In [721…

```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181
To: /content/delhivery_data.csv?1642751181
100% 55.6M/55.6M [00:00<00:00, 119MB/s]
```

About Delhivery:

Delhivery is a leading player in India's commerce landscape, distinguished by its robust infrastructure, top-tier logistics operations, and advanced engineering and technology capabilities. Their ambition is to establish the operating system for commerce, leveraging their strengths to provide exceptional quality, efficiency, and profitability. The Data team plays a crucial role by harnessing the company's data to gain insights and capabilities that give Delhivery a competitive edge over rivals. Their efforts focus on enhancing business quality, streamlining operations, and boosting profitability compared to competitors.

Business Problem

The goal of this case study is to help Delhivery's Data team use the company's data better. They need to clean and organize the data so they can make accurate predictions and create machine learning models. We'll focus on making the data processing easier for them, which will help them make smarter decisions and grow the business.

The objective is:

- Clean, sanitize and manipulate data to get useful features out of raw fields.
- Make sense out of the raw data and help the data science team to build forecasting models on it

Importing Libraries

In [722…

```
import pandas as pd
from scipy import stats
from scipy.stats import ttest_ind
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

Data Overview:

Load Dataset

In [723…

```
data = pd.read_csv('delhivery_data.csv?1642751181')
data.head()
```

Out[723]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB

5 rows × 24 columns

In [724…

```
data[['od_start_time','od_end_time','is_cutoff']]
```

Out[724]:

	od_start_time	od_end_time	is_cutoff
0	2018-09-20 03:21:32.418600	2018-09-20 04:47:45.236797	True
1	2018-09-20 03:21:32.418600	2018-09-20 04:47:45.236797	True
2	2018-09-20 03:21:32.418600	2018-09-20 04:47:45.236797	True
3	2018-09-20 03:21:32.418600	2018-09-20 04:47:45.236797	True
4	2018-09-20 03:21:32.418600	2018-09-20 04:47:45.236797	False
...	...	...	...
144862	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	True
144863	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	True
144864	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	True
144865	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	True
144866	2018-09-20 16:24:28.436231	2018-09-20 23:32:09.618069	False

144867 rows × 3 columns

In [725…

```
data.shape
```

Out[725]:

```
(144867, 24)
```

Inference

- size of data is 144867 X 24
- total rows 144867
- total columns 24

In [726…

```
data.columns
```

Out[726]:

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

In [727…

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   data                            144867 non-null  object 
 1   trip_creation_time              144867 non-null  object 
 2   route_schedule_uuid             144867 non-null  object 
 3   route_type                      144867 non-null  object 
 4   trip_uuid                       144867 non-null  object 
 5   source_center                   144867 non-null  object 
 6   source_name                     144574 non-null  object 
 7   destination_center              144867 non-null  object 
 8   destination_name                144606 non-null  object 
 9   od_start_time                   144867 non-null  object 
 10  od_end_time                     144867 non-null  object 
 11  start_scan_to_end_scan          144867 non-null  float64
 12  is_cutoff                       144867 non-null  bool   
 13  cutoff_factor                   144867 non-null  int64  
 14  cutoff_timestamp                144867 non-null  object 
 15  actual_distance_to_destination  144867 non-null  float64
 16  actual_time                     144867 non-null  float64
 17  osrm_time                       144867 non-null  float64
 18  osrm_distance                   144867 non-null  float64
 19  factor                          144867 non-null  float64
 20  segment_actual_time             144867 non-null  float64
 21  segment_osrm_time               144867 non-null  float64
 22  segment_osrm_distance           144867 non-null  float64
 23  segment_factor                  144867 non-null  float64
dtypes: bool(1), float64(10), int64(1), object(12)
memory usage: 25.6+ MB
```

Inference

- There is a need to adjust the data types of certain columns
  - *trip_creation_time --> datetime*
  - *od_start_time --> datetime*
  - *od_end_time -->datetime*
- There is also a need to change columns name of certain columns.
  - *start_scan_to_end_scan to total_trip_time*
  - *source_center to source_id*
  - *destination_center to destination_id*
- There are some null values in columns *source_name,destiation_name*.

In [728…

```
##changing datatype and column name
data['trip_creation_time'] = pd.to_datetime(data['trip_creation_time'])
data['od_start_time'] = pd.to_datetime(data['od_start_time'])
data['od_end_time'] = pd.to_datetime(data['od_end_time'])
data.rename(columns = {'start_scan_to_end_scan':'total_trip_time','source_center':'source_id','destination_center':'des
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144867 non-null  object
 1   trip_creation_time              144867 non-null  datetime64[ns]
 2   route_schedule_uuid             144867 non-null  object
 3   route_type                      144867 non-null  object
 4   trip_uuid                       144867 non-null  object
 5   source_id                       144867 non-null  object
 6   source_name                     144574 non-null  object
 7   destination_id                  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  total_trip_time                 144867 non-null  float64
 12  is_cutoff                       144867 non-null  bool
 13  cutoff_factor                   144867 non-null  int64
 14  cutoff_timestamp                144867 non-null  object
 15  actual_distance_to_destination  144867 non-null  float64
 16  actual_time                     144867 non-null  float64
 17  osrm_time                       144867 non-null  float64
 18  osrm_distance                   144867 non-null  float64
 19  factor                          144867 non-null  float64
 20  segment_actual_time             144867 non-null  float64
 21  segment_osrm_time               144867 non-null  float64
 22  segment_osrm_distance           144867 non-null  float64
 23  segment_factor                  144867 non-null  float64
dtypes: bool(1), datetime64[ns](3), float64(10), int64(1), object(9)
memory usage: 25.6+ MB
```

In [729…

```
print(data['trip_creation_time'].dt.date.min(),data['trip_creation_time'].dt.date.max())
```

```
2018-09-12 2018-10-03
```

Inference

Data is given from **12 september 2018** to **3 october 2018**

Missing Value Analysis

In [730…

```
data.isnull().sum()
```

Out[730]:

```
data                                0
trip_creation_time                  0
route_schedule_uuid                 0
route_type                          0
trip_uuid                           0
source_id                           0
source_name                       293
destination_id                      0
destination_name                  261
od_start_time                       0
od_end_time                         0
total_trip_time                     0
is_cutoff                           0
cutoff_factor                       0
cutoff_timestamp                    0
actual_distance_to_destination      0
actual_time                         0
osrm_time                           0
osrm_distance                       0
factor                              0
segment_actual_time                 0
segment_osrm_time                   0
segment_osrm_distance               0
segment_factor                      0
dtype: int64
```

Inference

there are 293 null values in source_name and 261 in destination_name

In [731…

```
data[data['source_name'].isnull()]
```

Out[731]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_id	source_name	destination_id	de
112	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	
113	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	
114	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	
115	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	
116	training	2018-09-25 08:53:04.377810	thanos::sroute:4460a38d-ab9b-484e-bd4e-f4201d0...	FTL	trip-153786558437756691	IND342902A1B	NaN	IND302014AAA	
...	...	...	...	...	...	...	...	...	
144484	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003AAA	G (
144485	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003AAA	G (
144486	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003AAA	G (
144487	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003AAA	G (
144488	test	2018-10-03 09:06:06.690094	thanos::sroute:cbef3b6a-79ea-4d5e-a215-b558a70...	FTL	trip-153855756668984584	IND282002AAD	NaN	IND474003AAA	G (

293 rows × 24 columns

checking source_name column wheather we can fill null value.

In [732…

```
data[~(data['source_name'].isnull()) & data['source_id'] == "IND342902A1B" ]
```

Out[732]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_id	source_name	destination_id	destination_name	od_start_time

0 rows × 24 columns

checking destination_name column wheather we can fill null value.

In [733…

```
lis = data[data['destination_name'].isnull()]['destination_id'].to_list()
lis = set(lis)
```

In [734…

```
data[~ data['destination_name'].isnull() & data['destination_id'].isin(lis)]
```

Out[734]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_id	source_name	destination_id	destination_name	od_start_time

0 rows × 24 columns

Inference

- there are total 293 null values in source_name column and 261 in destination_name column. there are 3 rows with both nulls.
- we need to drop 551 rows.

