```
#Importing all the necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('delhivery_data.csv')
df.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_cent
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AA
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AA
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AA
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AA
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AA

5 rows × 24 columns

```
df.describe()
```

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	factor	se
count	19129.000000	19129.000000	19129.000000	19129.000000	19129.000000	19129.000000	19129.000000	
mean	869.031314	212.431282	213.533286	377.047101	196.268859	260.531285	2.073866	
std	962.423313	325.977085	326.180058	552.513550	292.917878	400.164692	1.369946	
min	25.000000	9.000000	9.000267	9.000000	6.000000	9.101900	0.250000	
25%	149.000000	22.000000	23.086633	50.000000	26.000000	29.089500	1.597285	
50%	402.000000	54.000000	55.362397	119.000000	59.000000	71.873800	1.852890	
75%	1352.000000	242.000000	242.666012	438.000000	220.000000	291.838300	2.206897	
max	3560.000000	1722.000000	1722.009755	3276.000000	1611.000000	2191.166400	77.387097	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19130 entries, 0 to 19129
Data columns (total 24 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   data                            19130 non-null  object
 1   trip_creation_time              19130 non-null  object
 2   route_schedule_uuid             19130 non-null  object
 3   route_type                      19130 non-null  object
 4   trip_uuid                       19130 non-null  object
 5   source_center                   19130 non-null  object
 6   source_name                     19069 non-null  object
 7   destination_center              19129 non-null  object
 8   destination_name                19087 non-null  object
 9   od_start_time                   19129 non-null  object
 10  od_end_time                     19129 non-null  object
 11  start_scan_to_end_scan          19129 non-null  float64
 12  is_cutoff                       19129 non-null  object
 13  cutoff_factor                   19129 non-null  float64
 14  cutoff_timestamp                19129 non-null  object
 15  actual_distance_to_destination  19129 non-null  float64
 16  actual_time                     19129 non-null  float64
```

```
 17  osrm_time                     19129 non-null  float64
 18  osrm_distance                 19129 non-null  float64
 19  factor                        19129 non-null  float64
 20  segment_actual_time           19129 non-null  float64
 21  segment_osrm_time             19129 non-null  float64
 22  segment_osrm_distance         19129 non-null  float64
 23  segment_factor                19129 non-null  float64
dtypes: float64(11), object(13)
memory usage: 3.5+ MB
```

```
df.isnull().sum().sort_values(ascending=False)[:10]
```

	0
source_name	61
destination_name	43
is_cutoff	1
cutoff_factor	1
segment_osrm_distance	1
segment_osrm_time	1
segment_actual_time	1
factor	1
osrm_distance	1
osrm_time	1

The dataset contains null values in 'source_name' and 'destination_name' features.

```
# Removing null values

df = df.dropna(how='any')
df = df.reset_index(drop=True)
```

```
# Converting the data type to datetime
df['od_start_time'] =  pd.to_datetime(df['od_start_time'])
df['od_end_time']  =  pd.to_datetime(df['od_end_time'])
```

Grouping by sub-journey in the trip

```
df['segment_key']=df['trip_uuid']+df['source_center']+df['destination_center']
df['segment_actual_time_sum']=df.groupby('segment_key')['segment_actual_time'].transform('cumsum')
df['segment_osrm_distance_sum']=df.groupby('segment_key')['segment_osrm_distance'].transform('cumsum')
df['segment_osrm_time_sum']=df.groupby('segment_key')['segment_osrm_time'].transform('cumsum')
```

Aggregating at sub-journey level

```
segment_dict = {

    'data' : 'first',
    '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' : 'first',
    'start_scan_to_end_scan' : 'first',


    'actual_distance_to_destination' : 'last',
```

```
    'actual_time' : 'last',

    'osrm_time' : 'last',
    'osrm_distance' : 'last',

    'segment_actual_time_sum' : 'last',
    'segment_osrm_distance_sum' : 'last',
    'segment_osrm_time_sum' : 'last',

    }
```

Grouping mini-trips, sorting by time

```
segment = df.groupby('segment_key').agg(segment_dict).reset_index()
segment = segment.sort_values(by=['segment_key','od_end_time'], ascending=True).reset_index()
segment
```

	index	segment_key	data	trip_creation_time	route_schedule_uuid	route_type	trip_
0	0	trip-153671191949943656IND487001AABIND487551AAA	training	2018-09-12 00:25:19.499696	thanos::sroute:0ac760f3-96cb-4046-bfd0-8bc4678...	FTL	15367119194994
1	1	trip-153671191949943656IND487551AAAIND464668AAA	training	2018-09-12 00:25:19.499696	thanos::sroute:0ac760f3-96cb-4046-bfd0-8bc4678...	FTL	15367119194994
2	2	trip-153671237597058150IND785690AABIND785682AAA	training	2018-09-12 00:32:55.970840	thanos::sroute:db0f8027-8ade-4411-9aff-b26adaa...	Carting	15367123759705
3	3	trip-153671262893947351IND500055AACIND501401AAC	training	2018-09-12 00:37:08.939733	thanos::sroute:beb73e7f-71ff-4501-bc17-191b4f3...	Carting	15367126289394
4	4	trip-153671262893947351IND501401AACIND500010AAA	training	2018-09-12 00:37:08.939733	thanos::sroute:beb73e7f-71ff-4501-bc17-191b4f3...	Carting	15367126289394
...	...	...	...	...	...	...	
3633	3633	trip-153861089872028474IND602024AAAIND602001AAA	test	2018-10-03 23:54:58.720536	thanos::sroute:27463ea7-5903-4530-92e7-6a4feca...	Carting	15386108987202
3634	3634	trip-153861106442901555IND208006AAAIND209304AAA	test	2018-10-03 23:57:44.429324	thanos::sroute:5609c268-e436-4e0a-8180-3db4a74...	Carting	15386110644290
3635	3635	trip-153861106442901555IND209304AAAIND208006AAA	test	2018-10-03 23:57:44.429324	thanos::sroute:5609c268-e436-4e0a-8180-3db4a74...	Carting	15386110644290
3636	3636	trip-153861118270144424IND583119AAAIND583101AAA	test	2018-10-03 23:59:42.701692	thanos::sroute:412fea14-6d1f-4222-8a5f-a517042...	FTL	15386111827014
3637	3637	trip-153861118270144424IND583201AAAIND583119AAA	test	2018-10-03 23:59:42.701692	thanos::sroute:412fea14-6d1f-4222-8a5f-a517042...	FTL	15386111827014

3638 rows × 21 columns

```
segment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3638 entries, 0 to 3637
Data columns (total 21 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   index                       3638 non-null   int64
 1   segment_key                 3638 non-null   object
 2   data                        3638 non-null   object
 3   trip_creation_time          3638 non-null   object
 4   route_schedule_uuid         3638 non-null   object
 5   route_type                  3638 non-null   object
 6   trip_uuid                   3638 non-null   object
 7   source_center               3638 non-null   object
 8   source_name                 3638 non-null   object
 9   destination_center          3638 non-null   object
```

```
 10  destination_name                3638 non-null   object
 11  od_start_time                   3638 non-null   datetime64[ns]
 12  od_end_time                     3638 non-null   datetime64[ns]
 13  start_scan_to_end_scan          3638 non-null   float64
 14  actual_distance_to_destination  3638 non-null   float64
 15  actual_time                     3638 non-null   float64
 16  osrm_time                       3638 non-null   float64
 17  osrm_distance                   3638 non-null   float64
 18  segment_actual_time_sum         3638 non-null   float64
 19  segment_osrm_distance_sum       3638 non-null   float64
 20  segment_osrm_time_sum           3638 non-null   float64
dtypes: datetime64[ns](2), float64(8), int64(1), object(10)
memory usage: 597.0+ KB
```

