Notebook

October 1, 2019

0.0.1 Question 0

Question 0A What is the granularity of the data (i.e. what does each row represent)?
Write your answer here, replacing this text.

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?

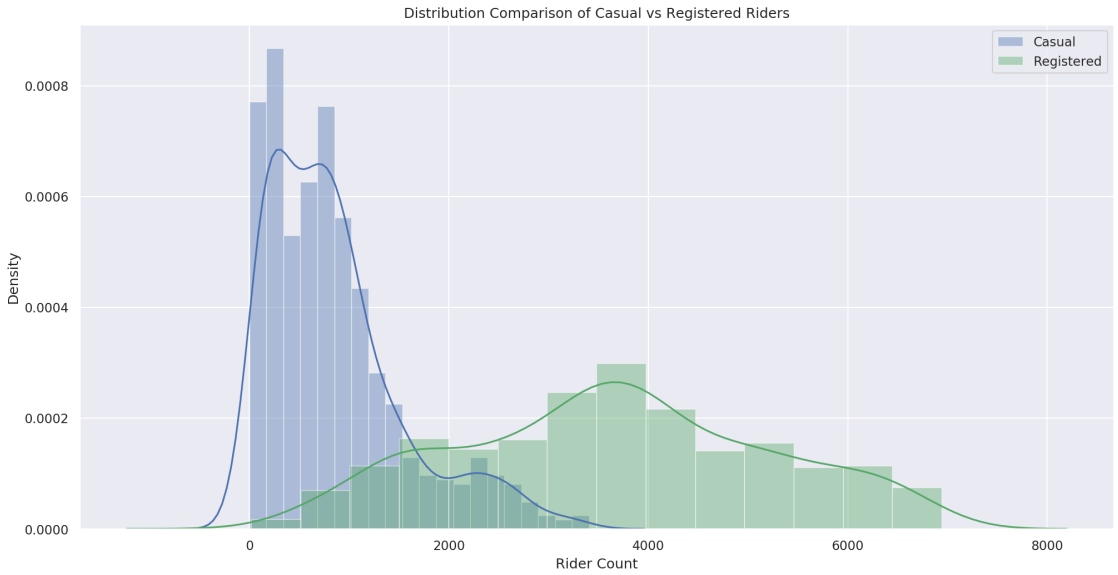
Write your answer here, replacing this text.

0.0.2 Question 2

Question 2a Use the `sns.distplot` 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.

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 [85]: sns.distplot(daily_counts['casual'], kde=True, color='b')
         sns.distplot(daily_counts['registered'], kde=True, color='g')
         plt.xlabel("Rider Count")
         plt.ylabel("Density")
         plt.title('Distribution Comparison of Casual vs Registered Riders')
         plt.legend(['Casual','Registered'])

         plt.show()
```



0.0.3 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.

Looking at the density curves for the casual and registered riders, we see that the highest density for casual riders will have a rider count of around 1000 on a given day. Thus, the mean and mode for the casual riders is around 1000. For the registered riers, we see a mean and mode of around 4000 riders on a given day. The graphs for both casual and registered riders are symmetrical, with casual riders bbeing more left skewed. The data for casual riders is also right tailed. Casual riders has more outliers very high and low density values at a rider count of around 1000. The spread of registered riders is also much more wide compared to the spread of the casual riders.

0.0.4 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.

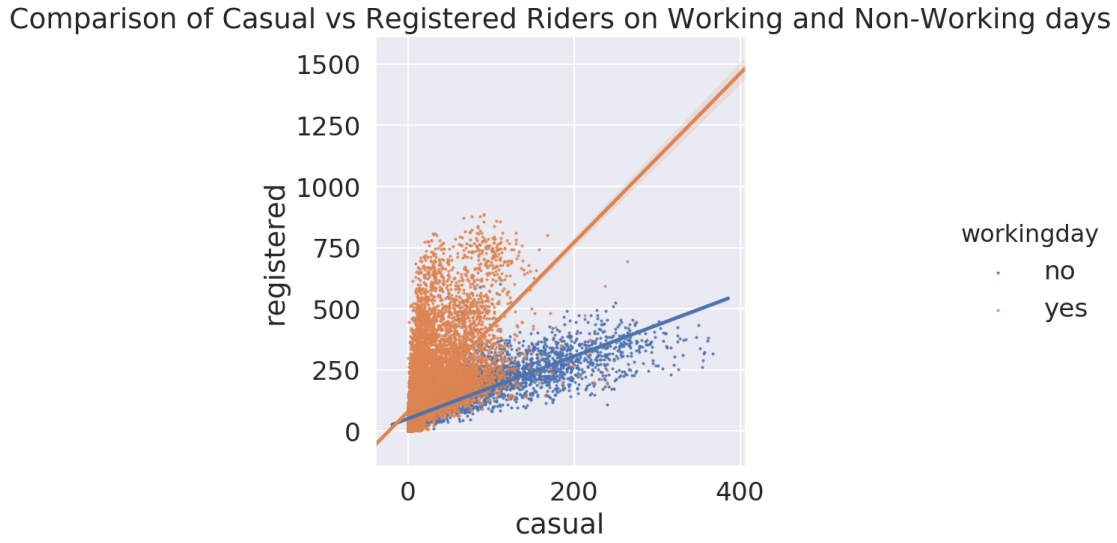
The `lmplot` function will also try to draw a linear regression line (just as you saw in Data 8). Color the points in the scatterplot according to whether or not the day is a working day (your colors do not have to match ours exactly, but they should be different based on whether the day is a working 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 [103]: # Make the font size a bit bigger
          sns.set(font_scale=1.5)

          sns.lmplot(x='casual',y='registered', data=bike, hue='workingday', scatter_kws={'s':1}, fit_r
          plt.xlabel("casual")
          plt.ylabel("registered")
          plt.title('Comparison of Casual vs Registered Riders on Working and Non-Working days')

          plt.show()
```



0.0.5 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?

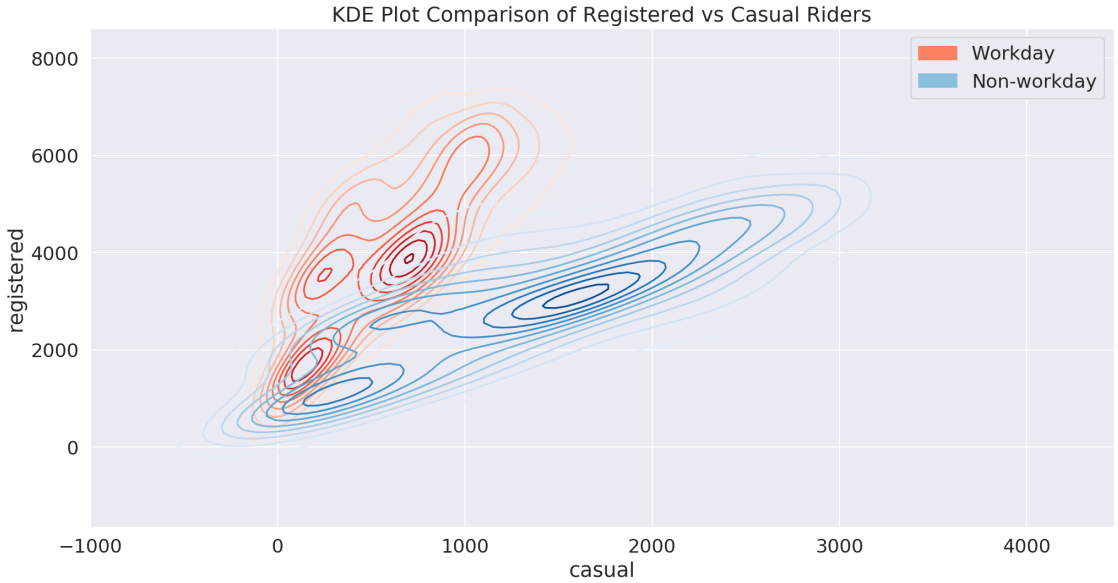
It looks like there are more registered riders during working days, and less registered riders during the weekend. Conversely, there are more casual riders during the weekend and less casual riders during working days.

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 however 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. * Check out this tutorial to see an example of how to set colors for each dataset and how to create a legend. The legend part uses some weird matplotlib syntax that we haven't learned! You'll probably find creating the legend annoying, but it's 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). 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 [116]:
```
import matplotlib.patches as mpatches  # see the tutorial for how we use mpatches to generate

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

# Bivariate KDEs require two data inputs.
# In this case, we will need the daily counts for casual and registered riders on workdays
casual_workday = is_workingday['casual']
registered_workday = is_workingday['registered']

# Use sns.kdeplot on the two variables above to plot the bivariate KDE for weekday rides
sns.kdeplot(casual_workday,registered_workday,cmap="Reds", shade=False,legend=True)

# Repeat the same steps above but for rows corresponding to non-workingdays
is_non_workingday = daily_counts[daily_counts['workingday'] == 'no']
casual_non_workday = is_non_workingday['casual']
registered_non_workday = is_non_workingday['registered']

# Use sns.kdeplot on the two variables above to plot the bivariate KDE for non-workingday rid
sns.kdeplot(casual_non_workday,registered_non_workday,cmap="Blues", shade=False,legend=True)

plt.title('KDE Plot Comparison of Registered vs Casual Riders');
plt.xlabel("casual")
plt.ylabel("registered")
r = sns.color_palette("Reds")[2]
b = sns.color_palette("Blues")[2]

red_patch = mpatches.Patch(color=r, label='Workday')
blue_patch = mpatches.Patch(color=b, label='Non-workday')

plt.legend(handles=[red_patch,blue_patch])

plt.show()
```


KDE Plot Comparison of Registered vs Casual Riders
Workday
Non-workday
registered
casual
8000
6000
4000
2000
0
−1000
0
1000
2000
3000
4000

Question 3b What additional details can you identify from this contour plot that were difficult to determine from the scatter plot?

Looking at this contour plot, we could specifically see in which counts, are there a high number of people. For example, we could see its very concentrated for 2000 registered and 0 casual. Also very concentrated at 4000 registered and 800 casual.

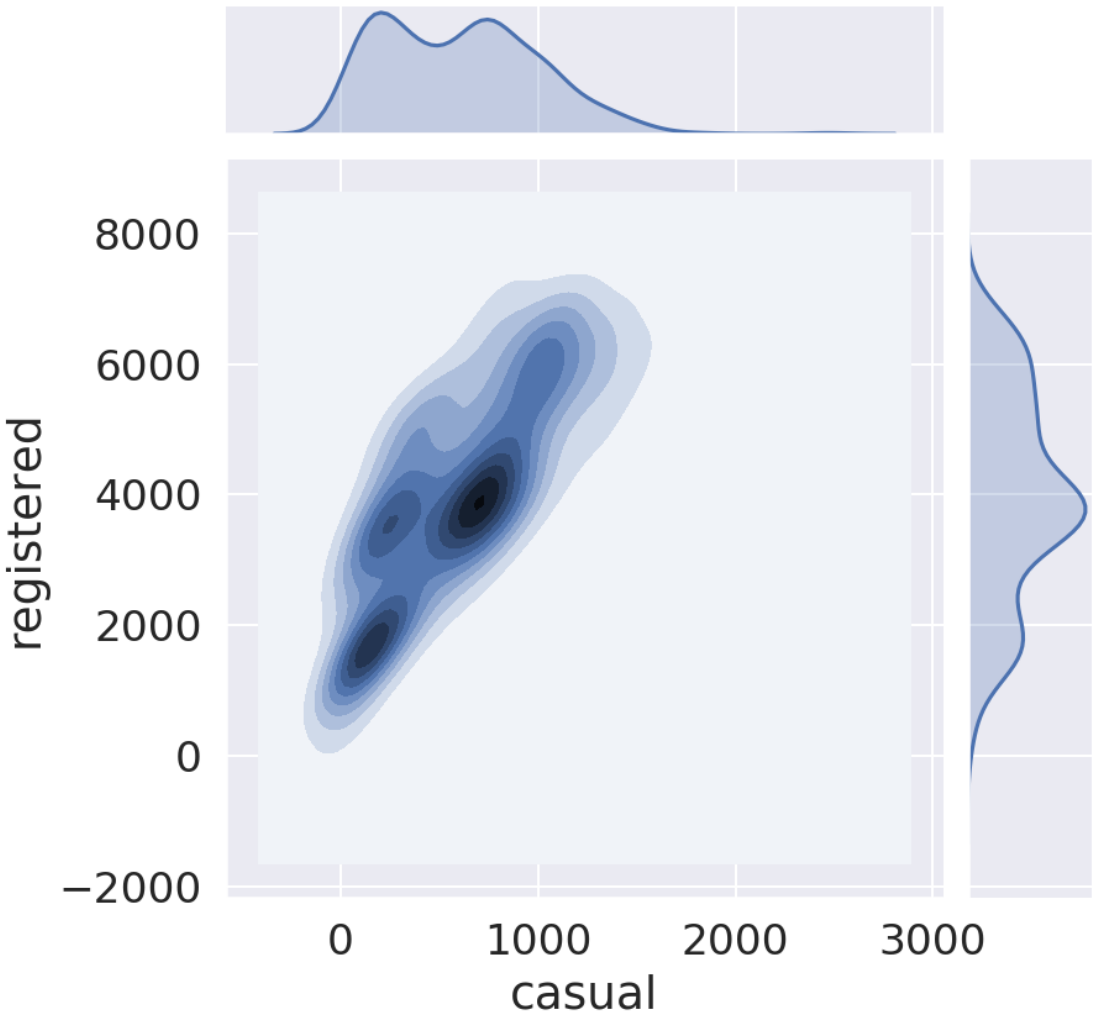
0.1 4: Joint Plot

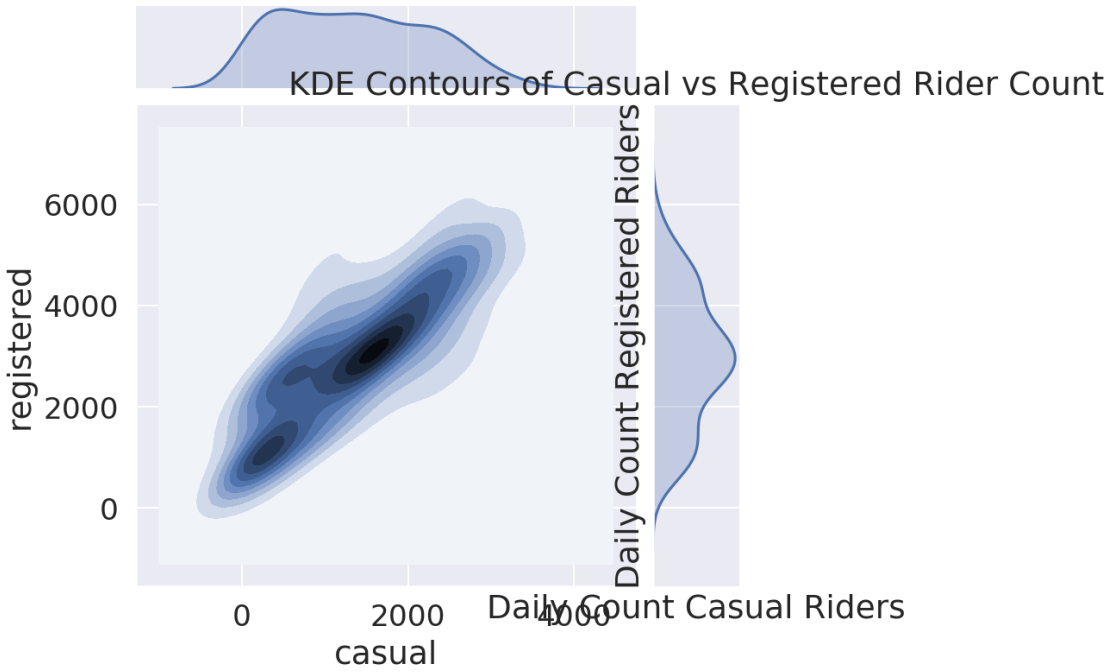
As an alternative approach to visualizing the data, construct the following set of three plots where the main plot shows the contours of the kernel density estimate of daily counts for registered and casual riders plotted together, and the two "margin" plots (at the top and right of the figure) provide 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. * `plt.suptitle` from lab 1 can be handy for setting the title where you want. * `plt.subplots_adjust(top=0.9)` can help if your title overlaps with your plot

We do not expect you to match our colors exactly, but the colors you choose should not distract from the information your plot conveys!

In [163]:
```
sns.jointplot(casual_workday,registered_workday,kind="kde");
sns.jointplot(casual_non_workday,registered_non_workday,kind="kde");

plt.title('KDE Contours of Casual vs Registered Rider Count');
plt.xlabel("Daily Count Casual Riders")
plt.ylabel("Daily Count Registered Riders")
plt.subplots_adjust(top=0.9)

plt.show()
```


8000
6000
4000
2000
0
−2000
registered
0
1000
2000
3000
casual


KDE Contours of Casual vs Registered Rider Count
registered
6000
4000
2000
0
0
2000
4000
casual
Daily Count Registered Riders
Daily Count Casual Riders

0.2 5: Understanding Daily Patterns

0.2.1 Question 5

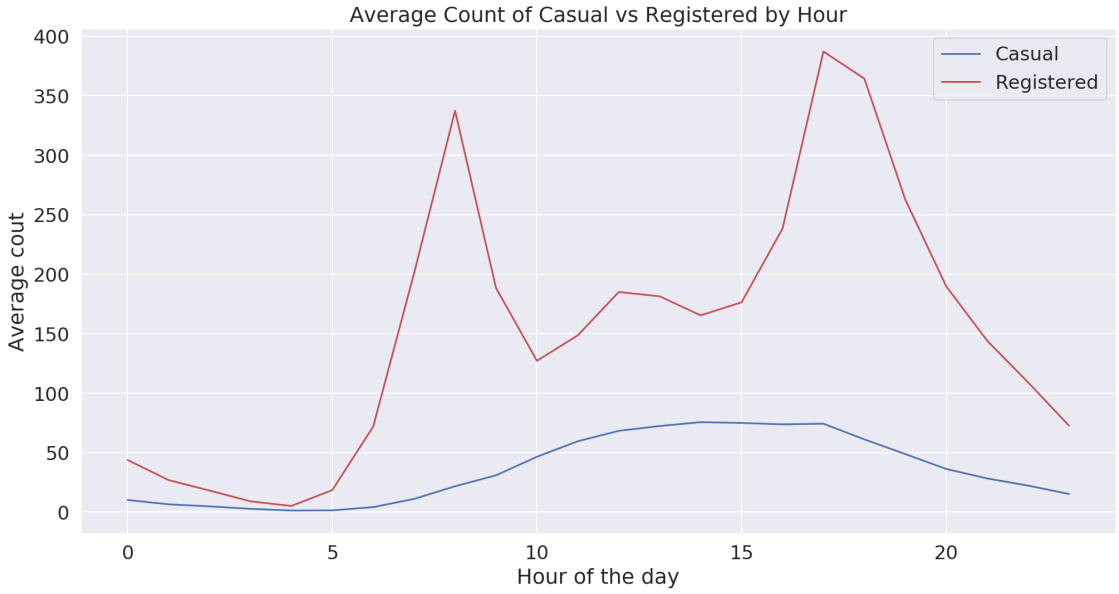
Question 5a Let's examine the behavior of riders by plotting the average number of riders for each hour of the day over the **entire dataset**, stratified by rider type.

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 [164]: Q5_c0=np.mean(bike[bike['hr']==0]['casual'])
          Q5_c1=np.mean(bike[bike['hr']==1]['casual'])
          Q5_c2=np.mean(bike[bike['hr']==2]['casual'])
          Q5_c3=np.mean(bike[bike['hr']==3]['casual'])
          Q5_c4=np.mean(bike[bike['hr']==4]['casual'])
          Q5_c5=np.mean(bike[bike['hr']==5]['casual'])
          Q5_c6=np.mean(bike[bike['hr']==6]['casual'])
          Q5_c7=np.mean(bike[bike['hr']==7]['casual'])
          Q5_c8=np.mean(bike[bike['hr']==8]['casual'])
          Q5_c9=np.mean(bike[bike['hr']==9]['casual'])
          Q5_c10=np.mean(bike[bike['hr']==10]['casual'])
          Q5_c11=np.mean(bike[bike['hr']==11]['casual'])
          Q5_c12=np.mean(bike[bike['hr']==12]['casual'])
          Q5_c13=np.mean(bike[bike['hr']==13]['casual'])
          Q5_c14=np.mean(bike[bike['hr']==14]['casual'])
          Q5_c15=np.mean(bike[bike['hr']==15]['casual'])
          Q5_c16=np.mean(bike[bike['hr']==16]['casual'])
          Q5_c17=np.mean(bike[bike['hr']==17]['casual'])
          Q5_c18=np.mean(bike[bike['hr']==18]['casual'])
          Q5_c19=np.mean(bike[bike['hr']==19]['casual'])
          Q5_c20=np.mean(bike[bike['hr']==20]['casual'])
          Q5_c21=np.mean(bike[bike['hr']==21]['casual'])
          Q5_c22=np.mean(bike[bike['hr']==22]['casual'])
          Q5_c23=np.mean(bike[bike['hr']==23]['casual'])


          Q5_r0=np.mean(bike[bike['hr']==0]['registered'])
          Q5_r1=np.mean(bike[bike['hr']==1]['registered'])
          Q5_r2=np.mean(bike[bike['hr']==2]['registered'])
          Q5_r3=np.mean(bike[bike['hr']==3]['registered'])
          Q5_r4=np.mean(bike[bike['hr']==4]['registered'])
          Q5_r5=np.mean(bike[bike['hr']==5]['registered'])
          Q5_r6=np.mean(bike[bike['hr']==6]['registered'])
          Q5_r7=np.mean(bike[bike['hr']==7]['registered'])
          Q5_r8=np.mean(bike[bike['hr']==8]['registered'])
          Q5_r9=np.mean(bike[bike['hr']==9]['registered'])
          Q5_r10=np.mean(bike[bike['hr']==10]['registered'])
          Q5_r11=np.mean(bike[bike['hr']==11]['registered'])
          Q5_r12=np.mean(bike[bike['hr']==12]['registered'])
          Q5_r13=np.mean(bike[bike['hr']==13]['registered'])
          Q5_r14=np.mean(bike[bike['hr']==14]['registered'])
          Q5_r15=np.mean(bike[bike['hr']==15]['registered'])
          Q5_r16=np.mean(bike[bike['hr']==16]['registered'])
          Q5_r17=np.mean(bike[bike['hr']==17]['registered'])
```

```
Q5_r18=np.mean(bike[bike['hr']==18]['registered'])
Q5_r19=np.mean(bike[bike['hr']==19]['registered'])
Q5_r20=np.mean(bike[bike['hr']==20]['registered'])
Q5_r21=np.mean(bike[bike['hr']==21]['registered'])
Q5_r22=np.mean(bike[bike['hr']==22]['registered'])
Q5_r23=np.mean(bike[bike['hr']==23]['registered'])

plt.plot([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23],[Q5_c0,Q5_c1,Q5_c2,Q5
plt.plot([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23],[Q5_r0,Q5_r1,Q5_r2,Q5

plt.title('Average Count of Casual vs Registered by Hour');
plt.xlabel("Hour of the day")
plt.ylabel("Average cout")
plt.legend(['Casual','Registered'])
```

Out[164]: <matplotlib.legend.Legend at 0x7fd5b7c7af60>



Question 5b What can you observe from the plot? Hypothesize about the meaning of the peaks in the registered riders' distribution.

Looking at the plots for the registered riders, I see that there are peaks when the hour is 8 and when the hour is 17. This makes sense because this is during rush hour times when it is 8 am and 5 pm. Looking at the plot for casual riders, we see that they are highest during the afternoon, from 10am to 5 pm.

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.

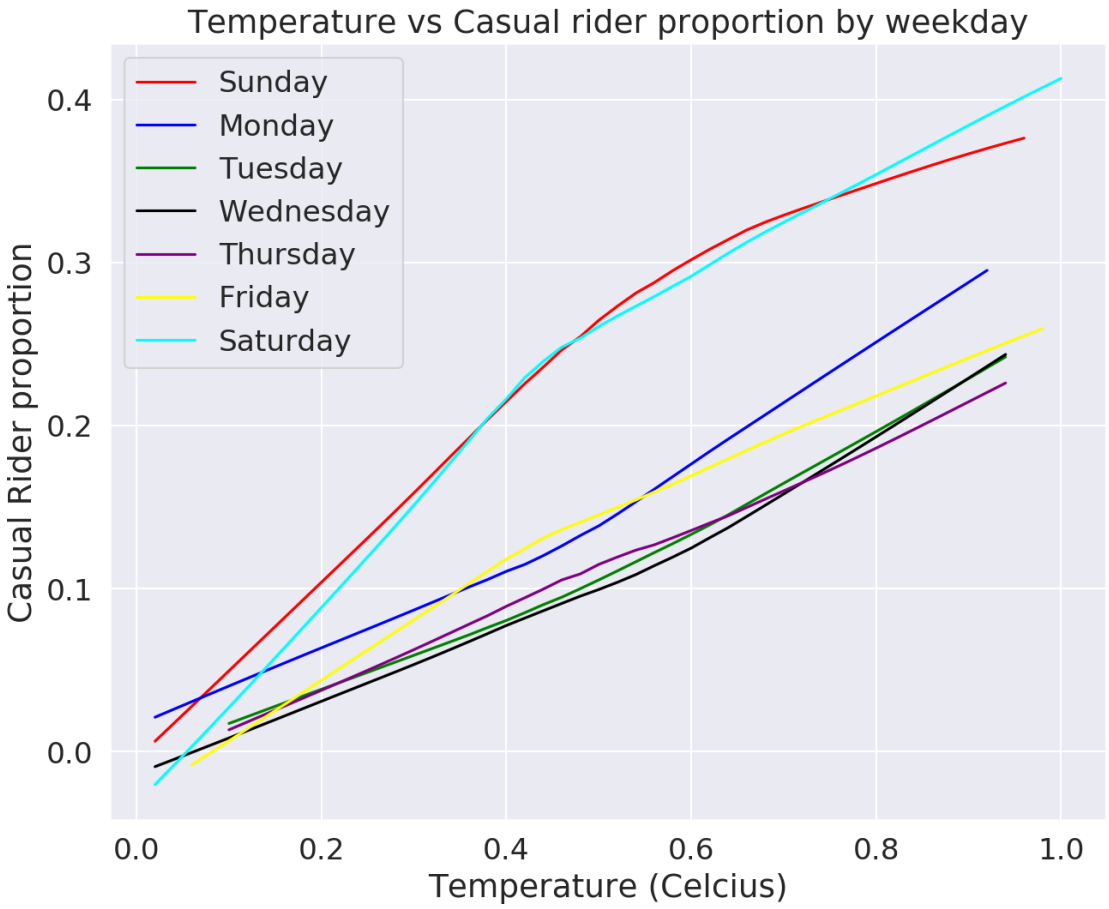
You do not need to match the colors on our sample plot as long as the colors in your plot make it easy to distinguish which day they represent.

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.
- 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, Fahrenheit = Celsius $* \frac{9}{5} + 32$.

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

In [190]:
```
from statsmodels.nonparametric.smoothers_lowess import lowess

plt.figure(figsize=(10,8))

Sunday = bike[bike['weekday']=='Sun']
ysmoothSun = lowess(Sunday['prop_casual'], Sunday['temp'], return_sorted=False)
sns.lineplot(Sunday['temp'], ysmoothSun, label="Sunday", color='red')

Monday = bike[bike['weekday']=='Mon']
ysmoothMon = lowess(Monday['prop_casual'], Monday['temp'], return_sorted=False)
sns.lineplot(Monday['temp'], ysmoothMon, label="Monday", color='blue')

Tuesday = bike[bike['weekday']=='Tue']
ysmoothTue = lowess(Tuesday['prop_casual'], Tuesday['temp'], return_sorted=False)
sns.lineplot(Tuesday['temp'], ysmoothTue, label="Tuesday", color='green')

Wednesday = bike[bike['weekday']=='Wed']
ysmoothWed = lowess(Wednesday['prop_casual'], Wednesday['temp'], return_sorted=False)
sns.lineplot(Wednesday['temp'], ysmoothWed, label="Wednesday", color='black')

Thursday = bike[bike['weekday']=='Thu']
ysmoothThu = lowess(Thursday['prop_casual'], Thursday['temp'], return_sorted=False)
sns.lineplot(Thursday['temp'], ysmoothThu, label="Thursday", color='purple')

Friday = bike[bike['weekday']=='Fri']
ysmoothFri = lowess(Friday['prop_casual'], Friday['temp'], return_sorted=False)
sns.lineplot(Friday['temp'], ysmoothFri, label="Friday", color='yellow')

Saturday = bike[bike['weekday']=='Sat']
ysmoothSat = lowess(Saturday['prop_casual'], Saturday['temp'], return_sorted=False)
sns.lineplot(Saturday['temp'], ysmoothSat, label="Saturday", color='cyan')

plt.title('Temperature vs Casual rider proportion by weekday');
plt.xlabel("Temperature (Celcius)")
plt.ylabel("Casual Rider proportion")

plt.legend();
```


Temperature vs Casual rider proportion by weekday
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Casual Rider proportion
0.0
0.1
0.2
0.3
0.4
0.0
0.2
0.4
0.6
0.8
1.0
Temperature (Celcius)

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?

Looking at the curve plot, we see that as temperature increases, prop_casual increases. I guess this would make sense as more people would use bikes in warmer weather. Looking at the curves, we also see that bikesharing is more popular in the weekends(Saturday and Sunday).

0.2.2 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.

Write your answer here, replacing this text.

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!

Write your answer here, replacing this text.