In [148…]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency, ttest_ind, f_oneway, kruskal, shapiro, ttes
from sklearn.preprocessing import OneHotEncoder, StandardScaler, MinMaxScaler
```

In [2]:
```
df=pd.read_csv("delhivery_data.csv")
```

In [4]:
```
df.head()
```

Out[4]:

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

5 rows × 24 columns

In [5]:
```
df.shape
```

Out[5]:
```
(144867, 24)
```

In [6]:
```
df.info()
```

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

In [7]:
```
df.describe()
```

Out[7]:

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	osrm
count	144867.000000	144867.000000	144867.000000	144867.000000	144867.0
mean	961.262986	232.926567	234.073372	416.927527	213.8
std	1037.012769	344.755577	344.990009	598.103621	308.(
min	20.000000	9.000000	9.000045	9.000000	6.0
25%	161.000000	22.000000	23.355874	51.000000	27.0
50%	449.000000	66.000000	66.126571	132.000000	64.0
75%	1634.000000	286.000000	286.708875	513.000000	257.0
max	7898.000000	1927.000000	1927.447705	4532.000000	1686.0

In [8]:
```
df.dtypes
```

Out[8]:

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

In [9]:

```
df.isna().sum()
```

Out[9]:

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

In [10]:

```
df.nunique()
```

Out[10]:
```
data                                2
trip_creation_time                  14817
route_schedule_uuid                 1504
route_type                          2
trip_uuid                           14817
source_center                       1508
source_name                         1498
destination_center                  1481
destination_name                    1468
od_start_time                       26369
od_end_time                         26369
start_scan_to_end_scan              1915
is_cutoff                           2
cutoff_factor                       501
cutoff_timestamp                    93180
actual_distance_to_destination      144515
actual_time                         3182
osrm_time                           1531
osrm_distance                       138046
factor                              45641
segment_actual_time                 747
segment_osrm_time                   214
segment_osrm_distance               113799
segment_factor                      5675
dtype: int64
```

In [11]:
```
df.duplicated().sum()
```

Out[11]:
```
0
```

In [12]:
```
#dropping unnecessary attributes
un_att=["data", "is_cutoff" ,"cutoff_factor", "cutoff_timestamp","factor", "segment_f
df=df.drop(columns=un_att)
```

In [13]:
```
df.head(2)
```

Out[13]:

	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	sc
0	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_V
1	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_V

In [14]:
```
df.shape
```

Out[14]:
```
(144867, 18)
```

Missing value treatment:As dataset is very large and we have very few missing values so it's better to drop those values as replacing them will create unnecessary complication of the dataset.

In [17]:
```
df=df.dropna()
```

In [18]:
```
df.isna().sum()
```

Out[18]:
```
trip_creation_time               0
route_schedule_uuid              0
route_type                       0
trip_uuid                        0
source_center                    0
source_name                      0
destination_center               0
destination_name                 0
od_start_time                    0
od_end_time                      0
start_scan_to_end_scan           0
