Data Cleaning and Pre-processing

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

In [2]:
```
df=pd.read_csv("C:/Users/Nishanth/Desktop/SENCHOLA INTERNSHIP/Task11/Global YouTube Statistics.csv", encoding='latin')
df
```

Out[2]:

	rank	Youtuber	subscribers	video views	category	Title	uploads	Country	Abbreviation	channel_type	...	subscribers_for_last_30_days	created
0	1	T-Series	245000000	2.280000e+11	Music	T-Series	20082	India	IN	Music	...	2000000.0	2
1	2	YouTube Movies	170000000	0.000000e+00	Film & Animation	youtubemovies	1	United States	US	Games	...	NaN	2
2	3	MrBeast	166000000	2.836884e+10	Entertainment	MrBeast	741	United States	US	Entertainment	...	8000000.0	2
3	4	Cocomelon - Nursery Rhymes	162000000	1.640000e+11	Education	Cocomelon - Nursery Rhymes	966	United States	US	Education	...	1000000.0	2
4	5	SET India	159000000	1.480000e+11	Shows	SET India	116536	India	IN	Entertainment	...	1000000.0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	
990	991	Natan por Aï¿	12300000	9.029610e+09	Sports	Natan por Aï¿	1200	Brazil	BR	Entertainment	...	700000.0	2
991	992	Free Fire India Official	12300000	1.674410e+09	People & Blogs	Free Fire India Official	1500	India	IN	Games	...	300000.0	2
992	993	Panda	12300000	2.214684e+09	NaN	HybridPanda	2452	United Kingdom	GB	Games	...	1000.0	2
993	994	RobTopGames	12300000	3.741235e+08	Gaming	RobTopGames	39	Sweden	SE	Games	...	100000.0	2
994	995	Make Joke Of	12300000	2.129774e+09	Comedy	Make Joke Of	62	India	IN	Comedy	...	100000.0	2

995 rows × 28 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 28 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   rank                                    995 non-null    int64  
 1   Youtuber                                995 non-null    object 
 2   subscribers                             995 non-null    int64  
 3   video views                             995 non-null    float64
 4   category                                949 non-null    object 
 5   Title                                   995 non-null    object 
 6   uploads                                 995 non-null    int64  
 7   Country                                 873 non-null    object 
 8   Abbreviation                            873 non-null    object 
 9   channel_type                            965 non-null    object 
 10  video_views_rank                        994 non-null    float64
 11  country_rank                            879 non-null    float64
 12  channel_type_rank                       962 non-null    float64
 13  video_views_for_the_last_30_days        939 non-null    float64
 14  lowest_monthly_earnings                 995 non-null    float64
 15  highest_monthly_earnings                995 non-null    float64
 16  lowest_yearly_earnings                  995 non-null    float64
 17  highest_yearly_earnings                 995 non-null    float64
 18  subscribers_for_last_30_days            658 non-null    float64
 19  created_year                            990 non-null    float64
 20  created_month                           990 non-null    object 
 21  created_date                            990 non-null    float64
 22  Gross tertiary education enrollment (%) 872 non-null    float64
 23  Population                              872 non-null    float64
 24  Unemployment rate                       872 non-null    float64
 25  Urban_population                        872 non-null    float64
 26  Latitude                                872 non-null    float64
 27  Longitude                               872 non-null    float64
dtypes: float64(18), int64(3), object(7)
memory usage: 217.8+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

	rank	subscribers	video views	uploads	video_views_rank	country_rank	channel_type_rank	video_views_for_the_last_30_days	lowest_monthly_earni
count	995.00000	9.950000e+02	9.950000e+02	995.000000	9.940000e+02	879.000000	962.000000	9.390000e+02	995.000
mean	498.00000	2.298241e+07	1.103954e+10	9187.125628	5.542489e+05	386.053470	745.719335	1.756103e+08	36886.148
std	287.37606	1.752611e+07	1.411084e+10	34151.352254	1.362782e+06	1232.244746	1944.386561	4.163782e+08	71858.724
min	1.00000	1.230000e+07	0.000000e+00	0.000000	1.000000e+00	1.000000	1.000000	1.000000e+00	0.000
25%	249.50000	1.450000e+07	4.288145e+09	194.500000	3.230000e+02	11.000000	27.000000	2.013750e+07	2700.000
50%	498.00000	1.770000e+07	7.760820e+09	729.000000	9.155000e+02	51.000000	65.500000	6.408500e+07	13300.000
75%	746.50000	2.460000e+07	1.355470e+10	2667.500000	3.584500e+03	123.000000	139.750000	1.688265e+08	37900.000
max	995.00000	2.450000e+08	2.280000e+11	301308.000000	4.057944e+06	7741.000000	7741.000000	6.589000e+09	850900.000

8 rows × 21 columns

In [5]:
```
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    df[col].fillna(df[col].mean(), inplace=True)

for col in df.select_dtypes(include=['object']).columns:
    df[col].fillna(df[col].mode()[0], inplace=True)

df.isnull().sum()
```

Out[5]:
```
rank                                  0
Youtuber                              0
subscribers                           0
video views                           0
category                              0
Title                                 0
uploads                               0
Country                               0
Abbreviation                          0
channel_type                          0
video_views_rank                      0
country_rank                          0
channel_type_rank                     0
video_views_for_the_last_30_days      0
lowest_monthly_earnings               0
highest_monthly_earnings              0
lowest_yearly_earnings                0
highest_yearly_earnings               0
subscribers_for_last_30_days          0
created_year                          0
created_month                         0
created_date                          0
Gross tertiary education enrollment (%)    0
Population                            0
Unemployment rate                     0
Urban_population                      0
Latitude                              0
Longitude                             0
dtype: int64
```

In [6]:
```
print(df.duplicated().any())
print(df.shape)
```

```
False
(995, 28)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['rank', 'Youtuber', 'subscribers', 'video views', 'category', 'Title',
       'uploads', 'Country', 'Abbreviation', 'channel_type',
       'video_views_rank', 'country_rank', 'channel_type_rank',
       'video_views_for_the_last_30_days', 'lowest_monthly_earnings',
       'highest_monthly_earnings', 'lowest_yearly_earnings',
       'highest_yearly_earnings', 'subscribers_for_last_30_days',
       'created_year', 'created_month', 'created_date',
       'Gross tertiary education enrollment (%)', 'Population',
       'Unemployment rate', 'Urban_population', 'Latitude', 'Longitude'],
      dtype='object')
```

In [8]:
```
for dtype in ['object', 'float', 'int']:
    print(f'Columns of {dtype} type:')
    print(df.select_dtypes(include=[dtype]).columns.tolist())
    print()
```

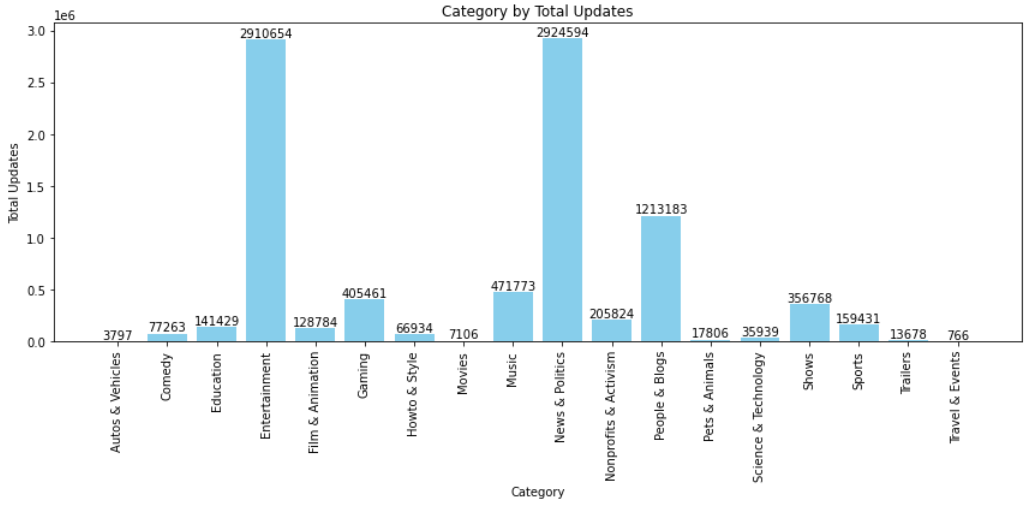
```
Columns of object type:
['Youtuber', 'category', 'Title', 'Country', 'Abbreviation', 'channel_type', 'created_month']

Columns of float type:
['video views', 'video_views_rank', 'country_rank', 'channel_type_rank', 'video_views_for_the_last_30_days', 'lowest_monthly_earnings',
'highest_monthly_earnings', 'lowest_yearly_earnings', 'highest_yearly_earnings', 'subscribers_for_last_30_days', 'created_year', 'create
d_date', 'Gross tertiary education enrollment (%)', 'Population', 'Unemployment rate', 'Urban_population', 'Latitude', 'Longitude']

Columns of int type:
['rank', 'subscribers', 'uploads']
```

Data Visualization

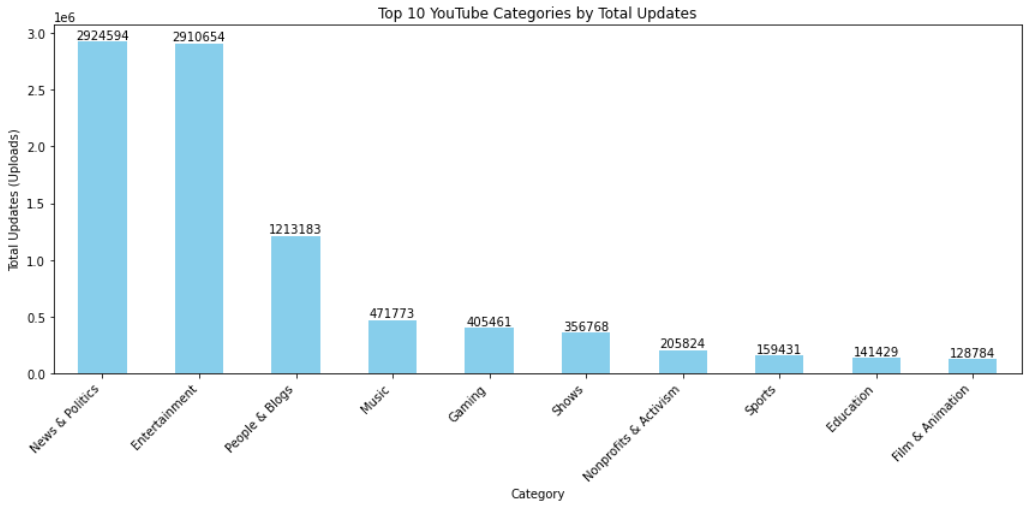
In [9]:
```
df['category'].unique()
```

Out[9]:
```
array(['Music', 'Film & Animation', 'Entertainment', 'Education', 'Shows',
       'People & Blogs', 'Gaming', 'Sports', 'Howto & Style',
       'News & Politics', 'Comedy', 'Trailers', 'Nonprofits & Activism',
       'Science & Technology', 'Movies', 'Pets & Animals',
       'Autos & Vehicles', 'Travel & Events'], dtype=object)
```

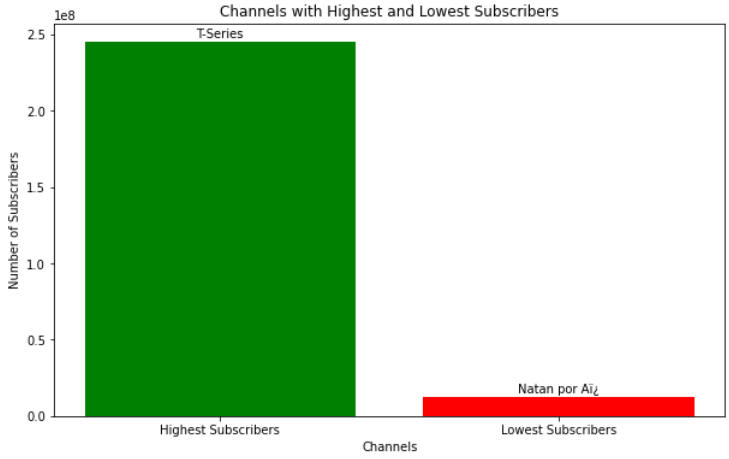
In [10]:
```
df['category'] = df['category'].astype(str)
category_total_uploads = df.groupby('category')['uploads'].sum().reset_index()
categories = category_total_uploads['category']
values = category_total_uploads['uploads']

plt.figure(figsize=(12, 6))
bars = plt.bar(categories, values, color='skyblue')
plt.title("Category by Total Updates")
plt.xlabel("Category")
plt.ylabel("Total Updates")
plt.xticks(rotation=90)

for bar, value in zip(bars, values):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 5, str(value), ha='center', va='bottom')

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



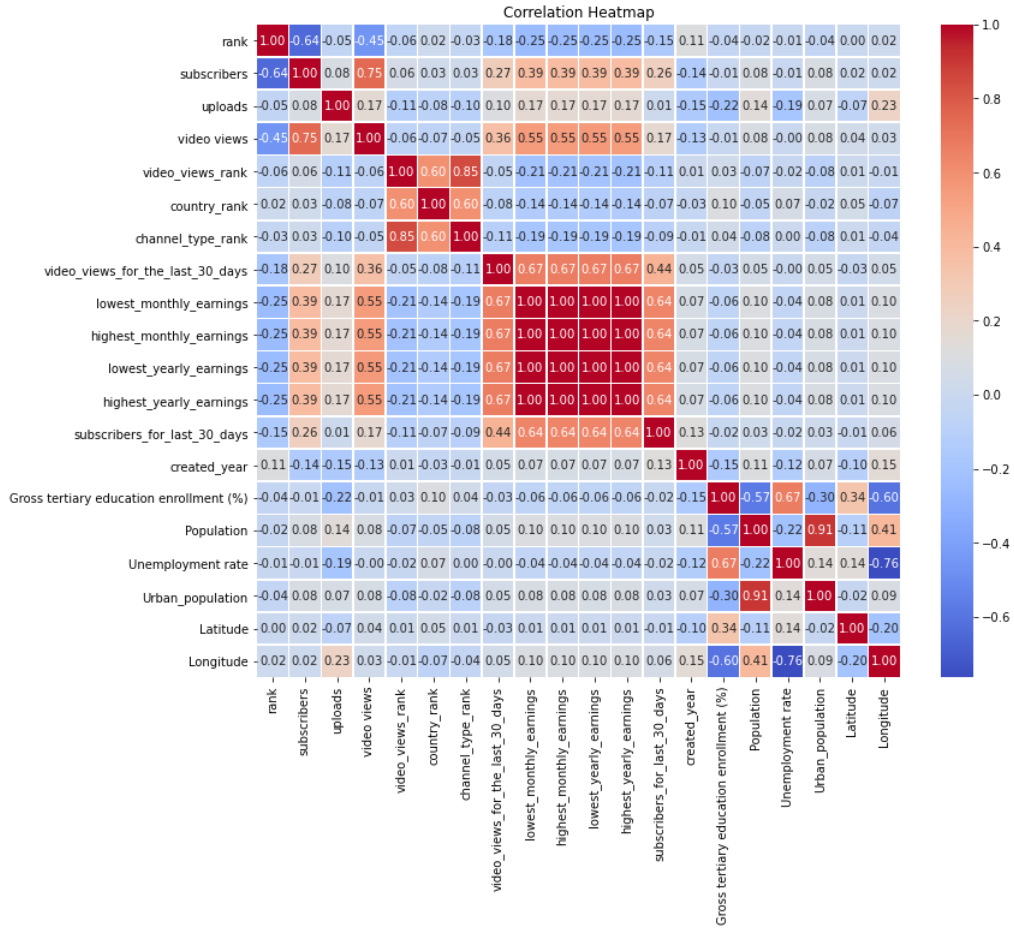
In [11]:
```
category_updates = df.groupby('category')['uploads'].sum().nlargest(10)
plt.figure(figsize=(12, 6))
bars = category_updates.plot(kind='bar', color='skyblue')
plt.xlabel('Category')
plt.ylabel('Total Updates (Uploads)')
plt.title('Top 10 YouTube Categories by Total Updates')
plt.xticks(rotation=45, ha='right')

for idx, value in enumerate(category_updates):
    plt.text(idx, value, str(int(value)), ha='center', va='bottom', fontsize=10)

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



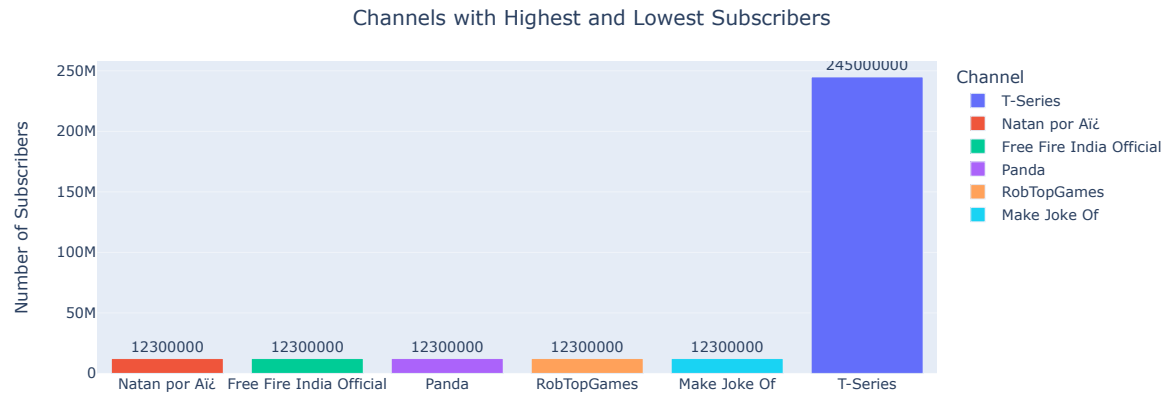
In [12]:
```
highest_subscribers = df[df['subscribers'] == df['subscribers'].max()]
lowest_subscribers = df[df['subscribers'] == df['subscribers'].min()]
highest_subscribers_channel_name = highest_subscribers['Youtuber'].values[0]
lowest_subscribers_channel_name = lowest_subscribers['Youtuber'].values[0]

plt.figure(figsize=(10, 6))
plt.bar(['Highest Subscribers', 'Lowest Subscribers'],[highest_subscribers['subscribers'].values[0],
        lowest_subscribers['subscribers'].values[0]], color=['green', 'red'])
plt.xlabel('Channels')
plt.ylabel('Number of Subscribers')
plt.title('Channels with Highest and Lowest Subscribers')
plt.text(0,highest_subscribers['subscribers'].values[0] + 1000000, f'{highest_subscribers_channel_name}', ha='center', va='bottom', fonts
plt.text(1,lowest_subscribers['subscribers'].values[0] + 1000000, f'{lowest_subscribers_channel_name}', ha='center', va='bottom', fontsiz
plt.show()
```



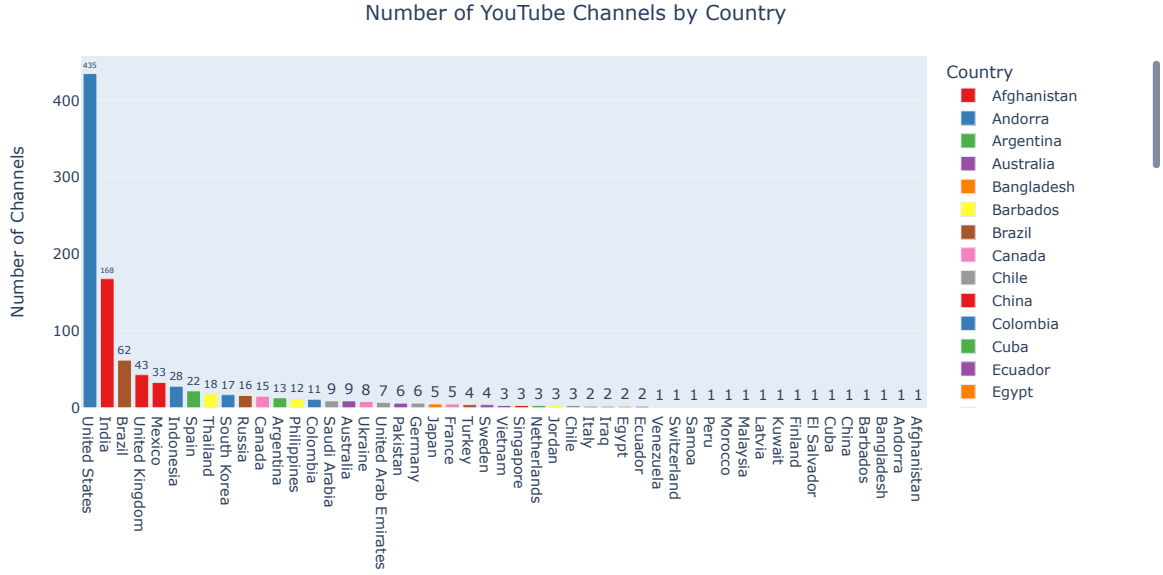
In [13]:
```
columns = ['rank','subscribers', 'uploads', 'video views', 'video_views_rank', 'country_rank','channel_type_rank',
           'video_views_for_the_last_30_days', 'lowest_monthly_earnings','highest_monthly_earnings', 'lowest_yearly_earnings',
           'highest_yearly_earnings','subscribers_for_last_30_days', 'created_year', 'Gross tertiary education enrollment (%)',
            'Population', 'Unemployment rate', 'Urban_population', 'Latitude', 'Longitude']

correlation_matrix = df[columns].corr()

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



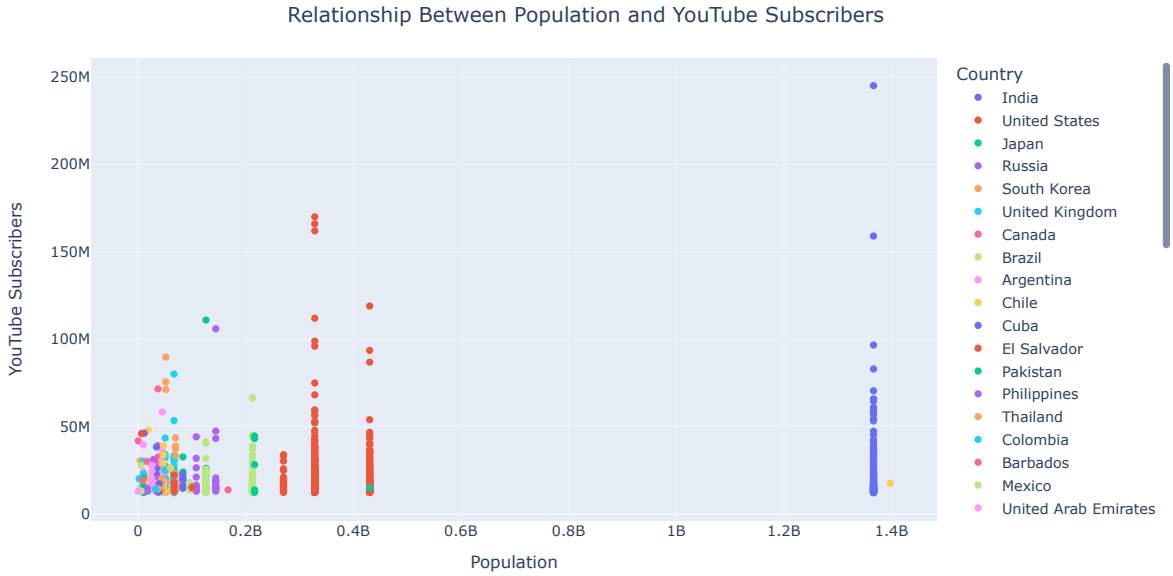
In [14]:
```
highest_subscribers = df[df['subscribers'] == df['subscribers'].max()]
lowest_subscribers = df[df['subscribers'] == df['subscribers'].min()]
channels = pd.concat([highest_subscribers, lowest_subscribers])
fig = px.bar(channels, x='Youtuber', y='subscribers', color='Youtuber',
             labels={'Youtuber': 'Channel', 'subscribers': 'Subscribers'}, text='subscribers', height=400)
fig.update_layout(title_text='Channels with Highest and Lowest Subscribers', title_x=0.5)
fig.update_xaxes(categoryorder='total ascending', title_text='')
fig.update_yaxes(title_text='Number of Subscribers')
fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.show()
```



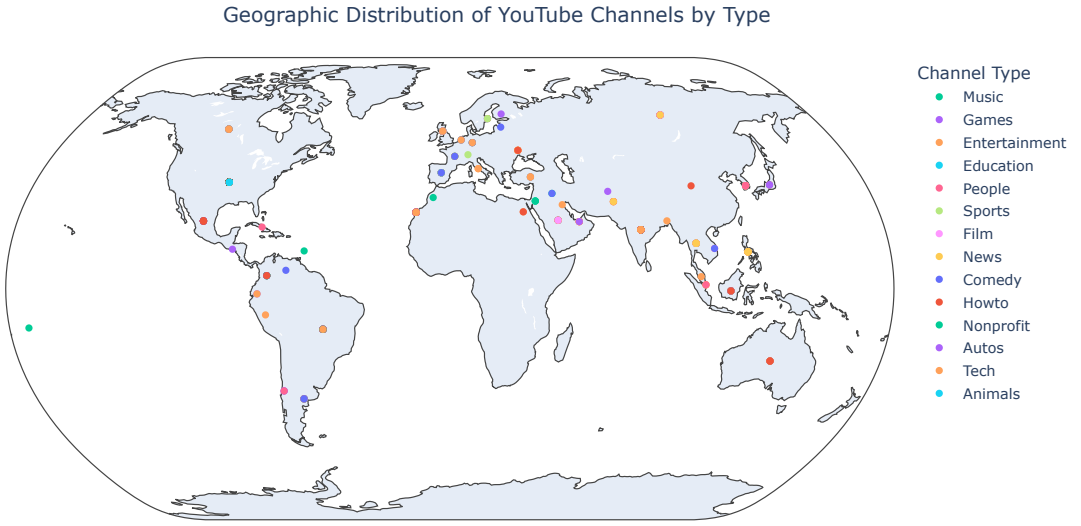
In [15]:
```
channel_counts = df['Youtuber'].groupby(df['Country']).count().reset_index()
color_scale = px.colors.qualitative.Set1[:len(channel_counts)]
fig = px.bar(channel_counts, x='Country', y='Youtuber', text='Youtuber', labels={'Country': 'Country', 'Youtuber': 'Number of Channels'},
             color='Country', color_discrete_sequence=color_scale)
fig.update_layout(title_text='Number of YouTube Channels by Country', title_x=0.5)
fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.update_xaxes(categoryorder='total descending', title_text='')
fig.update_yaxes(title_text='Number of Channels')
fig.show()
```



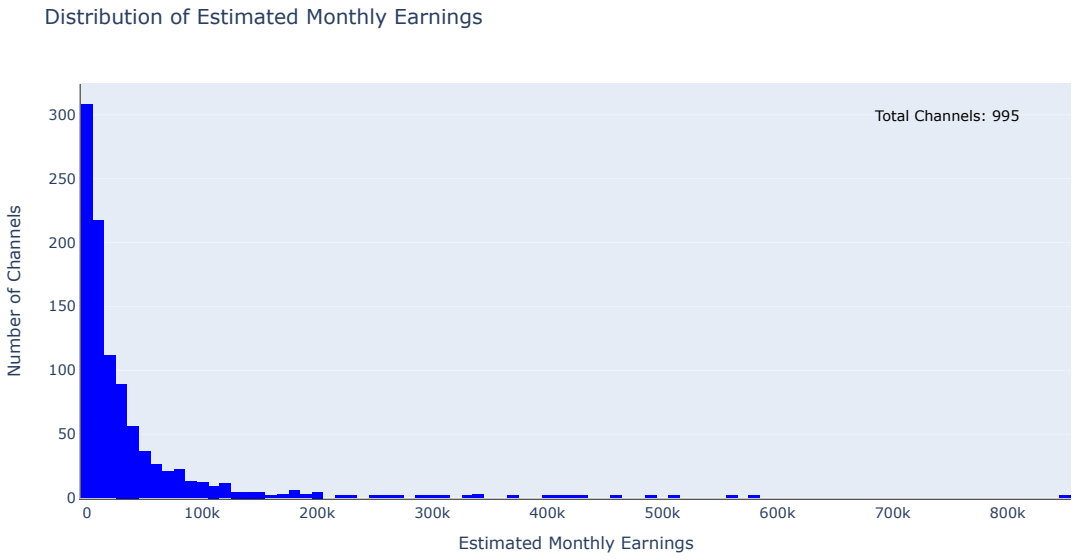
In [16]:
```
fig = px.scatter(df, x='Population', y='subscribers', color='Country',
                 labels={'Population': 'Population', 'subscribers': 'YouTube Subscribers'})
fig.update_layout(title_text='Relationship Between Population and YouTube Subscribers', title_x=0.5)
fig.update_layout(xaxis_title="Population", yaxis_title="YouTube Subscribers", legend_title="Country", showlegend=True)
fig.show()
```

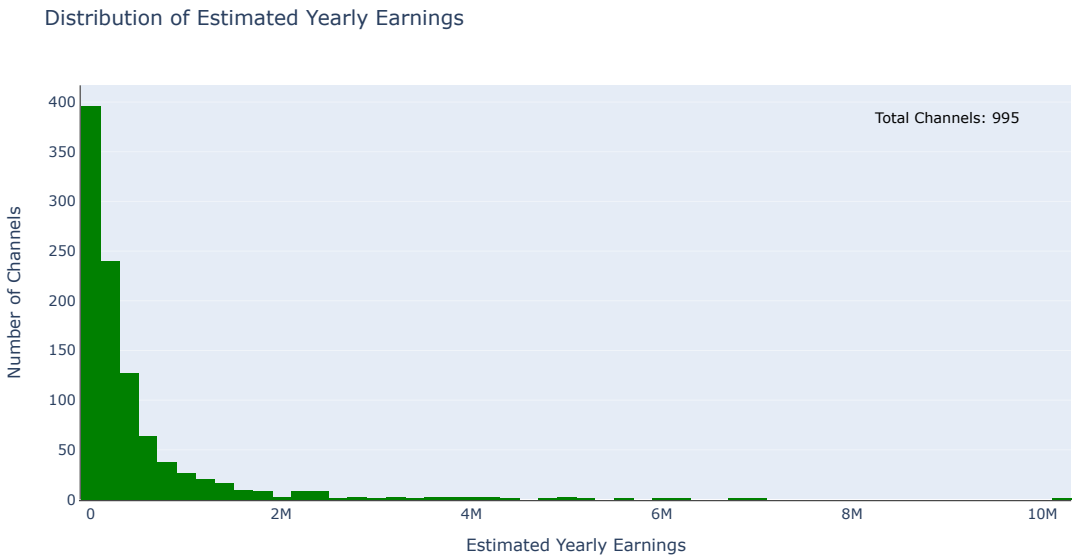


In [17]:
```
fig = px.scatter_geo(df, lat='Latitude', lon='Longitude', color='channel_type', hover_name='Country',
                     labels={'channel_type': 'Channel Type'}, color_discrete_map={'individual': 'blue', 'brand': 'red'})
fig.update_layout(title_text='Geographic Distribution of YouTube Channels by Type', title_x=0.5)
fig.update_geos(projection_type="natural earth", showcoastlines=True)
fig.update_layout(legend_title_text='Channel Type')
fig.show()
```



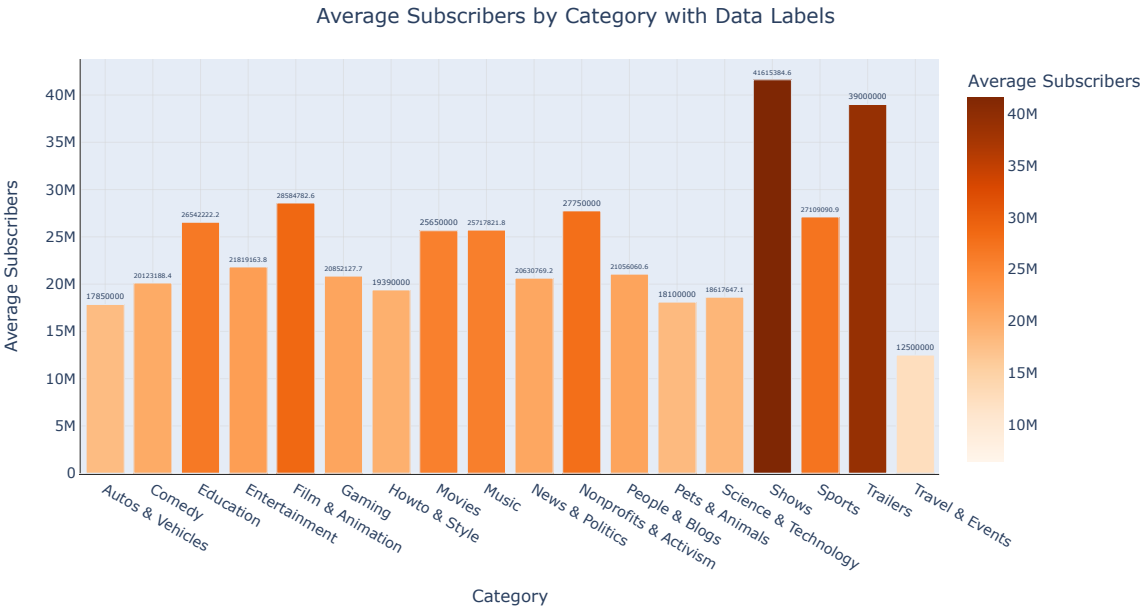
In [18]:
```
channels_monthly = len(df)
channels_yearly = len(df)

fig_monthly = px.histogram(df, x='lowest_monthly_earnings', title='Distribution of Estimated Monthly Earnings', color_discrete_sequence=[
fig_yearly = px.histogram(df, x='lowest_yearly_earnings', title='Distribution of Estimated Yearly Earnings', color_discrete_sequence=['gr

fig_monthly.update_xaxes(title='Estimated Monthly Earnings', showline=True, linewidth=2, linecolor='black')
fig_monthly.update_yaxes(title='Number of Channels', showline=True, linewidth=2, linecolor='black')
fig_monthly.update_layout(showlegend=False)
fig_yearly.update_xaxes(title='Estimated Yearly Earnings', showline=True, linewidth=2, linecolor='black')
fig_yearly.update_yaxes(title='Number of Channels', showline=True, linewidth=2, linecolor='black')
fig_yearly.update_layout(showlegend=False)

fig_monthly.add_annotation(
    text=f'Total Channels: {channels_monthly}',
    xref='paper', yref='paper',
    x=0.95, y=0.95,
    showarrow=False,
    font=dict(size=12, color='black')
)
fig_yearly.add_annotation(
    text=f'Total Channels: {channels_yearly}',
    xref='paper', yref='paper',
    x=0.95, y=0.95,
    showarrow=False,
    font=dict(size=12, color='black')
)
fig_monthly.show()
fig_yearly.show()
```



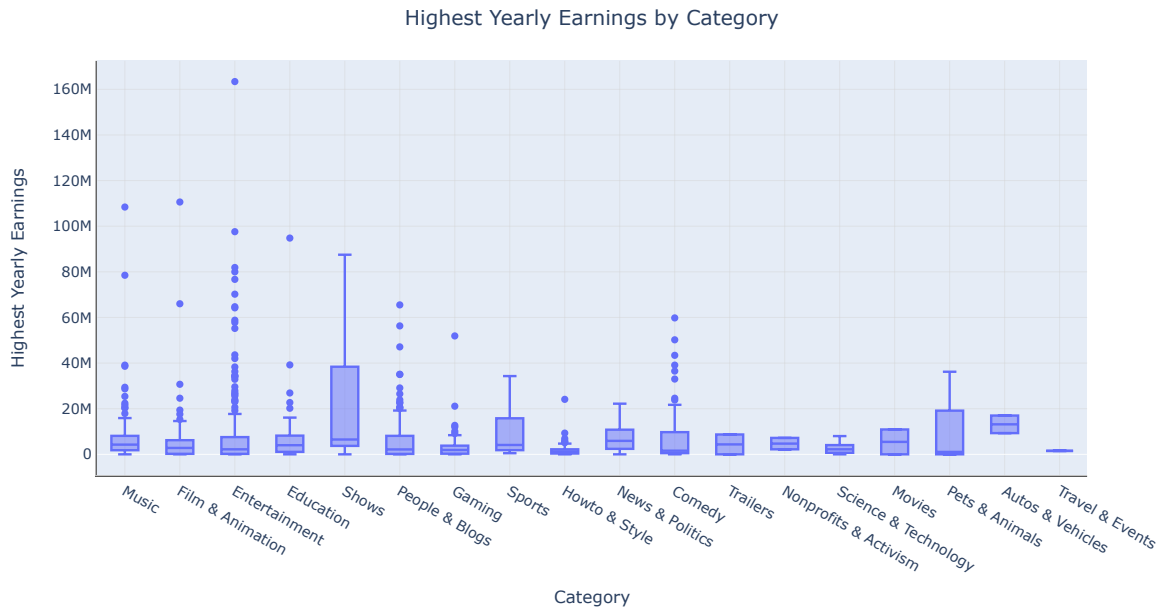


In [19]:
```
fig = px.scatter(df, x='subscribers', y='lowest_yearly_earnings',
                 labels={'subscribers': 'Subscribers', 'lowest_yearly_earnings': 'Yearly Earnings'}, trendline='ols')
fig.update_layout(title_text='Relationship Between Subscribers and Yearly Earnings', title_x=0.5)
fig.update_traces(marker=dict(size=10, opacity=0.7), selector=dict(mode='markers+text'))
fig.update_layout(showlegend=False)
fig.show()
```

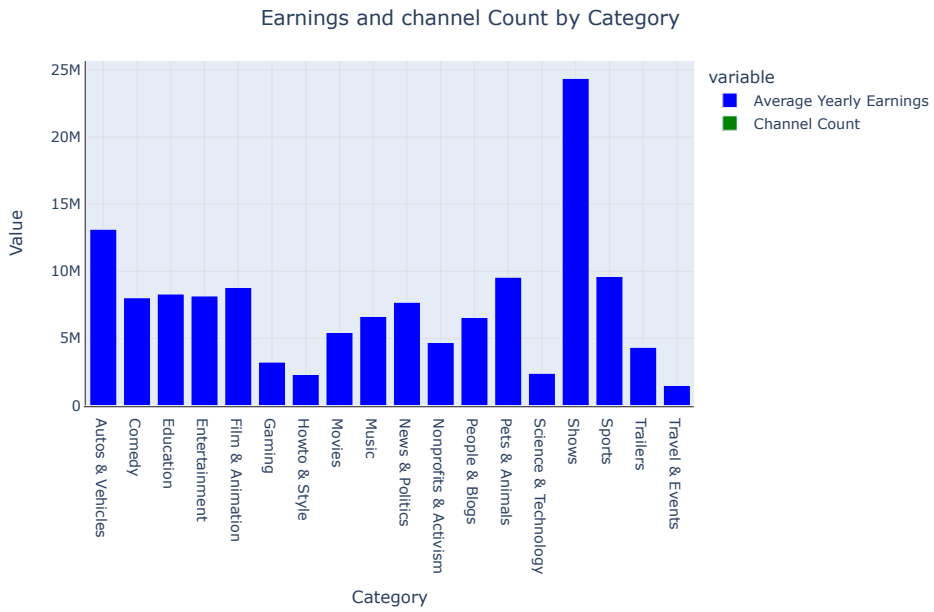
In [20]:
```
average_subscribers= df.groupby('category')['subscribers'].mean().reset_index()

fig = px.bar(average_subscribers, x='category', y='subscribers',
             labels={'category': 'Category', 'subscribers': 'Average Subscribers'},
             color='subscribers',
             color_continuous_scale='oranges',
             color_continuous_midpoint=average_subscribers['subscribers'].mean())

fig.update_layout(title_text='Average Subscribers by Category with Data Labels', title_x=0.5)
fig.update_traces(text=average_subscribers['subscribers'].round(1),
                  textposition='outside')

fig.update_xaxes(showline=True, linewidth=2, linecolor='black', showgrid=True, gridcolor='lightgray')
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', showgrid=True, gridcolor='lightgray')
fig.show()
```



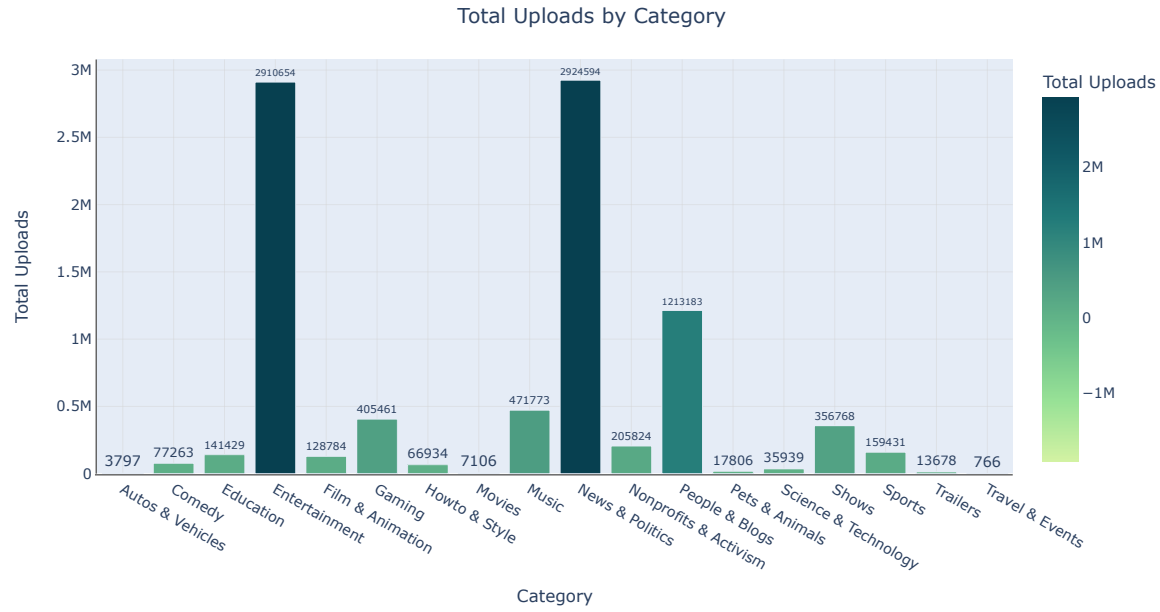
In [21]:
```
fig = px.box(df, x='category', y='highest_yearly_earnings',
             labels={'category': 'Category', 'highest_yearly_earnings': 'Highest Yearly Earnings'})
fig.update_layout(title_text='Highest Yearly Earnings by Category', title_x=0.5)

fig.update_xaxes(showline=True, linewidth=2, linecolor='black', showgrid=True, gridcolor='lightgray')
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', showgrid=True, gridcolor='lightgray')
fig.show()
```



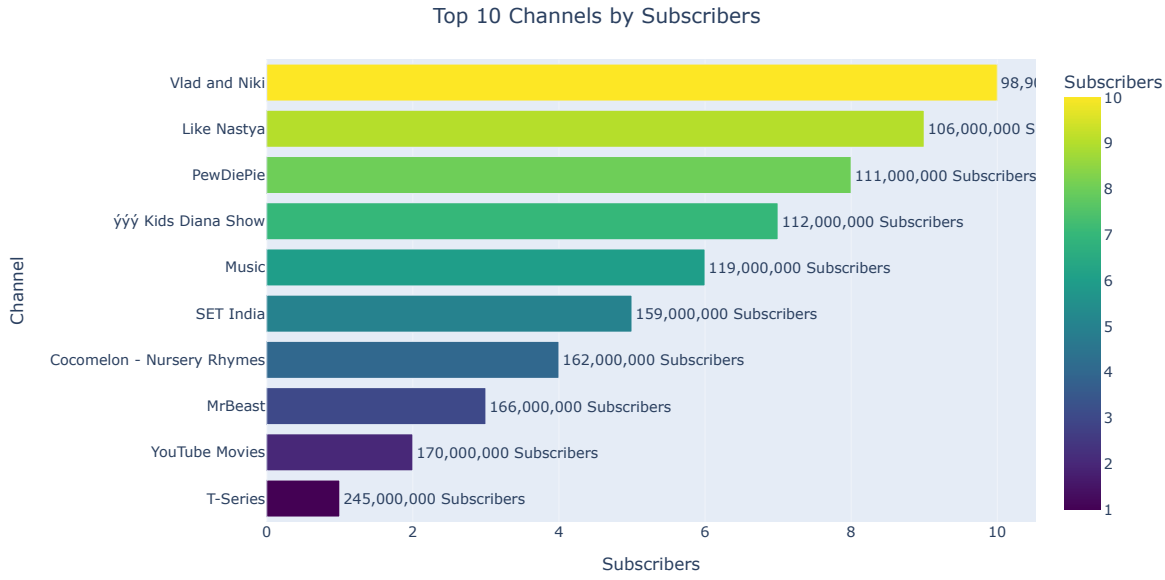
In [22]:
```
category = df.groupby('category').agg({'highest_yearly_earnings': 'mean', 'Youtuber': 'count'}).reset_index()
category.rename(columns={'highest_yearly_earnings': 'Average Yearly Earnings', 'Youtuber': 'Channel Count'}, inplace=True)
fig = px.bar(category, x='category', y=['Average Yearly Earnings', 'Channel Count'],
             labels={'category': 'Category', 'value': 'Value'},
             color_discrete_sequence=['blue', 'green'], width=800)
fig.update_layout(title_text='Earnings and channel Count by Category', title_x=0.5)

fig.update_xaxes(showline=True, linewidth=2, linecolor='black', showgrid=True, gridcolor='lightgray')
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', showgrid=True, gridcolor='lightgray')
fig.show()
```



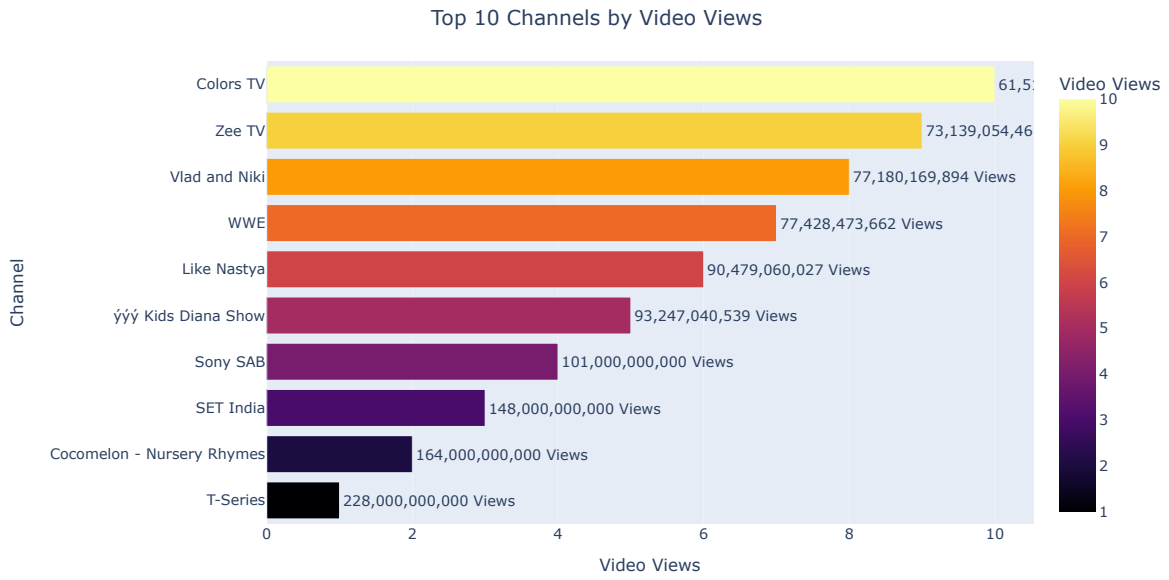
In [23]:
```
uploads_by_category = df.groupby('category')['uploads'].sum().reset_index()

fig = px.bar(uploads_by_category, x='category', y='uploads',
             labels={'category': 'Category', 'uploads': 'Total Uploads'}, color='uploads', color_continuous_scale='emrld',
             color_continuous_midpoint=uploads_by_category['uploads'].mean())
fig.update_layout(title_text='Total Uploads by Category', title_x=0.5)
fig.update_traces(text=uploads_by_category['uploads'].round(1).astype(str),
                  textposition='outside')
fig.update_xaxes(showline=True, linewidth=2, linecolor='black', showgrid=True, gridcolor='lightgray')
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', showgrid=True, gridcolor='lightgray')
fig.show()
```



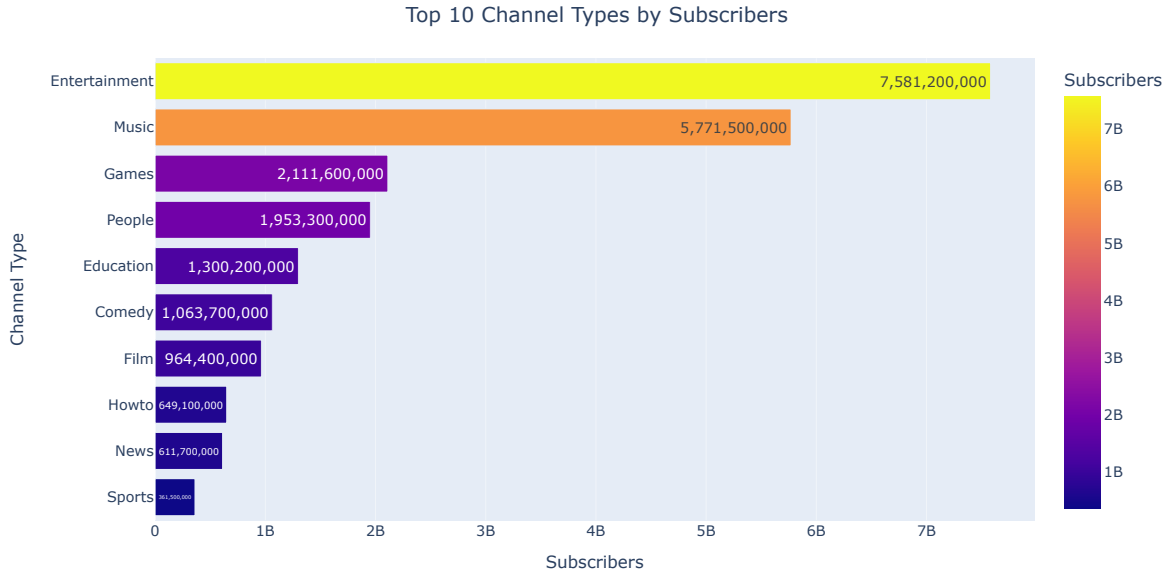
In [24]:
```
df['subscribers_rank'] = df['subscribers'].rank(ascending=False, method='min')
df = df.sort_values(by='subscribers_rank')

top_subscribers = df[['Youtuber', 'subscribers', 'subscribers_rank']].head(10)
fig_subscribers = px.bar(top_subscribers, x='subscribers_rank', y='Youtuber',
                         labels={'Youtuber': 'Channel', 'subscribers_rank': 'Subscribers'},
                         color='subscribers_rank', color_continuous_scale='Viridis')
fig_subscribers.update_layout(title_text='Top 10 Channels by Subscribers', title_x=0.5)
fig_subscribers.update_layout(yaxis_categoryorder='total ascending', xaxis_title='Subscribers')
fig_subscribers.update_traces(text=top_subscribers['subscribers'].apply(lambda x: f'{int(x):,} Subscribers'),
                              textposition='outside', textfont_size=12)
fig_subscribers.show()
```



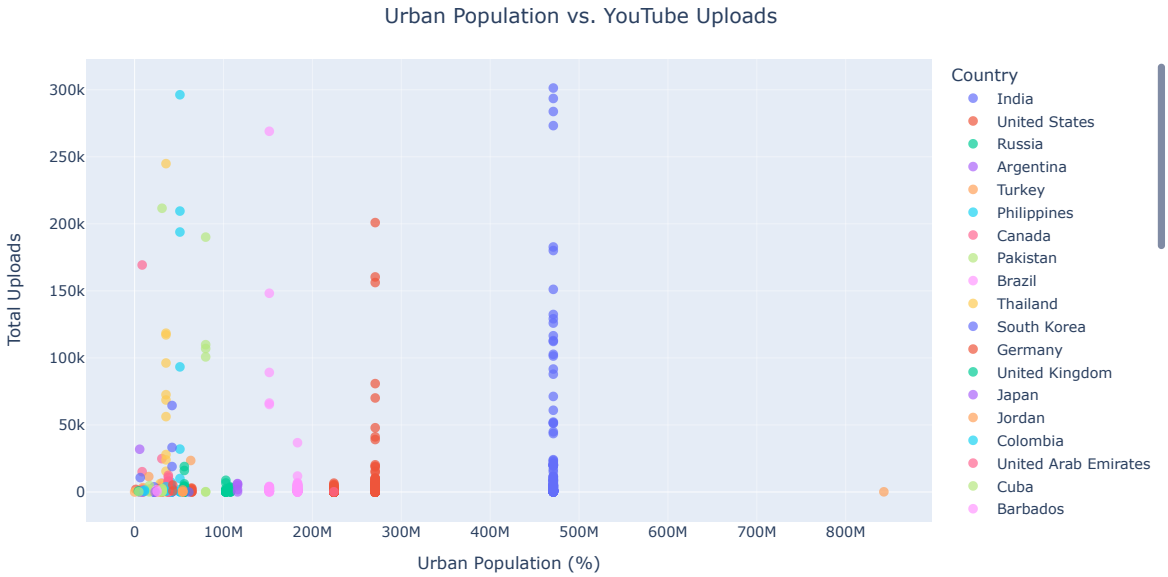
In [25]:
```
df['video_views_rank'] = df['video views'].rank(ascending=False, method='min')
df = df.sort_values(by='video_views_rank')

top_video_views = df[['Youtuber', 'video views', 'video_views_rank']].head(10)
fig_video_views = px.bar(top_video_views, x='video_views_rank', y='Youtuber',
                         labels={'Youtuber': 'Channel', 'video_views_rank': 'Video Views'},
                         color='video_views_rank', color_continuous_scale='Inferno')
fig_video_views.update_layout(title_text='Top 10 Channels by Video Views', title_x=0.5)

fig_video_views.update_layout(yaxis_categoryorder='total ascending', xaxis_title='Video Views')
fig_video_views.update_traces(text=top_video_views['video views'].apply(lambda x: f'{int(x):,} Views'),
                              textposition='outside', textfont_size=12)
fig_video_views.show()
```



In [26]:
```
channel_type_subscribers = df.groupby('channel_type')['subscribers'].sum().reset_index()
channel_type_subscribers = channel_type_subscribers.sort_values(by='subscribers', ascending=False)

top_10_channel_types = channel_type_subscribers.head(10)
fig_channel_type_subscribers = px.bar(top_10_channel_types, x='subscribers', y='channel_type',
                                      labels={'channel_type': 'Channel Type', 'subscribers': 'Subscribers'},
                                      color='subscribers', color_continuous_scale='Plasma',
                                      orientation='h')
fig_channel_type_subscribers.update_layout(title_text='Top 10 Channel Types by Subscribers', title_x=0.5)
fig_channel_type_subscribers.update_yaxes(categoryorder='total ascending', title='Channel Type')
fig_channel_type_subscribers.update_xaxes(title='Subscribers')
fig_channel_type_subscribers.update_traces(text=top_10_channel_types['subscribers'].apply(lambda x: f'{int(x):,}'),
                                           textposition='inside', textfont_size=12)
fig_channel_type_subscribers.show()
```

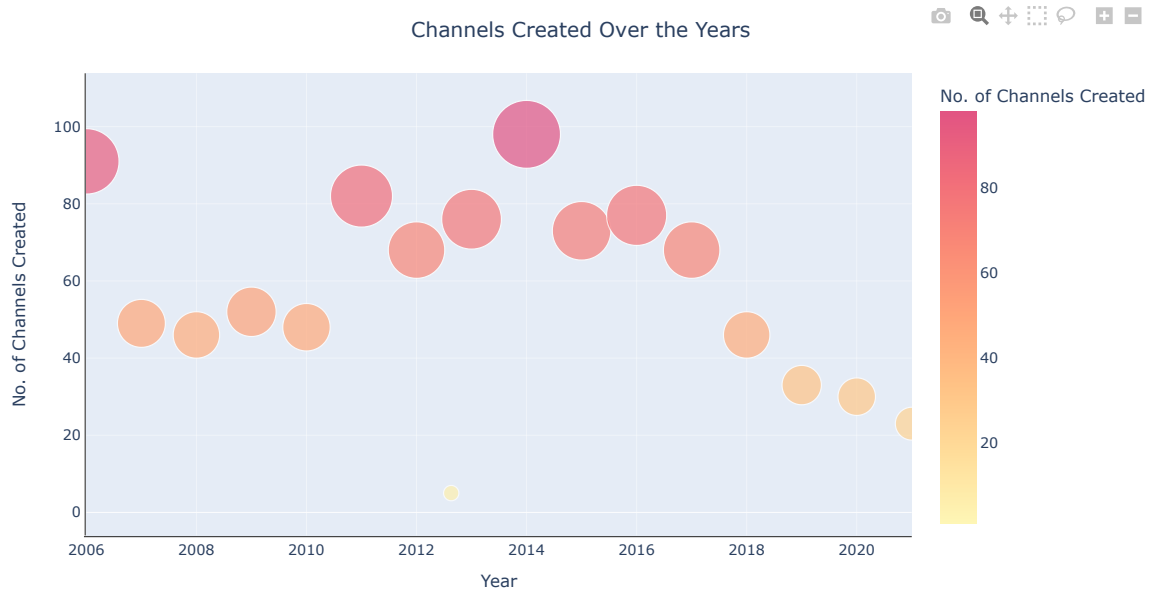


In [27]:
```
fig = px.scatter(df, x='Urban_population', y='uploads',
                 labels={'Urban_population': 'Urban Population (%)', 'uploads': 'Total Uploads'},
                 color='Country', hover_name='Country')
fig.update_layout(title_text='Urban Population vs. YouTube Uploads', title_x=0.5)
fig.update_xaxes(title='Urban Population (%)')
fig.update_yaxes(title='Total Uploads')
fig.update_layout(legend_title_text='Country')
fig.update_traces(marker=dict(size=8, opacity=0.7))
fig.show()

correlation_coefficient = df['Urban_population'].corr(df['uploads'])
print(f'Correlation Coefficient: {correlation_coefficient:.2f}')
```




```
Correlation Coefficient: 0.07
```

In [28]:
```
yearly_counts = df.groupby('created_year').size().reset_index(name='count')
fig = px.scatter(yearly_counts, x='created_year', y='count',
                 labels={'created_year': 'Year', 'count': 'No. of Channels Created'},
                 color=yearly_counts['count'], size=yearly_counts['count'], size_max=40,
                 color_continuous_scale='pinkyl' )

fig.update_layout(title_text='Channels Created Over the Years', title_x=0.5)
fig.update_xaxes(title='Year', showline=True, linewidth=2, linecolor='black')
fig.update_yaxes(title='No. of Channels Created', showline=True, linewidth=2, linecolor='black')
fig.update_layout(xaxis_range=[2006, 2021])

fig.show()
```



In [ ]: