# fde46a5b-1f70-4e88-99dd-f425a1ce2e17

September 16, 2024

¡ Hola Christian! Como te va?

Mi nombre es Facundo Lozano! Y tengo el agrado de ser tu revisor hoy :)

Como siempre, a continuación un poco sobre la modalidad de revisión que usaremos:

Cuando enccuentro un error por primera vez, simplemente lo señalaré, te dejaré encontrarlo y arreglarlo tú cuenta. Además, a lo largo del texto iré haciendo algunas observaciones sobre mejora en tu código y también haré comentarios sobre tus percepciones sobre el tema. Pero si aún no puedes realizar esta tarea, te daré una pista más precisa en la próxima iteración y también algunos ejemplos prácticos. Estaré abierto a comentarios y discusiones sobre el tema.

Encontrará mis comentarios a continuación: **no los mueva, modifique ni elimine**.

Puedes encontrar mis comentarios en cuadros verdes, amarillos o rojos como este:

Comentario del revisor.

Exito. Todo se ha hecho de forma exitosa.

Comentario del revisor.

Observación. Algunas recomendaciones.

Comentario del revisor.

Necesita arreglos. Este apartado necesita algunas correcciones. El trabajo no puede ser aceptado con comentarios rojos.

Puedes responder utilizando esto:

Respuesta de estudiante.

## 1 Proyecto Sprint 10

El siguiente proyecto busca preparar un estudio de mercado sobre restaurantes en Los Ángeles que ayuden a obtener información que sea de utilidad para un restaurante atendido por robots en la misma ciudad. La intención es realizar una presentación con el objetivo de atraer inversionistas para el negocio.

El Data set en el que se basará la investigación para obtener los datos y realizar las visualizaciones pertinentes contiene la siguiente información: - object_name — nombre del establecimiento - chain — establecimiento que pertenece a una cadena (TRUE/FALSE) - object_type — tipo de establecimiento - address — dirección - number — número de asientos

Review General. (Iteración 3)

Ahora esi excelente Christian hemos cumplido con todo lo necesario para aprobar el proyecto, felicitaciones! Saludos y éxitos!

Review General. (Iteración 2)

Un gran avance Christian! Has realizado importantes avances en el trabajo y corregido el link de acceso a la presentación, que es excelente. Sola resta revisar la consigna del histograma para poder dar este proyecto como aprobado. Te dejé un nuevo comentario con más detalles para que puedas buscar la forma de resolverlo.

Espero nuevamente a nuestra próxima iteración que estoy seguro que sera nuestra ultima :)

Exitos y saludos!

Review General. (Iteración 1)

Christian, siempre me tomo este tiempo al inicio del proyecto para comentar mis apreciaciones generales de esta primera iteración de la entrega.

Siempre me gusta comenzar dando la bienvenida al mundo de los datos a los estudiantes, te deseo lo mejor y espero que consigas lograr tus objetivos. Personalmente siempre me gusta brindar el siguiente consejo, "Está bien equivocarse, es normal y es lo mejor que te puede pasar. Aprendemos de los errores y eso te hará mejor programando ya que podrás descubrir cosas a medida que avances y son estas cosas las que te darán esa experiencia para ser mejor como Data Analyst"

Ahora si yendo a esta notebook. Quiero felicitarte y agradecerte por este proyecto Christian, lo has resuelto de una forma espectacular, se ha notado a lo largo de todo el proceso tu gran manejo sobre python y las librerías que debían utilizarse. A la vez quiero destacar tu compromiso porque no solo has resuelto sino que lo has resuelto con creces. Por otro lado, simplementenos hay unos detalles a modificar y faltò ademàs el link a la presentación. Devuelvo el proyecto para que lo agreguemos :)

Éxitos y saludos Christian!

### 1.1 Importación de librerias y datos.

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import re
import plotly.graph_objects as go
```

[3]:
```
rest_data = pd.read_csv('/datasets/rest_data_us_upd.csv')

display(rest_data.head(3))
```

```
      id          object_name                  address  chain object_type  \
0  11786  HABITAT COFFEE SHOP   3708 N EAGLE ROCK BLVD  False        Cafe
1  11787             REILLY'S        100 WORLD WAY 120  False  Restaurant
2  11788       STREET CHURROS  6801 HOLLYWOOD BLVD 253  False   Fast Food
```

```
   number
0      26
1       9
2      20
```

### 1.2 Análisis exploratorio de datos.

[4]:
```
print('Forma de la base rest_data')
display(rest_data.shape)
print('Información general de la base rest_data')
display(rest_data.info())
print('Primeras 5 filas de la base rest_data')
display(rest_data.head())
print('Últimas 5 filas de la base rest_data')
display(rest_data.tail())
print('Descripción estadística de la base rest_data')
display(rest_data.describe())
print('Filas duplicadas de la base rest_data')
display(f'La base tiene: {rest_data.duplicated().sum()} filas duplicadas')
```

```
Forma de la base rest_data
```

```
(9651, 6)
```

```
Información general de la base rest_data
<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
```

```
None
```

```
Primeras 5 filas de la base rest_data
```

```
      id          object_name                  address  chain object_type  \
0  11786  HABITAT COFFEE SHOP   3708 N EAGLE ROCK BLVD  False        Cafe   
1  11787            REILLY'S        100 WORLD WAY 120  False  Restaurant   
2  11788       STREET CHURROS  6801 HOLLYWOOD BLVD 253  False   Fast Food   
3  11789    TRINITI ECHO PARK       1814 W SUNSET BLVD  False  Restaurant   
4  11790              POLLEN       2100 ECHO PARK AVE  False  Restaurant   
```

```
   number
0      26
1       9
2      20
3      22
4      20
```

```
Últimas 5 filas de la base rest_data
```

```
         id      object_name                      address  chain object_type  \
9646  21432  HALL OF JUSTICE            217 W TEMPLE AVE  False  Restaurant
9647  21433      FIN-MELROSE            5750 MELROSE AVE  False  Restaurant
9648  21434      JUICY WINGZ          6741 HOLLYWOOD BLVD   True   Fast Food
9649  21435  MEDIDATE COFFEE      548 S SPRING ST STE 100  False        Cafe
9650  21436     CAFE SPROUTS  1300 S SAN PEDRO ST STE 111   True  Restaurant

      number
9646     122
9647      93
9648      15
9649       6
9650      19
```

```
Descripción estadística de la base rest_data
```

```
                 id       number
count   9651.000000  9651.000000
mean   16611.000000    43.695161
std     2786.148058    47.622874
min    11786.000000     1.000000
25%    14198.500000    14.000000
50%    16611.000000    27.000000
75%    19023.500000    46.000000
max    21436.000000   229.000000
```

```
Filas duplicadas de la base rest_data
```

```
'La base tiene: 0 filas duplicadas'
```

### 1.2.1 Conclusiones del análisis exploratorio de datos.

La base de datos cuenta con 9651 filas y 6 columnas, en su mayoría podemos contemplarlos como valores categóricos. Solo tiene 3 valores nulos en la columna de chain los cuales se analizarán y se verá su tratamiento en la siguiente parte del proceso.

### 1.2.2 Tratamiento de datos.

[5]:
```
rest_data_isnull = rest_data[rest_data['chain'].isnull()]
display(rest_data_isnull.head())

rest_data_notnull = rest_data[~rest_data['chain'].isnull()]
```

```
display(rest_data_notnull.head())

len_isnull = len(rest_data_isnull)
len_notnull = len(rest_data_notnull)

null_rate = len_isnull / len_notnull

print(null_rate)
```

```
        id              object_name               address chain object_type  \
7408  19194  TAQUERIA LOS 3 CARNALES  5000 E WHITTIER BLVD   NaN  Restaurant
7523  19309     JAMMIN JIMMY'S PIZZA   1641 FIRESTONE BLVD   NaN       Pizza
8648  20434    THE LEXINGTON THEATER          129 E 3RD ST   NaN  Restaurant

      number
7408      14
7523       1
8648      35
```

```
      id          object_name                   address  chain object_type  \
0  11786  HABITAT COFFEE SHOP   3708 N EAGLE ROCK BLVD  False        Cafe
1  11787             REILLY'S        100 WORLD WAY 120  False  Restaurant
2  11788       STREET CHURROS  6801 HOLLYWOOD BLVD 253  False   Fast Food
3  11789    TRINITI ECHO PARK       1814 W SUNSET BLVD  False  Restaurant
4  11790               POLLEN       2100 ECHO PARK AVE  False  Restaurant

   number
0      26
1       9
2      20
3      22
4      20
```

```
0.0003109452736318408
```

[6]:
```
rest_data = rest_data.dropna()

rest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9648 entries, 0 to 9650
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           9648 non-null   int64
 1   object_name  9648 non-null   object
 2   address      9648 non-null   object
 3   chain        9648 non-null   object
 4   object_type  9648 non-null   object
```

```
 5   number       9648 non-null   int64
dtypes: int64(2), object(4)
memory usage: 527.6+ KB
```

Los valores nulos fueron elimindados ya que representaban una cantidad muy poco representativa para que afecten en el resultado final.

Comentario del revisor. (Iteración 1)

Christian. Bien realizada la importaciòn de la base de datos. Y aquì un buen anàlisis para identificar duplicdos y valores ausentes. Bien hecho!

## 1.3 Análisis y visualizaciones de datos.

[7]:
```
rest_data.head(2)
```

[7]:
```
      id          object_name                  address  chain object_type  \
0  11786  HABITAT COFFEE SHOP  3708 N EAGLE ROCK BLVD  False        Cafe
1  11787             REILLY'S       100 WORLD WAY 120  False  Restaurant

   number
0      26
1       9
```

### 1.3.1 Proporciones de los distintos tipos de establecimientos.

[8]:
```
rest_data_type = rest_data['object_type'].value_counts(normalize = True).
 ↪reset_index()
rest_data_type.columns = ['object_type', 'proportion']

fig = px.bar(
    rest_data_type,
    x = 'object_type',
    y = 'proportion',
    title = 'Proporción de tipos de establelcimiento',
    labels = {
        'object_type': 'Tipo de establecimiento',
        'proportion': 'Proporción'
    },
    color = 'object_type'
)

fig.show(renderer="iframe")
```

Comentario del revisor. (Iteración 1)

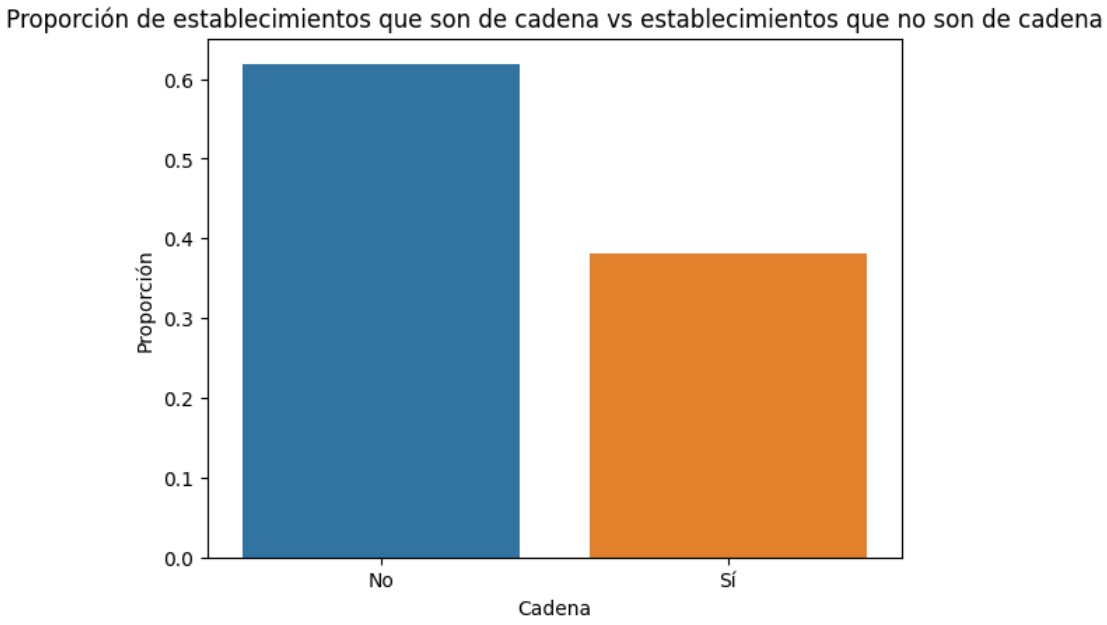
Excelente análisis y comprensión del mismo Christian, hemos logrado el objetivo de visualizar las proporciones de los establecimientos, excelente!

**Conclusiones de las representaciones gráficas de los distintos tipos de establecimientos.** La gran mayoría de los establecimientos en nuestra base de datos son restaurantes siendo poco más de un 75% del total. Los siguientes son la comida chatarra con 11% y el resto se encuentran por debajo del 5%.

### 1.3.2 Proporciones de establecimientos que pertenecen a una cadena vs las no pertenecientes.

[9]:
```
chain_vs = rest_data['chain'].value_counts(normalize = True).reset_index()
chain_vs.columns = ['chain', 'proportion']
chain_vs['chain'] = chain_vs['chain'].replace({True: 'Sí', False: 'No'})

ax = sns.barplot(x = 'chain', y = 'proportion', data = chain_vs)
ax.set_xlabel('Cadena')
ax.set_ylabel('Proporción')
ax.set_title('Proporción de establecimientos que son de cadena vs␣
 ↪establecimientos que no son de cadena')
```

[9]:
```
Text(0.5, 1.0, 'Proporción de establecimientos que son de cadena vs
establecimientos que no son de cadena')
```




Comentario del revisor. (Iteración 1)

Y aquí un gráfico ideal para poder visualizar por tipo de establecimiento las cadenas, impresionante!

**Grafica de establecimientos pertenecientes a una cadena vs no pertenecientes.** Poco más del 62% de los establecimientos registrados no pertenecen a una cadena comercial.

### 1.3.3 Tipos de establecimientos que son habitualmente una cadena.

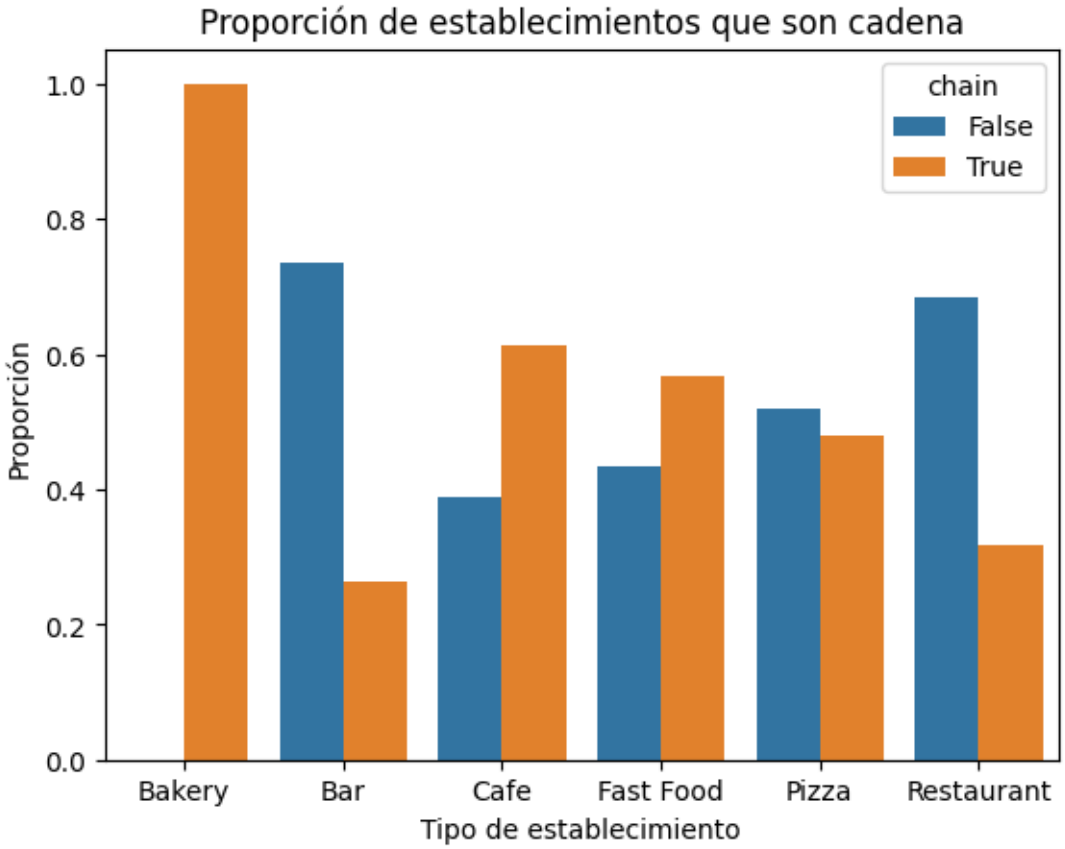
[10]:
```
count = (
    rest_data
    .groupby(['object_type', 'chain'])
    .size()
    .reset_index(name = 'count')
)

total_stablishment = (
    rest_data
    .groupby('object_type')
    .size()
    .reset_index(name = 'total')
)

total_merged = (
    count
    .merge(
        total_stablishment,
        on = 'object_type',
        how = 'left'
    )
)

total_merged['proportion'] = total_merged['count'] / total_merged['total']

total_merged
```

[10]:

| | object_type | chain | count | total | proportion |
|---|---|---|---|---|---|
| 0 | Bakery | True | 283 | 283 | 1.000000 |
| 1 | Bar | False | 215 | 292 | 0.736301 |
| 2 | Bar | True | 77 | 292 | 0.263699 |
| 3 | Cafe | False | 169 | 435 | 0.388506 |
| 4 | Cafe | True | 266 | 435 | 0.611494 |
| 5 | Fast Food | False | 461 | 1066 | 0.432458 |
| 6 | Fast Food | True | 605 | 1066 | 0.567542 |
| 7 | Pizza | False | 166 | 319 | 0.520376 |
| 8 | Pizza | True | 153 | 319 | 0.479624 |
| 9 | Restaurant | False | 4961 | 7253 | 0.683993 |
| 10 | Restaurant | True | 2292 | 7253 | 0.316007 |

[11]:
```
sns.barplot(
    data = total_merged,
    x = 'object_type',
    y = 'proportion',
    hue = 'chain'
```

```
)

handles, labels = ax.get_legend_handles_labels()

plt.title('Proporción de establecimientos que son cadena')
plt.xlabel('Tipo de establecimiento')
plt.ylabel('Proporción')

plt.show()
```




Comentario del revisor. (Iteración 1)

Cuidado aquí Christian, la intención es correcta pero deberíamos cambiar el enfoque. Si queremos corroborar que tipo de establecimiento es habitualmente una cadena no debemos simplemente contabilizar la cantidad de cadenas que hay por establecimiento a nivel general sino ver el porcentaje del mismo, es decir sobre los restaurantes, ¿cuál es el porcentaje de restaurantes que son cadenas? ¿y sobre bakery?, es decir que si bakery tiene 9 cadenas sobre 10 establecimientos tiene un 90% de cadenas mientras que si restaurante tiene 20 de 100 significa que un 20% es habitualmente cadena, etc. Tratemos de corregirlo pero con este nuevo enfoque que te comento :)

Respuesta de estudiante.

Muchas gracias por el comentario. Realmente era la gráfica que más me hacía ruido dejarla así, pero no se me había ocurrido manejarla de esta forma.

Comentario del revisor. (Iteración 2)

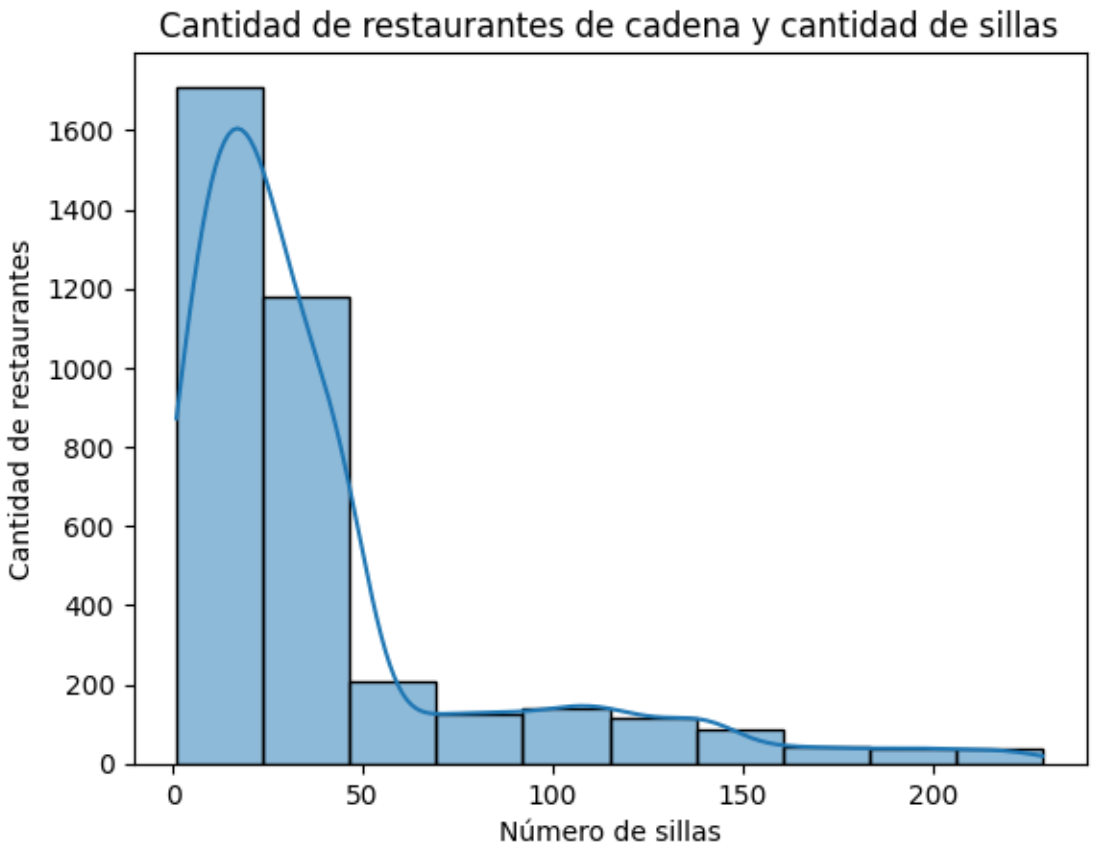
Ahora sí! Has resuelto perfectamente el error. Efectivamente son las Bakery los tipos de establecimiento que son, en su totalidad, cadenas. Excelente.

**Conclusiones de tipos de establecimientos más comunes en las cadenas.** Dentro de los datos que tenemos, todos los establecimientos de panaderías pertenecen a una cadena, mientras que en el caso de cafeterías y establecimientos de comida rápida la mayoría pertenecen a alguna cadena. Las pizzerías tienen una distribución similar entre cadena y no establecimiento de cadena. Los restaurantes y bares se caracterizan por no pertenecer a una cadena.

### 1.3.4 ¿Qué carácteriza a una cadena?

[38]:
```
chain = (
    rest_data[rest_data['chain'] == True]
)

median = chain['number'].median()

print(f'La mediana de la cantidad de sillas es: {median}')

sns.histplot(
    data = chain,
    x = 'number',
    bins = 10,
    kde = True
)

plt.title('Cantidad de restaurantes de cadena y cantidad de sillas')
plt.xlabel('Número de sillas')
plt.ylabel('Cantidad de restaurantes')

plt.show()
```

```
La mediana de la cantidad de sillas es: 25.0
```




Comentario del revisor. (Iteración 1) Has realizado un buen anàlisis de los datos. Sin embargo, para este caso, una mejor forma de visualizacòn para poder responder a la consigna sería mediante la implementación de un gráfico de tipo de histograma. Prueba hacerlo asì.

Respuesta de estudiante.

Intenté corregirlo, pero llego a una visualización igual. No supe cómo resolver este problema.

Comentario del revisor. (Iteración 2)

Christian, repasemos nuevamente la consigna. Se solicita que analices las características de los establecimientos que sí son una cadena. Cuando son cadenas, ¿tenemos muchos establecimientos con un pequeño número de asientos o unos pocos establecimientos con un montón de asientos? Te recomiendo que pruebes filtrando cuando la condición de cadena sea true, veas la cantidad de establecimientos y analices la mediana. De esta manera podrías aplicas un gráfico tipo histograma viendo con qué frecuencia aparecen establecimientos según distintas agrupaciones de cantidades de asientos. Probemos de nuevo!

`Ahora si excelente Christian, esto es lo que se buscaba! La cantidad de bins puede variar, por`

Respuesta de estudiante.

Listo, Facundo. Creo que ya entendí lo que se buscaba y al menos ahora me hace mucho sentido la gráfica. En caso de que sea lo buscado, me gustaría saber cuál sería la cantidad de bins más

adecuada. Intenté con varias opciones, pero no supe cómo tomar una decición fundamentada.

[14]:
```
type_merged_grouped = (
    type_merged
    .groupby('object_type')['rate'].mean()
    .reset_index()
)

mean_rate = type_merged_grouped['rate'].mean()
median_rate = type_merged_grouped['rate'].median()

# Crear el gráfico de líneas
fig_line = go.Figure()

# Agregar la línea de datos
fig_line.add_trace(go.Scatter(
    x=type_merged_grouped['object_type'],
    y=type_merged_grouped['rate'],
    mode='lines+markers',
    name='Promedio de Sillas por Establecimiento'
))

# Agregar línea de promedio
fig_line.add_trace(go.Scatter(
    x=type_merged_grouped['object_type'],
    y=[mean_rate] * len(type_merged_grouped),
    mode='lines',
    name=f'Promedio ({mean_rate:.2f})',
    line=dict(color='red', dash='dash')
))

# Agregar línea de mediana
fig_line.add_trace(go.Scatter(
    x=type_merged_grouped['object_type'],
    y=[median_rate] * len(type_merged_grouped),
    mode='lines',
    name=f'Mediana ({median_rate:.2f})',
    line=dict(color='blue', dash='dash')
))

# Actualizar el diseño del gráfico
fig_line.update_layout(
    title='Promedio de sillas por establecimiento',
    xaxis_title='Tipo de establecimiento',
    yaxis_title='Promedio',
    xaxis_tickangle=60
)
```

```
# Mostrar el gráfico
fig_line.show(renderer="iframe")
```

**Conclusiones de características de cadenas.** La gran mayoría de los establecimientos que pertenecen a una cadena tienen menos de 50 sillas por cada uno. Menos de 500 tienen entre 50 y 100 sillas. En promedio los establecimientos registrados. El promedio de sillas en los establecimientos es de 32.72, los restaurantes y los bares están por arriba del promedio mienntras que el resto se encuentran por debajo.

### 1.3.5 Promedio de cada tipo de restaurante.

[15]:
```
type_merged['chair_average'] = type_merged['chair_number'] /␣
 ↪type_merged['total_type']

type_merged
```

[15]:

| | chain | object_type | total_type | chair_number | rate | chair_average |
|---|---|---|---|---|---|---|
| 1 | False | Restaurant | 4961 | 245688 | 49.523886 | 49.523886 |
| 9 | False | Bar | 215 | 9972 | 46.381395 | 46.381395 |
| 0 | True | Restaurant | 2292 | 102810 | 44.856021 | 44.856021 |
| 8 | True | Bar | 77 | 3100 | 40.259740 | 40.259740 |
| 4 | True | Fast Food | 605 | 23044 | 38.089256 | 38.089256 |
| 3 | False | Pizza | 166 | 5200 | 31.325301 | 31.325301 |
| 6 | True | Cafe | 266 | 6894 | 25.917293 | 25.917293 |
| 2 | True | Pizza | 153 | 3906 | 25.529412 | 25.529412 |
| 5 | False | Fast Food | 461 | 10895 | 23.633406 | 23.633406 |
| 7 | False | Cafe | 169 | 3981 | 23.556213 | 23.556213 |
| 10 | True | Bakery | 283 | 6162 | 21.773852 | 21.773852 |

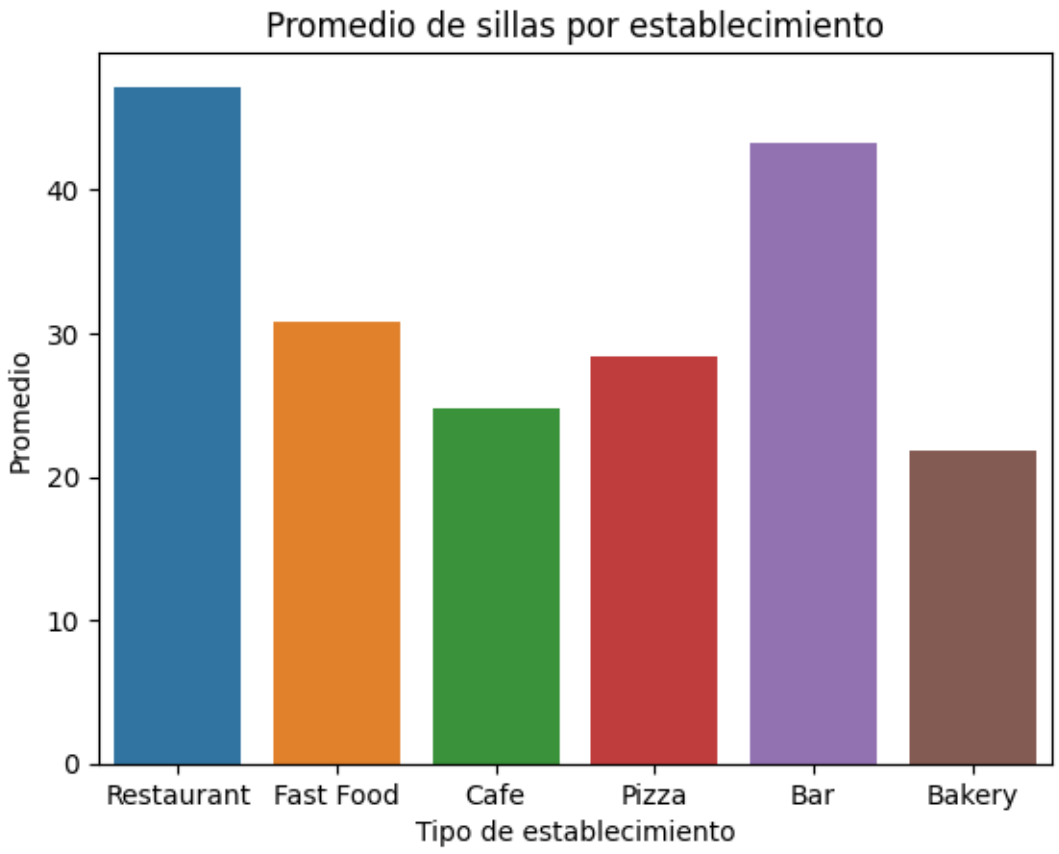
[16]:
```
average_chairs_per_type = (
    type_merged.groupby('object_type')
    .agg({
        'total_type': 'sum',
        'chair_average': 'mean'
    })
    .sort_values(by = 'total_type', ascending = False)
    .reset_index()
)

average_chairs_per_type
```

[16]:

| | object_type | total_type | chair_average |
|---|---|---|---|
| 0 | Restaurant | 7253 | 47.189954 |
| 1 | Fast Food | 1066 | 30.861331 |
| 2 | Cafe | 435 | 24.736753 |

```
3       Pizza          319      28.427356
4         Bar          292      43.320568
5      Bakery          283      21.773852
```

[17]:
```
average_chairs_per_type

sns.barplot(
    data = average_chairs_per_type,
    x = 'object_type',
    y = 'chair_average'
)
plt.xlabel('Tipo de establecimiento')
plt.ylabel('Promedio')
plt.title('Promedio de sillas por establecimiento')

plt.show()
```




Comentario del revisor. (Iteración 1)

Muy bien resuelto. Simple y conciso.

**Los restaurantes tienen la mayor cantidad de**

[18]:
```
average_chairs_per_type
```

[18]:

| | object_type | total_type | chair_average |
|---|---|---|---|
| 0 | Restaurant | 7253 | 47.189954 |
| 1 | Fast Food | 1066 | 30.861331 |
| 2 | Cafe | 435 | 24.736753 |
| 3 | Pizza | 319 | 28.427356 |
| 4 | Bar | 292 | 43.320568 |
| 5 | Bakery | 283 | 21.773852 |

### 1.3.6 Nombres de las calles.

[19]:
```
def extrac_street_name(address):
    match = re.search(r'\d+\s*(.*)', address)
    return match.group(1).strip() if match else address

rest_data['street_name'] = rest_data['address'].apply(extrac_street_name)

# Mostrar el DataFrame resultante
display(rest_data.head())
```

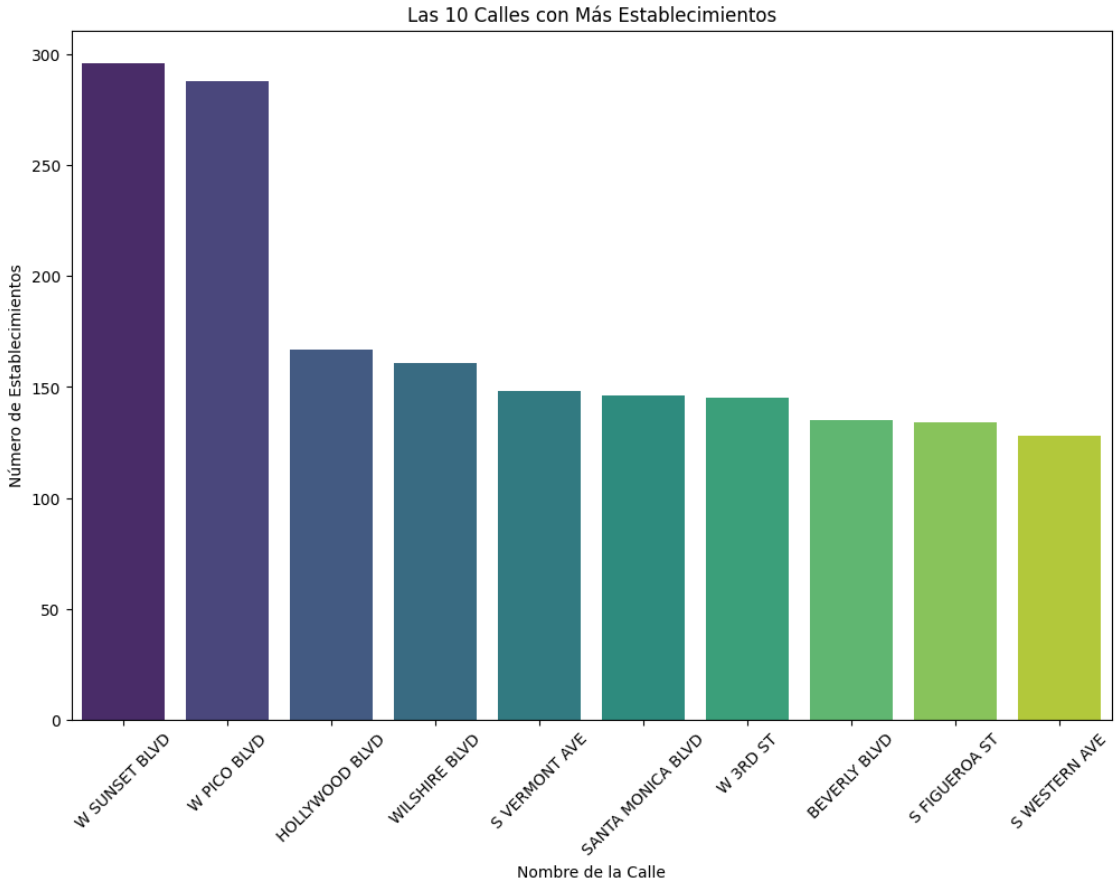
| | id | object_name | address | chain | object_type | \ |
|---|---|---|---|---|---|---|
| 0 | 11786 | HABITAT COFFEE SHOP | 3708 N EAGLE ROCK BLVD | False | Cafe | |
| 1 | 11787 | REILLY'S | 100 WORLD WAY 120 | False | Restaurant | |
| 2 | 11788 | STREET CHURROS | 6801 HOLLYWOOD BLVD 253 | False | Fast Food | |
| 3 | 11789 | TRINITI ECHO PARK | 1814 W SUNSET BLVD | False | Restaurant | |
| 4 | 11790 | POLLEN | 2100 ECHO PARK AVE | False | Restaurant | |

| | number | street_name |
|---|---|---|
| 0 | 26 | N EAGLE ROCK BLVD |
| 1 | 9 | WORLD WAY 120 |
| 2 | 20 | HOLLYWOOD BLVD 253 |
| 3 | 22 | W SUNSET BLVD |
| 4 | 20 | ECHO PARK AVE |

### 1.3.7 10 mejores calles por números de restaurante.

[20]:
```
street_counts = rest_data['street_name'].value_counts().reset_index()
street_counts.columns = ['street_name', 'establishment_count']

top_streets = street_counts.head(10)

plt.figure(figsize=(12, 8))
sns.barplot(
    data=top_streets,
    x='street_name',
```

```
    y='establishment_count',
    palette='viridis'
)

plt.xlabel('Nombre de la Calle')
plt.ylabel('Número de Establecimientos')
plt.title('Las 10 Calles con Más Establecimientos')
plt.xticks(rotation = 45)

plt.show()
```




Comentario del revisor. (Iteración 1)

Excelente Christian! No todos los estudiantes lo logran a la primera, excelente gráfico derivado de la transformación correcta sobre los nombres de las calles, bien hecho!

### 1.3.8 Calles con un solo restaurante.

[21]:
```
one_object_streets = street_counts[street_counts['establishment_count'] == 1]

print(f'La cantidad de calles con un solo restaurante es:␣
 ↪{len(one_object_streets)}')
```

```
La cantidad de calles con un solo restaurante es: 2442
```

Comentario del revisor. (Iteración 1)

Fantástico. Has llegado al número correcto.

### 1.3.9 Calles con gran número de restaurantes y distribución del número de asientos.

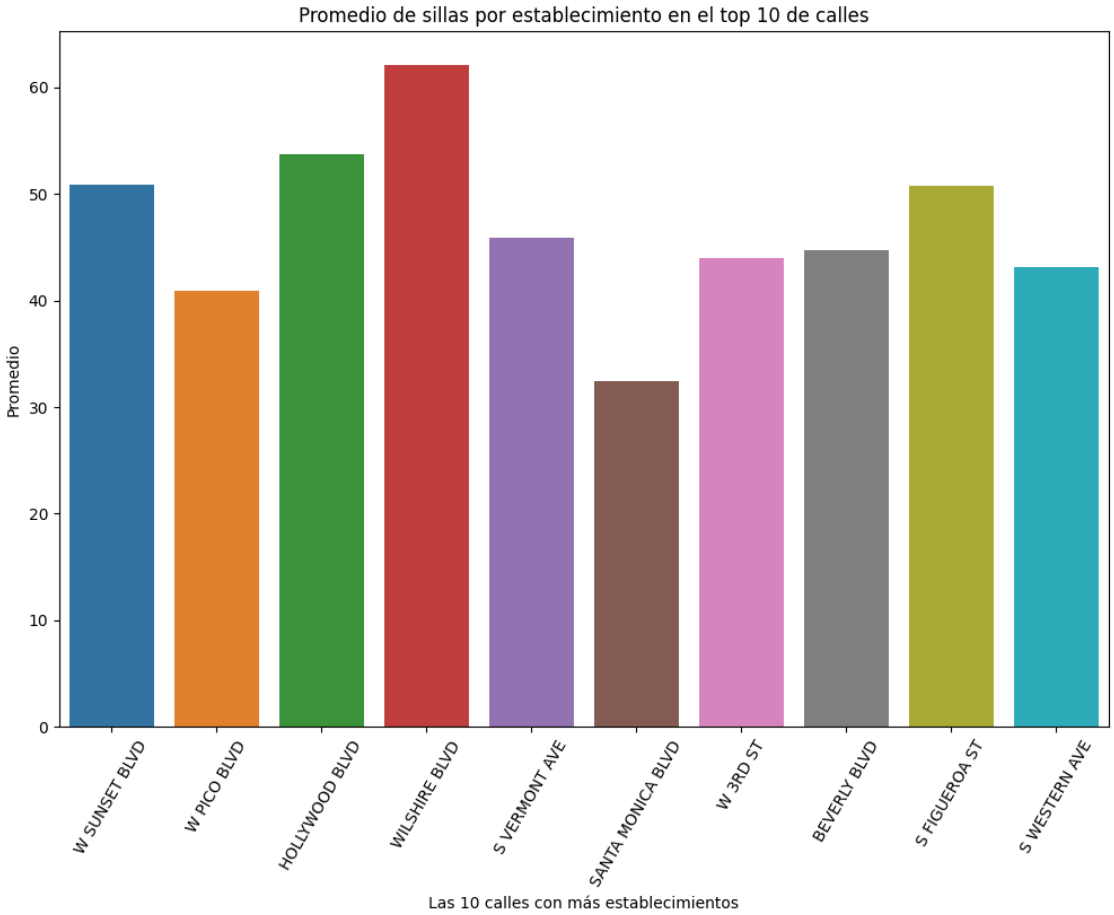
[22]:
```
t = top_streets['street_name'].tolist()

filtered_data = (
    rest_data[rest_data['street_name']
    .isin(t)]
    .reset_index()
)

filtered_merged = (
    filtered_data
    .merge(
        street_counts,
        on = 'street_name',
        how = 'left'
    )
)

filtered_grouped = (
    filtered_merged
    .groupby('street_name')
    .agg({
        'establishment_count': 'first',
        'number': 'sum'
    })
    .sort_values(by = 'establishment_count', ascending = False)
    .reset_index()
)

filtered_grouped['chair_rate'] = (
    filtered_grouped['number']
    / filtered_grouped['establishment_count']
)
```

```
filtered_grouped = filtered_grouped.sort_values(by ='establishment_count',␣
 ↪ascending = False)

filtered_grouped
```

[22]:

| | street_name | establishment_count | number | chair_rate |
|---|---|---|---|---|
| 0 | W SUNSET BLVD | 296 | 15072 | 50.918919 |
| 1 | W PICO BLVD | 288 | 11773 | 40.878472 |
| 2 | HOLLYWOOD BLVD | 167 | 8973 | 53.730539 |
| 3 | WILSHIRE BLVD | 161 | 10003 | 62.130435 |
| 4 | S VERMONT AVE | 148 | 6790 | 45.878378 |
| 5 | SANTA MONICA BLVD | 146 | 4729 | 32.390411 |
| 6 | W 3RD ST | 145 | 6370 | 43.931034 |
| 7 | BEVERLY BLVD | 135 | 6044 | 44.770370 |
| 8 | S FIGUEROA ST | 134 | 6802 | 50.761194 |
| 9 | S WESTERN AVE | 128 | 5517 | 43.101562 |

[23]:
```
plt.figure(figsize=(12, 8))
sns.barplot(
    data = filtered_grouped,
    x = 'street_name',
    y = 'chair_rate'
)
plt.xlabel('Las 10 calles con más establecimientos')
plt.ylabel('Promedio')
plt.title('Promedio de sillas por establecimiento en el top 10 de calles')
plt.xticks(rotation = 60)

plt.show()
```




Comentario del revisor. (Iteración 1)

Un trabajo espectacular Christian, estos últimos agregados profundizando las distribuciones en las calles más populares, felicitaciones!

[24]:
```
mean_rate_top = filtered_grouped['chair_rate'].mean()
median_rate_top = filtered_grouped['chair_rate'].median()


fig_line_top = go.Figure()

fig_line_top.add_trace(go.Scatter(
    x=filtered_grouped['street_name'],
    y=filtered_grouped['chair_rate'],
    mode='lines+markers',
    name='Promedio de Sillas por calle'
))
```

```
fig_line_top.add_trace(go.Scatter(
    x=filtered_grouped['street_name'],
    y=[mean_rate_top] * len(filtered_grouped),
    mode='lines',
    name=f'Promedio ({mean_rate_top:.2f})',
    line=dict(color='red', dash='dash')
))

fig_line_top.add_trace(go.Scatter(
    x=filtered_grouped['street_name'],
    y=[median_rate_top] * len(filtered_grouped),
    mode='lines',
    name=f'Mediana ({median_rate_top:.2f})',
    line=dict(color='blue', dash='dash')
))

fig_line_top.update_layout(
    title='Promedio de sillas por calle',
    xaxis_title='Calle',
    yaxis_title='Promedio',
    xaxis_tickangle=60
)

fig_line_top.show(renderer="iframe")
```

Comentario del revisor. (Iteración 1) Buena forma de llegar a determinas las calles con màs sillas. Bien hecho.

**Conclusiones de calles con gran número de restaurantes y distribución del número de asientos.** En promedio, las sillas que tienen los establecimientos en las calles más con mayor número de ellos supera al promedio de las sillas de toda la base de datos. Podría pensar que ya que hay más opciones, más gente suele asistir a estos lugares y buscan tener una mayor posibilidad de recibir clientes para dar el servicio. También, hay 5 calles que superan o están muy cerca del promedio de sillas.

### 1.4 Presentación para inversionistas.

Presentación: https://drive.google.com/file/d/1wY5Od8SDGt2xmiRyA8OLDwhHAUHVWd7s/view?usp=sharing

### 1.5 Conclusiones generales.

- La realización de este proyecto está basada en un solo data set. Este data set fue tratado para procesar solo 3 valores nulos en la columna de chain que contenía 3 valores nulos. Los valores fueron eliminados ya que no representaban un porcentaje representativo de los datos.
- Una mayoría de establecimientos que no son de cadena, estos son representados por el 62% de los registros el resto son establecimientos de cadena.

- Los restaurantes que no son de cadena superan por una cantidad considerable a los que sí lo son.
- La proporción de sillas disponibles con la cantidad de establecimientos no parece tener una relación en si son o no de cadena.
- En promedio los restaurantes y los bares son los establecimientos con mayor número de asientos.
- Las calles con un mayor número de establecimientos tienen un mayor promedio de sillas disponibles.

Comentario del revisor. (Iteración 1)

Christian, te solicito que habilites el link para que cualquier persona pueda tener acceso al mismo asì puedo corregir la presentación. Espero ese cambio para poder aprobar esta parte del trabajo!

Comentario del revisor. (Iteración 2)

Ahora si, la presentación está muy clara. Está bien pensada la comunicación hacia un potencial cliente. Felicitaciones!