Importing the necessary libraries

In [848…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from scipy.stats import ttest_ind
from scipy.stats import levene
from scipy.stats import shapiro
```

Downloading the file

In [849…
```
!gdown --id 1ARd4VJqTul74XQmDLerT-rKOpwISnPcY
```

```
/usr/local/lib/python3.10/dist-packages/gdown/__main__.py:132: FutureWarning: Option `--id` was deprecated in version
4.3.1 and will be removed in 5.0. You don't need to pass it anymore to use a file ID.
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1ARd4VJqTul74XQmDLerT-rKOpwISnPcY
To: /content/delhivery_data.csv
100% 55.6M/55.6M [00:00<00:00, 186MB/s]
```

Reading the file

In [850…
```
df = pd.read_csv("delhivery_data.csv")
df.head()
```

Out[850]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_cente
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAE
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAE
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAE
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAE
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAE

5 rows × 24 columns

Problem Statement:

Delhivery, aiming to solidify its position as a leader in the logistics industry, needs to effectively leverage its vast amount of operational data to optimize its services and drive business growth. The challenge is to transform raw data from engineering pipelines into actionable insights that can improve operational efficiency, enhance decision-making, and support the development of accurate forecasting models.

- Specifically, the company needs to:

- Develop a robust data processing framework to clean, sanitize, and structure the raw data from various sources.

- Extract and engineer meaningful features from the processed data to support advanced analytics and machine learning models.
- Conduct in-depth analysis to identify patterns, trends, and anomalies in logistics operations, with a focus on route optimization and delivery efficiency.
- Perform hypothesis testing to validate assumptions about operational performance and identify areas for improvement.

- Create a foundation for predictive modeling that will enable more accurate forecasting of delivery times, resource needs, and potential operational challenges.
- Generate actionable recommendations based on the analysis to enhance overall operational excellence, improve customer satisfaction, and maintain Delhivery's competitive edge in the logistics market.

Understanding the columns

1. data - tells whether the data is testing or training data
2. trip_creation_time – Timestamp of trip creation
3. route_schedule_uuid – Unique ID for a particular route schedule
4. route_type – Transportation type a. FTL – Full Truck Load: FTL shipments get to the destination sooner, as the truck is making no other pickups or drop-offs along the way b. Carting: Handling system consisting of small vehicles (carts)
5. trip_uuid - Unique ID given to a particular trip (A trip may include different source and destination centers)
6. source_center - Source ID of trip origin
7. source_name - Source Name of trip origin
8. destination_cente – Destination ID
9. destination_name – Destination Name
10. od_start_time – Trip start time
11. od_end_time – Trip end time
12. start_scan_to_end_scan – Time taken to deliver from source to destination
13. is_cutoff – Unknown field
14. cutoff_factor – Unknown field
15. cutoff_timestamp – Unknown field
16. actual_distance_to_destination – Distance in kms between source and destination warehouse
17. actual_time – Actual time taken to complete the delivery (Cumulative)
18. osrm_time – An open-source routing engine time calculator which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) and gives the time (Cumulative)
19. osrm_distance – An open-source routing engine which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) (Cumulative)
20. factor – Unknown field
21. segment_actual_time – This is a segment time. Time taken by the subset of the package delivery

22. segment_osrm_time – This is the OSRM segment time. Time taken by the subset of the package delivery
23. segment_osrm_distance – This is the OSRM distance. Distance covered by subset of the package delivery
24. segment_factor – Unknown field

In [851…
```
df.shape
```
Out[851]:
```
(144867, 24)
```

In [852…
```
df.ndim
```
Out[852]:
```
2
```

Removing the unknown columns

In [853…
```
unknown_cols = df[['is_cutoff','cutoff_factor','cutoff_timestamp','factor','segment_factor']]
```

In [854…
```
df.drop(columns = unknown_cols,inplace = True)
```

In [855…
```
df.columns
```
Out[855]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance'],
      dtype='object')
```

In [856…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 19 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  actual_distance_to_destination  144867 non-null  float64
 13  actual_time                     144867 non-null  float64
 14  osrm_time                       144867 non-null  float64
 15  osrm_distance                   144867 non-null  float64
 16  segment_actual_time             144867 non-null  float64
 17  segment_osrm_time               144867 non-null  float64
 18  segment_osrm_distance           144867 non-null  float64
dtypes: float64(8), object(11)
memory usage: 21.0+ MB
```

Number of unique elements in each column

In [857…

```
for i in df.columns:
  print(i,"-->",df[i].nunique())
```

```
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
actual_distance_to_destination --> 144515
actual_time --> 3182
osrm_time --> 1531
osrm_distance --> 138046
segment_actual_time --> 747
segment_osrm_time --> 214
segment_osrm_distance --> 113799
```

Selecting the columns which have object data type

In [858…

```
object_columns = df.select_dtypes(include='object')
object_columns.columns
```

Out[858]:

```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time'],
      dtype='object')
```

Changing the data type of the following columns from object to datetime data type

- trip_creation_time
- od_start_time
- od_end_time

In [859…
```
obj_to_dt = df[['trip_creation_time','od_start_time','od_end_time']]
for i in obj_to_dt:
  df[i] = pd.to_datetime(df[i])
```

Changing the data type of the following columns from object to category data type

- data
- route type

In [860…
```
obj_to_category = df[['data','route_type']]
for i in obj_to_category:
  df[i] = (df[i].astype('category'))
```

Selecting the columns which have float64 data type

In [861…
```
float64_columns = df.select_dtypes(include='float64')
float64_columns.columns
```

Out[861]:
```
Index(['start_scan_to_end_scan', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance'],
      dtype='object')
```

Changing the float64 ---> float32 to reduce the overall file size

In [862…
```
for i in float64_columns:
  df[i] = df[i].astype('float32')
```

In [863…
```
df.dtypes
```

Out[863]:

	0
data	category
trip_creation_time	datetime64[ns]
route_schedule_uuid	object
route_type	category
trip_uuid	object
source_center	object
source_name	object
destination_center	object
destination_name	object
od_start_time	datetime64[ns]
od_end_time	datetime64[ns]
start_scan_to_end_scan	float32
actual_distance_to_destination	float32
actual_time	float32
osrm_time	float32
osrm_distance	float32
segment_actual_time	float32
segment_osrm_time	float32
segment_osrm_distance	float32

dtype: object

In [864…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 19 columns):
 #   Column                          Non-Null Count   Dtype         
---  ------                          --------------   -----         
 0   data                            144867 non-null  category      
 1   trip_creation_time              144867 non-null  datetime64[ns]
 2   route_schedule_uuid             144867 non-null  object        
 3   route_type                      144867 non-null  category      
 4   trip_uuid                       144867 non-null  object        
 5   source_center                   144867 non-null  object        
 6   source_name                     144574 non-null  object        
 7   destination_center              144867 non-null  object        
 8   destination_name                144606 non-null  object        
 9   od_start_time                   144867 non-null  datetime64[ns]
 10  od_end_time                     144867 non-null  datetime64[ns]
 11  start_scan_to_end_scan          144867 non-null  float32       
 12  actual_distance_to_destination  144867 non-null  float32       
 13  actual_time                     144867 non-null  float32       
 14  osrm_time                       144867 non-null  float32       
 15  osrm_distance                   144867 non-null  float32       
 16  segment_actual_time             144867 non-null  float32       
 17  segment_osrm_time               144867 non-null  float32       
 18  segment_osrm_distance           144867 non-null  float32       
dtypes: category(2), datetime64[ns](3), float32(8), object(6)
memory usage: 14.6+ MB
```

- Observation: File size reduce from 21.0+ MB to 16.4+ MB Section

Checking for Missing Values

In [865…

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

Out[865]:

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

dtype: int64

- Observation : Only Source name and destination columns have missing values in them

In [866…

```
total_missing_df = df.isnull().sum().sort_values(ascending =False)
percent_missing_df = ((df.isnull().sum()/df.isna().count())*100).sort_values(ascending=False)
```

```
missing_data_df = pd.concat([total_missing_df, percent_missing_df], axis=1, keys=['Total', 'Percent'])
missing_data_df[missing_data_df['Percent']>0]
```

Out[866]:

	Total	Percent
source_name	293	0.202254
destination_name	261	0.180165

- Observation:
- In source name column we have ~0.2% of missing values and
- In destination name column we have around ~0.18% of missing values

Observing each trip_uuid to understand it's segments and clubbing them into single row

In [867…
```
df[df['trip_uuid']=='trip-153741093647649320']
```

Out[867]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_ce
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND38862(
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND38862(
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND38862(
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND38862(
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND38862(
5	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND38832(
6	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND38832(
7	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND38832(
8	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND38832(
9	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	IND38832(

- Observation:
- The delivery information for each package is distributed across multiple rows in the dataset. This structure indicates that each delivery trip is broken down into distinct segments or stages.

In [868…

```
df['segment_id'] = df['trip_uuid']+df['source_center']+df['destination_center']

df['segment_actual_time'] = df.groupby('segment_id')['segment_actual_time'].cumsum()

df['segment_osrm_time'] = df.groupby('segment_id')['segment_osrm_time'].cumsum()

df['segment_osrm_distance'] = df.groupby('segment_id')['segment_osrm_distance'].cumsum()
```

In [869…

```
df.columns
```

Out[869]:

```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'segment_id'],
      dtype='object')
```

In [870…

```
merge_segment = {'trip_creation_time':'first',
              'route_schedule_uuid':'first',
              'route_type':'first',
              'trip_uuid':'first',
              'source_center':'first',
              'source_name':'first',
              'destination_center':'last',
              'destination_name':'last',
              'od_start_time':'first',
              'od_end_time':'last',
              'start_scan_to_end_scan':'last',
              'actual_distance_to_destination':'last',
              'actual_time':'last',
              'osrm_time':'last',
              'osrm_distance':'last',
              'segment_actual_time':'last',
              'segment_osrm_time':'last',
              'segment_osrm_distance':'last'
}
```

In [871…

```
segment_df = df.sort_values(['segment_id','actual_time']).groupby('segment_id').agg(merge_segment).reset_index()
segment_df.head()
```

Out[871]:

	segment_id	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_cer
0	trip-153671041653548748IND209304AAAIND000000ACB	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip-153671041653548748	IND209304/
1	trip-153671041653548748IND462022AAAIND209304AAA	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip-153671041653548748	IND462022/
2	trip-153671042288605164IND561203AABIND562101AAA	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	trip-153671042288605164	IND561203/
3	trip-153671042288605164IND572101AAAIND561203AAB	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	trip-153671042288605164	IND572101/
4	trip-153671043369099517IND000000ACBIND160002AAC	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	trip-153671043369099517	IND000000/

In [872…

```
segment_df.shape
```

Out[872]:

```
(26368, 19)
```

In [873…

```
segment_df.loc[segment_df['trip_uuid']=='trip-153741093647649320']
```

Out[873]:

	segment_id	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source
10374	trip-153741093647649320IND388121AAAIND388620AAB	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
10375	trip-153741093647649320IND388620AABIND388320AAA	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388

Observation:

- After creating the segment id and merge_segments with appropriate aggregate values ,here we can observe that

- Each trip_uuid has multiple segment_id's within
- For example trip_uuid ---> 153741093647649320 has 2 segment_id's

1. Anand_VUNagar_DC (Gujarat) -----> Khambhat_MotvdDPP_D (Gujarat)
2. Khambhat_MotvdDPP_D (Gujarat) ----> Anand_Vaghasi_IP (Gujarat)

In [874…
```
segment_df['trip_uuid'].value_counts()
```

Out[874]:

trip_uuid	count
trip-153758895506669465	8
trip-153717306559016761	8
trip-153710494321650505	8
trip-153799754114520189	7
trip-153723070696812593	7
...	...
trip-153710150497989359	1
trip-153710140173080801	1
trip-153773116749798625	1
trip-153710128882927165	1
trip-153700061930228958	1

14817 rows × 1 columns

dtype: int64

In [875…
```
merge_trip = {'trip_creation_time':'first',
              'route_schedule_uuid':'first',
              'route_type':'first',
              'source_center':'first',
              'source_name':'first',
              'destination_center':'last',
```

```
                'destination_name':'last',
                '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'}
```

In [876…

```
trip_df = segment_df.sort_values(['trip_uuid','od_start_time']).groupby('trip_uuid').agg(merge_trip).reset_index()
trip_df.head()
```

Out[876]:

	trip_uuid	trip_creation_time	route_schedule_uuid	route_type	source_center	source_name	destination_center	
0	trip-153671041653548748	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	IND462022AAA	Bhopal_Trnsport_H (Madhya Pradesh)	IND000000ACB	
1	trip-153671042288605164	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	IND572101AAA	Tumkur_Veersagr_I (Karnataka)	IND562101AAA	
2	trip-153671043369099517	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND160002AAC	Ch
3	trip-153671046011330457	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	IND400072AAB	Mumbai Hub (Maharashtra)	IND401104AAA	
4	trip-153671052974046625	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	IND583101AAA	Bellary_Dc (Karnataka)	IND583101AAA	

In [877…

```
trip_df.shape
```

Out[877]:

```
(14817, 18)
```

In [878…

```
trip_df.loc[trip_df['trip_uuid']=='trip-153741093647649320']
```

Out[878]:

	trip_uuid	trip_creation_time	route_schedule_uuid	route_type	source_center	source_name	destination_center	de:
5919	trip-153741093647649320	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388320AAA	Ar

Observation:

- After creating the trip_df,here we can observe that
- Each trip_uuid has multiple segments and when we combine them we get a single row where in we will be able to find the totals of

1. start_scan_to_end_scan,
2. actual_distance_to_destination
3. actual_time
4. osrm_time
5. osrm_distance
6. segment_actual_time
7. segment_osrm_time
8. segment_osrm_distance

Identifying any missing values after merging the rows per trip_uuid

In [879…

```
trip_df.isna().sum()
```

Out[879]:

	0
trip_uuid	0
trip_creation_time	0
route_schedule_uuid	0
route_type	0
source_center	0
source_name	10
destination_center	0
destination_name	8
od_start_time	0
od_end_time	0
start_scan_to_end_scan	0
actual_distance_to_destination	0
actual_time	0
osrm_time	0
osrm_distance	0
segment_actual_time	0
segment_osrm_time	0
segment_osrm_distance	0

dtype: int64

Observation: Only very few nulls exists in source name(10) and destination name columns(8) respectively we can drop them

In [880…

```
trip_df.dropna(inplace=True)
trip_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 14800 entries, 0 to 14816
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   trip_uuid                       14800 non-null  object
 1   trip_creation_time              14800 non-null  datetime64[ns]
 2   route_schedule_uuid             14800 non-null  object
 3   route_type                      14800 non-null  category
 4   source_center                   14800 non-null  object
 5   source_name                     14800 non-null  object
 6   destination_center              14800 non-null  object
 7   destination_name                14800 non-null  object
 8   od_start_time                   14800 non-null  datetime64[ns]
 9   od_end_time                     14800 non-null  datetime64[ns]
 10  start_scan_to_end_scan          14800 non-null  float32
 11  actual_distance_to_destination  14800 non-null  float32
 12  actual_time                     14800 non-null  float32
 13  osrm_time                       14800 non-null  float32
 14  osrm_distance                   14800 non-null  float32
 15  segment_actual_time             14800 non-null  float32
 16  segment_osrm_time               14800 non-null  float32
 17  segment_osrm_distance           14800 non-null  float32
dtypes: category(1), datetime64[ns](3), float32(8), object(6)
memory usage: 1.6+ MB
```

In [881…
```
trip_df.shape
```

Out[881]:
```
(14800, 18)
```

In [882…
```
trip_df.isnull().sum()
```

Out[882]:

	0
trip_uuid	0
trip_creation_time	0
route_schedule_uuid	0
route_type	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
actual_distance_to_destination	0
actual_time	0
osrm_time	0
osrm_distance	0
segment_actual_time	0
segment_osrm_time	0
segment_osrm_distance	0

dtype: int64

Data after thorough cleaning

In [883…

```
trip_df.head()
```

Out[883]:

	trip_uuid	trip_creation_time	route_schedule_uuid	route_type	source_center	source_name	destination_center	
0	trip-153671041653548748	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	IND462022AAA	Bhopal_Trnsport_H (Madhya Pradesh)	IND000000ACB	
1	trip-153671042288605164	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	IND572101AAA	Tumkur_Veersagr_I (Karnataka)	IND562101AAA	
2	trip-153671043369099517	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND160002AAC	Chi
3	trip-153671046011330457	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	IND400072AAB	Mumbai Hub (Maharashtra)	IND401104AAA	
4	trip-153671052974046625	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	IND583101AAA	Bellary_Dc (Karnataka)	IND583101AAA	

Seperating state and city from destination name column

In [884…

```
trip_df['destination_name'].head()
```

Out[884]:

	destination_name
0	Gurgaon_Bilaspur_HB (Haryana)
1	Chikblapur_ShntiSgr_D (Karnataka)
2	Chandigarh_Mehmdpur_H (Punjab)
3	Mumbai_MiraRd_IP (Maharashtra)
4	Bellary_Dc (Karnataka)

dtype: object

In [885…

```
# separating destination state from destination name column
def sep_state(a):
```

```python
    return a.split('(')[1][:-1]
```

In [886…

```python
trip_df['destination_st'] = trip_df['destination_name'].apply(sep_state)
trip_df['destination_st'].head()
trip_df['destination_st'].value_counts().reset_index()
```

Out[886]:

	destination_st	count
0	Maharashtra	2591
1	Karnataka	2275
2	Haryana	1667
3	Tamil Nadu	1072
4	Telangana	838
5	Gujarat	746
6	Uttar Pradesh	732
7	West Bengal	708
8	Punjab	693
9	Delhi	675
10	Rajasthan	523
11	Andhra Pradesh	414
12	Bihar	363
13	Madhya Pradesh	337
14	Kerala	273
15	Assam	234
16	Jharkhand	168
17	Orissa	119
18	Uttarakhand	113
19	Goa	65
20	Chhattisgarh	43
21	Himachal Pradesh	40
22	Chandigarh	29
23	Arunachal Pradesh	23
24	Dadra and Nagar Haveli	17

	destination_st	count
25	Jammu & Kashmir	15
26	Pondicherry	10
27	Meghalaya	8
28	Mizoram	6
29	Nagaland	1
30	Tripura	1
31	Daman & Diu	1

In [887…

```
# separating destination city from destination column
def sep_city(a):
  return a.split('_')[0]
```

In [888…

```
trip_df['destination_city'] = trip_df['destination_name'].apply(sep_city)
trip_df['destination_city'].head()
```

Out[888]:

	destination_city
0	Gurgaon
1	Chikblapur
2	Chandigarh
3	Mumbai
4	Bellary

dtype: object

Seperating state and city from source name column

In [889…

```
trip_df['source_name'].head()
```

Out[889]:

	source_name
0	Bhopal_Trnsport_H (Madhya Pradesh)
1	Tumkur_Veersagr_I (Karnataka)
2	Bangalore_Nelmngla_H (Karnataka)
3	Mumbai Hub (Maharashtra)
4	Bellary_Dc (Karnataka)

dtype: object

In [890…

```
# separating source state from source name column
def sep_state(a):
  return a.split('(')[1][:-1]
```

In [891…

```
trip_df['source_st'] = trip_df['source_name'].apply(sep_state)
trip_df['source_st'].head()
```

Out[891]:

	source_st
0	Madhya Pradesh
1	Karnataka
2	Karnataka
3	Maharashtra
4	Karnataka

dtype: object

In [892…

```
# separating source city from source name column
def sep_city(a):
  return a.split('_')[0]
```

In [893…

```
trip_df['source_city'] = trip_df['source_name'].apply(sep_city)
trip_df['source_city'].head()
```

Out[893]:

	source_city
0	Bhopal
1	Tumkur
2	Bangalore
3	Mumbai Hub (Maharashtra)
4	Bellary

dtype: object

seperating datetime from trip creation column into trip_year,trip_month,trip_day,trip_hour,trip_time,trip_day_of_week

In [894…
```
trip_df['trip_creation_time'].head()
```

Out[894]:

	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

dtype: datetime64[ns]

In [895…
```
trip_df['trip_yr'] = trip_df['trip_creation_time'].dt.year
trip_df['trip_mon'] = trip_df['trip_creation_time'].dt.month
trip_df['trip_day'] = trip_df['trip_creation_time'].dt.day
trip_df['trip_hr'] = trip_df['trip_creation_time'].dt.hour
trip_df['trip_time'] = trip_df['trip_creation_time'].dt.day
trip_df['trip_dayofweek'] = trip_df['trip_creation_time'].dt.day_of_week
```

In [896…

```
trip_df[['trip_yr','trip_mon', 'trip_day','trip_hr','trip_time','trip_dayofweek']].sample(10)
```

Out[896]:

	trip_yr	trip_mon	trip_day	trip_hr	trip_time	trip_dayofweek
5331	2018	9	19	7	19	2
3532	2018	9	16	22	16	6
9584	2018	9	25	13	25	1
9695	2018	9	25	18	25	1
7082	2018	9	21	21	21	4
11354	2018	9	28	1	28	4
3018	2018	9	16	0	16	6
5094	2018	9	18	23	18	1
8453	2018	9	23	20	23	6
14175	2018	10	2	23	2	1

In [897…

```
trip_df['trip_time'] = trip_df['od_end_time']-trip_df['od_start_time']
trip_df['trip_time_mins'] = (trip_df['od_end_time'] - trip_df['od_start_time']).dt.total_seconds()/(60) # trip duratic
trip_df[['od_start_time','od_end_time','trip_time_mins','trip_time']].head()
```

Out[897]:

	od_start_time	od_end_time	trip_time_mins	trip_time
0	2018-09-12 00:00:16.535741	2018-09-13 13:40:23.123744	2260.109800	1 days 13:40:06.588003
1	2018-09-12 00:00:22.886430	2018-09-12 03:01:59.598855	181.611874	0 days 03:01:36.712425
2	2018-09-12 00:00:33.691250	2018-09-14 17:34:55.442454	3934.362520	2 days 17:34:21.751204
3	2018-09-12 00:01:00.113710	2018-09-12 01:41:29.809822	100.494935	0 days 01:40:29.696112
4	2018-09-12 00:02:09.740725	2018-09-12 12:00:30.683231	718.349042	0 days 11:58:20.942506

Listing out all the numerical columns into num_col

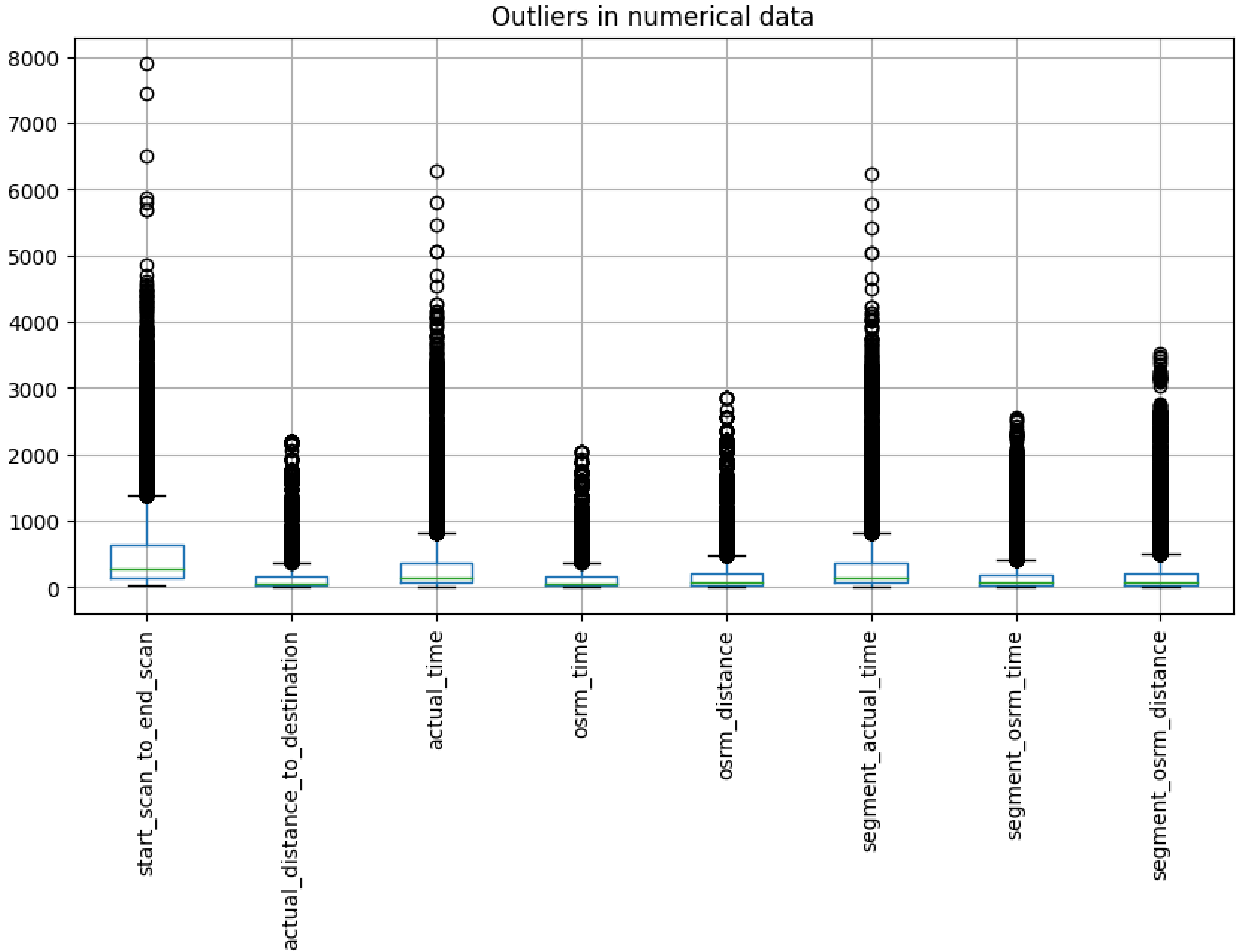
In [898…

```
num_col =['start_scan_to_end_scan', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance']
```

Univariate analysis using Boxplot of all the numerical data

In [899…

```
trip_df[num_col].boxplot(figsize=(10,5))
plt.xticks(rotation=90)
plt.title("Outliers in numerical data")
plt.show()
```



Outliers in numerical data
8000
7000
6000
5000
4000
3000
2000
1000
0
start_scan_to_end_scan
actual_distance_to_destination
actual_time
osrm_time
osrm_distance
segment_actual_time
segment_osrm_time
segment_osrm_distance

Identifying the data points at 25 th and 75 th percentile and compuitng Interquartile range

In [900…
```
q1 = trip_df[num_col].quantile(0.25)
q3 = trip_df[num_col].quantile(0.75)
IQR = q3-q1
IQR
```

Out[900]:

	0
start_scan_to_end_scan	489.000000
actual_distance_to_destination	141.919181
actual_time	303.000000
osrm_time	139.250000
osrm_distance	177.857743
segment_actual_time	301.000000
segment_osrm_time	155.000000
segment_osrm_distance	186.299987

dtype: float64

Handling the outliers

In [901…
```
trip_df = trip_df[~((trip_df[num_col]<q1-(1.5*IQR)) | (trip_df[num_col]>q3+(1.5*IQR))).any(axis=1)].reset_index()
```

Shape of the data after removing the outliers

In [902…
```
trip_df.shape
```

Out[902]:
```
(12744, 30)
```

Exploratory data analysis

Statistical Summary

In [903…
```
trip_df[num_col].describe()
```

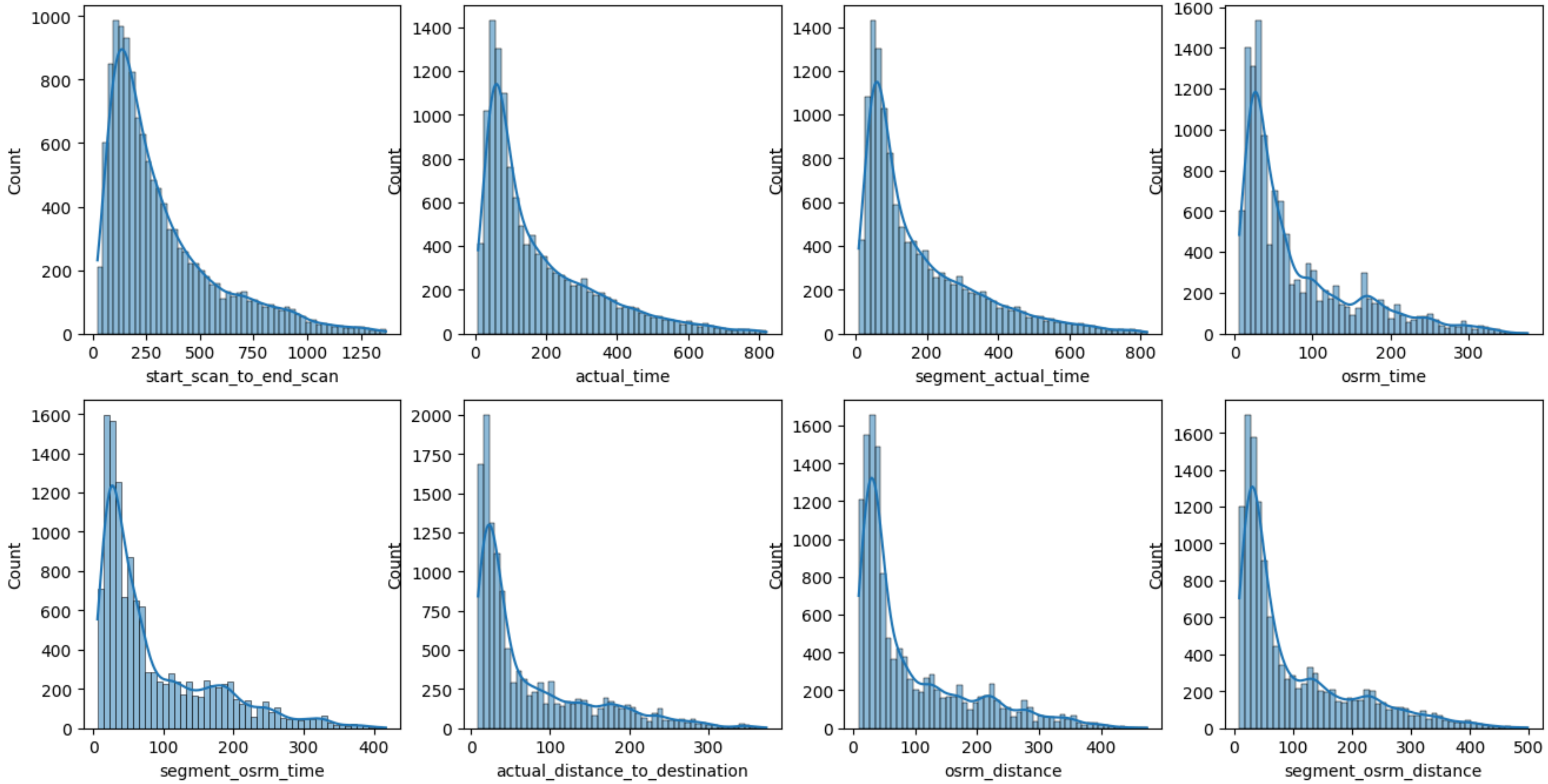
Out[903]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time	segment_o
count	12744.000000	12744.000000	12744.000000	12744.000000	12744.000000	12744.000000	1274
mean	322.166809	72.840584	178.612839	78.992546	92.396942	176.948685	8
std	257.550323	72.611122	159.187897	72.898178	90.244339	158.133560	8
min	23.000000	9.002461	9.000000	6.000000	9.072900	9.000000	
25%	136.000000	21.403780	61.000000	27.000000	28.357800	60.000000	2
50%	234.000000	38.657143	115.000000	50.000000	48.619051	114.000000	5
75%	427.000000	103.167301	254.000000	111.000000	132.065826	251.000000	12
max	1366.000000	373.441223	820.000000	376.000000	474.133698	818.000000	41

Distribution plots of continuous features

In [904…
```
fig,ax = plt.subplots(2,4,figsize=(16,8))
sns.histplot(data = trip_df['start_scan_to_end_scan'],kde = True,ax = ax[0,0])
sns.histplot(data = trip_df['actual_time'],kde = True,ax = ax[0,1])
sns.histplot(data = trip_df['segment_actual_time'],kde = True,ax = ax[0,2])
sns.histplot(data = trip_df['osrm_time'],kde =True,ax = ax[0,3])
sns.histplot(data = trip_df['segment_osrm_time'],kde = True,ax = ax[1,0])
sns.histplot(data = trip_df['actual_distance_to_destination'],kde = True,ax = ax[1,1])
sns.histplot(data = trip_df['osrm_distance'],kde = True,ax = ax[1,2])
sns.histplot(data=trip_df['segment_osrm_distance'], kde=True, ax=ax[1,3])
```

```
plt.show()
```



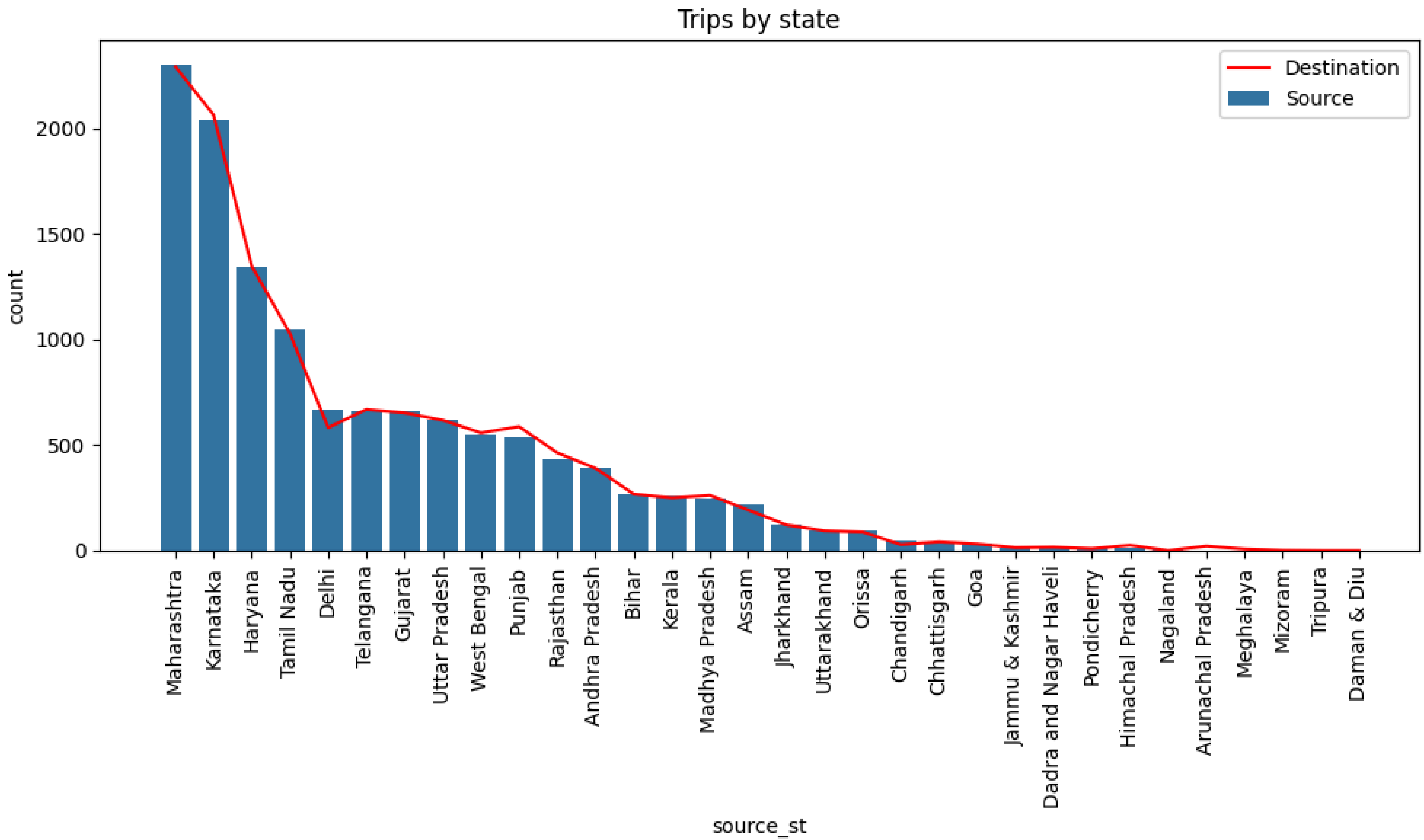
Observation:

- All distribution plots are right skewed.
- Indicates that a large number of trips have shorter durations, while fewer trips have longer durations.
- The segment_actual_time and segment_osrm_time follow a similar pattern, with the actual time generally being more spread out compared to the OSRM time.

- The actual distances (both actual_distance_to_destination and segment_actual_distance) have a wider distribution compared to the OSRM-predicted distances.

In [905…

```
source = trip_df['source_st'].value_counts().reset_index()
source
destination = trip_df['destination_st'].value_counts().reset_index()
destination
plt.figure(figsize=(10,6))
plt.xticks(rotation=90)
plt.title("Trips by state")
sns.barplot(data = source, x = 'source_st', y = 'count',label='Source')
sns.lineplot(data = destination, x = 'destination_st' , y = 'count',color = 'red',label = 'Destination')
plt.tight_layout()
plt.show()
```

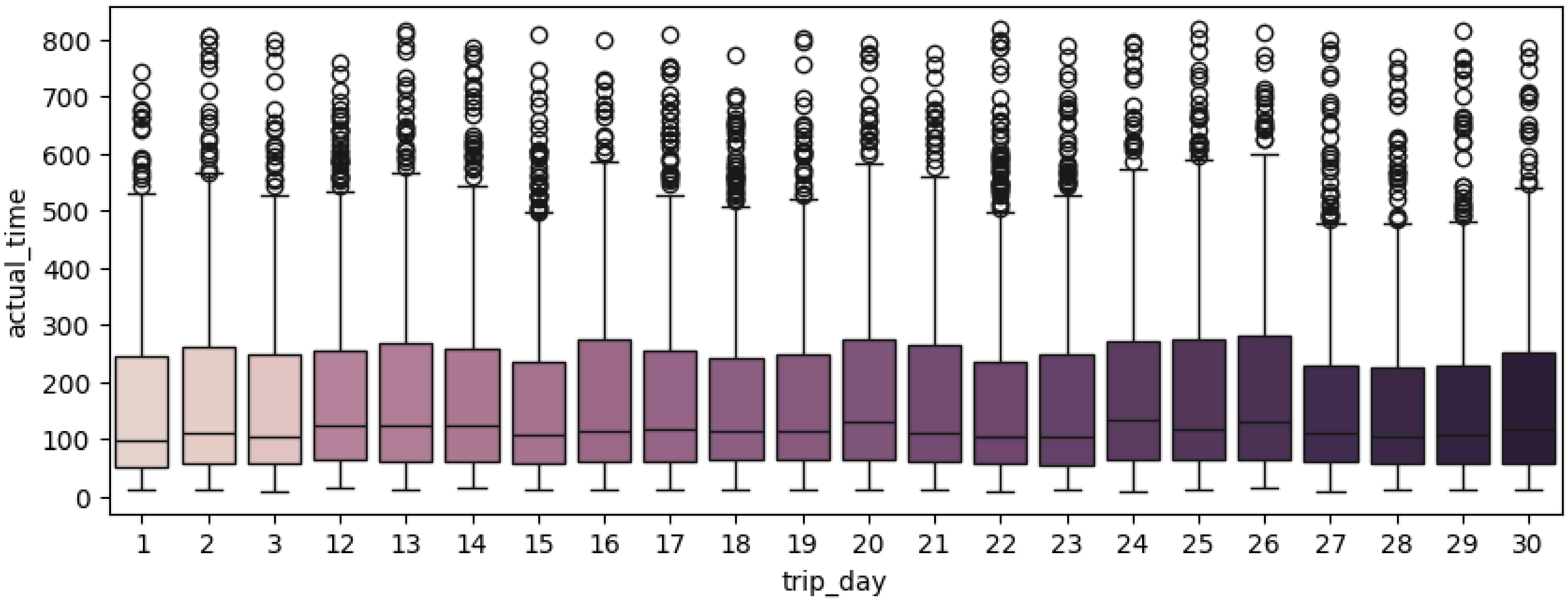


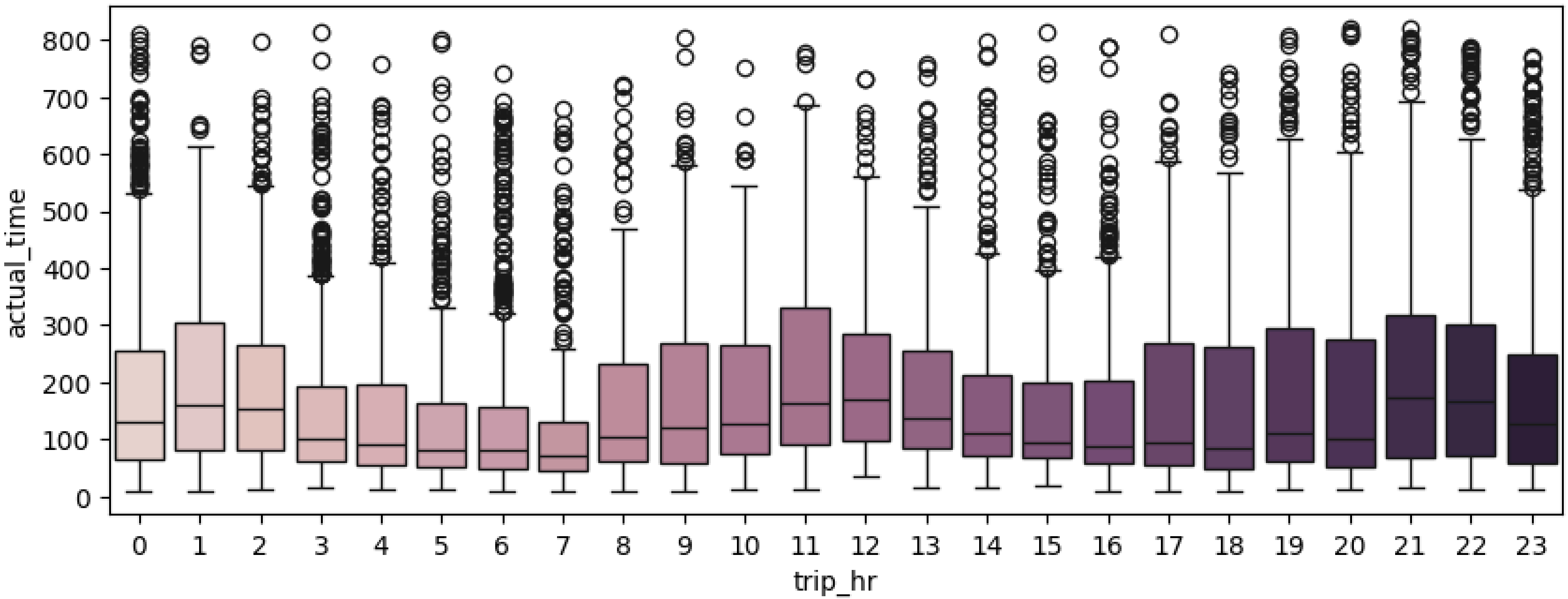
Observation

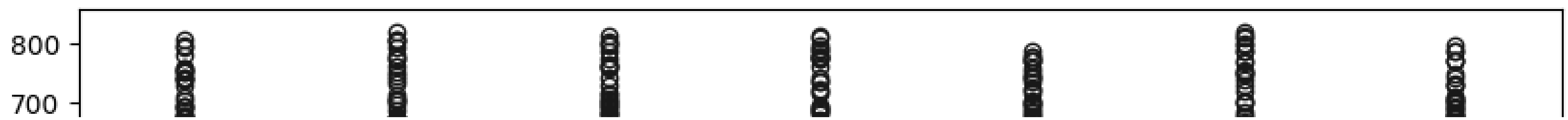
- Maharashtra, Karnataka, and Haryana are the top three source states and as well as destination states with the highest number of trips.
- There are extremely low number of trips in states like Nagaland,Arunachal Pradesh,Meghalaya,Mizoram,Tripura,Daman and Diu

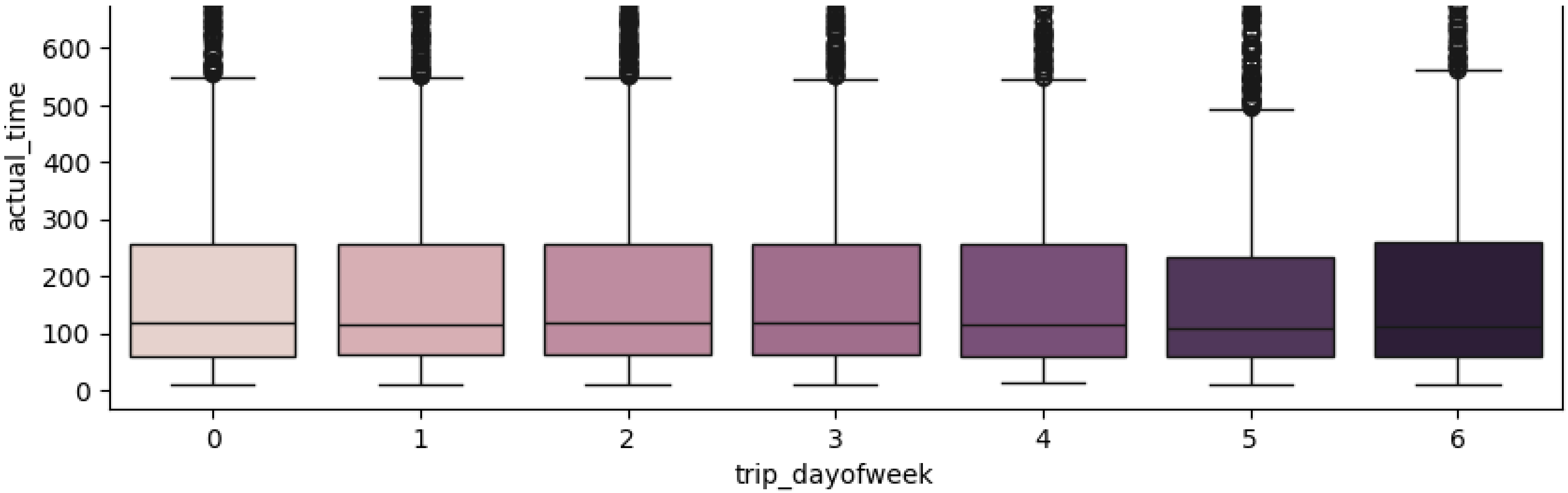
In [906…

```
fig, ax = plt.subplots(3,1,figsize=(10,12))

sns.boxplot(data=trip_df, x='trip_day', y='actual_time', ax=ax[0],hue = 'trip_day',legend = False)
sns.boxplot(data=trip_df, x='trip_hr', y='actual_time', ax=ax[1],hue = 'trip_hr',legend = False)
sns.boxplot(data=trip_df, x='trip_dayofweek', y='actual_time', ax=ax[2],hue = 'trip_dayofweek',legend = False)

plt.show();
```


actual_time
0
100
200
300
400
500
600
700
800
1 2 3 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
trip_day


actual_time
0
100
200
300
400
500
600
700
800
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
trip_hr


800
700

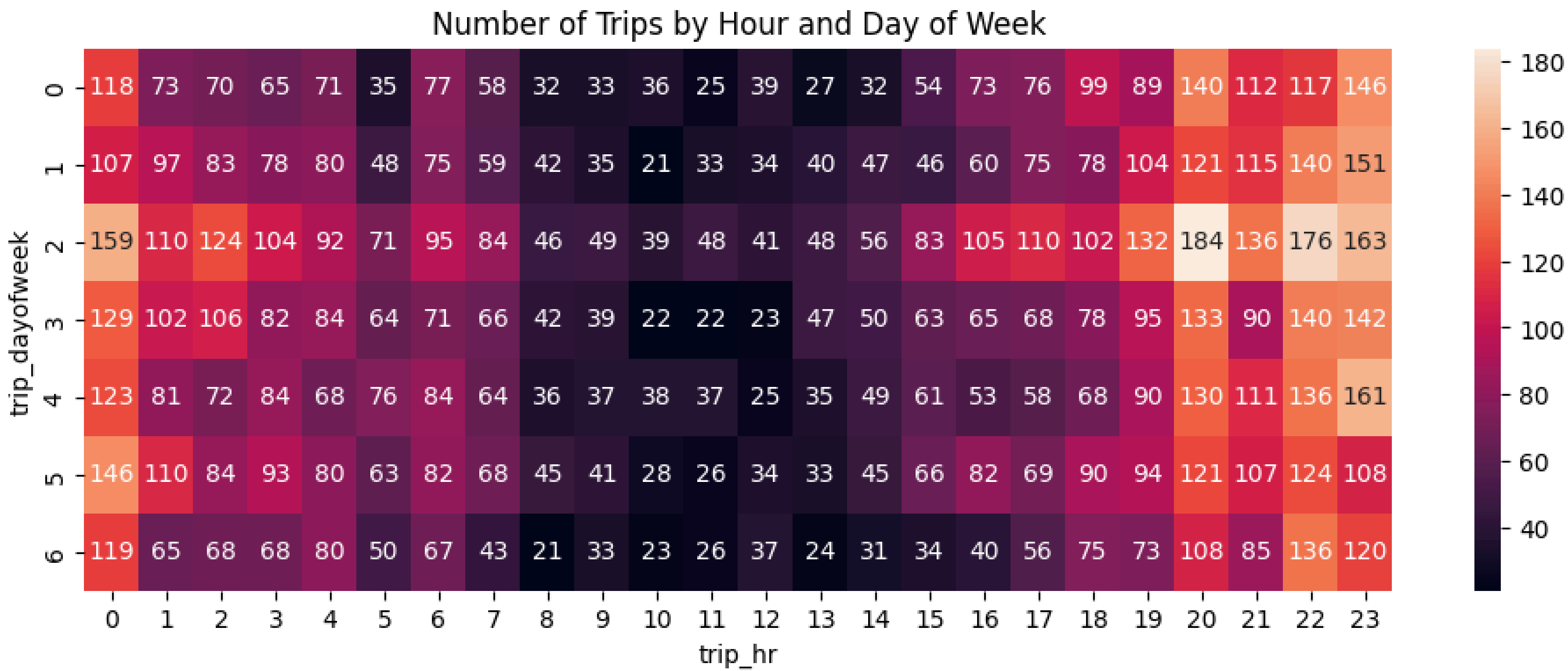


Observation

- Trip day doesn't have any impact on Actual delivery time
- Deliveries are quicker at 3,4,5,6,7 am and 14,15,16,17,18 pm where as in between these hours median delivery times are comparitively higher.
- Weekday of delivery has no impact on the actual delivery time

In [907…

```
#Number of Trips by hour and Day of week
data = pd.pivot_table(data=trip_df, index='trip_dayofweek', columns='trip_hr', values='trip_uuid', aggfunc='count')
plt.figure(figsize=(12,4))
sns.heatmap(data, annot=True, fmt='d')
plt.title('Number of Trips by Hour and Day of Week')
plt.show()
```



Observation:

- Wednesday is the busiest day of the week with maximum number of trips
- 10pm-1am is the busiest time of the day having maximum number of trips (probably because the delivery time is least during these hours - less traffic on the roads)

In [908…

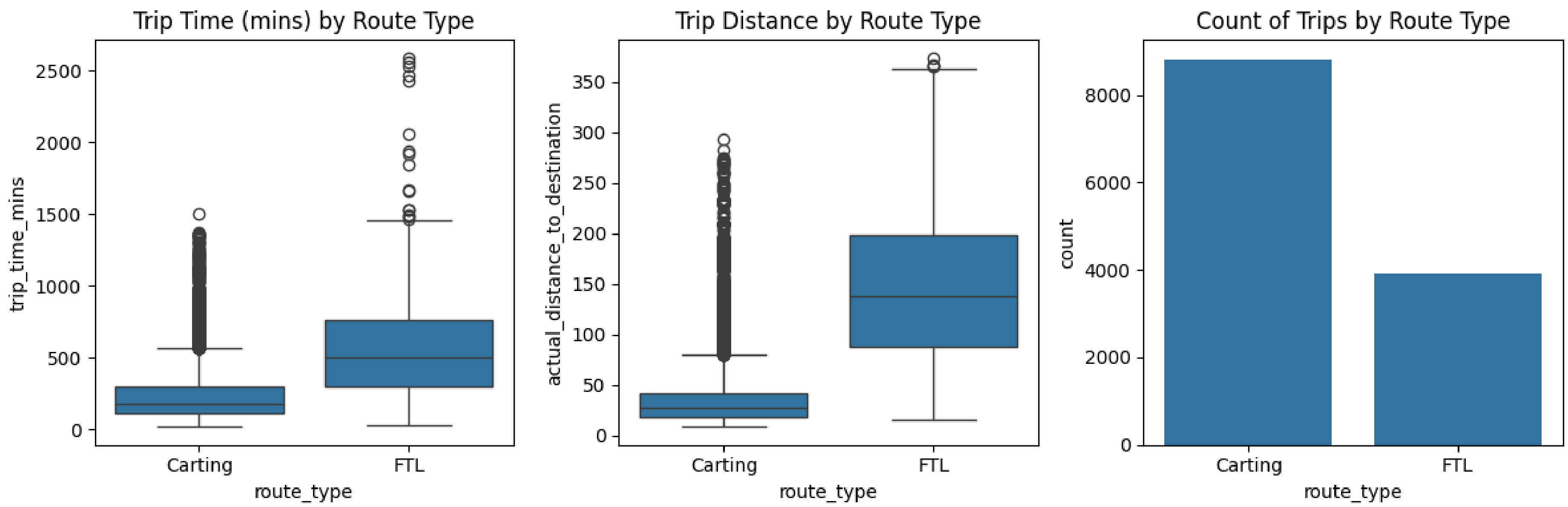
```
# Most frequent pathways
trip_df.groupby(['source_name','destination_name'])['trip_uuid'].count().sort_values(ascending=False).reset_index().he
```

Out[908]:

	source_name	destination_name	trip_uuid
0	Bangalore_Nelmngla_H (Karnataka)	Bengaluru_KGAirprt_HB (Karnataka)	151
1	Chandigarh_Mehmdpur_H (Punjab)	Chandigarh_Mehmdpur_H (Punjab)	122
2	Bengaluru_Bomsndra_HB (Karnataka)	Bengaluru_KGAirprt_HB (Karnataka)	121
3	Bhiwandi_Mankoli_HB (Maharashtra)	Bhiwandi_Mankoli_HB (Maharashtra)	111
4	Bengaluru_KGAirprt_HB (Karnataka)	Bangalore_Nelmngla_H (Karnataka)	108
5	Ahmedabad_East_H_1 (Gujarat)	Ahmedabad_East_H_1 (Gujarat)	107
6	Bhiwandi_Mankoli_HB (Maharashtra)	Mumbai Hub (Maharashtra)	105
7	Mumbai_Chndivli_PC (Maharashtra)	Bhiwandi_Mankoli_HB (Maharashtra)	99
8	Bangalore_Nelmngla_H (Karnataka)	Bengaluru_Bomsndra_HB (Karnataka)	97
9	Gurgaon_Bilaspur_HB (Haryana)	Sonipat_Kundli_H (Haryana)	92

In [909…

```
#time and distance by route_type

fig, ax = plt.subplots(1,3,figsize=(12,4))

sns.boxplot(data=trip_df, x='route_type', y='trip_time_mins', ax=ax[0])
sns.boxplot(data=trip_df, x='route_type', y='actual_distance_to_destination', ax=ax[1])
sns.countplot(x=trip_df['route_type'], ax=ax[2])

ax[0].set_title('Trip Time (mins) by Route Type')
ax[1].set_title('Trip Distance by Route Type')
ax[2].set_title('Count of Trips by Route Type')

plt.tight_layout()
plt.show();
```



Observation

- Carting Routes used for shorter trips within a distance range of 0 to 100 km or duration Less than 500 minutes (about 8.3 hours)
- FTL (Full Truckload) Routes used for longer trips of distance Greater than 100 km and duration of More than 300 minutes (5 hours)
- The number of FTL trips is half that of Carting trips
- In other words, for every two Carting trips, there is one FTL trip

In [910…

```
# Extract unique state codes from both source_state and destination_state columns
source_names = trip_df['source_st'].unique()
destination_names = trip_df['destination_st'].unique()
```

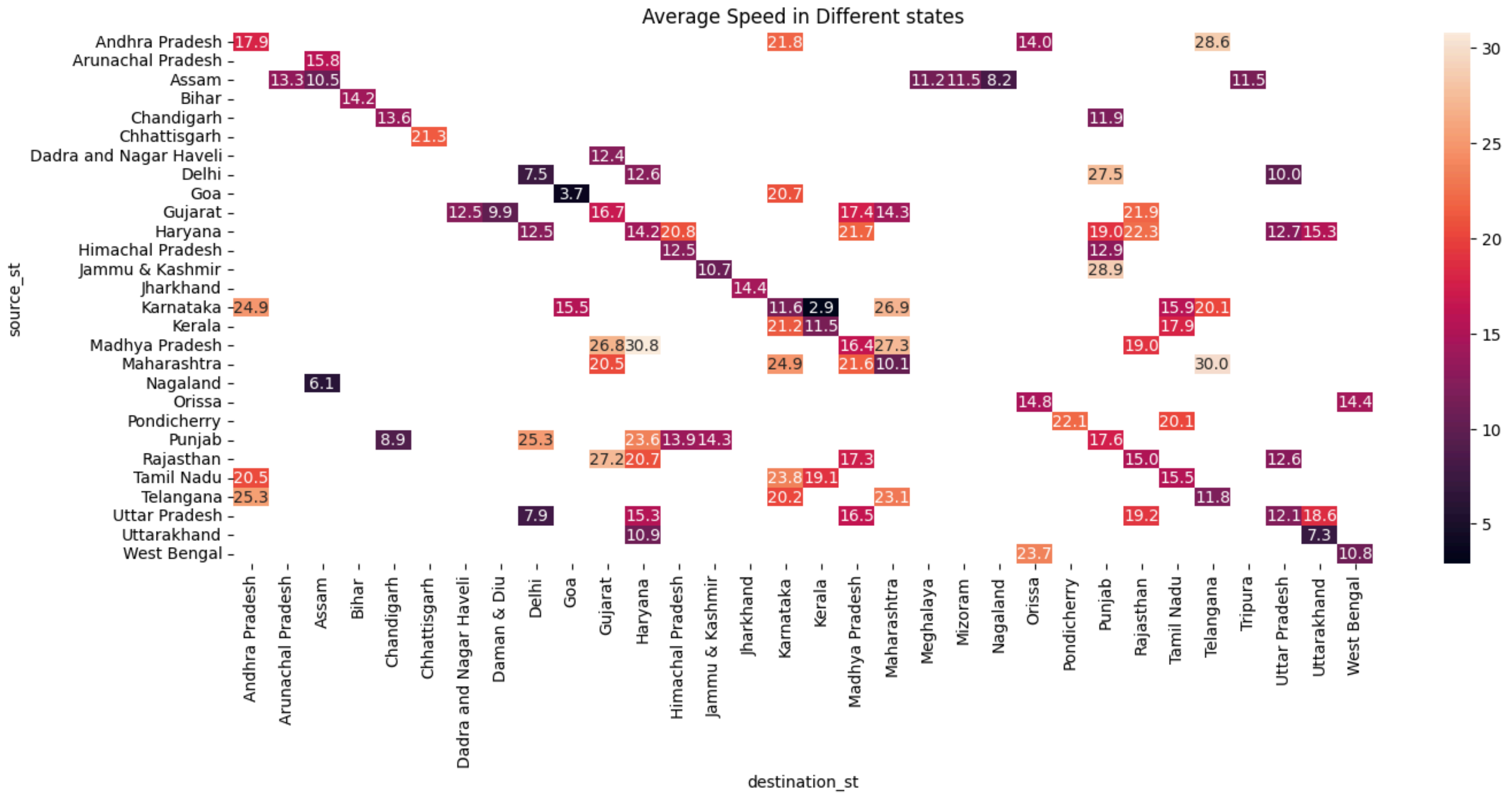
In [911…

```
unique_states = pd.concat([pd.Series(source_names), pd.Series(destination_names)]).unique()
colors = sns.color_palette('husl', len(unique_states))

# Create the color dictionary
state_colors = dict(zip(unique_states, colors))
```

In [912…

```
#Average trip time by source and destination state
flow = trip_df[(trip_df["source_st"].isin(state_colors.keys())) & (trip_df["destination_st"].isin(state_colors.keys())
```

```
flow['speed'] = flow['actual_distance_to_destination']/(flow['trip_time_mins']/60)
data = pd.pivot_table(data=flow, index='source_st', columns='destination_st', values='speed', aggfunc='mean')
plt.figure(figsize=(16,6))
sns.heatmap(data, annot=True, fmt='.1f')
plt.title('Average Speed in Different states')
plt.show()
```



Observation:

1. Inter-state vs. Intra-state Delivery Speeds:

- Inter-state deliveries show significantly higher average speeds compared to intra-state deliveries.

1. State-specific Intra-state Delivery Speed Variations:

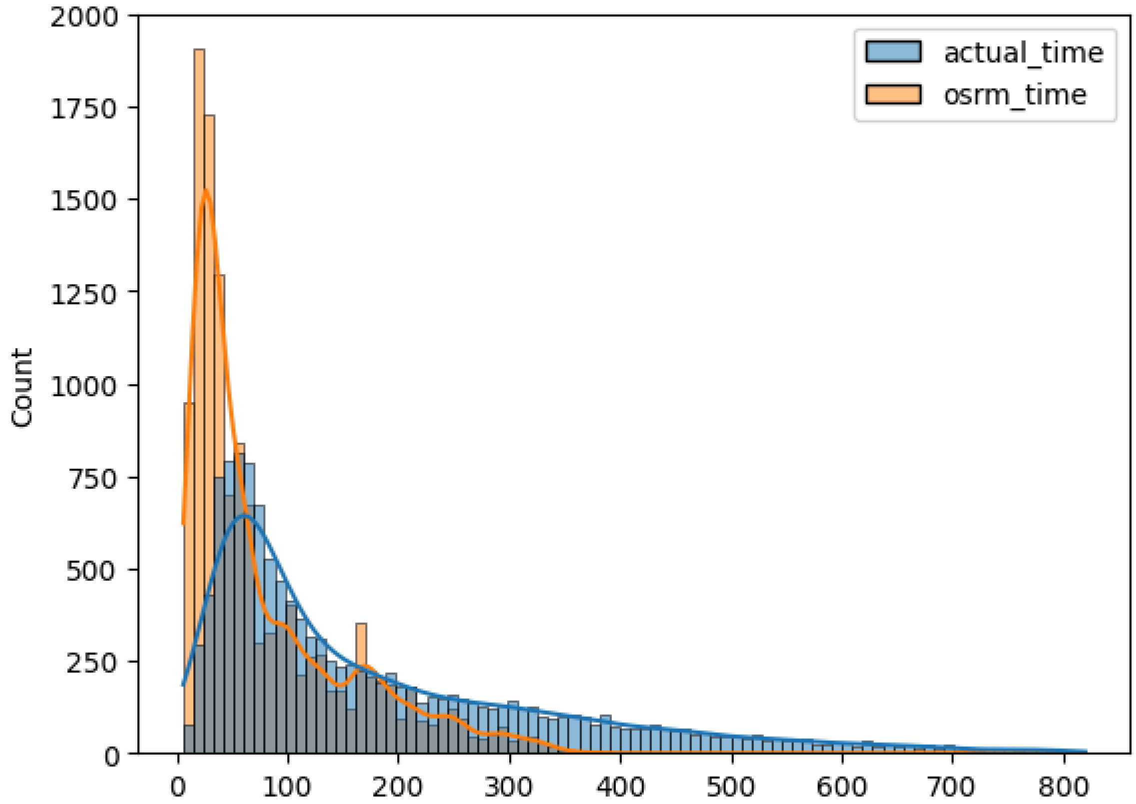
- Among intra-state deliveries: Delhi exhibits the slowest average speed. Punjab demonstrates the fastest average speed.

Hypothesis testing and visual analysis

Actual time vs OSRM time

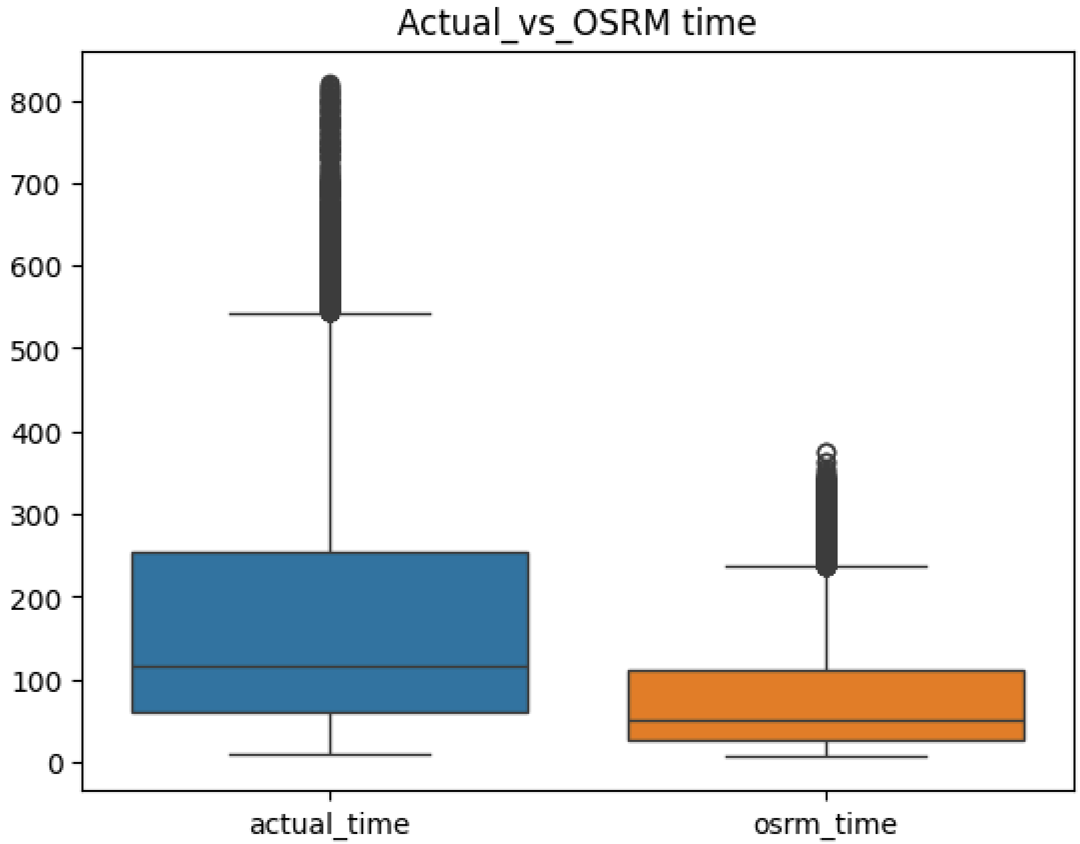
In [913…

```
sns.histplot(data = trip_df[['actual_time','osrm_time']],kde = True)
plt.show()
```



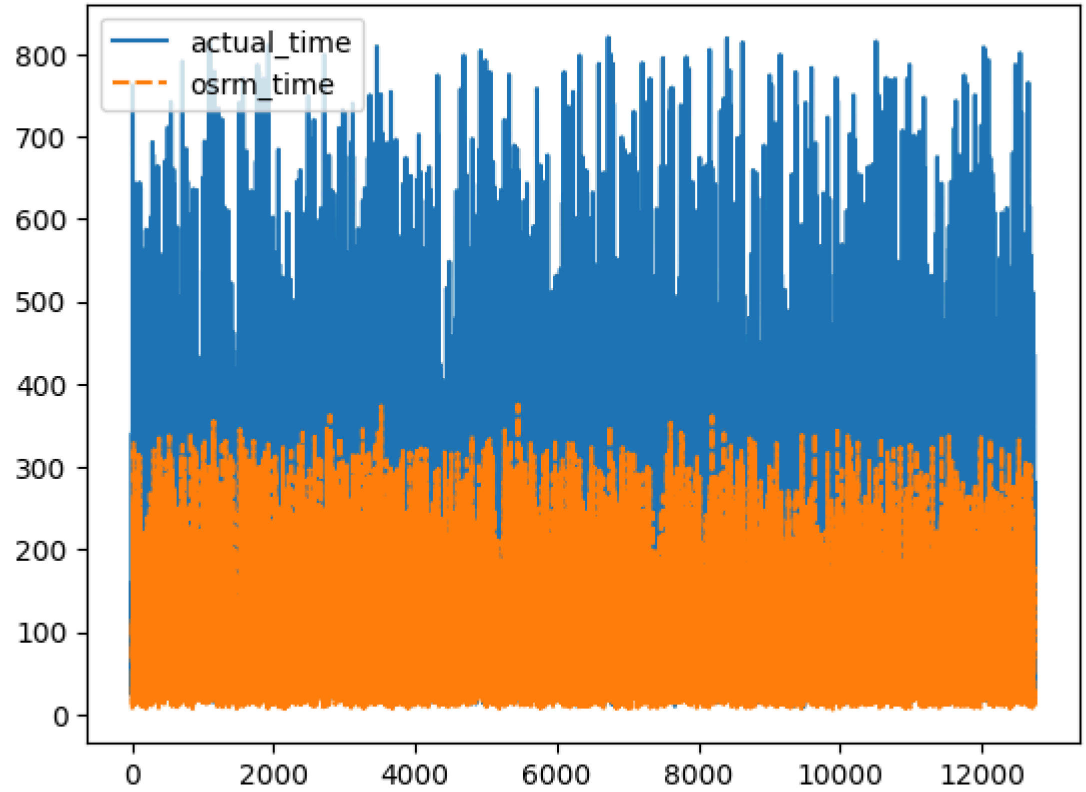
In [914…

```
sns.boxplot(data = trip_df[['actual_time','osrm_time']])
plt.title('Actual_vs_OSRM time')
plt.show()
```



