In [1921]:

```
import numpy as np
import pandas as pd
from scipy import stats

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

ProblemStatement : To create and Analyse new features in Losgistics data set then Provide recomendations & Insights Observed through Data

In [1922]:

```
df = pd.read_csv("C:\\Users\\bolla\\OneDrive\\Desktop\\Notes\\delhivery_data.txt")
df.head(50)
```

Out[1922]:

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

In [1923]:

```
df.shape
```

Out[1923]:

```
(144867, 24)
```

The Given data has 144867 Rows and 24 Columns

In [1924]:

```
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  
```

In [1925]:

```
df.data.unique()
```

Out[1925]:

```
array(['training', 'test'], dtype=object)
```

The given data has two kinds of data that Training and test data

In [1926]:

```
print("No of rows in  training data : ",df[df["data"]=="training"].data.count())

print("No of rows in  test data : ",df[df["data"]=="test"].data.count())
```

```
No of rows in  training data :  104858
No of rows in  test data :  40009
```

We can see that majority of the data is training data

In [1927]:

```
df = df.drop(["is_cutoff","cutoff_factor","cutoff_timestamp","factor","segment_factor"], axis=1)
```

Removal of all columns that are Unknown field

In [1928]:

```
df
```

Out[1928]:

	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	Anand
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand
...	...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sor
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sor
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sor
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sor
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sor

144867 rows × 19 columns

In [1929]:

```
time_val = df.groupby(["trip_uuid","source_name","destination_name"]).actual_time.last()
time_val = pd.DataFrame(time_val)
time_val = time_val.reset_index()

actual_time_val = time_val.groupby("trip_uuid").actual_time.sum()
actual_time_val = pd.DataFrame(actual_time_val)
actual_time_val = actual_time_val.reset_index()
actual_time_val
```

Out[1929]:

	trip_uuid	actual_time
0	trip-153671041653548748	1562.0
1	trip-153671042288605164	143.0
2	trip-153671043369099517	3347.0
3	trip-153671046011330457	59.0
4	trip-153671052974046625	341.0
...	...	...
14782	trip-153861095625827784	83.0
14783	trip-153861104386292051	21.0
14784	trip-153861106442901555	282.0
14785	trip-153861115439069069	264.0
14786	trip-153861118270144424	275.0

14787 rows × 2 columns

In [1930]:

```
osrmtime_val = df.groupby(["trip_uuid","source_name","destination_name"]).osrm_time.last()
osrmtime_val = pd.DataFrame(osrmtime_val)
osrmtime_val = osrmtime_val.reset_index()

osrm_time_val = osrmtime_val.groupby("trip_uuid").osrm_time.sum()
osrm_time_val = pd.DataFrame(osrm_time_val)
osrm_time_val = osrm_time_val.reset_index()
osrm_time_val
```

Out[1930]:

	trip_uuid	osrm_time
0	trip-153671041653548748	717.0
1	trip-153671042288605164	68.0
2	trip-153671043369099517	1740.0
3	trip-153671046011330457	15.0
4	trip-153671052974046625	117.0
...	...	...
14782	trip-153861095625827784	62.0
14783	trip-153861104386292051	12.0
14784	trip-153861106442901555	48.0
14785	trip-153861115439069069	179.0
14786	trip-153861118270144424	68.0

14787 rows × 2 columns

In [1931]:

```
seg_actual_time_val = df.groupby(["trip_uuid","source_name","destination_name"]).segment_actual_time.sum(
seg_actual_time_val = pd.DataFrame(seg_actual_time_val)
seg_actual_time_val = seg_actual_time_val.reset_index()

segment_actual_time_val = seg_actual_time_val.groupby("trip_uuid").segment_actual_time.sum()
segment_actual_time_val = pd.DataFrame(segment_actual_time_val)
segment_actual_time_val = segment_actual_time_val.reset_index()
segment_actual_time_val
```

Out[1931]:

	trip_uuid	segment_actual_time
0	trip-153671041653548748	1548.0
1	trip-153671042288605164	141.0
2	trip-153671043369099517	3308.0
3	trip-153671046011330457	59.0
4	trip-153671052974046625	340.0
...	...	...
14782	trip-153861095625827784	82.0
14783	trip-153861104386292051	21.0
14784	trip-153861106442901555	281.0
14785	trip-153861115439069069	258.0
14786	trip-153861118270144424	274.0

14787 rows × 2 columns

In [1932]:

```
osrm_dst_val = df.groupby(["trip_uuid","source_name","destination_name"]).osrm_distance.last()
osrm_dst_val = pd.DataFrame(osrm_dst_val)
osrm_dst_val = osrm_dst_val.reset_index()

osrm_distance_val = osrm_dst_val.groupby("trip_uuid").osrm_distance.sum()
osrm_distance_val = pd.DataFrame(osrm_distance_val)
osrm_distance_val = osrm_distance_val.reset_index()
osrm_distance_val
```

Out[1932]:

	trip_uuid	osrm_distance
0	trip-153671041653548748	991.3523
1	trip-153671042288605164	85.1110
2	trip-153671043369099517	2354.0665
3	trip-153671046011330457	19.6800
4	trip-153671052974046625	146.7918
...	...	...
14782	trip-153861095625827784	73.4630
14783	trip-153861104386292051	16.0882
14784	trip-153861106442901555	58.9037
14785	trip-153861115439069069	171.1103
14786	trip-153861118270144424	80.5787

14787 rows × 2 columns

In [1933]:

```
seg_osrm_dst_val = df.groupby(["trip_uuid","source_name","destination_name"]).segment_osrm_distance.sum()
seg_osrm_dst_val = pd.DataFrame(seg_osrm_dst_val)
seg_osrm_dst_val = seg_osrm_dst_val.reset_index()

segment_osrm_distance_val = seg_osrm_dst_val.groupby("trip_uuid").segment_osrm_distance.sum()
segment_osrm_distance_val = pd.DataFrame(segment_osrm_distance_val)
segment_osrm_distance_val = segment_osrm_distance_val.reset_index()
segment_osrm_distance_val
```

Out[1933]:

	trip_uuid	segment_osrm_distance
0	trip-153671041653548748	1320.4733
1	trip-153671042288605164	84.1894
2	trip-153671043369099517	2545.2678
3	trip-153671046011330457	19.8766
4	trip-153671052974046625	146.7919
...	...	...
14782	trip-153861095625827784	64.8551
14783	trip-153861104386292051	16.0883
14784	trip-153861106442901555	104.8866
14785	trip-153861115439069069	223.5324
14786	trip-153861118270144424	80.5787

14787 rows × 2 columns

In [1934]:

```
seg_osrm_time_val = df.groupby(["trip_uuid","source_name","destination_name"]).segment_osrm_time.sum()
seg_osrm_time_val = pd.DataFrame(seg_osrm_time_val)
seg_osrm_time_val = seg_osrm_time_val.reset_index()

segment_osrm_time_val = seg_osrm_time_val.groupby("trip_uuid").segment_osrm_time.sum()
segment_osrm_time_val = pd.DataFrame(segment_osrm_time_val)
segment_osrm_time_val = segment_osrm_time_val.reset_index()
segment_osrm_time_val
```

Out[1934]:

	trip_uuid	segment_osrm_time
0	trip-153671041653548748	1008.0
1	trip-153671042288605164	65.0
2	trip-153671043369099517	1941.0
3	trip-153671046011330457	16.0
4	trip-153671052974046625	115.0
...	...	...
14782	trip-153861095625827784	62.0
14783	trip-153861104386292051	11.0
14784	trip-153861106442901555	88.0
14785	trip-153861115439069069	221.0

Each Trip_uuid has more than one stops or relay points in its journey to reach from source to destination , hence group-by and Aggregate functions should be used careful i,e while grouping ["trip_uuid","source_name","destination_name"] use “last()” as we have multiple sources and destinations for each trip and while grouping ("trip_uuid") use sum() , also while using the grouping of

segmented values we can directly use sum() as each value is individual recording .

In [1935]:

```
condensed_df = df.copy()
```

In [1936]:

```
condensed_df = pd.merge(condensed_df,actual_time_val, on = "trip_uuid", how="right")
condensed_df = pd.merge(condensed_df,osrm_time_val, on = "trip_uuid", how="right")
condensed_df = pd.merge(condensed_df,segment_actual_time_val, on = "trip_uuid", how="right")
condensed_df = pd.merge(condensed_df,osrm_distance_val, on = "trip_uuid", how="right")
condensed_df = pd.merge(condensed_df,segment_osrm_distance_val, on = "trip_uuid", how="right")
condensed_df = pd.merge(condensed_df,segment_osrm_time_val, on = "trip_uuid", how="right")

condensed_df
```

srm_time_x	segment_osrm_distance_x	actual_time_y	osrm_time_y	segment_actual_time_y	osrm_distance_y	segment_osrm_dist
39.0	55.2597	1562.0	717.0	1548.0	991.3523	132
52.0	73.8647	1562.0	717.0	1548.0	991.3523	132
16.0	23.0634	1562.0	717.0	1548.0	991.3523	132
15.0	21.4162	1562.0	717.0	1548.0	991.3523	132

In [ ]:

```
```

In [1937]:

```
condensed_df = condensed_df.drop(["route_schedule_uuid","destination_center","source_center"], axis=1)
```

In [1938]:

```
condensed_df.isna().sum()/len(df)*100
```

Out[1938]:

```
data                              0.000000
trip_creation_time                0.000000
route_type                        0.000000
trip_uuid                         0.000000
source_name                       0.133916
destination_name                  0.107685
od_start_time                     0.000000
od_end_time                       0.000000
start_scan_to_end_scan            0.000000
actual_distance_to_destination    0.000000
actual_time_x                     0.000000
osrm_time_x                       0.000000
osrm_distance_x                   0.000000
segment_actual_time_x             0.000000
segment_osrm_time_x               0.000000
segment_osrm_distance_x           0.000000
actual_time_y                     0.000000
osrm_time_y                       0.000000
segment_actual_time_y             0.000000
osrm_distance_y                   0.000000
segment_osrm_distance_y           0.000000
segment_osrm_time_y               0.000000
dtype: float64
```

Columns Source-name and Destination-name have missing values with less than 0.2% , dropping off these rows as such minor percent of values may not affect the data.

In [1939]:

```
## Dropping missing value rows from source_name and destination_name as the missing values are less than 0

condensed_df.dropna(axis =0, how="any", inplace = True)
```

In [1940]:

```
condensed_df.isna().sum()/len(df)*100
```

Out[1940]:

```
data                              0.0
trip_creation_time                0.0
route_type                        0.0
trip_uuid                         0.0
source_name                       0.0
destination_name                  0.0
od_start_time                     0.0
od_end_time                       0.0
start_scan_to_end_scan            0.0
actual_distance_to_destination    0.0
actual_time_x                     0.0
osrm_time_x                       0.0
osrm_distance_x                   0.0
segment_actual_time_x             0.0
segment_osrm_time_x               0.0
segment_osrm_distance_x           0.0
actual_time_y                     0.0
osrm_time_y                       0.0
```

In [1941]:

```
condensed_df[["source_city","source_place","source_code&state"]] = condensed_df.source_name.str.split("_"
condensed_df[["source_code","source_state"]] = condensed_df["source_code&state"].str.split(" ", n=1, expa


condensed_df[["destination_city","destination_place",
              "destination_code&state"]] = condensed_df.destination_name.str.split("_", n=2, expand = Tru
condensed_df[["destination_code","destination_state"]] = condensed_df["destination_code&state"].str.split


condensed_df.drop(["source_code&state","destination_code&state"],axis = 1, inplace = True)
condensed_df
```

Out[1941]:

	data	trip_creation_time	route_type	trip_uuid	source_name	destination_name	
0	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	Bhopal_Trnsport_H (Madhya Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	C
1	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	Bhopal_Trnsport_H (Madhya Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	C
2	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	Bhopal_Trnsport_H (Madhya Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	C
3	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	Bhopal_Trnsport_H (Madhya Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	C
4	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	Bhopal_Trnsport_H (Madhya Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	C
...	...	...	...	...	...	...	
144661	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	Peikulam_SriVnktpm_D (Tamil Nadu)	Tirunelveli_VdkkuSrt_I (Tamil Nadu)	C
144662	test	2018-10-03 23:59:42.701692	FTL	trip-153861118270144424	Hospet (Karnataka)	Sandur_WrdN1DPP_D (Karnataka)	C
144663	test	2018-10-03 23:59:42.701692	FTL	trip-153861118270144424	Hospet (Karnataka)	Sandur_WrdN1DPP_D (Karnataka)	C
144664	test	2018-10-03 23:59:42.701692	FTL	trip-153861118270144424	Sandur_WrdN1DPP_D (Karnataka)	Bellary_Dc (Karnataka)	C
144665	test	2018-10-03 23:59:42.701692	FTL	trip-153861118270144424	Sandur_WrdN1DPP_D (Karnataka)	Bellary_Dc (Karnataka)	C

144316 rows × 30 columns

In [1942]:

```
condensed_df.isna().sum()/len(df)*100
```

Out[1942]:

```
data                                0.000000
trip_creation_time                  0.000000
route_type                          0.000000
trip_uuid                           0.000000
source_name                         0.000000
destination_name                    0.000000
od_start_time                       0.000000
od_end_time                         0.000000
start_scan_to_end_scan              0.000000
actual_distance_to_destination      0.000000
actual_time_x                       0.000000
osrm_time_x                         0.000000
osrm_distance_x                     0.000000
segment_actual_time_x               0.000000
segment_osrm_time_x                 0.000000
segment_osrm_distance_x             0.000000
actual_time_y                       0.000000
osrm_time_y                         0.000000
segment_actual_time_y               0.000000
osrm_distance_y                     0.000000
segment_osrm_distance_y             0.000000
segment_osrm_time_y                 0.000000
source_city                         0.000000
source_place                        1.448225
source_code                        10.097538
source_state                       10.097538
destination_city                    0.000000
destination_place                   1.678781
destination_code                   10.653220
destination_state                  10.653220
dtype: float64
```

After dividing Source-name and Destination-name into city, place , code , state , it is seen that the address format is not the same for all orders and missing values can be found in place -1.5% , code-10% , state-10% , filling the null values with value of column city of the same respective rows

In [1943]:

```
## Dropping missing value rows from source_name and destination_name as the missing values are less than 6

condensed_df.dropna(axis =0, how="any", inplace = True)
```

In [1944]:

```
condensed_df["source_place"].fillna(condensed_df["source_city"],inplace=True)

condensed_df["destination_place"].fillna(condensed_df["destination_city"],inplace=True)

condensed_df["source_code"].fillna(condensed_df["source_place"],inplace=True)

condensed_df["source_state"].fillna(condensed_df["source_code"],inplace=True)

condensed_df["destination_code"].fillna(condensed_df["destination_place"],inplace=True)

condensed_df["destination_state"].fillna(condensed_df["destination_code"],inplace=True)
```

In [1945]:

```
condensed_df.drop(['source_name','destination_name'], axis=1, inplace=True)

condensed_df.isna().sum()/len(df)*100
```

Out[1945]:

```
data                            0.0
trip_creation_time              0.0
route_type                      0.0
trip_uuid                       0.0
od_start_time                   0.0
od_end_time                     0.0
start_scan_to_end_scan          0.0
actual_distance_to_destination  0.0
actual_time_x                   0.0
osrm_time_x                     0.0
osrm_distance_x                 0.0
segment_actual_time_x           0.0
segment_osrm_time_x             0.0
segment_osrm_distance_x         0.0
actual_time_y                   0.0
osrm_time_y                     0.0
segment_actual_time_y           0.0
osrm_distance_y                 0.0
segment_osrm_distance_y         0.0
segment_osrm_time_y             0.0
source_city                     0.0
source_place                    0.0
source_code                     0.0
source_state                    0.0
destination_city                0.0
destination_place               0.0
destination_code                0.0
destination_state               0.0
dtype: float64
```

In [1946]:

```
condensed_df["trip_creation_time"]= condensed_df.trip_creation_time.apply(pd.to_datetime)

condensed_df["Trip_Year"]=condensed_df.trip_creation_time.dt.year
condensed_df["Trip_Month"]=condensed_df.trip_creation_time.dt.month
condensed_df["Trip_Day"]=condensed_df.trip_creation_time.dt.day
```

In [1947]:

```
print("Year :", condensed_df.Trip_Year.value_counts())
print("")
print("Month :", condensed_df.Trip_Month.value_counts())
print("")

print("Day :", condensed_df.Trip_Day.value_counts())
```

```
Year : 2018    115535
Name: Trip_Year, dtype: int64

Month : 9     101530
10     14005
Name: Trip_Month, dtype: int64

Day : 21    6089
20    5926
18    5895
25    5817
15    5804
13    5713
17    5646
26    5589
12    5577
22    5462
16    5401
19    5333
14    5311
24    5265
3     4955
27    4946
23    4920
1     4818
28    4658
29    4390
2     4232
30    3788
Name: Trip_Day, dtype: int64
```

Column trip_creation_time can be divided into Trip_Year, Trip_Month, Trip_Day , from these we can infer that data is from year 2018, month september(9) but data from days 4 to 11 are not present in data .

In [1948]:

```
import datetime as DT

condensed_df[["od_start_time","od_end_time"]]= condensed_df[["od_start_time","od_end_time"]].apply(pd.to_
condensed_df["Total_timetaken"] = ((condensed_df.od_end_time)-(condensed_df.od_start_time))
condensed_df["Total_timetaken"] = condensed_df["Total_timetaken"].apply(lambda x: x.total_seconds()/60)

condensed_df.drop(['od_end_time','od_start_time'], axis=1, inplace=True)
```

Column start_scan_to_end_scan contains time taken to deliver from source to destination in Mins

Using columns 'od_end_time','od_start_time' a new feature can be created Total time taken by checking the difference between 'od_end_time','od_start_time' .

In [1949]:

```
start_to_end= condensed_df.groupby("trip_uuid").start_scan_to_end_scan.sum()
start_to_end = pd.DataFrame(start_to_end)
start_to_end = start_to_end.reset_index()

condensed_df = pd.merge(condensed_df,start_to_end, on = "trip_uuid", how="right")

agg_Total_timetaken = condensed_df.groupby("trip_uuid").Total_timetaken.sum()
agg_Total_timetaken = pd.DataFrame(agg_Total_timetaken)
agg_Total_timetaken = agg_Total_timetaken.reset_index()

condensed_df = pd.merge(condensed_df,agg_Total_timetaken, on = "trip_uuid", how="right")
```

In [1950]:

```
condensed_df.Total_timetaken_y = condensed_df.Total_timetaken_y.round(1)
condensed_df
```

Out[1950]:

	data	trip_creation_time	route_type	trip_uuid	start_scan_to_end_scan_x	actual_distance_to_
0	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
1	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
2	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
3	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
4	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
...	...	...	...	...	...	
115530	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	45.0	
115531	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	91.0	
115532	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	91.0	
115533	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	91.0	
115534	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	91.0	

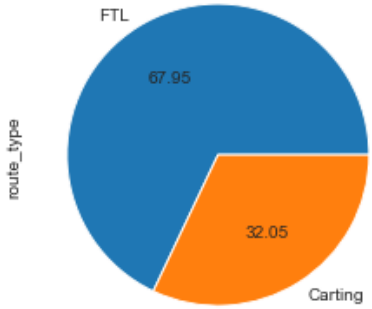
115535 rows × 32 columns

In [1951]:

```
condensed_df['route_type'].value_counts().plot(kind='pie',autopct="%.2f")
```

Out[1951]:

```
<AxesSubplot:ylabel='route_type'>
```

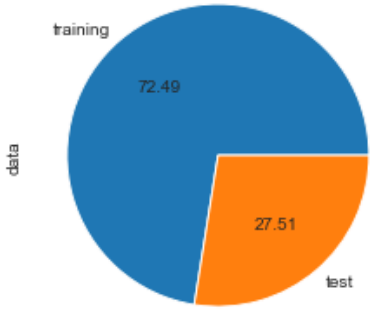


In [1952]:

```
condensed_df['data'].value_counts().plot(kind='pie',autopct="%.2f")
```

Out[1952]:

```
<AxesSubplot:ylabel='data'>
```



Before grouping the data by trip_uuid Route type FTL comprises 67.95% and Carting comprises 32.05% of the total orders .

In [1953]:

```
x_df = condensed_df.groupby("trip_uuid")[["data","route_type","source_code","destination_code","actual_ti
                                          "osrm_distance_y","source_state","destination_state","Trip_Day"
x_df = pd.DataFrame(x_df)
x_df = x_df.reset_index()

x_df
```

Out[1953]:

	trip_uuid	data	route_type	source_code	destination_code	actual_time_y	osrm_time_y	osrm
0	trip-153671041653548748	training	FTL	H_6	HB	1562.0	717.0	
1	trip-153671042288605164	training	Carting	D	D	143.0	68.0	
2	trip-153671043369099517	training	FTL	HB	H	3347.0	1740.0	
3	trip-153671066201138152	training	Carting	DPC	Dc	24.0	13.0	
4	trip-153671074033284934	training	Carting	D_12	I_4	161.0	29.0	
...	...	...	...	...	...	...	...	
11875	trip-153861089872028474	test	Carting	DC	DC	62.0	28.0	
11876	trip-153861095625827784	test	Carting	C	H	83.0	62.0	
11877	trip-153861104386292051	test	Carting	DPC	DC	21.0	12.0	
11878	trip-153861106442901555	test	Carting	DC	H_6	282.0	48.0	
11879	trip-153861115439069069	test	Carting	D	I	264.0	179.0	

11880 rows × 11 columns

In [1954]:

```
# Top 10 Cities with Highest number of trips
pd.set_option("display.max_rows", 1000)
y_df = condensed_df.groupby(["source_city","destination_city"])["actual_time_y"].count()
y_df = pd.DataFrame(y_df)
y_df = y_df.reset_index()

y_df.sort_values(by=['actual_time_y'], ascending=False , inplace = True)
y_df.head(10)
```

Out[1954]:

	source_city	destination_city	actual_time_y
669	Gurgaon	Bangalore	4976
169	Bangalore	Gurgaon	3316
687	Gurgaon	Kolkata	2862
230	Bengaluru	Bengaluru	2062
163	Bangalore	Bengaluru	1741
684	Gurgaon	Hyderabad	1639
670	Gurgaon	Bhiwandi	1617
674	Gurgaon	Delhi	1345
282	Bhiwandi	Gurgaon	1269
228	Bengaluru	Bangalore	1268

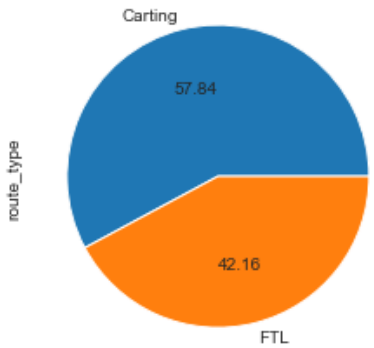
Top cities that have more number of Trips are Bangalore, Gurgaon, Kolkata, Hyderabad, Bhiwandi, Delhi.

In [1955]:

```
x_df['route_type'].value_counts().plot(kind='pie',autopct="%.2f")
```

Out[1955]:

```
<AxesSubplot:ylabel='route_type'>
```

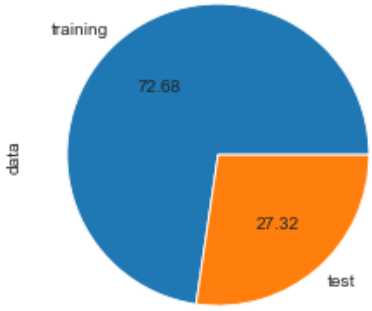


In [1956]:

```
x_df['data'].value_counts().plot(kind='pie',autopct="%.2f")
```

Out[1956]:

```
<AxesSubplot:ylabel='data'>
```



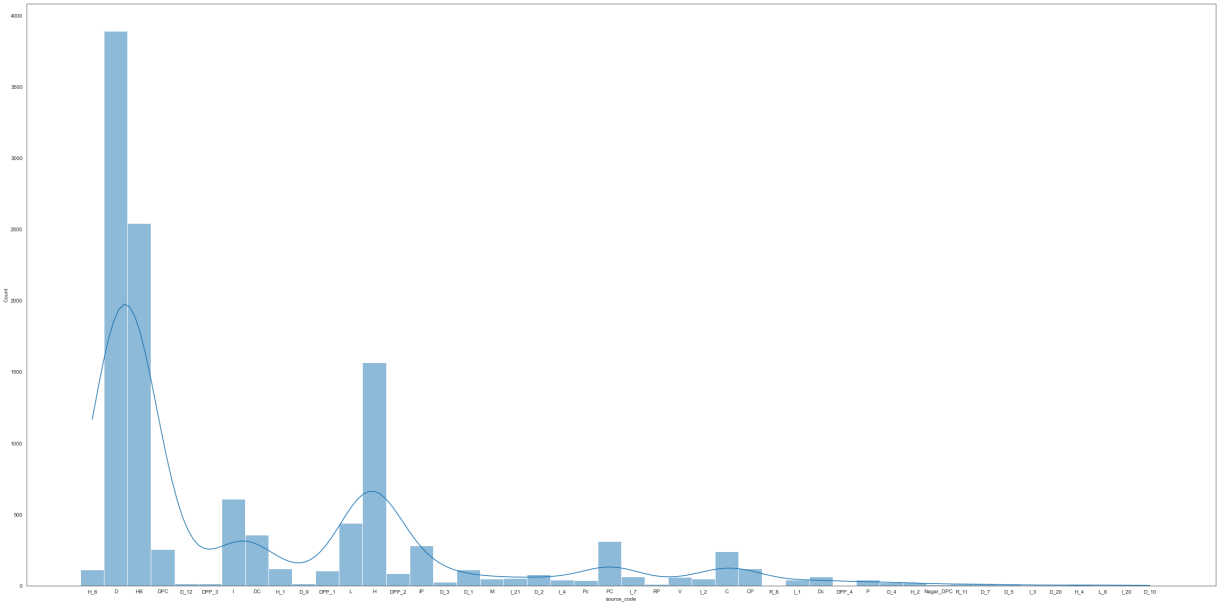
From the above difference between before and after grouping by trip_uuid we can infer that FTL type has more relay point although it is not making no other pickups or drop-offs along the way Among the given data Training data comprises 72.49% and test data comprises 27.51%

In [1957]:

```
### Quantative attributes in data are
### source_code,source_state,destination_code,destination_state,Trip_Day

fig1, ax1 = plt.subplots(figsize=(40, 20))


sns.histplot(data =x_df , x ='source_code', kde = True )
```

Out[1957]:

```
<AxesSubplot:xlabel='source_code', ylabel='Count'>
```

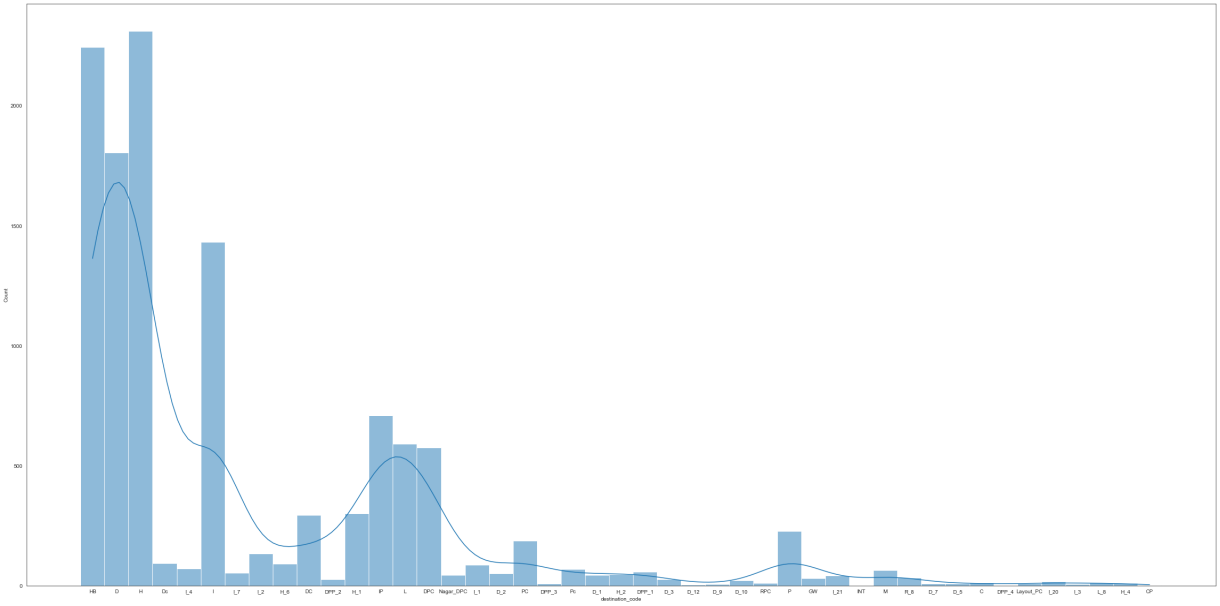


In [1958]:

```
fig1, ax1 = plt.subplots(figsize=(40, 20))


sns.histplot(data =x_df , x ='destination_code', kde = True )
```

Out[1958]:

```
<AxesSubplot:xlabel='destination_code', ylabel='Count'>
```

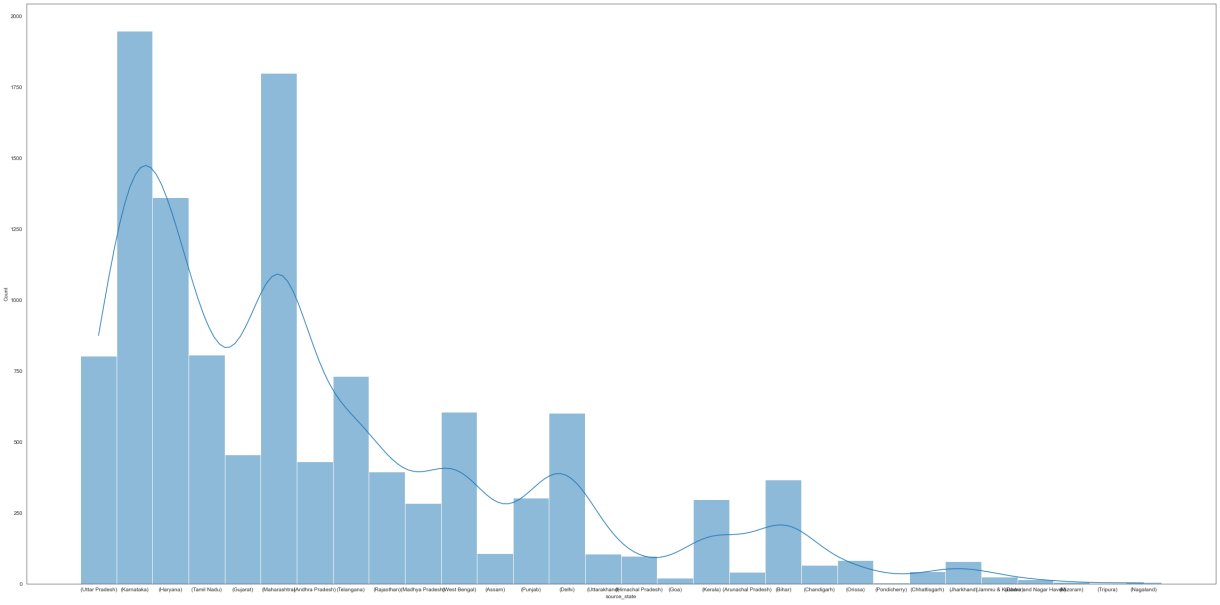


In [1959]:

```
fig1, ax1 = plt.subplots(figsize=(40, 20))


sns.histplot(data =x_df , x ='source_state', kde = True )
```

Out[1959]:

```
<AxesSubplot:xlabel='source_state', ylabel='Count'>
```

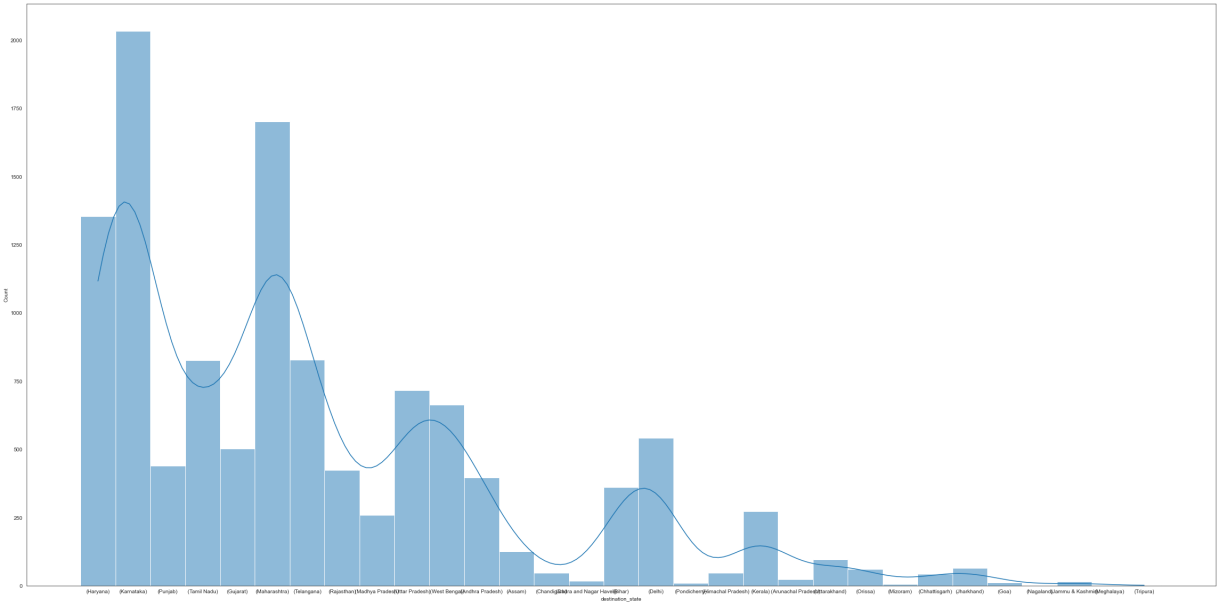


In [1960]:

```
fig1, ax1 = plt.subplots(figsize=(40, 20))


sns.histplot(data =x_df , x ='destination_state', kde = True )
```

Out[1960]:

```
<AxesSubplot:xlabel='destination_state', ylabel='Count'>
```

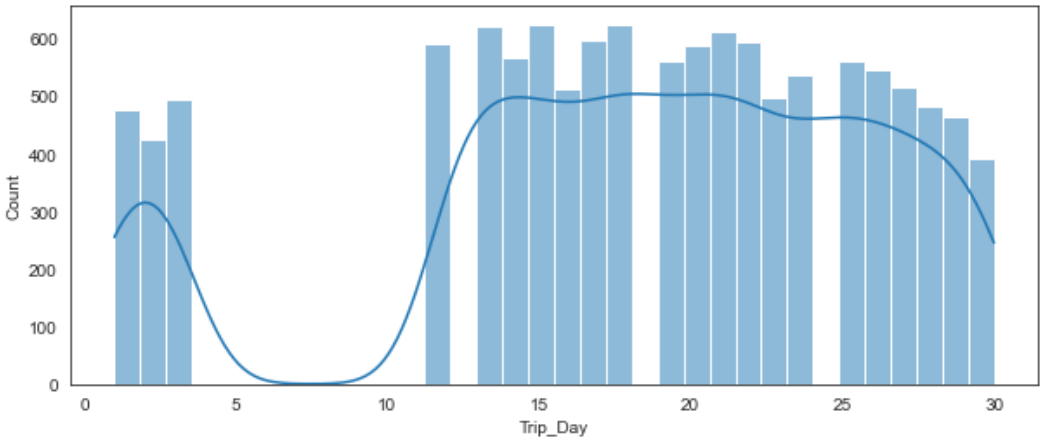


In [1961]:

```
fig1, ax1 = plt.subplots(figsize=(10, 4))


sns.histplot(data =x_df , x ='Trip_Day', kde = True )
```

Out[1961]:

```
<AxesSubplot:xlabel='Trip_Day', ylabel='Count'>
```



The most number of trips are from source code “D” & “HB” The most number of trips are to destination code “D” , “HB”,”H” & ”I” The most number of trips are from source state “Karnataka” , “Maharashtra”, “Haryana” The most number of trips are to Destination state “Karnataka” , “Maharashtra”, “Haryana” We can see that there is a slight decline in number of trips at the end of month

In [1962]:

```
print("Number of states the pacakages are being received from : ",x_df.source_state.nunique())
print("Number of states the pacakages are being received from : ",x_df.destination_state.nunique())
```

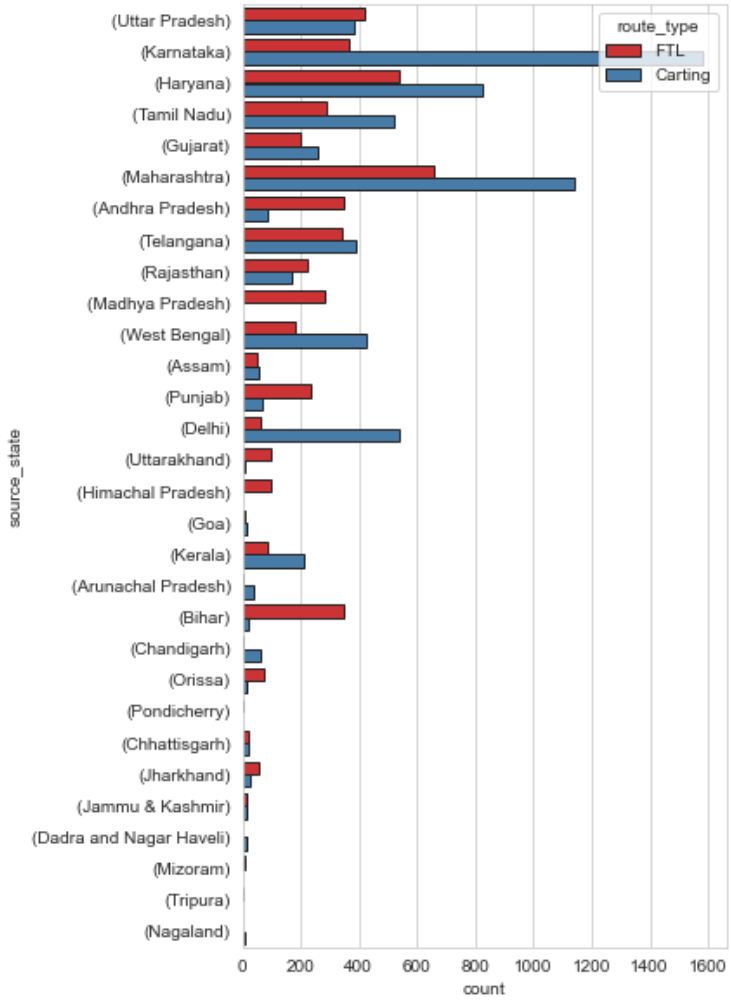
```
Number of states the pacakages are being received from :  30
Number of states the pacakages are being received from :  31
```

In [1963]:

```
sns.set_style("whitegrid")

fig1, ax1 = plt.subplots(figsize=(5, 10))


sns.countplot(data =x_df , y ='source_state',  hue = "route_type",edgecolor="0.15", palette='Set1')
```

Out[1963]:

```
<AxesSubplot:xlabel='count', ylabel='source_state'>
```

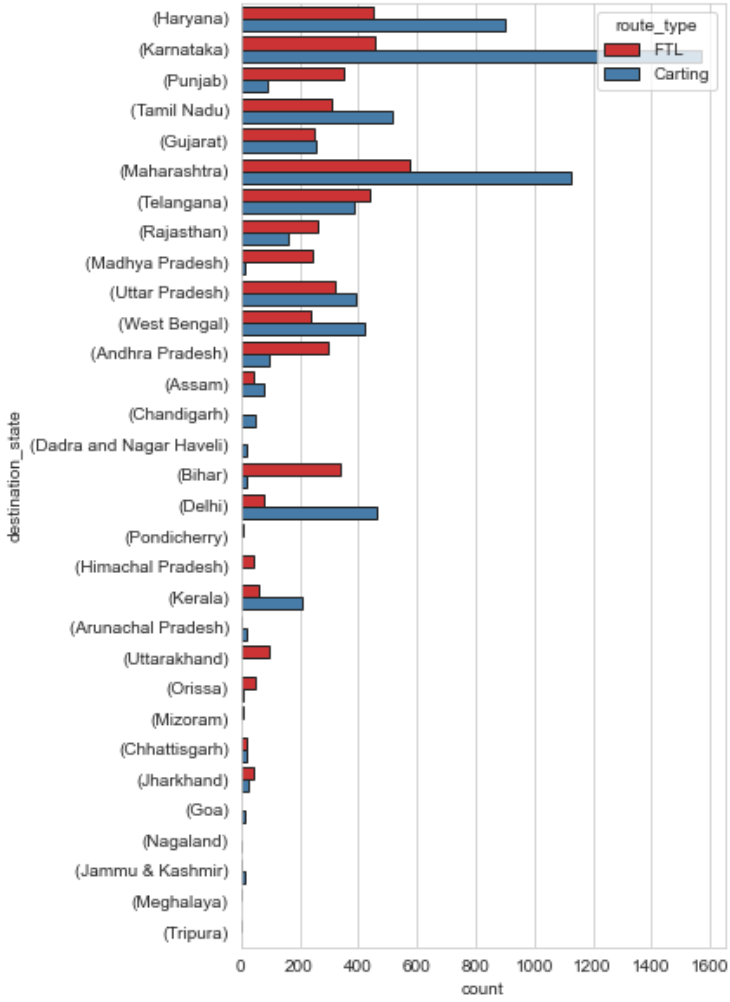


In [1964]:

```
fig1, ax1 = plt.subplots(figsize=(5, 10))


sns.countplot(data =x_df , y ='destination_state',  hue = "route_type",edgecolor="0.15", palette='Set1')
```

Out[1964]:

```
<AxesSubplot:xlabel='count', ylabel='destination_state'>
```

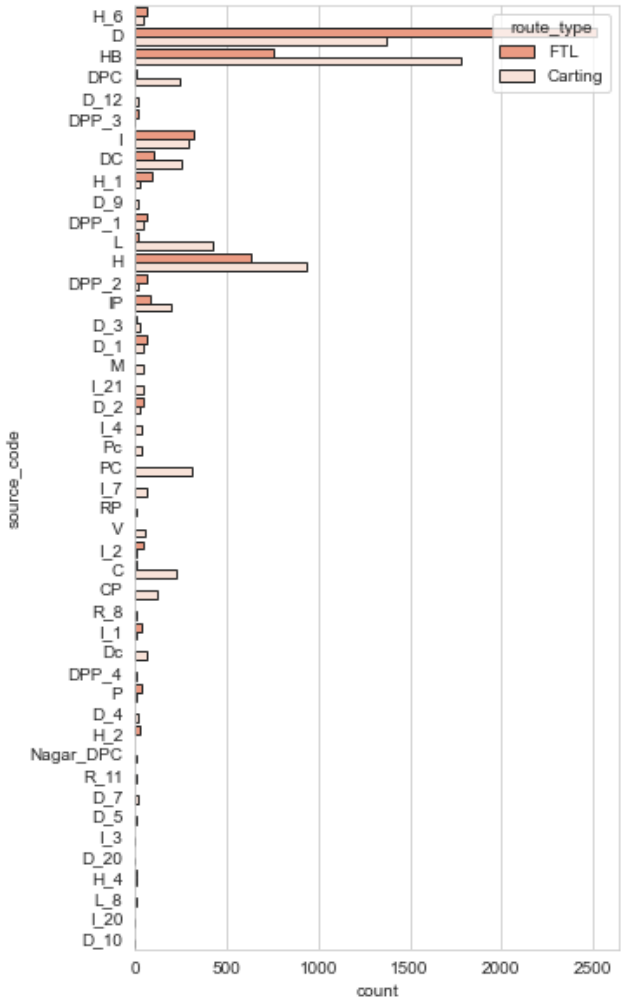


In [1965]:

```
fig1, ax1 = plt.subplots(figsize=(5, 10))


sns.countplot(data =x_df , y ='source_code',  hue = "route_type",edgecolor="0.15", palette=["#fc9272","#f
```

Out[1965]:

```
<AxesSubplot:xlabel='count', ylabel='source_code'>
```

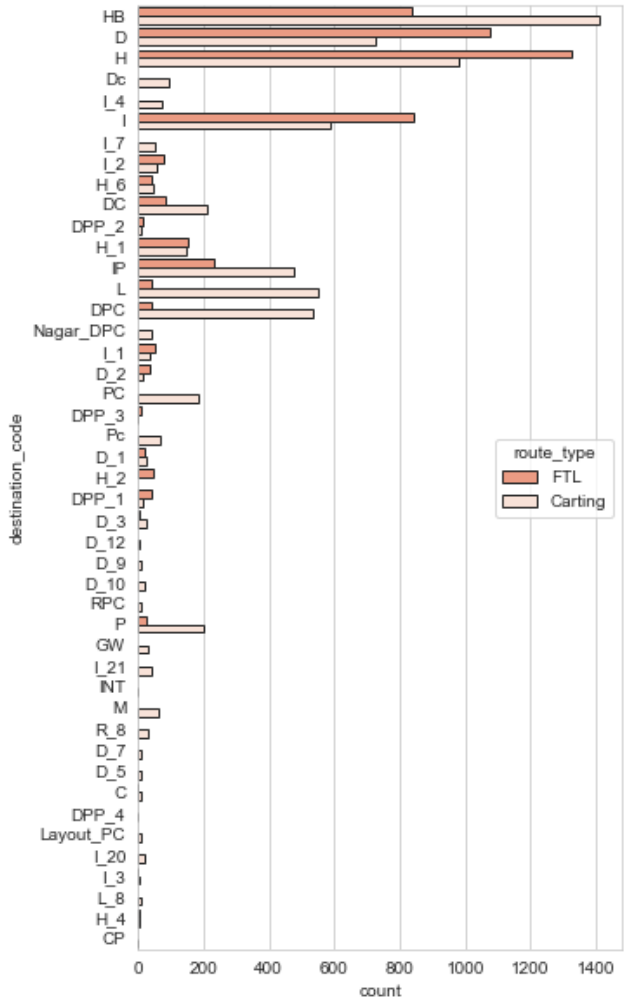


In [1966]:

```
fig1, ax1 = plt.subplots(figsize=(5, 10))


sns.countplot(data =x_df , y ='destination_code',  hue = "route_type",edgecolor="0.15", palette=["#fc9272
```

Out[1966]:

```
<AxesSubplot:xlabel='count', ylabel='destination_code'>
```



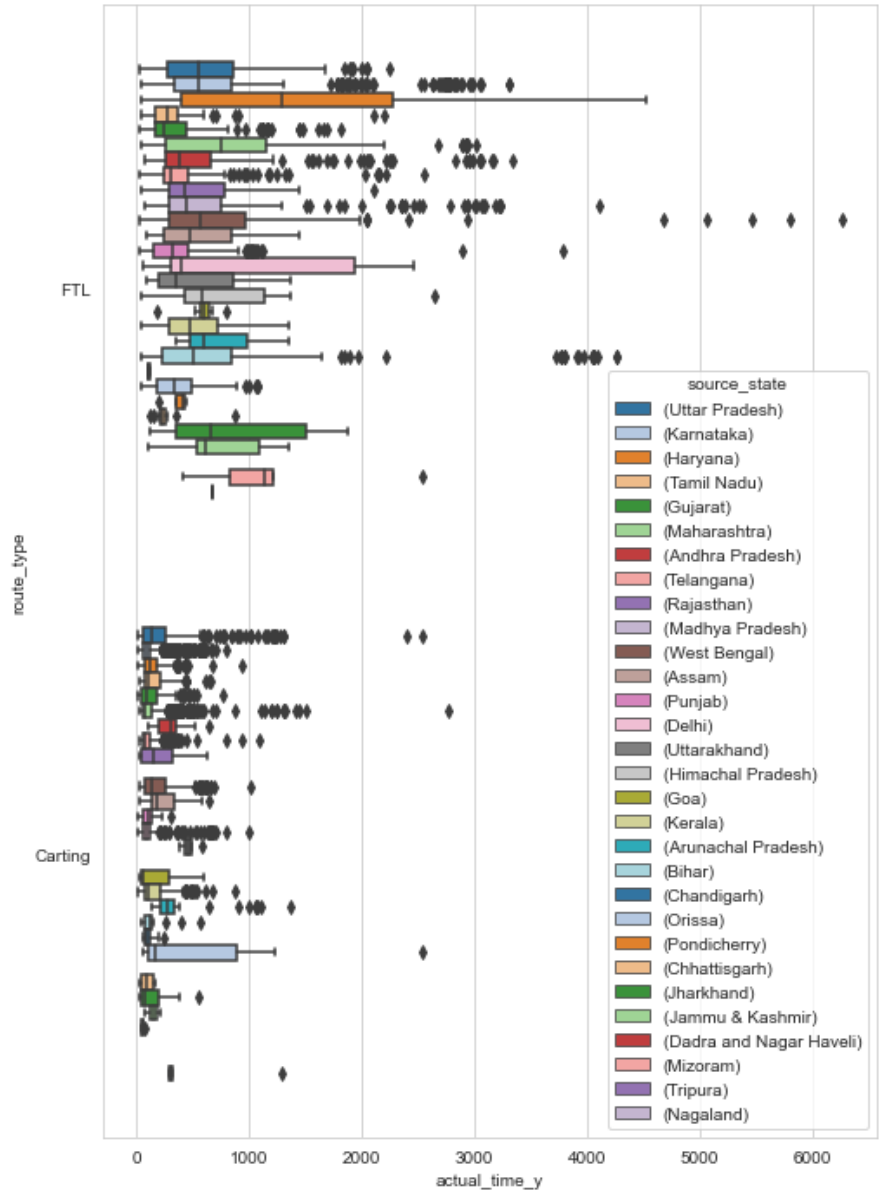
Although Karnataka has the highest number of trips from Source district still the majority of those trips come under non-dominant route-type i,e Carting , whereas Maharashtra has highest FTL type trips still which is dominated by Carting . Insights that are observed in trips from source district resemble the insights observed in trips to destination district Source codes D and HB are quiet opposite when it comes to dominance of FTL and carting, D has more number of FTL's whereas HB has more number of Carting's In Destination codes D , HB are same as source codes and H , I codes have FTL in dominance By comparing source code and destination code we can understand that these codes are again divided internally as sub-codes .

In [1967]:

```
fig1, ax1 = plt.subplots(figsize=(8,12))
custom_palette = sns.color_palette("tab20", n_colors=30)

sns.boxplot(data=x_df, x='actual_time_y', y='route_type', hue='source_state', palette=custom_palette)
```

Out[1967]:

```
<AxesSubplot:xlabel='actual_time_y', ylabel='route_type'>
```

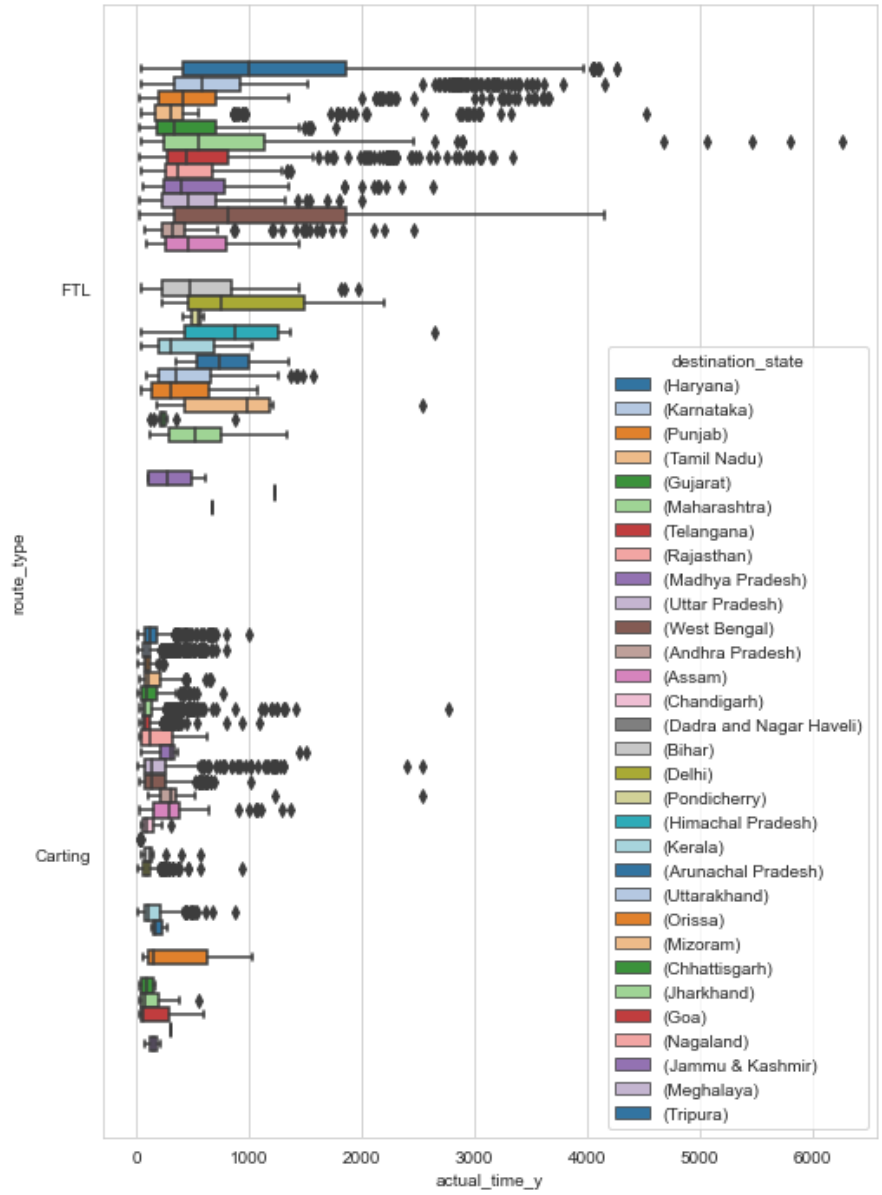


In [1968]:

```
fig1, ax1 = plt.subplots(figsize=(8,12))
custom_palette = sns.color_palette("tab20", n_colors=30)

sns.boxplot(data=x_df, x='actual_time_y', y='route_type', hue='destination_state', palette=custom_palette
```

Out[1968]:

```
<AxesSubplot:xlabel='actual_time_y', ylabel='route_type'>
```

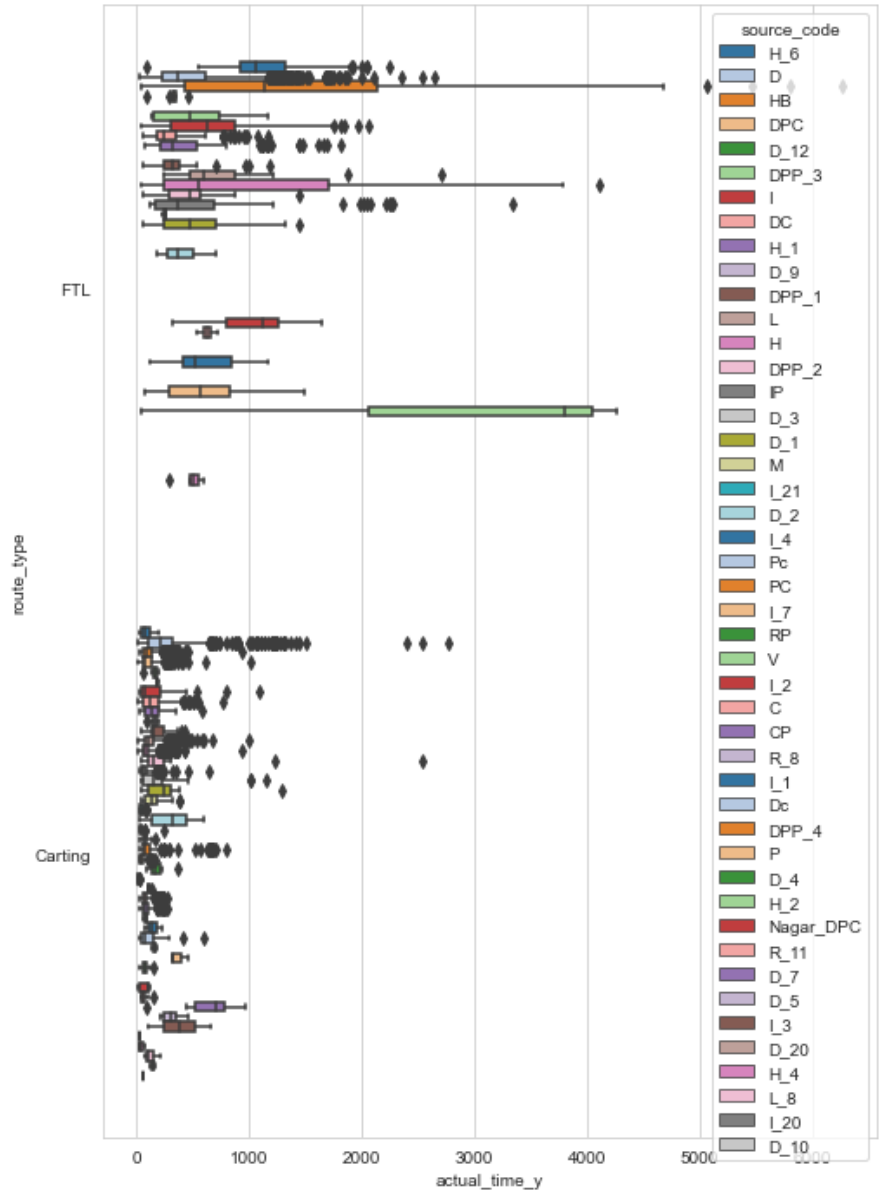


In [1969]:

```
fig1, ax1 = plt.subplots(figsize=(8,12))
custom_palette = sns.color_palette("tab20", n_colors=30)

sns.boxplot(data=x_df, x='actual_time_y', y='route_type', hue='source_code', palette=custom_palette)
```

Out[1969]:

```
<AxesSubplot:xlabel='actual_time_y', ylabel='route_type'>
```

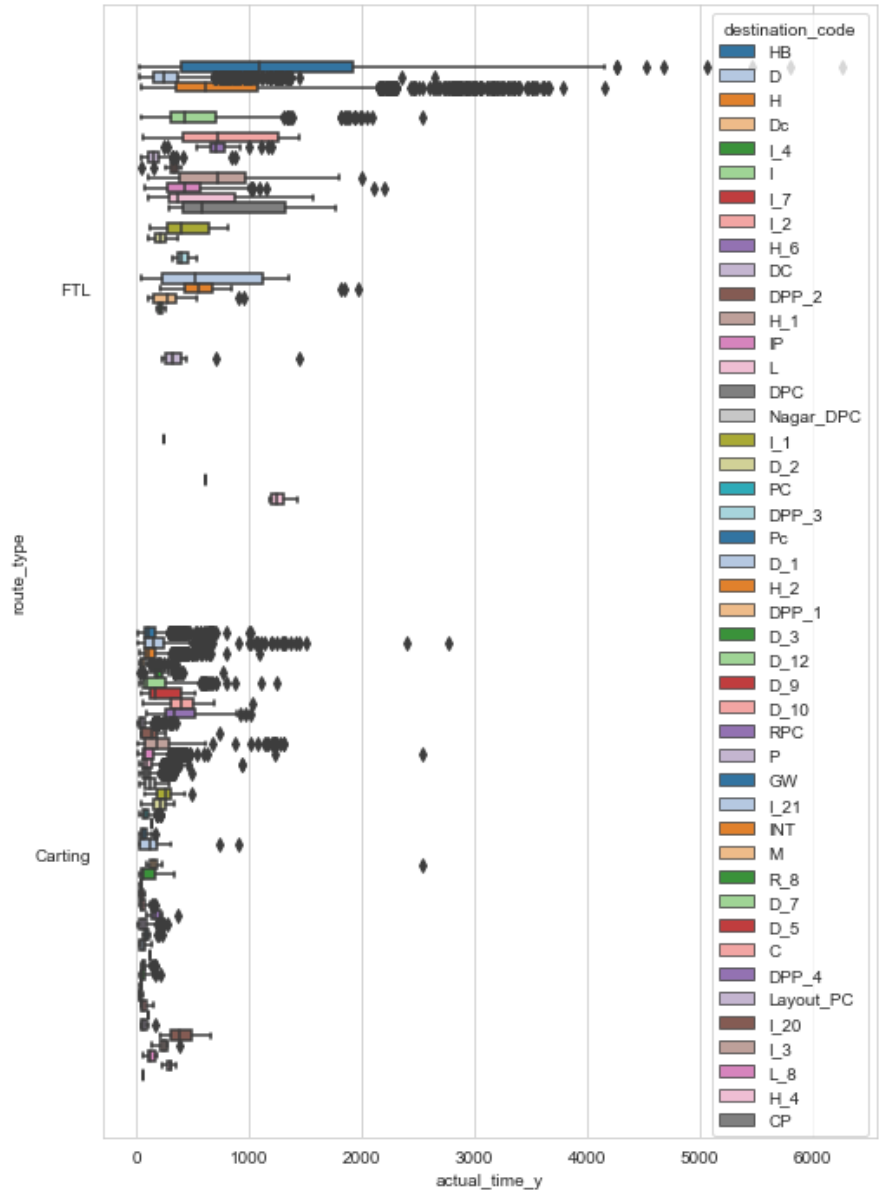


In [1970]:

```
fig1, ax1 = plt.subplots(figsize=(8,12))
custom_palette = sns.color_palette("tab20", n_colors=30)

sns.boxplot(data=x_df, x='actual_time_y', y='route_type', hue='destination_code', palette=custom_palette)
```

Out[1970]:

```
<AxesSubplot:xlabel='actual_time_y', ylabel='route_type'>
```



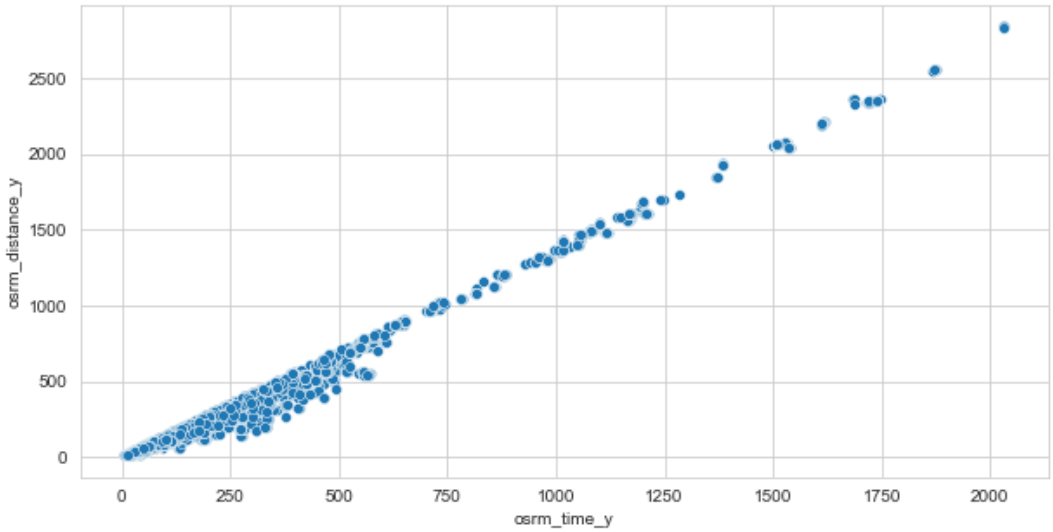
Haryana doesn't have any outliers even though it has one of the highest number of trips from source district West Bengal took the highest number of minutes to complete the trip from source district by which we can also conclude West bengal also has the most number of far away destinations Carting type took very less time to complete the trip when compared to FTL even though FTL does not have any other pickups or drop-offs along the way . In Destination district West bengal doesn't have any outliers Maharashtra took the highest number of minutes to complete the trip to destination district by which we can also conclude Maharashtra receives from most number of far away source districts Carting type took very less time to complete the trip when compared to FTL even though FTL does not have any other pickups or drop-offs along the way . Source code HB has the highest number of minutes to complete the trip by which we can also conclude HB also has the most number of far away destinations Carting type took very less time to complete the trip when compared to FTL even though FTL does not have any other pickups or drop-offs along the way . Destination code HB has the highest number of minutes to complete the trip by which we can also conclude HB receives from most number of far away source districts Carting type took very less time to complete the trip when compared to FTL even though FTL does not have any other pickups or drop-offs along the way .

In [1971]:

```
fig1, ax1 = plt.subplots(figsize=(10, 5))


sns.scatterplot(data =x_df , x ='osrm_time_y', y = 'osrm_distance_y' )
```

Out[1971]:

```
<AxesSubplot:xlabel='osrm_time_y', ylabel='osrm_distance_y'>
```



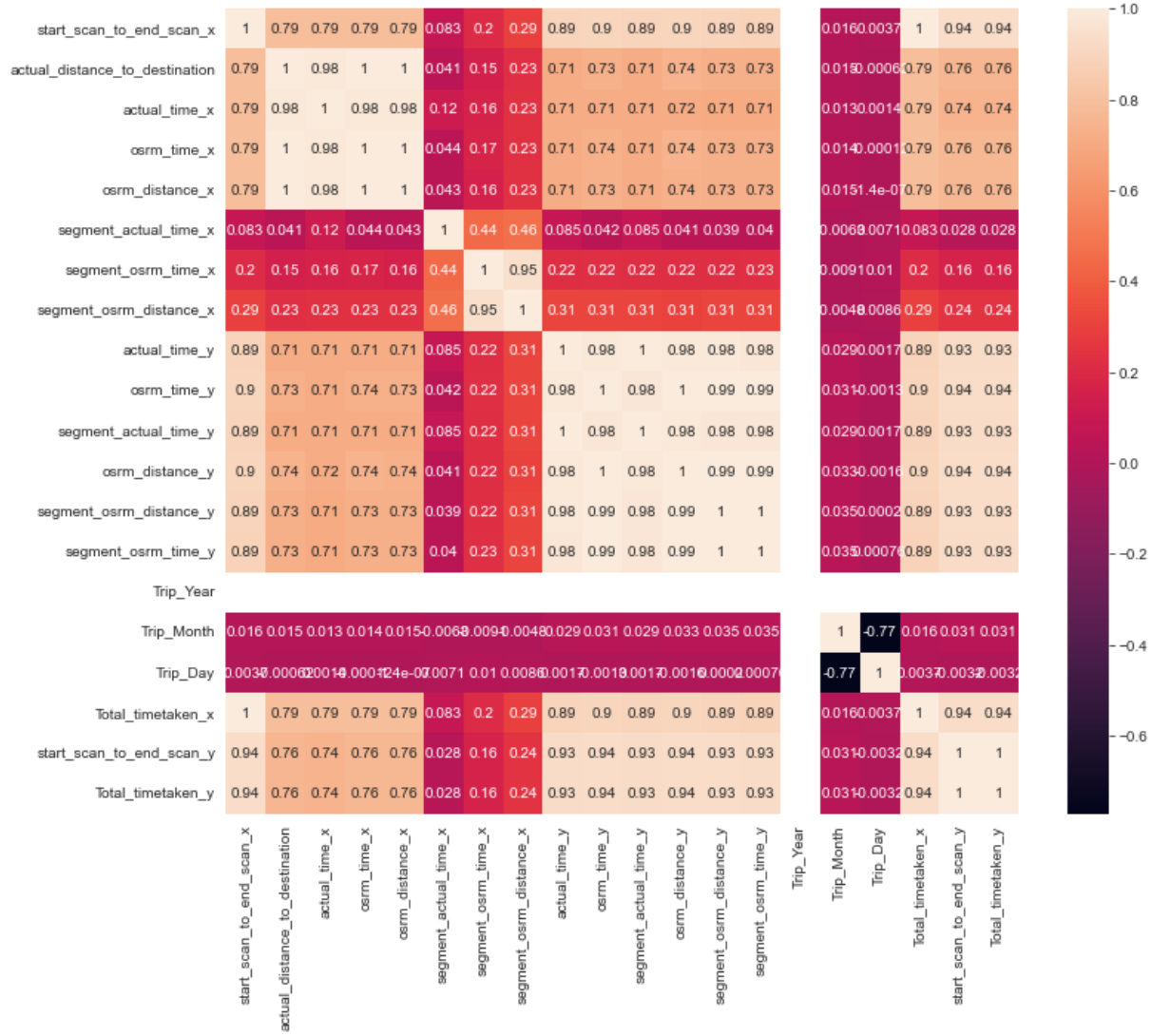
From the scatter plot of osrm_time, osrm_distance (Aggregate values) we can see the trend is linear as the distance increases time increases

In [1972]:

```
fig1, ax1 = plt.subplots(figsize=(12, 10))


sns.heatmap(condensed_df.corr(),annot=True)
```

Out[1972]:

<AxesSubplot:>

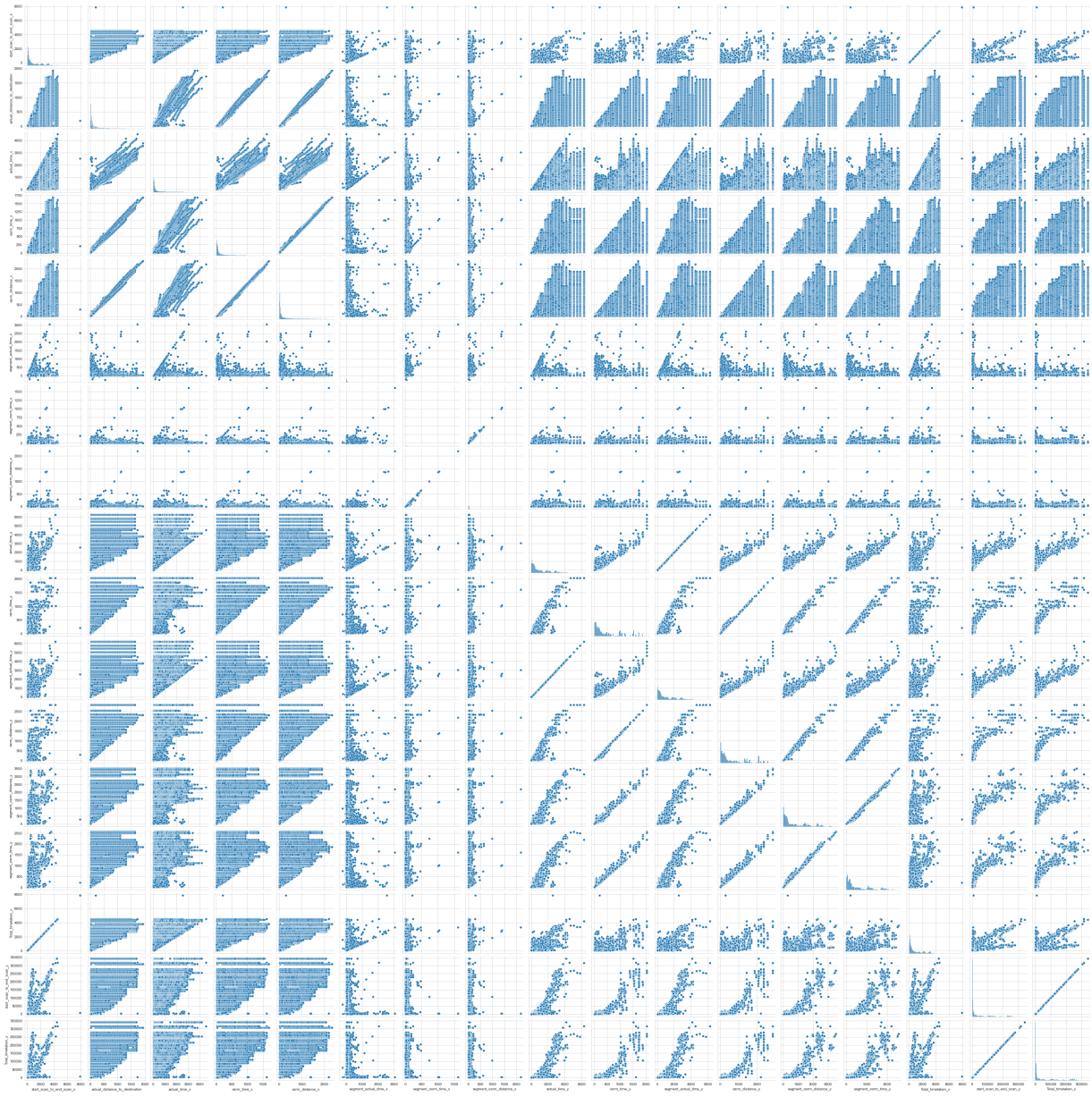


In [1973]:

```
sns.pairplot(condensed_df.select_dtypes(include='float64'))
```

Out[1973]:

```
<seaborn.axisgrid.PairGrid at 0x22df5f4b040>
```



In [1974]:

```
condensed_df
```

Out[1974]:

	data	trip_creation_time	route_type	trip_uuid	start_scan_to_end_scan_x	actual_distance_to_
0	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
1	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
2	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
3	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
4	training	2018-09-12 00:00:16.535741	FTL	trip-153671041653548748	999.0	
...	...	...	...	...	...	
115530	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	45.0	
115531	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	91.0	
115532	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	91.0	
115533	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	91.0	
115534	test	2018-10-03 23:59:14.390954	Carting	trip-153861115439069069	91.0	

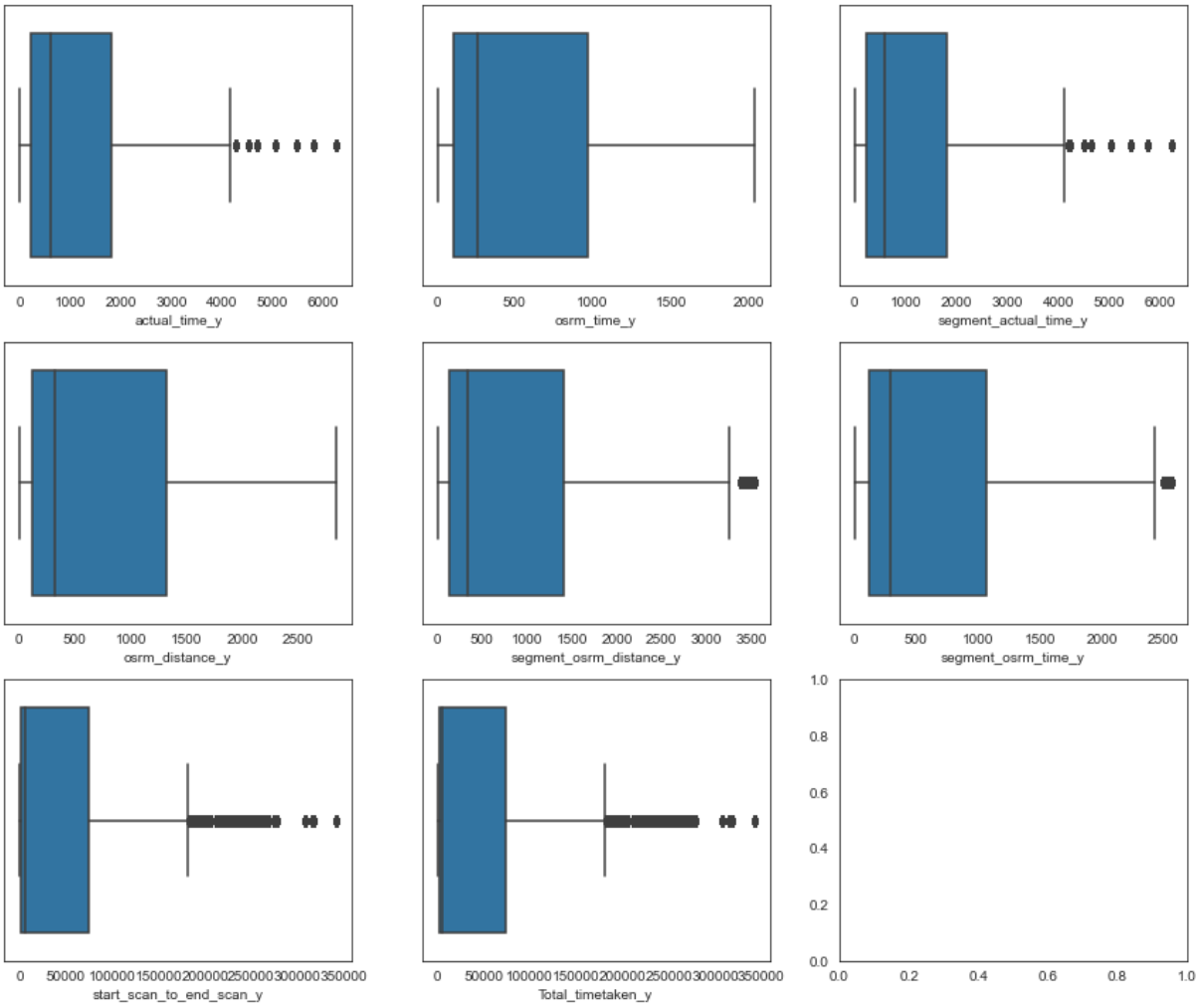
115535 rows × 32 columns

In [1975]:

```
sns.set_style("white")

fig , axis = plt.subplots(nrows = 3 , ncols = 3 , figsize = (15,5))

fig.subplots_adjust(top=2)


sns.boxplot(data =condensed_df , x ='actual_time_y',  ax =axis[0,0])
sns.boxplot(data =condensed_df , x ='osrm_time_y',  ax =axis[0,1])
sns.boxplot(data =condensed_df , x ='segment_actual_time_y',  ax =axis[0,2])
sns.boxplot(data =condensed_df , x ='osrm_distance_y',  ax =axis[1,0])
sns.boxplot(data =condensed_df , x ='segment_osrm_distance_y',  ax =axis[1,1])
sns.boxplot(data =condensed_df , x ='segment_osrm_time_y',  ax =axis[1,2])
sns.boxplot(data =condensed_df , x ='start_scan_to_end_scan_y',  ax =axis[2,0])
sns.boxplot(data =condensed_df , x ='Total_timetaken_y',  ax =axis[2,1])
```

Out[1975]:

<AxesSubplot:xlabel='Total_timetaken_y'>



trend is linear as the distance increases time increases Outliers are found in Columns actual_time_y, segment_actual_time_y, segment_osrm_distance_y, segment_osrm_time_y, start_scan_to_end_scan_y, Total_timetaken_y. NOTE : the letter "_y" after each Column name represents its Aggregated data .

In [1976]:

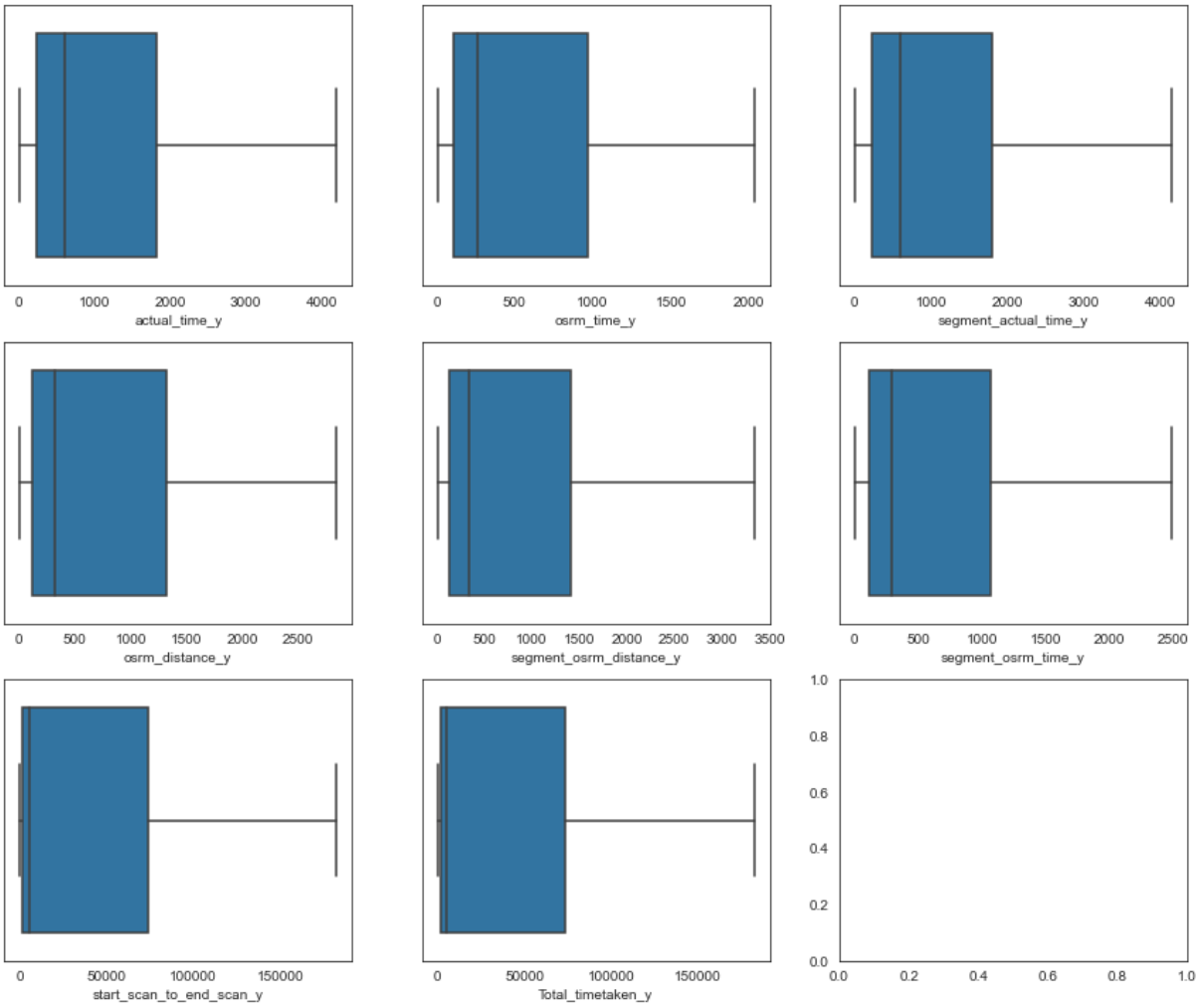
```
# Outlier treatment

cols_with_outliers = ['actual_time_y','segment_actual_time_y','segment_osrm_distance_y','segment_osrm_time
                      'start_scan_to_end_scan_y','Total_timetaken_y']

for col in cols_with_outliers:
    Q1 = condensed_df[col].quantile(0.25)
    Q3 = condensed_df[col].quantile(0.75)
    IQR = Q3 - Q1
    upper = Q3 + (1.5 * IQR)
    condensed_df[col] = np.where(condensed_df[col] > upper, upper, condensed_df[col])
```

In [1977]:

```
sns.set_style("white")

fig , axis = plt.subplots(nrows = 3 , ncols = 3 , figsize = (15,5))

fig.subplots_adjust(top=2)


sns.boxplot(data =condensed_df , x ='actual_time_y',  ax =axis[0,0])
sns.boxplot(data =condensed_df , x ='osrm_time_y',  ax =axis[0,1])
sns.boxplot(data =condensed_df , x ='segment_actual_time_y',  ax =axis[0,2])
sns.boxplot(data =condensed_df , x ='osrm_distance_y',  ax =axis[1,0])
sns.boxplot(data =condensed_df , x ='segment_osrm_distance_y',  ax =axis[1,1])
sns.boxplot(data =condensed_df , x ='segment_osrm_time_y',  ax =axis[1,2])
sns.boxplot(data =condensed_df , x ='start_scan_to_end_scan_y',  ax =axis[2,0])
sns.boxplot(data =condensed_df , x ='Total_timetaken_y',  ax =axis[2,1])
```

Out[1977]:

<AxesSubplot:xlabel='Total_timetaken_y'>



In [1978]:

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

condensed_df.route_type = label_encoder.fit_transform(condensed_df.route_type)
condensed_df.route_type.value_counts()
```

Out[1978]:

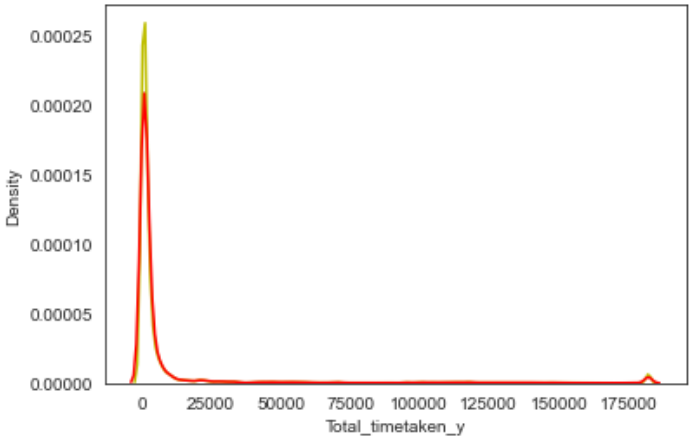
```
1    78506
0    37029
Name: route_type, dtype: int64
```

In [1979]:

```
### Comparing Total_timetaken & start_scan_to_end_scan

x_df = condensed_df.groupby("trip_uuid")['Total_timetaken_y'].last()
x_df = pd.DataFrame(x_df)
x_df = x_df.reset_index()

y_df = condensed_df.groupby("trip_uuid")['start_scan_to_end_scan_y'].last()
y_df = pd.DataFrame(y_df)
y_df = y_df.reset_index()

sns.kdeplot(x_df.Total_timetaken_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.start_scan_to_end_scan_y,bw_adjust=0.3,color = "r")
```

Out[1979]:

```
<AxesSubplot:xlabel='Total_timetaken_y', ylabel='Density'>
```

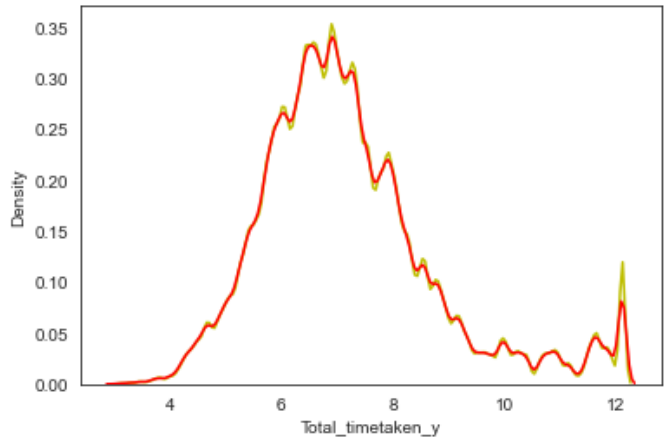


In [1980]:

```
#Normalizing the data using Log transformation
x_df.Total_timetaken_y = np.log(x_df.Total_timetaken_y)

y_df.start_scan_to_end_scan_y = np.log(y_df.start_scan_to_end_scan_y)


sns.kdeplot(x_df.Total_timetaken_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.start_scan_to_end_scan_y,bw_adjust=0.3,color = "r")
```

Out[1980]:

```
<AxesSubplot:xlabel='Total_timetaken_y', ylabel='Density'>
```



In [1981]:

```
# Hypothesis testing of Time taken to complete one whole trip
# H0 : Total_timetaken_y = start_scan_to_end_scan_y
# H1 : Total_timetaken_y != start_scan_to_end_scan_y

from scipy.stats import ttest_ind


T_statistic , p = ttest_ind(x_df.Total_timetaken_y,y_df.start_scan_to_end_scan_y)

if p > 0.01 :
    print("H0 Null hypothesis rejected , there is significant difference between the two groups")
else :
    print("H1 failed to reject Null hypothesis , start_scan_to_end_scan = Total_timetaken")
print("")
print("T_statistic : ",T_statistic, " & ", "P-value : ", p )
```

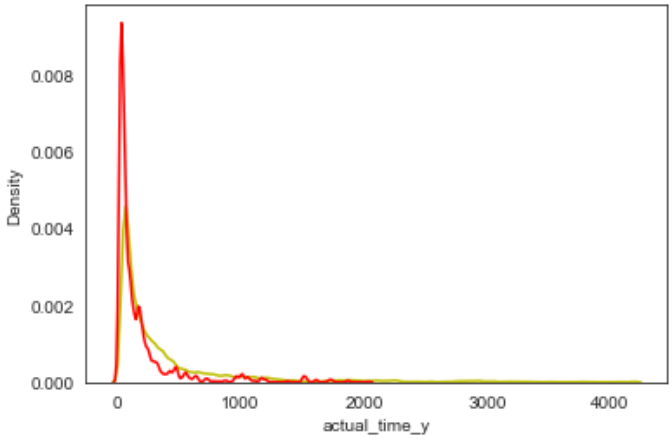
```
H0 Null hypothesis rejected , there is significant difference between the two groups

T_statistic :  0.14574946442066897  &  P-value :  0.8841204077997561
```

In [1982]:

```
### Comparing actual_time_y & osrm_time_y

x_df = condensed_df.groupby("trip_uuid")['actual_time_y'].last()
x_df = pd.DataFrame(x_df)
x_df = x_df.reset_index()

y_df = condensed_df.groupby("trip_uuid")['osrm_time_y'].last()
y_df = pd.DataFrame(y_df)
y_df = y_df.reset_index()

sns.kdeplot(x_df.actual_time_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.osrm_time_y,bw_adjust=0.3,color = "r")
```

Out[1982]:

```
<AxesSubplot:xlabel='actual_time_y', ylabel='Density'>
```

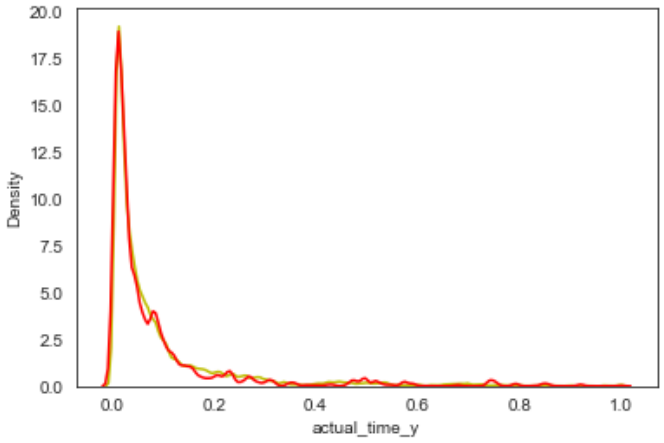


In [1983]:

```
#Normalizing the data using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

x_df = scaler.fit_transform(x_df[["actual_time_y"]])
x_df = pd.DataFrame(x_df, columns=["actual_time_y"])

y_df = scaler.fit_transform(y_df[["osrm_time_y"]])
y_df = pd.DataFrame(y_df, columns=["osrm_time_y"])

sns.kdeplot(x_df.actual_time_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.osrm_time_y,bw_adjust=0.3,color = "r")
```

Out[1983]:

```
<AxesSubplot:xlabel='actual_time_y', ylabel='Density'>
```



In [1984]:

```
# Hypothesis testing of Time taken to complete one whole trip
# H0 : actual_time = osrm_time
# H1 : actual_time != osrm_time

from scipy.stats import ttest_ind


T_statistic , p = ttest_ind(x_df.actual_time_y,y_df.osrm_time_y)

if p > 0.01 :
    print("H0 Null hypothesis rejected , there is significant difference between the two groups")
else :
    print("H1 failed to reject Null hypothesis , actual_time = osrm_time")
print("")
print("T_statistic : ",T_statistic, " & ", "P-value : ", p )
```

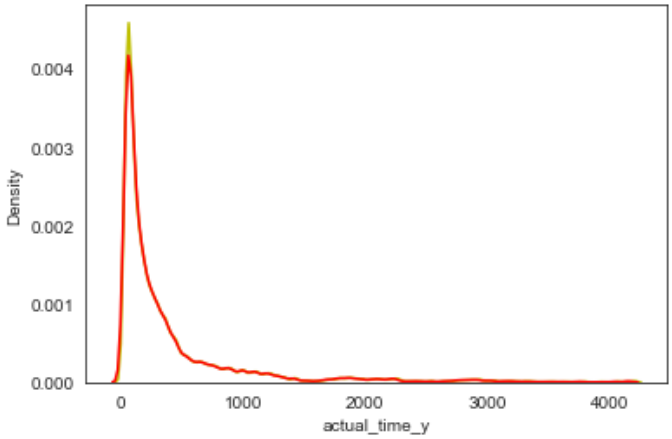
```
H1 failed to reject Null hypothesis , actual_time = osrm_time

T_statistic :  3.5837178063824946  &  P-value :  0.0003394158272098638
```

In [1985]:

```
### Comparing actual_time_y & segment_actual_time_y

x_df = condensed_df.groupby("trip_uuid")['actual_time_y'].last()
x_df = pd.DataFrame(x_df)
x_df = x_df.reset_index()

y_df = condensed_df.groupby("trip_uuid")['segment_actual_time_y'].last()
y_df = pd.DataFrame(y_df)
y_df = y_df.reset_index()

sns.kdeplot(x_df.actual_time_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.segment_actual_time_y,bw_adjust=0.3,color = "r")
```

Out[1985]:

```
<AxesSubplot:xlabel='actual_time_y', ylabel='Density'>
```

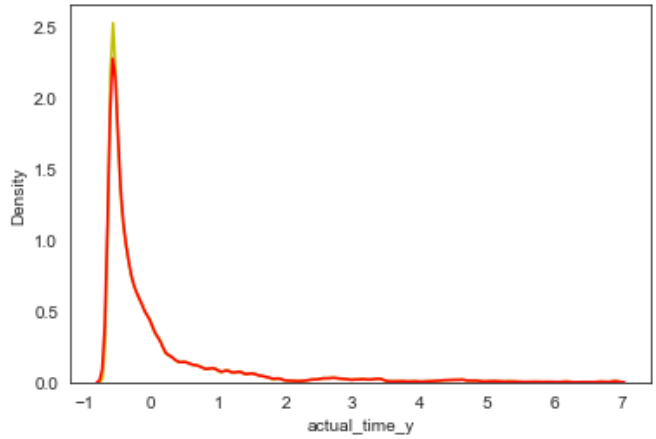


In [1986]:

```
#  Standardizing the data using StandardScaler
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x_df = scaler.fit_transform(x_df[["actual_time_y"]])
x_df = pd.DataFrame(x_df, columns=["actual_time_y"])

y_df = scaler.fit_transform(y_df[["segment_actual_time_y"]])
y_df = pd.DataFrame(y_df, columns=["segment_actual_time_y"])

sns.kdeplot(x_df.actual_time_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.segment_actual_time_y,bw_adjust=0.3,color = "r")
```

Out[1986]:

```
<AxesSubplot:xlabel='actual_time_y', ylabel='Density'>
```



In [1987]:

```
# Hypothesis testing of Time taken to complete one whole trip
# H0 : actual_time = segment_actual_time_y
# H1 : actual_time > segment_actual_time_y

from scipy.stats import ttest_ind


T_statistic , p = ttest_ind(x_df.actual_time_y,y_df.segment_actual_time_y,alternative = "greater")

if p > 0.01 :
    print("H0 Null hypothesis rejected , there is significant difference between the two groups")
else :
    print("H1 failed to reject Null hypothesis , actual_time = segment_actual_time")
print("")
print("T_statistic : ",T_statistic, " & ", "P-value : ", p )
```

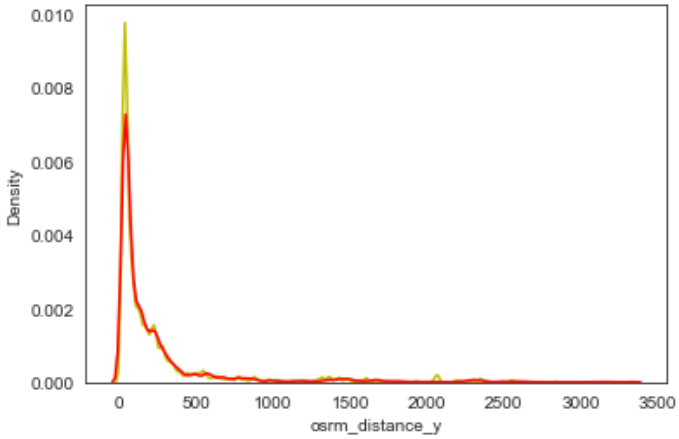
```
H0 Null hypothesis rejected , there is significant difference between the two groups

T_statistic :  1.6594004642163501e-15  &  P-value :  0.49999999999999933
```

In [1988]:

```
### Comparing osrm_distance_y & segment_osrm_distance_y

x_df = condensed_df.groupby("trip_uuid")['osrm_distance_y'].last()
x_df = pd.DataFrame(x_df)
x_df = x_df.reset_index()

y_df = condensed_df.groupby("trip_uuid")['segment_osrm_distance_y'].last()
y_df = pd.DataFrame(y_df)
y_df = y_df.reset_index()

sns.kdeplot(x_df.osrm_distance_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.segment_osrm_distance_y,bw_adjust=0.3,color = "r")
```

Out[1988]:

```
<AxesSubplot:xlabel='osrm_distance_y', ylabel='Density'>
```

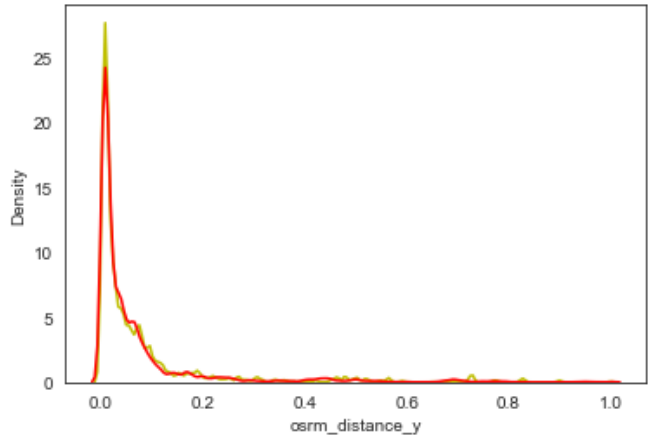


In [1989]:

```
#Normalizing the data using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

x_df = scaler.fit_transform(x_df[["osrm_distance_y"]])
x_df = pd.DataFrame(x_df, columns=["osrm_distance_y"])

y_df = scaler.fit_transform(y_df[["segment_osrm_distance_y"]])
y_df = pd.DataFrame(y_df, columns=["segment_osrm_distance_y"])

sns.kdeplot(x_df.osrm_distance_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.segment_osrm_distance_y,bw_adjust=0.3,color = "r")
```

Out[1989]:

```
<AxesSubplot:xlabel='osrm_distance_y', ylabel='Density'>
```



In [1990]:

```
# Hypothesis testing of Distance travelled to complete one whole trip
# H0 : actual_time = segment_osrm_distance
# H1 : actual_time != segment_osrm_distance

from scipy.stats import ttest_ind


T_statistic , p = ttest_ind(x_df.osrm_distance_y,y_df.segment_osrm_distance_y)

if p > 0.01 :
    print("H0 Null hypothesis rejected , there is significant difference between the two groups")
else :
    print("H1 failed to reject Null hypothesis , osrm_distance = segment_osrm_distance")
print("")
print("T_statistic : ",T_statistic, " & ", "P-value : ", p )
```

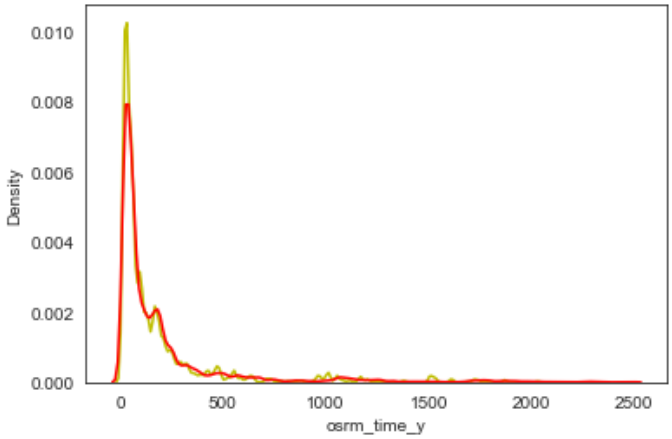
```
H1 failed to reject Null hypothesis , osrm_distance = segment_osrm_distance

T_statistic :  3.006606175860401  &  P-value :  0.002644578637034996
```

In [1991]:

```
### Comparing osrm_time_y & segment_osrm_time_y

x_df = condensed_df.groupby("trip_uuid")['osrm_time_y'].last()
x_df = pd.DataFrame(x_df)
x_df = x_df.reset_index()

y_df = condensed_df.groupby("trip_uuid")['segment_osrm_time_y'].last()
y_df = pd.DataFrame(y_df)
y_df = y_df.reset_index()

sns.kdeplot(x_df.osrm_time_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.segment_osrm_time_y,bw_adjust=0.3,color = "r")
```

Out[1991]:

```
<AxesSubplot:xlabel='osrm_time_y', ylabel='Density'>
```

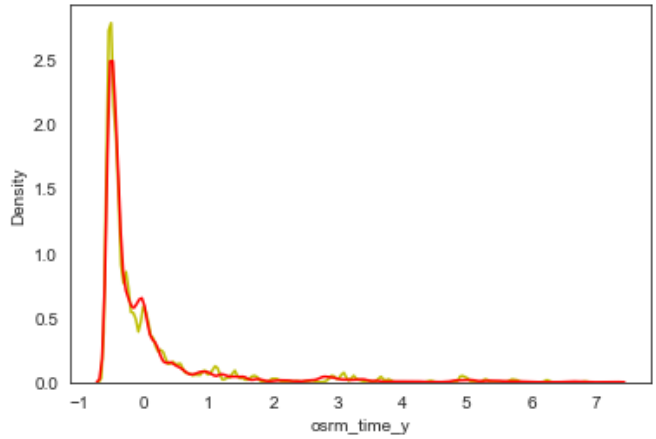


In [1992]:

```
#  Standardizing the data using StandardScaler
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x_df = scaler.fit_transform(x_df[["osrm_time_y"]])
x_df = pd.DataFrame(x_df, columns=["osrm_time_y"])

y_df = scaler.fit_transform(y_df[["segment_osrm_time_y"]])
y_df = pd.DataFrame(y_df, columns=["segment_osrm_time_y"])

sns.kdeplot(x_df.osrm_time_y,bw_adjust=0.2,color = "y")
sns.kdeplot(y_df.segment_osrm_time_y,bw_adjust=0.3,color = "r")
```

Out[1992]:

```
<AxesSubplot:xlabel='osrm_time_y', ylabel='Density'>
```



In [1993]:

```
# Hypothesis testing of Time taken to complete one whole trip
# H0 : osrm_time = segment_osrm_time
# H1 : osrm_time > segment_osrm_time

from scipy.stats import ttest_ind


T_statistic , p = ttest_ind(x_df.osrm_time_y,y_df.segment_osrm_time_y,alternative = "greater")

if p > 0.01 :
    print("H0 Null hypothesis rejected , there is significant difference between the two groups")
else :
    print("H1 failed to reject Null hypothesis , actual_time = segment_actual_time")
print("")
print("T_statistic : ",T_statistic, " & ", "P-value : ", p )
```

```
H0 Null hypothesis rejected , there is significant difference between the two groups

T_statistic :  2.6734785256818978e-15  &  P-value :  0.49999999999999895
```

INSIGHTS

The Given data has 144867 Rows and 24 Columns The given data has two kinds of data that Training and test data No of rows in training data : 104858 , No of rows in test data : 40009 , We can see that majority of the data is training data Each Trip_uuid has more than one stops or relay points in its journey to reach from source to destination , hence group-by and Aggregate functions should be used careful i,e while grouping ["trip_uuid","source_name","destination_name"] use “last()” as we have multiple sources and destinations for each trip and while grouping ("trip_uuid") use sum() , also while using the grouping of segmented values we can directly use sum() as each value is individual recording . Columns Source-name and Destination-name have missing values with less than 0.2% , dropping off these rows as such minor percent of values may not affect the data. After dividing Source-name

and Destination-name into city, place , code , state , it is seen that the address format is not the same for all orders and missing values can be found in place -1.5% , code-10% , state-10% , filling the null values with value of column city of the same respective rows Column trip_creation_time can be divided into Trip_Year, Trip_Month, Trip_Day , from these we can infer that data is from year 2018, month september(9) but data from days 4 to 11 are not present in data . Column start_scan_to_end_scan contains time taken to deliver from source to destination in Mins Using columns 'od_end_time','od_start_time' a new feature can be created Total time taken but checking the difference between 'od_end_time','od_start_time' . Before grouping the data by trip_uuid Route type FTL comprises 67.95% and Carting comprises 32.05% of the total orders . Top cities that have more number of Trips are Bangalore, Gurgaon, Kolkata, Hyderabad, Bhiwandi, Delhi. FTL deliveries are mostly Inter-state and Carting are mostly Intra-state services. After grouping the data by trip_uuid Route type FTL comprises 57.84% and Carting comprises 42.16% of the total orders . From the above difference between before and after grouping by trip_uuid we can infer that FTL type has more relay point although it is not making no other pickups or drop-offs along the way Among the given data Training data comprises 72.49% and test data comprises 27.51% Grouping data on trip_uuid to perform Visual Analysis Univariate Analysis : The most number of trips are from source code “D” & “HB” The most number of trips are to destination code “D” , “HB”,”H” & ”I” The most number of trips are from source state “Karnataka” , “Maharashtra”, “Haryana” The most number of trips are to Destination state “Karnataka” , “Maharashtra”, “Haryana” We can see that there is a slight decline in number of trips at the end of month Bivariate Analysis : Although Karnataka has the highest number of trips from Source district still the majority of those trips come under non-dominant route-type i,e Carting , whereas Maharashtra has highest FTL type trips still which is dominated by Carting . Insights that are observed in trips from source district resemble the insights observed in trips to destination district Source codes D and HB are quiet opposite when it comes to dominance of FTL and carting, D has more number of FTL’s whereas HB has more number of Carting’s In Destination codes D , HB are same as source codes and H , I codes have FTL in dominance By comparing source code and destination code we can understand that these codes are again divided internally as sub-codes . Multivariate Analysis : Haryana doesn't have any outliers even though it has one of the highest number of trips from source district West Bengal took the highest number of minutes to complete the trip from source district by which we can also conclude West bengal also has the most number of far away destinations Carting type took very less time to complete the trip when compared to FTL even though FTL does not have any other pickups or drop-offs along the way . In Destination district West bengal doesn't have any outliers Maharashtra took the highest number of minutes to complete the trip to destination district by which we can also conclude Maharashtra receives from most number of far away source districts Carting type took very less time to complete the trip when compared to FTL even though FTL does not have any other pickups or drop-offs along the way . Source code HB has the highest number of minutes to complete the trip by which we can also conclude HB also has the most number of far away destinations Carting type took very less time to complete the trip when compared to FTL even though FTL does not have any other pickups or drop-offs along the way . Destination code HB has the highest number of minutes to complete the trip by which we can also conclude HB receives from most number of far away source districts Carting type took very less time to complete the trip when compared to FTL even though FTL does not have any other pickups or drop-offs along the way . From the scatter plot of osrm_time, osrm_distance (Aggregate values) we can see the trend is linear as the distance increases time increases Outliers are found in Columns actual_time_y, segment_actual_time_y, segment_osrm_distance_y, segment_osrm_time_y, start_scan_to_end_scan_y, Total_timetaken_y. NOTE : the letter “_y” after each Column name represents its Aggregated data . Visual analysis of Total_timetaken & start_scan_to_end_scan shows that both the data are similar & Hypothesis test Result - H0 Null hypothesis rejected , there is significant difference between the two groups Visual analysis of actual_time_y & osrm_time_y shows that both the data are similar & Hypothesis test Result - H1 failed to reject Null hypothesis , actual_time = osrm_time Visual analysis of actual_time_y & segment_actual_time_y shows that both the data are similar & Hypothesis test Result - H0 Null hypothesis rejected , there is significant difference between the two groups. Visual analysis of osrm_distance_y & segment_osrm_distance_y shows that both the data are similar & Hypothesis test Result - H1 failed to reject Null hypothesis , osrm_distance = segment_osrm_distance Visual analysis of osrm_time_y & segment_osrm_time_y shows that both the data are similar & Hypothesis test Result - H0 Null hypothesis rejected , there is significant difference between the two groups

Recommendations

States and cities that have Higher traffic of Orders like Maharashtra, Karnataka ,Haryana also Bangalore , Hyderabad etc need to check and increase warehouse capacity and workers. States and cities that have Lower traffic of Orders like Jummu & Kashmir, Uttarakhand etc , the company has to increase their exposure for their services by using difference means of advertising . Delhivary mostly provides services in Bulk and B to B , so creating weight margin discounts can increase sales dramatically. Reviewing the used OSRM can be beneficial as it failed to match the actual observations in some cases like osrm_time & segment_osrm_time .

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: