Hello Ameer!

My name is Olga. I'm happy to review your project today.

When I will see mistake at the first time, I will just point it out. I let you find it and fix it by yourself. I'm trying to prepare you to work as an Data Analyst. SO, at a real job, your team lead will do the same. But if you can't handle this task yet, I will give you a more accurate hint at the next iteration.

You will find my comments below - **please do not move, modify or delete them**.

You can find my comments in green, yellow or red boxes like this:

**Reviewer's comment**

Success. Everything is done succesfully.

**Reviewer's comment**

Remarks. Some recommendations.

**Reviewer's comment**

Needs fixing. The block requires some corrections. Work can't be accepted with the red comments.

You can answer me by using this:

**Student answer.**

**Overallv reviewer's comment**

Hi Ameer! Thank you for sending your project.

It is awesome! I really like your excellent graphs, conclusions and clear code. I see you understand this sprint very good. I am really impressed! AlsoIwant to say that your presentation is perfect. I think investors will be subjugated.

I'm glad to say that your project has been accepted. Congratulations!

You hit the target. :)

# Introduction

We've decided to open a small robot-run cage in Los Angeles. The project is promising but expensive, therefore we have decided to try to attract investors. They are interested in the current market conditions, will we be able to maintain our success when the novelty of robot waiters wear off? As an expert in analytics, we decided to prepare a market research, using an open-source data on the restaurants in Los Angeles.

## Purpose of the research

The purpose of our research is to calculate our success chances and try to find ways and trends in the current market that will help us attract more investors and increase our chances for success.

**Reviewer's comment**

Great start with an introduction!

## Table of content

## 1 Initialization

### 1.1 Loading the libraries and data

We'll start by loading the necessary libraries and the open-source data we we're given:

In [1]:

```
# loading libraries that we may use
import pandas as pd
import plotly.express as px
from plotly import graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
!pip install usaddress
import usaddress
import warnings
warnings.filterwarnings("ignore")
```

```
Collecting usaddress
  Downloading usaddress-0.5.10-py2.py3-none-any.whl (63 kB)
     |████████████████████████████████| 63 kB 681 kB/s eta 0:00:011
Collecting future>=0.14
  Downloading future-0.18.2.tar.gz (829 kB)
     |████████████████████████████████| 829 kB 5.4 MB/s eta 0:00:01
Collecting python-crfsuite>=0.7
  Downloading python_crfsuite-0.9.8-cp39-cp39-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl (1.0 MB)
     |████████████████████████████████| 1.0 MB 45.9 MB/s eta 0:00:01
Collecting probableparsing
  Downloading probableparsing-0.0.1-py2.py3-none-any.whl (3.1 kB)
Building wheels for collected packages: future
  Building wheel for future (setup.py) ... done
  Created wheel for future: filename=future-0.18.2-py3-none-any.whl size=491
059 sha256=5837e19b9f11f391c29a46defb329b64de0e7ca0a26891062971d68b882ea6e7
  Stored in directory: /home/jovyan/.cache/pip/wheels/2f/a0/d3/4030d9f80e6b3
be787f19fc911b8e7aa462986a40ab1e4bb94
Successfully built future
Installing collected packages: python-crfsuite, probableparsing, future, usa
ddress
Successfully installed future-0.18.2 probableparsing-0.0.1 python-crfsuite-
0.9.8 usaddress-0.5.10
```

**Reviewer's comment**

Libraries at the beginning are good! :)

In [2]:

```
# loading the dataset
rest_data = pd.read_csv('/datasets/rest_data_us.csv')
rest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9651 entries, 0 to 9650
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   id           9651 non-null   int64 
 1   object_name  9651 non-null   object
 2   address      9651 non-null   object
 3   chain        9648 non-null   object
 4   object_type  9651 non-null   object
 5   number       9651 non-null   int64 
dtypes: int64(2), object(4)
memory usage: 452.5+ KB
```

## 1.2 Checking for problematic values

What we can immediately notice is a few missing values and probably some type changes that we'll need to perform. But before we start doing so, lets make sure that our column names are representative of their values:

In [3]:

```
rest_data.columns = ['id', 'rest_name', 'address', 'chain', 'rest_type', 'seats_number']
# according to the instructions, object_name and object_type are the establishment name and
```

Next we'll check the values of the columns 'chain' and 'rest_type' to see if we can convert them into a **category** type:

In [4]:

```
# Lets check the values of the columns, to see which we can convert into a category type
print(rest_data['chain'].value_counts())
print()
print(rest_data['rest_type'].value_counts())
```

```
False    5972
True     3676
Name: chain, dtype: int64

Restaurant    7255
Fast Food     1066
Cafe           435
Pizza          320
Bar            292
Bakery         283
Name: rest_type, dtype: int64
```

The **chain** column can be converted to boolean, the **rest_type** can be converted to **category** type:

In [5]:

```
# converting the chain and rest_type into boolean and category respectively.
rest_data['chain'] = rest_data['chain'].astype(bool)
rest_data['rest_type'] = rest_data['rest_type'].astype('category')
display(rest_data.info())
display(rest_data.sample(10)) # having a look at our DF
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9651 entries, 0 to 9650
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            9651 non-null   int64
 1   rest_name     9651 non-null   object
 2   address       9651 non-null   object
 3   chain         9651 non-null   bool
 4   rest_type     9651 non-null   category
 5   seats_number  9651 non-null   int64
dtypes: bool(1), category(1), int64(2), object(2)
memory usage: 320.8+ KB
```

```
None
```

| | id | rest_name | address | chain | rest_type | seats_number |
|---|---|---|---|---|---|---|
| **2541** | 14327 | JUDI'S DELI | 8631 W 3RD ST STE 305 | True | Restaurant | 49 |
| **4909** | 16695 | THE THREE CLUB | 1123 N VINE ST | False | Restaurant | 48 |
| **798** | 12584 | WET DOCUMENT | 3850 WILSHIRE BLVD #107 | False | Restaurant | 31 |
| **5005** | 16791 | GUATEPAN BAKERY | 982 W VERNON AVE | True | Bakery | 22 |
| **6049** | 17835 | KEIKO GRILL | 3650 MARTIN LUTHER KING JR BLVD STE 132 | False | Restaurant | 23 |
| **6195** | 17981 | SOUTHLAND BEER | 740 S WESTERN AVE STE 112 | False | Restaurant | 22 |
| **9005** | 20791 | KOBUNGA KOREAN GRILL | 929 W JEFFERSON BLVD | False | Restaurant | 23 |
| **2254** | 14040 | J - N - J BURGER SHACK | 5754 W ADAMS BLVD | False | Fast Food | 28 |
| **6310** | 18096 | IGUANA COFFEE BAR | 6320 HOLLYWOOD BLVD | False | Cafe | 17 |
| **7046** | 18832 | BKK101 THAI CUISINE | 11127 VENICE BLVD STE 10 | False | Restaurant | 16 |

**Now** that we're done with the minor updates to our columns, lets look for any duplicates or missing values:

In [6]:

```
# checking for dups
rest_data.duplicated().sum()
```

Out[6]:

```
0
```

In [7]:

```
# Checking for NA values
rest_data.isna().sum()
```

Out[7]:

```
id              0
rest_name       0
address         0
chain           0
rest_type       0
seats_number    0
dtype: int64
```

The initial check showed that we had 3 missing values in the **chain** column but now we have 0 missing values. So what exactly happened?

Well, once we converted the **chain** column from int to boolean, all NaNs were converted to **False**, leaving us with no missing values. We can back-track and remove these rows, or we can just keep them, they won't make much difference.

Lets perform another check, looking for zeros where zeros shouldn't be:

In [8]:

```
# Look for zeros in our dataframe:
for i in rest_data:
    print(i,len(rest_data[rest_data[i]==0]))
```

```
id 0
rest_name 0
address 0
chain 5972
rest_type 0
seats_number 0
```

The results look alright, we have no zeros other than in the chain column, in our case this just means **False**.

**Reviewer's comment**

Excellent preparation step! Let's start!

# 2 Data Analysis

## 2.1 Initial data analysis

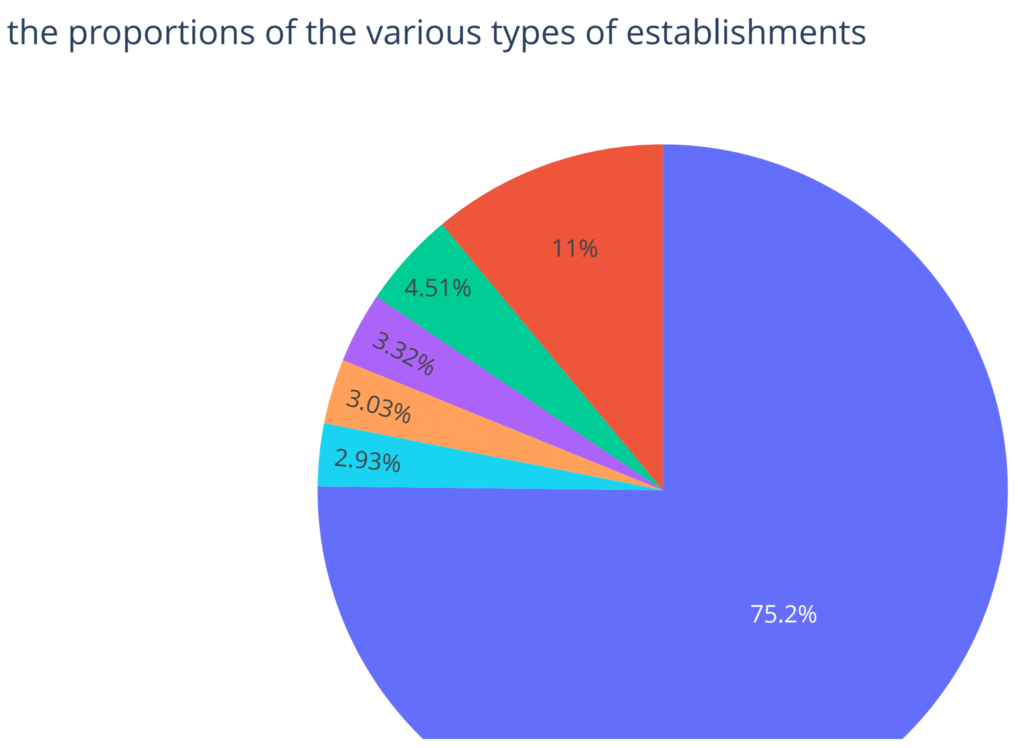
Now that we have a clean dataframe to work with, lets start our research!

We have the data in our hands, lets begin by drawing a graph describing the **proportions of the various types of establishments** to see how is the market divided:

In [9]:

```
rest_type_counts = rest_data.groupby('rest_type', as_index = False).agg({'id': 'count'})
rest_type_counts.columns = ['rest_type', 'count']
display(rest_type_counts.sort_values(by = 'count'))
px.pie(rest_type_counts, values='count',
title = 'the proportions of the various types of establishments', names='rest_type')
```

| | rest_type | count |
|---|---|---|
| 0 | Bakery | 283 |
| 1 | Bar | 292 |
| 4 | Pizza | 320 |
| 2 | Cafe | 435 |
| 3 | Fast Food | 1066 |
| 5 | Restaurant | 7255 |



What a surprise! Restaurants have a major lead among the establishment types, with 75.2% of the total establishment types. After it comes the **Fast Food** and then **Cafe**s.

1. Restaurants are the most popular establishment type.

**Reviewer's comment**

Restaurants are so popular!

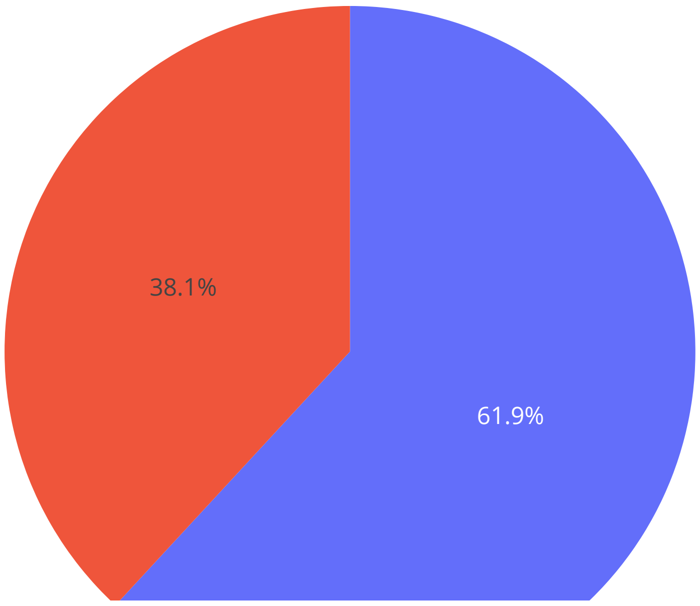
**Investigate the proportions of chain and nonchain establishments. Plot a graph** Lets move on and continue with our research, now that we know which establishment is the most popular among the other establishments, lets see the establishments that belongs to a chain and those that don't. Which one is more popular?

In [10]:

```
rest_chain_count = rest_data.groupby('chain', as_index = False).agg({'id': 'count'})
rest_chain_count.columns = ['chain', 'count']
display(rest_chain_count)
px.pie(rest_chain_count, values='count',
title = 'the proportions of chain and nonchain establishments', names='chain')
```

| | chain | count |
|---|---|---|
| **0** | False | 5972 |
| **1** | True | 3679 |

the proportions of chain and nonchain establishments



**Reviewer's comment**

Yes, most of our establishments are non-chain.

Establishments that don't belong to a **chain** are more than x1.5 of those that belong to a chain.

2. Establishments that don't belong to chain is more popular.

**Which type of establishment is typically a chain?** We've found out which establishments are the most popular, we've found out how much belong to a chain and how much doesn't. Now lets mix both of these parameters and find out the proportions of each establishment type that that belong to a chain:

In [11]:

```
# Lets find the percentage of each of the establishment that  belong to a chain
chain_rest_types_count = rest_data.query('chain == True').groupby('rest_type', as_index = F
chain_rest_types_count.columns = ['rest_type', 'count']
chain_rest_types_count['chain_no_chain_ratio'] = chain_rest_types_count['count'] / rest_typ
display(chain_rest_types_count.sort_values(by = 'chain_no_chain_ratio', ascending = False))
```

| | rest_type | count | chain_no_chain_ratio |
|---|---|---|---|
| **0** | Bakery | 283 | 1.000000 |
| **2** | Cafe | 266 | 0.611494 |
| **3** | Fast Food | 605 | 0.567542 |
| **4** | Pizza | 154 | 0.481250 |
| **5** | Restaurant | 2294 | 0.316196 |
| **1** | Bar | 77 | 0.263699 |

This is actually surprising, turns out that all **bakeries** are part of a bigger chain. Next comes **Cafe** with 61% of its establishments belong to a chain.

3. All bakeries belong to a chain

**Reviewer's comment**

Yes, little surprise for bakeries. :)

**Reviewer's comment**

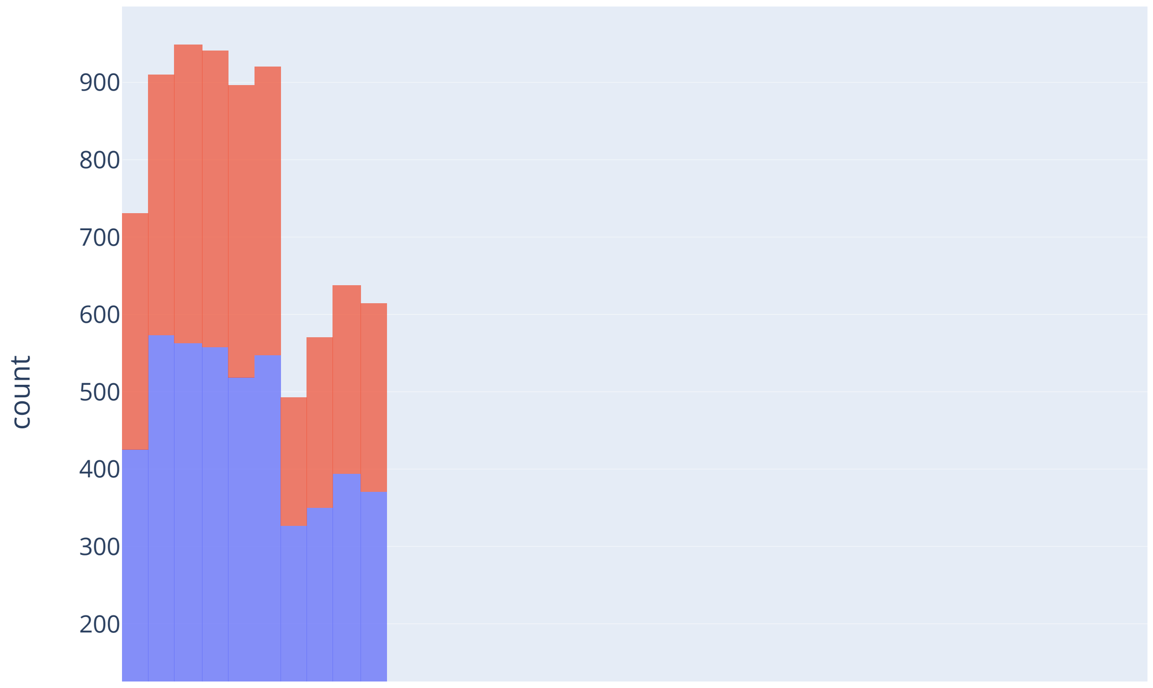
It would be informative to add barplot.

## 2.2 The impact of seat numbers

What is a feature that characterizes a chain? **many establishments with a small number of seats** or **few establishments with a lot of seats**?

In [12]:

```
fig = px.histogram(rest_data, x = 'seats_number', color = 'chain', opacity =0.75)
fig.show()
```



Judging by the graph, we can say that establishments that belong to a chain has less seat numbers among all categories, this means that:

4. Lower seat numbers is a characterestic for establishments that belongs to a chain.

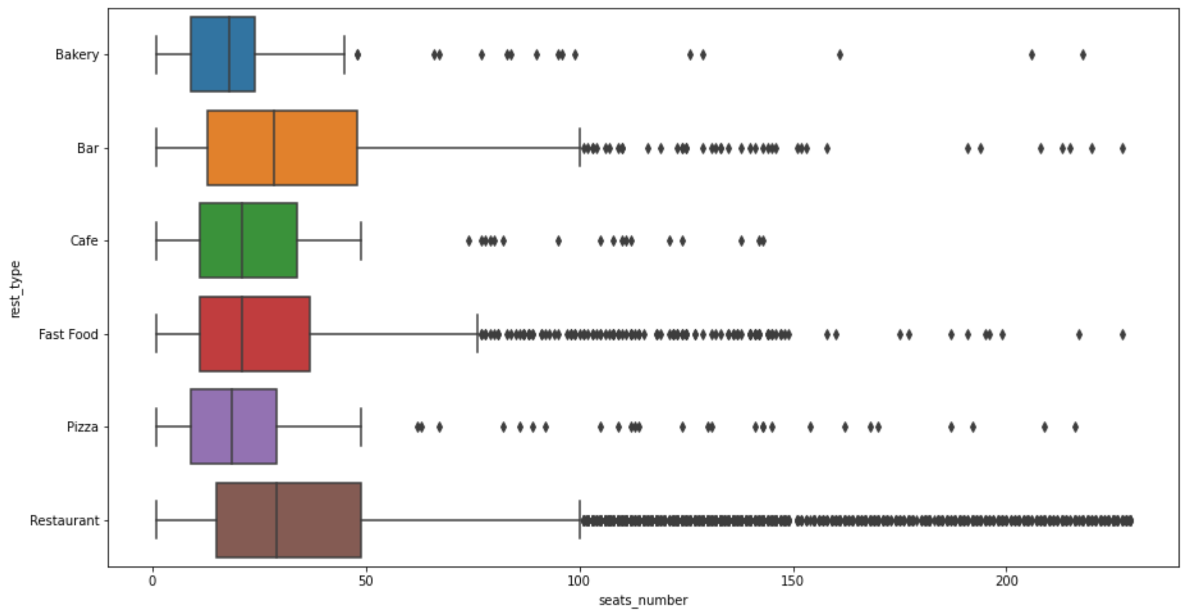
We've determined which **(Chained vs un-chained)** establishments usually has more seats. How is that distributed among the different types of establishments? Lets calculate the average seat numbers among these types and look at a graph describing the amount of seats on the different types of establishments:

In [13]:

```
plt.figure(figsize=(15,8))
ax = sns.boxplot(x = "seats_number", y = "rest_type", data = rest_data)
rest_seats = rest_data.groupby('rest_type', as_index = False).agg({'seats_number': 'mean'})
rest_seats.columns = ['rest_type', 'average_number_of_seats']
rest_seats
```

Out[13]:

| | rest_type | average_number_of_seats |
|---|---|---|
| **0** | Bakery | 21.773852 |
| **1** | Bar | 44.767123 |
| **2** | Cafe | 25.000000 |
| **3** | Fast Food | 31.837711 |
| **4** | Pizza | 28.459375 |
| **5** | Restaurant | 48.042316 |



Restaurants have the biggest amount of seats while Bakeries have the smallest amount. This makes a lot of sense because people come to a restaurant to sit and enjoy their time with other people , having some quality time together while bakeries customers usually just buy and leave with the products home.

According to the results above, we can see that **Bars** and **Restaurants** have the greatest amount of seats, this is very reasonable since people come to these places to sit and have a good time. While on the other hand, **Bakeries** and **Pizza** shops have the fewest amount of and this also makes sense because people usually don't spend much time at cafes so too many seats isn't needed, and Pizza shops usually work more with delieveries!

**Reviewer's comment**

Yep! Very informative graphs. We need to think about number of seats and type of our cafe.

## 2.3 The impact of the location

We've determined how are seat numbers distributed among the different establishment types. Now, we'll see how much impact does streets and street names have on the success of the establishments.

First of all, we'll get the street names out of the address using a python library that's made for that; **usaddress**.

Lets test the library:

In [14]:

```
# getting the street names out of the address
street_example = rest_data.iloc[0]['address']
street_example
```

Out[14]:

```
'3708 N EAGLE ROCK BLVD'
```

We use .parse() method and pass to it any unstructed address. The library will split it into components:

In [15]:

```
usaddress.parse(street_example)
```

Out[15]:

```
[('3708', 'AddressNumber'),
 ('N', 'StreetNamePreDirectional'),
 ('EAGLE', 'StreetName'),
 ('ROCK', 'StreetName'),
 ('BLVD', 'StreetNamePostType')]
```

In [16]:

```
raw_address = usaddress.parse(street_example)
raw_address
```

Out[16]:

```
[('3708', 'AddressNumber'),
 ('N', 'StreetNamePreDirectional'),
 ('EAGLE', 'StreetName'),
 ('ROCK', 'StreetName'),
 ('BLVD', 'StreetNamePostType')]
```

In [17]:

```
# saving the data of the raw_address in a dict_address (StreetName will be stored in one ce
dict_address={}
for i in raw_address:
    if i[1] not in dict_address.keys():
        dict_address.update({i[1]:i[0]})
    else:
        old_value = dict_address[i[1]]
        dict_address.update({i[1]: old_value + ' ' +  i[0]})
```

In [18]:

```
dict_address
```

Out[18]:

```
{'AddressNumber': '3708',
 'StreetNamePreDirectional': 'N',
 'StreetName': 'EAGLE ROCK',
 'StreetNamePostType': 'BLVD'}
```

With the new dictionary that we created, we can easily access any part of any street

In [19]:

```
dict_address['StreetName']
```

Out[19]:

```
'EAGLE ROCK'
```

Now that we're finished with the test sample, it seems to be working fine, lets apply it to the whole dataframe! (While exprеminting with address names we found out that some addresses don't have 'StreetName' but has 'PlaceName' instead therefore we'll include an option to return the PlaceName instead of StreetName when there's no StreetName)

In [20]:

```
def get_street_name(row):
    raw_address = usaddress.parse(row)
    dict_address = {}
    for i in raw_address:
        if i[1] not in dict_address.keys():
            dict_address.update({i[1]:i[0]})
        else:
            old_value = dict_address[i[1]]
            dict_address.update({i[1]: old_value + ' ' +  i[0]})

    if 'StreetName' in dict_address.keys():
        clean_adress=str(dict_address['StreetName'])
        return clean_adress
    else:
        clean_adress=str(dict_address['PlaceName'])
        return clean_adress
```

Before we apply the function to the whole dataframe, lets test it on a couple rows to see if we get valid results:

In [21]:

```
# testing our function
rest_data.address.sample(5).apply(get_street_name)
```

Out[21]:

```
2108     SLAUSON
589      OLYMPIC
9       CAHUENGA
4685    GLENDALE
8220    BROADWAY
Name: address, dtype: object
```

It seems to be working fine! Lets apply it to the whole dataframe:

In [22]:

```
# applying the function on the dataframe:
rest_data['street_name'] = rest_data.address.apply(get_street_name)
```

Now that we have establishment street names and address in different columns, lets look at the graph of top ten streets in terms of **restaurants**: (I didn't know if you wanted only restaurants or you meant all establishments by the word *restaurant* so I used all establishment types)

In [23]:

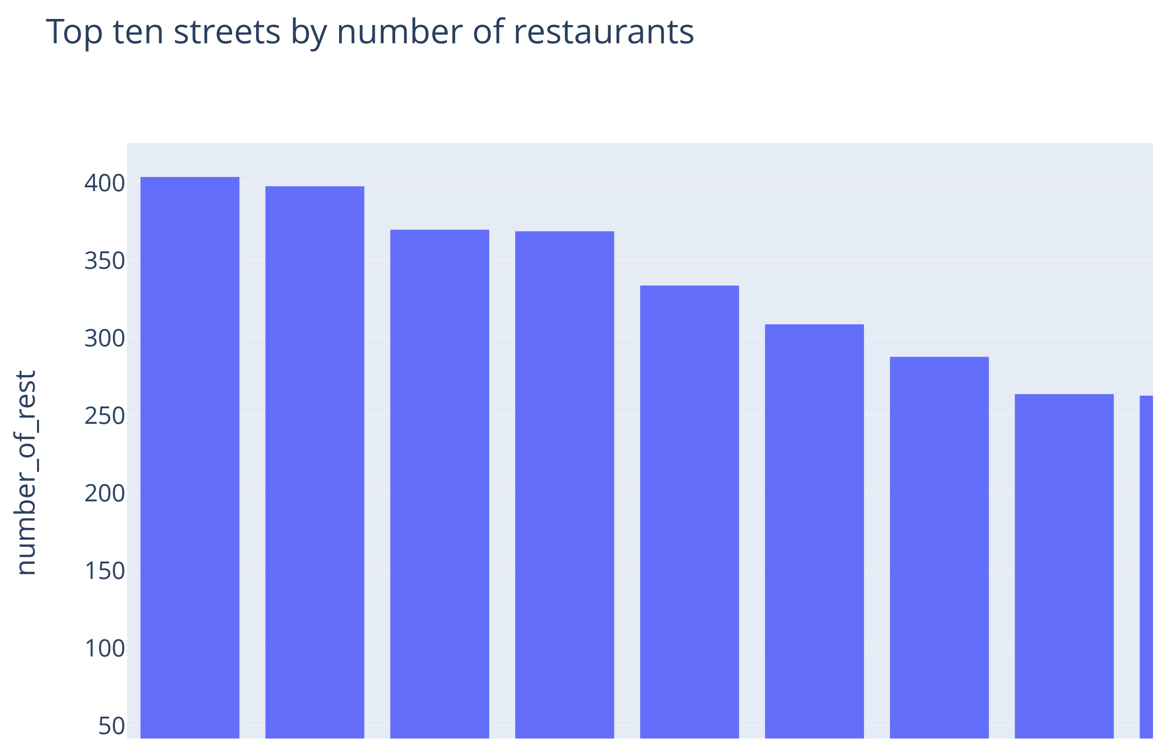
```
# grouping by the street name to get how much restaurants each street has:
res_street = rest_data.groupby('street_name', as_index = False).agg({'id': 'count', 'seats_
res_street.columns = ['street', 'number_of_rest', 'avg_number_of_seats']

# getting the top 10 streets
top_10 = res_street.sort_values(by='number_of_rest', ascending = False).head(10)

display(top_10)

fig = px.bar(top_10, x= 'street', y='number_of_rest', title = 'Top ten streets by number of
fig.show()
```

| | street | number_of_rest | avg_number_of_seats |
|---|---|---|---|
| **423** | SUNSET | 404 | 47.997525 |
| **473** | WILSHIRE | 398 | 52.655779 |
| **363** | PICO | 370 | 40.632432 |
| **463** | WESTERN | 369 | 41.344173 |
| **181** | FIGUEROA | 334 | 45.050898 |
| **345** | OLYMPIC | 309 | 49.339806 |
| **447** | VERMONT | 288 | 45.371528 |
| **393** | SANTA MONICA | 264 | 35.045455 |
| **25** | 3RD | 263 | 40.532319 |
| **227** | HOLLYWOOD | 253 | 57.047431 |



The results all look pretty close one to another, with **SUNSET** AND **WILSHIRE** Streets having the greatest amount of restaurants in them

**Reviewer's comment**

Great work with address! Sunset and Wilshire are top. :)

We've determined which Streets had the lead on number of restaurants, lets continue the research by checking which streets had the lowest numbers of restaurants, meaning streets with one restaurant:

In [24]:

```
res_street_1 = res_street.query('number_of_rest == 1')
display(len(res_street_1))
display(len(res_street_1)/len(rest_data['street_name'].unique()))
```

```
202
```

```
0.4122448979591837
```

What a surprise, over 200 streets have only one restaurant in them therefore we can't determine which streets actually have the least amount of restaurants but what we can actually tell is that almost half (41.2%...) of the streets listed have only one restaurnt, and that's a big percentage!

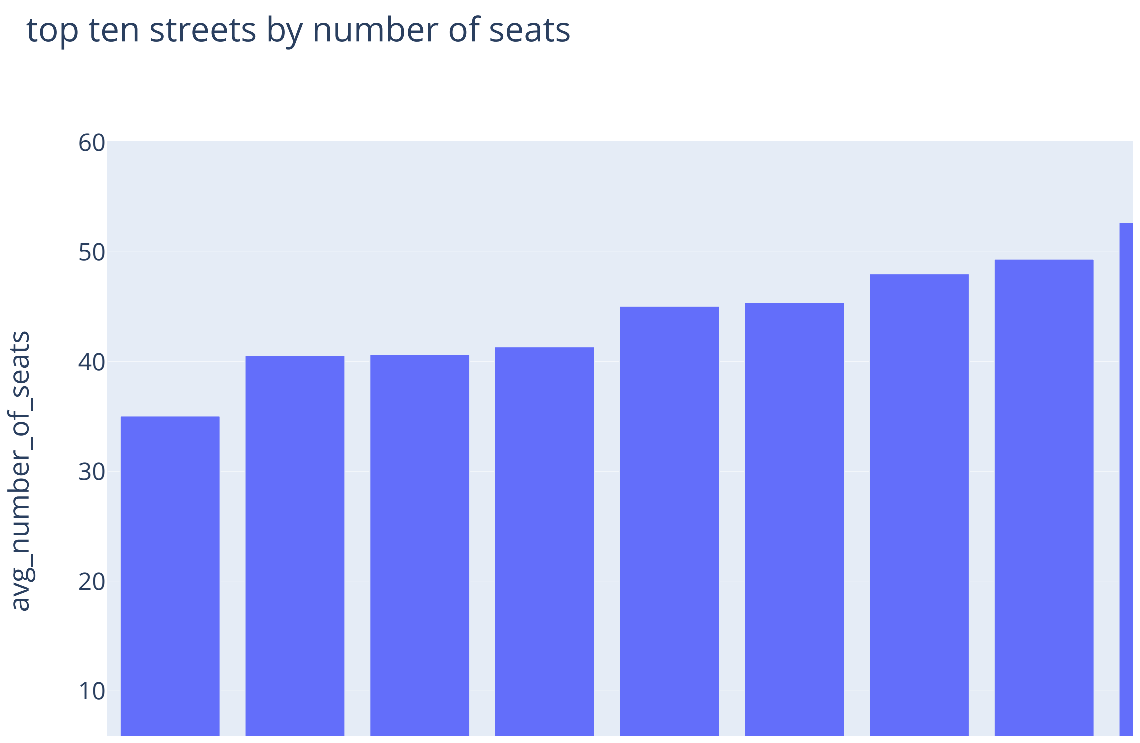
**Reviewer's comment**

LA is a big city. I think it have many streets not in the city center. This number is normal.

**Next:** we'll see how the distribution of seat numbers are among the streets with the highest numbers of restaurants in them:

In [25]:

```
fig = px.bar(top_10.sort_values(by = 'avg_number_of_seats'), x= 'street', y='avg_number_of_
fig.show()
```



I have personally expected that the more restaurants a street had the less seats it would have, but that's not the case here. We can see that **Hollywood** that had the least amount of establishments among the top 10 streets actually have the highest average of seats but that still doesn't conclude anything. If we continue looking, we can see that **Monica** and **3RD** streets that were among the lowest in restuarant numbers in the last chart, are also among the lowest here too.

**Reviewer's comment**

Excellent step! Thank you for such wonderful graph!

## 3 Preparing a presentation

presentation: https://drive.google.com/file/d/1WXa4iXlpFeXFJ_eksDedQzCNqSmukTZ0/view?usp=sharing (https://drive.google.com/file/d/1WXa4iXlpFeXFJ_eksDedQzCNqSmukTZ0/view?usp=sharing)

**Reviewer's comment**

Perfect presentation with nice style! I think investors will give you money. :)

## 4 Conclusions

First of all, lets list up all the results that we gathered below:

1. Restaurants are the most popular type of establishment, followed by Fast Foods.
2. Unchained establishments are the great majority among the establishments.
3. All bakeries belong to a chain (100%!), 61% of cafes belong to a chain and around 50% of Pizza shops and Fast Foods belong to a chain.
4. Percentage of restaurants and Bars that belong to a chain are way lower than the other establishments.
5. Establishments that belong to a chain has lower seats number than establishments that don't belong to a chain.
6. Bars & Restaurants have the highest average of seat numbers while Cafes & Bakeries have the lowest averages among the establishments.
7. SUNSET, PICO and WILSHIRE are the streets with the most amount of establishments in them.

So, after we've listed all these conclusions, how does it help us to succeed as a new establishment with a new idea?

Well, from (1) we learn that we should get away from having our establishment as a **restaurant** or maybe even as a **fast food** since they constitute the majority of the establishments thus we will be facing a very big competition. From (2) and (3) we learn that as a **cafe** or a **bakery** it is preferable to have this establishment as a part of a chain rather than have it as a solo establishment. This could be due to being able to sell our products to the largest audience to make the best profit. From (4), (5) and (6) we learn that **cafes** and **bakeries** has less seat numbers than **restaurants** and **bars**, this means that these types of establishment tend to invest more in expanding their chains to reach a larger audience rather than invest in each establishment it-self. Finally, from (7) we learn in which streets we don't want to build our establishments at to avoid high competition and early failure. It is safer to build or establishments in streets with low amounts of competition and then move on to higher and more crowded streets as we gain more and more success.

**Reviewer's comment**

Thank you for conclusion and recommendations! Very good last part.:)