In [915…

```
sns.lineplot(data = trip_df[['actual_time','osrm_time']])
plt.show()
```



Observation:

- OSRM time has less spread than Actual time in the above histplot, and individually both the plots are right skewed
- Similarly even in boxplot and lineplot Actual time is more than OSRM time

Attempt to normalise data using log noraml or box cox

In [916…

```
a=np.log(trip_df['actual_time'])
b=np.log(trip_df['osrm_time'])

from scipy.stats import shapiro
a_sample = a.sample(1000)
```

```
stat,p_val = shapiro(a_sample)
if p_val<0.05:
  print("distribution is not normal after log transforamtion")
else:
  print("distribution is normal after log transformation")
```

```
distribution is not normal after log transforamtion
```

In [917…

```
from scipy.stats import shapiro
b_sample = b.sample(1000)
stat,p_val = shapiro(b_sample)
if p_val<0.05:
  print("distribution is not normal after log transforamtion")
else:
  print("distribution is normal after log transformation")
```

```
distribution is not normal after log transforamtion
```

In [918…

```
from scipy.stats import boxcox

a_boxcox, lambda_a = boxcox(trip_df['actual_time'] + 1)  # Add 1 if there are zeros
b_boxcox, lambda_b = boxcox(trip_df['osrm_time'] + 1)  # Add 1 if there are zeros
```

In [919…

```
a_boxcox
sample_size = 1000
a_boxcox_sample = np.random.choice(a_boxcox, size=sample_size, replace=False)
stat,p_val = shapiro(a_boxcox_sample)
if p_val<0.05:
  print("distribution is not normal after boxcox transformation")
else:
  print("distribution is normal after boxcox transformation")
```

```
distribution is not normal after boxcox transformation
```

In [920…

```
b_boxcox
sample_size = 1000
b_boxcox_sample = np.random.choice(b_boxcox, size=sample_size, replace=False)
stat,p_val = shapiro(b_boxcox_sample)
if p_val<0.05:
  print("distribution is not normal after boxcox transformation")
else:
  print("distribution is normal after boxcox transformation")
```

```
distribution is not normal after boxcox transformation
```

In [921…

```
a = trip_df['osrm_time']
b = trip_df['actual_time']
alpha = 0.05
# performing levene's test to check if the variances are equal or not
stat,p_val = levene(a,b)
print(stat,p_val)
if p_val<alpha:
  print("reject null hypotheis and conclude variances are not equal")
else:
  print("failed to reject null hypothesis and conclude variances are equal")
```

```
2646.4350925441204 0.0
reject null hypotheis and conclude variances are not equal
```

- Observation:
- Data is not normally distributed even after applying log normal or boxcox transformation
- variances are not equal so therefore it doesn't obey ttest independent criteria's
- But since size of the data is large and by virtue of CLT we can perform ttest_ind to support the visual outcomes

Hypothesis:

- Ho : The distributions of actual_time and osrm_time are the same.
- Ha : The distribution of actual_time is greater than the distribution of osrm_time.

In [922…

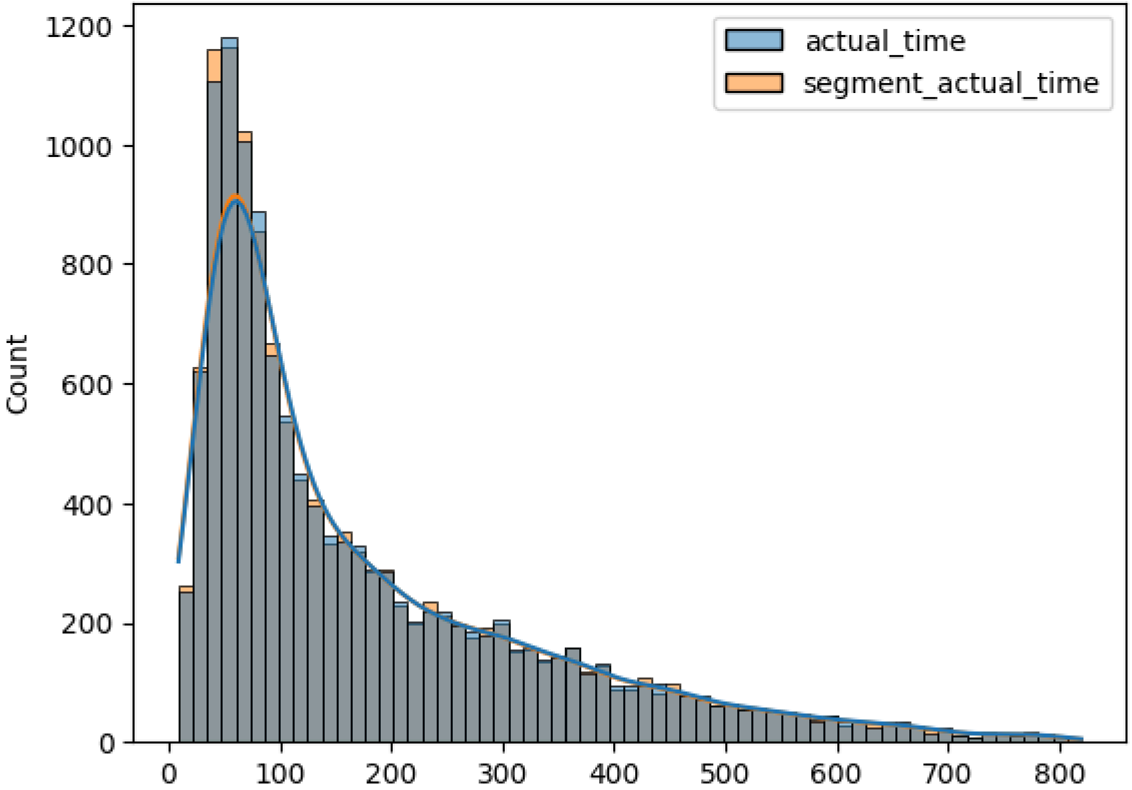
```
a = trip_df['actual_time']
b = trip_df['osrm_time']

t_stat, p_value = ttest_ind(a, b, alternative='greater',equal_var = False)

print(f"t_stat: {stat},P-value: {p_value}")


# Interpretation
if p_value < 0.05:
    print("Reject null hypothesis. The distribution of 'actual_time' is significantly greater than 'osrm_time'.")
else:
    print("Failed to reject null hypothesis. There is no significant evidence that 'actual_time' is greater than 'osrm
```

```
t_stat: 2646.4350925441204,P-value: 0.0
Reject null hypothesis. The distribution of 'actual_time' is significantly greater than 'osrm_time'.
```

Actual Time Vs Segment Actual Time

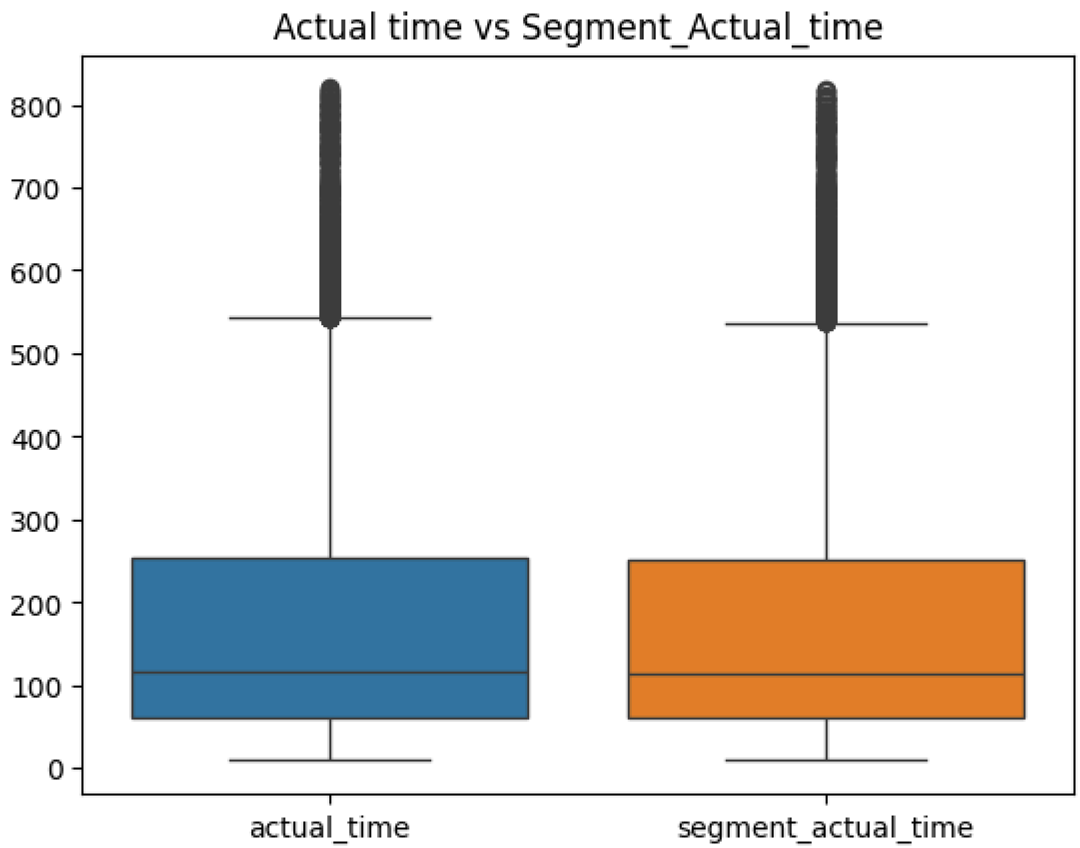
In [923…

```
sns.histplot(trip_df[['actual_time','segment_actual_time']],kde = True)
plt.show()
```



In [924…

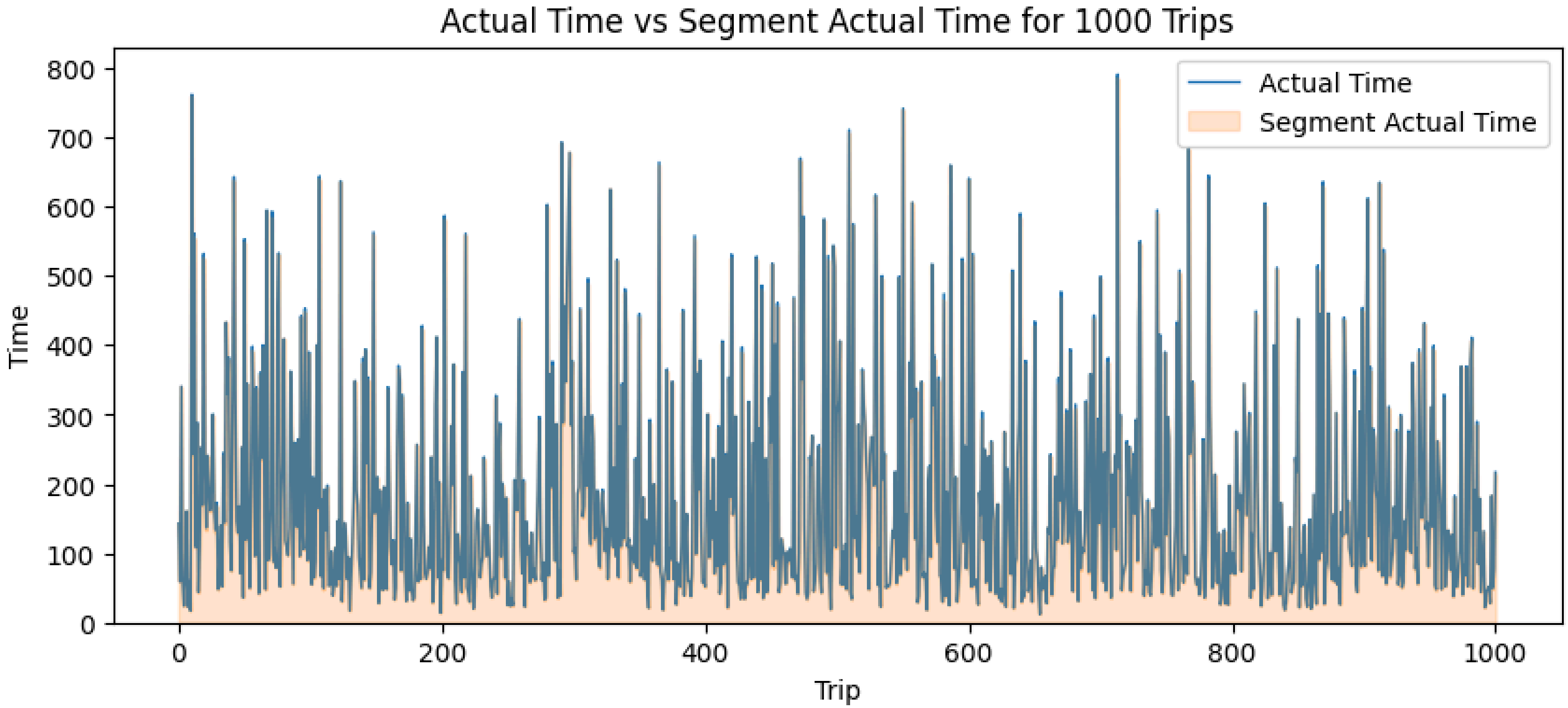
```
sns.boxplot(data = trip_df[['actual_time','segment_actual_time']])
plt.title('Actual time vs Segment_Actual_time')
plt.show()
```



In [925…

```
plt.figure(figsize=(10,4))

# Line plot for 'actual_time'
sns.lineplot(data=trip_df['actual_time'].loc[:1000], label='Actual Time', lw=1)

# Area plot for 'segment_actual_time'
trip_df['segment_actual_time'].loc[:1000].plot(kind='area', alpha=0.2, label='Segment Actual Time')

# Adding titles and labels
plt.title('Actual Time vs Segment Actual Time for 1000 Trips')
plt.xlabel('Trip')
plt.ylabel('Time')

# Displaying the legend
plt.legend()
```

```python
# Show the plot
plt.show()
```



Observation:

- Both actual time and segment actual time are right-skewed and not normally distributed
- We can see from the boxplot and the lineplot that the actual time and segment actual time do not differ much.

In [926…

```python
a = trip_df['actual_time']
b = trip_df['segment_actual_time']

stat,p_val = levene(a,b)
print(f"levenes_stat--> {stat},p_value--> {p_val}")
if p_val<0.05:
  print("variances are not equal and actual and segment actual time differ ")
```

```
else:
  print("variances are equal and they do not differ much")
```

```
levenes_stat--> 0.3096519091893021,p_value--> 0.5778987520533407
variances are equal and they do not differ much
```

- Since the sample size is greater than 30 and variances of both the samples are almost equal we can perform t test independent to support the observation from our visual analysis

Hypothesis:

- Ho : The distributions of actual_time and segment_actual_time are the same.
- Ha : The distribution of actual_time and segment_actual_time are not same

In [927…

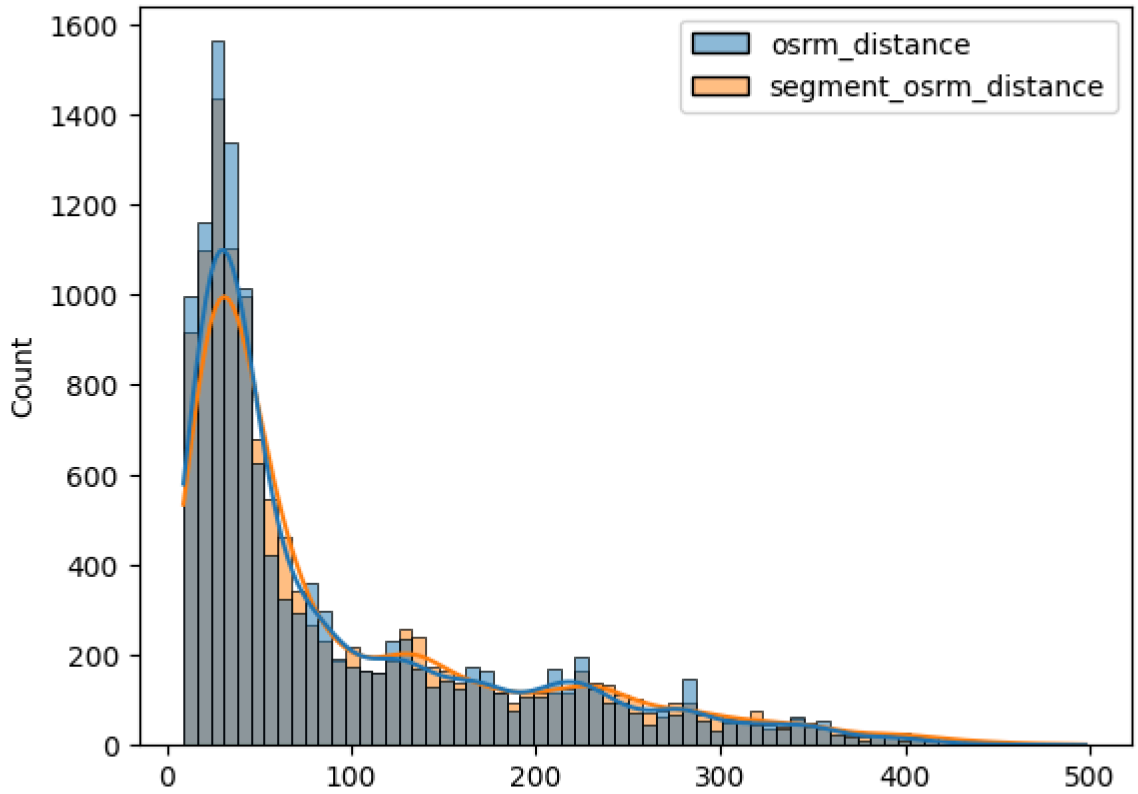
```
from scipy.stats import ttest_ind
stat,p_val = ttest_ind(a,b,alternative = 'two-sided')
alpha= 0.05
print(stat,p_val)
if p_val < alpha:
  print("Actual time and segment actual time are not equal")
else:
  print("Actual time and segment actual time are equal")
```

```
0.8372585086100397 0.40245512700089525
Actual time and segment actual time are equal
```

OSRM distance vs segment OSRM distance

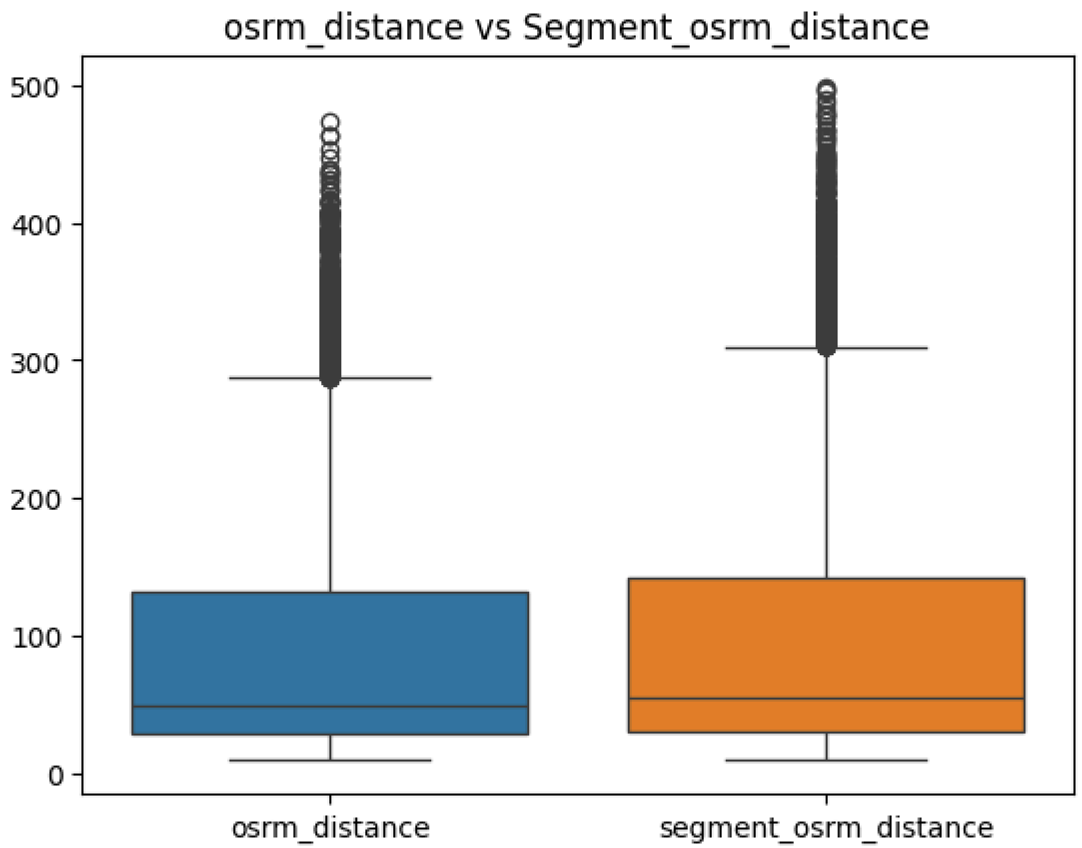
In [928…

```
sns.histplot(trip_df[['osrm_distance','segment_osrm_distance']],kde = True)
plt.show()
```



In [929…

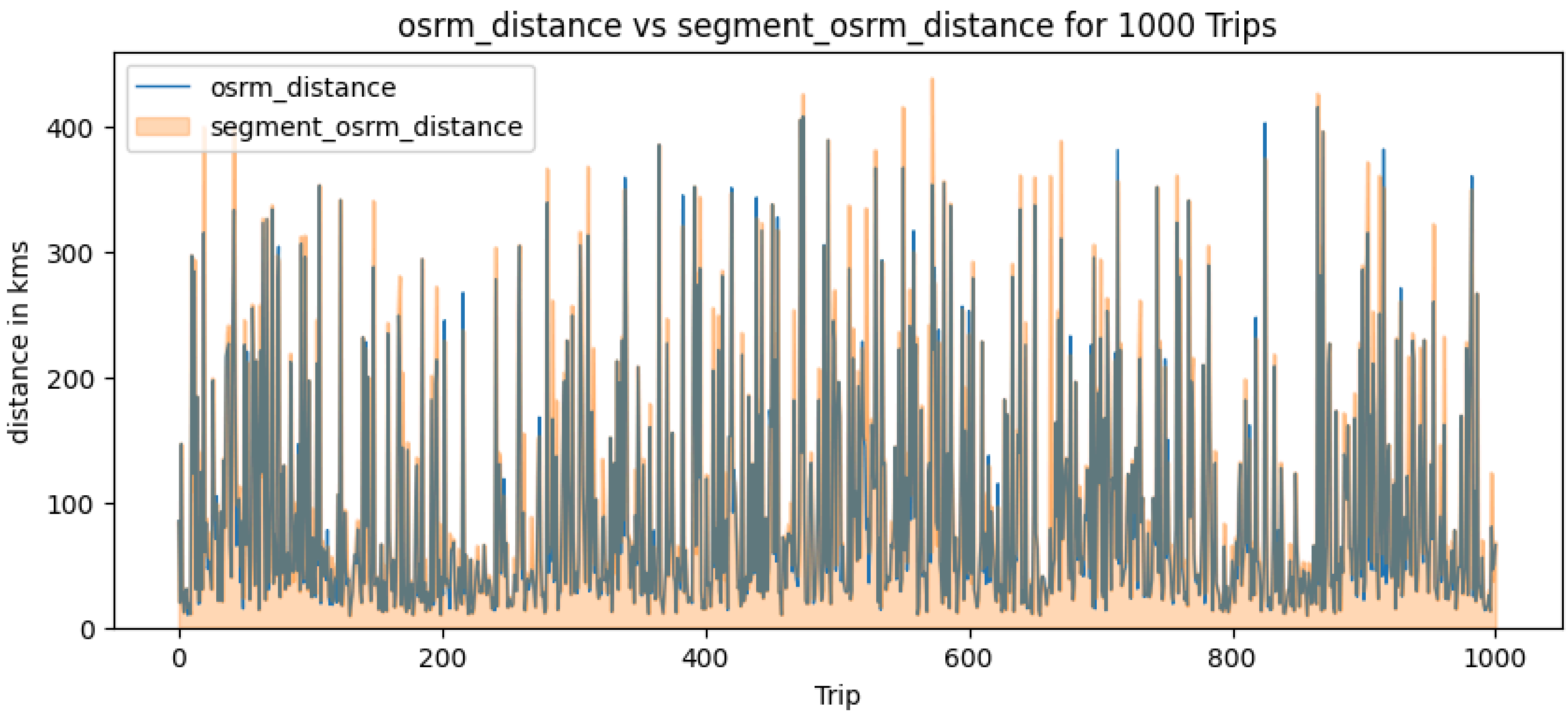
```
sns.boxplot(data = trip_df[['osrm_distance','segment_osrm_distance']])
plt.title('osrm_distance vs Segment_osrm_distance')
plt.show()
```



In [930…

```
plt.figure(figsize=(10,4))
# Line plot for 'osrm_time'
sns.lineplot(data=trip_df['osrm_distance'].loc[:1000], label='osrm_distance', lw=1)

# Area plot for 'segment_osrm_distance'
trip_df['segment_osrm_distance'].loc[:1000].plot(kind='area', alpha=0.3, label='segment_osrm_distance')

# Adding titles and labels
plt.title('osrm_distance vs segment_osrm_distance for 1000 Trips')
plt.xlabel('Trip')
plt.ylabel('distance in kms')

# Displaying the legend
plt.legend()
```

```
# Show the plot
plt.show()
```



Observation:

- Distributions for both parameters are very similar with right-skew
- The box plot shows a small difference between the mean values of osrm distance and segment osrm distance
- In the sample data of 1000 trips (lineplot), we see that osrm distance is lesser than segment osrm distance in most cases

Hypothesis:

- Ho : The distributions of osrm_distance and segment_osrm_distance are the same.
- Ha : The distributions of osrm_distance is lesser than segment_osrm_distance are the same.

In [931…
```
a = trip_df['osrm_distance'].var()
b = trip_df['segment_osrm_distance'].var()

print(f"Variance of osrm distance :---> {a}\nVariance of segment osrm distance:--->{b}" )
```

```
Variance of osrm distance :---> 8144.041015625
Variance of segment osrm distance:--->9135.052734375
```

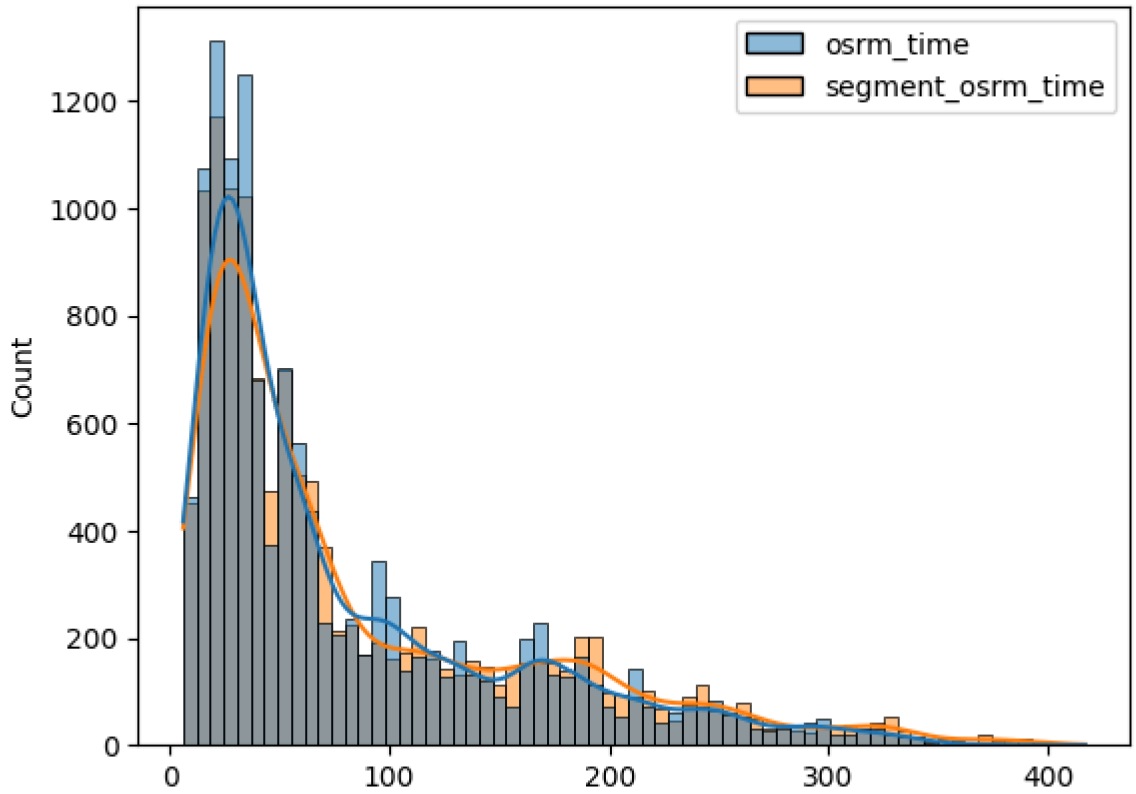
- Though variances are not equal yet the sample size is greater, by virtue of CLT we can perform ttest independent test

In [932…
```
from scipy.stats import ttest_ind

a = trip_df['osrm_distance']
b = trip_df['segment_osrm_distance']
alpha = 0.05

t_stat, p_val = ttest_ind(a, b, equal_var=False, alternative='less')

print('Test Statistic:', t_stat)
print('P value:', p_val)

if p_val<alpha:
  print("Reject null hypothesis and conclude osrm_distance is lesser than segment_osrm_distance")
else:
  print("Fail to reject null hypothesisa and osrm_distance is not lesser than segment_osrm_distance")
```

```
Test Statistic: -5.410997487356375
P value: 3.1620587525789336e-08
Reject null hypothesis and conclude osrm_distance is lesser than segment_osrm_distance
```

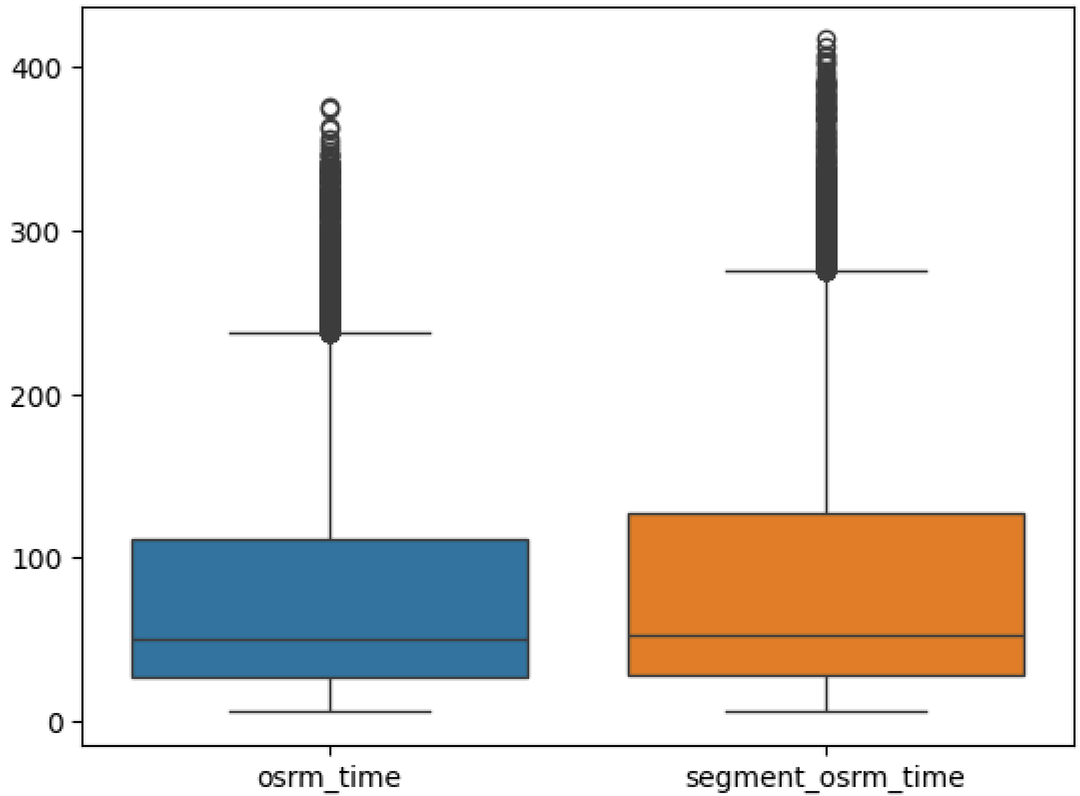
OSRM time VS Segment OSMR time

In [933…
```
sns.histplot(trip_df[['osrm_time','segment_osrm_time']],kde = True)
plt.show()
```



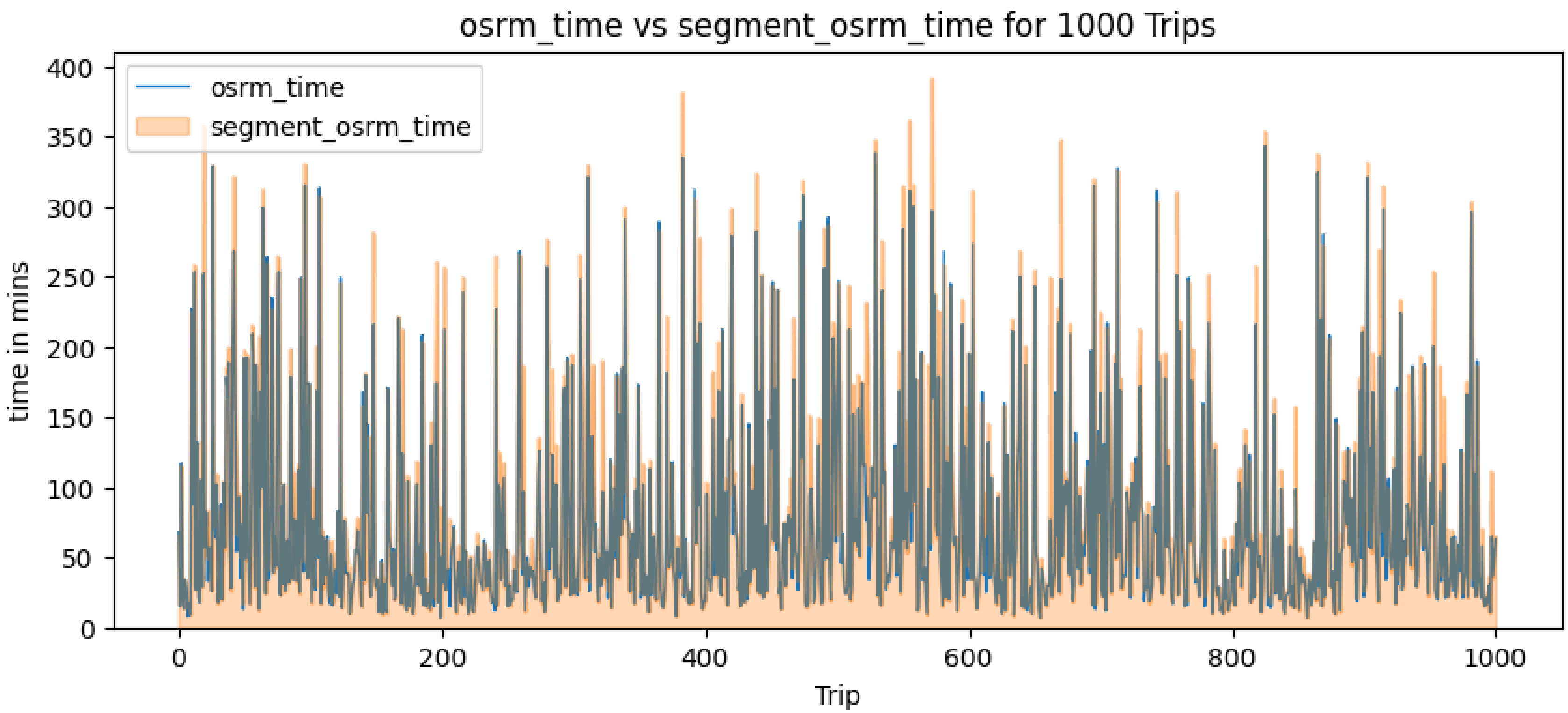
In [934…

```
sns.boxplot(trip_df[['osrm_time','segment_osrm_time']])
plt.show()
```



In [935…

```
plt.figure(figsize=(10,4))
# Line plot for 'osrm_time'
sns.lineplot(data=trip_df['osrm_time'].loc[:1000], label='osrm_time', lw=1)

# Area plot for 'segment_osrm_distance'
trip_df['segment_osrm_time'].loc[:1000].plot(kind='area', alpha=0.3, label='segment_osrm_time')

# Adding titles and labels
plt.title('osrm_time vs segment_osrm_time for 1000 Trips')
plt.xlabel('Trip')
plt.ylabel('time in mins')

# Displaying the legend
plt.legend()

# Show the plot
plt.show()
```



Observation:

- The boxplot and the lineplot of 1000 trips shows that osrm_time is lesser than segment_osrm_time
- The distributions are right skewed

Hypothesis:

- Ho : The distributions of osrm_time and segment_osrm_time are the same.
- Ha : The distributions of osrm_time is lesser than segment_osrm_time are the same.

In [936…

```
print('Variance of OSRM Time:',trip_df.osrm_time.var())
print('Variance of Segment OSRM Time', trip_df.segment_osrm_time.var())
```

```
Variance of OSRM Time: 5314.1445
Variance of Segment OSRM Time 6439.15
```

In [937…

```
from scipy.stats import ttest_ind

a = trip_df.osrm_time
b = trip_df.segment_osrm_time
alpha = 0.05

t_stat, p_value = ttest_ind(a, b, equal_var=False,alternative='less')

print('Test Statistic:', t_stat)
print('P value:', p_value)

if p_value < alpha:
  print("Reject null hypothesis and conclude osrm_time is lesser than segment_osrm_time")
else:
  print("Fail to reject null hypothesis and conclude osrm_time is not lesser than segment_osrm_time")
```

```
Test Statistic: -7.84573574222561
P value: 2.2369342780892427e-15
Reject null hypothesis and conclude osrm_time is lesser than segment_osrm_time
```

Standardising the numerical columns

In [938…

```
num_col
```

Out[938]:

```
['start_scan_to_end_scan',
 'actual_distance_to_destination',
 'actual_time',
 'osrm_time',
 'osrm_distance',
 'segment_actual_time',
 'segment_osrm_time',
 'segment_osrm_distance']
```

In [939…

```
from sklearn.preprocessing import StandardScaler
```

In [940…

```
trip = trip_df.copy()
scaler = StandardScaler()
scaler.fit(trip[num_col])
```

```python
trip[num_col] = scaler.transform(trip[num_col])
trip[num_col]
```

Out[940]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time	segment_osrm_
0	-0.552018	0.004770	-0.223725	-0.150799	-0.080739	-0.227340	-0.26
1	-0.862649	-0.766652	-0.751424	-0.877869	-0.805810	-0.745909	-0.87
2	1.533093	0.752090	1.020137	0.521398	0.602775	1.031139	0.35
3	-0.517072	-0.664436	-0.738859	-0.768123	-0.712895	-0.739585	-0.79
4	-0.870415	-0.877862	-0.971298	-0.905306	-0.890712	-0.967250	-0.91
...	...	...	...	...	...	...	
12739	-0.253036	-0.207666	-0.600652	-0.233109	-0.209816	-0.600457	-0.30
12740	-1.017965	-0.789535	-0.990144	-0.919024	-0.845612	-0.986222	-0.94
12741	0.383758	-0.470411	0.649492	-0.425165	-0.371154	0.658022	0.01
12742	0.096424	0.852289	0.536413	1.371933	0.872259	0.512570	1.67
12743	0.119722	-0.093089	0.605517	-0.150799	-0.130963	0.613754	-0.24

12744 rows × 8 columns

Insights

1. OSRM vs. Segment OSRM: On average, the OSRM time and distance are lower compared to segment OSRM time and distance.
2. Actual Time vs. Segment Actual Time: There is no significant difference between actual time and segment actual time.
3. Actual vs. OSRM Time: The average actual delivery time is notably higher than the OSRM estimated time. While the maximum OSRM time reaches 400 minutes (6.6 hours), the actual delivery time can extend up to 800 minutes (13 hours), nearly doubling the estimated time.
4. Peak Delivery Times: Delivery times are notably longer between 9:00 AM to 12:00 PM and 5:00 PM to 10:00 PM.

5. Busiest Day: Wednesday experiences the highest number of trips, making it the busiest day of the week.
6. Busiest Time: The hours between 10:00 PM and 1:00 AM see the highest number of trips, likely due to shorter delivery times during these low-traffic hours.
7. Route Types: Carting routes are typically used for short-distance (0-100 km) and short-duration (<500 minutes) trips, while FTL (Full Truck Load) routes are employed for longer distances (>100 km) and extended durations (>300 minutes).
8. FTL vs. Carting Trips: FTL trips account for 50% of the count of carting trips.
9. Inter-State vs. Intra-State Deliveries: The average speed for inter-state deliveries is considerably higher than for intra-state deliveries.
10. State Delivery Speeds: Among states, Delhi records the lowest intra-state delivery speed, whereas Punjab has the highest.

Recommendations

1. Given that actual delivery times consistently exceed OSRM estimates, it's crucial for the company to enhance forecasting accuracy or pinpoint the underlying causes of delivery delays.
2. Extract best practices from high-volume states like Maharashtra and Karnataka to boost delivery operations in other regions.
3. To minimize delivery times, prioritize dispatching orders during off-peak hours.
4. Optimize routing along high-speed corridors to further reduce delivery durations.

In [940…