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.

```
In [107]: vio.head(20)
```

```
Out[107]:                                        description  risk_category     vid
          0   Consumer advisory not provided for raw or unde…  Moderate Risk  103128
          1                    Contaminated or adulterated food      High Risk  103108
          2            Discharge from employee nose mouth or eye  Moderate Risk  103117
          3                         Employee eating or smoking  Moderate Risk  103118
          4                             Food in poor condition  Moderate Risk  103123
          5   Food safety certificate or food handler card n…       Low Risk  103157
          6               Foods not protected from contamination  Moderate Risk  103133
          7                   High risk food holding temperature      High Risk  103103
          8                         High risk vermin infestation      High Risk  103114
          9                  Improper cooking time or temperatures      High Risk  103106
          10                           Improper cooling methods      High Risk  103105
          11     Improper food labeling or menu misrepresentation       Low Risk  103141
          12                              Improper food storage       Low Risk  103139
          13                       Improper or defective plumbing       Low Risk  103150
          14                          Improper reheating of food      High Risk  103107
          15     Improper storage of equipment utensils or linens       Low Risk  103145
          16  Improper storage use or identification of toxi…       Low Risk  103138
          17                           Improper thawing methods  Moderate Risk  103132
          18  Improperly displayed mobile food permit or sig…  Moderate Risk  103175
          19            Improperly washed fruits and vegetables       Low Risk  103137
```

bus.csv - the name column is not uniform and latitude and longitude have a lot of missing values which have been depicted by -9999. The phone number column also has missing values depicted by the same number

ins.csv - The score column (which seems to be an important column) has a lot of missing values depicted by -1. The type column, which should ideally have categorical data has a lot of different categories which might make it difficult to group

vio.csv - The description column is not is all lower cases.

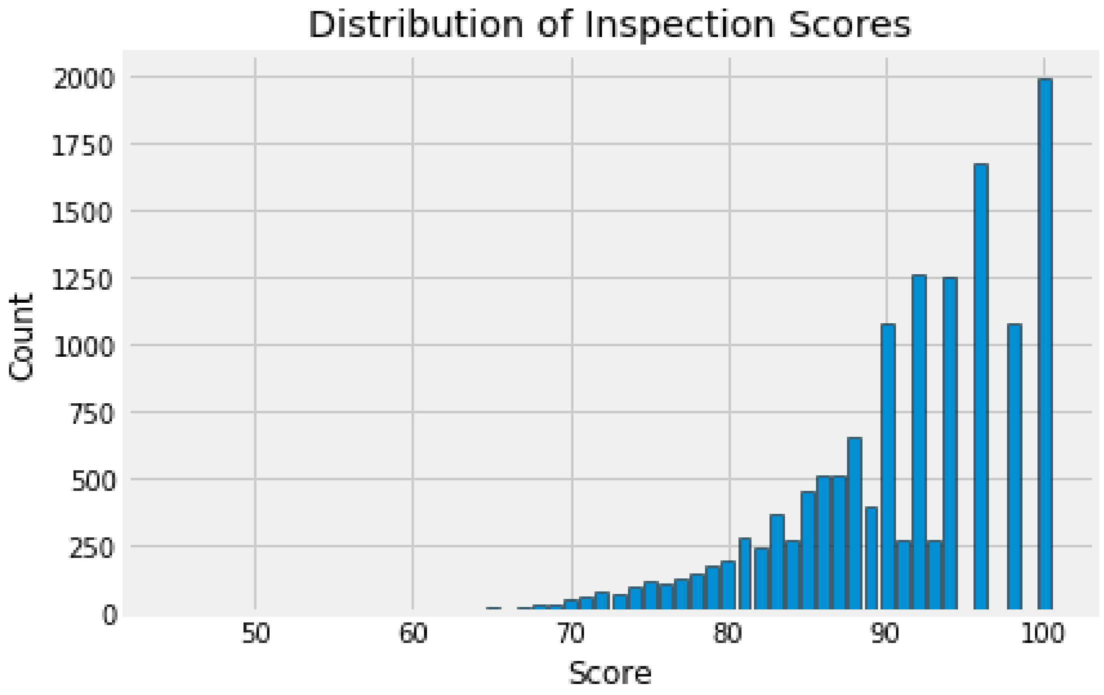
In the cell below, write the name of the restaurant with the lowest inspection scores ever. You can also head to yelp.com and look up the reviews page for this restaurant. Feel free to add anything interesting you want to share.

It's Lollipot with a score of 45. Yelp.com has a surprisingly decent number of good reviews with a few bad reviews here and there.

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.

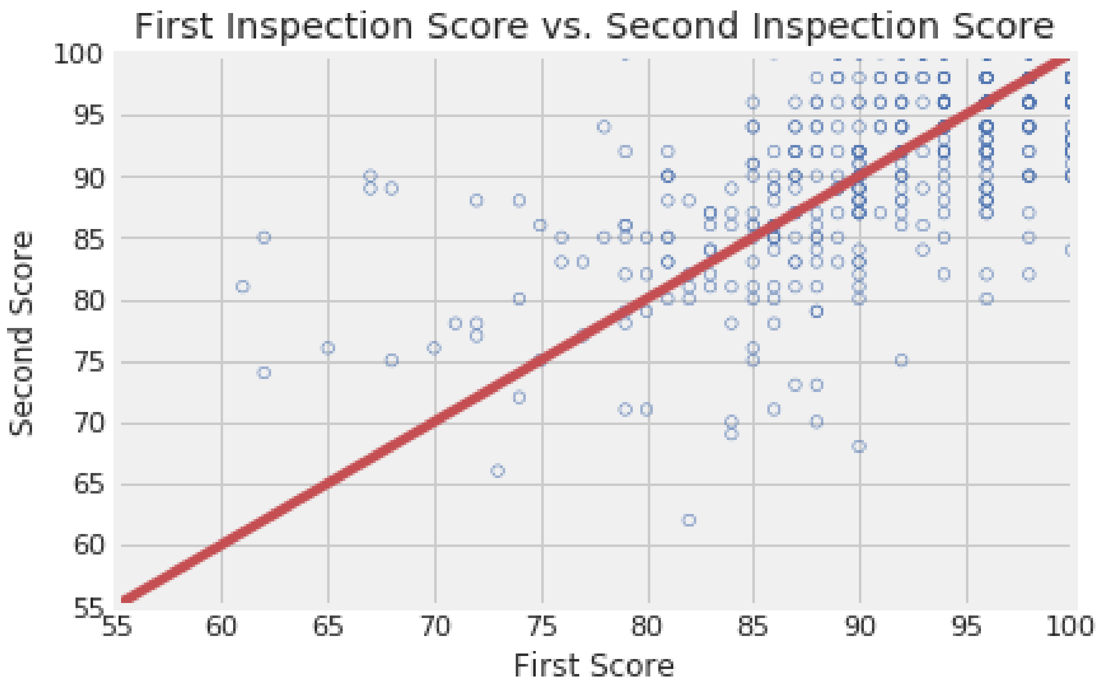
```
In [166]: score_counts = ins_named["score"].value_counts()
```

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?

It's a vertical bar chart. It is skewed to the right with a tail extending on the left. While there are fewer gaps for the lower values, there are significant gaps as the scores increase. This means that those scoring a higher value are few.

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.

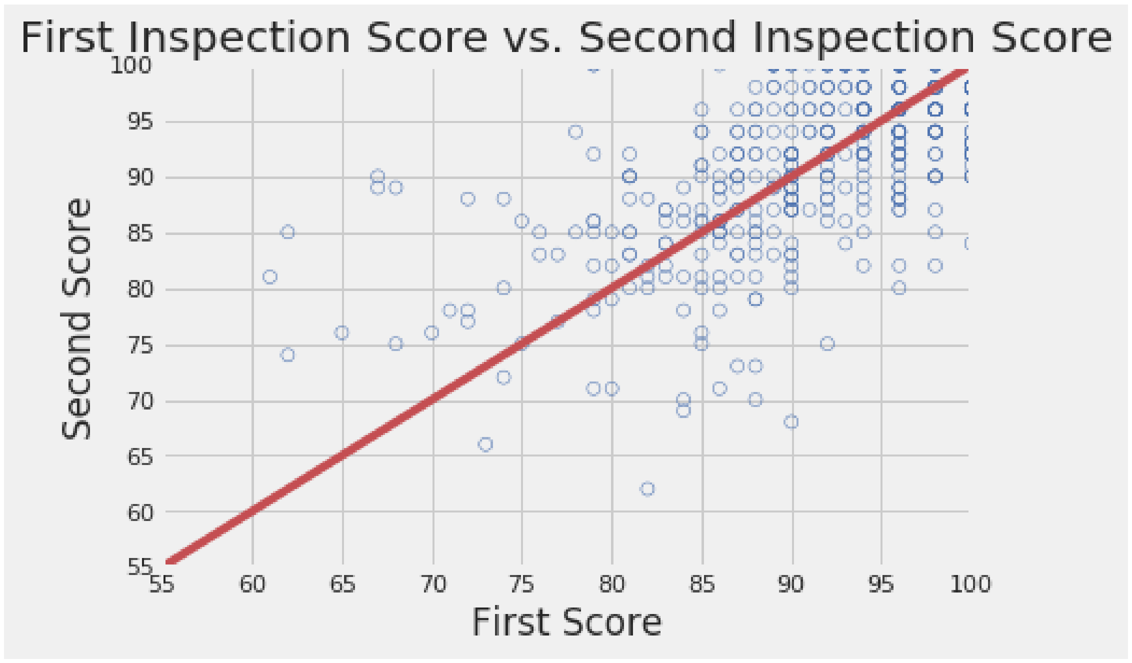
`plt.xlabel`, `plt.ylabel`, `plt.axis`, and `plt.title`.

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

```
In [175]: zipped = scores_pairs_by_business["score_pair"].to_list()
          unzipped_values = zip(*zipped)
          unzipped_list = list(unzipped_values)
          first_val = unzipped_list[0]
          second_val = unzipped_list[1]
          plt.scatter(x=first_val, y = second_val, facecolors="none", edgecolors="b")
          plt.ylim(55,100)
          plt.xlim(55,100)
          plt.xlabel("First Score")
          plt.ylabel("Second Score")
          plt.title("First Inspection Score vs. Second Inspection Score")
```

```
x_line = np.linspace(55,100, 1000)
plt.plot(x_line, x_line, color="r")
```

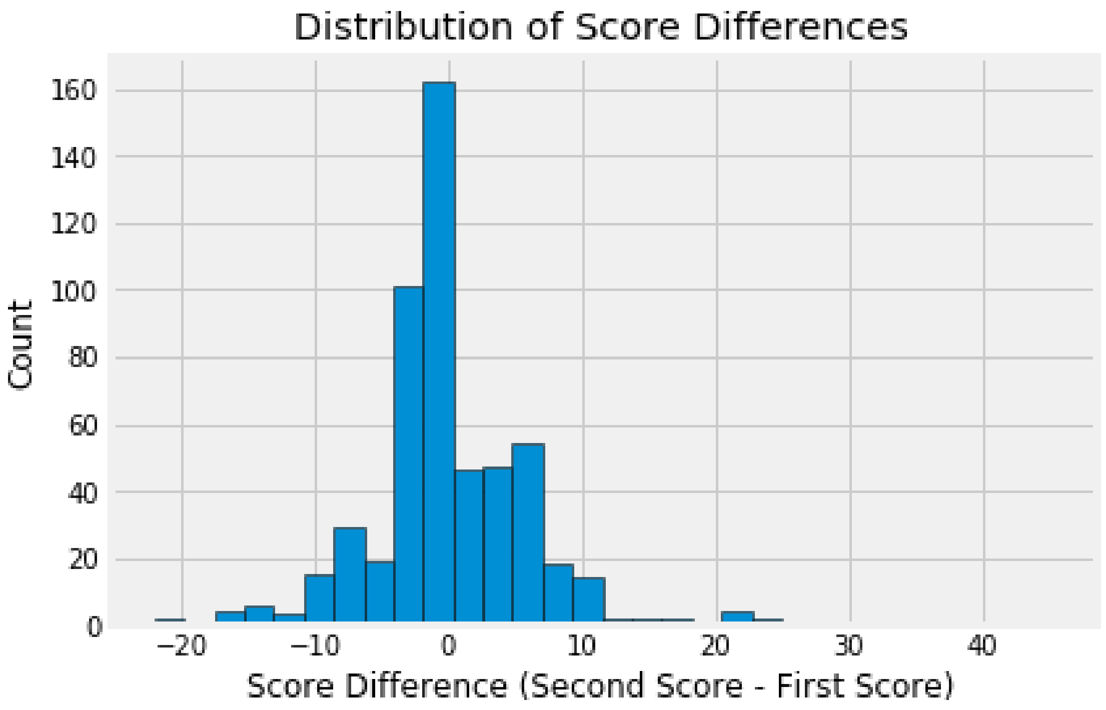
```
Out[175]: [<matplotlib.lines.Line2D at 0x7f3e6e965e20>]
```



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.

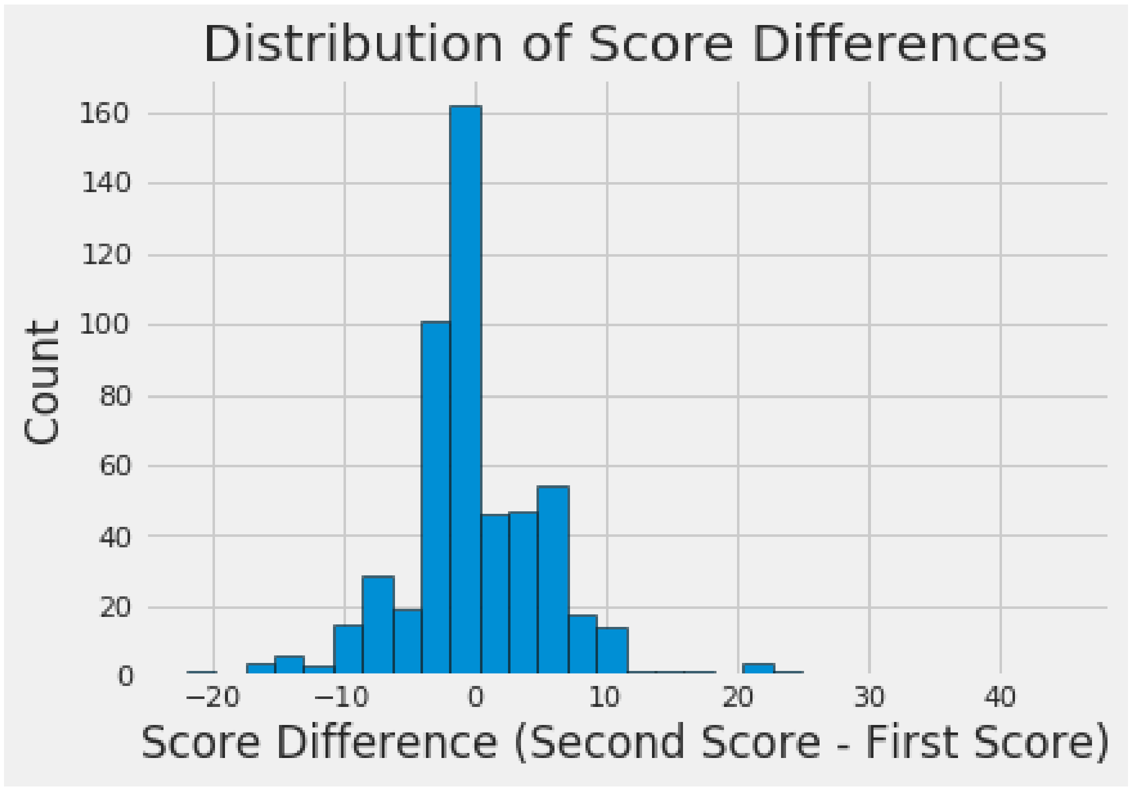
Hint: Convert the scores into numpy arrays to make them easier to deal with.

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

```
In [176]: first_arr = np.asarray(first_val)
          second_arr = np.asarray(second_val)
          diff_score = second_arr-first_arr
          plt.hist(diff_score, bins=30, edgecolor="black")
          plt.xlabel("Score Difference (Second Score - First Score)")
```

```
        plt.ylabel("Count")
        plt.title("Distribution of Score Differences")
```

```
Out[176]: 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?

If the restaurants had a better score the second time, the values of first score would be above the plotted red line. Instead, we observe that the values are equally distributed on both sides of the slope. So no, it is not consistent. The number of restaurants improving is equal to those doing badly.

0.1.4 Question 7f

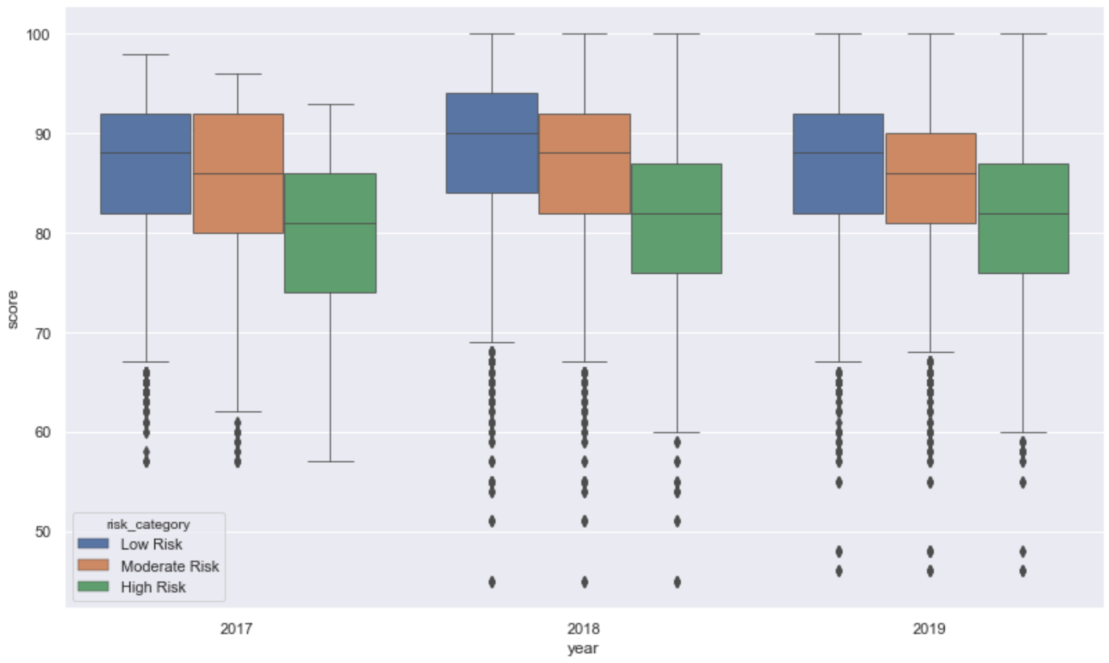
If a restaurant's score improves from the first to the second inspection, how would this be reflected in the histogram of the difference in the scores that you made in question 7d? What do you oberve from the plot? Are your observations consistent with your expectations? Explain your observations in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

If there was a increase in the score after the second inspection, one would see the point above the y=x line. As for the histogram, it would be skewed to the right beyond 0. But that is not the case for the histogram. The scatterplot shows an equal distribution of positive and negative differences. The histogram is centered at 0, with an unimodal distribution. This could indicate that on average difference between the first and second score is 0. The distribution also has a long tail on both sides with a range of -20 to 25.

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!



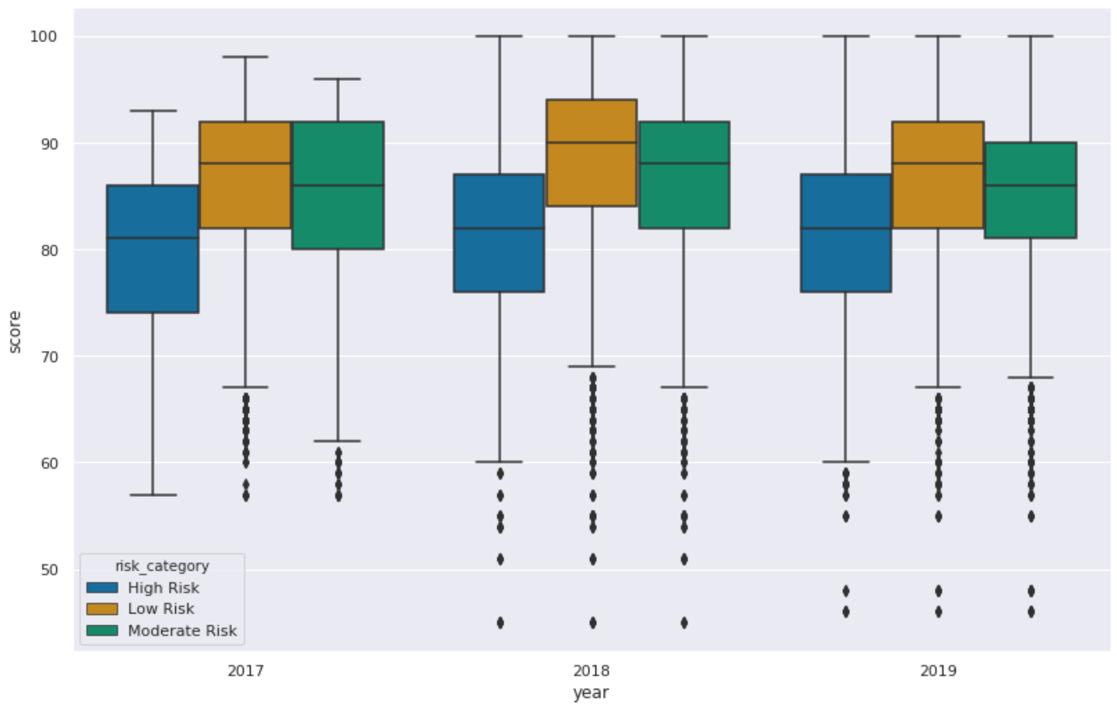
Hint: Use `sns.boxplot()`. Try taking a look at the first several parameters. The documentation is linked here!

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

```
In [201]: # Do not modify this line
          sns.set()

          three_years = ins_named[ins_named["year"].isin([2017,2018,2019])]
          ins_named2 = pd.merge(ins, bus, on="bid",how="left" )
          m1 = pd.merge(three_years, ins2vio, on="iid", how="left")
          m2 = pd.merge(m1, vio, on="vid", how="left")
          plt.figure(figsize=(12,8))
          sns.boxplot(y='score', x='year',
                           data=m2,
                           palette="colorblind",
                           hue='risk_category')
```

```
Out[201]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4e55d56b20>
```



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 [203]: #YOUR CODE HERE
          ins["year"].value_counts()
          #let us take a look at the months of the inspections.
          #also the types of routines are uncheduled so we want to see which month was most frequent in
          ins["month"] = ins['timestamp'].dt.month
          ins["type"].value_counts()
          group_by_month = ins.groupby(by="month", as_index=False).agg("size").sort_values("size", ascen
          #we can see that April is the month with the maximum number of scheduled inspections. Digging
          only_april = ins[ins['month']==4]
          only_april
```

```
#now checking number of violations for each violation categories
for_vio = pd.merge(only_april, ins2vio, on="iid", how="left")
for_vio2 = pd.merge(for_vio,vio, on="vid", how="left")
#now group by descriptions
group_by_desc = for_vio2.groupby(by="description", as_index=False).agg("size").sort_values("s
group_by_desc
#the description for highest count
highest_desc = group_by_desc.iloc[0].description
highest_desc

#All the inspections were unscheduled, so I wanted to see what month had maximum number of vi
#It was in April and in this month, max violations were 'Inadequately cleaned or sanitized fo
```

```
<ipython-input-203-b8b7e1360b51>:5: 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.
  ins["month"] = ins['timestamp'].dt.month
```

```
Out[203]: 'Inadequately cleaned or sanitized food contact surfaces'
```

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!)

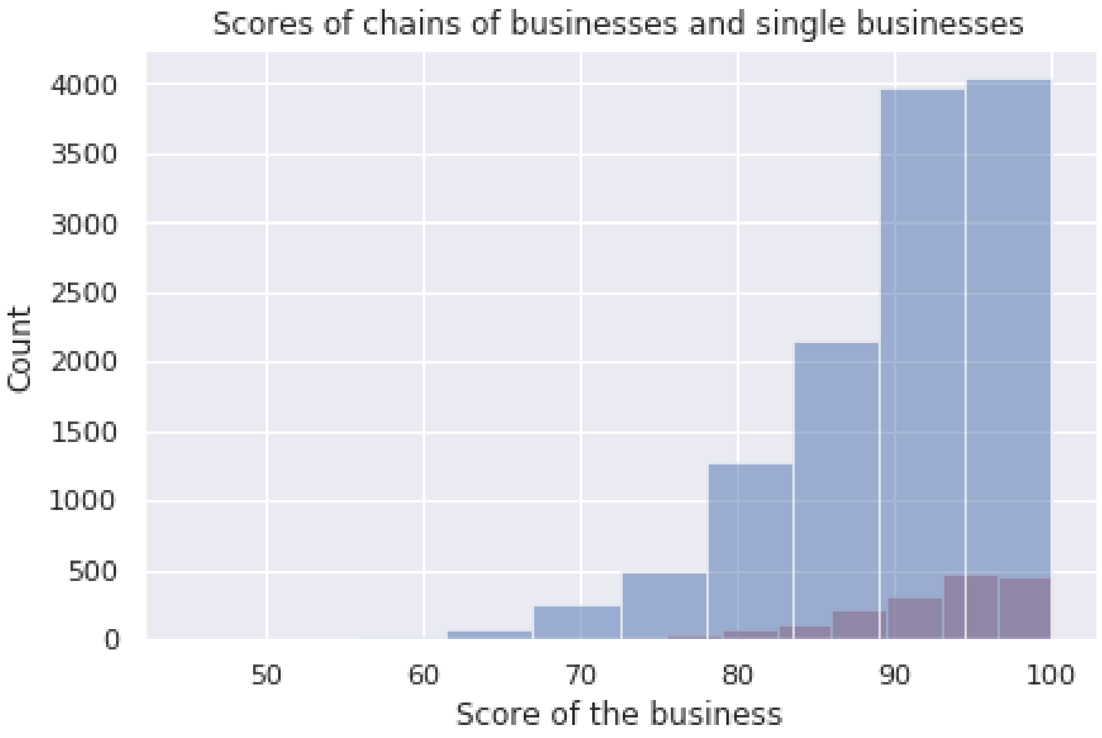
Please note that you will only receive support in OH and Piazza for Matplotlib and seaborn questions. However, you may use some other Python libraries to help you create you visualizations. If you do so, make sure it is compatible with the PDF export (e.g., Plotly does not create PDFs properly, which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

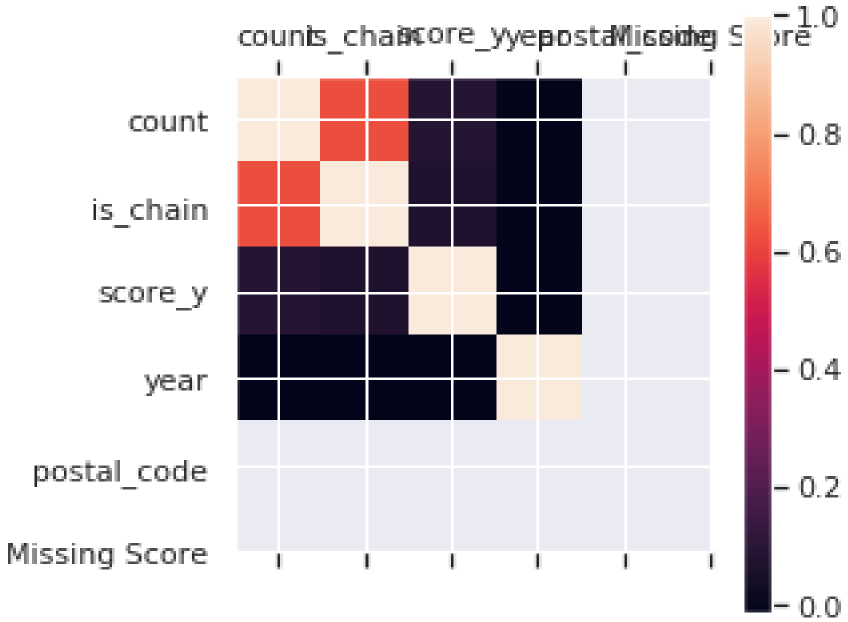
```
In [204]: # YOUR DATA PROCESSING AND PLOTTING HERE
          # want to check what is the difference in the scores of restaurant-chains and individual rest
          list_of_bus = bus["name"].value_counts().to_frame().reset_index().rename(columns={"index":"na
          #merge bus and ins
          group_by_name = ins_named.groupby(as_index=False, by="name")["score"].agg("max")
          #merge with list_of_bus to add number of businesses to the group_by_name dataframe
          group_by_name_count = pd.merge(group_by_name, list_of_bus, on="name", how="left").sort_values
          group_by_name_count["is_chain"] = [False if x==1 else True for x in group_by_name_count["coun
          #merging with original ins_named
          fin_merge = pd.merge(group_by_name_count, ins_named, on="name", how="right")
          display(fin_merge["is_chain"].value_counts())
          yeschain = fin_merge[fin_merge["is_chain"]==True]["score_y"].to_list()
          nochain = fin_merge[fin_merge["is_chain"]==False]["score_y"].to_list()
          fig, ax = plt.subplots()
          plt.hist(yeschain, color="r", alpha = 0.5, )
          plt.hist(nochain, color="b", alpha=0.5)
          plt.xlabel("Score of the business")
          plt.ylabel("Count")
          plt.title("Scores of chains of businesses and single businesses")


          #correlation matrix to understand the relation between the different
          #vairables
          df = fin_merge[["count", "is_chain","score_y","year","postal_code","Missing Score"]]
```

```
plt.matshow(df.corr())
#plt.figure(figsize=[10,12])
plt.xticks(range(len(df.columns)), df.columns)
plt.yticks(range(len(df.columns)), df.columns)
plt.colorbar()
plt.show()
#The histogram depicts the scores distribution for the chain businesses and non-chain busines
#difference in the heights of the 2 histograms is due to the number of entries (shown in the
#that there is not much difference in the distributions so the chain businesses are not doing
#non-chain counterparts. Postal codes and missing scores in the second chart have the grey ar
```

```
False    12292
True      1739
Name: is_chain, dtype: int64
```




count
is_chain
score_y
year
postal_code
Missing Score
1.0
0.8
0.6
0.4
0.2
0.0