Calculating time taken between od_start_time and od_end_time

```
segment['od_time_diff_hour'] = (segment['od_end_time'] - segment['od_start_time']).dt.total_seconds() / (60)
segment['od_time_diff_hour']
```

	od_time_diff_hour
0	86.055592
1	204.606678
2	252.076999
3	59.704450
4	210.319679
...	...
3633	53.684203
3634	248.409092
3635	173.710775
3636	287.474007
3637	66.933565

3638 rows × 1 columns

```
trip_dict = {

    'data' : 'first',
    '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',

    'start_scan_to_end_scan' : 'sum',
    'od_time_diff_hour' : 'sum',

    'actual_distance_to_destination' : 'sum',
    'actual_time' : 'sum',
    'osrm_time' : 'sum',
    'osrm_distance' : 'sum',

    'segment_actual_time_sum' : 'sum',
    'segment_osrm_distance_sum' : 'sum',
    'segment_osrm_time_sum' : 'sum',

    }

trip = segment.groupby('trip_uuid').agg(trip_dict).reset_index(drop = True)
trip
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destinati
0	training	2018-09-12 00:25:19.499696	thanos::sroute:0ac760f3-96cb-4046-bfd0-8bc4678...	FTL	trip-153671191949943656	IND487001AAB	Narsinghpur_KndliDPP_D (Madhya Pradesh)	IND4
1	training	2018-09-12 00:32:55.970840	thanos::sroute:db0f8027-8ade-4411-9aff-b26adaa...	Carting	trip-153671237597058150	IND785690AAB	Sonari_Central_DPP_1 (Assam)	IND7
2	training	2018-09-12 00:37:08.939733	thanos::sroute:beb73e7f-71ff-4501-bc17-191b4f3...	Carting	trip-153671262893947351	IND500055AAC	Hyderabad_North_D_2 (Telangana)	IND5
3	training	2018-09-12 00:39:30.747127	thanos::sroute:42969f47-47af-4473-9f2c-cf747fe...	FTL	trip-153671277074687197	IND624001AAA	Dindigul_Central_D_1 (Tamil Nadu)	IND6
4	training	2018-09-12 00:46:48.079257	thanos::sroute:8c5ab716-198a-4395-b83f-5672773...	Carting	trip-153671320807895983	IND121004AAB	FBD_Balabhgarh_DPC (Haryana)	IND1
...	...	...	...	...	...	...	...	
2001	test	2018-10-03 23:33:29.015349	thanos::sroute:a02b2c7d-49c4-4c7f-956e-b4b22a1...	Carting	trip-153860960901509071	IND424006AAA	Dhule_MIDCAvdn_I (Maharashtra)	IND4
2002	test	2018-10-03 23:45:48.025062	thanos::sroute:1396edcd-faad-4029-a574-f71a85a...	Carting	trip-153861034802474617	IND245101AAA	Hapur_Swargash_D (Uttar Pradesh)	IND2
2003	test	2018-10-03 23:54:58.720536	thanos::sroute:27463ea7-5903-4530-92e7-6a4feca...	Carting	trip-153861089872028474	IND600116AAB	Chennai_Porur_DPC (Tamil Nadu)	IND6
2004	test	2018-10-03 23:57:44.429324	thanos::sroute:5609c268-e436-4e0a-8180-3db4a74...	Carting	trip-153861106442901555	IND208006AAA	Kanpur_GovndNgr_DC (Uttar Pradesh)	IND2
2005	test	2018-10-03 23:59:42.701692	thanos::sroute:412fea14-6d1f-4222-8a5f-a517042...	FTL	trip-153861118270144424	IND583119AAA	Sandur_WrdN1DPP_D (Karnataka)	IND5

2006 rows × 18 columns

Next steps: Generate code with trip | View recommended plots | New interactive sheet