In [735…

```
data.dropna(subset=['source_name','destination_name'],inplace=True)
data.shape
```

Out[735]:

```
(144316, 24)
```

In [736…

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 144316 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144316 non-null  object
 1   trip_creation_time              144316 non-null  datetime64[ns]
 2   route_schedule_uuid             144316 non-null  object
 3   route_type                      144316 non-null  object
 4   trip_uuid                       144316 non-null  object
 5   source_id                       144316 non-null  object
 6   source_name                     144316 non-null  object
 7   destination_id                  144316 non-null  object
 8   destination_name                144316 non-null  object
 9   od_start_time                   144316 non-null  datetime64[ns]
 10  od_end_time                     144316 non-null  datetime64[ns]
 11  total_trip_time                 144316 non-null  float64
 12  is_cutoff                       144316 non-null  bool
 13  cutoff_factor                   144316 non-null  int64
 14  cutoff_timestamp                144316 non-null  object
 15  actual_distance_to_destination  144316 non-null  float64
 16  actual_time                     144316 non-null  float64
 17  osrm_time                       144316 non-null  float64
 18  osrm_distance                   144316 non-null  float64
 19  factor                          144316 non-null  float64
 20  segment_actual_time             144316 non-null  float64
 21  segment_osrm_time               144316 non-null  float64
 22  segment_osrm_distance           144316 non-null  float64
 23  segment_factor                  144316 non-null  float64
dtypes: bool(1), datetime64[ns](3), float64(10), int64(1), object(9)
memory usage: 26.6+ MB
```

In [737…

```
data.columns.to_list()
```

Out[737]:

```
['data',
 'trip_creation_time',
 'route_schedule_uuid',
 'route_type',
 'trip_uuid',
 'source_id',
 'source_name',
 'destination_id',
 'destination_name',
 'od_start_time',
 'od_end_time',
 'total_trip_time',
 'is_cutoff',
 'cutoff_factor',
 'cutoff_timestamp',
 'actual_distance_to_destination',
 'actual_time',
 'osrm_time',
 'osrm_distance',
 'factor',
 'segment_actual_time',
 'segment_osrm_time',
 'segment_osrm_distance',
 'segment_factor']
```

Inference

To keep our analysis clear and focused, we should remove columns that have unclear or unknown meanings. this are some columns which are unknown.

- segment_factor – Unknown field
- factor – Unknown field
- is_cutoff – Unknown field
- cutoff_factor – Unknown field
- cutoff_timestamp – Unknown field

In [738...

```python
#dropping unkown fields
data1 = data.drop(columns=['segment_factor','factor','is_cutoff','cutoff_factor','cutoff_timestamp'])
```

In [739...

```python
data1.columns.to_list()
```

Out[739]:

```
['data',
 'trip_creation_time',
 'route_schedule_uuid',
 'route_type',
 'trip_uuid',
 'source_id',
 'source_name',
 'destination_id',
 'destination_name',
 'od_start_time',
 'od_end_time',
 'total_trip_time',
 'actual_distance_to_destination',
 'actual_time',
 'osrm_time',
 'osrm_distance',
 'segment_actual_time',
 'segment_osrm_time',
 'segment_osrm_distance']
```

In [740…

```
data1[(data1['trip_uuid'] == "trip-153786558437756691") & (data1['source_name'] =='Jaipur_Hub (Rajasthan)') & (data1['d
                                                       'od_start_time','od_end_time','total_trip_time',
                                                       'actual_distance_to_destination','actual_time',
                                                       'osrm_time','osrm_distance','segment_actual_time',
                                                       'segment_osrm_time','segment_osrm_distance']]
```

Out[740]:

	trip_uuid	source_name	destination_name	od_start_time	od_end_time	total_trip_time	actual_distance_to_destination	actua
91	trip-153786558437756691	Jaipur_Hub (Rajasthan)	Ajmer_FoySGRRD_I (Rajasthan)	2018-09-25 08:53:04.377810	2018-09-25 16:37:20.428705	464.0	22.564533	
92	trip-153786558437756691	Jaipur_Hub (Rajasthan)	Ajmer_FoySGRRD_I (Rajasthan)	2018-09-25 08:53:04.377810	2018-09-25 16:37:20.428705	464.0	45.388095	
93	trip-153786558437756691	Jaipur_Hub (Rajasthan)	Ajmer_FoySGRRD_I (Rajasthan)	2018-09-25 08:53:04.377810	2018-09-25 16:37:20.428705	464.0	78.204766	
94	trip-153786558437756691	Jaipur_Hub (Rajasthan)	Ajmer_FoySGRRD_I (Rajasthan)	2018-09-25 08:53:04.377810	2018-09-25 16:37:20.428705	464.0	88.010616	
95	trip-153786558437756691	Jaipur_Hub (Rajasthan)	Ajmer_FoySGRRD_I (Rajasthan)	2018-09-25 08:53:04.377810	2018-09-25 16:37:20.428705	464.0	115.305349	
96	trip-153786558437756691	Jaipur_Hub (Rajasthan)	Ajmer_FoySGRRD_I (Rajasthan)	2018-09-25 08:53:04.377810	2018-09-25 16:37:20.428705	464.0	125.698345	

In [741...

```
# Appling aggregation
agg_data = data1.groupby(['trip_uuid','source_id','destination_id']).agg({
  'data':'first',
 'trip_creation_time':'first',
 'route_schedule_uuid':'first',
 'route_type':'first',
 'source_name':'first',
 'destination_name':'first',
 'od_start_time':'first',
 'od_end_time':'first',
 'total_trip_time':'first',
 'actual_distance_to_destination':'last',
 'actual_time':'last',
 'osrm_time':'last',
 'osrm_distance':'last',
 'segment_actual_time':'sum',
 'segment_osrm_time':'sum',
 'segment_osrm_distance':'sum'
 }).reset_index()
```

In [742...

```
agg_data.head()
```

Out[742]:

	trip_uuid	source_id	destination_id	data	trip_creation_time	route_schedule_uuid	route_type	source_name
0	trip-153671041653548748	IND209304AAA	IND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	Kanpur_Central_H_6 (Uttar Pradesh)
1	trip-153671041653548748	IND462022AAA	IND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	Bhopal_Trnsport_H (Madhya Pradesh)
2	trip-153671042288605164	IND561203AAB	IND562101AAA	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	Doddablpur_ChikaDPP_D (Karnataka)
3	trip-153671042288605164	IND572101AAA	IND561203AAB	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	Tumkur_Veersagr_ (Karnataka)
4	trip-153671043369099517	IND000000ACB	IND160002AAC	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	Gurgaon_Bilaspur_HB (Haryana)

In [743…

```
agg_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26222 entries, 0 to 26221
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype         
---  ------                          --------------  -----         
 0   trip_uuid                       26222 non-null  object        
 1   source_id                       26222 non-null  object        
 2   destination_id                  26222 non-null  object        
 3   data                            26222 non-null  object        
 4   trip_creation_time              26222 non-null  datetime64[ns]
 5   route_schedule_uuid             26222 non-null  object        
 6   route_type                      26222 non-null  object        
 7   source_name                     26222 non-null  object        
 8   destination_name                26222 non-null  object        
 9   od_start_time                   26222 non-null  datetime64[ns]
 10  od_end_time                     26222 non-null  datetime64[ns]
 11  total_trip_time                 26222 non-null  float64       
 12  actual_distance_to_destination  26222 non-null  float64       
 13  actual_time                     26222 non-null  float64       
 14  osrm_time                       26222 non-null  float64       
 15  osrm_distance                   26222 non-null  float64       
 16  segment_actual_time             26222 non-null  float64       
 17  segment_osrm_time               26222 non-null  float64       
 18  segment_osrm_distance           26222 non-null  float64       
dtypes: datetime64[ns](3), float64(8), object(8)
memory usage: 3.8+ MB
```

Inference

- The dataset was aggregated based on specific columns, including **'trip_uuid', 'source_id', and 'destination_id'.**
- Aggregation functions such as 'sum' and 'first' and 'last were applied to relevant columns to consolidate information and summarize data for each group.
- The aggregation process condensed the dataset, resulting dataset consist
  - **rows --> 26222**
  - **columns --> 19**

Feature Extraction

Extracting city,state,place from source and destination column.

In [744…

```
# checking source_name data format
agg_data[~agg_data['source_name'].str.contains('_')]['source_name']
```

Out[744]:

```
6                Mumbai Hub (Maharashtra)
9                      Hospet (Karnataka)
13              HBR Layout PC (Karnataka)
71               Mumbai Hub (Maharashtra)
81                       Palwal (Haryana)
                      ...
26132    Mumbai Antop Hill (Maharashtra)
26137           HBR Layout PC (Karnataka)
26181                  Darbhanga (Bihar)
26209            Mumbai Hub (Maharashtra)
26221                  Hospet (Karnataka)
Name: source_name, Length: 823, dtype: object
```

There are some value which don't follow underscore format

In [745…

```
# spliting source_name to city,place,code,state
# for source_name with '_'
def with_(s):
  lis = s.split('(')
  lis2 = lis[0].split('_')
  city=lis2[0]
  place=lis2[1]
  state=lis[1][:-1]
  return city,place,state

# for source_name without '_'
def without(s):
  lis = s.split('(')
  lis2 = lis[0].split(' ')
  city=lis2[0]
  place=lis2[1]
  state=lis[1][:-1]
  return city,place,state

agg_data[['source_city','source_place','source_state']] = tuple(agg_data['source_name'].apply(lambda x: with_(x) if (Tr
agg_data.drop(columns=['source_name'],inplace=True)
```

In [746…

