    print("Fail to reject Null Hypothesis.")
```

```
p-value is  0.0
Reject Null Hypothesis.
```

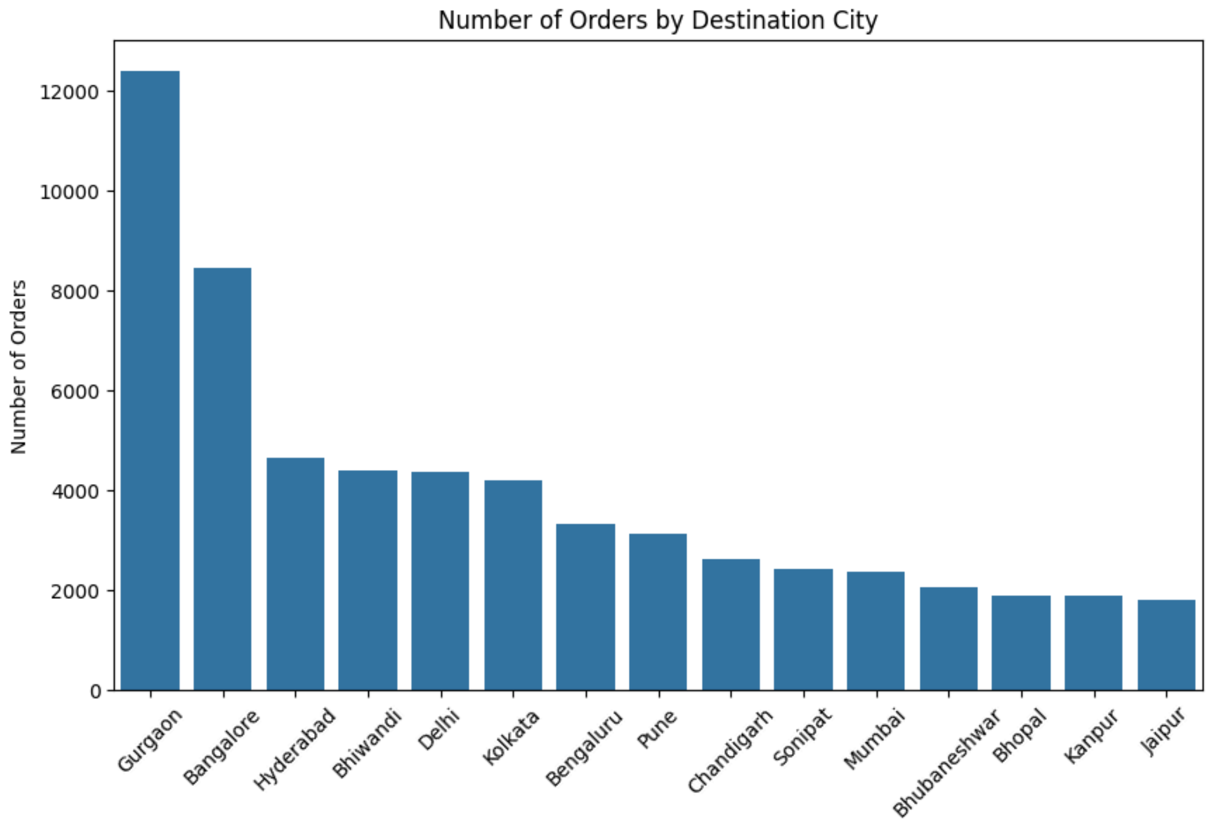
In [59]:
```
#Do hypothesis testing/ visual analysis between osrm time aggregated value and segm
#H0 - osrm time and segment osrm time are not significantly different. H1 - osrm ti
# Perform a KS test since the distribution is not normal

from scipy.stats import kstest

t_ks, p_value = kstest(aggregated_trip_data['osrm_time'], aggregated_trip_data['seg
t_ks, p_value

# Set significance level
alpha = 0.05

print("p-value is ",p_value)

if p_value < alpha:
    print("Reject Null Hypothesis.")
else:
    print("Fail to reject Null Hypothesis.")
```

```
p-value is  1.091363497855281e-299
Reject Null Hypothesis.
```

In [60]:
```
# From which city is the most number of orders created by ?
# Group by 'Source City' and count the number of orders
city_counts = data.groupby('Source_City').size()

# Sort the cities by the number of orders in descending order
city_counts = city_counts.sort_values(ascending=False).head(15) # limiting the numb

# Plot the bar chart
city_counts.plot(kind='bar', figsize=(10, 6))
plt.title('Number of Orders by Source City')
plt.xlabel('Source City')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
plt.show()
```



In [61]:
```
# Where is most of the orders getting delivered to ?
# Group by 'Source City' and count the number of orders
city_counts = data.groupby('Dest_City').size()

# Sort the cities by the number of orders in descending order
city_counts = city_counts.sort_values(ascending=False).head(15) # Limiting the numb

# Plot the bar chart#
city_counts.plot(kind='bar', figsize=(10, 6))
sns.barplot(x=city_counts.index, y=city_counts.values)
plt.title('Number of Orders by Destination City')
plt.xlabel('Destination City')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
plt.show()
```



In [62]:
```
data['destination_name'].value_counts().sort_values(ascending=False)
```

Out[62]:

destination_name	count
Gurgaon_Bilaspur_HB (Haryana)	12269
Bangalore_Nelmngla_H (Karnataka)	8399
Bhiwandi_Mankoli_HB (Maharashtra)	4327
Hyderabad_Shamshbd_H (Telangana)	4109
Kolkata_Dankuni_HB (West Bengal)	3925
...	...
Kumta_Central_DPP_2 (Karnataka)	1
Baghpat_Barout_D (Uttar Pradesh)	1
Shirur_Central_DPP_3 (Maharashtra)	1
Salem_Kadtmpty_D (Tamil Nadu)	1
Vijayawada (Andhra Pradesh)	1

1440 rows × 1 columns

dtype: int64

In [63]:

```
data['source_name'].value_counts().sort_values(ascending=False)
```

Out[63]:

source_name	count
Gurgaon_Bilaspur_HB (Haryana)	18715
Bangalore_Nelmngla_H (Karnataka)	8292
Bhiwandi_Mankoli_HB (Maharashtra)	7279
Pune_Tathawde_H (Maharashtra)	3249
Hyderabad_Shamshbd_H (Telangana)	2616
...	...
Kanhangad_Arangadi_D (Kerala)	1
Mahasamund_RajpurRD_D (Chhattisgarh)	1
Berhampur_Khajuria_I (Orissa)	1
Badkulla_Central_DPP_1 (West Bengal)	1
Bhubaneswar_Patia (Orissa)	1

1475 rows × 1 columns

dtype: int64

In [64]:

```
#understand which state has max orders generated
data['Source_State'].value_counts().sort_values(ascending=False)
```

Out[64]:

Source_State	count
Haryana	21964
Maharashtra	16953
Karnataka	15899
Uttar Pradesh	5894
Tamil Nadu	5890
Gujarat	5499
Telangana	5047
West Bengal	4499
Andhra Pradesh	4379
Rajasthan	4287
Punjab	3644
Delhi	3553
Bihar	3262
Madhya Pradesh	3182
Assam	2273
Jharkhand	1953
Kerala	1879
Orissa	1695
Uttarakhand	921
Himachal Pradesh	480
Goa	413
Chandigarh	371
Chhattisgarh	201
Arunachal Pradesh	180
Jammu & Kashmir	177
Meghalaya	78
Pondicherry	45
Nagaland	33
Dadra and Nagar Haveli	28

	count
Source_State	
Mizoram	26
Tripura	5

dtype: int64

In [65]:
```
aggregated_trip_data
```

Out[65]:

	trip_uuid	source_center	destination_center	segment_actual_time	segm
0	trip-153671042288605164	IND572101AAA	IND561203AAB	141.0	
1	trip-153671046011330457	IND400072AAB	IND401104AAA	59.0	
2	trip-153671052974046625	IND583101AAA	IND583201AAA	340.0	
3	trip-153671055416136166	IND600116AAB	IND600056AAA	60.0	
4	trip-153671066201138152	IND600044AAD	IND600048AAA	24.0	
...	...	...	...	...	
11741	trip-153861095625827784	IND160002AAC	IND140603AAA	82.0	
11742	trip-153861104386292051	IND121004AAB	IND121004AAA	21.0	
11743	trip-153861106442901555	IND209304AAA	IND208006AAA	281.0	
11744	trip-153861115439069069	IND627005AAA	IND628801AAA	258.0	
11745	trip-153861118270144424	IND583201AAA	IND583119AAA	274.0	

11746 rows × 10 columns

In [66]:
```
#Find the busiest corridor

# Group by 'corridor' and count the number of trips
corridor_counts = data.groupby('segment_key').size() #corridor combination of sourc

# Find the busiest corridor
busiest_corridor = corridor_counts.idxmax()
```

```
busiest_corridor_count = corridor_counts.max()

print("Busiest Corridor:", busiest_corridor)
print("Number of Trips:", busiest_corridor_count)
```

```
Busiest Corridor: trip-153755502932196495 - IND160002AAC - IND562132AAA
Number of Trips: 81
```

In [67]:
```
#What is the avg distance and time for the busiest corridor
# Filter data for the busiest corridor
busiest_corridor_data = data[data['segment_key'] == busiest_corridor]

# Calculate average distance and time
avg_distance = busiest_corridor_data['actual_distance_to_destination'].mean()
avg_time = busiest_corridor_data['actual_time'].mean()

print("Average Distance:", avg_distance)
print("Average Time:", avg_time)
```

```
Average Distance: 1050.7516678097484
Average Time: 1682.7283950617284
```

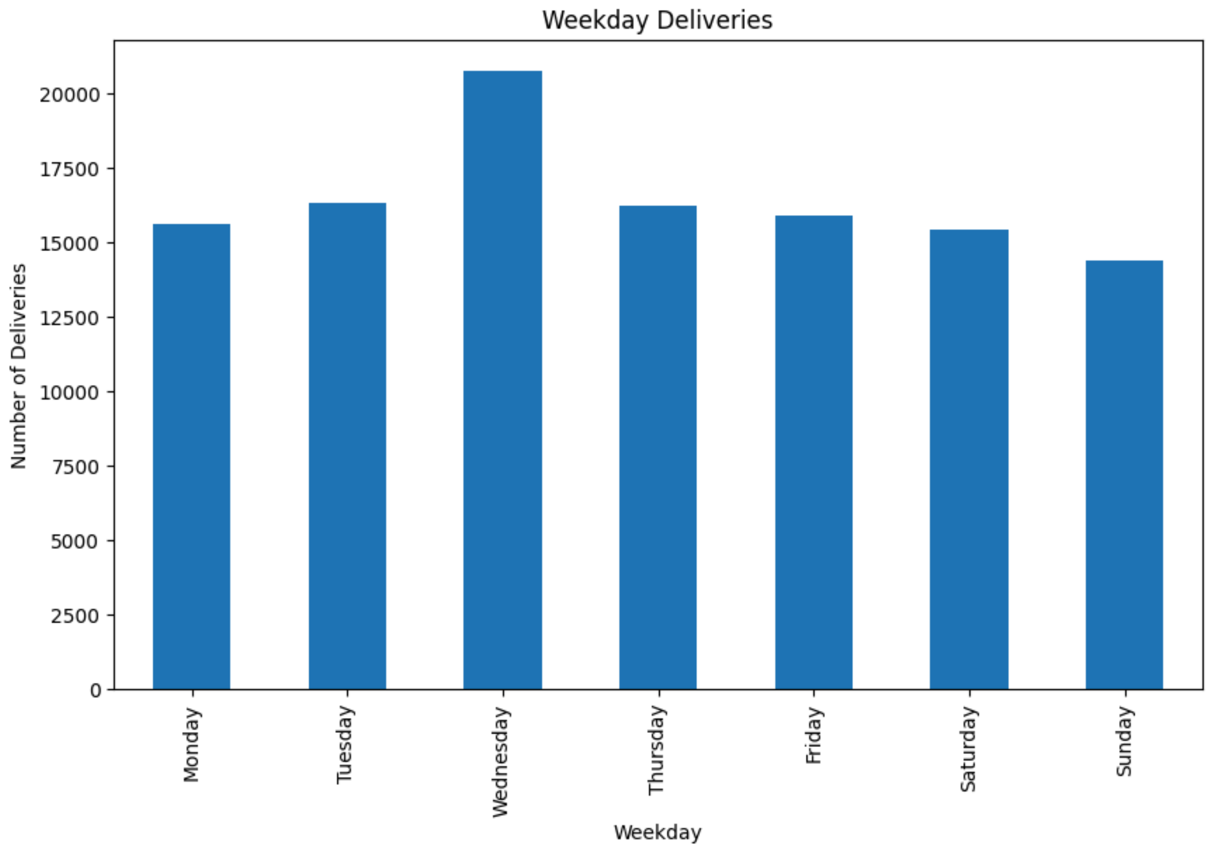
In [67]:
```
```

In [68]:
```
data.info()
```

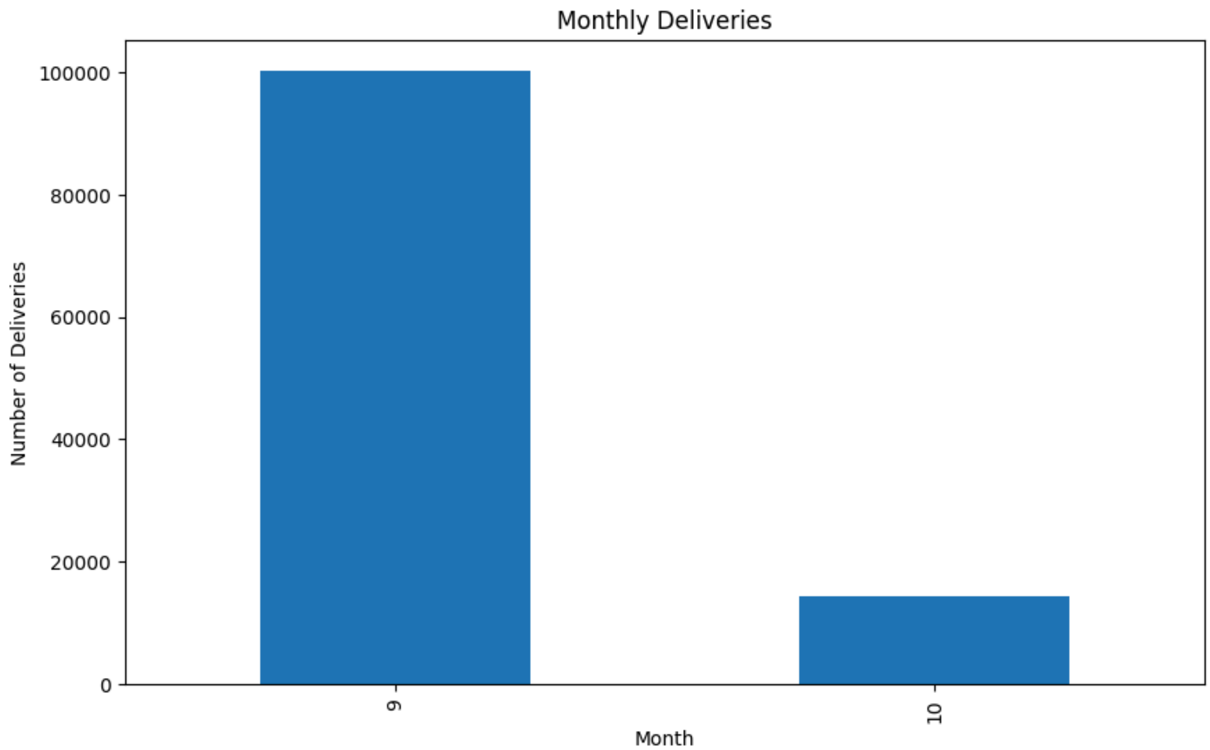
```
<class 'pandas.core.frame.DataFrame'>
Index: 114710 entries, 0 to 114709
Data columns (total 39 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            114710 non-null  object
 1   trip_creation_time              114710 non-null  datetime64[ns]
 2   route_schedule_uuid             114710 non-null  object
 3   route_type                      114710 non-null  object
 4   trip_uuid                       114710 non-null  object
 5   source_center                   114710 non-null  object
 6   source_name                     114710 non-null  object
 7   destination_center              114710 non-null  object
 8   destination_name                114710 non-null  object
 9   od_start_time                   114710 non-null  datetime64[ns]
 10  od_end_time                     114710 non-null  datetime64[ns]
 11  start_scan_to_end_scan          114710 non-null  float64
 12  is_cutoff                       114710 non-null  object
 13  cutoff_factor                   114710 non-null  float64
 14  cutoff_timestamp                114710 non-null  object
 15  actual_distance_to_destination  114710 non-null  float64
 16  actual_time                     114710 non-null  float64
 17  osrm_time                       114710 non-null  float64
 18  osrm_distance                   114710 non-null  float64
 19  factor                          114710 non-null  float64
 20  segment_actual_time             114710 non-null  float64
 21  segment_osrm_time               114710 non-null  float64
 22  segment_osrm_distance           114710 non-null  float64
 23  segment_factor                  114710 non-null  float64
 24  segment_key                     114710 non-null  object
 25  Source_City                     114710 non-null  object
 26  Source_Place                    114710 non-null  object
 27  Source_Code                     114710 non-null  object
 28  Source_State                    114710 non-null  object
 29  Dest_City                       114710 non-null  object
 30  Dest_Place                      114710 non-null  object
 31  Dest_Code                       114710 non-null  object
 32  Dest_State                      114710 non-null  object
 33  trip_creation_month             114710 non-null  int32
 34  trip_creation_year              114710 non-null  int32
 35  trip_creation_day               114710 non-null  int32
 36  trip_creation_hour              114710 non-null  int32
 37  trip_creation_minute            114710 non-null  int32
 38  od_time_diff_hour               114710 non-null  timedelta64[ns]
dtypes: datetime64[ns](3), float64(11), int32(5), object(19), timedelta64[ns](1)
memory usage: 32.8+ MB
```

In [69]:
```
# Are there differences in delivery performance on different weekdays?
# Group by day of the week and count the number of deliveries
weekday_deliveries = data.groupby(data['trip_creation_time'].dt.dayofweek)['trip_uu
weekday_labels = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday

# Plot the weekday deliveries
weekday_deliveries.plot(kind='bar', figsize=(10, 6))
plt.title('Weekday Deliveries')
plt.xlabel('Weekday')
```

```
plt.ylabel('Number of Deliveries')
plt.xticks(range(7), weekday_labels)
plt.show()
```

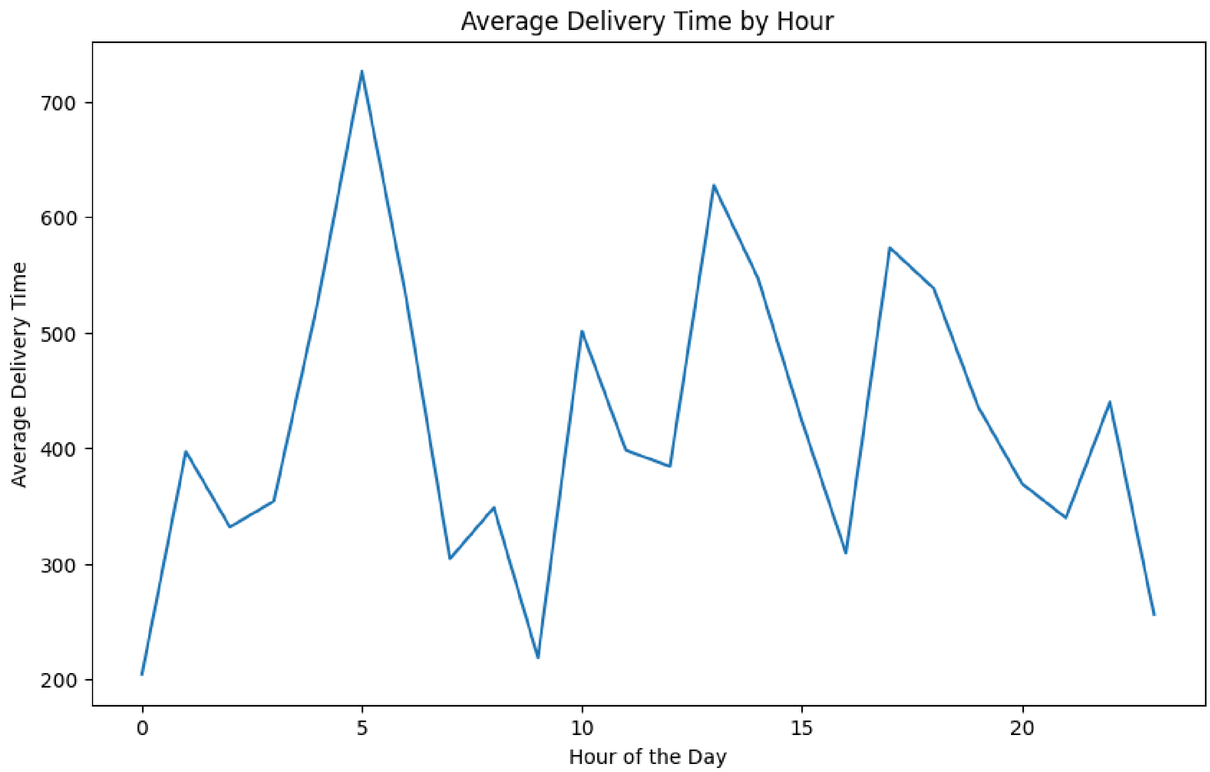


In [70]:
```
#what is the variation of deliveries across months
monthly_deliveries = data.groupby('trip_creation_month')['trip_uuid'].count()

# Plot the monthly deliveries
monthly_deliveries.plot(kind='bar', figsize=(10, 6))
plt.title('Monthly Deliveries')
plt.xlabel('Month')
plt.ylabel('Number of Deliveries')
plt.show()
```



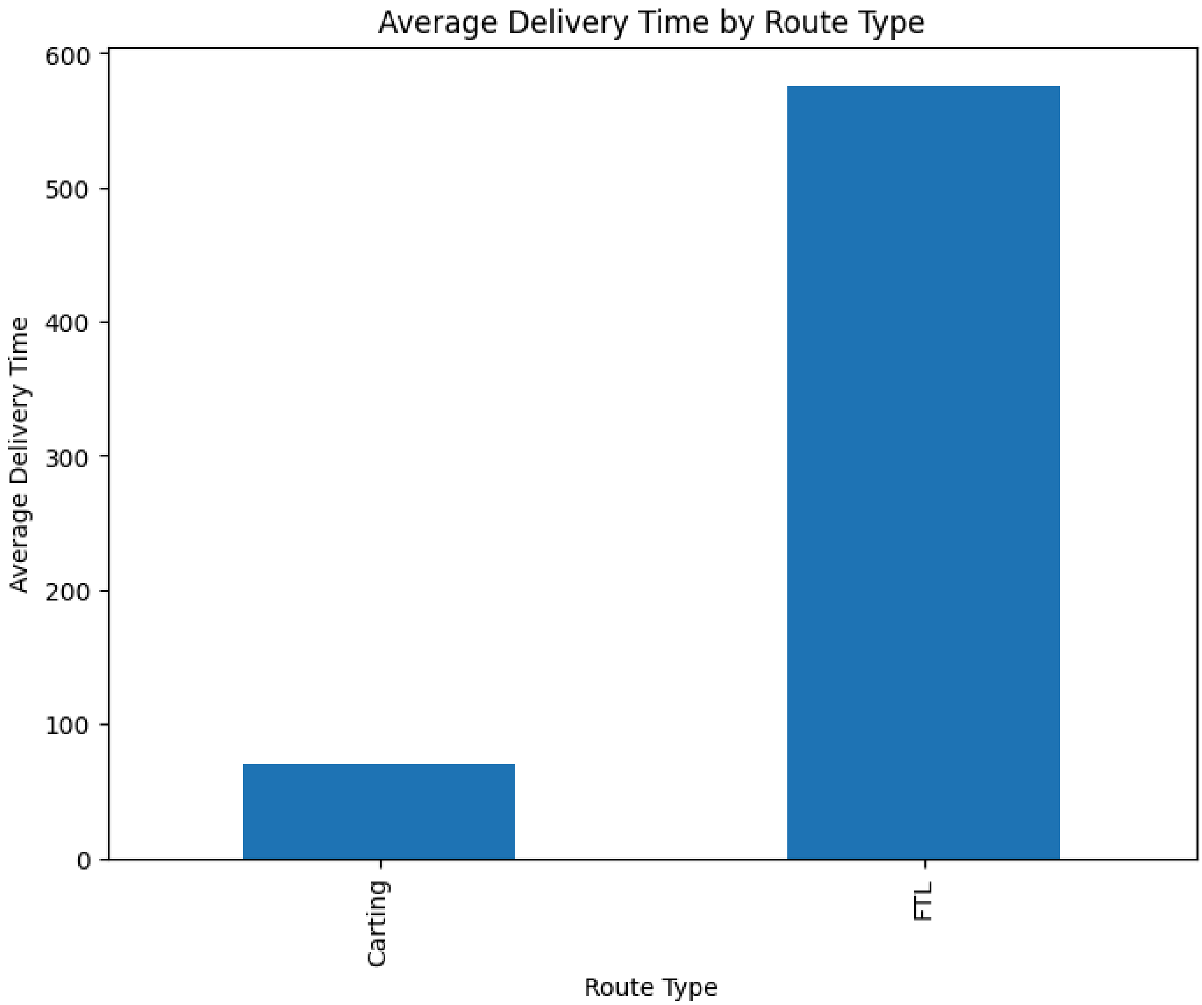
In [71]:

```
#  How do delivery times vary across different times of the day?


# Group by hour and calculate average delivery time
hourly_avg_delivery_time = data.groupby('trip_creation_hour')['actual_time'].mean()

# Plot the hourly average delivery time
hourly_avg_delivery_time.plot(kind='line', figsize=(10, 6))
plt.title('Average Delivery Time by Hour')
plt.xlabel('Hour of the Day')
plt.ylabel('Average Delivery Time')
plt.show()
```

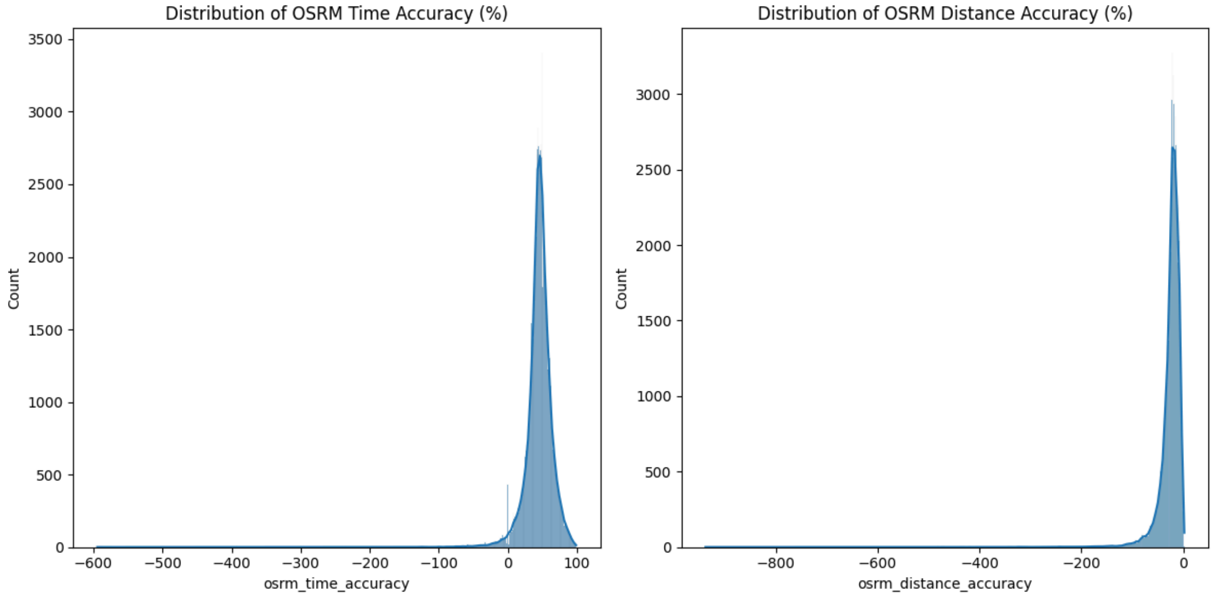


In [72]:

```
#How does the average delivery time vary by route type (FTL, Carting)?
# Group by route type and calculate average delivery time
avg_delivery_time_by_route_type = data.groupby('route_type')['actual_time'].mean()

# Plot the average delivery time by route type
avg_delivery_time_by_route_type.plot(kind='bar', figsize=(8, 6))
plt.title('Average Delivery Time by Route Type')
plt.xlabel('Route Type')
plt.ylabel('Average Delivery Time')
plt.show()
```



In [73]:
```
# How accurate are the OSRM time and distance estimates compared to the actual valu
# Calculate the percentage difference between actual and OSRM time and distance
data['osrm_time_accuracy'] = ((data['actual_time'] - data['osrm_time']) / data['act
data['osrm_distance_accuracy'] = ((data['actual_distance_to_destination'] - data['o

# Visualize the distribution of accuracy
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.histplot(data['osrm_time_accuracy'], kde=True)
plt.title('Distribution of OSRM Time Accuracy (%)')

plt.subplot(1, 2, 2)
sns.histplot(data['osrm_distance_accuracy'], kde=True)
plt.title('Distribution of OSRM Distance Accuracy (%)')

plt.tight_layout()
plt.show()

# Calculate summary statistics
print("Mean OSRM Time Accuracy:", data['osrm_time_accuracy'].mean())
print("Mean OSRM Distance Accuracy:", data['osrm_distance_accuracy'].mean())
```




```
Mean OSRM Time Accuracy: 45.182117028868454
Mean OSRM Distance Accuracy: -25.03150717069437
```

In [73]:

In [74]:
```
#data[data['source_center'] =='IND160002AAC' & data['destination_center'] == 'IND56
```

In [76]:
```
data[data['destination_center'] == 'IND562132AAA']
```

Out[76]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uui
470	training	2018-09-24 17:40:43.210450	thanos::sroute:366da0f3-1979-4793-973f-27da635...	FTL	trip 1537810843210180
471	training	2018-09-24 17:40:43.210450	thanos::sroute:366da0f3-1979-4793-973f-27da635...	FTL	trip 1537810843210180
472	training	2018-09-24 17:40:43.210450	thanos::sroute:366da0f3-1979-4793-973f-27da635...	FTL	trip 1537810843210180
473	training	2018-09-24 17:40:43.210450	thanos::sroute:366da0f3-1979-4793-973f-27da635...	FTL	trip 1537810843210180
474	training	2018-09-24 17:40:43.210450	thanos::sroute:366da0f3-1979-4793-973f-27da635...	FTL	trip 1537810843210180
...	...	...	...	...	
114563	test	2018-09-29 19:13:45.341446	thanos::sroute:eb0c8030-4969-4bc1-83ff-8e9e25d...	FTL	trip 1538248425341188
114675	test	2018-10-02 01:05:55.850345	thanos::sroute:500aa87c-3d54-4159-a296-0b93c15...	Carting	trip 1538442355850096
114676	test	2018-10-02 01:05:55.850345	thanos::sroute:500aa87c-3d54-4159-a296-0b93c15...	Carting	trip 1538442355850096
114677	test	2018-10-02 01:05:55.850345	thanos::sroute:500aa87c-3d54-4159-a296-0b93c15...	Carting	trip 1538442355850096
114687	training	2018-09-22 06:09:34.298350	thanos::sroute:369397e5-7b19-49be-aeed-abcc29b...	Carting	trip 1537596574298095

8399 rows × 41 columns