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
```

Now data is pretty clean and we don't have any null or missing values

In [23]:
```
#converting datetime column into date & time format
datetime_cols=["trip_creation_time", "od_start_time","od_end_time"]
for i in datetime_cols:
    df[i]=pd.to_datetime(df[i])
```

In [25]:
```
df.dtypes
```

Out[25]:
```
trip_creation_time               datetime64[ns]
route_schedule_uuid                      object
route_type                               object
trip_uuid                                object
source_center                            object
source_name                              object
destination_center                       object
destination_name                         object
od_start_time                    datetime64[ns]
od_end_time                      datetime64[ns]
start_scan_to_end_scan                  float64
actual_distance_to_destination          float64
actual_time                             float64
osrm_time                               float64
osrm_distance                           float64
segment_actual_time                     float64
segment_osrm_time                       float64
segment_osrm_distance                   float64
dtype: object
```

Grouping by segment

In [30]:
```
df["segment_key"]=df["trip_uuid"] +"_"+ df["source_center"] +"_"+ df["destination_cen
```

In [31]:
```
#Merging rows in columns based on segment key
seg_agg_cols=["segment_actual_time","segment_osrm_distance","segment_osrm_time"]
for i in seg_agg_cols:
    df[i+"_sum"]=df.groupby("segment_key")[i].cumsum()
```

In [32]:
```
df.head(1)
```

Out[32]:

	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	sc
0	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_V

1 rows × 22 columns

Aggregating at segment level

In [33]:
```
create_segment_dict = {
    "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",}
```

In [36]:
```
seg_agg_data=df.groupby("segment_key").aggregate(create_segment_dict).reset_index()
seg_agg_data=seg_agg_data.sort_values(by=["segment_key","od_end_time"])
seg_agg_data
```

Out[36]:

	segment_key	trip_creation_time	route_schedule_uuid
0	trip-153671041653548748_IND209304AAA_IND000000ACB	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...
1	trip-153671041653548748_IND462022AAA_IND209304AAA	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...
2	trip-153671042288605164_IND561203AAB_IND562101AAA	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...
3	trip-153671042288605164_IND572101AAA_IND561203AAB	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...
4	trip-153671043369099517_IND000000ACB_IND160002AAC	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...
...	...	...	...
26217	trip-153861115439069069_IND628204AAA_IND627657AAA	2018-10-03 23:59:14.390954	thanos::sroute:c5f2ba2c-8486-4940-8af6-d1d2a6a...
26218	trip-153861115439069069_IND628613AAA_IND627005AAA	2018-10-03 23:59:14.390954	thanos::sroute:c5f2ba2c-8486-4940-8af6-d1d2a6a...
26219	trip-153861115439069069_IND628801AAA_IND628204AAA	2018-10-03 23:59:14.390954	thanos::sroute:c5f2ba2c-8486-4940-8af6-d1d2a6a...
26220	trip-153861118270144424_IND583119AAA_IND583101AAA	2018-10-03 23:59:42.701692	thanos::sroute:412fea14-6d1f-4222-8a5f-a517042...
26221	trip-153861118270144424_IND583201AAA_IND583119AAA	2018-10-03 23:59:42.701692	thanos::sroute:412fea14-6d1f-4222-8a5f-a517042...

26222 rows × 19 columns

Feature Engineering

Time difference between order start and order end time

In [38]:
```
seg_agg_data["od_time_diff_hour"]=(seg_agg_data["od_end_time"]-seg_agg_data["od_start
seg_agg_data.head(2)
```

Out[38]:

	segment_key	trip_creation_time	route_schedule_uuid	route
0	trip-153671041653548748_IND209304AAA_IND000000ACB	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	
1	trip-153671041653548748_IND462022AAA_IND209304AAA	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	

Source state and Destination state

In [40]:
```
seg_agg_data["source_state_name"]=seg_agg_data["source_name"].apply(lambda x:str(x).s
seg_agg_data["destination_state_name"]=seg_agg_data["destination_name"].apply(lambda
```

In [41]:
```
seg_agg_data.head(2)
```

Out[41]:

	segment_key	trip_creation_time	route_schedule_uuid	route
0	trip-153671041653548748_IND209304AAA_IND000000ACB	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	
1	trip-153671041653548748_IND462022AAA_IND209304AAA	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	

2 rows × 22 columns

Trip creation time in months and years

In [44]:
```
seg_agg_data["trip_cr_year"]=seg_agg_data["trip_creation_time"].dt.year
seg_agg_data["trip_cr_month"]=seg_agg_data["trip_creation_time"].dt.month
seg_agg_data["trip_cr_week"]=seg_agg_data["trip_creation_time"].dt.isocalendar().week
seg_agg_data["trip_cr_day"]=seg_agg_data["trip_creation_time"].dt.day
seg_agg_data["trip_cr_day_of_week"]=seg_agg_data["trip_creation_time"].dt.dayofweek
seg_agg_data["trip_cr_hour"]=seg_agg_data["trip_creation_time"].dt.hour
```

In [45]:
```
seg_agg_data.head(2)
```

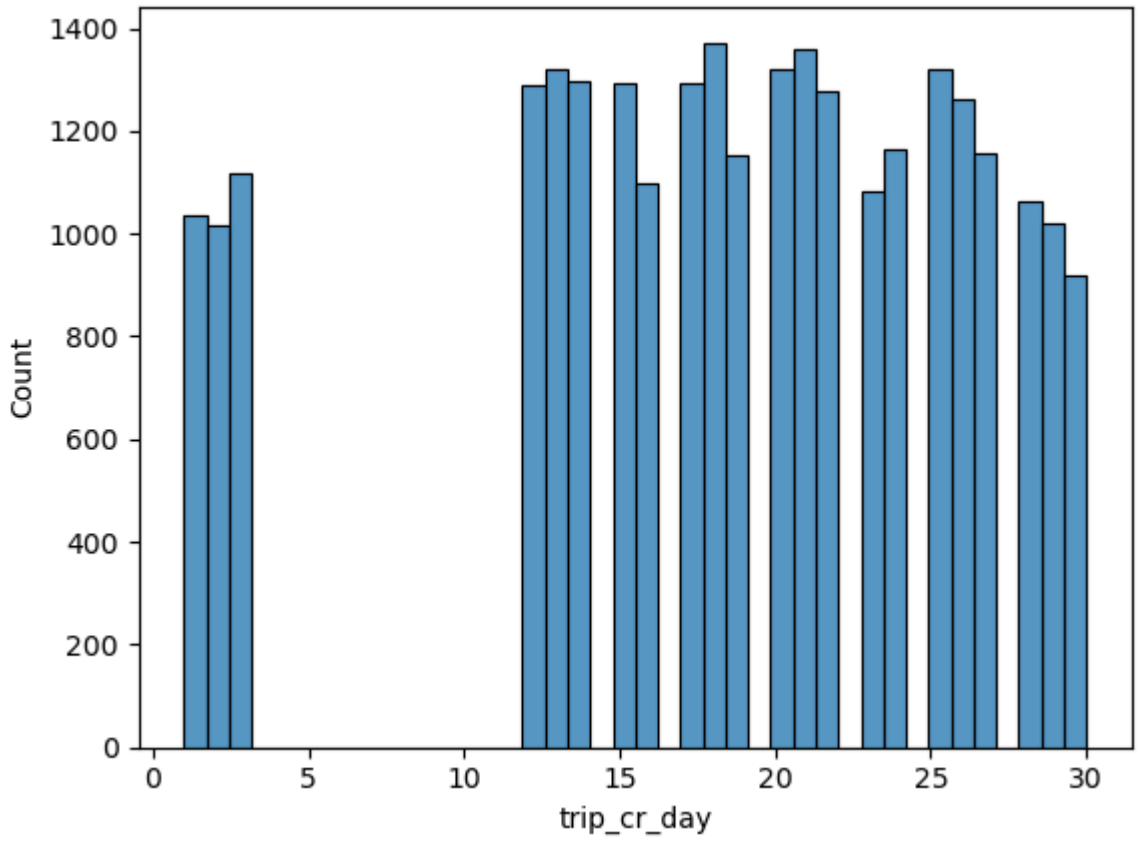
Out[45]:

	segment_key	trip_creation_time	route_schedule_uuid	route
0	trip-153671041653548748_IND209304AAA_IND000000ACB	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	
1	trip-153671041653548748_IND462022AAA_IND209304AAA	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	

2 rows × 28 columns

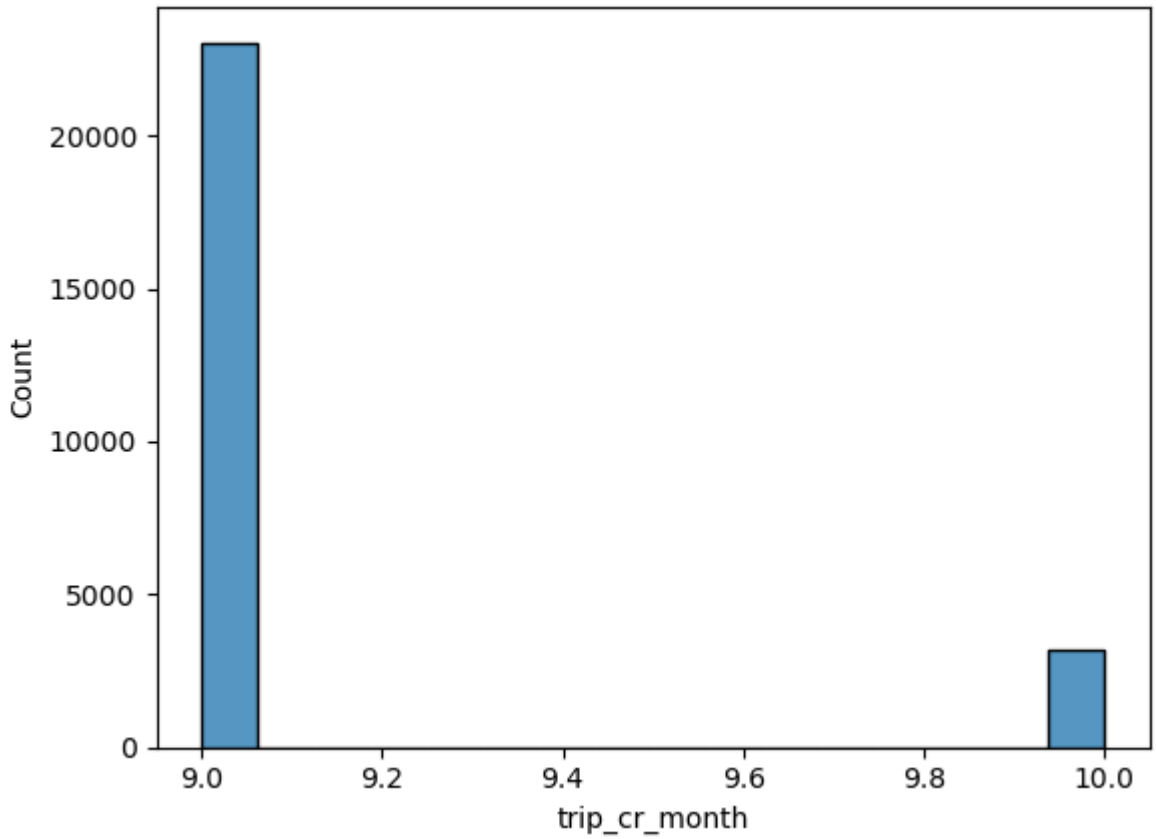
In [202…
```
sns.histplot(data=seg_agg_data,x="trip_cr_day")
```

Out[202]:
```
<Axes: xlabel='trip_cr_day', ylabel='Count'>
```



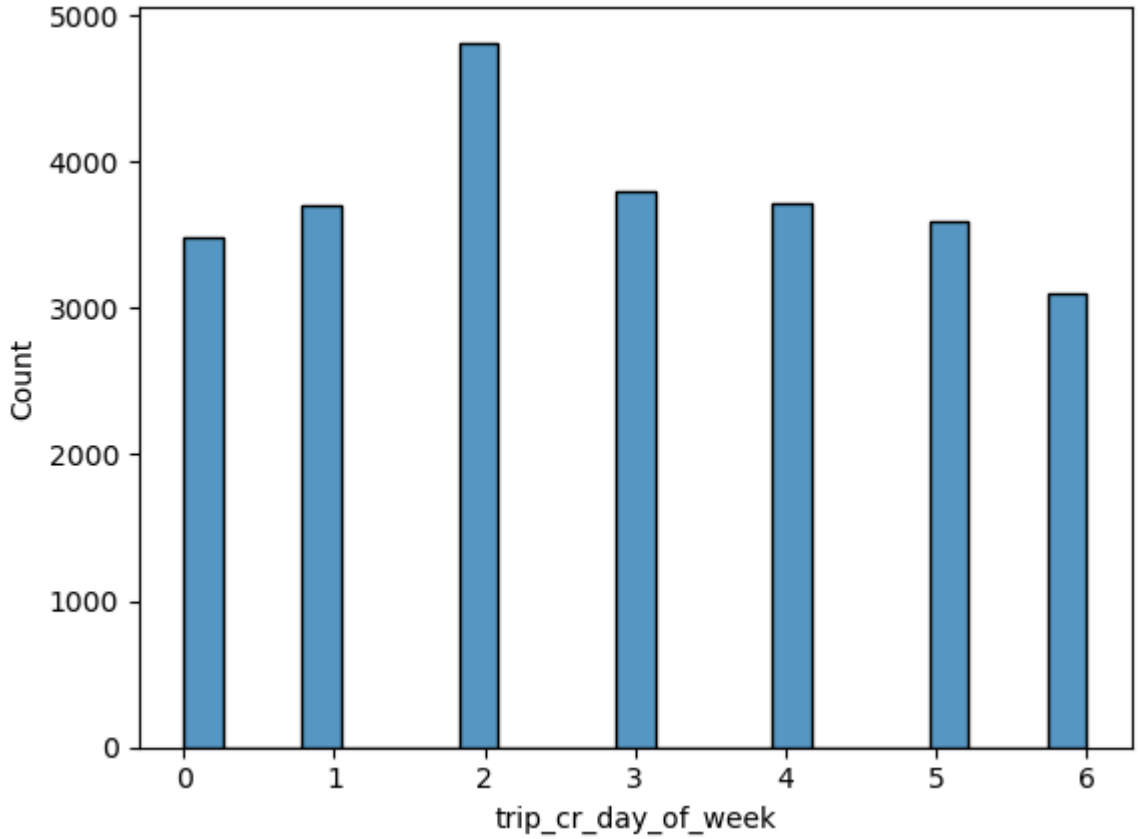
In [203…
```
sns.histplot(data=seg_agg_data,x="trip_cr_month")
```

Out[203]:
```
<Axes: xlabel='trip_cr_month', ylabel='Count'>
```



In [204…
```
sns.histplot(data=seg_agg_data,x="trip_cr_day_of_week")
```

Out[204]:
```
<Axes: xlabel='trip_cr_day_of_week', ylabel='Count'>
```



In-depth analysis

In [90]:
```
trip_df=seg_agg_data.copy()
```

In [91]:
```
trip_df.head(1)
```

Out[91]:

	segment_key	trip_creation_time	route_schedule_uuid	route
0	trip-153671041653548748_IND209304AAA_IND000000ACB	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	

1 rows × 28 columns

In [92]:
```
create_trip_dict= {
    "trip_creation_time":"first",
    "route_schedule_uuid":"first",
    "route_type":"first",
    "trip_uuid":"first",
    "source_center":"first",
    "source_name":"first",
    "destination_center":"first",
    "destination_name":"first",
    "od_start_time":"first",
    "od_end_time":"last",
    "start_scan_to_end_scan":"sum",
    "actual_distance_to_destination":"sum",
    "actual_time":"sum",
    "osrm_time":"sum",
    "osrm_distance":"sum",
    "segment_actual_time_sum":"sum",
    "segment_osrm_distance_sum":"sum",
    "segment_osrm_time_sum":"sum",
    "od_time_diff_hour":"sum",
    "source_state_name":"first",
```

```
    "destination_state_name":"first",
    "trip_cr_year":"first",
    "trip_cr_month":"first",
    "trip_cr_week":"first",
    "trip_cr_day":"first",
    "trip_cr_day_of_week":"first",
    "trip_cr_hour":"first"
}
```

In [93]:
```
trip_agg_data=trip_df.groupby("trip_uuid").agg(create_trip_dict).reset_index(drop=Tru
trip_agg_data.head(2)
```

Out[93]:

	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	
0	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip-153671041653548748	IND209304AAA	Ka
1	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	trip-153671042288605164	IND561203AAB	Doddab

2 rows × 27 columns

Outlier Detection & Treatment

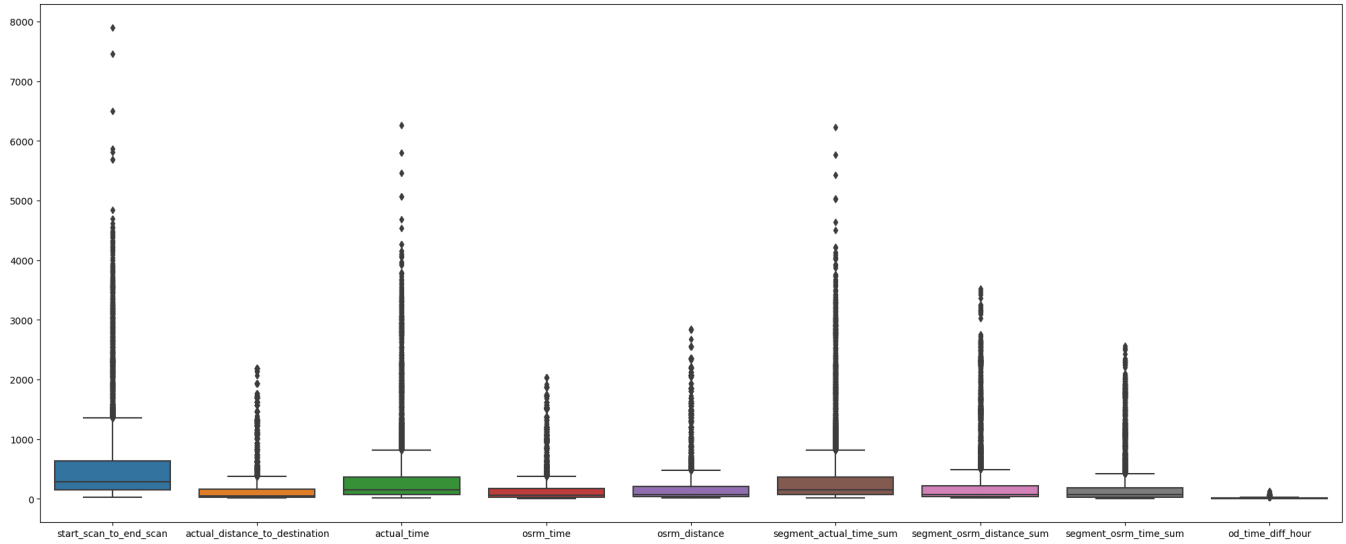
In [94]:
```
num_cols=trip_agg_data.select_dtypes(include=np.float64)
num_cols
```

Out[94]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance
0	2259.0	824.732854	1562.0	717.0	991.3523
1	180.0	73.186911	143.0	68.0	85.1110
2	3933.0	1927.404273	3347.0	1740.0	2354.0665
3	100.0	17.175274	59.0	15.0	19.6800
