Problem Statement:

To help Delhivery process and understand the data coming from their data engineering pipelines and suggesting a structured approach that involves data cleaning, feature creation, data exploration, and visualization.

```
# Importing required Packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import ttest_ind
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

```
# Loading the data

df = pd.read_csv("/kaggle/input/delhivery-data/delhivery_data.csv")
df.head()
```

	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)	IND3886
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND3886
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND3886
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND3886
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND3886

5 rows × 24 columns

Basic Data Cleaning and Exploration

Handling Missing Values

Identify and handle missing values in the dataset.

```
# Check for missing values
length_of_df = len(df)

missing_values = df.isnull().sum()
print(missing_values)
```

```
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
```

```
total_missing_values = max(missing_values)
print(f"Percentage of missing values: {(total_missing_values*100)/length_of_df}")
```

```
Percentage of missing values: 0.20225448169700483
```

```
#Due to low missing value percentage, to the categorical data, solution is to drop the rows
df.dropna(inplace=True, axis=0)
df.isnull().sum()
```

```
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
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
```

Analyzing Dataset Structure

```
# Data structure
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 144316 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144316 non-null  object
 1   trip_creation_time              144316 non-null  object
 2   route_schedule_uuid             144316 non-null  object
 3   route_type                      144316 non-null  object
 4   trip_uuid                       144316 non-null  object
 5   source_center                   144316 non-null  object
 6   source_name                     144316 non-null  object
 7   destination_center              144316 non-null  object
 8   destination_name                144316 non-null  object
 9   od_start_time                   144316 non-null  object
 10  od_end_time                     144316 non-null  object
 11  start_scan_to_end_scan          144316 non-null  float64
 12  is_cutoff                       144316 non-null  bool
 13  cutoff_factor                   144316 non-null  int64
 14  cutoff_timestamp                144316 non-null  object
 15  actual_distance_to_destination  144316 non-null  float64
 16  actual_time                     144316 non-null  float64
 17  osrm_time                       144316 non-null  float64
 18  osrm_distance                   144316 non-null  float64
 19  factor                          144316 non-null  float64
 20  segment_actual_time             144316 non-null  float64
 21  segment_osrm_time               144316 non-null  float64
 22  segment_osrm_distance           144316 non-null  float64
 23  segment_factor                  144316 non-null  float64
dtypes: bool(1), float64(10), int64(1), object(12)
memory usage: 26.6+ MB
```

```
df['cutoff_timestamp'] = pd.to_datetime(df['cutoff_timestamp'], format='%Y-%m-%d %H:%M:%S', errors='coerce')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 144316 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144316 non-null  object
 1   trip_creation_time              144316 non-null  object
 2   route_schedule_uuid             144316 non-null  object
 3   route_type                      144316 non-null  object
 4   trip_uuid                       144316 non-null  object
 5   source_center                   144316 non-null  object
 6   source_name                     144316 non-null  object
 7   destination_center              144316 non-null  object
 8   destination_name                144316 non-null  object
 9   od_start_time                   144316 non-null  object
 10  od_end_time                     144316 non-null  object
 11  start_scan_to_end_scan          144316 non-null  float64
 12  is_cutoff                       144316 non-null  bool
 13  cutoff_factor                   144316 non-null  int64
 14  cutoff_timestamp                140909 non-null  datetime64[ns]
 15  actual_distance_to_destination  144316 non-null  float64
 16  actual_time                     144316 non-null  float64
 17  osrm_time                       144316 non-null  float64
 18  osrm_distance                   144316 non-null  float64
 19  factor                          144316 non-null  float64
 20  segment_actual_time             144316 non-null  float64
 21  segment_osrm_time               144316 non-null  float64
 22  segment_osrm_distance           144316 non-null  float64
 23  segment_factor                  144316 non-null  float64
dtypes: bool(1), datetime64[ns](1), float64(10), int64(1), object(11)
memory usage: 26.6+ MB
```

```
df.head()
```

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

5 rows × 24 columns

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

	start_scan_to_end_scan	cutoff_factor	cutoff_timestamp	actual_distance_to_d
count	144316.000000	144316.000000	140909	144
mean	963.697698	233.561345	2018-09-23 03:15:10.623693568	
min	20.000000	9.000000	2018-09-12 00:10:27	
25%	161.000000	22.000000	2018-09-17 19:18:34	
50%	451.000000	66.000000	2018-09-22 21:15:24	
75%	1645.000000	286.000000	2018-09-28 06:12:35	
max	7898.000000	1927.000000	2018-10-06 23:44:12	
std	1038.082976	345.245823	NaN	

Feature Creation and Aggregation

Feature Extraction

Extract useful features from existing columns:

- Trip Creation Time: Extract year, month, and day.
- Source Name and Destination Name: Split into city and state.

```
# Extract features from trip_creation_time
df['trip_creation_year'] = pd.to_datetime(df['trip_creation_time']).dt.year
df['trip_creation_month'] = pd.to_datetime(df['trip_creation_time']).dt.month
df['trip_creation_day'] = pd.to_datetime(df['trip_creation_time']).dt.day

# Function to handle splitting and ensuring two parts
def split_and_expand(column):
    split_col = column.str.split('_', n=1, expand=True)
    split_col.columns = ['city', 'state']
    split_col['state'] = split_col['state'].fillna('Unknown')  # Fill missing parts with 'Unknown'
    return split_col

# Apply the function to source_name and destination_name
df[['source_city', 'source_state']] = split_and_expand(df['source_name'])
df[['destination_city', 'destination_state']] = split_and_expand(df['destination_name'])
df.head()
```

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

5 rows × 31 columns

```
# Calculate 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['time_taken'] = (df['od_end_time'] - df['od_start_time']).dt.total_seconds()
```

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

# Compare the difference between calculated time and start_scan_to_end_scan
df['time_difference'] = df['time_taken'] - df['start_scan_to_end_scan']
```

Aggregating Data

Aggregate data based on trip_uuid, source_center, and destination_center.

```
# Group by trip_uuid and aggregate
aggregated_df = df.groupby('trip_uuid').agg({
    # Keep sum for continuous
    'actual_time': 'sum',
    'osrm_time': 'sum',
    'actual_distance_to_destination': 'sum',
    'osrm_distance': 'sum',
    'start_scan_to_end_scan': 'sum',
    'time_taken': 'sum',
    # Keep First occurence for categorical
    'route_type': 'first',
    'source_center': 'first',
    'destination_center': 'first',
    'trip_creation_year': 'first',
    'trip_creation_month': 'first',
    'trip_creation_day': 'first',
    'source_state': 'first',
    'destination_state': 'first',
}).reset_index()

aggregated_df.head()
```

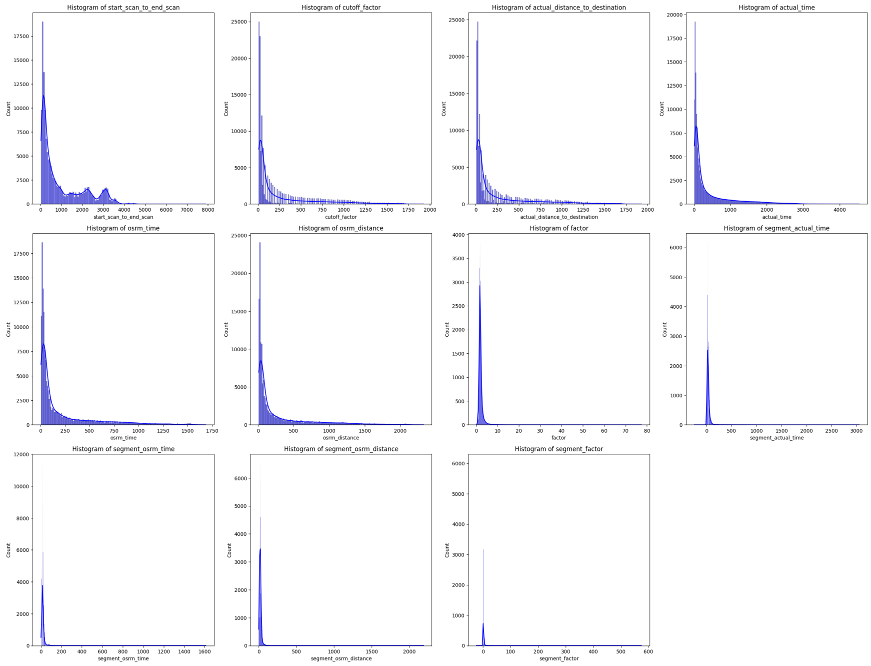
	trip_uuid	actual_time	osrm_time	actual_distance_to_destination	osrm_(
0	trip-153671041653548748	15682.0	7787.0	8860.812105	10
1	trip-153671042288605164	399.0	210.0	240.208306	
2	trip-153671043369099517	112225.0	65768.0	68163.502238	89
3	trip-153671046011330457	82.0	24.0	28.529648	
4	trip-153671052974046625	556.0	207.0	239.007304	

Visual Analysis

Univariate Analysis

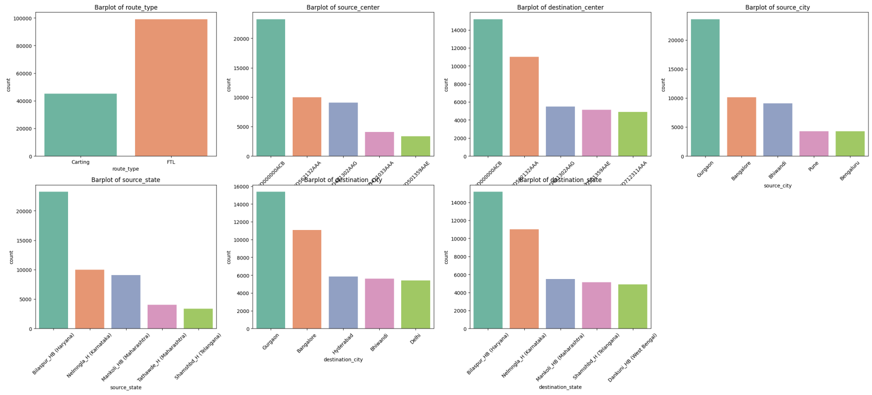
```
def plot_continuous_variable(data, variable):
    # Histogram
    sns.histplot(data[variable].dropna(), color='blue', kde=True)
    plt.title(f'Histogram of {variable}')
    plt.tight_layout()

def plot_categorical_variable(data, category, rotation=0):
    # Boxplot
    if len(data[category].value_counts()) > 5:
        top_5_categories = data[category].value_counts().nlargest(5).index
    # Filter the data to include only the top 5 categories
        filter_data = data[data[category].isin(top_5_categories)]
        sns.countplot(data=filter_data, x=category, palette='Set2', order=filter_data[category].value_counts().index)
        plt.xticks(rotation=rotation)
    else:
        sns.countplot(data=data, x=category, palette='Set2')
    plt.title(f'Barplot of {category}')
    plt.xlabel(category)


numeric_columns = ['start_scan_to_end_scan', 'cutoff_factor','actual_distance_to_destination', 'actual_time', 'osrm_time','osrm_distance
numeric_data = df[numeric_columns]
i = 1
plt.figure(figsize=(24, 24))
for cat in numeric_columns:
    plt.subplot(4, 4, i)
    plot_continuous_variable(df, cat)
    i += 1
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
```



While most numerical data has huge range, this is highly caused due to the influence of outliers as mean and median of the data is far smaller than max_values.

```
categorical_columns = ['route_type', 'source_center', 'destination_center', 'source_city', 'source_state', 'destination_city','destinat:

i = 1
plt.figure(figsize=(30, 24))
for cat in categorical_columns:
    plt.subplot(4, 4, i)
    plot_categorical_variable(df, cat, rotation=45)
    i += 1
plt.show()
```



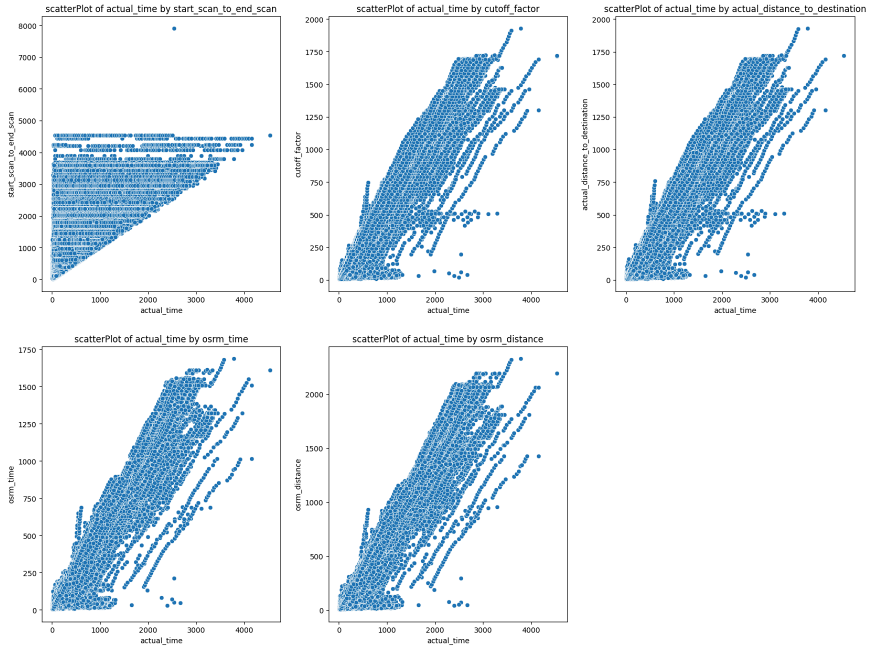
Majority of the categorical values have uneven distribution among its groups.

Bi-variate Analysis

```
# Define a function to create the required bivariate plots for continuous-categorical variables
def plot_bivariate_plot_NC(data, category, variable):
    sns.violinplot(x=df[category].dropna(), y=df[variable].dropna())
    plt.title(f'Boxplot(Boxenplot for better understanding) of {variable} by {category}')
    plt.xlabel(category)
    plt.ylabel(variable)
    plt.tight_layout()
    plt.show()

def plot_bivariate_plot_CC(data, category_1, category_2, rotation=0):
    # Grouped Baxplot
    if len(data[category_1].value_counts()) > 5:
        top_5_categories = data[category_1].value_counts().nlargest(5).index
    # Filter the data to include only the top 5 categories
        filter_data = data[data[category_1].isin(top_5_categories)]
    if len(data[category_2].value_counts()) > 5:
        top_5_categories = data[category_2].value_counts().nlargest(5).index
    # Filter the data to include only the top 5 categories
        filter_data = data[data[category_2].isin(top_5_categories)]
    sns.countplot(data=filter_data, x=category_1, hue=category_2, palette='Set2', order=data[category_1].value_counts().nlargest(5).inde
    plt.xticks(rotation=rotation)
    plt.title(f'Grouped Barplot of {category_1} and {category_2}')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

# Define a function to create the required bivariate plots for continuous-continuous variables
def plot_bivariate_plot_NN(data, variable_1, variable_2):
    # Scatterplot
    sns.scatterplot(x=variable_1, y=variable_2, data=data)
    plt.title(f'scatterPlot of {variable_1} by {variable_2}')
    plt.xlabel(variable_1)
    plt.ylabel(variable_2)
```

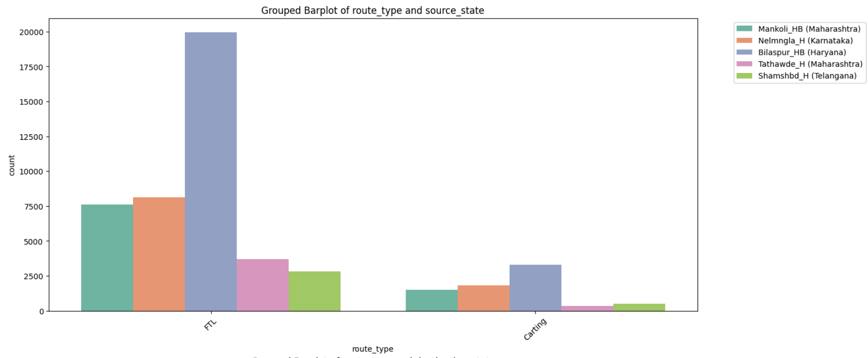
```
select_num_cols = [ 'actual_time','start_scan_to_end_scan', 'cutoff_factor','actual_distance_to_destination', 'osrm_time','osrm_distance
select_cat_cols = ['route_type','source_state','destination_state']
```

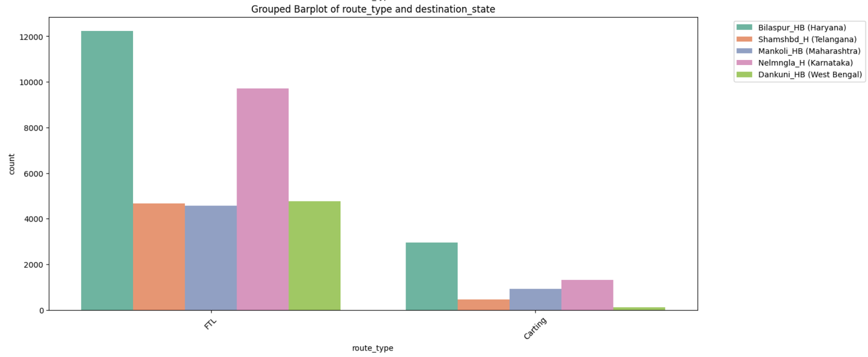
```
i = 1
plt.figure(figsize=(20, 15))
for col in select_num_cols[1:]:
    plt.subplot(2, 3, i)
    plot_bivariate_plot_NN(df, select_num_cols[0], col)
    i += 1
plt.show()
```



Double-click (or enter) to edit

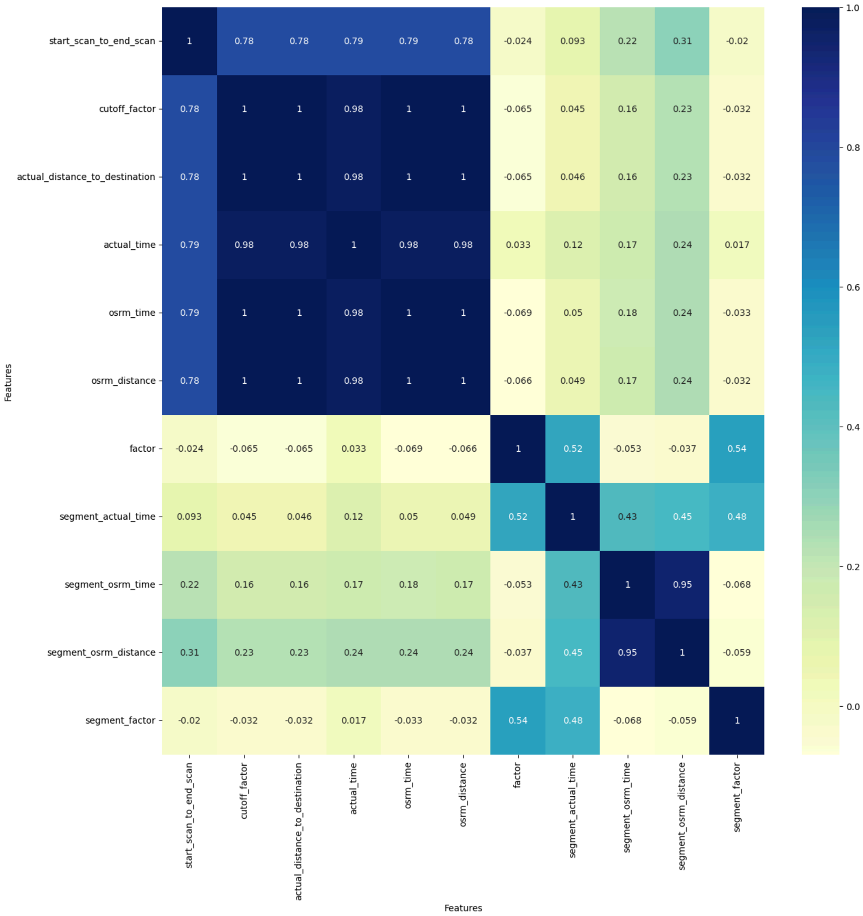
```
i = 1
plt.figure(figsize=(15, 15))
for col in select_cat_cols[1:]:
    plt.subplot(2, 1, i)
    plot_bivariate_plot_CC(df, select_cat_cols[0], col, rotation=45)
    i += 1
plt.show()
```





Correlation

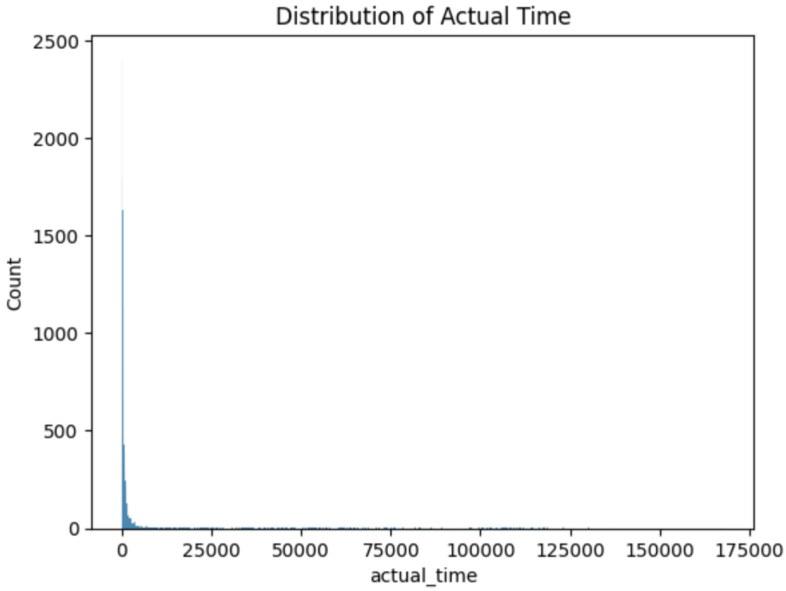
```
correlation_matrix = numeric_data.corr()

plt.figure(figsize=(15, 15))
sns.heatmap(correlation_matrix, annot=True, cmap="YlGnBu")
plt.xlabel('Features')
plt.ylabel('Features')
plt.show()
```



Features 'start_scan_to_end_scan', 'cutoff_factor','actual_distance_to_destination', 'osrm_time','osrm_distance' all have high positive correlation among them

```
# Distribution of actual time
sns.histplot(aggregated_df['actual_time'])
plt.title('Distribution of Actual Time')
plt.show()
```

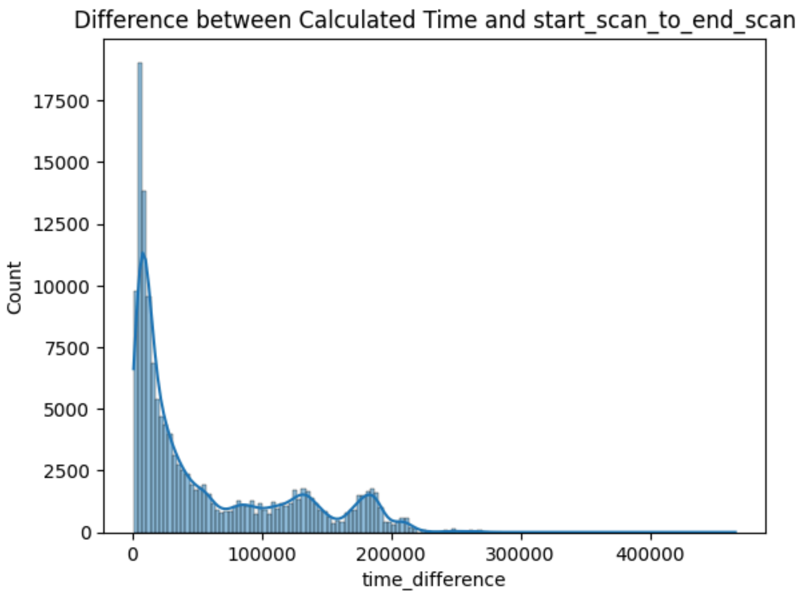
```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
```



Hypothesis Testing

```
# Perform hypothesis testing/visual analysis between calculated time and start_scan_to_end_scan
# H0 (Null Hypothesis): There is no significant difference between the means of time_taken and start_scan_to_end_scan.
# Ha (Alternative Hypothesis): There is a significant difference between the means of time_taken and start_scan_to_end_scan.

sns.histplot(df['time_difference'], kde=True)
plt.title('Difference between Calculated Time and start_scan_to_end_scan')
plt.show()

ttest_result_1 = ttest_ind(df['time_taken'], df['start_scan_to_end_scan'])
print(f'T-test result between time_taken and start_scan_to_end_scan: {ttest_result_1}')
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_
  with pd.option_context('mode.use_inf_as_na', True):
```

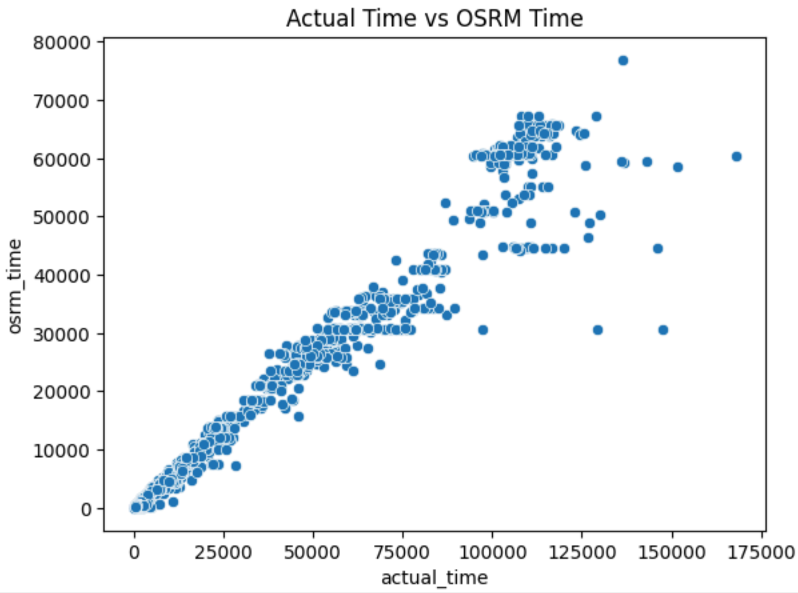



```
T-test result between time_taken and start_scan_to_end_scan: TtestResult(statistic=34
```

Since the p-value (0.0) is less than 0.05, we reject the null hypothesis (H0). This suggests that there is a significant difference between the means of time_taken and start_scan_to_end_scan.

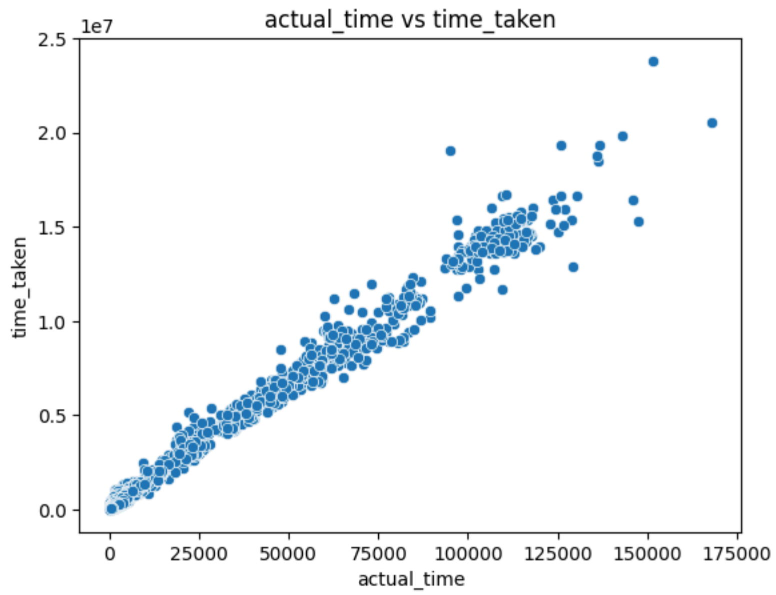
```
# H0 (Null Hypothesis): There is no significant difference between the means of actual_time and osrm_time.
# Ha (Alternative Hypothesis): There is a significant difference between the means of actual_time and osrm_time.

ttest_result_2 = ttest_ind(aggregated_df['actual_time'], aggregated_df['osrm_time'])
print(f'T-test result between actual_time and osrm_time: {ttest_result_2}')
# Relationship between actual_time and osrm_time
sns.scatterplot(x=aggregated_df['actual_time'], y=aggregated_df['osrm_time'])
plt.title('Actual Time vs OSRM Time')
plt.show()
```

```
T-test result between actual_time and osrm_time: TtestResult(statistic=14.05476572196
```

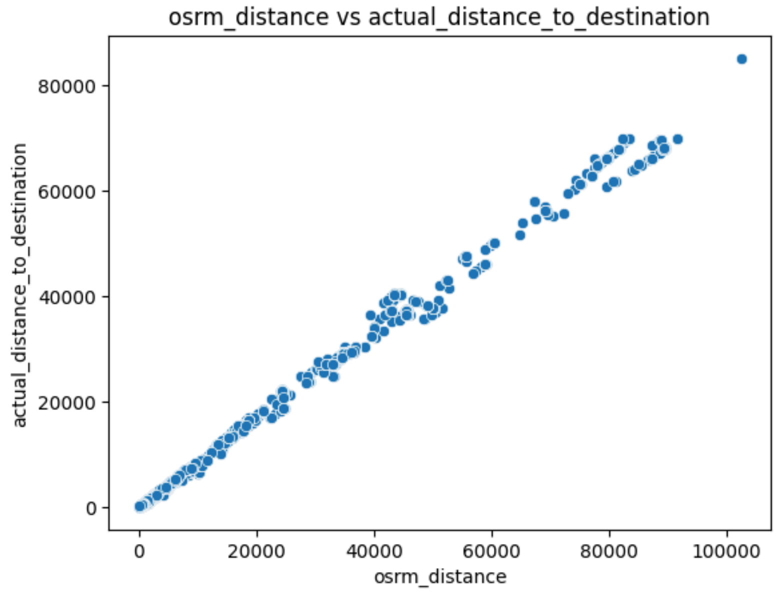


Since the p-value (1.003515078231707e-44) is less than 0.05, we reject the null hypothesis (H0). This indicates that there is a significant difference between the means of actual_time and osrm_time.

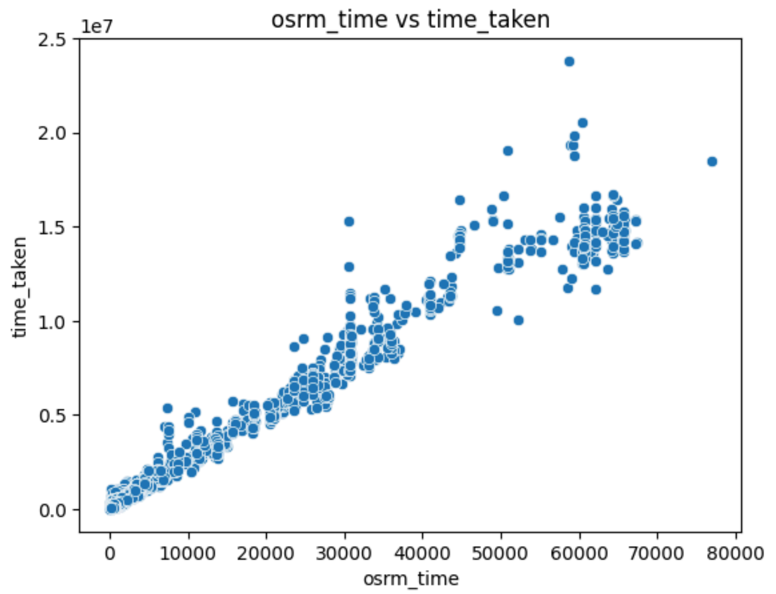
```
# Hypothesis testing/visual analysis for other comparisons
def hypothesis_test_and_plot(dataframe, col1, col2):
    sns.scatterplot(x=dataframe[col1], y=dataframe[col2])
    plt.title(f'{col1} vs {col2}')
    plt.show()
    ttest_result = ttest_ind(dataframe[col1], dataframe[col2])
    print(f'T-test result between {col1} and {col2}: {ttest_result}')

# Apply the function to required comparisons

# H0 (Null Hypothesis): There is no significant difference between the means of actual_time and time_taken.
# Ha (Alternative Hypothesis): There is a significant difference between the means of actual_time and time_taken.
hypothesis_test_and_plot(aggregated_df, 'actual_time', 'time_taken')

# H0 (Null Hypothesis): There is no significant difference between the means of osrm_distance and actual_distance_to_destination.
# Ha (Alternative Hypothesis): There is a significant difference between the means of osrm_distance and actual_distance_to_destination.
hypothesis_test_and_plot(aggregated_df, 'osrm_distance', 'actual_distance_to_destination')

# H0 (Null Hypothesis): There is no significant difference between the means of osrm_time and time_taken.
# Ha (Alternative Hypothesis): There is a significant difference between the means of osrm_time and time_taken.
hypothesis_test_and_plot(aggregated_df, 'osrm_time', 'time_taken')
```




```
T-test result between actual_time and time_taken: TtestResult(statistic=-33.6681927
```




```
T-test result between osrm_distance and actual_distance_to_destination: TtestResult
```



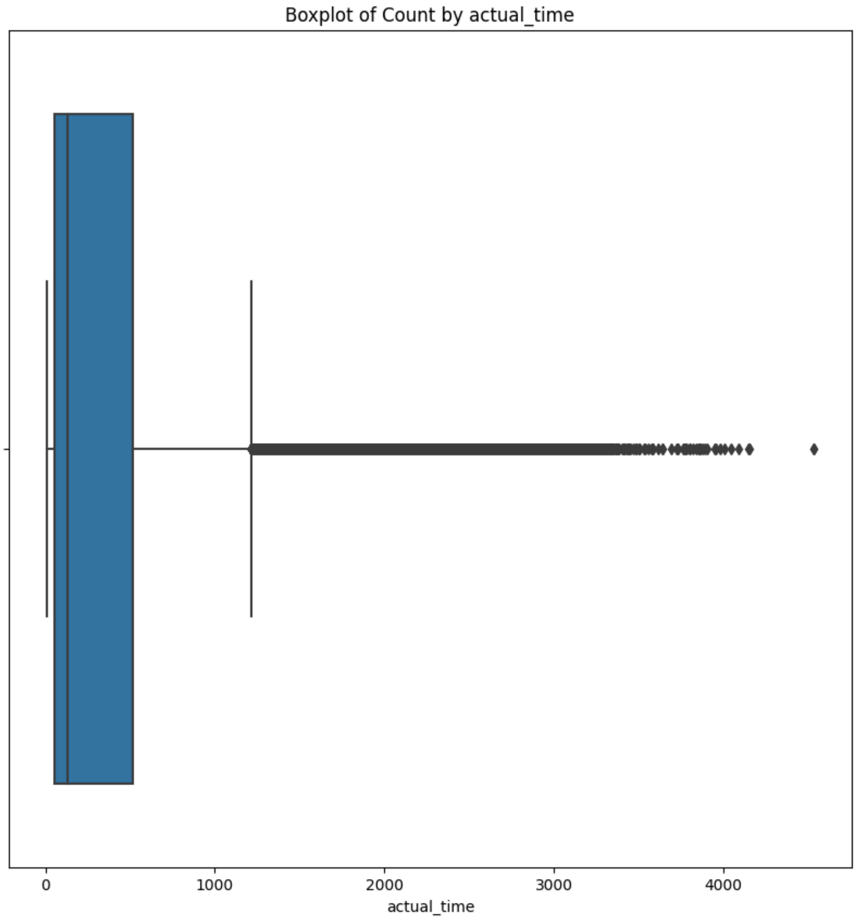

```
T-test result between osrm_time and time_taken: TtestResult(statistic=-33.788216115
```

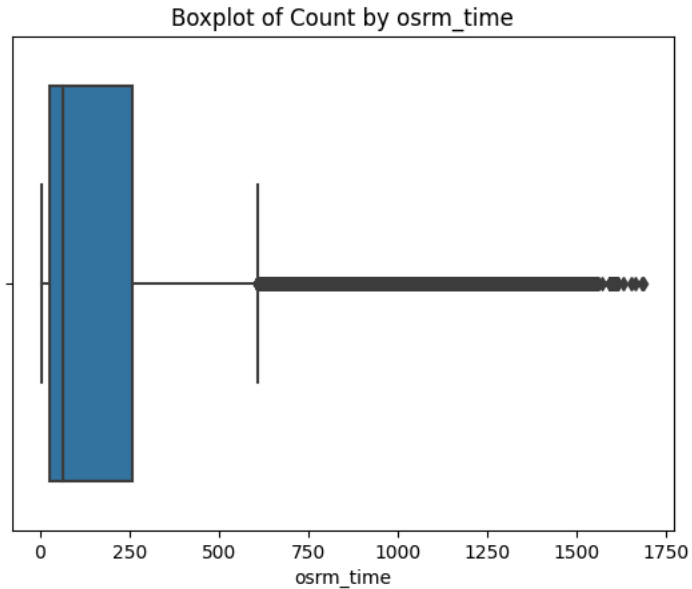
- Since the p-value (6.859303598499063e-244) is less than 0.05, we reject the null hypothesis (H0). This suggests that there is a significant difference between the means of actual_time and time_taken.

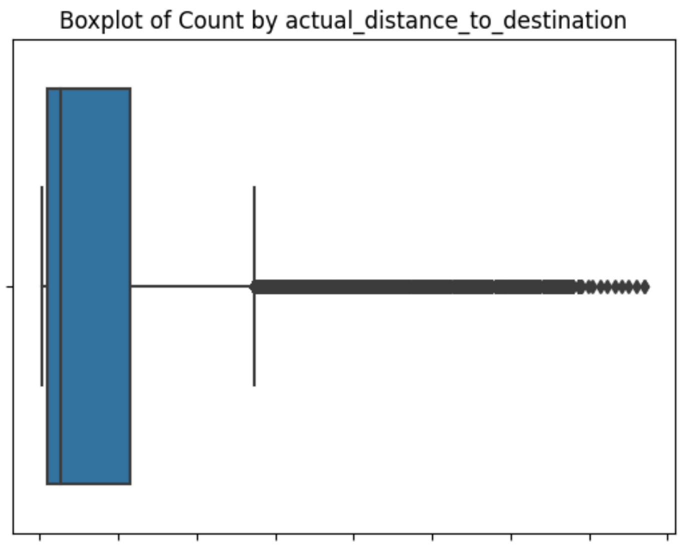
- Since the p-value (1.4486329729906885e-05) is less than 0.05, we reject the null hypothesis (H0). This indicates that there is a significant difference between the means of osrm_distance and actual_distance_to_destination.
- Since the p-value (1.3861868853733896e-245) is less than 0.05, we reject the null hypothesis (H0). This suggests that there is a significant difference between the means of osrm_time and time_taken.

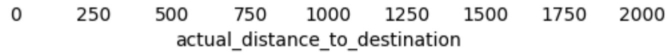
Finding Outliers and Treatment

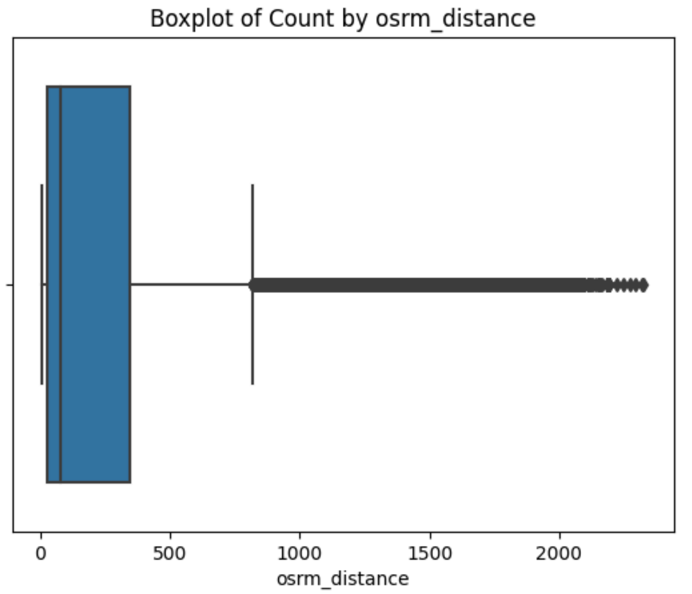
```
# Detect outliers
aggregated_df_num = aggregated_df.select_dtypes(include=np.number)
aggregated_df_num = aggregated_df_num[aggregated_df_num.columns[:-3]]
plt.figure(figsize=(10, 10))
for var in aggregated_df_num.columns:
    sns.boxplot(x=var, data=df)
    plt.title(f'Boxplot of Count by {var}')
    plt.show()
```

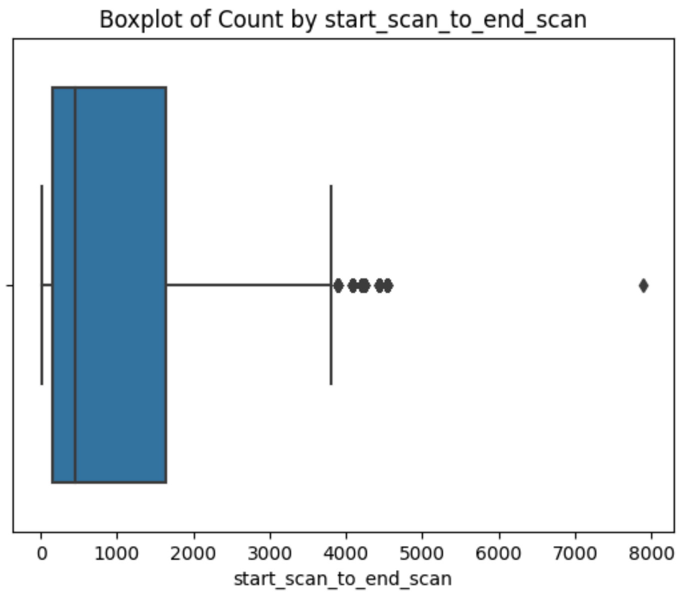

Boxplot of Count by actual_time
0
1000
2000
3000
4000
actual_time

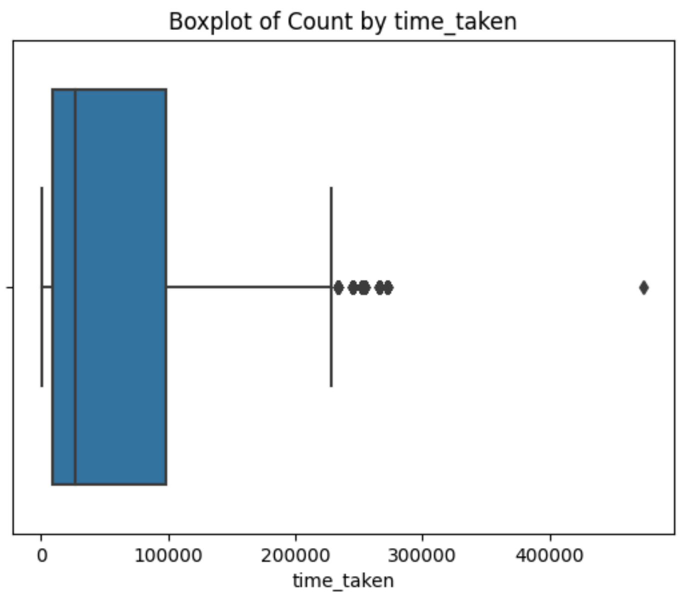

Boxplot of Count by osrm_time
0
250
500
750
1000
1250
1500
1750
osrm_time


Boxplot of Count by actual_distance_to_destination


0
250
500
750
1000
1250
1500
1750
2000
actual_distance_to_destination


Boxplot of Count by osrm_distance
0
500
1000
1500
2000
osrm_distance


Boxplot of Count by start_scan_to_end_scan
0
1000
2000
3000
4000
5000
6000
7000
8000
start_scan_to_end_scan


Boxplot of Count by time_taken
0
100000
200000
300000
400000
time_taken

```
# Find outliers using the IQR method
def find_and_handle_outliers(dataframe, columns):
    for col in columns:
        Q1 = dataframe[col].quantile(0.25)
        Q3 = dataframe[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        dataframe = dataframe[(dataframe[col] >= lower_bound) & (dataframe[col] <= upper_bound)]
    return dataframe

numerical_columns = ['actual_time', 'osrm_time', 'actual_distance_to_destination', 'osrm_distance', 'time_taken']
filtered_df = find_and_handle_outliers(aggregated_df, numerical_columns)
```

```
filtered_df.head()
```

	trip_uuid	actual_time	osrm_time	actual_distance_to_destination	osrm_(
1	trip-153671042288605164	399.0	210.0	240.208306	
3	trip-153671046011330457	82.0	24.0	28.529648	
4	trip-153671052974046625	556.0	207.0	239.007304	
5	trip-153671055416136166	92.0	30.0	34.407865	
6	trip-153671066201138152	24.0	13.0	9.100510	

Encoding Categorical Variables

```
filtered_df.select_dtypes(include='object').nunique()
```

```
trip_uuid             10148
route_type                2
source_center           739
destination_center      898
source_state            655
destination_state       801
dtype: int64
```

```
# One-hot encoding
encoded_df = pd.get_dummies(filtered_df, columns=['route_type'])
```

Normalization/Standardization

```
from sklearn.preprocessing import StandardScaler

# Standardize numerical features
scaler = StandardScaler()
numerical_columns = ['actual_time', 'osrm_time', 'actual_distance_to_destination', 'osrm_distance']
encoded_df[numerical_columns] = scaler.fit_transform(encoded_df[numerical_columns])

# Display the processed DataFrame
encoded_df.head()
```

	trip_uuid	actual_time	osrm_time	actual_distance_to_destination	osrm_(
1	trip-153671042288605164	0.557651	0.798465	1.338076	
3	trip-153671046011330457	-0.824265	-0.958499	-0.826426	-
4	trip-153671052974046625	1.242071	0.770127	1.325795	
5	trip-153671055416136166	-0.780672	-0.901823	-0.766319	-
6	trip-153671066201138152	-1.077108	-1.062405	-1.025097	-

Business Insights and Recommendations

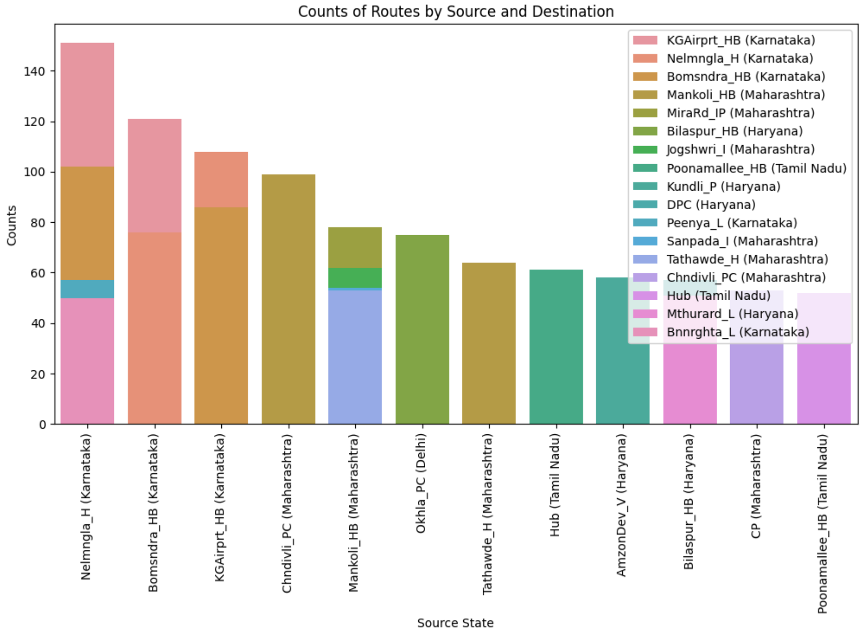
```
busiest_corridors = encoded_df.groupby(['source_state', 'destination_state']).size().reset_index(name='counts').sort_values(by='counts',
busiest_corridors.head(10)
```

	source_state	destination_state	counts
825	Nelmngla_H (Karnataka)	KGAirprt_HB (Karnataka)	151
140	Bomsndra_HB (Karnataka)	KGAirprt_HB (Karnataka)	121
545	KGAirprt_HB (Karnataka)	Nelmngla_H (Karnataka)	108
704	Mankoli_HB (Maharashtra)	Unknown	105
821	Nelmngla_H (Karnataka)	Bomsndra_HB (Karnataka)	102
290	Chndivli_PC (Maharashtra)	Mankoli_HB (Maharashtra)	99
1077	Tathawde_H (Maharashtra)	Unknown	92
540	KGAirprt_HB (Karnataka)	Bomsndra_HB (Karnataka)	86
695	Mankoli_HB (Maharashtra)	MiraRd_IP (Maharashtra)	78
143	Bomsndra_HB (Karnataka)	Nelmngla_H (Karnataka)	76

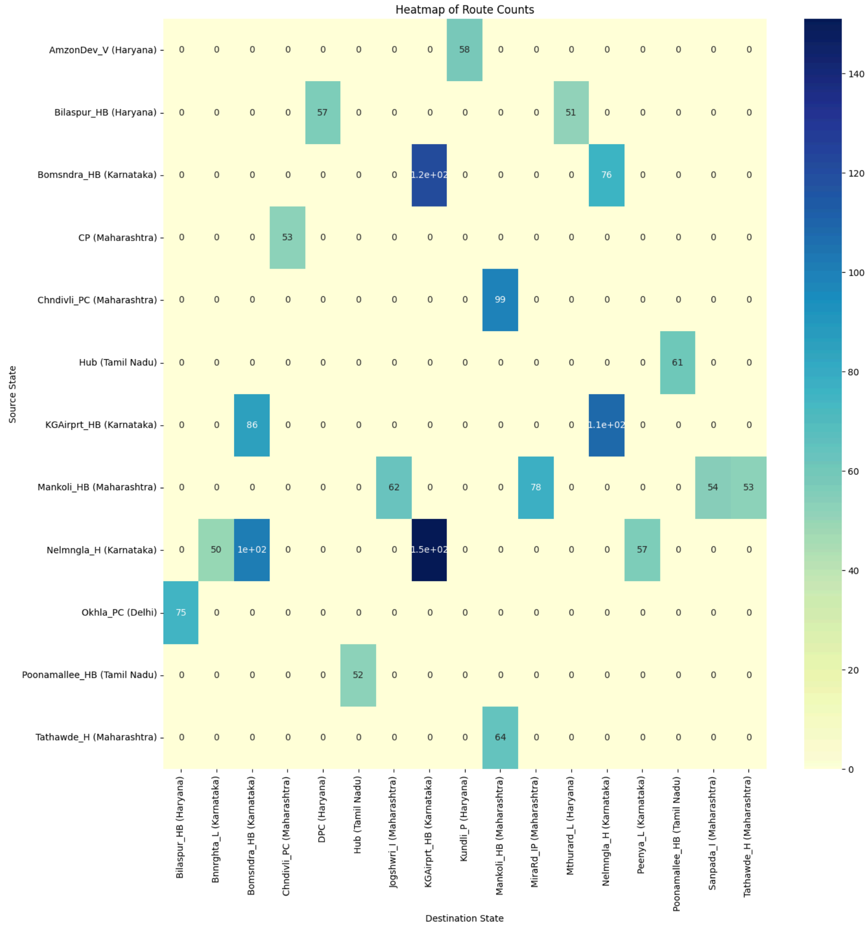
```
data = busiest_corridors.head(30)
data = data[~((data['source_state']=='Unknown') | (data['destination_state']=='Unknown'))]
data.head(10)
```

	source_state	destination_state	counts
825	Nelmngla_H (Karnataka)	KGAirprt_HB (Karnataka)	151
140	Bomsndra_HB (Karnataka)	KGAirprt_HB (Karnataka)	121
545	KGAirprt_HB (Karnataka)	Nelmngla_H (Karnataka)	108
821	Nelmngla_H (Karnataka)	Bomsndra_HB (Karnataka)	102
290	Chndivli_PC (Maharashtra)	Mankoli_HB (Maharashtra)	99
540	KGAirprt_HB (Karnataka)	Bomsndra_HB (Karnataka)	86
695	Mankoli_HB (Maharashtra)	MiraRd_IP (Maharashtra)	78
143	Bomsndra_HB (Karnataka)	Nelmngla_H (Karnataka)	76
850	Okhla_PC (Delhi)	Bilaspur_HB (Haryana)	75
1073	Tathawde_H (Maharashtra)	Mankoli_HB (Maharashtra)	64

```
# Bar plot

plt.figure(figsize=(12, 6))
sns.barplot(data=data, x='source_state', y='counts', hue='destination_state', dodge=False)
plt.xticks(rotation=90)
plt.title('Counts of Routes by Source and Destination')
plt.ylabel('Counts')
plt.xlabel('Source State')
plt.legend(loc='upper right')
plt.show()
```




```
# Pivot table for heatmap
pivot_df = data.pivot_table(index='source_state', columns='destination_state', values='counts', fill_value=0)

# Heatmap
plt.figure(figsize=(15, 15))
sns.heatmap(pivot_df, annot=True, cmap="YlGnBu")
plt.title('Heatmap of Route Counts')
plt.xlabel('Destination State')
plt.ylabel('Source State')
plt.show()
```


Heatmap of Route Counts
Source State
Destination State
AmzonDev_V (Haryana)
Bilaspur_HB (Haryana)
Bomsndra_HB (Karnataka)
CP (Maharashtra)
Chndivli_PC (Maharashtra)
Hub (Tamil Nadu)
KGAirprt_HB (Karnataka)
Mankoli_HB (Maharashtra)
Nelmngla_H (Karnataka)
Okhla_PC (Delhi)
Poonamallee_HB (Tamil Nadu)
Tathawde_H (Maharashtra)
Bilaspur_HB (Haryana)
Bnnrghta_L (Karnataka)
Bomsndra_HB (Karnataka)
Chndivli_PC (Maharashtra)
DPC (Haryana)
Hub (Tamil Nadu)
Jogshwri_I (Maharashtra)
KGAirprt_HB (Karnataka)
Kundli_P (Haryana)
Mankoli_HB (Maharashtra)
MiraRd_IP (Maharashtra)
Mthurard_L (Haryana)
Nelmngla_H (Karnataka)
Peenya_L (Karnataka)
Poonamallee_HB (Tamil Nadu)
Sanpada_I (Maharashtra)
Tathawde_H (Maharashtra)