Aeropuertos - Locales y Rentas

May 1, 2024

1 Aeropuertos - Renta Locales

Creado por:

- **V. D. Betancourt**

1.1 Introducción

1.1.1 Objetivo

El objetivo del presente proyecto consiste en proponer un panorama general para la gestión de las **rentas de locales** en distintos aeropuertos. El análisis estará enfocado en:

- **Cantidad de Locales Rentados y No Rentados**
- **Montos por Locales Rentados**
- **Clientes Activos y Clientes con Pagos Atrasados**

Para ello, se ha creado un dataset con **datos sintéticos**. Sin embargo, puede llevarse a cabo con datos reales si es que se cuenta con ellos, siempre y cuando se respeten los campos requeridos, señalados en la Sección "**Descripción**" (se sugiere revisar el código o los datasets generados para corroborar los campos necesarios).

1.1.2 Descripción

El análisis de la renta de locales en distintos aeropuertos depende totalmente de la información disponible.

Los **datos sintéticos** que se proponen para este proyecto constan de **16 variables (columnas)**, algunas de las cuales contienen un catálogo de opciones posibles (que llamaremos `'diccionarios'`).

1.2 Settings

1.2.1 Instalar Librerías

`'faker'` es una librería en Python que se utiliza para generar **datos sintéticos**. Esto es especialmente útil cuando necesitamos simular datos realistas para pruebas sin utilizar información real.

En el caso de nuestro proyecto, utilizamos Faker únicamente para crear los campos:

- `'local_id'`

- `'telefono'`

También podríamo usarlo para crear nombres de arrendatarios, identificaciones de locales, y otros datos que nos ayudan a llenar nuestro dataset de forma que se asemeje a un escenario real.

Sin embargo, despuúes de las primeras pruebas iniciales, se concluyó que era mejor diseñar catálogos (diccionarios) para otros campos como los antes mencionados, con tal de darle un toque aún más realista y ad-hoc.

[ ]:
```
# Instalar Faker
!pip install faker
```

```
Collecting faker
  Downloading Faker-25.0.0-py3-none-any.whl (1.8 MB)
                            1.8/1.8 MB
15.1 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.4 in
/usr/local/lib/python3.10/dist-packages (from faker) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.4->faker) (1.16.0)
Installing collected packages: faker
Successfully installed faker-25.0.0
```

1.2.2 Importar Librerías

[ ]:
```
import pandas as pd
import numpy as np
from faker import Faker
```

1.3 Generar Datos Sintéticos

Warning!

- Si **no se tienen "datos reales"** todavía, entonces se debe ejecutar esta sección de código para generar los datos sintéticos (ficticios).

Los datos sintéticos generados con código de Python, constan de 16 variables (columnas), que son:

- `'fecha'`: Son fechas mensuales desde el `'2023-01-31'` hasta el `'2024-03-31'`.
- `'nombre_aeropuerto'`: Se han creado nombres genéricos para 12 aeropuertos para efectos de este proyecto, con la flexibilidad de poder sustitiuirlos en cualquier momento por los nombres reales.
- `'terminal'`: Se asume que cada aeropuerto puede tener 2 terminales: `'T1', 'T2'`.
- `'planta'`: Se asume que cada aeropuerto tiene 2 pisos o plantas: `'Baja', 'Alta'`.
- `'local_id'`: Es un código alfanumérico que consta de 3 letras y 2 números (**'???-##'**), creado con una herramienta especializada.
- `'local_categoria'`: Es la categoría principal de los locales, puediendo ser:

  - `'Transportes', 'Hoteles', 'Compras', 'Alimentos', 'Agencias de Turismo'`.

- `'local_subcateg'`: Corresponde a las posibles subcategorías para cada categoría principal, pudiendo ser:
  - `'Transportes': ['Taxis']`,
  - `'Hoteles': ['Hotel']`,
  - `'Compras': ['Duty free', 'Ropa, accesorios y zapatería', 'Regalos y souvenirs', 'Libros, revistas y música', 'Sorteos y pronósticos']`,
  - `'Alimentos': ['Comida empaquetada', 'Restaurante bar', 'Bares y cafeterías', 'Comida rápida', 'Cafetería']`,
  - `'Agencias de Turismo': ['Agencias de viaje', 'Guía de Turistas']`
- `'nombre_arrendatario'`: Se ha creado un catálogo (diccionario) para los nombres de las empresas (arrendatarios) posibles para las diferentes subcategorias (`'local_subcateg'`).
- `'telefono'`: Es un número generado aleatoriamente que consta de 10 dígitos con el formato: `'##-####-####'`.
- `'horario'`: Se ha creado un catálogo (diccionario) para los posibles horarios asignados a las diferentes subcategorias (`'local_subcateg'`).
- `'monto_renta'`: Es el monto del alquiler. Se asume que será un número aleatorio entre `'50,000'` y `'150,000'` pesos.
- `'monto_renta_usd'`: Es la conversión a dólares del `'monto_renta'` dado por `monto_renta / tipo_cambio`, con un 'tipo_cambio'=20, parametrizable.
- `'deposito'`: Se asume que se requiere pagar un mes de alquiler por adelantado como depósito de seguridad.
- `'fecha_corte'`: Esta es la fecha en la que se evalúa o determina el pago del alquiler del local.
- `'fecha_pago'`: Es la fecha en la que efectivamente se realiza el pago del alquiler.
- `'monto_pago'`: Es el monto real pagado en la fecha de pago.
  - Si el pago se realiza en o antes de la fecha de corte (`días_para_pago  0`), el monto de pago es igual al monto del alquiler (`'monto_renta'`).
  - Si el pago se hace después de la fecha de corte, se podría aplicar una deducción, reflejada en esta columna, posiblemente como un incentivo para el pago puntual o una penalidad por pago tardío.

Adicionalmente, algunas de las variables anteriores dependen de otras variables auxiliares:

- `'dias_para_pago'`: Esta columna indica el número de días de diferencia entre la fecha de corte y la fecha real en que se realiza el pago. Puede ser negativo, lo que significa que el pago se realizó antes de la fecha de corte, o positivo, lo que indica que el pago se realizó después de la fecha de corte.

[ ]:
```
import pandas as pd
import numpy as np
```

```
from faker import Faker

fake = Faker()

# Generar fechas mensuales
fechas = pd.date_range(start='2023-01-31', end='2024-03-31', freq='M')

# Definir Diccionarios
# Diccionario de Categorías
categorias = ['Transportes', 'Hoteles', 'Compras', 'Alimentos', 'Agencias de␣
 ↪Turismo']

# Diccionario de Subcategorías
subcategorias = {
    'Transportes': ['Taxis'],
    'Hoteles': ['Hotel'],
    'Compras': ['Duty free', 'Ropa, accesorios y zapatería', 'Regalos y␣
 ↪souvenirs', 'Libros, revistas y música', 'Sorteos y pronósticos'],
    'Alimentos': ['Comida empaquetada', 'Restaurante bar', 'Bares y␣
 ↪cafeterías', 'Comida rápida', 'Cafetería'],
    'Agencias de Turismo': ['Agencias de viaje', 'Guía de Turistas']
}

# Diccionario de Nombres de Empresas por Subcategoría
nombre_empresas = {
    'Taxis': ['Taxis Ejecutivos', 'Cabina Naranja', 'Taxis Apariencia', 'Sitio␣
 ↪Taxis 150'],
    'Hotel': ['Hotel H', 'Hotel M', 'Hotel I', 'Hotel R', 'Hotel F'],
    'Duty free': ['Tienda 51', 'Tienda 52'],
    'Ropa, accesorios y zapatería': ['Moda con M', 'Zapatería Elegante'],
    'Regalos y souvenirs': ['Regalos para Volar'],
    'Libros, revistas y música': ['El Lector Viajero'],
    'Sorteos y pronósticos': ['Sorteos 110'],
    'Comida empaquetada': ['Cuadrado O', 'Tienda X', 'Once por Siete'],
    'Restaurante bar': ['Rincón Urbano'],
    'Bares y cafeterías': ['Snacks de Volada'],
    'Cafetería': ['Café S', 'Café C', 'Café V'],
    'Comida rápida': ['Delicia Urbana', 'Hamburguesas M', 'Pekín para Llevar'],
    'Guía de Turistas': ['Oficina Gubernamental'],
    'Agencias de viaje': ['Viaja Joven', 'Viaja México', 'Viajes Euro']
}

# Diccionario de Horarios por Subcategoría
horarios_empresas = {
    'Taxis': ['24hrs'],
    'Hotel': ['24hrs'],
    'Duty free': ['9hrs-22hrs'],
```

```
    'Ropa, accesorios y zapatería': ['10hrs-20hrs'],
    'Regalos y souvenirs': ['8hrs-20hrs'],
    'Libros, revistas y música': ['10hrs-20hrs'],
    'Sorteos y pronósticos': ['9hrs-22hrs'],
    'Comida empaquetada': ['24hrs'],
    'Restaurante bar': ['24hrs'],
    'Bares y cafeterías': ['24hrs'],
    'Cafetería': ['24hrs'],
    'Comida rápida': ['7hrs-22hrs'],
    'Guía de Turistas': ['10hrs-20hrs'],
    'Agencias de viaje': ['10hrs-20hrs']
}



# Diccionario de Nombres Reales de Aeropuertos
# Modificar diccionario cuando se tengan disponibles los nombres reales
nombres_aeropuertos = {
    'Aeropuerto_1': 'Primer_Aeropuerto',
    'Aeropuerto_2': 'Segundo_Aeropuerto',
    'Aeropuerto_3': 'Tercer_Aeropuerto',
    'Aeropuerto_4': 'Cuarto_Aeropuerto',
    'Aeropuerto_5': 'Quinto_Aeropuerto',
    'Aeropuerto_6': 'Sexto_Aeropuerto',
    'Aeropuerto_7': 'Séptimo_Aeropuerto',
    'Aeropuerto_8': 'Octavo_Aeropuerto',
    'Aeropuerto_9': 'Noveno_Aeropuerto',
    'Aeropuerto_10': 'Décimo_Aeropuerto',
    'Aeropuerto_11': 'Undécimo_Aeropuerto',
    'Aeropuerto_12': 'Duodécimo_Aeropuerto'
}

# Suposición del total de locales disponibles en cada aeropuerto
locales_disponibles = {
    'Primer_Aeropuerto': 25,
    'Segundo_Aeropuerto': 20,
    'Tercer_Aeropuerto': 15,
    'Cuarto_Aeropuerto': 10,
    'Quinto_Aeropuerto': 20,
    'Sexto_Aeropuerto': 25,
    'Séptimo_Aeropuerto': 10,
    'Octavo_Aeropuerto': 15,
    'Noveno_Aeropuerto': 15,
    'Décimo_Aeropuerto': 10,
    'Undécimo_Aeropuerto': 20,
    'Duodécimo_Aeropuerto': 10
    }
```

```python
# Datos sintéticos usando nombres reales
data = []
for fecha in fechas:
    for num in range(1, 13):  # Para cada aeropuerto (12 en total)
        aeropuerto_nombre = f'Aeropuerto_{num}'
        nombre_real = nombres_aeropuertos[aeropuerto_nombre]  # Usar el nombre␣
 ↪real mapeado
        num_locales_rentados = np.random.randint(0,␣
 ↪locales_disponibles[nombre_real] + 1)

        for _ in range(num_locales_rentados):
            categoria = np.random.choice(categorias)
            subcategoria = np.random.choice(subcategorias[categoria])
            compania = np.random.choice(nombre_empresas[subcategoria])
            horario = horarios_empresas[subcategoria]

            monto_renta = np.round(np.random.uniform(50000, 150000), 2)
            tipo_cambio = 20  # Supuesto tipo de cambio
            monto_renta_usd = np.round(monto_renta / tipo_cambio, 2)

            fecha_corte = fecha
            dias_para_pago = np.random.choice([-3, 0, 1, 3, 5])
            fecha_pago = fecha_corte + pd.DateOffset(days=int(dias_para_pago))

            monto_pago = monto_renta if dias_para_pago <= 0 else np.
 ↪round(monto_renta - np.random.uniform(0, 5000), 2)

            data.append({
                'fecha': fecha,
                'nombre_aeropuerto': nombre_real,
                'terminal': np.random.choice(['T1', 'T2']),
                'planta': np.random.choice(['Baja', 'Alta']),
                'local_id': fake.bothify(text='???-##'),
                'local_categoria': categoria,
                'local_subcateg': subcategoria,
                'nombre_arrendatario': compania,
                'telefono': fake.numerify(text='##-####-####'),
                'horario': horario,
                'monto_renta': monto_renta,
                'monto_renta_usd': monto_renta_usd,
                'deposito': monto_renta,
                'fecha_corte': fecha_corte,
                'fecha_pago': fecha_pago,
                'monto_pago': monto_pago,
            })
```

```
df = pd.DataFrame(data)
df.head()
```

[ ]:
```
        fecha  nombre_aeropuerto terminal planta local_id local_categoria  \
0 2023-01-31  Primer_Aeropuerto       T1   Baja   syO-71         Compras
1 2023-01-31  Primer_Aeropuerto       T1   Baja   Jna-27     Transportes
2 2023-01-31  Primer_Aeropuerto       T1   Alta   DdH-79         Hoteles
3 2023-01-31  Primer_Aeropuerto       T1   Baja   Twk-23         Compras
4 2023-01-31  Primer_Aeropuerto       T1   Alta   Oir-73     Transportes

                 local_subcateg nombre_arrendatario      telefono  \
0     Libros, revistas y música   El Lector Viajero  95-8860-6482
1                         Taxis    Taxis Apariencia  62-9504-5426
2                         Hotel             Hotel M  47-5503-6658
3  Ropa, accesorios y zapatería  Zapatería Elegante  49-1949-5123
4                         Taxis     Sitio Taxis 150  34-9776-2841

         horario  monto_renta  monto_renta_usd   deposito fecha_corte  \
0  [10hrs-20hrs]    112390.81          5619.54  112390.81  2023-01-31
1        [24hrs]    120331.40          6016.57  120331.40  2023-01-31
2        [24hrs]    106680.23          5334.01  106680.23  2023-01-31
3  [10hrs-20hrs]    128252.87          6412.64  128252.87  2023-01-31
4        [24hrs]    123237.71          6161.89  123237.71  2023-01-31

  fecha_pago  monto_pago
0 2023-01-28   112390.81
1 2023-01-28   120331.40
2 2023-02-01   104511.78
3 2023-01-28   128252.87
4 2023-02-01   118622.05
```

[ ]:
```
# Filas y Columnas
print("Cantidad de Filas y Columnas en el DataFrame")
df.shape
```

```
Cantidad de Filas y Columnas en el DataFrame
```

[ ]:
```
(1480, 16)
```

[ ]:
```
# Info General
print("Información de Variables, Cantidad de Registros No Nulos, y Tipos de␣
 ↪Datos")
df.info()
```

```
Información de Variables, Cantidad de Registros No Nulos, y Tipos de Datos
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1480 entries, 0 to 1479
```

```
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fecha                 1480 non-null   datetime64[ns]
 1   nombre_aeropuerto     1480 non-null   object
 2   terminal              1480 non-null   object
 3   planta                1480 non-null   object
 4   local_id              1480 non-null   object
 5   local_categoria       1480 non-null   object
 6   local_subcateg        1480 non-null   object
 7   nombre_arrendatario   1480 non-null   object
 8   telefono              1480 non-null   object
 9   horario               1480 non-null   object
 10  monto_renta           1480 non-null   float64
 11  monto_renta_usd       1480 non-null   float64
 12  deposito              1480 non-null   float64
 13  fecha_corte           1480 non-null   datetime64[ns]
 14  fecha_pago            1480 non-null   datetime64[ns]
 15  monto_pago            1480 non-null   float64
dtypes: datetime64[ns](3), float64(4), object(9)
memory usage: 185.1+ KB
```

[ ]:
```
# Variables (Columnas)
print("Nombres de las Variables (Columnas)")
df.columns
```

```
Nombres de las Variables (Columnas)
```

[ ]:
```
Index(['fecha', 'nombre_aeropuerto', 'terminal', 'planta', 'local_id',
       'local_categoria', 'local_subcateg', 'nombre_arrendatario', 'telefono',
       'horario', 'monto_renta', 'monto_renta_usd', 'deposito', 'fecha_corte',
       'fecha_pago', 'monto_pago'],
      dtype='object')
```

1.3.1 Data Wrangling

Campos Calculados Para algunos tipos de análisis, será necesario calcular campos adicionales como estos:

- **'pago_pendiente_mes'**: Este monto permite determinar si el monto que debe pagar el cliente por concepto de renta (**'monto_renta'**) es igual o diferente con el monto efectivamente pagado por el mismo (**'monto_pago'**).
- **'cliente_con_atraso_mes'**: Se considerará que un cliente tiene atraso en el mes si su **'pago_pendiente_mes'>0** y la **'fecha_pago'** es mayor que su **'fecha_corte'**.

[ ]:
```
# Calcular campos adicionales
df['pago_pendiente_mes'] = df['monto_renta'] - df['monto_pago']
```

```
df['cliente_con_atraso_mes'] = ((df['pago_pendiente_mes'] > 0) |␣
 ↪(df['fecha_pago'] > df['fecha_corte'])).astype(int)
```

1.4 Exportar CSV

[ ]:
```
# Exportar a CSV
df.to_csv('dataset_aeropuertos_rentas.csv', index=False)
```

1.5 Cargar CSV en DataFrame

Warning!

- Si **ya se tienen "datos reales"** en formato .csv, asegurarse de que tengan los mismos campos y continuar ejecutando el código desde esta sección. En este caso, la sección anterior para Generar Datos Sintéticos no sería necesaria.

[ ]:
```
# Cargar el DataFrame desde un archivo CSV
df = pd.read_csv('dataset_aeropuertos_rentas.csv')
df.head()
```

[ ]:

	fecha	nombre_aeropuerto	terminal	planta	local_id	local_categoria	local_subcateg	nombre_arrendatario	telefono	horario	monto_renta	monto_renta_usd	deposito	fecha_corte	fecha_pago	monto_pago	pago_pendiente_mes	cliente_con_atraso_mes
0	2023-01-31	Primer_Aeropuerto	T1	Baja	syO-71	Compras	Libros, revistas y música	El Lector Viajero	95-8860-6482	['10hrs-20hrs']	112390.81	5619.54	112390.81	2023-01-31	2023-01-28	112390.81	0.00	0
1	2023-01-31	Primer_Aeropuerto	T1	Baja	Jna-27	Transportes	Taxis	Taxis Apariencia	62-9504-5426	['24hrs']	120331.40	6016.57	120331.40	2023-01-31	2023-01-28	120331.40	0.00	0
2	2023-01-31	Primer_Aeropuerto	T1	Alta	DdH-79	Hoteles	Hotel	Hotel M	47-5503-6658	['24hrs']	106680.23	5334.01	106680.23	2023-01-31	2023-02-01	104511.78	2168.45	1
3	2023-01-31	Primer_Aeropuerto	T1	Baja	Twk-23	Compras	Ropa, accesorios y zapatería	Zapatería Elegante	49-1949-5123	['10hrs-20hrs']	128252.87	6412.64	128252.87	2023-01-31	2023-01-28	128252.87	0.00	0
4	2023-01-31	Primer_Aeropuerto	T1	Alta	Oir-73	Transportes	Taxis	Sitio Taxis 150	34-9776-2841	['24hrs']	123237.71	6161.89	123237.71	2023-01-31	2023-02-01	118622.05	4615.66	1

En caso de que los datos reales no tengan los siguientes campos calculados, deberán calcularse:

```
# Calcular campos adicionales
df['pago_pendiente_mes'] = df['monto_renta'] - df['monto_pago']
df['cliente_con_atraso_mes'] = (df['pago_pendiente_mes'] > 0).astype(int)
```

1.6 Exploración Inicial de los Datos

1.6.1 Información Básica

[ ]:
```
import pandas as pd

# Mostrar las primeras filas del DataFrame para tener una idea del contenido
df.head()
```

[ ]:

	fecha	nombre_aeropuerto	terminal	planta	local_id	local_categoria	local_subcateg	nombre_arrendatario	telefono	horario	monto_renta	monto_renta_usd	deposito	fecha_corte	fecha_pago	monto_pago	pago_pendiente_mes	cliente_con_atraso_mes
0	2023-01-31	Primer_Aeropuerto	T1	Baja	syO-71	Compras	Libros, revistas y música	El Lector Viajero	95-8860-6482	['10hrs-20hrs']	112390.81	5619.54	112390.81	2023-01-31	2023-01-28	112390.81	0.00	0
1	2023-01-31	Primer_Aeropuerto	T1	Baja	Jna-27	Transportes	Taxis	Taxis Apariencia	62-9504-5426	['24hrs']	120331.40	6016.57	120331.40	2023-01-31	2023-01-28	120331.40	0.00	0
2	2023-01-31	Primer_Aeropuerto	T1	Alta	DdH-79	Hoteles	Hotel	Hotel M	47-5503-6658	['24hrs']	106680.23	5334.01	106680.23	2023-01-31	2023-02-01	104511.78	2168.45	1
3	2023-01-31	Primer_Aeropuerto	T1	Baja	Twk-23	Compras	Ropa, accesorios y zapatería	Zapatería Elegante	49-1949-5123	['10hrs-20hrs']	128252.87	6412.64	128252.87	2023-01-31	2023-01-28	128252.87	0.00	0
4	2023-01-31	Primer_Aeropuerto	T1	Alta	Oir-73	Transportes	Taxis	Sitio Taxis 150	34-9776-2841	['24hrs']	123237.71	6161.89	123237.71	2023-01-31	2023-02-01	118622.05	4615.66	1

[ ]:
```
# Resumen de los tipos de datos y valores faltantes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1480 entries, 0 to 1479
Data columns (total 18 columns):
```

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   fecha                   1480 non-null   object
 1   nombre_aeropuerto       1480 non-null   object
 2   terminal                1480 non-null   object
 3   planta                  1480 non-null   object
 4   local_id                1480 non-null   object
 5   local_categoria         1480 non-null   object
 6   local_subcateg          1480 non-null   object
 7   nombre_arrendatario     1480 non-null   object
 8   telefono                1480 non-null   object
 9   horario                 1480 non-null   object
 10  monto_renta             1480 non-null   float64
 11  monto_renta_usd         1480 non-null   float64
 12  deposito                1480 non-null   float64
 13  fecha_corte             1480 non-null   object
 14  fecha_pago              1480 non-null   object
 15  monto_pago              1480 non-null   float64
 16  pago_pendiente_mes      1480 non-null   float64
 17  cliente_con_atraso_mes  1480 non-null   int64
dtypes: float64(5), int64(1), object(12)
memory usage: 208.2+ KB
```

[ ]:
```
# Resumen estadístico de las columnas numéricas
df.describe()
```

[ ]:
```
         monto_renta  monto_renta_usd       deposito     monto_pago  \
count    1480.000000      1480.000000    1480.000000    1480.000000
mean   101136.867446      5056.843392  101136.867446   99616.113473
std     28691.102289      1434.555176   28691.102289   28735.749908
min     50019.660000      2500.980000   50019.660000   46831.670000
25%     76528.347500      3826.417500   76528.347500   74883.152500
50%    101585.275000      5079.265000  101585.275000  100424.485000
75%    126357.537500      6317.875000  126357.537500  124713.615000
max    149855.660000      7492.780000  149855.660000  149855.660000

       pago_pendiente_mes  cliente_con_atraso_mes
count         1480.000000             1480.000000
mean          1520.753973                0.598649
std           1657.122652                0.490338
min              0.000000                0.000000
25%              0.000000                0.000000
50%            972.065000                1.000000
75%           2948.317500                1.000000
max           4994.400000                1.000000
```

[ ]:
```
# Contar valores faltantes por columna
df.isnull().sum()
```

[ ]:
```
fecha                    0
nombre_aeropuerto        0
terminal                 0
planta                   0
local_id                 0
local_categoria          0
local_subcateg           0
nombre_arrendatario      0
telefono                 0
horario                  0
monto_renta              0
monto_renta_usd          0
deposito                 0
fecha_corte              0
fecha_pago               0
monto_pago               0
pago_pendiente_mes       0
cliente_con_atraso_mes   0
dtype: int64
```

1.6.2 Data Cleaning

[ ]:
```
# Valores Faltantes
import pandas as pd

# Filtrar filas donde faltan datos
registros_con_datos_faltantes = df[df[['nombre_arrendatario', 'monto_renta',␣
 ↪'local_id', 'nombre_aeropuerto']].isna().any(axis=1)]

# Mostrar los registros con datos faltantes
print("Registros con datos faltantes en columnas esenciales:")
print(registros_con_datos_faltantes)
```

```
Registros con datos faltantes en columnas esenciales:
Empty DataFrame
Columns: [fecha, nombre_aeropuerto, terminal, planta, local_id, local_categoria,
local_subcateg, nombre_arrendatario, telefono, horario, monto_renta,
monto_renta_usd, deposito, fecha_corte, fecha_pago, monto_pago,
pago_pendiente_mes, cliente_con_atraso_mes]
Index: []
```

[ ]:
```
# Corrección de Tipos de Datos
# Convertir 'fecha' y 'fecha_pago' a datetime si no lo están
df['fecha'] = pd.to_datetime(df['fecha'])
```

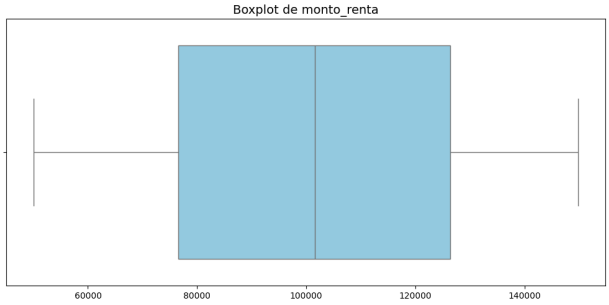
```
df['fecha_pago'] = pd.to_datetime(df['fecha_pago'])

# Asegurarse de que 'monto_renta' y 'monto_pago' sean float
df['monto_renta'] = df['monto_renta'].astype(float)
df['monto_pago'] = df['monto_pago'].astype(float)
```

[ ]:
```
# Eliminar Duplicados
# Eliminar duplicados, manteniendo la primera ocurrencia
#df.drop_duplicates(inplace=True)
```

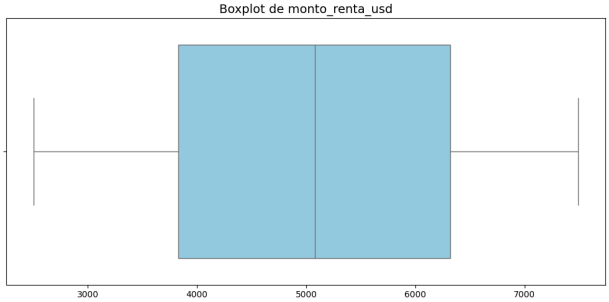
[ ]:
```
# Outliers (Valores Atípicos) en Variables Numéricas
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Supongamos que df es tu DataFrame

# Seleccionar solo columnas numéricas
df_numericas = df.select_dtypes(include=['float64', 'int64'])

# Crear subplots para cada variable numérica
fig, axes = plt.subplots(nrows=len(df_numericas.columns), ncols=1, figsize=(10,␣
 ↪5 * len(df_numericas.columns)))

# Titulo general
fig.suptitle('Visualización de Outliers en Variables Numéricas', fontsize=16,␣
 ↪y=1.02)

# Iterar sobre las columnas y crear un boxplot para cada una
for ax, column in zip(axes.flatten(), df_numericas.columns):  # asegúrate de␣
 ↪usar flatten() si axes es multidimensional
    sns.boxplot(data=df_numericas, x=column, ax=ax, color="skyblue")  # Usar␣
 ↪color en lugar de palette
    ax.set_title(f'Boxplot de {column}', fontsize=14)
    ax.set_xlabel('')

# Ajustar el layout para evitar superposiciones
plt.tight_layout()
plt.show()
```

Visualización de Outliers en Variables Numéricas

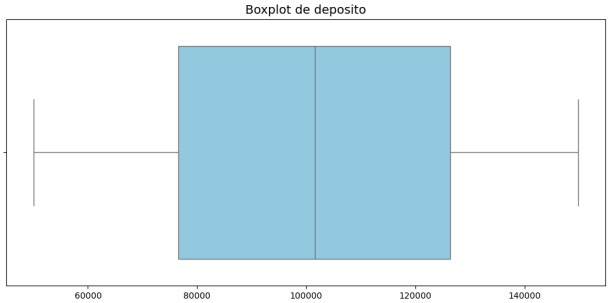
Boxplot de monto_renta



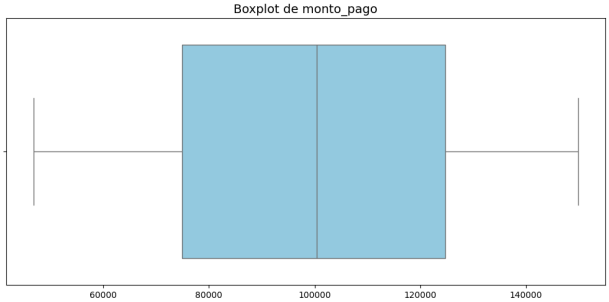
Boxplot de monto_renta_usd



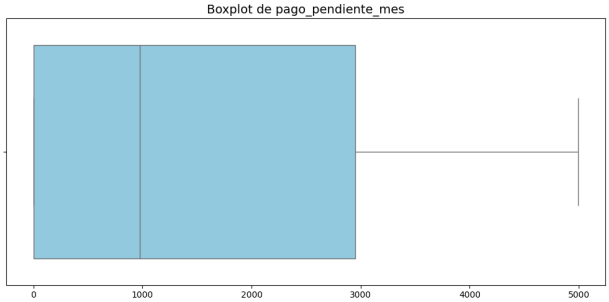
Boxplot de deposito



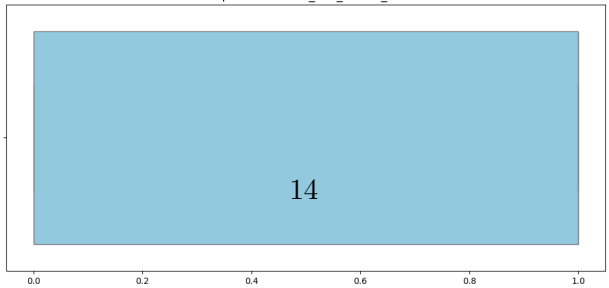
Boxplot de monto_pago



Boxplot de pago_pendiente_mes



Boxplot de cliente_con_atraso_mes



1.6.3 Generar Respaldo y Exportar CSV

[ ]:
```
# Crear un respaldo
df_aero_rentas_bkp = df.copy()
```

[ ]:
```
# Exportar a CSV
df_aero_rentas_bkp.to_csv('bkp_aeropuertos_rentas_datos_limpios.csv',␣
↪index=False)
```

1.6.4 Explorar Datos Último Mes

[ ]:
```
import pandas as pd

# Asegurarte de que 'fecha' es de tipo datetime
df['fecha'] = pd.to_datetime(df['fecha'])

# Encontrar la fecha más reciente en el DataFrame
fecha_reciente = df['fecha'].max()
print("Fecha más reciente en el dataset:", fecha_reciente)
```

```
Fecha más reciente en el dataset: 2024-03-31 00:00:00
```

[ ]:
```
# Filtrar el DataFrame para solo incluir registros de la fecha más reciente
df_reciente = df[df['fecha'] == fecha_reciente]
```

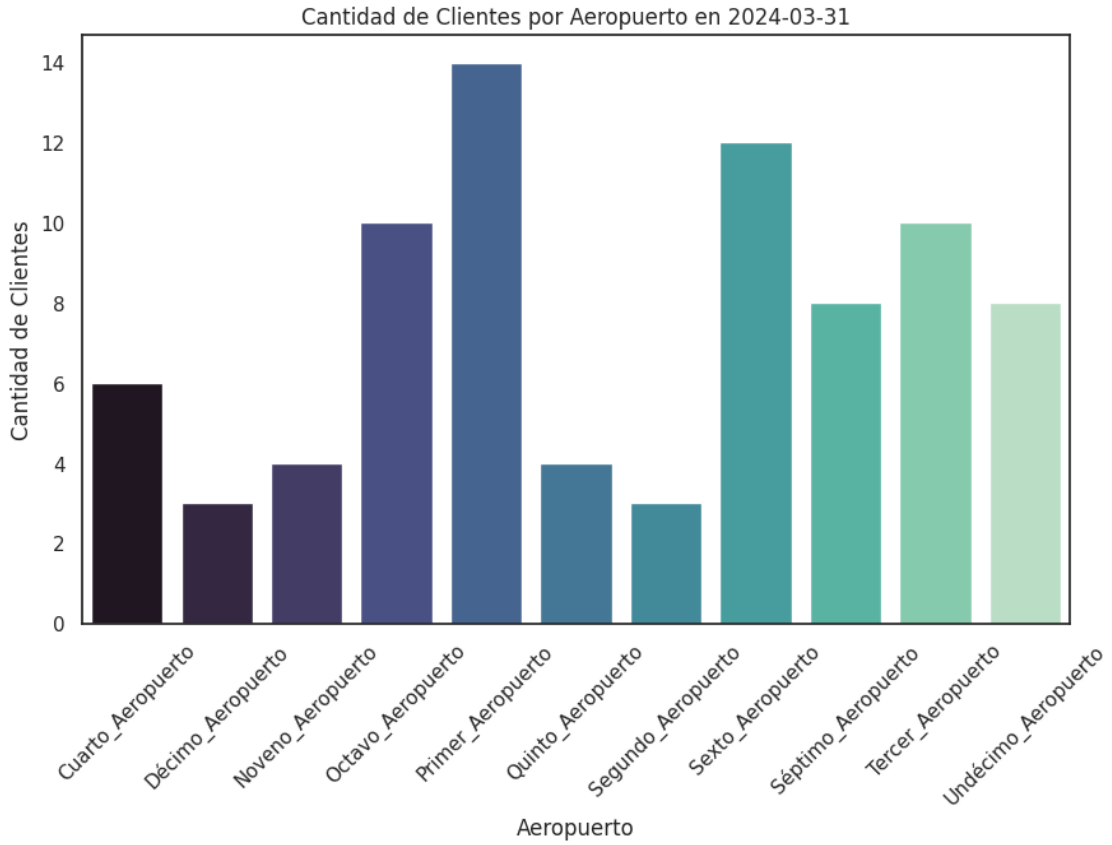
1.6.5 Cantidad de Clientes por Aeropuerto, Categoría y Subcategoría

[ ]:
```
# Total de clientes
# Contar el total de clientes únicos
total_clientes = df_reciente['nombre_arrendatario'].nunique()

# Mostrar
print("Total de Clientes en " + fecha_reciente.strftime('%Y-%m-%d') + ": ",␣
↪total_clientes)
```

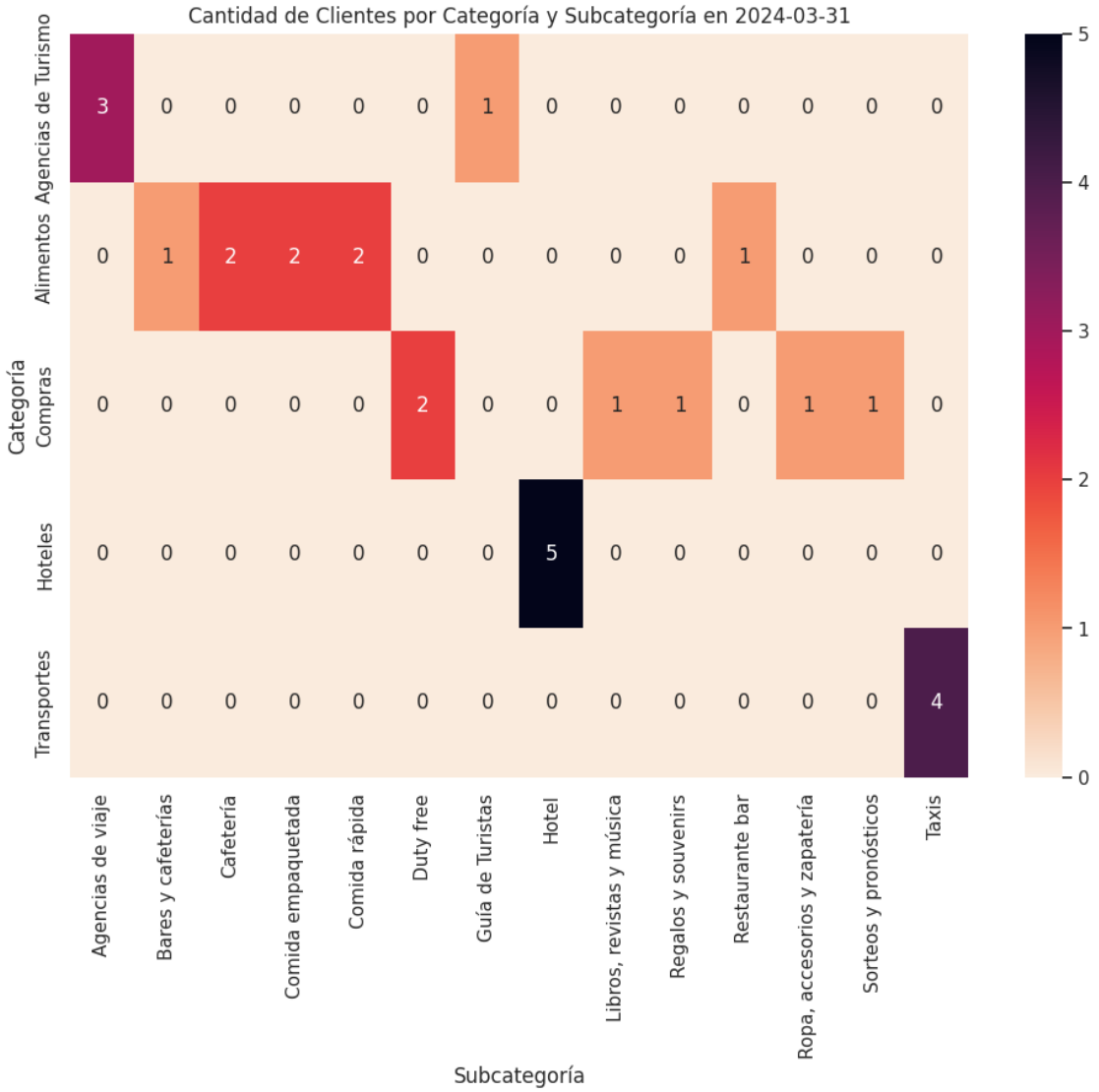
```
Total de Clientes en 2024-03-31:  27
```

[ ]:
```
# Cantidad de Clientes por Aeropuerto
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Configuración de estilos de Seaborn
sns.set(style="white")
```

```
# Cantidad de clientes únicos por Aeropuerto
clientes_por_aeropuerto = df_reciente.
 ↪groupby('nombre_aeropuerto')['nombre_arrendatario'].nunique().reset_index()
clientes_por_aeropuerto.columns = ['Aeropuerto', 'Cantidad de Clientes']

# palette:
# 'deep', 'muted', 'bright', 'pastel', 'dark', 'colorblind', 'husl'
# 'RdYlBu', 'magma', 'YlOrBr', 'crest', 'rocket_r', 'mako'

# Visualización
plt.figure(figsize=(10, 6))
sns.barplot(data=clientes_por_aeropuerto, x='Aeropuerto', y='Cantidad de␣
 ↪Clientes', hue='Aeropuerto', palette='mako')
plt.title('Cantidad de Clientes por Aeropuerto en ' + fecha_reciente.
 ↪strftime('%Y-%m-%d'))
plt.xticks(rotation=45)
plt.show()
```



[ ]:
```
# Cantidad de Clientes (únicos) Categoría y Subcategoría
clientes_por_categoria = df_reciente.groupby(['local_categoria',␣
 ↪'local_subcateg'])['nombre_arrendatario'].nunique().unstack().fillna(0)

# Visualización
plt.figure(figsize=(12, 8))
sns.heatmap(clientes_por_categoria, annot=True, fmt=".0f", cmap='rocket_r')
plt.title('Cantidad de Clientes por Categoría y Subcategoría en ' +␣
 ↪fecha_reciente.strftime('%Y-%m-%d'))
plt.ylabel('Categoría')
plt.xlabel('Subcategoría')
plt.show()
```

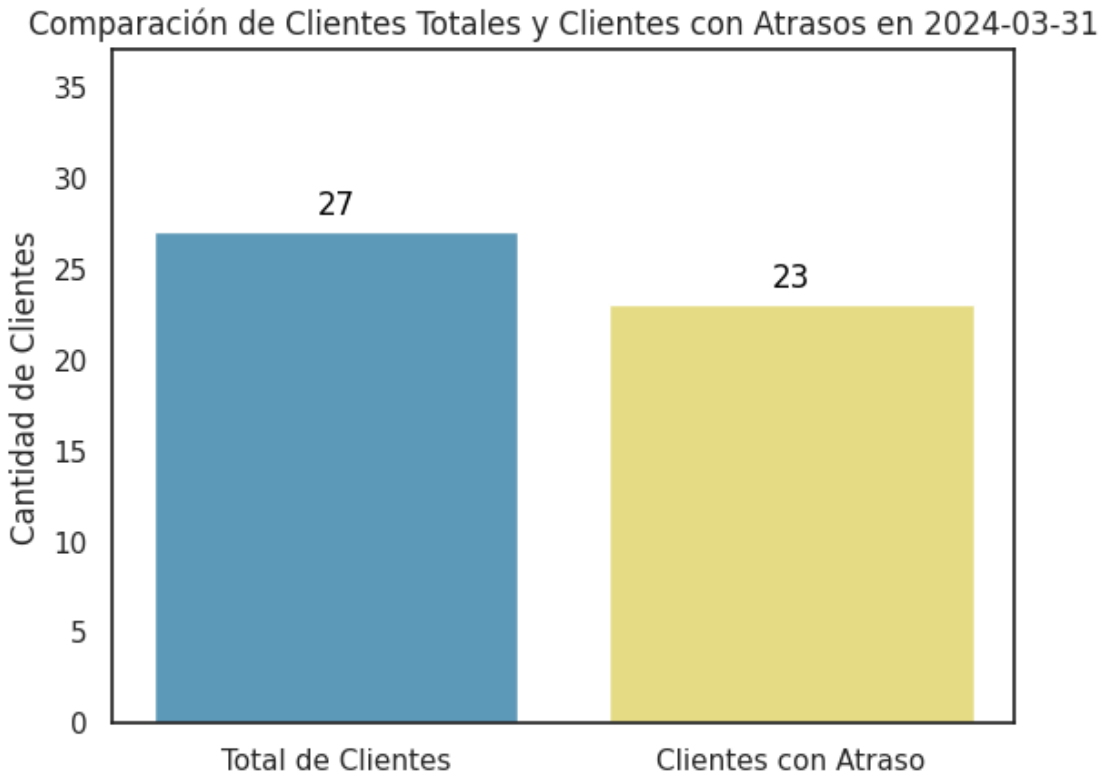


1.6.6 Cantidad de Clientes con Atraso

[ ]:
```
# Clientes con Atraso
import pandas as pd

# Contar clientes con atraso
clientes_con_atraso = df_reciente[df_reciente['cliente_con_atraso_mes'] ==␣
 ↪1]['nombre_arrendatario'].nunique()

# Contar el total de clientes únicos
total_clientes = df_reciente['nombre_arrendatario'].nunique()

print(f"Cantidad de Clientes con Atraso: {clientes_con_atraso}")
print(f"Total de Clientes Únicos: {total_clientes}")
```

```
Cantidad de Clientes con Atraso: 23
Total de Clientes Únicos: 27
```

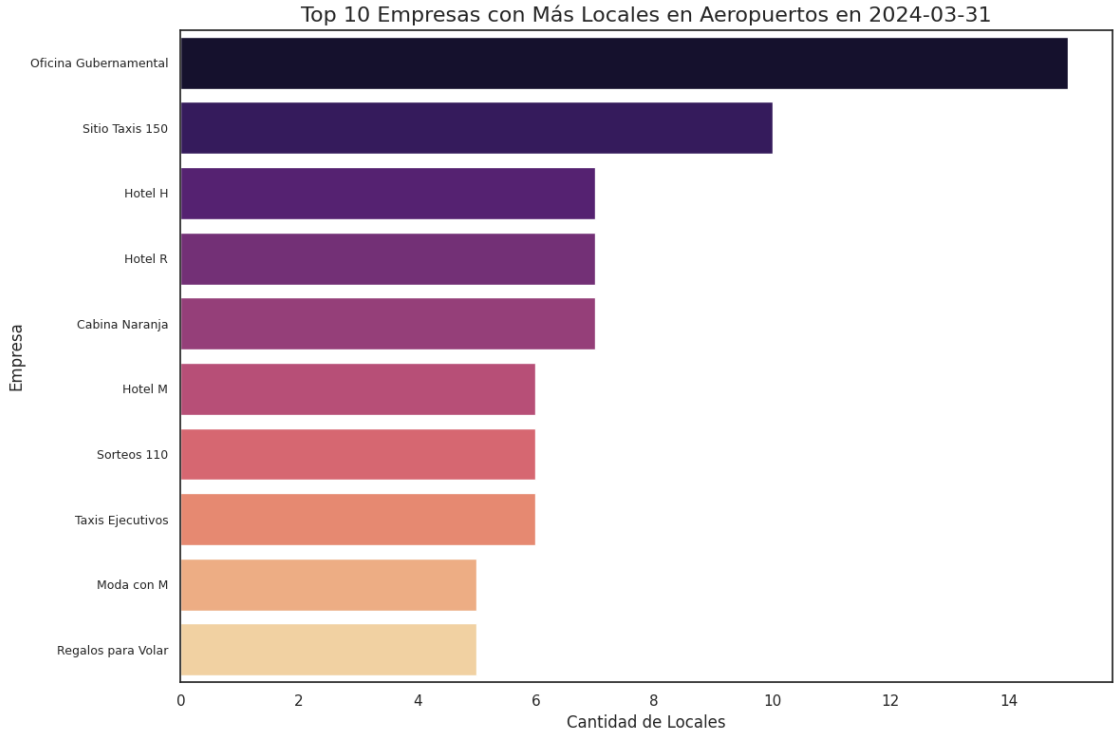
[ ]:
```
# Clientes Totales en Mes vs Clientes con Atrasos en Mes
import matplotlib.pyplot as plt

# Configurar datos para la visualización
categorias = ['Total de Clientes', 'Clientes con Atraso']
valores = [total_clientes, clientes_con_atraso]

# Gráfico de barras para comparar clientes totales y con atraso
fig, ax = plt.subplots()
ax.bar(categorias, valores, color=['#5c98b7', '#e5db84'])
ax.set_ylabel('Cantidad de Clientes')
ax.set_title('Comparación de Clientes Totales y Clientes con Atrasos en ' +␣
 ↪fecha_reciente.strftime('%Y-%m-%d'))
ax.set_ylim(0, max(valores) + 10)  # Ajustar espacio para visualización

# Añadir etiquetas con los valores exactos en las barras
for i, v in enumerate(valores):
    ax.text(i, v + 1, str(v), color='black', ha='center')

plt.show()
```



1.6.7 Concentración de las Empresas

[ ]:
```
# Top 10 Empresas con Más Locales
import seaborn as sns
import matplotlib.pyplot as plt

# Empresas con más locales
empresas_con_mas_locales = df_reciente['nombre_arrendatario'].value_counts().
 ↪head(10).reset_index()
empresas_con_mas_locales.columns = ['Empresa', 'Cantidad de Locales']

# Configuración de la visualización
plt.figure(figsize=(12, 8))  # Ajusta el tamaño para mejor visualización
ax = sns.barplot(data=empresas_con_mas_locales, y='Empresa', x='Cantidad de␣
 ↪Locales', hue='Empresa', palette='magma')

# Añadir título y etiquetas
plt.title('Top 10 Empresas con Más Locales en Aeropuertos en ' + fecha_reciente.
 ↪strftime('%Y-%m-%d'), fontsize=16)
plt.xlabel('Cantidad de Locales', fontsize=12)
plt.ylabel('Empresa', fontsize=12)
```

```
# Ajustar el tamaño del nombre de las empresas
# Reducir el tamaño de la fuente para el eje y
ax.tick_params(axis='y', labelsize=9)

# Mejorar layout y mostrar el gráfico
plt.tight_layout()
plt.show()
```



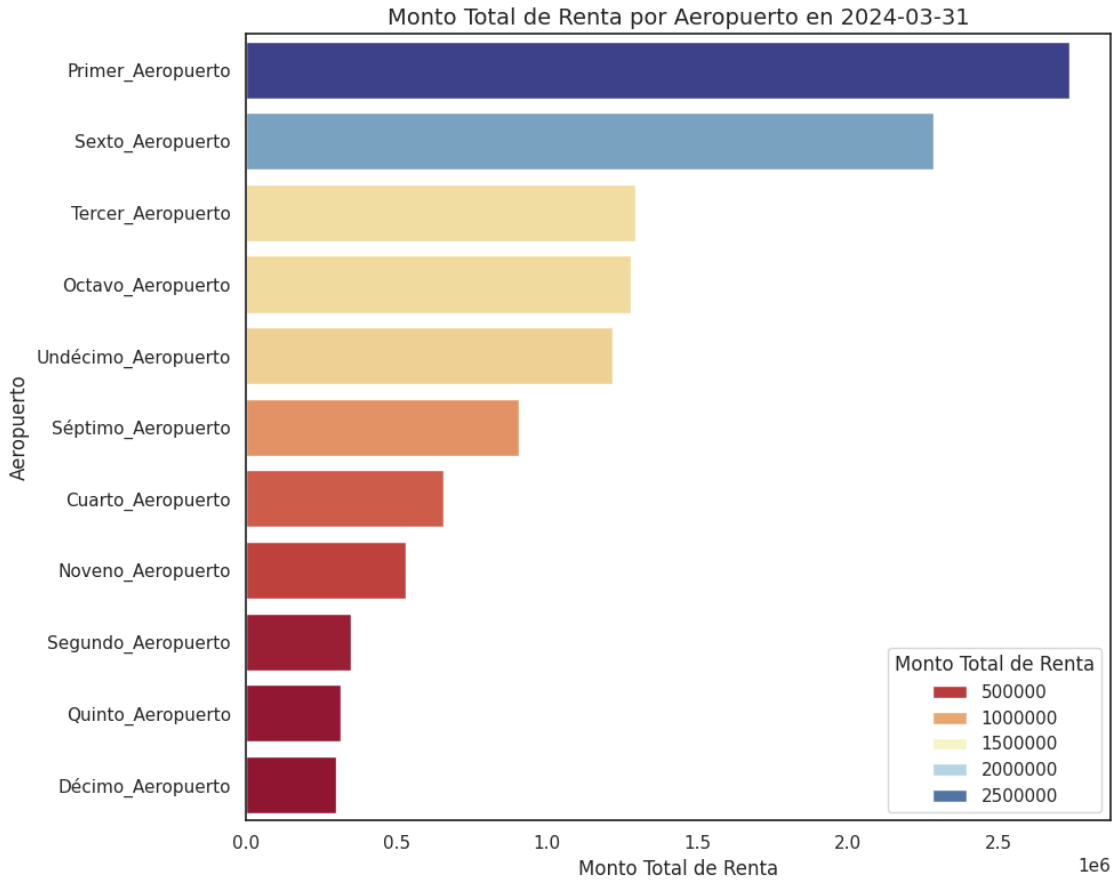
1.6.8 Monto Total de Renta por Aeropuerto

[ ]:
```
# palette:
# 'deep', 'muted', 'bright', 'pastel', 'dark', 'colorblind', 'husl'
# 'RdYlBu', 'magma', 'YlOrBr', 'crest', 'rocket_r', 'mako'
```

[ ]:
```
# Monto Total de Renta por Aeropuerto
import seaborn as sns
import matplotlib.pyplot as plt

# Calcular el monto total de renta por aeropuerto en la fecha más reciente
monto_renta_por_aeropuerto = df_reciente.
  ↪groupby('nombre_aeropuerto')['monto_renta'].sum().reset_index()
monto_renta_por_aeropuerto.columns = ['Aeropuerto', 'Monto Total de Renta']
```

```
# Ordenar los datos por 'Monto Total de Renta' de forma descendente
monto_renta_por_aeropuerto.sort_values('Monto Total de Renta', ascending=False,␣
 ↪inplace=True)
```

[ ]:
```
# Monto Total de Renta por Aeropuerto
import seaborn as sns
import matplotlib.pyplot as plt

# Configuración de la visualización
plt.figure(figsize=(10, 8))
sns.barplot(data=monto_renta_por_aeropuerto, y='Aeropuerto', x='Monto Total de␣
 ↪Renta', hue='Monto Total de Renta', palette='RdYlBu')

# Añadir título y etiquetas
plt.title('Monto Total de Renta por Aeropuerto en ' + fecha_reciente.
 ↪strftime('%Y-%m-%d'), fontsize=14)
plt.xlabel('Monto Total de Renta', fontsize=12)
plt.ylabel('Aeropuerto', fontsize=12)

# Mejorar layout y mostrar el gráfico
plt.tight_layout()
plt.show()
```



1.7 Análisis de Datos Históricos

Se abordarán los siguientes enfoques para analizar los datos:

- Análisis de Locales
  - Rentados y No Rentados
  - Montos Totales de Rentas
- Análisis de Clientes Activos
- Análisis de Pagos Atrasados

1.7.1 Análisis de Locales

Cálculos, Agrupaciones-Filtros

[ ]:
```
import pandas as pd

# Asegurando que 'fecha' es tipo datetime y extraer año y mes
df['fecha'] = pd.to_datetime(df['fecha'])
df['year_month'] = df['fecha'].dt.to_period('M')
```

```
# Suponiendo un total de locales disponibles
```

Locales Rentados

Cantidad de Locales Rentados

[ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Aseguramos que 'locales_rentados' es un DataFrame con 'nombre_aeropuerto' como índice y 'year_month' como columnas
locales_rentados = df.groupby(['nombre_aeropuerto', 'year_month']).size().unstack(fill_value=0)

print("Locales Rentados por Aeropuerto y Mes:\n")
locales_rentados
```

```
Locales Rentados por Aeropuerto y Mes:
```

[ ]:

year_month	2023-01	2023-02	2023-03	2023-04	2023-05	2023-06	\
nombre_aeropuerto							
Cuarto_Aeropuerto	3	8	1	2	0	1	
Duodécimo_Aeropuerto	6	4	8	9	3	10	
Décimo_Aeropuerto	4	5	8	10	3	6	
Noveno_Aeropuerto	0	10	4	10	7	14	
Octavo_Aeropuerto	8	12	1	12	3	4	
Primer_Aeropuerto	7	19	8	6	20	16	
Quinto_Aeropuerto	12	4	18	6	16	0	
Segundo_Aeropuerto	20	12	10	8	11	11	
Sexto_Aeropuerto	12	21	11	20	9	1	
Séptimo_Aeropuerto	4	3	1	4	2	3	
Tercer_Aeropuerto	11	4	14	10	5	6	
Undécimo_Aeropuerto	15	18	11	11	4	1	

year_month	2023-07	2023-08	2023-09	2023-10	2023-11	2023-12	\
nombre_aeropuerto							
Cuarto_Aeropuerto	2	0	5	2	8	5	
Duodécimo_Aeropuerto	8	6	6	9	0	9	
Décimo_Aeropuerto	10	7	1	9	7	6	
Noveno_Aeropuerto	10	1	0	4	4	15	
Octavo_Aeropuerto	15	11	11	14	4	12	
Primer_Aeropuerto	3	19	25	2	18	16	
Quinto_Aeropuerto	12	5	8	8	6	16	
Segundo_Aeropuerto	19	3	1	10	11	2	

```
Sexto_Aeropuerto            11        0        11        25        21        4
Séptimo_Aeropuerto           2       10         5         6         6        2
Tercer_Aeropuerto            2       14         2        15         3        5
Undécimo_Aeropuerto         15       18        16         1        13        9
```

```
year_month            2024-01  2024-02  2024-03
nombre_aeropuerto
Cuarto_Aeropuerto           7       10        7
Duodécimo_Aeropuerto        3        9        0
Décimo_Aeropuerto           3        2        3
Noveno_Aeropuerto           3        0        5
Octavo_Aeropuerto           9        3       13
Primer_Aeropuerto           9        9       25
Quinto_Aeropuerto           6       10        4
Segundo_Aeropuerto          4        5        3
Sexto_Aeropuerto           21       23       23
Séptimo_Aeropuerto          9        9        9
Tercer_Aeropuerto          13        0       14
Undécimo_Aeropuerto        11        3       11
```

[ ]:
```
locales_rentados.shape
```

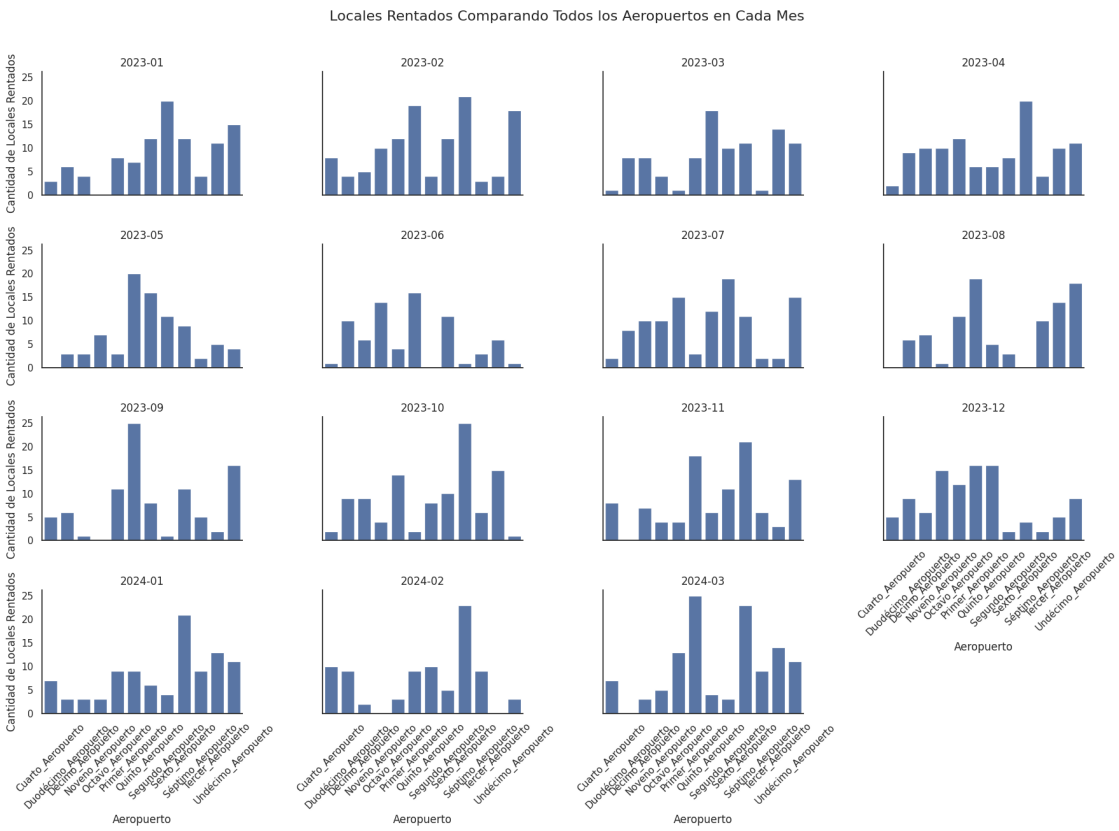
[ ]:
```
(12, 15)
```

Cantidad de Locales Rentados Comparando Todos los Aeropuertos en Cada Mes

Barplot Sencillo

[ ]:
```
# Visualizar Locales Rentados Comparando Todos los Aeropuertos en Cada Mes
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Aseguramos que 'locales_rentados' es un DataFrame con 'nombre_aeropuerto'␣
 ↪como índice y 'year_month' como columnas
#locales_rentados = df.groupby(['nombre_aeropuerto', 'year_month']).size().
 ↪unstack(fill_value=0)

# Transformar el DataFrame de formato ancho a formato largo
locales_rentados_long = locales_rentados.reset_index().
 ↪melt(id_vars='nombre_aeropuerto', var_name='year_month', value_name='Locales␣
 ↪Rentados')

# Convertir 'year_month' a string para facilitar la visualización
locales_rentados_long['year_month'] = locales_rentados_long['year_month'].
 ↪astype(str)

# Configuración de Seaborn
```

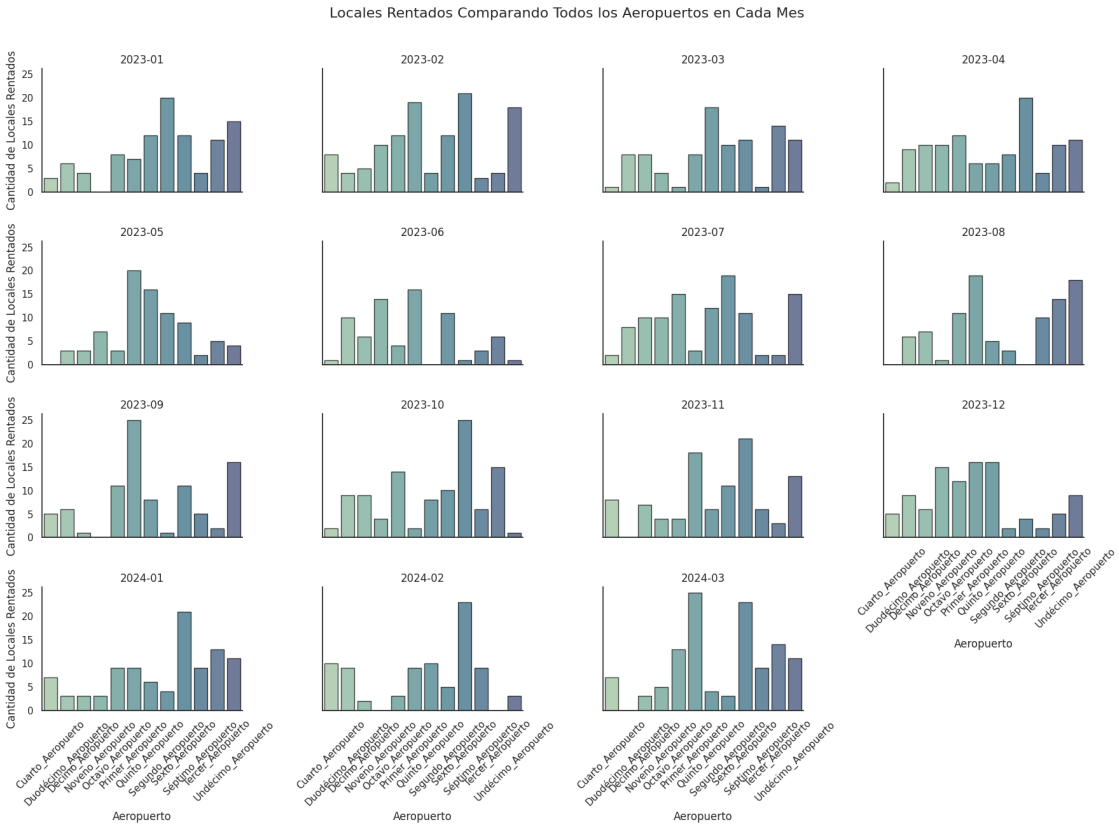
```
sns.set(style="white")

# Crear un FacetGrid para graficar un barplot por cada mes
g = sns.FacetGrid(locales_rentados_long, col='year_month', col_wrap=4,␣
 ↪height=3, aspect=1.5)
g.map_dataframe(sns.barplot, x='nombre_aeropuerto', y='Locales Rentados',␣
 ↪color='b', order=locales_rentados.index)

# Ajustar títulos y etiquetas
g.set_titles('{col_name}')
g.set_axis_labels('Aeropuerto', 'Cantidad de Locales Rentados')
for ax in g.axes.flatten():
    for label in ax.get_xticklabels():
        label.set_rotation(45)

# Ajustar espacio entre gráficos para evitar solapamiento
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Añadir un título general sobre todos los subgráficos
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Locales Rentados Comparando Todos los Aeropuertos en Cada Mes',␣
 ↪fontsize=16)

# Mostrar el gráfico
plt.show()
```



Barplot Estilizado

[ ]:
```
# Visualizar Locales Rentados Comparando Todos los Aeropuertos en Cada Mes
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Aseguramos que 'locales_rentados' es un DataFrame con 'nombre_aeropuerto'␣
 ↪como índice y 'year_month' como columnas
#locales_rentados = df.groupby(['nombre_aeropuerto', 'year_month']).size().
 ↪unstack(fill_value=0)

# Transformar el DataFrame de formato ancho a formato largo
#locales_rentados_long = locales_rentados.reset_index().
 ↪melt(id_vars='nombre_aeropuerto', var_name='year_month', value_name='Locales␣
 ↪Rentados')

# Convertir 'year_month' a string para facilitar la visualización
#locales_rentados_long['year_month'] = locales_rentados_long['year_month'].
 ↪astype(str)
```

```
# Configuración de Seaborn
sns.set(style="white")

# Configurar una paleta de colores
palette = sns.color_palette("crest", n_colors=len(locales_rentados.index))

# Crear el FacetGrid
g = sns.FacetGrid(locales_rentados_long, col='year_month', col_wrap=4,␣
 ↪height=3, aspect=1.5)
g.map_dataframe(sns.barplot, x='nombre_aeropuerto', y='Locales Rentados',␣
 ↪hue='nombre_aeropuerto', palette=palette, order=locales_rentados.index,␣
 ↪legend=False)

# Ajustar propiedades de las barras después de crearlas
for ax in g.axes.flat:
    for bar in ax.patches:
        bar.set_edgecolor('black')  # Ajustar el color de borde
        bar.set_alpha(0.7)  # Ajustar la transparencia


# Ajustar títulos y etiquetas
g.set_titles('{col_name}')
g.set_axis_labels('Aeropuerto', 'Cantidad de Locales Rentados')
for ax in g.axes.flatten():
    for label in ax.get_xticklabels():
        label.set_rotation(45)

# Ajustar espacio entre gráficos para evitar solapamiento
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Añadir un título general sobre todos los subgráficos
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Locales Rentados Comparando Todos los Aeropuertos en Cada Mes',␣
 ↪fontsize=16)

# Mostrar el gráfico
plt.show()
```



Evolución Mensual de Locales Rentados por Aeropuerto

[ ]:
```
locales_rentados
```

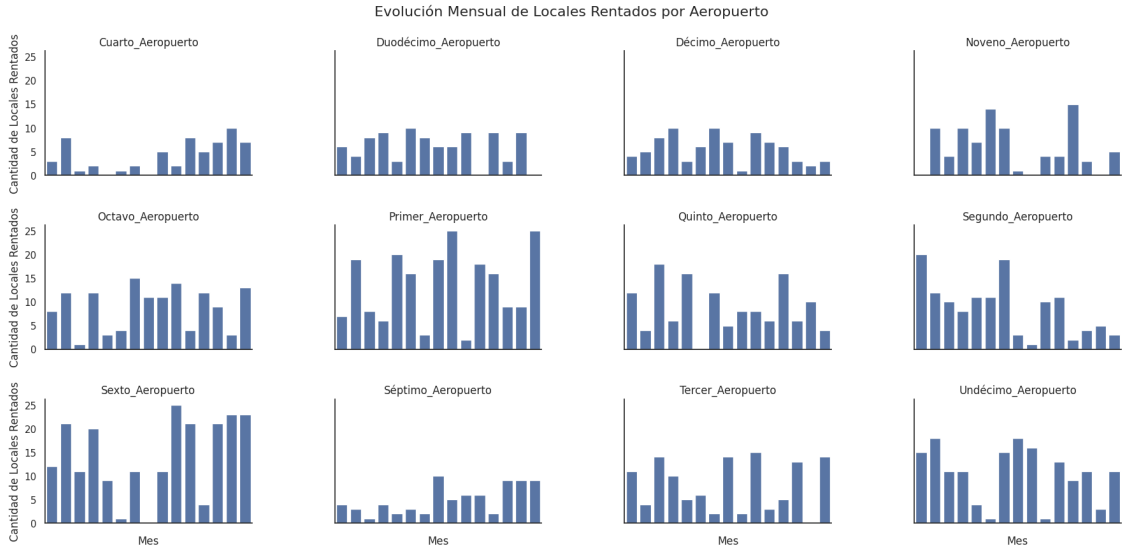
[ ]:
```
year_month              2023-01  2023-02  2023-03  2023-04  2023-05  2023-06  \
nombre_aeropuerto
Cuarto_Aeropuerto             3        8        1        2        0        1
Duodécimo_Aeropuerto          6        4        8        9        3       10
Décimo_Aeropuerto             4        5        8       10        3        6
Noveno_Aeropuerto             0       10        4       10        7       14
Octavo_Aeropuerto             8       12        1       12        3        4
Primer_Aeropuerto             7       19        8        6       20       16
Quinto_Aeropuerto            12        4       18        6       16        0
Segundo_Aeropuerto           20       12       10        8       11       11
Sexto_Aeropuerto             12       21       11       20        9        1
Séptimo_Aeropuerto            4        3        1        4        2        3
Tercer_Aeropuerto            11        4       14       10        5        6
Undécimo_Aeropuerto          15       18       11       11        4        1

year_month              2023-07  2023-08  2023-09  2023-10  2023-11  2023-12  \
nombre_aeropuerto
```

Cuarto_Aeropuerto	2	0	5	2	8	5
Duodécimo_Aeropuerto	8	6	6	9	0	9
Décimo_Aeropuerto	10	7	1	9	7	6
Noveno_Aeropuerto	10	1	0	4	4	15
Octavo_Aeropuerto	15	11	11	14	4	12
Primer_Aeropuerto	3	19	25	2	18	16
Quinto_Aeropuerto	12	5	8	8	6	16
Segundo_Aeropuerto	19	3	1	10	11	2
Sexto_Aeropuerto	11	0	11	25	21	4
Séptimo_Aeropuerto	2	10	5	6	6	2
Tercer_Aeropuerto	2	14	2	15	3	5
Undécimo_Aeropuerto	15	18	16	1	13	9

year_month	2024-01	2024-02	2024-03
nombre_aeropuerto			
Cuarto_Aeropuerto	7	10	7
Duodécimo_Aeropuerto	3	9	0
Décimo_Aeropuerto	3	2	3
Noveno_Aeropuerto	3	0	5
Octavo_Aeropuerto	9	3	13
Primer_Aeropuerto	9	9	25
Quinto_Aeropuerto	6	10	4
Segundo_Aeropuerto	4	5	3
Sexto_Aeropuerto	21	23	23
Séptimo_Aeropuerto	9	9	9
Tercer_Aeropuerto	13	0	14
Undécimo_Aeropuerto	11	3	11

[ ]:
```python
# Visualizar Evolución Mensual de Locales Rentados por Aeropuerto
import seaborn as sns
import matplotlib.pyplot as plt

# Configurar el estilo visual de Seaborn
sns.set(style="white")

# Calcular locales rentados por mes y aeropuerto
locales_rentados_aero = df.groupby(['nombre_aeropuerto', 'year_month']).size().
 ↪reset_index(name='Locales Rentados')

# Convertir 'year_month' a string para facilitar la visualización
locales_rentados_aero['year_month'] = locales_rentados_aero['year_month'].
 ↪astype(str)

# Crear un FacetGrid para graficar un histograma por cada aeropuerto
g = sns.FacetGrid(locales_rentados_aero, col='nombre_aeropuerto', col_wrap=4,␣
 ↪height=3, aspect=1.5)
```

```
g.map(sns.barplot, 'year_month', 'Locales Rentados',␣
  ↪order=sorted(locales_rentados_aero['year_month'].unique()), color='b')

# Ajustar títulos y etiquetas
g.set_titles('{col_name}')
g.set_axis_labels('Mes', 'Cantidad de Locales Rentados')
g.set_xticklabels(rotation=90)

# Ajustar espacio entre gráficos para evitar solapamiento
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Añadir un título general sobre todos los subgráficos
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Evolución Mensual de Locales Rentados por Aeropuerto',␣
  ↪fontsize=16)

# Mostrar el gráfico
plt.show()
```



Montos por Locales Rentados Monto Total de Renta Comparando Todos los Aeropuertos en Cada Mes

[ ]:
```
# Visualizar Monto Total de Renta Comparando Todos los Aeropuertos en Cada Mes
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Asumiendo que tienes monto_total_renta calculado de esta manera:
```

```
monto_total_renta = df.groupby(['nombre_aeropuerto',␣
 ↪'year_month'])['monto_renta'].sum().unstack(fill_value=0)

# Asegurar que 'monto_total_renta' es un DataFrame con 'year_month' como índice␣
 ↪y cada aeropuerto como una columna
monto_total_renta_long = monto_total_renta.reset_index().
 ↪melt(id_vars='nombre_aeropuerto', var_name='year_month', value_name='Monto␣
 ↪Total de Renta')

# Convertir 'year_month' a string para facilitar la visualización
monto_total_renta_long['year_month'] = monto_total_renta_long['year_month'].
 ↪astype(str)

# Mostrar Tabla de Datos
print("Monto Total de Renta Comparando Todos los Aeropuertos en Cada Mes")
monto_total_renta_long
```

```
Monto Total de Renta Comparando Todos los Aeropuertos en Cada Mes
```

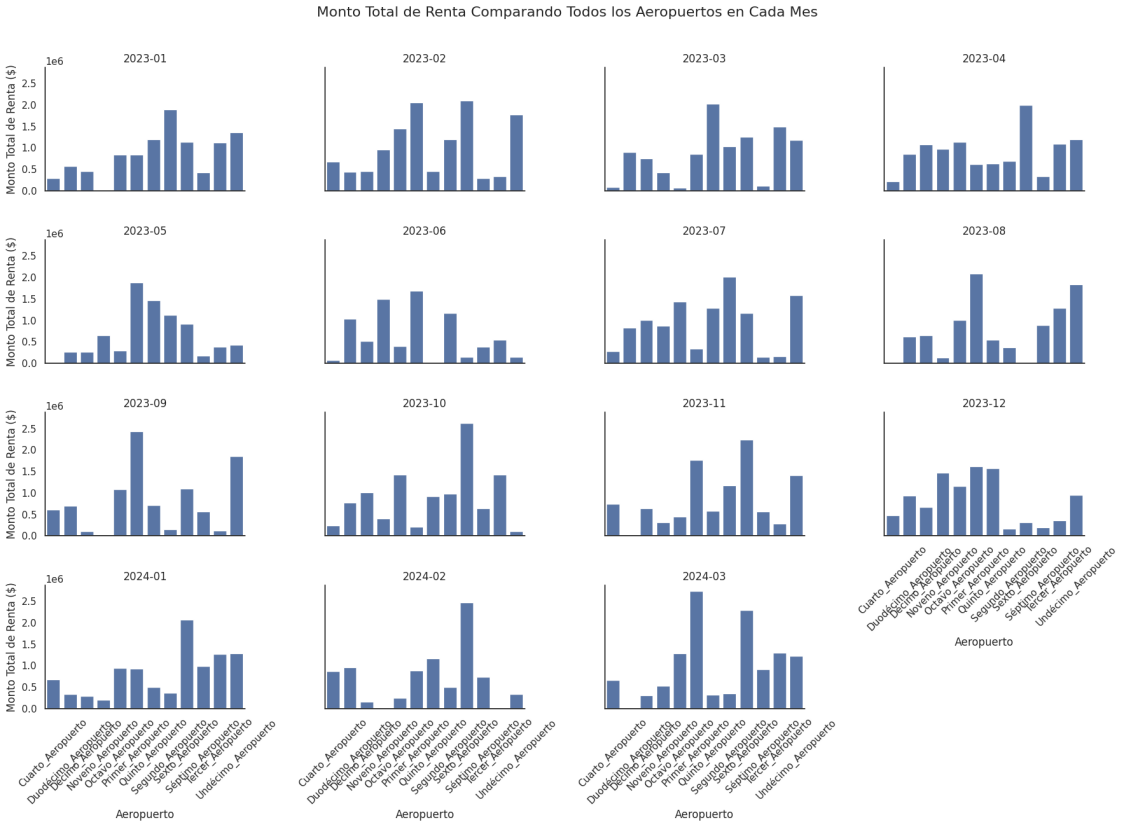
[ ]:

	nombre_aeropuerto	year_month	Monto Total de Renta
0	Cuarto_Aeropuerto	2023-01	298478.72
1	Duodécimo_Aeropuerto	2023-01	569890.66
2	Décimo_Aeropuerto	2023-01	449884.55
3	Noveno_Aeropuerto	2023-01	0.00
4	Octavo_Aeropuerto	2023-01	843791.25
..	…	…	…
175	Segundo_Aeropuerto	2024-03	350250.47
176	Sexto_Aeropuerto	2024-03	2283737.68
177	Séptimo_Aeropuerto	2024-03	905677.99
178	Tercer_Aeropuerto	2024-03	1292176.17
179	Undécimo_Aeropuerto	2024-03	1218812.92

[180 rows x 3 columns]

[ ]:
```
# Visualizar Monto Total de Renta Comparando Todos los Aeropuertos en Cada Mes
# Configuración de Seaborn
sns.set(style="white")

# Crear un FacetGrid para graficar un barplot por cada mes
g = sns.FacetGrid(monto_total_renta_long, col='year_month', col_wrap=4,␣
 ↪height=3, aspect=1.5)
g.map_dataframe(sns.barplot, x='nombre_aeropuerto', y='Monto Total de Renta',␣
 ↪color='b', order=sorted(monto_total_renta_long['nombre_aeropuerto'].
 ↪unique()))

# Ajustar títulos y etiquetas
g.set_titles('{col_name}')
```

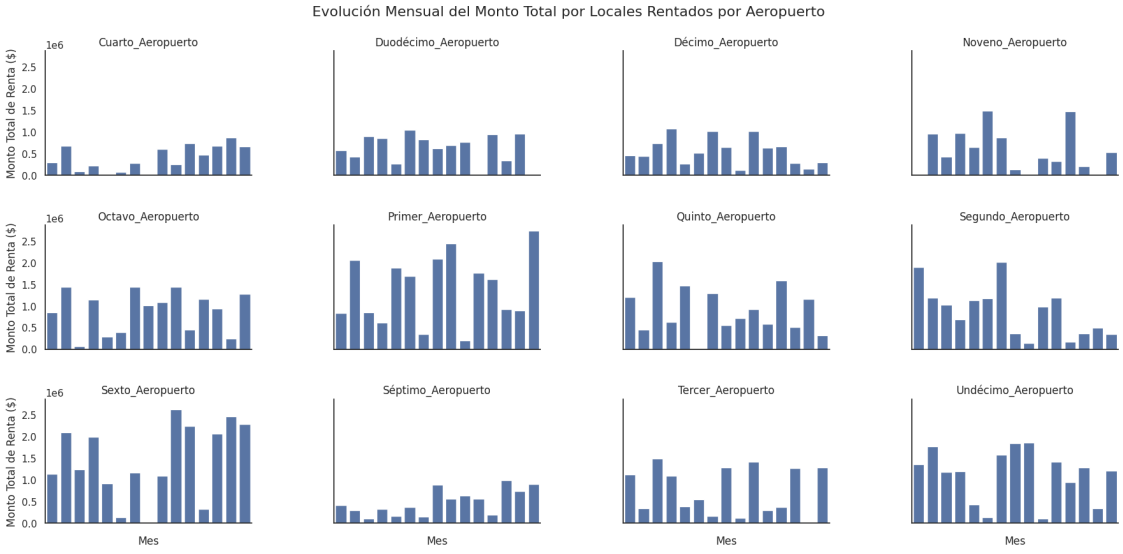
```
g.set_axis_labels('Aeropuerto', 'Monto Total de Renta ($)')
for ax in g.axes.flatten():
    for label in ax.get_xticklabels():
        label.set_rotation(45)

# Ajustar espacio entre gráficos para evitar solapamiento
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Añadir un título general sobre todos los subgráficos
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Monto Total de Renta Comparando Todos los Aeropuertos en Cada␣
 ↪Mes', fontsize=16)

# Mostrar el gráfico
plt.show()
```



Evolución Mensual del Monto Total por Locales Rentados por Aeropuerto

[ ]:
```
# Visualizar la Evolución Mensual del Monto Total por Locales Rentados por␣
 ↪Aeropuerto
import seaborn as sns
```

```
import matplotlib.pyplot as plt

# Supongamos que monto_total_renta_long es tu DataFrame preparado para Seaborn
# Configurar el estilo visual de Seaborn
sns.set(style="white")

# Convertir 'year_month' a string para facilitar la visualización
monto_total_renta_long['year_month'] = monto_total_renta_long['year_month'].
 ↪astype(str)

# Crear el FacetGrid para graficar un histograma por cada aeropuerto
g = sns.FacetGrid(monto_total_renta_long, col='nombre_aeropuerto', col_wrap=4,␣
 ↪height=3, aspect=1.5)
g.map(sns.barplot, 'year_month', 'Monto Total de Renta',␣
 ↪order=sorted(monto_total_renta_long['year_month'].unique()), color='b')

# Ajustar títulos y etiquetas
g.set_titles('{col_name}')
g.set_axis_labels('Mes', 'Monto Total de Renta ($)')
g.set_xticklabels(rotation=90)

# Ajustar espacio entre gráficos para evitar solapamiento
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Añadir un título general sobre todos los subgráficos
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Evolución Mensual del Monto Total por Locales Rentados por␣
 ↪Aeropuerto', fontsize=16)

# Mostrar el gráfico
plt.show()
```



Locales No Rentados

Cantidad de Locales No Rentados

[ ]:
```
# Calcular Locales No Rentados
locales_no_rentados = locales_rentados.apply(lambda row:␣
 ↪locales_disponibles[row.name] - row, axis=1)

# Mostrar resultados
print("Locales No Rentados por Aeropuerto y Mes:\n")
locales_no_rentados
```

```
Locales No Rentados por Aeropuerto y Mes:
```

[ ]:
```
year_month            2023-01  2023-02  2023-03  2023-04  2023-05  2023-06  \
nombre_aeropuerto
Cuarto_Aeropuerto           7        2        9        8       10        9
Duodécimo_Aeropuerto        4        6        2        1        7        0
Décimo_Aeropuerto           6        5        2        0        7        4
Noveno_Aeropuerto          15        5       11        5        8        1
Octavo_Aeropuerto           7        3       14        3       12       11
Primer_Aeropuerto          18        6       17       19        5        9
Quinto_Aeropuerto           8       16        2       14        4       20
Segundo_Aeropuerto          0        8       10       12        9        9
Sexto_Aeropuerto           13        4       14        5       16       24
Séptimo_Aeropuerto          6        7        9        6        8        7
Tercer_Aeropuerto           4       11        1        5       10        9
Undécimo_Aeropuerto         5        2        9        9       16       19

year_month            2023-07  2023-08  2023-09  2023-10  2023-11  2023-12  \
nombre_aeropuerto
Cuarto_Aeropuerto           8       10        5        8        2        5
Duodécimo_Aeropuerto        2        4        4        1       10        1
Décimo_Aeropuerto           0        3        9        1        3        4
Noveno_Aeropuerto           5       14       15       11       11        0
Octavo_Aeropuerto           0        4        4        1       11        3
Primer_Aeropuerto          22        6        0       23        7        9
Quinto_Aeropuerto           8       15       12       12       14        4
Segundo_Aeropuerto          1       17       19       10        9       18
Sexto_Aeropuerto           14       25       14        0        4       21
Séptimo_Aeropuerto          8        0        5        4        4        8
Tercer_Aeropuerto          13        1       13        0       12       10
Undécimo_Aeropuerto         5        2        4       19        7       11
```

year_month	2024-01	2024-02	2024-03
nombre_aeropuerto			
Cuarto_Aeropuerto	3	0	3
Duodécimo_Aeropuerto	7	1	10
Décimo_Aeropuerto	7	8	7
Noveno_Aeropuerto	12	15	10
Octavo_Aeropuerto	6	12	2
Primer_Aeropuerto	16	16	0
Quinto_Aeropuerto	14	10	16
Segundo_Aeropuerto	16	15	17
Sexto_Aeropuerto	4	2	2
Séptimo_Aeropuerto	1	1	1
Tercer_Aeropuerto	2	15	1
Undécimo_Aeropuerto	9	17	9

Exportar Locales No Rentados

[ ]:
```
# Exportar a CSV Locales No Rentados
locales_no_rentados.to_csv('aeropuertos_locales_no_rentados.csv', index=False)
```

1.7.2 Análisis de Clientes Activos

[ ]:
```
# Análisis de continuidad de clientes
df.sort_values(by=['nombre_arrendatario', 'fecha'], inplace=True)
df['continuidad'] = df.groupby('nombre_arrendatario')['fecha'].diff().dt.days␣
 ↪<= 30
clientes_continuos = df[df['continuidad']]
```

[ ]:
```
# Mostrar
print("Análisis de Continuidad de Clientes:\n")
clientes_continuos.head()
```

```
Análisis de Continuidad de Clientes:
```

[ ]:

	fecha	nombre_aeropuerto	terminal	planta	local_id	local_categoria	\
30	2023-01-31	Tercer_Aeropuerto	T1	Baja	UQM-16	Transportes	
40	2023-01-31	Cuarto_Aeropuerto	T2	Alta	Nhp-03	Transportes	
66	2023-01-31	Séptimo_Aeropuerto	T2	Baja	SAI-79	Transportes	
67	2023-01-31	Séptimo_Aeropuerto	T2	Baja	umJ-59	Transportes	
78	2023-01-31	Décimo_Aeropuerto	T1	Alta	wrN-89	Transportes	

	local_subcateg	nombre_arrendatario	telefono	horario	monto_renta	\
30	Taxis	Cabina Naranja	74-9578-9421	['24hrs']	72155.00	
40	Taxis	Cabina Naranja	40-6301-1293	['24hrs']	67083.05	
66	Taxis	Cabina Naranja	96-3050-8397	['24hrs']	53716.13	
67	Taxis	Cabina Naranja	73-7667-4714	['24hrs']	137216.42	

```
78          Taxis      Cabina Naranja  73-5342-7845  ['24hrs']    137264.92

    monto_renta_usd   deposito fecha_corte fecha_pago  monto_pago  \
30          3607.75   72155.00  2023-01-31 2023-02-03    69548.68
40          3354.15   67083.05  2023-01-31 2023-02-01    63999.81
66          2685.81   53716.13  2023-01-31 2023-02-05    53410.38
67          6860.82  137216.42  2023-01-31 2023-02-01   132437.98
78          6863.25  137264.92  2023-01-31 2023-02-05   134633.70

    pago_pendiente_mes  cliente_con_atraso_mes year_month  continuidad
30             2606.32                       1    2023-01         True
40             3083.24                       1    2023-01         True
66              305.75                       1    2023-01         True
67             4778.44                       1    2023-01         True
78             2631.22                       1    2023-01         True
```

[ ]:
```python
# Filas y Columnas
clientes_continuos.shape
```

[ ]:
```
(1218, 20)
```

Heatmap Este gráfico muestra un **Heatmap (Mapa de Calor)** donde las filas representan clientes y las columnas representan meses.

Un color más intenso indica la presencia de continuidad en los pagos (true), mientras que un color más claro indica la ausencia de la misma (false).

Warning!

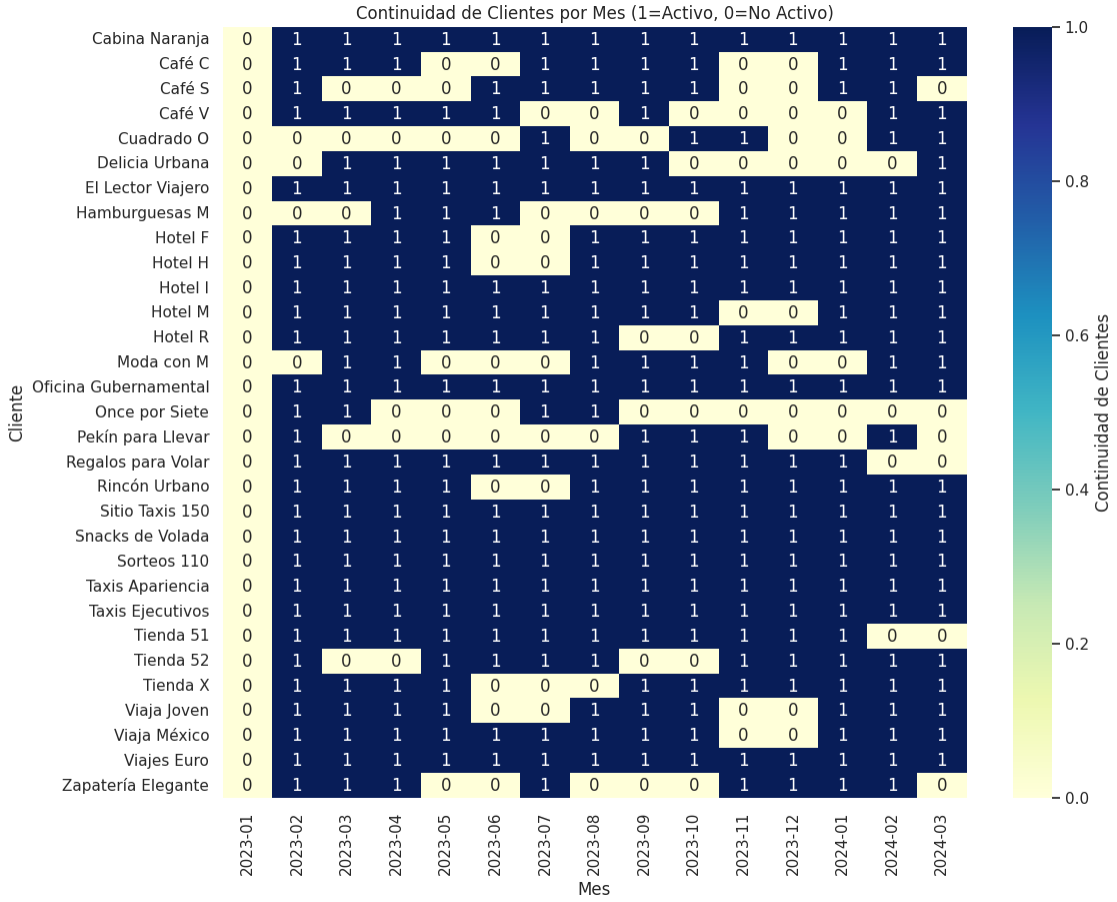
- El primer mes del Heatmap mostrará todos los valores en **'0'**, debido a que no hay mes previo con el cual comparar si el cliente (la empresa) existía o no.

Interpretación:

- **'0'**: Significa que ese mes, la Empresa en cuestión ***no fue Cliente Activo***.
- **'1'**: Significa que ese mes, la Empresa en cuestión ***fue Cliente Activo***.

[ ]:
```python
# Heatmap de Clientes Activos
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Asegurando que la columna 'fecha' está en formato datetime
df['fecha'] = pd.to_datetime(df['fecha'])

# Crear una columna 'Mes' para agrupar por mes de manera más sencilla
df['Mes'] = df['fecha'].dt.to_period('M')
```

```
# Ordenar los datos por cliente y fecha para asegurar que el cálculo de␣
 ↪continuidad es correcto
df.sort_values(by=['nombre_arrendatario', 'fecha'], inplace=True)

# Calcular la continuidad como la presencia de transacciones en meses␣
 ↪consecutivos
df['continuidad'] = df.groupby('nombre_arrendatario')['Mes'].diff().
 ↪apply(lambda x: x.n == 1 if pd.notna(x) else False)

# Crear un DataFrame resumido que indica si un cliente tuvo transacciones␣
 ↪continuas cada mes
resumen_continuidad = df.groupby(['nombre_arrendatario', 'Mes'])['continuidad'].
 ↪any().unstack(fill_value=False)

# Heatmap para visualizar la continuidad de clientes
plt.figure(figsize=(12, 10))
sns.heatmap(resumen_continuidad, annot=True, cmap="YlGnBu", fmt="g",␣
 ↪cbar_kws={'label': 'Continuidad de Clientes'})
plt.title('Continuidad de Clientes por Mes (1=Activo, 0=No Activo)')
plt.xlabel('Mes')
plt.ylabel('Cliente')
plt.show()
```



1.7.3 Análisis de Pagos Atrasados

[ ]:
```
# Agrupar
# Suponiendo que df es tu DataFrame
agrupado = df.groupby(['nombre_arrendatario', 'nombre_aeropuerto'])
```

[ ]:
```
# Analizar Pagos
# Añadir una columna que indique si el pago fue tardío
df['pago_tardio'] = df['fecha_pago'] > df['fecha_corte']
```

[ ]:
```
# Empresas con Atrasos en Múltiples Aeropuertos
# Filtrar empresas con pagos tardíos
empresas_con_atrasos = df[df['pago_tardio']]

# Agrupar por empresa y aeropuerto para contar en cuántos aeropuertos están␣
 ↪presentes con atrasos
atrasos_por_aeropuerto = empresas_con_atrasos.
 ↪groupby('nombre_arrendatario')['nombre_aeropuerto'].nunique()
```

```
# Filtrar empresas presentes en más de un aeropuerto con atrasos
empresas_problemas_multiples = atrasos_por_aeropuerto[atrasos_por_aeropuerto >␣
 ↪1].index

# Detalles de las empresas con problemas en múltiples aeropuertos
detalles_empresas_problemas = df[df['nombre_arrendatario'].
 ↪isin(empresas_problemas_multiples)]
```

[ ]:
```
# Visualizar Clientes con Atrasos
detalles_empresas_problemas[['nombre_arrendatario', 'nombre_aeropuerto',␣
 ↪'fecha_pago', 'monto_pago', 'pago_tardio']]
print("Clientes (Empresas) con Atrasos en Pagos:\n")
detalles_empresas_problemas.head()
```

```
Clientes (Empresas) con Atrasos en Pagos:
```

[ ]:
```
        fecha   nombre_aeropuerto terminal planta local_id local_categoria  \
22 2023-01-31  Segundo_Aeropuerto       T2   Baja   QWI-44     Transportes
30 2023-01-31   Tercer_Aeropuerto       T1   Baja   UQM-16     Transportes
40 2023-01-31   Cuarto_Aeropuerto       T2   Alta   Nhp-03     Transportes
66 2023-01-31  Séptimo_Aeropuerto       T2   Baja   SAI-79     Transportes
67 2023-01-31  Séptimo_Aeropuerto       T2   Baja   umJ-59     Transportes

   local_subcateg nombre_arrendatario      telefono    horario  …  \
22          Taxis      Cabina Naranja  75-0011-8971  ['24hrs']  …
30          Taxis      Cabina Naranja  74-9578-9421  ['24hrs']  …
40          Taxis      Cabina Naranja  40-6301-1293  ['24hrs']  …
66          Taxis      Cabina Naranja  96-3050-8397  ['24hrs']  …
67          Taxis      Cabina Naranja  73-7667-4714  ['24hrs']  …

     deposito  fecha_corte  fecha_pago monto_pago pago_pendiente_mes  \
22   61903.04   2023-01-31  2023-02-01   61411.94             491.10
30   72155.00   2023-01-31  2023-02-03   69548.68            2606.32
40   67083.05   2023-01-31  2023-02-01   63999.81            3083.24
66   53716.13   2023-01-31  2023-02-05   53410.38             305.75
67  137216.42   2023-01-31  2023-02-01  132437.98            4778.44

    cliente_con_atraso_mes  year_month  continuidad      Mes  pago_tardio
22                       1     2023-01        False  2023-01         True
30                       1     2023-01        False  2023-01         True
40                       1     2023-01        False  2023-01         True
66                       1     2023-01        False  2023-01         True
67                       1     2023-01        False  2023-01         True

[5 rows x 22 columns]
```

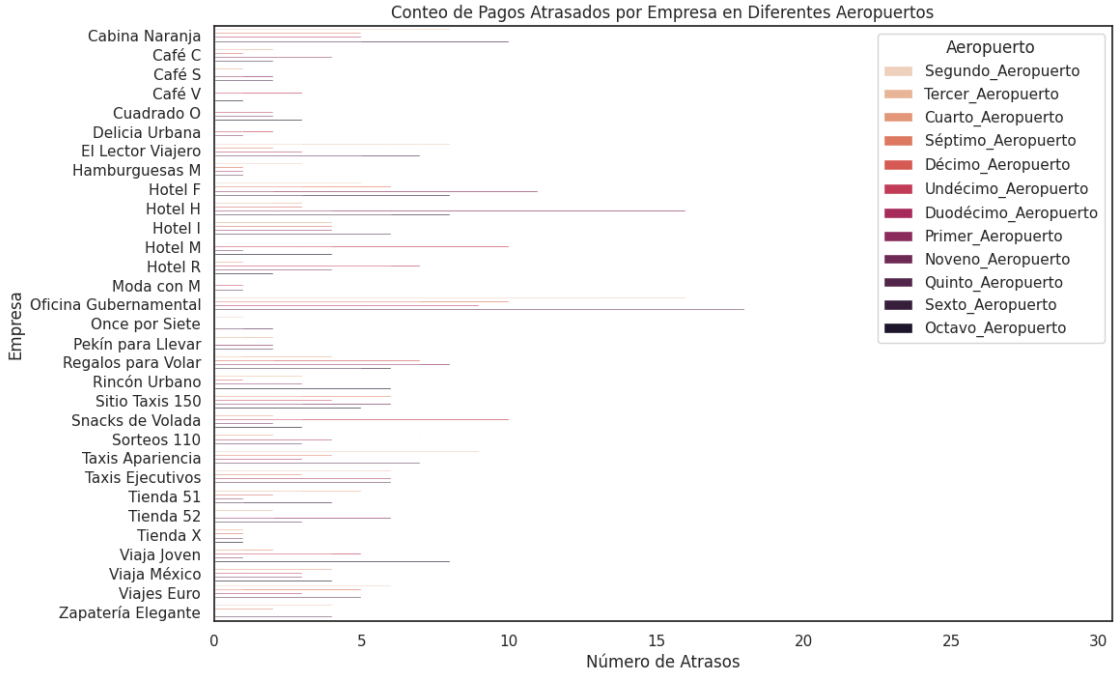
Exportar Clientes (Empresas) con Atraso en Pagos

[ ]:
```
# Exportar a CSV Clientes con Atraso en Pagos
detalles_empresas_problemas.to_csv('aeropuertos_empresas_atraso_pagos.csv',␣
↪index=False)
```

Countplot

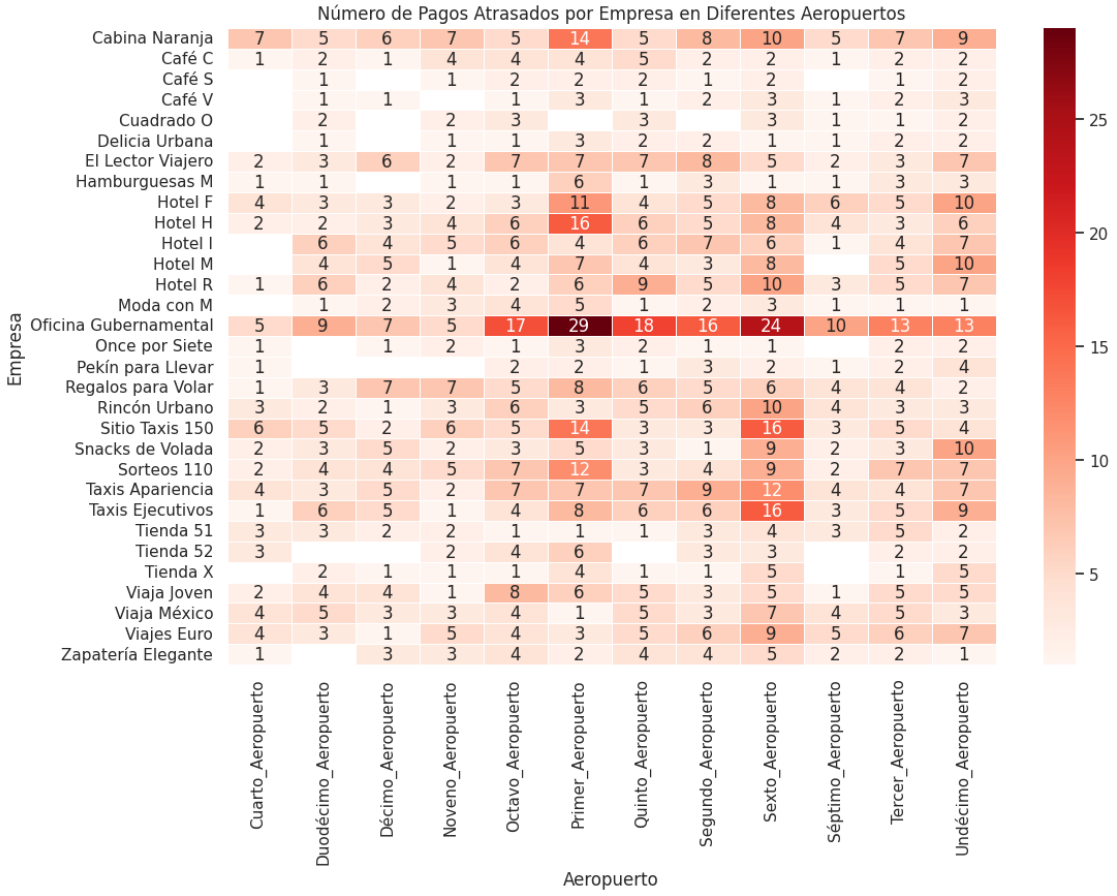
[ ]:
```
# palette:
# 'deep', 'muted', 'bright', 'pastel', 'dark', 'colorblind', 'husl'
# 'RdYlBu', 'magma', 'YlOrBr', 'crest', 'rocket_r', 'mako'
```

[ ]:
```
# Countplot de Pagos Atrasados por Empresa en Diferentes Aeropuertos
# Crear un DataFrame para visualización
visualizacion_atrasos = detalles_empresas_problemas.
↪groupby(['nombre_arrendatario', 'nombre_aeropuerto']).size().
↪reset_index(name='Numero_de_Atrasos')

# Countplot
plt.figure(figsize=(12, 8))
sns.countplot(data=detalles_empresas_problemas, y='nombre_arrendatario',␣
↪hue='nombre_aeropuerto', palette='rocket_r')
plt.title("Conteo de Pagos Atrasados por Empresa en Diferentes Aeropuertos")
plt.xlabel("Número de Atrasos")
plt.ylabel("Empresa")
plt.legend(title="Aeropuerto", loc='upper right')
plt.show()
```



Heatmap

[ ]:
```
# Heatmap de Pagos Atrasados por Empresa en Diferentes Aeropuertos
import seaborn as sns
import matplotlib.pyplot as plt

# Correcta transformación del DataFrame para el heatmap
heatmap_data = visualizacion_atrasos.pivot(index="nombre_arrendatario",␣
 ↪columns="nombre_aeropuerto", values="Numero_de_Atrasos")

# Crear el heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, cmap="Reds", fmt="g", linewidths=.5)  #␣
 ↪'g' es el especificador general para números en `fmt`
plt.title("Número de Pagos Atrasados por Empresa en Diferentes Aeropuertos")
plt.ylabel("Empresa")
plt.xlabel("Aeropuerto")
plt.show()
```



1.8 Fin