# Lab 5: Bike Sharing

## Exploratory Data Analysis (EDA) and Visualization

## Introduction

Bike sharing systems are a new generation of traditional bike rentals where the process of signing up, renting and returning is automated. Through these systems, users are able to easily rent a bike from one location and return them to another. We will be analyzing bike sharing data from Washington D.C.

In this assignment, you will perform tasks to clean, visualize, and explore the bike sharing data. You will also investigate open-ended questions. These open-ended questions ask you to think critically about how the plots you have created provide insight into the data.

After completing this assignment, you should be comfortable with:

- reading plaintext delimited data into `pandas`
- wrangling data for analysis
- using EDA to learn about your data
- making informative plots

In [ ]:
```
# Run this cell to set up your notebook.
import seaborn as sns
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import zipfile
from pathlib import Path

# Default plot configurations
%matplotlib inline
plt.rcParams['figure.figsize'] = (16,8)
plt.rcParams['figure.dpi'] = 150
sns.set()
```

```
import warnings
warnings.filterwarnings("ignore")

from IPython.display import display, Latex, Markdown
```

# Loading Bike Sharing Data

The data we are exploring is collected from a bike sharing system in Washington D.C.

The variables in this data frame are defined as:

| Variable | Description |
|---|---|
| instant | record index |
| dteday | date |
| season | 1. spring<br>2. summer<br>3. fall<br>4. winter |
| yr | year (0: 2011, 1:2012) |
| mnth | month ( 1 to 12) |
| hr | hour (0 to 23) |
| holiday | whether day is holiday or not |
| weekday | day of the week |
| workingday | if day is neither weekend nor holiday |
| weathersit | 1. clear or partly cloudy<br>2. mist and clouds<br>3. light snow or rain<br>4. heavy rain or snow |
| temp | normalized temperature in Celsius (divided by 41) |
| atemp | normalized "feels-like" temperature in Celsius (divided by 50) |
| hum | normalized percent humidity (divided by 100) |
| windspeed | normalized wind speed (divided by 67) |
| casual | count of casual users |
| registered | count of registered users |

| | |
|---|---|
| cnt | count of total rental bikes including casual and registered |

## Loading the data

The following code loads the data into a Pandas DataFrame.

In [ ]:
```
# Run this cell to load the data.  No further action is ne
bike = pd.read_csv('data/bikeshare.txt')
bike.head()
```

Out[ ]:

| | instant | dteday | season | yr | mnth | hr | holiday | weekday | workingday |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2011-01-01 | 1 | 0 | 1 | 0 | 0 | 6 | 0 |
| **1** | 2 | 2011-01-01 | 1 | 0 | 1 | 1 | 0 | 6 | 0 |
| **2** | 3 | 2011-01-01 | 1 | 0 | 1 | 2 | 0 | 6 | 0 |
| **3** | 4 | 2011-01-01 | 1 | 0 | 1 | 3 | 0 | 6 | 0 |
| **4** | 5 | 2011-01-01 | 1 | 0 | 1 | 4 | 0 | 6 | 0 |

Below, we show the shape of the file. You should see that the size of the DataFrame matches the number of lines in the file, minus the header row.

In [ ]:
```
bike.shape
```

Out[ ]:
```
(17379, 17)
```

# 0: Examining the Data

Before we start working with the data, let's examine its granularity.

## Question 0

### Question 0A

What is the granularity of the data (i.e. what does each row represent)?

The bike sharing data including environment in Washington D.C. and count of users is recorded every hour of the day.

### Question 0B

For this assignment, we'll be using this data to study bike usage in Washington D.C. Based on the granularity and the variables present in the data, what might some limitations of using this data be? What are two additional data categories/variables that you can collect to address some of these limitations?

Problems:

- No geographic data: While we have overall usage statistics, we don't know where the bikes are being picked up or dropped off. This prevents us from assessing if certain neighborhoods or areas are underserved.
- Lack of Price data: There is no data about how much money should customers pay to rent the bike.

Solutions:

- Add geographic data: Add the variables 'location', indicating where the users rent the sharing bike.
- Add the Price data: Add the variables 'Average price'.

---

# 1: Data Preparation

A few of the variables that are numeric/integer actually encode categorical data. These include `holiday`, `weekday`, `workingday`,

and `weathersit`. In the following problem, we will convert these four variables to strings specifying the categories. In particular, use 3-letter labels (`Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, and `Sat`) for `weekday`. You may simply use `yes` / `no` for `holiday` and `workingday`.

In this exercise we will *mutate* the data frame, **overwriting the corresponding variables in the data frame.** However, our notebook will effectively document this in-place data transformation for future readers. Make sure to leave the underlying datafile `bikeshare.txt` unmodified.

# Question 1

## Question 1a (Decoding `weekday`, `workingday`, and `weathersit`)

Decode the `holiday`, `weekday`, `workingday`, and `weathersit` fields:

1. `holiday`: Convert to `yes` and `no`. Hint: There are fewer holidays...
2. `weekday`: It turns out that Monday is the day with the most holidays. Mutate the `'weekday'` column to use the 3-letter label (`'Sun'`, `'Mon'`, `'Tue'`, `'Wed'`, `'Thu'`, `'Fri'`, and `'Sat'` ...) instead of its current numerical values. Assume `0` corresponds to `Sun`, `1` to `Mon` and so on, in order of the previous sentence.
3. `workingday`: Convert to `yes` and `no`.
4. `weathersit`: You should replace each value with one of `Clear`, `Mist`, `Light`, or `Heavy`. Assume `1` corresponds to `Clear`, `2` corresponds to `Mist`, and so on in order of the previous sentence.

**Note:** If you mutate any of the tables above, then they will not be in the format of their original `.csv` file. As a debugging tip, if you want to revert changes, run the cell that reloads the csv.

**Hint:** One approach is to use the replace method of the pandas DataFrame class. Take a look at the link by clicking on the word `replace` in the previous sentence. We have already included `replace` in the cell below so you can focus on creating the "nested-dictonaries" described in the documentation.

In [ ]:
```
# Modify holiday weekday, workingday, and weathersit here
factor_dict = {
    'holiday': {0: 'no', 1: 'yes'},
    'weekday': {0: 'Sun', 1: 'Mon', 2: 'Tue', 3: 'Wed', 4:
    'workingday': {0: 'no', 1: 'yes'},
    'weathersit': {1: 'Clear', 2: 'Mist', 3: 'Light', 4: '
}
bike.replace(factor_dict, inplace=True)
bike.head()
```

Out[ ]:

| | instant | dteday | season | yr | mnth | hr | holiday | weekday | workingday |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2011-01-01 | 1 | 0 | 1 | 0 | no | Sat | no |
| **1** | 2 | 2011-01-01 | 1 | 0 | 1 | 1 | no | Sat | no |
| **2** | 3 | 2011-01-01 | 1 | 0 | 1 | 2 | no | Sat | no |
| **3** | 4 | 2011-01-01 | 1 | 0 | 1 | 3 | no | Sat | no |
| **4** | 5 | 2011-01-01 | 1 | 0 | 1 | 4 | no | Sat | no |

## Question 1b (Holidays)

How many entries in the data correspond to holidays? Set the variable `num_holidays` to this value.

In [ ]:
```
num_holidays = bike[bike['holiday'] == 'yes'].shape[0]
num_holidays
```

Out[ ]:
```
500
```

## Question 1c (Computing Daily Total Counts)

In the next few questions we will be analyzing the daily number of registered and unregistered users.

Construct a data frame named `daily_counts` indexed by `dteday` with the following columns:

- `casual` : total number of casual riders for each day
- `registered` : total number of registered riders for each day
- `workingday` : whether that day is a working day or not ( `yes` or `no` )

**Hint**: `groupby` and `agg` . For the `agg` method, please check the documentation for examples on applying different aggregations per column. If you use the capability to do different aggregations by column, you can do this task with a single call to `groupby` and `agg` . For the `workingday` column we can take any of the values since we are grouping by the day, thus the value will be the same within each group. Take a look at the `'first'` or `'last'` aggregation functions.

In [ ]:
```
daily_counts = bike.groupby('dteday').agg({
    'casual': 'sum',
    'registered': 'sum',
    'workingday': 'first'
})
daily_counts.head()
```

Out[ ]:

| dteday | casual | registered | workingday |
|---|---|---|---|
| **2011-01-01** | 331 | 654 | no |
| **2011-01-02** | 131 | 670 | no |
| **2011-01-03** | 120 | 1229 | yes |
| **2011-01-04** | 108 | 1454 | yes |
| **2011-01-05** | 82 | 1518 | yes |

# 2: Exploring the Distribution of Riders

Let's begin by comparing the distribution of the daily counts of casual and registered riders. Questions 2-7 require using many visualization methods so for your convenience, we have summarized a few useful ones below.

## Matplotlib and Seaborn Table of Common Functions

`x` and `y` are sequences of values (i.e. arrays, lists, or Series).

| Function | Description |
|---|---|
| `plt.plot(x, y)` | Creates a line plot of `x` against `y` |
| `plt.title(name)` | Adds a title `name` to the current plot |
| `plt.xlabel(name)` | Adds a label `name` to the x-axis |
| `plt.ylabel(name)` | Adds a label `name` to the y-axis |
| `plt.scatter(x, y)` | Creates a scatter plot of `x` against `y` |
| `plt.hist(x, bins=None)` | Creates a histogram of `x`; `bins` can be an integer or a sequence |
| `plt.bar(x, height)` | Creates a bar plot of categories `x` and corresponding heights `height` |
| `sns.histplot(data, x, y, hue, kde)` | Creates a distribution plot; `data` is a DataFrame; `x`, `y` are column names in `data` that specify positions on the x and y axes; `hue` is a column name in `data` that adds subcategories to the plot based on `hue`; `kde` is a boolean that determines whether to overlay a KDE curve |
| `sns.lineplot(data, x, y, hue)` | Creates a line plot |
| `sns.scatterplot(data, x, y, hue, size)` | Creates a scatter plot; `size` is a vector that contains the size of point for each subcategory based on `hue` |
| `sns.kdeplot(x, y)` | Creates a kernel density estimate plot; `x`, `y` are series of data that indicate positions on the `x` and `y` axis |
| `sns.jointplot(x, y,` | Creates a joint plot of 2 variables with KDE plot in the middle and a distribution plot for each |

| | |
|---|---|
| `data, kind)` | variable on the sides; `kind` determines the visualization type for the distribution plot, can be `scatter`, `kde` or `hist` |

**Note**: This list of functions and parameters is **not** exhaustive. You may need to reference and explore more documentation to answer the following questions, but we will help you through that process.

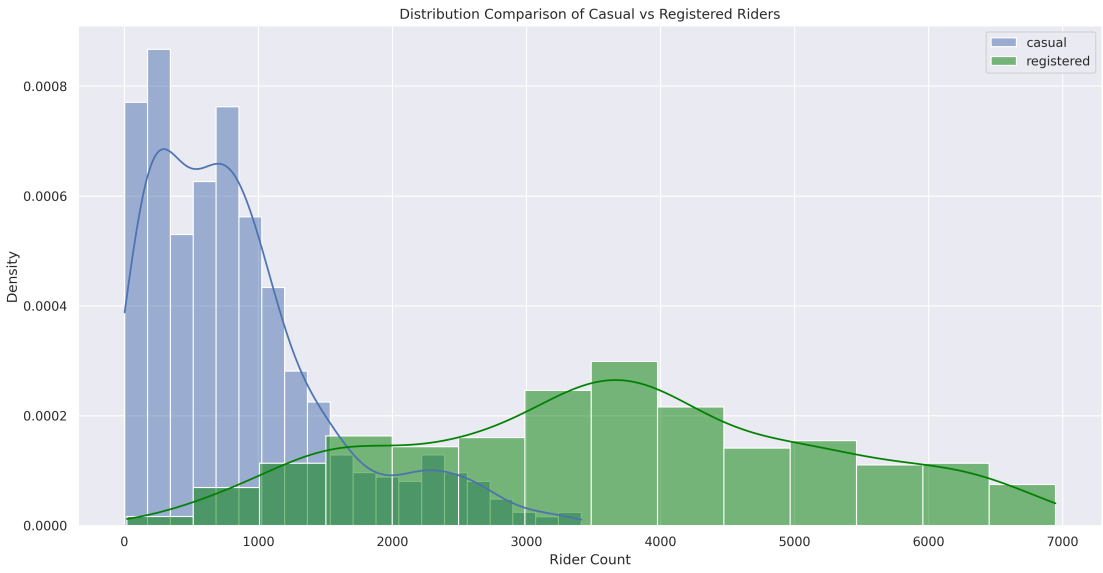
# Question 2

## Question 2a

Use the `sns.histplot` function to create a plot that overlays the distribution of the daily counts of bike users, using blue to represent `casual` riders, and green to represent `registered` riders. The temporal granularity of the records should be daily counts, which you should have after completing question 1c.

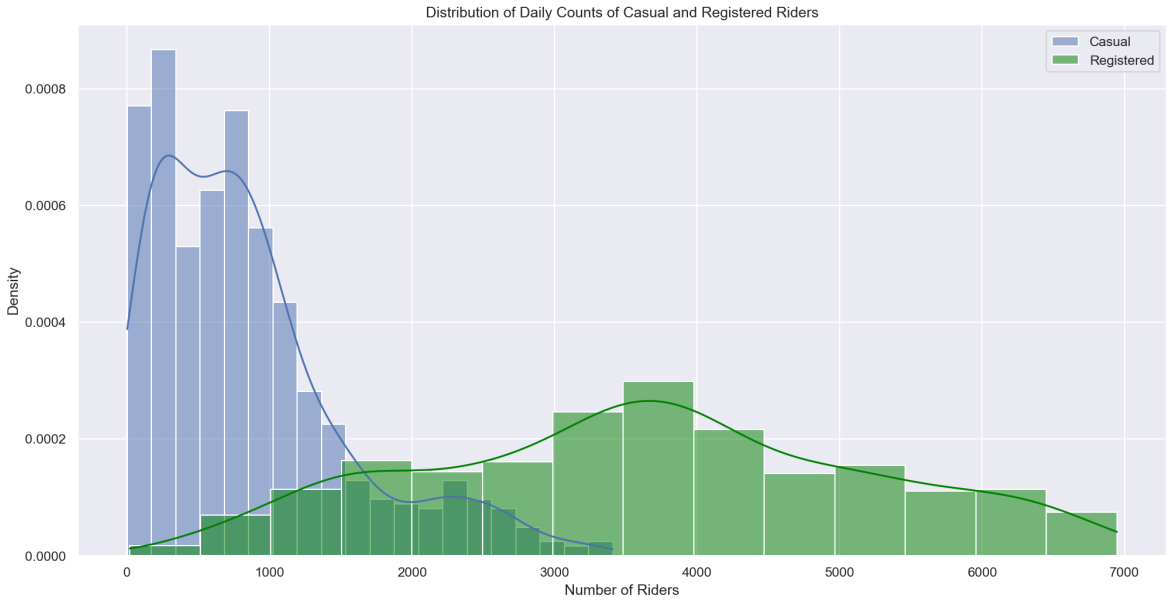
**Hint:** You will need to set the `stat` parameter appropriately to match the desired plot.

Include a legend, xlabel, ylabel, and title. Read the seaborn plotting tutorial if you're not sure how to add these. After creating the plot, look at it and make sure you understand what the plot is actually telling us, e.g on a given day, the most likely number of registered riders we expect is ~4000, but it could be anywhere from nearly 0 to 7000.




In [ ]:
```
sns.histplot(data=daily_counts, x='casual', kde=True, stat
```

```
sns.histplot(data=daily_counts, x='registered', kde=True,

plt.title('Distribution of Daily Counts of Casual and Regi
plt.xlabel('Number of Riders')
plt.ylabel('Density')
plt.legend()
plt.show()
```




## Question 2b

In the cell below, descibe the differences you notice between the density curves for casual and registered riders. Consider concepts such as modes, symmetry, skewness, tails, gaps and outliers. Include a comment on the spread of the distributions.

The registered riders' distribution is centered around a higher value (about 3500-4000) compared to casual riders (centered around 500-1000). This indicates that on average, there are more registered riders per day than casual riders.

Spread:

- The registered riders' distribution has a wider spread, ranging from about 0 to 7000, while the casual riders' distribution is narrower, mostly between 0 and 2500. This suggests more variability in the daily count of registered riders.

Shape:

- The registered riders' distribution is more symmetrical, closer to a normal distribution, but with a slight right skew.

## Question 2c

The density plots do not show us how the counts for registered and casual riders vary together. Use `sns.lmplot` to make a scatter plot to investigate the relationship between casual and registered counts. This time, let's use the `bike` DataFrame to plot hourly counts instead of daily counts.

The `lmplot` function will also try to draw a linear regression line. Color the points in the scatterplot according to whether or not the day is a working day (your colors do not have to match ours exactly, but they should be different based on whether the day is a working day).

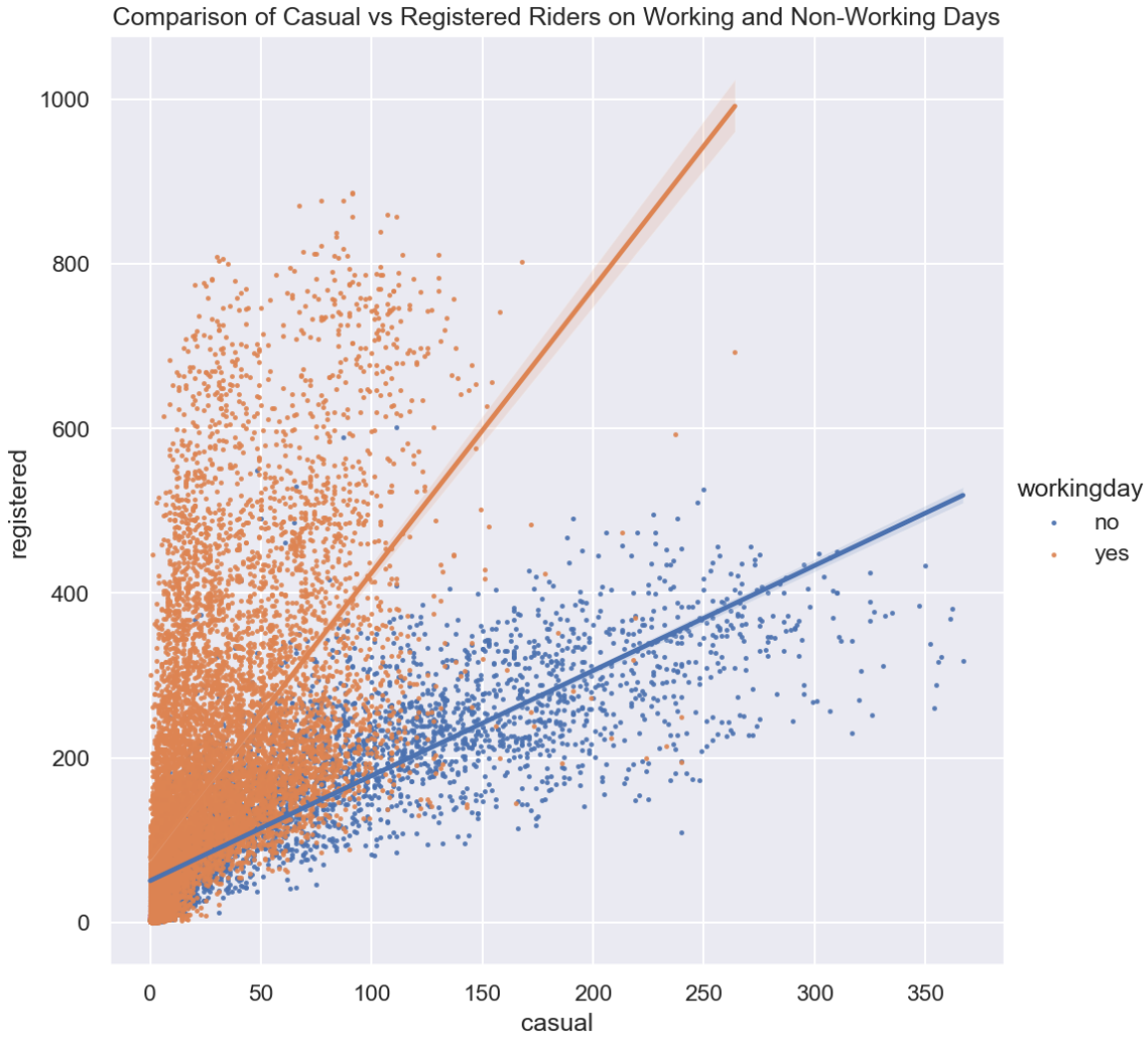
There are many points in the scatter plot, so make them small to help reduce overplotting. Also make sure to set `fit_reg=True` to generate the linear regression line. You can set the `height` parameter if you want to adjust the size of the `lmplot`.


**Hints:**

- Checkout this helpful tutorial on `lmplot`.
- You will need to set `x`, `y`, and `hue` and the `scatter_kws` in the `sns.lmplot` call.
- You will need to call `plt.title` to add a title for the graph.

In [ ]:
```
# Make the font size a bit bigger
sns.set(font_scale=1)
sns.lmplot(data=bike, x='casual', y='registered', hue='wor

plt.title('Comparison of Casual vs Registered Riders on Wo
```

Out[ ]:
```
Text(0.5, 1.0, 'Comparison of Casual vs Registered Riders
on Working and Non-Working Days')
```




## Question 2d

What does this scatterplot seem to reveal about the relationship (if any) between casual and registered riders and whether or not the day is on the weekend? What effect does overplotting have on your ability to describe this relationship?

Relation:

- There is a positive relation between the number of casual riders and registered riders. As one increases, the other tends to increase as well.
- And non-workingdays tends to have more casual users than the registered users.

Overplotting:

- It's challenging to identify true outliers as they may be obscured by or appear to blend with the mass of overlapping points.

- In areas where many points overlap, it's difficult to determine the true density of data points. This makes it hard to assess the strength of the relationship in different regions of the plot.

# 3: Visualization

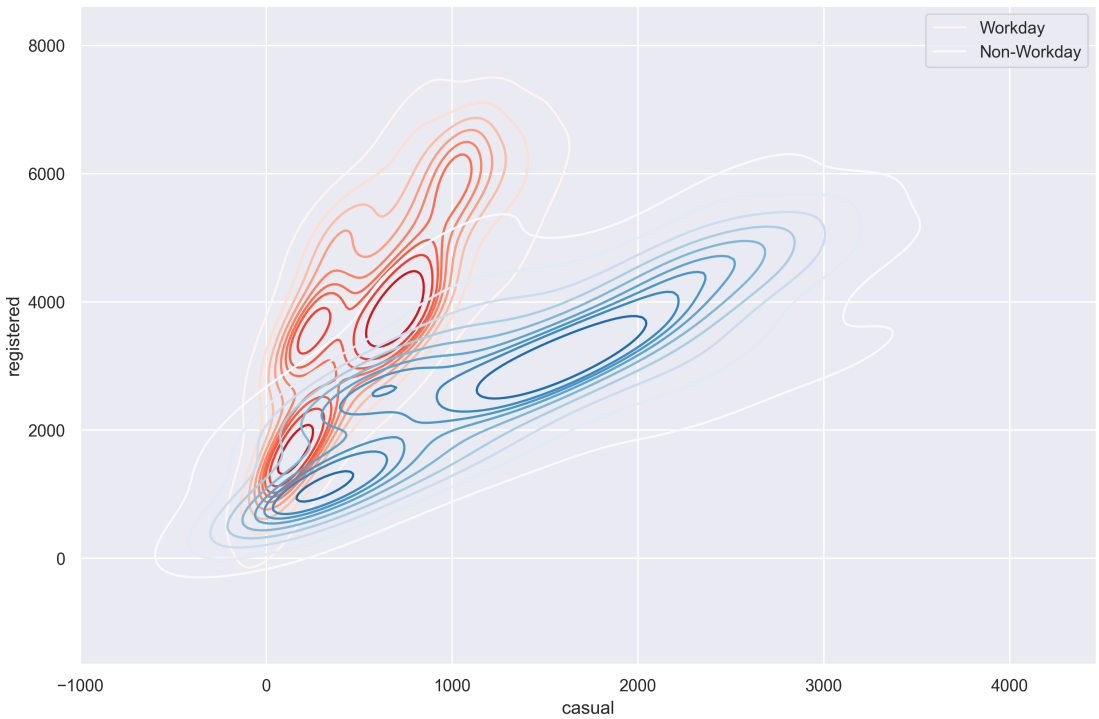
## Question 3

### Question 3a Bivariate Kernel Density Plot

To address overplotting, let's try visualizing the data with another technique, the bivariate kernel density estimate.

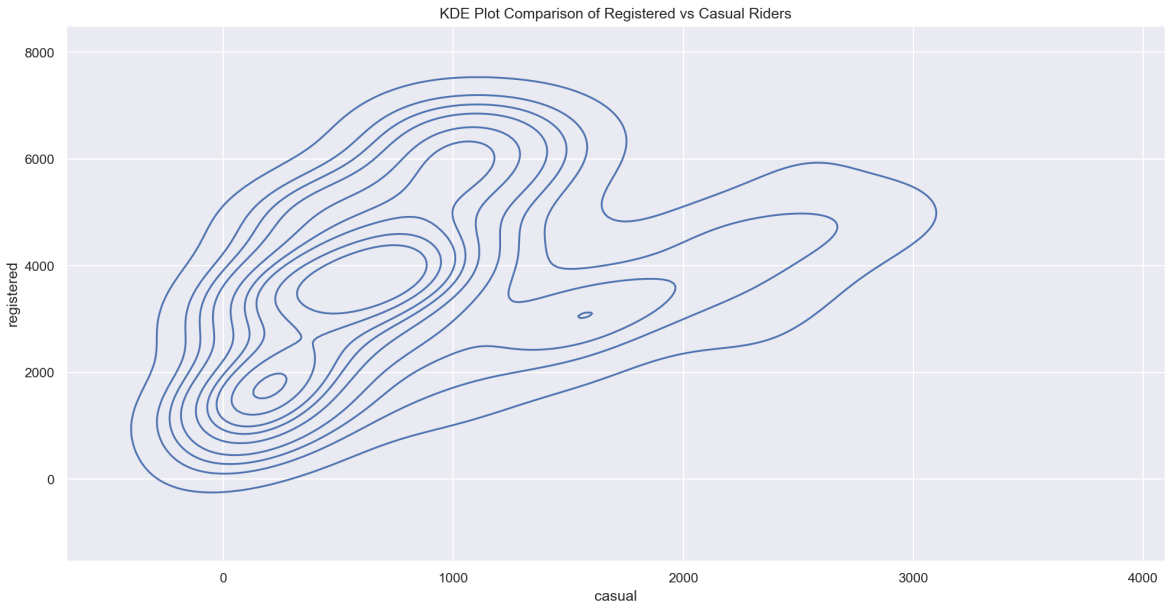
You will want to read up on the documentation for `sns.kdeplot`, which can be found here.

The result we wish to achieve should be a plot that looks like this:




A basic kde plot of all the data is quite easy to generate. However, this plot includes both weekend and weekday data, which isn't what we want (see example figure above).

In [ ]:
```
sns.kdeplot(x=daily_counts['casual'], y=daily_counts['regi
plt.title('KDE Plot Comparison of Registered vs Casual Rid
```




Generating the plot with weekend and weekday separated can be complicated so we will provide a walkthrough below, feel free to use whatever method you wish if you do not want to follow the walkthrough.

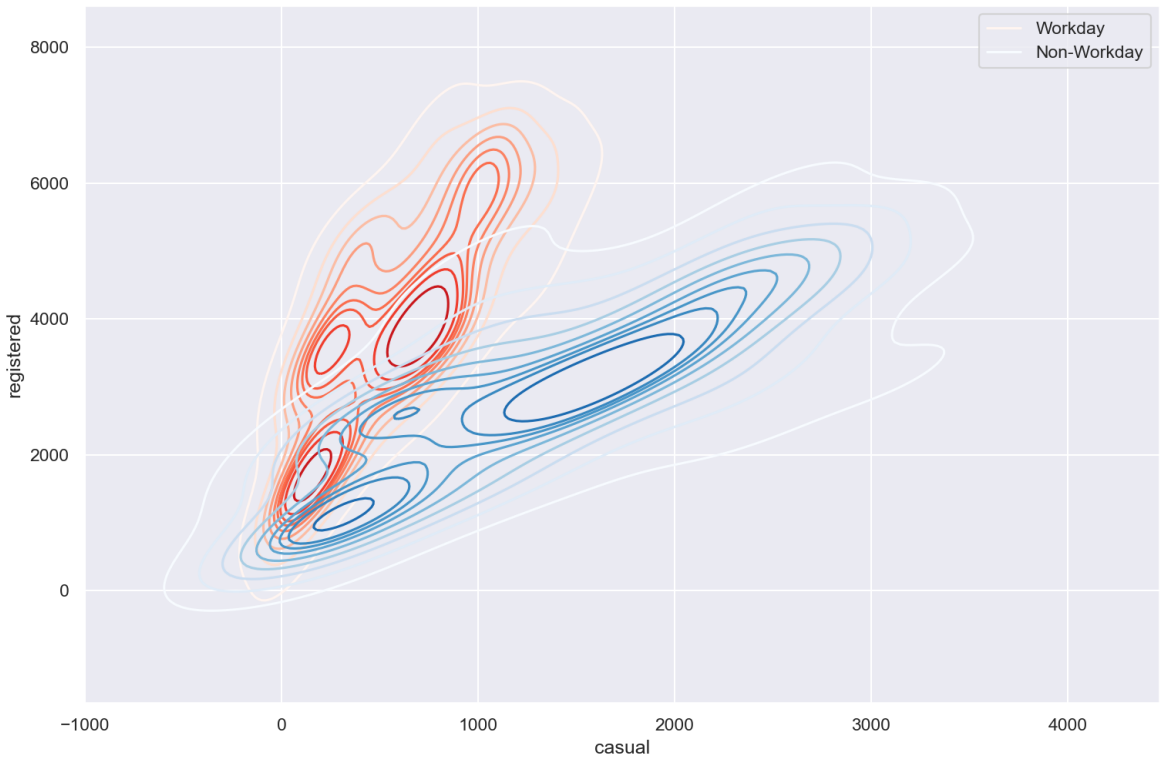
**Hints:**

- You can use `loc` with a boolean array and column names at the same time
- You will need to call kdeplot twice, each time drawing different data from the `daily_counts` table.
- Check out this guide to see an example of how to create a legend. In particular, look at how the example in the guide makes use of the `label` argument in the call to `plt.plot()` and what the `plt.legend()` call does. This is a good exercise to learn how to use examples to get the look you want.
- You will want to set the `cmap` parameter of `kdeplot` to `"Reds"` and `"Blues"` (or whatever two contrasting colors you'd like), and also set the `label` parameter to address which type of day you want to plot. You are required for this question to use two sets of contrasting colors for your plots.

After you get your plot working, experiment by setting `shade=True` in `kdeplot` to see the difference between the shaded and unshaded version. Please submit your work with `shade=False`.

In [ ]:
```
# Set the figure size for the plot
plt.figure(figsize=(12,8))

# Set 'is_workingday' to a boolean array that is true for
is_workingday = daily_counts['workingday'] == 'yes'

# Bivariate KDEs require two data inputs.
# In this case, we will need the daily counts for casual a
# Hint: consider using the .loc method here.
casual_workday = daily_counts.loc[is_workingday, 'casual']
registered_workday = daily_counts.loc[is_workingday, 'regi

# Use sns.kdeplot on the two variables above to plot the b
sns.kdeplot(x=casual_workday, y=registered_workday, cmap='

not_workingday = ~is_workingday
# Repeat the same steps above but for rows corresponding t
# Hint: Again, consider using the .loc method here.
casual_non_workday = daily_counts.loc[not_workingday, 'cas
registered_non_workday = daily_counts.loc[not_workingday,

# Use sns.kdeplot on the two variables above to plot the b
sns.kdeplot(x=casual_non_workday, y=registered_non_workday

plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1936944d5c0>
```




## Question 3bi

In your own words, describe what the lines and the color shades of the lines signify about the data.

- The lines in the plot represent contours of equal density in the bivariate distribution of casual and registered riders.
- The color shades of the lines indicate the relative density of points in that area - darker shades represent higher density.

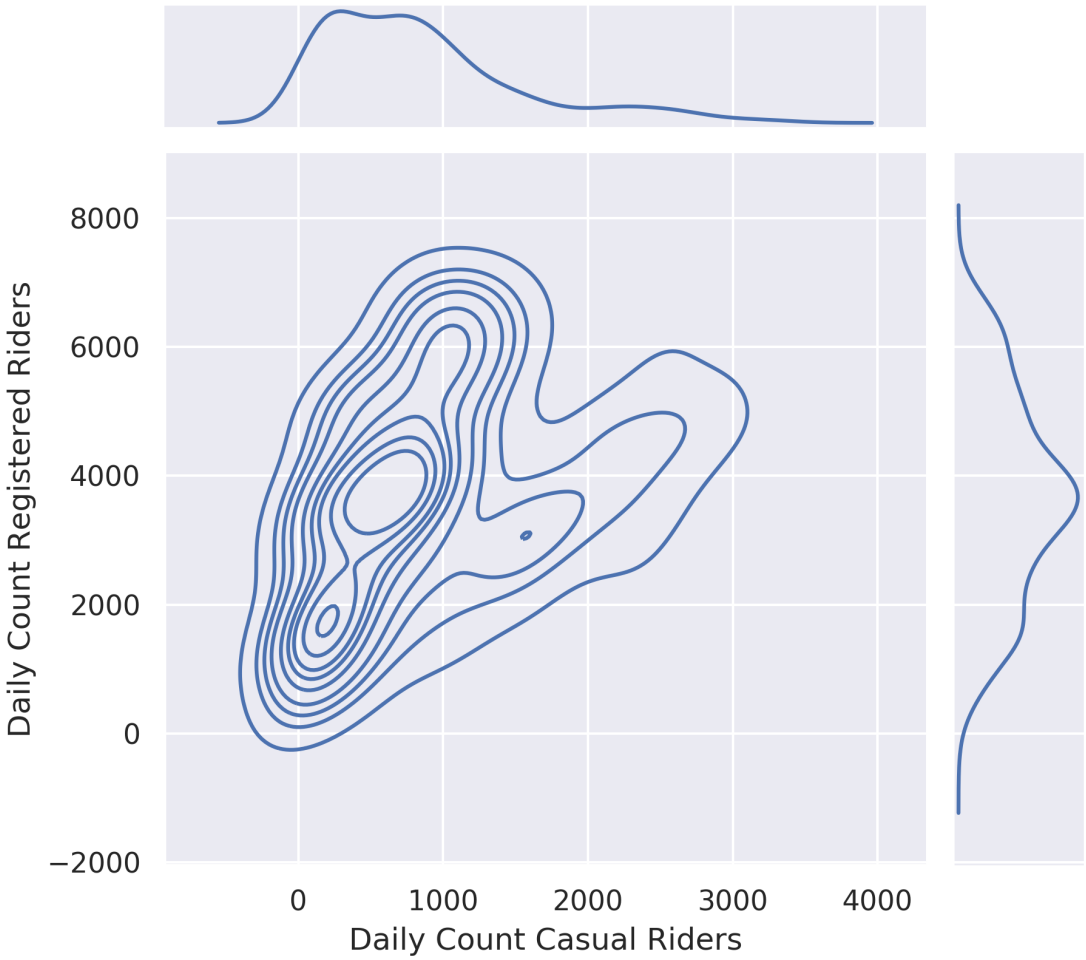
### Question 3bii

What additional details can you identify from this contour plot that were difficult to determine from the scatter plot?

