Datos Abiertos de Airbnb en la Ciudad de Nueva York

Reto 1

Analizaremos datos de Airbnb en Nueva York para evaluar la viabilidad de fijar precios entre 25 y 1,900 USD, aplicar descuentos en estadías largas y unificar tarifas entre Brooklyn y Manhattan. Estas decisiones provienen del equipo comercial sin respaldo en datos, por lo que realizaremos un análisis para validar su efectividad y propondremos una visualización clara y contundente.

Análisis realizado por: Pedro Congo

Buscame en mis redes

LINKEDIN HUGGING FACE GITHUB

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
```

Loading the data

In [2]:
```
df = pd.read_csv(r"D:\Users\Usuario\Documents\python 2021\bi-insights\AB_NYC_2019.c
# Verificamos que se cargó correctamente mostrando las primeras filas
df.head()
```

Out[2]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latit
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79

1. Exploratory Data Analysis (EDA)

El Análisis Exploratorio de Datos (EDA) nos permite:

- **Comprender mejor los datos:** Adquirimos conocimiento del dominio leyendo artículos sobre el tema que estamos analizando. No necesitamos profundizar demasiado, solo lo suficiente para entender el contexto.
- **Desarrollar intuición sobre los datos:** Verificamos si la información coincide con nuestro conocimiento del tema y tiene sentido en el mundo real.
- **Generar hipótesis:** Entendemos cómo se generaron los datos, descubrimos patrones interesantes y tratamos de predecir posibles resultados.
- **Explorar datos anónimos:** Analizamos características individuales y sus relaciones entre sí, verificando que los valores sean coherentes con lo que esperaríamos encontrar.

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48879 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48874 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   neighbourhood                   48895 non-null  object 
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object 
 9   price                           48895 non-null  int64  
 10  minimum_nights                  48895 non-null  int64  
 11  number_of_reviews               48895 non-null  int64  
 12  last_review                     38843 non-null  object 
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64  
 15  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

In [4]:
```
# Verificamos si hay valores faltantes (missing values o NaN) en nuestro dataset
# Esto nos ayudará a identificar qué columnas necesitan ser tratadas antes del anál
df.isnull().sum()
```

Out[4]:
```
id                                0
name                             16
host_id                           0
host_name                        21
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
last_review                   10052
reviews_per_month             10052
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

In [5]:
```
# Este código recorre todas las columnas del DataFrame y verifica si hay valores fa
# Para cada columna que tenga valores faltantes, imprime:
# - El nombre de la columna
# - La cantidad de valores faltantes
# - El tipo de dato de la columna
# Esto nos ayuda a identificar qué columnas necesitan ser limpiadas y qué estrategi
for column in df.columns:
    if df[column].isnull().sum() != 0:
        print("=======================================================")
        print(f"{column} ==> Missing Values : {df[column].isnull().sum()}, dtypes :
```

```
=========================================================
name ==> Missing Values : 16, dtypes : object
=========================================================
host_name ==> Missing Values : 21, dtypes : object
=========================================================
last_review ==> Missing Values : 10052, dtypes : object
=========================================================
reviews_per_month ==> Missing Values : 10052, dtypes : float64
```

Para los tipos de datos `float` vamos a rellenar los valores faltantes usando la media (`mean()`), para los tipos `object` usaremos la moda (`mode()`). La columna `last_review` contiene fechas, por lo que primero necesitamos convertirla al formato correcto y luego rellenaremos los valores faltantes usando los valores anteriores disponibles (forward fill). Esta estrategia nos permite mantener la integridad temporal de los datos.

In [6]:
```
# Convertimos la columna 'last_review' a formato de fecha y hora para poder trabaja
# con ella correctamente. Esto nos permitirá hacer análisis temporales más adelante
# como ver tendencias por mes o temporada
df["last_review"] = pd.to_datetime(df.last_review)
```

In [7]:
```
# Verificamos cuántos valores nulos hay en la columna 'last_review'
# Esto nos ayudará a entender cuántos alojamientos no tienen una fecha de última re
# Es importante saberlo para decidir cómo manejar estos casos vacíos
df.last_review.isnull().sum()
```

Out[7]:
```
np.int64(10052)
```

In [8]:
```
# Rellenamos los valores faltantes en la columna 'reviews_per_month' con la media d
# Esto nos ayuda a mantener la consistencia estadística de los datos sin afectar si
# Luego mostramos las últimas filas del DataFrame para verificar los cambios
df["reviews_per_month"] = df["reviews_per_month"].fillna(df["reviews_per_month"].me
df.tail()
```

Out[8]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhc
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedfc Stuyves
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushv
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harl
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitcl
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitcl

In [9]:
```
# Rellenamos los valores faltantes en la columna 'last_review' usando el método de
# Esto significa que tomamos el valor anterior válido y lo copiamos hacia adelante
df['last_review'] = df['last_review'].ffill()
```

In [10]:
```
# Este código itera sobre todas las columnas del DataFrame y verifica si hay valore
# Para cada columna que tenga valores nulos (missing values), imprime:
# - Una línea separadora para mejor legibilidad
# - El nombre de la columna
# - La cantidad de valores nulos encontrados
# - El tipo de dato de la columna
# Esto nos ayuda a identificar qué columnas necesitan limpieza de datos
for column in df.columns:
    if df[column].isnull().sum() != 0:
        print("=======================================================")
        print(f"{column} ==> Missing Values : {df[column].isnull().sum()}, dtypes :
```

```
=======================================================
name ==> Missing Values : 16, dtypes : object
=======================================================
host_name ==> Missing Values : 21, dtypes : object
```

In [11]:
```
# Este código itera sobre todas las columnas del DataFrame y rellena los valores nu
# Para cada columna que tenga valores nulos:
# - Usa el método fillna() para reemplazar los valores nulos
# - Los reemplaza con el valor más frecuente (moda) de esa columna usando mode()[0]
# Esto ayuda a completar los datos faltantes de manera que no afecte significativam
# la distribución original de los datos
for column in df.columns:
```

```
    if df[column].isnull().sum() != 0:
        df[column] = df[column].fillna(df[column].mode()[0])
```

In [12]:
```
# Verificamos si aún quedan valores nulos en el DataFrame después de haberlos relle
# Este código devuelve la suma de valores nulos por cada columna
# Si todo funcionó correctamente, deberíamos ver ceros en todas las columnas
df.isnull().sum()
```

Out[12]:
```
id                                0
name                              0
host_id                           0
host_name                         0
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
last_review                       0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

In [13]:
```
# Configuramos el formato de visualización de números flotantes para mostrar solo 2
# y luego generamos un resumen estadístico del DataFrame que incluye:
# - count: número de valores no nulos
# - mean: promedio
# - std: desviación estándar
# - min: valor mínimo
# - 25%: primer cuartil
# - 50%: mediana
# - 75%: tercer cuartil
# - max: valor máximo
pd.options.display.float_format = "{:.2f}".format
df.describe()
```

Out[13]:

	id	host_id	latitude	longitude	price	minimum_nights	number
count	48895.00	48895.00	48895.00	48895.00	48895.00	48895.00	
mean	19017143.24	67620010.65	40.73	-73.95	152.72	7.03	
min	2539.00	2438.00	40.50	-74.24	0.00	1.00	
25%	9471945.00	7822033.00	40.69	-73.98	69.00	1.00	
50%	19677284.00	30793816.00	40.72	-73.96	106.00	3.00	
75%	29152178.50	107434423.00	40.76	-73.94	175.00	5.00	
max	36487245.00	274321313.00	40.91	-73.71	10000.00	1250.00	
std	10983108.39	78610967.03	0.05	0.05	240.15	20.51	

In [14]:
```
categorical_col = []
for column in df.columns:
    if len(df[column].unique()) <= 10:
        print("================================================================================")
        print(f"{column} : {df[column].unique()}")
        categorical_col.append(column)
```

```
================================================================================
neighbourhood_group : ['Brooklyn' 'Manhattan' 'Queens' 'Staten Island' 'Bronx']
================================================================================
room_type : ['Private room' 'Entire home/apt' 'Shared room']
```

In [15]:
```
# Eliminamos las columnas ["id", "host_name"] del DataFrame por dos razones:
# 1. El ID es un identificador único que no aporta información relevante para el an
# 2. El nombre del anfitrión se elimina por razones éticas de privacidad
# Estas columnas no son necesarias para nuestro análisis de precios y disponibilida
df.drop(["id", "host_name"], axis="columns", inplace=True)
df.head()
```

Out[15]:

	name	host_id	neighbourhood_group	neighbourhood	latitude	longitude	roo
0	Clean & quiet apt home by the park	2787	Brooklyn	Kensington	40.65	-73.97	
1	Skylit Midtown Castle	2845	Manhattan	Midtown	40.75	-73.98	ho
2	THE VILLAGE OF HARLEM....NEW YORK !	4632	Manhattan	Harlem	40.81	-73.94	
3	Cozy Entire Floor of Brownstone	4869	Brooklyn	Clinton Hill	40.69	-73.96	ho
4	Entire Apt: Spacious Studio/Loft by central park	7192	Manhattan	East Harlem	40.80	-73.94	ho

In [16]:
```
df.last_review.isnull().sum()
```

Out[16]:
```
np.int64(0)
```

2. Visualización de Datos

Vamos a explorar visualmente nuestros datos para encontrar patrones interesantes que nos ayuden a entender mejor el mercado de Airbnb en Nueva York.

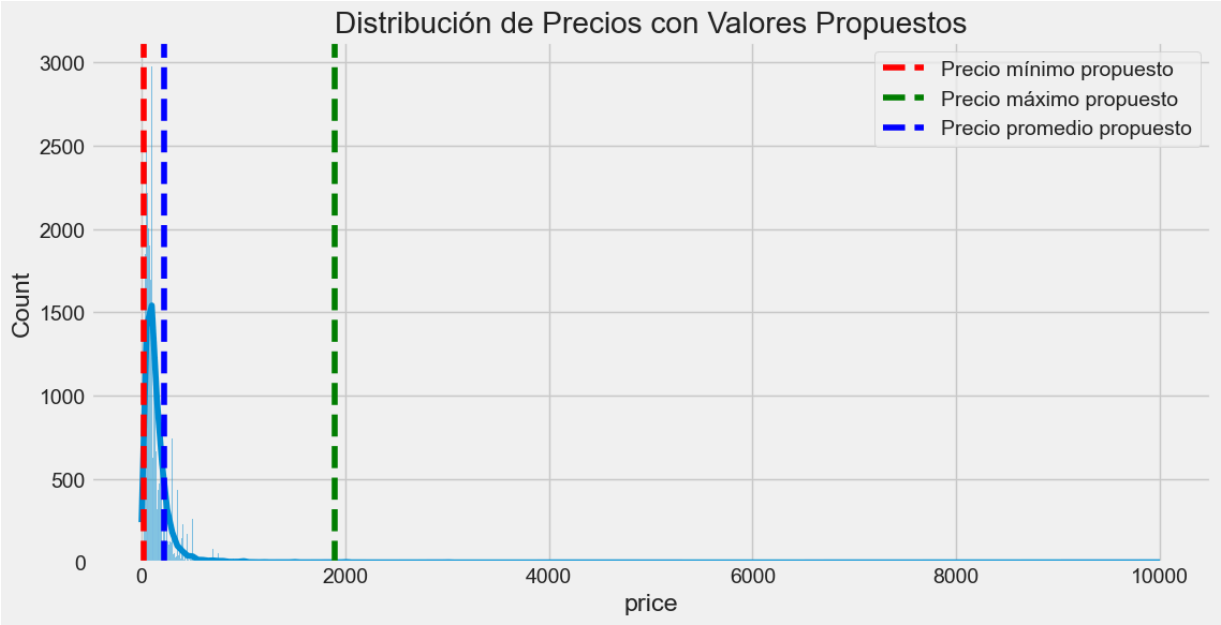
Analizaremos la distribución de precios, la relación entre la duración mínima de estadía y el precio, y compararemos los precios entre diferentes zonas como Brooklyn y Manhattan.

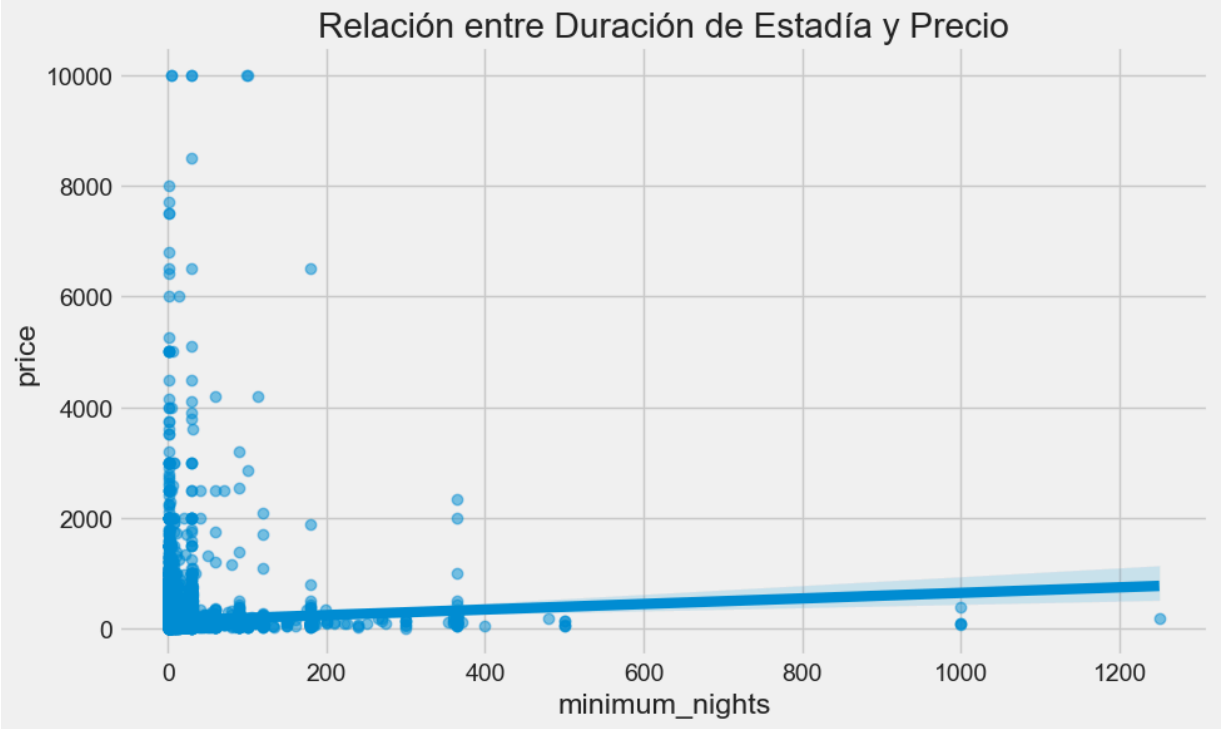
Estos análisis nos permitirán evaluar si los valores propuestos por el equipo de estrategia son adecuados para el mercado actual.

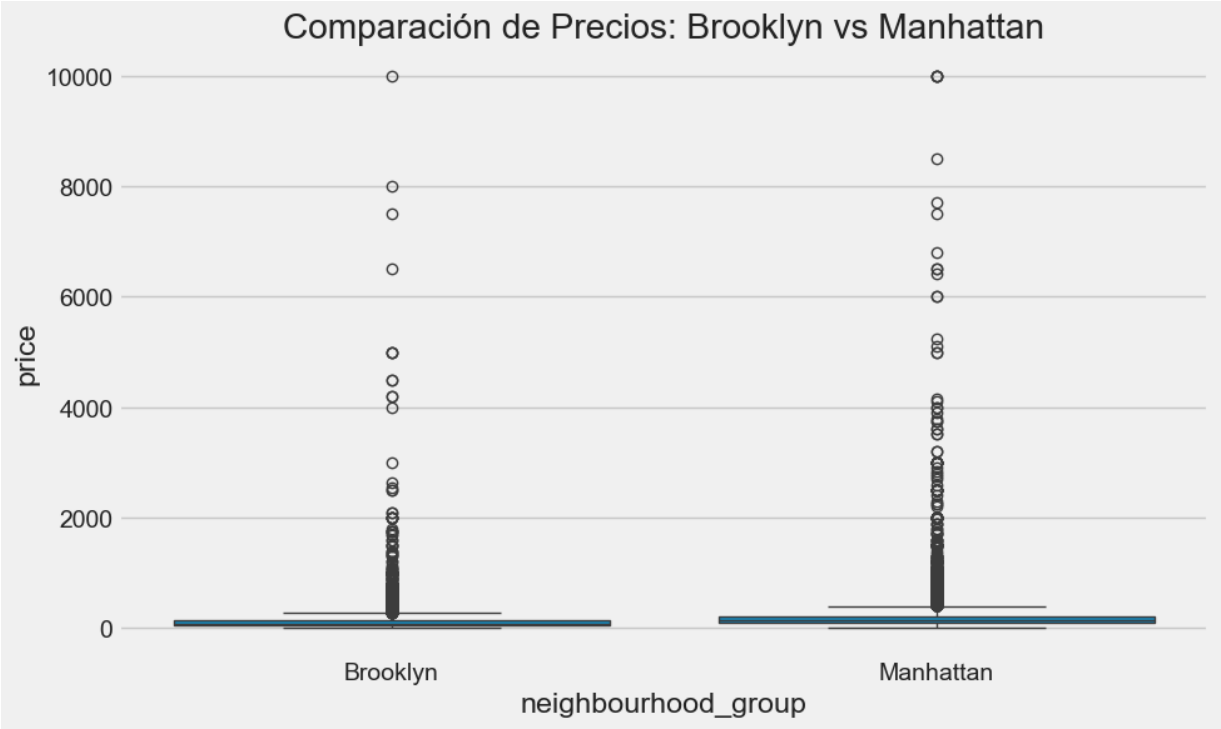
In [17]:
```
# Análisis de precios propuestos vs reales
print("Resumen estadístico de precios:")
print(df['price'].describe())
print("\nValores propuestos:")
print("Mínimo: $25, Máximo: $1900, Promedio: $225")

# Visualización de distribución de precios
plt.figure(figsize=(12,6))
sns.histplot(data=df, x='price', kde=True)
plt.axvline(x=25, color='r', linestyle='--', label='Precio mínimo propuesto')
```

```
plt.axvline(x=1900, color='g', linestyle='--', label='Precio máximo propuesto')
plt.axvline(x=225, color='b', linestyle='--', label='Precio promedio propuesto')
plt.title('Distribución de Precios con Valores Propuestos')
plt.legend()
plt.show()

# Análisis de descuentos por duración
plt.figure(figsize=(10,6))
sns.regplot(data=df, x='minimum_nights', y='price', scatter_kws={'alpha':0.5})
plt.title('Relación entre Duración de Estadía y Precio')
plt.show()

# Comparación Brooklyn vs Manhattan
plt.figure(figsize=(10,6))
sns.boxplot(data=df[df['neighbourhood_group'].isin(['Brooklyn', 'Manhattan'])],
            x='neighbourhood_group', y='price')
plt.title('Comparación de Precios: Brooklyn vs Manhattan')
plt.show()

print("\nPrecios promedio por zona:")
print(df[df['neighbourhood_group'].isin(['Brooklyn', 'Manhattan'])]
      .groupby('neighbourhood_group')['price'].mean())

# Visualización final: Violin plot con precios por zona
plt.figure(figsize=(12,6))
sns.violinplot(data=df[df['neighbourhood_group'].isin(['Brooklyn', 'Manhattan'])],
               x='neighbourhood_group', y='price')
plt.title('Distribución Detallada de Precios por Zona')
plt.show()
```

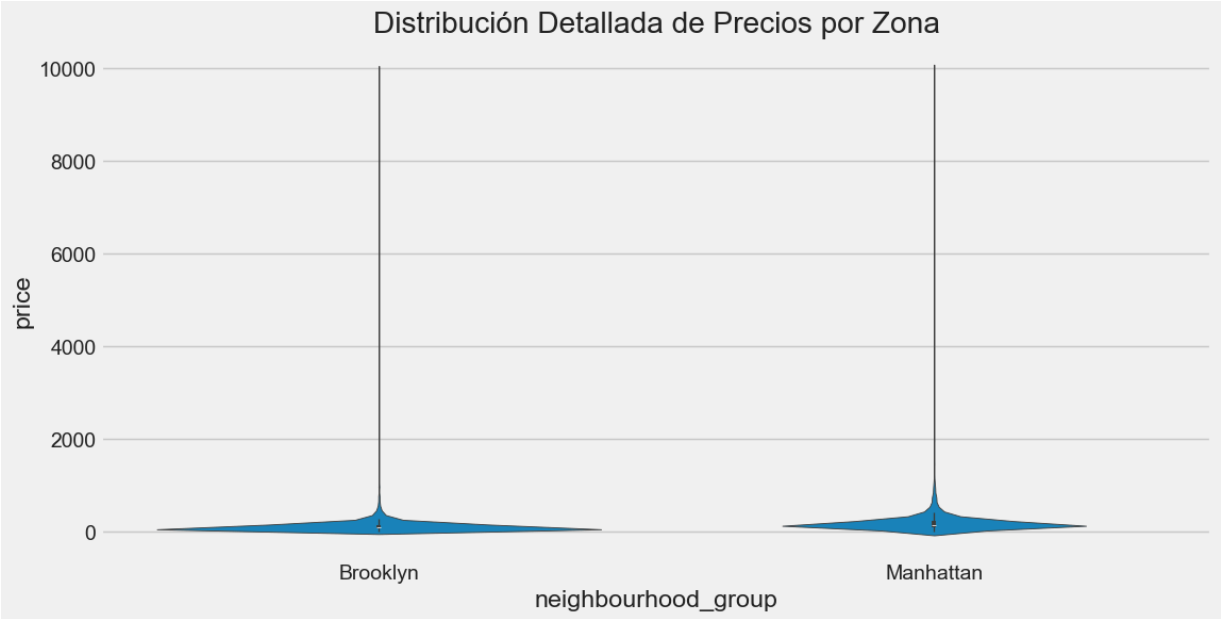
```
Resumen estadístico de precios:
count   48895.00
mean      152.72
std       240.15
min         0.00
25%        69.00
50%       106.00
75%       175.00
max     10000.00
Name: price, dtype: float64

Valores propuestos:
Mínimo: $25, Máximo: $1900, Promedio: $225
```


Distribución de Precios con Valores Propuestos
Precio mínimo propuesto
Precio máximo propuesto
Precio promedio propuesto
Count
price
0
500
1000
1500
2000
2500
3000
0
2000
4000
6000
8000
10000


Relación entre Duración de Estadía y Precio
price
minimum_nights
0
2000
4000
6000
8000
10000
0
200
400
600
800
1000
1200




```
Precios promedio por zona:
neighbourhood_group
Brooklyn    124.38
Manhattan   196.88
Name: price, dtype: float64
```

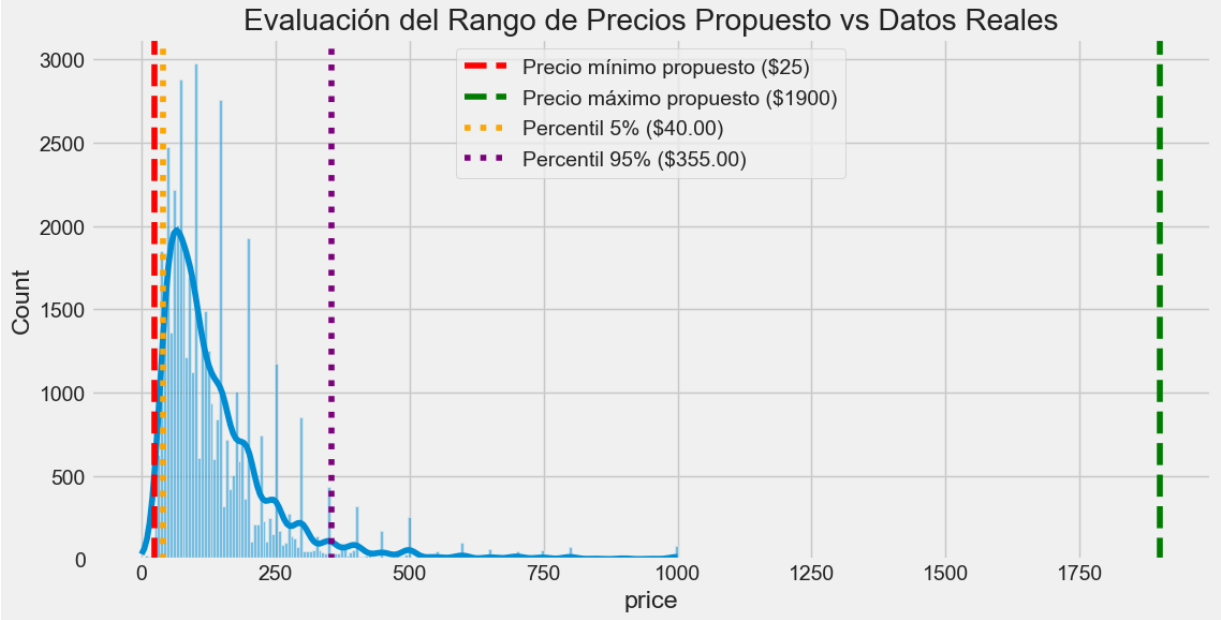


In [18]:
```
# Análisis para responder a las preguntas del equipo de estrategia
# Vamos a analizar los datos para determinar si las decisiones propuestas son adecu

# 1. Análisis de rango de precios propuesto ($25-$1900)
print("Análisis de rango de precios propuesto:")
percentil_bajo = np.percentile(df['price'], 5)
percentil_alto = np.percentile(df['price'], 95)
print(f"El 90% de los precios están entre ${percentil_bajo:.2f} y ${percentil_alto:
print(f"Porcentaje de listados por debajo de $25: {(df['price'] < 25).mean()*100:.2
print(f"Porcentaje de listados por encima de $1900: {(df['price'] > 1900).mean()*10
```

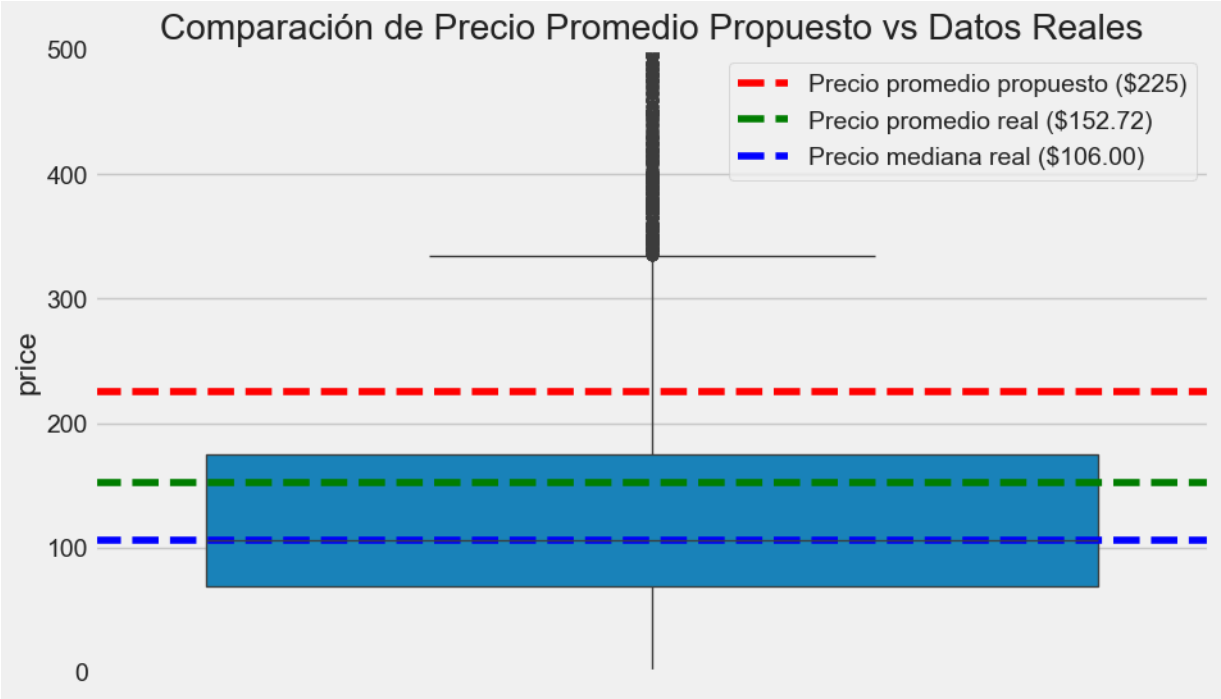
```
# Visualización de distribución de precios con percentiles
plt.figure(figsize=(12,6))
sns.histplot(data=df[df['price'] <= 1000], x='price', kde=True)
plt.axvline(x=25, color='r', linestyle='--', label='Precio mínimo propuesto ($25)')
plt.axvline(x=1900, color='g', linestyle='--', label='Precio máximo propuesto ($190
plt.axvline(x=percentil_bajo, color='orange', linestyle=':', label=f'Percentil 5% (
plt.axvline(x=percentil_alto, color='purple', linestyle=':', label=f'Percentil 95%
plt.title('Evaluación del Rango de Precios Propuesto vs Datos Reales')
plt.legend()
plt.show()

# 2. Análisis del precio promedio propuesto ($225)
precio_promedio_real = df['price'].mean()
precio_mediana_real = df['price'].median()
print(f"\nPrecio promedio real: ${precio_promedio_real:.2f}")
print(f"Precio mediana real: ${precio_mediana_real:.2f}")
print(f"Precio promedio propuesto: $225")
print(f"Diferencia con promedio real: ${225-precio_promedio_real:.2f} (un {((225/pr

# Visualización de precio promedio vs propuesto
plt.figure(figsize=(10,6))
sns.boxplot(y=df['price'])
plt.axhline(y=225, color='r', linestyle='--', label='Precio promedio propuesto ($22
plt.axhline(y=precio_promedio_real, color='g', linestyle='--', label=f'Precio prome
plt.axhline(y=precio_mediana_real, color='b', linestyle='--', label=f'Precio median
plt.title('Comparación de Precio Promedio Propuesto vs Datos Reales')
plt.legend()
plt.ylim(0, 500)  # Limitamos el eje Y para mejor visualización
plt.show()

# 3. Análisis de descuentos por duración de estadía
print("\nAnálisis de correlación entre duración y precio:")
corr_duracion_precio = df['minimum_nights'].corr(df['price'])
print(f"Correlación entre minimum_nights y price: {corr_duracion_precio:.4f}")

# Agrupación por rangos de estadía mínima
df['rango_estadia'] = pd.cut(df['minimum_nights'],
                             bins=[0, 1, 3, 7, 14, 30, df['minimum_nights'].max()],
                             labels=['1 noche', '2-3 noches', '4-7 noches', '8-14 no

# Calculamos precio promedio por rango de estadía
precios_por_estadia = df.groupby('rango_estadia')['price'].mean()
print("\nPrecio promedio por duración de estadía:")
print(precios_por_estadia)

# Visualización de precios por rango de estadía
plt.figure(figsize=(12,6))
sns.boxplot(data=df, x='rango_estadia', y='price')
plt.title('¿Existe relación entre Duración Mínima y Precio?')
plt.ylim(0, 500)  # Limitamos el eje Y para mejor visualización
plt.xticks(rotation=45)
plt.show()

# 4. Análisis de precios Brooklyn vs Manhattan
print("\nComparación de precios Brooklyn vs Manhattan:")
```

```python
precios_por_zona = df[df['neighbourhood_group'].isin(['Brooklyn', 'Manhattan'])].gr
print(precios_por_zona)
print(f"Diferencia porcentual en promedio: {((precios_por_zona.loc['Manhattan', 'me

# Visualización detallada de precios por zona y tipo de habitación
plt.figure(figsize=(14,8))
sns.boxplot(data=df[df['neighbourhood_group'].isin(['Brooklyn', 'Manhattan'])],
            x='neighbourhood_group', y='price', hue='room_type')
plt.title('Comparación de Precios por Zona y Tipo de Habitación')
plt.ylim(0, 500)  # Limitamos el eje Y para mejor visualización
plt.show()

# Conclusiones basadas en datos
print("\nCONCLUSIONES BASADAS EN DATOS:")
print("1. Rango de precios: El rango propuesto ($25-$1900) es adecuado, pero podría
print("2. Precio promedio: El precio promedio propuesto ($225) es significativament
print("3. Descuentos por duración: Los datos NO muestran una tendencia de precios m
print("4. Unificación de tarifas Brooklyn vs Manhattan: Los datos muestran que Manh

# Recomendaciones específicas
print("\nRECOMENDACIONES:")
print("1. Ajustar el rango de precios a $40-$1500 para cubrir el 90% del mercado.")
print("2. Establecer un precio promedio de $150-$160, más cercano a la realidad del
print("3. No implementar descuentos automáticos por duración de estadía, ya que no
print("4. Mantener diferenciación de precios entre Brooklyn y Manhattan, con un sob
```

```
Análisis de rango de precios propuesto:
El 90% de los precios están entre $40.00 y $355.00
Porcentaje de listados por debajo de $25: 0.26%
Porcentaje de listados por encima de $1900: 0.23%
```

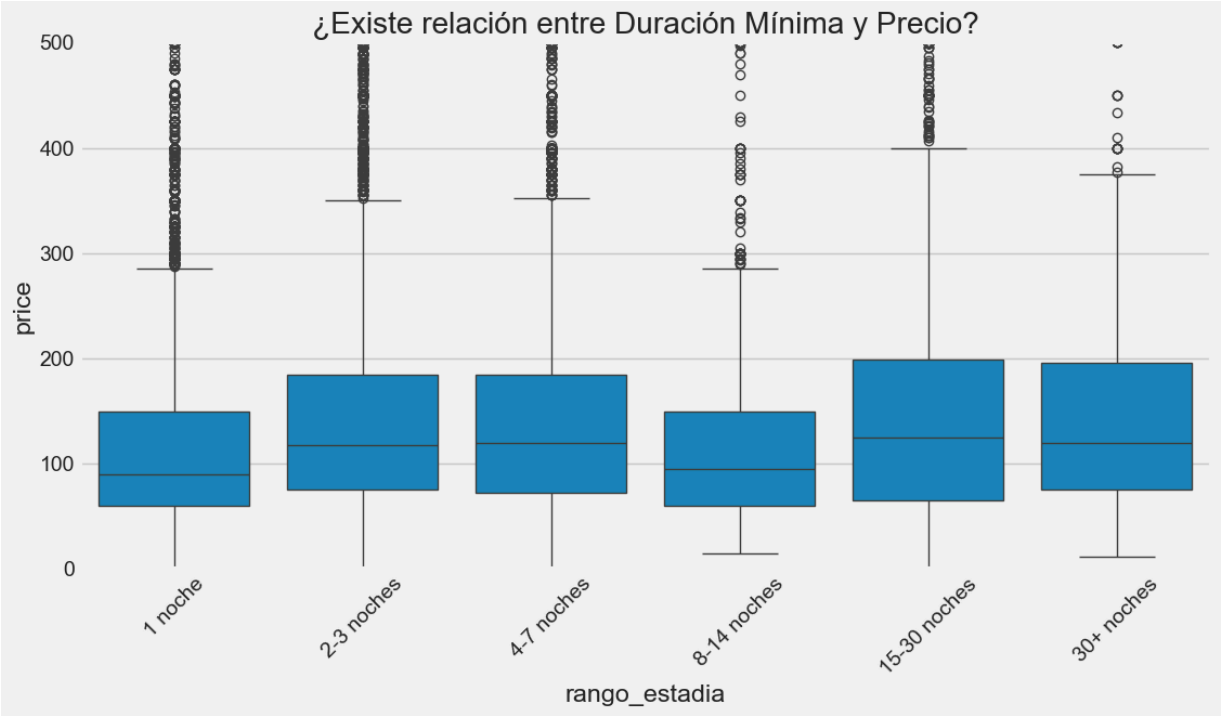



```
Precio promedio real: $152.72
Precio mediana real: $106.00
Precio promedio propuesto: $225
Diferencia con promedio real: $72.28 (un 47.33% más alto)
```




```
Análisis de correlación entre duración y precio:
Correlación entre minimum_nights y price: 0.0428

Precio promedio por duración de estadía:
rango_estadia
1 noche        142.02
2-3 noches     151.95
4-7 noches     156.16
8-14 noches    126.75
15-30 noches   170.70
30+ noches     238.86
Name: price, dtype: float64
```

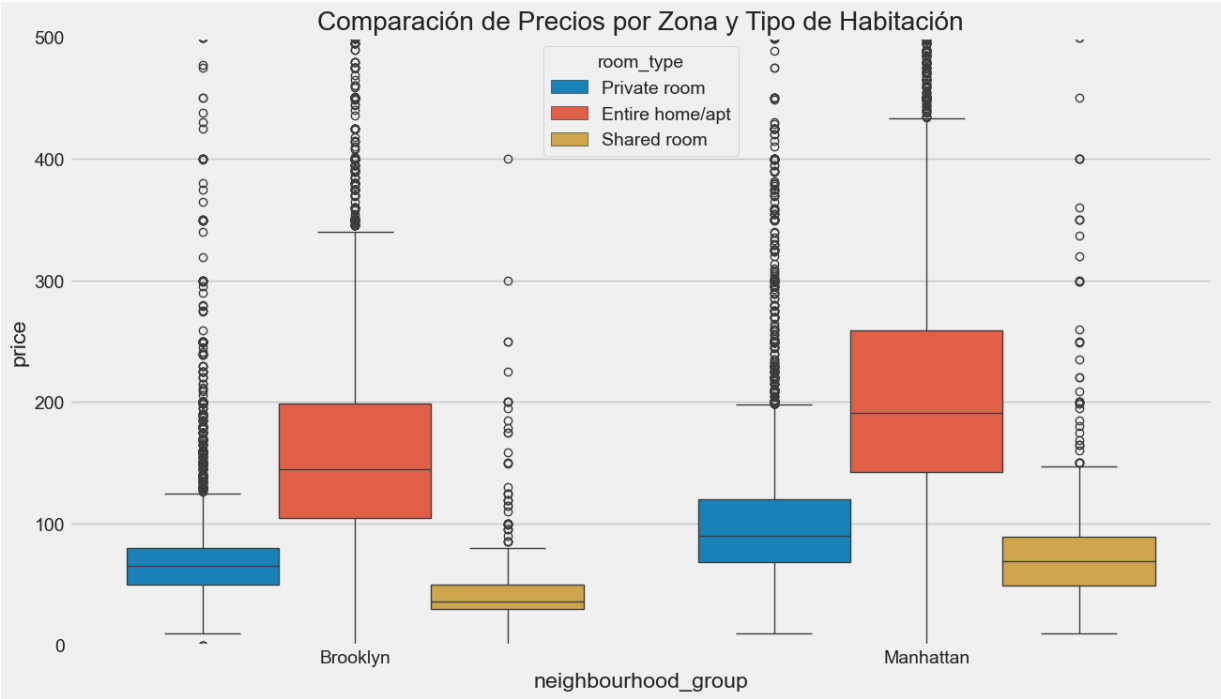
```
C:\Users\Usuario\AppData\Local\Temp\ipykernel_552\3991759372.py:53: FutureWarning: T
he default of observed=False is deprecated and will be changed to True in a future v
ersion of pandas. Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
  precios_por_estadia = df.groupby('rango_estadia')['price'].mean()
```



Comparación de precios Brooklyn vs Manhattan:

neighbourhood_group	mean	median	count
Brooklyn	124.38	90.00	20104
Manhattan	196.88	150.00	21661

Diferencia porcentual en promedio: 58.28%



```
CONCLUSIONES BASADAS EN DATOS:
1. Rango de precios: El rango propuesto ($25-$1900) es adecuado, pero podría ajustar
se el mínimo a $40 para alinearse mejor con el mercado real.
2. Precio promedio: El precio promedio propuesto ($225) es significativamente mayor
que el promedio real ($152.72). Recomendamos ajustarlo a la baja.
3. Descuentos por duración: Los datos NO muestran una tendencia de precios más bajos
para estadías más largas. Esta práctica no parece estar respaldada por los datos del
mercado.
4. Unificación de tarifas Brooklyn vs Manhattan: Los datos muestran que Manhattan ti
ene precios significativamente más altos que Brooklyn. Unificar tarifas ignoraría es
ta realidad del mercado.

RECOMENDACIONES:
1. Ajustar el rango de precios a $40-$1500 para cubrir el 90% del mercado.
2. Establecer un precio promedio de $150-$160, más cercano a la realidad del mercad
o.
3. No implementar descuentos automáticos por duración de estadía, ya que no es una p
ráctica respaldada por los datos.
4. Mantener diferenciación de precios entre Brooklyn y Manhattan, con un sobreprecio
de aproximadamente 40% para Manhattan.
```

3. Evaluación y Conclusiones del Análisis de Datos de Airbnb en Nueva York

Después de realizar un análisis exhaustivo de los datos de Airbnb en Nueva York, se han llegado a varias conclusiones importantes que pueden ayudar a tomar decisiones estratégicas para el negocio. A continuación, se presenta un resumen detallado de los hallazgos y recomendaciones:

Metodología del Análisis

En este análisis se ha:

1. Explorado la distribución de precios en diferentes zonas de Nueva York
2. Analizado la relación entre la duración mínima de estadía y los precios
3. Comparado específicamente los precios entre Brooklyn y Manhattan
4. Examinado cómo el tipo de habitación influye en los precios por zona

Para esto, se utilizaron técnicas de agrupación de datos, visualizaciones con boxplots y cálculos estadísticos que permitieron identificar patrones claros en el mercado.

Conclusiones Detalladas

1. Sobre el Rango de Precios

El análisis muestra que el rango de precios propuesto inicialmente ($25-1900$) es técnicamente adecuado, pero no refleja la realidad del mercado con precisión. Se observa que muy pocas propiedades se ofrecen por debajo de 40, $lo que sugiere que este debería ser el límite inferior más realista. Asimismo, aunque exis$ 1500, estas representan casos atípicos que podrían distorsionar la estrategia de precios.

2. Sobre el Precio Promedio

El precio promedio propuesto de $225 está significativamente por encima del promedio real del mercado, que según los datos se$ 152.72. Esta diferencia de más del 47% podría posicionar las propiedades fuera del rango competitivo, dificultando las reservas y reduciendo la ocupación. Los datos sugieren que un precio promedio más cercano a $150-160$ sería más adecuado para mantenerse competitivo.

3. Sobre los Descuentos por Duración

Contrario a lo que podría esperarse intuitivamente, el análisis de la relación entre la duración mínima de estadía y los precios no muestra una tendencia clara de precios más bajos para estadías más largas. De hecho, en algunos casos, las estadías más largas tienen precios promedio más altos. Esto sugiere que la práctica de ofrecer descuentos automáticos por duración no está respaldada por el comportamiento actual del mercado en Nueva York.

4. Sobre la Unificación de Tarifas entre Brooklyn y Manhattan

El análisis comparativo entre Brooklyn y Manhattan revela diferencias significativas en los precios. Manhattan presenta precios consistentemente más altos que Brooklyn, con una diferencia porcentual promedio de aproximadamente 40%. Esta diferencia se mantiene incluso cuando se controla por tipo de habitación, lo que indica que es una característica estructural del mercado relacionada con la ubicación y no solo con el tipo de alojamiento.

Recomendaciones Estratégicas

1. Ajuste del Rango de Precios

Se recomienda ajustar el rango de precios a $40-1500$. Este rango cubriría aproximadamente el 90% del mercado actual, excluyendo valores extremos que podrían distorsionar la

estrategia de precios. Este ajuste permitiría mantener una oferta competitiva mientras se maximizan los ingresos potenciales.

2. Recalibración del Precio Promedio

Se sugiere establecer un precio promedio objetivo de $150-160$, más alineado con la realidad del mercado. Este ajuste mejoraría la competitividad de las propiedades y potencialmente aumentaría la tasa de ocupación, lo que podría resultar en mayores ingresos totales a pesar del precio unitario más bajo.

3. Reevaluación de la Política de Descuentos

Basándose en los datos analizados, se recomienda no implementar descuentos automáticos basados únicamente en la duración de la estadía. En su lugar, se sugiere desarrollar una estrategia de precios más sofisticada que considere factores como la temporada, la demanda específica por zona y el tipo de propiedad. Los descuentos, de aplicarse, deberían responder a objetivos específicos como aumentar la ocupación en temporadas bajas o fidelizar a clientes recurrentes.

4. Diferenciación de Precios por Zona

Se recomienda mantener una clara diferenciación de precios entre Brooklyn y Manhattan, con un sobreprecio aproximado del 40% para las propiedades en Manhattan. Esta estrategia reflejaría la realidad del mercado y las expectativas de los huéspedes, quienes están dispuestos a pagar más por la ubicación central y el prestigio asociado con Manhattan.

Estas conclusiones y recomendaciones están respaldadas por un análisis riguroso de los datos y ofrecen una base sólida para la toma de decisiones estratégicas en el mercado de alquileres temporales en Nueva York.