INSIGHTREACH
CONECTANDO NEGOCIOS LOCALES CON CLIENTES IDEALES

**Introducción**

Este proyecto integrador tiene como objetivo analizar el mercado gastronómico de una ciudad elegida ( Miami en este caso), integrando datos internos de comportamiento del consumidor con información externa obtenida a través de la API de Yelp y un dataset principal de clientes proporcionado para hacer análisis exploratorio EDA, limpieza y determinar hallazgos entre estos dos con visualizaciones que lo sustenten.

La estructura del documento está dada por el mismo proyecto integrador por avances 1,2 y 3. Cumpliendo las tareas y sugerencias de Henry luego aportadas por los mismos hallazgos en el proceso.

El análisis busca identificar patrones de consumo, estructura de precios, reputación de restaurantes y tendencias alimentarias, con el fin de generar recomendaciones estratégicas basadas en datos.

# 1. OBJETIVOS

## objetivos PI

- Explorar y comprender estructuras de datos reales provenientes de fuentes diversas.
- Aplicar técnicas de limpieza y transformación de datos para preparar datasets para análisis.
- Utilizar librerías de Python como pandas, numpy, matplotlib y seaborn para realizar análisis exploratorio de datos (EDA).
- Conectar con una API externa (Yelp) para enriquecer un dataset con información contextual para extraer datos relevantes de páginas web.
- Interpretar los resultados del análisis para generar recomendaciones accionables.
- Documentar el proceso de análisis de forma clara, estructurada y reproducible.

## Objetivos para el cliente

Identificar oportunidades de alto impacto económico en el mercado gastronómico de Miami mediante el análisis del comportamiento del consumidor y la oferta gastronómica local.

## Objetivos Específicos

- Analizar patrones de gasto y frecuencia de consumo.
- Evaluar la estructura del mercado por rangos de precio y categorías.
- Analizar la relación entre reputación (ratings), precio y demanda.
- Detectar tendencias hacia alimentación saludable y plant-based.
- Generar recomendaciones estratégicas orientadas a clientes premium y fidelización.

## Tecnologías Utilizadas

Python-Pandas-Matplotlib-Seaborn-Jupyter Notebook-API REST (Yelp).

Informe Técnico (el paso a paso) y resultados del proceso. AVANCE #1

1. Carga de datos
   a. Se carga y explora el data set base_datos_restaurantes_USA_v2.csv y yelp_restaurants.csv. Se destaca que es un archivo CSV-Comma-Separated Values (valores separados por comas). Es un formato de archivo de texto muy usado para guardar y compartir datos en forma de tabla.
   b. Se cargan dichos archivos en VSC (Visual Studio Code) haciendo uso de la librería Pandas previamente instalada a través de la terminal y se importa pandas as pd y se llama el dataframe para su visualización y manejo.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df_0 = pd.read_csv(r'C:\Users\ASUS\Desktop\DATA SCIENCE-CURSE\EDA\base_datos_restaurantes_USA_v2 (nuevo).csv')
```

   c. Se realiza una visualización de los mismos para observar visualmente de que se está haciendo referencia. con la función

```
df.head()
df.tail()
```

   d. Se observa una lista con una fila de información tales como: id_persona, nombre, apellido, edad, genero, ciudad_residencial, estrato_socieconomico, frecuencia_visita, promedio_gasto_comida, ocio, consume_licor, preferencia_alimenticias, membresia_premium, telefono_contacto, correo_electronico, tipo_de_pago_mas_usado, ingresos_mensuales.

| | id_persona | nombre | apellido | edad | genero | ciudad_residencia | estrato_socioeconomico | frecuencia_visita | promedio_gasto_comida | ocio | consume_licor | preferencias_alimenticias | membresia_premium | telefono_contacto | correo_electronico | tipo_de_pago_mas_usado | ingresos_mensuales |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 29995 | 4862097674 | Robert | Cortez | 20.0 | Masculino | Houston | Alto | -3 | 30.82 | Si | Si | Vegano | Si | NaN | garciagregory@example.net | Tarjeta | 5781 |
| 29996 | 9458262482 | Michael | Holt | 78.0 | Masculino | Denver | Alto | 5 | 45.04 | No | No | Mariscos | No | NaN | jimmy77@example.org | Efectivo | 7652 |
| 29997 | 3412365931 | Rebecca | Henry | 77.0 | Femenino | San Diego | Muy Alto | 7 | 93.55 | No | Si | Mariscos | Si | NaN | NaN | Efectivo | 12639 |
| 29998 | 8853079811 | Tamara | Griffin | 77.0 | Femenino | Chicago | Bajo | 1 | 6.18 | Si | No | Otro | No | (243)658-6543x11668 | NaN | Tarjeta | 1057 |
| 29999 | 4553644223 | Tracey | Flynn | 57.0 | Femenino | Chicago | Bajo | 1 | 6.96 | No | Si | Carnes | No | 295.679.7926x321 | NaN | Efectivo | 1348 |

3. Exploración inicial (EDA)

*Uso de funciones básicas e iniciales para reconocer que tiene dicho dataframe. entre eso tenemos tamaño de df_0, nombres de categorías, tipo de datos, cantidad de la muestra etc.*

```
df_0.shape #muestra la cantidad de filas y columnas del DataFrame.
```

```
0.0s
(30000, 17)
```

```
df_0.columns #muestra los nombres de las columnas del DataFrame.
```
✓ 0.0s

```
Index(['id_persona', 'nombre', 'apellido', 'edad', 'genero',
       'ciudad_residencia', 'estrato_socioeconomico', 'frecuencia_visita',
       'promedio_gasto_comida', 'ocio', 'consume_licor',
       'preferencias_alimenticias', 'membresia_premium', 'telefono_contacto',
       'correo_electronico', 'tipo_de_pago_mas_usado', 'ingresos_mensuales'],
      dtype='object')
```

```
df_0.dtypes #muestra los tipos de datos de cada columna en el DataFrame.
#Edad es un float64 y debe ser int64 (valor a ajustar en limpieza de datos).
```
✓ 0.0s

| | 0 |
|---|---|
| id_persona | int64 |
| nombre | object |
| apellido | object |
| edad | float64 |
| genero | object |
| ciudad_reside | object |
| estrato_socio | object |
| frecuencia_vis | int64 |
| promedio_ga | float64 |
| ocio | object |

*.info() y .describe() muy utiles para identificar valores nulos*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 17 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   id_persona                 30000 non-null  int64
 1   nombre                     30000 non-null  object
 2   apellido                   30000 non-null  object
 3   edad                       29899 non-null  float64
 4   genero                     30000 non-null  object
 5   ciudad_residencia          30000 non-null  object
 6   estrato_socioeconomico     30000 non-null  object
 7   frecuencia_visita          30000 non-null  int64
 8   promedio_gasto_comida      29855 non-null  float64
 9   ocio                       30000 non-null  object
 10  consume_licor              30000 non-null  object
 11  preferencias_alimenticias  28597 non-null  object
 12  membresia_premium          30000 non-null  object
 13  telefono_contacto          14834 non-null  object
 14  correo_electronico         14928 non-null  object
 15  tipo_de_pago_mas_usado     30000 non-null  object
 16  ingresos_mensuales         30000 non-null  int64
dtypes: float64(2), int64(3), object(12)
memory usage: 3.9+ MB
```

*reconociendo los duplicados con un* `df_0.duplicated().sum()` *que suma el numero de filas duplicadas en el data frame. Se destaca que este data frame no tiene filas duplicadas.*

*4. Limpieza de datos por categoría.*

*Se toma la decisión de realizar una limpieza y manejo de cada columna de categorías.*

4.1 CATEGORIA-EDAD: Dado que la variable edad presenta valores extremos y no constituye el foco principal del análisis, se optó por la imputación mediante la mediana, al ser una medida robusta frente a outliers y errores de registro.

```
df_0 ['edad'].describe() #muestra estadísticas descriptivas de la columna 'edad'.
```

✓ 0.0s

| | # edad |
|---|---|
| count | 29899.0 |
| mean | 49.665005518579214 |
| std | 23.839549601481316 |
| min | -5.0 |
| 25% | 33.0 |
| 50% | 49.0 |
| 75% | 65.0 |
| max | 300.0 |

se perciben valores negativos y edades de 300 años, que no son coherentes con la realidad.

```
(df_0['edad'] >100).sum() #cuenta cuántos valores mayores a 100 hay en la columna 'edad'.
```

✓ 0.0s

```
np.int64(110)
```

```
(df_0['edad'] <0).sum() #cuenta cuántos valores menores a 0 hay en la columna 'edad'.
```

✓ 0.0s

```
np.int64(97)
```

Se selecciona un rango coherente

```
edad_valida = (
    df_0['edad'].notna() &
    (df_0['edad'] >= 0) &
    (df_0['edad'] <= 100)
)
```

✓ 0.0s

```
mediana_edad = df_0.loc[edad_valida, 'edad'].median()
print ("Mediana de edad válida:", mediana_edad)
```

✓ 0.0s

```
Mediana de edad válida: 49.0
```

Se carga la imputación al data frame de la edad con la mediana. ya que este no es una variable de análisis fuerte no es agresivo realizar una mediana en este campo.

4.2 CATEGORIA-Frecuencia visitas

De igual manera se verifican valores nulos, si hay negativos u otros factores que no le den coherencia al campo de datos.

```
df_0['frecuencia_visita'].describe()
```

0.0s

| | # frecuencia_visita |
|---|---|
| count | 30000.0 |
| mean | 3.8961333333333332 |
| std | 2.741531612452236 |
| min | -3.0 |
| 25% | 2.0 |
| 50% | 4.0 |
| 75% | 5.0 |
| max | 10.0 |

Cantidad de valores negativos sobre el 5% tratable con imputación de la mediana.

```
frecuencia_valida = ( df_0['frecuencia_visita'].notna() & (df_0['frecuencia_visita'] >= 0) )
mediana_frecuencia = df_0.loc[frecuencia_valida, 'frecuencia_visita'].median()
print("Mediana de frecuencia de visita válida:", mediana_frecuencia)
```

0.0s

```
Mediana de frecuencia de visita válida: 4.0
```

se crea un frecuencia valida como mascara booleana para identificar valores validos, los no negativos. luego se calcula la mediana con .median()

```
#imputa los valores no válidos en la columna 'frecuencia_visita' con la mediana calculada anteriormente.
df=df_0.copy() # Crea una copia del DataFrame df_0 para evitar modificar el original.
df.loc[~frecuencia_valida, 'frecuencia_visita'] = mediana_frecuencia  # Asigna la mediana a los valores no válidos.
df['frecuencia_visita'] = df['frecuencia_visita'].round().astype('int64')
```

se crea una copia por el manejo de los cambios que se han venido dando. y se introduce la mediana a los valores no válidos, finalmente se redondea para tener valores enteros.

4..3 PROMEDIO GASTO COMIDA: Para análisis de correlación es preferible trabajar únicamente con valores observados reales, evitando introducir valores imputados que puedan distorsionar la relación entre variables. a demás de ellos los valores nulos corresponden a 145 registros es <1% de la muestra.

```
df['promedio_gasto_comida'].describe()
```

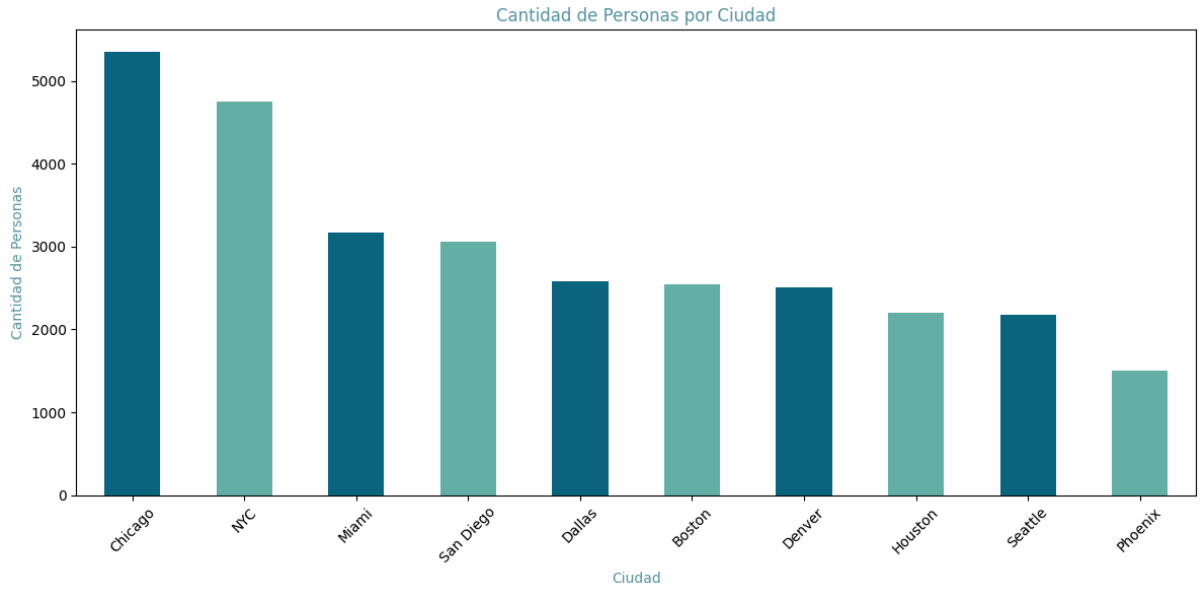
0.0s

| | # promedio_gasto_comida |
|---|---|
| count | 29855.0 |
| mean | 32.60345168313515 |
| std | 26.402600946984904 |
| min | 0.0 |
| 25% | 13.29 |
| 50% | 25.51 |
| 75% | 44.4 |
| max | 149.97 |

La variable promedio de gasto en comida constituye un indicador económico central del análisis. Por este motivo, los registros con valores nulos fueron excluidos, ya que la imputación de esta variable implicaría introducir supuestos no verificables y podría distorsionar las relaciones estadísticas y económicas evaluadas. El análisis se realizó únicamente sobre valores observados reales, garantizando la integridad y coherencia de los resultados. Se refuerza esta decisión con el porcentaje de nulos que es menos del 0.5% del dataset.
Creación de un nuevo dataframe (df_1) como resultado de una eliminación de nulos.

4.4 Categorías-Telefono y correo
> Las variables correo electrónico y teléfono no fueron consideradas dentro del análisis estadístico, ya que no constituyen variables analíticas ni aportan información cuantitativa o categórica relevante para la generación de correlaciones, segmentaciones o inferencias económicas. Su inclusión en los modelos analíticos no genera valor estadístico y podría inducir interpretaciones erróneas.
> No obstante, dichas variables no fueron eliminadas del dataset, pese a presentar una alta proporción de valores nulos y errores en el registro. La razón es que representan información de contacto con alto valor estratégico desde una perspectiva de negocio, particularmente para el diseño y ejecución de campañas de marketing, activaciones comerciales o acciones de fidelización por parte del cliente final.
> En este sentido, aunque podrían haberse eliminado sin afectar los resultados del análisis, se decidió preservarlas como atributos informativos, permitiendo que el cliente disponga de datos potencialmente útiles para futuras iniciativas operativas o comerciales, sin comprometer la integridad del análisis estadístico realizado.

5. Visualizaciones

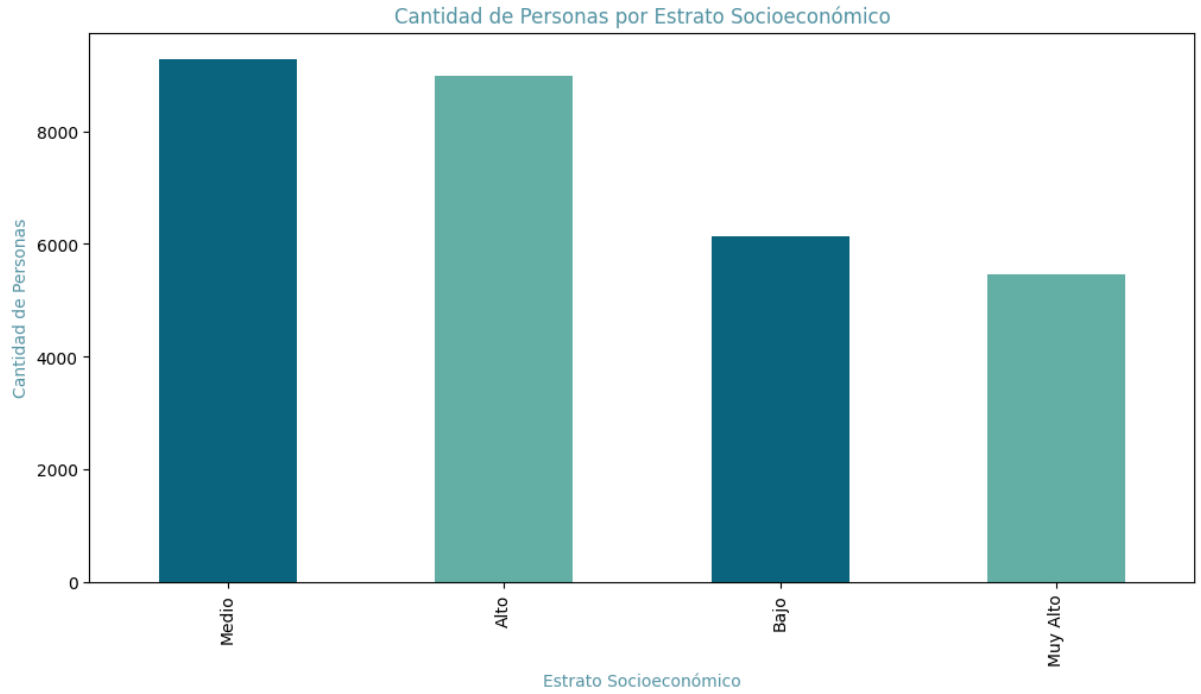



>Pese a que Miami tiene una menor cantidad de personas por ciudad posee mayor frecuencia de visitas a restaurantes y gasto promedio mayor por visita. adicional a esto miami es conocido por ser un sitio turístico por ende su gastronomía es más internacional y ciudades como chicago son más de estilo tradicional.

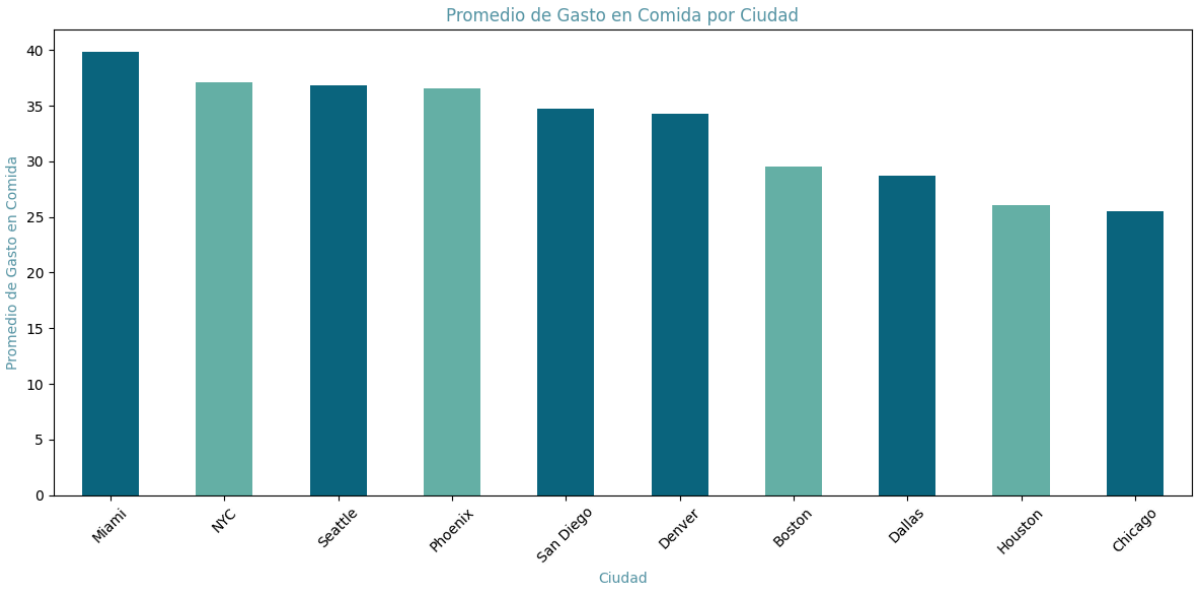
CÓDIGO

```
ciudad_counts = df_1['ciudad_residencia'].value_counts()
plt.figure(figsize=(12, 6))
ciudad_counts.plot(kind='bar', color= ['#0c677e','#64b2a7'])
plt.xlabel('Ciudad',color='#5897a7')
plt.ylabel('Cantidad de Personas',color='#5897a7')
plt.title('Cantidad de Personas por Ciudad', color='#5897a7')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

El código cuenta la cantidad de valores “personas” en la columna que se llama del df_1 ciudad residencia y crea a traves de la libreria de matplot un diagrama de barras (kind=’bar’) para los colores se decidio establecer los mismos en una paleta de color del logo creado. los demas son las etiquetas que se desean poner en los ejes y de que color, xticks me da la rotación de las etiquetas. Similarmente se manejaron todas las visualizaciones de aqui en adelante.




> la población se concentra en estratos Medio-Alto (intermedios) y menor representación en estratos bajo y muy altos. Clase media dominante y el estrato bajo representa aprox 20% del total siendo el tercer grupo más grande.

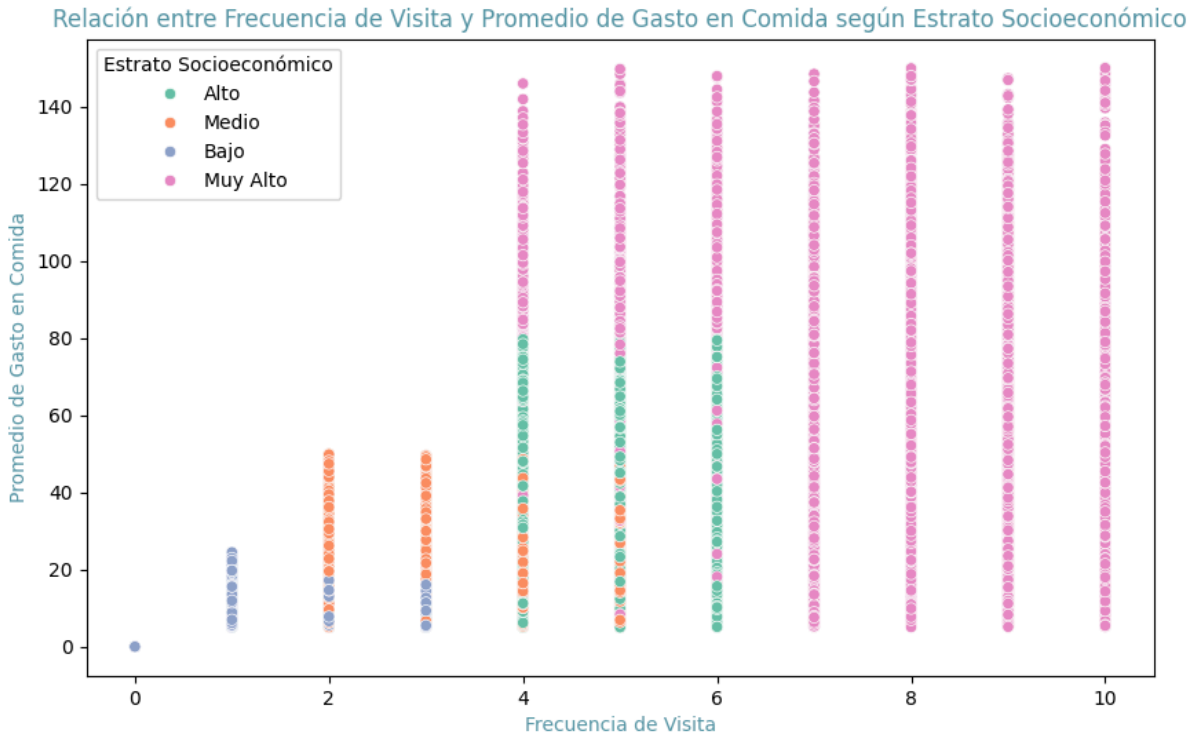



> Miami con el promedio de gasto más alto razón por la cual se tomó como ciudad de análisis. contrastando con Chicago que pese a que tiene la mayor población representa el menor gasto haciéndolo menos atractivo como potencial económico para alguna propuesta.

CODIGO

```
ciudad_gasto = df_1.groupby('ciudad_residencia')['promedio_gasto_comida'].mean().sort_values(ascending=False)
```

Se realiza un groupby para agrupar tanto por promedio de gasto y ciudad y se ordenan de mayor a menor.




> hallazgo nivel socioeconómico muy alto gastan más y van con más frecuencia. En coherencia los de bajo nivel socioeconómico gastan menos y frecuentan menos. Los

estratos alto no superan el 80 en promedio de gasto y su frecuencia esta entre el 4 y 6 visitas al mes.

El gráfico muestra una segmentación definida por estrato, pues más o menos cada estrato ocupa una zona relativamente distinta del gráfico. con una correlación de que a mayor estrato socioeconómico mayor frecuencia y mayor gasto. Brecha socioeconómica: Diferencia significativa entre estratos bajo y muy alto (hasta 10x en gasto).

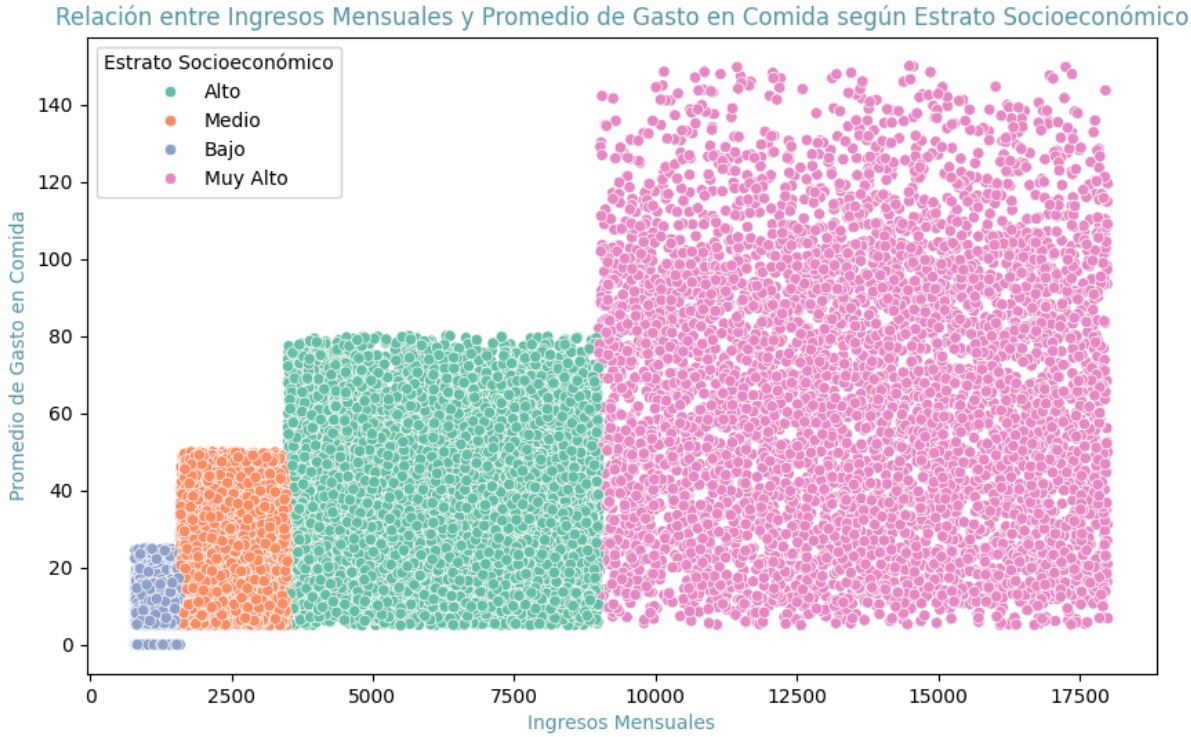
CODIGO

```
sns.scatterplot(data=df_1, x='frecuencia_visita', y='promedio_gasto_comida', hue='estrato_socioeconomico', palette='Set2')
```

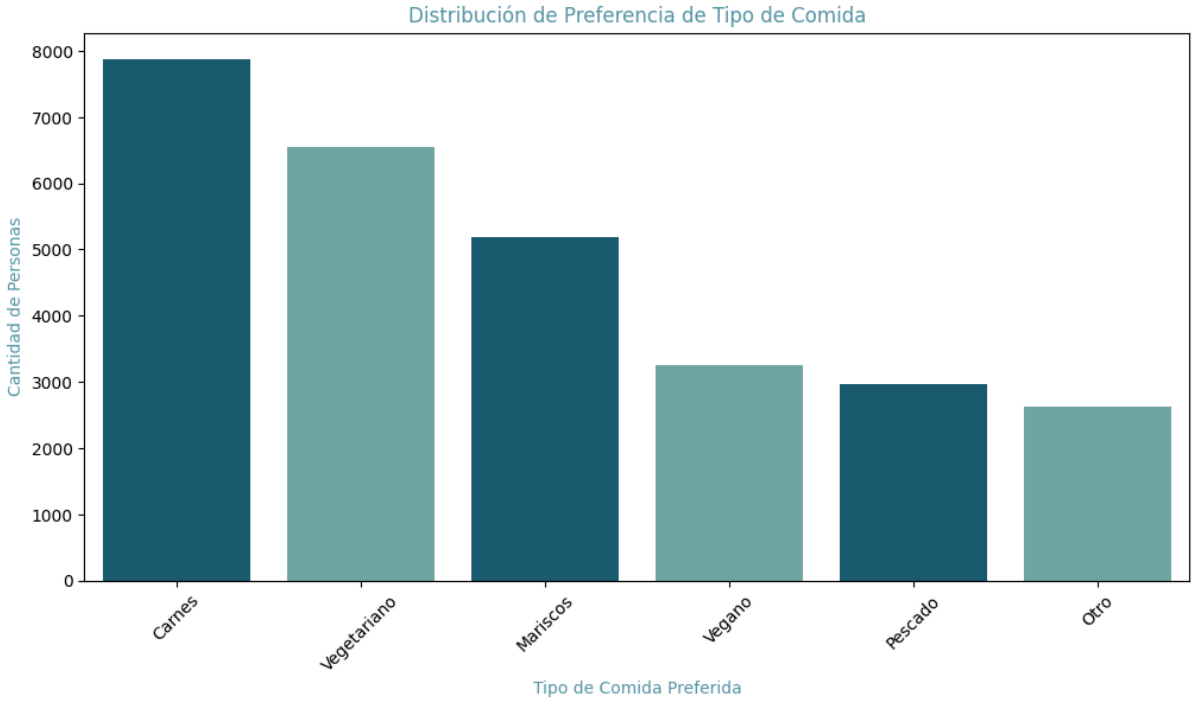
Se destaca el uso de la librería seaborn llamado desde el inicio del código.

```
import seaborn as sns
```

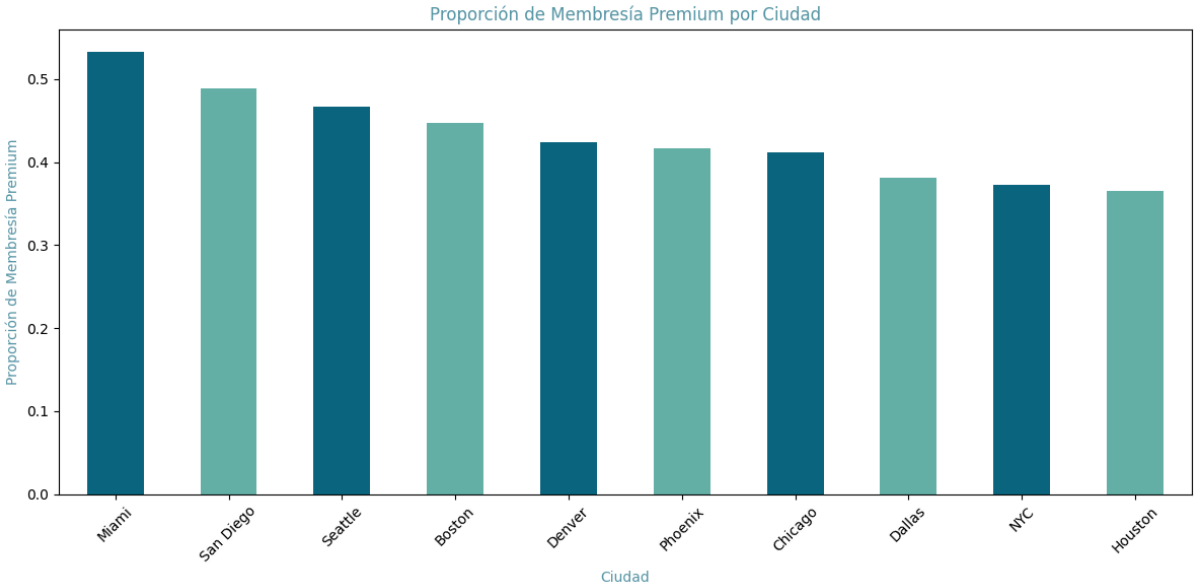
y el tipo de gráfico definido es scatterplot es decir diagrama de dispersión para relacionar varias variables-frecuencia, socioeconómico y gasto. o este otro




> Gráfico representando que los estratos bajos con ingresos bajos realizan gastos entre 0-25 dolares (bajos) y así en escalera y dimensionamiento refiriéndose a que los estratos altos tienen mayor rango de gasto y los volúmenes de la gráfica si hacen discrepancia a que fuese una distribución poblacional real, posiblemente este dataset está dado de datos sintéticos armado a conveniencia.




Existe una tendencia clara hacia opciones saludables Las preferencias del consumidor muestran una inclinación significativa hacia opciones vegetarianas, veganas y proteínas ligeras, integradas de forma transversal en categorías de alto volumen (carnes).



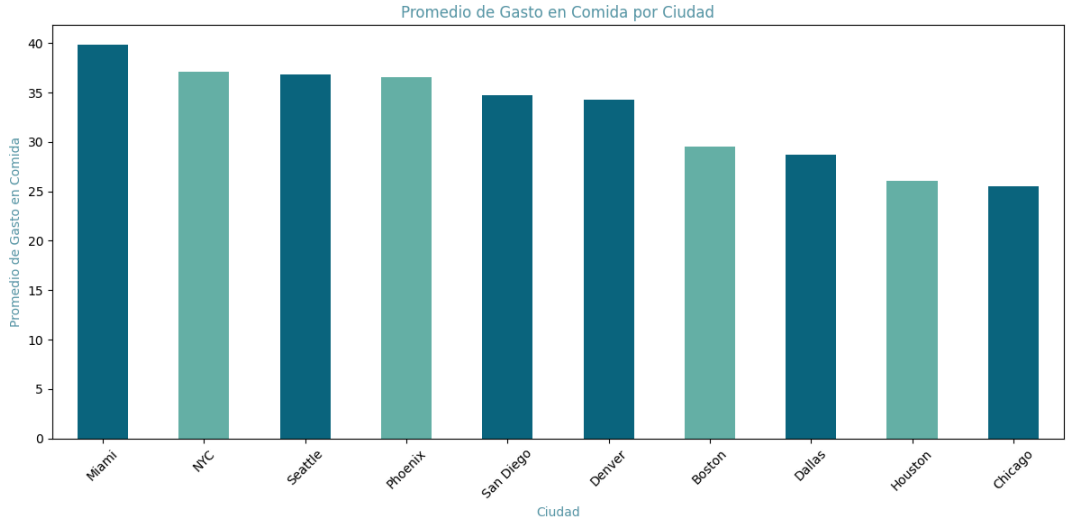

Codigo

```
#definir columna binaria de membresia_premium
df_1.loc[:, 'membresia_premium_binaria'] = np.where(df_1['membresia_premium'] == 'Sí', 1, 0)
```

El reto en esta parte fue generar una columna binaria de 1 para los ‘Sí’ y 0 para los ‘No’ para poder realizar un conteo

.

5. Filtrado por ciudad (Análisis)




> Toma de decisión ciudad MIAMI por tener el mayor gasto promedio por cliente y frecuencia de visitas. se diferencia con Chicago pues aunque esta ciudad tiene el mayor número de registros es en contraste la de menor atractivo económico porque en el gráfico anterior es la de menor gasto promedio.
se continuará la limpieza de datos y análisis en base a esta ciudad. #insight: La ciudad de Miami destaca por tener el mayor gasto promedio por cliente y una alta frecuencia de visitas a restaurantes, lo que la convierte en un mercado atractivo para estrategias de marketing y expansión de negocios en el sector gastronómico.

```
df_miami = df_1[df_1['ciudad_residencia'] == 'Miami']
```

Se trabaja de ahora en adelante con el data frame filtrado (df_miami) del data frame limpiado (df_1) por ende este dataframe de miami ya se encuentra en un estado de uso para análisis.

El ejercicio nos solicita filtrar inicialmente por una ciudad; sin embargo, esta elección no se realiza de manera aleatoria. Los criterios utilizados para aplicar este filtro se definieron con base en variables cuantitativas, priorizando una ciudad con un alto número de registros y un gasto promedio elevado en consumo de alimentos. Esto permite garantizar tanto la representatividad estadística de la muestra como la relevancia económica del análisis. En consecuencia, se asegura que la muestra cuente con suficientes observaciones para obtener correlaciones más estables y, a su vez, facilitar el análisis de ingresos y la identificación de clientes de alto valor (premium), con el objetivo de generar un impacto económico significativo.

| ciudad_residencia | # clientes | # gasto_promedio | # frecuencia |
|---|---|---|---|
| Missing: 0 (0%) Distinct: 10 (100%) | Missing: 0 (0%) Distinct: 10 (100%) | Missing: 0 (0%) Distinct: 10 (100%) | Missing: 0 (0%) Distinct: 10 (100%) |
| Miami 10% NYC 10% Seattle 10% Other 70% | Min 1511 Max 5384 | Min 25.531781641... Max 39.82154258... | Min 3.6030614384... Max 4.358757062... |
| Miami | 3186 | 39.82154258675079 | 4.358757062146893 |
| NYC | 4769 | 37.123200084157375 | 3.6030614384566997 |
| Seattle | 2191 | 36.86178358551123 | 4.03149246919215 |
| Phoenix | 1511 | 36.54477045908184 | 3.843150231634679 |
| San Diego | 3075 | 34.75153871283894 | 4.264065040650406 |
| Denver | 2523 | 34.26409561752988 | 3.896551724137931 |
| Boston | 2547 | 29.494765840220385 | 3.9807616804083237 |
| Dallas | 2602 | 28.695237727097023 | 3.66141429669485 |
| Houston | 2212 | 26.020559090909092 | 3.60623869801085 |
| Chicago | 5384 | 25.531781641428303 | 3.823922734026746 |

Aunque Miami no presenta el mayor número de registros, su selección se justifica por exhibir los valores más altos en gasto promedio y frecuencia de consumo. Este enfoque permite analizar patrones asociados a un consumo más intensivo y de mayor valor económico. Asimismo, este perfil resulta especialmente adecuado para correlacionar el comportamiento de los consumidores con variables externas provenientes de una API de restaurantes, como el precio y la calificación de los establecimientos, maximizando la relevancia del análisis.

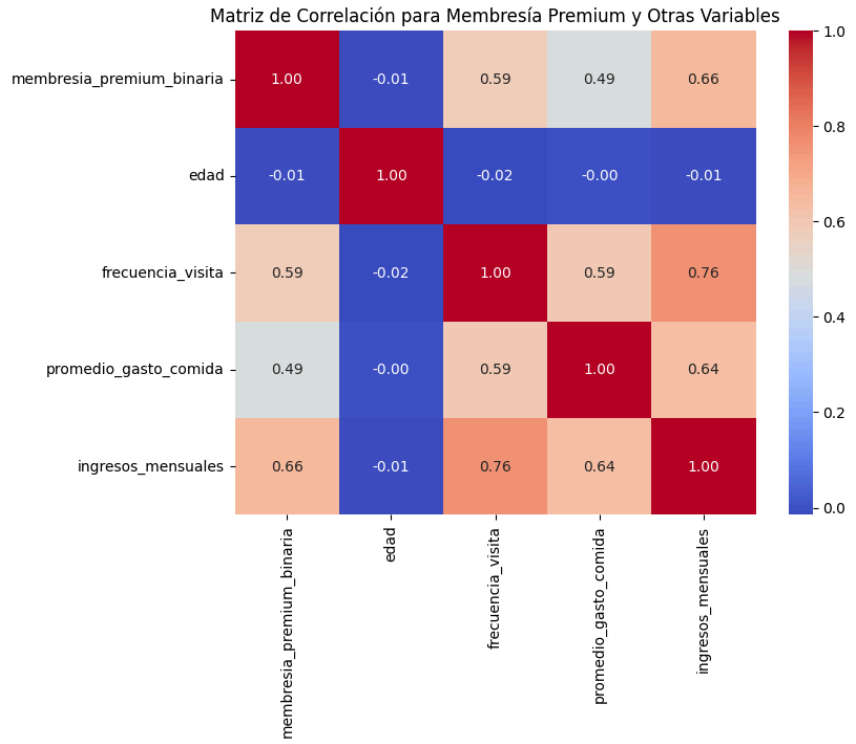



gráfico heatmap, **matriz de correlación** analiza cómo la **membresía premium (binaria)** se relaciona con variables clave de comportamiento y capacidad económica del cliente.

- No existe relación significativa entre edad y membresía premium.
- Clientes premium visitan con mayor frecuencia.
- La recurrencia es un factor determinante para pertenecer al segmento premium.

Membresía Premium vs Gasto Promedio → 0.49 (moderada)

- Existe una relación positiva clara.
- Clientes premium gastan más por visita, pero no es el único factor.

Membresía Premium vs Ingresos Mensuales → 0.66 (fuerte)

- Relación sólida entre capacidad económica y membresía.
- Clientes con mayores ingresos tienen más probabilidad de ser premium.

Frecuencia vs Gasto Promedio → 0.59 (moderada-alta)

- Clientes frecuentes también tienden a gastar más.
- Se refuerza el perfil de cliente recurrente + rentable.

Recomendaciones derivadas (Para mayor detalle ver archivo RECOMENDACIONES)

-Crear programas premium basados en frecuencia de visitas.
-Identificar clientes no premium con alta frecuencia + alto gasto (candidatos naturales).
-Evitar segmentación por edad.
Segmentación de mercado:
Estratos bajos: Menús económicos, promociones
Estratos altos: Experiencias premium, servicio diferenciado
Programas de fidelización:
Estratos muy altos visitan hasta 10 veces → Programas VIP
Estratos bajos visitan poco → Incentivos para aumentar frecuencia
Pricing strategy:
Estratos altos toleran precios de $40-150
Estratos bajos buscan opciones bajo $25

## AVANCE #2

1. Conexión código Pre-armado Henry

Conectarse a la API de Yelp para obtener información de negocios locales de una ciudad (elegida de la base de datos del avance anterior). Como ya se definió en el avance uno la ciudad elegida fue MIAMI. A continuación se expresa el código pre-armado de Henry para trabajar como parte del avance.

2.1 Seguridad (Variable de entorno al APIKEY)

> la apikey es la autenticación real no requiere del cliente ID. como recomendación de seguridad, no se comparte esta clave públicamente, para ello se usan variables de entorno o archivos de configuración seguros. mas adelante se explicara como se realizo este tratamiento.

Es recomendable que los API key no esten expuestos, pues un codigo que se comparta publicamente incurre en problemas de seguridad. a demas consume los limites en el yelp o API y puede llegar a ser bloqueada. para ello es recomendable gestionarla a traves de variables de entorno el sistema operativo, evitando su exposición directa y cumpliendo con buenas prácticas de seguridad en el consumo de APIs.

```
import os # Importar el módulo os para manejar variables de entorno
apikey=os.getenv('YELP_API_KEY') # Obtener la clave API desde una variable de entorno por seguridad
print(apikey is not None) # Verificar que la clave API se haya obtenido correctamente
```

✓ 0.0s

```
response.status_code
```

✓ 0.0s

```
200
```

200 ES PETICIÓN EXITOSA

Confirmando que la peticion con la modificacion del apikey por variable de entorno salio correctamente.

2.2 Extracción de datos relevantes

Se utilizó la Yelp API para recolectar información de restaurantes, incluyendo:
Nombre del restaurante, Categorías gastronómicas, Rango de precios, Ubicación, Rating, Número de reseñas.

La API presenta un límite de 50 resultados por request y un máximo de 1000 registros, por lo que se implementó paginación con offset para obtener el mayor volumen de datos posible.

Proceso de Recolección de Datos

- Se realizaron múltiples llamadas a la API usando paginación (offset).
- Se validó el límite máximo de registros entregados por la API.
- Los datos fueron almacenados en un DataFrame para su posterior análisis.

Se unificaron las respuestas del endpoint en una sola estructura tabular.

para ello se extraer información relevante de cada negocio y almacenarla en una lista

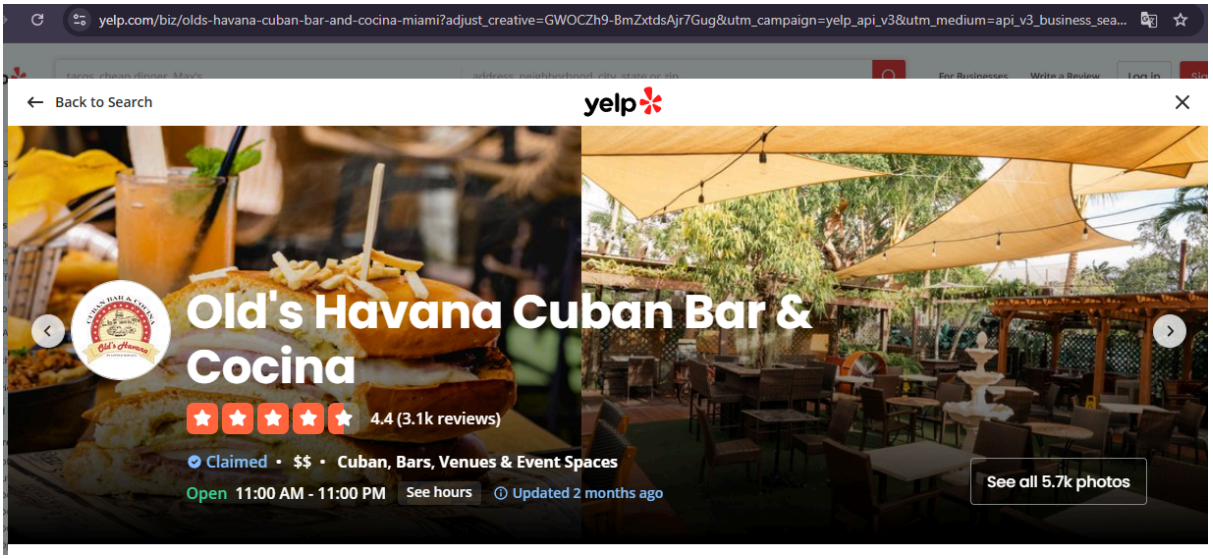
Conocer los limites diarios, llamadas y tiempo de reseteo

```
print("Límite diario:", response.headers.get('RateLimit-DailyLimit')) # Muestra el límite diario de llamadas a la API
print("Llamadas restantes:", response.headers.get('RateLimit-Remaining')) # Muestra las llamadas restantes permitidas
print("Reset time:", response.headers.get('RateLimit-ResetTime')) # Muestra el tiempo de reinicio del límite de llamadas
```

✓ 0.0s

```
Límite diario: 5000
Llamadas restantes: 4732
Reset time: 2026-02-06T00:00:00+00:00
```

2.2.1 Aproximación ejm de respuesta de la API




URL:https://www.yelp.com/biz/salty-flame-miami?adjust_creative=GWOCZh9-BmZxtdsAjr7Gug&utm_campaign=yelp_api_v3&utm_medium=api_v3_business_search&utm_source=GWOCZh9-BmZxtdsAjr7Gug&dd_referrer=

La respuesta del endpoint `/businesses/search` fue validada exitosamente mediante la verificación del código de estado HTTP, la estructura del JSON devuelto y la consistencia entre el límite solicitado y los resultados obtenidos. Asimismo, se confirmó la disponibilidad de variables clave como rating, rango de precios y ubicación, necesarias para el análisis posterior.

El error en el codigo por llamar a más de 240.

```
{'error': {'code': 'VALIDATION_ERROR', 'description': 'Too many results requested, limit+offset must be <= 240.'}}
```

Se define reajustar el código para limitar a 40 por paginación y el bucle pare adecuadamente

```
Obteniendo resultados desde 0 hasta 50...
Total de negocios analizados hasta ahora: 40

Obteniendo resultados desde 40 hasta 90...
Total de negocios analizados hasta ahora: 80

Obteniendo resultados desde 80 hasta 130...
Total de negocios analizados hasta ahora: 120
```

Primera Paginación (Primeros 50)

```python
API_NEGOCIOS_ANALIZADOS = []  # Lista para almacenar los negocios analizados
for offset in range(0, 240, 40):  # Limitar a 240 resultados debido a la restricción de la API de Yelp
    params={'term':'restaurants','location': ciudad,'limit':40, 'offset': offset}
    print(f"\nObteniendo resultados desde {offset} hasta {offset + 40}...")

    response=requests.get(api_url,params=params,headers=headers)

    if response.status_code != 200: # Verificar si la solicitud fue exitosa
        print(f"Error en la solicitud: {response.status_code}")
        print(response.json())
        break  # Salir del bucle si hay un error

    data=response.json()

    businesses=data.get('businesses', [])

    if not businesses:
        print("No hay más negocios para procesar.")
        break  # Salir del bucle si no hay más negocios

    API_NEGOCIOS_ANALIZADOS.extend(businesses)
    print(f"Obtenidos: {len(businesses)} negocios")
    print(f"Total acumulado: {len(API_NEGOCIOS_ANALIZADOS)}")
    time.sleep(0.2)  # Pausa para respetar los límites de la API
    print(f"Total de negocios analizados hasta ahora: {len(API_NEGOCIOS_ANALIZADOS)}")
```

Código inicial para extraer los primeros 50 locales y bucle para obtener en total 240 primeros. se crea una lista de almacenamiento y un offset como paginación cada 40, también se establecen indicaciones de advertencia como código 200 por error en la solicitud y un conteo final para saber cuántos negocios se captaron.

Se vio la necesidad de generar más datos a través de criterios, extraer de forma sistemática restaurantes desde la API de Yelp, considerando múltiples criterios de ordenamiento y segmentación de precios, con el fin de:

Maximizar la cobertura del mercado gastronómico

Reducir el sesgo por un único criterio de búsqueda

Obtener un dataset más representativo y diverso

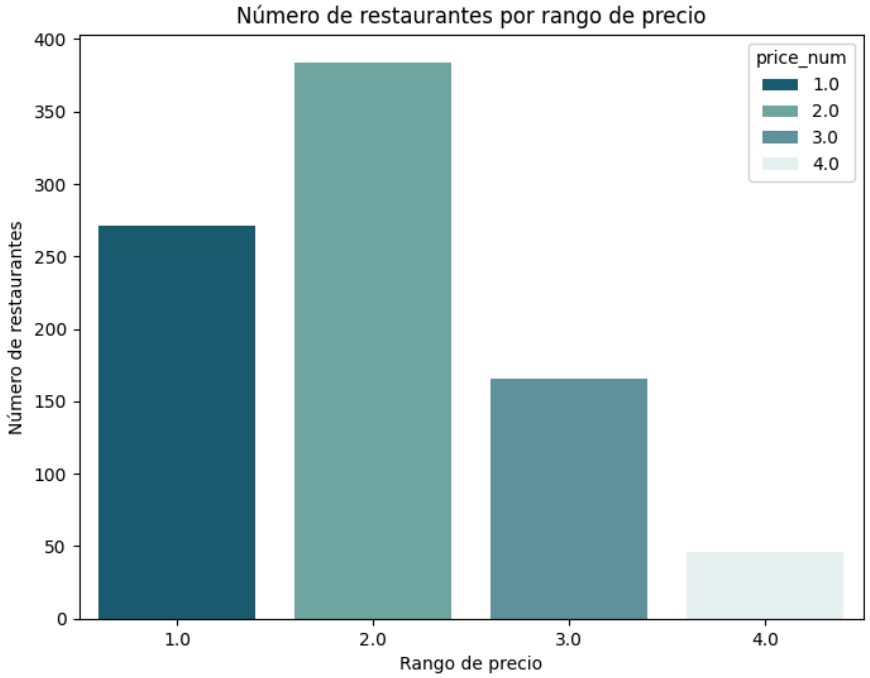
Yelp no permite ordenar por precio directamente, por lo que los rangos de precio deben tratarse como un **filtro independiente**, lo que justifica su manejo separado dentro del flujo de consulta.

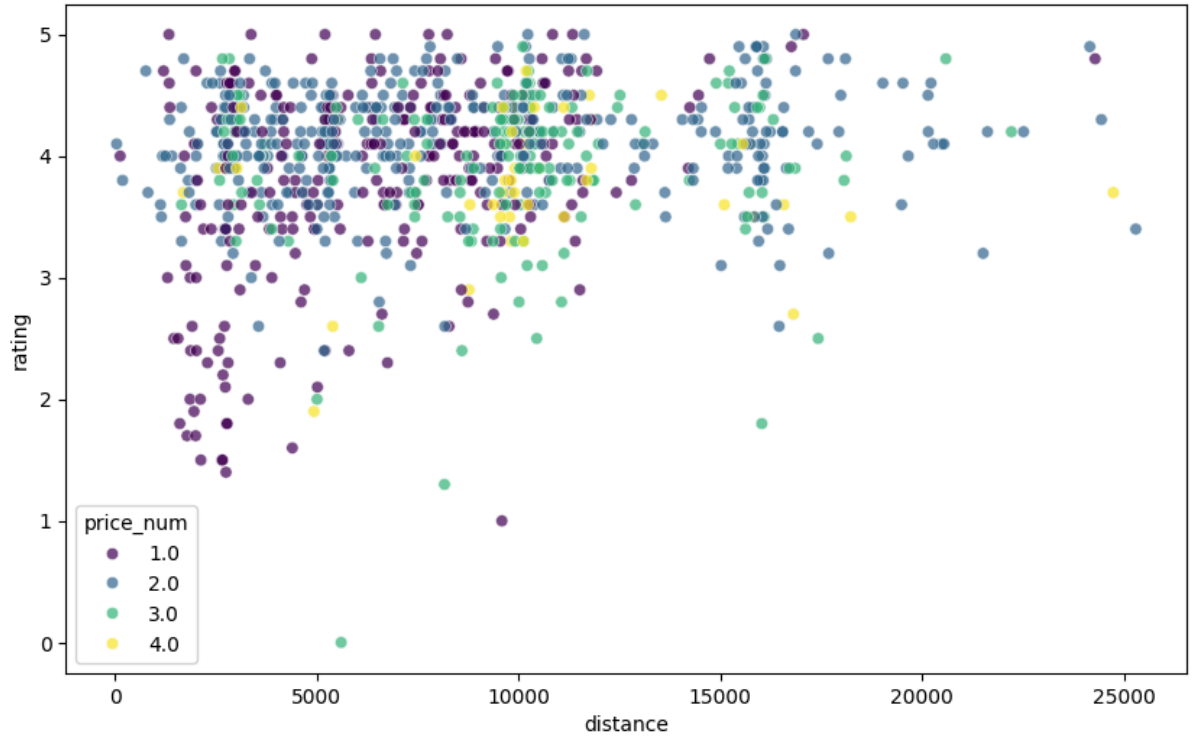
el codigo funciona con bucles en donde cada criterio se consulta independiente, capta un subconjunto distinto, genera una lista temporal por criterio y finalmente la almacena en nuestra lista principal-all_businesses.
tiene un corte por ausencia de resultados que es el break Esto indica que:

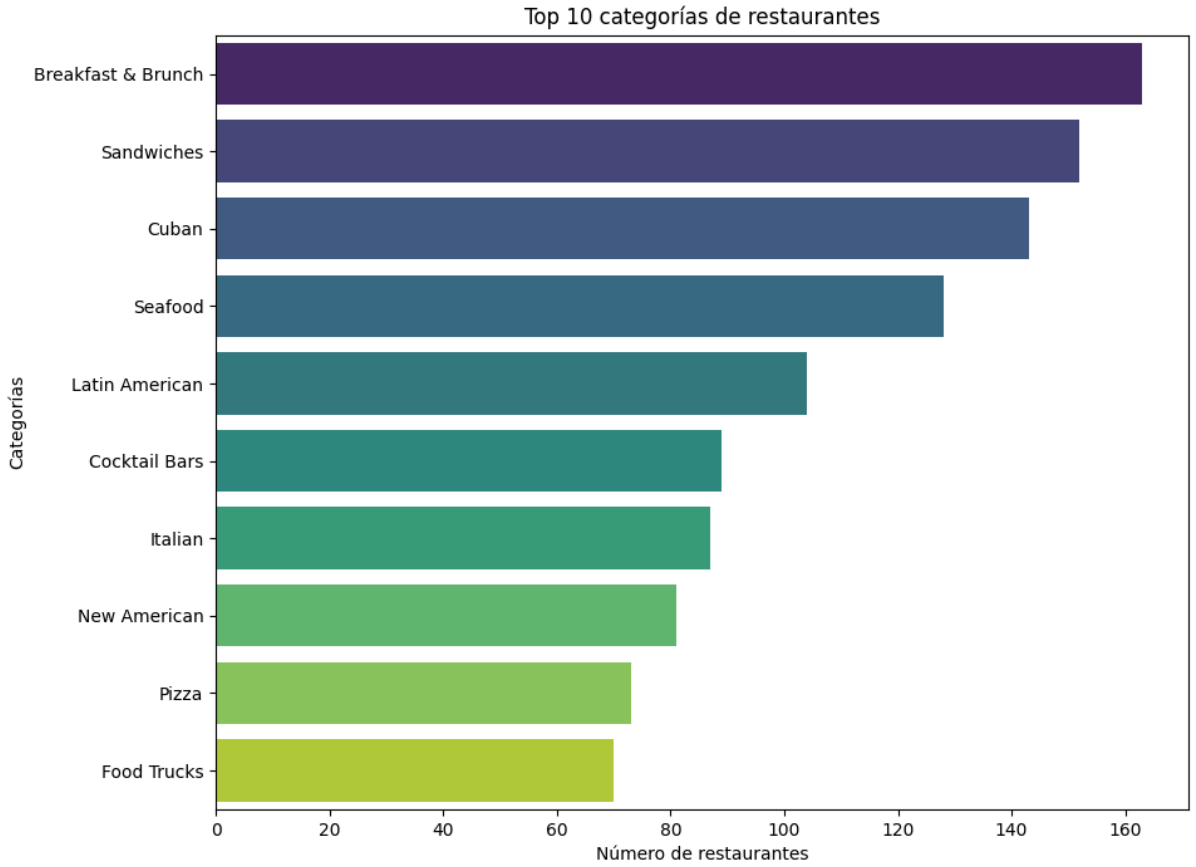
- Ya no hay más restaurantes disponibles para ese criterio
- Se evita hacer solicitudes innecesarias

```
No hay más negocios para procesar.
Total acumulado global: 1670
```

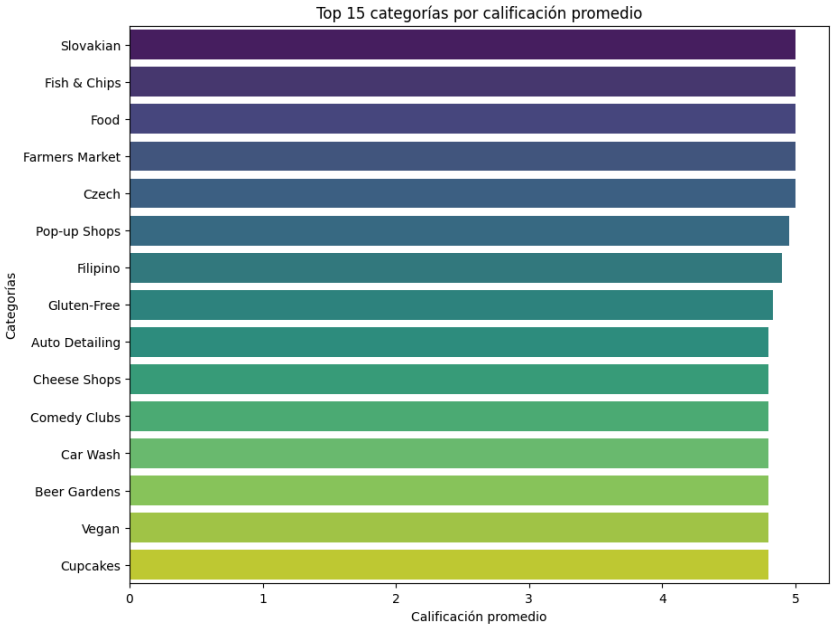
Como resultado 1670 negocios captados a los cuales esta lista se convirtió a dataframe de pandas y se le realizo un proceso de exploración de datos EDA similar al mencionado anteriormente, se puede evidenciar en el codigo el tratamiento a los datos tipo diccionario como las categorias, las direcciones y la localización que se lograron dividir en columnas separadas para poder trabajar mejor en su analisis. a continuación las visualizaciones mas relevantes






> El mercado esta dominado por comida de tipo breakfast and brunch, seguramente tendencia cultural y turismo asi como una fuerte presencia de comida latina especialmente de la gastronomia cubana. como cuarto lugar esta la comida de mar por la ubicación de miami costera asi como los bares por la vida nocturna turistica.  como estrategia se puede optar por nicho especifico ejm cuban bruch o comida mexicana posibles tendencias de salud vegana o vegetariana, nichos seguros con poca competencia como la comida italiana o hibridos con licores.



AVANCE #3. Analizar los datos integrados (clientes, Yelp, tendencias externas) para identificar patrones relevantes que puedan guiar decisiones de marketing. Formular preguntas clave del negocio y responderlas con visualizaciones y análisis descriptivo.

## 3 Análisis del Mercado Gastronómico (Yelp)

Predominan restaurantes de precio medio (2 y 3), lo que indica un mercado amplio pero con fuerte presencia de consumidores con poder adquisitivo medio–alto.

Categorías más frecuentes:
Breakfast & Brunch
Sandwiches
Cuban
Seafood
Cocktail Bars
Este patrón refuerza el perfil de Miami como ciudad de consumo premium y experiencial.

### 3.1 Tendencias Alimenticias

A partir de las preferencias de los clientes:

Aunque el consumo de carnes sigue siendo dominante, se observa una alta participación de opciones vegetarianas y veganas.
La presencia de categorías saludables sugiere una tendencia creciente hacia estilos de vida health & wellness.

### 3.2 Análisis del Cliente Premium

Mediante matrices de correlación se identificó que: La edad no tiene relación significativa con la membresía premium. La membresía premium está fuertemente asociada con: Frecuencia de visita
Ingresos mensuales
Gasto promedio por comida
* El cliente premium se define por su comportamiento y capacidad económica, no por factores demográficos.

### 3.3 Insights Clave del Proyecto

- Miami presenta un mercado gastronómico con fuerte orientación premium.
- La frecuencia de visita es el principal driver de valor del cliente.
- Los ingresos mensuales influyen directamente en el consumo recurrente.
- Existe una tendencia clara hacia opciones saludables y veganas.
- El perfil premium no depende de la edad del consumidor.

### 3.4 Recomendaciones Estratégicas

Diseñar programas de fidelización basados en frecuencia, no en edad.
Identificar clientes con alta frecuencia y gasto para convertirlos en premium.
Priorizar campañas en restaurantes de precio medio–alto.
Impulsar alianzas con restaurantes de comida saludable y vegana.
Enfocar estrategias de marketing en experiencias, no solo en descuentos.

## 4. Conclusión

El mercado gastronómico de Miami se caracteriza por una alta frecuencia de consumo, una estructura de precios equilibrada y una creciente preferencia por experiencias y alimentación saludable. El valor económico se concentra en clientes recurrentes más que en variables demográficas como la edad, lo que resalta la importancia de estrategias orientadas a la fidelización y la experiencia del cliente. En este contexto, las oportunidades de mayor impacto se encuentran en la combinación de calidad, reputación y propuestas flexibles que permitan capturar segmentos de consumo medio–alto y premium de forma sostenible.

1. Miami es un mercado gastronómico económicamente atractivo
   La ciudad presenta altos niveles de gasto promedio y frecuencia de consumo, lo que la convierte en un entorno ideal para analizar estrategias orientadas a la captura de valor y clientes premium.

2. La edad no es un determinante del comportamiento de consumo
   El análisis de correlación muestra que la edad no guarda relación significativa con el gasto, la frecuencia ni los ingresos, lo que valida su exclusión como variable prioritaria en la segmentación.

3. La frecuencia de consumo es el principal driver de ingresos
   Existe una relación fuerte entre la frecuencia de visitas y los ingresos mensuales, indicando que los clientes recurrentes generan mayor valor que aquellos con consumo ocasional, incluso si su gasto unitario es

alto.

4. El gasto promedio en comida complementa, pero no sustituye, la frecuencia
   El gasto y la frecuencia están positivamente relacionados, pero el mayor impacto económico se alcanza cuando ambas variables se combinan, definiendo al cliente premium real.

5. El mercado presenta una estructura de precios equilibrada
   Predominan los restaurantes de precio medio, con una presencia relevante del segmento medio–alto y un nicho premium reducido pero estratégico, lo que refleja una capacidad real de absorción de precios elevados.

6. La reputación es un factor clave de diferenciación
   Los restaurantes con mejores calificaciones y mayor volumen de reseñas concentran la demanda, evidenciando que la calidad percibida pesa más que el precio en la decisión del consumidor.

7. Existe una tendencia clara hacia el consumo saludable
   Las preferencias del consumidor muestran una inclinación significativa hacia opciones vegetarianas, veganas y proteínas ligeras, lo que confirma una transición hacia estilos de alimentación más saludables y conscientes.

8. La tendencia health no es un nicho aislado, sino transversal
   Esta preferencia se integra dentro de categorías de alto volumen como Breakfast & Brunch, New American y Seafood, lo que permite escalar propuestas saludables sin limitarse a restaurantes especializados.

9. La oferta y la demanda presentan coherencia estratégica
   Los datos del consumidor y la información obtenida de la API de Yelp muestran alineación entre preferencias alimentarias y la estructura del mercado, reduciendo el riesgo de desajustes estratégicos.

10. El análisis basado en datos fortalece la toma de decisiones
    La integración de fuentes internas y externas permitió generar insights accionables, superando decisiones basadas en intuición y proporcionando una base sólida para estrategias comerciales y de posicionamiento.