```
agg_data.head()
```

Out[746]:

	trip_uuid	source_id	destination_id	data	trip_creation_time	route_schedule_uuid	route_type	destination_nam
0	trip-153671041653548748	IND209304AAA	IND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	Gurgaon_Bilaspur_H (Haryana
1	trip-153671041653548748	IND462022AAA	IND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	Kanpur_Central_H_ (Uttar Pradesl
2	trip-153671042288605164	IND561203AAB	IND562101AAA	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	Chikblapur_ShntiSgr_ (Karnataka
3	trip-153671042288605164	IND572101AAA	IND561203AAB	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	Doddablpur_ChikaDPP_ (Karnataka
4	trip-153671043369099517	IND000000ACB	IND160002AAC	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	Chandigarh_Mehmdpur_ (Punjal

5 rows × 21 columns

In [747…

```
# checking wheather destination_name data follow underscore format.
agg_data[~agg_data['destination_name'].str.contains('_')]['destination_name']
```

Out[747]:

```
7                     Hospet (Karnataka)
14              HBR Layout PC (Karnataka)
18         PNQ Rahatani DPC (Maharashtra)
19                   Faridabad (Haryana)
33                     Janakpuri (Delhi)
                            ...
26082                  Bidar (Karnataka)
26090            Nalasopara (Maharashtra)
26138           HBR Layout PC (Karnataka)
26153           Jabalpur (Madhya Pradesh)
26180                  Darbhanga (Bihar)
Name: destination_name, Length: 984, dtype: object
```

destination_name column data is same as source*name column data it also contain some value with* '' and some are just seperated by space.

In [748…

```
# spliting source_name to city,place,code,state
# for source_name with '_'
agg_data[['destination_city','destination_place','destination_state']] = tuple(agg_data['destination_name'].apply(lambd
agg_data.drop(columns=['destination_name'],inplace=True)
```

In [749…

```
agg_data.head()
```

Out[749]:

	trip_uuid	source_id	destination_id	data	trip_creation_time	route_schedule_uuid	route_type	od_start_time	od_
0	trip-153671041653548748	IND209304AAA	IND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	2018-09-12 16:39:46.858469	20 13:40:2
1	trip-153671041653548748	IND462022AAA	IND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	2018-09-12 00:00:16.535741	20 16:39:4
2	trip-153671042288605164	IND561203AAB	IND562101AAA	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	2018-09-12 02:03:09.655591	20 03:01:5
3	trip-153671042288605164	IND572101AAA	IND561203AAB	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	2018-09-12 00:00:22.886430	20 02:03:0
4	trip-153671043369099517	IND000000ACB	IND160002AAC	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	2018-09-14 03:40:17.106733	20 17:34:5

5 rows × 23 columns

Inference: Data Preprocessing Decisions for Source and Destination Names

- During the data preprocessing phase, it was observed that some entries in the 'source_name' and 'destination_name' columns did not follow the standard format of 'city_place_code (state*name)'. These entries lacked the underscore character (*'') which typically separates the city, place, and code information.
- Furthermore, certain rows in the 'destination_name' column were missing the place information, containing only the city and state details. To maintain consistency and facilitate accurate analysis, it was decided to represent these missing place values with blanks ('').

Processing: Extracting month, year, day from Trip_creation_time.

In [750…

```
# extracting day,month,year and .
agg_data['trip_day'] = agg_data['trip_creation_time'].dt.day
agg_data['trip_month'] = agg_data['trip_creation_time'].dt.month
agg_data['trip_year'] = agg_data['trip_creation_time'].dt.year
# Extract hour from 'trip_creation_time'
agg_data['hour'] = agg_data['trip_creation_time'].dt.hour

# Create categorical column for time intervals
bins = [-1, 6, 12, 18, 23]
labels = ['Night', 'Morning', 'Afternoon', 'Evening']
agg_data['trip_time_category'] = pd.cut(agg_data['hour'], bins=bins, labels=labels)
```

In [751…

```
#dropping hour and trip_creation_time column
agg_data.drop(columns=['hour','trip_creation_time'],inplace=True)
```

In [752…

```
agg_data.head()
```

Out[752]:

	trip_uuid	source_id	destination_id	data	route_schedule_uuid	route_type	od_start_time	od_end_time	total_trip
0	trip-153671041653548748	IND209304AAA	IND000000ACB	training	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	2018-09-12 16:39:46.858469	2018-09-13 13:40:23.123744	
1	trip-153671041653548748	IND462022AAA	IND209304AAA	training	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	2018-09-12 00:00:16.535741	2018-09-12 16:39:46.858469	
2	trip-153671042288605164	IND561203AAB	IND562101AAA	training	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	2018-09-12 02:03:09.655591	2018-09-12 03:01:59.598855	
3	trip-153671042288605164	IND572101AAA	IND561203AAB	training	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	2018-09-12 00:00:22.886430	2018-09-12 02:03:09.655591	
4	trip-153671043369099517	IND000000ACB	IND160002AAC	training	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	2018-09-14 03:40:17.106733	2018-09-14 17:34:55.442454	

5 rows × 26 columns

Inference: Feature Extraction and Categorization

- Day, Month, and Year Extraction:
    - 'day', 'month', and 'year' columns were created to represent the day of the month, the month of the year, and the year, of **'trip_creation_time'** column.
    - This feature extraction enables a deeper understanding of data in monthly,yearly, day wise.
- Time Categorization:
    - The **'trip_creation_time'** column was further analyzed to categorize time intervals into distinct categories. Time intervals were defined, dividing the day into four categories: **'Morning', 'Afternoon', 'Evening', and 'Night'.**
    - Using these intervals, a new categorical column named **'time_category'** was created to represent the time of day during which each trip was created.

- Such categorization is userful for understanding time-dependent patterns.

Processing: Creating total time taken column

In [753…

```
agg_data['od_start_time'] = pd.to_datetime(agg_data['od_start_time'].dt.strftime("%y:%m:%d %H:%M:%S"))
agg_data['od_end_time'] = pd.to_datetime(agg_data['od_end_time'].dt.strftime("%y:%m:%d %H:%M:%S"))
```

In [754…

```
agg_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26222 entries, 0 to 26221
Data columns (total 26 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   trip_uuid                       26222 non-null  object
 1   source_id                       26222 non-null  object
 2   destination_id                  26222 non-null  object
 3   data                            26222 non-null  object
 4   route_schedule_uuid             26222 non-null  object
 5   route_type                      26222 non-null  object
 6   od_start_time                   26222 non-null  datetime64[ns]
 7   od_end_time                     26222 non-null  datetime64[ns]
 8   total_trip_time                 26222 non-null  float64
 9   actual_distance_to_destination  26222 non-null  float64
 10  actual_time                     26222 non-null  float64
 11  osrm_time                       26222 non-null  float64
 12  osrm_distance                   26222 non-null  float64
 13  segment_actual_time             26222 non-null  float64
 14  segment_osrm_time               26222 non-null  float64
 15  segment_osrm_distance           26222 non-null  float64
 16  source_city                     26222 non-null  object
 17  source_place                    26222 non-null  object
 18  source_state                    26222 non-null  object
 19  destination_city                26222 non-null  object
 20  destination_place               26222 non-null  object
 21  destination_state               26222 non-null  object
 22  trip_day                        26222 non-null  int64
 23  trip_month                      26222 non-null  int64
 24  trip_year                       26222 non-null  int64
 25  trip_time_category              26222 non-null  category
dtypes: category(1), datetime64[ns](2), float64(8), int64(3), object(12)
memory usage: 5.0+ MB
```

In [755…

```
agg_data['total_trip_od_time'] = agg_data['od_end_time']-agg_data['od_start_time']
```

In [756…

```
agg_data.head()
```

Out[756]:

	trip_uuid	source_id	destination_id	data	route_schedule_uuid	route_type	od_start_time	od_end_time	total_trip_tim
0	trip-153671041653548748	IND209304AAA	IND000000ACB	training	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	2024-02-14 16:39:46	2024-02-14 13:40:23	1260
1	trip-153671041653548748	IND462022AAA	IND209304AAA	training	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	2024-02-14 00:00:16	2024-02-14 16:39:46	999
2	trip-153671042288605164	IND561203AAB	IND562101AAA	training	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	2024-02-14 02:03:09	2024-02-14 03:01:59	58
3	trip-153671042288605164	IND572101AAA	IND561203AAB	training	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	2024-02-14 00:00:22	2024-02-14 02:03:09	122
4	trip-153671043369099517	IND000000ACB	IND160002AAC	training	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	2024-02-14 03:40:17	2024-02-14 17:34:55	834

5 rows × 27 columns

In [757…

```
agg_data[agg_data['total_trip_od_time'] < pd.to_timedelta(" 0 days")][['trip_uuid','source_id','source_city','od_start_
```

Out[757]:

	trip_uuid	source_id	source_city	od_start_time	destination_id	destination_city	destination_state	od_end_time	total_t
0	trip-153671041653548748	IND209304AAA	Kanpur	2024-02-14 16:39:46	IND000000ACB	Gurgaon	Haryana	2024-02-14 13:40:23	
128	trip-153671547254076660	IND507002AAA	Khammam	2024-02-14 20:40:11	IND508213AAB	Suryapet	Telangana	2024-02-14 00:33:53	
169	trip-153671723500134877	IND209304AAA	Kanpur	2024-02-14 19:05:32	IND211002AAB	Allahabad	Uttar Pradesh	2024-02-14 03:46:17	
172	trip-153671726684858447	IND263153AAB	Rudrapur	2024-02-14 18:25:01	IND131028AAB	Sonipat	Haryana	2024-02-14 04:33:32	
178	trip-153671742249756615	IND000000ACB	Gurgaon	2024-02-14 01:57:02	IND821115AAB	Sasaram	Bihar	2024-02-14 01:07:23	
...	...	...	...	...	...	...	...	...	
26209	trip-153861091843037040	IND400072AAB	Mumbai	2024-02-14 23:55:18	IND401104AAA	Mumbai	Maharashtra	2024-02-14 01:23:31	
26210	trip-153861095625827784	IND160002AAC	Chandigarh	2024-02-14 23:55:56	IND140603AAA	Zirakpur	Punjab	2024-02-14 02:28:43	
26212	trip-153861104386292051	IND121004AAB	FBD	2024-02-14 23:57:23	IND121004AAA	Faridabad	Haryana	2024-02-14 00:57:59	
26214	trip-153861106442901555	IND209304AAA	Kanpur	2024-02-14 23:57:44	IND208006AAA	Kanpur	Uttar Pradesh	2024-02-14 02:51:27	
26215	trip-153861115439069069	IND627005AAA	Tirunelveli	2024-02-14 23:59:14	IND628801AAA	Eral	Tamil Nadu	2024-02-14 01:44:53	

5017 rows × 9 columns

In [758…

```
agg_data[agg_data['trip_uuid'] == "trip-153861115439069069"][['trip_uuid','source_id','source_city','source_state','od_
```

Out[758]:

	trip_uuid	source_id	source_city	source_state	od_start_time	destination_id	destination_city	destination_state	od_er
26215	trip-153861115439069069	IND627005AAA	Tirunelveli	Tamil Nadu	2024-02-14 23:59:14	IND628801AAA	Eral	Tamil Nadu	202 (
26216	trip-153861115439069069	IND627657AAA	Thisayanvilai	Tamil Nadu	2024-02-14 03:31:11	IND628613AAA	Peikulam	Tamil Nadu	202 (
26217	trip-153861115439069069	IND628204AAA	Tirchchndr	Tamil Nadu	2024-02-14 02:29:04	IND627657AAA	Thisayanvilai	Tamil Nadu	202 (
26218	trip-153861115439069069	IND628613AAA	Peikulam	Tamil Nadu	2024-02-14 04:16:39	IND627005AAA	Tirunelveli	Tamil Nadu	202 (
26219	trip-153861115439069069	IND628801AAA	Eral	Tamil Nadu	2024-02-14 01:44:53	IND628204AAA	Tirchchndr	Tamil Nadu	202 (

Inference

During the creation of the **'total_trip_time'** column by subtracting **'od_start_time'** from **'od_end_time'**, it was observed that certain rows resulted in negative values. This discrepancy arises when the **'od_end_time'** precedes the **'od_start_time'**, leading to incorrect interpretations of the trip duration.

To rectify this issue and ensure accurate representation of trip durations, an alternative approach was adopted. The absolute time difference between 'od_start_time' and 'od_end_time' was calculated, disregarding the directionality of the time interval. This methodology guarantees non-negative values, accurately reflecting the duration of each trip.

In [759…

```
agg_data['total_trip_od_time'] = abs(agg_data['od_end_time']-agg_data['od_start_time'])
```

In [760…

```
# checking for negative values in total_trip_time column
agg_data[agg_data['total_trip_od_time'] < pd.to_timedelta(" 0 days")]
```

Out[760]:

trip_uuid	source_id	destination_id	data	route_schedule_uuid	route_type	od_start_time	od_end_time	total_trip_time	actual_distance_to_de

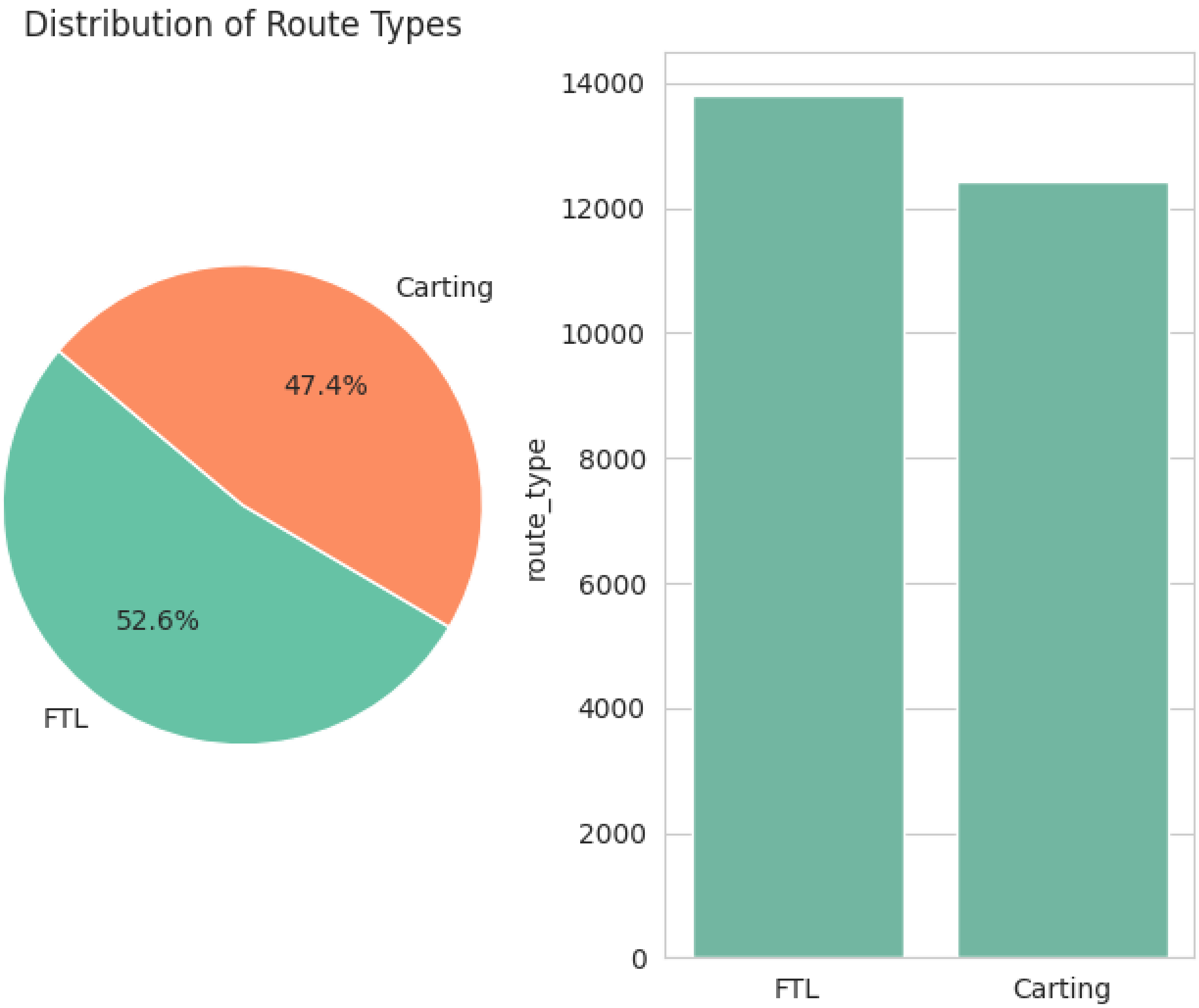
0 rows × 27 columns

Finding insights

Analysis: Comparing percentage of different route type trips.

In [761…

```
route_type_count = agg_data['route_type'].value_counts()
plt.figure(figsize=(8, 6))
plt.subplot(1,2,1)
sns.set_style("whitegrid")  # Set the style of the plot
sns.set_palette("Set2")     # Set the color palette
plt.pie(route_type_count, labels=route_type_count.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Route Types')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle

plt.subplot(1,2,2)
sns.barplot(route_type_count)
plt.subplots_adjust(wspace=0.3)
plt.show()
```



Inference

The pie and bar graphs show that about 51% of orders are catering, while 49% are FTL. This means catering orders are slightly more common. It suggests there's a higher demand for smaller shipments handled by catering services, like carts. FTL orders, on the other hand, are for larger shipments needing dedicated transportation.

Analysis: Indentifying states with the higest number trips

In [762...

```
agg_data.groupby('destination_state')['trip_uuid'].nunique().sort_values(ascending=False)
```

```
Out[762]: destination_state
          Maharashtra               2637
          Karnataka                 2425
          Haryana                   1800
          Tamil Nadu                1097
          Uttar Pradesh              882
          Telangana                  856
          Gujarat                    791
          West Bengal                713
          Punjab                     693
          Delhi                      674
          Rajasthan                  574
          Andhra Pradesh             516
          Madhya Pradesh             423
          Bihar                      384
          Kerala                     303
          Assam                      249
          Jharkhand                  197
          Orissa                     187
          Uttarakhand                159
          Himachal Pradesh           101
          Chandigarh                  91
          Goa                         74
          Chhattisgarh                43
          Arunachal Pradesh           42
          Jammu & Kashmir             25
          Pondicherry                 24
          Dadra and Nagar Haveli      17
          Meghalaya                   11
          Mizoram                      7
          Daman & Diu                  1
          Nagaland                     1
          Tripura                      1
          Name: trip_uuid, dtype: int64
```

Inference

By grouping the data by destination state and counting the number of unique trip_uuids, we identified the states with the highest and lowest trip volumes.

According to the sorted results:

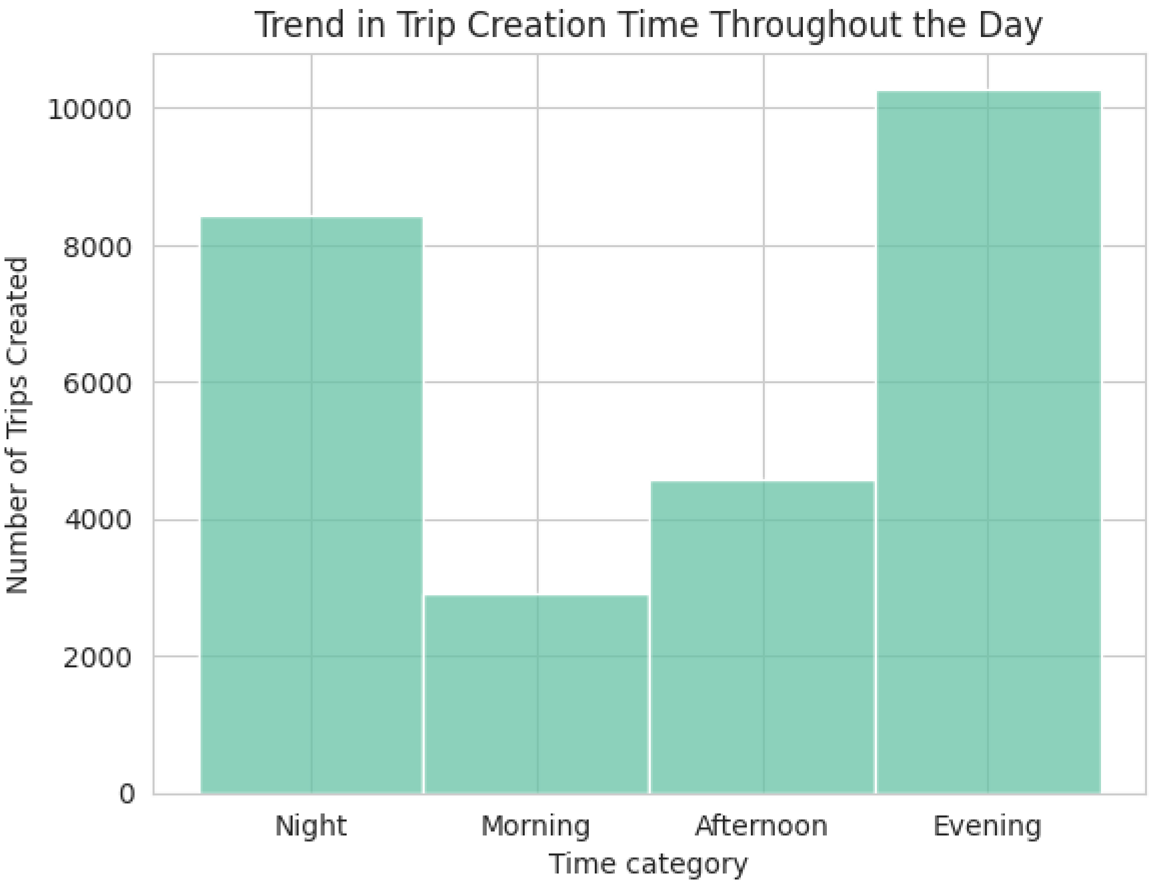
- Maharashtra had the highest number of unique trip_uuids, indicating a high volume of deliveries to this state.
- Maharashtra and Karnataka followed closely behind, suggesting significant delivery activity in these states as well.

- **(Himachal Pradesh,Chandigarh,Goa,Chhattisgarh,Arunachal Pradesh,Jammu & Kashmir,Pondicherry,Dadra and Nagar Haveli,Meghalaya, Mizoram,Nagaland,Daman & Diu,Tripura)** had the lowest number of unique trip_uuids, indicating relatively lower delivery volume compared to other states.

Analysis: Indentifying trend in trip_creation_time

In [763…

```
# Plotting
# plt.figure(figsize=(10, 6))
sns.histplot(agg_data['trip_time_category'])
plt.xlabel('Time category')
plt.ylabel('Number of Trips Created')
plt.title('Trend in Trip Creation Time Throughout the Day')
# plt.xticks(range(24))  # Set x-axis ticks to show all 24 hours
# plt.grid(True)
plt.show()
```



Inference

- Trip creation exhibits a notable trend with the highest activity observed during evening and night hours.
- Evening and night-time periods experience a surge in trip planning and logistics activities.
- Possible reasons for this trend include preparations for next-day deliveries, optimization of nighttime transportation routes, and customer preferences for evening deliveries.
- Reduced traffic congestion during these hours may contribute to more efficient trip planning and execution.

Analysis: Identifying Top 10 Cities for Delhivery Services in Maharashtra, Karnataka, Haryana, and Tamil Nadu

In [764…

```
states = ['Maharashtra','Karnataka','Haryana','Tamil Nadu']
agg_data[(agg_data['source_state'].isin(states)) & (agg_data['destination_state'].isin(states))].groupby(['source_city'
```

Out[764]:

```
source_city  destination_city
Mumbai       Mumbai              588
Bengaluru    Bengaluru           528
Bhiwandi     Mumbai              512
Bangalore    Bengaluru           492
Bengaluru    Bangalore           356
Mumbai       Bhiwandi            345
Chennai      Chennai             205
MAA          Chennai             204
Chennai      MAA                 141
Pune         PNQ                 122
Name: trip_uuid, dtype: int64
```

Inference

- **Maharashtra (Mumbai):**
  - Mumbai is the top city for Delhivery services in state Maharashtra.
  - It shows a lot of delivery activity, indicating it's a major hub for sending and receiving packages.
- **Karnataka (Bengaluru):**
  - Bengaluru is the top city for Delhivery in state Karnataka.
  - It's a big deal for deliveries, likely because it's a big city with lots of businesses and people.

Analysis: Finding average distance of trip

In [765…

```python
average_distance = round(agg_data['actual_distance_to_destination'].mean(),2)
print("Average distance to destion is :",average_distance,"KM")
```

```
Average distance to destion is : 92.53 KM
```

Inference

- Average trips source to destination distance is close to **93 KM.**

Analysis: Finding average time taken by trip

In [766…

```python
print("Average time of trip :",agg_data['total_trip_od_time'].mean())
print("Minimum time of trip :",agg_data['total_trip_od_time'].max())
print("Maximum time of trip :",agg_data['total_trip_od_time'].max())
```

```
Average time of trip : 0 days 05:55:50.411677217
Minimum time of trip : 0 days 23:29:45
Maximum time of trip : 0 days 23:29:45
```

Inference

- Average time taken by a trip is close to **6 hours**

Processing: Handling categorical column

In [767…

```
agg_data['route_type'].unique()
```

Out[767]:

```
array(['FTL', 'Carting'], dtype=object)
```

In [768…

```
agg_data['route_type_encoded'] = agg_data['route_type'].apply(lambda x: 1 if x == 'FTL' else 0)
```

In [769…

```
agg_data.route_type.unique()
```

Out[769]:

```
array(['FTL', 'Carting'], dtype=object)
```

Inference

- Using One-Hot Encoding:
  - We changed the **'route_type'** column to numbers, with 'FTL' as 1 and 'Carting' as 0.

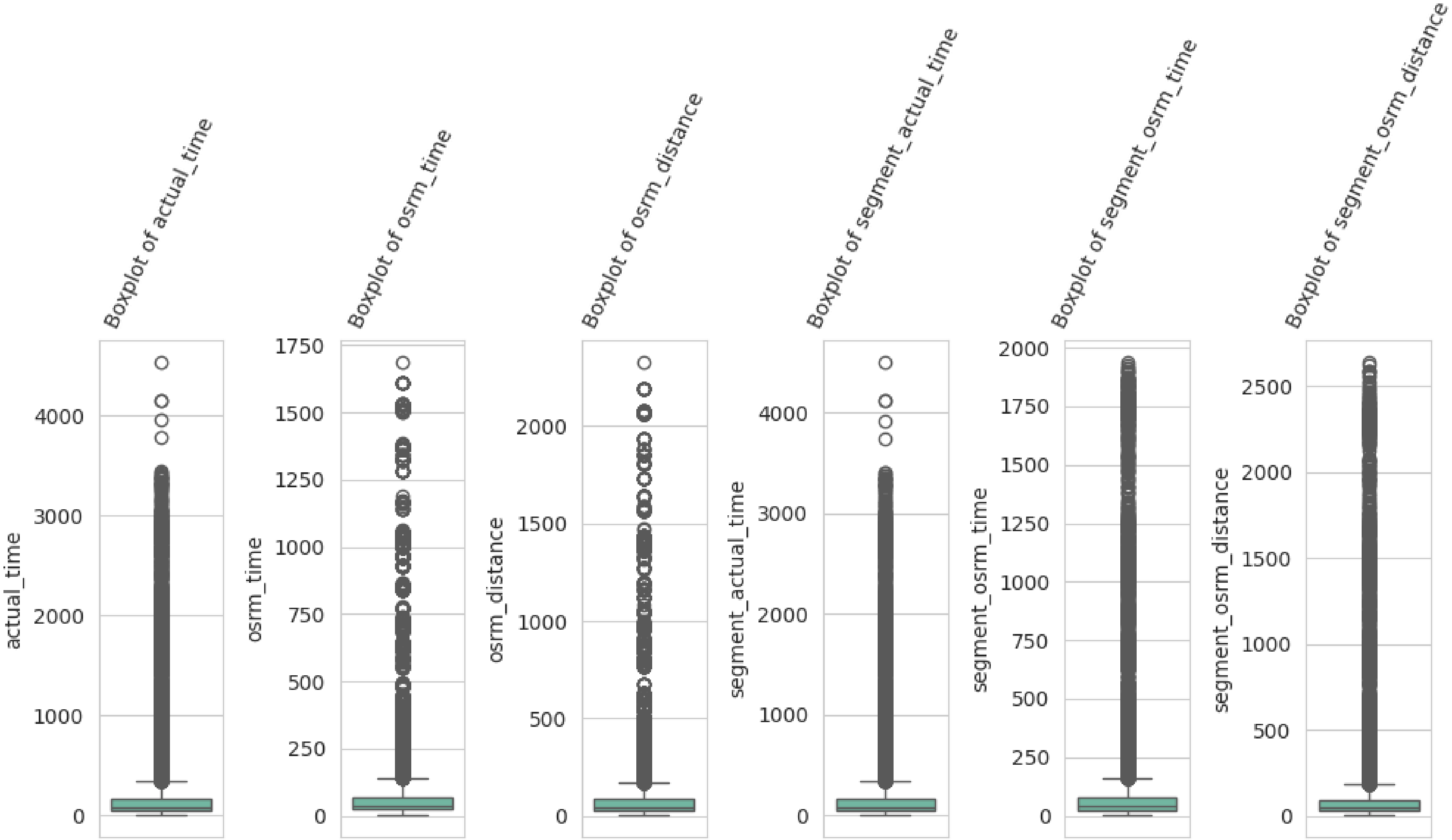
Outlier Detection

List of numerical columns in which we need to check for outliners

- actual_time aggregated
- OSRM time aggregated value
- segment_actual_time
- osrm distance aggregated
- segment osrm distance aggregated value
- segment osrm time aggregated value

In [770…

```
# # Visualize outliers using box plots
numerical_columns = ['actual_time','osrm_time','osrm_distance','segment_actual_time','segment_osrm_time','segment_osrm_
plt.figure(figsize=(10, 6))
for i in range(len(numerical_columns)):
    plt.subplot(1, len(numerical_columns), i+1)
    sns.boxplot(y=agg_data[numerical_columns[i]])
    plt.title(f'Boxplot of {numerical_columns[i]}',rotation=65,fontsize=10)
plt.tight_layout()
plt.show()
```



Inference

As we can see that all numerical columns consist of outliners and plot also showing skewness in data. we can take log to make the data normally distibuted.

In [771…

```
## making copy of data
agg_data_1 = agg_data
```

Processing: Replacing outliers using IQR method.

In [772…

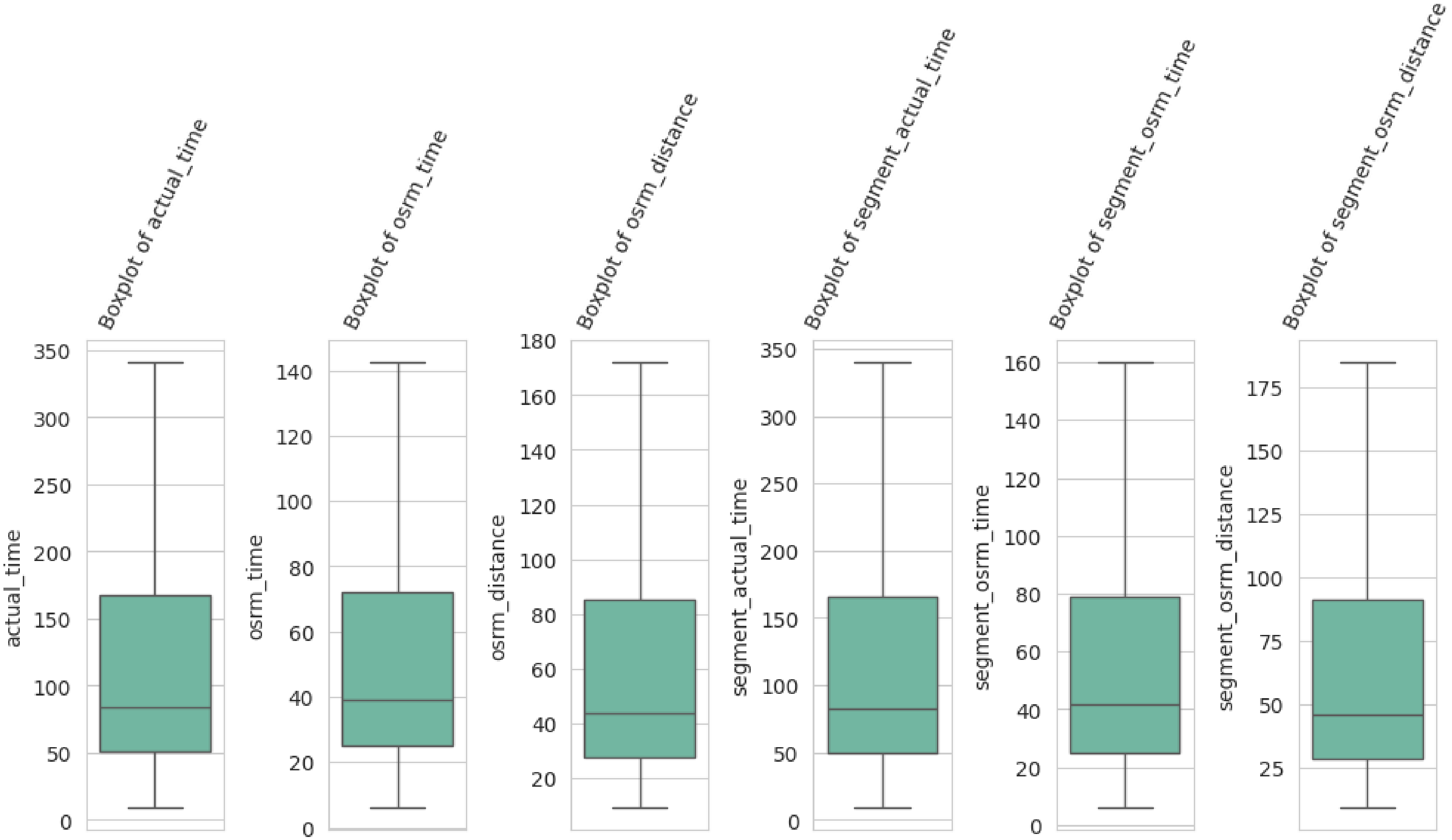
```
## copy of agg_data
agg_data_1 = agg_data
```

In [773…

```
for i in numerical_columns:
    q1 = agg_data_1[i].quantile(0.25)
    q3 = agg_data_1[i].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    # Replace outliers with the nearest non-outlier value
    agg_data_1[i] = agg_data_1[i].apply(lambda x: lower_bound if x < lower_bound else (upper_bound if x > upper_bound e
```

In [774…

```
## visualizing outliers
plt.figure(figsize=(10, 6))
for i in range(len(numerical_columns)):
    plt.subplot(1, len(numerical_columns), i+1)
    sns.boxplot(y=agg_data_1[numerical_columns[i]])
    plt.title(f'Boxplot of {numerical_columns[i]}',rotation=65,fontsize=10)
plt.tight_layout()
plt.show()
```



Inference

- Using IQR method outiler completely removed from data. still visual analysis is need to check the data normality.

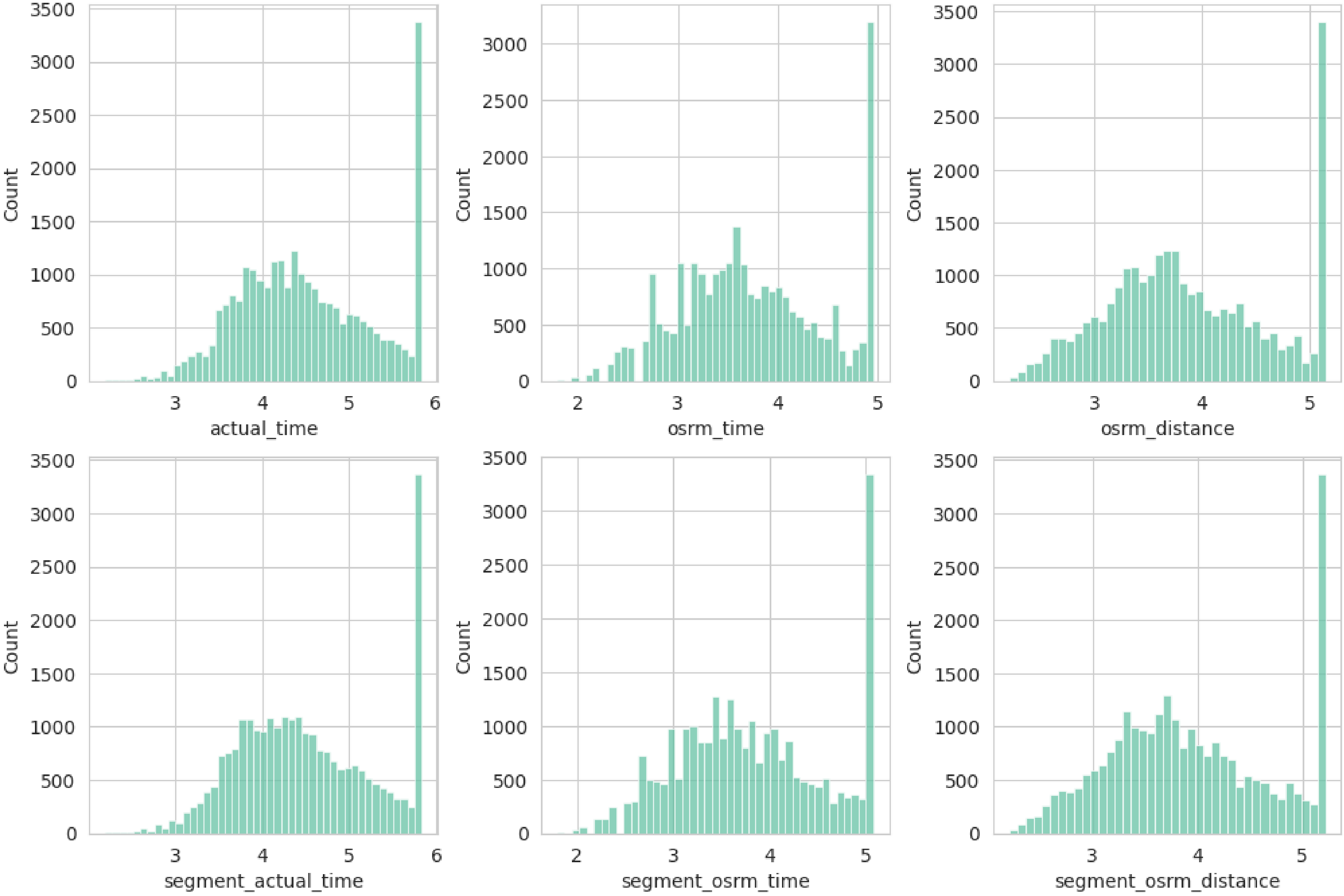
Processing: Trying to make data nomrally distributed using log tranformation

In [775…

```
# # aplying log transormation
for name in numerical_columns:
    agg_data_1[name] = np.log(agg_data_1[name])
```

In [776…

```
## checking the distribution of numerical columns
numerical_columns = ['actual_time','osrm_time','osrm_distance','segment_actual_time','segment_osrm_time','segment_osrm_
plt.figure(figsize =(12,8))
plt.subplot(2,3,1)
sns.histplot(agg_data[numerical_columns[0]])

plt.subplot(2,3,2)
sns.histplot(agg_data[numerical_columns[1]])

plt.subplot(2,3,3)
sns.histplot(agg_data[numerical_columns[2]])

plt.subplot(2,3,4)
sns.histplot(agg_data[numerical_columns[3]])

plt.subplot(2,3,5)
sns.histplot(agg_data[numerical_columns[4]])

plt.subplot(2,3,6)
sns.histplot(agg_data[numerical_columns[5]])
plt.subplots_adjust(wspace=0.3)
plt.show()
```



In [777…

```
## applying sphiro-wilk test to check for noramlity
for i in numerical_columns:
  stat,p_value = stats.shapiro(agg_data_1[i].sample(300))
  if p_value > 0.05:
    print(i,": data is normally distibuted")
  else:
    print(i,": data is not normally distibuted")
```

```
actual_time : data is not normally distibuted
osrm_time : data is not normally distibuted
osrm_distance : data is not normally distibuted
segment_actual_time : data is not normally distibuted
segment_osrm_time : data is not normally distibuted
segment_osrm_distance : data is not normally distibuted
```

Inference

- After applying the Interquartile Range (IQR) method to remove outliers for a specific data point, the count of values significantly increases, making it challenging to achieve a normal distribution.
- The substantial increase in the count of values suggests a substantial number of outliers were present in the dataset, impacting the data's distribution.
- Despite attempts to address outliers using the IQR method, the data remains skewed or exhibits non-normal behavior, posing challenges for normalization.
- In such cases, to ensure the integrity and reliability of subsequent analyses, it may be necessary to resort to dropping all rows containing outliers, allowing for a more robust and interpretable dataset.

In [778…

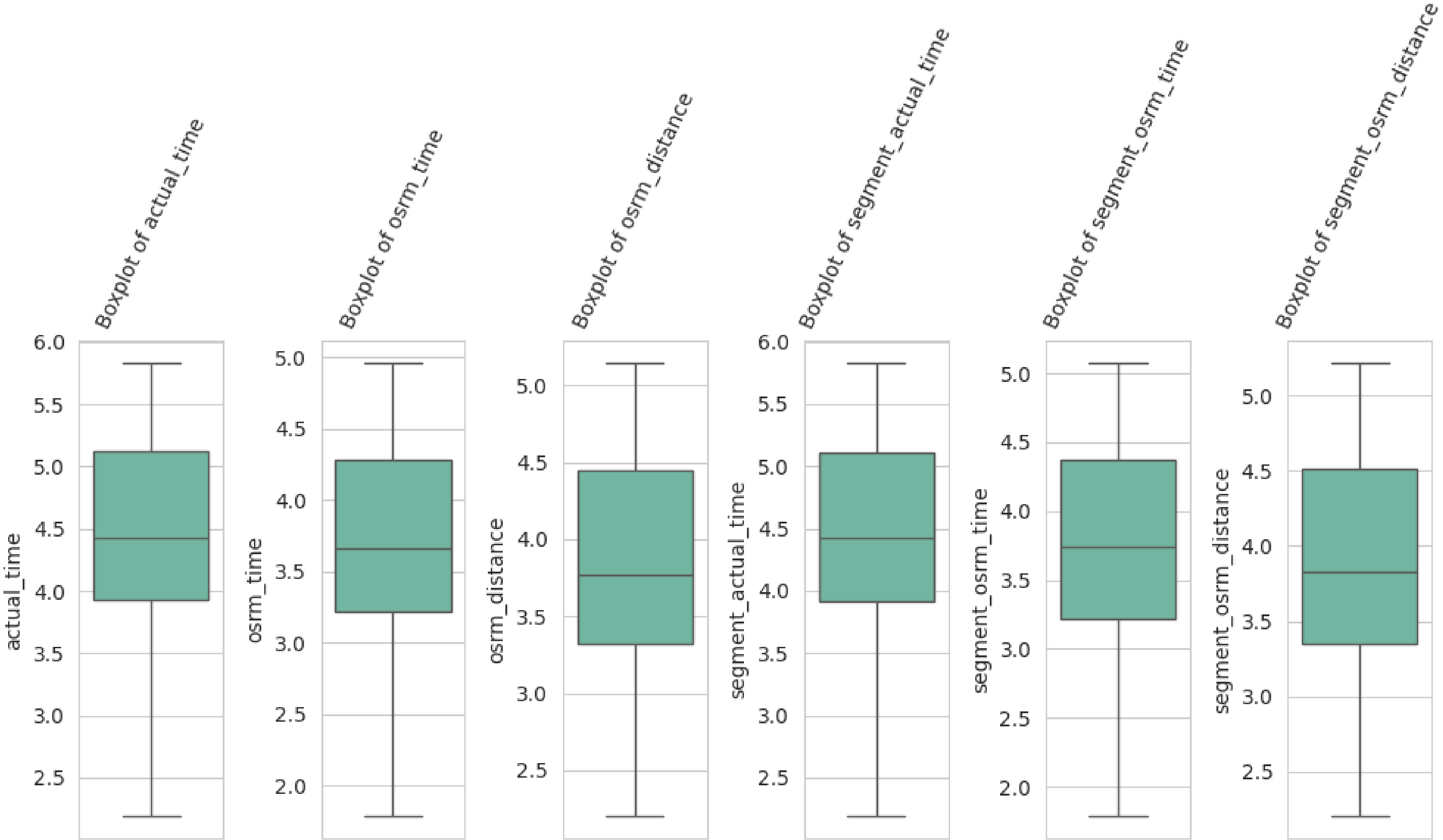
```python
### making a copy of data
agg_data_2 = agg_data
```

In [779…

```python
# dropping all outliner rows
for i in numerical_columns:
  q1 = agg_data_2[i].quantile(0.25)
  q3 = agg_data_2[i].quantile(0.75)
  iqr = q3 - q1
  lower_bound = q1 - 1.5 * iqr
  upper_bound = q3 + 1.5 * iqr
  outlier_indices = (agg_data_2[i] < lower_bound) | (agg_data_2[i] > upper_bound)
  # print(outlier_indices)
  agg_data_2.drop(agg_data_2.index[outlier_indices],inplace=True)
```

In [780…

```python
## checking for outliers
plt.figure(figsize=(10, 6))
for i in range(len(numerical_columns)):
    plt.subplot(1, len(numerical_columns), i+1)
    sns.boxplot(y=agg_data[numerical_columns[i]])
    plt.title(f'Boxplot of {numerical_columns[i]}',rotation=65,fontsize=10)
plt.tight_layout()
plt.show()
```



In [781...

```
### Applying log trangomation to outlier free data.
for name in numerical_columns:
    agg_data_2[name] = np.log(agg_data_2[name])
```

In [782...

```
## applying sphiro-wilk test to check for noramlity
for i in numerical_columns:
  stat,p_value = stats.shapiro(agg_data_2[i].sample(300))
  if p_value > 0.05:
    print(i,": data is normally distibuted")
  else:
    print(i,": data is not normally distibuted")
```

```
actual_time : data is not normally distibuted
osrm_time : data is not normally distibuted
osrm_distance : data is not normally distibuted
segment_actual_time : data is not normally distibuted
segment_osrm_time : data is not normally distibuted
segment_osrm_distance : data is not normally distibuted
```

Inference

Despite efforts to normalize the data through various methods, including log transformation, outlier removal, and other techniques, the data still did not conform to a normal distribution. Non-normality can introduce challenges when using traditional parametric tests that rely on the assumption of normality.

Given the persistent non-normality of the data, it was decided to proceed with alternative approaches that do not rely on the assumption of normality. Robust statistical tests, such as the Mann-Whitney U test, were chosen as alternatives to traditional tests like the t-test, which depend on normality assumptions.

Hypothesis testing

Comparison of Actual Time and OSRM Time

To perform this comparison we use a Mann-Whitney U test.

The Mann-Whitney U test is used to determine whether there is a statistically significant difference between the medians of two independent groups. It does not assume normality of the data and is robust to deviations from normality.

Null Hypothesis (H0): The median of actual_time is equal to the median of orsm_time.

Alternative Hypothesis (H1): The median of actual_time is not equal to the median of orsm_time.

In [783…

```python
# Perform Mann-Whitney U test
statistic, p_value = stats.mannwhitneyu(agg_data['actual_time'],agg_data['osrm_time'])

# Print the test statistic and p-value
print("Mann-Whitney U Test Statistic:", statistic)
print("P-value:", p_value)

# Set the significance level
alpha = 0.05
```

```python
if p_value > alpha:
    print("Failed to reject the null hypothesis (no significant difference)")
else:
    print("Reject the null hypothesis (significant difference)")
```

```
Mann-Whitney U Test Statistic: 521422996.0
P-value: 0.0
Reject the null hypothesis (significant difference)
```

Comparing actual_time aggregated value and segment actual time aggregated value.

Null Hypothesis (H0): The median of '*actual_time*' is equal to the median of '*segment_actual_time*'.

Alternative Hypothesis (H1): The median of '*actual_time*' is not equal to the median of *segment_actual_time*.

In [784…

```python
# Perform Mann-Whitney U test
statistic, p_value = stats.mannwhitneyu(agg_data['actual_time'],agg_data['segment_actual_time'])

# Print the test statistic and p-value
print("Mann-Whitney U Test Statistic:", statistic)
print("P-value:", p_value)

# Set the significance level
alpha = 0.05
if p_value > alpha:
    print("Failed to reject the null hypothesis (no significant difference)")
else:
    print("Reject the null hypothesis (significant difference)")
```

```
Mann-Whitney U Test Statistic: 351202639.5
P-value: 1.9263138185709147e-05
Reject the null hypothesis (significant difference)
```

Comparing OSRM distance aggregated value and segment OSRM distance aggregated value.

Null Hypothesis (H0): The median of '*osrm_distance*' is equal to the median of '*segment_osrm_distance*'.

Alternative Hypothesis (H1): The median of '*osrm_distance*' is not equal to the median of '*segment_orsm_distance*'.

In [785…

```python
# Perform Mann-Whitney U test
statistic, p_value = stats.mannwhitneyu(agg_data['osrm_distance'],agg_data['segment_osrm_distance'])

# Print the test statistic and p-value
print("Mann-Whitney U Test Statistic:", statistic)
print("P-value:", p_value)

# Set the significance level
alpha = 0.05
if p_value > alpha:
    print("Failed to reject the null hypothesis (no significant difference)")
else:
    print("Reject the null hypothesis (significant difference)")
```

```
Mann-Whitney U Test Statistic: 327935102.0
P-value: 5.599075722834906e-20
Reject the null hypothesis (significant difference)
```

Comparing OSRM time aggregated value and segment OSRM time aggregated value.

Null Hypothesis (H0): The median of '*OSRM_time*' is equal to the median of '*segment_osrm_time*'.

Alternative Hypothesis (H1): The median of '*osrm_time*' is not equal to the median of '*segment_orsm_time*'.

In [786…

```
# Perform Mann-Whitney U test
statistic, p_value = stats.mannwhitneyu(agg_data['osrm_time'],agg_data['segment_osrm_distance'])

# Print the test statistic and p-value
print("Mann-Whitney U Test Statistic:", statistic)
print("P-value:", p_value)

# Set the significance level
alpha = 0.05
if p_value > alpha:
    print("Failed to reject the null hypothesis (no significant difference)")
else:
    print("Reject the null hypothesis (significant difference)")
```

```
Mann-Whitney U Test Statistic: 297491700.5
P-value: 2.9932680690345243e-157
Reject the null hypothesis (significant difference)
```

Business Insights & Recommendations

Insights

- Catering orders comprise approximately **51%** of total orders, while FTL orders make up the remaining **49%**. This indicates a slightly higher demand for smaller shipments handled by catering services.
- The data suggests a significant volume of deliveries to states like **Maharashtra and Karnataka**, as they have the highest number of unique trip_uuids. Conversely, states like Himachal Pradesh, Chandigarh, and Goa show lower delivery volumes.
- **Evening and night-time** periods experience a surge in trip planning and logistics activities, possibly due to preparations for next-day deliveries, optimization of nighttime transportation routes, and customer preferences for evening deliveries.
- **Mumbai** emerges as the top city for Delhivery services in Maharashtra, indicating its importance as a major hub for sending and receiving packages. Similarly, **Bengaluru** leads in Delhivery services in Karnataka, reflecting its significance as a bustling city with high delivery demand.
- The average distance traveled from source to destination is approximately **93 kilometers**, highlighting the typical distance covered by deliveries.
- On average, trips take close to **6 hours** to complete, indicating the typical duration of delivery operations.
- The hypothesis testing conducted on actual time, distance, and segmented distance and time compared to estimated time, distance, and segmented distance and time resulted in rejecting the null hypothesis. This suggests that there are significant differences between the actual and estimated values, indicating potential areas for further investigation and optimization in delivery operations. This implies a need to improve the open-source routing engine to enhance accuracy and efficiency in route planning and delivery estimations.

Recommendation

- More customers prefer smaller shipments handled by catering services, like carts, than larger shipments needing dedicated transportation. **So, businesses can focus on catering services to cater to this higher demand.**

- Maharashtra, Karnataka have the highest delivery volumes, **so businesses can focus on expanding their delivery services in these states and cities to attract more customers.**

- Evening and night-time periods experience a surge in trip planning and logistics activities, **so businesses can optimize their delivery routes and schedules during these hours to improve efficiency and reduce delivery times.**

- The average distance of a trip is close to **93 KM**, and the average time taken by a trip is close to **6 hours**. This information can help businesses optimize their delivery routes and estimate delivery times more accurately.

- Businesses can invest in data analytics to study and analyze their delivery data to identify trends, patterns, and opportunities for improvement, and make data-driven business decisions.

- Business should switch to other better open source routing engine which give more accurate preditons about time and distance.