PHASE 4:DEVELOPMENT – PART2

Project Title: Public Transport Efficiency Analysis

Introduction:

In Phase 4, We Start To Build the Project Public Transportation Efficiency Analysis By Loading and Preprocessing the Provided dataset from Kaggle (https://www.kaggle.com/datasets/rednivrug/unisys?select=20140711.CSV).

We can define the analysis objectives and collect transportation data from the source shared.

Process and clean the collected data to ensure its quality and accuracy.

Steps Followed:

Step1: Import the Libraries

We need to Import the required Libraries to load and Preprocess the dataset . we have used libraries like Pandas , Matplotlib , Seaborn and Numpy.

```
In [1]: import numpy as np
        import pandas as pd
```

Step2: Load the Dataset

To use the data set for our analysis we need to import the dataset. We can import it using pandas **read_csv()**build in function.

```
In [2]: print("Load the dataset")
        import pandas as pd
        data = pd.read_csv('20140711.csv', low_memory=False)
        data.shape
        data.head(5)
```

```
Load the dataset
```

Out[2]:

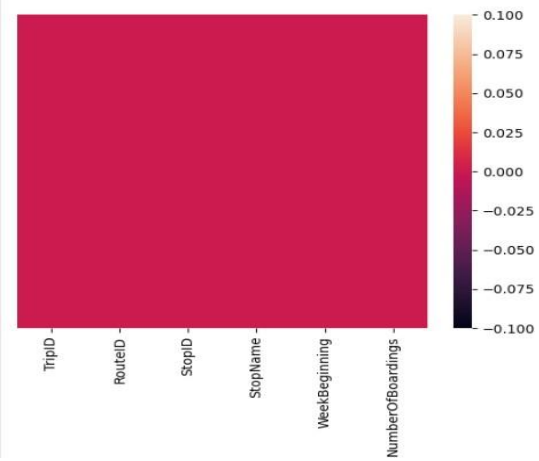
	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings
0	23631	100	14156	181 Cross Rd	2013-06-30 00:00:00	1
1	23631	100	14144	177 Cross Rd	2013-06-30 00:00:00	1
2	23632	100	14132	175 Cross Rd	2013-06-30 00:00:00	1
3	23633	100	12266	Zone A Arndale Interchange	2013-06-30 00:00:00	2
4	23633	100	14147	178 Cross Rd	2013-06-30 00:00:00	1

Step3: Check Data Types and Drop Duplicates

In This Step we drop the duplicate columns or data and then plot an heatmap using seaborn to virtualize the null values of the dataset .after we will check for the data type using dataframe_name.**dtypes** built-in function

```
In [3]: data = data.drop_duplicates()
        import seaborn as sns
        sns.heatmap(data.isnull(),yticklabels= False)
        print("\nCheck data types of columns")
        print(data.dtypes)
```

```
Check data types of columns
TripID                int64
RouteID              object
StopID                int64
StopName             object
WeekBeginning        object
NumberOfBoardings     int64
dtype: object
```



Step4: Handle Mixed Data Types

Here We will Handle the data types which are irregular and not manageable into an manageable datatype for example convert a **string**datatype but the data is in number we can convert it into an **int**datatype

```
In [4]: data['RouteID'] = pd.to_numeric(data['RouteID'], errors='coerce')
        print("Handle mixed data types")
        print(data.dtypes)
```

```
Handle mixed data types
TripID                int64
RouteID             float64
StopID                int64
StopName             object
WeekBeginning        object
NumberOfBoardings     int64
dtype: object
```

Step5: Handle Missing Values

In this step we will delete all the rows that having an null value by using the built-in function called **dropna()** function and then show the remaining data using **shape** function

```
In [4]: data = data.dropna()
        print("\nHandle missing values")
        print(data.shape)
```

```
Handle missing values
(10857234, 6)
```

Step6: Change Data Types

In this step we change the date field data which is in **string** data type to the **datetime** format which is available in the pandasdataframe

```
In [5]: data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning'], errors='coerce')
        print("\nConvert 'WeekBeginning' column to datetime format")
        print(data['WeekBeginning'].head())
```

```
Convert 'WeekBeginning' column to datetime format
0   2013-06-30
1   2013-06-30
2   2013-06-30
3   2013-06-30
4   2013-06-30
Name: WeekBeginning, dtype: datetime64[ns]
```

Step7: Convert Into Convenient Data

Here we clean the data of every column which contains the **whitespace** in beginning and the end of the data given in the dataset using **strip()** function

```
In [6]: data['StopName'] = data['StopName'].str.strip()
        print("\nClean 'StopName' column")
        print(data['StopName'].head())
```

```
Clean 'StopName' column
0                  181 Cross Rd
1                  177 Cross Rd
2                  175 Cross Rd
3    Zone A Arndale Interchange
4                  178 Cross Rd
Name: StopName, dtype: object
```

Step8 : Check Number Of Unique Values

```
In [7]: print(data.nunique())
```

```
TripID               39282
RouteID                605
StopID                7397
StopName              4165
WeekBeginning           54
NumberOfBoardings      400
dtype: int64
```

Step 9: Check the Cleaned Dataset

```
In [8]: data.shape
        data.columns
        data.head(3)
```

Out[8]:

	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings
0	23631	100	14156	181 Cross Rd	2013-06-30	1
1	23631	100	14144	177 Cross Rd	2013-06-30	1
2	23632	100	14132	175 Cross Rd	2013-06-30	1

Step 10: Check for NULL Values

```
In [9]: data.isnull().sum()
```

```
Out[9]: TripID               0
        RouteID              0
        StopID               0
        StopName             0
        WeekBeginning        0
        NumberOfBoardings    0
        dtype: int64
```

Step 11: Check the Uniqueness

```
In [10]: data['WeekBeginning'].unique()
```

```
Out[10]: <DatetimeArray>
['2013-06-30 00:00:00', '2013-07-07 00:00:00', '2013-07-14 00:00:00',
 '2013-07-21 00:00:00', '2013-07-28 00:00:00', '2013-08-04 00:00:00',
 '2013-08-11 00:00:00', '2013-08-18 00:00:00', '2013-08-25 00:00:00',
 '2013-09-01 00:00:00', '2013-09-08 00:00:00', '2013-09-15 00:00:00',
 '2013-09-22 00:00:00', '2013-09-29 00:00:00', '2013-10-06 00:00:00',
 '2013-10-13 00:00:00', '2013-10-20 00:00:00', '2013-10-27 00:00:00',
 '2013-11-03 00:00:00', '2013-11-10 00:00:00', '2013-11-17 00:00:00',
 '2013-11-24 00:00:00', '2013-12-01 00:00:00', '2013-12-08 00:00:00',
 '2013-12-15 00:00:00', '2013-12-22 00:00:00', '2013-12-29 00:00:00',
 '2014-01-05 00:00:00', '2014-01-12 00:00:00', '2014-01-19 00:00:00',
 '2014-01-26 00:00:00', '2014-02-02 00:00:00', '2014-02-09 00:00:00',
 '2014-02-16 00:00:00', '2014-02-23 00:00:00', '2014-03-02 00:00:00',
 '2014-03-09 00:00:00', '2014-03-16 00:00:00', '2014-03-23 00:00:00',
 '2014-03-30 00:00:00', '2014-04-06 00:00:00', '2014-04-13 00:00:00',
 '2014-04-20 00:00:00', '2014-04-27 00:00:00', '2014-05-04 00:00:00',
 '2014-05-11 00:00:00', '2014-05-18 00:00:00', '2014-05-25 00:00:00',
 '2014-06-01 00:00:00', '2014-06-08 00:00:00', '2014-06-15 00:00:00',
 '2014-06-22 00:00:00', '2014-06-29 00:00:00', '2014-07-06 00:00:00']
Length: 54, dtype: datetime64[ns]
```

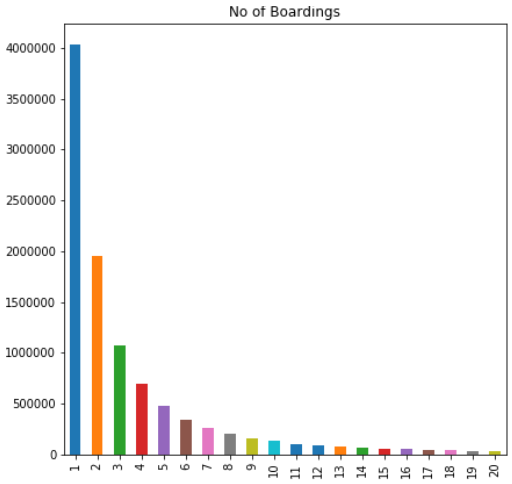
Step12: Save the Cleaned Dataset

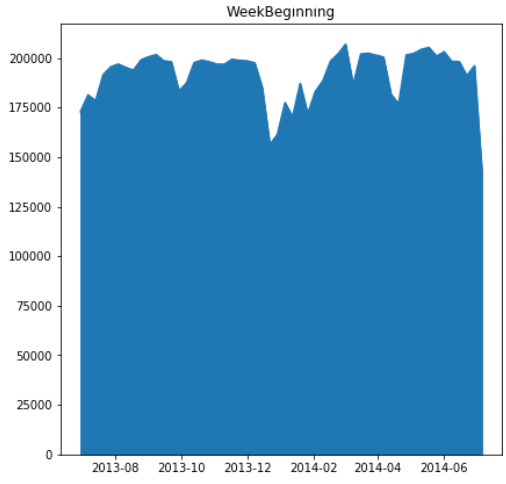
```
In [12]: data.to_csv('cleaned_data.csv', index=False)
         print("\nSave the cleaned dataset to a new CSV file")
         print("Cleaned dataset saved successfully.")
```

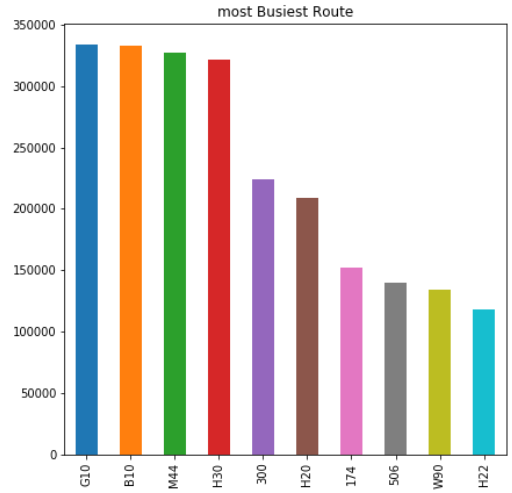
```
Save the cleaned dataset to a new CSV file
Cleaned dataset saved successfully.
```

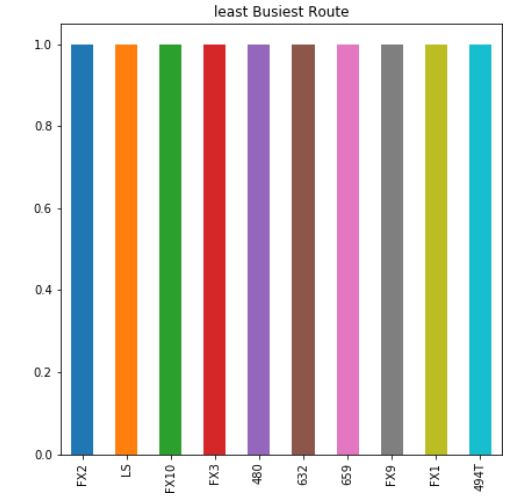
Step13: Data Virtulization

```
fig,axrr=plt.subplots(2,2,figsize=(15,15))

ax=axrr[0][0]
ax.set_title("No of Boardings")
data['NumberOfBoardings'].value_counts().sort_index().head(20).plot.bar(ax=axrr[0][0])

ax=axrr[0][1]
ax.set_title("WeekBeginning")
data['WeekBeginning'].value_counts().plot.area(ax=axrr[0][1])

ax=axrr[1][0]
ax.set_title("most Busiest Route")
data['RouteID'].value_counts().head(10).plot.bar(ax=axrr[1][0])

ax=axrr[1][1]
ax.set_title("least Busiest Route")
data['RouteID'].value_counts().tail(10).plot.bar(ax=axrr[1][1])
```









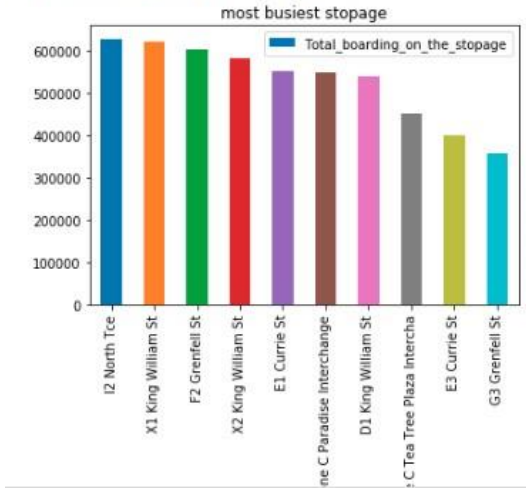

```
stopageName_with_boarding = stopageName_with_boarding.sort_values('Total_boarding_on_the_stopage', ascending = False)
#stopage with most no of boarding
stopageName_with_boarding.head(10)
```

	StopName	Total_boarding_on_the_stopage
3054	I2 North Tce	628859
3125	X1 King William St	622099
3032	F2 Grenfell St	604149
3130	X2 King William St	583227
3021	E1 Currie St	550396
3207	Zone C Paradise Interchange	547709
3015	D1 King William St	541046
3211	Zone C Tea Tree Plaza Intercha	451960

Step14: Virtualization for Most Busiest Stop

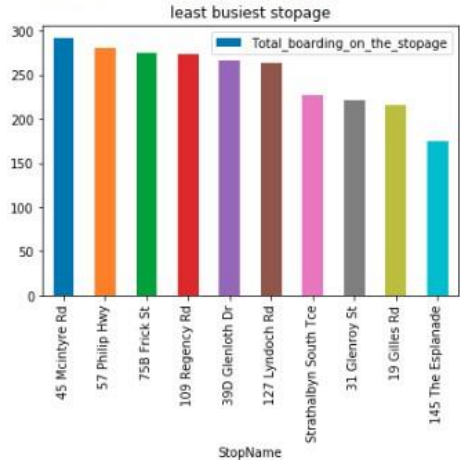
```
ax = stopageName_with_boarding.head(10).plot.bar(x='StopName', y='Total_boarding_on_the_stopage', rot=90)
ax.set_title("most busiest stopage")
```

```
Text(0.5,1,'most busiest stopage')
```




```
ax = stopageName_with_boarding.tail(10).plot.bar(x='StopName', y='Total_boarding_on_the_stopage', rot=90)
ax.set_title("least busiest stopage")
```

```
Text(0.5,1,'least busiest stopage')
```



```
data['WeekBeginning'].value_counts().mean()
```

```
191508.66666666666
```

```
# data['dist_from_centre'].nunique()
bb_grp = data.groupby(['dist_from_centre']).agg({'NumberOfBoardings': ['sum']}).reset_index()
bb_grp.columns = bb_grp.columns.get_level_values(0)
bb_grp.head()
bb_grp.columns
bb_grp.tail()
```

	dist_from_centre	NumberOfBoardings
0	0.000018	1892435
1	0.131368	167535
2	0.309089	356518
3	0.314937	1484824
4	0.326005	120061

```
Index(['dist_from_centre', 'NumberOfBoardings'], dtype='object')
```

	dist_from_centre	NumberOfBoardings
2392	86.471064	18905
2393	94.826409	321
2394	99.625655	1101
2395	99.665190	4373
2396	99.748995	21216