PyBer_Challenge

March 6, 2022

1 Pyber Challenge

1.0.1 4.3 Loading and Reading CSV files

[1]:
```
# Add Matplotlib inline magic command
%matplotlib inline
import pandas as pd

# File to Load (Remember to change these)
city_data_to_load = "Resources/city_data.csv"
#city_data_to_load = '/Users/lukeperrin/Documents/UCB Data Bootcamp/
 ↪module-repos/05-PyBer-lukeperrin/Pyber_Challenge/Resources/city_data.csv'
ride_data_to_load = "Resources/ride_data.csv"
# ride_data_to_load = '/Users/lukeperrin/Documents/UCB Data Bootcamp/
 ↪module-repos/05-PyBer-lukeperrin/Pyber_Challenge/Resources/ride_data.csv'



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

1.0.2 Merge the DataFrames

[2]:
```
# Combine the data into a single dataset
pyber_data_df = pd.merge(ride_data_df, city_data_df, how="left", on=["city",␣
 ↪"city"])

# Display the data table for preview
pyber_data_df
```

	city	date	fare	ride_id	driver_count	type
0	Lake Jonathanshire	2019-01-14 10:14:22	13.83	5739410935873	5	Urban
1	South Michelleport	2019-03-04 18:24:09	30.24	2343912425577	72	Urban
2	Port Samanthamouth	2019-02-24 04:29:00	33.44	2005065760003	57	Urban
3	Rodneyfort	2019-02-10 23:22:03	23.44	5149245426178	34	Urban
4	South Jack	2019-03-06 04:28:35	34.58	3908451377344	46	Urban
...	...	...	...	...	...	...
2370	Michaelberg	2019-04-29 17:04:39	13.38	8550365057598	6	Rural
2371	Lake Latoyabury	2019-01-30 00:05:47	20.76	9018727594352	2	Rural
2372	North Jaime	2019-02-10 21:03:50	11.11	2781339863778	1	Rural
2373	West Heather	2019-05-07 19:22:15	44.94	4256853490277	4	Rural
2374	Newtonview	2019-04-25 10:20:13	55.84	9990581345298	1	Rural

2375 rows × 6 columns

1.1 Challenge Deliverable 1. Generate a Ride-Sharing DataFrame by City Type

[3]:
```
#  1. Get the total rides for each city type
totalRides_byCityType = pyber_data_df.groupby(["type"]).count()["ride_id"]
totalRides_byCityType
```

[3]:
```
type
Rural        125
Suburban     625
Urban       1625
Name: ride_id, dtype: int64
```

[4]:
```
# 2. Get the total drivers for each city type
totalDrivers_byCityType = city_data_df.groupby(["type"]).sum()["driver_count"]
totalDrivers_byCityType
```

[4]:
```
type
Rural         78
Suburban     490
Urban       2405
Name: driver_count, dtype: int64
```

[5]:
```
#  3. Get the total amount of fares for each city type
totalFares_byCityType = pyber_data_df.groupby(["type"]).sum()["fare"]
totalFares_byCityType
```

[5]:
```
type
Rural        4327.93
```

```
Suburban    19356.33
Urban       39854.38
Name: fare, dtype: float64
```

[6]:
```
#  4. Get the average fare per ride for each city type.
avgFaresPerRide_byCityType = round(totalFares_byCityType /␣
 ↪totalRides_byCityType, 2)
avgFaresPerRide_byCityType
```

[6]:
```
type
Rural       34.62
Suburban    30.97
Urban       24.53
dtype: float64
```

[7]:
```
# 5. Get the average fare per driver for each city type.
avgFaresPerDriver_byCityType = round(totalFares_byCityType /␣
 ↪totalDrivers_byCityType, 2)
avgFaresPerDriver_byCityType
```

[7]:
```
type
Rural       55.49
Suburban    39.50
Urban       16.57
dtype: float64
```

[8]:
```
#  6. Create a PyBer summary DataFrame.
pyber_summary_df = pd.DataFrame({
    "Total Rides" : totalRides_byCityType,
    "Total Drivers" : totalDrivers_byCityType,
    "Total Fares" : totalFares_byCityType,
    "Average Fare per Ride" : avgFaresPerRide_byCityType,
    "Average Fare per Driver" : avgFaresPerDriver_byCityType
})
pyber_summary_df
```

	Total Rides	Total Drivers	Total Fares	Average Fare per Ride	Average Fare per Driver
type					
Rural	125	78	4327.93	34.62	55.49
Suburban	625	490	19356.33	30.97	39.50
Urban	1625	2405	39854.38	24.53	16.57

[9]:
```
#  7. Cleaning up the DataFrame. Delete the index name
pyber_summary_df.index.name = None
```

```
pyber_summary_df
```

	Total Rides	Total Drivers	Total Fares	Average Fare per Ride	Average Fare per Driver
Rural	125	78	4327.93	34.62	55.49
Suburban	625	490	19356.33	30.97	39.50
Urban	1625	2405	39854.38	24.53	16.57

[10]:
```
#  8. Format the columns.
pyber_summary_df["Total Rides"] = pyber_summary_df["Total Rides"].map("{:,}".
 ↪format)
pyber_summary_df["Total Drivers"] = pyber_summary_df["Total Drivers"].map("{:
 ↪,}".format)
pyber_summary_df["Total Fares"] = pyber_summary_df["Total Fares"].map("${:,.
 ↪2f}".format)
pyber_summary_df["Average Fare per Ride"] = pyber_summary_df["Average Fare per␣
 ↪Ride"].map("${:,.2f}".format)
pyber_summary_df["Average Fare per Driver"] = pyber_summary_df["Average Fare␣
 ↪per Driver"].map("${:,.2f}".format)
pyber_summary_df
```

	Total Rides	Total Drivers	Total Fares	Average Fare per Ride	Average Fare per Driver
Rural	125	78	$4,327.93	$34.62	$55.49
Suburban	625	490	$19,356.33	$30.97	$39.50
Urban	1,625	2,405	$39,854.38	$24.53	$16.57

[11]:
```
# Export pyber_summary_df to PNG image
# import dataframe_image as dfi
```

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

[12]:
```
# Print the merged DataFrame for reference.
pyber_data_df
```

	city	date	fare	ride_id	driver_count	type
0	Lake Jonathanshire	2019-01-14 10:14:22	13.83	5739410935873	5	Urban
1	South Michelleport	2019-03-04 18:24:09	30.24	2343912425577	72	Urban
2	Port Samanthamouth	2019-02-24 04:29:00	33.44	2005065760003	57	Urban
3	Rodneyfort	2019-02-10 23:22:03	23.44	5149245426178	34	Urban
4	South Jack	2019-03-06 04:28:35	34.58	3908451377344	46	Urban
...	...	...	...	...	...	...
2370	Michaelberg	2019-04-29 17:04:39	13.38	8550365057598	6	Rural
2371	Lake Latoyabury	2019-01-30 00:05:47	20.76	9018727594352	2	Rural
2372	North Jaime	2019-02-10 21:03:50	11.11	2781339863778	1	Rural
2373	West Heather	2019-05-07 19:22:15	44.94	4256853490277	4	Rural
2374	Newtonview	2019-04-25 10:20:13	55.84	9990581345298	1	Rural

2375 rows × 6 columns

1.2.1 1. Using groupby() to create a new DataFrame showing the sum of the fares for each date where the indices are the city type and date.

[13]:
```
sumFare_byDate_byType = pyber_data_df.groupby(['type', 'date']).sum()['fare']
sumFare_byDate_byType
```

[13]:
```
type   date
Rural  2019-01-01 09:45:36    43.69
       2019-01-02 11:18:32    52.12
       2019-01-03 19:51:01    19.90
       2019-01-04 03:31:26    24.88
       2019-01-06 07:38:40    47.33
                               ...
Urban  2019-05-08 04:20:00    21.99
       2019-05-08 04:39:49    18.45
       2019-05-08 07:29:01    18.55
       2019-05-08 11:38:35    19.77
       2019-05-08 13:10:18    18.04
Name: fare, Length: 2375, dtype: float64
```

1.2.2 2. Reset the index on the DataFrame you created in #1. This is needed to use the 'pivot()' function.

[14]:
```
sumFare_byDate_byType = sumFare_byDate_byType.reset_index()
```

1.2.3 3. Create a pivot table with the 'date' as the index, the columns ='type', and values='fare' to get the total fares for each type of city by the date.

```
[15]: sumFare_byDate_byType_pivot = sumFare_byDate_byType.pivot(index='date',␣
       ↪columns='type', values='fare')
      sumFare_byDate_byType_pivot
```

type	Rural	Suburban	Urban
date			
2019-01-01 00:08:16	NaN	NaN	37.91
2019-01-01 00:46:46	NaN	47.74	NaN
2019-01-01 02:07:24	NaN	24.07	NaN
2019-01-01 03:46:50	NaN	NaN	7.57
2019-01-01 05:23:21	NaN	NaN	10.75
...	...	...	...
2019-05-08 04:20:00	NaN	NaN	21.99
2019-05-08 04:39:49	NaN	NaN	18.45
2019-05-08 07:29:01	NaN	NaN	18.55
2019-05-08 11:38:35	NaN	NaN	19.77
2019-05-08 13:10:18	NaN	NaN	18.04

2375 rows × 3 columns

1.2.4 4. Create a new DataFrame from the pivot table DataFrame using loc on the given dates, '2019-01-01':'2019-04-29'.

```
[16]: deliv2_step4_df = sumFare_byDate_byType_pivot.loc['2019-01-01':'2019-04-29']
      deliv2_step4_df
```

type	Rural	Suburban	Urban
date			
2019-01-01 00:08:16	NaN	NaN	37.91
2019-01-01 00:46:46	NaN	47.74	NaN
2019-01-01 02:07:24	NaN	24.07	NaN
2019-01-01 03:46:50	NaN	NaN	7.57
2019-01-01 05:23:21	NaN	NaN	10.75
...	...	...	...
2019-04-28 14:28:36	NaN	NaN	11.46
2019-04-28 16:29:16	NaN	NaN	36.42
2019-04-28 17:26:52	NaN	NaN	31.43
2019-04-28 17:38:09	NaN	34.87	NaN
2019-04-28 19:35:03	NaN	16.96	NaN

2196 rows × 3 columns

1.2.5 5. Set the "date" index to datetime datatype. This is necessary to use the resample() method in Step 8.

[17]:
```
deliv2_step4_df.index = pd.to_datetime(deliv2_step4_df.index)
```

1.2.6 6. Check that the datatype for the index is datetime using df.info()

[18]:
```
deliv2_step4_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2196 entries, 2019-01-01 00:08:16 to 2019-04-28 19:35:03
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Rural     114 non-null    float64
 1   Suburban  573 non-null    float64
 2   Urban     1509 non-null   float64
dtypes: float64(3)
memory usage: 68.6 KB
```

1.2.7 7. Create a new DataFrame using the "resample()" function by week 'W' and get the sum of the fares for each week.

[19]:
```
deliv2_step4_df_resampled = deliv2_step4_df.resample('W').sum()
deliv2_step4_df_resampled
```

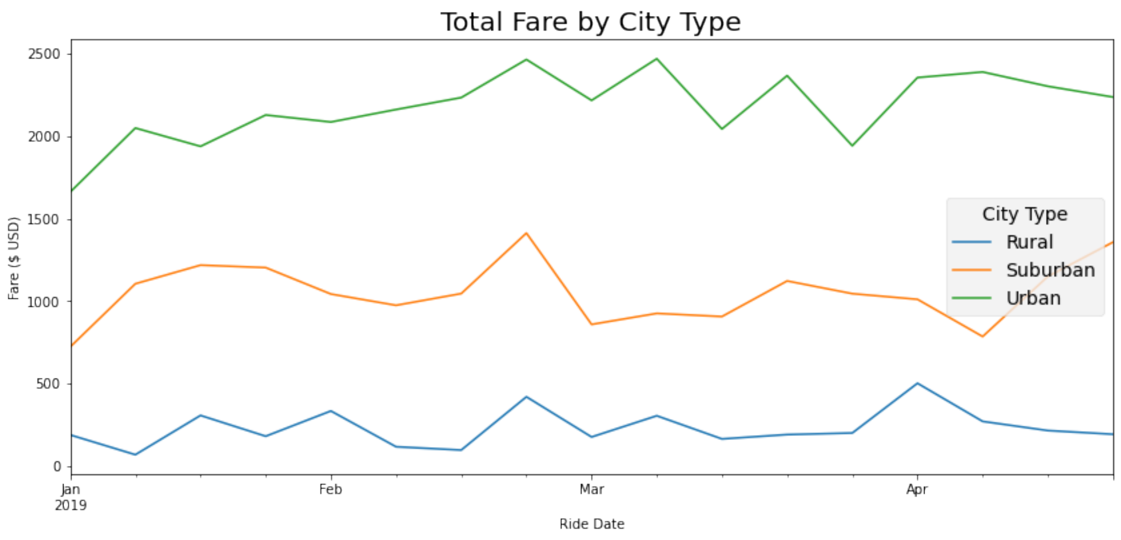
type	Rural	Suburban	Urban
date			
2019-01-06	187.92	721.60	1661.68
2019-01-13	67.65	1105.13	2050.43
2019-01-20	306.00	1218.20	1939.02
2019-01-27	179.69	1203.28	2129.51
2019-02-03	333.08	1042.79	2086.94
2019-02-10	115.80	974.34	2162.64
2019-02-17	95.82	1045.50	2235.07
2019-02-24	419.06	1412.74	2466.29
2019-03-03	175.14	858.46	2218.20
2019-03-10	303.94	925.27	2470.93
2019-03-17	163.39	906.20	2044.42
2019-03-24	189.76	1122.20	2368.37
2019-03-31	199.42	1045.06	1942.77
2019-04-07	501.24	1010.73	2356.70
2019-04-14	269.79	784.82	2390.72
2019-04-21	214.14	1149.27	2303.80
2019-04-28	191.85	1357.75	2238.29

[ ]:
```
import dataframe_image as dfi
>>> dfi.export(df_styled, 'df_styled.png')
```

1.2.8 8. Using the object-oriented interface method, plot the resample DataFrame using the df.plot() function.

[20]:
```
# Import the style from Matplotlib.
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib import style

# Define plot data sources and results destinations
output_img = 'Analysis/PyBer_fare_summary.png' #Multiline chart image name &␣
 ↪destination

# Plot from DataFrame
deliv2_step4_df_resampled.plot(title="Total Fare by City Type", xlabel='Date',␣
 ↪ylabel='Ride Fare ($ USD)', figsize=(14,6))

# Use the graph style fivethirtyeight
```

```
style.use('fivethirtyeight')

### Set plot parameters
plt.title('Total Fare by City Type')
plt.legend(loc='best', title='City Type')
plt.xlabel("Ride Date")
plt.ylabel("Fare ($ USD)")

# Save as PNG image
plt.savefig(output_img)
```



[20]:

```
```