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

path_file = '/content/delhivery_data.csv'

df = pd.read_csv(path_file)
df
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
...	...	...	...	...	...	...	...	
30046	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562
30047	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562
30048	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562
30049	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562
30050	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562

30051 rows × 24 columns

1. Data cleaning and exploration

```
#drop rows with missing values :

df_cleaned = df.dropna()
df_cleaned
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
...	...	...	...	...	...	...	...	
30045	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562
30046	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562
30047	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562
30048	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562
30049	training	2018-09-16 06:21:40.370304	thanos::sroute:dca6268f-741a-4d1a-b1b0-aab1309...	FTL	trip-153707890037004730	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND562

29902 rows × 24 columns

```
df['route_type'] = df['route_type'].fillna(df['route_type'].mode()[0])
df['route_type']
```

	route_type
0	Carting
1	Carting
2	Carting
3	Carting
4	Carting
...	...
144862	Carting
144863	Carting
144864	Carting
144865	Carting
144866	Carting

144867 rows × 1 columns

dtype: object

1.b Analyze the structure of the data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30051 entries, 0 to 30050
Data columns (total 24 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   data                            30051 non-null  object 
 1   trip_creation_time              30051 non-null  object 
 2   route_schedule_uuid             30051 non-null  object 
 3   route_type                      30051 non-null  object 
 4   trip_uuid                       30051 non-null  object 
 5   source_center                   30051 non-null  object 
 6   source_name                     29963 non-null  object 
 7   destination_center              30051 non-null  object 
 8   destination_name                29991 non-null  object 
 9   od_start_time                   30050 non-null  object 
 10  od_end_time                     30050 non-null  object 
 11  start_scan_to_end_scan          30050 non-null  float64
 12  is_cutoff                       30050 non-null  object 
 13  cutoff_factor                   30050 non-null  float64
 14  cutoff_timestamp                30050 non-null  object 
 15  actual_distance_to_destination  30050 non-null  float64
 16  actual_time                     30050 non-null  float64
 17  osrm_time                       30050 non-null  float64
 18  osrm_distance                   30050 non-null  float64
 19  factor                          30050 non-null  float64
 20  segment_actual_time             30050 non-null  float64
 21  segment_osrm_time               30050 non-null  float64
 22  segment_osrm_distance           30050 non-null  float64
 23  segment_factor                  30050 non-null  float64
dtypes: float64(11), object(13)
memory usage: 5.5+ MB
```

```
df.isnull().sum()
```

```
                                  0
              data                0
        trip_creation_time        0
       route_schedule_uuid        0
           route_type             0
            trip_uuid             0
          source_center           0
           source_name           88
        destination_center        0
         destination_name        60
          od_start_time           1
           od_end_time            1
      start_scan_to_end_scan      1
            is_cutoff             1
          cutoff_factor           1
         cutoff_timestamp         1
 actual_distance_to_destination   1
           actual_time            1
            osrm_time             1
          osrm_distance           1
              factor              1
       segment_actual_time        1
        segment_osrm_time         1
      segment_osrm_distance       1
          segment_factor          1

dtype: int64
```

```
df.describe()
```

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	factor	se
count	30050.000000	30050.000000	30050.000000	30050.000000	30050.000000	30050.000000	30050.000000	
mean	889.489651	217.402363	218.523992	386.967854	200.333145	266.046534	2.093752	
std	988.504498	332.861350	333.068487	567.799670	298.138106	407.047990	1.435717	
min	25.000000	9.000000	9.000267	9.000000	6.000000	9.101900	0.250000	
25%	150.000000	22.000000	23.043696	50.000000	26.000000	29.089850	1.597958	
50%	407.000000	55.000000	56.245459	120.500000	59.000000	72.470200	1.852459	
75%	1377.000000	242.000000	244.387826	456.000000	229.000000	300.930325	2.222222	
max	3702.000000	1722.000000	1722.009755	3382.000000	1611.000000	2191.166400	77.387097	

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

```
path_file = '/content/delhivery_data.csv'

df = pd.read_csv(path_file)
df
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388
...	...	...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	Sonipat_Kundli_H (Haryana)	IND000

144867 rows × 24 columns

```
# List unique values for categorical columns :

df['source_name'].unique()
```

```
array(['Anand_VUNagar_DC (Gujarat)', 'Khambhat_MotvdDPP_D (Gujarat)',
       'Bhiwandi_Mankoli_HB (Maharashtra)', ...,
       'Dwarka_StnRoad_DC (Gujarat)', 'Bengaluru_Nelmngla_L (Karnataka)',
       'Kulithalai_AnnaNGR_D (Tamil Nadu)'], dtype=object)
```

```
df['route_type'].unique()
```

```
array(['Carting', 'FTL'], dtype=object)
```

1.C Merging

```
# Group by 'Trip_uuid', 'Source_ID', and 'Destination_ID'
grouped = df.groupby(['trip_uuid', 'source_center', 'destination_center'])
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f7ba0752680>
```

```
 # Aggregation

 aggregated_df = grouped.agg({'start_scan_to_end_scan':'sum',
 'is_cutoff':'max',
 'cutoff_factor':'sum',
  'cutoff_timestamp': 'max',
    'actual_distance_to_destination': 'min',
    'actual_time': 'sum',
    'osrm_time': 'sum',
    'osrm_distance': 'sum',
    'factor': 'mean',
    'segment_actual_time': 'sum',
    'segment_osrm_time': 'sum',
    'segment_osrm_distance': 'sum',
    'segment_factor': 'mean',
    'source_name': 'first',
    'destination_name': 'first',
    'od_start_time': 'first',
    'od_end_time': 'last',
    'trip_creation_time': 'first',
    'route_schedule_uuid': 'first',
    'route_type': 'first'
}).reset_index()
```

```
aggregated_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26368 entries, 0 to 26367
Data columns (total 23 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   trip_uuid                       26368 non-null  object
 1   source_center                   26368 non-null  object
 2   destination_center              26368 non-null  object
 3   start_scan_to_end_scan          26368 non-null  float64
 4   is_cutoff                       26368 non-null  bool
 5   cutoff_factor                   26368 non-null  int64
 6   cutoff_timestamp                26368 non-null  object
 7   actual_distance_to_destination  26368 non-null  float64
 8   actual_time                     26368 non-null  float64
 9   osrm_time                       26368 non-null  float64
 10  osrm_distance                   26368 non-null  float64
 11  factor                          26368 non-null  float64
 12  segment_actual_time             26368 non-null  float64
 13  segment_osrm_time               26368 non-null  float64
 14  segment_osrm_distance           26368 non-null  float64
 15  segment_factor                  26368 non-null  float64
 16  source_name                     26302 non-null  object
 17  destination_name                26287 non-null  object
 18  od_start_time                   26368 non-null  object
 19  od_end_time                     26368 non-null  object
 20  trip_creation_time              26368 non-null  object
 21  route_schedule_uuid             26368 non-null  object
 22  route_type                      26368 non-null  object
dtypes: bool(1), float64(10), int64(1), object(11)
memory usage: 4.5+ MB
```

```
aggregated_df.describe()
```

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	factor	s
count	26368.000000	26368.000000	26368.000000	26368.000000	26368.000000	26368.000000	26368.000000	
mean	5281.222884	1279.709231	18.015383	2290.618932	1175.002086	1564.546549	2.366245	
std	24795.814754	6497.610268	20.376645	11238.249896	5893.392945	7966.272378	2.165496	
min	22.000000	9.000000	9.000045	9.000000	6.000000	9.072900	0.338322	
25%	204.000000	46.000000	9.638582	91.000000	47.000000	56.434800	1.565830	
50%	462.000000	81.000000	22.001099	181.000000	90.000000	100.604350	1.907209	
75%	1272.000000	201.000000	22.803610	450.000000	219.000000	251.385750	2.461538	
max	341880.000000	84757.000000	1722.045544	167920.000000	76953.000000	102415.868000	70.000000	

2.Build some features to prepare the data for actual analysis. Extract features from the below fields:

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

path_file = '/content/delhivery_data.csv'

df = pd.read_csv(path_file)

# Function to extract city, place code, and state

def extract_location_info(df, column_name, prefix):
    # Extract city
    df[f'{prefix}_city'] = df[column_name].str.extract(r'([^\-]+)')
    # Extract place code
    df[f'{prefix}_place_code'] = df[column_name].str.extract(r'\-([A-Za-z0-9]+)')
    # Extract state
    df[f'{prefix}_state'] = df[column_name].str.extract(r'\((\w+)\)')
    return df

# Extract features from source_name

df = extract_location_info(df, 'source_name', 'source')

# Extract features from destination_name

df = extract_location_info(df, 'destination_name', 'destination')

# Convert trip_creation_time to datetime

df['trip_creation_time'] = pd.to_datetime(df['trip_creation_time'])

# Extract year, month, and day

df['trip_year'] = df['trip_creation_time'].dt.year
df['trip_month'] = df['trip_creation_time'].dt.month
df['trip_day'] = df['trip_creation_time'].dt.day

# Display the DataFrame with new features

print(df[['source_city', 'source_place_code', 'source_state',
          'destination_city', 'destination_place_code', 'destination_state',
          'trip_year', 'trip_month', 'trip_day']])
```

```
                       source_city source_place_code source_state  \
0       Anand_VUNagar_DC (Gujarat)               NaN      Gujarat
1       Anand_VUNagar_DC (Gujarat)               NaN      Gujarat
2       Anand_VUNagar_DC (Gujarat)               NaN      Gujarat
3       Anand_VUNagar_DC (Gujarat)               NaN      Gujarat
4       Anand_VUNagar_DC (Gujarat)               NaN      Gujarat
...                            ...               ...          ...
```

```
144862  Sonipat_Kundli_H (Haryana)               NaN      Haryana
144863  Sonipat_Kundli_H (Haryana)               NaN      Haryana
144864  Sonipat_Kundli_H (Haryana)               NaN      Haryana
144865  Sonipat_Kundli_H (Haryana)               NaN      Haryana
144866  Sonipat_Kundli_H (Haryana)               NaN      Haryana

                     destination_city destination_place_code  \
0       Khambhat_MotvdDPP_D (Gujarat)                    NaN   
1       Khambhat_MotvdDPP_D (Gujarat)                    NaN   
2       Khambhat_MotvdDPP_D (Gujarat)                    NaN   
3       Khambhat_MotvdDPP_D (Gujarat)                    NaN   
4       Khambhat_MotvdDPP_D (Gujarat)                    NaN   
...                               ...                    ...   
144862  Gurgaon_Bilaspur_HB (Haryana)                    NaN   
144863  Gurgaon_Bilaspur_HB (Haryana)                    NaN   
144864  Gurgaon_Bilaspur_HB (Haryana)                    NaN   
144865  Gurgaon_Bilaspur_HB (Haryana)                    NaN   
144866  Gurgaon_Bilaspur_HB (Haryana)                    NaN   

       destination_state  trip_year  trip_month  trip_day  
0                Gujarat       2018           9        20  
1                Gujarat       2018           9        20  
2                Gujarat       2018           9        20  
3                Gujarat       2018           9        20  
4                Gujarat       2018           9        20  
...                  ...        ...         ...       ...  
144862           Haryana       2018           9        20  
144863           Haryana       2018           9        20  
144864           Haryana       2018           9        20  
144865           Haryana       2018           9        20  
144866           Haryana       2018           9        20  

[144867 rows x 9 columns]
```

3.In-depth analysis and feature engineering

```
from scipy import stats

# Calculate the time taken between od_start_time and od_end_time

df['od_start_time'] = pd.to_datetime(df['od_start_time'])
df['od_end_time'] = pd.to_datetime(df['od_end_time'])
df['od_time_diff'] = (df['od_end_time'] - df['od_start_time']).dt.total_seconds() / 60  # in minutes
```

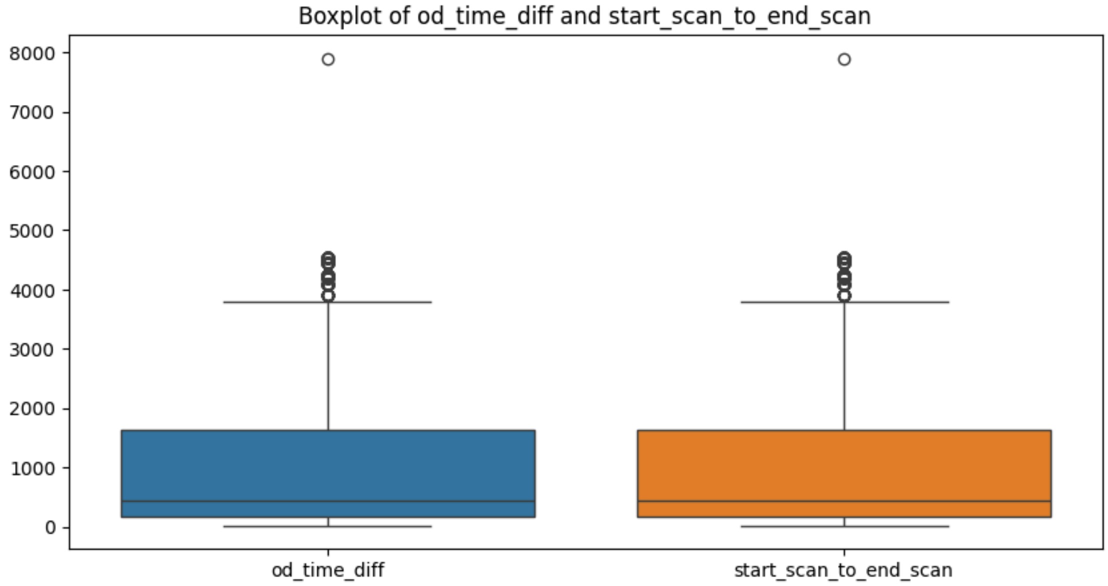
```
# Drop the original columns

df.drop(columns=['od_start_time', 'od_end_time'], inplace=True)
```

```
#Compare `od_time_diff` and `start_scan_to_end_scan` using hypothesis testing

t_stat, p_value = stats.ttest_rel(df['od_time_diff'], df['start_scan_to_end_scan'])
print(f"T-statistic: {t_stat}, P-value: {p_value}")
```

```
T-statistic: 651.1832057297116, P-value: 0.0
```

```
# Visual analysis (Box plot)

plt.figure(figsize=(10, 5))
sns.boxplot(data=df[['od_time_diff', 'start_scan_to_end_scan']])
plt.title('Boxplot of od_time_diff and start_scan_to_end_scan')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to p
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be remov
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to p
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be remov
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```

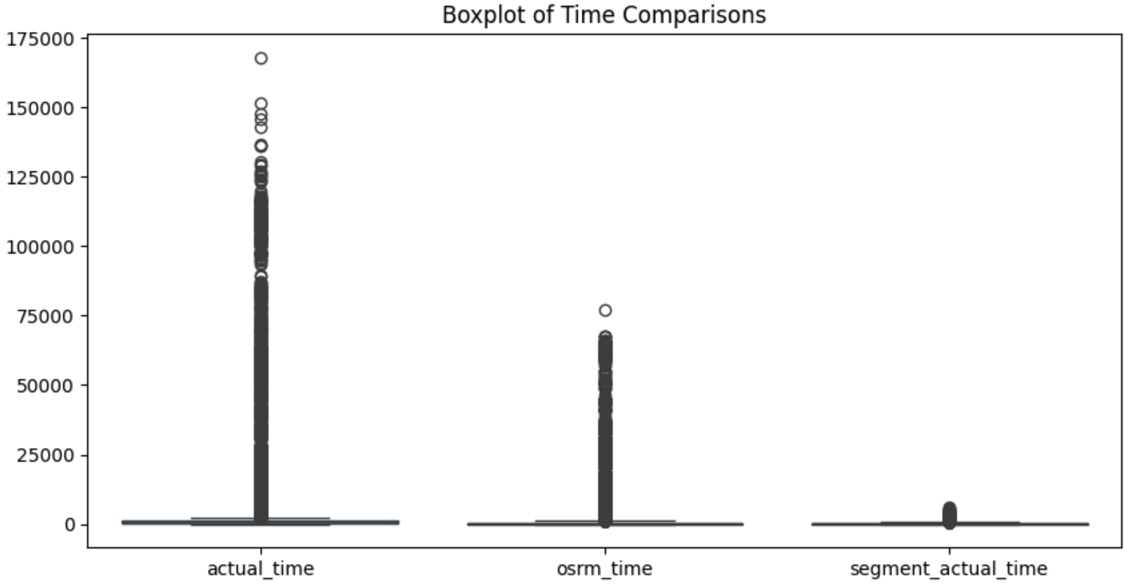


