Ejercicio 3: Creación de Nuevas Columnas y Análisis por Grupo

Utiliza el dataFrame top_youtube.

- Crea una nueva columna llamada "Likes_to_Views" que represente la proporción de "Likes" respecto a "Views", con dos decimales.
- Agrupa las canciones de top_youtube por "Album_type" y calcula:
- Promedio y mediana de "Energy" y "Danceability".
- Total de "Views" y "Stream" por tipo de álbum.
- Guarda el resultado de la agrupación en un nuevo dataFrame llamado album_analysis.

In [6]:
```
# Crear la nueva columna "Likes_to_Views" como proporción de Likes respecto a Views
top_youtube['Likes_to_Views'] = (top_youtube['Likes'] / top_youtube['Views']).round(2)

# Agrupar por "Album_type" y calcular estadísticas
album_analysis = top_youtube.groupby('Album_type').agg({
    'Energy': ['mean', 'median'],
    'Danceability': ['mean', 'median'],
    'Views': 'sum',
    'Stream': 'sum'
}).reset_index()

print(album_analysis.head())
```

```
   Album_type    Energy         Danceability                Views  \
                    mean  median         mean median           sum
0       album  0.683681  0.7095     0.670642  0.692  7.055829e+11
1  compilation  0.644563  0.6810     0.680406  0.734  2.843627e+10
2      single  0.683443  0.6720     0.694234  0.727  1.722522e+11

         Stream
            sum
0  5.216714e+11
1  9.327303e+09
2  9.019371e+10
```

```
C:\Users\isaia\AppData\Local\Temp\ipykernel_27692\3551823639.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  top_youtube['Likes_to_Views'] = (top_youtube['Likes'] / top_youtube['Views']).round(2)
```

Ejercicio 4: Identificación de Canciones con Baja Proporción de Likes

Utiliza el dataFrame top_youtube.

- Filtra todas las canciones cuya proporción "Likes_to_Views" sea menor a 0.01.
- Guarda este subconjunto en un dataFrame llamado low_likes.
- Dentro de low_likes, calcula el número total de canciones por "Album_type" y guarda el resultado en un dataFrame llamado low_likes_summary.

In [7]:
```python
# Filtrar canciones con proporción "Likes_to_Views" menor a 0.01
low_likes = top_youtube[top_youtube['Likes_to_Views'] < 0.01]

# Calcular el número total de canciones por "Album_type"
low_likes_summary = low_likes.groupby('Album_type').size().reset_index(name='count')

low_likes_summary.head()
```

Out[7]:

	Album_type	count
0	album	259
1	compilation	10
2	single	63

Ejercicio 5: Análisis de Tendencias de Canciones con Baja Proporción de Likes

Utiliza el dataFrame low_likes.

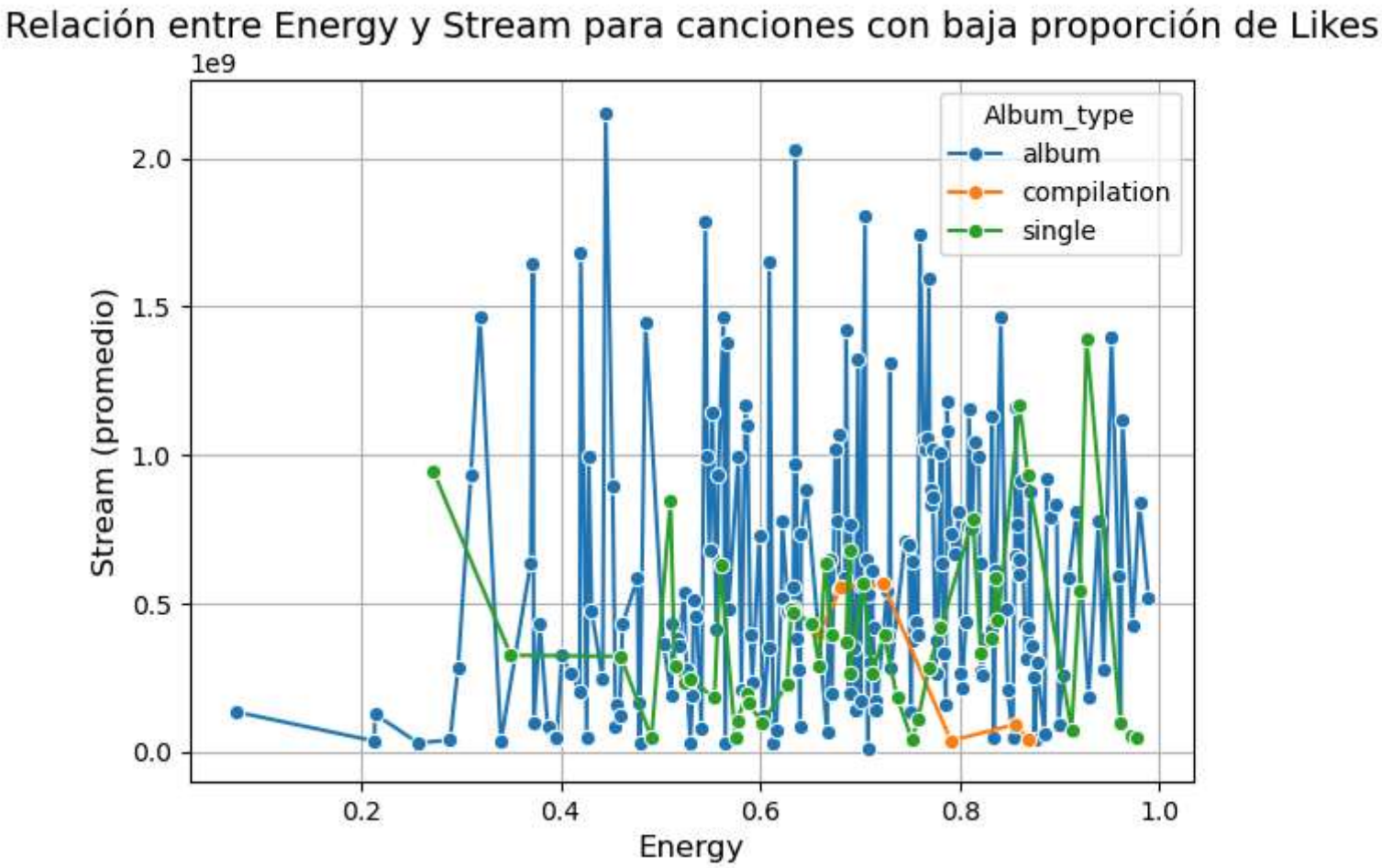
Crea un gráfico de líneas que muestre:

- La relación promedio entre "Stream" y "Energy" para las canciones en low_likes.
- En el eje X: "Energy".
- En el eje Y: promedio de "Stream".
- Diferencia los tipos de álbum ("Album_type") con colores en el gráfico.
- Asegúrate de incluir título, leyendas, y etiquetas de ejes.

In [8]:
```
# Promedio de "Stream" por "Energy" y "Album_type" en el subconjunto `low_likes`
trend_data = low_likes.groupby(['Album_type', 'Energy']).agg({'Stream': 'mean'}).reset_index()


# Personalizar el gráfico
plt.title('Relación entre Energy y Stream para canciones con baja proporción de Likes', fontsize=14)
plt.xlabel('Energy', fontsize=12)
plt.ylabel('Stream (promedio)', fontsize=12)
plt.legend(title='Album Type')
plt.grid(True)
plt.tight_layout()
sns.lineplot(data=trend_data, x='Energy', y='Stream', hue='Album_type', marker='o')
```

```
C:\Users\isaia\AppData\Local\Temp\ipykernel_27692\2126821775.py:9: UserWarning: No artists with labels found to put i
n legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argumen
t.
  plt.legend(title='Album Type')
```

Out[8]:
```
<Axes: title={'center': 'Relación entre Energy y Stream para canciones con baja proporción de Likes'}, xlabel='Energ
y', ylabel='Stream (promedio)'>
```



Ejercicio 6: Rango de valores de danceability

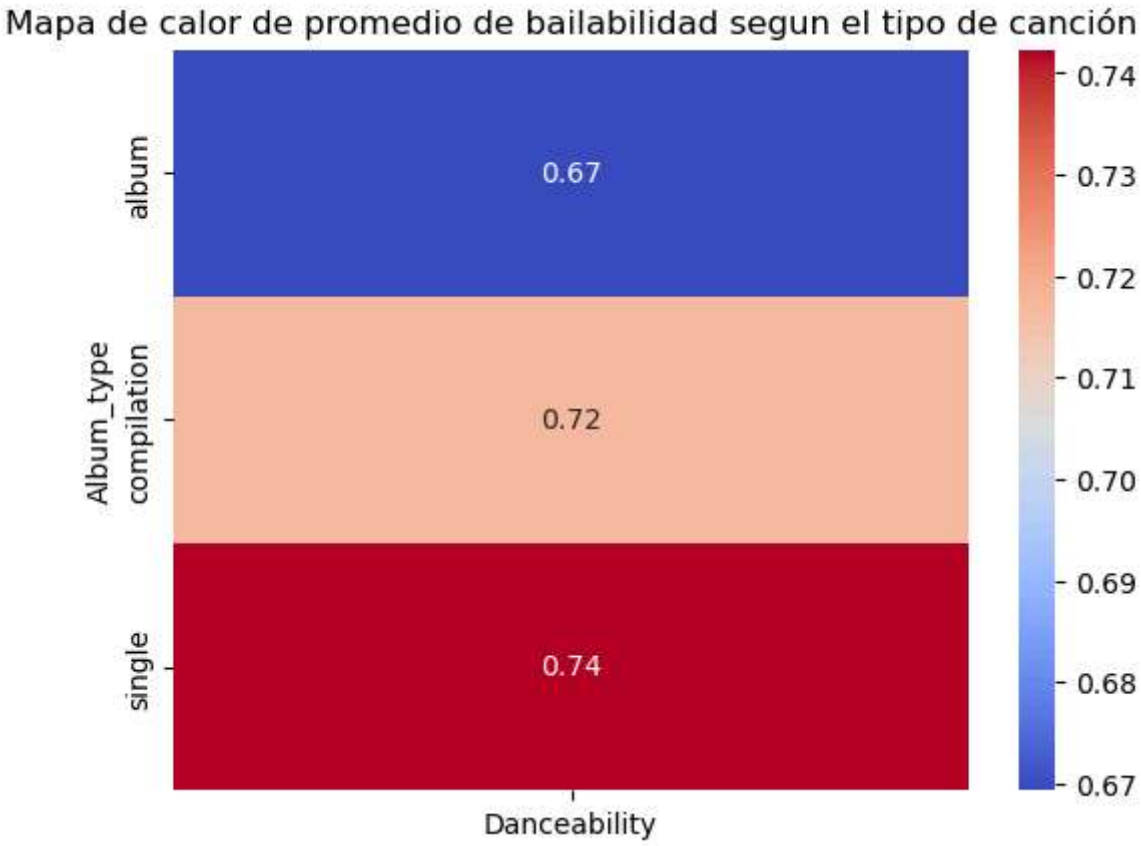
Utiliza el dataFrame low_likes.

- Muestra el rango de valores existente para los diferentes tipos de canciones en spotify (columna Album_type) para la columna danceability en low_likes en un mapa de calor.
- Asegúrate de incluir una barra de colores, etiquetas de ejes, y un título descriptivo.

In [9]:
```
heatmap_data = low_likes.groupby('Album_type', as_index=False)['Danceability'].mean().set_index('Album_type')
```

```
# Personalizar el mapa de calor
plt.title('Mapa de calor de promedio de bailabilidad segun el tipo de canción')
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm', fmt='.2f')
```

Out[9]: <Axes: title={'center': 'Mapa de calor de promedio de bailabilidad segun el tipo de canción'}, ylabel='Album_type'>



Ejercicio 7: Análisis de Artistas

Utiliza el dataFrame low_likes.

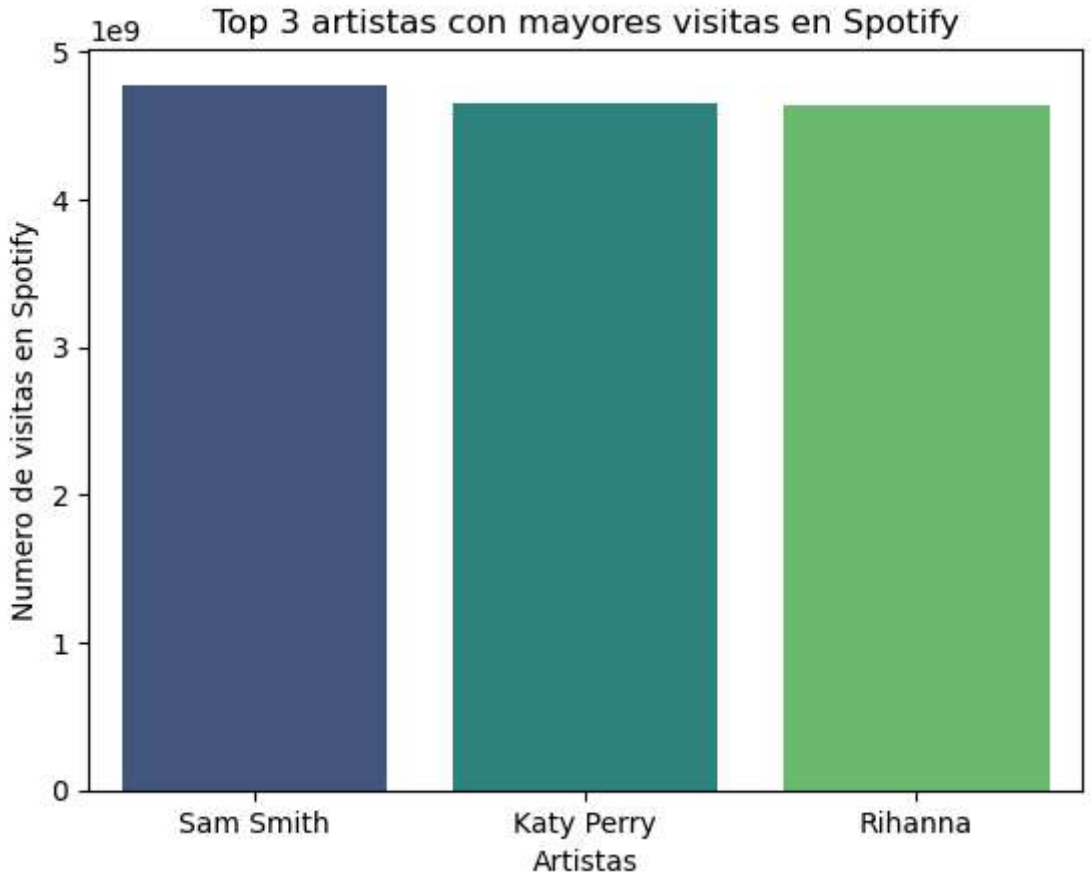
- Identifica los tres artistas con el mayor total de "Stream" en este subconjunto.
- Presenta un dataFrame con el nombre del artista y el total correspondiente, ordenados de mayor a menor.

- Genera un gráfico de barras que compare estos totales entre los artistas identificados, en el eje X debe ir la variable stream y en el eje Y la variable Artist, para ello utiliza el método barplot() de la librería seaborn.

In [10]:
```
# Identificar los tres artistas con el mayor total de "Stream"
top_artists = low_likes.groupby('Artist')['Stream'].sum().sort_values(ascending=False).reset_index().head(3)

# Personalizar el gráfico
plt.title('Top 3 artistas con mayores visitas en Spotify')
plt.xlabel('Artistas')
plt.ylabel('Numero de visitas en Spotify')

# Crear un gráfico de barras para comparar los totales
sns.barplot(data=top_artists, x='Artist', y='Stream', palette='viridis')

# Mostrar los datos del análisis
print(top_artists)
```

```
      Artist        Stream
0   Sam Smith  4.775585e+09
1  Katy Perry  4.655496e+09
2     Rihanna  4.641587e+09
```

```
C:\Users\isaia\AppData\Local\Temp\ipykernel_27692\1611777431.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `h
ue` and set `legend=False` for the same effect.

  sns.barplot(data=top_artists, x='Artist', y='Stream', palette='viridis')
```


Top 3 artistas con mayores visitas en Spotify
1e9
5
4
3
2
1
0
Numero de visitas en Spotify
Sam Smith
Katy Perry
Rihanna
Artistas