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 [711]: ins.head()
```

Out[711]:

	iid	date	score	type
0	100010_20190329	03/29/2019 12:00:00 AM	-1	New Construction
1	100010_20190403	04/03/2019 12:00:00 AM	100	Routine - Unscheduled
2	100017_20190417	04/17/2019 12:00:00 AM	-1	New Ownership
3	100017_20190816	08/16/2019 12:00:00 AM	91	Routine - Unscheduled
4	100017_20190826	08/26/2019 12:00:00 AM	-1	Reinspection/Followup

```
In [712]: vio.head(20)
```

Out[712]:

	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

In the bus dataframe, I immediately noticed that a restaurant is missing a phone number and thus the number is displayed as -9999; if ignored, this could jeopardize our analysis as it essentially indicates missing data. For the ins dataframe, many scores are also missing for the ”score” column, and denoted as -1. As of now, I don’t see any issues with the data itself.

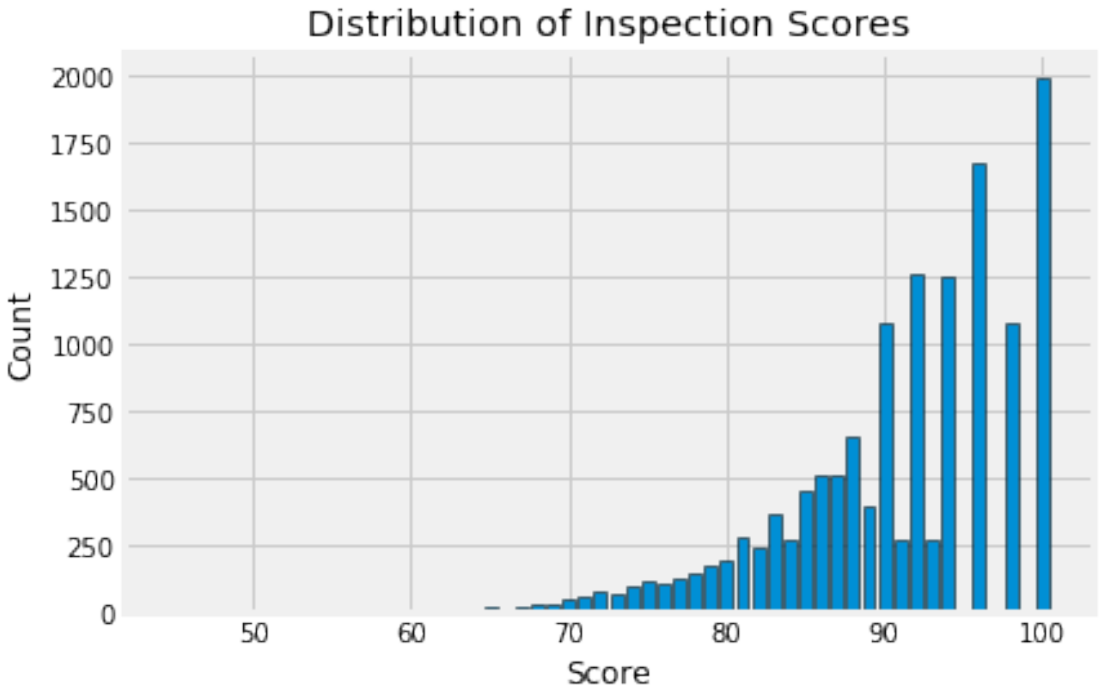
In the cell below, write the name of the restaurant with the lowest inspection scores ever. You can also head to yelp.com and look up the reviews page for this restaurant. Feel free to add anything interesting you want to share.

Lollipot has the lowest score rating.

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.

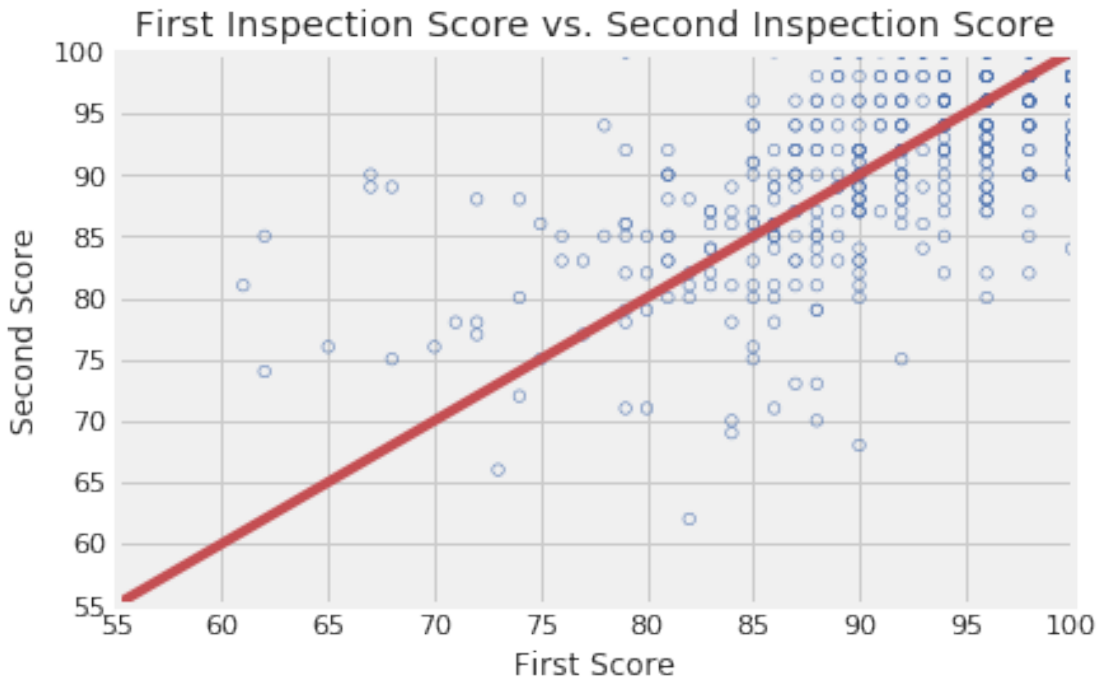
```
In [786]: data = ins.groupby("score").count()["Missing Score"]
```

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 output bar graph is skewed to the left with a long tail in the negative direction of the x-axis. There seem to be a few gaps in between some bins, especially between the 90-100 bins, which suggests that there are missing values in the inspection scores. What is also interesting is that most scores are above 65, which may be due to multiple reasons: maybe the inspectors are scoring too highly, or most restaurants registered in San Francisco are decent with their health inspection qualities, or most restaurants who have very low inspection scores were shut down, such that they are not included in the dataset anymore.

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.

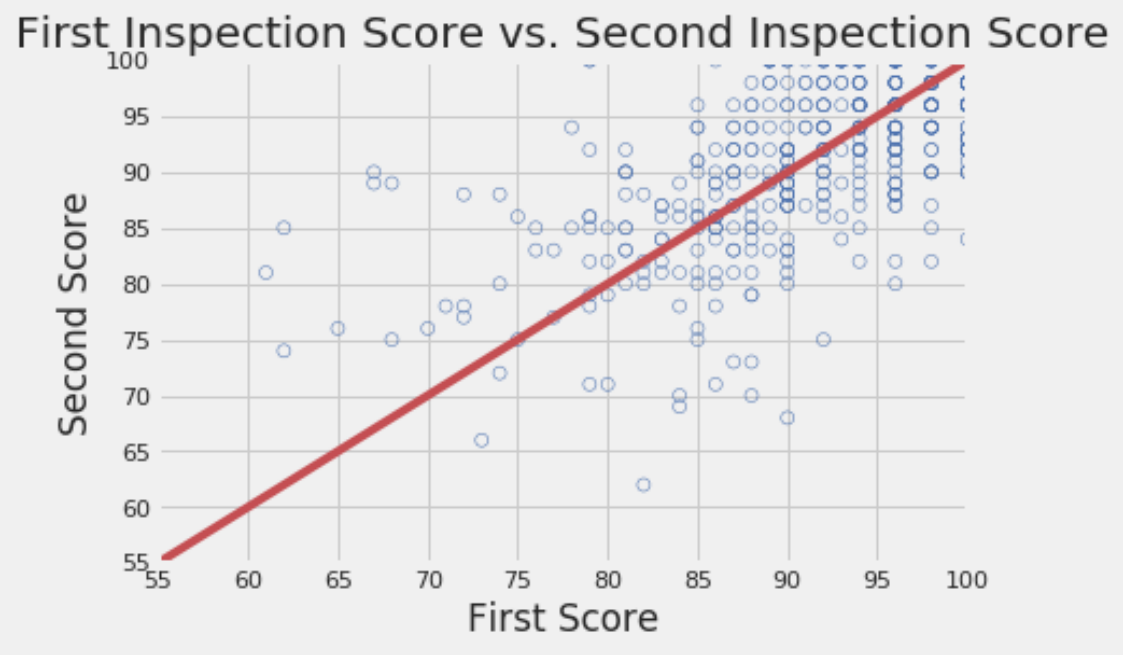
`plt.xlabel`, `plt.ylabel`, `plt.axis`, and `plt.title`.

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

```
In [800]: items = zip(*scores_pairs_by_business["score_pair"])
          items = list(items)
          first = items[0]
          second = items[1]

          plt.scatter(x=first, y=second, facecolors='none', edgecolors='b')
          plt.xlabel("First Score")
          plt.ylabel("Second Score")
          plt.title("First Inspection Score vs. Second Inspection Score")
          plt.axis([55, 100, 55, 100])
          x = np.linspace(0, 100)
```

```
plt.plot(x, x, 'r')
```

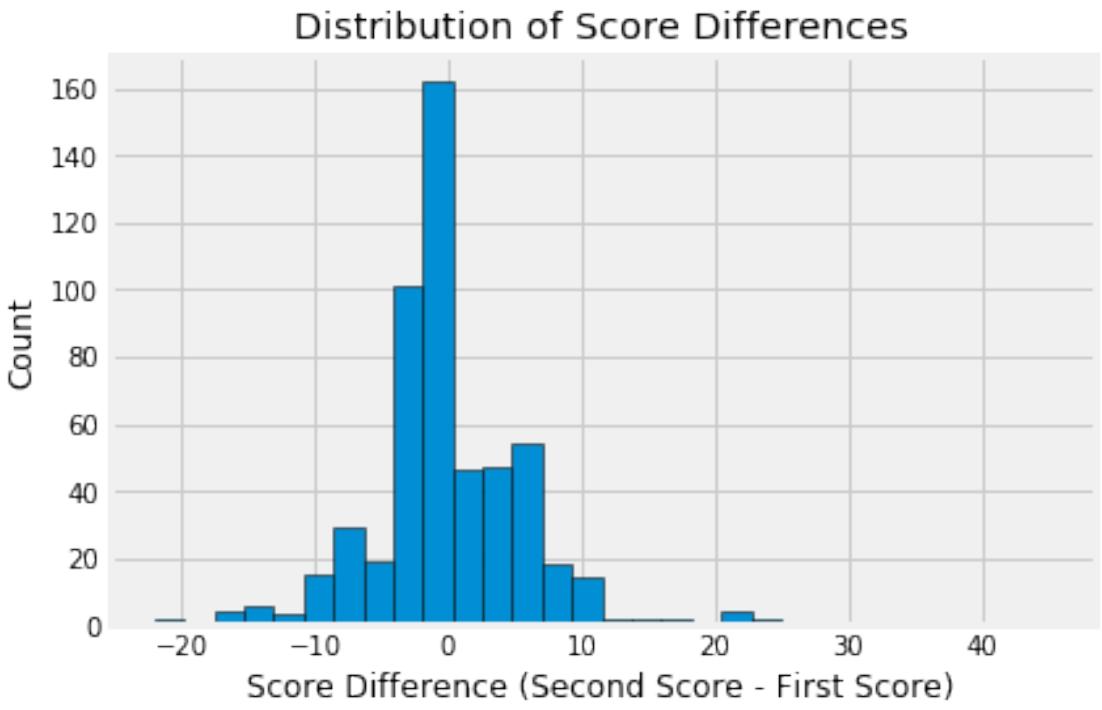
```
Out[800]: [<matplotlib.lines.Line2D at 0x7f2dc6d2ec70>]
```



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:



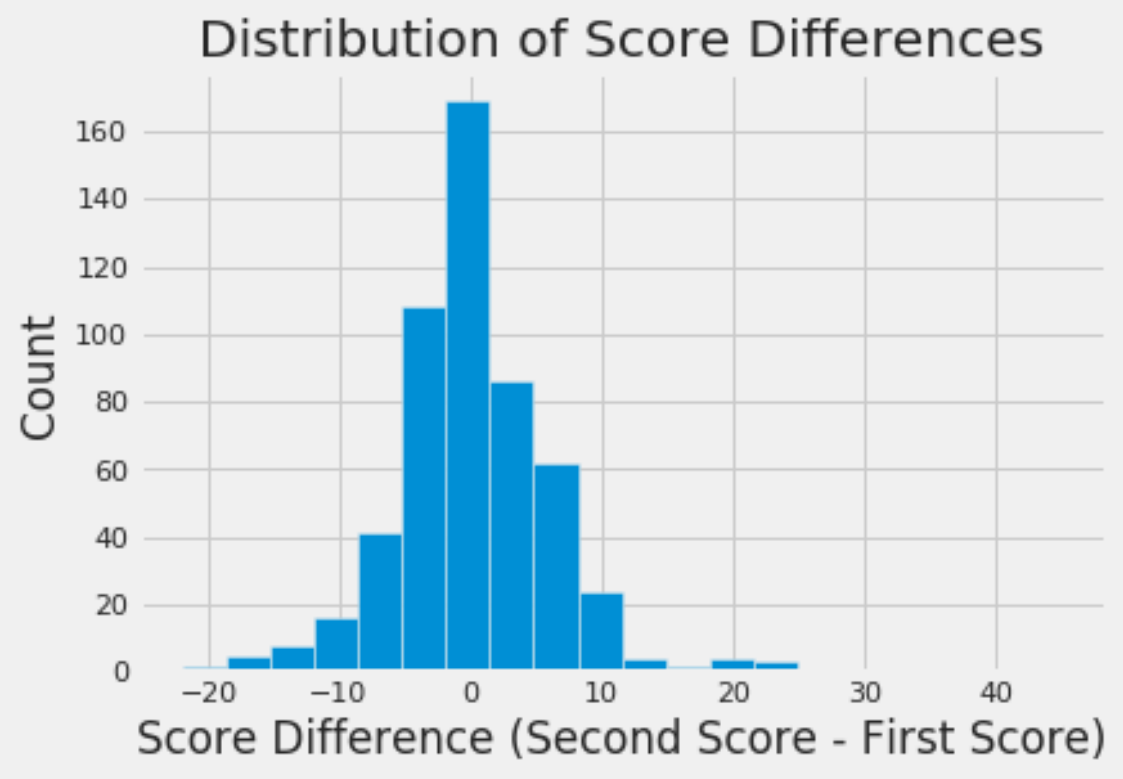
Hint: Use `second_score` and `first_score` created in the scatter plot code above.

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 [801]: s = np.subtract(second, first)
          plt.hist(s, bins=20)
          plt.xlabel("Score Difference (Second Score - First Score)")
          plt.ylabel("Count")
          plt.title("Distribution of Score Differences")
```

Out[801]: 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 restaurants' scores tend to improve from the first to the second inspection, the slope of the best fit line should be higher, such that the ratio of rise over run is higher. On the other hand, I observe that the slope of the best fit line is in fact, quite linear such that it would be similar to f(x) = x. This indicatest that the scores do not tend to improve noticeably between the first and second inspections. Thus, my observations are not consistent with my expectations.

0.1.4 Question 7f

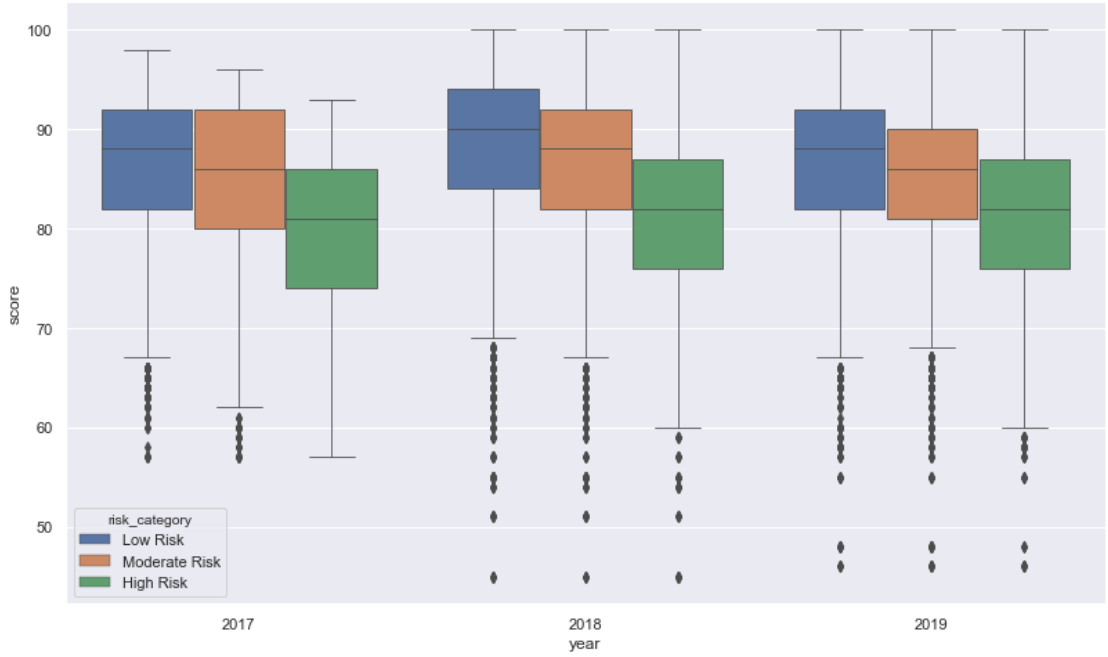
If a restaurant's score improves from the first to the second inspection, how would this be reflected in the histogram of the difference in the scores that you made in question 7d? What do you oberve from the plot? Are your observations consistent with your expectations? Explain your observations in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

If the scores were to be improved, the overall distribution should be centered around a higher mean value, i.e. the distribution itself should shift to the right as a whole such that its centre was a more postive value, since the x-axis indicates the score difference. I don't necessarily expect the spread or standard deviation to change noticeably. However, I am observing that the output histogram is a bell-curve centered around a mean of 0, indicating that the average score change is 0 such that the scores do not necessarily show a tend of improving between the two inspections. This is not consistent with my expectations.

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.

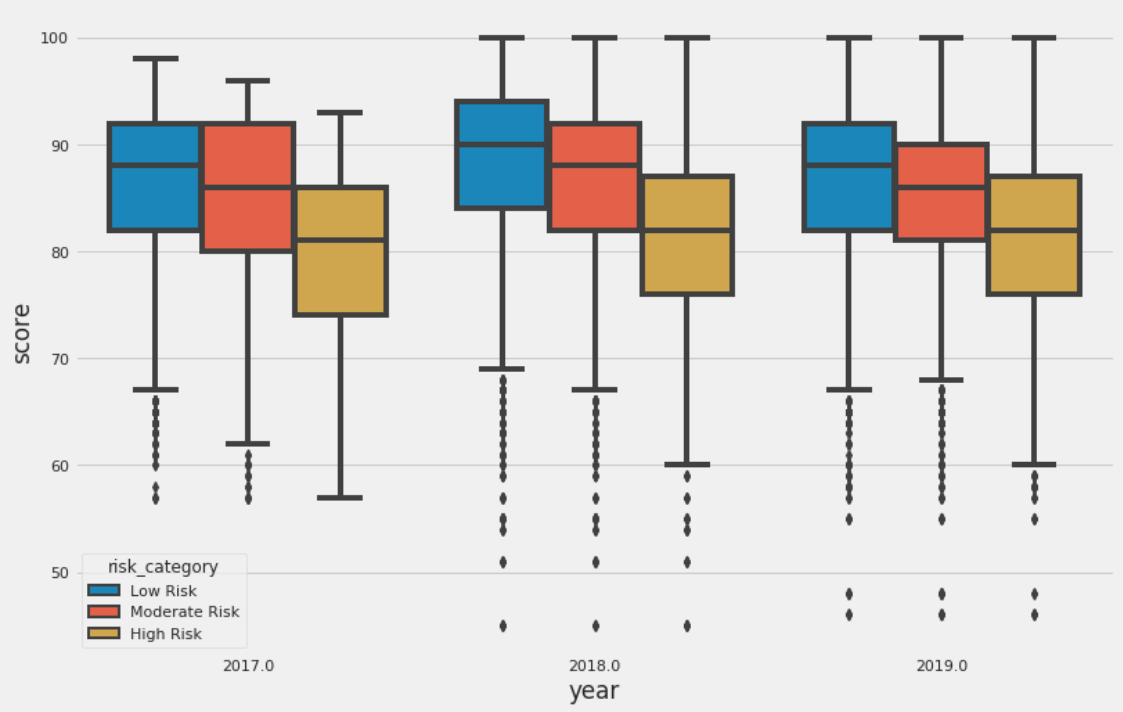
```
In [802]: table1 = vio.merge(ins2vio, how = "left", left_on = "vid", right_on = "vid")
          table2 = table1.merge(ins, how="left", left_on = "iid", right_on="iid")
          table2 = table2[table2["year"] != 2016.0]
          table2
```

```
Out[802]:                                              description  risk_category  \
          0      Consumer advisory not provided for raw or unde…  Moderate Risk
          1      Consumer advisory not provided for raw or unde…  Moderate Risk
```

```
2      Consumer advisory not provided for raw or unde…  Moderate Risk
3      Consumer advisory not provided for raw or unde…  Moderate Risk
4      Consumer advisory not provided for raw or unde…  Moderate Risk
…                                                    …              …
40204                          Worker safety hazards       Low Risk
40205                          Worker safety hazards       Low Risk
40207                          Worker safety hazards       Low Risk
40208                          Worker safety hazards       Low Risk
40209                          Worker safety hazards       Low Risk

         vid             iid                    date  score  \
0      103128  36861_20170504                     NaN    NaN
1      103128  80961_20180821  08/21/2018 12:00:00 AM   94.0
2      103128  62674_20190607  06/07/2019 12:00:00 AM   96.0
3      103128  85849_20181204  12/04/2018 12:00:00 AM   82.0
4      103128  57698_20190829  08/29/2019 12:00:00 AM   87.0
…       …             …                      …    …
40204  103159  89104_20171205  12/05/2017 12:00:00 AM   78.0
40205  103159  92642_20180613  06/13/2018 12:00:00 AM   92.0
40207  103159   1296_20170926  09/26/2017 12:00:00 AM   83.0
40208  103159  39047_20190515  05/15/2019 12:00:00 AM   94.0
40209  103159  38564_20170123  01/23/2017 12:00:00 AM   74.0

                       type      bid  timestamp    year  Missing Score
0                       NaN      NaN        NaT     NaN            NaN
1      Routine - Unscheduled  80961.0 2018-08-21  2018.0           94.0
2      Routine - Unscheduled  62674.0 2019-06-07  2019.0           96.0
3      Routine - Unscheduled  85849.0 2018-12-04  2018.0           82.0
4      Routine - Unscheduled  57698.0 2019-08-29  2019.0           87.0
…                        …      …        …     …             …
40204  Routine - Unscheduled  89104.0 2017-12-05  2017.0           78.0
40205  Routine - Unscheduled  92642.0 2018-06-13  2018.0           92.0
40207  Routine - Unscheduled   1296.0 2017-09-26  2017.0           83.0
40208  Routine - Unscheduled  39047.0 2019-05-15  2019.0           94.0
40209  Routine - Unscheduled  38564.0 2017-01-23  2017.0           74.0

[37619 rows x 11 columns]
```

In [803]:
```
plt.figure(figsize=(12, 8))
sns.boxplot(x=table2["year"], y=table2["score"], data=table2,
            orient="v", hue=table2["risk_category"], hue_order=["Low Risk", "Moderate Risk",
```

Out[803]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2dc53224c0>
```



In [804]:
```
# Do not modify this line
sns.set()
```

---

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 [805]: display(ins.head(), vio.head(), bus.head())
```

```
               iid                    date  score                    type  \
1   100010_20190403  04/03/2019 12:00:00 AM    100  Routine - Unscheduled
3   100017_20190816  08/16/2019 12:00:00 AM     91  Routine - Unscheduled
15  100041_20190520  05/20/2019 12:00:00 AM     83  Routine - Unscheduled
20  100055_20190425  04/25/2019 12:00:00 AM     98  Routine - Unscheduled
21  100055_20190912  09/12/2019 12:00:00 AM     82  Routine - Unscheduled
```

```
       bid  timestamp  year  Missing Score
1   100010 2019-04-03  2019            100
3   100017 2019-08-16  2019             91
15  100041 2019-05-20  2019             83
20  100055 2019-04-25  2019             98
21  100055 2019-09-12  2019             82
```

```
                                      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
```

```
      bid                          name                address           city  \
0    1000        HEUNG YUEN RESTAURANT            3279 22nd St  San Francisco
1  100010        ILLY CAFFE SF_PIER 39        PIER 39  K-106-B  San Francisco
2  100017  AMICI'S EAST COAST PIZZERIA             475 06th St  San Francisco
3  100026               LOCAL CATERING        1566 CARROLL AVE  San Francisco
4  100030            OUI OUI! MACARON  2200 JERROLD AVE STE C  San Francisco

  state postal_code     latitude    longitude  phone_number postal5
0    CA       94110    37.755282  -122.420493         -9999   94110
1    CA       94133 -9999.000000 -9999.000000   14154827284   94133
2    CA       94103 -9999.000000 -9999.000000   14155279839   94103
3    CA       94124 -9999.000000 -9999.000000   14155860315   94124
4    CA       94124 -9999.000000 -9999.000000   14159702675   94124
```

In [806]:
```
#YOUR CODE HERE
fun = ins.merge(bus, how="left", left_on="bid", right_on="bid")
fun = fun.iloc[:, [7, 8, 16]] #only select columns of interest--score, (name of restaurant), 
eachpost = fun.groupby(["postal5"]).agg(np.mean)
eachpost
#YOUR EXPLANATION HERE (in a comment)
#I wanted to answer the question of whether restaurants in certain postal codes
#(thus, "huddles" of regions within the city of San Francisco) had better inspection scores t
#I used the Missing Score column that we worked on earlier in the project to eliminate invali
#From the output, I have noticed that restaurants in the postal code 94129 have an average of
#which definitely stood out compared to the rest of the restaurants in different postal codes
```

Out[806]:
```
         Missing Score
postal5
94102        91.183967
94103        90.054276
94104        90.643312
```

94105	90.106346
94107	93.460499
94108	90.232514
94109	90.233407
94110	90.033333
94111	91.929254
94112	89.155172
94114	91.944934
94115	89.988327
94116	90.546729
94117	91.000000
94118	88.267829
94120	93.000000
94121	88.064965
94122	89.799263
94123	90.254902
94124	91.149746
94127	91.171975
94129	100.000000
94130	92.923077
94131	92.764331
94132	93.448485
94133	90.862605
94134	88.410138
94143	95.111111
94158	93.215385
94188	96.333333

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).

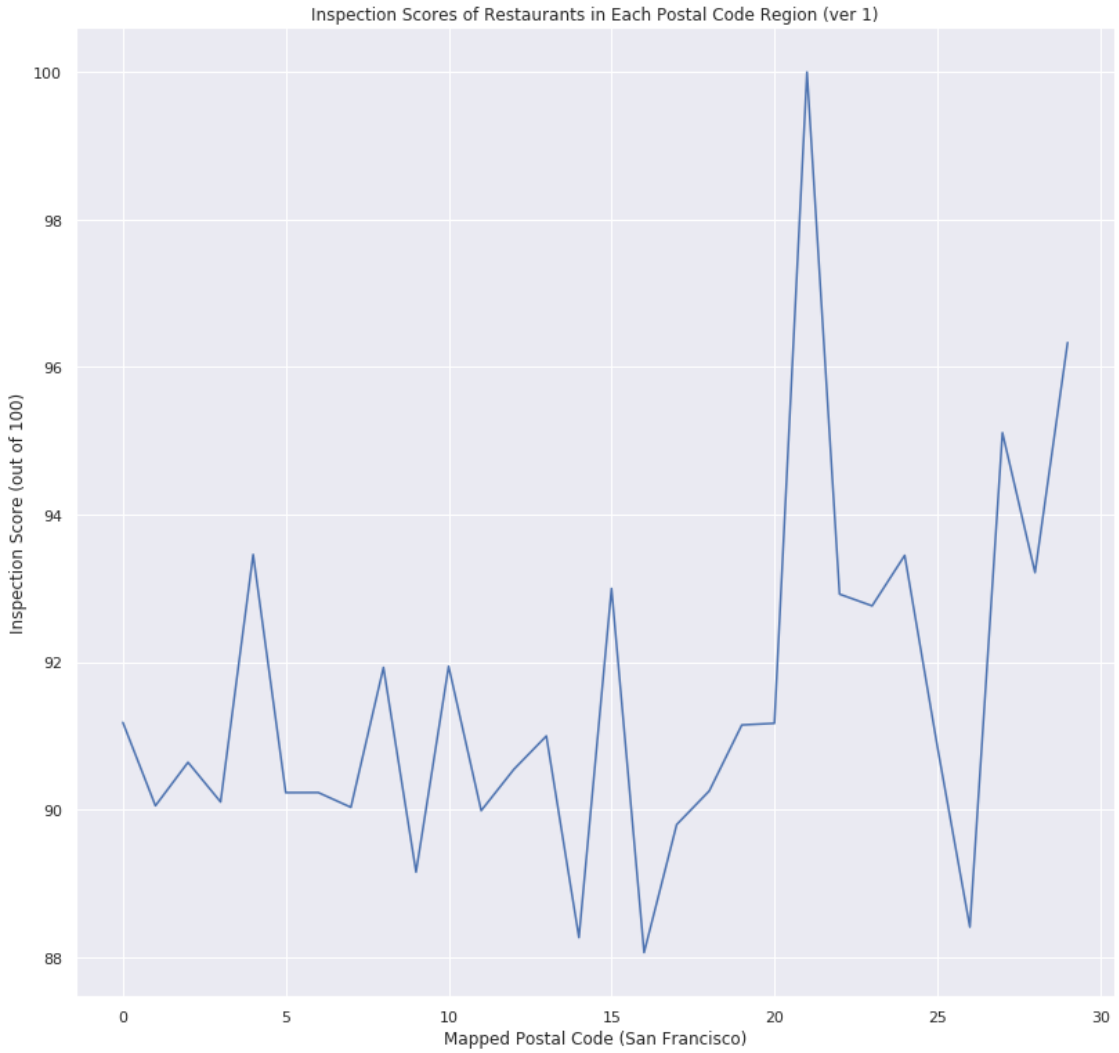
In [807]:
```
val = [i for i in range(len(eachpost.values))]
mappings = dict()
l = eachpost.index.tolist()
for i in val:
    mappings[i] = l[i]
mappings
```

Out[807]:
```
{0: '94102',
 1: '94103',
 2: '94104',
 3: '94105',
 4: '94107',
 5: '94108',
 6: '94109',
 7: '94110',
 8: '94111',
 9: '94112',
 10: '94114',
 11: '94115',
 12: '94116',
 13: '94117',
 14: '94118',
 15: '94120',
```

```
 16: '94121',
 17: '94122',
 18: '94123',
 19: '94124',
 20: '94127',
 21: '94129',
 22: '94130',
 23: '94131',
 24: '94132',
 25: '94133',
 26: '94134',
 27: '94143',
 28: '94158',
 29: '94188'}
```

In [808]:
```
# YOUR DATA PROCESSING AND PLOTTING HERE
plt.figure(figsize=(12, 12))
plt.plot(val, eachpost.values)
plt.xlabel("Mapped Postal Code (San Francisco)")
plt.ylabel("Inspection Score (out of 100)")
plt.title("Inspection Scores of Restaurants in Each Postal Code Region (ver 1)")
# YOUR EXPLANATION HERE (in a comment)

#The above mappings dictionary refers to the mapping of each integer key to its corresponding
#which is the postal code value. This was done in order to simplify the values displayed on t
#The first plot shows the mean inspection scores of restaurants in each postal code region (a
#where the y-axis has been automatically adjusted to best represent the graph.
#Here, we notice a huge spike near the "21" spot of the x-axis, which, as can be accessed fro
#postal code 94129. As noticed from the table computed in 8a, this postal code has an average
#which was higher than the other postal codes, so this explains the huge spike in the plot.
#Apart from that, there isn't a significant pattern in that most restaurants in the different
#hover around the 90-94 score rating apart from the outlier of 100 (postal code 94129).
```

Out[808]:
```
Text(0.5, 1.0, 'Inspection Scores of Restaurants in Each Postal Code Region (ver 1)')
```



In [809]:
```
plt.figure(figsize=(12, 12))
plt.ylim(0, 100)
plt.plot(val, eachpost.values)
plt.xlabel("Mapped Postal Code (San Francisco)")
plt.ylabel("Inspection Score (out of 100)")
plt.title("Inspection Scores of Restaurants in Each Postal Code Region (ver 2)")

#This second graph is very similar to the first one, except it computes the scores with the y
#This was computed to show that while in the first graph, the inspection score of restaurants
#looked significantly higher compared to the scores of other regions, looking at the "bigger"
#it is not necessarily so, as the spike definitely seems less noticeable than the first graph
#This graph indicates the importance of framing the axis of your data such that
#the graph can be computed to highlight or convey a certain point over another, a commonly no
#when interpreting data in the world of data science and statistics.
```

Out[809]: Text(0.5, 1.0, 'Inspection Scores of Restaurants in Each Postal Code Region (ver 2)')