```
# Aggregation and hypothesis testing for time and distance comparisons

# Aggregate by trip_uuid

aggregated_df = df.groupby('trip_uuid').agg({
    'actual_time': 'sum',
    'osrm_time': 'sum',
    'segment_actual_time': 'sum',
    'osrm_distance': 'sum',
    'segment_osrm_distance': 'sum',
    'segment_osrm_time': 'sum'
}).reset_index()

# Actual time vs OSRM time

t_stat_osrm, p_value_osrm = stats.ttest_rel(aggregated_df['actual_time'], aggregated_df['osrm_time'])
print(f"T-statistic (actual vs osrm time): {t_stat_osrm}, P-value: {p_value_osrm}")

# Actual time vs Segment actual time

t_stat_seg_time, p_value_seg_time = stats.ttest_rel(aggregated_df['actual_time'], aggregated_df['segment_actual_time'])
print(f"T-statistic (actual vs segment actual time): {t_stat_seg_time}, P-value: {p_value_seg_time}")

# OSRM distance vs Segment OSRM distance

t_stat_osrm_dist, p_value_osrm_dist = stats.ttest_rel(aggregated_df['osrm_distance'], aggregated_df['segment_osrm_distance'])
print(f"T-statistic (osrm vs segment osrm distance): {t_stat_osrm_dist}, P-value: {p_value_osrm_dist}")
```

```
T-statistic (actual vs osrm time): 32.468089449426905, P-value: 1.8633294618952604e-223
T-statistic (actual vs segment actual time): 30.75550616001704, P-value: 2.077325421800874e-201
T-statistic (osrm vs segment osrm distance): 30.03031541377046, P-value: 2.1753879024067997e-192
```

```
# Visual analysis (scatter plots or box plots)

plt.figure(figsize=(10, 5))
sns.boxplot(data=aggregated_df[['actual_time', 'osrm_time', 'segment_actual_time']])
plt.title('Boxplot of Time Comparisons')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to p
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be remov
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to p
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be remov
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to p
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be remov
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



```
# Outlier detection using IQR method

def handle_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[column] = np.where((df[column] < lower_bound) | (df[column] > upper_bound), np.nan, df[column])
    return df


# Apply on all numerical variables (replace 'column_name' with actual columns)

for column in ['actual_time', 'osrm_time', 'segment_actual_time', 'osrm_distance', 'segment_osrm_distance']:
    df = handle_outliers(df, column)


from sklearn.preprocessing import OneHotEncoder

# One-hot encoding for categorical variables (e.g., route_type)

encoder = OneHotEncoder(sparse_output=False)
route_type_encoded = encoder.fit_transform(df[['route_type']])
route_type_df = pd.DataFrame(route_type_encoded, columns=encoder.get_feature_names_out(['route_type']))
df = pd.concat([df, route_type_df], axis=1)
df.drop(columns=['route_type'], inplace=True)


from sklearn.preprocessing import MinMaxScaler

# Normalize/Standardize numerical features

scaler = MinMaxScaler()
numerical_columns = ['actual_time', 'osrm_time', 'segment_actual_time', 'osrm_distance', 'segment_osrm_distance']
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])

# Result
print(df.head())
```

```
       data         trip_creation_time  \
0  training 2018-09-20 02:35:36.476840
1  training 2018-09-20 02:35:36.476840
2  training 2018-09-20 02:35:36.476840
3  training 2018-09-20 02:35:36.476840
4  training 2018-09-20 02:35:36.476840

                                route_schedule_uuid                trip_uuid  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...  trip-153741093647649320
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...  trip-153741093647649320
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...  trip-153741093647649320
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...  trip-153741093647649320
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...  trip-153741093647649320

  source_center                 source_name destination_center  \
0  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB
1  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB
2  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB
3  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB
4  IND388121AAA  Anand_VUNagar_DC (Gujarat)       IND388620AAB

                destination_name  start_scan_to_end_scan  is_cutoff  ...  \
0  Khambhat_MotvdDPP_D (Gujarat)                    86.0       True  ...
1  Khambhat_MotvdDPP_D (Gujarat)                    86.0       True  ...
2  Khambhat_MotvdDPP_D (Gujarat)                    86.0       True  ...
3  Khambhat_MotvdDPP_D (Gujarat)                    86.0       True  ...
4  Khambhat_MotvdDPP_D (Gujarat)                    86.0      False  ...

   source_state              destination_city  destination_place_code  \
0       Gujarat  Khambhat_MotvdDPP_D (Gujarat)                     NaN
1       Gujarat  Khambhat_MotvdDPP_D (Gujarat)                     NaN
2       Gujarat  Khambhat_MotvdDPP_D (Gujarat)                     NaN
3       Gujarat  Khambhat_MotvdDPP_D (Gujarat)                     NaN
4       Gujarat  Khambhat_MotvdDPP_D (Gujarat)                     NaN

   destination_state  trip_year  trip_month  trip_day  od_time_diff  \
0            Gujarat       2018           9        20     86.213637
1            Gujarat       2018           9        20     86.213637
2            Gujarat       2018           9        20     86.213637
3            Gujarat       2018           9        20     86.213637
4            Gujarat       2018           9        20     86.213637

   route_type_Carting  route_type_FTL
0                 1.0             0.0
1                 1.0             0.0
2                 1.0             0.0
3                 1.0             0.0
4                 1.0             0.0

[5 rows x 33 columns]
```

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

path_file = '/content/delhivery_data.csv'

df = pd.read_csv(path_file)


# Most frequent sources:

most_orders_from_source = df['source_name'].value_counts().head(10)
print("Most orders from source:\n", most_orders_from_source)

# Most frequent destinations:

most_orders_from_destination = df['destination_name'].value_counts().head(10)
print("Most orders to destination:\n", most_orders_from_destination)
```

```
Most orders from source:
 source_name
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
Kolkata_Dankuni_HB (West Bengal)      2612
Chandigarh_Mehmdpur_H (Punjab)        2450
Surat_HUB (Gujarat)                   2189
```

```
Delhi_Airport_H (Delhi)               2013
Bengaluru_Bomsndra_HB (Karnataka)     1958
Name: count, dtype: int64
Most orders to destination:
 destination_name
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
Delhi_Airport_H (Delhi)               3769
Pune_Tathawde_H (Maharashtra)         3695
Chandigarh_Mehmdpur_H (Punjab)        2874
Sonipat_Kundli_H (Haryana)            2796
Bhubaneshwar_Hub (Orissa)             2524
Name: count, dtype: int64
```

```
# Create a new column for corridors :

df['corridor'] = df['source_name'] + " - " + df['destination_name']

# Find the busiest corridor (Top 5)
busiest_corridors = df['corridor'].value_counts().head(5)
print("Busiest corridors:\n", busiest_corridors)

# Busiest corridor
busiest_corridor = busiest_corridors.idxmax()
print("Busiest corridor:", busiest_corridor)
```

```
Busiest corridors:
 corridor
Gurgaon_Bilaspur_HB (Haryana) - Bangalore_Nelmngla_H (Karnataka)     4976
Bangalore_Nelmngla_H (Karnataka) - Gurgaon_Bilaspur_HB (Haryana)     3316
Gurgaon_Bilaspur_HB (Haryana) - Kolkata_Dankuni_HB (West Bengal)     2862
Gurgaon_Bilaspur_HB (Haryana) - Hyderabad_Shamshbd_H (Telangana)     1639
Gurgaon_Bilaspur_HB (Haryana) - Bhiwandi_Mankoli_HB (Maharashtra)    1617
Name: count, dtype: int64
Busiest corridor: Gurgaon_Bilaspur_HB (Haryana) - Bangalore_Nelmngla_H (Karnataka)
```

```
# Filter data for the busiest corridor

busiest_corridor_data = df[df['corridor'] == busiest_corridor]

# Calculate average actual distance

average_distance_actual = busiest_corridor_data['actual_distance_to_destination'].mean()
print(f"Average actual distance for {busiest_corridor}: {average_distance_actual} km")

# Calculate average OSRM predicted distance

average_distance_osrm = busiest_corridor_data['osrm_distance'].mean()
print(f"Average OSRM predicted distance for {busiest_corridor}: {average_distance_osrm} km")
```