Homework 6

In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight') # Use plt.style.available to see more styles
sns.set()
%matplotlib inline
```

world bank data

Loading the data

Let us load some World Bank data into a `pd.DataFrame` object named `wb` .

In [3]:

```python
wb = pd.read_csv("./data/world_bank_misc.csv", index_col=0)
wb.head()
```

Out[3]:

	Primary completion rate: Male: % of relevant age group: 2015	Primary completion rate: Female: % of relevant age group: 2015	Lower secondary completion rate: Male: % of relevant age group: 2015	Lower secondary completion rate: Female: % of relevant age group: 2015	Youth literacy rate: Male: % of ages 15-24: 2005-14	Youth literacy rate: Female: % of ages 15-24: 2005-14	Adult literacy rate: Male: % ages 15 and older: 2005-14	Adu litera rat Femal % ag 15 a olde 2005-
Afghanistan	NaN	NaN	NaN	NaN	62.0	32.0	45.0	18
Albania	108.0	105.0	97.0	97.0	99.0	99.0	98.0	96
Algeria	106.0	105.0	68.0	85.0	96.0	92.0	83.0	68
American Samoa	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Andorra	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 45 columns

This table contains some interesting columns:

In [4]:

```
list(wb.columns)
```

Out[4]:

```
['Primary completion rate: Male: % of relevant age group: 2015',
 'Primary completion rate: Female: % of relevant age group: 2015',
 'Lower secondary completion rate: Male: % of relevant age group: 2015',
 'Lower secondary completion rate: Female: % of relevant age group: 2015',
 'Youth literacy rate: Male: % of ages 15-24: 2005-14',
 'Youth literacy rate: Female: % of ages 15-24: 2005-14',
 'Adult literacy rate: Male: % ages 15 and older: 2005-14',
 'Adult literacy rate: Female: % ages 15 and older: 2005-14',
 'Students at lowest proficiency on PISA: Mathematics: % of 15 year-olds: 2015',
 'Students at lowest proficiency on PISA: Reading: % of 15 year-olds: 2015',
 'Students at lowest proficiency on PISA: Science: % of 15 year-olds: 2015',
 'Population: millions: 2016',
 'Surface area: sq. km thousands: 2016',
 'Population density: people per sq. km: 2016',
 'Gross national income, Atlas method: $ billions: 2016',
 'Gross national income per capita, Atlas method: $: 2016',
 'Purchasing power parity gross national income: $ billions: 2016',
 'per capita: $: 2016',
 'Gross domestic product: % growth : 2016',
 'per capita: % growth: 2016',
 'Prevalence of smoking: Male: % of adults: 2015',
 'Prevalence of smoking: Female: % of adults: 2015',
 'Incidence of tuberculosis: per 100,000 people: 2015',
 'Prevalence of diabetes: % of population ages 20 to 79: 2015',
 'Incidence of HIV: Total: % of uninfected population ages 15-49: 2015',
 'Prevalence of HIV: Total: % of population ages 15-49: 2015',
 "Prevalence of HIV: Women's share of population ages 15+ living with HIV: %: 2015",
 'Prevalence of HIV: Youth, Male: % of population ages 15-24: 2015',
 'Prevalence of HIV: Youth, Female: % of population ages 15-24: 2015',
 'Antiretroviral therapy coverage: % of people living with HIV: 2015',
 'Cause of death: Communicable diseases and maternal, prenatal, and nutrition condit
ions: % of population: 2015',
 'Cause of death: Non-communicable diseases: % of population: 2015',
 'Cause of death: Injuries: % of population: 2015',
 'Access to an improved water source: % of population: 1990',
 'Access to an improved water source: % of population: 2015',
 'Access to improved sanitation facilities: % of population: 1990',
 'Access to improved sanitation facilities: % of population: 2015',
 'Child immunization rate: Measles: % of children ages 12-23 months: 2015',
 'Child immunization rate: DTP3: % of children ages 12-23 months: 2015',
 'Children with acute respiratory infection taken to health provider: % of children
under age 5 with ARI: 2009-2016',
 'Children with diarrhea who received oral rehydration and continuous feeding: % of
children under age 5 with diarrhea: 2009-2016',
 'Children sleeping under treated bed nets: % of children under age 5: 2009-2016',
 'Children with fever receiving antimalarial drugs: % of children under age 5 with f
ever: 2009-2016',
 'Tuberculosis: Treatment success rate: % of new cases: 2014',
 'Tuberculosis: Cases detection rate: % of new estimated cases: 2015']
```

Scaling

At first, let's create a DataFrame with the `Adult literacy rate` and the `Gross national income per capita`:

In [5]:

```python
#creates a DataFrame with the appropriate index
df = pd.DataFrame(index=wb.index)

#copies the Series we want
df['lit'] = wb['Adult literacy rate: Female: % ages 15 and older: 2005-14']
df['inc'] = wb['Gross national income per capita, Atlas method: $: 2016']

#the line below drops all records that have a NaN value in either column
df.dropna(inplace=True)
print("Original records:", len(wb))
print("Final records:", len(df))
df.head(5)
```

```
Original records: 216
Final records: 147
```

Out[5]:

	lit	inc
Afghanistan	18.0	580.0
Albania	96.0	4250.0
Algeria	68.0	4270.0
Angola	60.0	3440.0
Antigua and Barbuda	99.0	13400.0

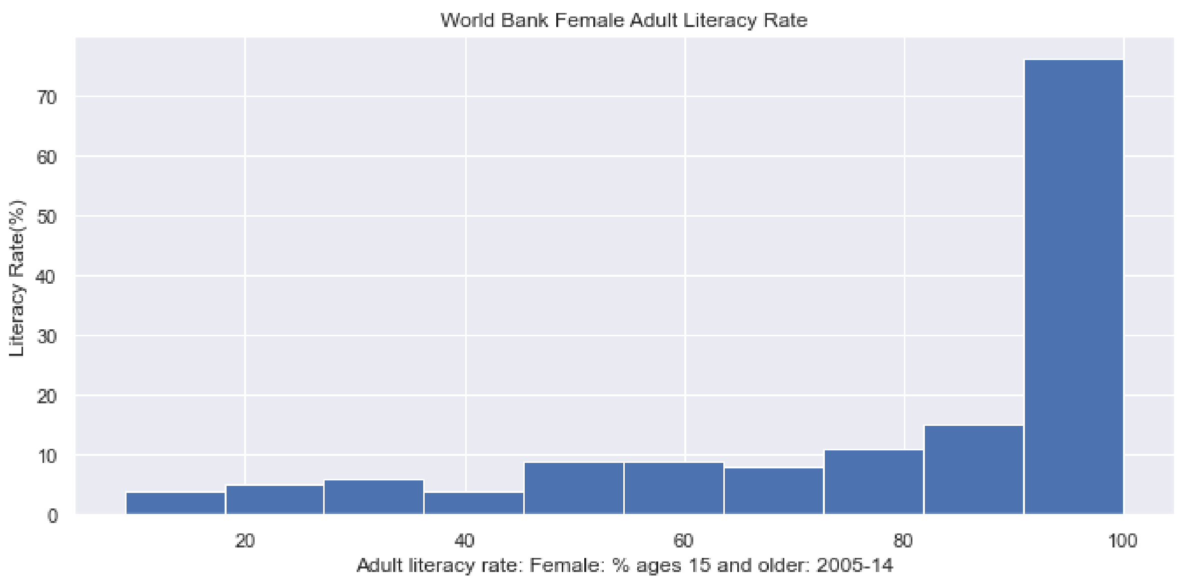
Question: Suppose we wanted to build a histogram of the distribution of literacy rates (i.e. `df["lit"]`).

In [6]:

```
plt.figure(figsize=(10,5))
plt.subplot(1,1,1)

# you can insert your code here:
plt.hist(df["lit"])
#
plt.xlabel("Adult literacy rate: Female: % ages 15 and older: 2005-14")
plt.ylabel("Literacy Rate(%)")
plt.title('World Bank Female Adult Literacy Rate')
```

Out[6]:

```
Text(0.5, 1.0, 'World Bank Female Adult Literacy Rate')
```



Question: In the cell below, create a plot of income per capita using the `displot` (https://seaborn.pydata.org/generated/seaborn.displot.html) function. When you call `displot` , set the `kde` parameter to false, e.g. `displot(s, kde=False)` .

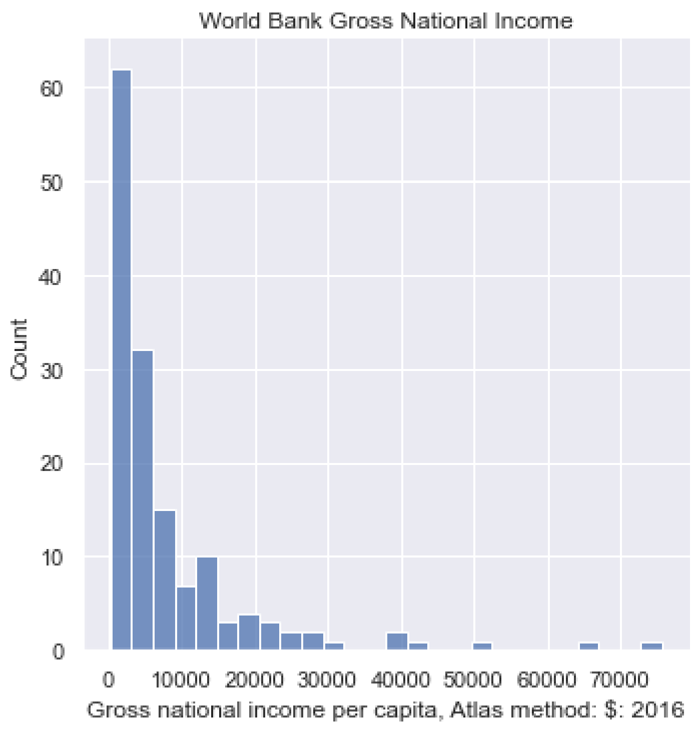
Don't forget to title the plot and label axes!

In [7]:

```python
# your code here
sns.displot(df["inc"],kde = False)
plt.xlabel("Gross national income per capita, Atlas method: $: 2016")
plt.title("World Bank Gross National Income")
```

Out[7]:

```
Text(0.5, 1.0, 'World Bank Gross National Income')
```



You should see histograms that show the counts of how many data points appear in each bin. `distplot` uses a heuristic called the Freedman-Diaconis rule to automatically identify the best bin sizes, though it is possible to set the bins yourself.

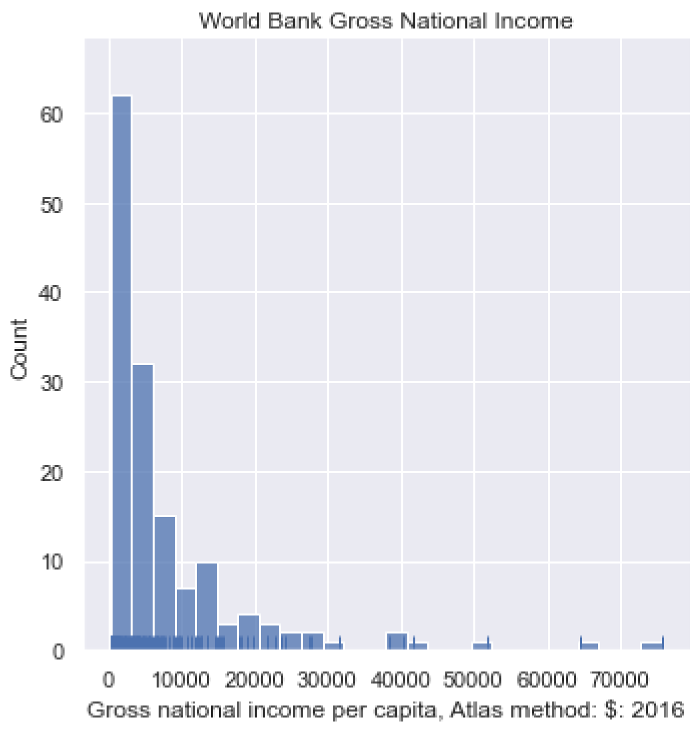
Question: In the cell below, set the `rug` parameter to True in `displot` with the same plot function.

In [8]:

```
# your code here
sns.displot(df["inc"],kde = False,rug = True)
plt.xlabel("Gross national income per capita, Atlas method: $: 2016")
plt.title("World Bank Gross National Income")
```

Out[8]:

```
Text(0.5, 1.0, 'World Bank Gross National Income')
```



Above, you should see little lines at the bottom of the plot showing the actual data points.

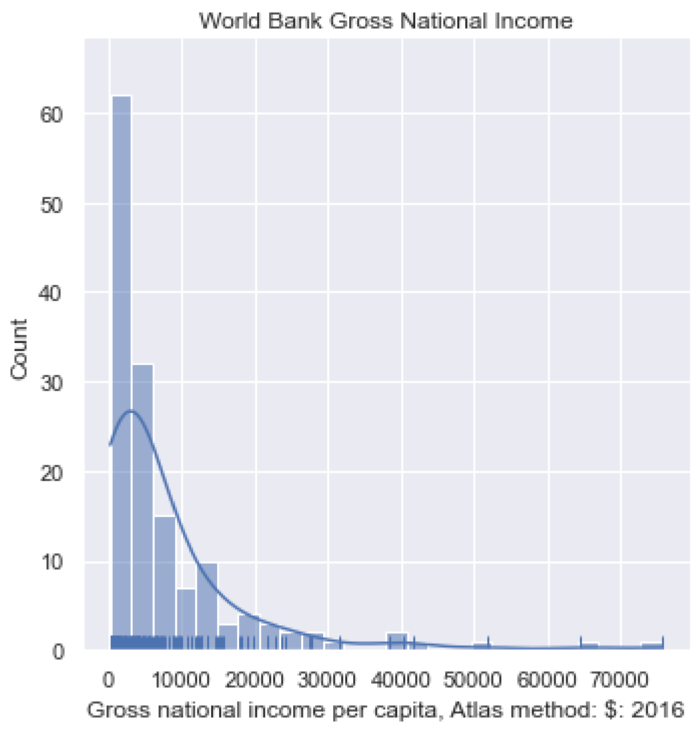
Question: In the cell below, further set the `kde` parameter to True in your `displot` to see what happens:

In [9]:

```python
# your code here
sns.displot(df["inc"],kde = True,rug = True)
plt.xlabel("Gross national income per capita, Atlas method: $: 2016")
plt.title("World Bank Gross National Income")
```

Out[9]:

```
Text(0.5, 1.0, 'World Bank Gross National Income')
```



Question: Transforming the `inc` data logarithmically gives us a more symmetric distribution of values. This can make it easier to see patterns.

In the cell below, make a distribution plot of `inc` with the data transformed using `np.log10` and `kde=True` . If you want to see the exact counts, just set `kde=False` . If you don't specify the `kde` parameter, it is by default set to True.

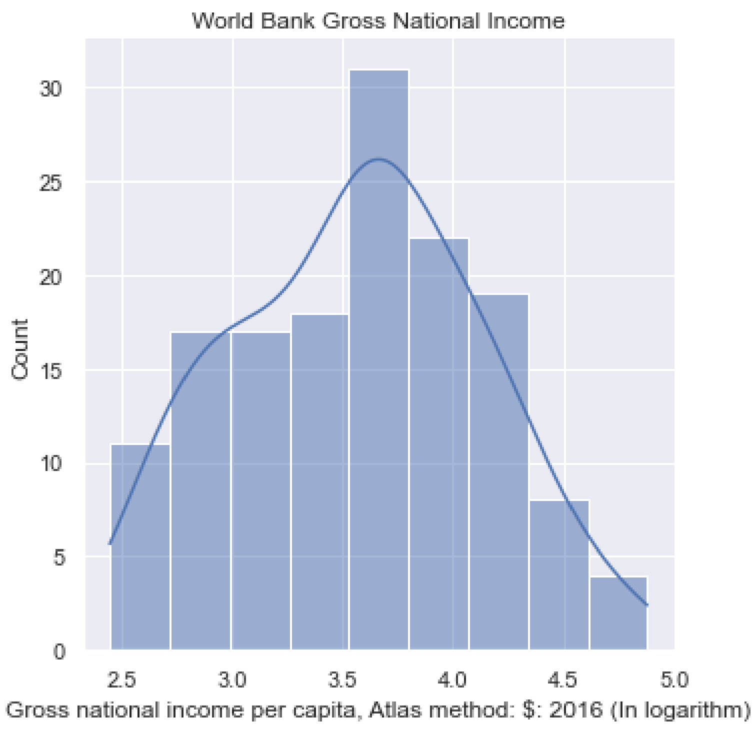
Don't forget to set the labels and title.

In [10]:

```
# your code here
sns.displot(np.log10(df["inc"]),kde = True)
plt.xlabel("Gross national income per capita, Atlas method: $: 2016 (In logarithm)")
plt.title("World Bank Gross National Income")
```

Out[10]:

```
Text(0.5, 1.0, 'World Bank Gross National Income')
```



Question: If we want to examine the relationship between the female adult literacy rate and the gross national income per capita, we need to make a scatter plot.

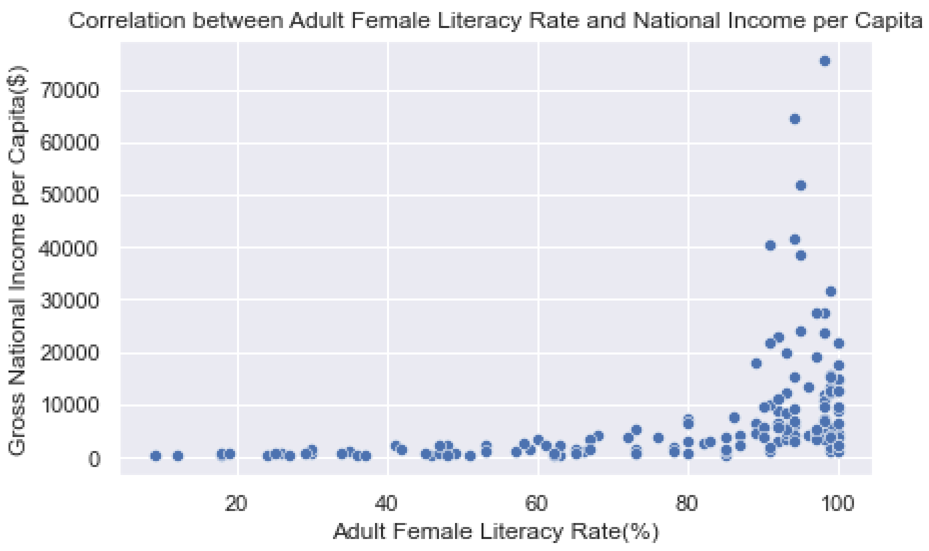
In the cell below, create a scatter plot of untransformed income per capita and literacy rate using the `sns.scatterplot` function. Make sure to label both axes.

In [11]:

```
# your code here
sns.scatterplot(data = df,x = "lit",y = "inc")
plt.xlabel("Adult Female Literacy Rate(%)")
plt.ylabel("Gross National Income per Capita($)")
plt.title("Correlation between Adult Female Literacy Rate and National Income per Capita")
```

Out[11]:

```
Text(0.5, 1.0, 'Correlation between Adult Female Literacy Rate and National Income p
er Capita')
```



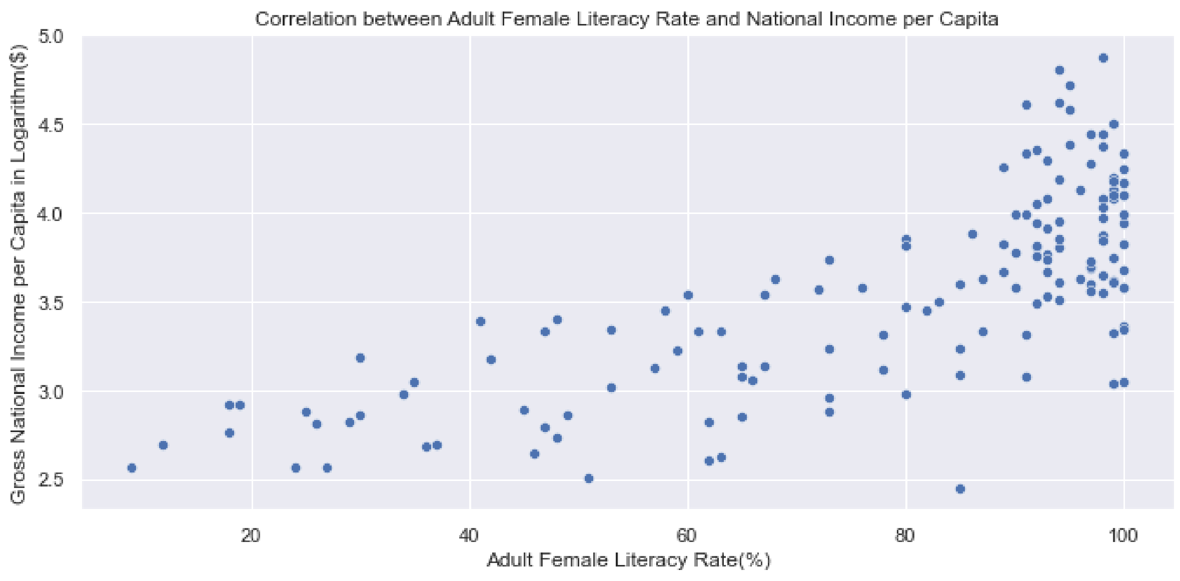
Question: In the cell below, make a scatter plot of log-transformed income per capita against literacy rate to better assess the relationship.

In [12]:

```
# your code here
plt.figure(figsize=(10,5))
df1 = df
df1["inc"] = np.log10(df["inc"])
sns.scatterplot(data = df1,x = "lit",y = "inc")
plt.xlabel("Adult Female Literacy Rate(%)")
plt.ylabel("Gross National Income per Capita in Logarithm($)")
plt.title("Correlation between Adult Female Literacy Rate and National Income per Capita")
```

Out[12]:

```
Text(0.5, 1.0, 'Correlation between Adult Female Literacy Rate and National Income p
er Capita')
```



bike sharing data

load and understand the Data

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.

The variables in this data frame are defined as:

Variable	Description

Variable	Description
instant	record index
dteday	date
season	1. spring 2. summer 3. fall 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 2. mist and clouds 3. light snow or rain 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

The data is a .zip file in csv format, we can use pandas to read the compressed data by setting the compression="zip":

In [13]:

```
# Run this cell to load the data.  No further action is needed
bike = pd.read_csv('./data/bikeshare.zip', compression="zip")
bike.head()
```

Out[13]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	ate
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2

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` .

Question: Decoding `weekday` , `holiday` , `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.
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` , ... `4` for `Heavy` .

In [14]:

```
# your code here
bike["weekday"] = bike["weekday"].map({1:"Mon",2:"Tue",3:"Wed",4:"Thu",5:"Fri",6:"Sat",0:"S
bike["holiday"] = bike["holiday"].map({1:"yes",0:"no"})
bike["workingday"] = bike["workingday"].map({1:"yes",0:"no"})
bike["weathersit"] = bike["weathersit"].map({1:"Clear",2:"Mist",3:"Light",4:"Heavy"})

bike.head()
```

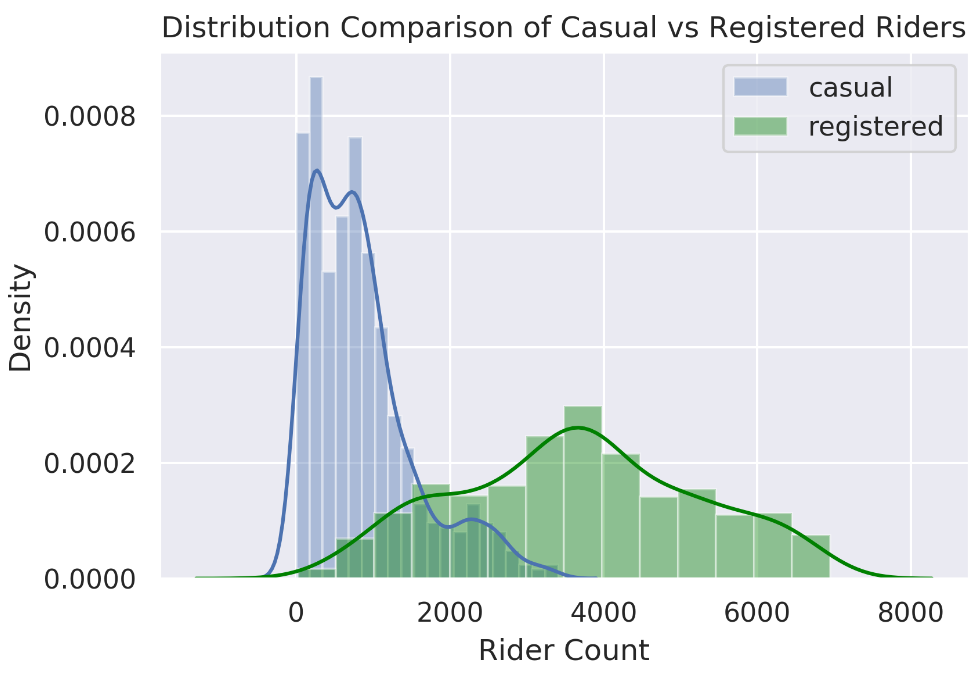
Out[14]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	ate
0	1	2011-01-01	1	0	1	0	no	Sat	no	Clear	0.24	0.2
1	2	2011-01-01	1	0	1	1	no	Sat	no	Clear	0.22	0.2
2	3	2011-01-01	1	0	1	2	no	Sat	no	Clear	0.22	0.2
3	4	2011-01-01	1	0	1	3	no	Sat	no	Clear	0.24	0.2
4	5	2011-01-01	1	0	1	4	no	Sat	no	Clear	0.24	0.2

Exploring the Distribution of Riders

Question: Use the `sns.displot` 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.

Include a legend, xlabel, ylabel, and title. Read the [seaborn plotting tutorial (https://seaborn.pydata.org/tutorial/distributions.html)](https://seaborn.pydata.org/tutorial/distributions.html) if you're not sure how to add these.



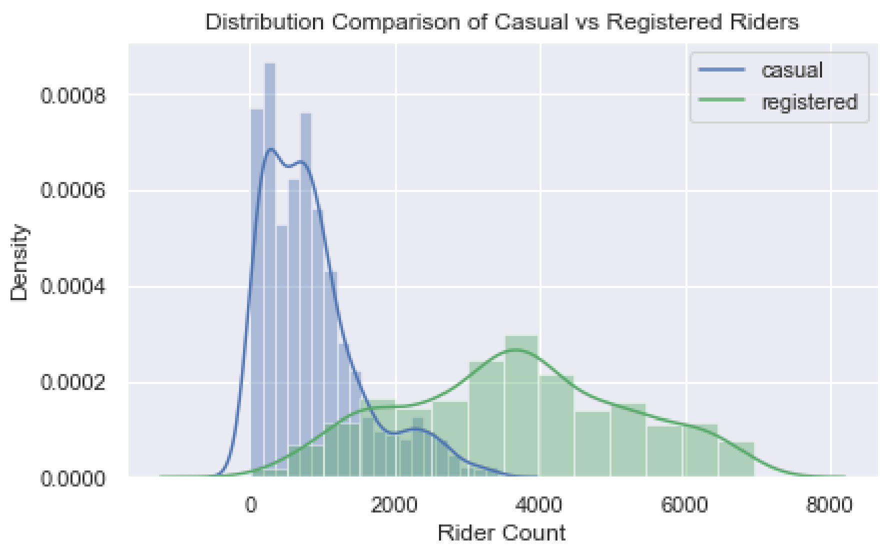
In [15]:

```
# your code here
df2 = bike.groupby("dteday").aggregate({"casual":np.sum,"registered":np.sum})[["casual","re
sns.distplot(df2["casual"],color = "b",kde = True)
sns.distplot(df2["registered"],color = "g",kde = True)
plt.xlabel("Rider Count")
plt.title("Distribution Comparison of Casual vs Registered Riders")
plt.legend(["casual","registered"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  This is separate from the ipykernel package so we can avoid doing imports until
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.githu
b.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  after removing the cwd from sys.path.
```

Out[15]:

```
<matplotlib.legend.Legend at 0x1c0bc71de48>
```

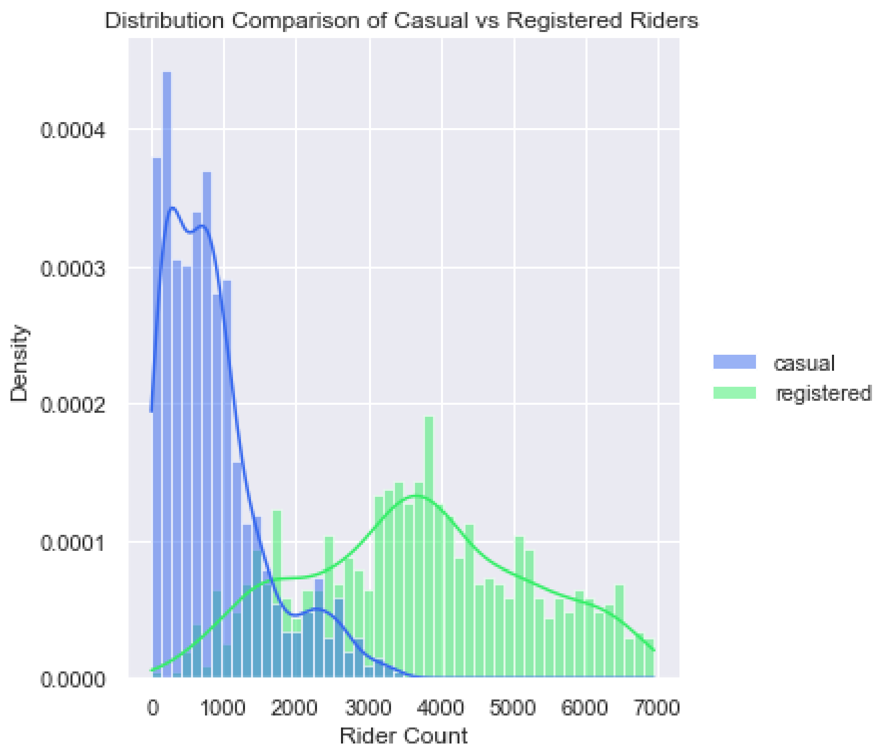


In [16]:

```
#Using displot instead of distplot
df2 = bike.groupby("dteday").aggregate({"casual":np.sum,"registered":np.sum})[["casual","re
sns.displot(df2,kde = True,palette = ["#3366EE","#33EE66"],stat = "density",bins = 50)
plt.xlabel("Rider Count")
plt.title("Distribution Comparison of Casual vs Registered Riders")
```

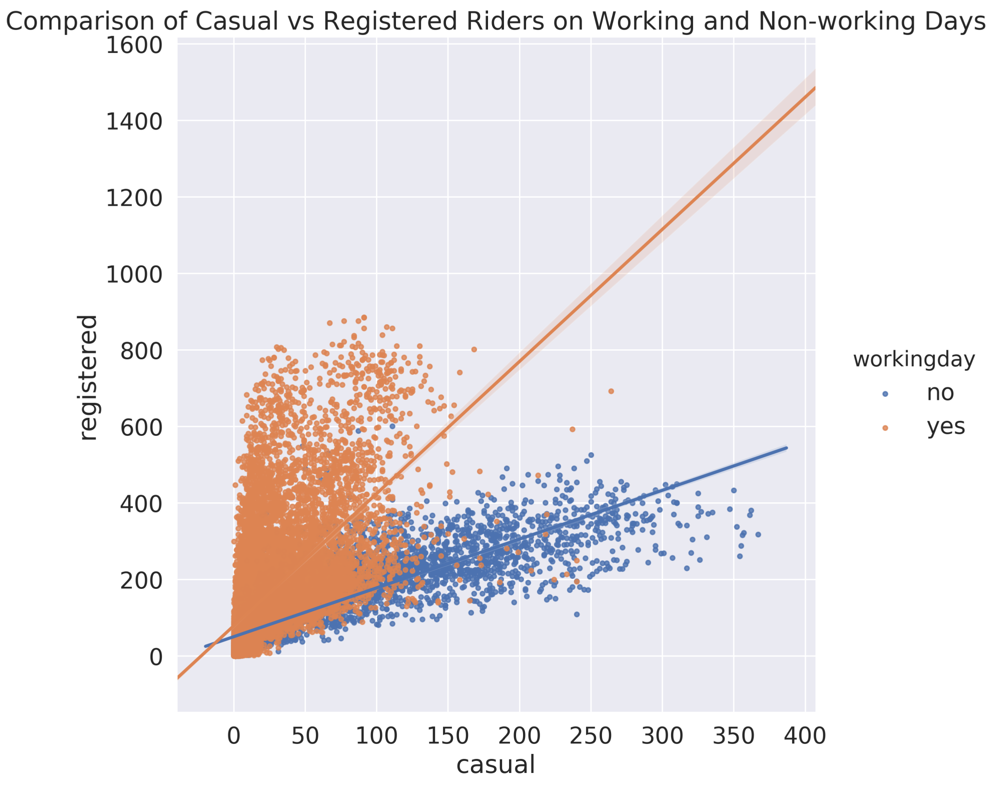
Out[16]:

```
Text(0.5, 1.0, 'Distribution Comparison of Casual vs Registered Riders')
```



Question: The density plots do not show us how the counts for registered and casual riders vary together. Use `sns.lmplot` [(https://seaborn.pydata.org/generated/seaborn.lmplot.html)](https://seaborn.pydata.org/generated/seaborn.lmplot.html) 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.

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` .

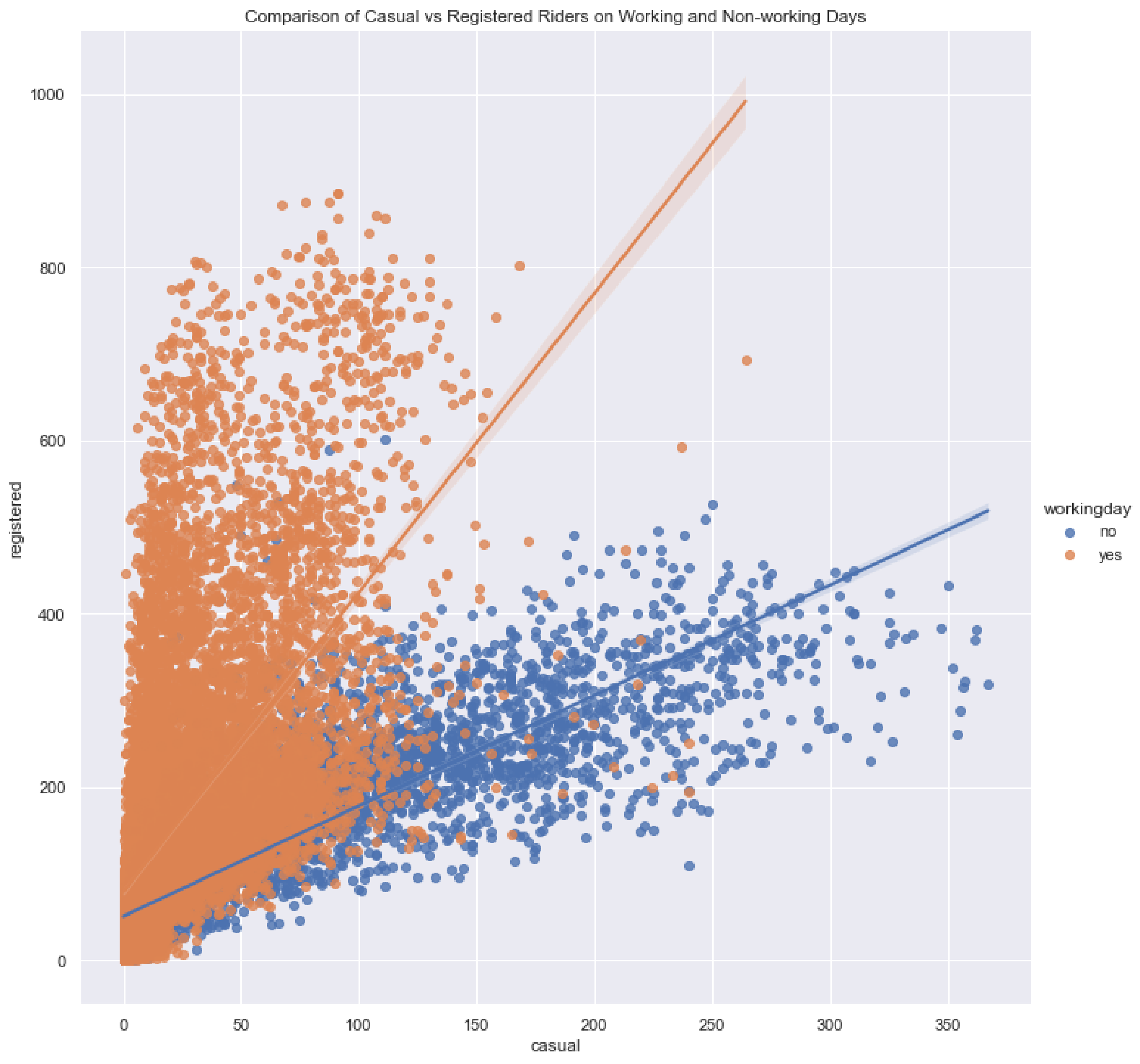

Comparison of Casual vs Registered Riders on Working and Non-working Days
registered
1600
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300
350
400
casual
workingday
no
yes

In [17]:

```
# Make the font size a bit bigger
sns.set(font_scale=1)
# your code here
sns.lmplot(data = bike,x = "casual",y = "registered",hue = "workingday",height = 10,fit_reg
plt.title("Comparison of Casual vs Registered Riders on Working and Non-working Days")
```

Out[17]:

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

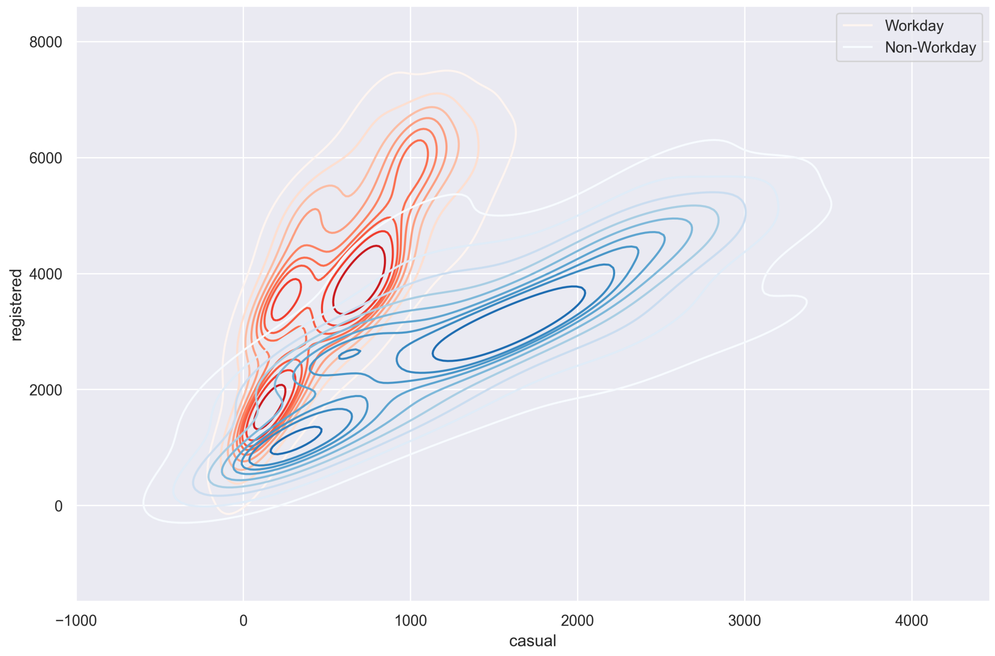


Bivariate Kernel Density Plot

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 (https://seaborn.pydata.org/generated/seaborn.kdeplot.html)](https://seaborn.pydata.org/generated/seaborn.kdeplot.html).

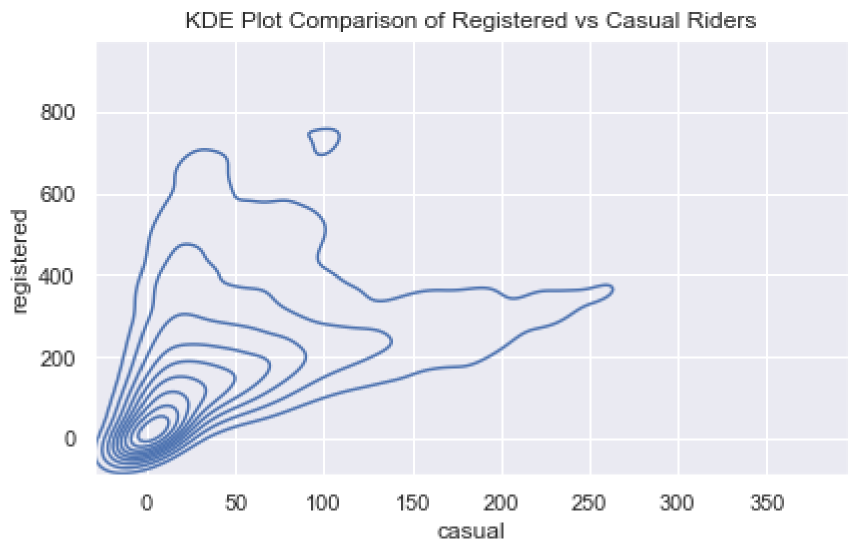
The result we wish to achieve should be a plot that looks like this:



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.

In [18]:

```python
sns.kdeplot(x='casual', y='registered', data=bike)
plt.title('KDE Plot Comparison of Registered vs Casual Riders');
```



Question: 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.

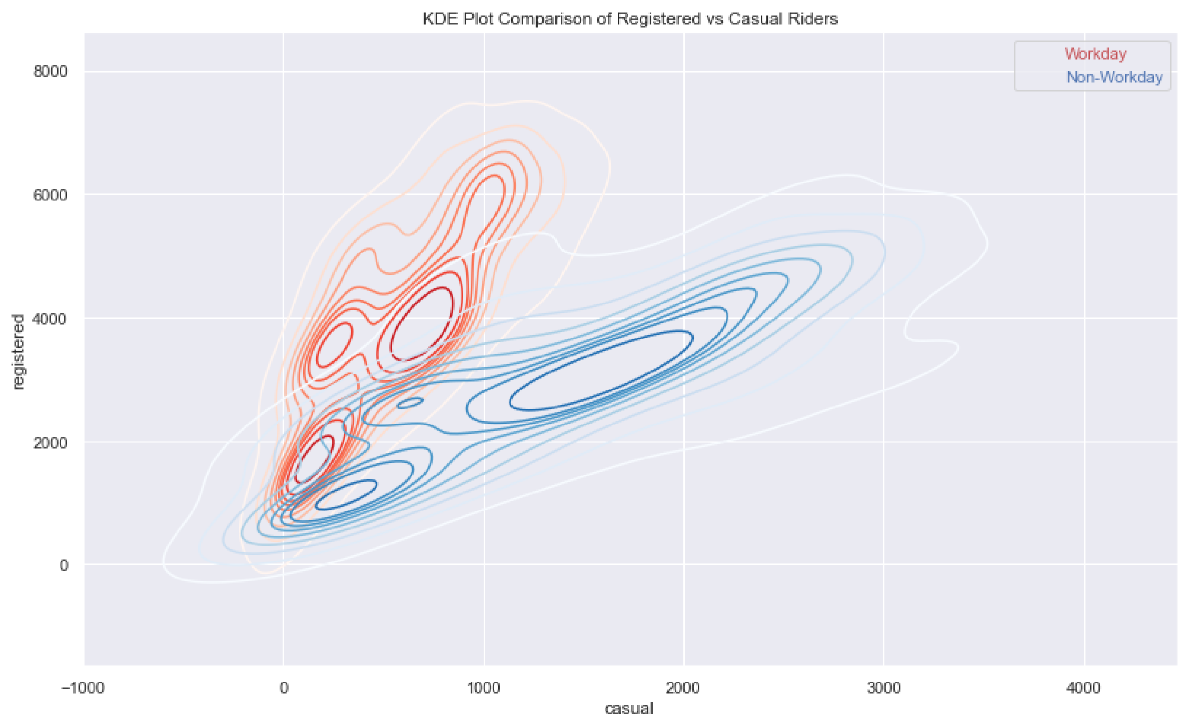
After you get your plot working, experiment by setting `shade=True` in `kdeplot` to see the difference between the shaded and unshaded version. Please submit your work with `shade=False` .

In [30]:

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

df5 = bike
df5["isworkingday"] = (bike["workingday"] == "yes").map({True:"Workday",False:"Non-Workday"
df6 = df5.groupby(["dteday","isworkingday"]).aggregate({"casual":np.sum,"registered":np.sum
df6 = df6.reset_index()
# sns.kdeplot(x='casual',y='registered',data=df6,hue = "isworkingday",shade = False)
sns.kdeplot(x = "casual",y = "registered",data = df6.loc[df6.isworkingday == "Workday"],cma
sns.kdeplot(x = "casual",y = "registered",data = df6.loc[df6.isworkingday == "Non-Workday"]
plt.legend(["Workday","Non-Workday"],labelcolor = ["r","b"])
plt.title('KDE Plot Comparison of Registered vs Casual Riders')
```

Out[30]:

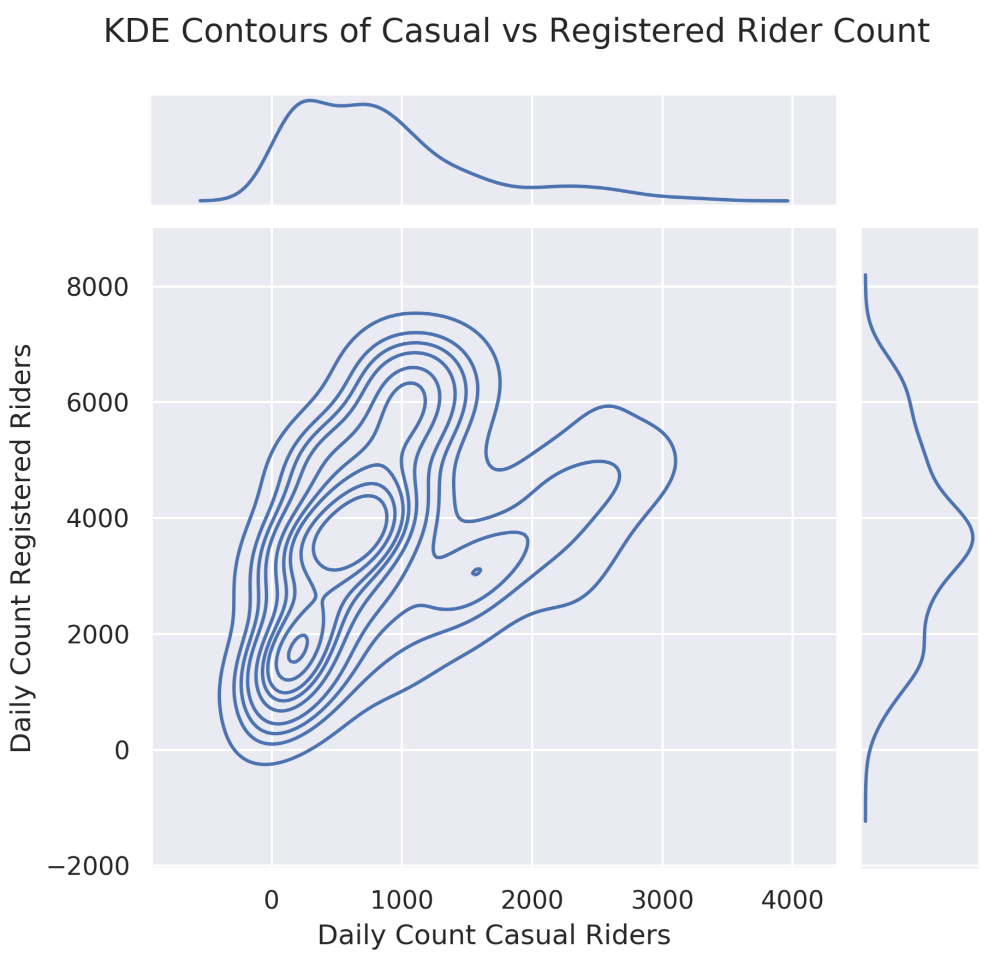
```
Text(0.5, 1.0, 'KDE Plot Comparison of Registered vs Casual Riders')
```



In [ ]:

<span>Question:</span> *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:

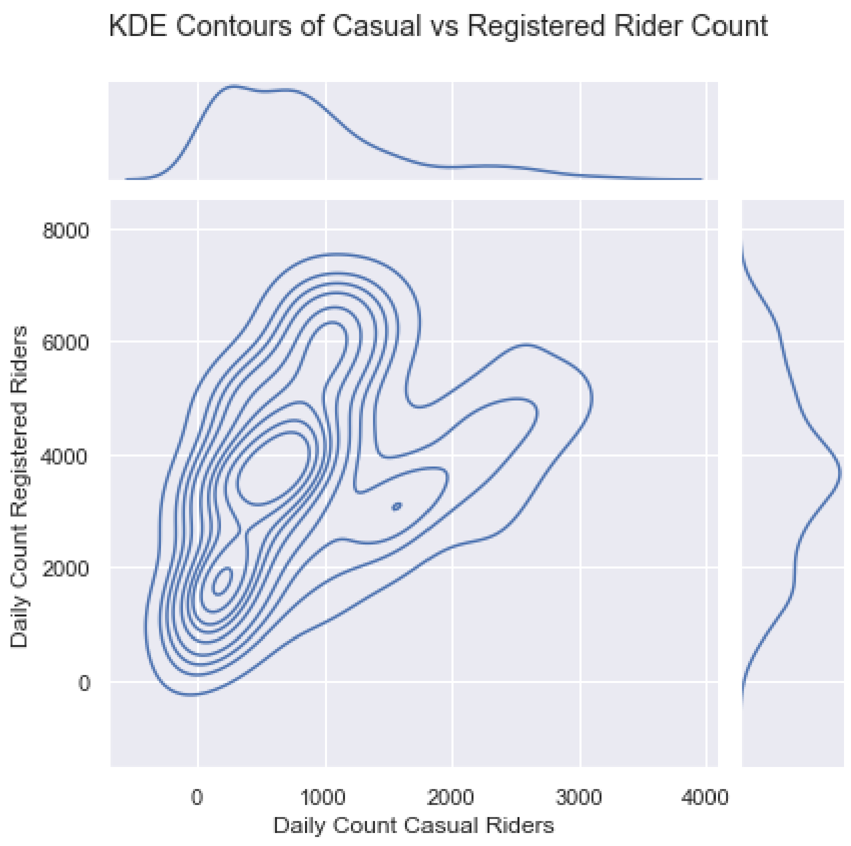
- The seaborn plotting tutorial (https://seaborn.pydata.org/tutorial/distributions.html) has examples that may be helpful.
- Take a look at `sns.jointplot` and its `kind` parameter.
- `set_axis_labels` can be used to rename axes on the contour plot.

In [24]:

```
# your code here
sns.jointplot(x='casual', y='registered', data=df6,kind = "kde")
plt.suptitle("KDE Contours of Casual vs Registered Rider Count")
plt.subplots_adjust(top=0.9)
plt.xlabel("Daily Count Casual Riders")
plt.ylabel("Daily Count Registered Riders")
```

Out[24]:

```
Text(5.560000000000002, 0.5, 'Daily Count Registered Riders')
```



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

In [25]:

```
# your code here
bike["prop_casual"] = bike["casual"] / (bike["casual"] + bike["registered"])
bike["prop_casual"]
```

Out[25]:

```
0        0.187500
1        0.200000
2        0.156250
3        0.230769
4        0.000000
           ...
17374    0.092437
17375    0.089888
17376    0.077778
17377    0.213115
17378    0.244898
Name: prop_casual, Length: 17379, dtype: float64
```

Question: In order to examine the relationship between proportion of casual riders and temperature, we can create a scatterplot using `sns.scatterplot` (https://seaborn.pydata.org/generated/seaborn.scatterplot.html). 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.

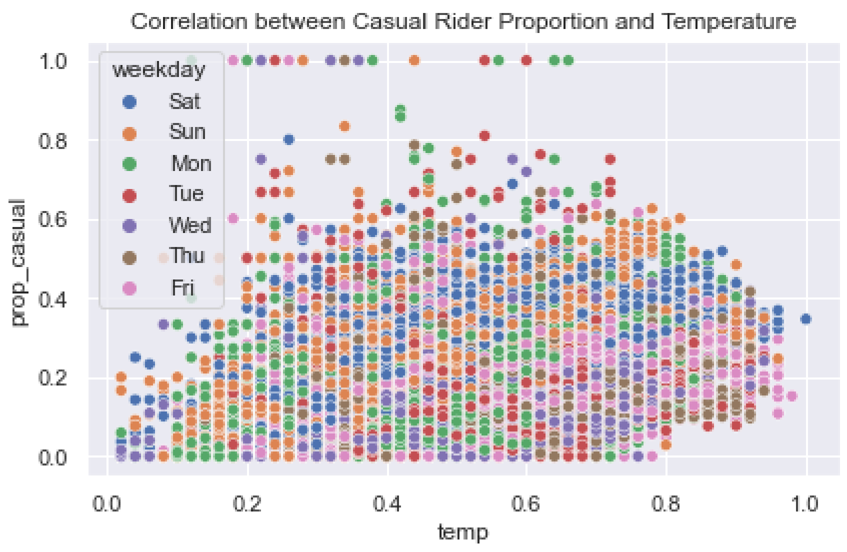
Hint: You will need to set the `data` , `x` , `y` , and `hue` in the `sns.scatterplot` call.

In [26]:

```
# your code here
sns.scatterplot(data = bike,x = "temp",y = "prop_casual",hue = "weekday")
plt.title("Correlation between Casual Rider Proportion and Temperature")
```

Out[26]:

```
Text(0.5, 1.0, 'Correlation between Casual Rider Proportion and Temperature')
```



The End

In [ ]: