

# HW 0 (Practice Mode)

Canvas assignment [link](#) for submission and due date.

Objective:

- familiarize with the use of basic functions in python packages
- familiarize with the use of python notebook: compile all cells to show results

Assessment:

- successful run of the Markdown and Python notebook cells.
- successful submission on git [link](#) by the due date.

Keep the following in mind for **all** notebooks you develop:

1. Structure your notebook. Use headings with meaningful levels in Markdown cells, and explain the questions each piece of code is to answer or the reason it is there.
2. Make sure your notebook can always be rerun from top to bottom.
3. **DO NOT** erase notebook cells provided.

## Setup

This section loads the relevant Python modules and does any configuration needed for the notebook to work.

Lets import python packages we will use in this homework:

- [numpy](#) - scientific computing package
- [pandas](#) - python data analysis package
- [seaborn](#) - statistical data visualization package

```
In [ ]: import numpy as np
import pandas as pd
import seaborn as sns
```

## Introduction to Pandas

In this chapter you will use pandas commands

**Q1:** Read the data file using Pandas. **Note** When we run your experiment to test for correctness, we assume that the day.csv is in the ../data/ folder relative to your HW1.ipynb.

1. **Download the Capital Bike Share data set** from <https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset>. Click 'Data Folder', download

the zip file, and extract the `day.csv` file.

- get used to downloading data files and saving them to correct hierarchy.
- big part of project and source versioning practice

2. Read the Data File: use Pandas `read_csv[]` function to read the file into `bikes` dataframe.

- Our data file does not have column headers, so we need to specify the names.

**A1** Replace the ? mark with your answer in the python cell below.

```
In [ ]: bikes = pd.read_csv("day.csv")
```

**Q2:** Use `head` to show the first few rows of the table:

- brief preview is a safety check you are exploring the correct data frame

**A2** Replace the ? mark with your answer

```
In [ ]: bikes.head()
```

```
Out[ ]:
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270

**Q3:** Use `info` to show a description of the columns, along with the shape and memory use of the data frame:

- `.info()` or `.head()` can be called in the same cell as data load
- we separate them out in this notebook so that we can discuss them in the markdown cells, but we can combine them in the future.

**A3** Replace the ? mark with your answer in the python cell below

```
In [ ]: bikes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   instant         731 non-null    int64
1   dteday          731 non-null    object
2   season          731 non-null    int64
3   yr              731 non-null    int64
4   mnth            731 non-null    int64
5   holiday         731 non-null    int64
6   weekday         731 non-null    int64
7   workingday      731 non-null    int64
8   weathersit       731 non-null    int64
9   temp            731 non-null    float64
10  atemp           731 non-null    float64
11  hum             731 non-null    float64
12  windspeed       731 non-null    float64
13  casual          731 non-null    int64
14  registered      731 non-null    int64
15  cnt             731 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

**Q4:** Pandas provide a very useful function for exploring statistical properties of dataframe, and allow us to see data composition for numerical columns. Use pandas build-in function and show statistical information for columns.

**A4:** Replace the ? mark with your answer in the python cell below

## Error

Do you mean the describe() function?

In [ ]: `bikes.describe()`

Out[ ]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathers
<b>count</b>	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
<b>mean</b>	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264	0.683995	1.39534
<b>std</b>	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787	0.465233	0.54489
<b>min</b>	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.00000
<b>25%</b>	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.00000
<b>50%</b>	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000	1.000000	1.00000
<b>75%</b>	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.00000
<b>max</b>	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.00000

**Q5:** Use pandas functions for filtering the dataframe rows based on some column values. Find out number of rows where the value of the **temp** column is more than the average value of

**temp** for the dataset.

*Steps:*

1. Find out the mean value of the temp column
2. Filter the rows where temp is greater than the mean value
3. Get the number of rows in the filtered dataframe.

**A5:** Replace the ? mark with your answer in python cell below

```
In [ ]: mean_value = bikes['temp'].mean()

print(f'Mean: {mean_value}')

filtered_dataframe = bikes[bikes['temp'] > mean_value]
row_count = filtered_dataframe.count()

print(row_count)
```

Mean: 0.495384788508892

```
instant      367
dteday       367
season       367
yr           367
mnth         367
holiday      367
weekday      367
workingday   367
weathersit    367
temp         367
atemp        367
hum          367
windspeed    367
casual        367
registered   367
cnt          367
dtype: int64
```

## Numpy

We now use Numpy for doing some mathematical calculation on the dataset.

**Q6:** Now, use numpy for working with below tasks

1. At first convert the dataframe into a numpy array
2. Print the shape of the numpy n-dimensional array
3. select and print rows from 100 to 105

**A6:** Replace the ? mark with your answer

```
In [ ]: num_array = bikes.values # conver the dataframe
print(num_array.shape) #print the shape
print(num_array[100:105]) # print rows from 100 to 105
```

```
(731, 16)
[[101 '2011-04-11' 2 0 4 0 1 1 2 0.595652 0.565217 0.716956 0.324474 855
 2493 3348]
 [102 '2011-04-12' 2 0 4 0 2 1 2 0.5025 0.493054 0.739167 0.274879 257
 1777 2034]
 [103 '2011-04-13' 2 0 4 0 3 1 2 0.4125 0.417283 0.819167 0.250617 209
 1953 2162]
 [104 '2011-04-14' 2 0 4 0 4 1 1 0.4675 0.462742 0.540417 0.1107 529 2738
 3267]
 [105 '2011-04-15' 2 0 4 1 5 0 1 0.446667 0.441913 0.67125 0.226375 642
 2484 3126]]
```

**Q7:** Lets put it all together

1. Create a new numpy array selecting column number 10 - 13.
2. Sort the numpy array in ascending order based on the 2nd column of our new numpy array.
  - Treadway - The 2nd column represents the date, which is already ordered.
3. Print first 5 rows of the sorted numpy array
  - Treadway - The new numpy array we just created only has 4 elements.

**A7:** Replace the ? mark with your answer in the cell below

```
In [ ]: new_array = num_array[10:14]
sorted_array = new_array[new_array[:, 1].argsort()]
print(sorted_array[0:5])

[[11 '2011-01-11' 1 0 1 0 2 1 2 0.169091 0.191464 0.686364 0.122132 43
 1220 1263]
 [12 '2011-01-12' 1 0 1 0 3 1 1 0.172727 0.160473 0.599545 0.304627 25
 1137 1162]
 [13 '2011-01-13' 1 0 1 0 4 1 1 0.165 0.150883 0.470417 0.301 38 1368
 1406]
 [14 '2011-01-14' 1 0 1 0 5 1 1 0.16087 0.188413 0.537826 0.126548 54
 1367 1421]]
```

## Seaborn Plotting 1

Seaborn package is used to plot the data. For every question in this chapter use the **bikes dataframe** for answering the questions.

**Q8:** Make a bar plot showing the mean number of riders (y-axis) per weekday (x-axis) using seaborn `catplot` method.

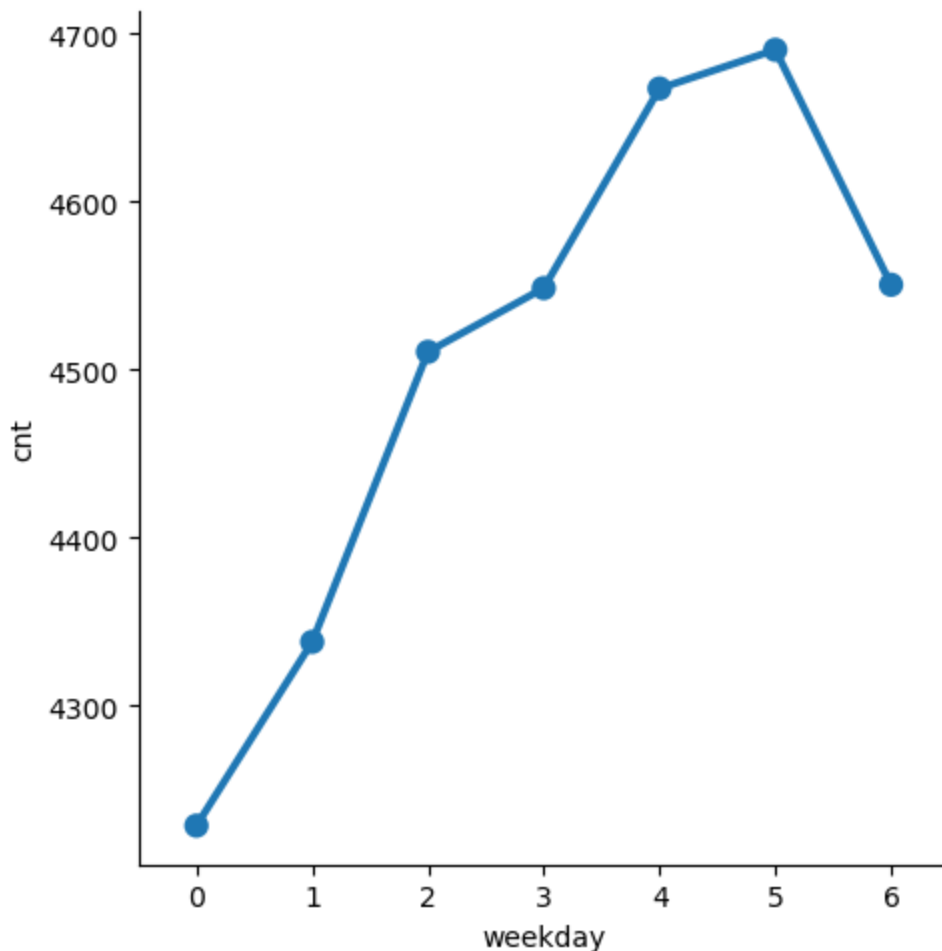
**A8:** Replace the ? mark with your answer

```
In [ ]: # Is "cnt" considered the number of riders?

mean_count = bikes.groupby('weekday')['cnt'].mean().reset_index()

mean_riders = sns.catplot(x='weekday', y='cnt', data=mean_count, kind='point')

c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The
figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



### Analysis

The X-axis labels are not very helpful as day 0 is not clear. This is a question about how the data is *coded*. We'll talk more about data encoding next week. Unfortunately, the data documentation doesn't actually say how weekdays are coded! But we can infer from the data in this case: first data point is January 1, 2011, which was a Saturday, coded as weekday 6; it then resets to 0 for the next day, and starts counting up. Often, we will not be able to infer the data encoding from the data itself - we need to consult the codebook or data set description. We got lucky this time. But looking at the data can help us make sense of the codebook.

*Lesson here is to always look at your data.*

**Q9:** Turn these weekday numbers into a *categorical* variable so Pandas knows how to label them. Hint: use `pandas.Categorical.from_codes()`.

**A9:** Replace the ? mark with your answer in the python cell below

```
In [ ]: codes = pd.CategoricalDtype(['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'])

bikes['day_names'] = pd.Categorical.from_codes(bikes['weekday'], dtype=codes)
bikes.head()
```

Out[ ]:

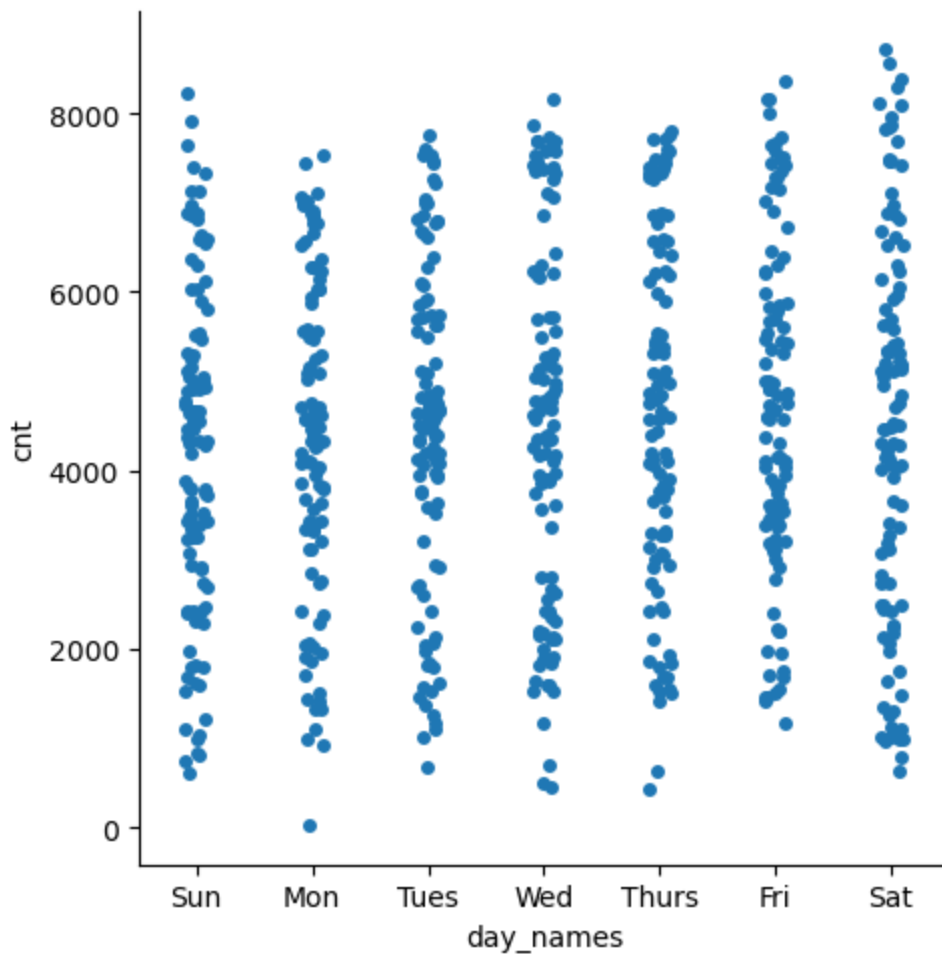
	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270

**A10:** Plot new data using seaborn `catplot`, where data=bikes, x-axis is `day_names` and y-axis is `cnt`

**A10:** Replace the ? mark with your answer in the python cell below

In [ ]: `mean_riders = sns.catplot(x='day_names', y='cnt', data=bikes)`

```
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The
figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



You have now now plotted the average rides per day.

**Note:** When we do not tell `catplot` what to do with multiple points for the same value (in this case the weekday name), it computes the mean and a bootstrapped 95% confidence interval.

## Seaborn Polotting 2: View Data over Time

Lets explore how did rides-per-day change over the course of the data set?

- This kind of data - a sequence of data points associated with times - is called a *time series*.
- This data set gives us an `instant` column that records the data number since the start of the data set

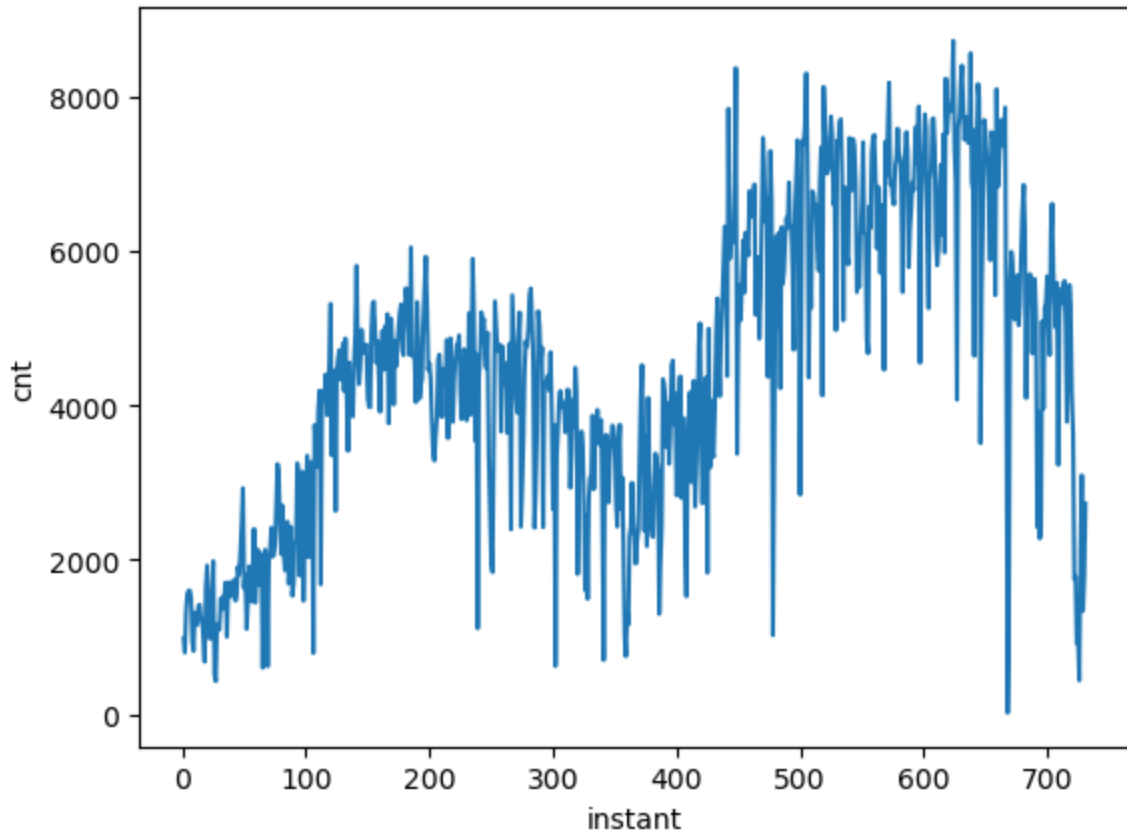
**Q11:** Use `seaborn.lineplot()` where data=bikes, x-axis is `instant` and y-axis is `cnt` value.

**A11:** Replace the ? mark with your answer in the python cell below

```
In [ ]: sns.lineplot(x='instant', y='cnt', data=bikes)
```



```
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
Out[ ]: <Axes: xlabel='instant', ylabel='cnt'>
```



Lets view this graph for actual times on x-axis. The `dteday` column records the date. We can transform `dteday` column to the actual date comlumn using `pandas.to_datetime()` method on the column:

```
In [ ]: bikes['dt'] = pd.to_datetime(bikes['dteday'])
```

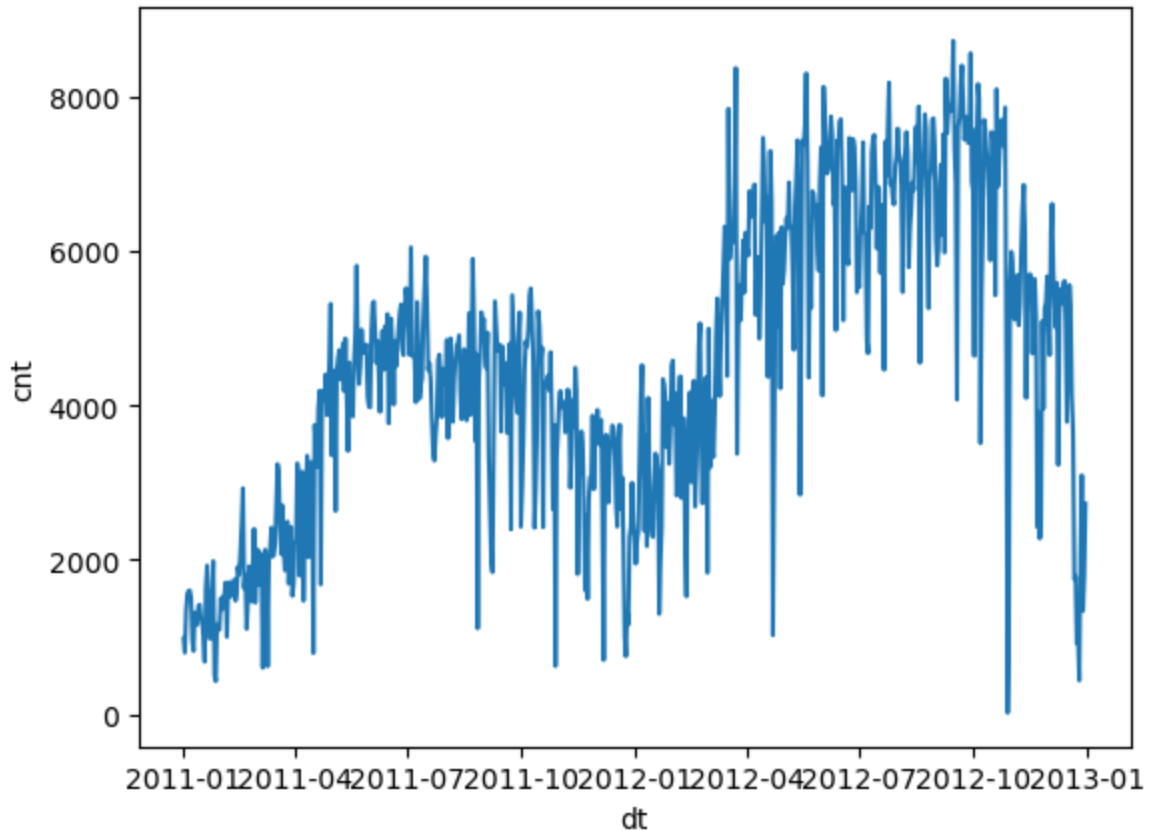
**Q12:** Now create a plot using `seaborn.lineplot()` where data=bikes, x-axis is `dt` and y-axis is `cnt` value.

**A12:** Replace the ? mark with your answer

```
In [ ]: sns.lineplot(x='dt', y='cnt', data=bikes)
```

```
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

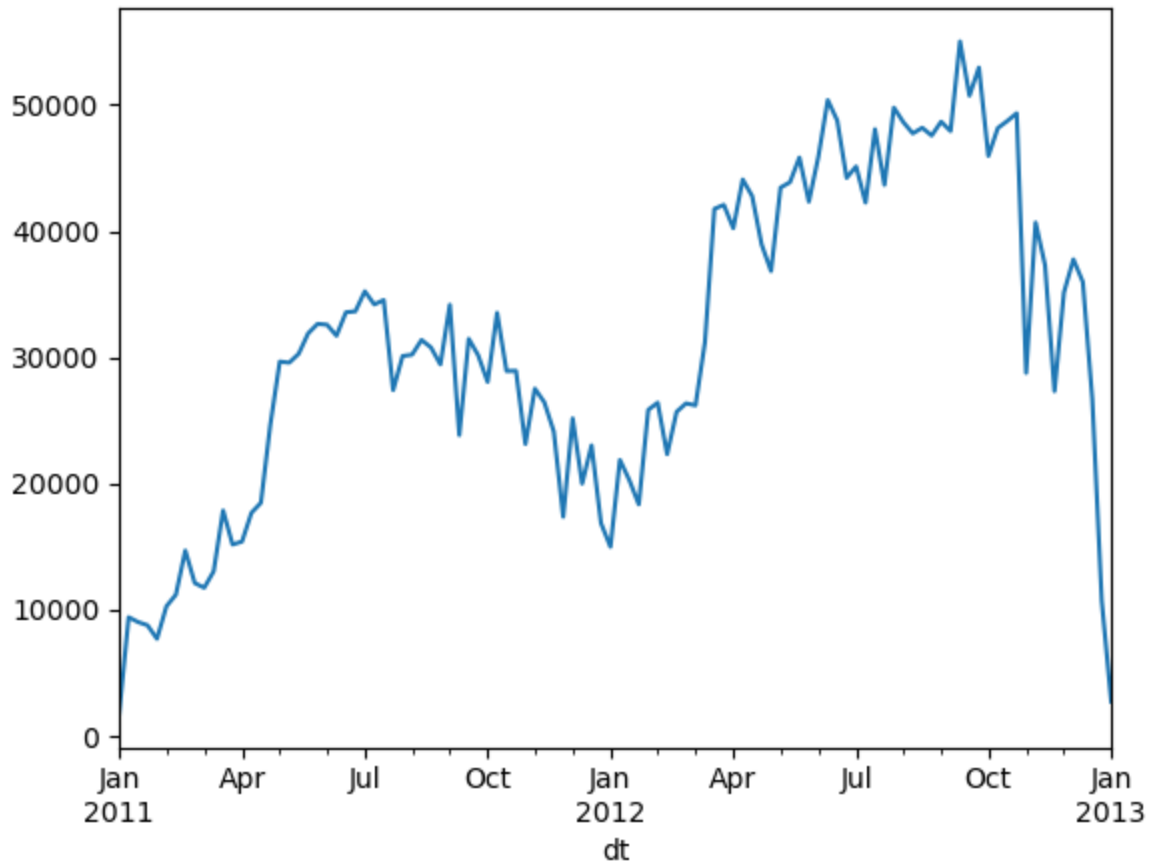
```
Out[ ]: <Axes: xlabel='dt', ylabel='cnt'>
```



Next, plot the *weekly* rides by resampling. Right now, our `bikes` data is indexed by row number in the CSV file. We can change its index to another column, such as our `dt` column with the date, which then lets us do things like resample by week:

```
In [ ]: bikes.set_index('dt')['cnt'].resample('1W').sum().plot()
```

```
Out[ ]: <Axes: xlabel='dt'>
```



What that code did, in one line, is:

1. Set the data frame's index to `dt` (`bikes.set_index('dt')`), returning a new DF
2. Select the count column (`['cnt']`), returning a series
3. Resample the series by week (`.resample('1W')`)
4. Combine measurements within each sample by summing them (`.sum()`)
5. Plotting the results using Pandas' defaults (`.plot()`)

Pandas default plotting functions are useful for quick plots to see what's in a data frame or series. They often are difficult to use to turn in to publication-ready charts.

**Q13:** Save the modified dataframe in csv format using pandas `to_csv()` function. Give the file name as **day\_output**

**A13:** Replace the ? mark with your answer

```
In [ ]: bikes.to_csv('day_output.csv')
```

## Submission Instructions

1. Run all cells in `ECpracticeSubmission.ipynb` and make sure there are no errors
2. Print `ECpracticeSubmission.ipynb` to pdf file
3. Create a Folder HW0 and Upload `ECpracticeSubmission.ipynb`, `ECpracticeSubmission.pdf` and `day_output.csv` files to your git repo allocated for

this course e.g: <https://git.txstate.edu/NetID/netid> before the deadline. Make Sure Instructor and TA has access for the repo.