In [18]:

```
#Creating toggle to make code into a drop down - easier to read blog

from IPython.display import HTML, display

def toggle_code():
    display(HTML('''<script>
    code_show=true;
    function code_toggle() {
        if (code_show){
            $('div.input').hide();
            $("#code_button").html("Click here if you want to see the code");
        } else {
            $('div.input').show();
            $("#code_button").html("Click here to hide the code");
        }
        code_show = !code_show
    }
    $( document ).ready(code_toggle);
    </script>
    <form action="javascript:code_toggle()"><input type="submit" id="code_button" value="Click here if you want to
```

In [19]:

```
toggle_code()
```

Click here if you want to see the code!

In [20]:

```
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap

# Creating a fun title for the blog
cmap = LinearSegmentedColormap.from_list('ombre_blue', ['#00008B', '#0000FF', '#1E90FF', '#87CEEB', '#ADD8E6'])

# Spacing looked strange so have manually adjusted it to look better
text = 'S p o tify S tre a m  in g :W  h a t a re w e lis te n in g to ?'
fig, ax = plt.subplots()
spacing = 0.15
for i, char in enumerate(text):
    ax.text(-5 + i*spacing, 0, char, font='Arial', fontsize=150, ha='left', va='center', color=cmap(i/len(text)))
ax.axis('off')

plt.show()
```

Analysing spotify 'Most Streamed' and 'Number One' songs over the past fifteen years.

As one of the first platforms to offer music streaming, Spotify created a whole new way to listen to music. Launching in 2006 in Sweden, Spotify hosts over 602 million monthly active users with over 100 million songs to choose from.

This blog aims to uncover how we have chosen to listen to music using Spotify, looking at data covering the songs with the highest total amount of streams on the platform and data on songs that spent at least a week at number one (according to weekly streams).

So... WHY are streams important?

Spotify compensates artists based on the amount of streams they have received - thus the more streams, the more income artists get. In the 21st century, where purchasing physical music is few and far between, this income is vital for artists and so understanding streaming trends is essential.

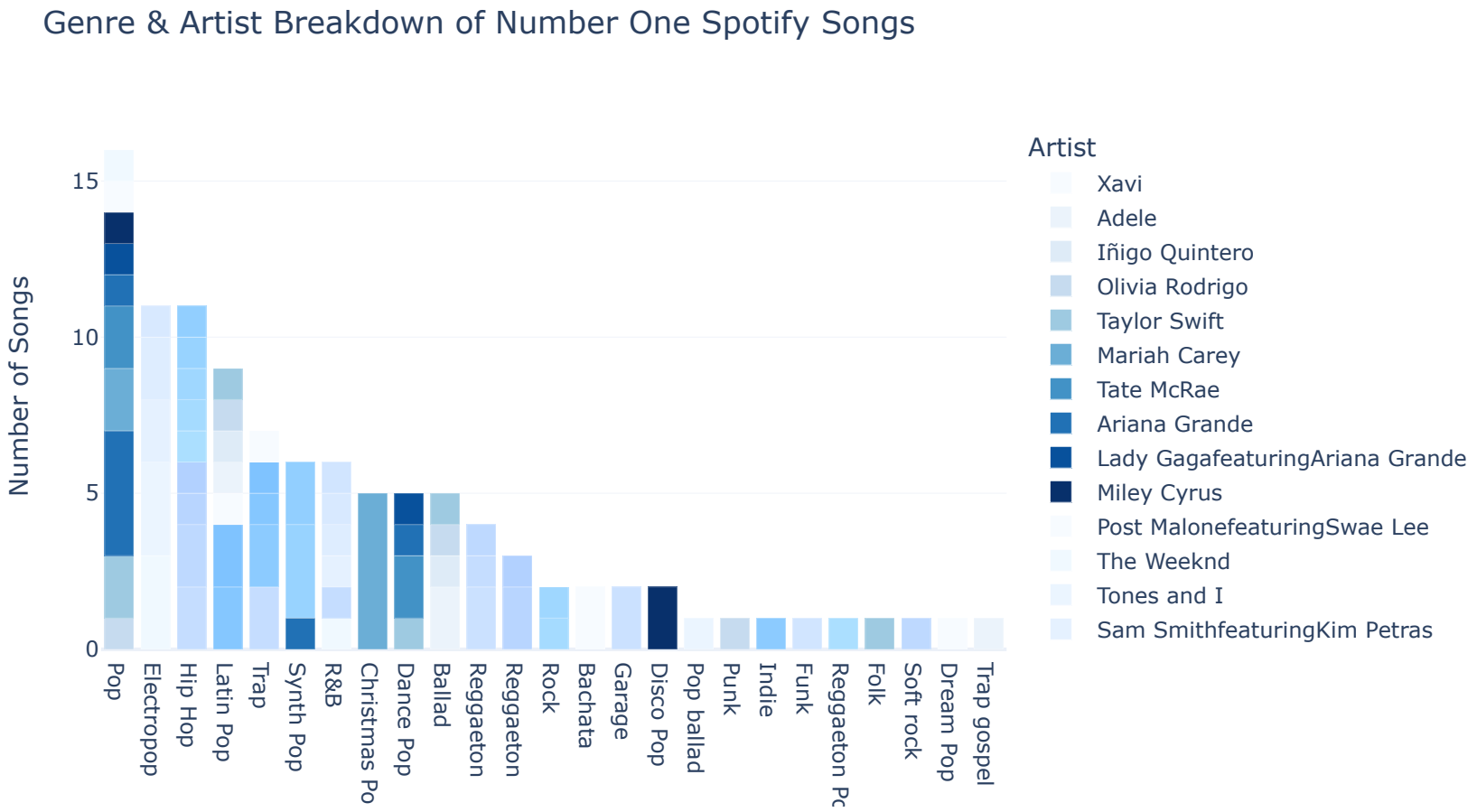
Furthermore, Spotify advertises certain songs to their listeners. Songs that have more streams are much more likely to be advertised, leading to more streams, leading to more advertisement ... a vicious never ending circle (until the next hit comes along!)

WHAT are we choosing to listen to?

In [21]:
```
#import important libraries
import pandas as pd
import numpy as np
import csv
import plotly.express as px

#load data
Number_one = pd.read_csv("List of number-one songs on Spotify.csv", encoding='latin1')
Most_streamed = pd.read_csv("Most-streamed songs.csv", encoding='latin1')

#Remove last column of Most streamed data as it only consists of references
Most_streamed = Most_streamed.iloc[:, :-1]
```

In [22]:

```
# Calculate total count of artists for each genre
counts = Number_one.groupby(['Genre', 'Artist']).size().reset_index(name='Count')
total_artists = counts.groupby('Genre')['Count'].sum().reset_index(name='Total_Artists')

# Make bar chart looking at genre and artist
counts_sorted = counts.sort_values(by=['Genre', 'Count'], ascending=[True, False])

genre_order = total_artists.sort_values(by='Total_Artists', ascending=False)['Genre']

# Colour chart
custom_blue_scale = [
    '#f7fbff', '#ebf3fb', '#deebf7', '#c6dbef', '#9ecae1',
    '#6baed6', '#4292c6', '#2171b5', '#08519c', '#08306b',
    '#f7fbff', '#f0f9ff', '#ebf5ff', '#e5f1ff', '#deedff',
    '#d8e9ff', '#d1e5ff', '#cbe1ff', '#c5ddff', '#bed9ff',
    '#b8d5ff', '#b2d1ff', '#abdfff', '#a5dbff', '#9fd7ff',
    '#98d3ff', '#92cfff', '#8ccbff', '#85c7ff', '#7fc3ff'
]

# Create bar chart
fig = px.bar(counts_sorted, x='Genre', y='Count', title='Genre & Artist Breakdown of Number One Spotify Songs',
             color='Artist',
             color_discrete_sequence=custom_blue_scale,
             template='plotly_white',
             barmode='stack',
             category_orders={"Genre": genre_order},
             labels={'Count': 'Number of Songs'}
            )
fig.show()
```



As you can see, (and to no surprise), 'Pop' music comes out as the most popular genre when looking at number one songs, followed by 'Electropop' and 'Hip Hop'. Unsurprisingly, sub genres of Pop are also thriving, with genres such as 'Synth Pop', 'Electropop' and 'Latin Pop' also ranking in the top six genres with the most number one songs. Interestingly, genres which were the most popular throughout the 20th century (such as Opera, Jazz and Country) now fail to even provide one song in the number one songs on Spotify. This signifies the change in listening habits which has been helped along by streaming services. As music is now readily available for free, consumers are able to change listening patterns instantly, instead of having to wait for the next chance to buy a record.

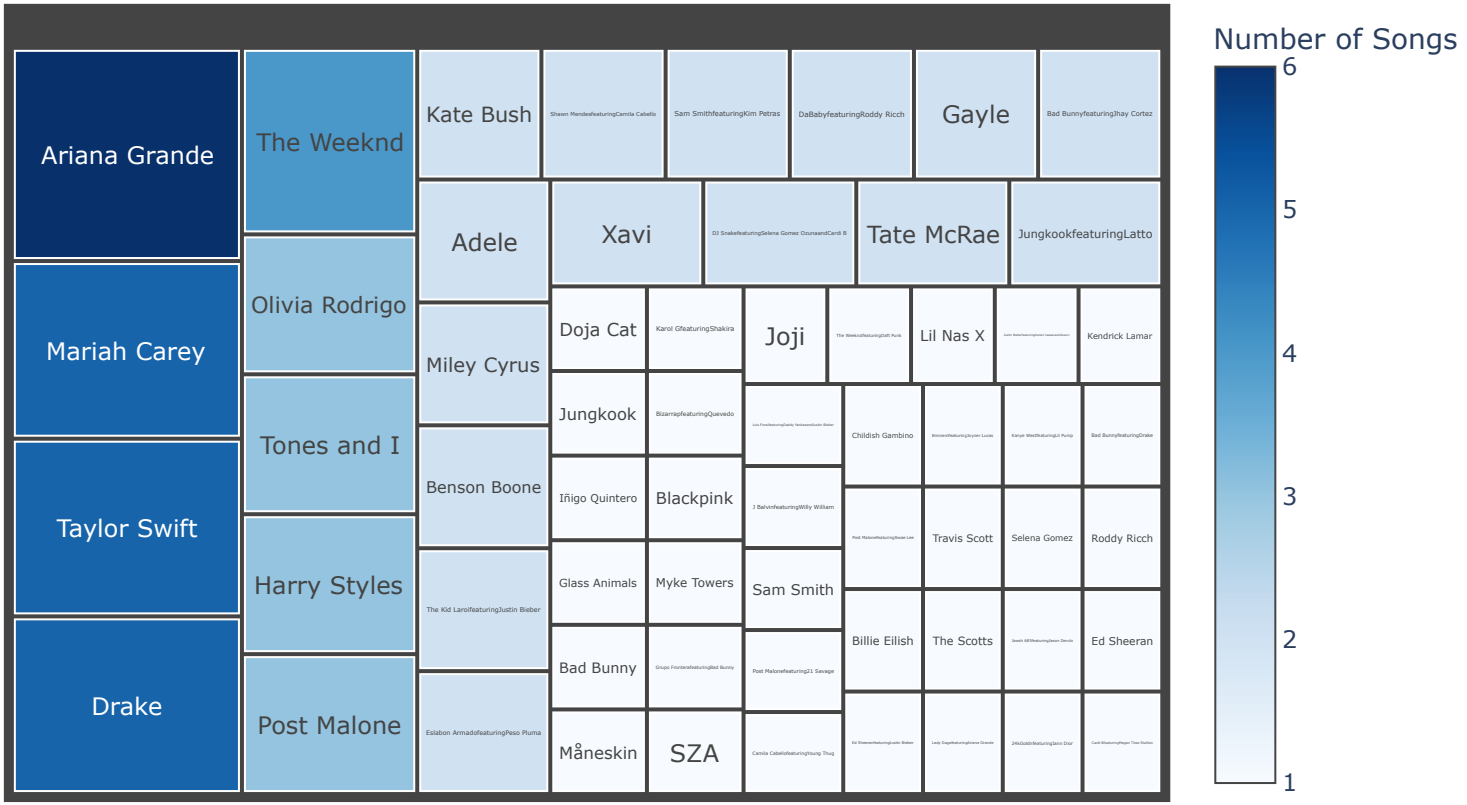
WHO are we choosing to listen to?

In [23]:

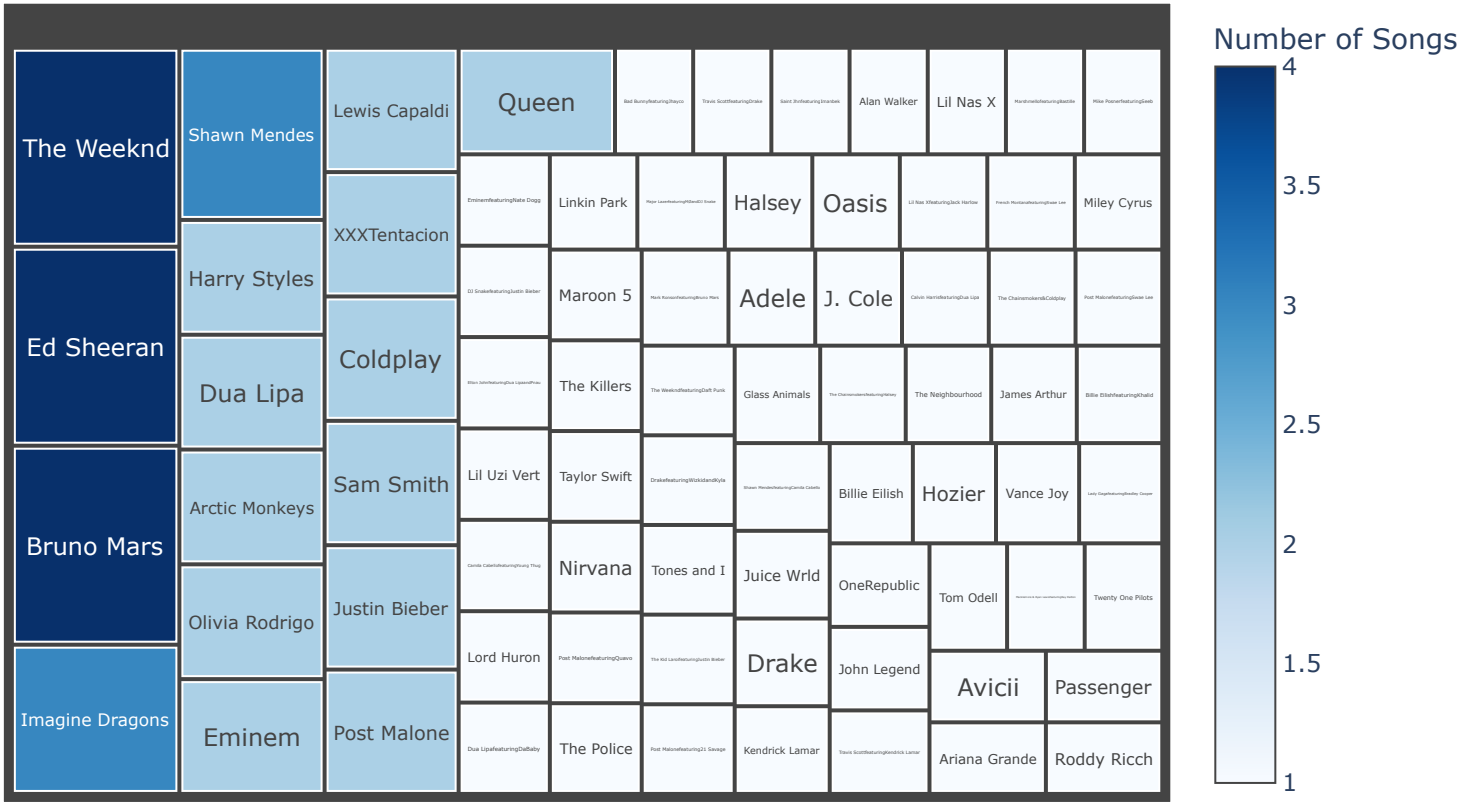
```
import plotly.graph_objects as go

# Count the number of songs per artist for Number One songs
song_counts_number_one = Number_one['Artist'].value_counts().reset_index()
song_counts_number_one.columns = ['Artist', 'Song']

# Count the number of songs per artist for Most Streamed songs
song_counts_most_streamed = Most_streamed['Artist'].value_counts().reset_index()
song_counts_most_streamed.columns = ['Artist', 'Song']

# Create the Treemap for Number One Songs
fig1 = go.Figure(go.Treemap(
    labels=song_counts_number_one['Artist'],
    parents=[''] * len(song_counts_number_one),
    values=song_counts_number_one['Song'],
    marker=dict(
        colorscale='Blues',
        colorbar=dict(title='Number of Songs')
    ),
    textposition='middle center',
    branchvalues='total',
))

fig1.update_layout(
    title='Artists with Number One Songs',
    height=500,
    width=800,  # Adjust the width to fit side by side
    margin=dict(t=50, b=50, l=50, r=50),
)

# Create the Treemap for Most Streamed Songs
fig2 = go.Figure(go.Treemap(
    labels=song_counts_most_streamed['Artist'],
    parents=[''] * len(song_counts_most_streamed),
    values=song_counts_most_streamed['Song'],
    marker=dict(
        colorscale='Blues',
        colorbar=dict(title='Number of Songs')
    ),
    textposition='middle center',
    branchvalues='total',
```

```
))

fig2.update_layout(
    title='Artists with Most Streamed Songs',
    height=500,
    width=800,
    margin=dict(t=50, b=50, l=50, r=50),
)

fig1.show()
fig2.show()
```

Artists with Number One Songs



Artists with Most Streamed Songs



Ariana Grande has had the most 'Number One' songs over the past two decades, with 6 holding the top spot on Spotify. This makes sense as all of her music falls under the most popular genre, 'Pop' or the sub genres that fall under pop music (e.g. 'Dance Pop'). Interestingly though, number ones do not necessarily imply a lot of streams as she only holds one song in the 'Most Streamed' list.

Similarly too, Ed Sheeran and Bruno Mars hold joint first place with most songs streamed (with 4 songs on the list) however Ed Sheeran has only ever received one number one song, and Bruno mars hasn't received any. Interestingly too, its more likely for artists to have lots of streams without having number ones, with 78 artists in the top streamed list (76% of the list being different artitsts) compared to 62 in that of number ones (with only 59% of the list being unique). This suggests that certain artists may have specific influential fans who help to determine top spots in

charts (at specific points in time), but fail to follow through on streams long term. Perhaps the artists on the most streamed list may be listened to by many people across long periods of time. The one outlier of this is The Weekend, who rates in the top 5 for most number ones (he has 4) and is also joint first for the most streamed songs.

Furthermore, an interesting artist to note is Drake, who is joint second for the most amount of number one songs. He is a bit of an anomoly as he does not produce pop music (or a pop subgenre). However, he does fail to excel in most streamed songs, only releasing one song which has

WHEN do the most number ones occur?

In [24]:

```
# Import necessary libraries
from wordcloud import WordCloud
from collections import Counter

# Duplicate datatsets because it was causing an issue
Number_one_copy = Number_one.copy()
Most_streamed_copy = Most_streamed.copy()

# Extract month from the date column
Number_one_copy['Month'] = Number_one_copy['Release Date'].str.split('-').str[1]
month_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

Most_streamed_copy['Month'] = Most_streamed_copy['ReleaseDate'].str.split('-').str[1]
month_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

# Count and sort the frequency of each month for Number_one_copy
month_counts = Counter(Number_one_copy['Month'])
sorted_month_counts = dict(sorted(month_counts.items(), key=lambda x: x[1], reverse=True))

# Create a WordCloud
wordcloud = WordCloud(width=800, height=600, colormap='Blues', background_color='white').generate_from_frequencies
plt.figure(figsize=(8, 6))

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('When Were Number One Songs Released?')

# Annotate the WordCloud with the count for each month
vertical_offset = 0
for month, count in sorted_month_counts.items():
    plt.text(1.2, 0.9 - vertical_offset, f"{month}: {count} songs", fontsize=12, ha='left', va='center', transform
    vertical_offset += 0.05

plt.show()

# Count and sort the frequency of each month for Most_streamed_copy
month_counts = Counter(Most_streamed_copy['Month'])
sorted_month_counts = dict(sorted(month_counts.items(), key=lambda x: x[1], reverse=True))

# Create a WordCloud
wordcloud = WordCloud(width=800, height=600, colormap='Blues', background_color='white').generate_from_frequencies
```

```
plt.figure(figsize=(8, 6))

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('When Where the Most Streamed Songs Released?')

# Annotate the WordCloud with the count for each month
vertical_offset = 0
for month, count in sorted_month_counts.items():
    plt.text(1.2, 0.9 - vertical_offset, f"{month}: {count} songs", fontsize=12, ha='left', va='center', transform
    vertical_offset += 0.05

plt.show()
```

When Were Number One Songs Released?

Nov: 14 songs
Sep: 12 songs
Oct: 12 songs
Apr: 11 songs
Jan: 10 songs
Aug: 10 songs
Mar: 9 songs
Jun: 9 songs
May: 7 songs
Jul: 7 songs
Dec: 3 songs
Feb: 1 songs

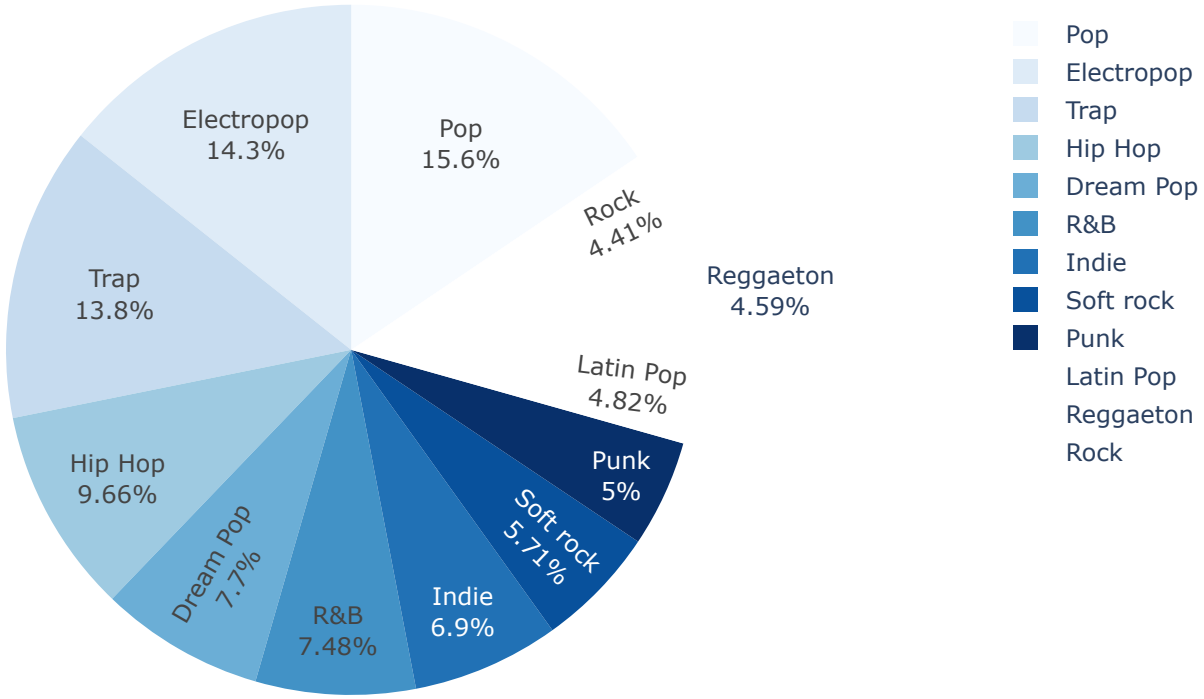
Interestingly, September performs well in both number one songs and most streamed songs. This may be because it is a common month for new things - people going back to school for example, who may be playing more music (for example in their journey to school) or people coming back from holidays (and listening to music on their way to work). April and October also seem to perform relatively well. Aside from this, there is little correlation between the months with the most streams and number ones. However, it does seem safe to say, if you want to produce music that performs well in the charts and streams, don't release it in Feburary, July or December - these months see very little songs hit either number one or the most streamed charts.

Of course this may be causation rather than correlation. For example, artists may take a break in December to enjoy the holidays and so not release new music. Similarly, according to https://www.artistandproducer.com/planning/best-time-to-release-music-hour-day-month/ (https://www.artistandproducer.com/planning/best-time-to-release-music-hour-day-month/), artists should avoid releasing new music in July because it is festival season and so music listeners are likely to be listening to the music in person, rather than online. These reasons may be why we fail to see as many successful songs in these months. Saying this, February is an anomoly month to be performing so poorly, with little to distract consumers from listening to music. Potential reasons could be to do with the shortness of the month (and so there are 3 less days for

Now lets have a look at songs that have reached 'Number One' for at least one week and are also in the 'Most Streamed' list.

In [25]:

```
# Merge the datasets
merged_df = pd.merge(Number_one, Most_streamed, on=['Song'], how='inner')

# Calculate total streams/percentage per genre
total_streams_per_genre = merged_df.groupby('Genre')['Streams(billions)'].sum().reset_index()
total_streams_per_genre['Percentage'] = (total_streams_per_genre['Streams(billions)'] / total_streams_per_genre['S

# Create pie chart
fig = px.pie(total_streams_per_genre,
             values='Streams(billions)',
             names='Genre',
             title='Share of Streams by Genre for Most Streamed Songs That Have Also Reached Number One',
             color_discrete_sequence=px.colors.sequential.Blues,
             )
fig.update_traces(hoverinfo='label+percent+value', textinfo='label+percent')
fig.show()
```



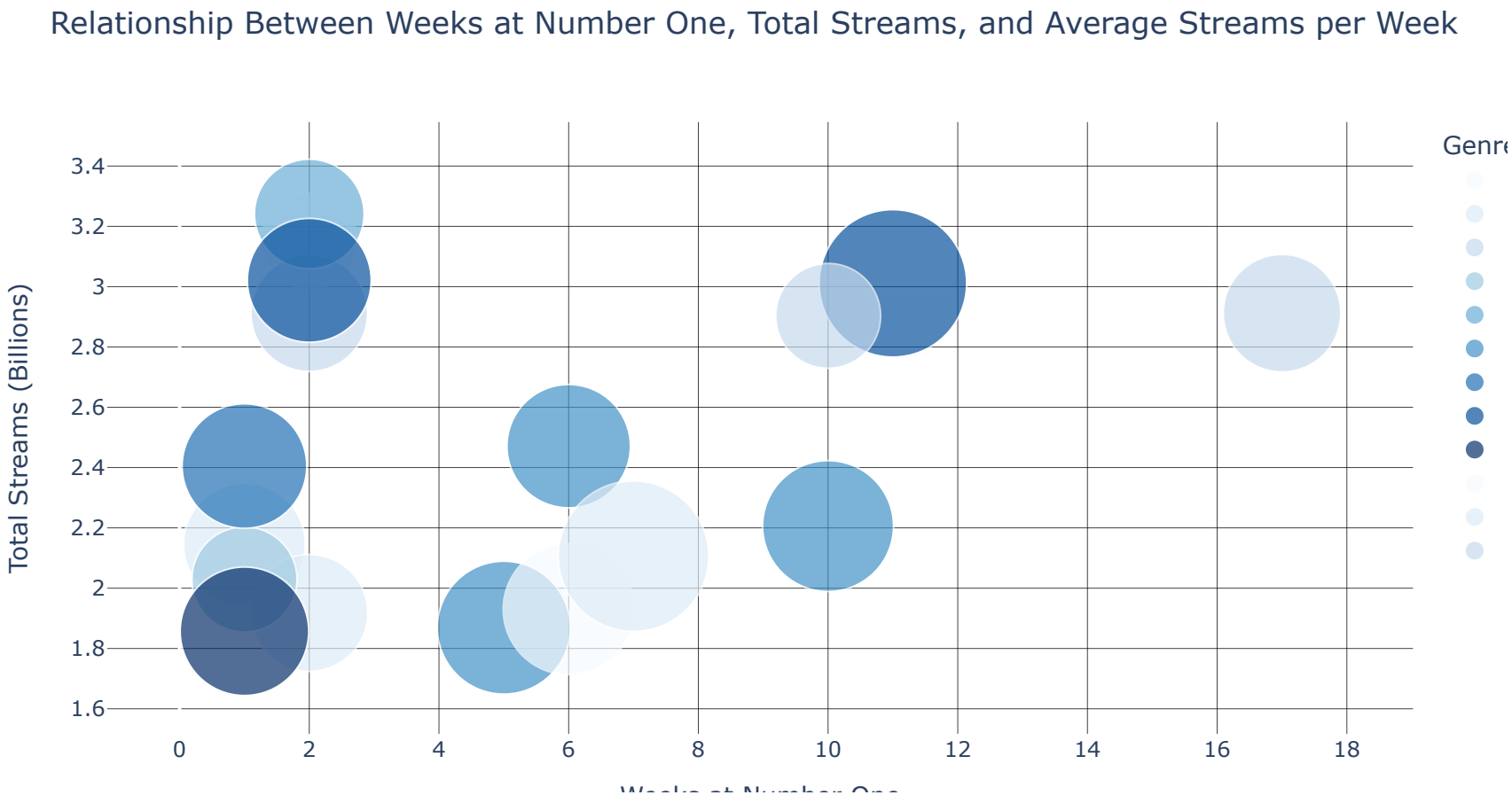
Initially looking at the combined datasets shows that only 14 songs managed to reach both number one and obtain one of the most amount of streams. This is interesting as both original datasets have over 100 songs, suggesting that only the most elite songs can manage to reach this status. Furthermore, looking at the breakdown by genre provides even more interesting analysis. I would expect a clear genre to stand out, and although 'Pop' music once again has the highest number of songs in this section, it only holds 15.6% of streams. In fact, there is a lot of genres that have reached this status, which shows that artists can perform exceptionally well, no matter the genre they produce. Saying this, some genres have failed to reach both 'Number One' and 'Most Streamed' such as Funk, Garage and Ballads. Christmas Pop, which has one of the

highest amount of number ones also fails to reach this list although this may be unsurprising when we think about it contextually - christmas

Finally, which songs are the most popular?

In [26]:

```
# Create scatter plot looking at streams, weeks at number one, and average streams per week
fig = px.scatter(merged_df,
                 x="Weeks",
                 y="Streams(billions)",
                 size="Avs.",
                 title="Relationship Between Weeks at Number One, Total Streams, and Average Streams per Week",
                 color="Genre",
                 hover_name="Song",
                 size_max=60,
                 color_discrete_sequence=px.colors.sequential.Blues)

fig.update_layout(
    plot_bgcolor='white',
    xaxis=dict(showgrid=True, gridcolor='black', title="Weeks at Number One", dtick=2),
    yaxis=dict(showgrid=True, gridcolor='black', title="Total Streams (Billions)", dtick=0.2),
)

fig.show()
```



The visualisation above shows how many weeks a song managed to stay at number one (x axis) while also taking into account the number of streams it reached (y axis) and the average weekly streams in millions (the size of the bubbles). As you can see there is a positive pattern between the highest performing songs, with more streams generally suggesting a song stays at number one for longer. Saying this, a lot of songs debut at the top spot for just one or two weeks but still reach a high number of streams.

The song that has received the most amount of streams while also receiving the number one top spot is 'Sunflower' by Post Malone. Extremely interestingly too, the song that has received the most weeks at number one is 'Rockstar' also by Post Malone! As an artist who performs reasonably well in both getting number ones and streams (although he is no where near the top 5 for either), producing music which generally falls in the 'Hip Hop' or 'Trap' categories, these are very interesting results. According to the Fader, (https://www.thefader.com/2017/10/17/post-malone-youtube-rockstar-chorus-loop (https://www.thefader.com/2017/10/17/post-malone-youtube-rockstar-chorus-loop)) a lot of the streams for 'Rockstar' may have come from a clever marketing scheme, whcih tricked listeners into streaming the song more. Furthermore, the success of 'Sunflower' may have emerged from the release of Spider-Man: Into the Spider-Verse, where the song featured. However it came about, Post Malone is the most successful Spotify artist according to my results!

All of this goes to show that it doesn't matter what kind of music you're producing (to an extent). Being clever about your marketing strategy and having a loyal fan base is the most important way to gain a lot of streams.

In the future, it would be great to be able to look at long term patterns using music downloads or record purchases to see how music listening has changed over the past century. In a world where music is changing so rapidly (and songs are now gaining popularity through social medias), it will be interesting to see what the most popular music will look like in 10 years time!