Data Bootcamp Group Project

Yelp Business Data: Restaurants in the United States

Yue Cui, Ewan (Yuanhao) Zhao, Heropeia Hu

Introduction

This project studies **restaurants in the United States** that are listed on ***Yelp.com*** with related datasets collected from the Yelp Dataset (https://www.kaggle.com/yelp-dataset/yelp-dataset/version/4) webpage on **Kaggle**. Given that the main focus of this project is business instead of users, the Yelp_business (https://www.kaggle.com/yelp-dataset/yelp-dataset/version/4#yelp_business.csv) and the Yelp_business_attributes (https://www.kaggle.com/yelp-dataset/yelp-dataset/version/4#yelp_business_attributes.csv) are the two most frequently used datasets in this case.

In this project, the first **research purpose** is to **analyze data relathionships** among restaurants categories, attributes and stars rating. After completion, we are able to tell the insights of which **cuisine categories** and **business attributes** contribute to **higher stars rating**. Secondly, based on the analysis, we create **search basis** for **western coast** restaurants **by names, cities, and parking facilities**. The customized recommendations could benefit both users and business owners to better locate friendly services and restaurants in local areas.

The project strucutres as follows:

- **I. Data Frame Presentation**
  - 1.1 *Business* Dataset Cleaning
  - 1.2 *Attribute* Dataset Cleaning
  - 1.3 *Business* and *Attribute* Data Frames Merging
  - 1.4 Data Visualization
- **II. Data Relationships**
  - 2.1 Relationship Analysis of Star Ratings
    - Relationship between Review Counts and Ratings
    - Relationship between Cuisines and Ratings
    - Relationship between States and Ratings
  - 2.2 Multiple Linear Regression
    - Attributes and Ratings
    - Cuisines and Ratings
  - 2.3 Linear Regression Model Using Machine Learning
- **III. West Coast Restaurant Recommendation Tool**
  - Restaurants located on the ***West Coast***: CA, NV, AZ, OR, WA
    - Restaurant recommendation by NAME
    - Restaurant recommendation by CITY
    - Restaurant recommendation by PARKING Facilities

Project

In [1]:
```
# importing necessery libraries for future analysis of the dat
aset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import seaborn as sns
import statsmodels.formula.api as smf
import matplotlib.dates as mdates
import statsmodels.api as sm
from statsmodels.api import add_constant
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerato
r
import re
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
```

In [2]:
```
# loading datasets
business = pd.read_csv('/Users/heropeia/Desktop/Final Project/
Data/yelp_business.csv')
attribute = pd.read_csv('/Users/heropeia/Desktop/Final Projec
t/Data/yelp_business_attributes.csv')
busi_attr = pd.read_csv('/Users/heropeia/Desktop/Final Projec
t/Data/yelp_business_attributes.csv')
```

I. Data Frame Presentation

In this section, two original data frames will be **cleaned**, **reshaped** and **merged** to develop into a **new data frame** showing both **basic information** and **detailed attributes** of restaurants in the United States. This data **cleaning process** would make analysis tasks in the next two sections much **easier** to complete.

1.1 *Business* Dataset Cleaning

A. Brief Overview of the Yelp_business Dataset

In [3]:
```
business
```

Out[3]:

	business_id	name	neighborhood	address	city	state
0	FYWN1wneV18bWNgQjJ2GNg	"Dental by Design"	NaN	"4855 E Warner Rd, Ste B9"	Ahwatukee	AZ
1	He-G7vWjzVUysIKrfNbPUQ	"Stephen Szabo Salon"	NaN	"3101 Washington Rd"	McMurray	PA
2	KQPW8lFf1y5BT2MxiSZ3QA	"Western Motor Vehicle"	NaN	"6025 N 27th Ave, Ste 1"	Phoenix	AZ
3	8DShNS-LuFqpEWIpoHxijA	"Sports Authority"	NaN	"5000 Arizona Mills Cr, Ste 435"	Tempe	AZ
4	PfOCPjBrlQAnz__NXj9h_w	"Brick House Tavern + Tap"	NaN	"581 Howe Ave"	Cuyahoga Falls	OH
...	...	...	...	...	...	...
174562	ALV5R8NkZ1KGOZeuZl3u0A	"Whitby Toyota"	NaN	"1025 Dundas Street W"	Whitby	ON
174563	gRGalHVu6BcaUDIAGVW_xQ	"Village Auto Body"	NaN	"3957 Brecksville Rd"	Richfield	OH
174564	XXvZBIHoJBU5d6-a-oyMWQ	"AAM"	NaN	"1600 W Broadway Rd, Ste 200"	Tempe	AZ
174565	lNpPGgM96nPIYM1shxciHg	"Bronze Beauty Spray Tanning"	NaN	"300 Camp Horne Rd, Ste 250"	Pittsburgh	PA
174566	viKaP26BcHU6cLx8sf4gKg	"Phoenix Pharmacy"	NaN	"1701 East Thomas Rd, Ste 105"	Phoenix	AZ

174567 rows × 13 columns

Columns

In [4]:
```
# check type of every column in the dataset
business.dtypes
```

Out[4]:
```
business_id      object
name             object
neighborhood     object
address          object
city             object
state            object
postal_code      object
latitude        float64
longitude       float64
stars           float64
review_count      int64
is_open           int64
categories       object
dtype: object
```

In [5]:
```
business.columns
```

Out[5]:
```
Index(['business_id', 'name', 'neighborhood', 'address', 'city', 'state',
       'postal_code', 'latitude', 'longitude', 'stars', 'review_count',
       'is_open', 'categories'],
      dtype='object')
```

Rows

In [6]:
```
# check the amount of rows in the given dataset to understand the size we are w
orking with
print('There are '+str(len(business))+' rows in this dataset.')
```

```
There are 174567 rows in this dataset.
```

B. Examine Missing Data

In [7]:
```
# find columns that have null values
# use'sum' function to show how many nulls are found in each column in dataset
obj = business.isnull().sum()
for key,value in obj.iteritems():
    print(key,":",value)
```

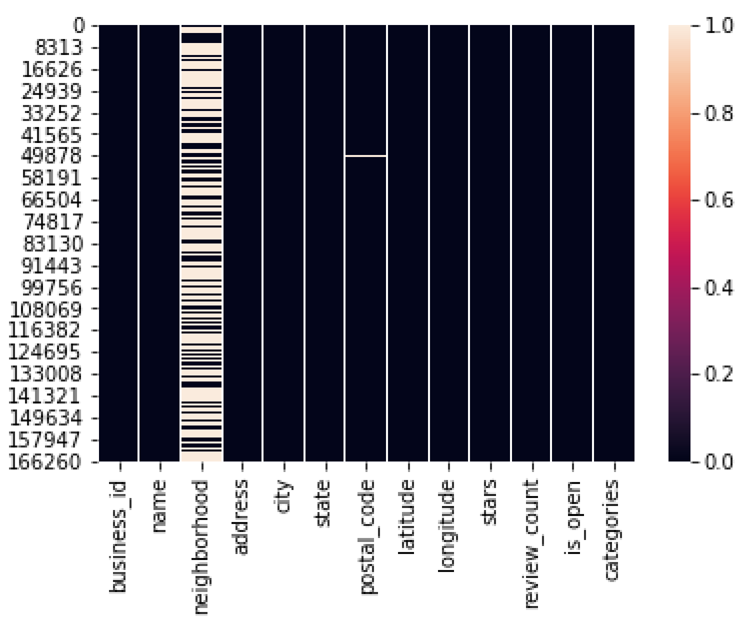
```
business_id : 0
name : 0
neighborhood : 106552
address : 0
city : 1
state : 1
postal_code : 623
latitude : 1
longitude : 1
stars : 0
review_count : 0
is_open : 0
categories : 0
```

Missing Data Heatmap

Figure 1. The **heat map** above demonstrates the proportion of **null values** in each column. More specifically, the lighter the strip is, the larger the amount of null values is within the column.

In [8]:
```
sns.heatmap(business.isnull())
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c182026d0>
```



In [9]:
```
# drop neighborhood column which has no actual meaning and has too many null va
lues
business.drop(['neighborhood'], axis=1, inplace=True)
business.drop(['is_open'],axis=1,inplace=True)
# remove quotation marks in name and address column
business.name=business.name.str.replace('"','')
business.address=business.address.str.replace('"','')
```

Summary of the Business Dataset after Preprocessing

In [10]:
```
print('Rows     :',business.shape[0])
print('Columns  :',business.shape[1])
print('\nFeatures :\n     :',business.columns.tolist())
print('\nMissing values    :',business.isnull().values.sum())
print('\nUnique values :  \n',business.nunique())
```

```
Rows     : 174567
Columns  : 11

Features :
     : ['business_id', 'name', 'address', 'city', 'state', 'postal_code', 'l
atitude', 'longitude', 'stars', 'review_count', 'categories']

Missing values    : 627

Unique values :
 business_id     174567
name            132618
address         138564
city              1093
state               67
postal_code      16004
latitude        138432
longitude       138844
stars                9
review_count      1061
categories       76419
dtype: int64
```

C. Filter and Categorize the U.S. Restaurants

Filter the U.S. Restaurants

In [11]:
```python
US_states = ["AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DC", "DE", "FL", "GA",
           "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD",
           "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ",
           "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC",
           "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY"]
usa=business.loc[business['state'].isin(US_states)]
us_restaurants=usa[usa['categories'].str.contains('Restaurants')]
us_restaurants
```

Out[11]:

	business_id	name	address	city	state	postal_code	]
4	PfOCPjBrlQAnz__NXj9h_w	Brick House Tavern + Tap	581 Howe Ave	Cuyahoga Falls	OH	44221	4
10	XOSRcvtaKc_Q5H1SAzN20A	East Coast Coffee	737 West Pike St	Houston	PA	15342	4(
14	fNMVV_ZX7CJSDWQGdOM8Nw	Showmars Government Center	600 E 4th St	Charlotte	NC	28202	3:
28	DjoS-Oe4ytRJzMGUPgYUkw	Panera Bread	38295 Chestnut Ridge Rd	Elyria	OH	44035	41
29	gAy4LYpsScrj8POnCW6btQ	Toast Cafe	2429 Hwy 160 W	Fort Mill	SC	29708	35
...	...	...	...	...	...	...	
174520	Gr-20Bg4XyduSKbvnE-i9g	Salt & Lime Modern Mexican Grill	9397 E Shea Blvd, Ste 115	Scottsdale	AZ	85260	3
174522	7wQXbUU2Mwvivg9wmBlrkg	Brown Bag Delis	1001 Liberty Ave	Pittsburgh	PA	15222	40
174523	nkDSE-yhvLX4ij5fSzvb5Q	Tonic Bar & Grill	971 Liberty Ave	Pittsburgh	PA	15222	40
174532	Ls_nR1MEcsOw5KuTlhodfQ	Cole's Public House	209 S Main St	Amherst	OH	44001	4:
174558	UdEmYOnk2iJDY9lpEPAlJQ	Floridino's Pizza & Pasta	590 N Alma School Rd, Ste 35	Chandler	AZ	85224	3.

32472 rows × 11 columns

In [12]:
```
us_restaurants.is_copy=False
us_restaurants['category']=pd.Series()
us_restaurants.loc[us_restaurants.categories.str.contains('American'),'category'] = 'American'
us_restaurants.loc[us_restaurants.categories.str.contains('Mexican'), 'category'] = 'Mexican'
us_restaurants.loc[us_restaurants.categories.str.contains('Italian'), 'category'] = 'Italian'
us_restaurants.loc[us_restaurants.categories.str.contains('Japanese'), 'category'] = 'Japanese'
us_restaurants.loc[us_restaurants.categories.str.contains('Chinese'), 'category'] = 'Chinese'
us_restaurants.loc[us_restaurants.categories.str.contains('Thai'), 'category'] = 'Thai'
us_restaurants.loc[us_restaurants.categories.str.contains('Mediterranean'), 'category'] = 'Mediterranean'
us_restaurants.loc[us_restaurants.categories.str.contains('French'), 'category'] = 'French'
us_restaurants.loc[us_restaurants.categories.str.contains('Vietnamese'), 'category'] = 'Vietnamese'
us_restaurants.loc[us_restaurants.categories.str.contains('Greek'),'category'] = 'Greek'
us_restaurants.loc[us_restaurants.categories.str.contains('Indian'),'category'] = 'Indian'
us_restaurants.loc[us_restaurants.categories.str.contains('Korean'),'category'] = 'Korean'
us_restaurants.loc[us_restaurants.categories.str.contains('Hawaiian'),'category'] = 'Hawaiian'
us_restaurants.loc[us_restaurants.categories.str.contains('African'),'category'] = 'African'
us_restaurants.loc[us_restaurants.categories.str.contains('Spanish'),'category'] = 'Spanish'
us_restaurants.loc[us_restaurants.categories.str.contains('Middle_eastern'),'category'] = 'Middle_eastern'
us_restaurants.category[:20]
```

```
/opt/anaconda3/lib/python3.7/site-packages/pandas/core/generic.py:5191: Futu
reWarning: Attribute 'is_copy' is deprecated and will be removed in a future
version.
  object.__getattribute__(self, name)
/opt/anaconda3/lib/python3.7/site-packages/pandas/core/generic.py:5192: Futu
reWarning: Attribute 'is_copy' is deprecated and will be removed in a future
version.
  return object.__setattr__(self, name, value)
```

Out[12]:

```
4      American
10          NaN
14     American
28          NaN
29     American
40     Japanese
44      Italian
45          NaN
46          NaN
52     American
53          NaN
54     American
64          NaN
72          NaN
75          NaN
76          NaN
80          NaN
81     American
88          NaN
91      Italian
Name: category, dtype: object
```

In [13]:

```
# drop null values in category, delete original column categories and reset the
index
us_restaurants = us_restaurants.dropna(axis=0, subset=['category'])
del us_restaurants['categories']
us_restaurants = us_restaurants.reset_index(drop=True)
```

D. Data Frame Presentation

In [14]:
```
us_restaurants
```

Out[14]:

	business_id	name	address	city	state	postal_code	la
0	PfOCPjBrlQAnz__NXj9h_w	Brick House Tavern + Tap	581 Howe Ave	Cuyahoga Falls	OH	44221	41
1	fNMVV_ZX7CJSDWQGdOM8Nw	Showmars Government Center	600 E 4th St	Charlotte	NC	28202	35
2	gAy4LYpsScrj8POnCW6btQ	Toast Cafe	2429 Hwy 160 W	Fort Mill	SC	29708	35.
3	tRVx2c89coruPRwYhGTcTw	Yuzu	13603 Madison Ave	Lakewood	OH	44107	41.
4	BnuzcebyB1AfxH0kjNWqSg	Carrabba's Italian Grill	245 Lancaster Ave	Frazer	PA	19355	40.
...	...	...	...	...	...	...	
19151	5zva2MTtB5IX6TaoVLL-NA	Zorbas Grill	440 W Warner Rd	Tempe	AZ	85284	33.
19152	vpwyL6NHm-pTWJ5IeJb9yw	Cocina Mendoza	300 Mt Lebanon Blvd	Pittsburgh	PA	15234	40
19153	Gr-20Bg4XyduSKbvnE-i9g	Salt & Lime Modern Mexican Grill	9397 E Shea Blvd, Ste 115	Scottsdale	AZ	85260	33
19154	nkDSE-yhvLX4ij5fSzvb5Q	Tonic Bar & Grill	971 Liberty Ave	Pittsburgh	PA	15222	40.
19155	UdEmYOnk2iJDY9lpEPAlJQ	Floridino's Pizza & Pasta	590 N Alma School Rd, Ste 35	Chandler	AZ	85224	33

19156 rows × 11 columns

1.2 *Attribute* Dataset Cleaning

A. Brief Overview of the Yelp_business_attribute Dataset

In [15]:
```
# replace all Na as NaN
attribute = attribute.replace('Na', np.nan)
attribute
```

Out[15]:

	business_id	AcceptsInsurance	ByAppointmentOnly	BusinessAccepts
0	FYWN1wneV18bWNgQjJ2GNg	NaN	NaN	
1	He-G7vWjzVUysIKrfNbPUQ	NaN	NaN	
2	8DShNS-LuFqpEWIp0HxijA	NaN	NaN	
3	PfOCPjBrlQAnz__NXj9h_w	NaN	NaN	
4	o9eMRCWt5PkpLDE0gOPtcQ	NaN	NaN	
...	...	...	...	
152036	kLFm_kehXNZkUc1oa2-Eaw	NaN	NaN	
152037	gRGalHVu6BcaUDIAGVW_xQ	NaN	NaN	
152038	XXvZBIHoJBU5d6-a-oyMWQ	NaN	NaN	
152039	lNpPGgM96nPIYM1shxciHg	NaN	NaN	
152040	viKaP26BcHU6cLx8sf4gKg	NaN	NaN	

152041 rows × 82 columns

Columns

In [16]:
```
attribute.dtypes
```

Out[16]:
```
business_id                          object
AcceptsInsurance                    float64
ByAppointmentOnly                    object
BusinessAcceptsCreditCards           object
BusinessParking_garage               object
BusinessParking_street               object
BusinessParking_validated            object
BusinessParking_lot                  object
BusinessParking_valet                object
HairSpecializesIn_coloring           object
HairSpecializesIn_africanamerican    object
HairSpecializesIn_curly              object
HairSpecializesIn_perms              object
HairSpecializesIn_kids               object
HairSpecializesIn_extensions         object
HairSpecializesIn_asian              object
HairSpecializesIn_straightperms      object
RestaurantsPriceRange2               object
GoodForKids                          object
WheelchairAccessible                 object
BikeParking                          object
Alcohol                              object
HasTV                                object
NoiseLevel                           object
RestaurantsAttire                    object
Music_dj                             object
Music_background_music               object
Music_no_music                       object
Music_karaoke                        object
Music_live                           object
Music_video                          object
Music_jukebox                        object
Ambience_romantic                    object
Ambience_intimate                    object
Ambience_classy                      object
Ambience_hipster                     object
Ambience_divey                       object
Ambience_touristy                    object
Ambience_trendy                      object
Ambience_upscale                     object
Ambience_casual                      object
RestaurantsGoodForGroups             object
Caters                               object
WiFi                                 object
RestaurantsReservations              object
RestaurantsTakeOut                   object
HappyHour                            object
GoodForDancing                       object
RestaurantsTableService              object
OutdoorSeating                       object
RestaurantsDelivery                  object
```

```
BestNights_monday                    object
BestNights_tuesday                   object
BestNights_friday                    object
BestNights_wednesday                 object
BestNights_thursday                  object
BestNights_sunday                    object
BestNights_saturday                  object
GoodForMeal_dessert                  object
GoodForMeal_latenight                object
GoodForMeal_lunch                    object
GoodForMeal_dinner                   object
GoodForMeal_breakfast                object
GoodForMeal_brunch                   object
CoatCheck                            object
Smoking                              object
DriveThru                            object
DogsAllowed                          object
BusinessAcceptsBitcoin               object
Open24Hours                          object
BYOBCorkage                          object
BYOB                                 object
Corkage                             float64
DietaryRestrictions_dairy-free      float64
DietaryRestrictions_gluten-free      object
DietaryRestrictions_vegan            object
DietaryRestrictions_kosher           object
DietaryRestrictions_halal            object
DietaryRestrictions_soy-free         object
DietaryRestrictions_vegetarian       object
AgesAllowed                          object
RestaurantsCounterService            object
dtype: object
```

Rows

In [17]:

```python
# check the amount of rows in the given dataset to understand the size we are w
orking with
print('There are '+str(len(attribute))+' rows in this dataset.')
```

```
There are 152041 rows in this dataset.
```

B. Examine Missing Data

In [18]:
```
# check null values for each column
obj = attribute.isnull().sum()
for key,value in obj.iteritems():
    print(key,":",value)
```

```
business_id : 0
AcceptsInsurance : 152041
ByAppointmentOnly : 151946
BusinessAcceptsCreditCards : 128460
BusinessParking_garage : 131649
BusinessParking_street : 112687
BusinessParking_validated : 112687
BusinessParking_lot : 113734
BusinessParking_valet : 112687
HairSpecializesIn_coloring : 112687
HairSpecializesIn_africanamerican : 152040
HairSpecializesIn_curly : 152040
HairSpecializesIn_perms : 152040
HairSpecializesIn_kids : 152040
HairSpecializesIn_extensions : 152040
HairSpecializesIn_asian : 152040
HairSpecializesIn_straightperms : 152040
RestaurantsPriceRange2 : 152040
GoodForKids : 146737
WheelchairAccessible : 131067
BikeParking : 112759
Alcohol : 141629
HasTV : 149207
NoiseLevel : 151824
RestaurantsAttire : 151955
Music_dj : 151958
Music_background_music : 152021
Music_no_music : 152021
Music_karaoke : 152021
Music_live : 152021
Music_video : 152021
Music_jukebox : 152021
Ambience_romantic : 152021
Ambience_intimate : 151979
Ambience_classy : 151979
Ambience_hipster : 151979
Ambience_divey : 151979
Ambience_touristy : 152008
Ambience_trendy : 151979
Ambience_upscale : 151979
Ambience_casual : 151979
RestaurantsGoodForGroups : 151979
Caters : 149872
WiFi : 151944
RestaurantsReservations : 149688
RestaurantsTakeOut : 151745
HappyHour : 145859
GoodForDancing : 151868
RestaurantsTableService : 149757
OutdoorSeating : 150305
RestaurantsDelivery : 151293
```

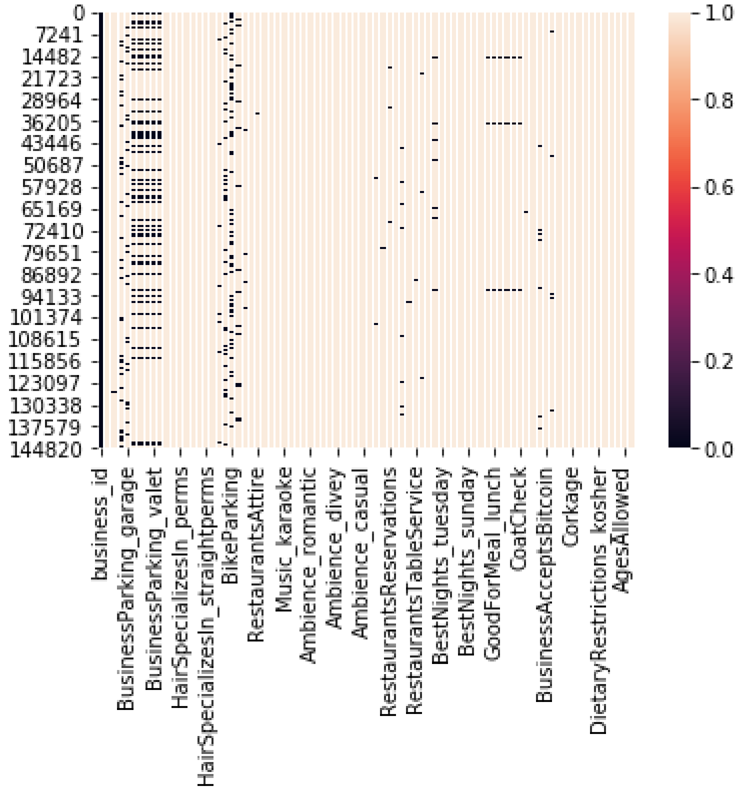
```
BestNights_monday : 148948
BestNights_tuesday : 152007
BestNights_friday : 152007
BestNights_wednesday : 152007
BestNights_thursday : 152007
BestNights_sunday : 152007
BestNights_saturday : 152007
GoodForMeal_dessert : 152007
GoodForMeal_latenight : 150227
GoodForMeal_lunch : 150227
GoodForMeal_dinner : 150227
GoodForMeal_breakfast : 150227
GoodForMeal_brunch : 150227
CoatCheck : 150227
Smoking : 151750
DriveThru : 151127
DogsAllowed : 146036
BusinessAcceptsBitcoin : 151637
Open24Hours : 150600
BYOBCorkage : 152037
BYOB : 151958
Corkage : 152041
DietaryRestrictions_dairy-free : 152041
DietaryRestrictions_gluten-free : 151931
DietaryRestrictions_vegan : 151931
DietaryRestrictions_kosher : 151931
DietaryRestrictions_halal : 151931
DietaryRestrictions_soy-free : 151931
DietaryRestrictions_vegetarian : 151931
AgesAllowed : 151931
RestaurantsCounterService : 152038
```

Missing Data Heatmap

Figure 2. The **heat map** above demonstrates the proportion of **null values** in each column. More specifically, the lighter the strip is, the larger the amount of null values is within the column.

In [19]:
```
sns.heatmap(attribute.isnull())
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c255da1d0>
```



Remove columns that are not important for restaurant if the column:

1. is hard to understand
2. has too many null values
3. irrelevant
4. categorical repetition

In [20]:
```
full_list = ['business_id', 'AcceptsInsurance', 'ByAppointmentOnly',
       'BusinessAcceptsCreditCards', 'BusinessParking_garage',
       'BusinessParking_street', 'BusinessParking_validated',
       'BusinessParking_lot', 'BusinessParking_valet',
       'HairSpecializesIn_coloring', 'HairSpecializesIn_africanamerican',
       'HairSpecializesIn_curly', 'HairSpecializesIn_perms',
       'HairSpecializesIn_kids', 'HairSpecializesIn_extensions',
       'HairSpecializesIn_asian', 'HairSpecializesIn_straightperms',
       'RestaurantsPriceRange2', 'GoodForKids', 'WheelchairAccessible',
       'BikeParking', 'Alcohol', 'HasTV', 'NoiseLevel', 'RestaurantsAttire',
       'Music_dj', 'Music_background_music', 'Music_no_music', 'Music_karaoke',
       'Music_live', 'Music_video', 'Music_jukebox', 'Ambience_romantic',
       'Ambience_intimate', 'Ambience_classy', 'Ambience_hipster',
       'Ambience_divey', 'Ambience_touristy', 'Ambience_trendy',
       'Ambience_upscale', 'Ambience_casual', 'RestaurantsGoodForGroups',
       'Caters', 'WiFi', 'RestaurantsReservations', 'RestaurantsTakeOut',
       'HappyHour', 'GoodForDancing', 'RestaurantsTableService',
       'OutdoorSeating', 'RestaurantsDelivery', 'BestNights_monday',
       'BestNights_tuesday', 'BestNights_friday', 'BestNights_wednesday',
       'BestNights_thursday', 'BestNights_sunday', 'BestNights_saturday',
       'GoodForMeal_dessert', 'GoodForMeal_latenight', 'GoodForMeal_lunch',
       'GoodForMeal_dinner', 'GoodForMeal_breakfast', 'GoodForMeal_brunch',
       'CoatCheck', 'Smoking', 'DriveThru', 'DogsAllowed',
       'BusinessAcceptsBitcoin', 'Open24Hours', 'BYOBCorkage', 'BYOB',
       'Corkage', 'DietaryRestrictions_dairy-free',
       'DietaryRestrictions_gluten-free', 'DietaryRestrictions_vegan',
       'DietaryRestrictions_kosher', 'DietaryRestrictions_halal',
       'DietaryRestrictions_soy-free', 'DietaryRestrictions_vegetarian',
       'AgesAllowed', 'RestaurantsCounterService']
list_to_keep = ['business_id', 'BusinessAcceptsCreditCards','BusinessParking_lo
t',
       'GoodForKids', 'WheelchairAccessible', 'Alcohol', 'RestaurantsAttire',
       'WiFi', 'RestaurantsReservations', 'RestaurantsTakeOut',
       'HappyHour', 'RestaurantsDelivery', 'Smoking', 'Open24Hours']
list_to_remove = [i for i in full_list if i not in list_to_keep]
print(str(list_to_remove))
```

```
['AcceptsInsurance', 'ByAppointmentOnly', 'BusinessParking_garage', 'Busines
sParking_street', 'BusinessParking_validated', 'BusinessParking_valet', 'Hai
rSpecializesIn_coloring', 'HairSpecializesIn_africanamerican', 'HairSpeciali
zesIn_curly', 'HairSpecializesIn_perms', 'HairSpecializesIn_kids', 'HairSpec
ializesIn_extensions', 'HairSpecializesIn_asian', 'HairSpecializesIn_straigh
tperms', 'RestaurantsPriceRange2', 'BikeParking', 'HasTV', 'NoiseLevel', 'Mu
sic_dj', 'Music_background_music', 'Music_no_music', 'Music_karaoke', 'Music
_live', 'Music_video', 'Music_jukebox', 'Ambience_romantic', 'Ambience_intim
ate', 'Ambience_classy', 'Ambience_hipster', 'Ambience_divey', 'Ambience_tou
risty', 'Ambience_trendy', 'Ambience_upscale', 'Ambience_casual', 'Restauran
tsGoodForGroups', 'Caters', 'GoodForDancing', 'RestaurantsTableService', 'Ou
tdoorSeating', 'BestNights_monday', 'BestNights_tuesday', 'BestNights_frida
y', 'BestNights_wednesday', 'BestNights_thursday', 'BestNights_sunday', 'Bes
tNights_saturday', 'GoodForMeal_dessert', 'GoodForMeal_latenight', 'GoodForM
eal_lunch', 'GoodForMeal_dinner', 'GoodForMeal_breakfast', 'GoodForMeal_brun
ch', 'CoatCheck', 'DriveThru', 'DogsAllowed', 'BusinessAcceptsBitcoin', 'BYO
BCorkage', 'BYOB', 'Corkage', 'DietaryRestrictions_dairy-free', 'DietaryRest
rictions_gluten-free', 'DietaryRestrictions_vegan', 'DietaryRestrictions_kos
her', 'DietaryRestrictions_halal', 'DietaryRestrictions_soy-free', 'DietaryR
estrictions_vegetarian', 'AgesAllowed', 'RestaurantsCounterService']
```

C. Data Frame Presentation

In [21]:

```
attribute.drop(['AcceptsInsurance', 'ByAppointmentOnly', 'BusinessParking_garage', 'BusinessParking_street', 'BusinessParking_validated', 'BusinessParking_valet', 'HairSpecializesIn_coloring', 'HairSpecializesIn_africanamerican', 'HairSpecializesIn_curly', 'HairSpecializesIn_perms', 'HairSpecializesIn_kids', 'HairSpecializesIn_extensions', 'HairSpecializesIn_asian', 'HairSpecializesIn_straightperms', 'RestaurantsPriceRange2', 'BikeParking', 'HasTV', 'NoiseLevel', 'Music_dj', 'Music_background_music', 'Music_no_music', 'Music_karaoke', 'Music_live', 'Music_video', 'Music_jukebox', 'Ambience_romantic', 'Ambience_intimate', 'Ambience_classy', 'Ambience_hipster', 'Ambience_divey', 'Ambience_touristy', 'Ambience_trendy', 'Ambience_upscale', 'Ambience_casual', 'RestaurantsGoodForGroups', 'Caters', 'GoodForDancing', 'RestaurantsTableService', 'OutdoorSeating', 'BestNights_monday', 'BestNights_tuesday', 'BestNights_friday', 'BestNights_wednesday', 'BestNights_thursday', 'BestNights_sunday', 'BestNights_saturday', 'GoodForMeal_dessert', 'GoodForMeal_latenight', 'GoodForMeal_lunch', 'GoodForMeal_dinner', 'GoodForMeal_breakfast', 'GoodForMeal_brunch', 'CoatCheck', 'DriveThru', 'DogsAllowed', 'BusinessAcceptsBitcoin', 'BYOBCorkage', 'BYOB', 'Corkage', 'DietaryRestrictions_dairy-free', 'DietaryRestrictions_gluten-free', 'DietaryRestrictions_vegan', 'DietaryRestrictions_kosher', 'DietaryRestrictions_halal', 'DietaryRestrictions_soy-free', 'DietaryRestrictions_vegetarian', 'AgesAllowed', 'RestaurantsCounterService'],axis = 1, inplace = True)
attribute
```

Out[21]:

	business_id	BusinessAcceptsCreditCards	BusinessParking_lot	Good
0	FYWN1wneV18bWNgQjJ2GNg	NaN	NaN	
1	He-G7vWjzVUysIKrfNbPUQ	NaN	NaN	
2	8DShNS-LuFqpEWIpoHxijA	NaN	NaN	
3	PfOCPjBrlQAnz__NXj9h_w	NaN	NaN	
4	o9eMRCWt5PkpLDE0gOPtcQ	NaN	False	
...	...	...	...	
152036	kLFm_kehXNZkUc1oa2-Eaw	NaN	False	
152037	gRGalHVu6BcaUDIAGVW_xQ	NaN	NaN	
152038	XXvZBIHoJBU5d6-a-oyMWQ	True	NaN	
152039	lNpPGgM96nPIYM1shxciHg	NaN	NaN	
152040	viKaP26BcHU6cLx8sf4gKg	NaN	NaN	

152041 rows × 14 columns

1.3 *Business* and *Attribute* Data Frames Merging

After merging the **Business** data frame with the **Attribute** data frame, several **attribute-related columns** have been **deleted** given that either that attribute is **categorized with varied levels** or there are far **too few "True" values** within the column. The **undesirable** columns are ***'GoodForKids'***, ***'Open24Hours'***, ***'RestaurantsReservations'***, ***'RestaurantsAttire'***, ***'BusinessAcceptsCreditCards'***.

Columns **retained** include ***'BusinessParking_lot'***, ***'WheelchairAccessible'***, ***'Alcohol'***, ***'RestaurantsTakeOut'***, ***'HappyHour'***, ***'RestaurantsDelivery'***, ***'Smoking'***. These attributes will be used as **independent variables** later in regressions.

Below is the combined data frame:

In [22]:
```python
us_restaurants_attribute = pd.merge(us_restaurants,attribute,left_on='business_id',right_on='business_id',how='inner')
us_restaurants_attribute = us_restaurants_attribute.fillna(0)
us_restaurants_attribute.replace('True', 1, inplace=True)
us_restaurants_attribute.replace('False', 0, inplace=True)
us_restaurants_attribute = us_restaurants_attribute[['business_id', 'name', 'address', 'city', 'state', 'postal_code',
                              'latitude', 'longitude', 'stars', 'review_count', 'category',
                              'BusinessParking_lot' ,'WheelchairAccessible' ,'Alcohol',
                              'RestaurantsTakeOut' ,'HappyHour', 'RestaurantsDelivery', 'Smoking']]
us_restaurants_attribute
```

Out[22]:

	business_id	name	address	city	state	postal_code	
0	PfOCPjBrlQAnz__NXj9h_w	Brick House Tavern + Tap	581 Howe Ave	Cuyahoga Falls	OH	44221	
1	fNMVV_ZX7CJSDWQGdOM8Nw	Showmars Government Center	600 E 4th St	Charlotte	NC	28202	
2	gAy4LYpsScrj8POnCW6btQ	Toast Cafe	2429 Hwy 160 W	Fort Mill	SC	29708	
3	tRVx2c89coruPRwYhGTcTw	Yuzu	13603 Madison Ave	Lakewood	OH	44107	
4	BnuzcebyB1AfxHokjNWqSg	Carrabba's Italian Grill	245 Lancaster Ave	Frazer	PA	19355	4
...	...	...	...	...	...	...	
18956	WUGbiFUhH6Iil_GCDoXU4g	Boston Market	829 Providence Rd	Charlotte	NC	28207	
18957	vpwyL6NHm-pTWJ5IeJb9yw	Cocina Mendoza	300 Mt Lebanon Blvd	Pittsburgh	PA	15234	
18958	Gr-2oBg4XyduSKbvnE-i9g	Salt & Lime Modern Mexican Grill	9397 E Shea Blvd, Ste 115	Scottsdale	AZ	85260	
18959	nkDSE-yhvLX4ij5fSzvb5Q	Tonic Bar & Grill	971 Liberty Ave	Pittsburgh	PA	15222	4
18960	UdEmYOnk2iJDY9lpEPAlJQ	Floridino's Pizza & Pasta	590 N Alma School Rd, Ste 35	Chandler	AZ	85224	

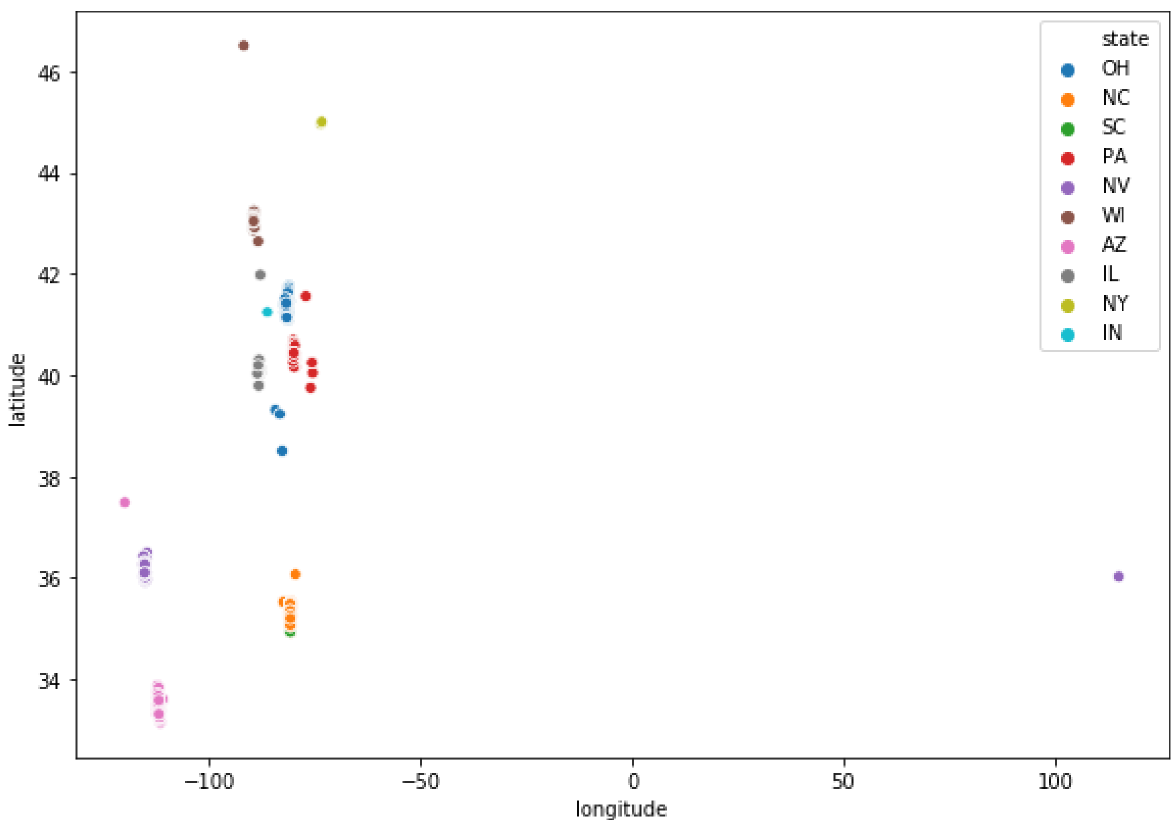
18961 rows × 18 columns

1.4 Data Visualization

A. Geographic Distribution of the U.S. Restaurants

Figure 3. This **geographic distribution** is plotted by using the columns - *longtitude* and *latitude* - from the data frame.

In [23]:
```
plt.figure(figsize=(10,7))
sns.scatterplot(us_restaurants.longitude,us_restaurants.latitude,hue=us_restaur
ants.state)
plt.ioff()
```



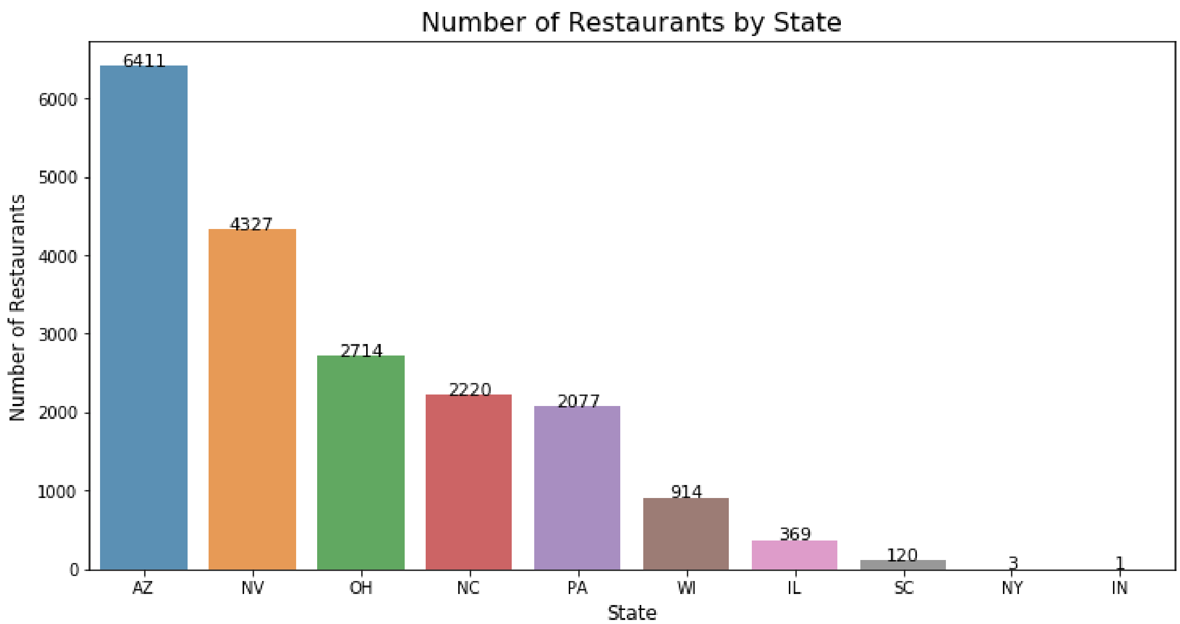
B. State-level and City-Level Restaurant Distributions

State-level

Figure 4. The **bar plot** presents the **number of restaurants in each state** in descending order. Within this *Yelp dataset*, the state with the **highet** amount of restaurants is **Arizona** of **6411** restaurants in total, whereas **Indiana** only have **one** restaurant.

In [24]:

```
state_count = us_restaurants['state'].value_counts().sort_values(ascending = False)
plt.figure(figsize = (12,6))
sns.barplot(state_count.index, state_count.values, alpha = 0.8, order = state_count.index )
plt.title('Number of Restaurants by State', fontsize = 16)
plt.ylabel('Number of Restaurants', fontsize = 12)
plt.xlabel('State', fontsize = 12)
for i, v in enumerate(state_count):
    plt.text(i, v, str(v), horizontalalignment ='center', fontsize=11)
plt.show()
```

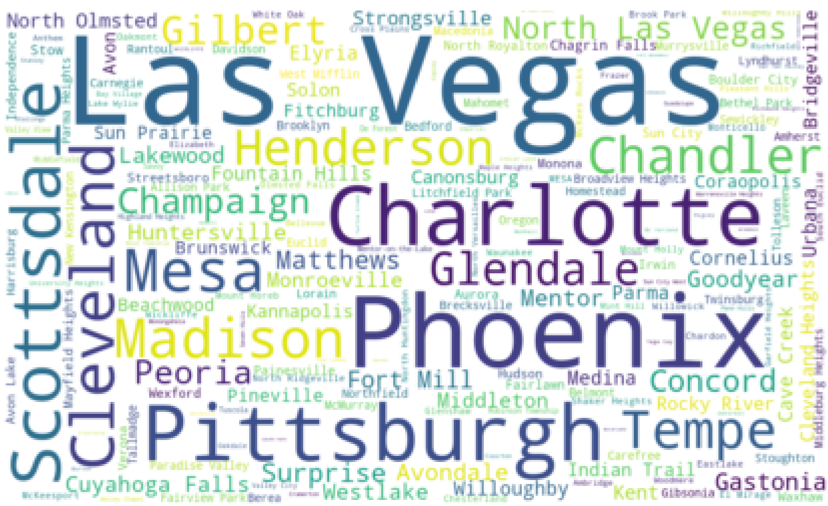


City-level

Figure 5. The **wordcloud** demonstrates the number of restaurants in each **city** through **font size**. **Las Vegas**, **Phoenix**, **Charlotte**, **Scottsdale**, **Cleveland** and **Pittsburgh** are cities with the **higest** number of restaurants.

In [25]:
```
city_count = us_restaurants['city'].value_counts()
wordcloud = WordCloud(width = 1000, height = 600,background_color='white', max_
font_size = 150,
                      min_font_size = 5, colormap = "viridis",
                      collocations = False).generate_from_frequencies(city_coun
t)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.tight_layout(pad = 1)
plt.figure(figsize = (200,100))
plt.show()
```



<Figure size 14400x7200 with 0 Axes>

C. Ratings and Cuisines

Rating

Figure 6. The **bar plot** presents the number of restaurants by **ratings (stars)**. In this dataset, there are only **375** restaurants winning **full scores** (5 stars) and **61** restaurants rated as **1-star**. Over **10,000** restaurants are rated between **3.0 and 4.0 stars**.

In [26]:
```
star_count = us_restaurants['stars'].value_counts().sort_values(ascending = False)
plt.figure(figsize = (12,5))
sns.barplot(star_count.index, star_count.values, alpha = 0.8, order = star_count.index)
plt.title('Number of Restaurants by Ratings', fontsize = 16)
plt.ylabel('Number of Restaurants', fontsize = 12)
plt.xlabel('Number of Stars', fontsize = 12)
for i, v in enumerate(star_count):
    plt.text(i, v, str(v), horizontalalignment ='center', fontsize = 11)
plt.show()
```

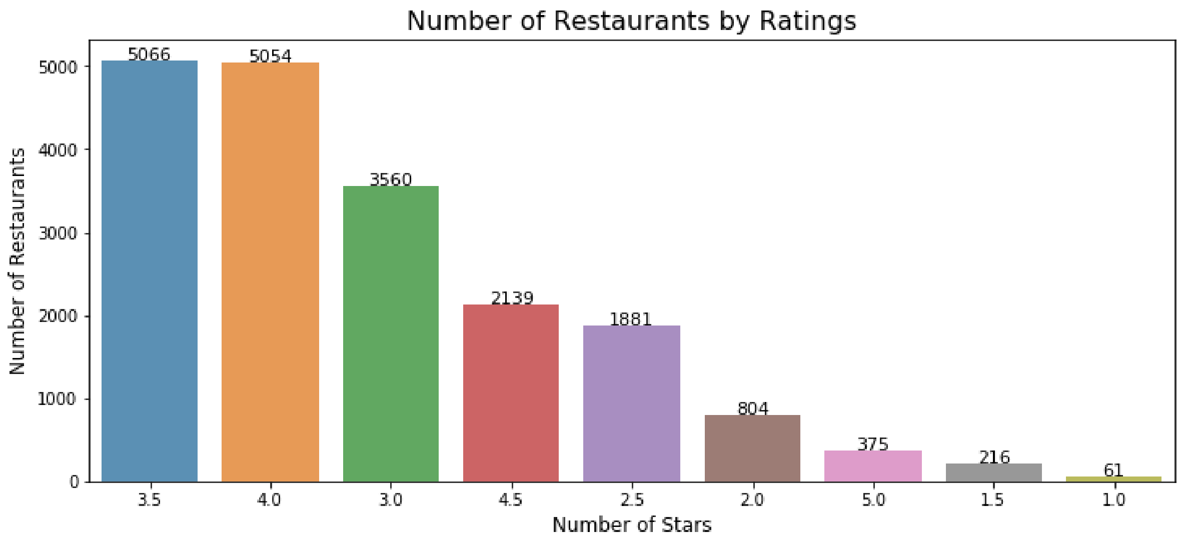
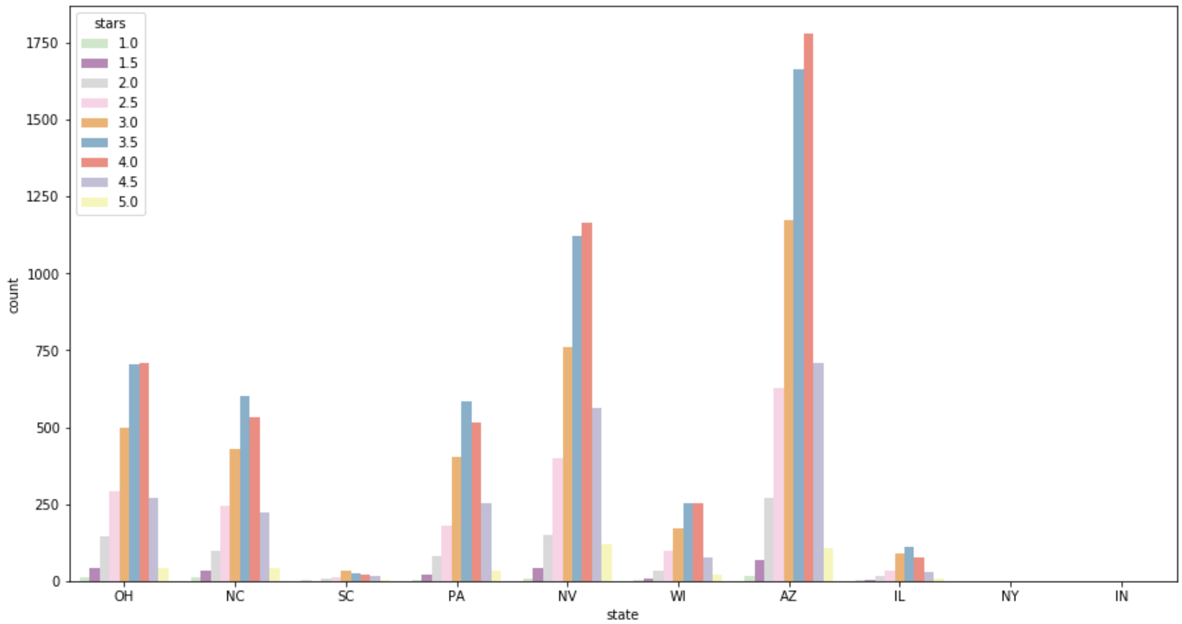


Figure 7. The **bar plot** presents the number of restaurants according to **ratings (stars)** in each state. Corresponding to the above plot, in most of the states, the **mode** is **3.5** or **4.0** stars.

In [27]:
```
fig,ax = plt.subplots()
fig.set_size_inches(15, 8)
sns.countplot(x="state", hue = 'stars', palette='Set3_r', data=us_restaurants)
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1c1b7fe810>



Cuisine

Figure 8. This **wordcloud** shows that the **three** main types of cuisine in the United States, based on the Yelp restaurant dataset, are **American**, **Mexican** and **Italian**. **Asian** and **Mediterranean** cusines are also highly welcomed according to the **number of each type of restaurants**.

In [28]:
```
category_count = us_restaurants['category'].value_counts()
wordcloud = WordCloud(width = 550, height = 300, background_color = 'white',
                      max_font_size = 160, min_font_size = 10, colormap = "nipy
_spectral",
                      collocations = False).generate_from_frequencies(category_
count)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.tight_layout(pad = 1)
plt.figure(figsize = (100,60))
plt.show()
```


```
<Figure size 7200x4320 with 0 Axes>
```

Figure 9. More explicitly, this **bar plot** sorts the number of restaurant **by cusine** in descending order and presents that there are **7314 American** restaurants and only **47 African** restaurants in the United States.

In [29]:
```
plt.figure(figsize=(15,6))
sns.barplot(category_count.index, category_count.values, alpha=0.8, order = cat
egory_count.index)
plt.title('Number of Restaurants by Cuisine', fontsize = 16)
plt.ylabel('Number of Restaurants', fontsize = 12)
plt.xlabel('Type of Cuisine', fontsize = 12)
for i, v in enumerate(category_count):
    plt.text(i, v, str(v), horizontalalignment ='center', fontsize = 11)
plt.show()
```

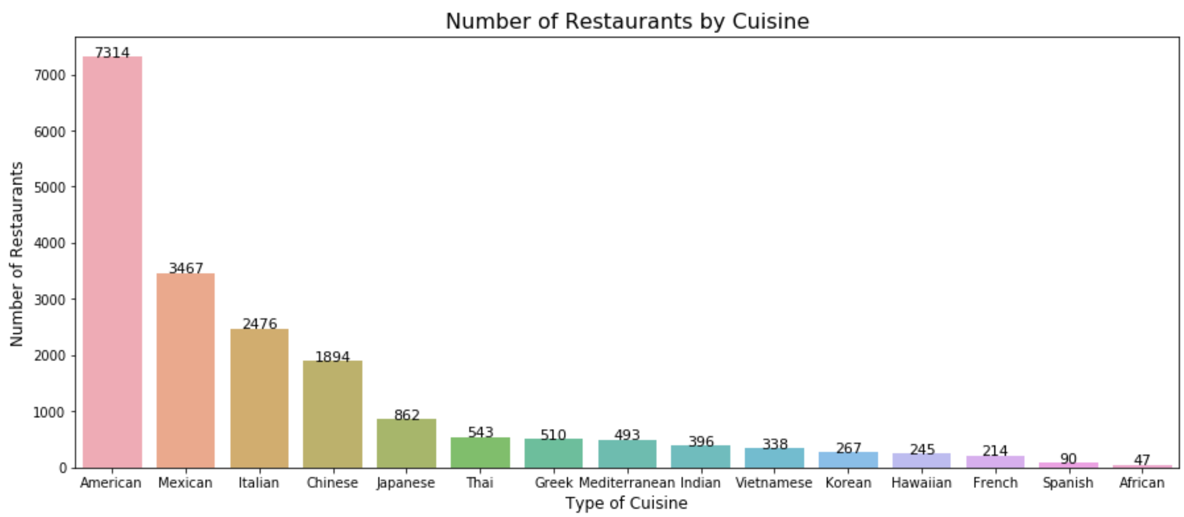
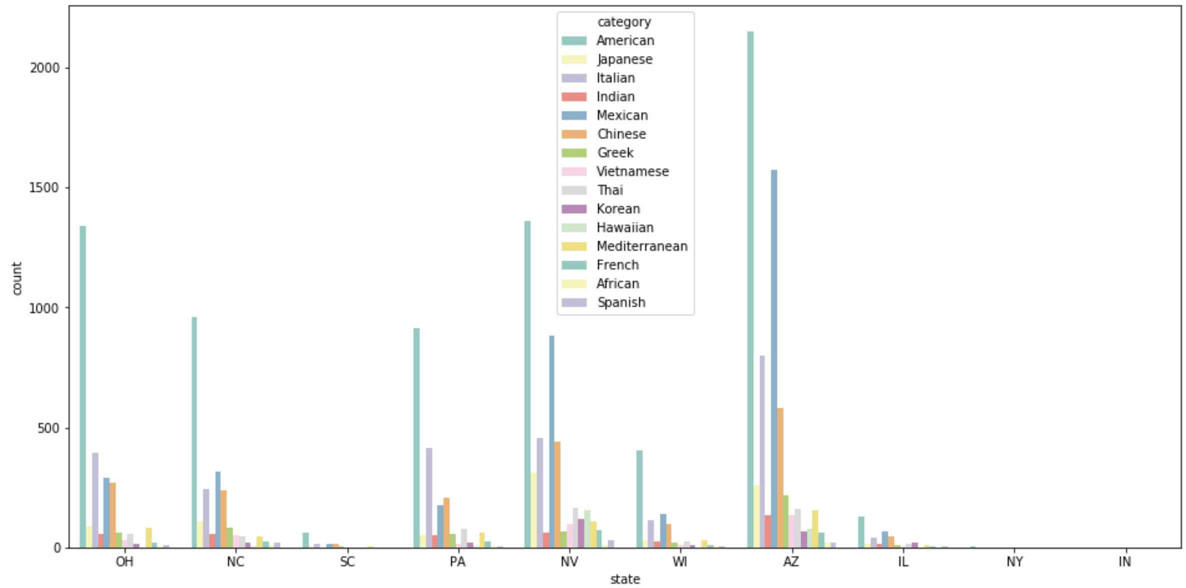


Figure 10. The **bar plot** presents the number of restaurants according to the **type of cuisines** in **each state**. It is evident that the **American cuisine** indeed dominates in nearly every state. Moreover, there are also a large number of **Mexican** restaurants in **Neveda** and **Arizona**.

In [30]:
```
fig,ax = plt.subplots()
fig.set_size_inches(16, 8)
sns.countplot(x="state", hue = 'category', palette="Set3", data=us_restaurants)
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1fd8a6d0>
```



II. Data Relationships

Yelp connects people with great local businesses. For customers, rating a business with stars and reviews helps elevate next experience. For restaurant business owners, deciding on which business attribute to improve and cuisine type to provide is crucial in order to receive more positive feedbacks.

This section concentrates mainly on **relationships** between **restaurant rating** and its **relevant factors** that include **attributes** and **cuisine types**. **Multiple linear regressions** have been presented in the **second part** of this section for a more **direct** demonstration.

2.1 Relationship Analysis of Star Ratings

A. Relationship between Review Counts and Ratings

In [31]:
```
print(smf.ols('stars ~ review_count',data=us_restaurants_attribute).fit().summa
ry())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                  stars   R-squared:                       0.0
30
Model:                            OLS   Adj. R-squared:                  0.0
30
Method:                 Least Squares   F-statistic:                     59
1.8
Date:                Wed, 13 May 2020   Prob (F-statistic):          9.44e-1
29
Time:                        01:40:19   Log-Likelihood:                -2042
9.
No. Observations:               18961   AIC:                         4.086e+
04
Df Residuals:                   18959   BIC:                         4.088e+
04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
====
                   coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
Intercept        3.4282      0.006    596.747      0.000       3.417
3.440
review_count     0.0006   2.56e-05     24.327      0.000       0.001
0.001
==============================================================================
==
Omnibus:                      516.880   Durbin-Watson:                   1.9
80
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              559.8
32
Skew:                          -0.412   Prob(JB):                    2.72e-1
22
Kurtosis:                       3.168   Cond. No.                         24
9.
==============================================================================
==

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

Independent variable - **'review_count'** - is **positively correlated** with **ratings** and its corresponding **coefficient** is **0.0006**. It indicates that the **rating** of a restaurant will have **increased by 0.0006** if there is **one more review**. The **Adjusted R-square** of this model is **3.0%**.

B. Relationship between Cuisines and Ratings

In [32]:
```
us_restaurants_attribute[['category','stars']].groupby('category').mean().sort_
values(by = ['stars'], ascending = False)
```

Out[32]:

category	stars
French	3.950000
Mediterranean	3.884774
Hawaiian	3.849794
Greek	3.819085
Thai	3.817343
Spanish	3.797753
African	3.776596
Vietnamese	3.760479
Korean	3.721374
Indian	3.676396
Japanese	3.658140
Italian	3.464809
Mexican	3.427841
American	3.420282
Chinese	3.299412

The above **data frame** presents the **average rating by cuisine**. According to the Yelp dataset, **French** restaurants in the United States have the **highest** average rating of **3.95 stars**, whereas **Chinese** restaurants only have **3.30 stars** on average.

C. Relationship between States and Ratings

In [33]:
```
us_restaurants_attribute[['state','stars']].groupby('state').mean().sort_values
(by = ['stars'], ascending = False)
```

Out[33]:

	stars
state	
NY	3.666667
NV	3.553055
PA	3.508041
IN	3.500000
AZ	3.495213
WI	3.471207
NC	3.433303
OH	3.432498
IL	3.380822
SC	3.368421

The above **data frame** presents the **average rating by state**. According to the Yelp dataset, restaurants in **New York State** have the **highest** average rating of **3.67 stars**, whereas restaurants in **South Carolina** only have **3.37 stars** on average.

2.2 Multiple Linear Regression

A. Multiple Linear Regression: Attributes and Ratings

Model 1. Relationship between Attributes and Restaurant Ratings (Full Model)

Dependent Variable: 'stars'

Independent Variable: 'review_count', 'BusinessParking_lot', 'WheelchairAccessible', 'Alcohol', 'RestaurantsTakeOut', 'HappyHour', 'RestaurantsDelivery', 'Smoking'

In [34]:
```
x = us_restaurants_attribute[['review_count','BusinessParking_lot' ,'WheelchairAccessi
ble' ,'Alcohol',
                              'RestaurantsTakeOut' ,'HappyHour', 'RestaurantsDelivery'
, 'Smoking']]
y = us_restaurants_attribute['stars']
x = sm.add_constant(x)
model = sm.OLS(y,x).fit()
predictions = model.predict(x)
print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  stars   R-squared:                       0.032
Model:                            OLS   Adj. R-squared:                  0.032
Method:                 Least Squares   F-statistic:                     78.39
Date:                Wed, 13 May 2020   Prob (F-statistic):          5.08e-128
Time:                        01:40:19   Log-Likelihood:                -20412.
No. Observations:               18961   AIC:                         4.084e+04
Df Residuals:                   18952   BIC:                         4.091e+04
Df Model:                           8
Covariance Type:            nonrobust
=================================================================================
===
                          coef    std err          t      P>|t|      [0.025      0.9
75]
---------------------------------------------------------------------------------
---
const                   3.4120      0.007    510.977      0.000       3.399       3.
425
review_count            0.0006   2.59e-05     24.661      0.000       0.001       0.
001
BusinessParking_lot     0.2890      0.081      3.561      0.000       0.130       0.
448
WheelchairAccessible    0.0539      0.012      4.387      0.000       0.030       0.
078
Alcohol                 0.3097      0.251      1.233      0.218      -0.183       0.
802
RestaurantsTakeOut     -0.1657      0.225     -0.737      0.461      -0.606       0.
275
HappyHour               0.0480      0.047      1.031      0.302      -0.043       0.
139
RestaurantsDelivery     0.1565      0.269      0.583      0.560      -0.370       0.
683
Smoking                 0.1890      0.251      0.752      0.452      -0.303       0.
681
==============================================================================
Omnibus:                      535.075   Durbin-Watson:                   1.980
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              581.303
Skew:                          -0.419   Prob(JB):                    5.91e-127
Kurtosis:                       3.179   Cond. No.                     1.17e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 1.17e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2495: FutureWarn
ing: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp
instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Analysis:

Coefficients of **'review_count'**, of **'BusinessParking_lot'** and of **'WheelchairAccessible'** are **significant** on **95%** level.

Model has a **3.2%** Adjusted R-square, which gives a good base and would improve as much.

Model 2. Relationship between Attributes and Restaurant Ratings (Only with Significant Regressors)

Dependent Variable: 'stars'

Independent Variable: 'review_count', 'BusinessParking_lot', 'WheelchairAccessible'

In [35]:
```
x = us_restaurants_attribute[['review_count','BusinessParking_lot' ,'WheelchairAccessi
ble']]
y = us_restaurants_attribute['stars']
x = sm.add_constant(x)
model = sm.OLS(y,x).fit()
predictions = model.predict(x)
print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  stars   R-squared:                       0.032
Model:                            OLS   Adj. R-squared:                  0.032
Method:                 Least Squares   F-statistic:                     207.7
Date:                Wed, 13 May 2020   Prob (F-statistic):          1.46e-132
Time:                        01:40:19   Log-Likelihood:                -20414.
No. Observations:               18961   AIC:                         4.084e+04
Df Residuals:                   18957   BIC:                         4.087e+04
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
=====================================================================================
===
                           coef    std err          t      P>|t|      [0.025      0.9
75]
-------------------------------------------------------------------------------------
---
const                    3.4133      0.007    516.770      0.000       3.400       3.
426
review_count             0.0006   2.59e-05     24.634      0.000       0.001       0.
001
BusinessParking_lot      0.2881      0.081      3.550      0.000       0.129       0.
447
WheelchairAccessible     0.0527      0.012      4.307      0.000       0.029       0.
077
==============================================================================
Omnibus:                      529.532   Durbin-Watson:                   1.980
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              574.777
Skew:                          -0.417   Prob(JB):                    1.54e-125
Kurtosis:                       3.177   Cond. No.                     3.52e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 3.52e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Analysis:

Independent variables - 'review_count', 'BusinessParking_lot' and **'WheelchairAccessible'** - are **positively** associated with **restaurant ratings** with coefficients **0.0006**, **0.2881** and **0.0527**, respectively.

This model has a **3.2%** Adjusted R-square.

B. Multiple Linear Regression: Cuisines and Ratings

In [36]:
```
dummy = pd.get_dummies(us_restaurants_attribute['category'])
us_restaurants_attribute_reg = pd.concat([us_restaurants_attribute,dummy],axis=
1)
us_restaurants_attribute_reg.head()
```

Out[36]:

	business_id	name	address	city	state	postal_code	latitu
0	PfOCPjBrlQAnz__NXj9h_w	Brick House Tavern + Tap	581 Howe Ave	Cuyahoga Falls	OH	44221	41.119
1	fNMVV_ZX7CJSDWQGdOM8Nw	Showmars Government Center	600 E 4th St	Charlotte	NC	28202	35.221
2	gAy4LYpsScrj8POnCW6btQ	Toast Cafe	2429 Hwy 160 W	Fort Mill	SC	29708	35.047
3	tRVx2c89coruPRwYhGTcTw	Yuzu	13603 Madison Ave	Lakewood	OH	44107	41.476
4	BnuzcebyB1AfxHokjNWqSg	Carrabba's Italian Grill	245 Lancaster Ave	Frazer	PA	19355	40.041

In [37]:
```
us_restaurants_attribute_reg.columns
```

Out[37]:
```
Index(['business_id', 'name', 'address', 'city', 'state', 'postal_code',
       'latitude', 'longitude', 'stars', 'review_count', 'category',
       'BusinessParking_lot', 'WheelchairAccessible', 'Alcohol',
       'RestaurantsTakeOut', 'HappyHour', 'RestaurantsDelivery', 'Smoking',
       'African', 'American', 'Chinese', 'French', 'Greek', 'Hawaiian',
       'Indian', 'Italian', 'Japanese', 'Korean', 'Mediterranean', 'Mexica
n',
       'Spanish', 'Thai', 'Vietnamese'],
      dtype='object')
```

Model 1. Relationship between Cuisines and Restaurant Ratings (Full Model)

Dependent Variable: 'stars'

Independent Variable: 'review_count', 'BusinessParking_lot', 'WheelchairAccessible','Alcohol', 'RestaurantsTakeOut', 'HappyHour', 'RestaurantsDelivery', 'Smoking', 'African', 'American', 'Chinese', 'French', 'Greek', 'Hawaiian', 'Indian', 'Italian', 'Japanese', 'Korean', 'Mediterranean', 'Mexican', 'Spanish', 'Thai', 'Vietnamese'

In [38]:
```
x = us_restaurants_attribute_reg[['review_count',
       'BusinessParking_lot', 'WheelchairAccessible', 'Alcohol',
       'RestaurantsTakeOut', 'HappyHour', 'RestaurantsDelivery', 'Smoking',
       'African', 'American', 'Chinese', 'French', 'Greek', 'Hawaiian',
       'Indian', 'Italian', 'Japanese', 'Korean', 'Mediterranean', 'Mexican',
       'Spanish', 'Thai', 'Vietnamese']]
y = us_restaurants_attribute_reg['stars']
x = sm.add_constant(x)
model = sm.OLS(y,x).fit()
predictions = model.predict(x)
print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  stars   R-squared:                       0.074
Model:                            OLS   Adj. R-squared:                  0.073
Method:                 Least Squares   F-statistic:                     68.67
Date:                Wed, 13 May 2020   Prob (F-statistic):         1.89e-294
Time:                        01:40:19   Log-Likelihood:                -19993.
No. Observations:               18961   AIC:                         4.003e+04
Df Residuals:                   18938   BIC:                         4.021e+04
Df Model:                          22
Covariance Type:            nonrobust
=====================================================================================
===
                           coef    std err          t      P>|t|      [0.025      0.9
75]
-------------------------------------------------------------------------------------
---
const                    3.3767      0.011    294.311      0.000       3.354       3.
399
review_count             0.0006   2.55e-05     22.807      0.000       0.001       0.
001
BusinessParking_lot      0.2355      0.079      2.964      0.003       0.080       0.
391
WheelchairAccessible     0.0517      0.012      4.254      0.000       0.028       0.
076
Alcohol                  0.4012      0.246      1.632      0.103      -0.081       0.
883
RestaurantsTakeOut      -0.1541      0.220     -0.701      0.483      -0.585       0.
277
HappyHour                0.0593      0.046      1.300      0.194      -0.030       0.
149
RestaurantsDelivery      0.1649      0.263      0.627      0.530      -0.350       0.
680
Smoking                  0.1971      0.246      0.801      0.423      -0.285       0.
679
African                  0.3005      0.095      3.148      0.002       0.113       0.
487
American                -0.0271      0.013     -2.076      0.038      -0.053      -0.
002
Chinese                 -0.1334      0.018     -7.225      0.000      -0.170      -0.
097
French                   0.4251      0.046      9.193      0.000       0.334       0.
516
Greek                    0.3795      0.031     12.297      0.000       0.319       0.
440
Hawaiian                 0.3825      0.043      8.889      0.000       0.298       0.
467
Indian                   0.2313      0.034      6.719      0.000       0.164       0.
299
Italian                  0.0216      0.017      1.281      0.200      -0.011       0.
055
Japanese                 0.1782      0.025      7.250      0.000       0.130       0.
226
Korean                   0.2435      0.042      5.861      0.000       0.162       0.
325
Mediterranean            0.4352      0.031     13.891      0.000       0.374       0.
497
Mexican                 -0.0119      0.015     -0.775      0.438      -0.042       0.
018
Spanish                  0.3173      0.070      4.550      0.000       0.181       0.
454
Thai                     0.3493      0.030     11.698      0.000       0.291       0.
408
Vietnamese               0.2851      0.037      7.680      0.000       0.212       0.
```

```
358
==============================================================================
Omnibus:                      486.381   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              526.848
Skew:                          -0.392   Prob(JB):                    3.95e-115
Kurtosis:                       3.229   Cond. No.                     1.78e+18
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The smallest eigenvalue is 2.99e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Analysis:

Coefficients of **'review_count'**, of **'BusinessParking_lot'**, of **'WheelchairAccessible'**, of **'African'**, of **'American'**, of **'Chinese'**, of **'French'**, of **'Greek'**, of **'Hawaiian'**, of **'Indian'**, of **'Japanese'**, of **'Korean'**, of **'Mediterranean'**, of **'Spanish'**, of **'Thai'**, and of **'Vietnamese'** are **significant** on **95%** level.

This model has a **7.3%** Adjusted R-square.

Model 2. Relationship between Cuisines and Restaurant Ratings (Only with Significant Regressors)

Dependent Variable: 'stars'

Independent Variable: 'review_count', 'BusinessParking_lot', 'WheelchairAccessible', 'African', 'Chinese', 'French', 'Greek', 'Hawaiian', 'Indian', 'American', 'Japanese', 'Korean', 'Mediterranean', 'Spanish', 'Thai', 'Vietnamese'

In [39]:
```
x = us_restaurants_attribute_reg[['review_count',
       'BusinessParking_lot', 'WheelchairAccessible',
       'African', 'Chinese', 'French', 'Greek', 'Hawaiian',
       'Indian', 'American', 'Japanese', 'Korean', 'Mediterranean',
       'Spanish', 'Thai', 'Vietnamese']]
y = us_restaurants_attribute_reg['stars']
x = sm.add_constant(x)
model = sm.OLS(y,x).fit()
predictions = model.predict(x)
print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  stars   R-squared:                       0.073
Model:                            OLS   Adj. R-squared:                  0.073
Method:                 Least Squares   F-statistic:                     93.83
Date:                Wed, 13 May 2020   Prob (F-statistic):          2.89e-298
Time:                        01:40:19   Log-Likelihood:                -19997.
No. Observations:               18961   AIC:                         4.003e+04
Df Residuals:                   18944   BIC:                         4.016e+04
Df Model:                          16                                         
Covariance Type:            nonrobust                                         
==============================================================================
===
                          coef    std err          t      P>|t|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
const                   3.3805      0.010    339.024      0.000       3.361       3.
400
review_count            0.0006   2.55e-05     22.780      0.000       0.001       0.
001
BusinessParking_lot     0.2347      0.079      2.954      0.003       0.079       0.
390
WheelchairAccessible    0.0491      0.012      4.056      0.000       0.025       0.
073
African                 0.2990      0.102      2.937      0.003       0.099       0.
499
Chinese                -0.1351      0.018     -7.313      0.000      -0.171      -0.
099
French                  0.4237      0.049      8.643      0.000       0.328       0.
520
Greek                   0.3769      0.032     11.673      0.000       0.314       0.
440
Hawaiian                0.3801      0.046      8.352      0.000       0.291       0.
469
Indian                  0.2287      0.036      6.322      0.000       0.158       0.
300
American               -0.0292      0.012     -2.377      0.017      -0.053      -0.
005
Japanese                0.1756      0.025      6.905      0.000       0.126       0.
225
Korean                  0.2421      0.044      5.514      0.000       0.156       0.
328
Mediterranean           0.4326      0.033     13.187      0.000       0.368       0.
497
Spanish                 0.3148      0.074      4.239      0.000       0.169       0.
460
Thai                    0.3466      0.031     11.101      0.000       0.285       0.
408
Vietnamese              0.2831      0.039      7.235      0.000       0.206       0.
360
==============================================================================
Omnibus:                      478.825   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              517.657
Skew:                          -0.390   Prob(JB):                    3.91e-113
Kurtosis:                       3.219   Cond. No.                     4.52e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 4.52e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Analysis:

There are **positive correlations** between **most of the independent variables** and **restaurant ratings**. However, regressors - **Chinese** and **American** - are **negatively** related to **ratings** with coefficients **-0.1351** and **-0.0292**, respectively. Therefore, according to these **negative coefficients**, restaurants that offer either **Chinese** or **American** cuisine receive **lower ratings**, holding other factors constant.

This model has a **7.3%** Adjusted R-square.

2.3 Linear Regression Model Using Machine Learning

The **linear model** is fitted. The **coefficient of determination** (**R-square**) of the prediction is stated as the **accuracy score** down below:

In [40]:
```
x = us_restaurants_attribute_reg[['review_count',
       'BusinessParking_lot', 'WheelchairAccessible', 'Alcohol',
       'RestaurantsTakeOut', 'HappyHour', 'RestaurantsDelivery', 'Smoking',
       'African', 'American', 'Chinese', 'French', 'Greek', 'Hawaiian',
       'Indian', 'Italian', 'Japanese', 'Korean', 'Mediterranean', 'Mexican',
       'Spanish', 'Thai', 'Vietnamese']]
y = us_restaurants_attribute_reg['stars']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y)
from sklearn.linear_model import LinearRegression
skl_lin_regr = LinearRegression().fit(x_train,y_train)
print('Accuracy Score', skl_lin_regr.score(x_test,y_test))
```

```
Accuracy Score 0.06091557478417298
```

Estimate the **intercept** and **coefficients** for the linear regression

In [41]:
```
coeff_df = pd.DataFrame(data = skl_lin_regr.coef_,index= [['review_count',
       'BusinessParking_lot', 'WheelchairAccessible', 'Alcohol',
       'RestaurantsTakeOut', 'HappyHour', 'RestaurantsDelivery', 'Smoking',
       'African', 'American', 'Chinese', 'French', 'Greek', 'Hawaiian',
       'Indian', 'Italian', 'Japanese', 'Korean', 'Mediterranean', 'Mexican',
       'Spanish', 'Thai', 'Vietnamese']], columns=['Coefficient'])
print('intercept',skl_lin_regr.intercept_)
print(coeff_df)
```

```
intercept 3.588553178637596
                      Coefficient
review_count             0.000647
BusinessParking_lot      0.227975
WheelchairAccessible     0.060645
Alcohol                  0.436569
RestaurantsTakeOut      -0.278883
HappyHour                0.075031
RestaurantsDelivery      0.174091
Smoking                  0.149317
African                  0.037444
American                -0.253741
Chinese                 -0.335352
French                   0.178142
Greek                    0.170937
Hawaiian                 0.142166
Indian                   0.013161
Italian                 -0.190130
Japanese                -0.063393
Korean                   0.028687
Mediterranean            0.240314
Mexican                 -0.236008
Spanish                  0.061829
Thai                     0.149302
Vietnamese               0.056641
```

Generate **predicted values (y_hat)** for restaurant ratings by using the **linear model**

In [42]:
```
lin_reg_predict_stars = pd.DataFrame(data=skl_lin_regr.predict(x_test))
lin_reg_predict_stars.rename(columns={0:'Stars_Rating_Predictions'}, inplace=True)
lin_reg_predict_stars.set_index('Stars_Rating_Predictions')
```

Out[42]:

Stars_Rating_Predictions
6.858890
3.429535
3.850802
3.342576
3.349046
...
3.255789
3.565400
3.412450
3.416161
3.388776

4741 rows × 0 columns

III. West Coast Restaurants Recommendation Tool

This section discovers **restaurants on the West Coast**. Three West Coast restaurant **recommendation tools** have been created and can be applied for restaurant search through entering **1) restaurant numbers**, **2) city indices** and **3) needs for parking facilities** with detailed instructions.

A. Filter Restaurants Located on the West Coast (CA, NV, AZ, OR, WA)

Filter Restaurants on the West Coast

In [43]:
```
west_states = ["CA","NV","AZ","OR","WA"]
western = business.loc[business['state'].isin(west_states)]
restaurants = western[western['categories'].str.contains('Restaurants')]
western = us_restaurants.loc[us_restaurants['state'].isin(west_states)]
restaurants
```

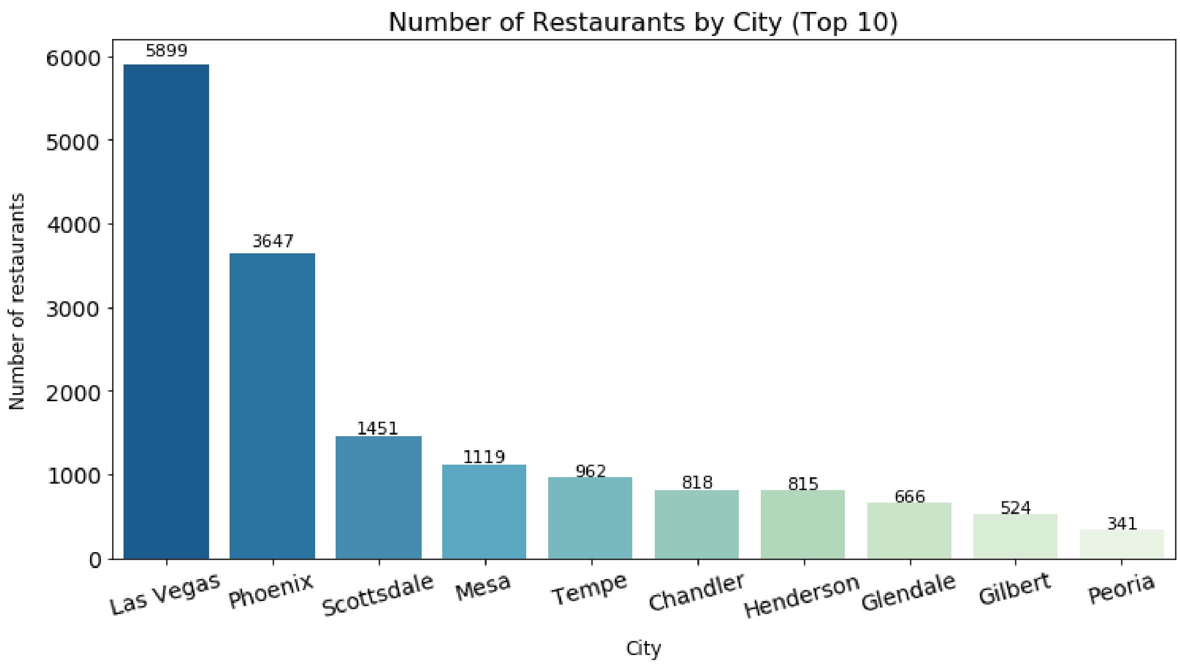
Out[43]:

	business_id	name	address	city	state	postal_code	
45	rDMptJYWtnMhpQu_rRXHng	McDonald's	719 E Thunderbird Rd	Phoenix	AZ	85022	3
46	1WBkAuQg81kokZIPMpn9Zg	Charr An American Burger Bar	777 E Thunderbird Rd, Ste 107	Phoenix	AZ	85022	3
52	Pd52CjgyEU3Rb8co6QfTPw	Flight Deck Bar & Grill	6730 S Las Vegas Blvd	Las Vegas	NV	89119	3
53	4srfPk1s8nlm1YusyDUbjg	Subway	6889 S Eastern Ave, Ste 101	Las Vegas	NV	89119	3
54	n7V4cD-KqqE3OXk0irJTyA	GameWorks	6587 Las Vegas Blvd S, Ste 171	Las Vegas	NV	89119	3
...	...	...	...	...	...	...	
174469	6HdzmAatFoB8UDep4n9QIw	IHOP	5170 S Fort Apache Rd	Las Vegas	NV	89148	3
174479	vGDhK2Lc4Np5iZYZ7FG0QA	Escobar Mexican Kitchen	1219 E Glendale Ave, Ste 14	Phoenix	AZ	85020	3
174504	5zva2MTtB5IX6TaoVLL-NA	Zorbas Grill	440 W Warner Rd	Tempe	AZ	85284	3
174520	Gr-20Bg4XyduSKbvnE-i9g	Salt & Lime Modern Mexican Grill	9397 E Shea Blvd, Ste 115	Scottsdale	AZ	85260	:
174558	UdEmYOnk2iJDY9lpEPAlJQ	Floridino's Pizza & Pasta	590 N Alma School Rd, Ste 35	Chandler	AZ	85224	3

17734 rows × 11 columns

Figure 11. The **bar plot** demonstrates **Top 10 cities on the West Coast** with the **highest number of restaurants**. The city with the **most** restaurants is **Las Vegas** and there are **5,899** restaurants located within that city. Other cities include **Phoenix**, **Scottsdale**, **Mesa**, **Tempe**, **Chandler**, **Henderson**, **Glendale**, **Gilbert**, **Peoria**.

In [44]:
```
city_rank = restaurants.city.value_counts()[:10]
plt.figure(figsize = (12,6))
sns.barplot(city_rank.index, city_rank.values, palette = sns.color_palette("GnB
u_r", len(city_rank)))
plt.title('Number of Restaurants by City (Top 10)', fontsize = 16)
plt.ylabel('Number of restaurants', fontsize = 12, labelpad = 10)
plt.xlabel('City', fontsize = 12, labelpad = 10)
plt.tick_params(labelsize = 14)
plt.xticks(rotation = 15)
for  i, v in enumerate(city_rank):
    plt.text(i, v*1.02, str(v), horizontalalignment = 'center', fontsize = 11)
```



Merge Business with Attributes for Restaurants on the West Coast

In [45]:
```
info = restaurants.merge(busi_attr,left_on='business_id',right_on='business_id'
,how='inner')
info
```

Out[45]:

	business_id	name	address	city	state	postal_code	l
0	rDMptJYWtnMhpQu_rRXHng	McDonald's	719 E Thunderbird Rd	Phoenix	AZ	85022	33
1	1WBkAuQg81kokZIPMpn9Zg	Charr An American Burger Bar	777 E Thunderbird Rd, Ste 107	Phoenix	AZ	85022	33
2	Pd52CjgyEU3Rb8co6QfTPw	Flight Deck Bar & Grill	6730 S Las Vegas Blvd	Las Vegas	NV	89119	36
3	4srfPk1s8nlm1YusyDUbjg	Subway	6889 S Eastern Ave, Ste 101	Las Vegas	NV	89119	36
4	n7V4cD-KqqE3OXkoirJTyA	GameWorks	6587 Las Vegas Blvd S, Ste 171	Las Vegas	NV	89119	36.
...	...	...	...	...	...	...	
17535	AEYNihHmGIjmUciRFo3qwA	Yin's Chinese Resturant	1950 W Indian School Rd	Phoenix	AZ	85015	33
17536	6HdzmAatFoB8UDep4n9QIw	IHOP	5170 S Fort Apache Rd	Las Vegas	NV	89148	36.
17537	vGDhK2Lc4Np5iZYZ7FG0QA	Escobar Mexican Kitchen	1219 E Glendale Ave, Ste 14	Phoenix	AZ	85020	33
17538	Gr-20Bg4XyduSKbvnE-i9g	Salt & Lime Modern Mexican Grill	9397 E Shea Blvd, Ste 115	Scottsdale	AZ	85260	3
17539	UdEmYOnk2iJDY9lpEPAlJQ	Floridino's Pizza & Pasta	590 N Alma School Rd, Ste 35	Chandler	AZ	85224	3

17540 rows × 92 columns

B. Restaurant Recommendation by *NAME*

In [46]:
```
def search_by_rest(df_name, sector_num):
    df_name = df_name.sort_values('name')
    df_name = df_name.dropna()
    dt = dict(enumerate(df_name['name'].unique()))
    if sector_num == 'library':
        print(dt)
        sector_num_2 = input("choose restaurant number (type 'library' to view
 options): ")
        return search_by_rest(df_name, sector_num_2)
    elif re.findall('[^0-9]',sector_num) != []:
        print('Invalid Input \n')
        sector_num_2 = input("choose restaurant number (type 'library' to view
 options): ")
        return search_by_sector(df_name, sector_num_2)
    elif int(sector_num) not in dt.keys():
        print('Invalid Input \n')
        sector_num_2 = input("choose restaurant number (type 'library' to view
 options): ")
        return search_by_sector(df_name, sector_num_2)
    else:
        result = df_name.groupby(['name']).get_group(dt[int(sector_num)])
        return result
```

Instruction:

This **recommendation tool** could be applied for searching **SPECIFIC restaurants on the West Coast** with a **list of restaurant numbers** in which **each number** corresponds to a **unique restaurant** in the above ***'info'*** data frame. Please type **'library'** in the space for the **entire restaurant list**. Or, enter a **number** directly for **restaurant search**.

In [47]:
```
search_by_rest(info,sector_num = input("choose sector number (type 'library' to
view options): "))
```

```
choose sector number (type 'library' to view options): 43
```

Out[47]:

	business_id	name	address	city	state	postal_code	latitude	longitude	st
6470	7SBtCKKeHbdB6-dIIkcZXw	2nd Annual Asian Food Festival	3333 Blue Diamond Rd	Las Vegas	NV	89139	36.041673	-115.184999	

C. Restaurant Recommendation by *CITY*

In [48]:
```
def search_by_city(df_city, city_num):
    df_city = df_city.sort_values('city')
    df_city = df_city.dropna()
    dt = dict(enumerate(df_city['city'].unique()))
    if city_num == 'library':
        print(dt)
        city_num_2 = input("choose city number (type 'library' to view option
s): ")
        return search_by_city(df_city, city_num_2)
    elif re.findall('[^0-9]', city_num) != []:
        print('Invalid Input \n')
        city_num_2 = input("choose city number (type 'library' to view option
s): ")
        return search_by_city(df_city, city_num_2)
    elif int(city_num) not in dt.keys():
        print('Invalid Input \n')
        city_num_2 = input("choose city number (type 'library' to view option
s): ")
        return search_by_city(df_city, city_num_2)
    else:
        result = df_city.groupby(['city']).get_group(dt[int(city_num)])
        return result
```

Instruction:

This **recommendation tool** could be applied for searching **West Coast restaurants by city index** with a **list of city indices** in which **each number** corresponds to a **city** on the West Coast. Please type **'library'** in the space for the **entire city list**. Or, enter a **number** directly for **city-level restaurant search**.

In [49]:
```
search_by_city(info, city_num = input("choose city index(type 'library' to view options): "))
```

```
choose city index(type 'library' to view options): 24
```

Out[49]:

	business_id	name	address	city	state	postal_code	latitu
16046	ro4h-3oXNxk4KLa6__jV5Q	Montesano's Italian Deli	3441 W Sahara Ave Ste B2	Las Vegas	NV	89102	36.1432
8898	7gIK3en6jVyiColBvlHNPA	Noble Roman's Take-n-Bake Pizza	4190 S Rainbow Blvd	Las Vegas	NV	89103	36.1125
16022	IF9C_h3NONhXZ7bbyhtnxA	Osi's Kitchen	4604 W Sahara Ave, Ste 6	Las Vegas	NV	89102	36.1449
8895	94GNGMxmruuqRLRhLqllWw	Wendy's	8900 W Charleston Blvd	Las Vegas	NV	89117	36.1595
8816	nt2-Zk4FmGY2SYSDBIogHw	Durango Taco Shop	7785 N Durango Dr	Las Vegas	NV	89131	36.3024
...	...	...	...	...	...	...	
3145	_8yxceldCT7oeowJC-yDPA	Benjarong Authentic Thai Cuisine	7425 S Durango Dr, Ste 101	Las Vegas	NV	89113	36.0533
459	OVTZNSkSfbl3gVB9XQIJfw	Cravings Buffet	3400 Las Vegas Blvd S	Las Vegas	NV	89109	36.1212
4564	VZQ7R-LYP8Ja9k_-ODnJpw	Z India	1116 S Rainbow Blvd	Las Vegas	NV	89146	36.1584
4562	ZCQa7CJxZ-53Zxd_pobWug	Le Cafe Ile St. Louis	3655 Las Vegas Blvd S	Las Vegas	NV	89109	36.1123
6883	FYqFfaxVRW6pdviONXIoDw	Fleming's Prime Steakhouse	6515 S Las Vegas Blvd	Las Vegas	NV	89119	36.0695

5818 rows × 92 columns

D. Restaurant Recommendations by *PARKING Facilities*

In [50]:
```
def search_by_parking(df_attribute,
                      garage=None, street=None, validated=None,
                     lot=None, valet=None):

    result = df_attribute
    temp =  ['False','Na','True']

    if garage == 'y':
        result = result.loc[result['BusinessParking_garage']=='True']
    else:
        result = result.loc[result['BusinessParking_garage'].isin(temp),:]

    if street == 'y':
        result = result.loc[result['BusinessParking_street'] == 'True']
    else:
        result = result.loc[result['BusinessParking_street'].isin(temp),:]

    if validated == 'y':
        result = result.loc[result['BusinessParking_validated']== 'True']
    else:
        result = result.loc[result['BusinessParking_validated'].isin(temp),:]

    if lot == 'y':
        result = result.loc[result['BusinessParking_lot']== 'True']
    else:
        result = result.loc[result['BusinessParking_lot'].isin(temp),:]

    if valet == 'y':
        result = result.loc[result['BusinessParking_valet']== 'True']
    else:
        result = result.loc[result['BusinessParking_valet'].isin(temp),:]

    return result
```

Instruction:

This **recommendation tool** could be applied for searching **West Coast restaurants by PARKING facilities**. Please type **'y/n'** in the space below for a **desirable restaurant search** according to your need for **parking****.

****Notes**: Most of the restaurants in this case **only** satisfy one of the conditions below. There are two restaurants in Las Vegas having both a garage and valet service.

In [51]:

```
search_by_parking(info,
                 garage=(input('Need a garage[y/n]:')),
                 street=(input('Need a street parking[y/n]:')),
                 validated=(input('Validated parking[y/n]:')),
                 lot=(input('Need a lot[y/n]:')),
                 valet=(input('Need a valet[y/n]:')))
```

```
Need a garage[y/n]:y
Need a street parking[y/n]:n
Validated parking[y/n]:n
Need a lot[y/n]:n
Need a valet[y/n]:y
```

Out[51]:

	business_id	name	address	city	state	postal_code	latitude	lon
2104	KjOpQ4QCf-sk_dbFJX_52w	Fido's Kitchen	7875 W Sahara Ave, Ste 103	Las Vegas	NV	89117	36.143204	-115.2
2433	tpZfJdRi64OTBN4g7lRM3Q	Seoul Market	1801 E Tropicana Ave, Ste 12	Las Vegas	NV	89119	36.100196	-115.1

- END -