The contour plot de-emphasizes outliers and focuses on the main distribution, giving a clearer picture of typical behavior

## 4: Joint Plot

As an alternative approach to visualizing the data, construct the following set of three plots where the main plot shows the contours of the kernel density estimate of daily counts for registered and casual riders plotted together, and the two "margin" plots (at the top and right of the figure) provide the univariate kernel density estimate of each of these variables. Note that this plot makes it harder see the linear relationships between casual and registered for the two different conditions (weekday vs. weekend).



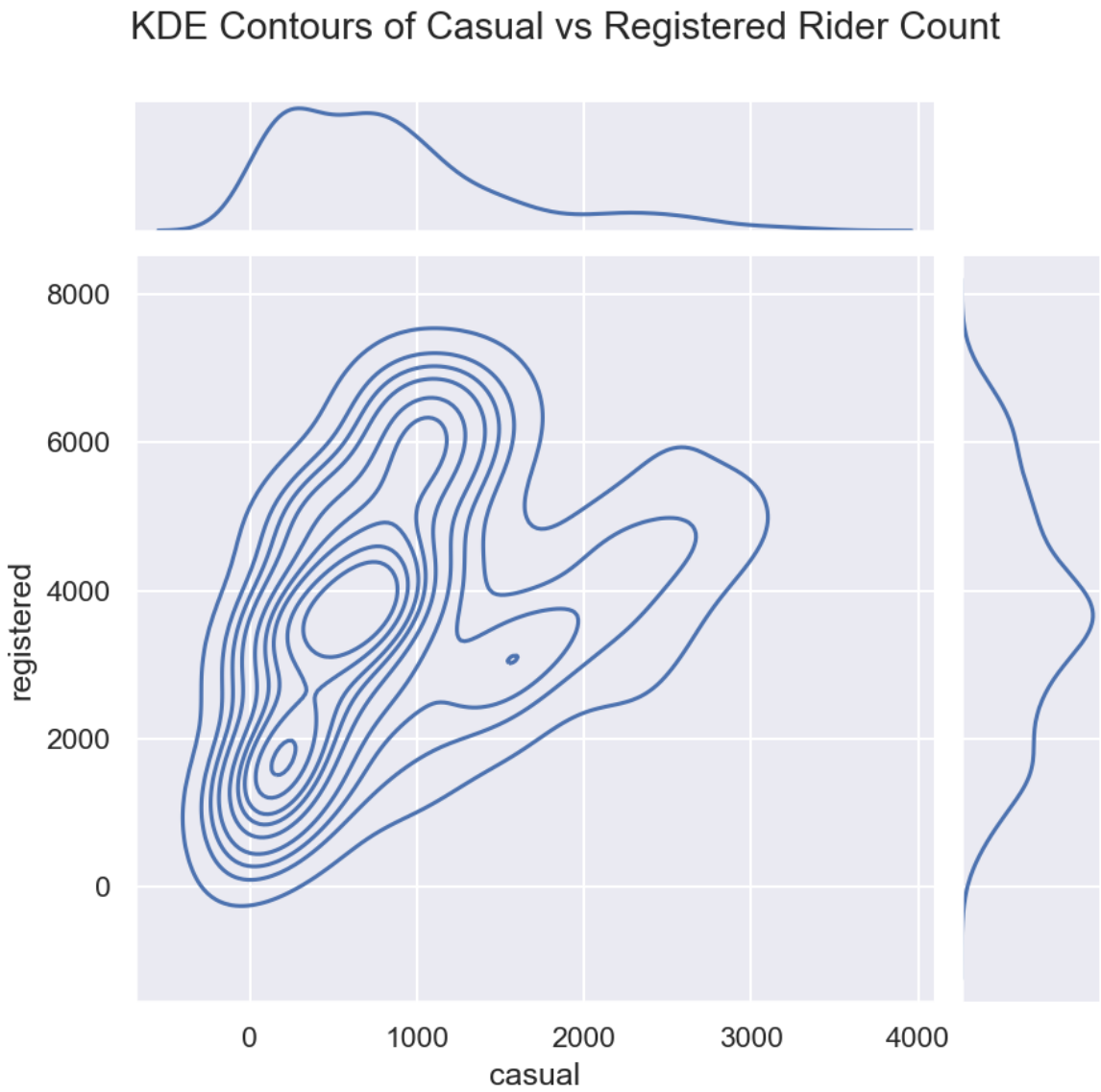

**Hints**:

- The seaborn plotting tutorial has examples that may be helpful.
- Take a look at `sns.jointplot` and its `kind` parameter.
- `set_axis_labels` can be used to rename axes on the contour plot.

**Note**:

- At the end of the cell, we called `plt.suptitle` to set a custom location for the title.
- We also called `plt.subplots_adjust(top=0.9)` in case your title overlaps with your plot.

In [ ]:
```
sns.jointplot(data=daily_counts, x='casual', y='registered
plt.suptitle("KDE Contours of Casual vs Registered Rider C
plt.subplots_adjust(top=0.9);
```



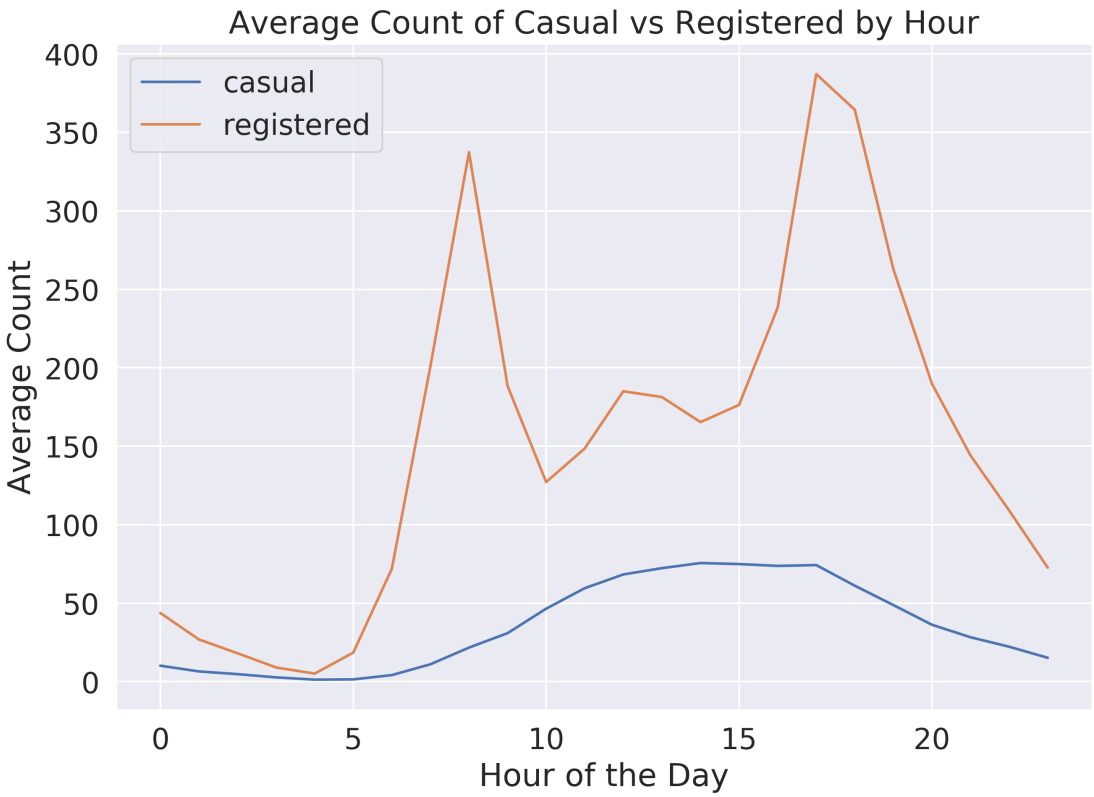

# 5: Understanding Daily Patterns

## Question 5

### Question 5a

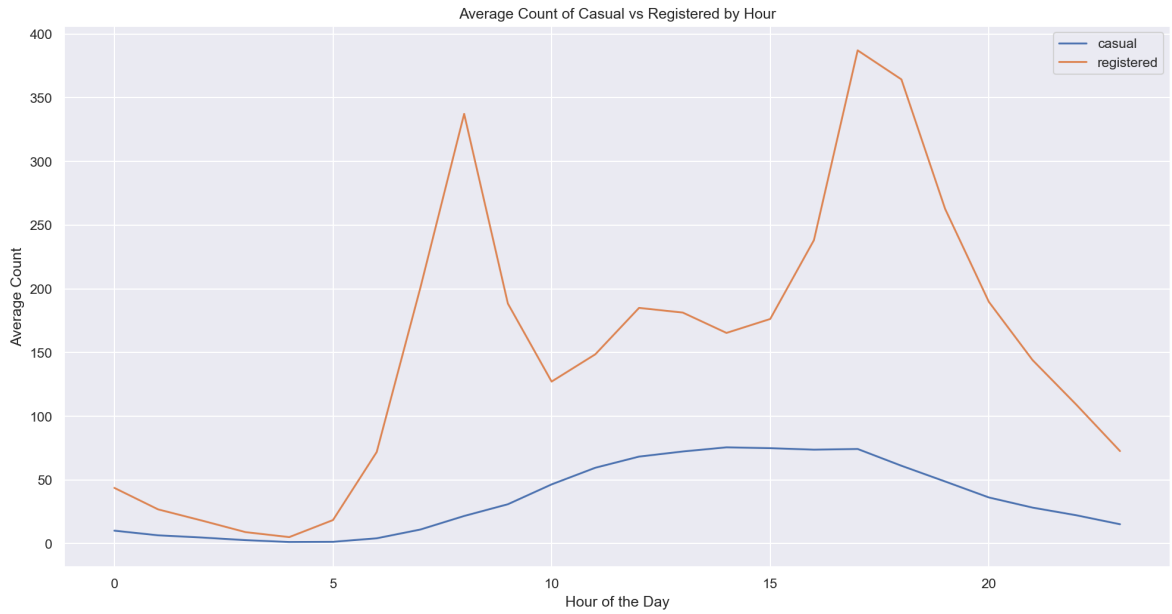
Let's examine the behavior of riders by plotting the average number of riders for each hour of the day over the **entire dataset**, stratified by rider type.

Your plot should look like the plot below. While we don't expect your plot's colors to match ours exactly, your plot should have different colored lines for different kinds of riders.




In [ ]:
```
bike_hourly = bike.groupby('hr').agg({
    'casual': 'mean',
    'registered': 'mean'
})
plt.plot(bike_hourly.index, bike_hourly['casual'], label='
plt.plot(bike_hourly.index, bike_hourly['registered'], lab
plt.xlabel('Hour of the Day')
plt.ylabel('Average Count')
plt.title('Average Count of Casual vs Registered by Hour')
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x193641b9da0>
```




## Question 5b

What can you observe from the plot? Hypothesize about the meaning of the peaks in the registered riders' distribution.

Observation:

- The registered users are much more than the casual users especially during the day time.
- Both registered users and casual users use sharing bikes more often during the day time.

Hypothesize:

- There are two peaks in the registered riders' distribution that one is around 9 a.m. and another one is around 5 p.m.. These are times that workers go to work and go back home thus they need sharing bikes as transportation.
- Another hypothesize is that many registered riders are workers who apply sharing bikes as transportation.

---

# 6: Exploring Ride Sharing and Weather

Now let's examine how the weather is affecting rider's behavior. First let's look at how the proportion of casual riders changes as weather changes.

## Question 6

### Question 6a

Create a new column `prop_casual` in the `bike` DataFrame representing the proportion of casual riders out of all riders for each record.

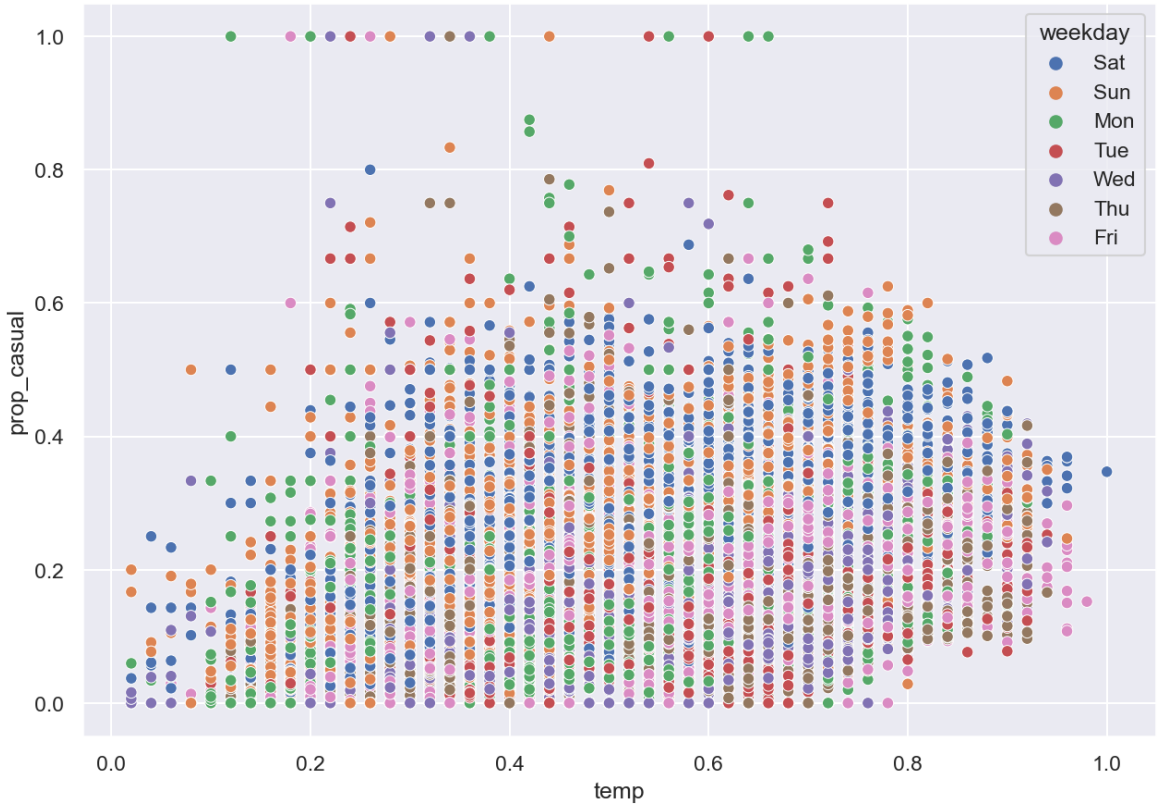
In [ ]:
```
bike['prop_casual'] = bike['casual'] / (bike['cnt'])
```

### Question 6b

In order to examine the relationship between proportion of casual riders and temperature, we can create a scatterplot using `sns.scatterplot`. We can even use color/hue to encode the information about day of week. Run the cell below, and you'll see we end up with a big mess that is impossible to interpret.

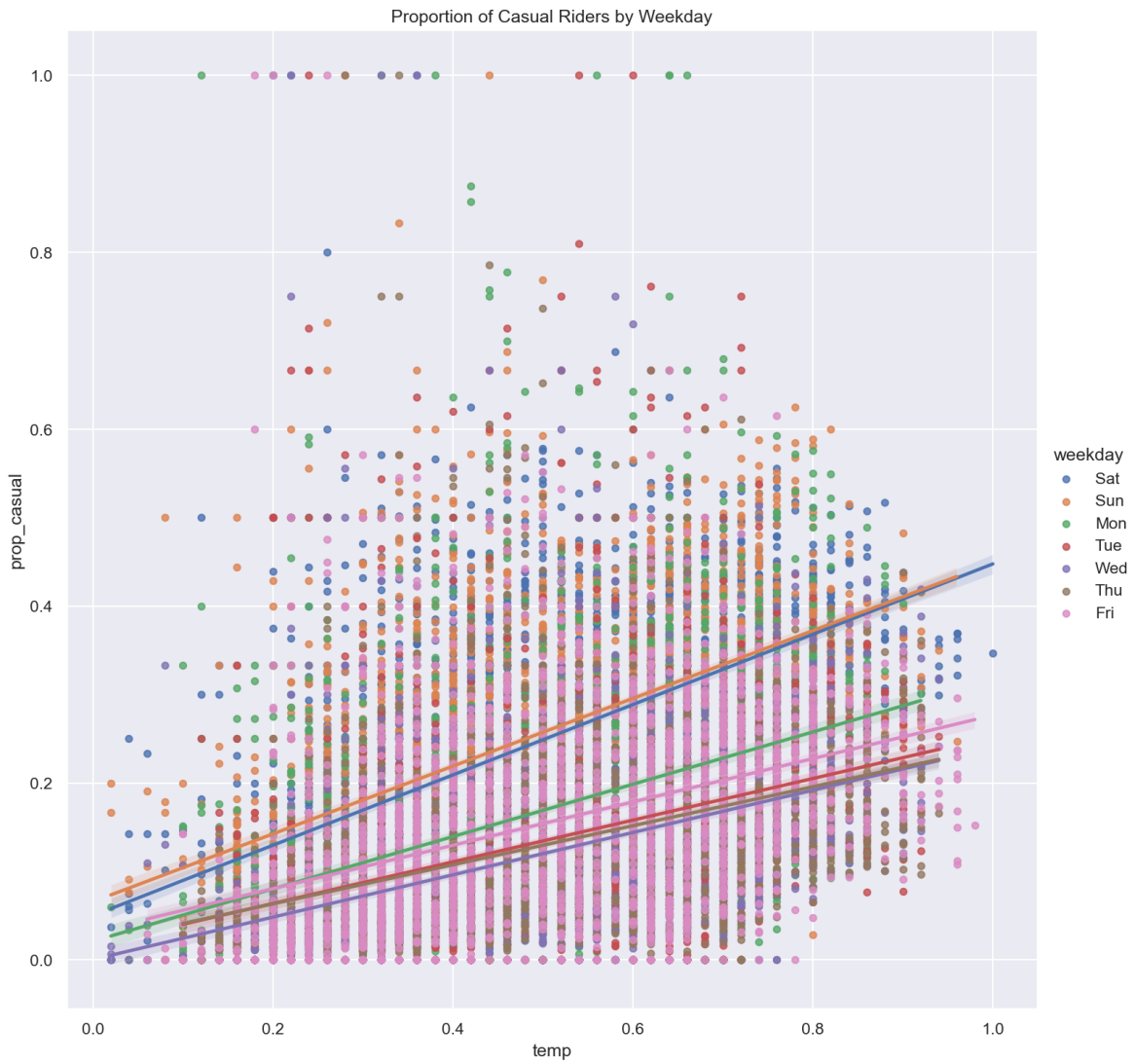
**Hint**: You will need to set the `data`, `x`, `y`, and `hue` in the `sns.scatterplot` call.

In [ ]:
```
plt.figure(figsize=(10, 7))
sns.scatterplot(data=bike, x="temp", y="prop_casual", hue=
```



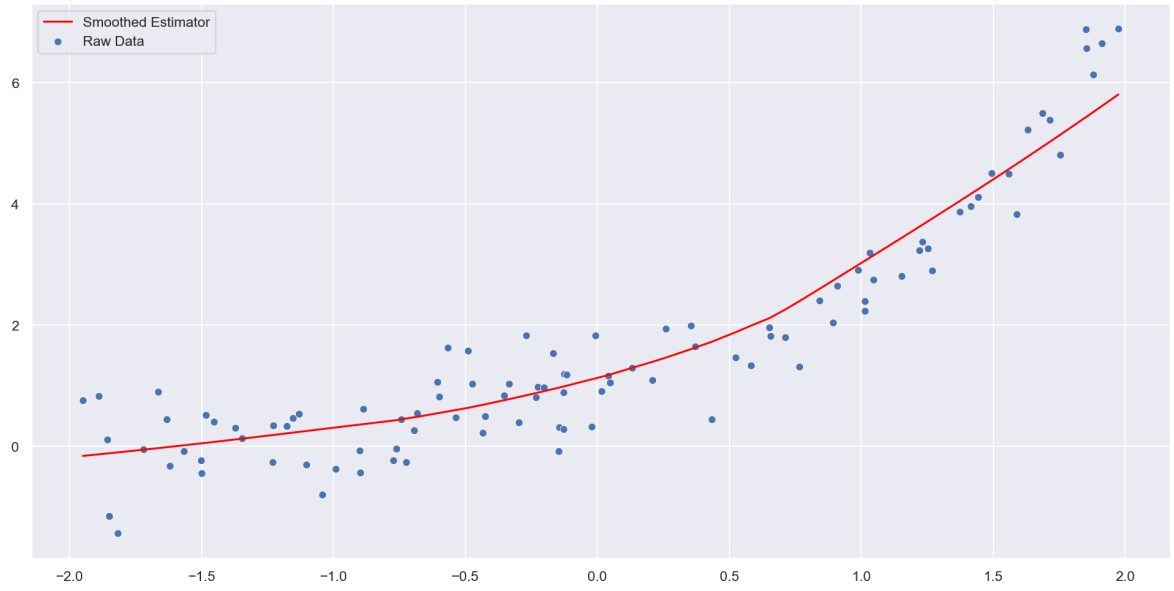

We could attempt linear regression using `sns.lmplot` as shown below, which hint at some relationships between temperature and proportional casual, but the plot is still fairly unconvincing.

In [ ]:
```
sns.lmplot(data=bike, x="temp", y="prop_casual", hue="week
plt.title("Proportion of Casual Riders by Weekday");
```

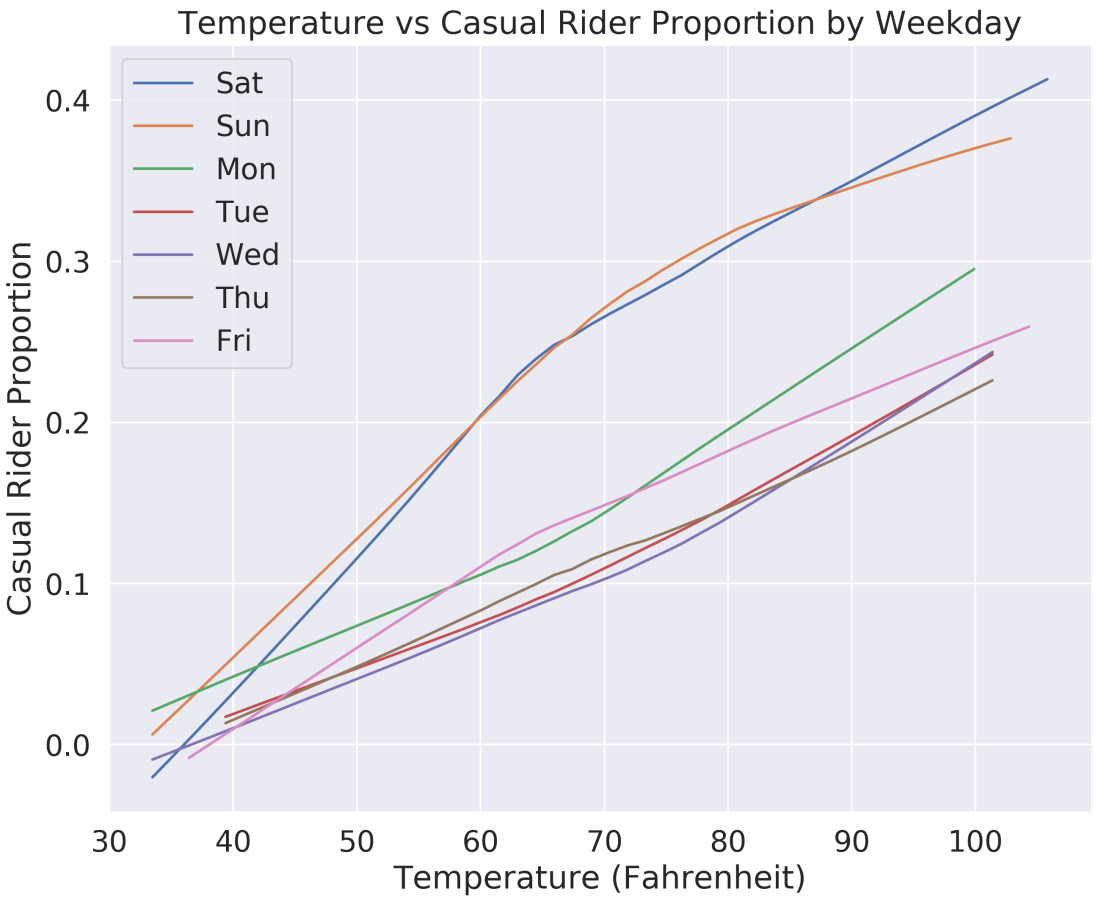



A better approach is to use local smoothing. The basic idea is that for each x value, we compute some sort of representative y value that captures the data close to that x value. One technique for local smoothing is "Locally Weighted Scatterplot Smoothing" or LOWESS. An example is below. The red curve shown is a smoothed version of the scatterplot.

In [ ]:
```
from statsmodels.nonparametric.smoothers_lowess import low
# Make noisy data
xobs = np.sort(np.random.rand(100)*4.0 - 2)
yobs = np.exp(xobs) + np.random.randn(100) / 2.0
sns.scatterplot(xobs, yobs, label="Raw Data")

# Predict 'smoothed' valued for observations
ysmooth = lowess(yobs, xobs, return_sorted=False)
sns.lineplot(xobs, ysmooth, label="Smoothed Estimator", co
plt.legend();
```




In our case with the bike ridership data, we want 7 curves, one for each day of the week. The x-axis will be the temperature and the y-axis will be a smoothed version of the proportion of casual riders.




You should use `statsmodels.nonparametric.smoothers_lowess.lowess` just like the example above. Unlike the example above, plot ONLY the lowess curve. Do not plot the actual data, which would result in overplotting. For this problem, the simplest way is to use a loop.

You do not need to match the colors on our sample plot as long as the colors in your plot make it easy to distinguish which day they represent.

**Hints:**

- Start by just plotting only one day of the week to make sure you can do that first.
- The `lowess` function expects y coordinate first, then x coordinate. You should also set the `return_sorted` field to `False`.
- Look at the top of this homework notebook for a description of the temperature field to know how to convert to Fahrenheit. By default, the temperature field ranges from 0.0 to 1.0. In case you need it, $\text{Fahrenheit} = \text{Celsius} * \frac{9}{5} + 32$.

Note: If you prefer plotting temperatures in Celsius, that's fine as well!

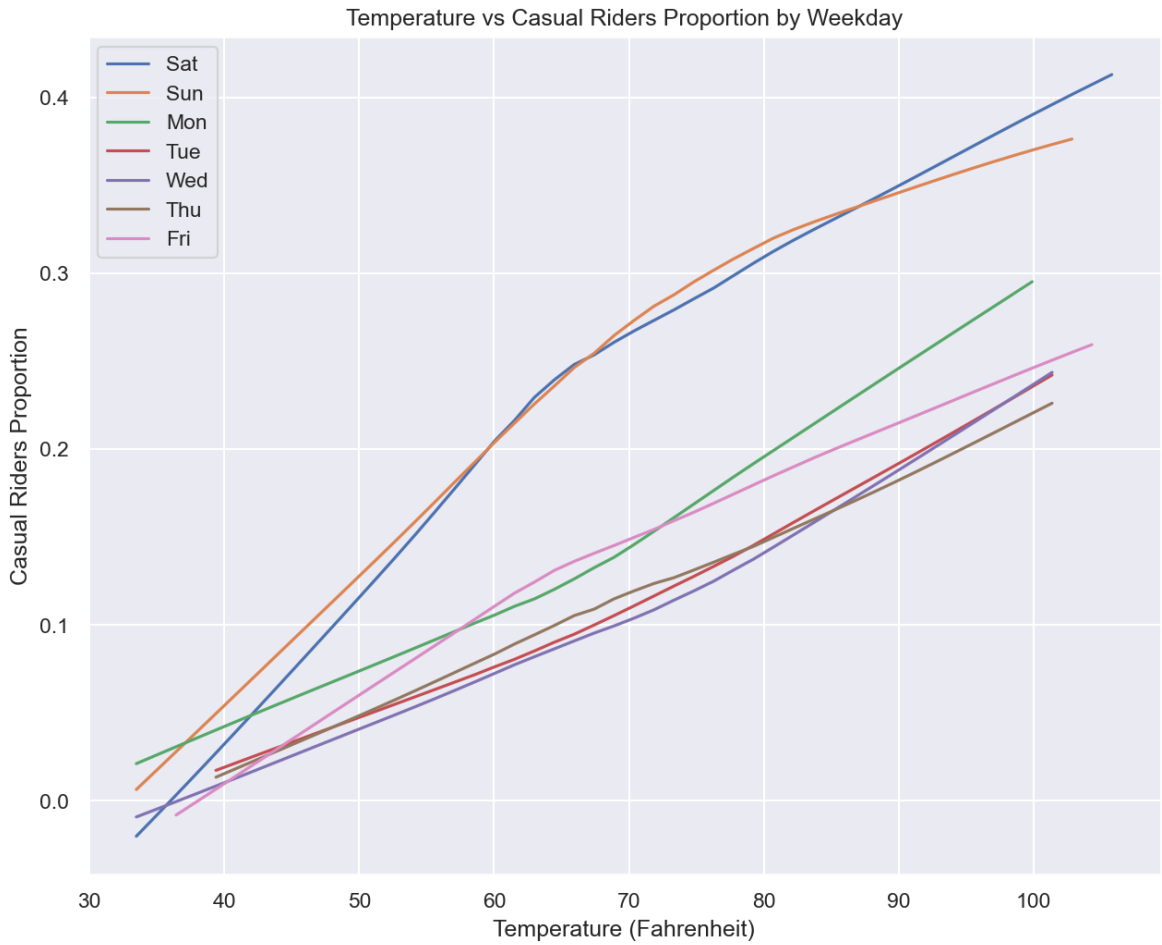
In [ ]:
```
bike.head()
```

Out[ ]:

| | instant | dteday | season | yr | mnth | hr | holiday | weekday | workingday |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2011-01-01 | 1 | 0 | 1 | 0 | no | Sat | no |
| **1** | 2 | 2011-01-01 | 1 | 0 | 1 | 1 | no | Sat | no |
| **2** | 3 | 2011-01-01 | 1 | 0 | 1 | 2 | no | Sat | no |
| **3** | 4 | 2011-01-01 | 1 | 0 | 1 | 3 | no | Sat | no |
| **4** | 5 | 2011-01-01 | 1 | 0 | 1 | 4 | no | Sat | no |

In [ ]:
```
from statsmodels.nonparametric.smoothers_lowess import low

plt.figure(figsize=(10,8))
for day in bike['weekday'].unique():
    xobs = bike[bike['weekday'] == day]['temp'] * 41 * 9/5
    yobs = bike[bike['weekday'] == day]['prop_casual']
```

```
    ysmooth = lowess(yobs, xobs, return_sorted=False)
    sns.lineplot(xobs, ysmooth, label=day)

plt.xlabel('Temperature (Fahrenheit)')
plt.ylabel('Casual Riders Proportion')
plt.title('Temperature vs Casual Riders Proportion by Week
```




## Question 6c

What do you see from the curve plot? How is `prop_casual` changing as a function of temperature? Do you notice anything else interesting?

- The proportion of casual riders is higher on weekends.
- The proportion of casual changes linearly as temperature changes.
- When it is cold, the proportion is nearly zero which means casual users might choose other transportation during the cold days. However, registered users may still take the sharing bikes because they pay for it.

# 7: Expanding our Analysis

## Question 7

### Question 7A

Imagine you are working for a Bike Sharing Company that collaborates with city planners, transportation agencies, and policy makers in order to implement bike sharing in a city. These stakeholders would like to reduce congestion and lower transportation costs. They also want to ensure the bike sharing program is implemented equitably. In this sense, equity is a social value that is informing the deployment and assessment of your bike sharing technology.

Equity in transportation includes: improving the ability of people of different socio-economic classes, genders, races, and neighborhoods to access and afford the transportation services, and assessing how inclusive transportation systems are over time.

Do you think the `bike` data as it is can help you assess equity? If so, please explain. If not, how would you change the dataset? You may discuss how you would change the granularity, what other kinds of variables you'd introduce to it, or anything else that might help you answer this question.

The bike data has a significant limitation in assessing equity since it lacks the variables related to genders, races and so on.

Improvement:

- Add data on pickup and drop-off locations, ideally with neighborhood or zip code information.
- Add data about genders. For example, how much males and females use the sharing bike one day.
- Add data about races.
- Add some user feedback.

In [ ]:
```
# Use this cell for scratch work. If you need to add more
```

## Question 7B

Bike sharing is growing in popularity and new cities and regions are making efforts to implement bike sharing systems that complement their other transportation offerings. The goals of these efforts are to have bike sharing serve as an alternate form of transportation in order to alleviate congestion, provide geographic connectivity, reduce carbon emissions, and promote inclusion among communities.

Bike sharing systems have spread to many cities across the country. The company you work for asks you to determine the feasibility of expanding bike sharing to additional cities of the U.S.

Based on your plots in this assignment, what would you recommend and why? Please list at least two reasons why, and mention which plot(s) you drew you analysis from.

**Note**: There isn't a set right or wrong answer for this question, feel free to come up with your own conclusions based on evidence from your plots!

1. From the plot in the Question 2c, we know that registered riders who may be workers use much more sharing bikes than the casual riders in workdays. So I think the planner of the city can provide more sharing bikes around company in workdays to maximize the utility of sharing bikes.
2. The plot in Question 6b shows a strong relationship between temperature and the proportion of casual riders. As temperature increases, so does the proportion of casual riders, particularly on weekends. This suggests that bike sharing systems might be more successful in cities with milder climates or longer warm seasons.

In [ ]:
```
# Use this cell for scratch work. If you need to add more
```

# Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. **Please save before exporting!**