```
import pandas as pd
import numpy as np
pd.set_option('display.max_columns',None)
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
from scipy.stats import spearmanr
from scipy.stats import norm
from statsmodels.stats.weightstats import ztest
from scipy.stats import ttest_1samp, ttest_ind
from scipy.stats import f_oneway
from scipy.stats import chi2_contingency
```

```
df= pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181')
df.head(5)
```

	data	trip_creation_time	route_schedule_uuid	route_type	
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	15374109
4	training	2018-09-20	thanos::sroute:eb7bfc78-b351-4c0e-a951-	Carting	

```
# Droping unnecessary columns
df.drop(columns=['is_cutoff','cutoff_factor','cutoff_timestamp','factor','segment_factor'],inplace=True)
```

```
df.shape
```

```
(144867, 19)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 19 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144867 non-null  object
 1   trip_creation_time              144867 non-null  object
 2   route_schedule_uuid             144867 non-null  object
 3   route_type                      144867 non-null  object
 4   trip_uuid                       144867 non-null  object
 5   source_center                   144867 non-null  object
 6   source_name                     144574 non-null  object
 7   destination_center              144867 non-null  object
 8   destination_name                144606 non-null  object
 9   od_start_time                   144867 non-null  object
 10  od_end_time                     144867 non-null  object
 11  start_scan_to_end_scan          144867 non-null  float64
 12  actual_distance_to_destination  144867 non-null  float64
 13  actual_time                     144867 non-null  float64
 14  osrm_time                       144867 non-null  float64
 15  osrm_distance                   144867 non-null  float64
 16  segment_actual_time             144867 non-null  float64
 17  segment_osrm_time               144867 non-null  float64
 18  segment_osrm_distance           144867 non-null  float64
dtypes: float64(8), object(11)
memory usage: 21.0+ MB
```

```
# Statistical summary
df.describe()
```

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time	segment_
count	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144
mean	961.262986	234.073372	416.927527	213.868272	284.771297	36.196111	
std	1037.012769	344.990009	598.103621	308.011085	421.119294	53.571158	
min	20.000000	9.000045	9.000000	6.000000	9.008200	-244.000000	
25%	161.000000	23.355874	51.000000	27.000000	29.914700	20.000000	
50%	449.000000	66.126571	132.000000	64.000000	78.525800	29.000000	
75%	1634.000000	286.708875	513.000000	257.000000	343.193250	40.000000	
max	7898.000000	1927.447705	4532.000000	1686.000000	2326.199100	3051.000000	1

```
# Missing value detection
df.isnull().sum()
```

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

```
# Missing values Treatment
df = df.dropna(how='any')
df = df.reset_index(drop=True)
```

```
# conversion of columns
df['od_start_time'] = pd.to_datetime(df['od_start_time'])
df['od_end_time'] = pd.to_datetime(df['od_end_time'])
```

Merging of rows

```
# Merge based on Trip_uuid, Source_ID, and Destination_ID
df['segment_key'] = df['trip_uuid'] + df['source_center'] + df['destination_center']

segment_cols = ['segment_actual_time', 'segment_osrm_distance', 'segment_osrm_time']

for col in segment_cols:
    df[col + '_sum'] = df.groupby('segment_key')[col].cumsum()

df[[col + '_sum' for col in segment_cols]]
```

	segment_actual_time_sum	segment_osrm_distance_sum	segment_osrm_time_sum
0	14.0	11.9653	11.0
1	24.0	21.7243	20.0
2	40.0	32.5395	27.0
3	61.0	45.5619	39.0
4	67.0	49.4772	44.0

```
create_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',

}


# segment_key conatins "Trip_uuid, Source_ID, and Destination_ID"
segment = df.groupby('segment_key').agg(create_segment_dict).reset_index()
segment = segment.sort_values(by=['segment_key','od_end_time'], ascending=True).reset_index()


#  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']
```

```
0        1260.604421
1         999.505379
2          58.832388
3         122.779486
4         834.638929
             ...
26217      62.115193
26218      91.087797
26219      44.174403
26220     287.474007
26221      66.933565
Name: od_time_diff_hour, Length: 26222, dtype: float64
```

```
segment.head(5)
```

	index	segment_key	data	trip_creation_time	route_schedule_uuid	route_type	trip_uui
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip 15367104165354874
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip 15367104165354874

```
segment[segment['trip_uuid'] == 'trip-153671041653548748' ]
```

	index	segment_key	data	trip_creation_time	route_schedule_uuid	route_type	trip_uui
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip 15367104165354874
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip 15367104165354874

Further aggregation of fields based on Trip_uuid

```
create_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(create_trip_dict).reset_index(drop = True)
```

```
trip.shape
```

```
(14787, 18)
```

```
#come back later
trip[['actual_time', 'segment_actual_time_sum']]
```

	actual_time	segment_actual_time_sum
0	1562.0	1548.0
1	143.0	141.0
2	3347.0	3308.0
3	59.0	59.0
4	341.0	340.0

Feature Creation

```
trip['source_name'] = trip['source_name'].str.lower() # lowering all columns
trip['destination_name'] = trip['destination_name']
```

```
def place2state(x):
    # transform "gurgaon_bilaspur_hb (haryana)" into "haryana"
    state = x.split('(')[1]

    return state[:-1] #removing ')' from ending

def place2city(x):
    #we will remove state
    city = x.split(' (')[0]

    city = city.split('_')[0]

    # Now daling with edge cases

    if city == 'pnq vadgaon sheri dpc': return 'vadgaonsheri'
    # ['PNQ Pashan DPC', 'Bhopal MP Nagar', 'HBR Layout PC',
    #  'PNQ Rahatani DPC', 'Pune Balaji Nagar', 'Mumbai Antop Hill']

    if city in ['pnq pashan dpc','pnq rahatani dpc', 'pune balaji nagar']:
        return 'pune'

    if city == 'hbr layout pc' :
        return 'bengaluru'
    if city == 'bhopal mp nagar':
        return 'bhopal'
    if city == 'mumbai antop hill':
        return 'mumbai'

    return city
def place2city_place(x):

    # we will remove state
    x = x.split('(')[0]

    len_ = len(x.split('_'))

    if len_ >= 3:
        return x.split('_')[1]

    # small cities have same city and place name
    if len_ == 2:
        return x.split('_')[0]

    # now we need to deal with edge cases or imporper name convention

    # if len(x.split('_')) == 2:

    return x.split(' ')[0]

def place2code(x):
    # we will remove state
    x = x.split('(')[0]

    if len(x.split('_')) >= 3:
        return x.split('_')[-1]

    return 'none'
```

```
trip['source_state'] = trip['source_name'].apply(lambda x: place2state(x))
trip['source_city']  = trip['source_name'].apply(lambda x: place2city(x))
trip['source_place'] = trip['source_name'].apply(lambda x: place2city_place(x))
trip['source_code']  = trip['source_name'].apply(lambda x: place2code(x))
```

ChatGPT

```
trip[['source_state','source_city','source_place','source_code']]
```

	source_state	source_city	source_place	source_code
0	Uttar Pradesh	Kanpur	Central	6
1	Karnataka	Doddablpur	ChikaDPP	D
2	Haryana	Gurgaon	Bilaspur	HB
3	Maharashtra	Mumbai Hub	Mumbai	none
4	Karnataka	Bellary	Bellary	none
...	...	...	...	...
14782	Punjab	Chandigarh	Mehmdpur	H
14783	Haryana	FBD	Balabhgarh	DPC
14784	Uttar Pradesh	Kanpur	GovndNgr	DC
14785	Tamil Nadu	Tirunelveli	VdkkuSrt	I
14786	Karnataka	Sandur	WrdN1DPP	D

14787 rows × 4 columns

```
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
...	...	...	...	...	...	...
14782	2018	10	23	3	40	2
14783	2018	10	23	3	40	2
14784	2018	10	23	3	40	2
14785	2018	10	23	3	40	2
14786	2018	10	23	3	40	2

14787 rows × 6 columns

Outlier detection & treatment

```
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']
```

```
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.shape
```

```
(12723, 22)
```

> Handling categorical values

```
trip['route_type'].value_counts()
```

```
Carting    8812
FTL        3911
Name: route_type, dtype: int64
```

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

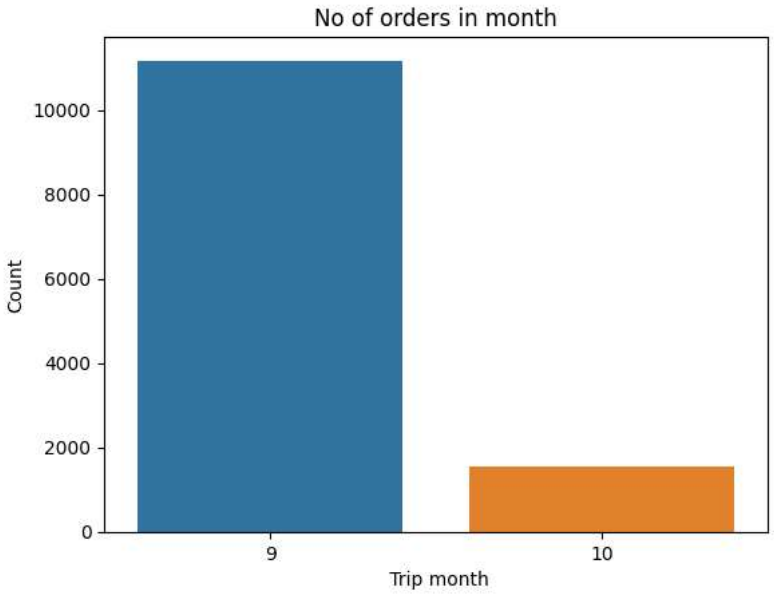
> **Visual Analysis (distribution plots of all the continuous variable(s), boxplots of all the categorical variables)**

```
trip.head(5)
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_ce
0	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	1	trip-153671042288605164	IND561203AAB	doddablpur_chikadpp_d (karnataka)	IND56120
1	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	1	trip-153671046011330457	IND400072AAB	mumbai hub (maharashtra)	IND40110
2	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	0	trip-153671052974046625	IND583101AAA	bellary_dc (karnataka)	IND58311
3	training	2018-09-12 00:02:34.161600	thanos::sroute:9bf03170-d0a2-4a3f-aa4d-9aaab3d...	1	trip-153671055416136166	IND600056AAA	chennai_poonamallee (tamil nadu)	IND60005
4	training	2018-09-12 00:04:22.011653	thanos::sroute:a97698cc-846e-41a7-916b-88b1741...	1	trip-153671066201138152	IND600044AAD	chennai_chrompet_dpc (tamil nadu)	IND60004

```
# Count of types of Route type
sns.countplot(data=trip,x='route_type')
plt.xticks(ticks=[0, 1], labels=['FTP', 'Carring'])
plt.show()
```

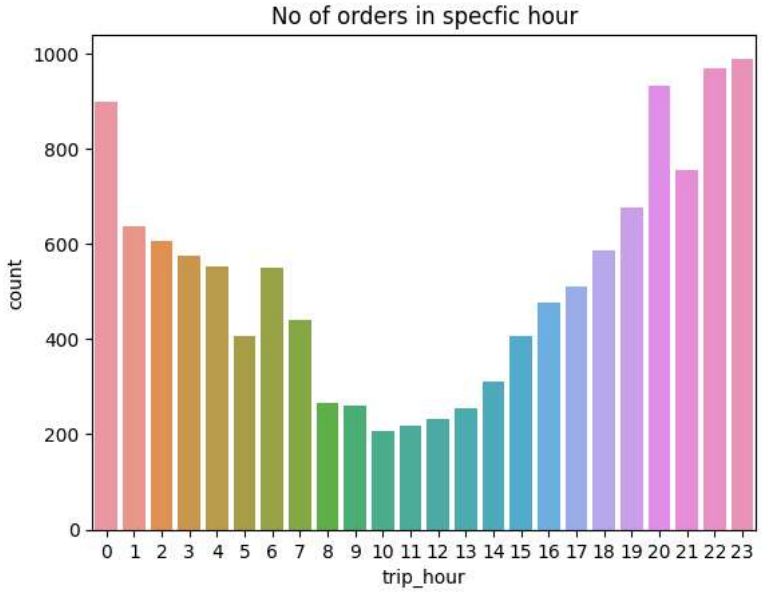

```
# Month in which most orders were created
sns.countplot(data=trip,x='trip_month')
plt.xlabel(' Trip month')
plt.ylabel('Count')
plt.title('No of orders in month')
plt.show()
```



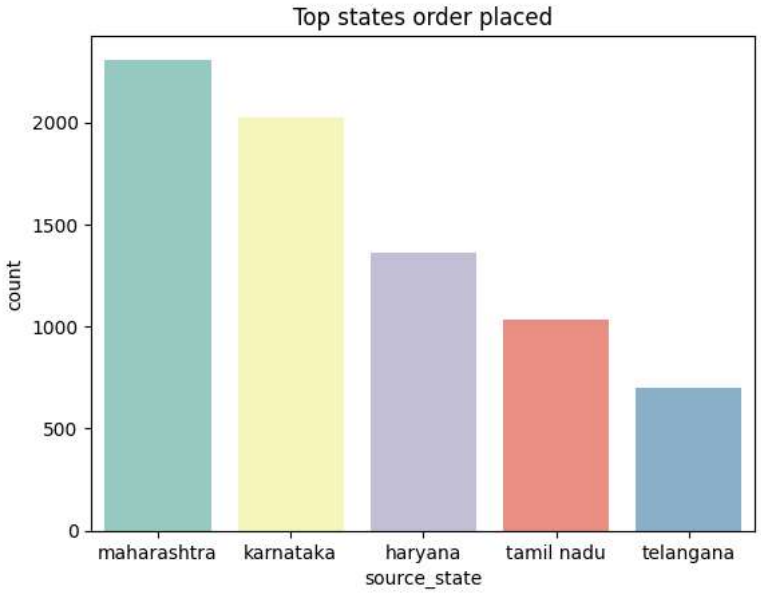

```
trip.value_counts('trip_month')
```

```
trip_month
9     11172
10     1551
dtype: int64
```

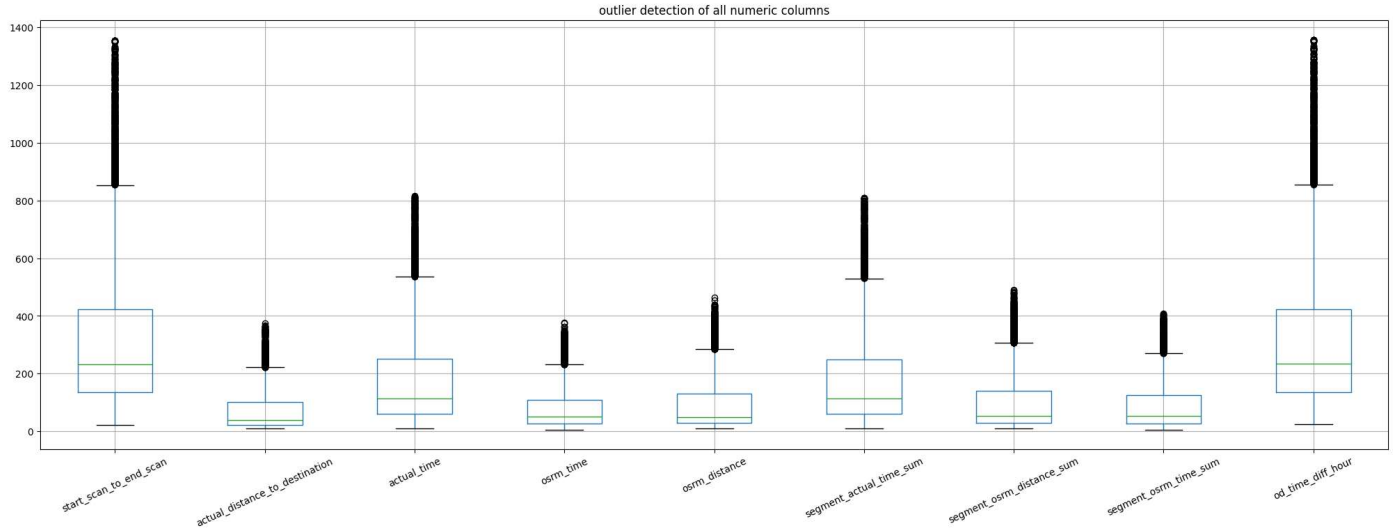
```
# Hour which most orders placed
sns.countplot(data=trip,x='trip_hour')
plt.title('No of orders in specfic hour')
plt.show()
```



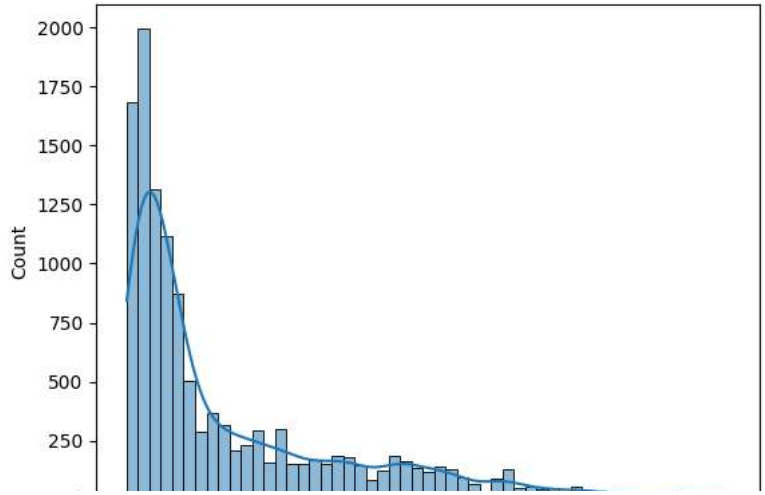
```
# states where most oders placed
categorical_column = "source_state"
top_categories = trip[categorical_column].value_counts().head(5).index

sns.countplot(data=trip[trip[categorical_column].isin(top_categories)],
              x=categorical_column,
              order=top_categories,
              palette='Set3')
plt.title('Top states order placed')
plt.show()
```



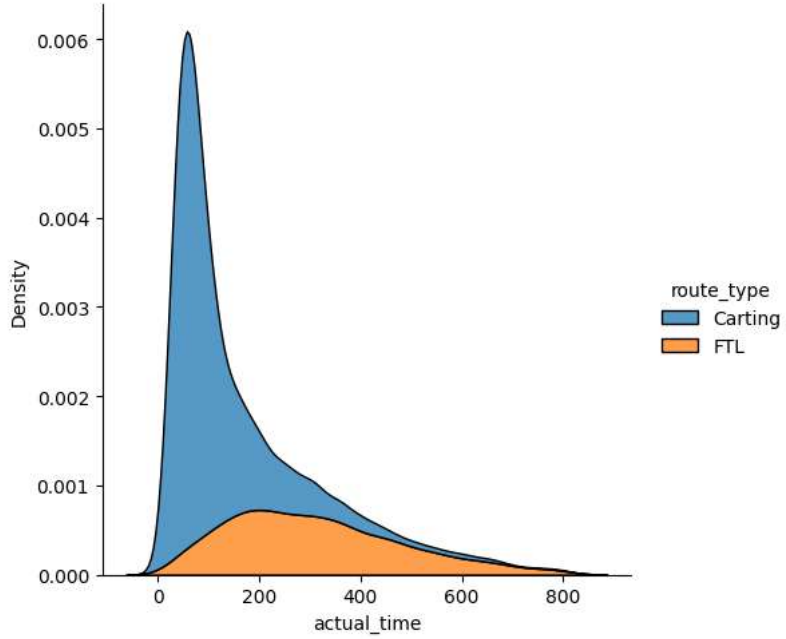

```
trip[num_cols].boxplot(rot=25, figsize=(25,8))
plt.title('outlier detection of all numeric columns')
plt.show()
```



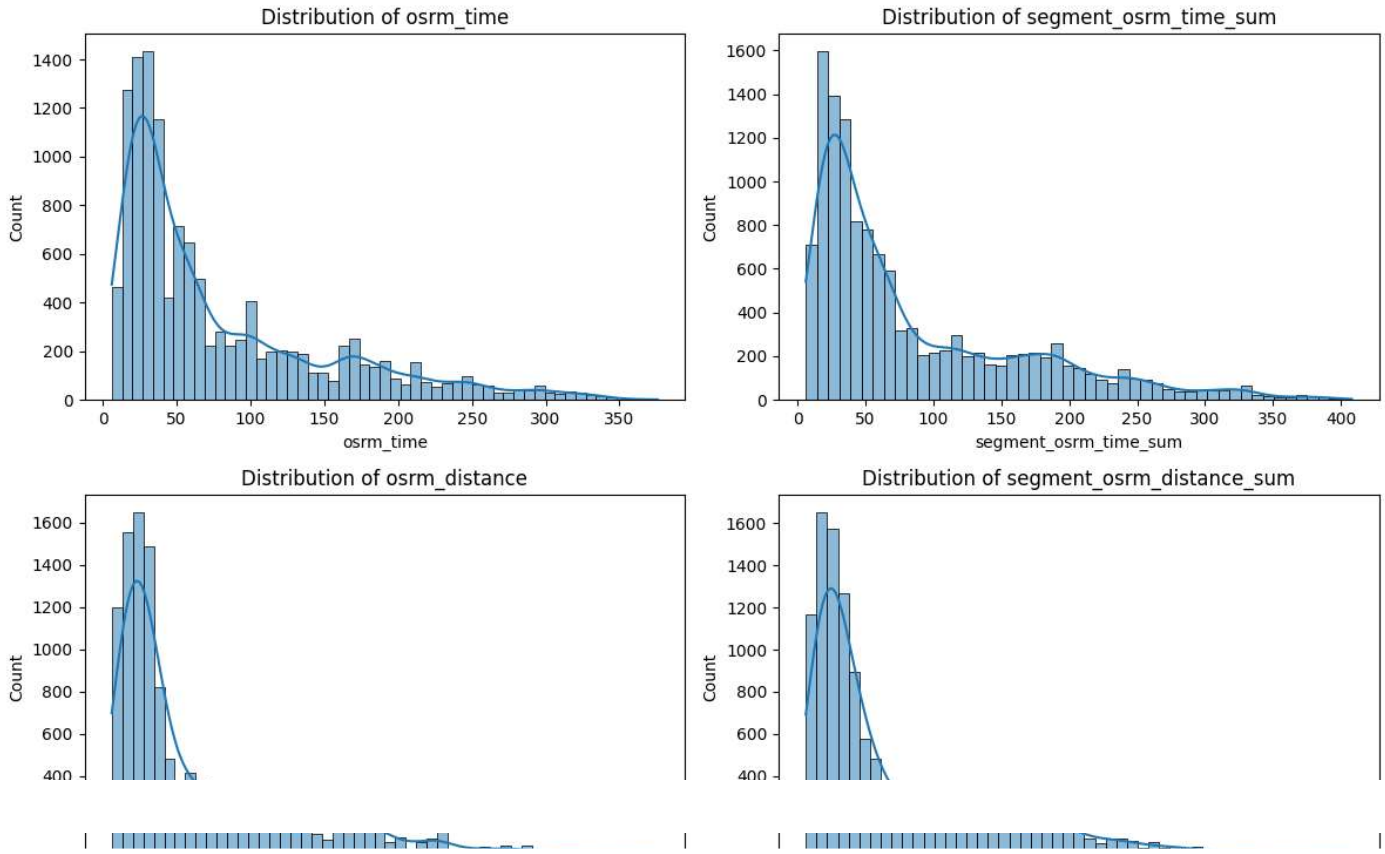

```
sns.histplot(trip['actual_distance_to_destination'], kde=True)
plt.show()
```




```
sns.displot(trip, x="actual_time", hue="route_type", kind="kde", multiple="stack")
plt.show()
```

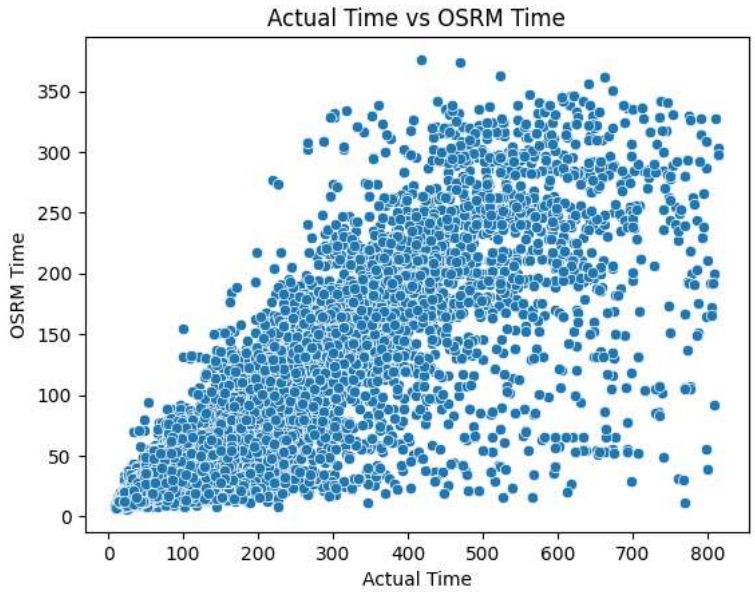



```
# distribution plots of all the continuous variable


plt.figure(figsize=(12, 8))

# Distribution plots for continuous variables
continuous_vars = ["osrm_time", "segment_osrm_time_sum", "osrm_distance", "segment_osrm_distance_sum"]
for var in continuous_vars:
    plt.subplot(2, 2, continuous_vars.index(var) + 1)
    sns.histplot(trip[var], kde=True)
    plt.title(f"Distribution of {var}")

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



> Checking relationship between aggregated fields

```
# Create a scatter plot to visualize the relationship between actual_time and OSRM_time

sns.scatterplot(data=trip, x="actual_time", y="osrm_time")
plt.xlabel('Actual Time')
plt.ylabel('OSRM Time')
plt.title('Actual Time vs OSRM Time')
plt.show()
```

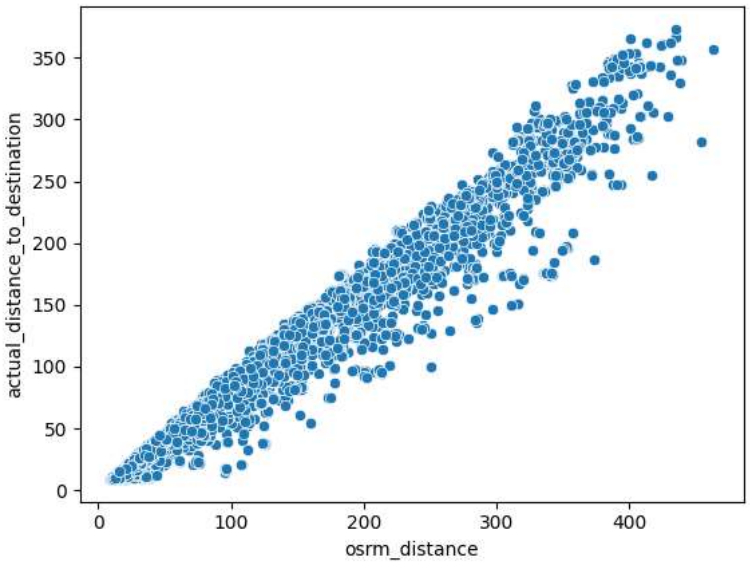



```
# Perform a two-sample t-test

# H0 :  Means are not significantly different.
# HA : Means are significantly different.
```

```
test_stat, p_value = ttest_ind(trip['actual_time'], trip['osrm_time'])
print(f'P-Value: {p_value}')

# Set the significance level
alpha = 0.05

# Check if the p-value is less than the significance level
if p_value < alpha:
    print('Reject the null hypothesis: Means are significantly different.')
else:
    print('Fail to reject the null hypothesis: Means are not significantly different.')
```

```
 P-Value: 0.0
 Reject the null hypothesis: Means are significantly different.
```

```
sns.scatterplot(data=trip, x="osrm_distance", y="actual_distance_to_destination")
plt.show()
```




```
# actual_distance_to_destination VS osrm_distance

# H0 :  Means are not significantly different.
# HA : Means are significantly different.

test_stat, p_value = ttest_ind(trip['actual_distance_to_destination'], trip['osrm_distance'])
print(f'P-Value: {p_value}')

# Set the significance level
alpha = 0.05

# Check if the p-value is less than the significance level
if p_value < alpha:
    print('Reject the null hypothesis: Means are significantly different.')
else:
    print('Fail to reject the null hypothesis: Means are not significantly different.')
```

```
 P-Value: 2.3652203422426466e-80
 Reject the null hypothesis: Means are significantly different.
```

```
# perform ANOVA


# H0 :  Means are not significantly different.
# HA : Means are significantly different.


f_oneway(trip['actual_distance_to_destination'], trip['start_scan_to_end_scan'],trip['actual_time'] ,trip['osrm_time'])

alpha = 0.05
if p_value < alpha:
    print('Reject the null hypothesis: Means are significantly different.')
else:
    print('Fail to reject the null hypothesis: Means are not significantly different.')
```

```
 Reject the null hypothesis: Means are significantly different.
```

```
# perform chi2_contingency

time = pd.crosstab(index=trip['actual_distance_to_destination'],columns=trip['start_scan_to_end_scan'])
time
```

start_scan_to_end_scan	23.0	26.0	27.0	28.0	29.0	30.0	31.0	32.0	33.0	34.0	35.0	36.0	37.0	38.0	39.0	40.0	41.0	42.0
actual_distance_to_destination																		
9.002461	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9.003578	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9.004038	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9.006255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9.006827	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
362.783667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
362.886493	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
365.945343	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
366.454138	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
373.441224	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12707 rows × 1171 columns

```
# H0 :  No association.
# HA :  Association btw actual_distance_to_destination and start_scan_to_end_scan.


chi2_contingency(time)
print(p_value)

alpha = 0.05
if p_value < alpha:
    print('No association.')
else:
    print('Association btw actual_distance_to_destination and start_scan_to_end_scan')
```

```
 2.3652203422426466e-80
 No association.
```

Normalize/Standarize the 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	
0	-0.548546	0.012060	-0.217856	-0.144341	
1	-0.861602	-0.765152	-0.749015	-0.877085	
2	1.552838	0.764988	1.034163	0.533102	
3	-0.513328	-0.662169	-0.736369	-0.766482	
4	-0.869428	-0.877197	-0.970332	-0.904736	
...	...	...	...	...	
12718	-0.247231	-0.201970	-0.597255	-0.227293	
12719	-1.018130	-0.788207	-0.989302	-0.918561	
12720	0.394533	-0.466688	0.661086	-0.420848	
12721	0.104957	0.865940	0.547267	1.390274	
12722	0.128436	-0.086534	0.616823	-0.144341	

12723 rows × 9 columns

Insights :

- The estimated time shows less time when compared to actual time.
- There are more Carting orders(small carts) comapared to FTL(full truck load)
- Most no of orders were placed in the 9th month(september)
- The pleak hrs where most orders placed are between 8pm to 12am
- Maharastra & Karnataka are the top 2 states where Delhivery is most active

Recommendations :

- We could show " Delivery time might be affected due to traffic " to show increase in estimated time.
- We could have more riders in peak states to deliver more with ease and increase our presence for more business.
- We require more data , not just 2 months (9th & 10th of 2018) to get more accurate findings and insights.