0.0.1 Question 0

Question 0A What is the granularity of the data (i.e. what does each row represent)?

Each row represents a single bike share. It also gives some info regarding the rental. The main giveaway that each row represents a single bike rental is the dteday column. There are multiple on the same day, indicating that each row must represent something more specific than just 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?

We don't know the specifics of certain things, like how long each bike was rented for, where the bike traveled to, and where bikes are rented/dropped off. If we want to study bike usage, these things, in addition to demographics of who is using these bike shares, would be useful.Two additional catergories that would be particularly helpful is where bikes are piked up/dropped off, and the age of the person renting the bike.

0.0.2 Question 2

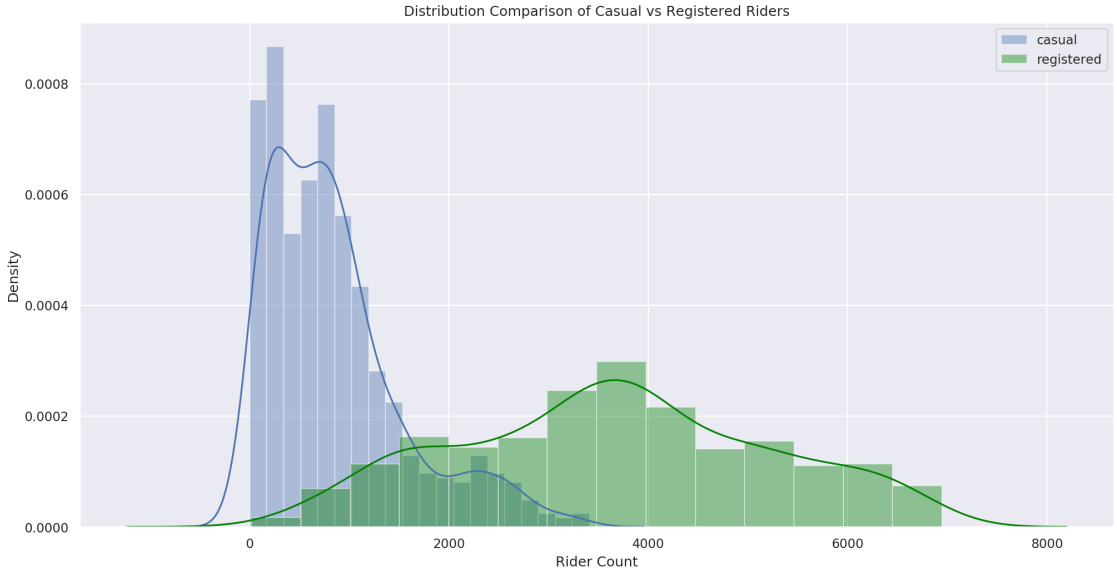
Question 2a Use the `sns.distplot` function to create a plot that overlays the distribution of the daily counts of bike users, using blue to represent `casual` riders, and green to represent `registered` riders. The temporal granularity of the records should be daily counts, which you should have after completing question 1c. **You can ignore all warnings that say `distplot` is a deprecated function.**

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 [45]:
```
plt.figure()
sns.distplot(daily_counts['casual'], label = 'casual')
sns.distplot(daily_counts['registered'], label = 'registered', color = 'green')
plt.xlabel("Rider Count")
plt.ylabel("Density")
plt.title('Distribution Comparison of Casual vs Registered Riders')
plt.legend()
```

```
/opt/conda/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support for mul
  ndim = x[:, None].ndim
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for multi-di
  x = x[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for multi-di
  y = y[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support for mul
  ndim = x[:, None].ndim
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for multi-di
  x = x[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for multi-di
  y = y[:, np.newaxis]
```

Out[45]: `<matplotlib.legend.Legend at 0x7f6b42957280>`


Distribution Comparison of Casual vs Registered Riders
casual
registered
Density
0.0000
0.0002
0.0004
0.0006
0.0008
0
2000
4000
6000
8000
Rider Count

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.

The density curve for casual riders is weakly bimodal, with peaks around 250 and 500. It is skewed right with a medium sized tail, and thus not symmtric. There don't appear to be outliers or gaps in the data. As for the registered rider density curve, it is unimodal centered around 4000 and roughly symmetric and thus not skewed. There appear to be no outliers or gaps. Compared to each other, the registered rider curve has a large spread and looks much more normal than the casual rider curve.

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 [46]:
```
# Make the font size a bit bigger
sns.set(font_scale=1)
sns.lmplot(x = 'casual', y = 'registered', hue = 'workingday', data = bike, fit_reg = True, sc
plt.title('Comparison of Casual vs Registered Riders on Wokring and Non-working Days')
```

Out[46]:
```
Text(0.5, 1, 'Comparison of Casual vs Registered Riders on Wokring and Non-working Days')
```


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

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?

There appear to be a linear relationship between the counts of casual and registered riders, and this relationship is dependent on whether or not it is a working day. The overplotting makes it difficult to assess the relationship as the bottom left corner is so saturated with data points that its hard to make anything out. Also, it seems the working day points in orange are always plotted above the nonworking day points in blue, so the blue points are hard to discern due to the overplotting.

Generating the plot with weekend and weekday separated can be complicated so we will provide a walk-through 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. * 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). You are required for this question to use two sets of contrasting colors for your plots.

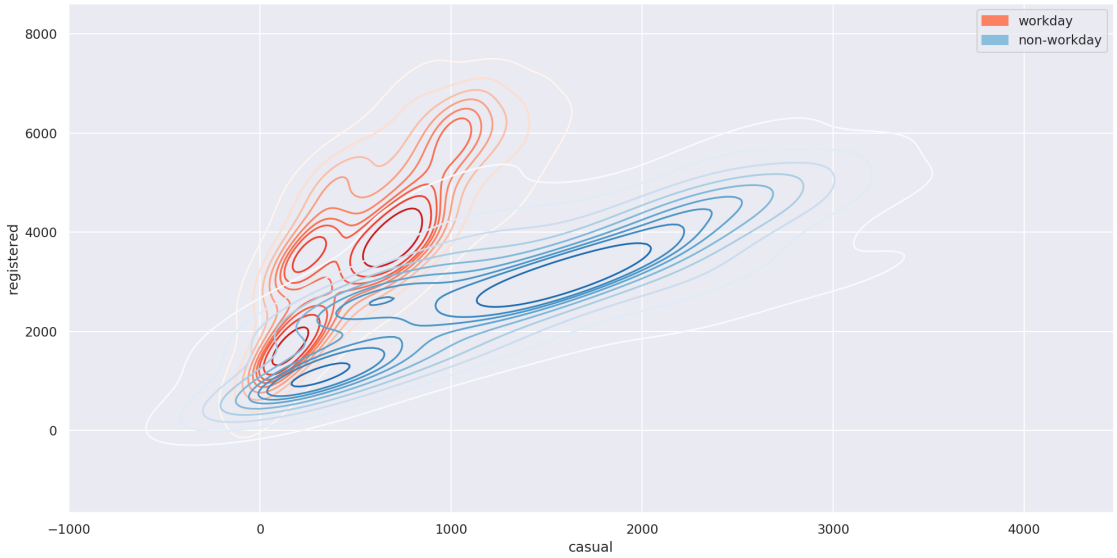
After you get your plot working, experiment by setting `shade=True` in `kdeplot` to see the difference between the shaded and unshaded version. Please submit your work with `shade=False`.

In [ ]:

In [48]:
```
import matplotlib.patches as mpatches

# Set 'is_workingday' to a boolean array that is true for all working_days
is_workingday = 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 = daily_counts.loc[is_workingday, 'casual']
registered_workday = daily_counts.loc[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')

# Repeat the same steps above but for rows corresponding to non-workingdays
not_workingday = ~is_workingday
casual_non_workday = daily_counts.loc[not_workingday, 'casual']
registered_non_workday = daily_counts.loc[not_workingday, 'registered']



# Use sns.kdeplot on the two variables above to plot the bivariate KDE for non-workingday ride
sns.kdeplot(casual_non_workday, registered_non_workday, cmap = 'Blues')

plt.xlabel('casual')
plt.ylabel('registered')
red = mpatches.Patch(color = sns.color_palette('Reds')[2], label = 'workday')
blue = mpatches.Patch(color = sns.color_palette('Blues')[2], label = 'non-workday')
plt.legend(handles = [red, blue])
```

```
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
```

Out[48]: <matplotlib.legend.Legend at 0x7f6b42a60fa0>



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

The association between casual and registered riders is linear, which was more difficult to tell on the scatterplot. The variability is higher for non-workdays than workdays. The non-workday distribution is bimodal while the workday distribution appears to be trimodal.

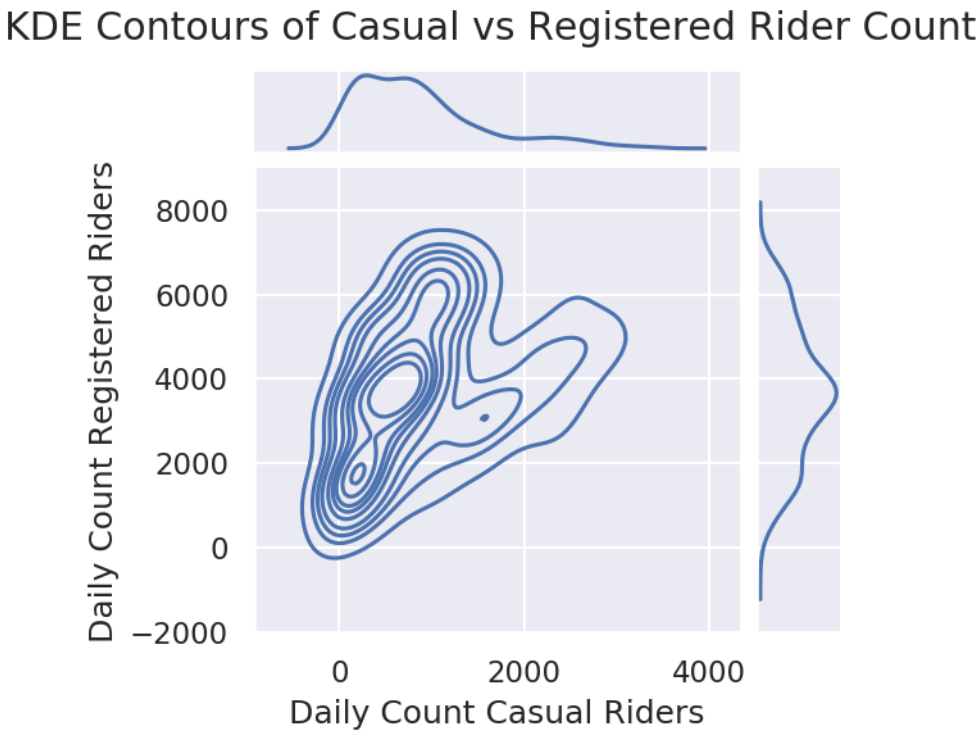
0.1 4: Joint Plot

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

In [57]:
```
joint_plot = sns.jointplot(x = "casual", y = "registered", data = daily_counts, kind = "kde", l
joint_plot.set_axis_labels("Daily Count Casual Riders", "Daily Count Registered Riders")
plt.suptitle("KDE Contours of Casual vs Registered Rider Count")
plt.subplots_adjust(top = 0.9)
```

```
/opt/conda/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support for mul
  ndim = x[:, None].ndim
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for multi-di
  x = x[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for multi-di
  y = y[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support for mul
  ndim = x[:, None].ndim
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for multi-di
  x = x[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for multi-di
  y = y[:, np.newaxis]
```


KDE Contours of Casual vs Registered Rider Count
Daily Count Registered Riders
8000
6000
4000
2000
0
−2000
0
2000
4000
Daily Count Casual Riders

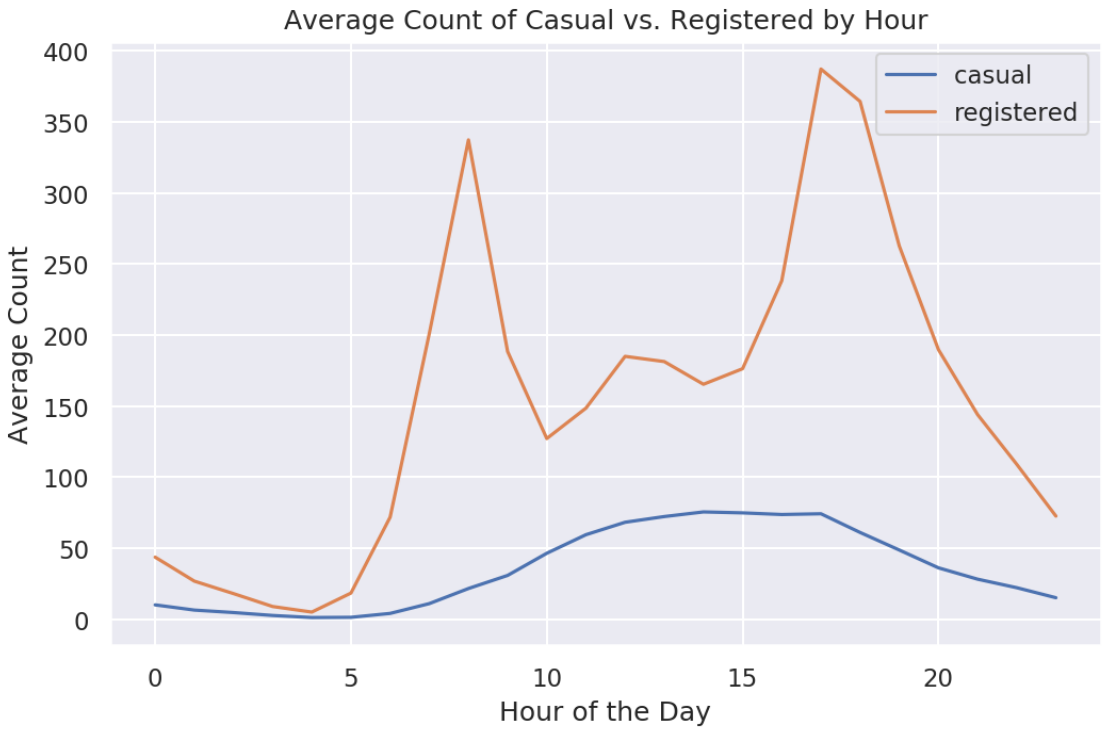
0.2 5: Understanding Daily Patterns

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 [60]: plt.figure(figsize = [8, 5])
         hour_means = bike.groupby('hr').mean()
         sns.lineplot(x = hour_means.index, y = hour_means['casual'], label = 'casual ')
         sns.lineplot(x = hour_means.index, y = hour_means['registered'], label = 'registered')
         plt.xlabel('Hour of the Day')
         plt.ylabel('Average Count')
         plt.title('Average Count of Casual vs. Registered by Hour');
```



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

Both casual and registered riders have patterns of usage. Casual riders tend to ride the most in the afternoon, around 3 pm, while registered riders ride the most around 8 am and 5 pm. However, there is still significant usage for registered riders in the afternoon, especially around 12 pm. This could follow the morning and evening commute hours, as well as lunch breaks.

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.

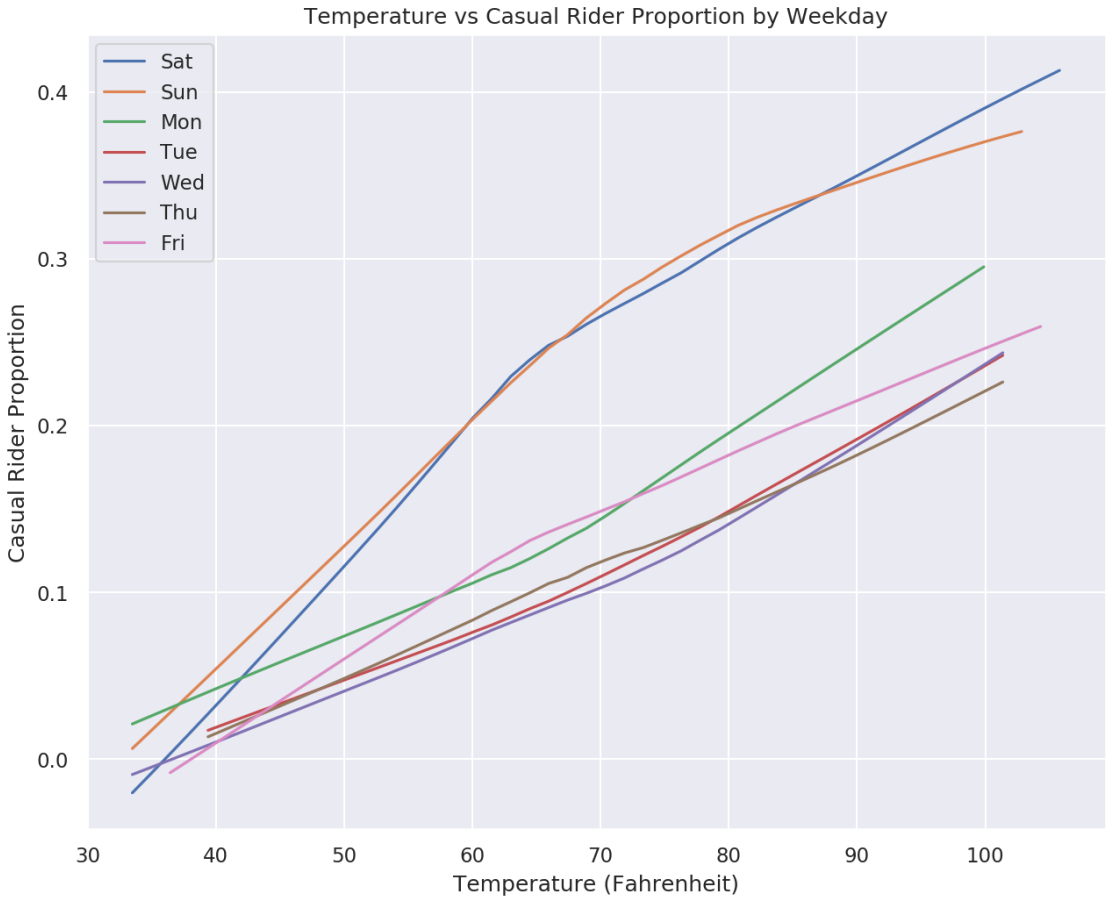
- The `lowess` function expects y coordinate first, then x coordinate.
- Look at the top of this homework notebook for a description of the temperature field to know how to convert to Fahrenheit. By default, the temperature field ranges from 0.0 to 1.0. In case you need it, Fahrenheit = Celsius $* \frac{9}{5} + 32$.

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

In [71]:
```
from statsmodels.nonparametric.smoothers_lowess import lowess

        plt.figure(figsize=(10,8))
        for day in bike['weekday'].unique():
            each_day = bike[bike['weekday'] == day]
            each_day['temp'] = each_day['temp'] * 41 * 9 / 5 + 32
            smooth = lowess(each_day['prop_casual'], each_day['temp'], return_sorted = False)
            sns.lineplot(each_day['temp'], smooth, label = day)


        plt.title("Temperature vs Casual Rider Proportion by Weekday")
        plt.xlabel("Temperature (Fahrenheit)")
        plt.ylabel("Casual Rider Proportion")
        plt.legend();
```

```
<ipython-input-71-7b9accb6252e>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.l
  each_day['temp'] = each_day['temp'] * 41 * 9 / 5 + 32
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
<ipython-input-71-7b9accb6252e>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
  each_day['temp'] = each_day['temp'] * 41 * 9 / 5 + 32
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
<ipython-input-71-7b9accb6252e>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
  each_day['temp'] = each_day['temp'] * 41 * 9 / 5 + 32
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
<ipython-input-71-7b9accb6252e>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
  each_day['temp'] = each_day['temp'] * 41 * 9 / 5 + 32
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
<ipython-input-71-7b9accb6252e>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
  each_day['temp'] = each_day['temp'] * 41 * 9 / 5 + 32
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
<ipython-input-71-7b9accb6252e>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
  each_day['temp'] = each_day['temp'] * 41 * 9 / 5 + 32
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
<ipython-input-71-7b9accb6252e>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
  each_day['temp'] = each_day['temp'] * 41 * 9 / 5 + 32
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following vari
  warnings.warn(
```


Temperature vs Casual Rider Proportion by Weekday
Sat
Sun
Mon
Tue
Wed
Thu
Fri
Casual Rider Proportion
0.0
0.1
0.2
0.3
0.4
30
40
50
60
70
80
90
100
Temperature (Fahrenheit)

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?

As temperatures increase, so do the proportion of casual riders, even for really hot temperatures (> 90). Weekends have a higher proportion of casual ridrs than weekdays. There appears to be 3 distinct groups: weekends, Monday and Friday, and midweek days, although Friday starts to overlap with midweek ways at higher temperatures.

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.

No, the bike data can't help assess equity. There is no information about socio-economic classes, genders, races, and neighborhoods in the dataset, nor anything about price. We only have data about the details of the rental, time, season, weather, etc. and whether or not the rider is registered or casual. None of this information is particularly useful in assessing the equity of the ride share service. I would change the data set to include where the bike was picked up and dropped off, as well as demographic data about the person who rented it, such as race, gender, and income. The price of the rental should also be included. This information would give us a picture of who is using the service in terms of socio-economic class, gender and race, as well as the areas in which the service is and isn't being used. This would give enough information to assess equity in some capacity.

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!

I would recommend expanding the bike sharing service to additional cities. The first reason being that it appears people do use bike sharing for commuting. In the graph titled "Average Count of Casual vs. Registered by Hour", we see registered riders using the service most around 8 am, 12 pm, and 5 pm, which coincides with the morning commute, lunch breaks, and eevening commutes. This points to people actually using the service as an alternative mode transportation like it was intended, helping with congestion and pollution. The second reason