```
trip[['actual_distance_to_destination','osrm_distance']]
```

	actual_distance_to_destination	osrm_distance
0	99.975595	124.5063
1	39.495954	46.9087
2	24.359086	30.4646
3	129.158279	197.6186
4	76.231506	79.9793
...	...	...
2001	49.732416	57.1276
2002	44.106290	46.5093
2003	27.010926	38.2867
2004	38.684839	58.9037
2005	66.081533	80.5787

2006 rows × 2 columns

Extracting City, Place, Code & State from Source and Destination names

```
def state(x):
  state=x.split('(')[1]
  return state[:-1]

def city(x):
```

```
    city=(x.split('(')[0]).split('_')[0]
    return city

  def place(x):
    x=x.split('(')[0]
    if len(x.split('_'))>=3:
      place=x.split('_')[1]
    elif len(x.split('_'))==2:
      place=x.split('_')[0]
    else:
      place=x.split(" ")[0]
    return place

  def code(x):
    x=x.split('(')[0]
    if len(x.split('_'))>=3:
      return x.split('_')[-1]
    return  "none"


  df['source_state']=df['source_name'].apply(state)
  df['source_city']=df['source_name'].apply(city)
  df['source_place']=df['source_name'].apply(place)
  df['source_code']=df['source_name'].apply(code)

  df['destination_state']=df['destination_name'].apply(state)
  df['destination_city']=df['destination_name'].apply(city)
  df['destination_place']=df['destination_name'].apply(place)
  df['destination_code']=df['destination_name'].apply(code)
```

State with Most Number of Orders

```
(df['source_state'].value_counts()).head()
```

source_state	count
Haryana	3897
Maharashtra	2654
Karnataka	2474
Tamil Nadu	1093
Telangana	898

```
(df['destination_state'].value_counts()).head()
```

destination_state	count
Maharashtra	2653
Karnataka	2611
Haryana	2575
Tamil Nadu	1136
Gujarat	1050

The source and destination state with most number of orders is Maharashtra

```
(df[(df['source_state']=="Maharashtra") & (df['destination_state']=="Maharashtra")]['source_city'].value_counts()).head()
```

source_city	count
Bhiwandi	336
Pune	299
Mumbai	182
Akola	109
Mumbai Hub	108

The source city with most number of orders is Bhiwandi.

```
(df[(df['source_state']=="Maharashtra") & (df['destination_state']=="Maharashtra") & (df['source_city']=="Bhiwandi")]["destination_city"].va
```

destination_city	count
Mumbai	142
Akola	63
Mumbai Hub	61
Pune	60
Nashik	6

The destination city with most number of orders is Mumbai.

```
trip['trip_creation_time'] =  pd.to_datetime(trip['trip_creation_time'])

trip['trip_year'] = trip['trip_creation_time'].dt.year
trip['trip_month'] = trip['trip_creation_time'].dt.month
trip['trip_hour'] = trip['trip_creation_time'].dt.hour
trip['trip_day'] = trip['trip_creation_time'].dt.day
trip['trip_week'] = trip['trip_creation_time'].dt.isocalendar().week
trip['trip_dayofweek'] = trip['trip_creation_time'].dt.dayofweek
```

```
trip[['trip_year', 'trip_month', 'trip_hour', 'trip_day', 'trip_week', 'trip_dayofweek']]
```

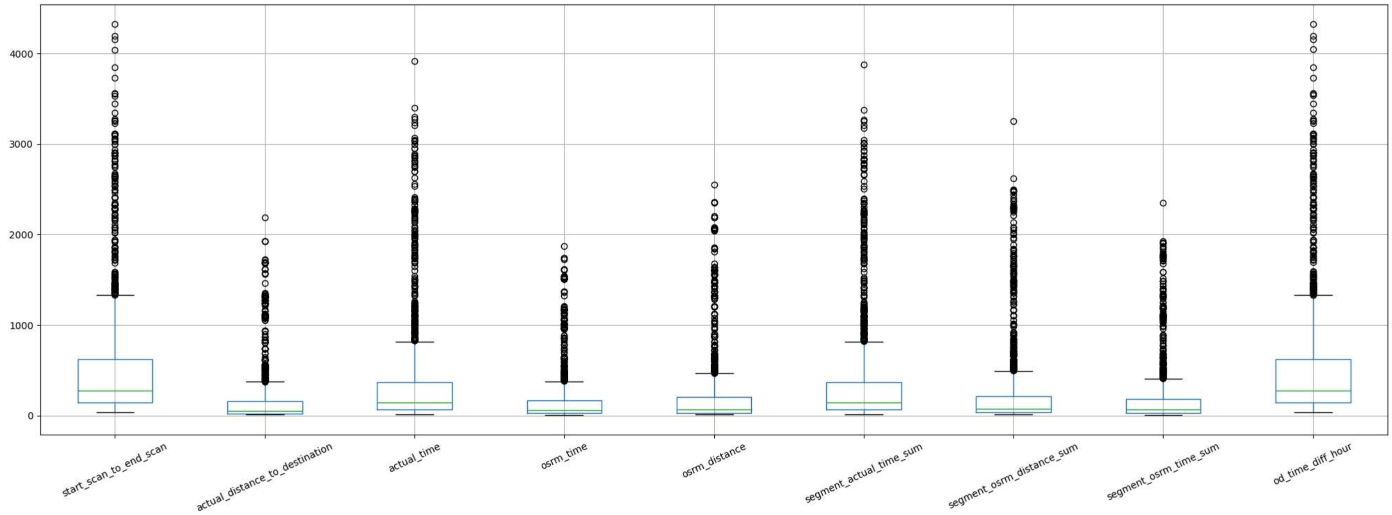
	trip_year	trip_month	trip_hour	trip_day	trip_week	trip_dayofweek
0	2018	9	0	12	37	2
1	2018	9	0	12	37	2
2	2018	9	0	12	37	2
3	2018	9	0	12	37	2
4	2018	9	0	12	37	2
...	...	...	...	...	...	...
2001	2018	10	23	3	40	2
2002	2018	10	23	3	40	2
2003	2018	10	23	3	40	2
2004	2018	10	23	3	40	2
2005	2018	10	23	3	40	2

2006 rows × 6 columns

Identifying outliers in numberical variable

```
num_cols = ['start_scan_to_end_scan', 'actual_distance_to_destination', 'actual_time', 'osrm_time',
            'osrm_distance', 'segment_actual_time_sum', 'segment_osrm_distance_sum',
            'segment_osrm_time_sum', 'od_time_diff_hour']
trip[num_cols].boxplot(rot=25, figsize=(25,8))
```

<Axes: >



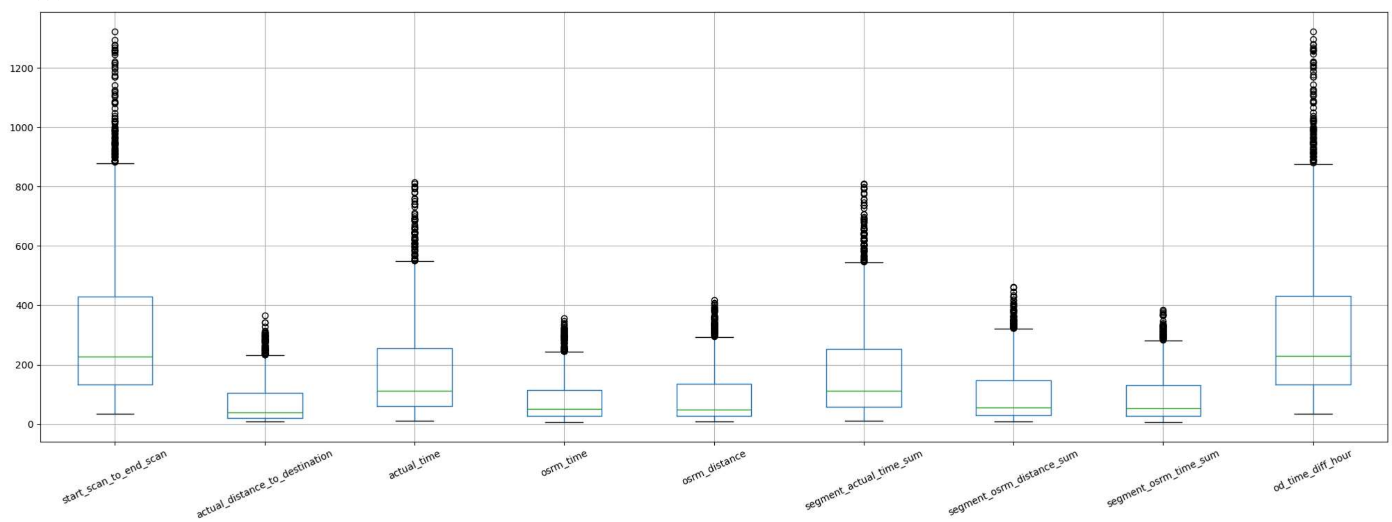
Outlier Handling using IQR

```
Q1 = trip[num_cols].quantile(0.25)
Q3 = trip[num_cols].quantile(0.75)

IQR = Q3 - Q1
trip = trip[~((trip[num_cols] < (Q1 - 1.5 * IQR)) | (trip[num_cols] > (Q3 + 1.5 * IQR))).any(axis=1)]
trip = trip.reset_index(drop=True)
```

```
trip[num_cols].boxplot(rot=25, figsize=(25,8))
```

<Axes: >



Handling Categorical Variables

```
# As there are only two route_type, one hot encoding is preffered
trip['route_type'].value_counts()
```

route_type	count
Carting	1198
FTL	546

```
trip['route_type'] = trip['route_type'].map({'FTL':0, 'Carting':1})
```

Standardization of Numerical Features

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

```
scaler = StandardScaler()
scaler.fit(trip[num_cols])
```

```
▾ StandardScaler
StandardScaler()
```

```
trip[num_cols] = scaler.transform(trip[num_cols])
```

```
trip[num_cols]
```

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time_sum	segment_os
0	-0.116423	0.354040	0.451797	0.332877	0.340357	0.458438	
1	-0.263444	-0.471556	0.378141	-0.638057	-0.513981	0.384329	
2	-0.197672	-0.678186	0.095794	-0.732453	-0.695028	0.106421	
3	0.347857	0.752407	0.746421	3.353561	1.145313	0.748696	
4	-0.236361	0.029914	-0.211106	-0.314412	-0.149878	-0.220892	
...	...	...	...	...	...	...	
1739	-0.665820	-0.331820	-0.315452	-0.530175	-0.401472	-0.313528	
1740	-0.665820	-0.408621	-0.346142	-0.408809	-0.518378	-0.350582	
1741	-0.538143	-0.641986	-0.720560	-0.705483	-0.608908	-0.721125	
1742	0.390416	-0.482628	0.629799	-0.435779	-0.381918	0.637533	
1743	0.127324	-0.108641	0.586833	-0.166075	-0.143279	0.594303	

1744 rows × 9 columns

Recommendations:

There is a notable disparity between OSRM parameters and actual metrics.

1. Action Points:

Review the data inputs provided to the routing engine for trip planning. Investigate any discrepancies with transporters and ensure the routing engine is optimized for accurate results.

2. Regional Presence Analysis:

North, South, and West zones experience high order volumes, whereas the Central, Eastern, and North-Eastern zones have comparatively lower activity. While this observation is based on only two months of data and requires further validation, it is worth exploring opportunities to expand operations in these regions.

3. State-Level Insights:

Maharashtra leads in traffic volume, followed by Karnataka, making these states key focus areas for resource planning. This is especially critical during festive seasons to ensure smooth operations.