A Research into NYC Restaurants' Food Safety

Yuan Yao (yy2884) & Yuming Wang (yw4054)

Abstract: NYC is a city with a huge population and diverse cultures. One result of this is the huge quantity of restaurants in NYC. On some important days (holiday, anniversary, friends gatherings), people would often dine in a restaurant instead of cooking on their own. And when choosing restaurants, an important criteria is food safety. Through this study, we tried to have a look into the food safety level in different NYC restaurants.

First we need to import some basic packages:

In [1]:

```
import pandas as pd
import numpy as np
import datetime
import re
import statsmodels.formula.api as smf
#This will be useful in plotting some graphs.
%matplotlib inline
import matplotlib.pyplot as plt
!pip install folium
import folium
from IPython.display import HTML
```

```
Requirement already satisfied: folium in /anaconda3/lib/python3.7/si
te-packages (0.7.0)
Requirement already satisfied: numpy in /anaconda3/lib/python3.7/sit
e-packages (from folium) (1.15.1)
Requirement already satisfied: branca>=0.3.0 in /anaconda3/lib/pytho
n3.7/site-packages (from folium) (0.3.1)
Requirement already satisfied: jinja2 in /anaconda3/lib/python3.7/si
te-packages (from folium) (2.10)
Requirement already satisfied: six in /anaconda3/lib/python3.7/site-
packages (from folium) (1.11.0)
Requirement already satisfied: requests in /anaconda3/lib/python3.7/
site-packages (from folium) (2.19.1)
Requirement already satisfied: MarkupSafe>=0.23 in /anaconda3/lib/py
thon3.7/site-packages (from jinja2->folium) (1.0)
Requirement already satisfied: urllib3<1.24,>=1.21.1 in /anaconda3/l
ib/python3.7/site-packages (from requests->folium) (1.23)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /anaconda3/l
ib/python3.7/site-packages (from requests->folium) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /anaconda3/lib/
python3.7/site-packages (from requests->folium) (2018.8.24)
Requirement already satisfied: idna<2.8,>=2.5 in /anaconda3/lib/pyth
on3.7/site-packages (from requests->folium) (2.7)
```

We found our data from *NYC open data* website. It's a dataframe about the inspection of the restaurants in NYC.

In [2]:

```
restaurant = pd.read_csv('~/Documents/Study/Data Bootcamp/DOHMH_New_York_City_Re
staurant_Inspection_Results.csv')
```

1. Data Cleaning

First let's have a look at our dataframe:

In [3]:

```
restaurant
```

Out[3]:

	CAMIS	DBA	BORO	BUILDING	STREET	ZIPCODE	
0	40931972	88 PALACE RESTAURANT	MANHATTAN	88	EAST BROADWAY	10002.0	212

1	50003390	NEW GOLDEN STAR RESTAURANT	BROOKLYN	638	BLAKE AVE	11207.0	718
2	40698807	MAMA MIA 44 SW	MANHATTAN	621	9 AVENUE	10036.0	212
3	41348161	DELICATESSEN MACBAR	MANHATTAN	54	PRINCE STREET	10012.0	212
4	41594717	AUNTIE ANNE'S PRETZELS	BROOKLYN	625	ATLANTIC AVENUE	11217.0	718
5	41491905	HILL COUNTRY CHICKEN	MANHATTAN	1123	BROADWAY	10010.0	646
6	50004809	BAR BACON	MANHATTAN	836	9TH AVE	10019.0	646
7	50033575	JACQUES TORRES ICE CREAM	MANHATTAN	89	E 42 ST	10017.0	212
8	50017223	FAIRFIELD INN & SUITES NEW YORK MANHATTAN FINA...	MANHATTAN	161	FRONT ST	10038.0	212
9	50013014	BE JUICE	MANHATTAN	93	3RD AVE	10003.0	212
10	41219576	EMPERADOR ELIAS RESTAURANT	BROOKLYN	274	BROADWAY	11211.0	718
11	40401002	JOHNNY MACK'S BAR	BROOKLYN	1114	8 AVENUE	11215.0	718
12	50042162	MILE 17	MANHATTAN	1446	1ST AVE	10021.0	212
13	41289876	EL MONTANERO BAKERY AND RESTAURANT	QUEENS	5521	MYRTLE AVENUE	11385.0	718

14	50060617	POQUITO PICANT	BROOKLYN	497	ATLANTIC AVE	11217.0	718
15	50049945	NEW SUKI SUSHI	BROOKLYN	9208	3RD AVE	11209.0	718
16	41678146	DESPIERTA CON ENERGIA/HERBALIFE	QUEENS	10442	CORONA AVENUE	11368.0	347
17	50042177	ABDULLAH SWEETS AND RESTAURANT	BROOKLYN	91	CHURCH AVE	11218.0	718
18	41508160	DUNKIN' DONUTS	MANHATTAN	266	1 AVENUE	10009.0	212
19	50037162	MEI PANDA HOUSE / TOP FRESH TORTILLAS	BRONX	277	E 206TH ST	10467.0	718
20	40999668	DUNKIN' DONUTS/BASKIN ROBBINS	MANHATTAN	269	8 AVENUE	10011.0	646
21	50053120	DELICIAS MEXICANOS	QUEENS	10214	ROOSEVELT AVE	11368.0	718
22	41702685	THE KEG ROOM	MANHATTAN	53	WEST 36 STREET	10018.0	212
23	50070312	LOUIE & ERNIE'S	BRONX	1300	CROSBY AVE	10461.0	718
24	40388218	WALKER'S RESTAURANT	MANHATTAN	16	NORTH MOORE STREET	10013.0	212
25	50060597	NU LOOK DELI	QUEENS	4916	VERNON BLVD	11101.0	718
26	40794103	MCDONALD'S	BRONX	1600	BRUCKNER BOULEVARD	10473.0	718

27	50042067	MALII	MANHATTAN	2028	2ND AVE	10029.0	212
28	41574435	PANDA KING HOUSE	QUEENS	7508	JAMAICA AVENUE	11421.0	718
29	50045872	PINA DULCE RESTAURANT	BROOKLYN	914	39TH ST	11219.0	718
...	...	...	...	...	...	...	
383137	50067074	BAZAR	MANHATTAN	31	W 26TH ST	10010.0	212
383138	41662314	JUSTINOS PIZZERIA	STATEN ISLAND	89	GUYON AVENUE	10306.0	718
383139	41714097	GANDHI INDIAN RESTAURANT	BROOKLYN	2032	BEDFORD AVENUE	11226.0	718
383140	41535549	PRONTO PIZZA	STATEN ISLAND	738	FOREST AVENUE	10310.0	718
383141	40363744	SONNY'S HEROS	BROOKLYN	1031	EAST 92 STREET	11236.0	718
383142	41698598	LULU JUICY FRUIT	BRONX	550	SOUTHERN BOULEVARD	10455.0	347
383143	41629883	CAFE MADELINE	BROOKLYN	1603	CORTELYOU ROAD	11226.0	718
383144	50015593	FUKUOKA SHABU-SHABU	QUEENS	6102	SPRINGFIELD BLVD	11364.0	718
383145	40654328	CAFE AL MERCATO	BRONX	2331	HUGHES AVENUE	10458.0	718
383146	50005075	APPLE JACK DINER	MANHATTAN	1725	BROADWAY	10019.0	212

383147	41046488	FRESH SALT	MANHATTAN	146	BEEKMAN STREET	10038.0	212
383148	41395526	CHASERS	QUEENS	6063	FLUSHING AVENUE	11378.0	718
383149	50072815	LA FONTANA SORELLENA	STATEN ISLAND	885	ANNADALE RD	10312.0	917
383150	50005692	LAYALY CAFE	QUEENS	4409	BROADWAY	11103.0	718
383151	41510846	218 RESTAURANT	MANHATTAN	218220	GRAND STREET	NaN	212
383152	41705075	DANIELA'S TRATTORIA	MANHATTAN	728	8 AVENUE	10036.0	212
383153	41217162	KINARAS INDIAN FOOD	BROOKLYN	368	MYRTLE AVENUE	11205.0	718
383154	50016961	NEW TOP TACO & CHINA	BROOKLYN	1654	SHEEPSHEAD BAY RD	11235.0	718
383155	41096132	TULCINGO	BROOKLYN	5520	5 AVENUE	11220.0	718
383156	40616799	AFGHAN KEBAB HOUSE #1	MANHATTAN	764	9 AVENUE	10019.0	212
383157	41254287	VINTAGE 61	MANHATTAN	36	PECK SLIP	10038.0	212
383158	41256036	KINGSTON BAKE SHOP	BROOKLYN	380	KINGSTON AVENUE	11225.0	718
383159	41392231	IVANA PIZZA	BRONX	2373	ARTHUR AVENUE	10458.0	718

383160	40401506	MINT	BROOKLYN	535	KINGS HIGHWAY	11223.0	718
383161	50051981	FIVESTUY CAFE	MANHATTAN	5	STUYVESANT OVAL	10009.0	347
383162	50043647	BODAI VEGETARIAN RESTAURANT	QUEENS	5908	MAIN ST	11355.0	718
383163	41621699	MI PASO CENTRO AMERICANO RESTAURANT	MANHATTAN	4129	BROADWAY	10033.0	212
383164	50033733	ICHIBANTEI	MANHATTAN	401	E 13TH ST	10009.0	646
383165	50040265	JERK PAN KUSINE	QUEENS	11302	SPRINGFIELD BLVD	11429.0	718
383166	50052504	MARKETPLACE AT MCGINLEY CENTER FORDHAM UNIVERS...	BRONX	441	E FORDHAM RD	10458.0	718

383167 rows × 18 columns

All of those column names are capitalized, some column names' meaning are hard to tell. so we should make some changes in the column names to make them more readable and clear. For future convenience, we are going to use lower case of those column names.

In [4]:

```
#create a list with all the column names.
list1=restaurant.columns
#turn all column names into lower case.
for i in list1:
    restaurant=restaurant.rename(columns={i:i.lower()})
#replace some column names to make them more understandable.
restaurant=restaurant.rename(columns={'dba':'name','cuisine description':'type'}
)
```

Since this is the inspection data of different restaurants, some restaurants have multiple data on different dates. If we apply this data directly in to our analysis, there would be potential errors.
For example, when analyzing the relationship between inspection scores and boroughs, too many inspections of the same restaurants with relatively higher scores would lead to an overestimation of the average score within a borough.
Therefore, we want to create a new dataframe, where very single restaurant only appears once, with the average scores of different restaurants.
In addition, as we hope to concentrate on recent trends, all the inspection data we choose are after 2015/1/1.

In [5]:

```
#drop the rows with no grade date and score.
restaurant2=restaurant.dropna(subset=['grade date','score'])
#select those rows with inspection date later than 2015/1/1
restaurant2['grade date'] = pd.to_datetime(restaurant2['grade date'])
mask = (restaurant2['grade date'] > '2015-1-1')
restaurant2=restaurant2.loc[mask]
#drop the irrelavent columns to make the rows of the same restaurant exactly the
same.
restaurant3=restaurant2.drop(['inspection date','action','violation code',
                              'violation description','critical flag','grade',
                              'grade date','record date','inspection type','scor
e'],axis=1)
#create a dataframe with only the average score of every restaurant.
camis_agg = restaurant2.groupby('camis').agg({'score':np.mean})
#drop the duplicates.
restaurant4=restaurant3.drop_duplicates(subset=['camis'], keep='first')
#merge and create the new dataframe
rescombo = pd.merge(restaurant4,camis_agg,on='camis')
rescombo = rescombo.rename(columns={'score':'avg_score'})
```

```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: Sett
ingWithCopyWarning:
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: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
```

Good job!
Now let's take a look at the DataFrame we have after cleaning:

In [6]:

```
rescombo
```

Out[6]:

	camis	name	boro	building	street	zipcode	phone
0	40931972	88 PALACE RESTAURANT	MANHATTAN	88	EAST BROADWAY	10002.0	2129418886
1	40698807	MAMA MIA 44 SW	MANHATTAN	621	9 AVENUE	10036.0	2123154582
2	41348161	DELICATESSEN MACBAR	MANHATTAN	54	PRINCE STREET	10012.0	2122260211
3	41594717	AUNTIE ANNE'S PRETZELS	BROOKLYN	625	ATLANTIC AVENUE	11217.0	7183984390
4	50004809	BAR BACON	MANHATTAN	836	9TH AVE	10019.0	6463620622
5	50033575	JACQUES TORRES ICE CREAM	MANHATTAN	89	E 42 ST	10017.0	2129837353
6	50013014	BE JUICE	MANHATTAN	93	3RD AVE	10003.0	2124886586
7	40401002	JOHNNY MACK'S BAR	BROOKLYN	1114	8 AVENUE	11215.0	7188327961
8	50042162	MILE 17	MANHATTAN	1446	1ST AVE	10021.0	2127721734
9	50060617	POQUITO PICANT	BROOKLYN	497	ATLANTIC AVE	11217.0	7182461700
10	50049945	NEW SUKI SUSHI	BROOKLYN	9208	3RD AVE	11209.0	7182382323
11	50037162	MEI PANDA HOUSE / TOP FRESH TORTILLAS	BRONX	277	E 206TH ST	10467.0	7186538888
12	40999668	DUNKIN' DONUTS/BASKIN ROBBINS	MANHATTAN	269	8 AVENUE	10011.0	6463968390
13	41702685	THE KEG ROOM	MANHATTAN	53	WEST 36 STREET	10018.0	2126431400
14	40388218	WALKER'S RESTAURANT	MANHATTAN	16	NORTH MOORE STREET	10013.0	2129410142
15	40794103	MCDONALD'S	BRONX	1600	BRUCKNER BOULEVARD	10473.0	7186170869
16	41574435	PANDA KING HOUSE	QUEENS	7508	JAMAICA AVENUE	11421.0	7182962172
17	50045872	PINA DULCE RESTAURANT	BROOKLYN	914	39TH ST	11219.0	7184355726
18	40841793	NELLY'S CAKES & PARTY SUPPLIES	BROOKLYN	597	NEW LOTS AVE	11207.0	7182726633
19	50003556	EASTERN RESTAURANT	BROOKLYN	1877	ROCKAWAY PKWY	11236.0	7182090649
20	40367005	DA VINCI PIZZA	BROOKLYN	6514	18 AVENUE	11204.0	7182325855

21	41670030	GRAMERCY BAGEL	MANHATTAN	246	3 AVENUE	10010.0	2123880080
22	50015640	LITTLE SKIPS OUTPOST	BROOKLYN	1158	MYRTLE AVE	11221.0	9292714311
23	40580969	CORAL DINER	MANHATTAN	3801	BROADWAY	10032.0	2129277545
24	41372809	LIBERTADOR	MANHATTAN	1725	2 AVENUE	10128.0	2123486222
25	41676904	PINCH FOOD DESIGN	MANHATTAN	545	WEST 27 STREET	10001.0	2122447000
26	50009466	GP SMOOTHIES & GIFTSHOP	BRONX	1152	CASTLE HILL AVE	10462.0	3479480957
27	41693374	ALEX CAFE & DELI	MANHATTAN	1018	LEXINGTON AVENUE	10021.0	2127449666
28	50045201	CHECKERS	QUEENS	12221	MERRICK BLVD	11434.0	7187122420
29	50048923	Milly's Pizzeria	BROOKLYN	834	BROADWAY	11206.0	7185994441
...	...	...	...	...	...	...	...
25239	50070600	PEELED N' PRESSED	BROOKLYN	808	UNION ST	11215.0	9292950550
25240	50074158	BARCLAYS - 745 CDR	MANHATTAN	745	7TH AVE	10019.0	2124123940
25241	41621393	SIDE STREET LOUNGE	BRONX	1330	BLONDELL AVENUE	10461.0	7184090001
25242	50073375	PENTAGRAM DESIGN	MANHATTAN	250	PARK AVE S	10003.0	2126837000
25243	50076463	LM &HY INC	QUEENS	13525	40TH RD	11354.0	2676166976
25244	50072244	MAISON BELJANSKI	MANHATTAN	317	E 53RD ST	10022.0	9292397313
25245	50077742	ASHOKA	MANHATTAN	1718	2ND AVE	10128.0	2128769100
25246	50069736	TRATTORIA ORA	QUEENS	1801	ASTORIA BLVD	11102.0	3478652685
25247	50069921	ELSEWHERE	BROOKLYN	599	JOHNSON AVE	11237.0	2032606095
25248	50081574	CHEN'S GARDEN	BROOKLYN	3920	9TH AVE	11232.0	9174067789
25249	50077286	KELLYS PUB	QUEENS	13611	41ST AVE	11355.0	7187957657
25250	50081066	LA CAFETERIA	QUEENS	9113	31ST AVE	11369.0	9293457571
25251	50072933	CHIP NYC	MANHATTAN	353	W 14TH ST	10014.0	9177676656
25252	50076091	ESAN	BROOKLYN	3041	AVENUE V	11229.0	7188775265
25253	50072182	DOUBLE CRISPY BAKERY I	MANHATTAN	230	GRAND ST	10013.0	2129666929
25254	50073050	TAJADAS COLUMBIANAS	BROOKLYN	241	BAY RIDGE AVE	11220.0	3474204082

25255	50068116	JAEWHON	QUEENS	7130	ROOSEVELT AVE	11372.0	3479353085
25256	50084075	YONGGOONG KALKOOKSOO	QUEENS	16324	NORTHERN BLVD	11358.0	9172852257
25257	50082148	SEOUL	MANHATTAN	11	W 32ND ST	10001.0	2129671678
25258	50074616	BOBWHITE LUNCH AND SUPPER COUNTER	MANHATTAN	570	LEXINGTON AVE	10022.0	9176553379
25259	50069329	ROGERS	QUEENS	203	BEACH 116TH ST	11694.0	7186343263
25260	50067514	ED'S ELBOW ROOM	MANHATTAN	308	E 78TH ST	10075.0	6466493764
25261	50058433	EAST & TACO	QUEENS	7917	MYRTLE AVE	11385.0	7183663667
25262	50073538	56TH & PARK F&B MANAGEMENT LLC	MANHATTAN	432	PARK AVE	10022.0	2122235412
25263	50076007	BLUE POINT BLEACHERS BAR 237	BRONX	1	E 161ST ST	10451.0	9172843260
25264	50072307	TRANS-PECOS	QUEENS	915	WYCKOFF AVE	11385.0	7183863718
25265	50074699	SUBWAY	BROOKLYN	400	MYRTLE AVE	11205.0	7187974394
25266	50073026	HIBACHI STATION	QUEENS	8312	37TH AVE	11372.0	6466332522
25267	50072728	DUNKIN DONUTS	QUEENS	3421	GREENPOINT AVE	11101.0	5164727100
25268	50060854	BARROW STREET THEATRE	MANHATTAN	27	BARROW ST	10014.0	2122436262

25269 rows × 9 columns

Now we have a dataframe with only one row for each restaurant, and the score is the average of all inspections. But there is no column for the us to know how they violated the regulations, so we still need a dataframe to include that.
We decided to use the column 'violation code', but we don't want to go too deep into different violation codes. According to the explanation of these codes we found online(https://www1.nyc.gov/assets/doh/downloads/pdf/rii/blue-book.pdf (https://www1.nyc.gov/assets/doh/downloads/pdf/rii/blue-book.pdf)), just by the numbers in the violation codes, we could have a basic idea of what kind of violation it is. For example, 02 means violation related to temperature, and 03 means violation related to food quality. So we can just keep the numerical parts.

In [7]:

```
#some of the violation codes are NaN, so we need to drop those rows
restaurant2=restaurant2.dropna(subset=['violation code'])
#reset the index.
restaurant2 = restaurant2.reset_index(drop=True)
#create a new column with only the numerical part in violation codes.
list2=[]
for i in range(0,189506):
    list2.append(restaurant2['violation code'][i])
    list2[i]=list2[i][0:2]
restaurant2['violation type']=pd.Series(list2)
#delete the original column
restaurant2=restaurant2.drop(['violation code'],axis=1)
restaurant2=restaurant2.dropna(subset=['zipcode'])
```

Now we have 3 data frame
restaurant is the original dataframe with some simple modifications
restaurant2 is the dataframe with only the inspection data after 2015-1-1, and the violation code replaced by violation type.
rescombo is the dataframe where each restaurant appears only once, and the score is the average score across all inspections.

2. Behind the Scores

The most objective and clearest indicator of the food safety of a restaurant is its score. Thanks to NYC Health Department, around 24,0000 restaurants' food handling, food temperature, personal hygiene, facility and equipment maintenance and vermin control are all carefully inspected. Every violation is associated with a certain number of points, and the restaurant's inspection score is the total points at the end of the inspection. **The lower the score, the better.** According to the Health Department, restaurants with a score between **0 and 13 points earn an A**, those with **14 to 27 points receive a B** and those with **28 or more a C**.
We are interested in where are the "good restaurants" (the restaurants with high food safety), and what makes a "good restaurant". Therefore, we want to explore the distribution and determination of the scores.

First, we are going to have a look at the overall situation of NYC food safety.

In [8]:

```
print('Average score for all NYC restaurants:',rescombo['avg_score'].mean())
A = rescombo[(rescombo['avg_score'] <= 13)]
B = rescombo[(rescombo['avg_score'] >13) & (rescombo['avg_score'] <=27 )]
C = rescombo[(rescombo['avg_score'] >27)]
print("Number of Grade A restaurants:",A.shape[0])
print("Number of Grade B restaurants:",B.shape[0])
print("Number of Grade C restaurants:",C.shape[0])
```

```
Average score for all NYC restaurants: 11.255175874805406
Number of Grade A restaurants: 20447
Number of Grade B restaurants: 4359
Number of Grade C restaurants: 463
```

So, for all restaurants in NYC, the average score is 11.26, nearly 81% of the restaurants have an overall grade of A. The overall health level of NYC restaurant is pretty good.

Which Borough to Eat in?

There are five boroughs in the New York City: Manhattan, Brooklyn, Queens, Bronx, and Staten Island. We want to explore the distribution of restaurant inspectin scores across boroughs, hoping to find some relationship between the food safety of a NYC restaurant and the borough in which it is located.

From our perspective, the average score of the restaurants within a borough can best reflect the overall food safety level of that specific borough. Therefore, we group our data by borough, and take the mean of the scores.

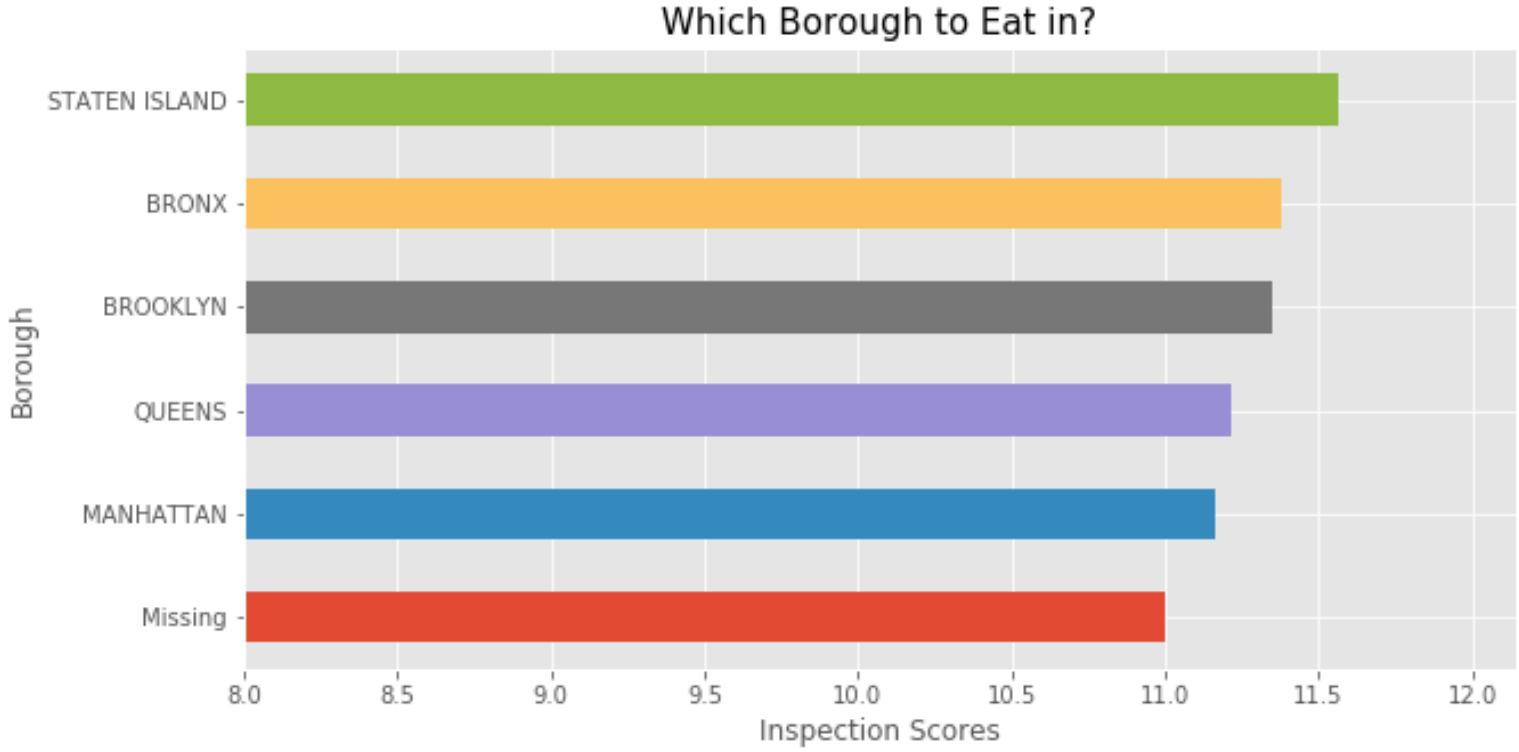
In [9]:

```
Score_by_boro = rescombo.groupby('boro')['avg_score'].mean()
print(Score_by_boro.sort_values())
#plot it so it will be more intuitive.
plt.style.use('ggplot')
fig, ax = plt.subplots()
Score_by_boro.sort_values().plot(kind='barh', ax=ax,figsize=(10,5))
ax.set_title('Which Borough to Eat in?', fontsize=15)
ax.set_xlabel('Inspection Scores')
ax.set_ylabel('Borough')
ax.set_xlim(xmin=8)
```

```
boro
Missing          11.000000
MANHATTAN        11.165765
QUEENS           11.213022
BROOKLYN         11.348740
BRONX            11.376621
STATEN ISLAND    11.564797
Name: avg_score, dtype: float64
```

Out[9]:

```
(8, 12.143036454908392)
```



As is shown in the bar chart, overall food safety of the five boroughs are basically at the same level, with Staten Island having the highest score (which indicates more violations), and Manhattan having the lowest. However, the differences are just marginal.
Also, based on the grading system listed above, all of the five boroughs receive an A for their overall food safety performance, which is definitely a good news!

Well, it seems hard to find solid evidances supporting differences between boroughs with regard to food safety.
We presume that this is because restaurants in these five boroughs of the New York City are basically competing in the same market, and supervised by the same authorities. These common factors contributes to there being no significant differences concerning food safety among New York restaurants.
However, there are yet a lot more to explore ...

Zipcode and Scores

Now, since there is no significant difference between different boroughs, we might as well dig deeper into differnt zipcode zones and try to see which parts of the city have the best (worst) food quality.

In [10]:

```
def create_map(table, zips, mapped_feature, add_text=""):
    nyc_geo = r'/Users/johnsonwang/Documents/Study/Data Bootcamp/f4129d9aa6dd4281bc98d0f701629b76nyczipcodetabulationareas.geojson.json'
    m = folium.Map(location = [40.7128, -74.0060], zoom_start=11)
    m.choropleth(
        geo_data=nyc_geo,
        fill_opacity=0.7,
        line_opacity=0.2,
        data=table,
        key_on = 'feature.properties.postalCode',
        columns=[zips,mapped_feature],
        fill_color='RdYlGn',
        legend_name=(' ').join(mapped_feature.split('_')).title()+' '+add_text+'Across NYC'
    )
    folium.LayerControl().add_to(m)
    m.save(outfile=mapped_feature+'_map.html')
    display(m)
```

In [11]:

```
NYC=pd.DataFrame(rescombo.groupby('zipcode',as_index=False)['avg_score'].mean())
```

In [12]:

```
NYC['zipcode']=NYC['zipcode'].astype(int)
NYC['zipcode']=NYC['zipcode'].astype(str)
```

In [13]:

```
create_map(NYC,'zipcode','avg_score','Borough')
```

```
/anaconda3/lib/python3.7/site-packages/folium/folium.py:432: FutureW
arning: The choropleth  method has been deprecated. Instead use the
new Choropleth class, which has the same arguments. See the example
notebook 'GeoJSON_and_choropleth' for how to do this.
  FutureWarning
```

Cuisine and Scores

What about the type of cuisine? Would the kind of food served by the restaurants have anything to do with food safety? That sounds an interesting aspect to probe into.
Before we start, let first take a look at how many cuisine types are there and how many restaurants each type have:

In [14]:

```
rescombo['type'].value_counts()
```

Out[14]:

```
American
5837
Chinese
2375
CafÃ©/Coffee/Tea
1677
Pizza
```

1176

Italian
979
Mexican
916
Japanese
849
Latin (Cuban, Dominican, Puerto Rican, South & Central American)
812
Bakery
717
Caribbean
671
Spanish
625
Donuts
530
Pizza/Italian
479
Chicken
478
Juice, Smoothies, Fruit Salads
409
Asian
401
Sandwiches
391
Hamburgers
376
Ice Cream, Gelato, Yogurt, Ices
333
Indian
325
Jewish/Kosher
323
French
320
Korean
300
Thai
293
Delicatessen
289
Mediterranean
266
Sandwiches/Salads/Mixed Buffet
239
Other
226
Seafood
193
Irish
190

...
Brazilian
29
Armenian
28
Hotdogs
28
Polish
27
Hotdogs/Pretzels
26
Australian
23
Creole
22
Chinese/Cuban
18
Ethiopian
16
English
15
Afghan
15
Pancakes/Waffles
14
Egyptian
14
Moroccan
11
Indonesian
9
Portuguese
9
Cajun
7
Not Listed/Not Applicable
7
Creole/Cajun
6
Soups
6
Southwestern
5
Scandinavian
5
Fruits/Vegetables
5
Nuts/Confectionary
4
Californian
3
Czech

```
3

Iranian
3
Chilean
1
Polynesian
1
Basque
1
Name: type, Length: 85, dtype: int64
```

Well, there are a total of 85 different types of cuisines in our dataset, and that would be too much for us to cope with. Some of our cuisines only have a few restaurants around NYC, they wouldn't be representative. So we decided to delete those cuisines with less than 50 restaurants.

Best Performing and Worst Performing Cuisine Types

In [15]:

```
#add a column with the number "1" to each data line for counting.
rescombo['count'] = 1
#create a dataframe that consist of the score and number for each type of restau
rant.
type_agg = rescombo.groupby('type').agg({'avg_score':np.mean,'count':np.sum})
#sort out those types with more than 50 restaurants.(delete small samples)
type_agg = type_agg.loc[type_agg['count']>50,:]
#sort them by their average score, from small to big.
type_agg.sort_values(by=['avg_score'])
```

Out[15]:

type	avg_score	count
Donuts	8.640052	530
CafÃ©/Coffee/Tea	9.203118	1677
Ice Cream, Gelato, Yogurt, Ices	9.273242	333
Sandwiches	9.397164	391
Sandwiches/Salads/Mixed Buffet	9.610423	239
Other	9.696453	226
Hamburgers	9.737953	376
Salads	9.816155	75
Bottled beverages, including water, sodas, juices, etc.	9.991082	102
Juice, Smoothies, Fruit Salads	10.483352	409

American	10.770475	5837
Tex-Mex	10.784144	139
Continental	10.801696	52
French	10.876421	320
Vegetarian	10.983645	112
Bakery	11.070206	717
Pizza	11.080506	1176
Bagels/Pretzels	11.142852	173
Chicken	11.168262	478
Irish	11.182732	190
Russian	11.395584	71
Eastern European	11.453185	73
Mediterranean	11.497404	266
Seafood	11.550182	193
Italian	11.669477	979
Pizza/Italian	11.685414	479
Greek	11.804771	143
Middle Eastern	11.864508	177
Japanese	11.874613	849
Soul Food	11.964494	60
Chinese	11.988011	2375
Mexican	12.058089	916
Steak	12.093866	82
Thai	12.400091	293
Jewish/Kosher	12.437701	323
Asian	12.450688	401
Korean	12.506828	300
Vietnamese/Cambodian/Malaysia	12.652363	86
Caribbean	12.838229	671
Delicatessen	12.947688	289
Spanish	13.164288	625
Latin (Cuban, Dominican, Puerto Rican, South & Central American)	13.237098	812
Indian	13.375021	325
Turkish	13.791381	64

African	13.905366	69
Peruvian	14.659722	76

We've sorted them from according to the average score, from small to big (best performing to worst performing). Now we see, the best performing type of restaurants is the **Donuts**,with an average score of only 8.64, despite their huge inspection numbers. And the worst performing one is **Peruvian**, with a score of 14.65.
Let's have a closer look at those two types.

First let's look at donut types.

In [16]:

```
#create a subset of only "donuts" type restaurants.
donuts=rescombo[rescombo.type=='Donuts']
print(donuts['avg_score'].describe())
donuts['name'].value_counts()
```

```
count    530.000000
mean       8.640052
std        3.010243
min        0.000000
25%        6.892857
50%        8.700000
75%       10.276786
max       24.500000
Name: avg_score, dtype: float64
```

Out[16]:

```
DUNKIN' DONUTS                          335
DUNKIN' DONUTS, BASKIN ROBBINS           86
DUNKIN DONUTS                            44
DUNKIN DONUTS & BASKIN ROBBINS            6
DUNKIN' DONUTS/ BASKIN ROBBINS            4
DUNKIN' DONUTS/BASKIN ROBBINS             4
DUNKIN DONUTS BASKIN ROBBINS              3
TWIN DONUT                                2
DOUGHNUTTERY                              2
DOUGHNUT PLANT                            2
DUNKIN' DONUTS BASKIN ROBBINS             2
DUN-WELL DOUGHNUTS                        2
THE DOUGHNUT PROJECT                      2
DOUGH                                     2
COUNTRY DONUTS                            2
DUNKIN DONUTS,BASKIN ROBBINS              1
Dunkin Donuts                             1
Dunkin Donuts / Baskin Robbins            1
DUNKIN DONUTS / BASKIN ROBBINS            1
7TH AVENUE DONUT SHOP                     1
DONUT PUB                                 1
DUNKIN DONUTS/BASKIN ROBBINS              1
DUNKIN' DONUTS, BASKINS ROBBINS           1
TAJ DONUT SHOP                            1
MIKE'S DONUTS                             1
DU'S DONUTS AND COFFEE                    1
DUNKIN' DONUTS - BASKIN ROBBINS           1
DUNKIN DONUTS/POPEYES                     1
DUNKIN' DONUTS & BASKIN ROBINS            1
CUZIN'S DUZIN                             1
DUNKIN' DONUTS-BASKIN ROBBINS             1
Underwest Donuts                          1
DUNKIN DONUTS ARRIVAL                     1
DONUT CONNECTIONS                         1
DUNKIN' DONUTS/SUBWAY                     1
DUNKIN' DONUTS/HUDSON NEWS                1
DUNKIN' DONUTS/Baskin Robbins             1
NOSTRAND DONUT SHOP                       1
KRISPY KREME                              1
Dunkin' Donuts/Baskin Robbins             1
HILLSIDE GOURMET                          1
DUNKIN DONUTS (west food court)           1
DUNKIN DONUTS-BASKIN ROBBINS              1
DUNKIN' DONUTS, BASKIN ROBBINS, SUBWAY    1
DUNKIN DONUTS/BASKIN ROBINS               1
DUNKIN DONUT                              1
DUNKIN' DONUTS/BR EXPRESS                 1
Name: name, dtype: int64
```

A lot of the these inspection datas are for Dunkin Donut. But their names are written in different ways. Let's try to gather them together and analyze them.

In [17]:

```
#create a dataframe for Dunkin Donuts
names=['DUNKIN','Dunkin']
dunkin=donuts[donuts['name'].str.contains('|'.join(names))]
print('Number of Dunkin Donuts:',dunkin.shape[0])
print('Deviation from the average donuts score:',dunkin['avg_score'].mean()-8.64
)
print('Donkin Donuts score description:\n',dunkin['avg_score'].describe())
#create a dataframe without dunkin donuts
without_dun=donuts[~donuts.name.str.contains('DUNKIN')][~donuts.name.str.contain
s('Dunkin')]
print('Non-Donkin Donuts score description:\n',without_dun['avg_score'].describe
())
```

```
Number of Dunkin Donuts: 505
Deviation from the average donuts score: -0.03011563793741523
Donkin Donuts score description:
 count    505.000000
mean       8.609884
std        2.990601
min        0.000000
25%        6.857143
50%        8.666667
75%       10.166667
max       24.500000
Name: avg_score, dtype: float64
Non-Donkin Donuts score description:
 count    25.000000
mean      9.249434
std       3.392404
min       2.500000
25%       7.222222
50%       9.875000
75%      11.285714
max      15.952381
Name: avg_score, dtype: float64

/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: User
Warning: Boolean Series key will be reindexed to match DataFrame ind
ex.
```

Among the 530 donuts restaurants, more than **95%** are Dunkin Donuts. And their **average score (8.61) is significantly lower** than the average score of other donut restaurants(9.24).

In [18]:

```
#create a subset of only "Peruvian" type restaurants.
peru=rescombo[rescombo.type=='Peruvian']
print(peru['avg_score'].describe())
peru['name'].value_counts()
```

```
count    76.000000
mean     14.659722
std       8.126878
min       7.000000
25%      10.104167
50%      12.000000
75%      17.014706
max      64.777778
Name: avg_score, dtype: float64
```

Out[18]:

```
DON ALEX RESTAURANT                  5
PIO PIO                              4
RIKO                                 2
RIKO PERUVIAN CUISINE                2
EL POLLO INKA PERU                   2
DON POLLO                            2
EL ANZUELO FINO RESTAURANT           1
MISTURA PERUANA                      1
CONEY ISLAND TASTE                   1
INTI N.Y.C.                          1
DESNUDA CEVICHERIA                   1
LIMA                                 1
INCA PERUVIAN GRILL KITCHEN          1
MISSION CEVICHE                      1
JORA RESTAURANT AND BAR              1
SABOR PERUANO II                     1
TU CASA RESTAURANT                   1
RAYMI/LATIN BEET KITCHEN             1
KANTU PERUVIAN CUISINE               1
PANCA                                1
SURFISH BISTRO                       1
MACA RESTAURANT                      1
COCO ROCO RESTAURANT                 1
LA BRASA PERUANA                     1
PIO PIO EXPRESS                      1
CUZCO PERU BBQ CHICKEN               1
SUPER POLLO                          1
PIO PIO BROOKLYN                     1
POLLO INKA PERU                      1
El Chalaco                           1
                                    ..
SABOR PERUANO RESTAURANT             1
AMARU                                1
LIMA RESTAURANT                      1
```

```
EL RINCON PERUANO RESTAURANT            1
CHIFA RESTAURANT                        1
COSTA VERDE                             1
URUBAMBA                                1
LIMA PERUVIAN RESTAURANT                1
BABY BRASA                              1
LA CABANA PERUANA                       1
EL ANCLA DE ASTORIA RESTAURANT & BAR    1
LA CASA DEL POLLO PERUANO II            1
PRIMA DONNA PERUVIAN RESTAURANT         1
CUZCO PERUVIAN  RESTAURANT              1
BIENVENIDOS AL CALLAO RESTAURANT        1
COLONIA VERDE                           1
LA CHACRA                               1
GILDA'S RESTAURANT                      1
CHIMU                                   1
LOS NISPEROS PERUVIAN RESTAURANT        1
QUECHUA NOSTRA                          1
Anzuelo Fino                            1
THE INKAN                               1
THE PATIO RESTAURANT                    1
LA UNION RESTAURANT                     1
NAZCA PERUVIAN RESTAURANT               1
LIMA 33                                 1
CHIRP                                   1
PIURA                                   1
SOL INCA                                1
Name: name, Length: 65, dtype: int64
```

Unlike donuts, there isn't a very famous Peruvian restaurant that has many chained restaurants. so we are not going to keep digging into that.

Major Cuisine Types

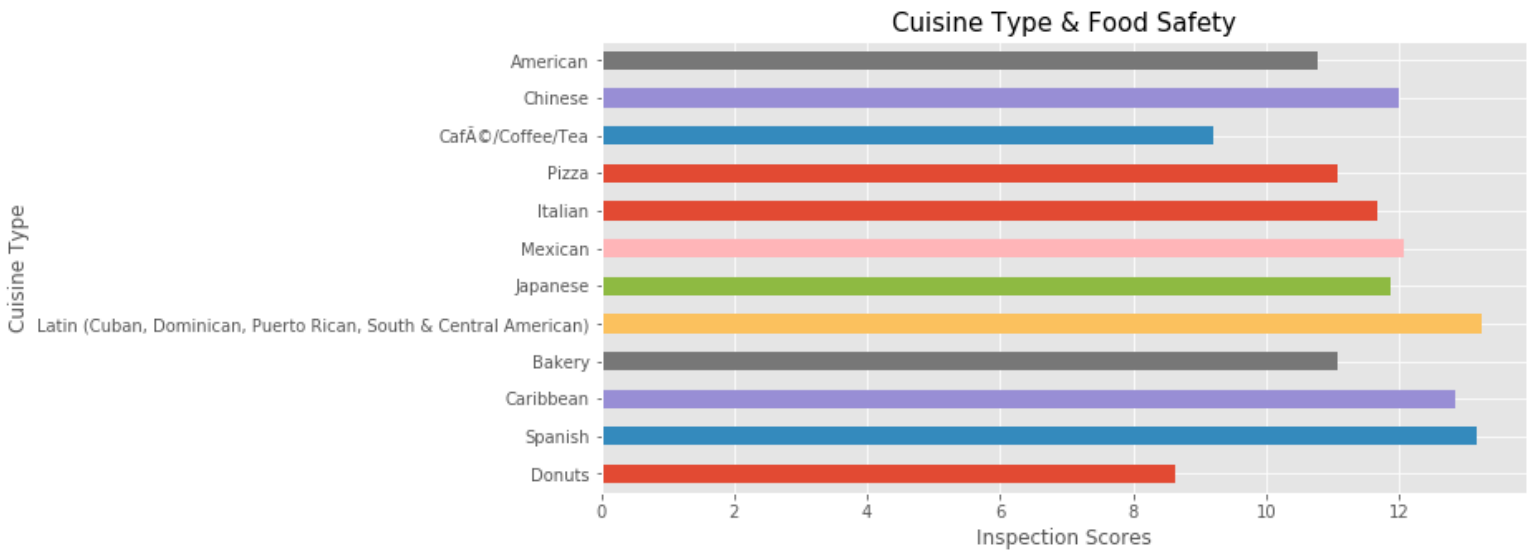
Now we are going to look at the some major cuisines (more than 500 restaurants) in NYC.

In [19]:

```
#selecting the cuisine types with more than 500 restaurants
Score_by_type = type_agg.loc[type_agg['count']>500,:]
Score_by_type=Score_by_type.sort_values(by=['count'],ascending=True)
#plot it out
plt.style.use('ggplot')
fig, ax = plt.subplots()
Score_by_type['avg_score'].plot(kind='barh', ax=ax,figsize=(10,5))
ax.set_title('Cuisine Type & Food Safety', fontsize=15)
ax.set_xlabel('Inspection Scores')
ax.set_ylabel('Cuisine Type')
```

Out[19]:

```
Text(0,0.5,'Cuisine Type')
```



We sort the cuisine types according to their restaurant numbers, from the most (American) to the least (Donuts), and there seems to be **no relationship between the number of restaurant and the average inspection score.**

Among those major cuisines, the best perfoming one is still donuts, while the worst performing one is the Latin. only two types of major cuisines **(Latin & Spanish) have average score of more than 13 (Grade B)**, **Cafe and Donuts have significantly lower average scores** than other cuisines.

And based on those trends and numbers we had an assumption: the cuisine that takes a long procedure to make is more likely to have a higher score, a lower grade, while foods that can be made fairly quickly wouldn't have many chances to violate the food safety law.

3. Behind the Violation Types

During the inspection of those restaurants, NYC Health Department have given different restaurants their violation codes to help consumers see what kind of inappropriate action those restaurants have made. In total, there are 99 different violation codes. But as mentioned before, we don't want to dig too deep into this. So we decided to use just the numerical part as indicators of what kind of violation it is.

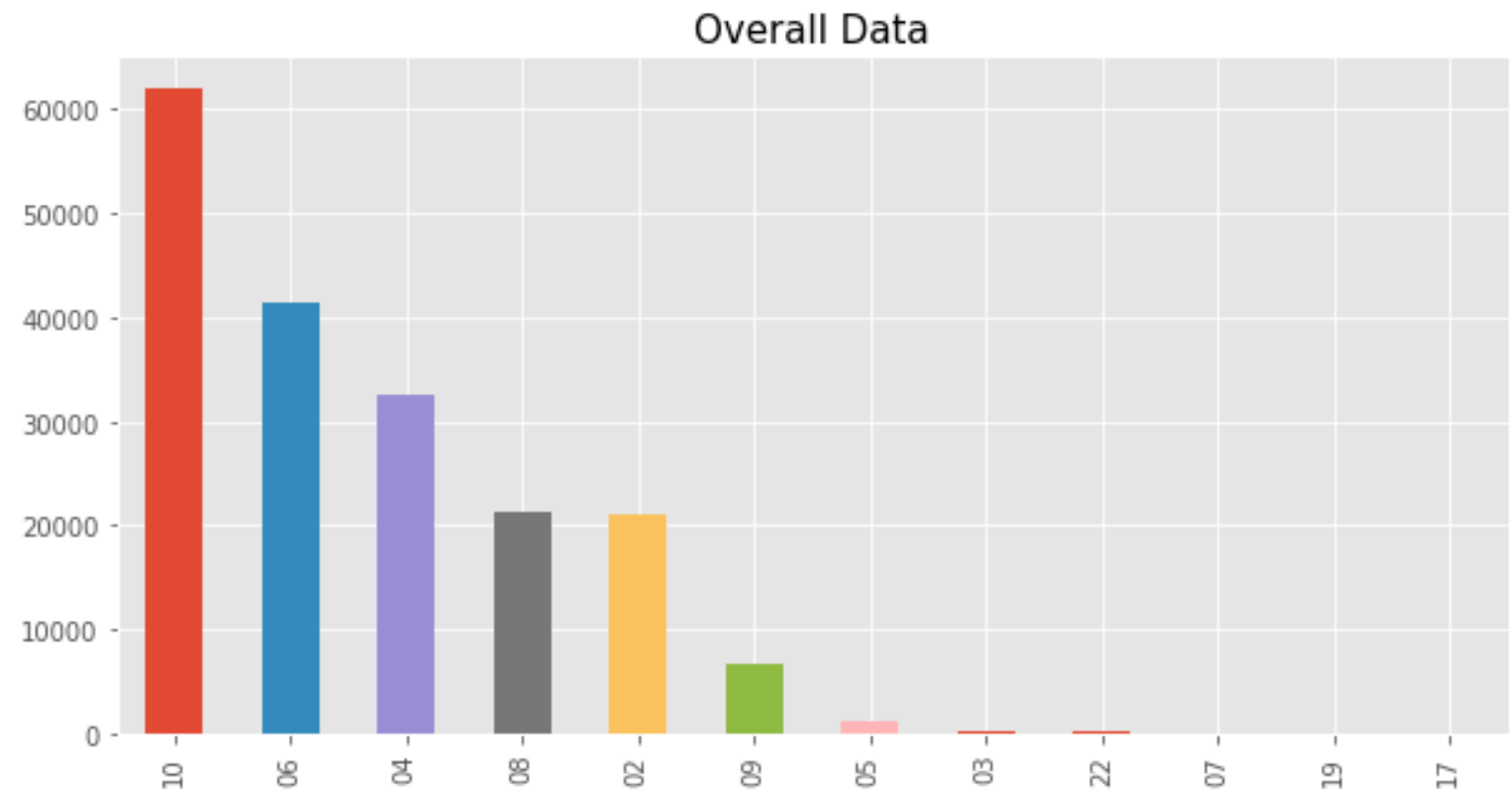
Violation types could also tell us something interesting. First, let's have a look at the overall trend.

Overall Trend

In [20]:

```
plt.style.use('ggplot')
fig, ax = plt.subplots()
restaurant2['violation type'].value_counts().plot(kind='bar',ax=ax,figsize=(10,5
))
ax.set_title('Overall Data',size=15)
print(restaurant2['violation type'].value_counts())
```

```
10    61879
06    41326
04    32414
08    21365
02    21063
09     6679
05     1183
03      335
22      266
07        6
19        1
17        1
Name: violation type, dtype: int64
```



Overall, the violation code appeared most is 10 (environment), that is over 21000 violations more than the second most: 06 (workers), almost the size of the fourth and fifth violation code: 08 (pest) and 02 (temperature).
04(contamination of food) is worth noticing as well, more than 30000 times of violation. Apart from these, other violation codes are relatively small, with less than 10000 violations.

Within the Boroughs

In [21]:

```
#create 5 dataframe with different boroughs.
manhattan_type=restaurant2[restaurant2.boro=='MANHATTAN']
brooklyn_type=restaurant2[restaurant2.boro=='BROOKLYN']
bronx_type=restaurant2[restaurant2.boro=='BRONX']
queens_type=restaurant2[restaurant2.boro=='QUEENS']
sta_island_type=restaurant2[restaurant2.boro=='STATEN ISLAND']

#count_values on the violation code in different boroughs.
```

```
a=manhattan_type['violation type'].value_counts()

df_mht=pd.DataFrame([a]).transpose()
b=brooklyn_type['violation type'].value_counts()
df_bkl=pd.DataFrame([b]).transpose()
c=bronx_type['violation type'].value_counts()
df_brx=pd.DataFrame([c]).transpose()
d=queens_type['violation type'].value_counts()
df_qns=pd.DataFrame([d]).transpose()
e=sta_island_type['violation type'].value_counts()
df_sta=pd.DataFrame([e]).transpose()

#create a new column in those dataframes, including the fraction of each violati
on codes.
df_mht['manhattan']=df_mht['violation type']/np.sum(df_mht['violation type'],axi
s=0)
df_bkl['brooklyn']=df_bkl['violation type']/np.sum(df_bkl['violation type'],axis
=0)
df_brx['bronx']=df_brx['violation type']/np.sum(df_brx['violation type'],axis=0)
df_qns['queens']=df_qns['violation type']/np.sum(df_qns['violation type'],axis=0
)
df_sta['staten island']=df_sta['violation type']/np.sum(df_sta['violation type']
,axis=0)

#turn the index into a new column, turn the type into integer.
df_mht['violation code']=df_mht.index.astype(str).astype(int)
df_bkl['violation code']=df_bkl.index.astype(str).astype(int)
df_brx['violation code']=df_brx.index.astype(str).astype(int)
df_qns['violation code']=df_qns.index.astype(str).astype(int)
df_sta['violation code']=df_sta.index.astype(str).astype(int)

#sort those dataframes based on violation code.
df_mht=df_mht.sort_values(by=['violation code'])
df_bkl=df_bkl.sort_values(by=['violation code'])
df_brx=df_brx.sort_values(by=['violation code'])
df_qns=df_qns.sort_values(by=['violation code'])
df_sta=df_sta.sort_values(by=['violation code'])

#merge those dataframe based on the violation code
vio_type=df_mht.merge(df_bkl,left_on='violation code', right_on='violation code'
, how='outer')
vio_type=vio_type.merge(df_brx,left_on='violation code', right_on='violation cod
e', how='outer')
vio_type=vio_type.merge(df_qns,left_on='violation code', right_on='violation cod
e', how='outer')
vio_type=vio_type.merge(df_sta,left_on='violation code', right_on='violation cod
e', how='outer')

#drop the counts of violation type, just leave the fraction of them
vio_type=vio_type.drop(['violation type_x','violation type_y','violation type'],
axis=1)
vio_type.fillna(0, inplace=True)
vio_type=vio_type.set_index('violation code')
```

In [22]:

```
vio_type
```

Out[22]:

violation code	manhattan	brooklyn	bronx	queens	staten island
2	0.111469	0.114189	0.116778	0.111488	0.119723
3	0.001697	0.001903	0.001052	0.002165	0.001695
4	0.166993	0.182950	0.204463	0.165043	0.161479
5	0.005920	0.006994	0.005842	0.006636	0.005855
6	0.230650	0.207139	0.186938	0.232811	0.239599
7	0.000027	0.000086	0.000000	0.000000	0.000000
8	0.109337	0.123150	0.133427	0.109205	0.097072
9	0.033961	0.033921	0.029326	0.042849	0.041448
10	0.338167	0.327637	0.321650	0.329074	0.332820
17	0.000014	0.000000	0.000000	0.000000	0.000000
19	0.000014	0.000000	0.000000	0.000000	0.000000
22	0.001752	0.002032	0.000526	0.000729	0.000308

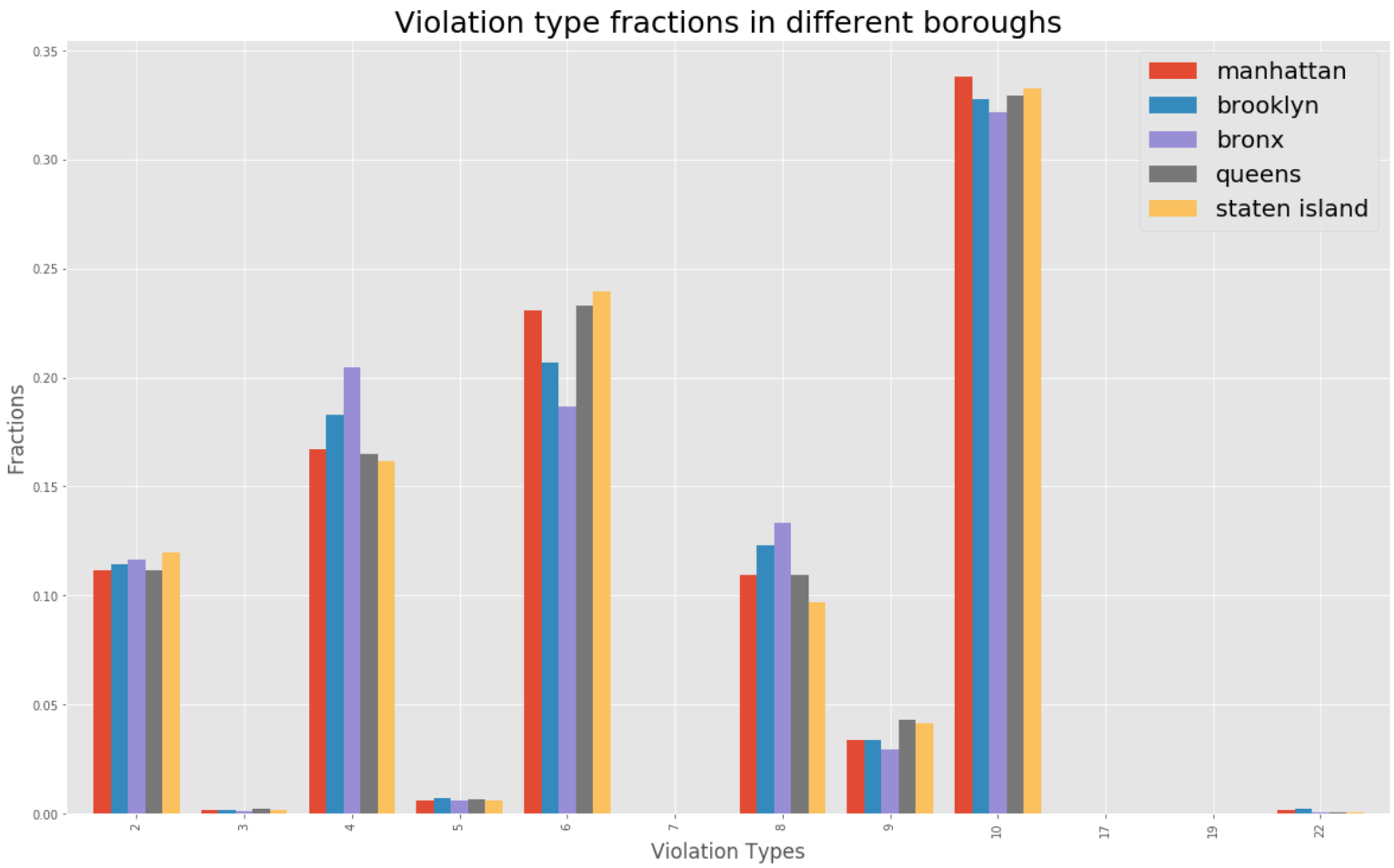
Now we have our database, let's try to plot it out.

In [23]:

```
plt.style.use('ggplot')
fig, ax = plt.subplots()
vio_type.plot(kind='bar', width=0.8,ax=ax,figsize=(20,12))
ax.set_title('Violation type fractions in different boroughs', fontsize=25)
ax.set_xlabel('Violation Types',fontsize=17)
ax.set_ylabel('Fractions',fontsize=17)
ax.legend(fontsize=20)
```

Out[23]:

```
<matplotlib.legend.Legend at 0x1a1ec864e0>
```



Cuisine Types and Violation Types

Will violation types have anything to do with the cuisine types?
Hypothetically, we think they are related.
For example, for Japanese restaurants, an important part of their food is sushi, thus it's less likely that they are going to violate rules with regards to temperature. Or, some Indian dishes might require a lot of manual work, while American restaurants and Cafe may just have a short cooking procedure. Thus Indian restaurants might have more violations towards workers.

To justify those assumptions, we need to look at different cuisines individually. Of course we are not going to look at every single cuisine type. First let's have a look at the best performing one: donut, and the worst performing one:Peruvian.

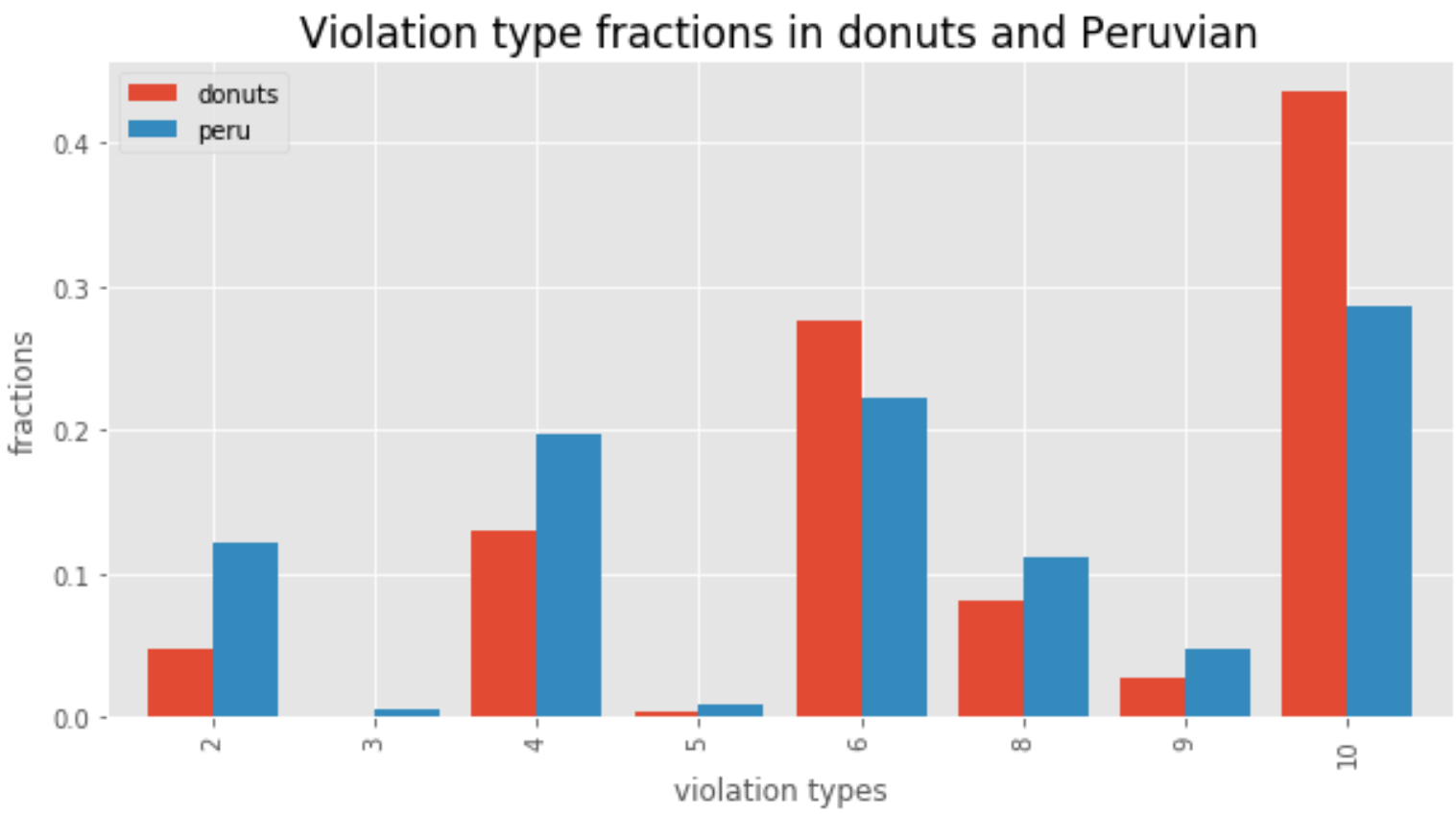
Best Performing and Worst Performing Ones

In [24]:

```
#create a dateframe for donuts.
donuts_type=restaurant2[restaurant2.type=='Donuts']
donuts_type=donuts_type['violation type'].value_counts()
df_donuts=pd.DataFrame([donuts_type]).transpose()
df_donuts['donuts']=df_donuts['violation type']/np.sum(df_donuts['violation type
'],axis=0)
df_donuts['violation code']=df_donuts.index.astype(str).astype(int)
df_donuts=df_donuts.sort_values(by=['violation code'])

#create a dataframe for Peruvian
peru_type=restaurant2[restaurant2.type=='Peruvian']
peru_type=peru_type['violation type'].value_counts()
df_peru=pd.DataFrame([peru_type]).transpose()
df_peru['peru']=df_peru['violation type']/np.sum(df_peru['violation type'],axis=
0)
df_peru['violation code']=df_peru.index.astype(str).astype(int)
df_peru=df_peru.sort_values(by=['violation code'])

#merge these two dataframes based on violation codes.
vio_cuisine_type=df_donuts.merge(df_peru,left_on='violation code', right_on='vio
lation code', how='outer')
vio_cuisine_type=vio_cuisine_type.drop(['violation type_x','violation type_y'],a
xis=1)
vio_cuisine_type=vio_cuisine_type.set_index('violation code')

#plt.style.use('ggplot')
fig, ax = plt.subplots()
vio_cuisine_type.plot(kind='bar', width=0.8,ax=ax,figsize=(10,5))
ax.set_title('Violation type fractions in donuts and Peruvian', fontsize=17)
ax.set_xlabel('violation types')
ax.set_ylabel('fractions')
```

Out[24]:

```
Text(0,0.5,'fractions')
```



In [25]:

```
vio_cuisine_type
```

Out[25]:

violation code	donuts	peru
2	0.047499	0.121658
3	0.000281	0.005348
4	0.129005	0.196524
5	0.003373	0.009358
6	0.276279	0.221925
8	0.080944	0.110963
9	0.027263	0.048128
10	0.435357	0.286096

Four Major Cuisine Types

After looking at the best performing and worst performing one, let's look at some of the largest cuisines. We decided to look at the 4 cuisines with the largest number of restaurants: American(5837),Chinese(2375),Cafe(1677),and Pizza(1176).

In [26]:

```
#create a dataframe for American
american_type=restaurant2[restaurant2.type=='American']
american_type=american_type['violation type'].value_counts()
df_american=pd.DataFrame([american_type]).transpose()
df_american['American']=df_american['violation type']/np.sum(df_american['violat
ion type'],axis=0)
df_american['violation code']=df_american.index.astype(str).astype(int)
df_american=df_american.sort_values(by=['violation code'])

#create a dataframe for Chinese
cn_type=restaurant2[restaurant2.type=='Chinese']
cn_type=cn_type['violation type'].value_counts()
df_cn=pd.DataFrame([cn_type]).transpose()
df_cn['Chinese']=df_cn['violation type']/np.sum(df_cn['violation type'],axis=0)
df_cn['violation code']=df_cn.index.astype(str).astype(int)
df_cn=df_cn.sort_values(by=['violation code'])

#create a dataframe for cafe
cafe_type=restaurant2[restaurant2.type=='CafÃ©/Coffee/Tea']
cafe_type=cafe_type['violation type'].value_counts()
df_cafe=pd.DataFrame([cafe_type]).transpose()
df_cafe['Cafe']=df_cafe['violation type']/np.sum(df_cafe['violation type'],axis=
0)
df_cafe['violation code']=df_cafe.index.astype(str).astype(int)
df_cafe=df_cafe.sort_values(by=['violation code'])

#create a dataframe for pizza
pizza_type=restaurant2[restaurant2.type=='Pizza']
pizza_type=pizza_type['violation type'].value_counts()
df_pizza=pd.DataFrame([pizza_type]).transpose()
df_pizza['Pizza']=df_pizza['violation type']/np.sum(df_pizza['violation type'],a
xis=0)
df_pizza['violation code']=df_pizza.index.astype(str).astype(int)
df_pizza=df_pizza.sort_values(by=['violation code'])

#concatenate the dataframes together.
vio_cuisine_type1=df_american.merge(df_cn,left_on='violation code', right_on='vi
olation code', how='outer')
vio_cuisine_type1=vio_cuisine_type1.merge(df_cafe,left_on='violation code', righ
t_on='violation code', how='outer')
vio_cuisine_type1=vio_cuisine_type1.merge(df_pizza,left_on='violation code', rig
ht_on='violation code', how='outer')
vio_cuisine_type1=vio_cuisine_type1.drop(['violation type_x','violation type_y']
,axis=1)
vio_cuisine_type1.fillna(0, inplace=True)
vio_cuisine_type1=vio_cuisine_type1.set_index('violation code')
vio_cuisine_type1
```

Out[26]:

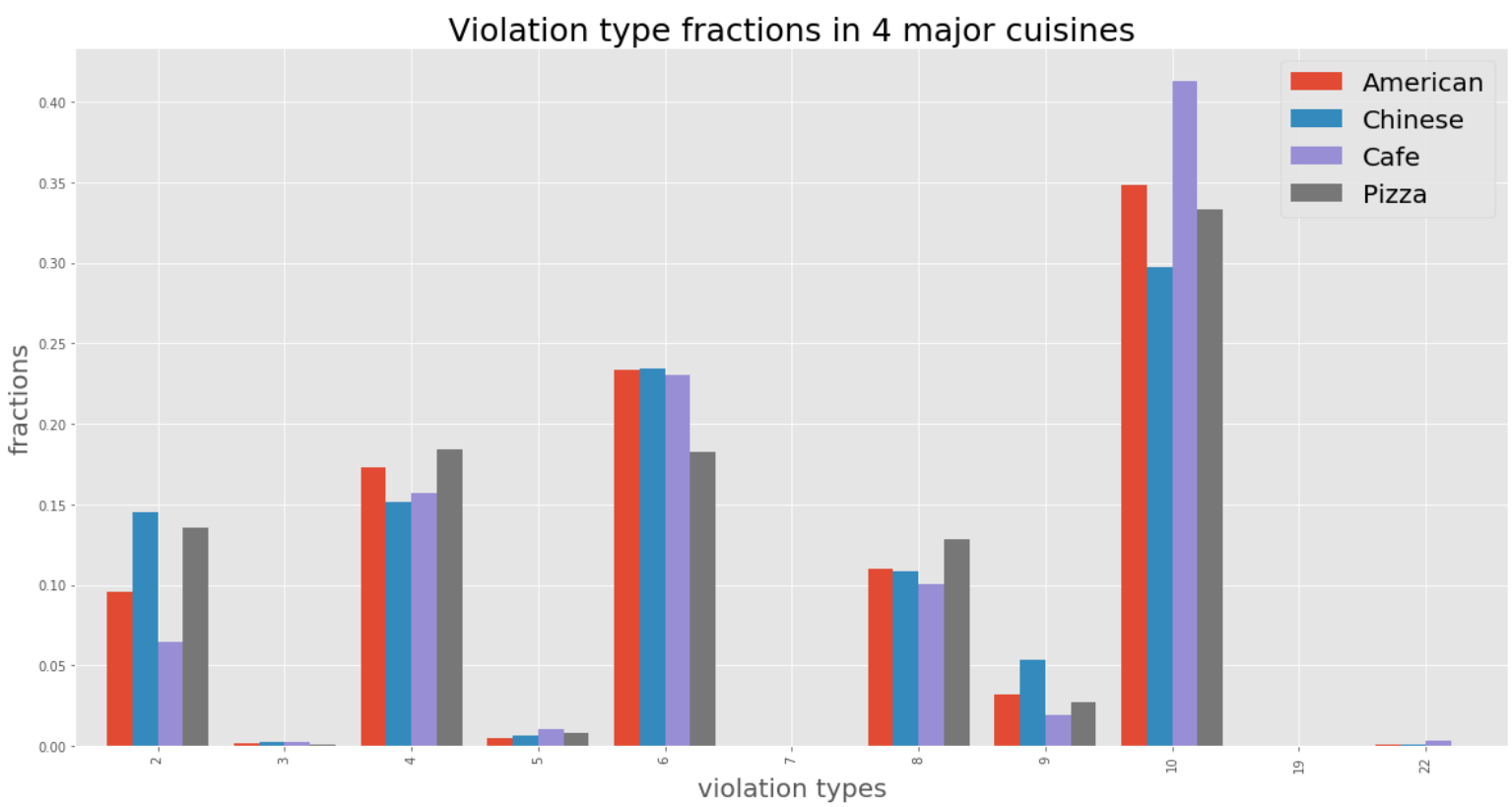
violation code	American	Chinese	Cafe	Pizza
2	0.095865	0.144865	0.064281	0.135587
3	0.001438	0.002308	0.002217	0.000942
4	0.173055	0.151684	0.156956	0.184003
5	0.005058	0.006189	0.010450	0.007657
6	0.233705	0.234449	0.230526	0.182825
7	0.000024	0.000000	0.000000	0.000000
8	0.110080	0.108675	0.100486	0.128401
9	0.031619	0.053289	0.019105	0.027329
10	0.348244	0.297336	0.412603	0.333255
19	0.000024	0.000000	0.000000	0.000000
22	0.000887	0.001206	0.003378	0.000000

In [27]:

```
fig, ax = plt.subplots()
vio_cuisine_type1.plot(kind='bar', width=0.8,ax=ax,figsize=(20,10))
ax.set_title('Violation type fractions in 4 major cuisines', fontsize=25)
ax.set_xlabel('violation types',fontsize=20)
ax.set_ylabel('fractions',fontsize=20)
ax.legend(fontsize=20)
```

Out[27]:

```
<matplotlib.legend.Legend at 0x1a1eb27550>
```



3. Trends in Restaurant Scores

Food safety of the restaurants is changing over time, and so do the restaurant scores. In addition to studying the restaurant scores in a static way, we also want to employ a more dynamic perspetive: we are curious about the trends in food safety. Have the restaurants become safer over the years? If so, which borough has experienced the greatest improvment? Which type of food has seen the most significant change? These are the topics that we are going to discuss in the following coding experiments.

In order to do so, we focus our attention on the data collected in 2015 and 2017. By taking the difference between these two years, we can find out the changes among all the restaurants that existed in both years:

In [28]:

```
#find the 2015 inspection data
mask1 = (restaurant2['grade date'] < '2016-1-1')
restaurant2015=restaurant2.loc[mask1]
#drop the irrelavent columns to make the rows of the same restaurant exactly the
same.
res2015=restaurant2015.drop(['inspection date','action','violation type',
                             'violation description','critical flag','grade',
                             'grade date','record date','inspection type','scor
e'],axis=1)
#drop the duplicates.
res2015=res2015.drop_duplicates(subset=['camis'], keep='first')
#create a dataframe with only the average score of every restaurant.
res2015_agg = restaurant2015.groupby('camis').agg({'score':np.mean})
#merge and create a new dataframe for 2015
combo2015 = pd.merge(res2015,res2015_agg,on='camis',how='inner')
combo2015 = combo2015.rename(columns={'score':'score2015'})

#find the 2017 inspection data
mask2 = (restaurant2['grade date'] > '2017-1-1')&(restaurant2['grade date'] < '2
018-1-1')
restaurant2017=restaurant2.loc[mask2]
#drop the irrelavent columns to make the rows of the same restaurant exactly the
same.
res2017=restaurant2017.drop(['name','boro','building','street','zipcode','phone'
,'type','inspection date','action','violation type',
                             'violation description','critical flag','grade',
                             'grade date','record date','inspection type','scor
e'],axis=1)
#drop the duplicates.
res2017=res2017.drop_duplicates(subset=['camis'], keep='first')
#create a dataframe with only the average score of every restaurant.
res2017_agg = restaurant2017.groupby('camis').agg({'score':np.mean})
#merge and create a new dataframe for 2017
combo2017 = pd.merge(res2017,res2017_agg,on='camis',how='inner')
combo2017 = combo2017.rename(columns={'score':'score2017'})

#combine combo2015 and combo2017
combo1517 = pd.merge(combo2015,combo2017,on='camis',how='inner')

# Take the difference of the score in 2017 and score in 2015 for every single re
staurant
combo1517['diff'] = combo1517['score2017'] - combo1517['score2015']
combo1517
```

Out[28]:

	camis	name	boro	building	street	zipcode	ph
0	40999668	DUNKIN' DONUTS/BASKIN ROBBINS	MANHATTAN	269	8 AVENUE	10011.0	6463968

1	40388218	WALKER'S RESTAURANT	MANHATTAN	16	NORTH MOORE STREET	10013.0	2129410
2	50003556	EASTERN RESTAURANT	BROOKLYN	1877	ROCKAWAY PKWY	11236.0	7182090
3	40367005	DA VINCI PIZZA	BROOKLYN	6514	18 AVENUE	11204.0	7182325
4	50015640	LITTLE SKIPS OUTPOST	BROOKLYN	1158	MYRTLE AVE	11221.0	9292714
5	41676904	PINCH FOOD DESIGN	MANHATTAN	545	WEST 27 STREET	10001.0	2122447
6	41257059	PANADERIA LA MIXTECA POBLANA & DELI	STATEN ISLAND	104	VICTORY BOULEVARD	10301.0	7187201
7	41388993	WESTVILLE	MANHATTAN	210	WEST 10 STREET	10014.0	2127417
8	41519860	LALLISSE	MANHATTAN	161	LEXINGTON AVENUE	10016.0	2126861
9	41597839	KIDO SUSHI (QUEENS CENTER MALL)	QUEENS	9015	QUEENS BOULEVARD	11373.0	7182713
10	50004706	ATLANTIC SOCIAL	BROOKLYN	673	ATLANTIC AVE	11217.0	7186232
11	50018554	CIN CHINESE RESTAURANT	QUEENS	4181	BOWNE ST	11355.0	7188881
12	41436878	CASA LEVER	MANHATTAN	390	PARK AVENUE	10022.0	2128882
13	41647741	THE STANDARD EAST VILLAGE	MANHATTAN	25	COOPER SQUARE	10003.0	2124755
14	40401834	COLUMBUS GOURMET FOOD	MANHATTAN	261	COLUMBUS AVE STORE #2	10023.0	2127217
15	50032779	VAN LEEUWEN ARTISAN ICE CREAM	MANHATTAN	181	WAVERLY PL	10014.0	9174751
16	40369165	REIF'S TAVERN	MANHATTAN	302	EAST 92 STREET	10128.0	2124260
17	50001789	RESTAURANTE & PANADERIA GUATELINDA	QUEENS	170-18	JAMAICA AVE	11432.0	2124706
18	41597385	JUGOS Y TACOS	QUEENS	7804	WOODSIDE AVENUE	11373.0	7184262
19	40729118	SEVEN STARS BAKERY	BROOKLYN	280	86 STREET	11209.0	7188364
20	50005881	LILLY O'BRIENS	MANHATTAN	18	MURRAY ST	10007.0	6469302
21	40393488	BELLA NAPOLI	MANHATTAN	130	MADISON AVENUE	10016.0	2126834
22	50032820	HOLIDAY INN EXPRESS	BROOKLYN	625	UNION ST	11215.0	7187971
23	50033000	LAS MARGARITAS RESTAURANT	BROOKLYN	7206	3RD AVE	11209.0	7182383

24	50002019	HOME SWEET HARLEM	MANHATTAN	1528	AMSTERDAM AVENUE	10031.0	2129269
25	41161594	SMOOCH	BROOKLYN	266	CARLTON AVENUE	11205.0	7186244
26	50001385	EL RANCHO RESTAURANT	BROOKLYN	1157	FLATBUSH AVE	11226.0	3476279
27	41385657	HOWL AT THE MOON BAR & GRILL	BRONX	585	EAST 189 STREET	10458.0	7185617
28	41558287	CHANG HENG	BROOKLYN	54	WILLOUGHBY STREET	11201.0	7184889
29	40394555	LUIGI'S RESTAURANT	QUEENS	26521	UNION TURNPIKE	11040.0	7183477
...	...	...	...	...	...	...	
6760	50039333	AUNTIE ANNE'S PRETZELS	STATEN ISLAND	2655	RICHMOND AVE	10314.0	7186983
6761	40706408	BLACK AND WHITE	MANHATTAN	86	EAST 10 STREET	10003.0	2122530
6762	50005457	BIRCH COFFEE	MANHATTAN	62	MADISON AVE	10016.0	6096101
6763	50036399	LEE LEE BUBBLE TEA	BROOKLYN	5718	5TH AVE	11220.0	7187654
6764	41277210	NAIDRE'S	BROOKLYN	384	7 AVENUE	11215.0	7189657
6765	50008918	HAHM JI BACH	QUEENS	4011	149TH PL	11354.0	6466102
6766	41622592	GENNARO'S AT COUNTRY LANES	STATEN ISLAND	1600	HYLAN BOULEVARD	10305.0	7183510
6767	41399316	NEW KING DRAGON RESTAURANT	BRONX	1749	RANDALL AVE	10473.0	7183288
6768	50003337	SEASONED VEGAN	MANHATTAN	55	ST. NICHOLAS AVENUE	10026.0	2122220
6769	50016565	HARLEM PIZZA CO.	MANHATTAN	135	W 116TH ST	10026.0	2122229
6770	41253845	Feile	MANHATTAN	133	WEST 33 STREET	10001.0	2126951
6771	41674167	FRANKS PIZZA	BROOKLYN	8025	FLATLANDS AVENUE	11236.0	7182518
6772	41380930	INATESSO PIZZA BAR CASANO	MANHATTAN	28	WEST STREET	10006.0	2122678
6773	41360469	SKY GARDEN AND TONIC AT SPA CASTLE	QUEENS	1111	131 STREET	11356.0	7189393
6774	50019178	PARIS BAGUETTE	BROOKLYN	97	COURT ST	11201.0	7187970
6775	41663033	EL ENCANTO CENTRO-AMERICANO RESTAURANT	QUEENS	14912	JAMAICA AVENUE	11435.0	7182629
6776	50002534	DRUNKEN MUNKEY	MANHATTAN	338	E 92ND STREET	10128.0	6469984

6777	41090957	PETRARCA	MANHATTAN	34	WHITE STREET	10013.0	2126252
6778	40932233	CELESTE	MANHATTAN	502	AMSTERDAM AVENUE	10024.0	2128744
6779	41265263	BAKEWAY NYC	QUEENS	2521	BROADWAY	11106.0	7185452
6780	50039890	NEW CHINA DELIGHT	QUEENS	9902	37TH AVE	11368.0	7188988
6781	40402089	HENRY ST. ALE HOUSE	BROOKLYN	62	HENRY STREET	11201.0	7185224
6782	50033970	DOMINO'S	QUEENS	9614	METROPOLITAN AVE	11375.0	7187932
6783	41542069	INSOMNIA COOKIES	MANHATTAN	405	AMSTERDAM AVENUE	10024.0	2153831
6784	50012576	Fruitea	QUEENS	5705	MAIN ST	11355.0	7184450
6785	50005295	LIVING ROOMRESTAURANT AND LOUNGE	BROOKLYN	178	AVENUE U	11223.0	7189968
6786	50038278	TACO REY RESTAURANT #2	BRONX	2319	HUGHES AVE	10458.0	7189750
6787	41317143	BAGEL VILLA	BROOKLYN	7221	5 AVENUE	11209.0	7187451
6788	41540529	PERIDANCE CAPEZIO CENTER CAFE	MANHATTAN	126	EAST 13 STREET	10003.0	2125050
6789	50014059	RED LOBSTER	QUEENS	8801	QUEENS BLVD	11373.0	7187603

6790 rows × 11 columns

Within the Boroughs

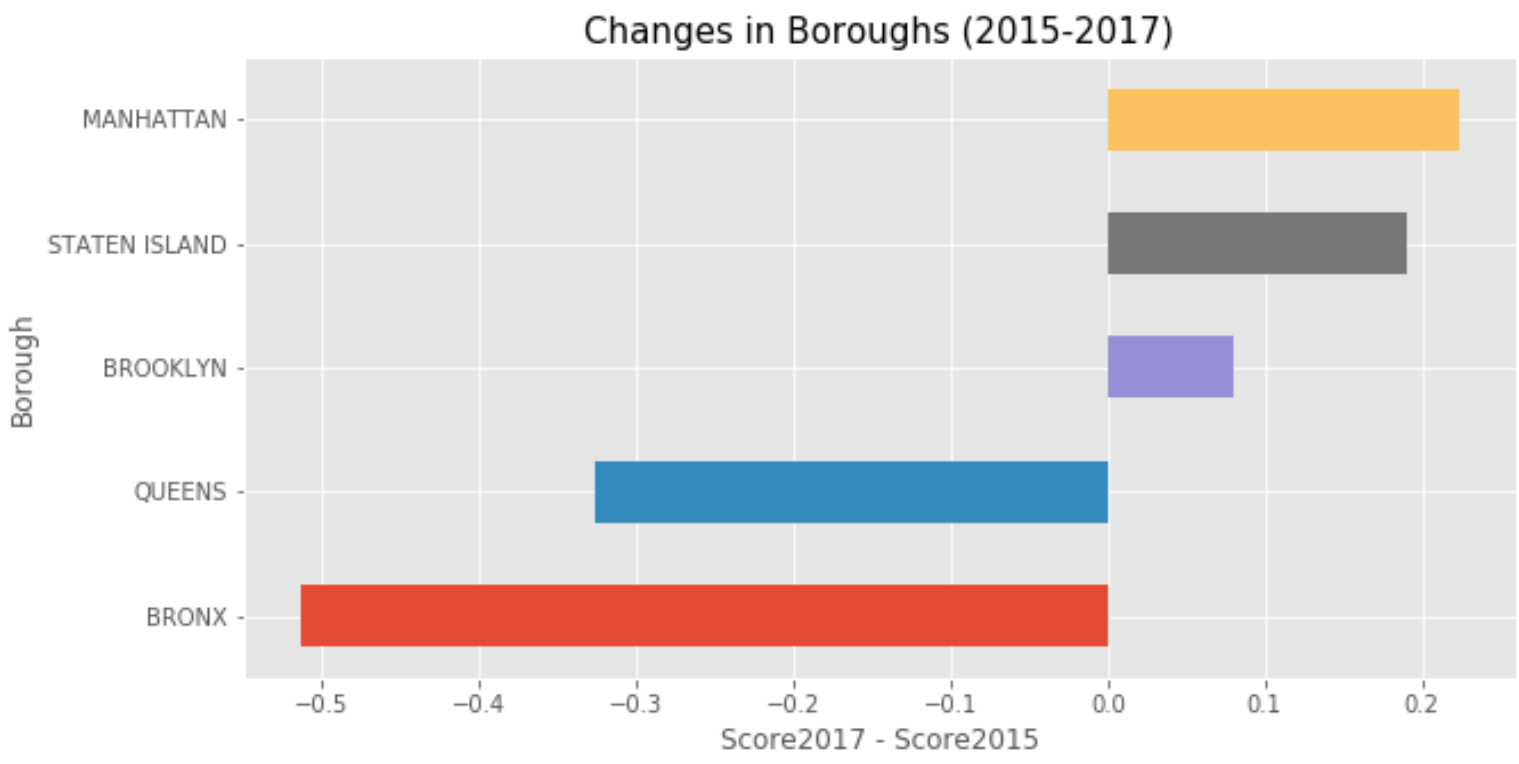
In [29]:

```
Diff_by_boro = combo1517.groupby('boro')['diff'].mean()
print(Diff_by_boro.sort_values())
#plot it so it will be more intuitive.
plt.style.use('ggplot')
fig, ax = plt.subplots()
Diff_by_boro.sort_values().plot(kind='barh', ax=ax,figsize=(10,5))
ax.set_title('Changes in Boroughs (2015-2017)', fontsize=15)
ax.set_xlabel('Score2017 - Score2015')
ax.set_ylabel('Borough')
```

```
boro
BRONX           -0.512409
QUEENS          -0.325835
BROOKLYN         0.080475
STATEN ISLAND    0.190476
MANHATTAN        0.222974
Name: diff, dtype: float64
```

Out[29]:

```
Text(0,0.5,'Borough')
```



Take a look!

Just a reminder: as restaurant scores are total points of food safety violations, the lower the score, the better. Therefore, **a nagative change in scores is associated with food safety improvement, while a positive change indicates that the food safety in 2017 was not as good as its 2015 level.**

The bar chart tells us that over the two years, food safety improved in Bronx and Queens, while deteriorated in Manhattan, Staten Island and Brooklyn.

Trends within Zipcode Zones

In [30]:

```
combo1517['zipcode']=combo1517['zipcode'].astype(int)
combo1517['zipcode']=combo1517['zipcode'].astype(str)
create_map(combo1517,'zipcode','diff','Borough')
```

```
/anaconda3/lib/python3.7/site-packages/folium/folium.py:432: FutureW
arning: The choropleth  method has been deprecated. Instead use the
new Choropleth class, which has the same arguments. See the example
notebook 'GeoJSON_and_choropleth' for how to do this.
  FutureWarning
```

Which Cuisine Experienced the Greatest Progress over the Years?

Just as what we have done in the previous parts of the project, we are eager to see the changes of food safety among different kinds of cuisines. After all, the production procedure, key ingredients, as well as the equipments involved can vary across different food types, and these may contribute to different changes in food safety.

To start with, we create a dataframe with the differences of restaurant scores between 2015 and 2017, and sort the values for further analysis:

In [31]:

```
combo1517['type'].value_counts()
```

Out[31]:

```
American
```

1607
Chinese
620
CafÃ©/Coffee/Tea
357
Pizza
352
Italian
298
Latin (Cuban, Dominican, Puerto Rican, South & Central American)
268
Mexican
252
Bakery
229
Japanese
202
Donuts
189
Caribbean
189
Spanish
158
Pizza/Italian
156
Hamburgers
139
Delicatessen
113
Chicken
112
Sandwiches
108
French
90
Jewish/Kosher
89
Thai
79
Indian
79
Asian
76
Korean
73
Mediterranean
67
Irish
63
Bagels/Pretzels
58
Juice, Smoothies, Fruit Salads
54
Seafood

49

Ice Cream, Gelato, Yogurt, Ices
49
Sandwiches/Salads/Mixed Buffet
47

...
Soul Food
12
Polish
11
Armenian
11
Ethiopian
10
Bangladeshi
10
German
10
Tapas
6
Australian
6
Pakistani
6
Creole
5
Portuguese
5
English
5
Afghan
4
Hotdogs
4
Brazilian
3
Chinese/Cuban
3
Pancakes/Waffles
3
Hotdogs/Pretzels
3
Indonesian
2
Egyptian
2
Fruits/Vegetables
2
Moroccan
2
Cajun
2

```
Polynesian

1
Iranian
1
Southwestern
1
Czech
1
Californian
1
Scandinavian
1
Not Listed/Not Applicable
1
Name: type, Length: 79, dtype: int64
```

In [32]:

```
#add a column with the number "1" to each data line for counting.
combo1517['count'] = 1
#create a dataframe that consist of the score difference and number for each typ
e of restaurant.
type_agg = combo1517.groupby('type').agg({'diff':np.mean,'count':np.sum})
#sort out those types with more than 50 restaurants.(delete small samples)
type_agg = type_agg.loc[type_agg['count']>50,:]
#sort them by their average score, from small to big.
type_agg.sort_values(by=['diff'])
```

Out[32]:

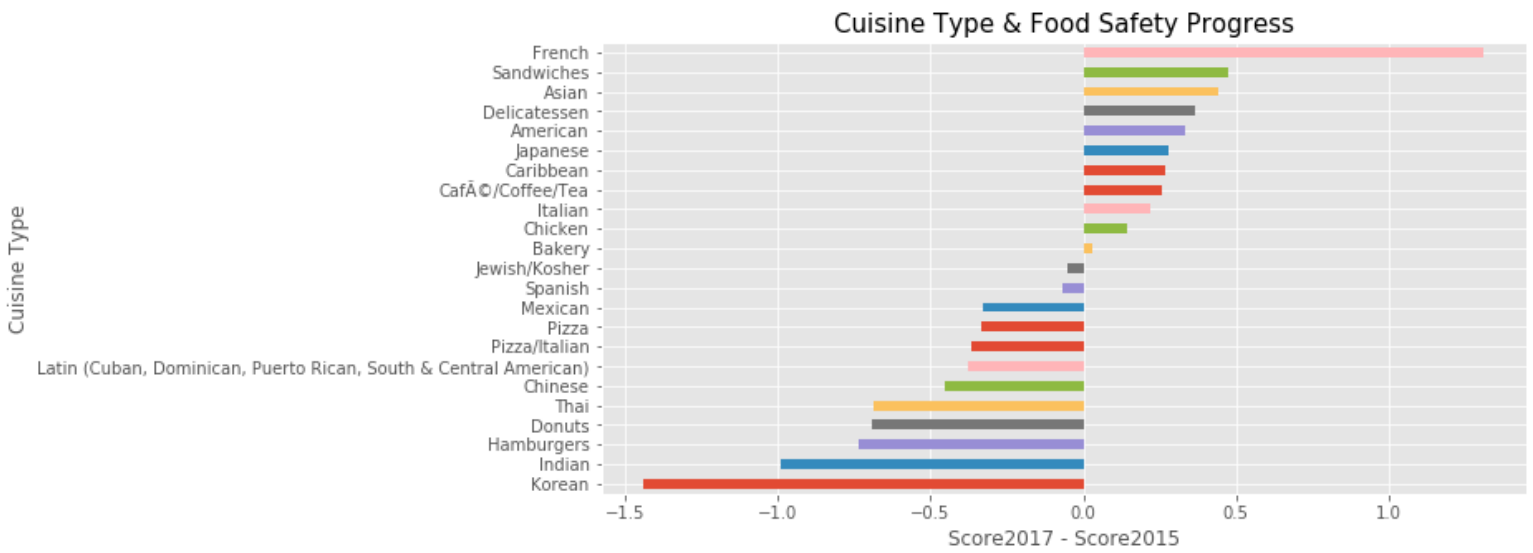
type	diff	count
Korean	-1.441923	73
Indian	-0.993279	79
Hamburgers	-0.734007	139
Donuts	-0.693462	189
Thai	-0.689688	79
Chinese	-0.455369	620
Latin (Cuban, Dominican, Puerto Rican, South & Central American)	-0.379562	268
Pizza/Italian	-0.367243	156
Pizza	-0.334125	352
Mexican	-0.328841	252
Bagels/Pretzels	-0.182184	58
Spanish	-0.071107	158
Jewish/Kosher	-0.054004	89
Mediterranean	-0.034488	67
Bakery	0.028079	229
Chicken	0.144196	112
Italian	0.221000	298
CafÃ©/Coffee/Tea	0.257036	357
Caribbean	0.268411	189
Japanese	0.278901	202
American	0.331896	1607
Juice, Smoothies, Fruit Salads	0.340278	54
Delicatessen	0.367517	113
Irish	0.434278	63
Asian	0.443212	76
Sandwiches	0.474383	108
French	1.311296	90

In [33]:

```
#selecting the cuisine types with more than 100 restaurants
Diff_by_type = type_agg.loc[type_agg['count']>70,:]
#plot it out
plt.style.use('ggplot')
fig, ax = plt.subplots()
Diff_by_type['diff'].sort_values().plot(kind='barh', ax=ax,figsize=(10,5))
ax.set_title('Cuisine Type & Food Safety Progress', fontsize=15)
ax.set_xlabel('Score2017 - Score2015')
ax.set_ylabel('Cuisine Type')
```

Out[33]:

```
Text(0,0.5,'Cuisine Type')
```



Wow! Beautiful!
The differences are significant here. Around half of the main cuisine types experienced food safety improvements, while the other half suffered different levels of deterioration.
It is French and Korean that attracts most of our attention: there was a 1.3 points increase in the overall restaurant scores of French type, and a 1.4 points decrease in that of Korean. What happened to them?

We now extract French and Korean from our dataframe to take a closer look at them:

In [34]:

```
#create a subset of only "French" type restaurants.
french=combo2015[combo2015.type=='French']
print(french['score2015'].describe())
```

```
count    116.000000
mean       9.722024
std        5.285862
min        2.000000
25%        7.000000
50%        9.000000
75%       12.000000
max       38.000000
Name: score2015, dtype: float64
```

In [35]:

```
#create a subset of only "Korean" type restaurants.
korean=combo2015[combo2015.type=='Korean']
print(korean['score2015'].describe())
```

```
count    99.000000
mean     12.787142
std       7.561093
min       2.000000
25%       8.750000
50%      11.000000
75%      13.000000
max      48.000000
Name: score2015, dtype: float64
```

OK, it seems to us that the restaurants are following an "approaching the average" trend. French restaurants, which started with a relatively lower score (9.72) in 2015, had its score increase to 11.03, while Korean restaurants, started with a relatively higher score (12.79) in 2015, had its score decrease to 11.35, making these two types of restaurants end up approximately the same level in 2017.

Trend for All Restaurants

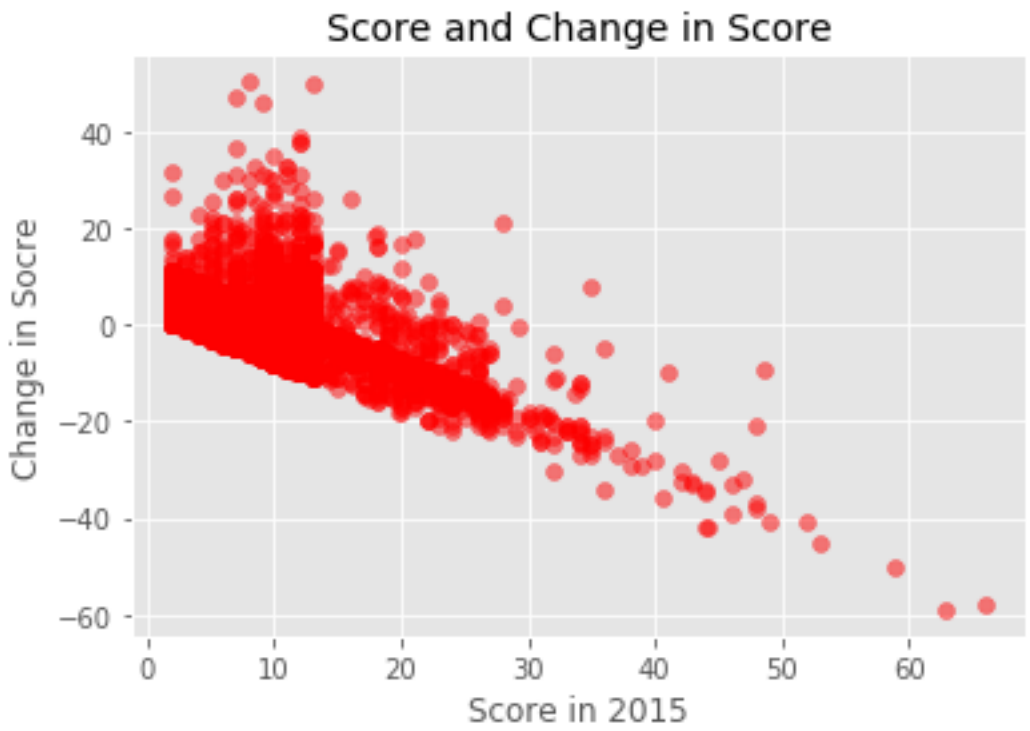
We want to check whether this trend applys to all the restaurants in our dataset, so we decide to visulize it in a scatter plot:

In [36]:

```
fig, ax = plt.subplots()
ax.scatter(combo1517['score2015'], combo1517['diff'], color='red',
           alpha=0.5)
ax.set_title('Score and Change in Score', fontsize=14)
ax.set_xlabel('Score in 2015')
ax.set_ylabel('Change in Socre')
```

Out[36]:

```
Text(0,0.5,'Change in Socre')
```



There seems to be a trend! Let's do a regression:

In [37]:

```
print(smf.ols('diff ~ score2015',data=combo1517).fit().summary())
```

```
                            OLS Regression Results
==============================================================================
==========
Dep. Variable:                   diff   R-squared:
0.447
Model:                            OLS   Adj. R-squared:
0.447
Method:                 Least Squares   F-statistic:
5491.
Date:                Fri, 21 Dec 2018   Prob (F-statistic):
0.00
Time:                        02:48:36   Log-Likelihood:
-20743.
No. Observations:                6790   AIC:
4.149e+04
Df Residuals:                    6788   BIC:
4.150e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
==========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
----------
Intercept      8.9223      0.136     65.722      0.000       8.656
9.188
score2015     -0.8444      0.011    -74.104      0.000      -0.867
-0.822
==============================================================================
==========
Omnibus:                     3613.012   Durbin-Watson:
2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
45624.717
Skew:                           2.267   Prob(JB):
0.00
Kurtosis:                      14.862   Cond. No.
26.1
==============================================================================
==========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
```

We can conclude from the regression that the change in score and the score in 2015 followed a negative relationship: every one point increase in the 2015 restaurant score was associated with a 0.8444 decrease in the score changes.

4. Conclusion

We all need to eat and food is important. After our research, we can conclude that food safely is related to both the cuisine type and the affluence of the area. So it might seem wise to eat donuts, ice cream or sandwiches in a rich area. But as the regression has shown, the restaurants' food quality in NYC tend to converge to the average. And the differences are not significant. So maybe, we don't need to care too much about food safety when choosing the restaurant. Just try new things out and enjoy your life!