YELP DATASET CODE

In [ ]:
```
#Import necessary libraries and packages
import pandas as pd
import numpy as np
from pandas.io.json import json_normalize
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
#Import primary CSV data file
business_original = pd.read_csv('business.csv')
```

In [ ]:
```
#First view of data
business_original.head(15)
```

In [ ]:
```
#Determine the actual span of our data
business_original.shape
```

In [ ]:
```
#Unique cities in dataset
business_original['city'].nunique()
```

In [ ]:
```
#Assess the data types for ease of analysis
business_original.info()
```

In [ ]:
```
#Check for duplicate records in the dataset
business_original.duplicated().sum()
```

In [ ]:
```
#Check for Null values
business_original.isnull().sum()
```

In [ ]:
```
#check what percentage of hours is null
(business_original.hours.isnull().sum()/len(business_original))*100
```

In [ ]:
```
#hours is not really useful and has a bad format so we drop it
business_original.drop('hours', axis=1, inplace=True)
```

In [ ]:
```
#updated span/size of dataset
business_original.shape
```

In [ ]:
```
#Analyze the state column to determine which states will be of use
business_original['state'].value_counts()
```

In [ ]:
```
#Visulaize the distribution above
ax = business_original['state'].value_counts()
ax.plot.bar(figsize = (16,4), title="Count of Business Records for each
 State")
```

In [ ]:
```
#Graph new order of states
filt = ['AZ','NV','NC','OH','PA']
state_filt= business_original['state'].isin(filt)
graph=business_original[state_filt]
```

In [ ]:
```
ax_1 = graph['state'].value_counts()
ax_1.plot.bar(figsize = (16,4), title="Count of Business Records for eac
h State")
```

In [ ]:
```
#Hence, filter needs only relevant states
filt1 = ['AZ','NV','NC','OH','PA']
state_filt1= business_original['state'].isin(filt1)
business = business_original[state_filt1]
business.head()
```

In [ ]:
```
business['state'].value_counts()
```

In [ ]:
```
#How many records do we have left to work with?
business.shape
```

In [ ]:
```
#Begin exploration of categories
#Check for null values
business['categories'].isnull().sum()
```

In [ ]:
```
#Replace null values
business["categories"].fillna("",inplace=True)
```

In [ ]:
```
#Reset index and drop unneccessry columns
business=business.reset_index().drop(columns=['Unnamed: 0','index'])
```

In [ ]:
```
#Filter out only records that fall into important categories
targets = ['Restaurants', 'Fast Food','Shopping','Beauty','Spa','Nightli
fe','Auto', 'Arts','Entertainment','Active Life']
business=business[business.categories.str.contains('|'.join(targets))]
```

In [ ]:
```
#What do we have left?
business.shape
```

In [ ]:
```
#CREATE FUNCTION TO SINGLE OUT AREA OF PRIMARY INTEREST FOR ANALYSIS
```

In [ ]:
```
def Restaurant(x):
     if ('restaurants' in x.lower()) or ('fast food' in x.lower()) or (
'restaurant' in x.lower()):
        return 1
     else:
        return 0
```

In [ ]:
```
business["Restaurant"] = business["categories"].apply(Restaurant)
business[["categories","Restaurant"]].head(10)
```

In [ ]:
```
business["Restaurant"].sum()
```

Extracting Attributes

In [ ]:
```
#Expand attributes columns by splitting and create dummy variables
business["attributes"]=business["attributes"].str.replace("{","")
business["attributes"]=business["attributes"].str.replace("}","")
business["attributes"]=business["attributes"].str.replace("'","")
business["attributes"]=business["attributes"].str.replace('"',"")
business["attributes"]=business["attributes"].astype(str)
pd.set_option('display.max_columns', 50)
business.head()
```

In [ ]:
```
#Create Parking variable
def Parking(x):
    if ('valet: True' in x) or ('garage: True' in x) or ('lot: True' in
x):
        return 1
    else:
        return 0
```

In [ ]:
```
business['Parking']=business['attributes'].apply(Parking)
```

In [ ]:
```
#Create Kid_friendly variable
def Kid_friendly(x):
    if 'GoodForKids: True' in x:
        return 1
    else:
        return 0
```

In [ ]:
```
business['Kid_friendly']=business['attributes'].apply(Kid_friendly)
```

In [ ]:
```
#Create Reservations variable
def Reservations(x):
    if 'RestaurantsReservations: True' in x:
        return 1
    else:
        return 0
```

In [ ]:
```
business['Reservations'] = business['attributes'].apply(Reservations)
```

In [ ]:
```
#Create Price range variable
def Price_Range(x):
    if 'RestaurantsPriceRange2: 1' in x:
        return 1
    elif 'RestaurantsPriceRange2: 2' in x:
        return 2
    elif 'RestaurantsPriceRange2: 3' in x:
        return 3
    else:
        return 4
```

In [ ]:
```
business['Price_Range'] = business['attributes'].apply(Price_Range)
```

In [ ]:
```
#Create creditcard variable
def Credit_card(x):
    if "BusinessAcceptsCreditCards: True" in x:
        return 1
    else:
        return 0
```

In [ ]:
```
business['Credit_card'] = business['attributes'].apply(Credit_card)
```

In [ ]:
```
#Create wheelchair access variable
def wheelchair_access(x):
    if 'WheelchairAccessible: True' in x:
        return 1
    else:
        return 0
```

In [ ]:
```
business['wheelchair_access'] = business['attributes'].apply(wheelchair_
access)
```

In [ ]:
```
#Create breakfast variable
def good_for_breakfast (x):
    if 'breakfast: True' in x:
        return 1
    else:
        return 0
```

In [ ]:
```
business['good_for_breakfast'] = business['attributes'].apply(good_for_b
reakfast)
```

In [ ]:
```
#Create lunch variable
def good_for_lunch (x):
    if 'lunch: True' in x:
        return 1
    else:
        return 0
```

In [ ]:
```
business['good_for_lunch'] = business['attributes'].apply(good_for_lunch
)
```

In [ ]:
```
#Create dinner variable
def good_for_dinner (x):
    if 'dinner: True' in x:
        return 1
    else:
        return 0
```

In [ ]:
```
business['good_for_dinner'] = business['attributes'].apply(good_for_dinn
er)
```

In [ ]:
```
#Create alcohol variable
def alcohol (x):
    if ('Alcohol: ufull_bar' in x) or ('Alcohol: ubeer_and_wine' in x):
        return 1
    else:
        return 0
```

In [ ]:
```
business['alcohol'] = business['attributes'].apply(alcohol)
```

In [ ]:
```
#Create happyhour variable
def happyhour (x):
    if 'HappyHour: True' in x :
        return 1
    else:
        return 0
```

In [ ]:
```
business['happyhour'] = business['attributes'].apply(happyhour)
```

In [ ]:
```
#Create wifi variable
def wifi (x):
    if ('WiFi: ufree' in x) or ('WiFi: free' in x) or ('WiFi: yes' in x)
or ('WiFi: uyes' in x) or ('WiFi: True' in x) or ('WiFi: uTrue' in x):
        return 1
    else:
        return 0
```

In [ ]:
```
business['wifi'] = business['attributes'].apply(wifi)
```

In [ ]:
```
#Create table service variable
def table_service (x):
    if 'RestaurantsTableService: True' in x :
        return 1
    else:
        return 0
```

In [ ]:
```
business['table_service'] = business['attributes'].apply(table_service)
```

In [ ]:
```
#Create Entertainment
def Entertainment (x):
    if ('HasTV: True' in x) or ('dj: True' in x) or ('background_music:
 True' in x) or ('jukebox: True' in x) or ('live: True' in x) or ('vide
o: True' in x) or ('karaoke: True' in x):
        return 1
    else:
        return 0
```

In [ ]:
```
business['Entertainment'] = business['attributes'].apply(Entertainment)
```

In [ ]:
```
#Create takeout variable
def takeout (x):
    if 'RestaurantsTakeOut: True' in x :
        return 1
    else:
        return 0
```

In [ ]:
```
business['Takeout'] = business['attributes'].apply(takeout)
```

In [ ]:
```
#Create Noise_Level variable

def Noise_Level(x):
    if ('NoiseLevel: uquiet' in x) or ('NoiseLevel: quiet' in x):
        return 1
    elif ('NoiseLevel: uaverage' in x) or ('NoiseLevel: average' in x):
        return 2
    elif ('NoiseLevel: uloud' in x) or ('NoiseLevel: loud' in x):
        return 3
    else:
        return 4
```

In [ ]:
```
business['Noise_Level'] = business['attributes'].apply(Noise_Level)
```

In [ ]:
```
#Create Reservations variable

def Reservations (x):
    if 'RestaurantsReservations: True' in x :
        return 1
    else:
        return 0
```

In [ ]:
```
business['Reservations'] = business['attributes'].apply(Reservations)
```

In [ ]:
```
#Create Delivery variable

def Delivery (x):
    if 'RestaurantsDelivery: True' in x :
        return 1
    else:
        return 0
```

In [ ]:
```
business['Delivery'] = business['attributes'].apply(Delivery)
```

Extracting Categories

In [ ]:
```
#Create FastFood variable
def FastFood (x):
    if 'Fast Food' in x :
        return 1
    else:
        return 0
```

In [ ]:
```
business['FastFood'] = business['categories'].apply(FastFood)
```

In [ ]:
```
#Create Ethnicity variable
def ethnicity (x):
    if ('american' in x.lower()) or ('burgers' in x.lower()):
        return 'American'
    elif 'chinese' in x.lower():
        return 'Chinese'
    elif ('mexican' in x.lower()) or ("tex-mex"in x.lower()):
        return 'Mexican'
    elif 'italian' in x.lower():
        return 'Italian'
    elif ('japanese' in x.lower()) or ('sushi' in x.lower()):
        return 'Japanese'
   # elif 'thai' in x.lower():
      #  return 'Thai'
   # elif 'indian' in x.lower():
      #  return 'Indian'
   # elif 'korean' in x.lower():
      #  return 'Korean'
    else:
        return 'other'
```

In [ ]:
```
business['Ethnicity'] = business['categories'].apply(ethnicity)
```

In [ ]:
```
#Remove foreign symbols from name to allow for counting chains
business["name"]=business["name"].str.replace(' ','')
business["name"]=business["name"].str.replace("'",'')
business["name"]=business["name"].str.replace(',','')
business["name"]=business["name"].str.replace('.','')

business["name"]=business["name"].astype(str)
business["name"]=business["name"].str.lower()
```

In [ ]:
```
#Select only restaurants for data analysis before chain is counted
Rest_filt= business["Restaurant"]==1
Restaurant=business[Rest_filt]
Restaurant.head(10)
```

In [ ]:
```
#Create chain counts column by counting occurence of names
Restaurant['Chain_Counts'] = Restaurant.groupby(['name'])['name'].transf
orm('count')
```

In [ ]:
```
#Declare chain if chain counts is 4 or more.
def Chain (x):
    if x >= 4 :
        return 1
    else:
        return 0
```

In [ ]:
```
#Create Is_Chain column
Restaurant['Is_Chain'] = Restaurant['Chain_Counts'].apply(Chain)
```

In [ ]:
```
#Drop longitude and latitude since they're not needed
Restaurant.drop(columns=['longitude','latitude'], inplace=True)
```

In [ ]:
```
#Confirm shape of DF
Restaurant.shape
```

In [ ]:
```
#Check for number of Open restaurants
Restaurant['is_open'].sum()
```

In [ ]:
```
#Check for number of Closed restaurants
len(Restaurant['is_open'])-(Restaurant['is_open'].sum())
```

In [ ]:
```
#Check again for null values
Restaurant.isnull().sum()
```

In [ ]:
```
#Make pie chart to show distribution of open and closed businesses'

# Pie chart
labels = ["Open", 'Closed']
sizes = [23867, 11438]
#colors
colors = ['Lime','Red']

fig1, ax1 = plt.subplots(figsize=(10,5))
fig1.subplots_adjust(0.3,0,1,1)
patches, texts, autotexts = ax1.pie(sizes, colors = colors, labels=label
s, autopct='%1.1f%%', startangle=90)
for text in texts:
    text.set_color('black')
    text.set_size(12)
for autotext in autotexts:
    autotext.set_color('black')
    autotext.set_size(14)

# Equal aspect ratio ensures that pie is drawn as a circle
ax1.axis('equal')
plt.tight_layout()
plt.show()
```

In [ ]:
```
Restaurant.state.value_counts()
```

In [ ]:
```
Restaurant.postal_code.value_counts() #Reject
```

In [ ]:
```
#Check for ethnicity distribution
#Looks very skewed so it may not be used. There are 600 levels. This doe
s not seem feasible for analysis within this time frame.
Restaurant.Ethnicity.value_counts()
```

In [ ]:
```
Restaurant.head()
```

In [ ]:
```
Restaurant.shape
```

In [ ]:
```

```