Use the `head` command on your three files again. This time, describe at least one potential problem with the data you see. Consider issues with missing values and bad data.

One potential problem I noticed was that there was some kind of data input that represents no value to a human but can be confusing when programming. For example, instead of NaN, there are -9999 values for latitude and longitude which, to me, indicates that there is no data existing for that point and that -9999 is simply a placeholder, but it makes it harder because a computer cannot automatically understand that.

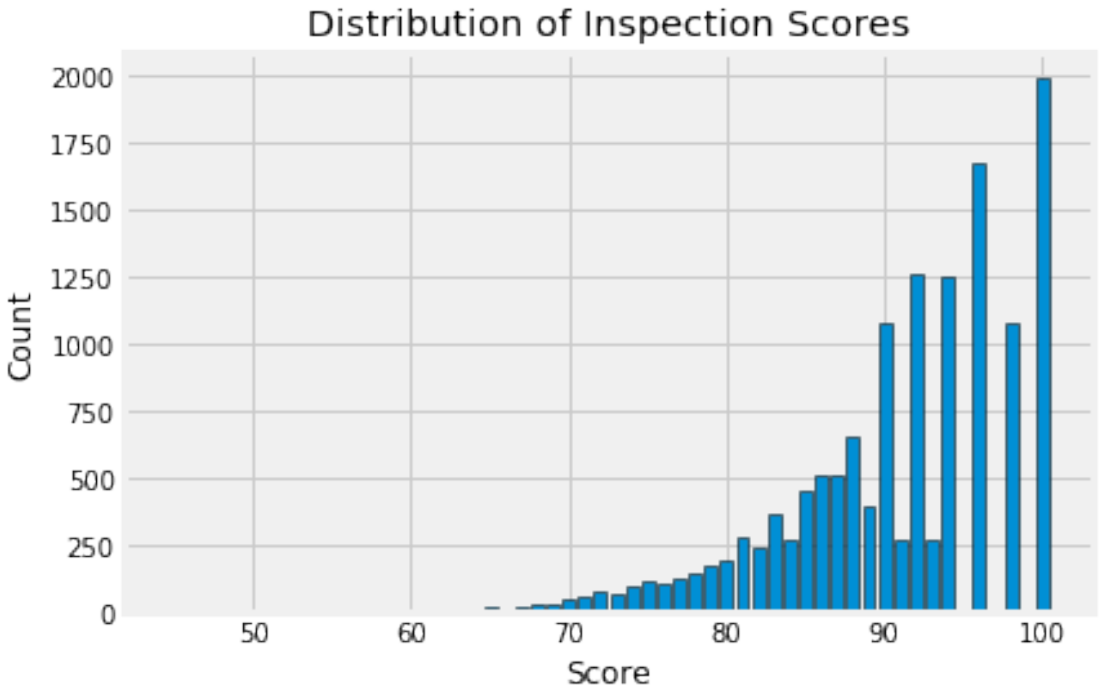
In the cell below, write the name of the restaurant with the lowest inspection scores ever. You can also head to yelp.com and look up the reviews page for this restaurant. Feel free to add anything interesting you want to share.

Lollipot has the lowest inspection score at 45. According to YELP, Lollipot is officially closed, but had 3.5 stars, which, all things considering, is quite impressive. There were also a lot of positive reviews, which indicates people really don't look at health inspection grades when choosing a place to dine at. Or maybe hotpot is just too tempting.

0.1 Question 6a

Let's look at the distribution of inspection scores. As we saw before when we called head on this data frame, inspection scores appear to be integer values. The discreteness of this variable means that we can use a barplot to visualize the distribution of the inspection score. Make a bar plot of the counts of the number of inspections receiving each score.

It should look like the image below. It does not need to look exactly the same (e.g., no grid), but make sure that all labels and axes are correct.



You might find this matplotlib.pyplot tutorial useful. Key syntax that you'll need:

```
plt.bar
plt.xlabel
plt.ylabel
plt.title
```

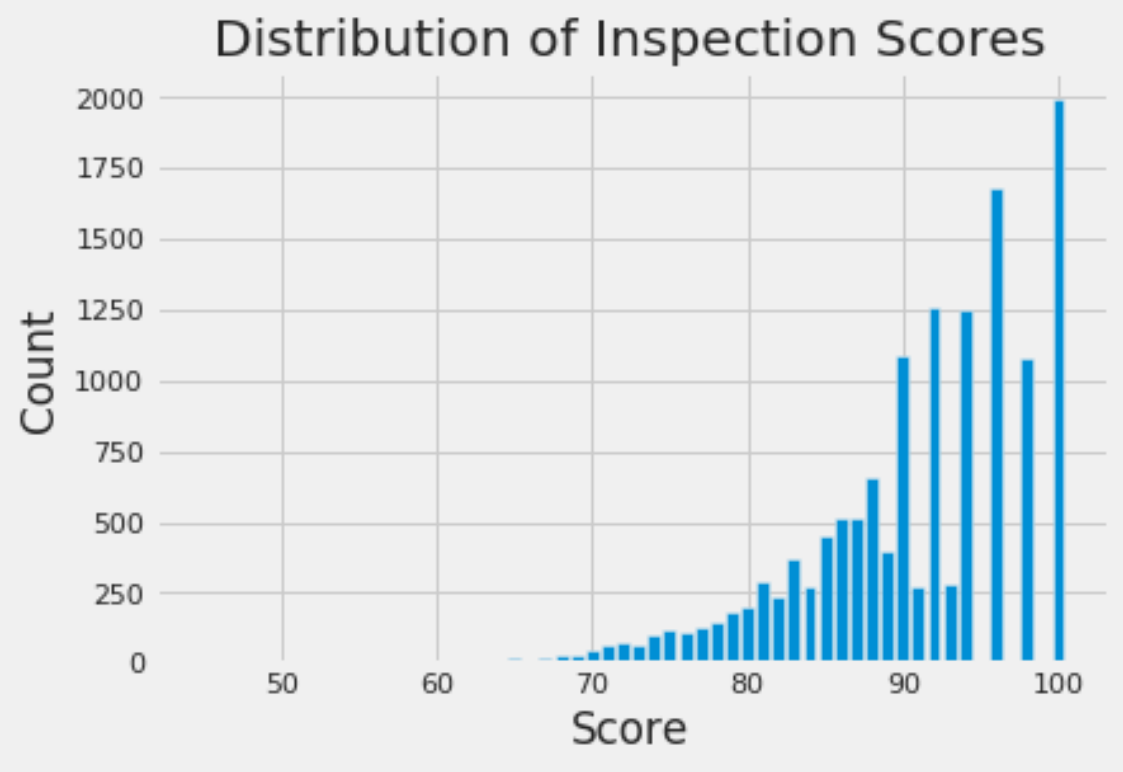
Note: If you want to use another plotting library for your plots (e.g. plotly, sns) you are welcome to use that library instead so long as it works on DataHub. If you use seaborn sns.countplot(), you may need to manually set what to display on xticks.

In [280]:
```
counted = ins[ins['score'] > 0].groupby('score').count()

x = counted.index
y = counted['iid']

plt.bar(x, y)
plt.xlabel('Score')
plt.ylabel('Count')
plt.title('Distribution of Inspection Scores')
```

Out[280]:
```
Text(0.5, 1.0, 'Distribution of Inspection Scores')
```



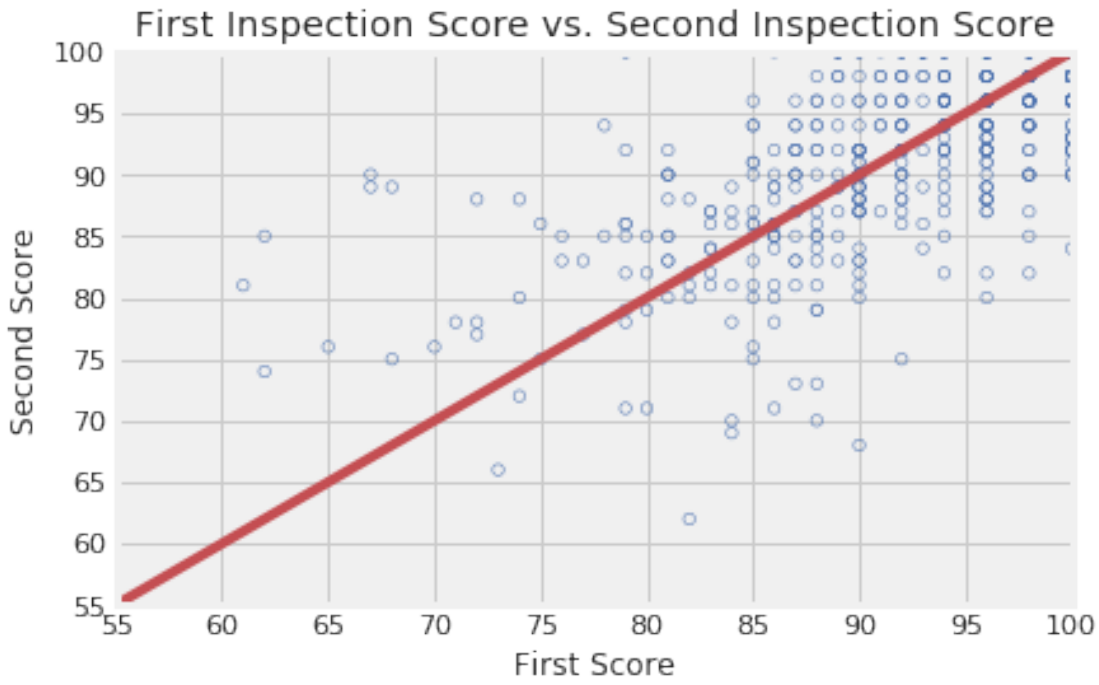
0.1.1 Question 6b

Describe the qualities of the distribution of the inspections scores based on your bar plot. Consider the mode(s), symmetry, tails, gaps, and anomalous values. Are there any unusual features of this distribution? What do your observations imply about the scores?

