In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

About Delhivery

Delhivery is the largest and fastest-growing fully integrated player in India by revenue in Fiscal 2021. They aim to build the operating system for commerce, through a combination of world-class infrastructure, logistics operations of the highest quality, and cutting-edge engineering and technology capabilities.

Problem Statement

The Data team builds intelligence and capabilities using this data that helps them to widen the gap between the quality, efficiency, and profitability of their business versus their competitors.

In [2]:
```
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/5
```

In [3]:
```
df.head()
```

Out[3]:

	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

Solution 1

In [4]:
```
# Checking the shape
df.shape
```

Out[4]: (144867, 24)

In [5]:
```
# checking null values #source_name and destination_name found missing
df.isnull().sum()
```

Out[5]:
```
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 [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 [96]:

```
# describing all the numerical columns
#OSRM stands for Open Source Routing Machine.
#It is an open-source routing engine that provides fast, accurate, and customizable ro
df.describe()
```

Out[96]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distan
count	144867.000000	1.448670e+05	1.448670e+05	1.448670e+05	1.448670e+(
mean	961.262986	-1.177150e-16	2.158109e-17	-6.748996e-17	5.689561e-1
std	1037.012769	1.000003e+00	1.000003e+00	1.000003e+00	1.000003e+(
min	20.000000	-6.524076e-01	-6.820372e-01	-6.748750e-01	-6.548359e-(
25%	161.000000	-6.107952e-01	-6.118150e-01	-6.066954e-01	-6.051907e-(
50%	449.000000	-4.868181e-01	-4.763865e-01	-4.865695e-01	-4.897572e-(
75%	1634.000000	1.525716e-01	1.606290e-01	1.400335e-01	1.387307e-(
max	7898.000000	4.908490e+00	6.880224e+00	4.779493e+00	4.847640e+(

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["source_name"].nunique()
```

Out[9]:
```
1498
```

In [10]:
```
df["source_name"].value_counts()
```

Out[10]:
```
Gurgaon_Bilaspur_HB (Haryana)            23347
Bangalore_Nelmngla_H (Karnataka)          9975
Bhiwandi_Mankoli_HB (Maharashtra)         9088
Pune_Tathawde_H (Maharashtra)             4061
Hyderabad_Shamshbd_H (Telangana)          3340
                                         ...
Shahjhnpur_NavdaCln_D (Uttar Pradesh)        1
Soro_UttarDPP_D (Orissa)                     1
Kayamkulam_Bhrnikvu_D (Kerala)               1
Krishnanagar_AnadiDPP_D (West Bengal)        1
Faridabad_Old (Haryana)                      1
Name: source_name, Length: 1498, dtype: int64
```

In [11]:
```
df["destination_name"].nunique()
```

Out[11]:
```
1468
```

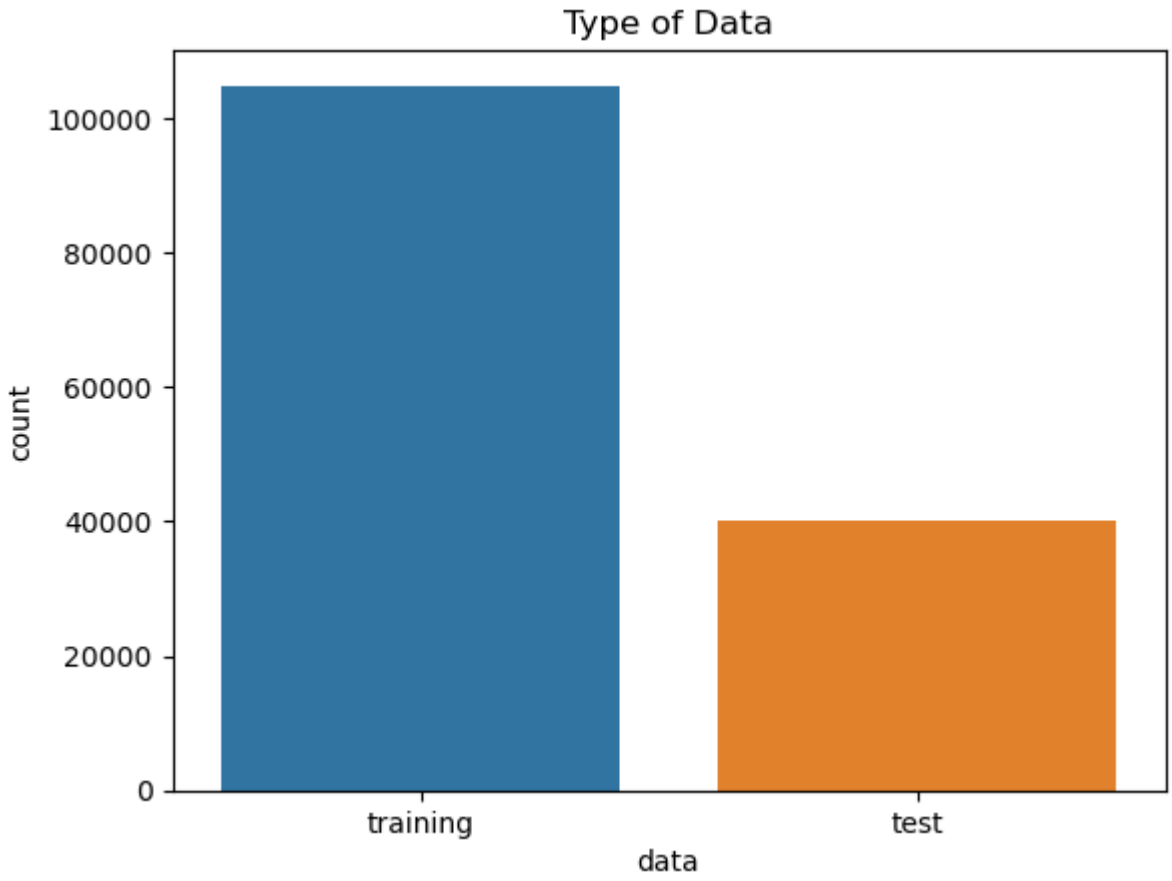
In [12]:
```
df["destination_name"].value_counts()
```

Out[12]:
```
Gurgaon_Bilaspur_HB (Haryana)        15192
Bangalore_Nelmngla_H (Karnataka)     11019
Bhiwandi_Mankoli_HB (Maharashtra)     5492
Hyderabad_Shamshbd_H (Telangana)      5142
Kolkata_Dankuni_HB (West Bengal)      4892
                                     ...
Hyd_Trimulgherry_Dc (Telangana)          1
Vijayawada (Andhra Pradesh)              1
Baghpat_Barout_D (Uttar Pradesh)         1
Mumbai_Sanpada_CP (Maharashtra)          1
Basta_Central_DPP_1 (Orissa)             1
Name: destination_name, Length: 1468, dtype: int64
```

In [13]:
```
df["data"].value_counts()
```

Out[13]:
```
training    104858
test         40009
Name: data, dtype: int64
```

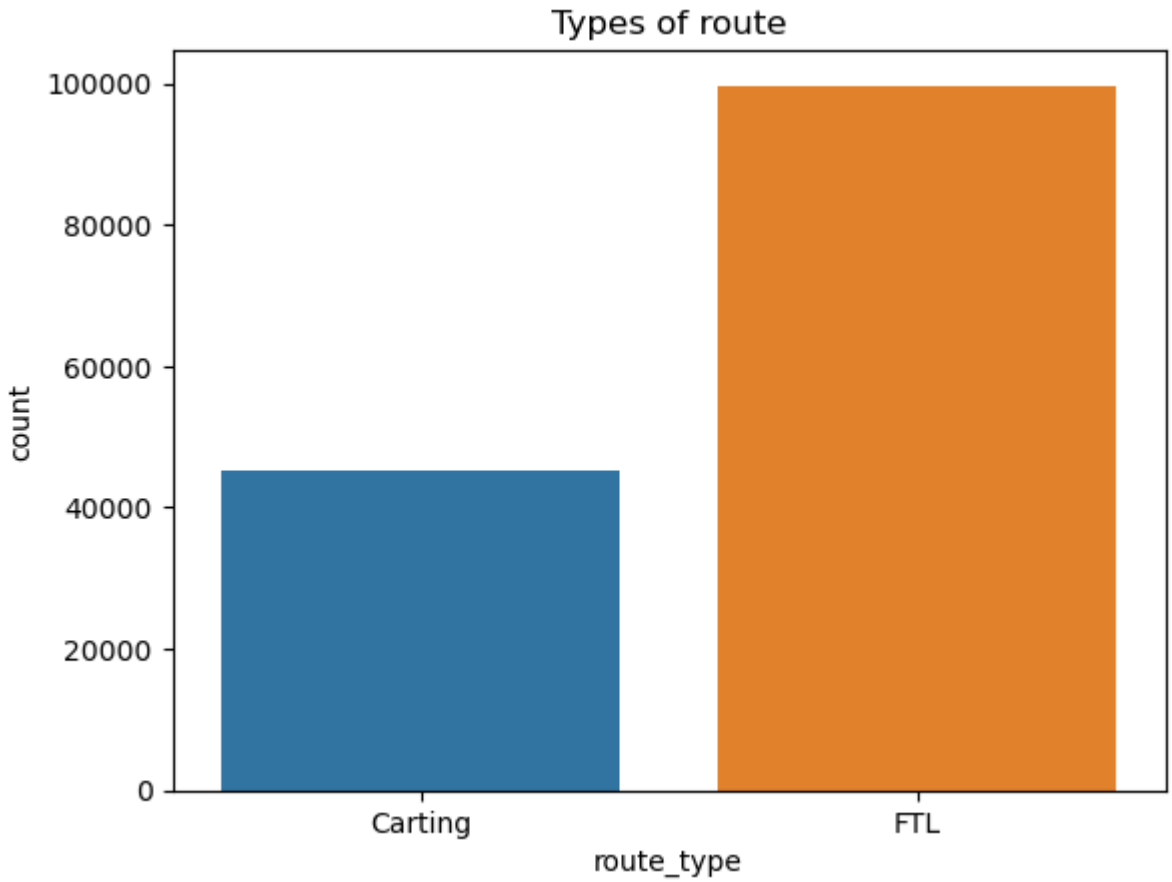
In [14]:
```
sns.countplot(x="data", data= df)
plt.title("Type of Data")
plt.show()
```



In [15]:
```
df["route_type"].value_counts()
```

Out[15]:
```
FTL        99660
Carting    45207
Name: route_type, dtype: int64
```

In [16]:
```
sns.countplot(x="route_type", data=df)
plt.title("Types of route")
plt.show()
```



In [17]:
```
df["trip_uuid"].value_counts()
```

Out[17]:
```
trip-153811219535896559    101
trip-153846035308581166    101
trip-153802363942560700    101
trip-153759210483476123    101
trip-153819749763881430    101
                          ...
trip-153807169820740041      1
trip-153815586768995663      1
trip-153823299365493206      1
trip-153733174477629450      1
trip-153694467298919626      1
Name: trip_uuid, Length: 14817, dtype: int64
```

In [18]:
```
df["source_center"].value_counts()
```

Out[18]:
```
IND000000ACB    23347
IND562132AAA     9975
IND421302AAG     9088
IND411033AAA     4061
IND501359AAE     3340
                ...
IND741121AAA        1
IND207123AAA        1
IND242001AAA        1
IND222001AAA        1
IND741101AAB        1
Name: source_center, Length: 1508, dtype: int64
```

Dropping Unknown Columns

In [19]:
```
# drops columns which were unknown :5 columns
df.drop(['is_cutoff', 'cutoff_factor' , 'cutoff_timestamp', 'factor', 'segment_factor'
```

In [20]:
```
df.head()
```

Out[20]:

	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

In [21]:
```
#Now 19 columns left
df.shape
```

Out[21]:
```
(144867, 19)
```

Checking for Dupliacte Rows

In [22]:
```
# chceking for duplicates, there are no duplicate rows
duplicateRows = df[df.duplicated()]
duplicateRows
```

Out[22]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	de

In [23]:
```
df.shape
```

Out[23]:
```
(144867, 19)
```

Soution - 5 : HANDLE missing values

In [24]:
```
# Chceking for missing values
df.isnull().sum()
```

Out[24]:
```
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
```

In [25]:
```
# filling a null values using fillna()
df["source_name"].fillna("source not Avialable", inplace = True)
```

In [26]:
```
# filling Destination name
df["destination_name"].fillna("Destination Name not Avialable", inplace = True)
```

In [27]:
```
df.isnull().sum()
```

Out[27]:
```
data                              0
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
```

Fixing Date and Time Column

In [28]:
```
df.head()
```

Out[28]:

	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

In [29]:
```
df.columns
```

Out[29]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance'],
      dtype='object')
```

In [30]:
```
df["trip_creation_time"].value_counts()
```

Out[30]:
```
2018-09-28 05:23:15.359220    101
2018-10-02 06:05:53.086094    101
2018-09-27 04:47:19.425867    101
2018-09-22 04:55:04.835022    101
2018-09-29 05:04:57.639067    101
                             ...
2018-09-27 18:08:18.207639      1
2018-09-28 17:31:07.690205      1
2018-09-29 14:56:33.655170      1
2018-09-19 04:35:44.776558      1
2018-09-14 17:04:32.989471      1
Name: trip_creation_time, Length: 14817, dtype: int64
```

In [31]:
```
# Coverting the columns in proper datetime format which were in integer

df["trip_creation_time"] = pd.to_datetime(df["trip_creation_time"])
df["od_start_time"]= pd.to_datetime(df["od_start_time"])
df["od_end_time"]= pd.to_datetime(df["od_end_time"])
```

In [32]:
```
df.dtypes
```

Out[32]:
```
data                                    object
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
```

In [33]:
```
df["actual_time"]
```

Out[33]:
```
0          14.0
1          24.0
2          40.0
3          62.0
4          68.0
          ...
144862     94.0
144863    120.0
144864    140.0
144865    158.0
144866    426.0
Name: actual_time, Length: 144867, dtype: float64
```

In [34]:
```
df.shape
```

Out[34]:
```
(144867, 19)
```

In [35]:
```
df["actual_time"].value_counts()
```

Out[35]:
```
32.0      1443
36.0      1420
30.0      1350
38.0      1329
42.0      1241
          ...
2709.0       1
2608.0       1
2910.0       1
2870.0       1
2980.0       1
Name: actual_time, Length: 3182, dtype: int64
```

Solution 3-Grouping by segment

In [36]:
```
# Creating a New Column "segment_key" by concatenating the values and then hashing the
df['segment_key'] = df[['trip_uuid', 'source_center', 'destination_center']].astype(st
df['segment_key'] = df['segment_key'].apply(lambda x: hash(x))
```

In [37]:
```
df['segment_key'][0]
```

Out[37]:
```
-2223622434856891013
```

In [38]:
```
print(df[['trip_uuid', 'source_center', 'destination_center', 'segment_key']])
```

```
                    trip_uuid source_center destination_center  \
0       trip-153741093647649320  IND388121AAA       IND388620AAB
1       trip-153741093647649320  IND388121AAA       IND388620AAB
2       trip-153741093647649320  IND388121AAA       IND388620AAB
3       trip-153741093647649320  IND388121AAA       IND388620AAB
4       trip-153741093647649320  IND388121AAA       IND388620AAB
...                         ...           ...                ...
144862  trip-153746066843555182  IND131028AAB       IND000000ACB
144863  trip-153746066843555182  IND131028AAB       IND000000ACB
144864  trip-153746066843555182  IND131028AAB       IND000000ACB
144865  trip-153746066843555182  IND131028AAB       IND000000ACB
144866  trip-153746066843555182  IND131028AAB       IND000000ACB

                segment_key
0      -2223622434856891013
1      -2223622434856891013
2      -2223622434856891013
3      -2223622434856891013
4      -2223622434856891013
...                     ...
144862 -3952487073946094192
144863 -3952487073946094192
144864 -3952487073946094192
144865 -3952487073946094192
144866 -3952487073946094192

[144867 rows x 4 columns]
```

In [39]:
```
# Cumulative sum segment wise of actual_time, osrm_distance, osrm_time
df['segment_actual_time_cumsum'] = df.groupby('segment_key')['segment_actual_time'].cu
df['segment_osrm_distance_cumsum'] = df.groupby('segment_key')['segment_osrm_distance'
df['segment_osrm_time_cumsum'] = df.groupby('segment_key')['segment_osrm_time'].cumsum
```

In [40]:
```
# showing column before and after cusmulatative sum of segment time
df[['segment_actual_time','segment_actual_time_cumsum']]
```

Out[40]:

	segment_actual_time	segment_actual_time_cumsum
0	14.0	14.0
1	10.0	24.0
2	16.0	40.0
3	21.0	61.0
4	6.0	67.0
...	...	...
144862	12.0	92.0
144863	26.0	118.0
144864	20.0	138.0
144865	17.0	155.0
144866	268.0	423.0

144867 rows × 2 columns

In [41]:
```
# showing column before and after cusmulatative sum of segment distance
df[['segment_osrm_distance','segment_osrm_distance_cumsum']]
```

Out[41]:

	segment_osrm_distance	segment_osrm_distance_cumsum
0	11.9653	11.9653
1	9.7590	21.7243
2	10.8152	32.5395
3	13.0224	45.5619
4	3.9153	49.4772
...	...	...
144862	8.1858	65.3487
144863	17.3725	82.7212
144864	20.7053	103.4265
144865	18.8885	122.3150
144866	8.8088	131.1238

144867 rows × 2 columns

In [42]:
```
# showing column before and after cusmulatative sum of segment osrm time
df[['segment_osrm_time', 'segment_osrm_time_cumsum']]
```

Out[42]:

	segment_osrm_time	segment_osrm_time_cumsum
0	11.0	11.0
1	9.0	20.0
2	7.0	27.0
3	12.0	39.0
4	5.0	44.0
...	...	...
144862	12.0	94.0
144863	21.0	115.0
144864	34.0	149.0
144865	27.0	176.0
144866	9.0	185.0

144867 rows × 2 columns

Solution 6: Aggregation at Segment Level

In [43]:
```
create_segment_dict = {
    'trip_creation_time': 'first',  # Keep the first value
    'route_schedule_uuid': 'first',  # Keep the first value
    'route_type': 'first',  # Keep the first value
    'source_name': 'first',  # Keep the first value
    'destination_name': 'first',  # Keep the first value
    'od_start_time': 'first',  # Keep the first value
    'od_end_time': 'last',  # Keep the last value
    'start_scan_to_end_scan': 'sum',  # Aggregate by sum
    'actual_distance_to_destination': 'sum',  # Aggregate by sum
    'actual_time': 'sum',  # Aggregate by sum
    'osrm_time': 'sum',  # Aggregate by sum
    'osrm_distance': 'sum',  # Aggregate by sum
    'segment_actual_time': 'last',  # Keep the last value
    'segment_osrm_distance': 'last',  # Keep the last value
    'segment_osrm_time': 'last'  # Keep the last value
}
```

In [44]:
```
df.columns
```

Out[44]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'segment_key',
       'segment_actual_time_cumsum', 'segment_osrm_distance_cumsum',
       'segment_osrm_time_cumsum'],
      dtype='object')
```

In [45]:
```
df.shape
```

Out[45]:
```
(144867, 23)
```

In [46]:
```
segment_aggregated = df.groupby('segment_key').agg(create_segment_dict).reset_index()
```

In [47]:
```
segment_aggregated
```

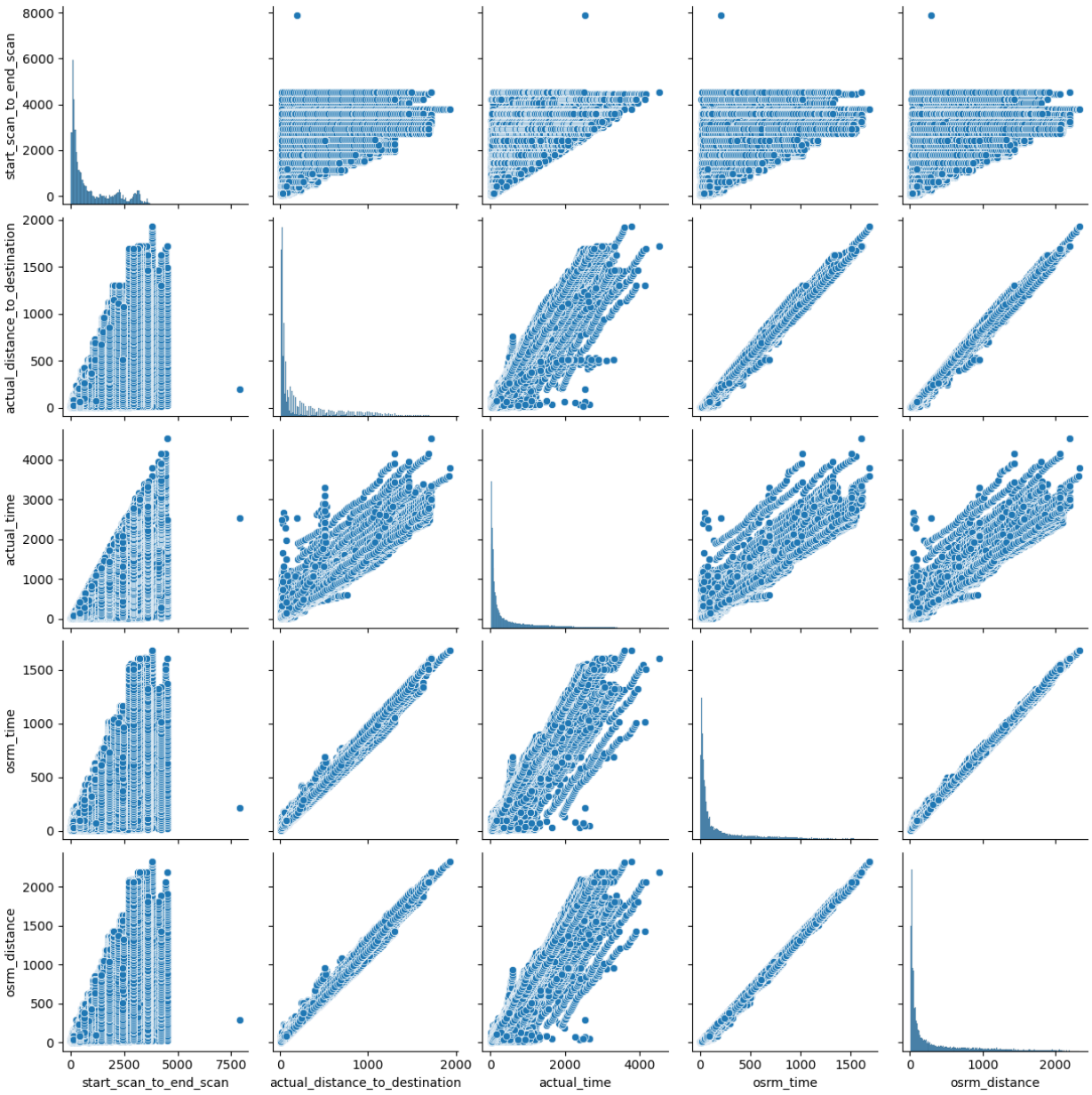
Out[47]:

	segment_key	trip_creation_time	route_schedule_uuid	route_type	source_na
0	-9222796547932529665	2018-10-03 19:22:39.293216	thanos::sroute:6e01eb09-11ca-4a88-839f-2fe0836...	Carting	Mumbai H (Maharash
1	-9220630053641770878	2018-09-15 01:58:24.742401	thanos::sroute:34330d20-1386-457b-8373-f94cc2f...	FTL	Bobbili_ColegR (Andhra Prade
2	-9218119143826161464	2018-09-22 21:00:28.082589	thanos::sroute:20f7c97f-614d-4501-96c7-fbd43d5...	Carting	Gurgaon_Bilaspur (Harya
3	-9216891155464067624	2018-09-22 00:28:35.499623	thanos::sroute:36941a6b-0e90-4582-a95a-96666dc...	Carting	Vaijiapur_YeolaRI (Maharash
4	-9216512112440739564	2018-10-01 00:02:31.857184	thanos::sroute:883e99fa-50a3-40e0-a2e2-9b12ed6...	FTL	Motihari_RajaBz (Bi
...	...	...	...	...	
26363	9219235042858240561	2018-09-27 17:57:56.350055	thanos::sroute:2c33e360-7e52-4d2c-a9db-fe24996...	FTL	Gurgaon_Bilaspur (Harya
26364	9219328082127877630	2018-10-03 10:53:58.045174	thanos::sroute:e2263654-53fa-4faa-ad8a-7956bca...	FTL	Hubli_Adargch (Karnata
26365	9220089546824066172	2018-09-28 23:06:33.589099	thanos::sroute:14870bd3-0ef2-4c83-8ba3-961539c...	Carting	Pune_Tathawd (Maharash
26366	9220227007014954146	2018-10-03 04:01:36.472494	thanos::sroute:9e7bb811-593f-47bc-ac49-ba03ed8...	Carting	Mumbai_Sanpa (Maharash
26367	9221140859336740686	2018-09-14 23:22:40.495686	thanos::sroute:17458102-0fe7-48eb-aefa-6b64e87...	Carting	Etah_CivlLin (Uttar Prade

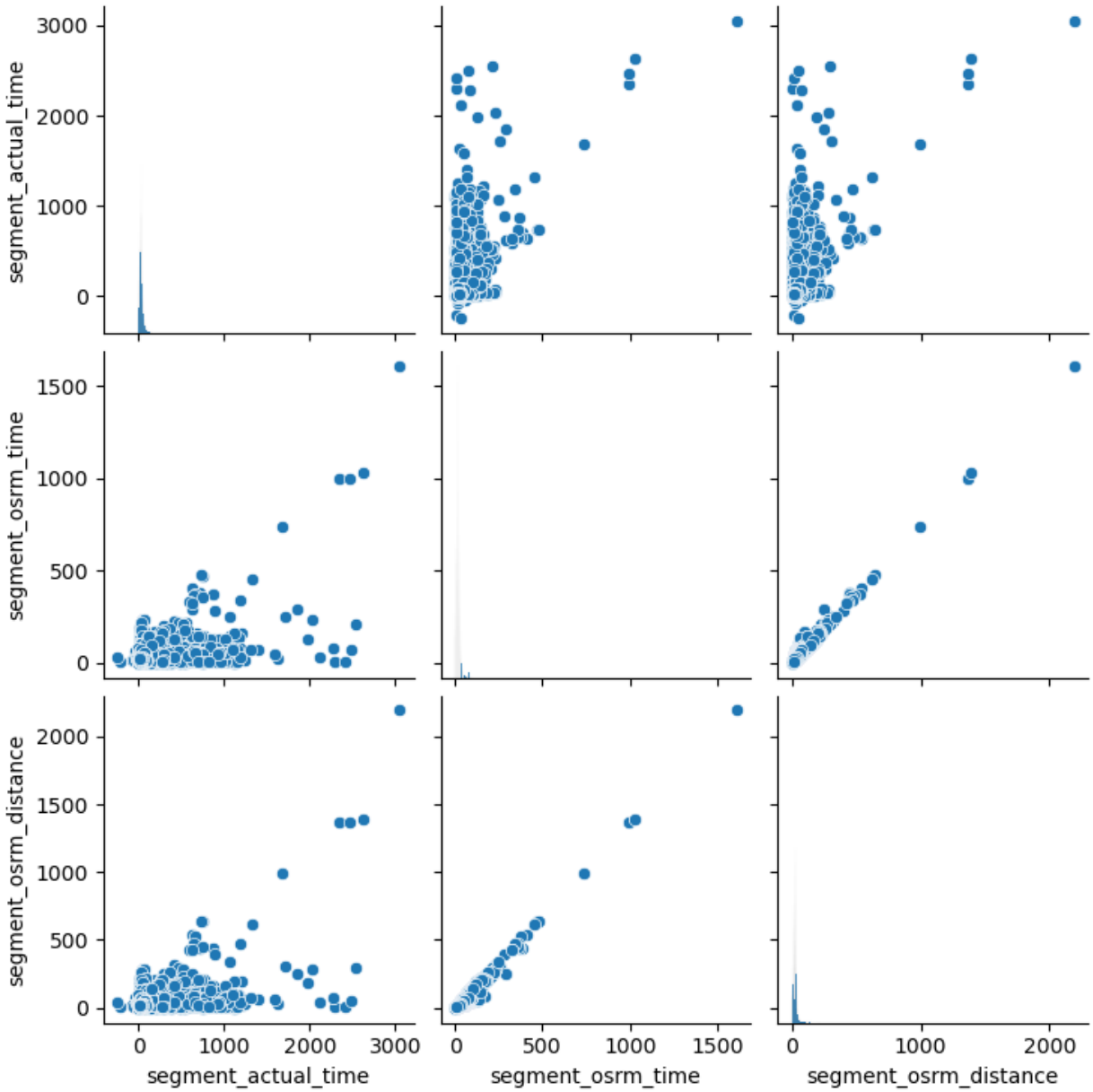
26368 rows × 16 columns

Soution -4

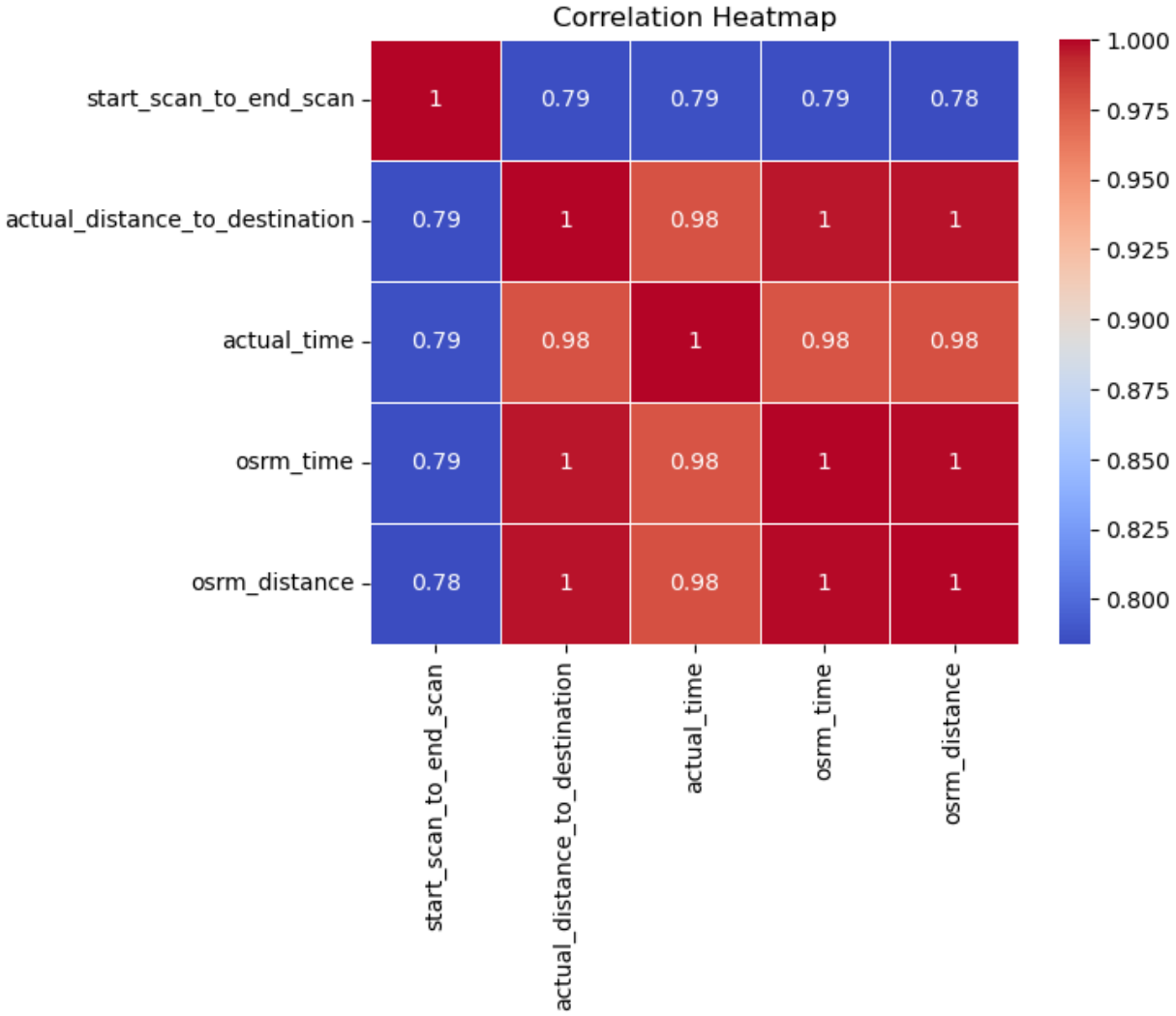
In [48]:
```
# Pairplot for numerical columns
sns.pairplot(df[['start_scan_to_end_scan', 'actual_distance_to_destination', 'actual_t
plt.show()
```



