Pyber Challenge

4.3 Loading and Reading CSV files

```
In [ ]: # Add Matplotlib inline magic command
        %matplotlib inline
        # Dependencies and Setup
        import matplotlib.pyplot as plt
        import pandas as pd

        # File to Load csv (Remember to change these)PyBer_ride_data.csv
        pyber_ride_df = pd.read_csv("Resources/PyBer_ride_data.csv")
        pyber_ride_df
        pyber_city_df = pd.read_csv("Resources/PyBer_city_data.csv")
        pyber_city_df

        #city_data_to_load = "city_data.csv"
        #ride_data_to_load = "ride_data.csv"

        # Read the City and Ride Data
        #city_data_df = pd.read_csv(pyber_city_df)
        #ride_data_df = pd.read_csv(pyber_ride_df)
```

Merge the DataFrames

```
In [ ]: # Combine the data into a single dataset
        pyber_data_df = pd.merge(pyber_ride_df, pyber_city_df, how="left", on=

        # Display the data table for preview
        pyber_data_df.dtypes
```

Deliverable 1: Get a Summary DataFrame

```
In [ ]: #Create the Urban city DateFrame
        urban_cities_df = pyber_data_df[pyber_data_df["type"] == "Urban"]
        urban_cities_df.head()
```

```
In [ ]: #Create the Suburban city DateFrame
        suburban_cities_df= pyber_data_df[pyber_data_df["type"] == "Suburban"]
        suburban_cities_df.head()
```

```
In [ ]: rural_cities_df = pyber_data_df[pyber_data_df["type"] == "Rural"]
        rural_cities_df.head()
```

In [ ]:
```
#  1. Get the total rides for each city type
total_rides_by_city_type = pyber_data_df.groupby(["type"]).count()["ri
total_rides_by_city_type.head()
```

In [ ]:
```
# 2. Get the total drivers for each city type
total_drivers_by_city_type = pyber_city_df.groupby(["type"]).count()["
total_drivers_by_city_type.head()
```

In [ ]:
```
#  3. Get the total amount of fares for each city type
total_fare_city_type = pyber_data_df.groupby(["type"]).sum()["fare"]
total_fare_city_type.head()
```

In [ ]:
```
#  4. Get the average fare per ride for each city type.
average_fare_per_ride_city_type = total_fare_city_type / total_rides_b
average_fare_per_ride_city_type.head()
```

In [ ]:
```
#5 Get the average number of drivers for each city.
average_number_drivers_city = total_fare_city_type / total_drivers_by_
average_number_drivers_city.head()
```

In [ ]:
```
#  7. Cleaning up the DataFrame. Delete the index name
total_rides_by_city_type.index.name = None
total_rides_by_city_type
```

In [ ]:
```
#  6. Create a PyBer summary DataFrame.
# Combine the data into a single dataset
pyber_summary_df = pd.DataFrame(
            {'Total Rides':total_rides_by_city_type,
             'Total Drivers': total_drivers_by_city_type,
             'Total Fares':total_fare_city_type,
             #'Average Fare per Ride':average_fare_per_ride_city_type,
             'Average Fare per Driver':average_number_drivers_city})
pyber_summary_df
# Display the data table for preview
pyber_summary_df
```

In [ ]:
```
total_drivers_by_city_type.index.name = None
total_drivers_by_city_type
```

In [ ]:
```
total_fare_city_type.index.name = None
total_fare_city_type
```

In [ ]:
```
#  8. Format the columns.
summary_df = pd.DataFrame()

summary_df["Total Rides"] = total_rides_by_city_type
summary_df["Total Drivers"] = total_drivers_by_city_type
summary_df["Total Fares"] = total_fare_city_type
summary_df["Average Fare per Ride"] = average_fare_per_ride_city_type
summary_df["Average Fare per Driver"] = total_drivers_by_city_type

summary_df
```

## Deliverable 2. Create a multiple line plot that shows the total weekly of the fares for each type of city.

In [ ]:
```
# 1. Read the merged DataFrame (use groupby(), and summ())

sum_fares = pyber_cities_fares_df.groupby(["City Type","Date"]).sum()[
sum_fares
```

In [5]:
```
# 2. Using groupby() to create a new DataFrame showing the sum of the

pyber_cities_fares= pyber_data_df[["Date","City Type","Fare"]].copy()
pyber_cities_fares.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
Input In [5], in <cell line: 3>()
      1 # 2. Using groupby() to create a new DataFrame showing the su
m of the fares
----> 3 pyber_cities_fares= pyber_data_df[["Date","City Type","Far
e"]].copy()
      4 pyber_cities_fares.head()

NameError: name 'pyber_data_df' is not defined
```

In [ ]:
```
# 3. Reset the index on the DataFrame you created in #1. This is neede
# df = df.reset_index()
sum_fares_df = sum_fares_df.reset_index()
sum_fares_df.head(10)
```

In [ ]:
```
# 4. Create a pivot table with the 'date' as the index, the columns ='
# to get the total fares for each type of city by the date.
sum_fares_pivot = sum_fares_df.pivot(index="date",columns="city type")
sum_fares_pivot
```

In [ ]:
```
# 5. Create a new DataFrame from the pivot table DataFrame using loc o
fares_Jan_to_Apr_df = sum_fares_pivot.loc['2019-01-01':'2019-04-29']
fares_Jan_Apr_df
```

In [4]:
```
# 6. Set the "date" index to datetime datatype. This is necessary to u
# df.index = pd.to_datetime(df.index)
pyber_cities_fares.index = pd.to_datetime(pyber_date_df.index)
pyber_cities_fares.head(10)
```

```
---------------------------------------------------------------------
------
NameError                                 Traceback (most recent call
last)
Input In [4], in <cell line: 3>()
      1 # 6. Set the "date" index to datetime datatype. This is neces
sary to use the resample() method in Step 8.
      2 # df.index = pd.to_datetime(df.index)
----> 3 pyber_cities_fares.index = pd.to_datetime(pyber_date_df.inde
x)
      4 pyber_cities_fares.head(10)

NameError: name 'pd' is not defined
```

In [ ]:
```
# 7. Check that the datatype for the index is datetime using df.info()
jan_to_apr_df.info()
```

In [ ]:
```
# 8. Create a new DataFrame using the "resample()" function by week 'W
weekly_fares_df = fares_Jan_Apr_df.resample("W").sum()
weekly_fares_df
```

In [ ]:
```
# 8. Using the object-oriented interface method, plot the resample Dat

plt.sytle.use'fivethirtyeight')
weekly_fares_df.plot(figsize = 14,8))
plt.gcf().subplots_adjust(bottom= 0.15)

#Add graph properties
plt.title("Total Fare by City Type")
plt.ylabel("Fare($USD)")
plt.xlabel("Month",fontsize = 14)

#Create a legend
lgnd= plt.lengend(fontsize="12", loc = "best", title = "City Type")
lgnd= get.tlte().set_fountsize(12)

#save figure
plt.savefig(Analysis/Fig5.png)

#show figure
plt.show()
```

In [ ]: