HW02 – SF Bike Share Data

This assignment looks at the bike share data from the San Francisco Metro Transit Authority. You are to take four datasets and import them into a Jupyter Notebook and create a notebook according to the instructions listed below.

FILES

1. status.csv **– records of bike and dock availability by minute**
   - station_id: station ID number (corresponds to "station_id" in "station.csv" for corresponding station information)
   - bikes_available: number of available bikes
   - time: date and time, PST
2. station.csv **– records of station ID, name, latitude, longitude, dockcount, city, installation date**
   - station_id: station ID number (corresponds to "station_id" in "status.csv")
   - name: name of station
   - lat: latitude
   - long: longitude
   - dockcount: number of total docks at station
   - landmark: city (San Francisco, Redwood City, Palo Alto, Mountain View, San Jose)
   - Zip: 94107=San Francisco, 94063=Redwood City, 94301=Palo Alto, 94041=Mountain View, 95113= San Jose"
3. trip.csv **– records of individual trips**
   - Trip ID: numeric ID of bike trip
   - Duration: time of trip in seconds (trips <1 min and >24 hours are excluded)
   - Start Date: start date of trip with date and time, in PST
   - Start Station: station name of start station
   - Start Terminal: numeric reference for start station
   - End Date: end date of trip with date and time, in PST
   - End Station: station name for end station
   - End Terminal: numeric reference for end station
   - Bike #: ID of bike used
   - Subscription Type: Subscriber = annual or 30-day member; Customer = 24-hour

4. weather.csv **– Records of daily weather by city.** Daily weather information per service area, provided from Weather Underground in PST. Weather is listed from north to south (San Francisco, Redwood City, Palo Alto, Mountain View, San Jose).
    - Precipitation_Inches "numeric, in form x.xx but alpha ""T""= trace when amount less than .01 inch"
    - Cloud_Cover "scale of 0-8, 0=clear"
    - Zip: 94107=San Francisco, 94063=Redwood City, 94301=Palo Alto, 94041=Mountain View, 95113= San Jose"

Files contain data from 9/1/15 to 8/31/16. This is the third year of Bay Area Bike Share's operation.

Format of this Homework

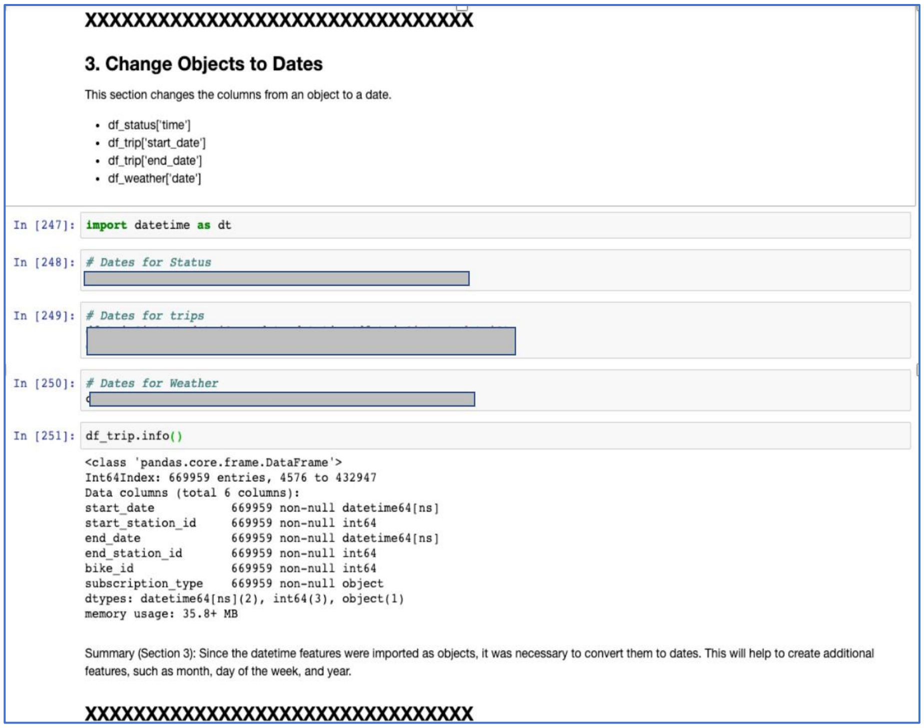
It is very important that your Jupyter Notebook is formatted correctly with markdown, comments, and code that works.

You are to do the following for each section (See Figure 1 below for title examples):

- Include a title as markdown Heading 2, for example: “Section 3. View Data”
- Include a description of the section detailing its purpose (*markdown*)
- Include your code and make sure it is executable and correct. (*code*)
- At the end of the section, include a brief summary describing results. Your summaries should get more detailed as the course progresses. (*markdown*)

How to turn it in:

- Your Jupyter notebook file must be named HW02_LastnameFirstInitial.ipynb. For example, HW02_ApigianC.ipynb.
- You are to turn in your Jupyter notebook file only. No data files and no folders.
- It is assumed that you created your Jupyter notebook in a folder named HW02_student and in that folder is a data folder. It is expected the path for importing data is in “data” folder, for example ‘**data/status.csv’**.



XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

3. Change Objects to Dates

This section changes the columns from an object to a date.

- df_status['time']
- df_trip['start_date']
- df_trip['end_date']
- df_weather['date']

```
In [247]: import datetime as dt
```

```
In [248]: # Dates for Status
```

[redacted]

```
In [249]: # Dates for trips
```

[redacted]

```
In [250]: # Dates for Weather
```

[redacted]

```
In [251]: df_trip.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 669959 entries, 4576 to 432947
Data columns (total 6 columns):
start_date           669959 non-null datetime64[ns]
start_station_id     669959 non-null int64
end_date             669959 non-null datetime64[ns]
end_station_id       669959 non-null int64
bike_id              669959 non-null int64
subscription_type    669959 non-null object
dtypes: datetime64[ns](2), int64(3), object(1)
memory usage: 35.8+ MB
```

Summary (Section 3): Since the datetime features were imported as objects, it was necessary to convert them to dates. This will help to create additional features, such as month, day of the week, and year.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

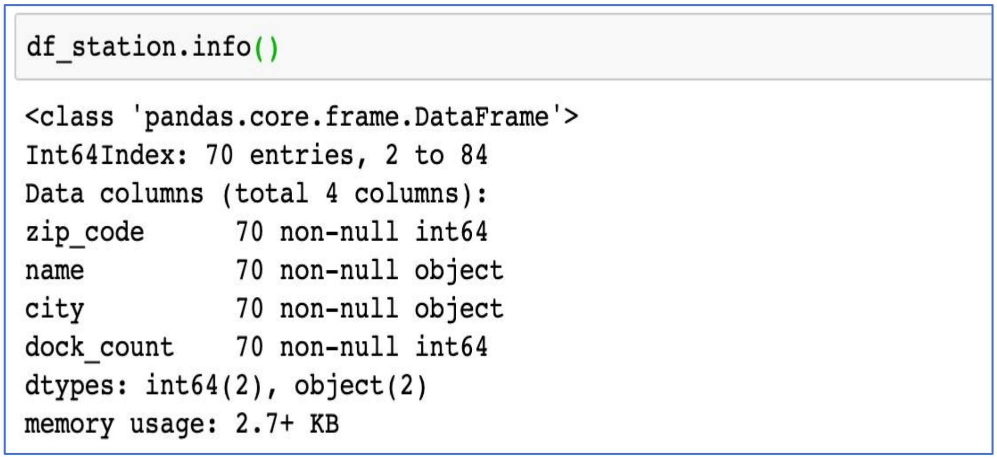
Figure 1: Screenshot of Section 3, with a title, description, code, and summary 1. Import Libraries

1. Import Libraries

- numpy, pandas, matplotlib, seaborn, and datetime
- You will also want to add the following:
  - %matplotlib inline
  - pd.set_option('display.max_columns',500)
  - plt.style.use('seaborn')

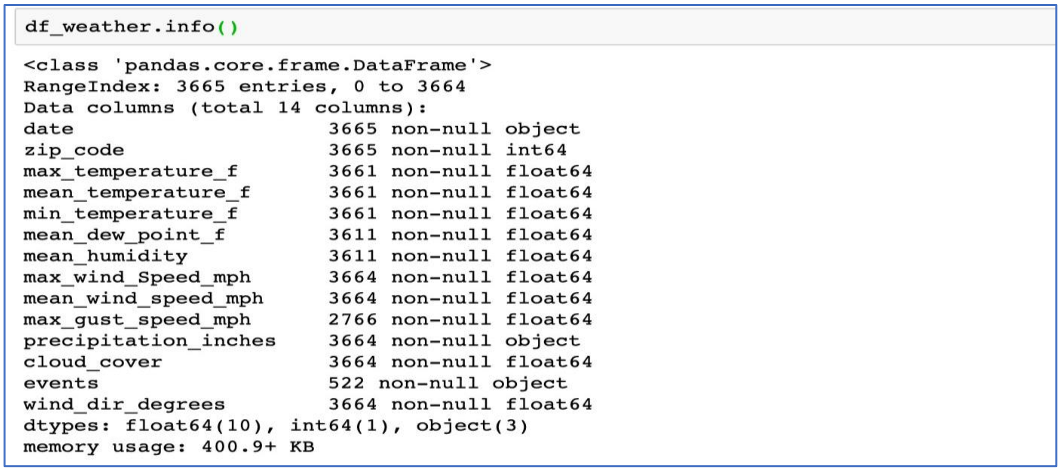
2. Import Data

- station.csv as df_station with index_col = 0 and header=0
  - Then select the columns in the order shown below.



```
df_station.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 70 entries, 2 to 84
Data columns (total 4 columns):
zip_code      70 non-null int64
name          70 non-null object
city          70 non-null object
dock_count    70 non-null int64
dtypes: int64(2), object(2)
memory usage: 2.7+ KB
```

- weather.csv as df_weather with index_col = None and header=0
  - Then select the columns in the order shown below.



```
df_weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3665 entries, 0 to 3664
Data columns (total 14 columns):
date                     3665 non-null object
zip_code                 3665 non-null int64
max_temperature_f        3661 non-null float64
mean_temperature_f       3661 non-null float64
min_temperature_f        3661 non-null float64
mean_dew_point_f         3611 non-null float64
mean_humidity            3611 non-null float64
max_wind_Speed_mph       3664 non-null float64
mean_wind_speed_mph      3664 non-null float64
max_gust_speed_mph       2766 non-null float64
precipitation_inches     3664 non-null object
cloud_cover              3664 non-null float64
events                   522 non-null object
wind_dir_degrees         3664 non-null float64
dtypes: float64(10), int64(1), object(3)
memory usage: 400.9+ KB
```

- status.csv as df_status with index_col = None and header=0
  - Keep all features
- trip.csv as df_trip with index_col = 0 and header=0
  - Keep all features

3. Change Objects to Dates

- df_status['time'], df_trip['start_date'], df_trip['end_date'], and df_weather['date']

4. Create additional Date Features

- For df_weather:
  - Create a ['day'] feature with number for dayofweek
  - Create a ['month'] feature with number for month
  - Create a ['year'] feature with number for year
- For df_trip – all based on ['start_date']:
  - Create a ['date'] feature with number for date
  - Create a ['day'] feature with number for dayofweek
  - Create a ['month'] feature with number for month iv. Create a ['year'] feature with number for year
  - Create a ['day_of_week'] feature with number for day_name( )
  - Create a ['trip_time'] feature by finding the difference between ['start_date'] and ['end_date']
  - Create a ['trip_time_m'] features by converting ['trip_time'] to minutes

Helpful code:

df_trip['trip_time_m'] = pd.to_timedelta(df_trip['trip_time']).astype('timedelta64[m]').astype(int)

- For df_status
  - Create a ['date'] feature based on ['time']

5. **Explore and Visualize Weather (df_weather)**
   - Show the mean for all features based on [‘month’]
   - Show the mean for [‘mean_temperature_f’] based on [‘year’] and [‘month’]
   - Show a barplot for [‘mean_temperature_f’] for each [‘month’]

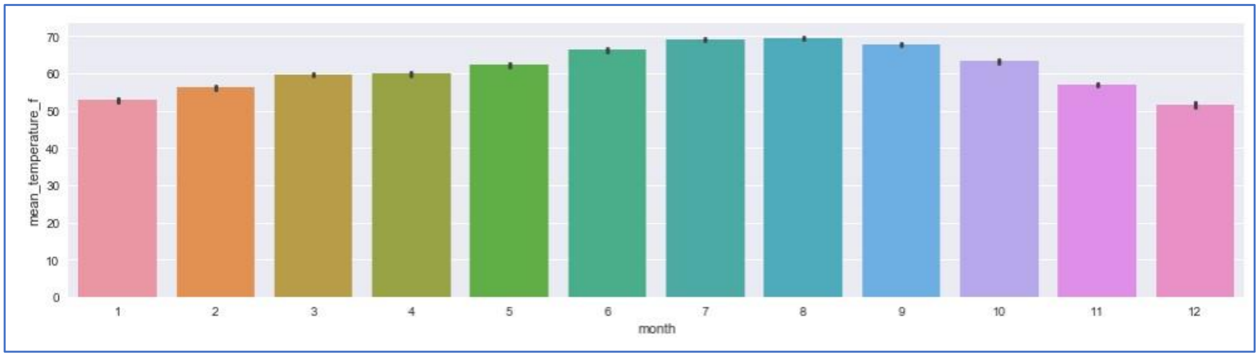
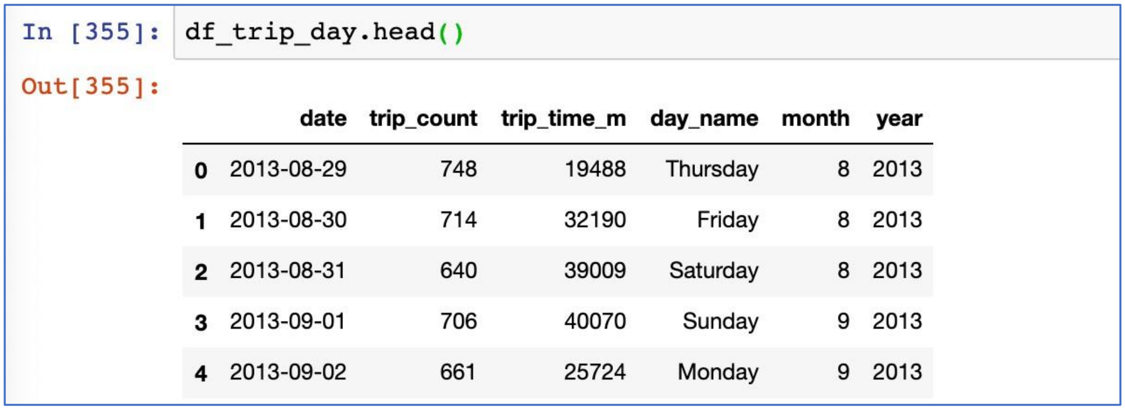


Figure 2: Barplot for mean_temperature_f’ for each month (your barplot may look different)

6. **Explore and Visualize Trips**
   - Create a new dataframe named df_trip_day which groups trips based on [‘date’] and looks like:



```
In [355]: df_trip_day.head()
```

Out[355]:

	date	trip_count	trip_time_m	day_name	month	year
0	2013-08-29	748	19488	Thursday	8	2013
1	2013-08-30	714	32190	Friday	8	2013
2	2013-08-31	640	39009	Saturday	8	2013
3	2013-09-01	706	40070	Sunday	9	2013
4	2013-09-02	661	25724	Monday	9	2013

   - To do this:
     - Create the Dataframe, df_trip_day, by grouping based on [‘date’] and include the count of [‘bike_id’]. Then reset the index and rename the count feature to [‘trip_count’].
     - Create another Dataframe named df_trip_m, by grouping based on [‘date’] and include the sum of [‘trip_time_m]. Then reset the index.

Helpful Code:
df_trip_m = pd.DataFrame(df_trip.groupby(['date'])['trip_time_m'].sum())
df_trip_m = df_trip_m.reset_index()

     - Merge df_trip_day and df_trip_m on [‘date’] and name it df_trip_day.
     - Add a day_name, month, and year feature to df_trip_day.

- Create a visualization (**Figure 3**) that shows two boxplots – one for trip_count and one for trip_time_m for each day of the week.

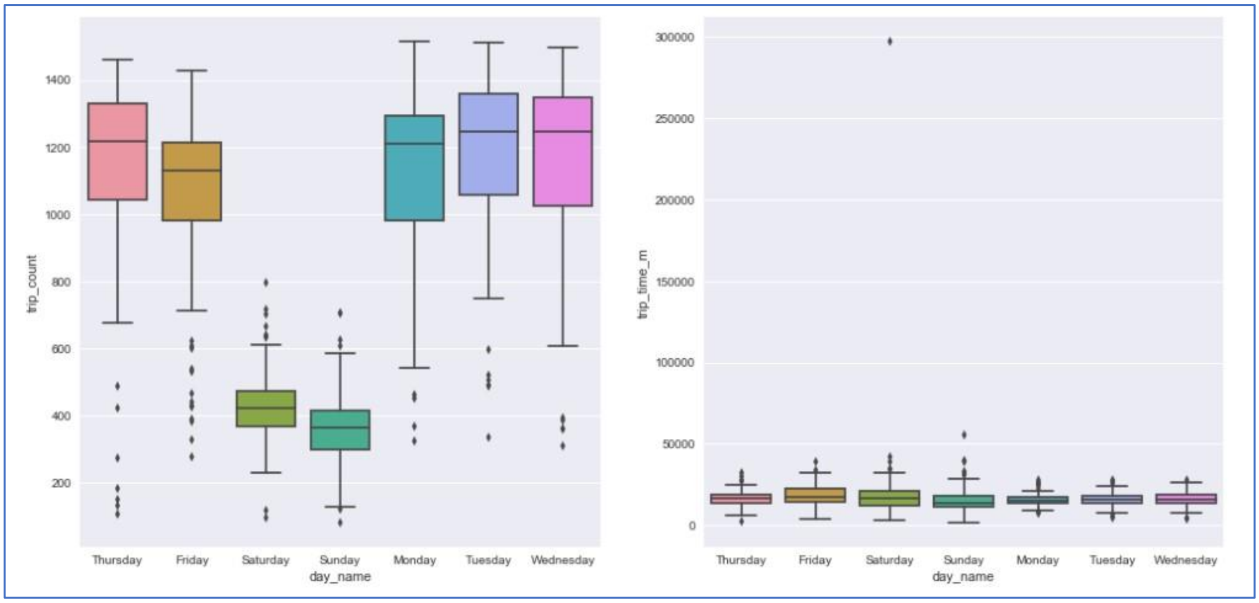


Figure 3: Boxplots for trip_count and trip_time_m per week day

7. Find the outlier and recreate visualization

- What happened in the trip_time_m on Saturday?
  - Review the data and find the outlier.
  - Drop the outlier from the DataFrame.
  - Redo the visualizations to look like Figure 4.

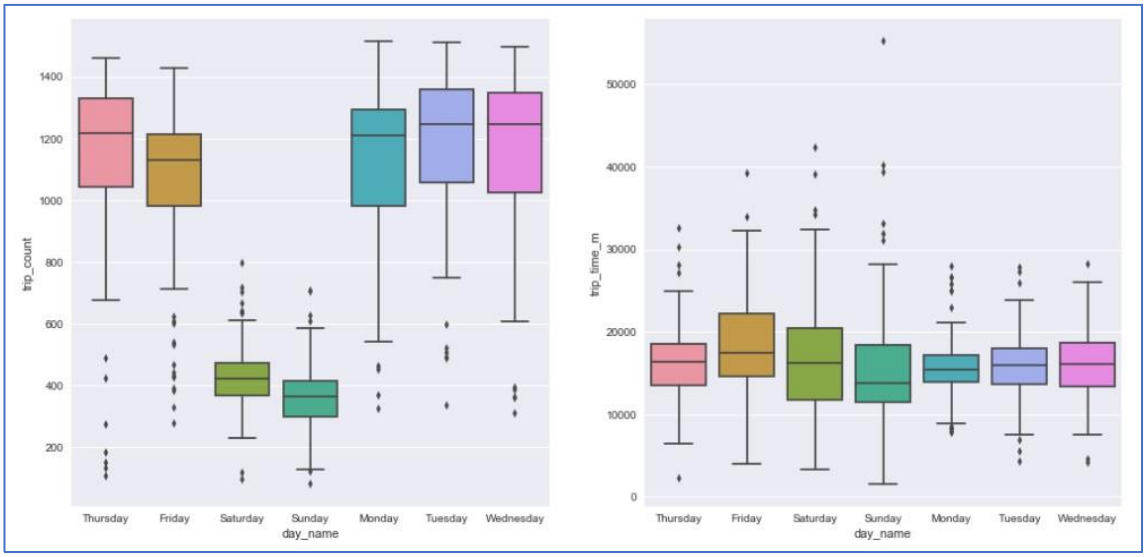
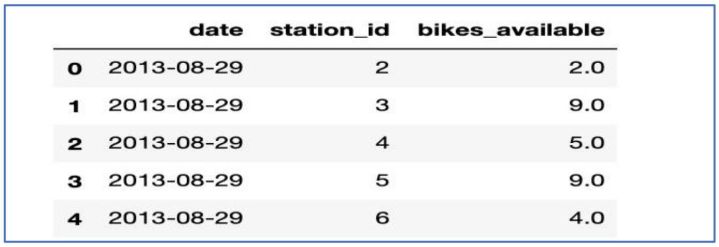


Figure 4: Boxplots for trip_count and trip_time_m per week day

8. Merge datasets together into one main Dataframe

- Create a new Dataframe from df_status named df_status_day
  - Group based on [‘date’ and ‘station_id’] and calculate the median for all features
  - Reset the index.
  - df_status_day show be similar to:



	date	station_id	bikes_available
0	2013-08-29	2	2.0
1	2013-08-29	3	9.0
2	2013-08-29	4	5.0
3	2013-08-29	5	9.0
4	2013-08-29	6	4.0

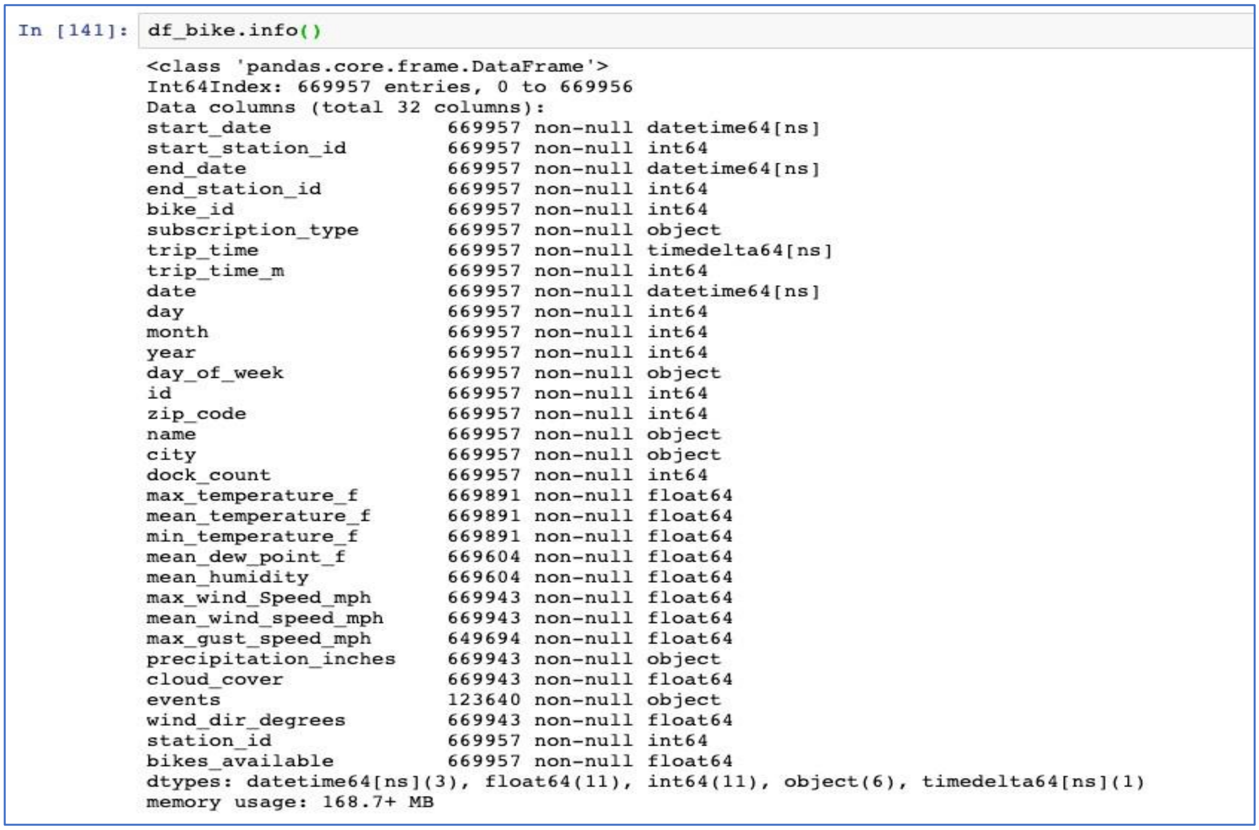
- Create a df_bike Dataframe by merging the four datasets together using:
  - df_weather
  - df_trip
  - df_status_day
  - df_station
- In order to merge all 4 DataFrames together, you will need to identify the process and primary and secondary keys.
  - Look at the screenshot below, it will give you a clear idea of how everything was brought into df_bike. (Hint: the features at the bottom would have been the last items merged and the features near the top would have been merged first.)
- Make sure to delete any duplicate columns and rename to their original name. For example, you will have two month columns, so it adds a x to one column and y to the other, [‘month_x’] and [‘month_y’]. Delete one and change the other back to [‘month’].

Helpful Code:

```
df_bike = df_bike.drop('month_y', axis = 1)
df_bike = df_bike.rename(columns = { 'month_x' : 'month' } )
```

Example process for merging the DataFrames:

- Start with df_trip and merged it with df_station based on the station ID. (It is ***start_station_id*** in df_trip and ***id*** in df_station.) Name the new DataFrame df_bike.
- Merge df_bike with df_weather based on ***date*** AND ***zip_code***.
- Merge df_bike with df_status_day based on date AND station_id.
  *Note: **station_id** is named **start_station_id** in df_bike.*



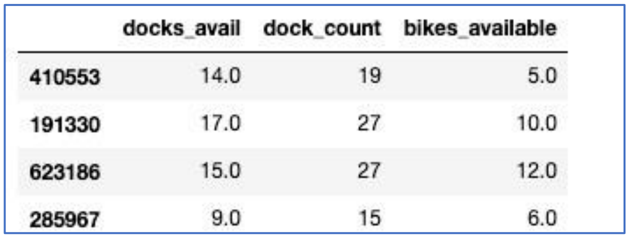
```
In [141]: df_bike.info()

          <class 'pandas.core.frame.DataFrame'>
          Int64Index: 669957 entries, 0 to 669956
          Data columns (total 32 columns):
          start_date              669957 non-null datetime64[ns]
          start_station_id        669957 non-null int64
          end_date                669957 non-null datetime64[ns]
          end_station_id          669957 non-null int64
          bike_id                 669957 non-null int64
          subscription_type       669957 non-null object
          trip_time               669957 non-null timedelta64[ns]
          trip_time_m             669957 non-null int64
          date                    669957 non-null datetime64[ns]
          day                     669957 non-null int64
          month                   669957 non-null int64
          year                    669957 non-null int64
          day_of_week             669957 non-null object
          id                      669957 non-null int64
          zip_code                669957 non-null int64
          name                    669957 non-null object
          city                    669957 non-null object
          dock_count              669957 non-null int64
          max_temperature_f       669891 non-null float64
          mean_temperature_f      669891 non-null float64
          min_temperature_f       669891 non-null float64
          mean_dew_point_f        669604 non-null float64
          mean_humidity           669604 non-null float64
          max_wind_Speed_mph      669943 non-null float64
          mean_wind_speed_mph     669943 non-null float64
          max_gust_speed_mph      649694 non-null float64
          precipitation_inches    669943 non-null object
          cloud_cover             669943 non-null float64
          events                  123640 non-null object
          wind_dir_degrees        669943 non-null float64
          station_id              669957 non-null int64
          bikes_available         669957 non-null float64
          dtypes: datetime64[ns](3), float64(11), int64(11), object(6), timedelta64[ns](1)
          memory usage: 168.7+ MB
```

9. Fill in NaN values

- For [‘events’], fill in all of the NaN values with the word ‘None’.
- For [‘precipitation_inches'], replace all ‘T’ values with 0.001.
  df_bike['precipitation_inches'] = df_bike['precipitation_inches'].replace('T', 0.001)
- For [‘max_temperature_f’], [‘mean_temperature_f‘], and [‘min_temperature_f’], fill in the NaN values with the mean( ).

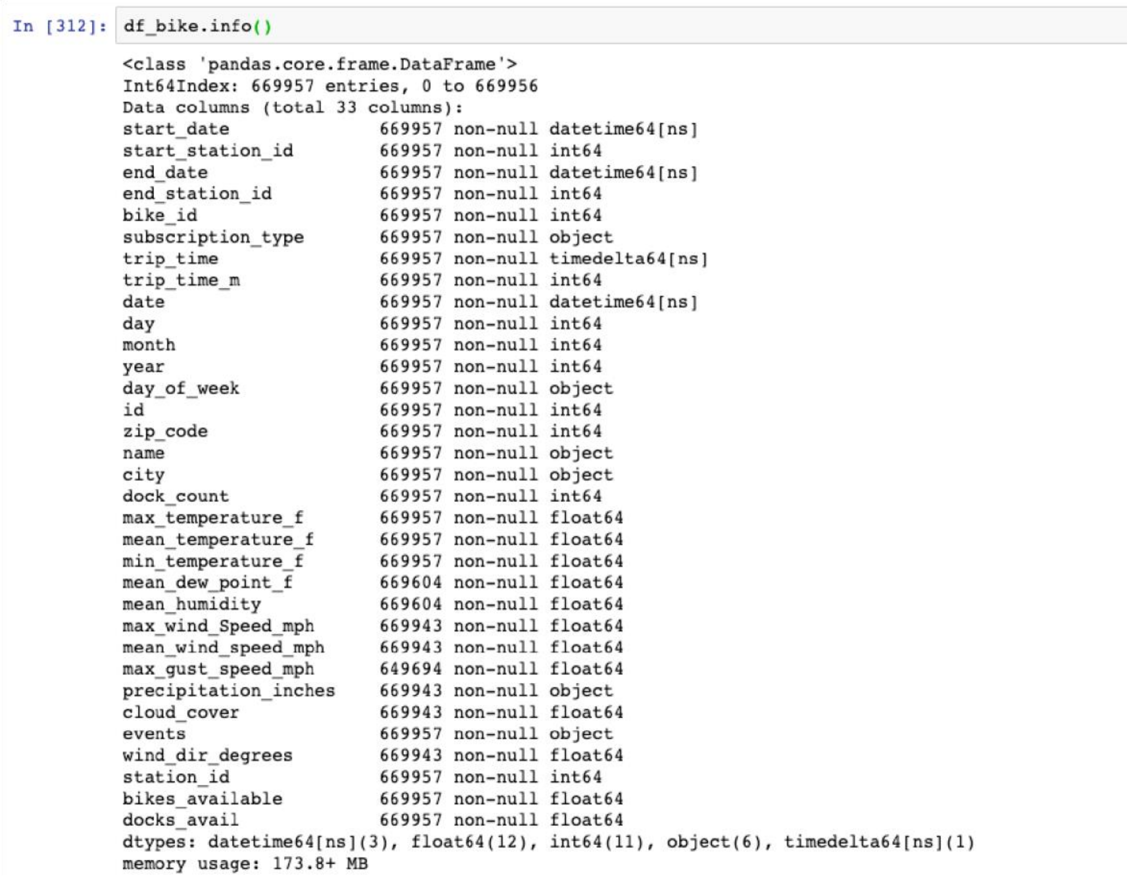
10. Create a new column

- Create a new column named [‘docks_avail’]. This column will take the [‘dock_count’] and subtract the [‘bikes_available’] away to indicate the number of docks that remain empty.



	docks_avail	dock_count	bikes_available
410553	14.0	19	5.0
191330	17.0	27	10.0
623186	15.0	27	12.0
285967	9.0	15	6.0

NOTE: Dataframe columns do not need to be in the exact order, and it may be okay if your number of records do not match.



```
In [312]: df_bike.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 669957 entries, 0 to 669956
Data columns (total 33 columns):
start_date            669957 non-null datetime64[ns]
start_station_id      669957 non-null int64
end_date              669957 non-null datetime64[ns]
end_station_id        669957 non-null int64
bike_id               669957 non-null int64
subscription_type     669957 non-null object
trip_time             669957 non-null timedelta64[ns]
trip_time_m           669957 non-null int64
date                  669957 non-null datetime64[ns]
day                   669957 non-null int64
month                 669957 non-null int64
year                  669957 non-null int64
day_of_week           669957 non-null object
id                    669957 non-null int64
zip_code              669957 non-null int64
name                  669957 non-null object
city                  669957 non-null object
dock_count            669957 non-null int64
max_temperature_f     669957 non-null float64
mean_temperature_f    669957 non-null float64
min_temperature_f     669957 non-null float64
mean_dew_point_f      669604 non-null float64
mean_humidity         669604 non-null float64
max_wind_Speed_mph    669943 non-null float64
mean_wind_speed_mph   669943 non-null float64
max_gust_speed_mph    649694 non-null float64
precipitation_inches  669943 non-null object
cloud_cover           669943 non-null float64
events                669957 non-null object
wind_dir_degrees      669943 non-null float64
station_id            669957 non-null int64
bikes_available       669957 non-null float64
docks_avail           669957 non-null float64
dtypes: datetime64[ns](3), float64(12), int64(11), object(6), timedelta64[ns](1)
memory usage: 173.8+ MB
```

11. Create your own visual

- Based on the df_bike Dataframe, create a visual(s) that show some complexity. It can be on anything that you think makes sense. Please avoid visuals that do not have any expect meaning.

Submission

Save your file as HW02_LastNameFirstInitial.ipynb and turn it in to D2L per the Dropbox instructions. (file only, no data)