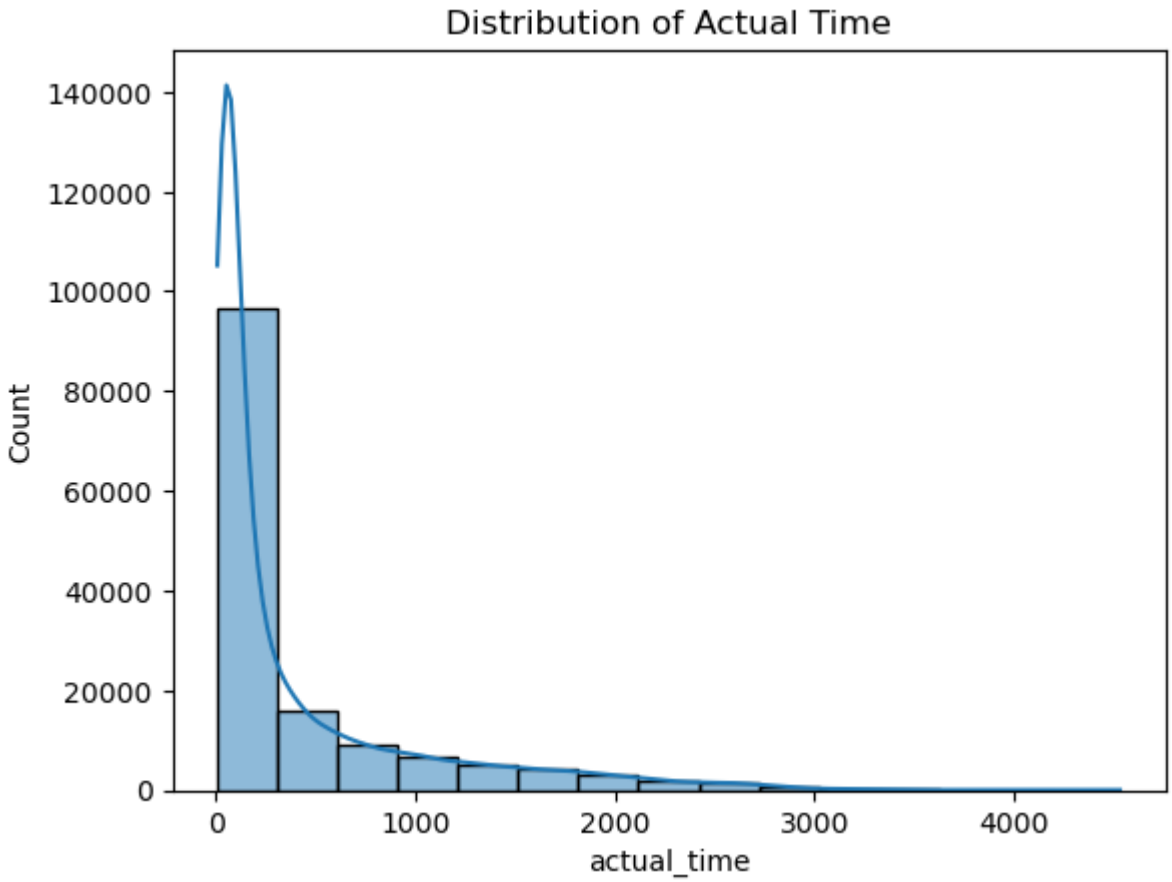
In [49]:
```
# Pairplot for numerical columns
sns.pairplot(df[['segment_actual_time', 'segment_osrm_time', 'segment_osrm_distance']]
plt.show()
```



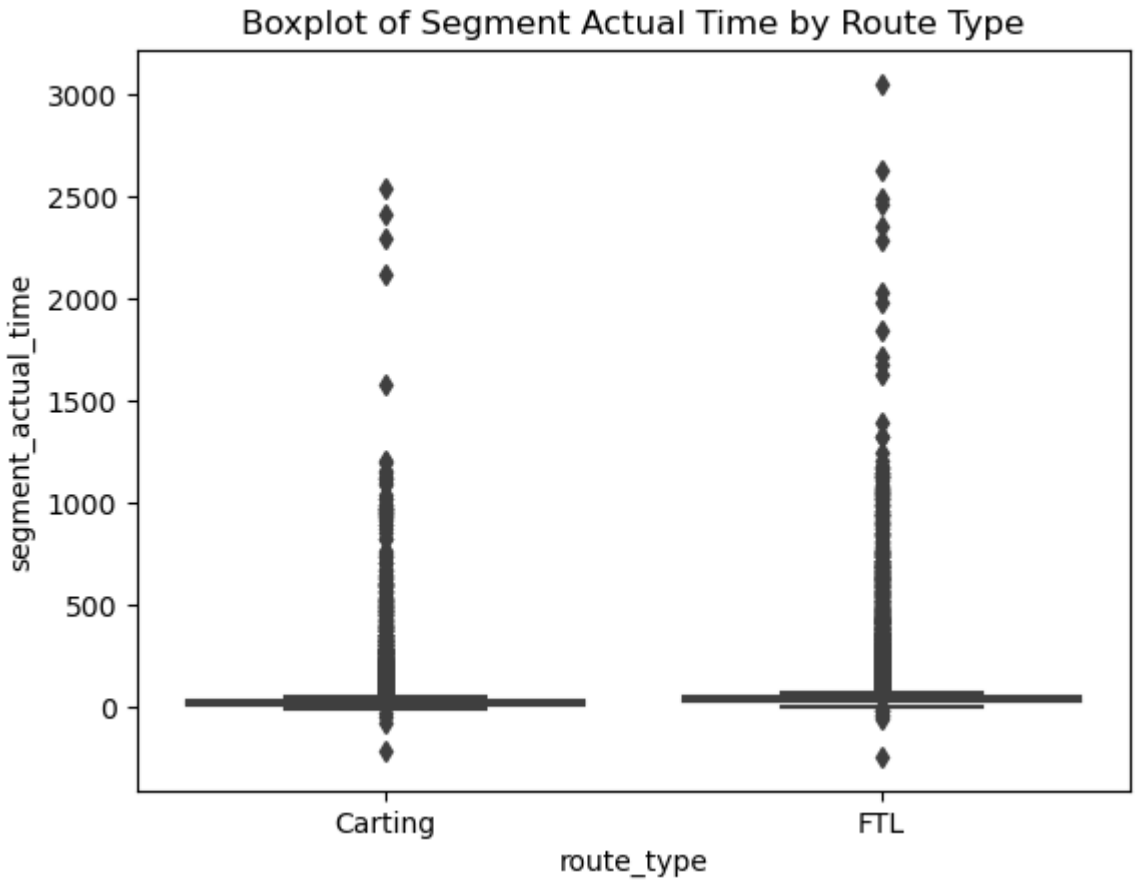
In [50]:
```
# Correlation heatmap
correlation_matrix = df[['start_scan_to_end_scan', 'actual_distance_to_destination', '
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Heatmap')
plt.show()
```



In [51]:
```
sns.histplot(df['actual_time'], bins=15, kde=True)
plt.title('Distribution of Actual Time')
plt.show()
```



In [52]:
```
# Boxplot for 'segment_actual_time' across different categories
sns.boxplot(x='route_type', y='segment_actual_time', data=df)
plt.title('Boxplot of Segment Actual Time by Route Type')
plt.show()
```



In [53]:
```
# Grouping specific columns by segment_key
grouped_data = df.groupby('segment_key').agg({
    'start_scan_to_end_scan': 'sum',
    'actual_distance_to_destination': 'sum',
    'actual_time': 'sum',
    'osrm_time': 'sum',
    'osrm_distance': 'sum',
    'segment_actual_time': 'last',
    'segment_osrm_time': 'last',
    'segment_osrm_distance': 'last'
}).reset_index()
```

In [54]:
```
grouped_data
```

Out[54]:

	segment_key	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm
0	-9222796547932529665	170.0	40.421223	96.0	
1	-9220630053641770878	132.0	47.177546	70.0	
2	-9218119143826161464	412.0	87.303874	136.0	
3	-9216891155464067624	1575.0	472.168572	1051.0	
4	-9216512112440739564	284.0	57.304936	116.0	
...	...	...	...	...	
26363	9219235042858240561	178809.0	36392.721633	77036.0	3
26364	9219328082127877630	4430.0	316.983741	621.0	
26365	9220089546824066172	546.0	49.508988	107.0	
26366	9220227007014954146	124.0	20.577676	56.0	
26367	9221140859336740686	396.0	93.075649	169.0	

26368 rows × 9 columns

In [55]:
```
create_segment_dict = {
    'trip_creation_time': 'first',  # Keep the first value
    'route_schedule_uuid': 'first',  # Keep the first value
    'route_type': 'first',  # Keep the first value
    'source_name': 'first',  # Keep the first value
    'destination_name': 'first',  # Keep the first value
    'od_start_time': 'first',  # Keep the first value
    'od_end_time': 'last',  # Keep the last value
    'start_scan_to_end_scan': 'sum',  # Aggregate by sum
    'actual_distance_to_destination': 'sum',  # Aggregate by sum
    'actual_time': 'sum',  # Aggregate by sum
    'osrm_time': 'sum',  # Aggregate by sum
    'osrm_distance': 'sum',  # Aggregate by sum
    'segment_actual_time': 'last',  # Keep the last value
    'segment_osrm_distance': 'last',  # Keep the last value
    'segment_osrm_time': 'last'  # Keep the last value
}

dictionary_grouped_data = df.groupby('segment_key').agg(create_segment_dict).reset_inc
```

In [56]:
```
dictionary_grouped_data
```

Out[56]:

	segment_key	trip_creation_time	route_schedule_uuid	route_type	source_na
0	-9222796547932529665	2018-10-03 19:22:39.293216	thanos::sroute:6e01eb09-11ca-4a88-839f-2fe0836...	Carting	Mumbai H (Maharash
1	-9220630053641770878	2018-09-15 01:58:24.742401	thanos::sroute:34330d20-1386-457b-8373-f94cc2f...	FTL	Bobbili_ColegR (Andhra Prade
2	-9218119143826161464	2018-09-22 21:00:28.082589	thanos::sroute:20f7c97f-614d-4501-96c7-fbd43d5...	Carting	Gurgaon_Bilaspur (Harya
3	-9216891155464067624	2018-09-22 00:28:35.499623	thanos::sroute:36941a6b-0e90-4582-a95a-96666dc...	Carting	Vaijiapur_YeolaRl (Maharash
4	-9216512112440739564	2018-10-01 00:02:31.857184	thanos::sroute:883e99fa-50a3-40e0-a2e2-9b12ed6...	FTL	Motihari_RajaBz (Bi
...	...	...	...	...	
26363	9219235042858240561	2018-09-27 17:57:56.350055	thanos::sroute:2c33e360-7e52-4d2c-a9db-fe24996...	FTL	Gurgaon_Bilaspur (Harya
26364	9219328082127877630	2018-10-03 10:53:58.045174	thanos::sroute:e2263654-53fa-4faa-ad8a-7956bca...	FTL	Hubli_Adargch (Karnata
26365	9220089546824066172	2018-09-28 23:06:33.589099	thanos::sroute:14870bd3-0ef2-4c83-8ba3-961539c...	Carting	Pune_Tathawd (Maharash
26366	9220227007014954146	2018-10-03 04:01:36.472494	thanos::sroute:9e7bb811-593f-47bc-ac49-ba03ed8...	Carting	Mumbai_Sanpa (Maharash
26367	9221140859336740686	2018-09-14 23:22:40.495686	thanos::sroute:17458102-0fe7-48eb-aefa-6b64e87...	Carting	Etah_CivlLin (Uttar Prade

26368 rows × 16 columns

In [57]:
```
sorted_segment = dictionary_grouped_data.sort_values(by=['segment_key', 'od_end_time'])
```

In [58]:
```
sorted_segment.head()
```

Out[58]:

	segment_key	trip_creation_time	route_schedule_uuid	route_type	source_name
0	-9222796547932529665	2018-10-03 19:22:39.293216	thanos::sroute:6e01eb09-11ca-4a88-839f-2fe0836...	Carting	Mumbai Hub (Maharashtra)
1	-9220630053641770878	2018-09-15 01:58:24.742401	thanos::sroute:34330d20-1386-457b-8373-f94cc2f...	FTL	Bobbili_ColegRd_D (Andhra Pradesh)
2	-9218119143826161464	2018-09-22 21:00:28.082589	thanos::sroute:20f7c97f-614d-4501-96c7-fbd43d5...	Carting	Gurgaon_Bilaspur_HB (Haryana)
3	-9216891155464067624	2018-09-22 00:28:35.499623	thanos::sroute:36941a6b-0e90-4582-a95a-96666dc...	Carting	Vaijiapur_YeolaRD_D (Maharashtra)
4	-9216512112440739564	2018-10-01 00:02:31.857184	thanos::sroute:883e99fa-50a3-40e0-a2e2-9b12ed6...	FTL	Motihari_RajaBzr_D (Bihar)

In [59]:
```
sorted_segment.tail()
```

Out[59]:

	segment_key	trip_creation_time	route_schedule_uuid	route_type	source_na
26363	9219235042858240561	2018-09-27 17:57:56.350055	thanos::sroute:2c33e360-7e52-4d2c-a9db-fe24996...	FTL	Gurgaon_Bilaspur_ (Haryai
26364	9219328082127877630	2018-10-03 10:53:58.045174	thanos::sroute:e2263654-53fa-4faa-ad8a-7956bca...	FTL	Hubli_Adargchi (Karnata
26365	9220089546824066172	2018-09-28 23:06:33.589099	thanos::sroute:14870bd3-0ef2-4c83-8ba3-961539c...	Carting	Pune_Tathawde (Maharasht
26366	9220227007014954146	2018-10-03 04:01:36.472494	thanos::sroute:9e7bb811-593f-47bc-ac49-ba03ed8...	Carting	Mumbai_Sanpad (Maharasht
26367	9221140859336740686	2018-09-14 23:22:40.495686	thanos::sroute:17458102-0fe7-48eb-aefa-6b64e87...	Carting	Etah_CivlLine (Uttar Prade

In [60]:
```
sorted_segment.shape
```

Out[60]:
```
(26368, 16)
```

Solution -2 Feature Engineering

In [61]:
```
# Creating new Column od_time_diff_hour
df['od_time_diff_hour']= df['od_end_time']-df['od_start_time']
```