The immediate thing I notice is the general upward slope of the graph, indicating that more restaurants recieve higher scores than not. There is also a higher concentration between the scores of 70 and 90, after which it becomes more spread out. This indicates that, the higher the score, the more restaurants receive these scores BUT these higher scores aren't represented as often. This could be for a number of reasons, perhaps a nuance in the grading criteria that doesn't allow for partial points as scores get higher.

I also noticed the left tail, where scores are lower. I do think it's unexpected that this doesn't resemble a normal distribution, so that the highest concentration would be between 70 and 85. But I suppose for restauranteers, this is a positive signal as they can trust most restaurants they visit in Berkeley, at least.

Now, create your scatter plot in the cell below. It does not need to look exactly the same (e.g., no grid) as the sample below, but make sure that all labels, axes and data itself are correct.



Key pieces of syntax you'll need:

`plt.scatter` plots a set of points. Use `facecolors='none'` and `edgecolors=b` to make circle markers with blue borders.

`plt.plot` for the reference line.

`plt.xlabel`, `plt.ylabel`, `plt.axis`, and `plt.title`.

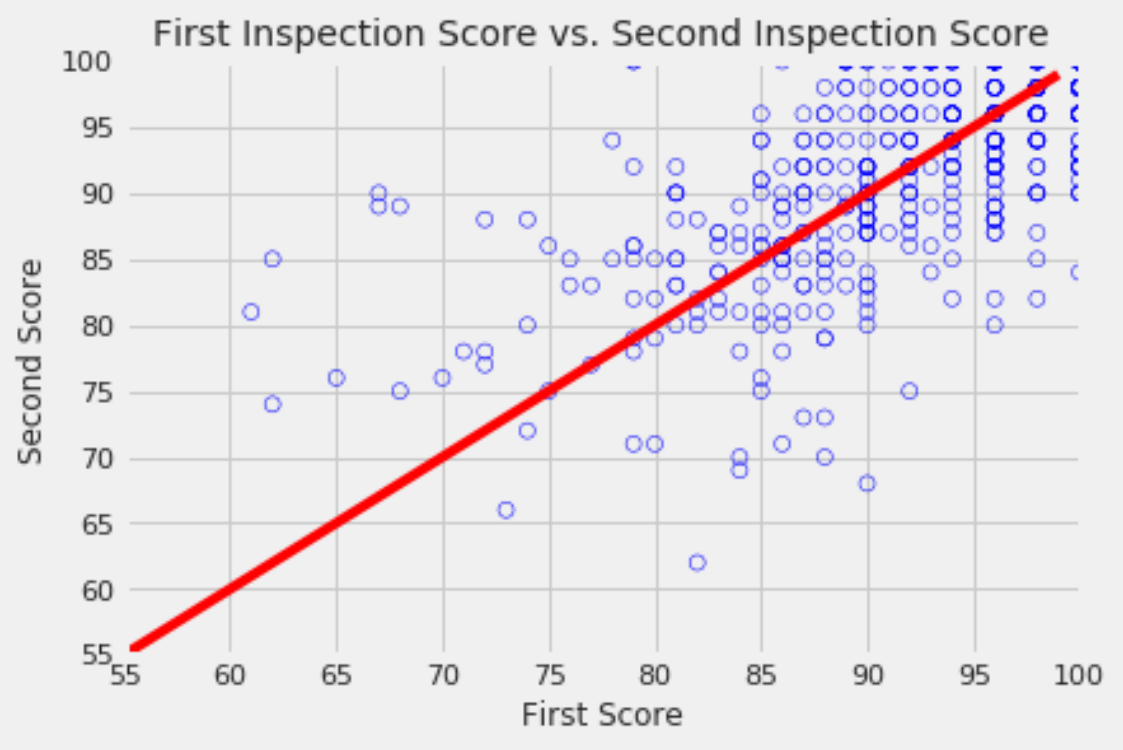
Hint: You may find it convenient to use the `zip()` function to unzip scores in the list.

```
In [289]: x = scores_pairs_by_business['score_pair'].map(lambda x: x[0])
          y = scores_pairs_by_business['score_pair'].map(lambda x: x[1])


          plt.plot(np.arange(55, 100), np.arange(55, 100), color = 'red')
          plt.axis([55, 100, 55, 100])
          plt.scatter(x, y, facecolors = 'none', edgecolors = 'blue')

          plt.xlabel('First Score', fontsize = 12)
          plt.ylabel('Second Score', fontsize = 12)
          plt.title('First Inspection Score vs. Second Inspection Score', fontsize = 14)
```