4	717.0	127.448500	341.0	117.0	146.7918
...	...	...	...	...	...
14782	257.0	57.762332	83.0	62.0	73.4630
14783	60.0	15.513784	21.0	12.0	16.0882
14784	421.0	38.684839	282.0	48.0	58.9037
14785	347.0	134.723836	264.0	179.0	171.1103
14786	353.0	66.081533	275.0	68.0	80.5787

14787 rows × 9 columns

In [95]:
```
fig = plt.figure(figsize=(25,10))
sns.boxplot(data=num_cols)
plt.show()
```

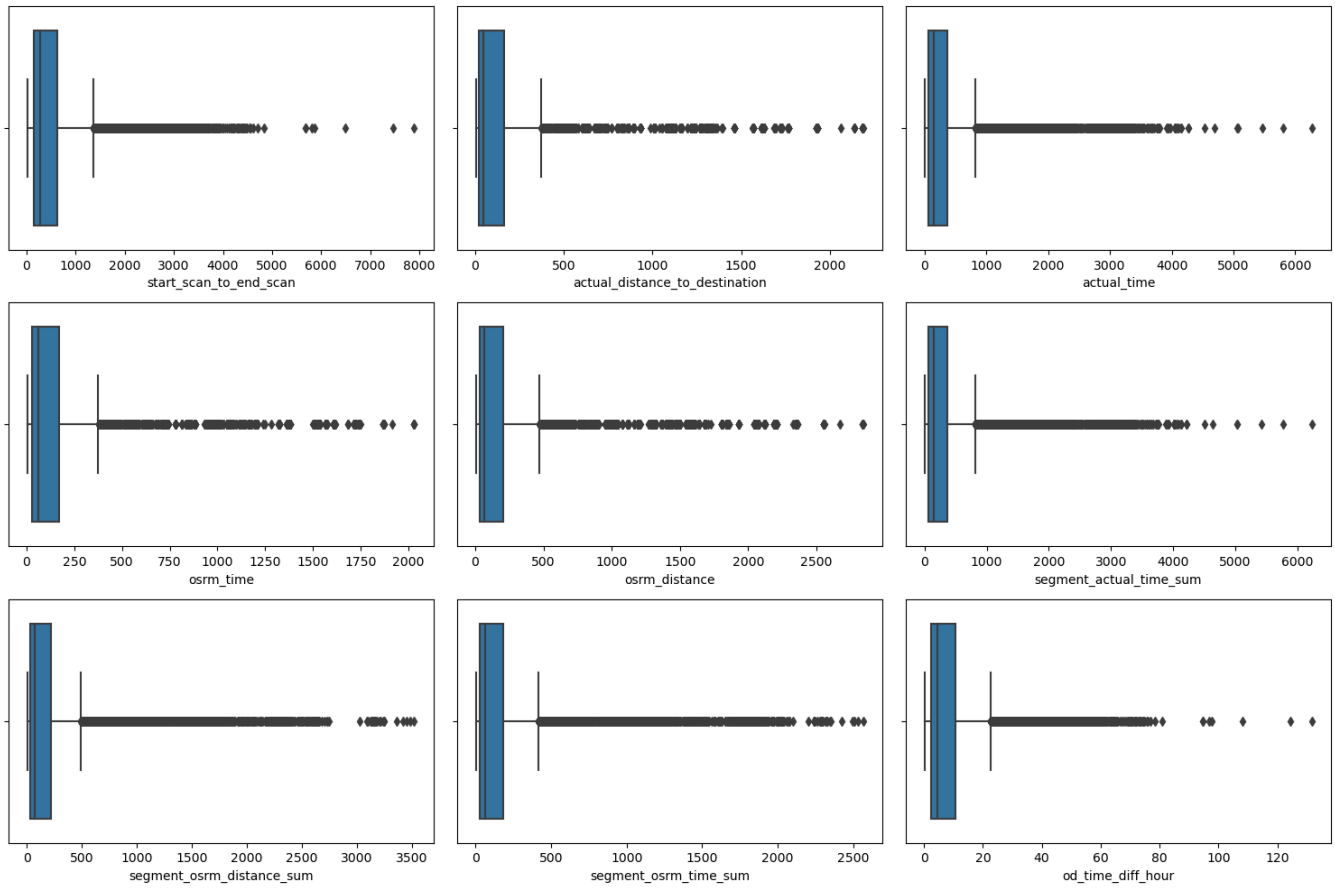


IQR Method

In [96]:
```
q1=num_cols.quantile(0.25)
q3=num_cols.quantile(0.75)
IQR=q3-q1
IQR
```

Out[96]:
```
start_scan_to_end_scan            483.000000
actual_distance_to_destination    140.814159
actual_time                       300.000000
osrm_time                         139.000000
osrm_distance                     175.887300
segment_actual_time_sum           298.000000
segment_osrm_distance_sum         183.981750
segment_osrm_time_sum             154.000000
od_time_diff_hour                   8.063987
dtype: float64
```

In [128…
```
plt.figure(figsize=(15,10))
for i, col in enumerate(num_cols,1):

    dt = trip_agg_data[col]

    Q1 = np.percentile(dt, 25)
    Q3 = np.percentile(dt, 75)
    IQR = Q3 - Q1

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

    filt_data = dt.loc[(da >= lower_bound) | (dt <= upper_bound)]

    plt.subplot(3,3,i)
    sns.boxplot(x=filt_data)
plt.tight_layout()
plt.show()
```



In [130…
```
num_cols
```

Out[130]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distanc
0	2259.0	824.732854	1562.0	717.0	991.3523
1	180.0	73.186911	143.0	68.0	85.1110
2	3933.0	1927.404273	3347.0	1740.0	2354.0665
3	100.0	17.175274	59.0	15.0	19.6800
4	717.0	127.448500	341.0	117.0	146.7918
...	...	...	...	...	..
14782	257.0	57.762332	83.0	62.0	73.4630
14783	60.0	15.513784	21.0	12.0	16.0882
14784	421.0	38.684839	282.0	48.0	58.9037
14785	347.0	134.723836	264.0	179.0	171.1103
14786	353.0	66.081533	275.0	68.0	80.5787

14787 rows × 9 columns

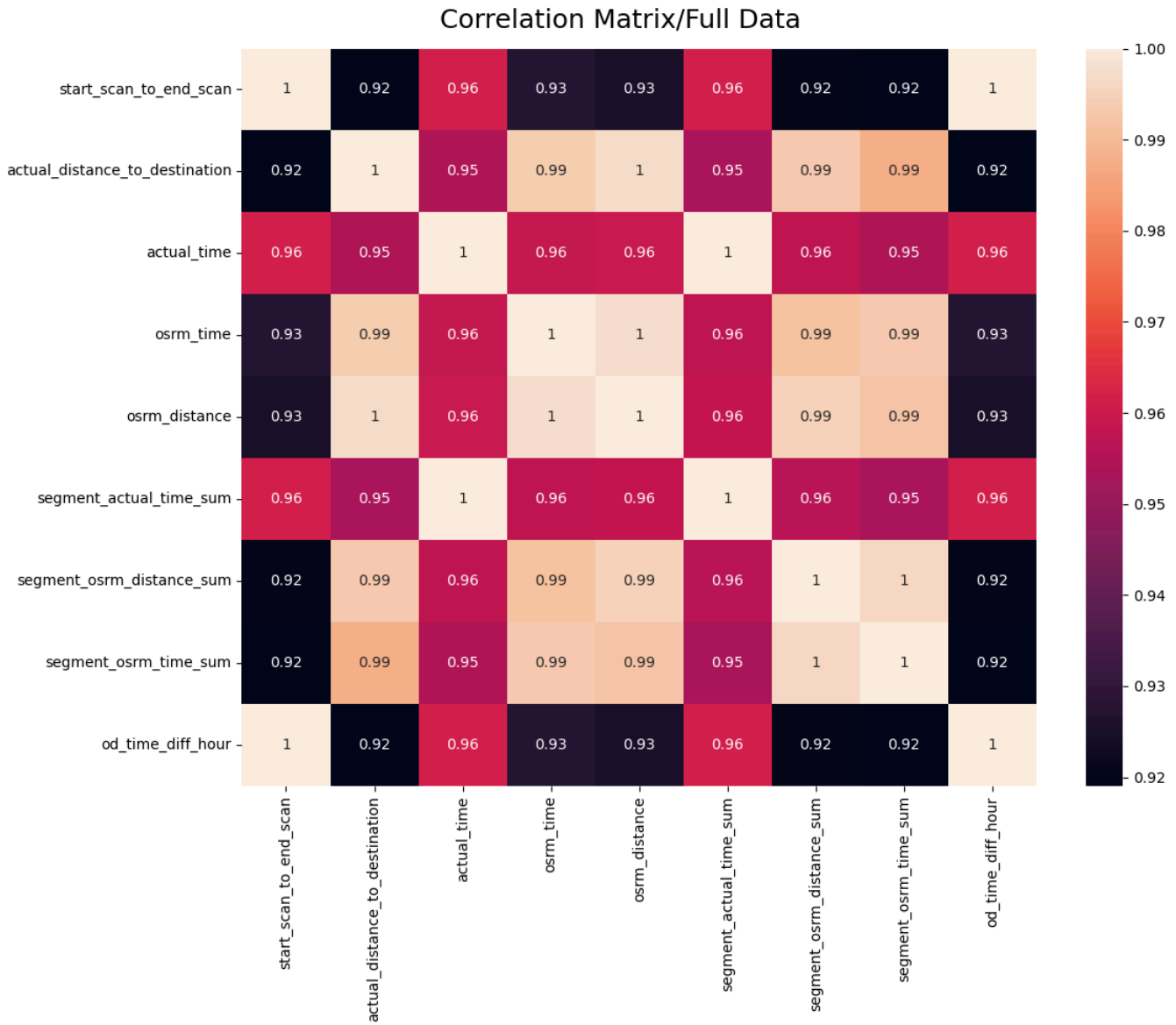
In [132…
```
num_cols_corr=num_cols.corr()
num_cols_corr
```

Out[132]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	os
start_scan_to_end_scan	1.000000	0.919159	0.961612	
actual_distance_to_destination	0.919159	1.000000	0.953920	
actual_time	0.961612	0.953920	1.000000	
osrm_time	0.927471	0.993568	0.958781	
osrm_distance	0.925205	0.997268	0.959398	
segment_actual_time_sum	0.961634	0.952987	0.999989	
segment_osrm_distance_sum	0.920191	0.993068	0.957151	
segment_osrm_time_sum	0.919429	0.987542	0.954044	
od_time_diff_hour	0.999999	0.919074	0.961560	

In [142…

```
fig = plt.figure(figsize=(12,10)).suptitle("Correlation Matrix/Full Data",fontsize=18
sns.heatmap(data=num_cols_corr,annot=True)
plt.tight_layout()
plt.show()
```



In [135…

```
num_dt=num_cols.copy()
```

In [137…

```
num_cols_ser = numerical_columns.columns.tolist()
num_cols_ser
```

Out[137]:
```
['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']
```

In [139…
```
q1 = np.percentile(num_dt[num_cols_ser], 25)
q3 = np.percentile(num_dt[num_cols_ser], 75)
IQR = q3 - q1

lower_bound = q1 - (1.5 * IQR)
upper_bound = q3 + (1.5 * IQR)

filt_num_data = num_dt[num_cols_ser][(num_dt[num_cols_ser] >= lower_bound) | (num_dt[
filt_num_data.head(2)
```

Out[139]:

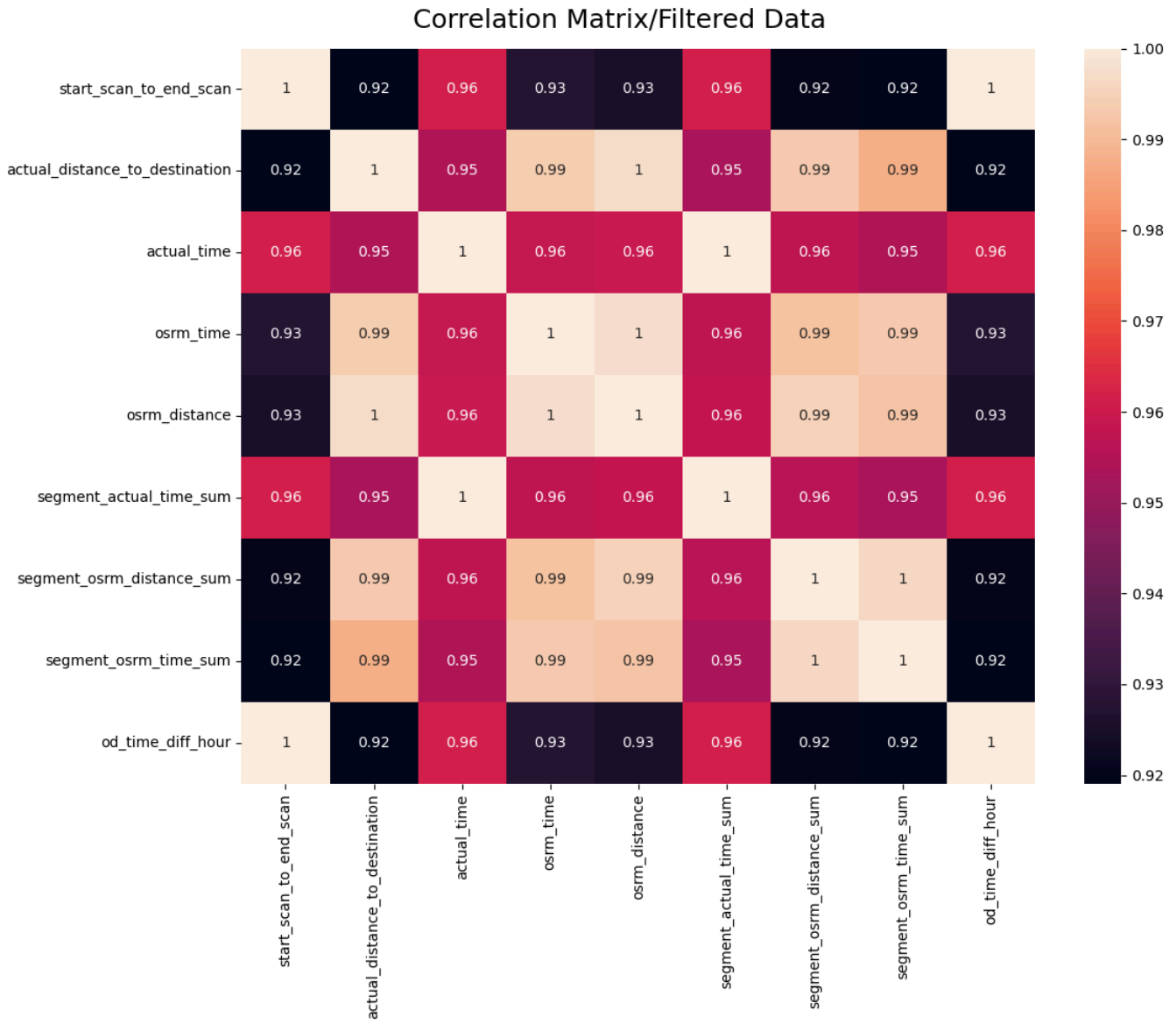
	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	se
0	2259.0	824.732854	1562.0	717.0	991.3523	
1	180.0	73.186911	143.0	68.0	85.1110	

In [140…
```
filt_num_data_corr=filtered_num_data.corr()
num_cols_corr
```

Out[140]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	o
start_scan_to_end_scan	1.000000	0.919159	0.961612	
actual_distance_to_destination	0.919159	1.000000	0.953920	
actual_time	0.961612	0.953920	1.000000	
osrm_time	0.927471	0.993568	0.958781	
osrm_distance	0.925205	0.997268	0.959398	
segment_actual_time_sum	0.961634	0.952987	0.999989	
segment_osrm_distance_sum	0.920191	0.993068	0.957151	
segment_osrm_time_sum	0.919429	0.987542	0.954044	
od_time_diff_hour	0.999999	0.919074	0.961560	

In [141…
```
fig = plt.figure(figsize=(12,10)).suptitle("Correlation Matrix/Filtered Data",fontsiz
sns.heatmap(data=num_cols_corr,annot=True)
plt.tight_layout()
plt.show()
```



One hot encoding

In [157…
```
cat_col=["route_type"]
```

In [158…
```
one_hot_encoder=OneHotEncoder()
df[cat_col]=one_hot_encoder.fit_transform(df[cat_col])
```

In [162…
```
df.head(2)
```

Out[162]:

	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	s
0	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	(0, 0)\t1.0	trip-153741093647649320	IND388121AAA	Anand_
1	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	(0, 0)\t1.0	trip-153741093647649320	IND388121AAA	Anand_

2 rows × 22 columns

Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler.

In [164…]
```
# normalizing using minmax scaler as the data is not gaussian and have too many outli
min_max_scaler = MinMaxScaler()
min_max_scaled_num = min_max_scaler.fit_transform(num_dt[num_cols_ser])
min_max_scaled_df = pd.DataFrame(min_max_scaled_num, columns=num_cols_ser)
min_max_scaled_df
```

Out[164]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distanc
0	0.283937	0.374613	0.248242	0.350938	0.34697
1	0.019937	0.029476	0.021419	0.030602	0.02685
2	0.496508	0.880999	0.533568	0.855874	0.82832
3	0.009778	0.003753	0.007992	0.004442	0.00374
4	0.088127	0.054395	0.053069	0.054788	0.04864
...	...	...	...	...	...
14782	0.029714	0.022392	0.011829	0.027641	0.02274
14783	0.004698	0.002990	0.001918	0.002962	0.00247
14784	0.050540	0.013631	0.043638	0.020731	0.01760
14785	0.041143	0.057736	0.040761	0.085390	0.05723
14786	0.041905	0.026213	0.042519	0.030602	0.02525

14787 rows × 9 columns

Hypothesis Testing

Step 1: Setup null and alternate hypothesis

- Null Hypothesis(Ho):mean of both column is same
- Alternate Hypothesis(Ha):mean of both column is different

Step 2: Determine the type of distribution

Step 3 :Determine p-value and set significance level(alpha)

Here we will take alpha as 0.05

Step 4:Compare p-value with significance level(alpha)

Actual_time aggregated value and OSRM time aggregated value

In [165…]
```
filt_actual_time = filt_num_data["actual_time"]
filt_osrm_time = filt_num_data["osrm_time"]
```

In [170…]
```
#Ho:actual time aggregated value and osrm time aggregated value is same
#ha:actual time aggregated value and osrm time aggregated value is different
#we will use ttest_ind here
t_stat, p_value = ttest_ind(filt_actual_time,filt_osrm_time,alternative="greater")
print("p_value:",p_value)

alpha = 0.05

if p_value < alpha:
    print("Reject H0")
    print("Actual time aggregated value is greater than osrm time aggregated value")
```

```
else:
    print("Fail to reject H0")
    print("Actual time aggregated value and osrm time aggregated value is same")
```

```
p_value: 4.1073095671733e-310
Reject H0
Actual time aggregated value is greater than osrm time aggregated value
```

Actual_time aggregated value and segment actual time aggregated value

In [174…
```
filt_actual_time = filt_num_data["actual_time"]
filt_segment_actual_time = filt_num_data["segment_actual_time_sum"]
```

In [176…
```
#Ho:actual time aggregated value and segment actual time aggregated value is same
#ha:actual time aggregated value and segment actual time aggregated value is differen
#we will use ttest_ind here
t_stat, p_value = ttest_ind(filt_actual_time,filt_segment_actual_time)
print("p_value:",p_value)

alpha = 0.05

if p_value < alpha:
    print("Reject H0")
    print("Actual time aggregated value and segment actual time aggregated value is d

else:
    print("Fail to reject H0")
    print("Actual time aggregated value and segment actual time aggregated value is s
```

```
p_value: 0.6174479719707524
Fail to reject H0
Actual time aggregated value and segment actual time aggregated value is same
```

OSRM distance aggregated value and segment OSRM distance aggregated value

In [178…
```
filt_osrm_distance = filt_num_data["osrm_distance"]
filt_segmented_osrm_distance = filt_num_data["segment_osrm_distance_sum"]
```

In [200…
```
#Ho:OSRM distance aggregated value and segment OSRM distance aggregated value is same
#ha:OSRM distance aggregated value and segment OSRM distance aggregated value is diff
#we will use ttest_ind here
t_stat, p_value = ttest_ind(filt_osrm_distance,filt_segmented_osrm_distance)
print("p_value:",p_value)

alpha = 0.05

if p_value < alpha:
    print("Reject H0")
    print("There is significant difference between OSRM distance aggregated value and

else:
    print("Fail to reject H0")
    print("OSRM distance aggregated value and segment OSRM distance aggregated value
```

```
p_value: 4.0929578191203324e-05
Reject H0
There is significant difference between OSRM distance aggregated value and segment OS
RM distance aggregated value
```

OSRM time aggregated value and segment OSRM time aggregated value

In [181…
```
filt_osrm_time = filt_num_data["osrm_time"]
filt_segmented_osrm_time = filt_num_data["segment_osrm_time_sum"]
```

In [184…

```
#Ho:OSRM time aggregated value and segment OSRM time aggregated value is same
#ha:OSRM time aggregated value and segment OSRM time aggregated value is different
#we will use ttest_ind here
t_stat, p_value = ttest_ind(filt_osrm_time,filt_segmented_osrm_time)
print("p_value:",p_value)

alpha = 0.05

if p_value < alpha:
    print("Reject H0")
    print("There is significant difference between OSRM time aggregated value and seg

else:
    print("Fail to reject H0")
    print("OSRM time aggregated value and segment OSRM time aggregated value is same"
```

```
p_value: 1.128703468644937e-08
Reject H0
There is significant difference between OSRM time aggregated value and segment OSRM t
ime aggregated
```

Business Insights

Busiest corridor, avg distance between them, avg time taken

In [198…

```
# To find the busiest corridor, we'll look at the most common combinations of source
corridor_freq = trip_agg_data.groupby(["source_state_name", "destination_state_name"]
busiest_corridor = corridor_freq.sort_values(by="Freq", ascending=False).head(1)

# Average distance and time taken for the busiest corridor
busiest_corridor_details = busiest_corridor.merge(trip_agg_data, on=["source_state_na
average_distance = busiest_corridor_details["actual_distance_to_destination"].mean()
average_time = busiest_corridor_details["od_time_diff_hour"].mean()

display(busiest_corridor)
print("Average distance:  ",average_distance)
print("Average time (in hours):  ",average_time)
```

	source_state_name	destination_state_name	Freq
85	Maharashtra	Maharashtra	2458

```
Average distance:   74.85284867694604
Average time (in hours):   5.346577921457034
```

- Busiest source and destination state is Maharashtra followed by Karnataka.
- Busiest source and destination city is Gurgaon and Bangalore.
- There is difference in actual time taken and predicted time through OSRM with actual more than OSRM in most of the cases.
- From correlation matrix it can be inferred that almost all features are highly correlated.
- From test we can infer that actual time and segment actual time is same, so we can tell that their is consistency in time measurement across segments.
- Order frequency is very high in mid to end of month.

Recommendations

- As Maharashtra and karnataka is busiest state, more optimised planning and mode of transportation should be applied.
- For busy cities like Gurgaon and Bangalore more number of inventories should be installed as well as more delivery persons should be hired for faster deliver in these busy cities.
- As actual time is more than OSRM time ,more robust time mangement system should be installed for accurate predictions.
- FTL mode of transportation should be preferred more as it helps is faster delivery.
- From the insights we can see that amount of orders are higher towards mind to end month and towards end of the year, so company should focus on increasing their transportation and shipment medium around this time.
- As end months of the year high high order volume so company should temporarily increase their workforce around these months.