5.3.6 Create a Bubble Chart for All Cities

Now for the final piece! You decide that instead of showing each bubble chart separately in your presentation, you'll show one bubble chart that that has all the city types. Given that the different city types have different colors, this will make your bubble chart stand out and show how each city type compares with one another.

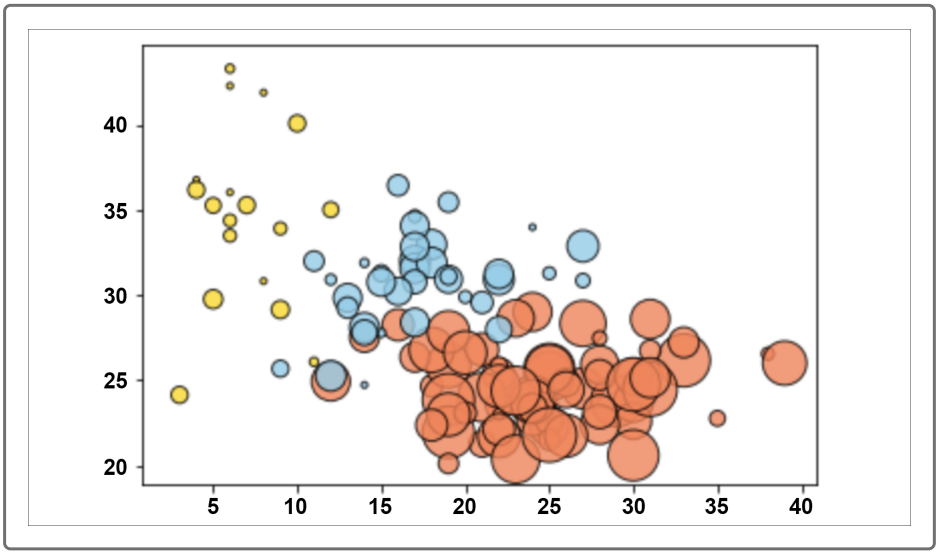
To create a bubble chart that showcases all the different city types in one chart, we'll combine our three scatter plot code blocks in one Jupyter Notebook cell.

Add the three `plt.scatter()` functions for each chart to one cell and run the cell.

```
# Add the scatter charts for each type of city.
plt.scatter(urban_ride_count,
      urban_avg_fare,
      s=10*urban_driver_count, c="coral",
      edgecolor="black", linewidths=1,
      alpha=0.8, label="Urban")

plt.scatter(suburban_ride_count,
      suburban_avg_fare,
      s=10*suburban_driver_count, c="skyblue",
      edgecolor="black", linewidths=1,
      alpha=0.8, label="Suburban")

plt.scatter(rural_ride_count,
      rural_avg_fare,
      s=10*rural_driver_count, c="gold",
      edgecolor="black", linewidths=1,
```

```
      alpha=0.8, label="Rural")

# Show the plot
plt.show()
```

In the output window, behold the (beautiful!) chart created from this code.



Did you notice that we did not have to change the x-limit? That's because plotting all the data on one chart formats the x-axis automatically. We could change the y-limit from 0 to 40, but that might crowd the bubbles in the middle of the chart, making it harder to see any differences in the data.

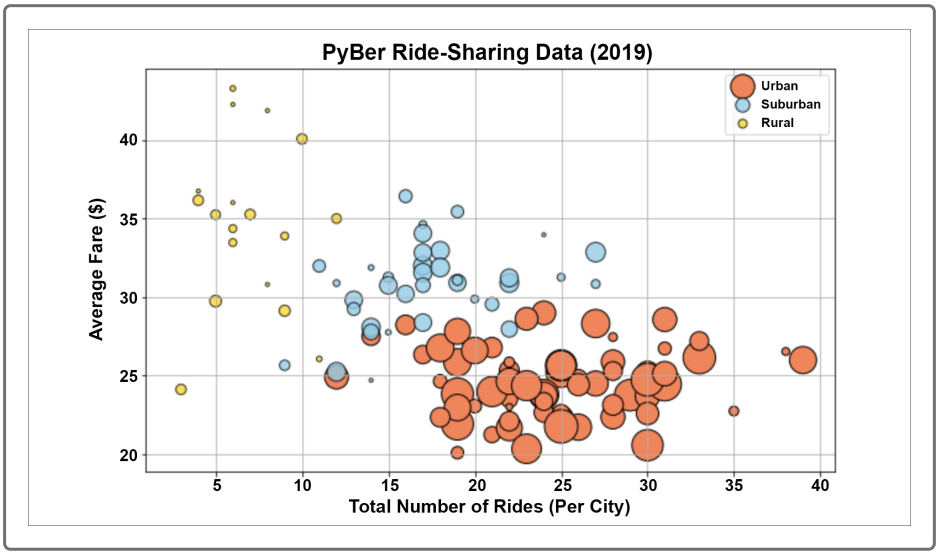
Let's add a title, labels for the axes, a legend, and a grid for all three charts and increase the font size of the axes labels to 12 and the title to 20. We'll also enlarge the figure so the markers are more spread out.

Edit the existing code to look like this.

```
# Build the scatter charts for each city type.
plt.subplots(figsize=(10, 6))
plt.scatter(urban_ride_count,
      urban_avg_fare,
      s=10*urban_driver_count, c="coral",
      edgecolor="black", linewidths=1,
      alpha=0.8, label="Urban")

plt.scatter(suburban_ride_count,
      suburban_avg_fare,
      s=10*suburban_driver_count, c="skyblue",
      edgecolor="black", linewidths=1,
      alpha=0.8, label="Suburban")
```

```
plt.scatter(rural_ride_count,
      rural_avg_fare,
      s=10*rural_driver_count, c="gold",
      edgecolor="black", linewidths=1,
      alpha=0.8, label="Rural")

# Incorporate the other graph properties
plt.title("PyBer Ride-Sharing Data (2019)", fontsize=20)
plt.ylabel("Average Fare ($)", fontsize=12)
plt.xlabel("Total Number of Rides (Per City)", fontsize=12)
plt.grid(True)

# Add the legend.
plt.legend()
# Show the plot
plt.show()
```

Next, run the cell, and in the output window you'll see this updated chart.



Did you notice something that we need to fix? The markers in the legend have different sizes, which are automatically determined based on the average size of the marker. Therefore, we'll need to customize the legend to scale them to the same size.

We can declare a variable for the legend function, `lgnd = plt.legend()`, and add parameters for font size, legend location, and legend title, along with some other features.

After we declare the variable for the legend, we can use `legendHandles[]._sizes` to set the font size of the marker in the legend to a

fixed size. Inside the brackets, we can add the list element for the number of markers in the legend.

Add the following code to your scatter plot code in place of `plt.legend()`:

```
# Create a legend
lgnd = plt.legend(fontsize="12", mode="Expanded",
         scatterpoints=1, loc="best", title="City Types")
lgnd.legendHandles[0]._sizes = [75]
lgnd.legendHandles[1]._sizes = [75]
lgnd.legendHandles[2]._sizes = [75]
lgnd.get_title().set_fontsize(12)
```

Let's break down what this code is doing for the legend:

1. We made the font size for the text "small" with `fontsize=`.
2. Then we expanded the legend horizontally using `mode=` to fit the area. Because the font size is small, this is optional.
3. We added the number of scatter points in the legend for each marker to be 1. We can add multiple marker points by increasing the number.
4. The location setting, `loc=`, for the legend is where it will fit the "best" based on the plotting of the data points.
5. We added a legend title.
6. We set each marker in the legend equal to 75-point font by using the `legendHandles[]._sizes` attribute and list indexing in the brackets to reference one of the three scatter plots.
7. Finally, we increased the font size of the legend title to 12.

Next, we need to add a note to the right of the chart to let the viewer know that the circle size correlates with the driver count for each city. Our note will say: "Note: Circle size correlates with driver count per city." To do this, we'll use the `plt.text()` function and add the text. Inside the function, we add the x and y coordinates for the chart and the text in quotes.

The x and y coordinates are based on the chart coordinates. We can see that our chart has a width (i.e., x, between 0 and 42; and y, between 18 and 50). Our x position can be 42, and the y position can be in the middle, 32–35.

Add the following code after the code that creates the legend:

```
# Incorporate a text label about circle size.
plt.text(42, 35, "Note: Circle size correlates with driver count per city.",
```

Finally, we need to save the chart in a folder using the `plt.savefig()` function, and provide a direct path to the folder and filename of the saved image.

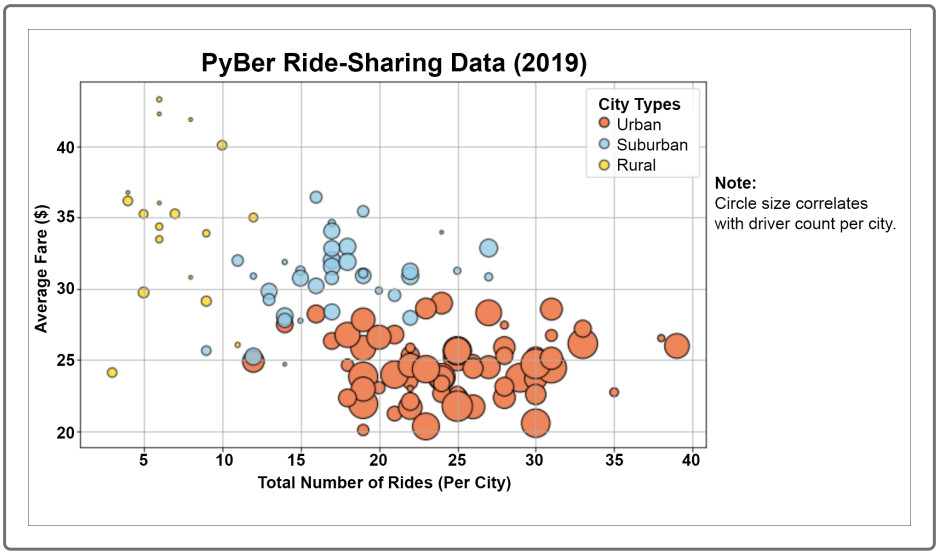
Add the following code after the code that added the text:

```
# Save the figure.
plt.savefig("analysis/Fig1.png")
```

IMPORTANT

> Before we run the whole code block, make sure you create a folder named "analysis" in the PyBer_Analysis folder.

After we run the cell, our final chart will look like this:



Congratulations on creating the ride-sharing bubble chart!

NOTE

> For further information on creating legends and adding text to charts, see the following documentation:

Matplotlib documentation on matplotlib.pyplot.legend (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.legend.html)

Matplotlib documentation on matplotlib.axes.Axes.legend (https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.legend.html)

Matplotlib documentation on matplotlib.pyplot.text (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.text.html)