ETL PROJECT

Toronto New Restaurant Location Analysis

Submitted By: Dong Yi Kim | Saif Gorges | Saloni Gupta | Sooyeon Kim

Date: December 17th, 2020

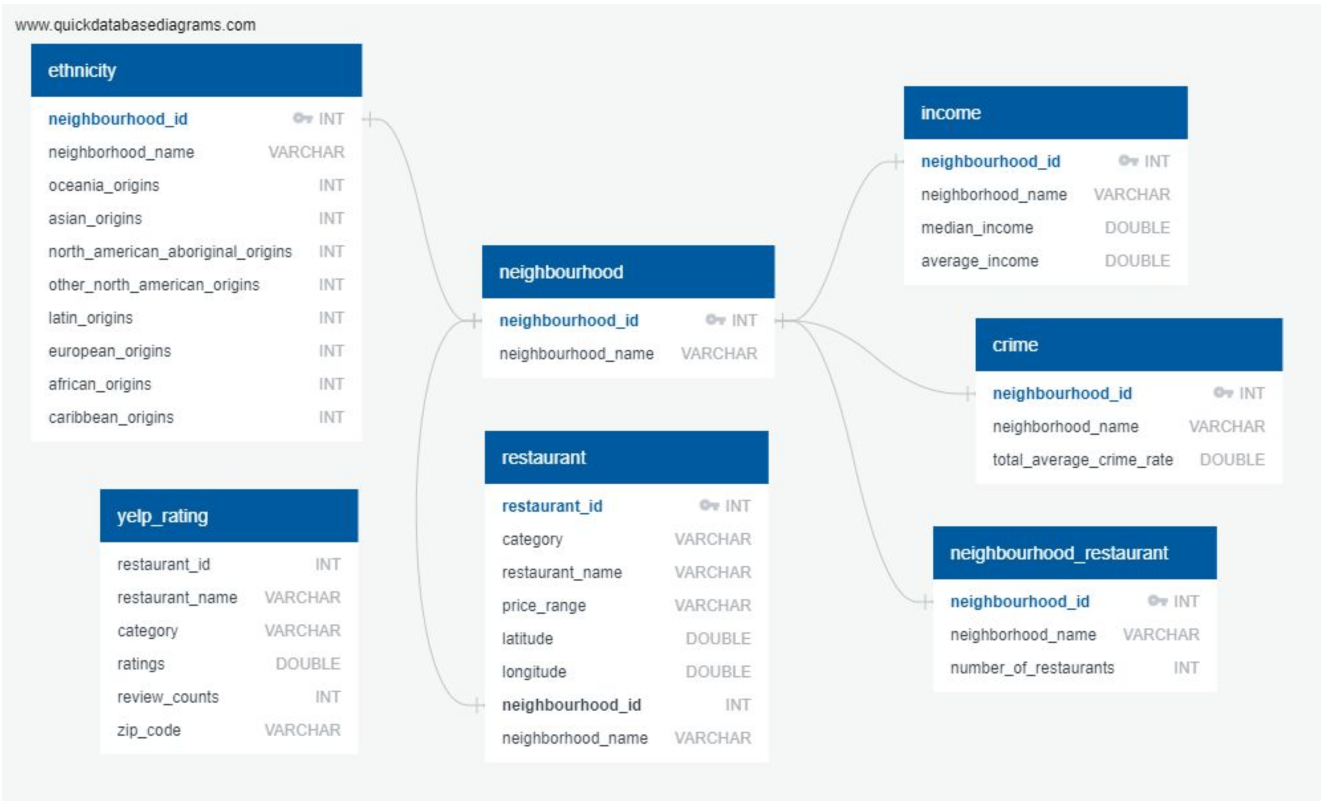
Introduction

Where to open a new restaurant? Choosing a new restaurant location is the most important, at the same time, the most difficult decision throughout the whole process. This project is to look at restaurant data in Toronto as well as ethnicity demographics, neighbourhood average income/crime. The goal of this new layer of analysis will be to help new restaurant owners decide as to where the best placement of a new restaurant could be. Our analysis will consider ethnicity, local competition, income and crime per neighborhood to help determine whether a restaurant could potentially be profitable or not each neighborhood.

Table of Contents

1. Data Modeling

ERD (Entity Relationship Diagram) of the tables was sketched out using a tool called 'Quick DBD.'



2. Data Extraction / Data Transformation

2.1 Datasets Sources:

In this project we extracted, transformed, and loaded these datasets:

- Toronto Neighbourhood Data - Toronto City Open Data
- Toronto Neighbourhood Income - Toronto City Open Data
- Toronto Crime Data - Toronto City Open Data
- Toronto Ethnicity Data - Toronto City Open Data Json API
- Toronto Restaurant Data - Kaggle
- Restaurant Ratings & number of reviews - Yelp API

The extraction of these datasets were completed using two methods; CSV's and API's. It was important to choose datasets from reliable sources such as Kaggle, 'Yelp API', 'Toronto Police API portal' and the 'City of Toronto Open API'. These three sources were all credible and trustworthy sources of data for our analysis. The primary key to almost every dataset was the neighborhood. Since our analysis primarily focuses on the behaviour of certain variables (crime, ethnicity, income, etc..) it was imperative that every dataset we sourced was in some way linkable to a neighborhood. The methods in which we chose to link to a neighborhood will be further discussed in detail below.

2.2 Toronto Restaurant Data

Extract

The restaurant data that we sourced from Kaggle came in a CSV format. From the screenshot below, we can see that we get all of the data we need for this analysis except for the neighborhood in which the restaurants belong to.

```
# read resources
restaurant_data = pd.read_csv("Resources/Restaurant_Data.csv")

# convert to dataframe
restaurant_df = pd.DataFrame(restaurant_data)
restaurant_df.head()
```

	Category	Restaurant Address	Restaurant Name	Restaurant Phone	Restaurant Price Range	Restaurant Website	Restaurant Yelp URL	Restaurant Latitude	Restaurant Longitude
0	Afghan	14 Prince Arthur Avenue\r\nToronto, ON M5R 1A9	The Host	(416) 962-4678	$11-30	welcometohost.com	https://www.yelp.ca /adredir?ad_business_id=OFA...	43.669935	-79.395858
1	Afghan	259 Wellington St W\r\nToronto, ON M5V	Aanch Modernist Indian Cuisine	(647) 558-1508	$11-30	aanch.ca	https://www.yelp.ca /adredir?ad_business_id=SZu...	43.644708	-79.390670

Transform

Initial data cleanup consisted of removing any unwanted columns and removing NaN rows.

```python
# Remove unwated columns in restaurant df
cleaned_restaurant_df = restaurant_df[["Category", "Restaurant Name", "Restaurant
cleaned_restaurant_df.reset_index(inplace=True)
cleaned_restaurant_df.drop('index', axis='columns', inplace=True)
cleaned_restaurant_df.head()
```

	Category	Restaurant Name	Restaurant Price Range	Restaurant Latitude	Restaurant Longitude
0	Afghan	The Host	$11-30	43.669935	-79.395858
1	Afghan	Aanch Modernist Indian Cuisine	$11-30	43.644708	-79.390670
2	Afghan	Silk Road Kabob House	Under $10	43.659816	-79.385591
3	Afghan	Naan & Kabob	$11-30	43.669058	-79.386100
4	Afghan	Afghan Cuisine	$11-30	43.708070	-79.341508

In order for us to connect this dataset to our main database, we need to identify which neighborhood each restaurant belongs to. This issue can be solved using GeoPandas. From the screenshots below, we load shapes file into Pandas so that we can get the borders of each neighborhood based on lat/long.

```python
filepath = os.path.join("Resources", "data.csv")
data = pd.read_csv(filepath)
nb = os.path.join('Resources', 'Neighbourhoods', "Neighbourhoods.shp")
regions = gpd.read_file(nb)
regions['neighbourhood'] = regions['FIELD_7'].str.replace(' \(.+\)', '').str.lower()
regions.to_csv('Resources/regions.csv')
regions.head()
```

	FIELD_1	FIELD_2	FIELD_3	FIELD_4	FIELD_5	FIELD_6	FIELD_7	FIELD_8	FIELD_9	FIELD_10	FIELD_11	FIELD_12	FIELD_13	FIELD_14	FIELI
0	2101	25886861	25926662	49885	94	94	Wychwood (94)	Wychwood (94)	None	None	-79.425515	43.676919	16491505	3.217960e+06	7515.77
1	2102	25886820	25926663	49885	100	100	Yonge-Eglinton (100)	Yonge-Eglinton (100)	None	None	-79.403590	43.704689	16491521	3.160334e+06	7872.02
2	2103	25886834	25926664	49885	97	97	Yonge-St.Clair (97)	Yonge-St.Clair (97)	None	None	-79.397871	43.687859	16491537	2.222464e+06	8130.41

We then create two new columns (Neighbourhood ID and Name) and link each restaurant based on lat/long with the above Geopandas table.

```python
# Add new columns, 'neighbourhood_id' and 'neighbourhood_name'
cleaned_restaurant_df['neighbourhood_id'] = 'NaN'
cleaned_restaurant_df['neighbourhood_name'] = 'NaN'

# Assign each restaurant to corresponded neighbourhood polygon using GeoPandas & neighbourhood geometry
for i in range(len(cleaned_restaurant_df)):

    lng = cleaned_restaurant_df.loc[i, 'Restaurant Longitude']
    lat = cleaned_restaurant_df.loc[i, 'Restaurant Latitude']
    point = Point(lng, lat)

    for j in np.arange(len(regions)):
        poly = regions.loc[j, 'geometry']

        if point.within(poly):
            cleaned_restaurant_df.loc[i, 'neighbourhood_id'] = regions.loc[j, 'FIELD_6']
            cleaned_restaurant_df.loc[i, 'neighbourhood_name'] = regions.loc[j, 'neighbourhood']

# Remove the restaurants which were not assigned to any Toronto neighbourhood polygon
cleaned_restaurant_df = cleaned_restaurant_df[cleaned_restaurant_df['neighbourhood_id'] != 'NaN']
cleaned_restaurant_df.head()
```

	Category	Restaurant Name	Restaurant Price Range	Restaurant Latitude	Restaurant Longitude	neighbourhood_id	neighbourhood_name
0	Afghan	The Host	$11-30	43.669935	-79.395858	95	annex
1	Afghan	Aanch Modernist Indian Cuisine	$11-30	43.644708	-79.390670	77	waterfront communities-the island
2	Afghan	Silk Road Kabob House	Under $10	43.659816	-79.385591	76	bay street corridor
3	Afghan	Naan & Kabob	$11-30	43.669058	-79.386100	75	church-yonge corridor
4	Afghan	Afghan Cuisine	$11-30	43.708070	-79.341508	55	thorncliffe park

Many of the duplicate values included a restaurant having more than one category, or multiple locations but same name. We dropped the duplicated that included having multiple categories.

```python
restaurant_df = cleaned_restaurant_df.drop_duplicates(subset=['Restaurant Name', 'neighbourhood_id'], keep='first')
restaurant_df.reset_index(inplace=True)
restaurant_df
```

	Category	Restaurant Name	Restaurant Price Range	Restaurant Latitude	Restaurant Longitude	neighbourhood_id	neighbourhood_name
0	Afghan	The Host	$11-30	43.669935	-79.395858	95	annex
1	Afghan	Aanch Modernist Indian Cuisine	$11-30	43.644708	-79.390670	77	waterfront communities-the island
2	Afghan	Silk Road Kabob House	Under $10	43.659816	-79.385591	76	bay street corridor

Finally, the total number of restaurants in each neighborhood were aggregated to come up with a cleaned list of the total number of restaurants per neighborhood.

```python
# number of restaurants for each neighbourhood
neighbourhood_restaurant = restaurant_df.groupby(['neighbourhood_id','neighbourhood_name'])['Restaurant Name'].count()
neighbourhood_restaurant.sort_values(ascending=False, inplace=True)
neighbourhood_restaurant = neighbourhood_restaurant.reset_index()
neighbourhood_restaurant.set_index('neighbourhood_id', inplace=True)
neighbourhood_restaurant.rename(columns={'Restaurant Name':'Number of Restaurants'}, inplace=True)
neighbourhood_restaurant
```

	neighbourhood_name	Number of Restaurants
neighbourhood_id		
76	bay street corridor	355
77	waterfront communities-the island	354
78	kensington-chinatown	280

2.3 Toronto Neighbourhood Data

Toronto Neighbourhood Data was the base of other tables as the objective was to look at the difference between neighbourhoods. The final neighbourhood table has only two columns: 'neighbourhood_id' and the other is 'neighbourhood_name', and both can be primary keys as the values are unique.

Extract

The Toronto Neighbourhood dataset was found in Toronto City Open Data. The raw dataset was a csv file and it has many columns including 'AREA_SHORT_CODE', 'AREA_NAME', 'LATITUDE', 'LONGITUDE', 'geometry' and so on. This dataset was imported by using Pandas library.

Transform

Firstly, the neighbourhood names had to be edited using Python string method replace(), as the raw data neighbourhood name had unrequired strings. For example, the raw data had a neighbourhood name, Yonge-Eglinton (100), from here '(100)' was not required. With the code below, 'neighbourhood_name' column was added to the dataframe.

```
# neighbourhood_name
neighbourhood['neighbourhood_name'] = neighbourhood['AREA_NAME'].str.replace(' \(.+\)', '')
```

	neighbourhood_name
neighbourhood_id	
1	West Humber-Clairville
2	Mount Olive-Silverstone-Jamestown
3	Thistletown-Beaumond Heights
4	Rexdale-Kipling
5	Elms-Old Rexdale
...	...
136	West Hill
137	Woburn
138	Eglinton East
139	Scarborough Village
140	Guildwood

140 rows × 1 columns

After that, the unwanted columns were dropped, renamed some columns based on our ERD, and finally set 'neighbourhood_id' as an index of the dataframe.

2.4 Toronto Ethnicity Data

Toronto Neighborhood profiles is a free open source data which is available to the public in many different file formats (CSV, XML and JSON). Using this data, our goal here was to extract the information about the ethnicity distribution by neighborhood in Toronto. The steps that were taken to achieve this goal will be explained in more details below.

Extract

<u>Step1</u>

As response to the API call, the ethnicity data from Toronto Neighborhood profiles was received in a JSON format. Toronto Open Data website provided sample code snippets on how to use the API to access the dataset. However, this sample code only fetched a maximum of 100 records (regardless of how many actual records existed) and this meant that if the result (from the API call) had more than 100 records, it would only fetch 100 records and the rest of the record would be disregarded. It was not the plausible scenario for our case because the result we get from the API call, would contain more than 100 records. In order to address this issue, the pagination had to be implemented. How this pagination works is by, using the while loop. First, the API call was made to try to fetch the first 100 records. If the records and the next page existed, while loop kicked in, and then made a call to the API again to fetch the next 100 records in the next page. In each iteration, the result was converted to the DataFrame and then appended to a list. The iteration was continued until there were no more records to fetch. At this point, it broke out of the while loop and stopped calling the API. And then, multiple DataFrames in the list were concatenated into a single DataFrame containing the total ethnicity data from Toronto Neighborhood profiles. To demonstrate how this was done, the actual code is shown below.

```
url = "https://ckan0.cf.opendata.inter.prod-toronto.ca/api/3/action/package_show"
params = { "id": "6e19a90f-971c-46b3-852c-0c48c436d1fc"}
package = requests.get(url, params = params).json()
print(package["result"])
```

```python
# Final Solution
offset = 0
total_record = 0
combined_dataframes = []

for resource in (package["result"]["resources"]):
    if resource["datastore_active"]:
        url = f'https://ckan0.cf.opendata.inter.prod-toronto.ca/api/3/action/datastore_search?\
        id={resource["id"]}&offset=0'
        while True:
            data = requests.get(url).json()
            next_page = data['result']['_links']['next']
            records = data['result']['records']
            if next_page and records:
                dataframe = pd.DataFrame(records)
                combined_dataframes.append(dataframe)
                url = f'https://ckan0.cf.opendata.inter.prod-toronto.ca{next_page}'
            else:
                break
    break

result = pd.concat(combined_dataframes).sort_index()
result.to_csv('ethnicity.csv')
```

Step2

Toronto Neighborhood profiles provide a portrait of the demographic, social and economic characteristics of the people and households in each City of Toronto neighborhood. From this data, we extracted the information about the ethnicity distribution (Ethnic origin and Ethnic origin population) in Toronto by using pandas.

```python
ethnicity_df = result.loc[((result["Category"] == "Ethnic origin") & \
                           (result["Topic"] == "Ethnic origin population"))]
```

Transform

Step3

The only 8 ethnic origins representing each ethnicity (Aboriginal, North American, European, Caribbean, Latins, African, Asian, Oceanian) were selected. Columns and rows are transposed, and the column called ‘characteristic’ is renamed to ‘neighbourhood_name’ which makes it easier to be merged with other data. Also, the columns which are not necessary for this project are eliminated.

```
characteristic_df = ethnicity_df.loc[((ethnicity_df["Characteristic"] == " North American Aboriginal origins") \
                                     | (ethnicity_df["Characteristic"] == " Other North American origins") \
                                     | (ethnicity_df["Characteristic"] == " European origins") \
                                     | (ethnicity_df["Characteristic"] == " Caribbean origins") \
                                     | (ethnicity_df["Characteristic"] == " Latin; Central and South American origins") \
                                     | (ethnicity_df["Characteristic"] == " African origins") \
                                     | (ethnicity_df["Characteristic"] == " Asian origins") \
                                     | (ethnicity_df["Characteristic"] == " Oceania origins"))]
```

```
for col in characteristic_df.columns:
    if col == '_id' or col == 'Category' or col == 'Topic' or col == 'Data Source':
        del characteristic_df[col]
```

```
transposed_df = characteristic_df.loc[:,'Characteristic':'Yorkdale-Glen Park']
ethnicity = transposed_df.transpose().reset_index()
header_row = 0
ethnicity.columns = ethnicity.iloc[header_row]
ethnicity = ethnicity.rename(columns={"Characteristic" : "neighbourhood_name"})
ethnicity.set_index('neighbourhood_name', inplace=True)
# ethnicity = ethnicity.drop('Characteristic')
ethnicity = ethnicity.drop(['Characteristic', 'City of Toronto'])
```

Step 4

Finally, ‘neighbourhood_id’ columns were added up and set as index by merging with the crime csv file.

```
# Create neighbourhood_id by merging
crime_csv = '../clean_data/crime.csv'
crime = pd.read_csv(crime_csv)
crime = crime.loc[:, ['neighbourhood_id', 'neighbourhood_name']]
```

```
ethnicity = pd.merge(ethnicity, crime, on = 'neighbourhood_name')
ethnicity = ethnicity.set_index('neighbourhood_id')
del ethnicity['neighbourhood_id_y']
```

neighbourhood_id	neighbourhood_name	Oceania origins	Asian origins	North American Aboriginal origins	Other North American origins	Latin; Central and South American origins	European origins	African origins	Caribbean origins
129	Agincourt North	10	24,305	40	1,345	470	3,055	535	1,445
128	Agincourt South-Malvern West	0	17,955	105	1,190	480	3,770	625	1,395
20	Alderwood	0	2,055	305	2,355	315	9,135	215	350
95	Annex	140	6,485	475	5,255	765	21,055	1,040	750
42	Banbury-Don Mills	20	12,025	230	3,230	585	13,435	990	815
...	...	...	...	...	...	...	...	...	...
94	Wychwood	90	2,500	335	2,010	645	9,685	610	740
100	Yonge-Eglinton	50	2,895	140	2,695	370	8,455	310	280
97	Yonge-St.Clair	80	2,330	215	2,525	300	9,460	370	295
27	York University Heights	20	12,550	220	2,045	2,055	8,735	2,450	3,345
31	Yorkdale-Glen Park	10	4,090	105	1,040	1,025	7,820	820	935

136 rows × 9 columns

2.5 Toronto Crime Data

Neighbourhood Crime Rates dataset from Toronto City Open Data includes the 2014-2019 Crime Data by Neighbourhood. Counts are available for Assault, Auto Theft, Break and Enter, Robbery, Theft Over and Homicide. Data also includes five-year averages and crime rates per 100,000 people by neighbourhood based on 2016 Census Population. Using this data, our goal was to extract the information about the crime rate by neighborhood in Toronto. The steps will be explained in more details below.

Extract

Step1

Neighbourhood Crime Rates dataset came in CSV format from Toronto City Open Data Website.

```
crime_csv = "../Resources/Neighbourhood_Crime_Rates.csv"
crime_df = pd.read_csv(crime_csv)
crime_df.head()
```

Transform

<u>Step2</u>

All unnecessary columns were eliminated since we only needed the 2019 crime data. Total average crime rates were calculated which is the total crime rate of crimes for 2019 per 100,00 population including Assault, Auto Theft, Break and Enter, Robbery, Theft Over and Homicide rates. Also, we created a new column for the values. Rest of the columns were removed and renamed until the data only has 'Neighbourhood Id', 'Neighbourhood Name' and 'Average Crime Rate' for columns. The result table and codes are down below.

```
# Remove unwanted columns
crime_df = crime_df.loc[:,["Neighbourhood", "Hood_ID", "Assault_Rate_2019", "AutoTheft_Rate_2019",\
                           "BreakandEnter_Rate_2019", "Homicide_Rate_2019", "Robbery_Rate_2019", \
                           "TheftOver_Rate_2019"]]

# Calculate total averate crime rates and create a new column for the values
# Crime Rates: Rate of crimes for 2019 per 100,000 population
crime_df["total_average_crime_rate"] = round((crime_df["Assault_Rate_2019"] + \
                                              crime_df["AutoTheft_Rate_2019"] + \
                                              crime_df["Homicide_Rate_2019"] + \
                                              crime_df["Robbery_Rate_2019"] + \
                                              crime_df["TheftOver_Rate_2019"]) / 6, 2)

# Remove unwanted columns
crime = crime_df.loc[:,["Neighbourhood", "Hood_ID", "total_average_crime_rate"]]


# Rename the columns names
crime = crime.rename(columns={"Hood_ID" : "neighbourhood_id",
                              "Neighbourhood" : "neighbourhood_name"
                             })

# Set index as neighbourhood_id
crime = crime.set_index('neighbourhood_id')
crime_rate = crime.sort_index()
```

neighbourhood_id	neighbourhood_name	total_average_crime_rate
1	West Humber-Clairville	507.32
2	Mount Olive-Silverstone-Jamestown	232.13
3	Thistletown-Beaumond Heights	236.50
4	Rexdale-Kipling	245.35
5	Elms-Old Rexdale	216.80
...	...	...
136	West Hill	383.95
137	Woburn	206.32
138	Eglinton East	231.97
139	Scarborough Village	262.10
140	Guildwood	84.03

140 rows × 2 columns

2.6 Restaurants Rating & Review Data from Yelp API

Yelp is a crowd-sourced local business review site where users can get lots of information about local restaurants, bars, cafes, etc. For the competition analysis, we wanted to look at which restaurants are popular to people, ratings, and the number of reviews information on restaurants on Yelp could be very valuable. Fortunately, Yelp allows the users to use API to get restaurant information on their site, and on Python, there is a library called 'YelpAPI', which makes it easier to use those API to get wanted information. Each step will be explained in detail.

Extract

Step 1

In order to use Yelp API, API key will be required from Yelp API site. You will need to use *'from yelpapi import YelpAPI'* to import YelpAPI library.

Step 2

Toronto neighbourhood information with latitude and longitude was required as well, since Yelp allowed us to query based on latitude and longitude. 200-245 restaurants per neighbourhood were queried. The number of Toronto neighbourhoods is 140. By using each neighbourhood's latitude and longitude, and for loop, we were able to get restaurant information about name, category, rating, the number of reviews, zip code, and hold these information in lists. You can see the code in the screenshot below:

```python
# Create a set of neighbourhood lat and lng combinations
lng_lats = []

lngs = neighbourhood['LONGITUDE']
lats = neighbourhood['LATITUDE']

lng_lats = zip(lngs, lats)


offset = 1
limit = 49
total_num_queries = 5

ids = []
names = []
categories = []
ratings = []
review_counts = []
zip_code = []


for lng_lats in zip(lngs, lats):

    for i in range(total_num_queries):

        response = yelp_api.search_query(latitude=lng_lats[1], longitude=lng_lats[0], radius=5000, limit=limit, offset=offset)

        for business in range(len(response['businesses'])):

            try:
                ids.append(response['businesses'][business]['id'])
                names.append(response['businesses'][business]['name'])
                categories.append(response['businesses'][business]['categories'][0]['title'])
                ratings.append(response['businesses'][business]['rating'])
                review_counts.append(response['businesses'][business]['review_count'])
                zip_code.append(response['businesses'][business]['location']['zip_code'])

            except:
                pass

        offset = offset + limit

    offset = 1
```

Transform

Step 3

With the lists created in Step 2, a dataframe was created by using Pandas. The duplicated rows were dropped, and set the 'restaurant_id' column as an index. There were no null values in the dataframe; removing null values was not needed.

restaurant_id	name	category	ratings	review_counts	zip_code
e41TP5cXZqSrz50xCBJqZw	Insomnia Restaurant & Lounge	Lounges	4.0	923	M5S 1Y6
r_BrlgzYcwo1NAuG9dLbpg	Pai Northern Thai Kitchen	Thai	4.5	2895	M5H 3G8
Uq-GOs9_lqweUsB5MdIl9w	Emma's Country Kitchen	Breakfast & Brunch	4.0	394	M6C 1B6
iGEvDk6hsizigmXhDKs2Vg	Seven Lives Tacos y Mariscos	Mexican	4.5	1323	M5T 2K1
-lCGmF2qUVKdvOehVNgPbg	Lamesa Filipino Kitchen	Filipino	4.0	352	M6C 1A9
...	...	...	...	...	...
RNdcUG1sCTLdUo8dEC9NJw	The Local	Bars	3.5	58	M6R 2M9
kbSSGo6zRPSdBT-CwG2cNg	Suvaiyakam Restaurant	Sri Lankan	4.5	11	M1W 3G5

2.7 Toronto Income Data

Toronto income dataset from Toronto City Open Data includes the Neighbourhood Profiles which provide a portrait of the demographic, social and economic characteristics of the people and households in each City of Toronto neighbourhood. The data is based on tabulations of 2016 Census of Population data from Statistics Canada. Using this data, we were extracting the information about household income by neighbourhood in Toronto.

Extract

Step 1

Toronto household income dataset came in CSV format from Toronto City Open Data Website.

```
income_df = pd.read_excel('../Resources/neighbourhood-income-data-2011.xlsx')
income_df.head()
```

Transform

Step 2

‘Income of households’ in the income category is selected as needed. Also, median and average of household total income is calculated for further analysis and all cities columns were transposed to one single column. Finally, income data and neighbourhood data was merged on neighbourhood name, and neighbourhood id was set as index. The result table and codes are down below.

```
income_df = income_df.loc[(income_df["Category"] == 'Income') & \
                          (income_df["Topic"]  == 'Income of households') & \
                          ((income_df["Attribute"] == 'Median household total income $')| \
                           (income_df["Attribute"]  == 'Average household total income $'))]
```

```
# Did transpose of all cities columns to 1 single column

transposed_df = income_df.loc[:,'Agincourt North':'Yorkdale-Glen Park']
income_data = transposed_df.transpose().reset_index()
income_data.columns = ["neighbourhood_name","median_income","average_income"]
```

```
neighbourhood = pd.read_csv('../Resources/Neighbourhoods.csv')

# For merge the DataFrames, add the neighbourhood_column
neighbourhood['neighbourhood_name'] = neighbourhood['AREA_NAME'].str.replace(' \(.+\)', '')
income_data['neighbourhood_name'] = income_data['neighbourhood_name'].str.replace(' \(.+\)', '')
```

```
income_neighbourhood = pd.merge(income_data, neighbourhood,  on='neighbourhood_name')
income = income_neighbourhood[['AREA_SHORT_CODE', 'neighbourhood_name', 'median_income', \
                               'average_income']]
income = income.rename(columns={'AREA_SHORT_CODE':'neighbourhood_id'})
income = income.set_index('neighbourhood_id')
```

	neighbourhood_name	median_income	average_income
neighbourhood_id			
1	West Humber-Clairville	66241.0	76228.0
2	Mount Olive-Silverstone-Jamestown	49934.0	58605.0
3	Thistletown-Beaumond Heights	62042.0	73512.0
4	Rexdale-Kipling	56545.0	66781.0
5	Elms-Old Rexdale	50846.0	63201.0
...	...	...	...
136	West Hill	49713.0	63461.0
137	Woburn	52018.0	63651.0
138	Eglinton East	46495.0	58035.0
139	Scarborough Village	42131.0	62141.0
140	Guildwood	76055.0	96885.0

140 rows × 3 columns

3. Data Load to SQL Database

3.1 Load onto PostgreSQL

Step 1

In pgAdmin4, create a table schema for each of the 7 transformed dataframe using the information and specified data types, primary keys, foreign keys, and other constraints.

Link to DB-Schema-Tables Creation:

[Neighbourhoods-DB-Schema-Tables-Creation.sql]

Step 2

Import the module sqlalchemy and create an engine with the parameters user, password, and database name to connect and log in to the PostgreSQL database.

Create database connection to Neighborhoods_DB

```
# Establish Connection to neighborhood database
engine = create_engine(f'postgresql://{username}:{password}@localhost:5432/Neighborhoods_DB')
conn = engine.connect()
```

```
# Confirm tables
engine.table_names()
```

```
['yelp_ratings',
 'restaurant',
 'income',
 'crime',
 'neighbourhood_restaurant',
 'neighbourhood',
 'ethnicity']
```

Step 3

Load Final Transformed data into the tables using the to_sql() function with the parameters table name, engine name, if_exists, and index.

This approach accomplishes data loading in a more direct way, and allows us to add a whole dataframe to a PostgreSQL database all at once.

Load DataFrames into database

```
neighbourhood_transformed.to_sql(name='neighbourhood', con=engine, if_exists='append', index=True)
```

```
income_transformed.to_sql(name='income', con=engine, if_exists='append', index=True)
```

```
crime_transformed.to_sql(name='crime', con=engine, if_exists='append', index=True)
```

```
ethnicity_transformed.to_sql(name='ethnicity', con=engine, if_exists='append', index=True)
```

```
restaurant_transformed.to_sql(name='restaurant', con=engine, if_exists='append', index=True)
```

```
neighbourhood_restaurant_transformed.to_sql(name='neighbourhood_restaurant', con=engine, if_exists='append', index=True)
```

```
yelp_rating_transformed.to_sql(name='yelp_ratings', con=engine, if_exists='append', index=True)
```

Checked if the tables are successfully populated with data in postgres - Neighbourhoods_DB using the SELECT command. It also shows the number of rows affected successfully.

```

-- Viewing the tables
Select * from neighbourhood;
Select * from income;
Select * from crime;
Select * from ethnicity;
Select * from restaurant;
Select * from neighbourhood_restaurant;
Select * from yelp_ratings;
```

Data Output Explain Messages Notifications

	restaurant_id [PK] integer	category character varying	restaurant_name character varying	price_range character varying	latitude real	longitude real	neighbourhood_id integer	neighbourhood_name character varying
1	1	Afghan	The Host	$11-30	43.669933	-79.39586	95	annex
2	2	Afghan	Aanch Modernist India...	$11-30	43.644707	-79.39067	77	waterfront communities-the ...
3	3	Afghan	Silk Road Kabob House	Under $10	43.659817	-79.38559	76	bay street corridor
4	4	Afghan	Naan & Kabob	$11-30	43.66906	-79.3861	75	church-yonge corridor
5	5	Afghan	Afghan Cuisine	$11-30	43.70807	-79.34151	55	thorncliffe park
6	6	Afghan	Pamier Kabob	$11-30	43.64721	-79.39547	77	waterfront communities-the ...
7	7	Afghan	Bamiyan Kabob	$11-30	43.77383	-79.49[illegible]	[illegible]	[illegible]
8	8	Afghan	Kandahar Kabab	$11-30	43.812546	-79.2[illegible]		

✓ Successfully run. Total query runtime: 368 msec. 5140 rows affected.

We further analysed the same set of data within Pandas using read_sql command for all the tables.

```
Neighbourhood_records = pd.read_sql("SELECT * FROM neighbourhood" , conn)
print (f'Data in Neighbourhood table\n-------------------------------------\n{Neighbourhood_records.head(10)}')
```

```
Data in Neighbourhood table
-------------------------------------
   neighbourhood_id                 neighbourhood_name
0                 1              West Humber-Clairville
1                 2  Mount Olive-Silverstone-Jamestown
2                 3       Thistletown-Beaumond Heights
3                 4                    Rexdale-Kipling
4                 5                   Elms-Old Rexdale
5                 6        Kingsview Village-The Westway
6                 7   Willowridge-Martingrove-Richview
7                 8             Humber Heights-Westmount
8                 9             Edenbridge-Humber Valley
9                10                  Princess-Rosethorn
```

3.2. Final Database

Confirm successful **Load** by querying database.

3.2.1 Restaurant Table

```sql
-- Viewing the tables
Select * from neighbourhood;
Select * from income;
Select * from crime;
Select * from ethnicity;
Select * from restaurant;
Select * from neighbourhood_restaurant;
Select * from yelp_ratings;
```

Data Output | Explain | Messages | Notifications

	restaurant_id [PK] integer	category character varying	restaurant_name character varying	price_range character varying	latitude real	longitude real	neighbourhood_id integer	neighbourhood_name character varying
1	1	Afghan	The Host	$11-30	43.669933	-79.39586	95	annex
2	2	Afghan	Aanch Modernist India...	$11-30	43.644707	-79.39067	77	waterfront communities-the ...
3	3	Afghan	Silk Road Kabob House	Under $10	43.659817	-79.38559	76	bay street corridor
4	4	Afghan	Naan & Kabob	$11-30	43.66906	-79.3861	75	church-yonge corridor
5	5	Afghan	Afghan Cuisine	$11-30	43.70807	-79.34151	55	thorncliffe park
6	6	Afghan	Pamier Kabob	$11-30	43.64721	-79.39547	77	waterfront communities-the ...

3.2.2 Neighbourhood_restaurant Table

```sql
Select * from neighbourhood_restaurant;
Select * from yelp_ratings;
```

Data Output | Explain | Messages | Notifications

	neighbourhood_id [PK] integer	neighbourhood_name character varying	number_of_restaurants integer
1	76	bay street corridor	355
2	77	waterfront communities-the ...	354
3	78	kensington-chinatown	280
4	75	church-yonge corridor	214
5	95	annex	180
6	81	trinity-bellwoods	136
7	14	islington-city centre west	120
8	70	south riverdale	106

3.2.3 Neighbourhood Table

```sql
-- Viewing the tables
Select * from neighbourhood;
Select * from income;
Select * from crime;
Select * from ethnicity;
Select * from restaurant;
Select * from neighbourhood_restauran
Select * from yelp_ratings;
```

Data Output | Explain | Messages | Notifications

	neighbourhood_id [PK] integer	neighbourhood_name character varying
1	1	West Humber-Clairville
2	2	Mount Olive-Silverstone-Ja...
3	3	Thistletown-Beaumond Heig...
4	4	Rexdale-Kipling
5	5	Elms-Old Rexdale
6	6	Kingsview Village-The West...

3.2.4 Income Table

```sql
-- Viewing the tables
Select * from neighbourhood;
Select * from income;
Select * from crime;
Select * from ethnicity;
Select * from restaurant;
Select * from neighbourhood_restaurant;
Select * from yelp_ratings;
```

Data Output | Explain | Messages | Notifications

	neighbourhood_id [PK] integer	neighbourhood_name character varying	median_income integer	average_income integer
1	1	West Humber-Clairville	66241	76228
2	2	Mount Olive-Silverstone-Ja...	49934	58605
3	3	Thistletown-Beaumond Heig...	62042	73512
4	4	Rexdale-Kipling	56545	66781
5	5	Elms-Old Rexdale	50846	63201
6	6	Kingsview Village-The West...	55454	71539

3.2.5 Crime Table

```sql
-- Viewing the tables
Select * from neighbourhood;
Select * from income;
Select * from crime;
Select * from ethnicity;
Select * from restaurant;
Select * from neighbourhood_restaurant;
Select * from yelp_ratings;
```

Data Output | Explain | Messages | Notifications

	neighbourhood_id [PK] integer	neighbourhood_name character varying	total_average_crime_rate double precision
1	1	West Humber-Clairville	507.32
2	2	Mount Olive-Silverstone-Ja...	232.13
3	3	Thistletown-Beaumond Heig...	236.5
4	4	Rexdale-Kipling	245.35
5	5	Elms-Old Rexdale	216.8

3.2.6 Ethnicity Table

```sql
Select * from ethnicity;
Select * from restaurant;
Select * from neighbourhood_restaurant;
Select * from yelp_ratings;
```

Data Output | Explain | Messages | Notifications

	neighbourhood_id [PK] integer	neighbourhood_name character varying	oceania_origins integer	asian_origins integer	north_american_a integer	other_north_americar integer	latin_origins_. integer	european_origins integer	african_origins. integer	caribbean_origins. integer
1	129	Agincourt North	10	24305	40	1345	470	3055	535	1445
2	128	Agincourt South-Malvern W...	0	17955	105	1190	480	3770	625	1395
3	20	Alderwood	0	2055	305	2355	315	9135	215	350
4	95	Annex	140	6485	475	5255	765	21055	1040	750
5	42	Banbury-Don Mills	20	12025	230	3230	585	13435	990	815

3.2.7 Yelp_Ratings Table

```sql
Select * from neighbourhood_restaurant;
Select * from yelp_ratings;
```

Data Output | Explain | Messages | Notifications

	restaurant_id [PK] character varying	restaurant_name character varying	category character varying	ratings double precision	review_counts integer	zip_code character varying
1	e41TP5cXZqSrz50xCBJqZw	Insomnia Restaurant & ...	Lounges	4	923	M5S 1Y6
2	r_BrlgzYcwo1NAuG9dLbpg	Pai Northern Thai Kitch...	Thai	4.5	2895	M5H 3G8
3	Uq-GOs9_lqweUsB5MdlI9w	Emma's Country Kitchen	Breakfast & Brunch	4	394	M6C 1B6
4	iGEvDk6hsizigmXhDKs2Vg	Seven Lives Tacos y M...	Mexican	4.5	1323	M5T 2K1
5	-ICGmF2qUVKdvOehVNgPbg	Lamesa Filipino Kitchen	Filipino	4	352	M6C 1A9

3.3 Join Tables

3.3.1 Join crime, income, ethnicity, and neighbourhood_restaurant Tables on 'neighbourhood_id'

Query-joining-Income-Crime-Ethnicity-Restaurant.sql

```
--Join the income crime ethnicity view and the neighbourhood_restaurant table -- 133 records

SELECT ice.neighbourhood_id, ice.neighbourhood_name, r.number_of_restaurants, ice.oceania_origins, ice.asian_origins,
ice.north_american_aboriginal_origins,ice.other_north_american_origins, ice.latin_origins,
ice.european_origins,ice.african_origins,ice.caribbean_origins
from income_crime_ethnicity_data AS ice inner join
neighbourhood_restaurant AS r on ice.neighbourhood_id = r.neighbourhood_id
ORDER BY r.number_of_restaurants DESC
```

ta Output Explain Messages Notifications

neighbourhood_id integer	neighbourhood_name character varying	number_of_restaurants integer	oceania_origins integer	asian_origins integer	north_american_aboriginal_origins integer
76	Bay Street Corridor	355	60	15040	250
77	Waterfront Communities-Th...	354	235	24810	965
78	Kensington-Chinatown	280	55	9140	265
75	Church-Yonge Corridor	214	125	10740	635
95	Annex	180	140	6485	475

3.3.2 Join restaurant table and yelp_ratings table on 'restaurant_name'

Query-joining-Restaurants-and-Yelp-Ratings.sql

Query Editor Query History

```
-- Join restaurant and yelp_ratings tables on restaruant_name
SELECT r.restaurant_name, r.category, r.price_range, y.ratings, y.review_counts, r.neighbourhood_id, r.neighbourhood_name
FROM restaurant AS r
INNER JOIN yelp_ratings AS y
ON r.restaurant_name = y.restaurant_name
ORDER BY y.ratings DESC, y.review_counts DESC;
```

Data Output Explain Messages Notifications

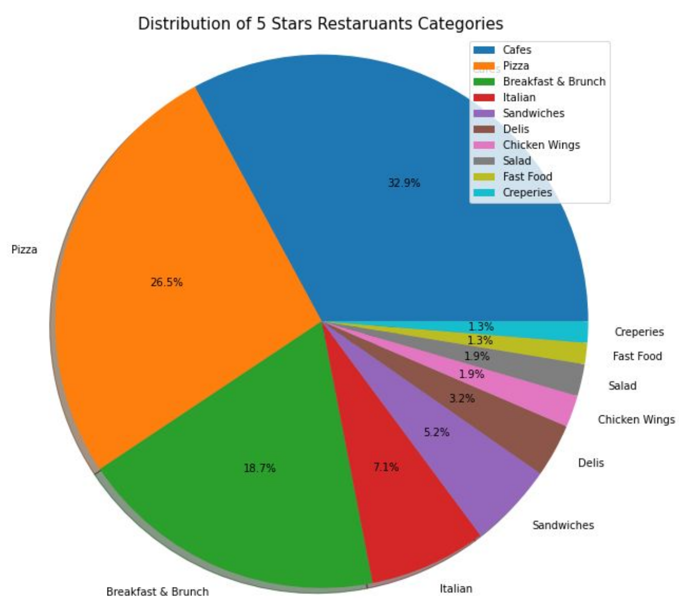
	restaurant_name character varying	category character varying	price_range character varying	ratings double precision	review_counts integer	neighbourhood_id integer	neighbourhood_name character varying
1	Baretto Caffe	Cafes	Under $10	5	326	42	banbury-don mills
2	Zeal Burgers	Burgers	$11-30	5	117	113	weston
3	Pomarosa Coffee Shop...	Cafes	$11-30	5	39	66	danforth

4. Sample Analysis

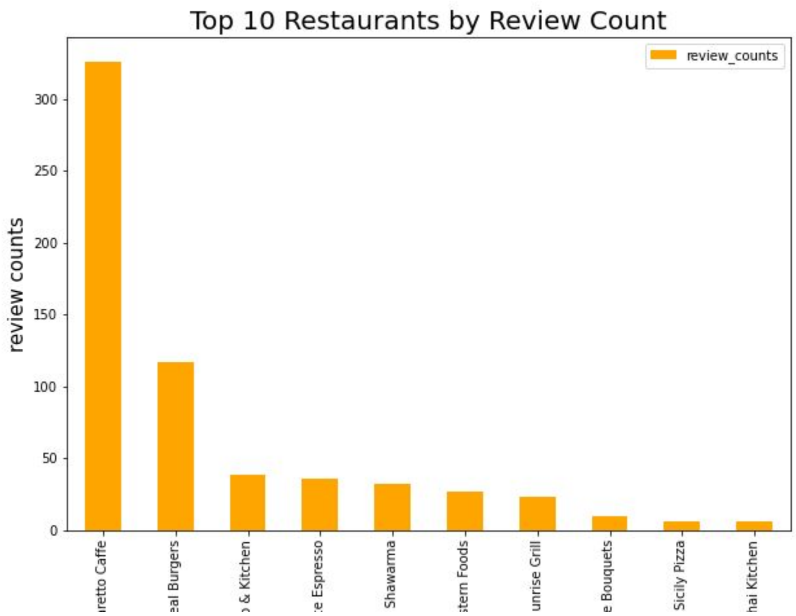
With the joined table, the bar plot and pie plot were created by using Pandas and matplotlib. We were able to look at the distribution of 5 stars restaurants categories in the pie chart. The table was filtered using the code 'restaurant_yelp_joined[restaurant_yelp_joined['ratings'] == 5.0]', grouped by 'category', and count() method, we were able to get the dataframe to plot the pie chart.

```
# Restaraunts have ratings of 5.0
rating_5 = restaurant_yelp_joined[restaurant_yelp_joined['ratings'] == 5.0]
rating_5_category = rating_5.groupby('category')['restaurant_name'].count()
rating_5_category = rating_5_category.to_frame()
rating_5_category.sort_values('restaurant_name', ascending = False, inplace=True)
rating_5_category = rating_5_category.rename(columns={'restaurant_name':'num_5_stars_restaurants'})
rating_5_category.head(5)
```

	num_5_stars_restaurants
category	
Cafes	51
Pizza	41
Breakfast & Brunch	29
Italian	11
Sandwiches	8



Also, the we sorted the joined table to get top 10 restaurants by review count, and created the bar chart:



5. Discussion

5.1 Limitations

Some of the limitations encountered in this project was the availability of lat/long data in many of the datasets. There were some incredibly interesting potentials for analysis from many of our sources such as date/time of police interaction. Lat/long data from the 'Toronto Police API' would have allowed us to pinpoint crime rates based on date and time **within each neighborhood**. This could allow for restaurant owners to make better decisions for opening/closing times, staffing volume during certain times of the day or even extra security measures during certain times of the day in order to minimize cost. As we searched for data sources, we found many other very interesting types of data that would have made a very positive impact on our analysis, however unfortunately many of these datasets do not contain information that we could link to a neighborhood.

5.2 Next Steps

Considering the limitations, we believe that we have consolidated a valuable set of data for the purpose of our analysis. Moving forward, we will be taking careful consideration for what types of data is available. We have learned that some data points are not as easy to collect as others such as geodata. However, with the onset of a new digital age where everyone has a phone, these limitations may not be around for long.

In the next part of the project, we will be analysing our data to answer the following questions:

- What percent of restaurants are getting more than 250 reviews?
- What percent of restaurants are getting a rating of more than 4?
- Which localities have higher footfall? (areas with higher review counts are likely to be in neighbourhoods where residents dine out more often)
- Localities with saturated restaurant market space - review count less than 250 & rating greater than 4 (worst areas to open Restaurants, since there are few people who dine out and neighbourhood has higher number of good restaurants already existing)
- Neighborhoods with high potential for restaurant opening(business review count greater than 300 and rating less than 4)
- Market segmentation based on ethnicity (most popular neighborhoods by ethnicity) - can create heat maps if time permits to showcase the most popular ethnicity in each neighborhood or horizontal bar graph.