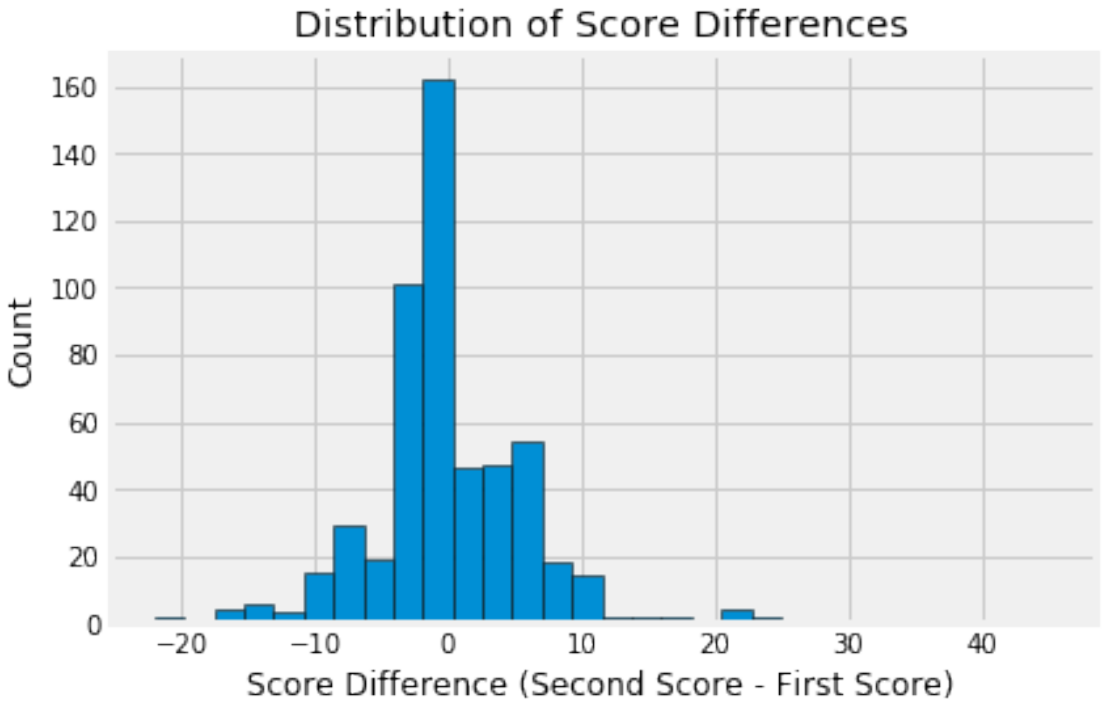
Out[289]: Text(0.5, 1.0, 'First Inspection Score vs. Second Inspection Score')



0.1.2 Question 7d

Another way to compare the scores from the two inspections is to examine the difference in scores. Subtract the first score from the second in `scores_pairs_by_business`. Make a histogram of these differences in the scores. We might expect these differences to be positive, indicating an improvement from the first to the second inspection.

The histogram should look like this:



Hint: Use `second_score` and `first_score` created in the scatter plot code above.

Hint: Convert the scores into numpy arrays to make them easier to deal with.

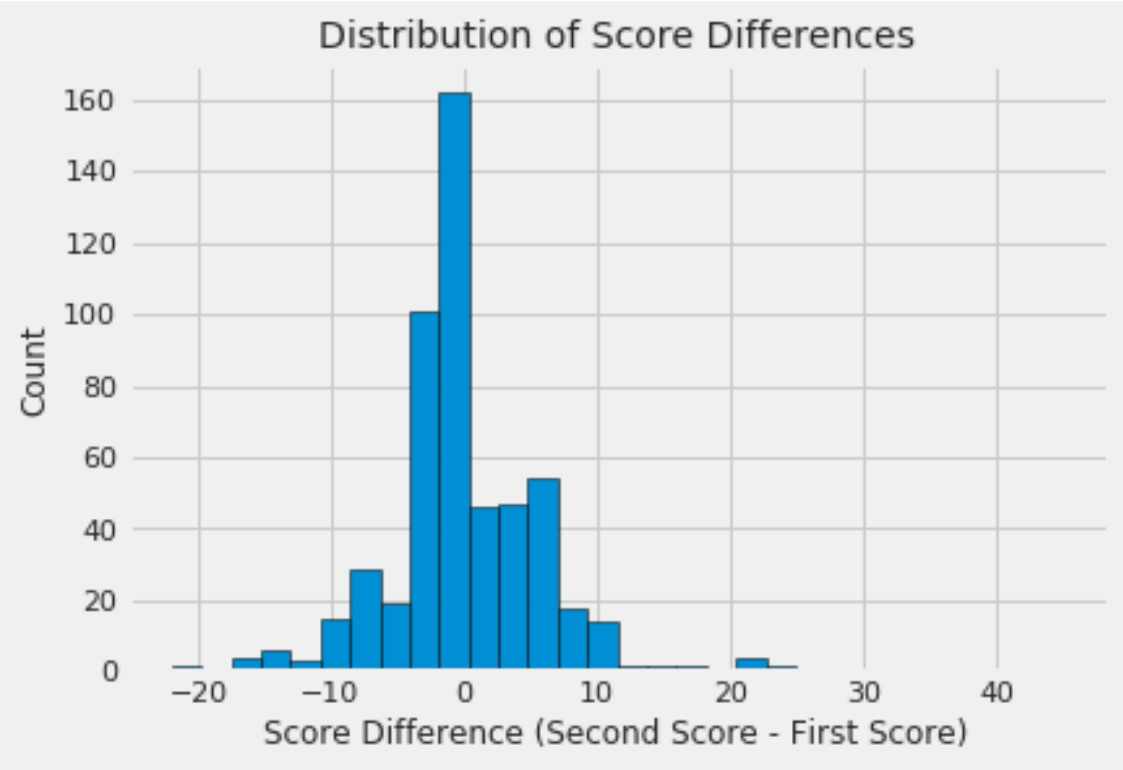
Hint: Use `plt.hist()` Try changing the number of bins when you call `plt.hist()`.

```
In [290]: diff_in_scores = y - x

          plt.hist(diff_in_scores.values, bins = 30, ec = 'black')
          plt.xlabel("Score Difference (Second Score - First Score)", fontsize = 12)
```

```
plt.ylabel("Count", fontsize = 12)
plt.title("Distribution of Score Differences", fontsize = 14)
```

```
Out[290]: Text(0.5, 1.0, 'Distribution of Score Differences')
```



0.1.3 Question 7e

If restaurants' scores tend to improve from the first to the second inspection, what do you expect to see in the scatter plot that you made in question 7c? What do you oberve from the plot? Are your observations consistent with your expectations?

Hint: What does the slope represent?

Since scores improve from first to second inspection, I would think the slope would be greater than 1 so that the points are mostly in the upper right instead of how it is right now (symmetrical around the line). This balanced distribution indicates that most businesses had the same score or around the same score in both inspections, or that the score increased/decreased the same amount. Essentially, it does not indicate that the restaurant's scores increased from the first to the second inspection.

0.1.4 Question 7f

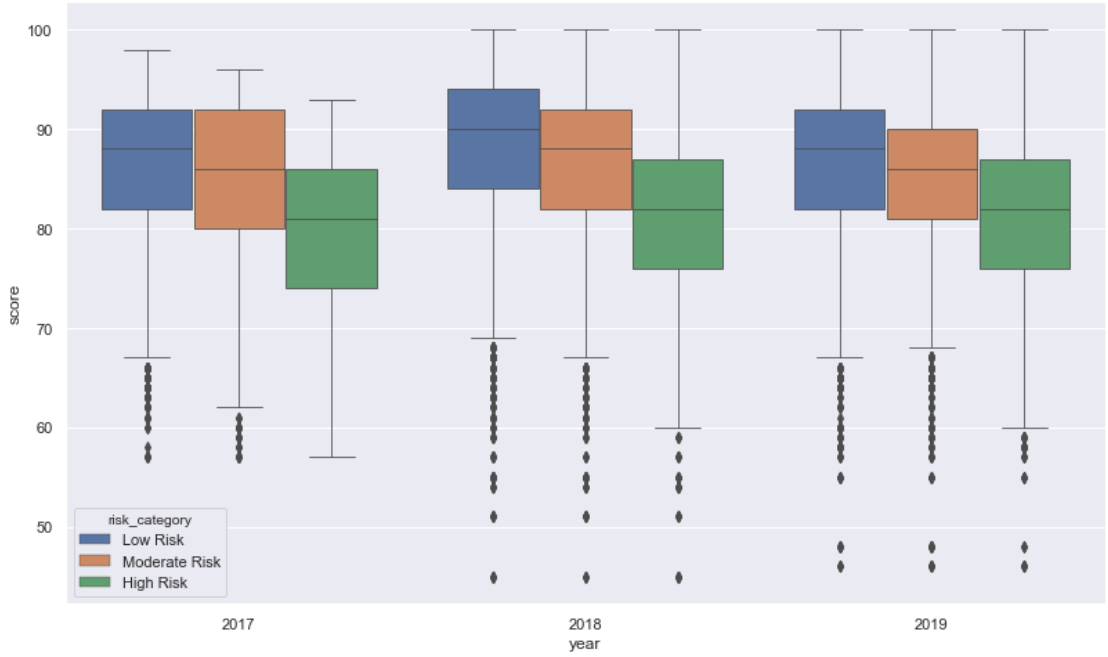
If a restaurant's score improves from the first to the second inspection, how would this be reflected in the histogram of the difference in the scores that you made in question 7d? What do you oberve from the plot? Are your observations consistent with your expectations? Explain your observations in the language of Statistics: for instance, the center, the spread, the deviation etc.

If a restaurant's score improves from the first to second inspection, it would be reflected in the histogram with the center being greater than 0 and the graph would be heavily skewed to the right to indicate the difference is positive. The spread of the graph would be relatively equal skewed to the right, indicating more restaurants have a positive diference. Instead, however, the histogram is centered around -2 and is skewed to the left rather than right. This indicates that, in fact, scores were worse from the first to the second inspection. This could be for various reasons, perhaps the inspectors are less lenient when issues aren't dealt with after the first inspection, although I personally expected that restaurants would attempt to address any issues from their first inspection in order to gain a better score and attract more business.

0.1.5 Question 7g

To wrap up our analysis of the restaurant ratings over time, one final metric we will be looking at is the distribution of restaurant scores over time. Create a side-by-side boxplot that shows the distribution of these scores for each different risk category from 2017 to 2019. Use a figure size of at least 12 by 8.

The boxplot should look similar to the sample below. Make sure the boxes are in the correct order!



Hint: Use `sns.boxplot()`. Try taking a look at the first several parameters. The documentation is linked here!

Hint: Use `plt.figure()` to adjust the figure size of your plot.

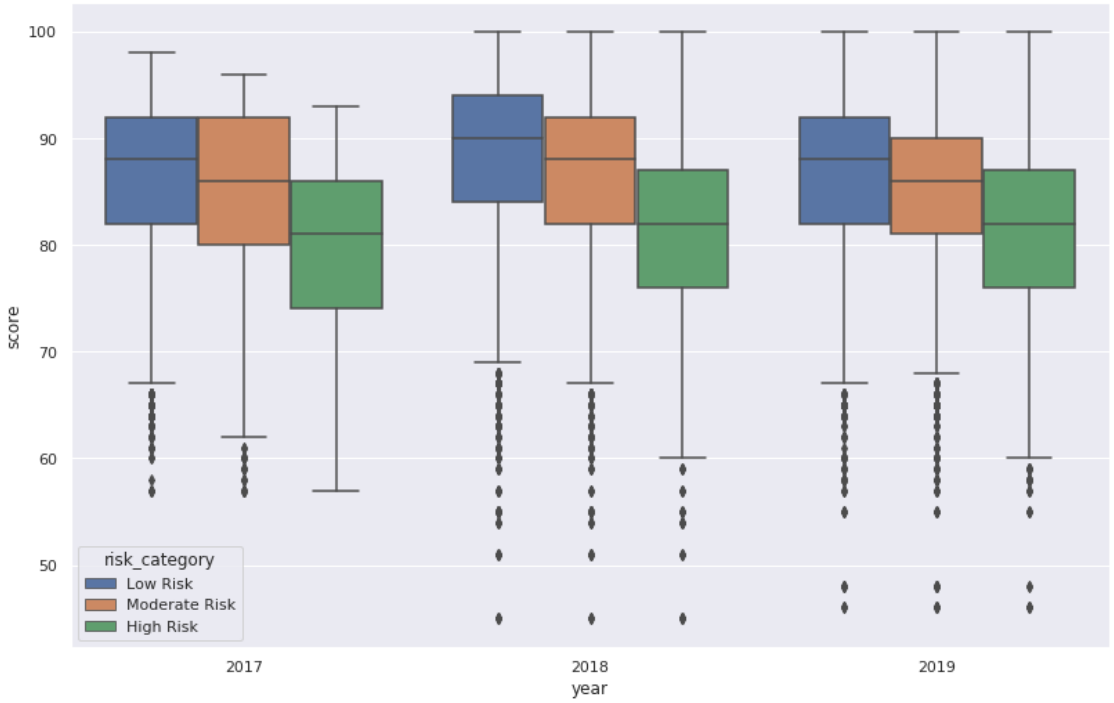
```
In [291]: # Do not modify this line
          sns.set()

          merge = ins2vio.merge(ins, how = 'outer', on = 'iid').merge(vio, how = 'outer', on = 'vid')
          merge['year'] = merge['timestamp'].dt.year

          positive_scores = merge[merge['score'] > 0]
          year_range = positive_scores[positive_scores['year']>2016]
```

```
convert = {'year': int}
year_range = year_range.astype(convert)

plt.figure(figsize = (12, 8))
plot = sns.boxplot(x = 'year', y = 'score', hue = 'risk_category', data = year_range, hue_ord
```



---

1 8: Open Ended Question

1.1 Question 8a

1.1.1 Compute Something Interesting

Play with the data and try to compute something interesting about the data. Please try to use at least one of groupby, pivot, or merge (or all of the above).

Please show your work in the cell below and describe in words what you found in the same cell. This question will be graded leniently but good solutions may be used to create future homework problems.

1.1.2 Grading

Since the question is more open ended, we will have a more relaxed rubric, classifying your answers into the following three categories:

- **Great** (4 points): Uses a combination of pandas operations (such as groupby, pivot, merge) to answer a relevant question about the data. The text description provides a reasonable interpretation of the result.
- **Passing** (1-3 points): Computation is flawed or very simple. The text description is incomplete but makes some sense.
- **Unsatisfactory** (0 points): No computation is performed, or a computation with completely wrong results.

Please have both your code and your explanation in the same one cell below. Any work in any other cell will not be graded.

```
In [292]: ins2vio['bid'] = ins2vio['iid'].str.split(pat='_', expand=True).astype(int)[0]
          ins2vio_dropped = ins2vio.drop("iid", axis=1)
          ins2vio_vio = ins2vio_dropped.merge(vio, how='outer', on='vid')

          ins2016 = ins[ins['year'] == 2016]
          ins2017 = ins[ins['year'] == 2017]
          ins2018 = ins[ins['year'] == 2018]
          ins2019 = ins[ins['year'] == 2019]
```

```
def helper(x):
    return list(x)


sort2016 = ins2016[ins2016['score'] > 0].sort_values(['bid', 'date'])
sort2016['num scores'] = sort2016.groupby('bid')['bid'].transform('count')
bid_score2016 = sort2016[sort2016['num scores'] == 2][['bid', 'score']]
score_pairs2016 = bid_score2016.groupby('bid').agg(helper).rename(columns = {'score': 'score_


sort2017 = ins2017[ins2017['score'] > 0].sort_values(['bid', 'date'])
sort2017['num scores'] = sort2017.groupby('bid')['bid'].transform('count')
bid_score2017 = sort2017[sort2017['num scores'] == 2][['bid', 'score']]
score_pairs2017 = bid_score2017.groupby('bid').agg(helper).rename(columns = {'score': 'score_


score_pairs2018 = scores_pairs_by_business


sort2019 = ins2019[ins2019['score'] > 0].sort_values(['bid', 'date'])
sort2019['num scores'] = sort2019.groupby('bid')['bid'].transform('count')
bid_score2019 = sort2019[sort2019['num scores'] == 2][['bid', 'score']]
score_pairs2019 = bid_score2019.groupby('bid').agg(helper).rename(columns = {'score': 'score_



dataframe_list = [score_pairs2016, score_pairs2017, score_pairs2018, score_pairs2019]


for dataframe in dataframe_list:
    x = dataframe['score_pair'].map(lambda x: x[0])
    y = dataframe['score_pair'].map(lambda x: x[1])

    data = y-x
    dataframe['Difference in Scores'] = data

reset = score_pairs2016.reset_index()

merged2016 = reset.merge(ins2vio_vio, how='left', on='bid')
merged2017 = score_pairs2017.merge(ins2vio_vio, how='left', on='bid')
merged2018 = score_pairs2018.merge(ins2vio_vio, how='left', on='bid')
merged2019 = score_pairs2019.merge(ins2vio_vio, how='left', on='bid')


negatives2016 = merged2016[merged2016['Difference in Scores'] < 0]
negatives2017 = merged2017[merged2017['Difference in Scores'] < 0]
negatives2018 = merged2018[merged2018['Difference in Scores'] < 0]
negatives2019 = merged2019[merged2019['Difference in Scores'] < 0]


negatives2016_relevant = negatives2016.sort_values(['Difference in Scores', 'risk_category'])
negatives2017_relevant = negatives2017.sort_values(['Difference in Scores', 'risk_category'])
negatives2018_relevant = negatives2018.sort_values(['Difference in Scores', 'risk_category'])
negatives2019_relevant = negatives2019.sort_values(['Difference in Scores', 'risk_category'])
```

```
risks2016 = negatives2016_relevant.groupby("risk_category").agg("count")
risks2017 = negatives2017_relevant.groupby("risk_category").agg("count")
risks2018 = negatives2018_relevant.groupby("risk_category").agg("count")
risks2019 = negatives2019_relevant.groupby("risk_category").agg("count")
```

```
risks_list = [risks2016, risks2017, risks2018, risks2019]

for risks in risks_list:
    print(risks)
```

```
#After seeing that there was a decent amount of businesses in 2018 whose score from a second
#lower than the first inspection, I wanted to see what the majority of violations looked like
#Before doing any computations, I assumed that, for all four years, there would be a large am
#violations and a small amount of moderate and high risk violations, simply because smaller v
#sometimes be overlooked by restaurants during their everyday work. For example, businesses a
#to check for pests and ensure food isn't contaminated than following safety guidelines to a
#the calculations, I saw that low risks were in the majority for all four years. However, wha
#that there was a substantial amount of moderate risks as well. In 2016, 2018, and 2019, mode
#not that far off from the amount of low risk violations, with moderate risk violations being
#85.3% of low risk violations in 2016, 2018, and 2019 respectfully. I did not expect this bec
#type of violations classified as moderate, I am surprised that these kinds of violations occ
```

```
               Difference in Scores
risk_category
High Risk                         5
Low Risk                         14
Moderate Risk                    10
               Difference in Scores
risk_category
High Risk                       239
Low Risk                       1043
Moderate Risk                   672
               Difference in Scores
risk_category
High Risk                       290
Low Risk                        831
Moderate Risk                   725
               Difference in Scores
risk_category
High Risk                       348
Low Risk                        945
Moderate Risk                   806
```

1.1.3 Grading

Since the question is more open ended, we will have a more relaxed rubric, classifying your answers into the following three categories:

- **Great** (4 points): The chart is well designed, and the data computation is correct. The text written articulates a reasonable metric and correctly describes the relevant insight and answer to the question you are interested in.
- **Passing** (1-3 points): A chart is produced but with some flaws such as bad encoding. The text written is incomplete but makes some sense.
- **Unsatisfactory** (0 points): No chart is created, or a chart with completely wrong results.

We will lean towards being generous with the grading. We might also either discuss in discussion or post on Piazza some examplar analysis you have done (with your permission)!

You should have the following in your answers: * a few visualizations; Please limit your visualizations to 5 plots. * a few sentences (not too long please!)

Please note that you will only receive support in OH and Piazza for Matplotlib and seaborn questions. However, you may use some other Python libraries to help you create you visualizations. If you do so, make sure it is compatible with the PDF export (e.g., Plotly does not create PDFs properly, which we need for Gradescope).

In [293]:
```
negatives2019 = merged2019[merged2019['Difference in Scores'] < 0]

merged2019.sort_values(['Difference in Scores', 'risk_category']) #to find out the range of D

fourth = 18*.25
half = 18*.5
three_quarters = 18*.75

percent25 = negatives2019[negatives2019['Difference in Scores'] >= -(fourth)]

mid1 = negatives2019[negatives2019['Difference in Scores'] < -(fourth)]
percent50 = mid1[mid1['Difference in Scores'] >= -(half)]

mid2 = negatives2019[negatives2019['Difference in Scores'] < -(half)]
percent75 = mid2[mid2['Difference in Scores'] >= -(three_quarters)]

mid3 = negatives2019[negatives2019['Difference in Scores'] < -(three_quarters)]
percent100 = mid3[mid3['Difference in Scores'] >= -18]

percent25_relevant = percent25.sort_values(['Difference in Scores', 'risk_category']).drop("b
percent50_relevant = percent50.sort_values(['Difference in Scores', 'risk_category']).drop("b
percent75_relevant = percent75.sort_values(['Difference in Scores', 'risk_category']).drop("b
percent100_relevant = percent100.sort_values(['Difference in Scores', 'risk_category']).drop(
```

```
risks25 = percent25_relevant.groupby("risk_category").agg("count")
risks50 = percent50_relevant.groupby("risk_category").agg("count")
risks75 = percent75_relevant.groupby("risk_category").agg("count")
risks100 = percent100_relevant.groupby("risk_category").agg("count")


percent_risks_list = [risks25, risks50, risks75, risks100]

for risks in percent_risks_list:
    print(risks)

reset_25 = risks25.reset_index()
reset_50 = risks50.reset_index()
reset_75 = risks75.reset_index()
reset_100 = risks100.reset_index()


fig = plt.figure(figsize = (15, 15))
fig.suptitle("Analysis of Risk Categories v. Difference in Scores in 2019")

bin_range = np.arange(40, 100)

fig.subplots_adjust(left = 0.125,
                    right = 0.9,
                    bottom = 0.1,
                    top = 0.9,
                    wspace = 0.5,
                    hspace = 0.5)



first_subgroup = fig.add_subplot(321)
plt.bar(reset_25['risk_category'], reset_25['Difference in Scores'], color = "pink")
first_subgroup.set_title("Number of Violations for Each Risk Category: up to 25% Points Lost"
first_subgroup.set_xlabel("Risk Categories")
first_subgroup.set_ylabel("Number of Violations")

second_subgroup = fig.add_subplot(322)
plt.bar(reset_50['risk_category'], reset_50['Difference in Scores'], color = "purple")
second_subgroup.set_title("Number of Violations for Each Risk Category: 26% to 50% Points Lost
second_subgroup.set_xlabel("Risk Categories")
second_subgroup.set_ylabel("Number of Violations")

third_subgroup = fig.add_subplot(323)
plt.bar(reset_75['risk_category'], reset_75['Difference in Scores'], color = "cyan")
third_subgroup.set_title("Number of Violations for Each Risk Category: 50% to 75% Points Lost
third_subgroup.set_xlabel("Risk Categories")
third_subgroup.set_ylabel("Number of Violations")

fourth_subgroup = fig.add_subplot(324)
plt.bar(reset_100['risk_category'], reset_100['Difference in Scores'], color = "orange")
fourth_subgroup.set_title("Number of Violations for Each Risk Category: 75% to 100% Points Los
fourth_subgroup.set_xlabel("Risk Categories")
```

```
fourth_subgroup.set_ylabel("Number of Violations")
```

```
#Continuing off of the calculations in 8a, I wanted to further break it down, choosing 2019 t
#specifically how the risk categories are related to the difference in scores. I hypothesized
#the difference, the lower the risk of violations. Thus, I expected there to be far more low
#the first quarter and a far more high risks in the last quarter. This was sort of true, gives
#violations were the highest in the first and second quarter. However, moderate risks really
#they were not far off in those first two quarters and then took over the most common type of
#and fourth quarter. I had expected high risks to be more frequent for the larger differences
#quite surprising.
```

```
               Difference in Scores
risk_category
High Risk                       170
Low Risk                        494
Moderate Risk                   433
               Difference in Scores
risk_category
High Risk                       130
Low Risk                        375
Moderate Risk                   280
               Difference in Scores
risk_category
High Risk                        27
Low Risk                         45
Moderate Risk                    57
               Difference in Scores
risk_category
High Risk                        21
Low Risk                         31
Moderate Risk                    36
```

```
Out[293]: Text(0, 0.5, 'Number of Violations')
```

Analysis of Risk Categories v. Difference in Scores in 2019