In [103…
```
df['od_time_diff_hour'].value_counts()
```

Out[103]:
```
2 days 15:22:06.369411    81
2 days 06:19:31.076681    79
2 days 04:39:37.670702    79
2 days 06:40:30.775605    79
2 days 09:46:38.866877    79
                          ..
0 days 08:58:43.684394     1
0 days 00:52:24.115918     1
0 days 06:42:15.958669     1
0 days 01:48:00.975066     1
0 days 12:47:50.508442     1
Name: od_time_diff_hour, Length: 26369, dtype: int64
```

In [63]:
```
# Optionally, drop the original columns if required
# od_start_time = Trip start time
# od_end_time= Trip end time
df = df.drop(['od_start_time', 'od_end_time'], axis=1)
```

In [64]:
```
df.head()
```

Out[64]:

	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 × 22 columns

In [65]:
```
# Display the DataFrame with the new feature and without the original columns
print(df[['trip_uuid', 'segment_key', 'od_time_diff_hour']])
```

```
                      trip_uuid          segment_key      od_time_diff_hour
0       trip-153741093647649320 -2223622434856891013 0 days 01:26:12.818197
1       trip-153741093647649320 -2223622434856891013 0 days 01:26:12.818197
2       trip-153741093647649320 -2223622434856891013 0 days 01:26:12.818197
3       trip-153741093647649320 -2223622434856891013 0 days 01:26:12.818197
4       trip-153741093647649320 -2223622434856891013 0 days 01:26:12.818197
...                         ...                  ...                    ...
144862  trip-153746066843555182 -3952487073946094192 0 days 07:07:41.181838
144863  trip-153746066843555182 -3952487073946094192 0 days 07:07:41.181838
144864  trip-153746066843555182 -3952487073946094192 0 days 07:07:41.181838
144865  trip-153746066843555182 -3952487073946094192 0 days 07:07:41.181838
144866  trip-153746066843555182 -3952487073946094192 0 days 07:07:41.181838

[144867 rows x 3 columns]
```

Solution -7 Handling Categorical Variable

In [66]:
```
df[["dest_name_split", "dest_state_split"]] = df['destination_name'].str.split('(', n=
df[["source_name_split", "source_state_split"]]= df["source_name"].str.split("(", n=1,
```

In [67]:
```
df["destination_name"]
```

Out[67]:
```
0         Khambhat_MotvdDPP_D (Gujarat)
1         Khambhat_MotvdDPP_D (Gujarat)
2         Khambhat_MotvdDPP_D (Gujarat)
3         Khambhat_MotvdDPP_D (Gujarat)
4         Khambhat_MotvdDPP_D (Gujarat)
                    ...
144862    Gurgaon_Bilaspur_HB (Haryana)
144863    Gurgaon_Bilaspur_HB (Haryana)
144864    Gurgaon_Bilaspur_HB (Haryana)
144865    Gurgaon_Bilaspur_HB (Haryana)
144866    Gurgaon_Bilaspur_HB (Haryana)
Name: destination_name, Length: 144867, dtype: object
```

In [68]:
```
# To Remove ")" from each splitted state name of destination
df["dest_state_split"] = df["dest_state_split"].str.strip(')')
df["source_state_split"]= df["source_state_split"].str.strip(')')
```

In [69]:
```
df.head()
```

Out[69]:

	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 × 26 columns

In [70]:
```
df= df.drop(["source_name", "destination_name"], axis=1)
```

In [71]:
```
df.head()
```

Out[71]:

	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 [80]:
```
df['trip_creation_year'] = df["trip_creation_time"].dt.year
df['trip_creation_month'] = df["trip_creation_time"].dt.month
df['trip_creation_day'] = df["trip_creation_time"].dt.day
df['trip_creation_hour'] = df["trip_creation_time"].dt.hour
```

```
df['trip_creation_minute'] = df["trip_creation_time"].dt.minute
df['trip_creation_second'] = df["trip_creation_time"].dt.second
```

In [84]:
```
df = df.drop(["trip_creation_time"], axis=1)
df.head()
```

Out[84]:

	data	route_schedule_uuid	route_type	trip_uuid	source_center	destination_center
0	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB
1	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB
2	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB
3	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB
4	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB

5 rows × 29 columns

In [72]:
```
df.dtypes
```

Out[72]:
```
data                                  object
trip_creation_time                datetime64[ns]
route_schedule_uuid                   object
route_type                            object
trip_uuid                             object
source_center                         object
destination_center                    object
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
segment_key                            int64
segment_actual_time_cumsum           float64
segment_osrm_distance_cumsum         float64
segment_osrm_time_cumsum             float64
od_time_diff_hour                timedelta64[ns]
dest_name_split                       object
dest_state_split                      object
source_name_split                     object
source_state_split                    object
dtype: object
```

Solution - 8 :Column Normalization (Min-Max Scaling)

In [73]:
```
#Min-max scaling is suitable when you want to preserve the original distribution and c
from sklearn.preprocessing import MinMaxScaler

# Assuming for specified columns
columns_to_normalize = ['actual_time', 'osrm_time', 'actual_distance_to_destination',

scaler = MinMaxScaler()
df[columns_to_normalize] = scaler.fit_transform(df[columns_to_normalize])
```

In [74]:
```
df.head()
```

Out[74]:

	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 [77]:
```
df[['actual_time', 'osrm_time', 'actual_distance_to_destination', 'osrm_distance']]
```

Out[77]:

	actual_time	osrm_time	actual_distance_to_destination	osrm_distance
0	0.001105	0.002976	0.000748	0.001276
1	0.003316	0.008333	0.005180	0.005488
2	0.006854	0.013095	0.009715	0.010155
3	0.011718	0.020238	0.014135	0.015775
4	0.013044	0.022619	0.015839	0.019511
...	...	...	...	...
144862	0.018793	0.032143	0.018900	0.025427
144863	0.024541	0.041667	0.023505	0.033090
144864	0.028963	0.048810	0.029797	0.038014
144865	0.032943	0.054762	0.033715	0.044132
144866	0.092195	0.052976	0.031817	0.034405

144867 rows × 4 columns

column standardization (or z-score normalization)

In [78]:
```
# z-score normalization is appropriate when you want to center the data around zero an
from sklearn.preprocessing import StandardScaler

# Assuming for specified Columns
columns_to_standardize = ['actual_time', 'osrm_time', 'actual_distance_to_destination'

scaler = StandardScaler()
df[columns_to_standardize] = scaler.fit_transform(df[columns_to_standardize])
```

In [79]:
```
df[['actual_time', 'osrm_time', 'actual_distance_to_destination', 'osrm_distance']]
```

Out[79]:

	actual_time	osrm_time	actual_distance_to_destination	osrm_distance
0	-0.673677	-0.658642	-0.648246	-0.647814
1	-0.656958	-0.629422	-0.623604	-0.624640
2	-0.630207	-0.603449	-0.598385	-0.598958
3	-0.593424	-0.564489	-0.573802	-0.568034
4	-0.583392	-0.551502	-0.564329	-0.547479
...	...	...	...	...
144862	-0.539921	-0.499556	-0.547308	-0.514923
144863	-0.496450	-0.447610	-0.521701	-0.472762
144864	-0.463011	-0.408650	-0.486711	-0.445666
144865	-0.432916	-0.376183	-0.464921	-0.412000
144866	0.015169	-0.385923	-0.475477	-0.465521

144867 rows × 4 columns

Solution 9

In [85]:
```
df["dest_name_split"].value_counts()
```

Out[85]:
```
Gurgaon_Bilaspur_HB      15192
Bangalore_Nelmngla_H     11019
Bhiwandi_Mankoli_HB       5492
Hyderabad_Shamshbd_H      5142
Kolkata_Dankuni_HB        4892
                         ...
Hyd_Trimulgherry_Dc          1
Vijayawada                   1
Baghpat_Barout_D             1
Mumbai_Sanpada_CP            1
Basta_Central_DPP_1          1
Name: dest_name_split, Length: 1469, dtype: int64
```

In [87]:
```
# Most order going to Karnataka then Haryana then Maharastra
df["dest_state_split"].value_counts()
```

Out[87]:
```
Karnataka                21065
Haryana                  20622
Maharashtra              18196
West Bengal               8499
Telangana                 8205
Tamil Nadu                8058
Uttar Pradesh             7834
Gujarat                   6714
Rajasthan                 6361
Andhra Pradesh            6265
Delhi                     5754
Punjab                    5105
Madhya Pradesh            4345
Bihar                     4238
Orissa                    3234
Jharkhand                 2552
Kerala                    2230
Assam                     2000
Uttarakhand                893
Goa                        580
Himachal Pradesh           553
Chandigarh                 389
Chhattisgarh               229
Arunachal Pradesh          211
Jammu & Kashmir            201
Pondicherry                154
Meghalaya                   37
Dadra and Nagar Haveli      34
Mizoram                     31
Tripura                      9
Nagaland                     7
Daman & Diu                  1
Name: dest_state_split, dtype: int64
```

In [88]:
```
# Source State where the most Number of order being placed
df["source_state_split"].value_counts()
```

Out[88]:
```
Haryana                   27499
Maharashtra               21401
Karnataka                 19578
Tamil Nadu                 7494
Gujarat                    7202
Uttar Pradesh              7137
Telangana                  6496
West Bengal                5963
Andhra Pradesh             5539
Rajasthan                  5267
Punjab                     4704
Delhi                      4398
Bihar                      4190
Madhya Pradesh             4021
Assam                      2875
Jharkhand                  2597
Kerala                     2413
Orissa                     2094
Uttarakhand                1162
Himachal Pradesh            587
Goa                         514
Chandigarh                  507
Arunachal Pradesh           245
Chhattisgarh                229
Jammu & Kashmir             226
Meghalaya                    86
Pondicherry                  49
Nagaland                     40
Dadra and Nagar Haveli       30
Mizoram                      26
Tripura                       5
Name: source_state_split, dtype: int64
```

In [90]:
```
# Addresses from where most order placed
df["source_name_split"].value_counts()
```

Out[90]:
```
Gurgaon_Bilaspur_HB         23347
Bangalore_Nelmngla_H         9975
Bhiwandi_Mankoli_HB          9088
Pune_Tathawde_H              4061
Hyderabad_Shamshbd_H         3340
                            ...
Shahjhnpur_NavdaCln_D           1
Soro_UttarDPP_D                 1
Kayamkulam_Bhrnikvu_D           1
Krishnanagar_AnadiDPP_D         1
Faridabad_Old                   1
Name: source_name_split, Length: 1499, dtype: int64
```

In [91]:
```
# Busiest souce center
df["source_center"].value_counts()
```

Out[91]:
```
IND000000ACB    23347
IND562132AAA     9975
IND421302AAG     9088
IND411033AAA     4061
IND501359AAE     3340
                ...
IND741121AAA        1
IND207123AAA        1
IND242001AAA        1
IND222001AAA        1
IND741101AAB        1
Name: source_center, Length: 1508, dtype: int64
```

In [92]:
```
# Busiest destination center
df["destination_center"].value_counts()
```

Out[92]:
```
IND000000ACB    15192
IND562132AAA    11019
IND421302AAG     5492
IND501359AAE     5142
IND712311AAA     4892
                ...
IND520011AAA        1
IND741201AAC        1
IND400705AAA        1
IND110046AAA        1
IND504215AAA        1
Name: destination_center, Length: 1481, dtype: int64
```

In [93]:
```
df['trip_creation_year'].value_counts()
```

Out[93]:
```
2018    144867
Name: trip_creation_year, dtype: int64
```

In [95]:
```
df['trip_creation_month'].value_counts()
```

Out[95]:
```
9     127349
10     17518
Name: trip_creation_month, dtype: int64
```

In [102…
```
df['trip_creation_hour'].value_counts()
```

Out[102]:
```
22    12255
20    10329
19    10197
23     9343
1      8771
21     8735
0      8299
18     7783
2      7321
4      6639
5      6183
17     5976
3      4976
6      4407
13     4294
15     4274
14     4273
16     3862
8      3596
10     2880
7      2708
11     2691
9      2612
12     2463
Name: trip_creation_hour, dtype: int64
```

In [97]:
```
df.describe()
```

Out[97]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distan
count	144867.000000	1.448670e+05	1.448670e+05	1.448670e+05	1.448670e+(
mean	961.262986	-1.177150e-16	2.158109e-17	-6.748996e-17	5.689561e-1
std	1037.012769	1.000003e+00	1.000003e+00	1.000003e+00	1.000003e+(
min	20.000000	-6.524076e-01	-6.820372e-01	-6.748750e-01	-6.548359e-(
25%	161.000000	-6.107952e-01	-6.118150e-01	-6.066954e-01	-6.051907e-(
50%	449.000000	-4.868181e-01	-4.763865e-01	-4.865695e-01	-4.897572e-(
75%	1634.000000	1.525716e-01	1.606290e-01	1.400335e-01	1.387307e-(
max	7898.000000	4.908490e+00	6.880224e+00	4.779493e+00	4.847640e+(

In [98]:
```
df['trip_creation_hour'].value_counts()
```

Out[98]:
```
22    12255
20    10329
19    10197
23     9343
1      8771
21     8735
0      8299
18     7783
2      7321
4      6639
5      6183
17     5976
3      4976
6      4407
13     4294
15     4274
14     4273
16     3862
8      3596
10     2880
7      2708
11     2691
9      2612
12     2463
Name: trip_creation_hour, dtype: int64
```

In [99]:
```
df['trip_creation_day'].value_counts()
```

Out[99]:
```
21    7639
15    7366
18    7354
20    7281
25    7168
13    7110
26    7059
17    7006
12    6995
22    6883
14    6809
16    6688
24    6661
19    6577
23    6262
3     6101
27    6090
1     5978
28    5794
29    5687
2     5439
30    4920
Name: trip_creation_day, dtype: int64
```

In [100…
```
df.head()
```

Out[100]:

	data	route_schedule_uuid	route_type	trip_uuid	source_center	destination_center
0	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB
1	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB
2	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB
3	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB
4	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620AAB

5 rows × 29 columns

In [101...

```
average_distance = df['actual_distance_to_destination'].mean()
average_time = df['actual_time'].mean()

print(f'Average Distance: {average_distance}')
print(f'Average Time Taken: {average_time}')
```

```
Average Distance: -1.1771504661684445e-16
Average Time Taken: 2.1581091879754814e-17
```

Solution 10 Recommendation and Finding

In [ ]:

```
# The Given data is only from 2018
# And All the trip made only in 9th(September) and 10th(October) month.
# Most people hour for the trip creation is evening and luch time
22    12255
20    10329
19    10197
23     9343
1      8771


# Most popular source station
Haryana                   27499
Maharashtra               21401
Karnataka                 19578
Tamil Nadu                 7494
Gujarat                    7202

#Least popular source station these states need proper strategy for Delhivery business
Pondicherry                 49
Nagaland                    40
Dadra and Nagar Haveli      30
```

```
Mizoram                      26
Tripura                       5

# # Most popular Destination station
Karnataka                 21065
Haryana                   20622
Maharashtra               18196
West Bengal                8499
Telangana                  8205

#Least popular Destination station these states need proper strategy for Delhivery bus
Dadra and Nagar Haveli        34
Mizoram                       31
Tripura                        9
Nagaland                       7
Daman & Diu                    1

# There are only 2 type of route_type
1. Full Truck Load (Most Used)
2. Carting

# Difference between Trip start time and end time is max 2 day and their order count
2 days 15:22:06.369411    81
2 days 06:19:31.076681    79
2 days 04:39:37.670702    79
2 days 06:40:30.775605    79
2 days 09:46:38.866877    79

Mimimum time Difference
0 days 08:58:43.684394     1
0 days 00:52:24.115918     1
0 days 06:42:15.958669     1
0 days 01:48:00.975066     1
0 days 12:47:50.508442     1

# Segement wise Trip Data is created by unique segment key.
```

Thank-You

In [ ]: