In [9]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Cargar los datos
customer = pd.read_parquet('customer.parquet')
film = pd.read_parquet('film.parquet')
inventory = pd.read_parquet('inventory.parquet')
rental = pd.read_parquet('rental.parquet')
store = pd.read_parquet('store.parquet')
```

In [10]:
```python
def convertir_columnas(df, columnas_fecha, columnas_texto):
    """
    Convierte las columnas de un DataFrame a los tipos de datos correctos.
    - columnas_fecha: Lista de columnas que deben ser convertidas a datetime.
    - columnas_texto: Lista de columnas que deben mantenerse como object (texto).
    """
    for columna in df.columns:
        if columna in columnas_fecha:
            # Convertir columnas de fecha a datetime
            try:
                df[columna] = pd.to_datetime(df[columna])
                #print(f"Columna '{columna}' convertida a datetime.")
            except (ValueError, TypeError):
                #print(f"Columna '{columna}' no pudo ser convertida a datetime. Manteniendo como object.")
                df[columna] = df[columna].astype(object)
        elif columna in columnas_texto:
            # Mantener columnas de texto como object
            df[columna] = df[columna].astype(object)
            #print(f"Columna '{columna}' mantenida como object.")
        else:
            # Intentar convertir columnas numéricas
            try:
                # Convertir a float (esto incluye enteros y decimales)
                df[columna] = pd.to_numeric(df[columna], errors='raise')

                # Verificar si todos los valores son enteros
                if (df[columna] % 1 == 0).all():
                    df[columna] = df[columna].astype(int)  # Convertir a entero
                    #print(f"Columna '{columna}' convertida a int.")
                else:
                    df[columna] = df[columna].astype(float)  # Convertir a float
                    #print(f"Columna '{columna}' convertida a float.")
            except (ValueError, TypeError):
                # Si no se puede convertir a numérico, mantener como object (texto)
                df[columna] = df[columna].astype(object)
                #print(f"Columna '{columna}' mantenida como object.")

    return df

# Definir las columnas que son fechas y las que son texto
columnas_fecha = ["create_date", "last_update", "rental_date", "return_date"]
columnas_texto = ["first_name", "last_name", "email", "title", "description", "rating", "special_features", "cus


# Aplicar la función a cada DataFrame
customer = convertir_columnas(customer, columnas_fecha, columnas_texto)
film = convertir_columnas(film, columnas_fecha, columnas_texto)
inventory = convertir_columnas(inventory, columnas_fecha, columnas_texto)
rental = convertir_columnas(rental, columnas_fecha, columnas_texto)
store = convertir_columnas(store, columnas_fecha, columnas_texto)
```

In [11]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Supongamos que ya tienes cargados los DataFrames: customer, film, inventory, rental, store

# Diccionario de DataFrames
dataframes = {
    "customer": customer,
    "film": film,
    "inventory": inventory,
    "rental": rental,
    "store": store
}



# Iterar sobre cada DataFrame
for i, (name, df) in enumerate(dataframes.items()):
```

```
    # Valores nulos
    null_counts = df.isnull().sum()

    # Tipos de datos
    dtypes = df.dtypes

    # Registros duplicados
    duplicated_counts = df.duplicated().sum()

    # Crear un DataFrame para mostrar la información
    info_df = pd.DataFrame({
        'Columna': df.columns,
        'Tipo de Dato': dtypes.astype(str),  # Convertir tipos de datos a cadena
        'Valores Nulos': null_counts
    })

    # Mostrar información en la consola
    print(f"Información para {name}:")
    print(info_df)
    print(f"Registros duplicados en {name}: {duplicated_counts}\n")
```

```
Información para customer:
                         Columna    Tipo de Dato  Valores Nulos
customer_id          customer_id           int32              0
store_id                store_id           int32              0
first_name           first_name          object              0
last_name             last_name          object              0
email                     email          object              0
address_id           address_id           int32              0
active                   active           int32              0
create_date          create_date  datetime64[ns]              0
last_update          last_update  datetime64[ns]              0
customer_id_old  customer_id_old          object              0
segment                  segment          object              0
Registros duplicados en customer: 0

Información para film:
                                  Columna    Tipo de Dato  Valores Nulos
film_id                           film_id           int32              0
title                               title          object              0
description                   description          object              0
release_year                 release_year           int32              0
language_id                   language_id           int32              0
original_language_id  original_language_id           int32              0
rental_duration            rental_duration           int32              0
rental_rate                    rental_rate         float64              0
length                              length           int32              0
replacement_cost          replacement_cost         float64              0
num_voted_users            num_voted_users           int32              0
rating                              rating          object              0
special_features          special_features          object              0
last_update                    last_update  datetime64[ns]              0
Registros duplicados en film: 0

Información para inventory:
                   Columna    Tipo de Dato  Valores Nulos
inventory_id  inventory_id           int32              0
film_id            film_id           int32              0
store_id          store_id           int32              0
last_update    last_update  datetime64[ns]              0
Registros duplicados en inventory: 0

Información para rental:
                   Columna    Tipo de Dato  Valores Nulos
rental_id        rental_id           int32              0
rental_date    rental_date  datetime64[ns]              0
inventory_id  inventory_id           int32              0
customer_id    customer_id           int32              0
return_date    return_date          object              0
staff_id          staff_id           int32              0
last_update    last_update  datetime64[ns]              0
Registros duplicados en rental: 0

Información para store:
                          Columna    Tipo de Dato  Valores Nulos
store_id                 store_id           int32              0
manager_staff_id  manager_staff_id           int32              0
address_id              address_id           int32              0
last_update            last_update  datetime64[ns]              0
Registros duplicados en store: 0
```

In [4]:
```
customer.head(5)
```

Out[4]:

	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
0	155	1	GAIL	KNIGHT	GAIL.KNIGHT@sakilacustomer.org	159	1	2006-02-15 03:04:36	2006-02-15 09:57:20
1	174	2	YVONNE	WATKINS	YVONNE.WATKINS@sakilacustomer.org	178	1	2006-02-15 03:04:36	2006-02-15 09:57:20
2	229	1	TAMARA	NGUYEN	TAMARA.NGUYEN@sakilacustomer.org	233	1	2006-02-15 03:04:36	2006-02-15 09:57:20
3	270	1	LEAH	CURTIS	LEAH.CURTIS@sakilacustomer.org	275	1	2006-02-15 03:04:36	2006-02-15 09:57:20
4	326	1	JOSE	ANDREW	JOSE.ANDREW@sakilacustomer.org	331	1	2006-02-15 03:04:37	2006-02-15 09:57:20

In [5]:
```
customer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1392 entries, 0 to 1391
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   customer_id      1392 non-null   int32         
 1   store_id         1392 non-null   int32         
 2   first_name       1392 non-null   object        
 3   last_name        1392 non-null   object        
 4   email            1392 non-null   object        
 5   address_id       1392 non-null   int32         
 6   active           1392 non-null   int32         
 7   create_date      1392 non-null   datetime64[ns]
 8   last_update      1392 non-null   datetime64[ns]
 9   customer_id_old  1392 non-null   object        
 10  segment          1392 non-null   object        
dtypes: datetime64[ns](2), int32(4), object(5)
memory usage: 98.0+ KB
```

In [6]:
```
customer.describe()
```

Out[6]:

	customer_id	store_id	address_id	active	create_date	last_update
count	1392.000000	1392.000000	1392.000000	1392.000000	1392	1392
mean	696.500000	1.480603	476.356322	0.989224	2014-01-25 05:15:09.965517056	2014-01-25 03:03:48.563218432
min	1.000000	1.000000	5.000000	0.000000	2006-02-15 03:04:36	2006-02-15 09:57:20
25%	348.750000	1.000000	353.750000	1.000000	2006-02-15 03:04:37	2006-02-15 09:57:20
50%	696.500000	1.000000	606.000000	1.000000	2020-01-25 14:02:20	2020-01-25 05:00:00
75%	1044.250000	2.000000	606.000000	1.000000	2020-01-25 14:02:20	2020-01-25 05:00:00
max	1392.000000	2.000000	606.000000	1.000000	2020-01-25 14:02:20	2020-01-25 05:00:00
std	401.980099	0.499803	187.717263	0.103283	NaN	NaN

In [7]:
```
film.head(5)
```

Out[7]:

	film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replace
0	141	CHICAGO NORTH	A Fateful Yarn of a Mad Cow And a Waitress wh...	2006	1	0	6	4.99	185	
1	260	DUDE BLINDNESS	A Stunning Reflection of a Husband And a Lumb...	2006	1	0	3	4.99	132	
2	292	EXCITEMENT EVE	A Brilliant Documentary of a Monkey And a Car...	2006	1	0	3	0.99	51	
3	584	MIXED DOORS	A Taut Drama of a Womanizer And a Lumberjack ...	2006	1	0	6	2.99	180	
4	641	ORANGE GRAPES	A Astounding Documentary of a Butler And a Wo...	2006	1	0	4	0.99	76	

In [8]:
```
film.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   film_id               1000 non-null   int32
 1   title                 1000 non-null   object
 2   description           1000 non-null   object
 3   release_year          1000 non-null   int32
 4   language_id           1000 non-null   int32
 5   original_language_id  1000 non-null   int32
 6   rental_duration       1000 non-null   int32
 7   rental_rate           1000 non-null   float64
 8   length                1000 non-null   int32
 9   replacement_cost      1000 non-null   float64
 10  num_voted_users       1000 non-null   int32
 11  rating                1000 non-null   object
 12  special_features      1000 non-null   object
 13  last_update           1000 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(2), int32(7), object(4)
memory usage: 82.2+ KB
```

In [9]:
```
film.describe()
```

Out[9]:

	film_id	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost
count	1000.000000	1000.0	1000.0	1000.0	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	2006.0	1.0	0.0	4.985000	2.980000	115.272000	19.984009
min	1.000000	2006.0	1.0	0.0	3.000000	0.990000	46.000000	9.990000
25%	250.750000	2006.0	1.0	0.0	4.000000	0.990000	80.000000	14.990000
50%	500.500000	2006.0	1.0	0.0	5.000000	2.990000	114.000000	19.990000
75%	750.250000	2006.0	1.0	0.0	6.000000	4.990000	149.250000	24.990000
max	1000.000000	2006.0	1.0	0.0	7.000000	4.990000	185.000000	29.990000
std	288.819436	0.0	0.0	0.0	1.411654	1.646393	40.426332	6.050834

In [10]:
```
inventory.head()
```

Out[10]:

	inventory_id	film_id	store_id	last_update
0	410	90	2	2006-02-15 10:09:17
1	596	130	2	2006-02-15 10:09:17
2	752	164	2	2006-02-15 10:09:17
3	809	176	2	2006-02-15 10:09:17
4	854	188	2	2006-02-15 10:09:17

In [11]:
```
inventory.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4581 entries, 0 to 4580
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   inventory_id  4581 non-null   int32
 1   film_id       4581 non-null   int32
 2   store_id      4581 non-null   int32
 3   last_update   4581 non-null   datetime64[ns]
dtypes: datetime64[ns](1), int32(3)
memory usage: 89.6 KB
```

In [12]:
```
inventory.describe()
```

Out[12]:

	inventory_id	film_id	store_id	last_update
count	4581.000000	4581.000000	4581.000000	4581
mean	2291.000000	500.936258	1.999127	2006-02-15 10:09:16.999999872
min	1.000000	1.000000	1.000000	2006-02-15 10:09:17
25%	1146.000000	253.000000	2.000000	2006-02-15 10:09:17
50%	2291.000000	496.000000	2.000000	2006-02-15 10:09:17
75%	3436.000000	753.000000	2.000000	2006-02-15 10:09:17
max	4581.000000	1000.000000	2.000000	2006-02-15 10:09:17
std	1322.565121	288.589650	0.029540	NaN

In [13]:
```
rental.head()
```

Out[13]:

	rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
0	297	2005-05-27 01:48:48	1594	48	2005-05-27 19:52:48	2	2006-02-16 02:30:53
1	472	2005-05-28 02:36:15	1338	528	2005-05-29 21:07:15	1	2006-02-16 02:30:53
2	480	2005-05-28 03:47:39	2108	220	2005-06-04 21:17:39	2	2006-02-16 02:30:53
3	510	2005-05-28 07:52:14	4338	113	2005-05-30 21:20:14	2	2006-02-16 02:30:53
4	1281	2005-06-15 13:21:39	2963	511	2005-06-17 11:03:39	1	2006-02-16 02:30:53

In [14]:
```
rental.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16044 entries, 0 to 16043
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   rental_id     16044 non-null  int32
 1   rental_date   16044 non-null  datetime64[ns]
 2   inventory_id  16044 non-null  int32
 3   customer_id   16044 non-null  int32
 4   return_date   16044 non-null  object
 5   staff_id      16044 non-null  int32
 6   last_update   16044 non-null  datetime64[ns]
dtypes: datetime64[ns](2), int32(4), object(1)
memory usage: 626.8+ KB
```

In [15]:
```
rental.describe()
```

Out[15]:

	rental_id	rental_date	inventory_id	customer_id	staff_id	last_update
count	16044.000000	16044	16044.000000	16044.000000	16044.000000	16044
mean	8025.371478	2005-07-23 13:12:53.217526912	2291.842558	297.143169	1.498878	2006-02-16 02:31:32.196895616
min	1.000000	2005-05-25 03:53:30	1.000000	1.000000	1.000000	2006-02-16 02:30:53
25%	4013.750000	2005-07-07 05:58:40.500000	1154.000000	148.000000	1.000000	2006-02-16 02:30:53
50%	8025.500000	2005-07-28 21:04:32.500000	2291.000000	296.000000	1.000000	2006-02-16 02:30:53
75%	12037.250000	2005-08-18 02:16:23	3433.000000	446.000000	2.000000	2006-02-16 02:30:53
max	16049.000000	2006-02-14 20:16:03	4581.000000	599.000000	2.000000	2006-02-23 09:12:08
std	4632.777249	NaN	1322.210643	172.453136	0.500014	NaN

In [16]:
```
store.head()
```

Out[16]:

	store_id	manager_staff_id	address_id	last_update
0	1	1	1	2016-02-15 09:57:12
1	2	2	2	2016-02-15 09:57:12

In [17]:
```
store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   store_id          2 non-null      int32         
 1   manager_staff_id  2 non-null      int32         
 2   address_id        2 non-null      int32         
 3   last_update       2 non-null      datetime64[ns]
dtypes: datetime64[ns](1), int32(3)
memory usage: 172.0 bytes
```

In [18]:
```
store.describe()
```

Out[18]:

	store_id	manager_staff_id	address_id	last_update
count	2.000000	2.000000	2.000000	2
mean	1.500000	1.500000	1.500000	2016-02-15 09:57:12
min	1.000000	1.000000	1.000000	2016-02-15 09:57:12
25%	1.250000	1.250000	1.250000	2016-02-15 09:57:12
50%	1.500000	1.500000	1.500000	2016-02-15 09:57:12
75%	1.750000	1.750000	1.750000	2016-02-15 09:57:12
max	2.000000	2.000000	2.000000	2016-02-15 09:57:12
std	0.707107	0.707107	0.707107	NaN

In [19]:
```
# Revisar valores nulos en cada tabla
for name, df in {"customer": customer, "film": film, "inventory": inventory, "rental": rental, "store": store}.
    print(f"Valores nulos en {name}:\n", df.isnull().sum(), "\n")

# Contar registros duplicados
for name, df in {"customer": customer, "film": film, "inventory": inventory, "rental": rental, "store": store}.
    print(f"Registros duplicados en {name}: {df.duplicated().sum()}")
```

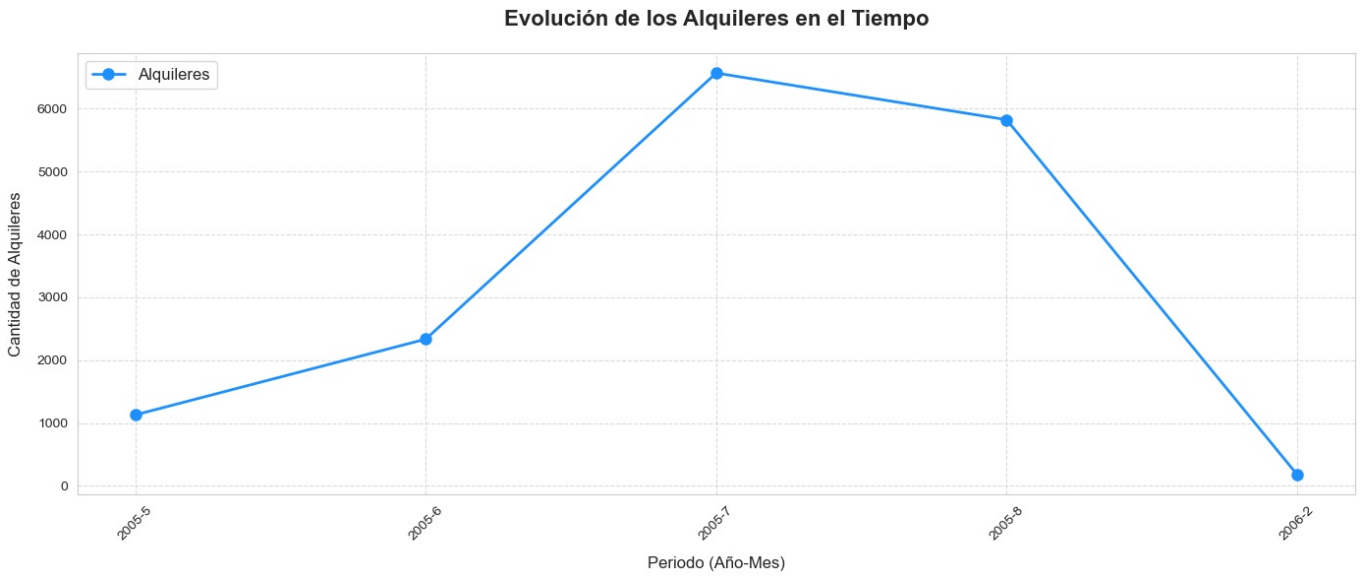
```
Valores nulos en customer:
 customer_id        0
store_id            0
first_name          0
last_name           0
email               0
address_id          0
active              0
create_date         0
last_update         0
customer_id_old     0
segment             0
dtype: int64

Valores nulos en film:
 film_id                  0
title                     0
description               0
release_year              0
language_id               0
original_language_id      0
rental_duration           0
rental_rate               0
length                    0
replacement_cost          0
num_voted_users           0
rating                    0
special_features          0
last_update               0
dtype: int64

Valores nulos en inventory:
 inventory_id    0
film_id          0
store_id         0
last_update      0
dtype: int64

Valores nulos en rental:
 rental_id       0
rental_date      0
inventory_id     0
customer_id      0
return_date      0
staff_id         0
last_update      0
dtype: int64

Valores nulos en store:
 store_id            0
manager_staff_id     0
address_id           0
last_update          0
dtype: int64

Registros duplicados en customer: 0
Registros duplicados en film: 0
Registros duplicados en inventory: 0
Registros duplicados en rental: 0
Registros duplicados en store: 0
```

Distribución de alquileres en el tiempo

In [20]:

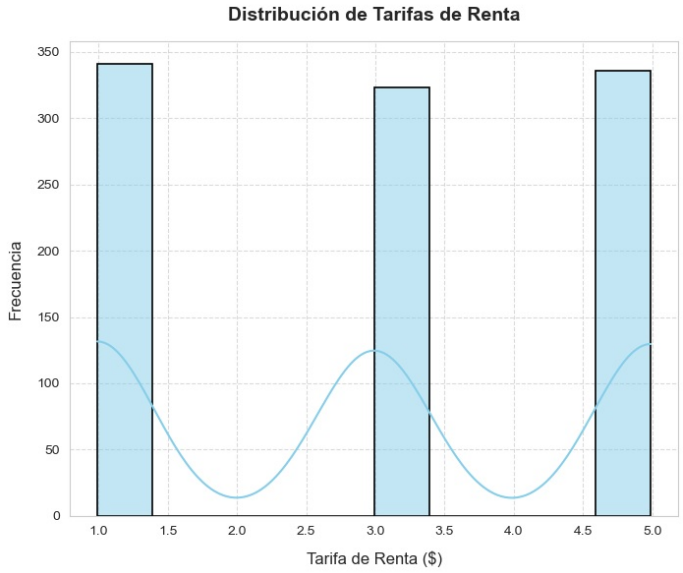
```
# Convertir rental_date a datetime
rental["rental_date"] = pd.to_datetime(rental["rental_date"])

# Crear columnas de año y mes
rental["year"] = rental["rental_date"].dt.year
rental["month"] = rental["rental_date"].dt.month

# Contar alquileres por mes
rentals_per_month = rental.groupby(["year", "month"]).size().reset_index(name="total_rentals")

# Crear una columna con formato Año-Mes para el eje X
rentals_per_month["periodo"] = rentals_per_month["year"].astype(str) + "-" + rentals_per_month["month"].astype(s


sns.set_style("whitegrid")  # Fondo con cuadrícula
sns.set_palette("husl")  # Paleta de colores atractiva

# Graficar
plt.figure(figsize=(14, 6))  # Tamaño más grande para mejor visualización
plt.plot(rentals_per_month["periodo"], rentals_per_month["total_rentals"],
```

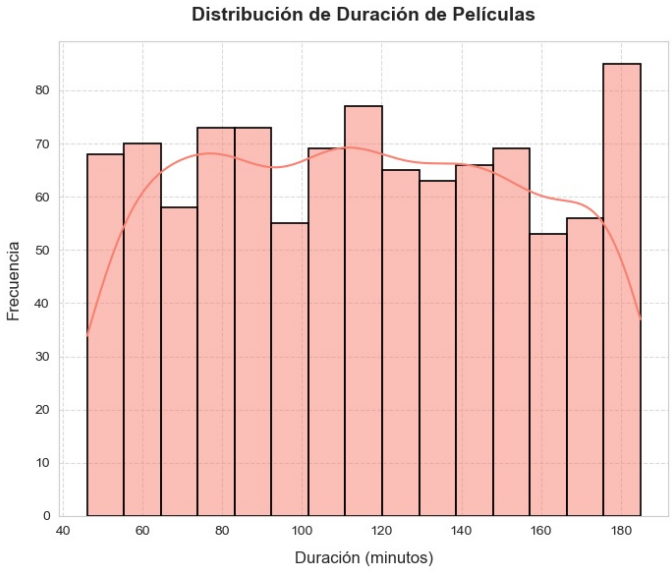
```
        marker="o", linestyle="-", color="dodgerblue", linewidth=2, markersize=8, label="Alquileres")

# título y las etiquetas
plt.title("Evolución de los Alquileres en el Tiempo", fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Periodo (Año-Mes)", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Alquileres", fontsize=12, labelpad=10)

# Rotacion de etiquetas del eje X
plt.xticks(rotation=45, fontsize=10)

# Cuadrícula suave
plt.grid(True, linestyle="--", alpha=0.7)

# Añadir leyenda
plt.legend(loc="upper left", fontsize=12)

# Ajuste del layout
plt.tight_layout()

plt.show()
```



Distribución de precios de renta y duración de películas

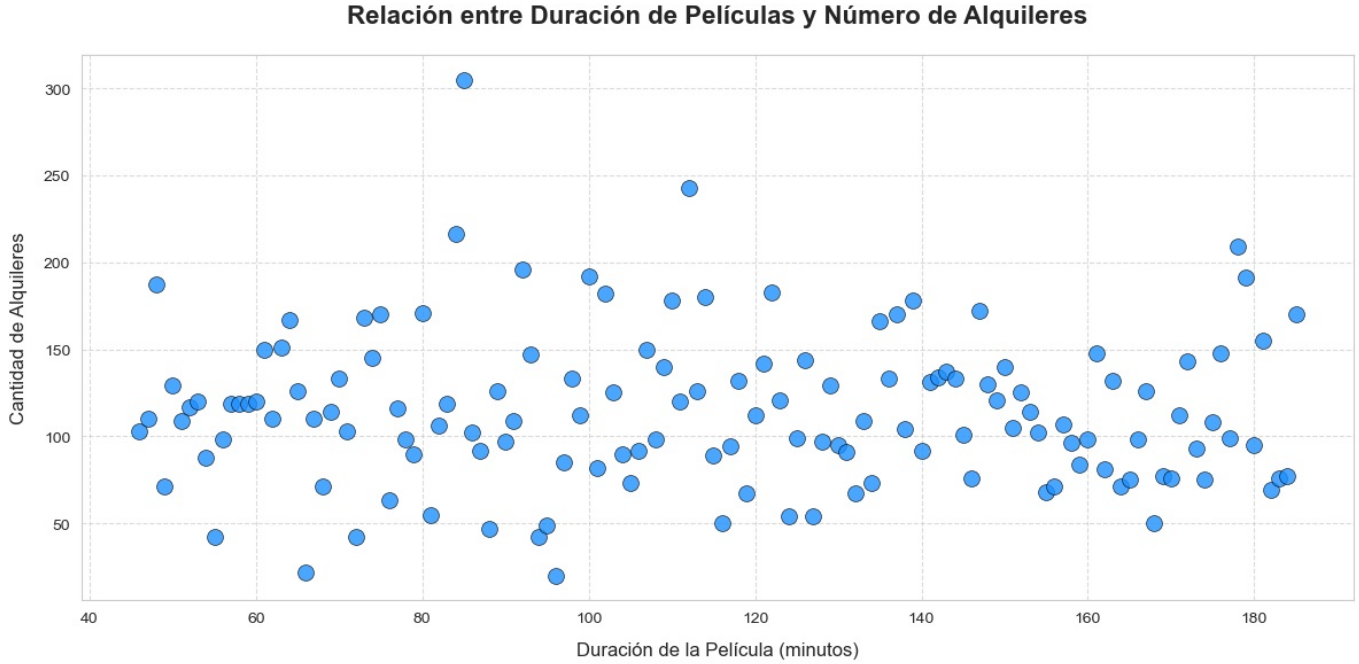
In [21]:
```
# Configuración del estilo visual para las gráficas
sns.set_style("whitegrid")
sns.set_palette("pastel")

# Creación de una figura con dos subplots para comparar distribuciones
plt.figure(figsize=(14, 6))

# Primer subplot: Distribución de las tarifas de renta
plt.subplot(1, 2, 1)
sns.histplot(film["rental_rate"], bins=10, kde=True, color="skyblue", edgecolor="black", linewidth=1.2)
plt.title("Distribución de Tarifas de Renta", fontsize=14, fontweight="bold", pad=15)
plt.xlabel("Tarifa de Renta ($)", fontsize=12, labelpad=10)
plt.ylabel("Frecuencia", fontsize=12, labelpad=10)
plt.grid(True, linestyle="--", alpha=0.7)

# Segundo subplot: Distribución de la duración de las películas
plt.subplot(1, 2, 2)
sns.histplot(film["length"], bins=15, kde=True, color="salmon", edgecolor="black", linewidth=1.2)
plt.title("Distribución de Duración de Películas", fontsize=14, fontweight="bold", pad=15)
plt.xlabel("Duración (minutos)", fontsize=12, labelpad=10)
plt.ylabel("Frecuencia", fontsize=12, labelpad=10)
plt.grid(True, linestyle="--", alpha=0.7)

# Ajuste del layout para evitar solapamientos
plt.tight_layout()

# Mostrar la gráfica
plt.show()
```





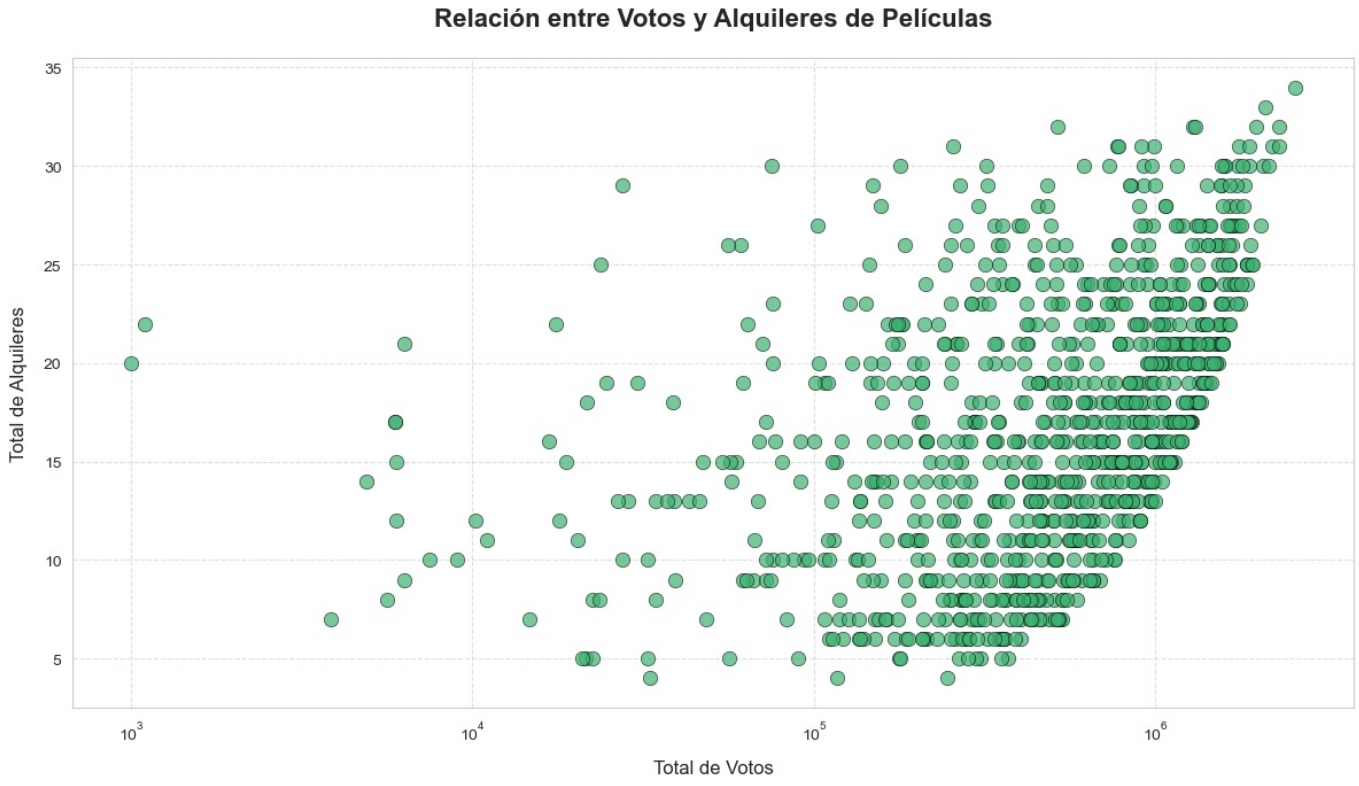
Relación entre duración de la película y cantidad de alquileres

In [22]:
```python
# Unión de las tablas film, rental e inventory para obtener datos completos
df_film_rental = rental.merge(inventory, on="inventory_id").merge(film, on="film_id")

# Agrupación de los datos por duración de la película y conteo de alquileres
rentals_by_length = df_film_rental.groupby("length")["rental_id"].count().reset_index()

# Configuración del estilo visual para la gráfica
sns.set_style("whitegrid")  # Fondo con cuadrícula
sns.set_palette("viridis")  # Paleta de colores

# Creación de una gráfica de dispersión
plt.figure(figsize=(12, 6))
sns.scatterplot(data=rentals_by_length, x="length", y="rental_id",
                color="dodgerblue", s=100, edgecolor="black", linewidth=0.5, alpha=0.8)

# Personalización del título y etiquetas
plt.title("Relación entre Duración de Películas y Número de Alquileres",
          fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Duración de la Película (minutos)", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Alquileres", fontsize=12, labelpad=10)

# Añadir cuadrícula suave
plt.grid(True, linestyle="--", alpha=0.7)

# Ajustar el layout para mejor visualización
plt.tight_layout()


plt.show()
```



Relación entre votos y cantidad de alquileres

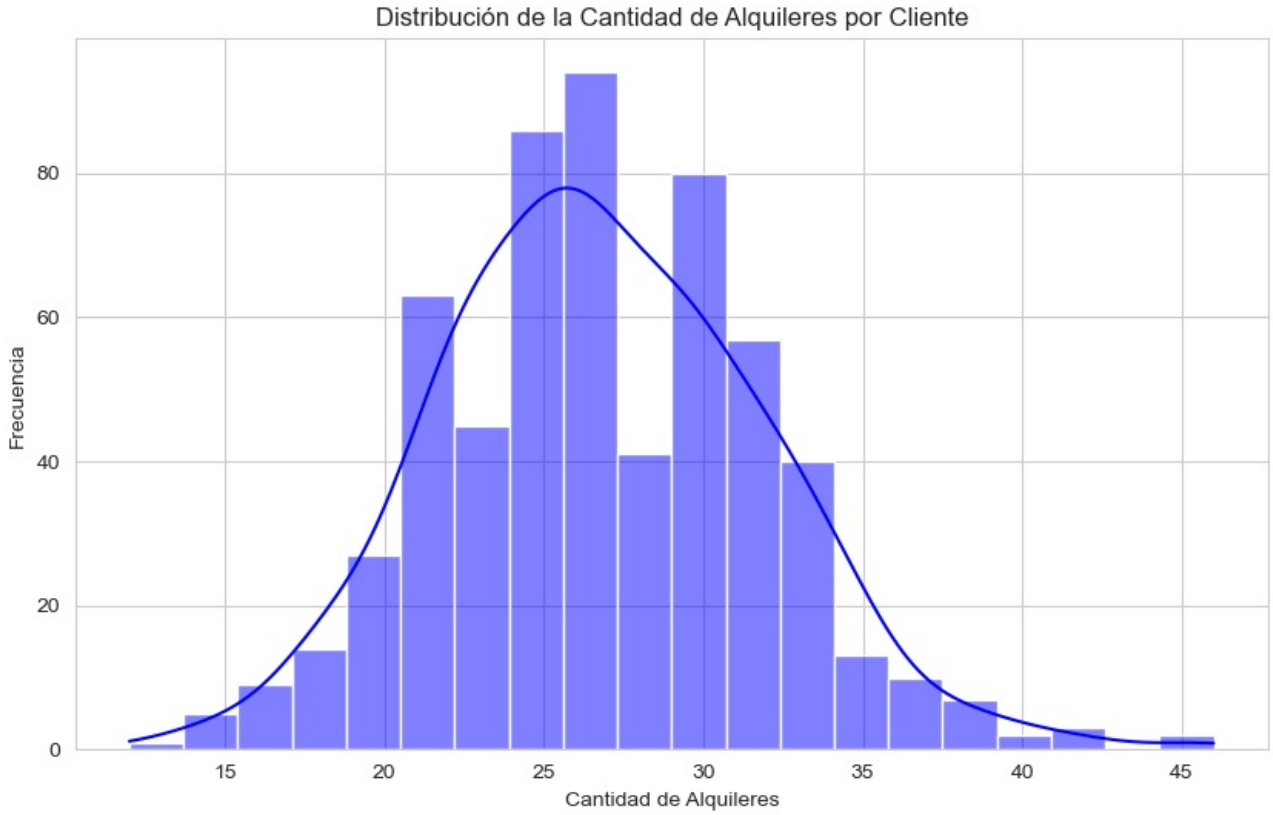
In [23]:
```
# Unión de las tablas rental, inventory y film, y agrupación por título para calcular votos y alquileres
df_votos_rentas = (rental.merge(inventory, on="inventory_id")
                         .merge(film, on="film_id")
                         .groupby("title", observed=False)
                         .agg(total_rentals=("rental_id", "count"),
                              total_votes=("num_voted_users", "sum"))
                         .reset_index()
                         .sort_values(by="total_rentals", ascending=False))

# Configuración del estilo visual para la gráfica
sns.set_style("whitegrid")  # Fondo con cuadrícula
sns.set_palette("rocket")  # Paleta de colores

# Creación de una gráfica de dispersión
plt.figure(figsize=(12, 7))
sns.scatterplot(x="total_votes", y="total_rentals", data=df_votos_rentas,
                color="mediumseagreen", s=80, alpha=0.7, edgecolor="black", linewidth=0.5)

# título y etiquetas
plt.title("Relación entre Votos y Alquileres de Películas",
          fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Total de Votos", fontsize=12, labelpad=10)
plt.ylabel("Total de Alquileres", fontsize=12, labelpad=10)

# Escala logarítmica en el eje X
plt.xscale("log")

# cuadrícula suave
plt.grid(True, linestyle="--", alpha=0.6)

# Ajustar el layout
plt.tight_layout()


plt.show()
```



Distribución de alquileres entre los clientes

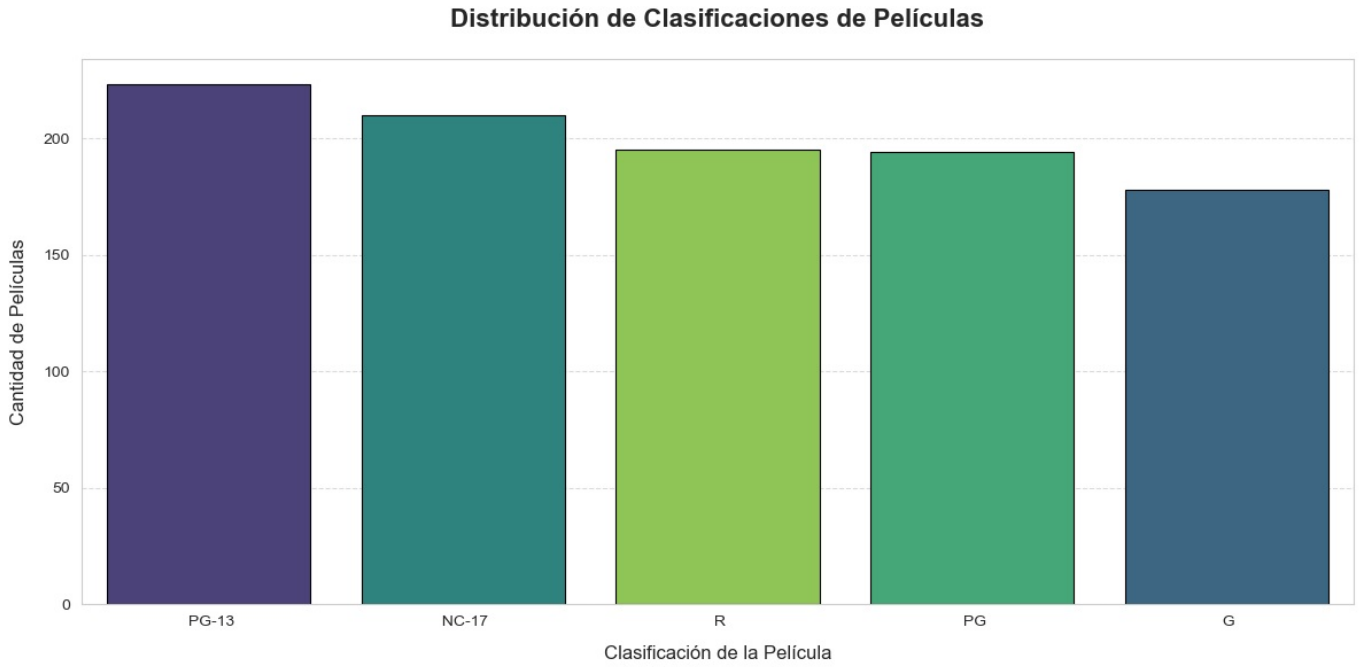
In [24]:
```
# Unir rental → customer
df_clientes_1 = rental.merge(customer, on="customer_id", suffixes=("_rental", "_cust"))

# Unir con inventory
df_clientes_1 = df_clientes_1.merge(inventory, on="inventory_id", suffixes=("", "_inv"))

# Unir con film
df_clientes_1 = df_clientes_1.merge(film, on="film_id", suffixes=("", "_film"))

# Agrupar por cliente para calcular total de rentas y gasto total
df_clientes_summary = (df_clientes_1.groupby(["customer_id", "first_name", "last_name"])
                                    .agg(total_rentals=("rental_id", "count"),
                                         total_spent=("rental_rate", "sum"))
```

```
                                .sort_values(by="total_spent", ascending=False))  # Ordenar por más gasto


# Histograma de total_rentals
plt.figure(figsize=(10, 6))
sns.histplot(df_clientes_summary["total_rentals"], bins=20, kde=True, color="blue")
plt.title("Distribución de la Cantidad de Alquileres por Cliente")
plt.xlabel("Cantidad de Alquileres")
plt.ylabel("Frecuencia")
plt.show()
```



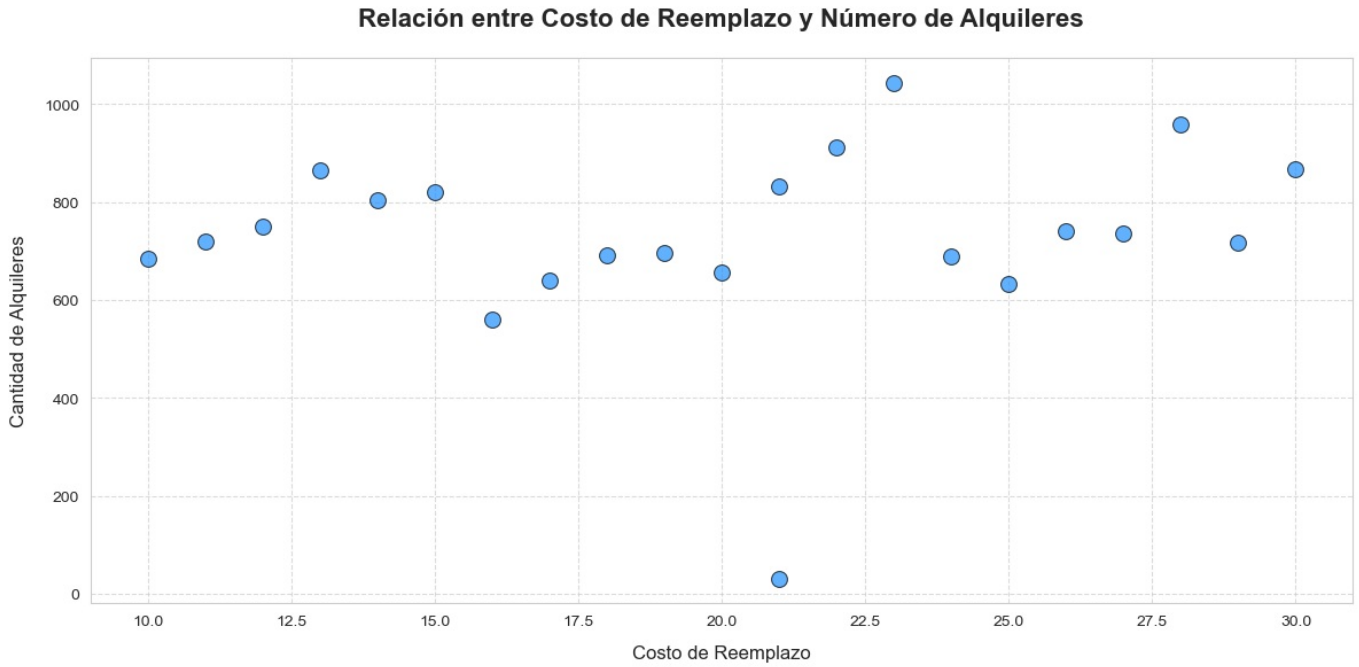
Distribucion de las clasificaciones de las películas

In [25]:
```
# Configuración del estilo visual
sns.set_style("whitegrid")
sns.set_palette("pastel")

# Creación de la gráfica de barras
plt.figure(figsize=(12, 6))
sns.countplot(data=film, x="rating", order=film["rating"].value_counts().index,
              hue="rating", palette="viridis", edgecolor="black", linewidth=0.8, legend=False)

# Personalización de títulos y etiquetas
plt.title("Distribución de Clasificaciones de Películas", fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Clasificación de la Película", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Películas", fontsize=12, labelpad=10)

# Cuadrícula suave en el eje Y
plt.grid(True, linestyle="--", alpha=0.7, axis="y")

# Ajuste del layout
plt.tight_layout()
plt.show()
```



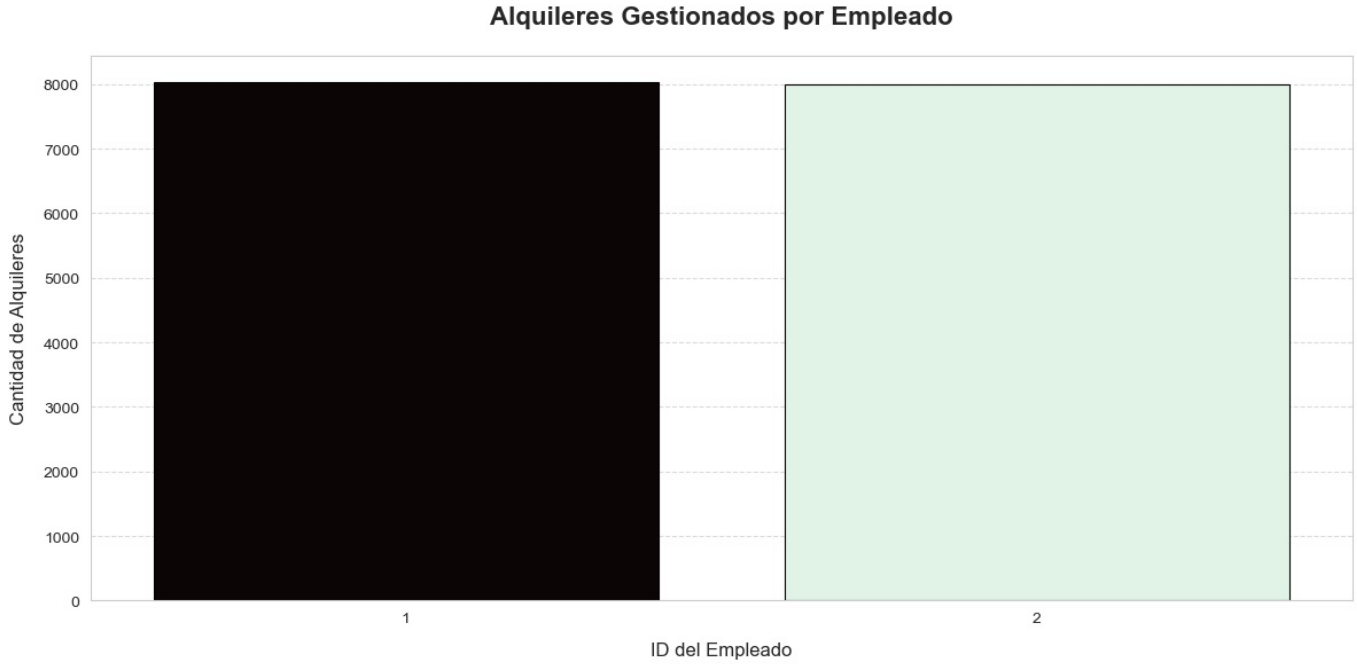
Costo de rempazo vs cantidad de alquileres

In [26]:
```python
# Unión de las tablas rental, inventory y film
df_cost_rentals = rental.merge(inventory, on="inventory_id").merge(film, on="film_id")

# Agrupación por costo de reemplazo y conteo de alquileres
rentals_by_cost = df_cost_rentals.groupby("replacement_cost")["rental_id"].count().reset_index()

# Configuración del estilo visual
sns.set_style("whitegrid")
sns.set_palette("rocket")

# Creación de la gráfica de dispersión mejorada
plt.figure(figsize=(12, 6))
sns.scatterplot(data=rentals_by_cost, x="replacement_cost", y="rental_id",
                color="dodgerblue", s=100, alpha=0.7, edgecolor="black", linewidth=0.8)

# Personalización de títulos y etiquetas
plt.title("Relación entre Costo de Reemplazo y Número de Alquileres",
          fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Costo de Reemplazo", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Alquileres", fontsize=12, labelpad=10)

# Cuadrícula suave
plt.grid(True, linestyle="--", alpha=0.7)

# Ajuste del layout
plt.tight_layout()
plt.show()
```



Alquileres gestionados por empleados

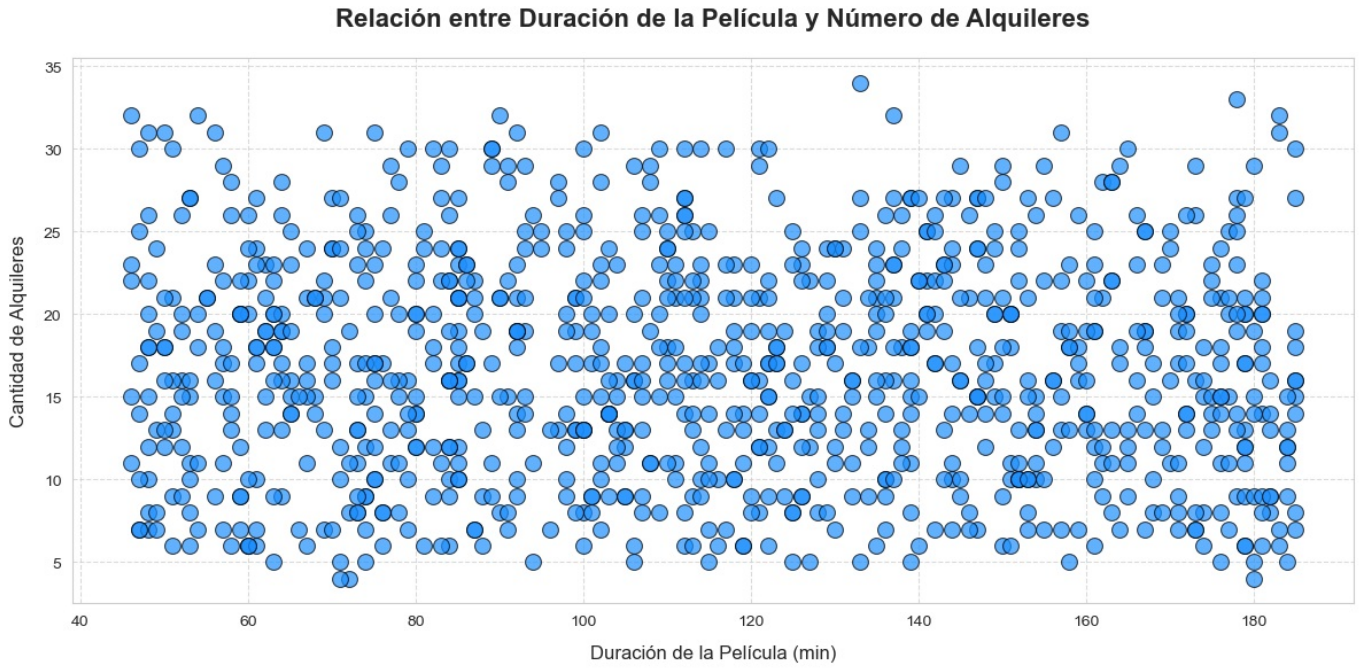
In [27]:
```
# Conteo de alquileres por empleado
rentals_per_staff = rental.groupby("staff_id")["rental_id"].count().reset_index()

# Configuración del estilo visual
sns.set_style("whitegrid")
sns.set_palette("mako")  # Paleta de colores

# Creación de la gráfica de barras
plt.figure(figsize=(12, 6))
sns.barplot(data=rentals_per_staff, x="staff_id", y="rental_id",
            hue="staff_id", palette="mako", edgecolor="black", linewidth=0.8, legend=False)

# Personalización de títulos y etiquetas
plt.title("Alquileres Gestionados por Empleado", fontsize=16, fontweight="bold", pad=20)
plt.xlabel("ID del Empleado", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Alquileres", fontsize=12, labelpad=10)

# Cuadrícula suave en el eje Y
plt.grid(True, linestyle="--", alpha=0.7, axis="y")

# Ajuste del layout
plt.tight_layout()
plt.show()
```



Duración de las peliculas vs Alquileres

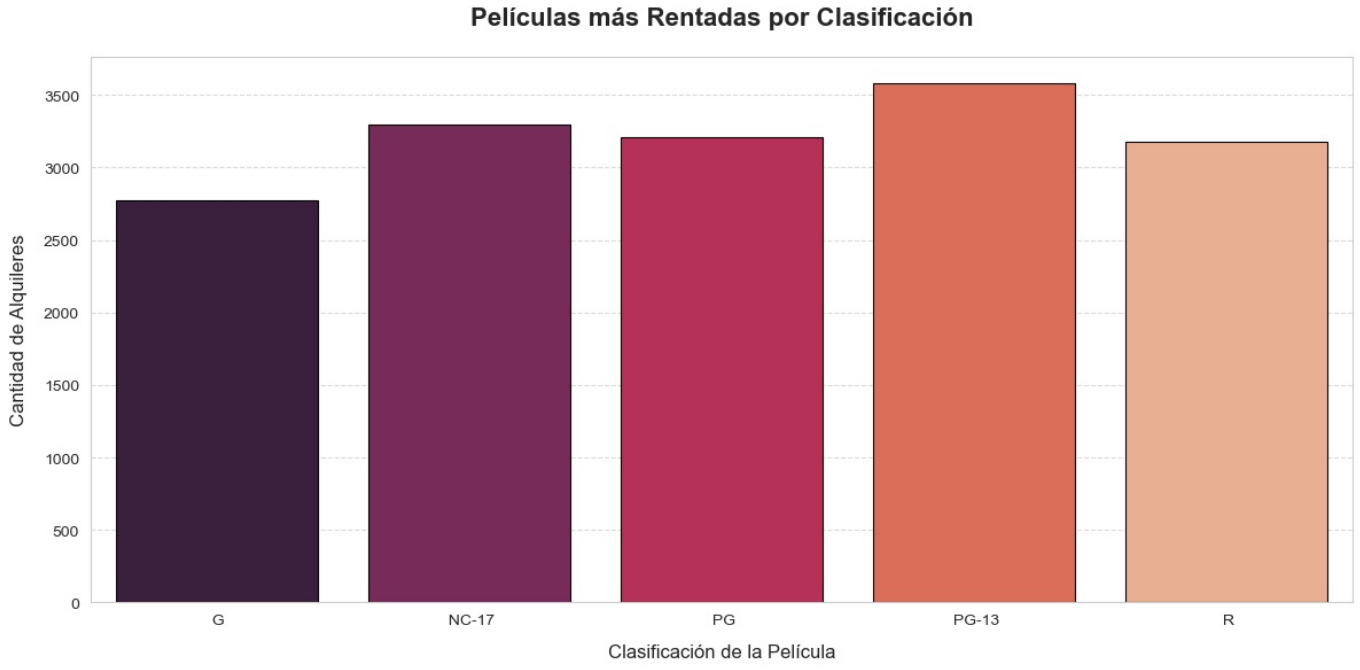
In [28]:
```
# Unión de las tablas rental, inventory y film
df_film_rentals = rental.merge(inventory, on="inventory_id").merge(film, on="film_id")

# Conteo de alquileres por película y agregar duración
top_rented_films = df_film_rentals.groupby("title").agg(
    total_rentals=("rental_id", "count"),
    length=("length", "first")  # Tomar la duración de la película
).reset_index().sort_values(by="total_rentals", ascending=False)

# Configuración del estilo visual
sns.set_style("whitegrid")
sns.set_palette("viridis")

# Creación de la gráfica de dispersión
plt.figure(figsize=(12, 6))
sns.scatterplot(data=top_rented_films, x="length", y="total_rentals",
                color="dodgerblue", s=100, alpha=0.7, edgecolor="black", linewidth=0.8)

# Personalización de títulos y etiquetas
plt.title("Relación entre Duración de la Película y Número de Alquileres",
          fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Duración de la Película (min)", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Alquileres", fontsize=12, labelpad=10)

# Cuadrícula suave
plt.grid(True, linestyle="--", alpha=0.7)

# Ajuste del layout
```

```
plt.tight_layout()
plt.show()
```



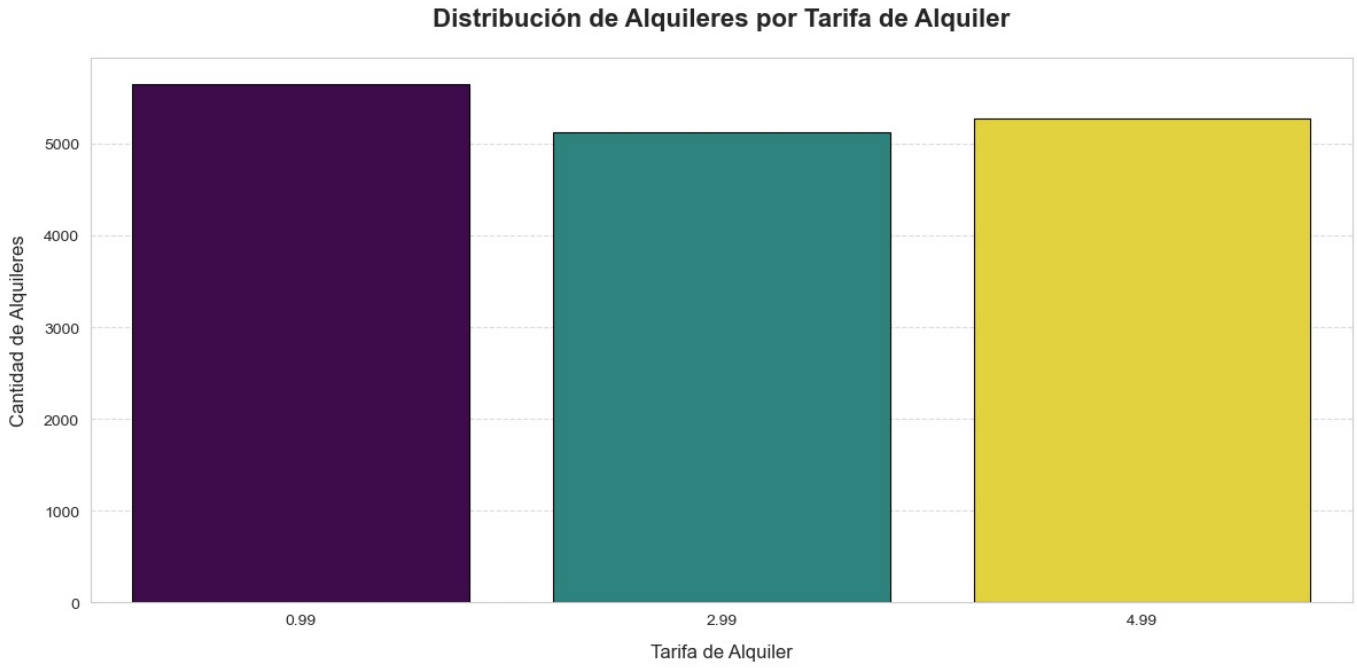
Peliculas alquiladas segun clasificacion

In [29]:
```
# Agrupación por clasificación y conteo de alquileres
rentals_by_rating = df_film_rentals.groupby("rating")["rental_id"].count().reset_index()

# Configuración del estilo visual
sns.set_style("whitegrid")
sns.set_palette("rocket")

# Creación de la gráfica de barras
plt.figure(figsize=(12, 6))
sns.barplot(data=rentals_by_rating, x="rating", y="rental_id",
            hue="rating", palette="rocket", edgecolor="black", linewidth=0.8, legend=False)

# Personalización de títulos y etiquetas
plt.title("Películas más Rentadas por Clasificación", fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Clasificación de la Película", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Alquileres", fontsize=12, labelpad=10)

# Cuadrícula suave en el eje Y
plt.grid(True, linestyle="--", alpha=0.7, axis="y")

# Ajuste del layout
plt.tight_layout()
plt.show()
```



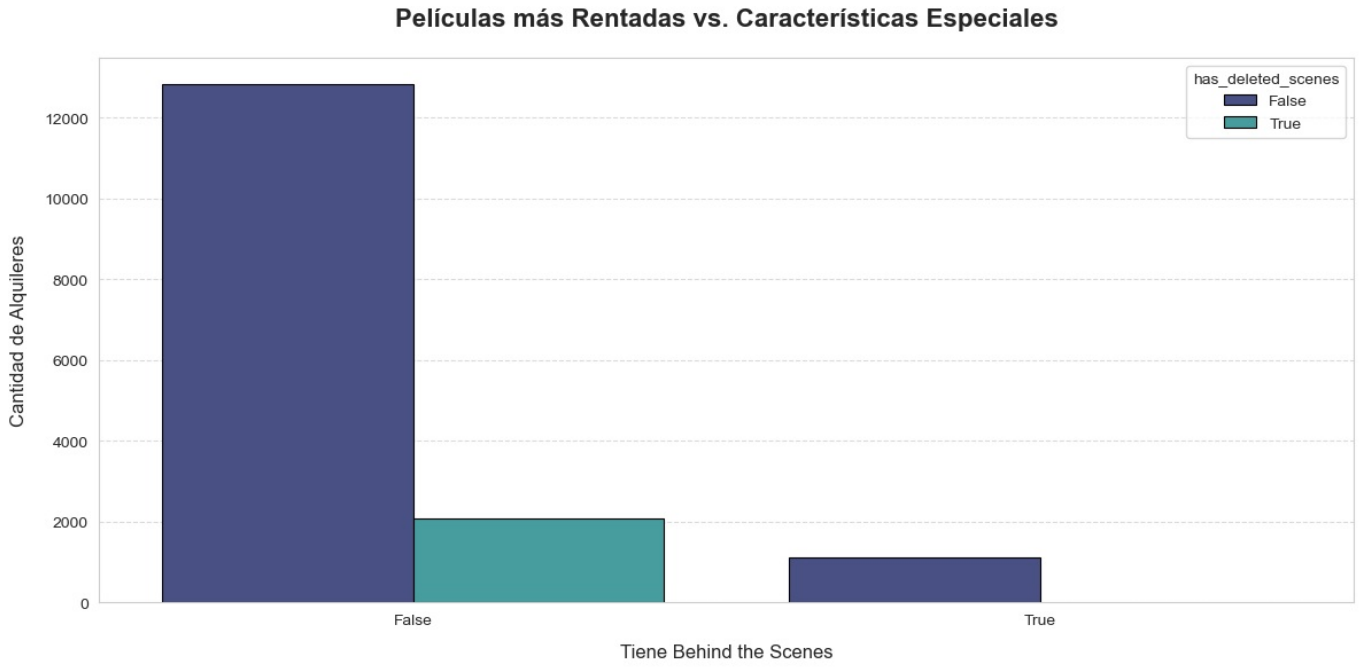
Relación entre la tarifa de alquiler y el numero de peliculas alquiladas

In [30]:
```
# Agrupación por tarifa de alquiler y conteo de alquileres
rentals_by_rate = df_film_rentals.groupby("rental_rate")["rental_id"].count().reset_index()

# Configuración del estilo visual
sns.set_style("whitegrid")
sns.set_palette("viridis")

# Creación de la gráfica de barras
plt.figure(figsize=(12, 6))
sns.barplot(data=rentals_by_rate, x="rental_rate", y="rental_id",
            hue="rental_rate", palette="viridis", edgecolor="black", linewidth=0.8, legend=False)

# Personalización de títulos y etiquetas
plt.title("Distribución de Alquileres por Tarifa de Alquiler", fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Tarifa de Alquiler", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Alquileres", fontsize=12, labelpad=10)

# Cuadrícula suave en el eje Y
plt.grid(True, linestyle="--", alpha=0.7, axis="y")

# Ajuste del layout
plt.tight_layout()
plt.show()
```



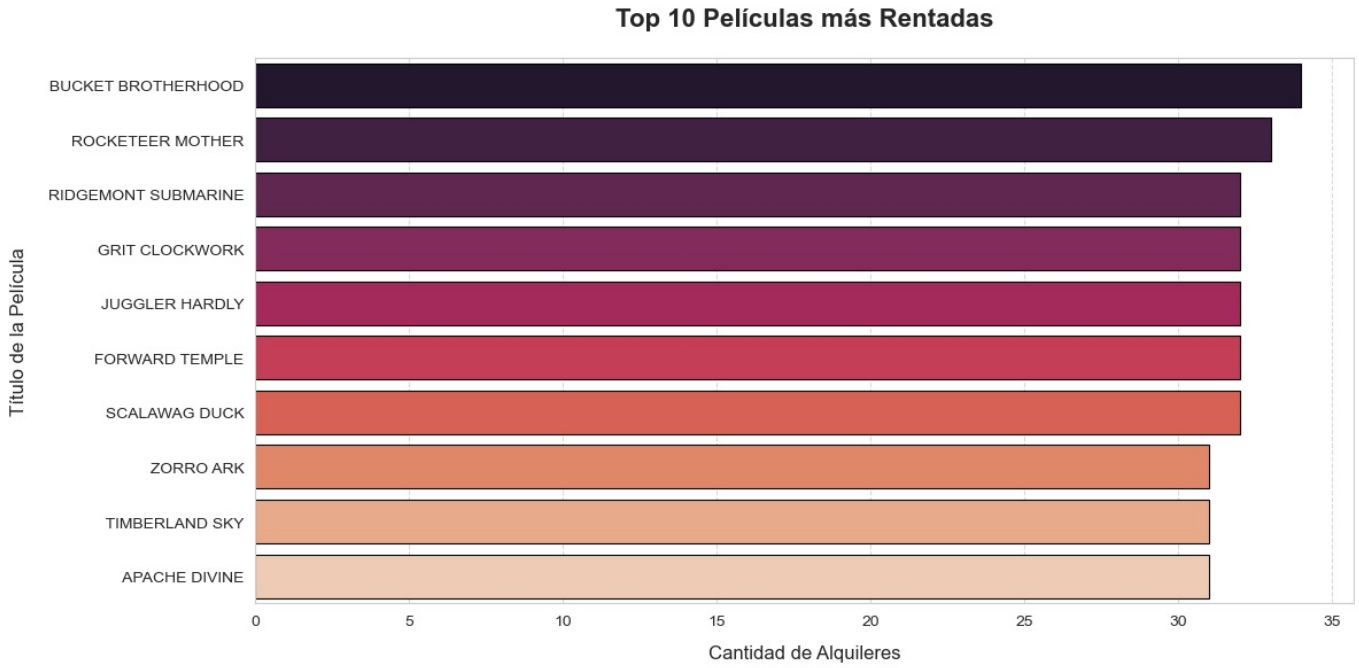
Relacion entre peliculas rentadas y caracteristicas especiales

In [31]:
```
# Crear columnas para características especiales
df_film_rentals["special_features"] = df_film_rentals["special_features"].astype(str)
df_film_rentals["has_behind_the_scenes"] = df_film_rentals["special_features"].str.contains("Behind the Scenes"
df_film_rentals["has_deleted_scenes"] = df_film_rentals["special_features"].str.contains("Deleted Scenes")

# Agrupación por características especiales y conteo de alquileres
rentals_by_features = df_film_rentals.groupby(["has_behind_the_scenes", "has_deleted_scenes"])["rental_id"].cou

# Configuración del estilo visual
sns.set_style("whitegrid")
sns.set_palette("mako")

# Creación de la gráfica de barras
plt.figure(figsize=(12, 6))
sns.barplot(data=rentals_by_features, x="has_behind_the_scenes", y="rental_id",
            hue="has_deleted_scenes", palette="mako", edgecolor="black", linewidth=0.8)

# Personalización de títulos y etiquetas
plt.title("Películas más Rentadas vs. Características Especiales", fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Tiene Behind the Scenes", fontsize=12, labelpad=10)
plt.ylabel("Cantidad de Alquileres", fontsize=12, labelpad=10)

# Cuadrícula suave en el eje Y
plt.grid(True, linestyle="--", alpha=0.7, axis="y")

# Ajuste del layout
plt.tight_layout()
plt.show()
```



Películas mas rentadas

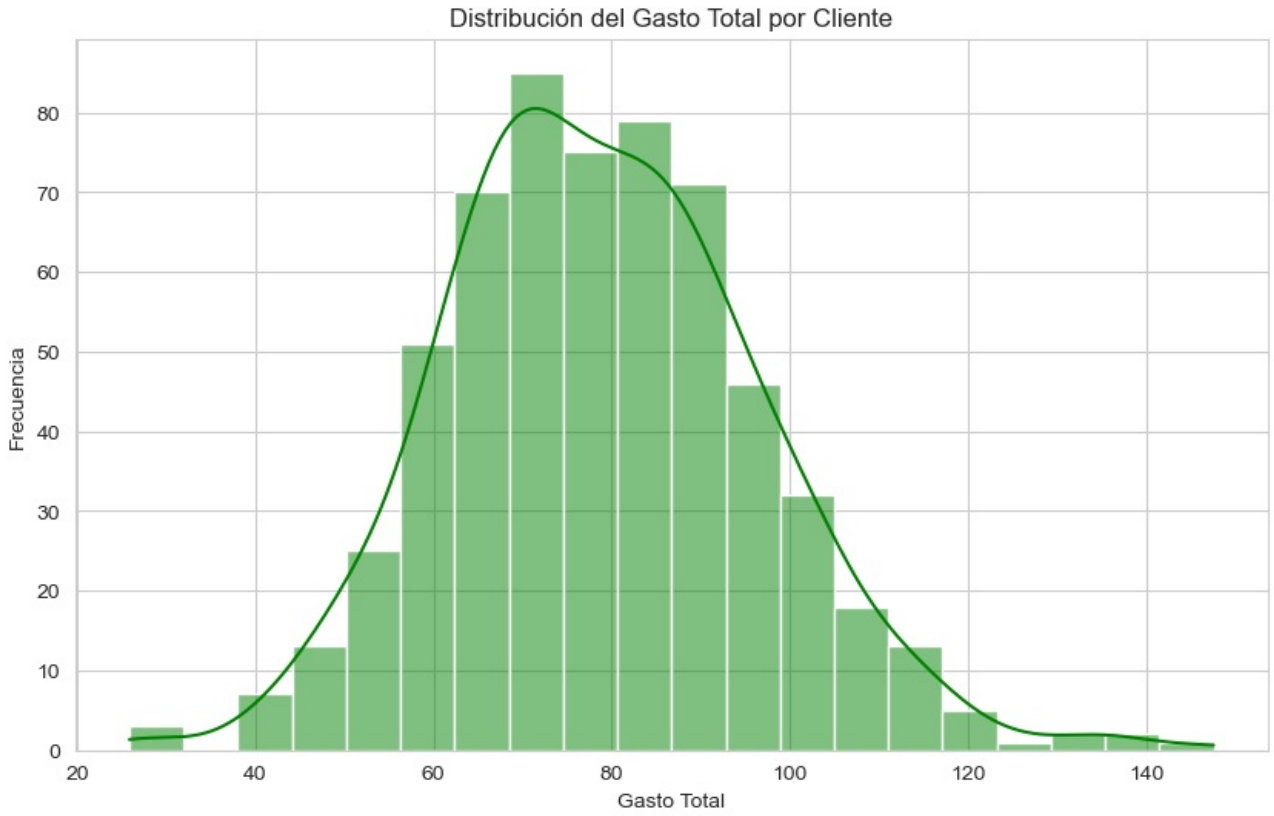
In [32]:
```
# Obtener las 10 películas más rentadas
top_10_rented_films = df_film_rentals.groupby("title")["rental_id"].count().reset_index().sort_values(by="renta

# Configuración del estilo visual
sns.set_style("whitegrid")
sns.set_palette("rocket")

# Creación de la gráfica de barras
plt.figure(figsize=(12, 6))
sns.barplot(data=top_10_rented_films, x="rental_id", y="title",
            hue="title", palette="rocket", edgecolor="black", linewidth=0.8, legend=False)

# Personalización de títulos y etiquetas
plt.title("Top 10 Películas más Rentadas", fontsize=16, fontweight="bold", pad=20)
plt.xlabel("Cantidad de Alquileres", fontsize=12, labelpad=10)
plt.ylabel("Título de la Película", fontsize=12, labelpad=10)

# Cuadrícula suave en el eje X
plt.grid(True, linestyle="--", alpha=0.7, axis="x")

# Ajuste del layout
plt.tight_layout()
plt.show()
```



distribución del gasto total

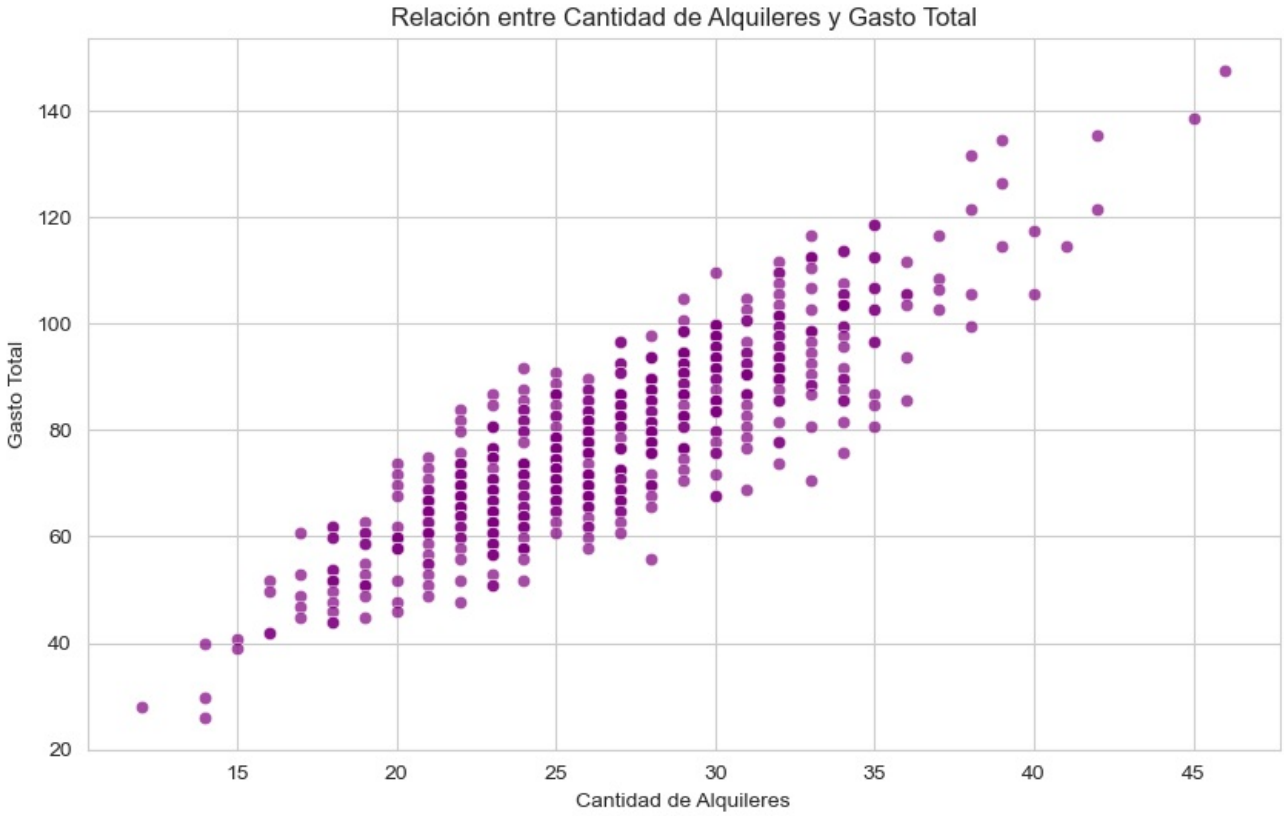
In [33]:
```
# Histograma de total_spent
```

```
plt.figure(figsize=(10, 6))
sns.histplot(df_clientes_summary["total_spent"], bins=20, kde=True, color="green")
plt.title("Distribución del Gasto Total por Cliente")
plt.xlabel("Gasto Total")
plt.ylabel("Frecuencia")
plt.show()
```



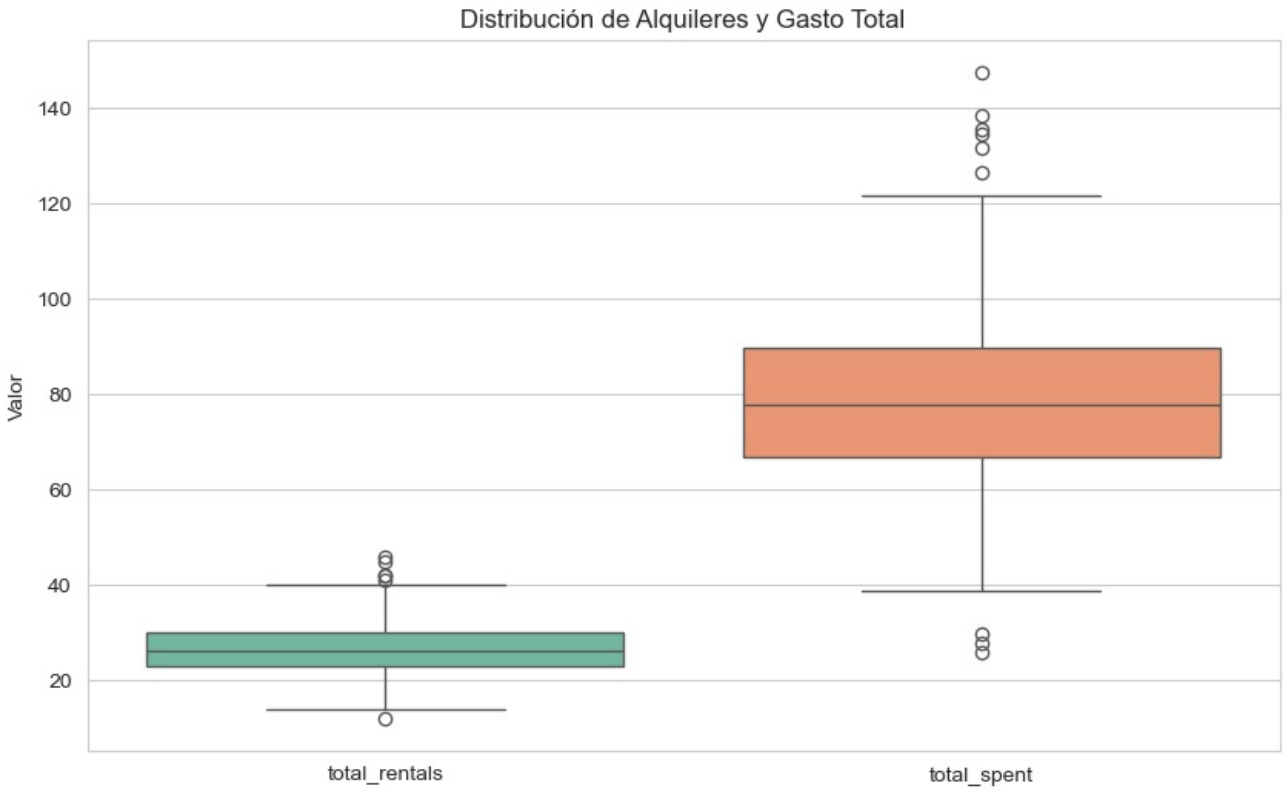
Relacion entre cantidad de alquileres y gasto total entre los clientes

In [34]:
```
# Gráfico de dispersión entre total_rentals y total_spent
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_clientes_summary, x="total_rentals", y="total_spent", color="purple", alpha=0.7)
plt.title("Relación entre Cantidad de Alquileres y Gasto Total")
plt.xlabel("Cantidad de Alquileres")
plt.ylabel("Gasto Total")
plt.show()
```



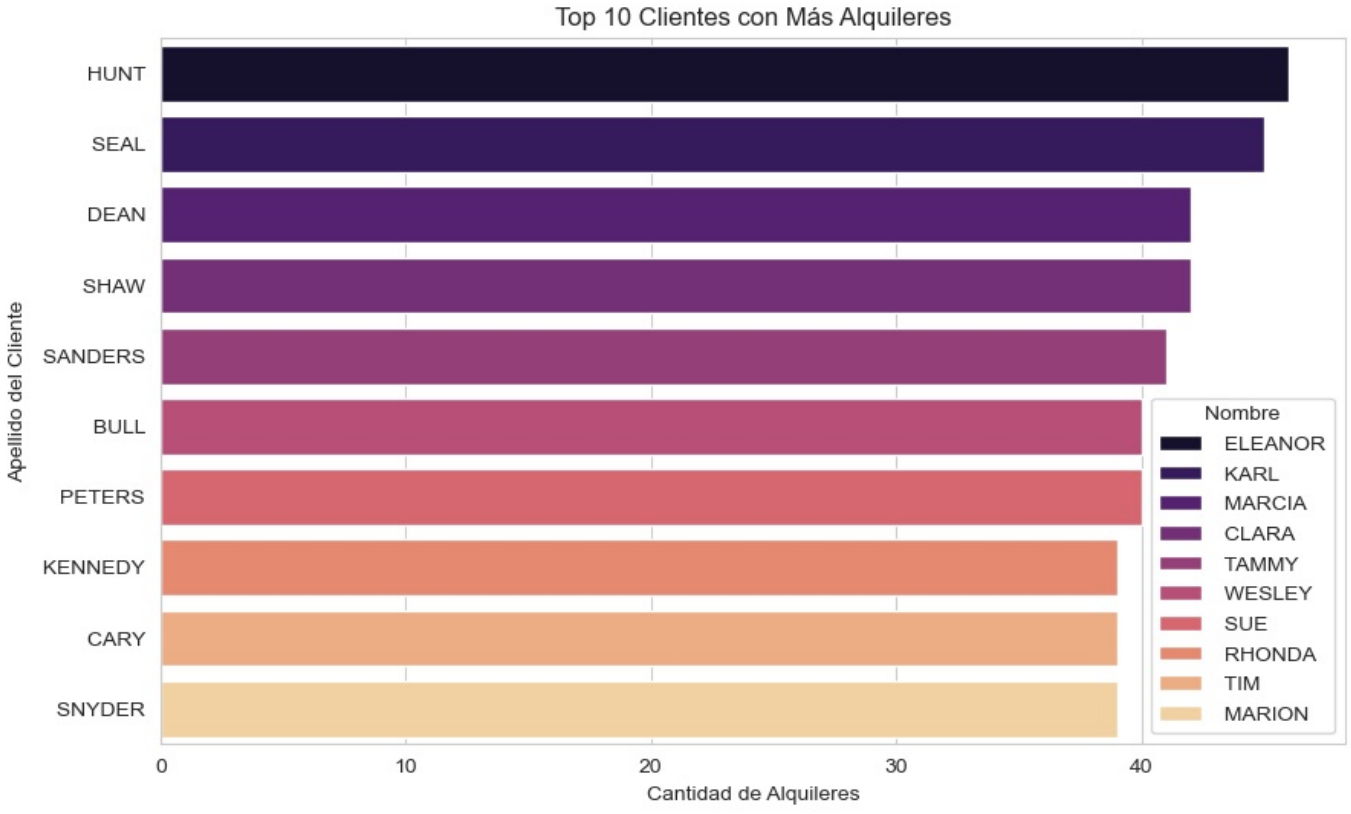
Distribucion cantidad de alquileres y gasto total entre los clientes

In [35]:
```
# Boxplot de total_rentals y total_spent
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_clientes_summary[["total_rentals", "total_spent"]], palette="Set2")
plt.title("Distribución de Alquileres y Gasto Total")
plt.ylabel("Valor")
plt.show()
```



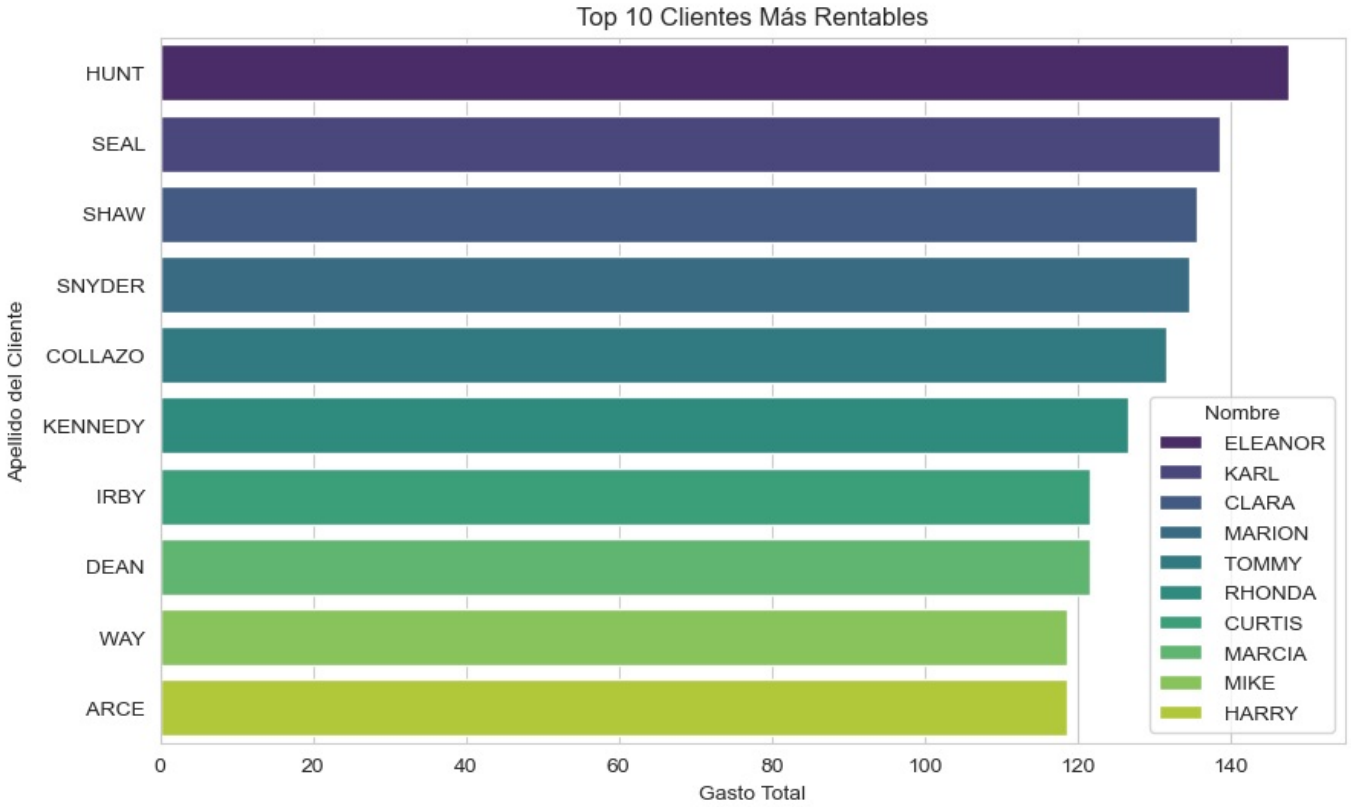
Clientes con más alquileres

In [36]:
```
# Gráfico de barras del top 10 de clientes con más alquileres
top_clientes_alquileres = df_clientes_summary.sort_values(by="total_rentals", ascending=False).head(10)
plt.figure(figsize=(10, 6))
sns.barplot(data=top_clientes_alquileres, x="total_rentals", y="last_name", hue="first_name", dodge=False, pale
plt.title("Top 10 Clientes con Más Alquileres")
plt.xlabel("Cantidad de Alquileres")
plt.ylabel("Apellido del Cliente")
plt.legend(title="Nombre")
plt.show()
```



Clientes que más han gastado

In [37]:
```
# Gráfico de barras del top 10 de clientes más rentables
top_clientes_rentables = df_clientes_summary.head(10)
plt.figure(figsize=(10, 6))
sns.barplot(data=top_clientes_rentables, x="total_spent", y="last_name", hue="first_name", dodge=False, palette
plt.title("Top 10 Clientes Más Rentables")
plt.xlabel("Gasto Total")
plt.ylabel("Apellido del Cliente")
plt.legend(title="Nombre")
plt.show()
```

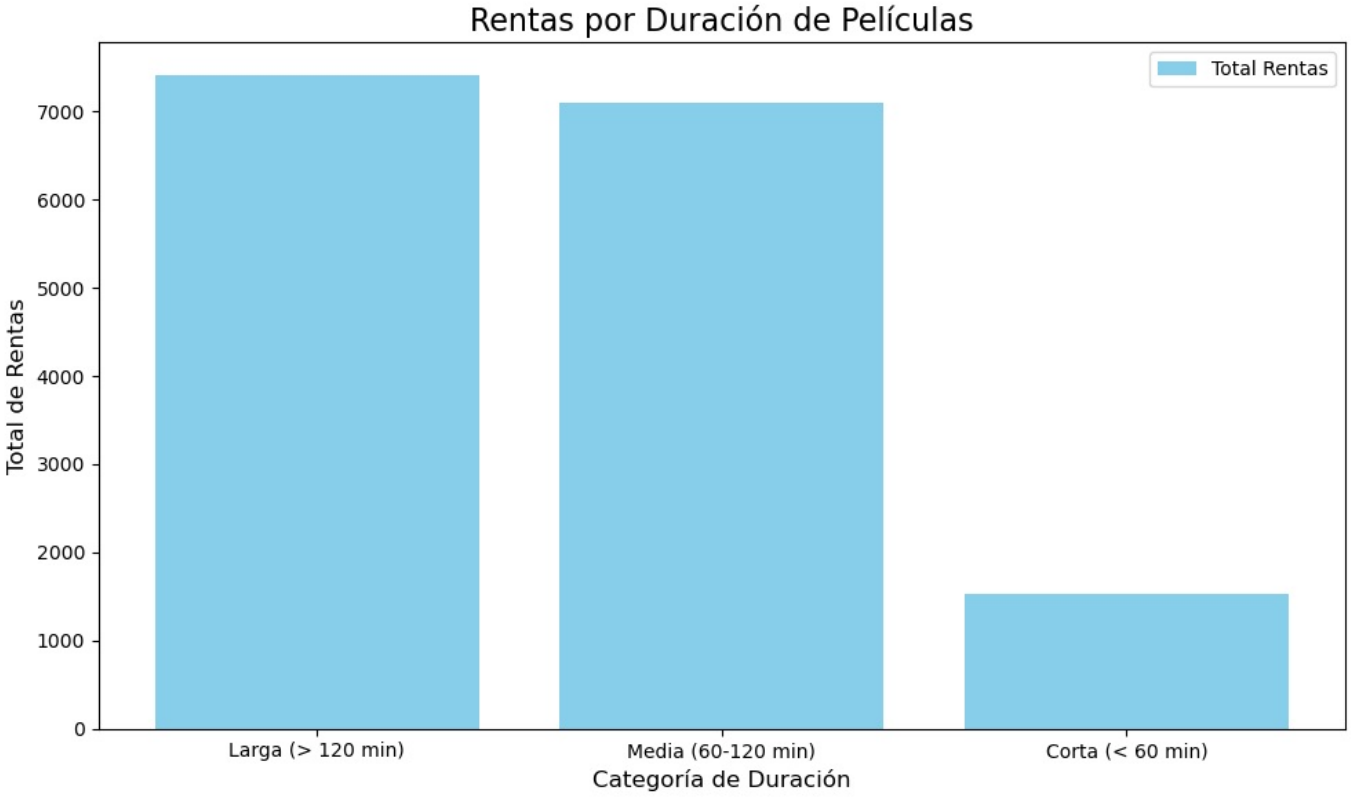


In [ ]:
```
```

In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Supongamos que ya tienes cargados los DataFrames: film, inventory, rental

# Definir categorías de duración de películas
bins = [0, 60, 120, np.inf]  # Definimos los intervalos
labels = ["Corta (< 60 min)", "Media (60-120 min)", "Larga (> 120 min)"]
film_length = film.copy()
film_length["length_category"] = pd.cut(film_length["length"], bins=bins, labels=labels, right=False)

# Unir rental → inventory → film
rental_inventory_film = rental.merge(inventory, on="inventory_id").merge(film_length, on="film_id")

# Agrupar por categoría de duración para calcular número de rentas e ingresos
length_summary = (rental_inventory_film.groupby("length_category", observed=False)
                  .agg(total_rentals=("rental_id", "count"),
                       total_income=("rental_rate", "sum"))
                  .sort_values(by="total_rentals", ascending=False))  # Ordenar por más rentas

# Mostrar resultados
print(length_summary)

# Crear el histograma (solo rentas)
fig, ax = plt.subplots(figsize=(10, 6))

# Configurar el gráfico de barras para las rentas
ax.bar(length_summary.index, length_summary["total_rentals"], color='skyblue', label='Total Rentas')

# Añadir títulos y etiquetas
ax.set_title("Rentas por Duración de Películas", fontsize=16)
ax.set_xlabel("Categoría de Duración", fontsize=12)
ax.set_ylabel("Total de Rentas", fontsize=12)

# Añadir leyenda
```

```
ax.legend(loc='upper right')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

```
                   total_rentals  total_income
length_category
Larga (> 120 min)           7412      22409.88
Media (60-120 min)          7101      20577.99
Corta (< 60 min)            1531       4223.69
```

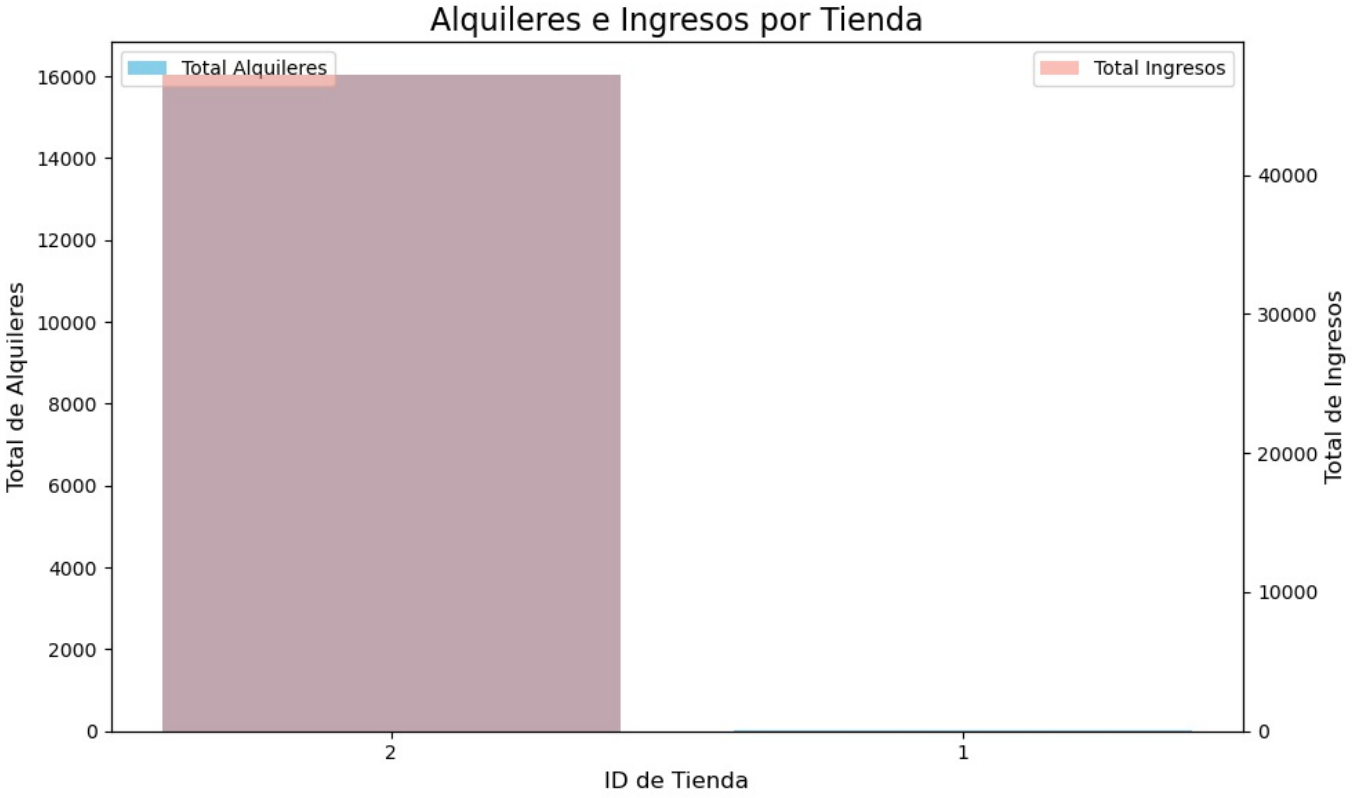


In [ ]:

In [18]:
```
# Asegurar que 'store_id' tenga el mismo tipo en ambas tablas
inventory["store_id"] = inventory["store_id"].astype(int)
store["store_id"] = store["store_id"].astype(int)

# Unir rental → inventory → store → film con sufijos para evitar duplicados
df_tiendas = (rental.merge(inventory, on="inventory_id", suffixes=("_rental", "_inventory"))
                    .merge(store, on="store_id", suffixes=("_inventory", "_store"))
                    .merge(film, on="film_id", suffixes=("_store", "_film")))

# Eliminar columnas 'last_update' redundantes si existen
df_tiendas = df_tiendas.loc[:, ~df_tiendas.columns.duplicated()]

# Agrupar por tienda y calcular total de alquileres e ingresos
df_tiendas_summary = (df_tiendas.groupby("store_id", observed=False)
                                .agg(total_rentals=("rental_id", "count"),
                                     total_income=("rental_rate", "sum"))
                                .sort_values(by="total_income", ascending=False))  # Ordenar por ingresos

# Mostrar resultados
print(df_tiendas_summary)

# Crear el gráfico de barras
fig, ax = plt.subplots(figsize=(10, 6))

# Configurar el gráfico de barras para los alquileres
ax.bar(df_tiendas_summary.index.astype(str), df_tiendas_summary["total_rentals"], color='skyblue', label='Total

# Configurar el gráfico de barras para los ingresos (en un segundo eje y)
ax2 = ax.twinx()
ax2.bar(df_tiendas_summary.index.astype(str), df_tiendas_summary["total_income"], color='salmon', alpha=0.5, lab

# Añadir títulos y etiquetas
ax.set_title("Alquileres e Ingresos por Tienda", fontsize=16)
ax.set_xlabel("ID de Tienda", fontsize=12)
ax.set_ylabel("Total de Alquileres", fontsize=12)
ax2.set_ylabel("Total de Ingresos", fontsize=12)

# # Añadir leyendas
```

```
ax.legend(loc='upper left')
ax2.legend(loc='upper right')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

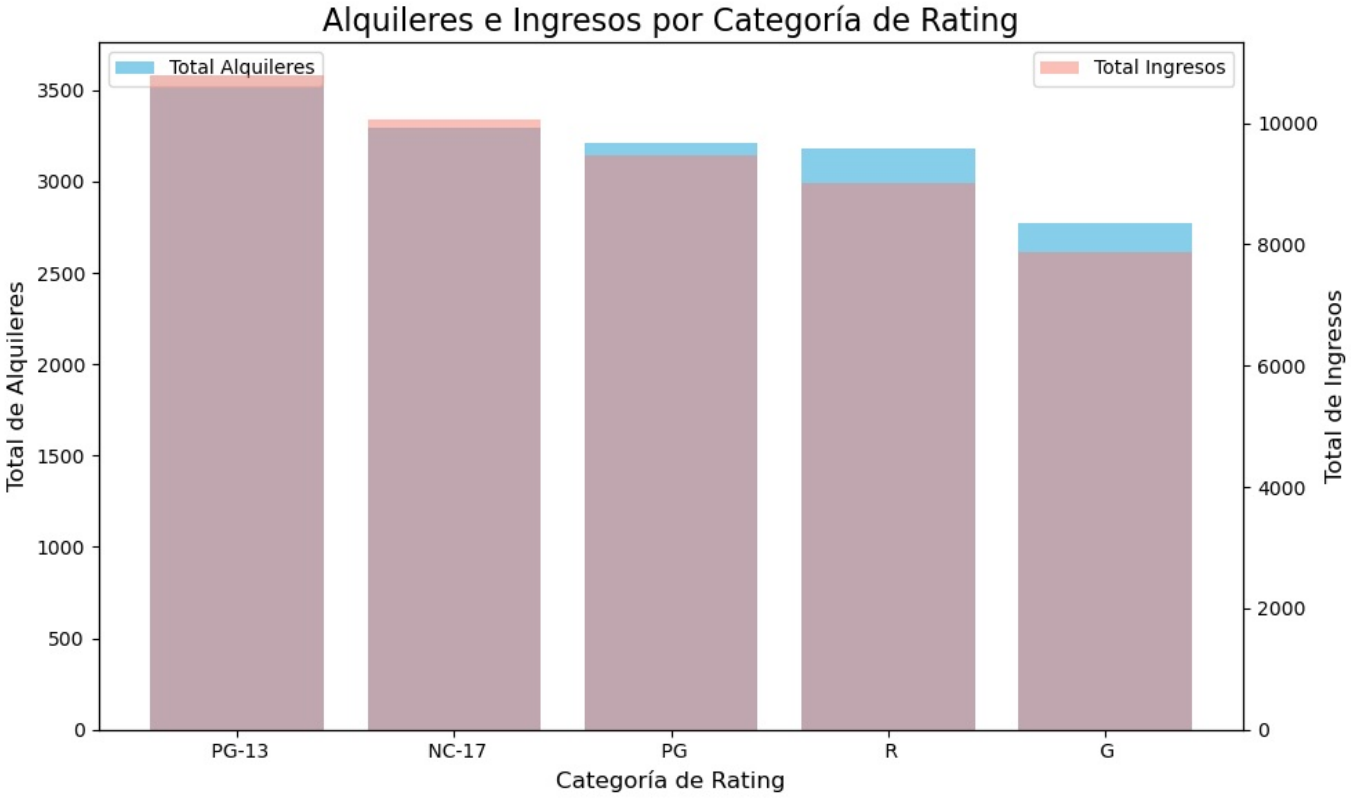
```
          total_rentals  total_income
store_id
2                 16032      47199.68
1                    12         11.88
```



In [ ]:

In [20]:
```
# Unir rental → inventory → film
df_reporte = (rental.merge(inventory, on="inventory_id")
                    .merge(film, on="film_id"))

# Agrupar por categoría (rating) y calcular total de alquileres e ingresos
df_reporte_summary = (df_reporte.groupby("rating", observed=False)
                                .agg(total_rentals=("rental_id", "count"),
                                     total_income=("rental_rate", "sum"))
                                .sort_values(by="total_rentals", ascending=False))  # Ordenar por rentas

# Mostrar resultados
print(df_reporte_summary)

# Crear el gráfico de barras
fig, ax = plt.subplots(figsize=(10, 6))

# Configurar el gráfico de barras para los alquileres
ax.bar(df_reporte_summary.index, df_reporte_summary["total_rentals"], color='skyblue', label='Total Alquileres'

# Configurar el gráfico de barras para los ingresos (en un segundo eje y)
ax2 = ax.twinx()
ax2.bar(df_reporte_summary.index, df_reporte_summary["total_income"], color='salmon', alpha=0.5, label='Total In

# Añadir títulos y etiquetas
ax.set_title("Alquileres e Ingresos por Categoría de Rating", fontsize=16)
ax.set_xlabel("Categoría de Rating", fontsize=12)
ax.set_ylabel("Total de Alquileres", fontsize=12)
ax2.set_ylabel("Total de Ingresos", fontsize=12)

# Añadir leyendas
ax.legend(loc='upper left')
ax2.legend(loc='upper right')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

```
        total_rentals  total_income
rating
PG-13            3585      10797.15
NC-17            3293      10062.07
PG               3212       9465.88
R                3181       9011.19
G                2773       7875.27
```



In [ ]:

Formulacion de preguntas y respuestas

1) ¿Cuáles son las películas más rentadas y cuáles generan mayores ingresos?

Tablas necesarias

rental (alquileres) → Para contar cuántas veces se alquiló cada película (rental_id).
inventory (inventario) → Para relacionar los alquileres con las películas (inventory_id → film_id).
film (películas) → Para obtener el título de la película (title), la cantidad de votos (num_voted_users) y la tarifa de alquiler (rental_rate).

In [38]:
```python
# Unir tablas rental → inventory → film
df_1 = (rental.merge(inventory, on="inventory_id")
              .merge(film, on="film_id"))

# Agrupar por título para calcular total de rentas e ingresos
df_summary_1 = (df_1.groupby("title")
                .agg(total_rentals=("rental_id", "count"),
                     total_income=("rental_rate", "sum"))
                .sort_values(by="total_rentals", ascending=False))  # Ordenar por más rentadas

# Mostrar el top 10 de películas más rentadas
print("Top 10 de películas más rentadas:")
print(df_summary_1.head(10))

# Ordenar por ingresos totales (total_income) en orden descendente
df_summary_2 = df_summary_1.sort_values(by="total_income", ascending=False)

# Mostrar el top 10 de películas con mayores ingresos
print("\nTop 10 de películas con mayores ingresos:")
print(df_summary_2.head(10))
```

```
Top 10 de películas más rentadas:
                    total_rentals  total_income
title
BUCKET BROTHERHOOD              34        169.66
ROCKETEER MOTHER                33         32.67
RIDGEMONT SUBMARINE             32         31.68
GRIT CLOCKWORK                  32         31.68
JUGGLER HARDLY                  32         31.68
FORWARD TEMPLE                  32         95.68
SCALAWAG DUCK                   32        159.68
ZORRO ARK                       31        154.69
TIMBERLAND SKY                  31         30.69
APACHE DIVINE                   31        154.69

Top 10 de películas con mayores ingresos:
                   total_rentals  total_income
title
BUCKET BROTHERHOOD             34        169.66
SCALAWAG DUCK                  32        159.68
GOODFELLAS SALUTE              31        154.69
APACHE DIVINE                  31        154.69
ZORRO ARK                      31        154.69
WIFE TURN                      31        154.69
DOGMA FAMILY                   30        149.70
HARRY IDAHO                    30        149.70
CAT CONEHEADS                  30        149.70
WITCHES PANIC                  30        149.70
```

Respuesta: película BUCKET BROTHERHOOD es la más rentada y la que más ingresos ha generado

In [ ]:

2) ¿Cuáles son los clientes más rentables?

Tablas necesarias

rental (alquileres) → Para contar cuántas veces cada cliente alquiló una película (rental_id).
customer (clientes) → Para obtener la información del cliente (customer_id, first_name, last_name).
inventory (inventario) → Para relacionar los alquileres con las películas (inventory_id → film_id).
film (películas) → Para obtener la tarifa de alquiler (rental_rate) y calcular el gasto total de cada cliente.

In [39]:
```python
# Unir rental → customer
df_clientes_1 = rental.merge(customer, on="customer_id", suffixes=("_rental", "_cust"))

# Unir con inventory
df_clientes_1 = df_clientes_1.merge(inventory, on="inventory_id", suffixes=("", "_inv"))

# Unir con film
df_clientes_1 = df_clientes_1.merge(film, on="film_id", suffixes=("", "_film"))

# Agrupar por cliente para calcular total de rentas y gasto total
df_clientes_summary = (df_clientes_1.groupby(["customer_id", "first_name", "last_name"])
                                    .agg(total_rentals=("rental_id", "count"),
                                         total_spent=("rental_rate", "sum"))
                                    .sort_values(by="total_spent", ascending=False))  # Ordenar por más gasto

# Mostrar el top 10 de clientes más rentables
print("Top 10 de clientes más rentables (por gasto total):")
print(df_clientes_summary.head(10))

# Ordenar por total de rentas (total_rentals) en orden descendente
df_clientes_summary_rentals = df_clientes_summary.sort_values(by="total_rentals", ascending=False)

# Mostrar el top 10 de clientes con más alquileres
print("\nTop 10 de clientes con más alquileres:")
print(df_clientes_summary_rentals.head(10))
```

```
Top 10 de clientes más rentables (por gasto total):
                                  total_rentals  total_spent
customer_id first_name last_name
148         ELEANOR    HUNT                  46       147.54
526         KARL       SEAL                  45       138.55
144         CLARA      SHAW                  42       135.58
178         MARION     SNYDER                39       134.61
459         TOMMY      COLLAZO               38       131.62
137         RHONDA     KENNEDY               39       126.61
410         CURTIS     IRBY                  38       121.62
236         MARCIA     DEAN                  42       121.58
403         MIKE       WAY                   35       118.65
368         HARRY      ARCE                  35       118.65

Top 10 de clientes con más alquileres:
                                  total_rentals  total_spent
customer_id first_name last_name
148         ELEANOR    HUNT                  46       147.54
526         KARL       SEAL                  45       138.55
236         MARCIA     DEAN                  42       121.58
144         CLARA      SHAW                  42       135.58
75          TAMMY      SANDERS               41       114.59
469         WESLEY     BULL                  40       105.60
197         SUE        PETERS                40       117.60
137         RHONDA     KENNEDY               39       126.61
468         TIM        CARY                  39       114.61
178         MARION     SNYDER                39       134.61
```

Respuesta: ELEANOR HUNT ha sido el cliente que más rentas ha realizado y el que más dinero ha gastado

In [ ]:

3) ¿Las películas más largas se rentan más o generan más ingresos que las más cortas?

Tablas necesarias

rental (alquileres) → Para contar cuántas veces se alquiló cada película.
inventory (inventario) → Para relacionar los alquileres con las películas.
film (películas) → Para obtener la duración (length) y la tarifa de alquiler (rental_rate).

In [40]:
```
# Definir categorías de duración de películas
bins = [0, 60, 120, np.inf]  # Definimos los intervalos
labels = ["Corta (< 60 min)", "Media (60-120 min)", "Larga (> 120 min)"]
film_length = film.copy()
film_length["length_category"] = pd.cut(film_length["length"], bins=bins, labels=labels, right=False)

# Unir rental → inventory → film
rental_inventory_film = rental.merge(inventory, on="inventory_id").merge(film_length, on="film_id")

# Agrupar por categoría de duración para calcular número de rentas e ingresos
length_summary = (rental_inventory_film.groupby("length_category", observed=False)
                  .agg(total_rentals=("rental_id", "count"),
                       total_income=("rental_rate", "sum"))
                  .sort_values(by="total_rentals", ascending=False))  # Ordenar por más rentas

# Mostrar resultados
print(length_summary)
```

```
                  total_rentals  total_income
length_category
Larga (> 120 min)          7412      22409.88
Media (60-120 min)         7101      20577.99
Corta (< 60 min)           1531       4223.69
```

Respuesta: las peliculas con duración mayor a dos horas han dado más ingresos

4) ¿Cuál es el rendimiento de cada tienda en términos de ingresos y alquileres?

Tablas necesarias

rental (alquileres) → Para contar cuántas veces se alquiló una película en cada tienda (rental_id).
inventory (inventario) → Para relacionar los alquileres con las películas y las tiendas (inventory_id → store_id, film_id).
store (tiendas) → Para identificar cada tienda (store_id).
film (películas) → Para obtener la tarifa de alquiler (rental_rate) y calcular los ingresos totales por tienda.

In [41]:
```
# Asegurar que 'store_id' tenga el mismo tipo en ambas tablas
inventory["store_id"] = inventory["store_id"].astype(int)
store["store_id"] = store["store_id"].astype(int)

# Unir rental → inventory → store → film con sufijos para evitar duplicados
df_tiendas = (rental.merge(inventory, on="inventory_id", suffixes=("_rental", "_inventory"))
                    .merge(store, on="store_id", suffixes=("_inventory", "_store"))
                    .merge(film, on="film_id", suffixes=("_store", "_film")))

# Eliminar columnas 'last_update' redundantes si existen
df_tiendas = df_tiendas.loc[:, ~df_tiendas.columns.duplicated()]

# Agrupar por tienda y calcular total de alquileres e ingresos
df_tiendas_summary = (df_tiendas.groupby("store_id", observed=False)
                                .agg(total_rentals=("rental_id", "count"),
                                     total_income=("rental_rate", "sum"))
                                .sort_values(by="total_income", ascending=False))  # Ordenar por ingresos

# Mostrar resultados
print(df_tiendas_summary)
```

```
          total_rentals  total_income
store_id
2                 16032      47199.68
1                    12         11.88
```

Respuesta: la tienda 2 es la que genera casi toda la ganancia

5) ¿Qué categorías de películas han dado mayor ingreso?

Tablas necesarias

rental (alquileres) → Para contar cuántas veces se alquiló cada película (rental_id).
inventory (inventario) → Para relacionar los alquileres con las películas (inventory_id → film_id).
film (películas) → Para obtener la clasificación de la película (rating) y la tarifa de alquiler (rental_rate), con lo que se calcula el ingreso total por categoría.

In [42]:
```
# Unir rental → inventory → film
df_reporte = (rental.merge(inventory, on="inventory_id")
                    .merge(film, on="film_id"))

# Agrupar por categoría (rating) y calcular total de alquileres e ingresos
df_reporte_summary = (df_reporte.groupby("rating", observed=False)
                                .agg(total_rentals=("rental_id", "count"),
                                     total_income=("rental_rate", "sum"))
                                .sort_values(by="total_rentals", ascending=False))  # Ordenar por rentas

# Mostrar resultados
print(df_reporte_summary)
```

```
        total_rentals  total_income
rating
PG-13            3585      10797.15
NC-17            3293      10062.07
PG               3212       9465.88
R                3181       9011.19
G                2773       7875.27
```

Respuesta: Las peliculas de categoria PG-13 son las que han dado mayor ingresos

¿Cuáles son los períodos de mayor y menor demanda en las tiendas?

Tablas necesarias

rental (alquileres) → Para contar cuántas veces se realizaron alquileres (rental_id).
rental (alquileres) → Para extraer el año y mes de la fecha de alquiler (rental_date).

In [43]:
```
# Convertir rental_date a tipo datetime
rental["rental_date"] = pd.to_datetime(rental["rental_date"])

# Extraer año y mes
rental["year"] = rental["rental_date"].dt.year
rental["month"] = rental["rental_date"].dt.month

# Agrupar por año y mes, contando el número de rentas
df_demand = (rental.groupby(["year", "month"])["rental_id"]
             .count()
             .reset_index()
             .rename(columns={"rental_id": "total_rentals"}))
```

```python
# Ordenar por año y mes
df_demand = df_demand.sort_values(by=["year", "month"])

# Mostrar los primeros 12 resultados
print(df_demand.head(12))
```

```
   year  month  total_rentals
0  2005      5           1131
1  2005      6           2336
2  2005      7           6569
3  2005      8           5826
4  2006      2            182
```

Respuesta: el mes 7 (julio) es el que ha generado mayor rentas