Pyber\_Challenge Report

**Pyber Challenge Over view:**

Overview of this report is to create a summary DataFrame to provide the Total Rides, Total Drivers, Total Fares, Average Fare per Ride, and Average Fare per Driver for each city. Second, a multiple-line plot to show the sum of the fares for each city type by using the Pandas and Matplotlib libraries to plot the sum of the fares for each city type.

**Pyber Challenge Summary:**

Summary DataFrame: We have received two set of data; First, data for three cities Urban, Suburban, and Rural, and Second, the data for rides in each city. Inspect the received data; for example, how many columns and rows are there, What types of data are present, is the data clean. Once we studied about the data next set is to find how many riders are in each individual city, total drivers, total fares in each city, average fare per ride, and average fare per driver.

Jupyter Notebook was used to generate a code to make things easy and readable format. Started with loading and converting the data into Pandas DataFrames. After completion of converting the data into DataFrames merge them together, see below the summery in detail.

Code to convert and Merge the data together:

Import matplotlb.pyplot as plt

Import pandas as pd

city\_data\_to\_load = “./Resources/city\_data.csv”

ride\_data\_to\_load = “./Resources/ride\_data.csv”

pyber\_data\_df = pd.merge(ride\_data\_df, city\_data\_df, how="left", on=["city "])

pyber\_data\_df

***Start doing the analysis once both city\_data and ride\_data DataFrames are merged.***

1. **Total Rides:**

To get the total number of rides in each city, we are using the groupby (), “Type”, and count () function. Where groupby(“type”) is used for splitting the data into different data types, “and count[“ride\_id”] is used to count the rides were taken in each city.

Groupby() will separate all the types in ride\_id and Count [] function will count the number of rider separately in each city. There are three cities Urban, Suburban, and Rural and with help of groupby() and count[] ride counts were split into three cities.

Code:

total\_rides\_by\_type = pyber\_data\_df.groupby(["type"]).count()["ride\_id"]

Result:

type

Rural : 125

Suburban : 625

Urban : 1625

Name: ride\_id, dtype: int64

1. **Total Drivers:**

To get the total number of drivers in each city, we are using the groupby (), “Type”, and sum () function. Where groupby(“type”) is used for splitting the three cities, and sum() function will provide the total number of drivers in each city.

Groupby() will select the column driver in data sheet and separate them city by city; and sum() function will provide us the total number of drivers in each city. There are three cities Urban, Suburban, and Rural and drivers were summed city by city.

Code:

total\_drivers\_by\_type = city\_data\_df.groupby(["type"]).sum()["driver\_count"]

Result:

type

Rural: 78

Suburban: 490

Urban: 2405

Name: driver\_count, dtype: int64

1. **Total Fares:**

To get the total fare amount for each city, we are using the groupby (), “Type” and sum () function. Where groupby(“type”) is used for splitting the three cities, and sum() function will provide the total fare amount was charged in each city.

Groupby() will select the column fare in data sheet and separate them city by city; and sum() function will provide us the total number of fare was charged in each city. Since there are three cities Urban, Suburban, and Rural and total fare was split into three cities.

Code:

total\_fare\_by\_type=pyber\_data\_df.groupby(["type"]).sum()["fare"]

Result:

type

Rural: 4327.93

Suburban: 19356.33

Urban: 39854.38

Name: fare, dtype: float64

1. **Average Fare per Ride:**

To find the average\_fare\_per\_ride we will divide the total\_fare\_by\_type with total\_ride\_by\_type. We have already worked in previous steps to get the the total\_fare\_by\_type per city and total\_rides\_by\_type in previous section.

Code:

average\_fare\_per\_ride = total\_fare\_by\_type / total\_rides\_by\_type

Result:

type

Rural 34.623440

Suburban 30.970128

Urban 24.525772

dtype: float64

1. **Average Fare per Driver:**

Similarly, to find the average\_fare\_per\_driver we will divide the total\_fare\_by\_type with total\_drivers\_by\_type. We have already worked in previous steps to get the the total\_fare\_by\_type per city and total\_drivers\_by\_type in previous section

Code:

average\_fare\_per\_driver = total\_fare\_by\_type / total\_drivers\_by\_type

Result:

type

Rural 55.486282

Suburban 39.502714

Urban 16.571468

dtype: float64

1. **Summary of DataFrame:**

To generate the final summary DataFrame, a list of dictionary was created. Summary for the DataFrame will help the reader to understand the total rides were taken, total drivers, total fare was charged, average fare per ride, and average fare per driver in each city.

We have created the list of columns and added the values in the columns, see the below code.

Code:

summary={

"Total Rides": total\_rides\_by\_type,

"Total Drivers": total\_drivers\_by\_type,

"Total Fares": total\_fare\_by\_type,

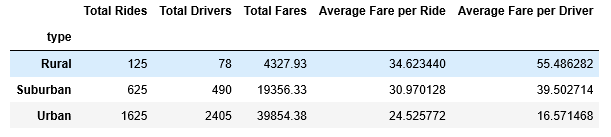
"Average Fare per Ride": average\_fare\_per\_ride,

"Average Fare per Driver": average\_fare\_per\_driver

}

pyber\_ride\_summary\_df = pd.DataFrame(summary)

Result:

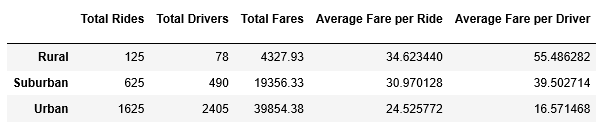


1. **Delete the Index mane from the summary:**

Code:

pyber\_ride\_summary\_df.index.name =""

Result:



1. **Format the pyber\_ride\_summary columns:**

Next step is to format the pyber\_ride\_summary columns and add two decimal places and add the Doller amount to average fare per ride and average fare per driver.

Code:

pyber\_ride\_summary\_df["Total Rides"]=pyber\_ride\_summary\_df["Total Rides"].map("{:,}".format)

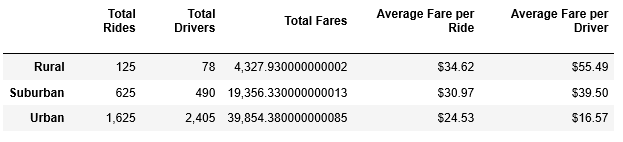
pyber\_ride\_summary\_df ["Total Drivers"] = pyber\_ride\_summary\_df["Total Drivers"].map("{:,}".format)

pyber\_ride\_summary\_df ["Total Fares"] = pyber\_ride\_summary\_df["Total Fares"].map("{:,}".format)

pyber\_ride\_summary\_df ["Average Fare per Ride"] = pyber\_ride\_summary\_df["Average Fare per Ride"].map("${:,.2f}".format)

pyber\_ride\_summary\_df ["Average Fare per Driver"] = pyber\_ride\_summary\_df["Average Fare per Driver"].map("${:,.2f}".format)

Result:



**Technical Analysis Deliverable 2: Multiple-Line Plot for the Sum of the Fares for Each City Type:**

1. **Rename All the columns**

We start with creating a new list of dictionary and rename all the columns with pyber\_data\_df. pyber\_data\_df is where we merge both city\_data and ride\_data. Changing the current column name to new column name.

"city": "City", "date": "Date", "fare": "Fare", "ride\_id": "Ride\_id", "driver\_count": "No. Drivers", "type": "City Type"

**Code:**

pyber\_data\_df=pyber\_data\_df.rename(columns={

"city": "City",

"date": "Date",

"fare": "Fare",

"ride\_id": "Ride\_id",

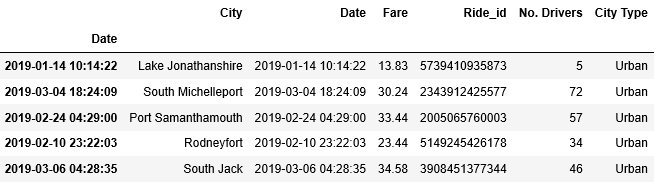
"driver\_count": "No. Drivers",

"type": "City Type"

})

pyber\_data\_df.head()

Result:



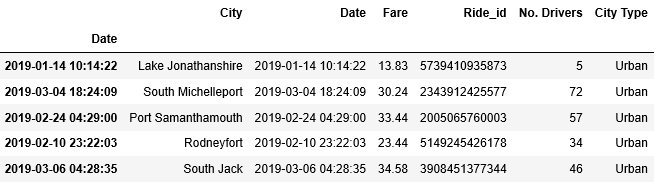
1. **Next step is to use date column as index.**

Code:

pyber\_data\_df.set\_index(pyber\_data\_df["Date"], inplace=True)

pyber\_data\_df.head()

Results:



1. **Make new Data Frame for fares and include only the Date City Type and Fare column:**

Create a new DataFrame list with must-needed columns. "Date","City Type","Fare"

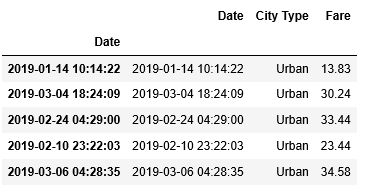
Code:

column\_names = ["Date","City Type","Fare"]

pyber\_cities\_fare = pyber\_data\_df[column\_names].copy()

pyber\_cities\_fare.head()

Result:



1. **Drop the extra columns:**

Since we convert the index to Date and we have another Date column showing, we will drop to duplicate columns.

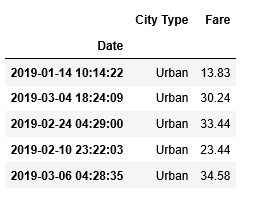
Code:

pyber\_cities\_fare.drop(["Date"], axis=1, inplace=True)

pyber\_cities\_fare.index = pd.to\_datetime(pyber\_data\_df.index)

pyber\_cities\_fare.head()

Result:



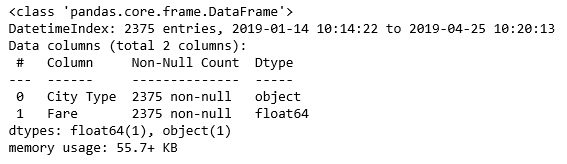
1. **Check the info about DataFram:**

Check to make sure the index is a datetime data type by using the info() method on the DataFrame.

Code:

pyber\_cities\_fare.info()

Result:



1. Calculate the Sum:

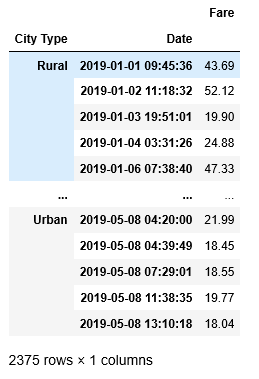
Calculate the sum of fares by the type of the city and date using groupby to create a series.

Code:

sum\_fare\_by\_type = pyber\_cities\_fare.groupby(["City Type", "Date"]).sum()[["Fare"]]

sum\_fare\_by\_type

Result:



1. **Convert the groupby Series into a DataFrame:**

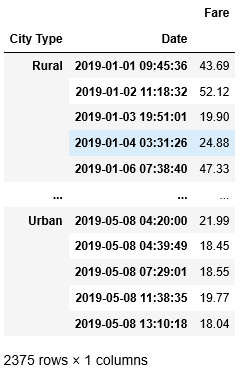
Convert using the pd.DataFrame method.

Code:

sum\_fare\_by\_type = pd.DataFrame(sum\_fare\_by\_type)

sum\_fare\_by\_type

Result:



1. **Reset the index,:**

Next step is to rest the index by using the reset\_index().

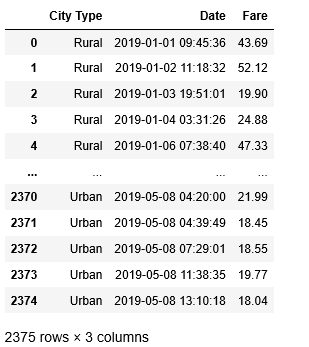
Code:

# Net step is to rest the index by using the reset\_index()

sum\_fare\_by\_type = sum\_fare\_by\_type.reset\_index()

sum\_fare\_by\_type

Result:



1. **Create a Pivot Table:**

Create a pivot table with the date as the index and column as the type with fare in each row.

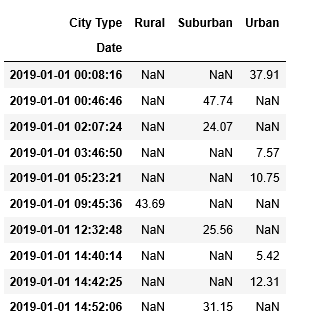
Code:

# To create a pivot table

sum\_fare\_by\_type\_pivot = sum\_fare\_by\_type.pivot(index="Date", columns="City Type")["Fare"]

sum\_fare\_by\_type\_pivot.head(10)

Result:



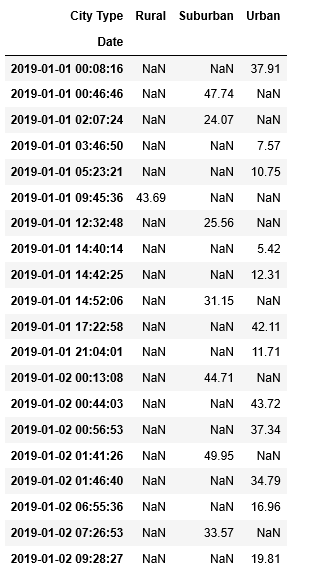
1. **Create a line chart that shows fares from Jan 1, 2019 to April 29, 2019:**

Code:

fare\_Jan\_April = sum\_fare\_by\_type\_pivot.loc["2019-01-01": "2019-04-28"]

fare\_Jan\_April.head(20)

Result:



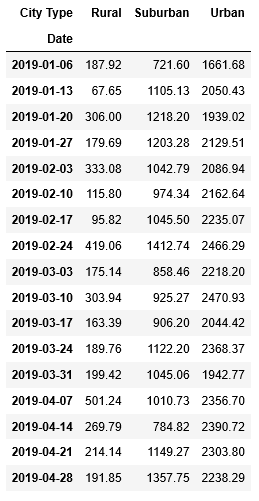
1. **Create a new DataFrame and use resample by week and get sum of the fares for each week:**

Code:

weekly\_fares\_df = fare\_Jan\_April.resample("W").sum()

weekly\_fares\_df

Result:



1. **Use axis plotting to add lables and titles**:

Using the object-oriented interface method, plot the DataFrame you created in previous step using the df.plot() function. Things to consider with your plotting:

Import the style from Matplotlib.

Use the FiveThirtyEight style sheet graph style.

Add a title.

Add x- and y-axis labels according to the final figure.

Save the figure to the “analysis” folder.

Make the figure size large enough so it’s not too small

Code:

from matplotlib import style

style.use("fivethirtyeight")

ax = weekly\_fares\_df.plot(figsize=(20,6))

# Adding a title

ax.set\_title("Total Fare by City Type")

# Add a X-axis and Y-axis label

ax.set\_xlabel("Month")

ax.set\_ylabel("Fare ($USD)")

plt.savefig("./challenge\_fare\_summary.png")

plt.show()

Result:

