In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import plotly.express as ple
```

1ST

In [4]:
```
cdata=pd.read_csv('countries of the world.csv')

cdata.head()
```

Out[4]:

	Country	Region
0	Afghanistan	ASIA (EX. NEAR EAST)
1	Albania	EASTERN EUROPE
2	Algeria	NORTHERN AFRICA
3	American Samoa	OCEANIA
4	Andorra	WESTERN EUROPE

In [5]:
```
cdata.count()
```

Out[5]:
```
Country     227
Region      227
dtype: int64
```

In [6]:
```
cdata.isna()
```

Out[6]:

	Country	Region
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
222	False	False
223	False	False
224	False	False
225	False	False
226	False	False

227 rows × 2 columns

In [7]:
```
cdata.isna().sum()
```

```
Out[7]: Country    0
        Region     0
        dtype: int64
```

```
In [8]: cdata.duplicated().sum()
```

```
Out[8]: 0
```

```
In [9]: cdatatxt=pd.read_csv("countries of the world.csv")
        cdatatxt.head()
```

Out[9]:

	Country	Region
0	Afghanistan	ASIA (EX. NEAR EAST)
1	Albania	EASTERN EUROPE
2	Algeria	NORTHERN AFRICA
3	American Samoa	OCEANIA
4	Andorra	WESTERN EUROPE

```
In [10]: cdatatxt.isna().sum()
```

```
Out[10]: Country    0
         Region     0
         dtype: int64
```

```
In [11]: cdatatxt.duplicated().sum()
```

```
Out[11]: 0
```

2nd

```
In [12]: customerdata=pd.read_excel("Customer Call List.xlsx")
         customerdata.head()
```

Out[12]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No
1	1002	Abed	Nadir	123/643/9775	93 West Main Street	No	Yes
2	1003	Walter	/White	7066950392	298 Drugs Driveway	N	NaN
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y
4	1005	Jon	Snow	876|678|3469	123 Dragons Road	Y	No

```
In [13]: customerdata.count()
```

Out[13]:
```
CustomerID            21
First_Name            21
Last_Name             20
Phone_Number          19
Address               21
Paying Customer       21
Do_Not_Contact        17
Not_Useful_Column     21
dtype: int64
```

In [14]:
```
customerdata.describe()
```

Out[14]:

	CustomerID
count	21.000000
mean	1010.952381
std	6.127611
min	1001.000000
25%	1006.000000
50%	1011.000000
75%	1016.000000
max	1020.000000

In [15]:
```
customerdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   CustomerID         21 non-null     int64 
 1   First_Name         21 non-null     object
 2   Last_Name          20 non-null     object
 3   Phone_Number       19 non-null     object
 4   Address            21 non-null     object
 5   Paying Customer    21 non-null     object
 6   Do_Not_Contact     17 non-null     object
 7   Not_Useful_Column  21 non-null     bool  
dtypes: bool(1), int64(1), object(6)
memory usage: 1.3+ KB
```

In [16]:
```
customerdata.isna().sum()
```

Out[16]:
```
CustomerID           0
First_Name           0
Last_Name            1
Phone_Number         2
Address              0
Paying Customer      0
Do_Not_Contact       4
Not_Useful_Column    0
dtype: int64
```

In [17]:
```
customerdata.dropna(inplace=True)
```

In [18]:
```
customerdata.isna().sum()
```

Out[18]:
```
CustomerID           0
First_Name           0
Last_Name            0
Phone_Number         0
Address              0
Paying Customer      0
Do_Not_Contact       0
Not_Useful_Column    0
dtype: int64
```

In [20]:
```
customerdata.duplicated().sum()
```

Out[20]:
```
1
```

In [23]:
```
dprows=customerdata[customerdata.duplicated()]
dprows
```

Out[23]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Nc
20	1020	Anakin	Skywalker	876|678|3469	910 Tatooine Road, Tatooine	Yes	N	

In [25]:
```
customerdata.drop_duplicates(inplace=True)
```

In [28]:
```
customerdata.duplicated().sum()
```

Out[28]:
```
0
```

In [32]:
```
customerdata.rename(columns={'Not_Useful_Column':'Not_Useful'},inplace=True)
```

In [33]:
```
customerdata
```

Out[33]:

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No
1	1002	Abed	Nadir	123/643/9775	93 West Main Street	No	Yes
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y
4	1005	Jon	Snow	876|678|3469	123 Dragons Road	Y	No
5	1006	Ron	Swanson	304-762-2467	768 City Parkway	Yes	Yes
7	1008	Sherlock	Holmes	876|678|3469	98 Clue Drive	N	No
9	1010	Peter	Parker	123-545-5421	25th Main Street, New York	Yes	No
12	1013	Don	Draper	123-543-2345	2039 Main Street	Yes	N
13	1014	Leslie	Knope	876|678|3469	343 City Parkway	Yes	No
14	1015	Toby	Flenderson_	304-762-2467	214 HR Avenue	N	No
15	1016	Ron	Weasley	123-545-5421	2395 Hogwarts Avenue	No	N
16	1017	Michael	Scott	123/643/9775	121 Paper Avenue, Pennsylvania	Yes	No
18	1019	Creed	Braton	N/a	N/a	N/a	Yes
19	1020	Anakin	Skywalker	876|678|3469	910 Tatooine Road, Tatooine	Yes	N

3rd

In [34]:
```python
fdata=pd.read_csv('Flavors.csv')
fdata
```

Out[34]:

	Flavor	Base Flavor	Liked	Flavor Rating	Texture Rating	Total Rating
0	Mint Chocolate Chip	Vanilla	Yes	10.0	8.0	18.0
1	Chocolate	Chocolate	Yes	8.8	7.6	16.6
2	Vanilla	Vanilla	No	4.7	5.0	9.7
3	Cookie Dough	Vanilla	Yes	6.9	6.5	13.4
4	Rocky Road	Chocolate	Yes	8.2	7.0	15.2
5	Pistachio	Vanilla	No	2.3	3.4	5.7
6	Cake Batter	Vanilla	Yes	6.5	6.0	12.5
7	Neapolitan	Vanilla	No	3.8	5.0	8.8
8	Chocolte Fudge Brownie	Chocolate	Yes	8.2	7.1	15.3

In [40]:
```
fdata.isna().sum()
```
Out[40]:
```
Flavor            0
Base Flavor       0
Liked             0
Flavor Rating     0
Texture Rating    0
Total Rating      0
dtype: int64
```

In [41]:
```
fdata.describe()
```
Out[41]:

	Flavor Rating	Texture Rating	Total Rating
count	9.0000	9.000000	9.000000
mean	6.6000	6.177778	12.800000
std	2.5387	1.478832	4.030509
min	2.3000	3.400000	5.700000
25%	4.7000	5.000000	9.700000
50%	6.9000	6.500000	13.400000
75%	8.2000	7.100000	15.300000
max	10.0000	8.000000	18.000000

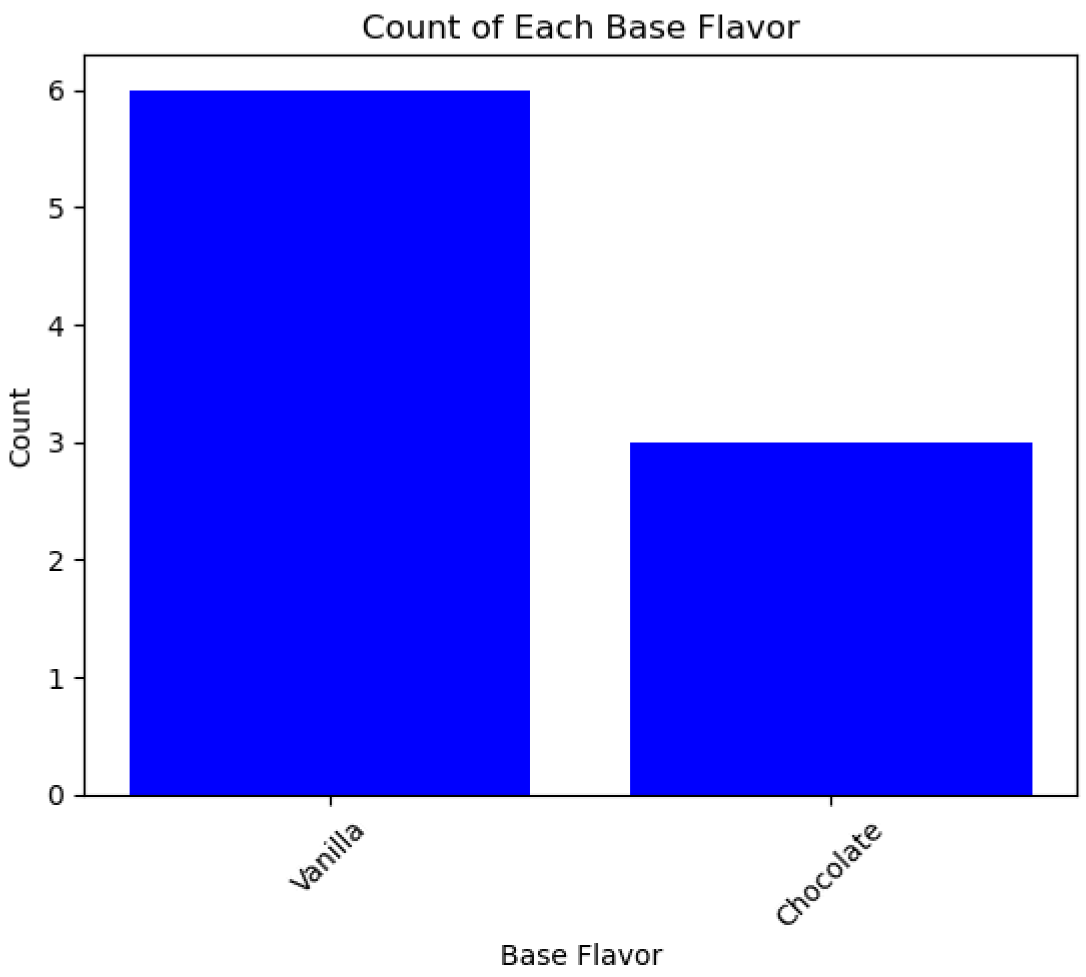
In [42]:
```
fdata.count()
```
Out[42]:
```
Flavor            9
Base Flavor       9
Liked             9
Flavor Rating     9
Texture Rating    9
Total Rating      9
dtype: int64
```

In [49]:
```
fdata['Base Flavor'].value_counts()
```
Out[49]:
```
Base Flavor
Vanilla      6
Chocolate    3
Name: count, dtype: int64
```

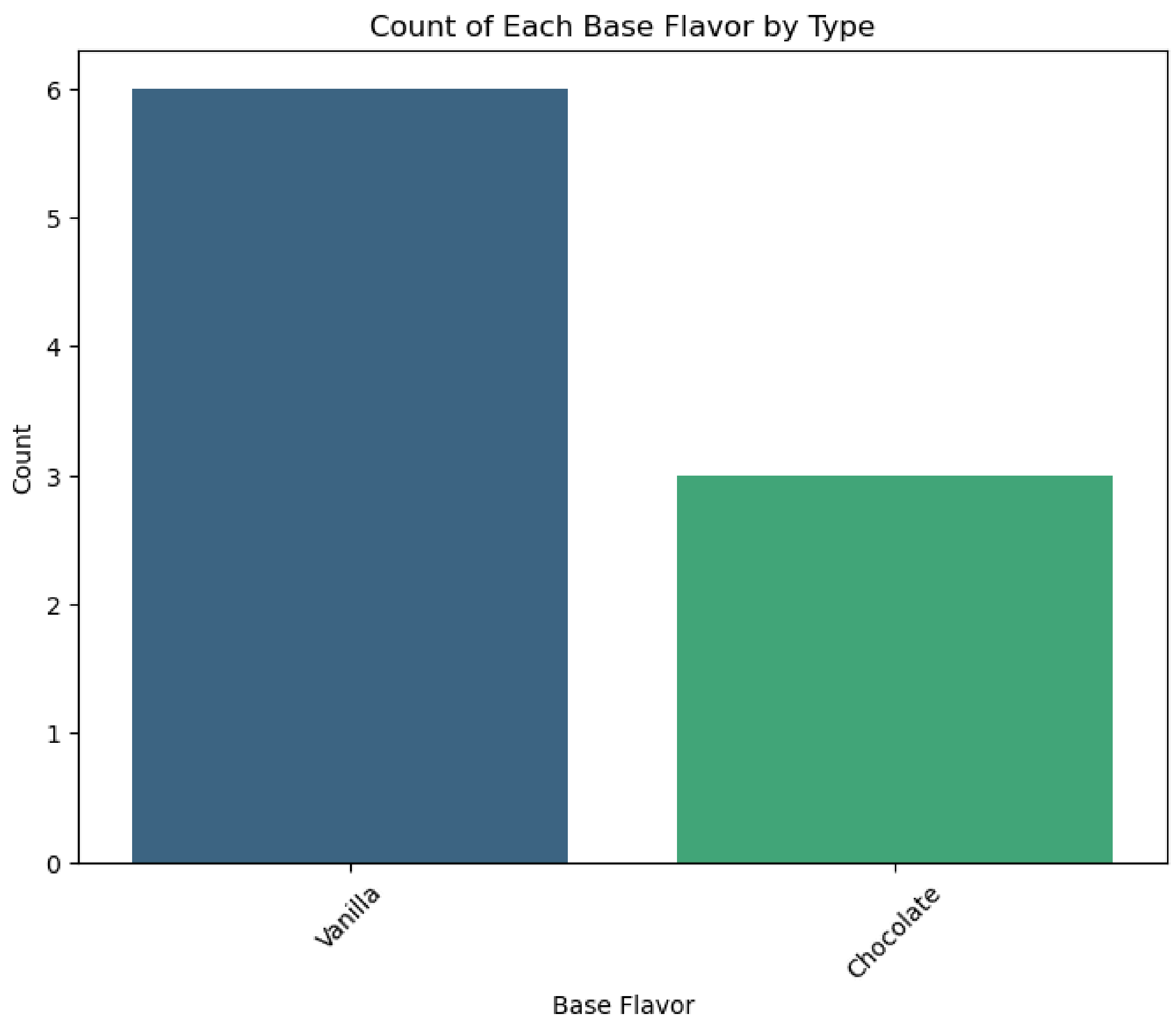
In [50]:
```
fdata['Base Flavor']
```

Out[50]:
```
0      Vanilla
1    Chocolate
2      Vanilla
3      Vanilla
4    Chocolate
5      Vanilla
6      Vanilla
7      Vanilla
8    Chocolate
Name: Base Flavor, dtype: object
```

In [59]:
```
flavor_counts = fdata['Base Flavor'].value_counts()

plt.bar(flavor_counts.index, flavor_counts.values,color='blue')
plt.xlabel('Base Flavor')
plt.ylabel('Count')
plt.title('Count of Each Base Flavor')
plt.xticks(rotation=45)
plt.show()
```



In [65]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=fdata, x='Base Flavor',  palette='viridis')
plt.xlabel('Base Flavor')
plt.ylabel('Count')
plt.title('Count of Each Base Flavor by Type')
plt.xticks(rotation=45)
plt.show()
```



4th

In [66]:
```
icedata=pd.read_csv("Ice Cream Ratings.csv")
icedata.head()
```

Out[66]:

	Date	Flavor Rating	Texture Rating	Overall Rating
0	1/1/2022	0.223090	0.040220	0.600129
1	1/2/2022	0.635886	0.938476	0.106264
2	1/3/2022	0.442323	0.044154	0.598112
3	1/4/2022	0.389128	0.549676	0.489353
4	1/5/2022	0.386887	0.519439	0.988280

In [68]:
```
icedata.count().sum()
```

Out[68]:
```
28
```

In [71]:
```
icedata.describe()
```

Out[71]:

	Flavor Rating	Texture Rating	Overall Rating
count	7.000000	7.000000	7.000000
mean	0.442328	0.372952	0.531445
std	0.249008	0.322952	0.334906
min	0.140995	0.040220	0.105147
25%	0.304989	0.118871	0.297808
50%	0.389128	0.325110	0.598112
75%	0.539104	0.534557	0.716478
max	0.877984	0.938476	0.988280

In [72]:
```
icedata.isnull().sum()
```
Out[72]:
```
Date              0
Flavor Rating     0
Texture Rating    0
Overall Rating    0
dtype: int64
```
In [74]:
```
icedata.duplicated().sum()
```
Out[74]:
```
0
```

5th

In [77]:
```
import json

with open('json_sample.json','r') as file:
    jdata=json.load(file)
```
In [78]:
```
print(jdata)
```

```
[{'12 Strong': {'Genre': 'Action', 'Gross': '$453,173', 'IMDB Metascore': '54', 'P
opcorn Score': 72, 'Rating': 'R', 'Tomato Score': 54}, 'A Fantastic Woman (Una Muj
er Fantástica)': {'popcornscore': 83, 'rating': 'R', 'tomatoscore': 90}, 'All The
Money In The World': {'popcornscore': 71, 'rating': 'R', 'tomatoscore': 77}, 'Bila
l: A New Breed Of Hero': {'popcornscore': 91, 'rating': 'PG13', 'tomatoscore': 5
7}, 'Call Me By Your Name': {'popcornscore': 87, 'rating': 'R', 'tomatoscore': 9
6}, 'Darkest Hour': {'popcornscore': 84, 'rating': 'PG13', 'tomatoscore': 86}, 'De
n Of Thieves': {'Genre': 'Action', 'Gross': '$491,898', 'IMDB Metascore': '49', 'P
opcorn Score': 69, 'Rating': 'R', 'Tomato Score': 40}, 'Ferdinand': {'popcornscor
e': 49, 'rating': 'PG', 'tomatoscore': 71}, 'Fifty Shades Freed': {'Genre': 'Dram
a', 'Gross': 'unknown', 'IMDB Metascore': '34', 'Popcorn Score': 'unknown', 'Ratin
g': 'unrated', 'Tomato Score': 'unkown'}, "Film Stars Don'T Die In Liverpool": {'p
opcornscore': 69, 'rating': 'R', 'tomatoscore': 78}, 'Forever My Girl': {'popcorns
core': 91, 'rating': 'PG', 'tomatoscore': 21}, 'Golden Exits': {'Genre': 'Drama',
'Gross': 'unknown', 'IMDB Metascore': '72', 'Popcorn Score': 'unknown', 'Rating':
'unrated', 'Tomato Score': 'unkown'}, 'Hostiles': {'Genre': 'Adventure', 'Gross':
'$548,886', 'IMDB Metascore': '65', 'Popcorn Score': 71, 'Rating': 'R', 'Tomato Sc
ore': 72}, 'I, Tonya': {'popcornscore': 89, 'rating': 'R', 'tomatoscore': 90}, 'In
sidious: The Last Key': {'popcornscore': 51, 'rating': 'PG13', 'tomatoscore': 32},
'Jumanji: Welcome To The Jungle': {'Genre': 'Action', 'Gross': '$760,867', 'IMDB M
etascore': '58', 'Popcorn Score': 89, 'Rating': 'PG13', 'Tomato Score': 76}, "Mary
And The Witch'S Flower": {'popcornscore': 78, 'rating': 'PG', 'tomatoscore': 84},
'Maze Runner: The Death Cure': {'Genre': 'Action', 'Gross': '$720,463', 'IMDB Meta
score': '51', 'Popcorn Score': 71, 'Rating': 'PG13', 'Tomato Score': 43}, "Molly'S
Game": {'popcornscore': 85, 'rating': 'R', 'tomatoscore': 82}, 'Paddington 2': {'G
enre': 'Animation', 'Gross': '$184,414', 'IMDB Metascore': '88', 'Popcorn Score':
89, 'Rating': 'PG', 'Tomato Score': 100}, 'Padmaavat': {'popcornscore': 62, 'ratin
g': 'NR', 'tomatoscore': 74}, 'Permission': {'Genre': 'Comedy', 'Gross': 'unknow
n', 'IMDB Metascore': '53', 'Popcorn Score': 'unknown', 'Rating': 'unrated', 'Toma
to Score': 'unkown'}, 'Peter Rabbit': {'Genre': 'Animation', 'Gross': 'unknown',
'IMDB Metascore': '56', 'Popcorn Score': 'unknown', 'Rating': 'unrated', 'Tomato S
core': 'unkown'}, 'Phantom Thread': {'popcornscore': 68, 'rating': 'R', 'tomatosco
re': 91}, 'Pitch Perfect 3': {'popcornscore': 52, 'rating': 'PG13', 'tomatoscore':
31}, 'Proud Mary': {'popcornscore': 56, 'rating': 'R', 'tomatoscore': 26}, 'Sanpo
Suru Shinryakusha': {'Genre': 'Drama', 'Gross': 'unknown', 'IMDB Metascore': '65',
'Popcorn Score': 'unknown', 'Rating': 'unrated', 'Tomato Score': 'unkown'}, 'Star
Wars: The Last Jedi': {'popcornscore': 48, 'rating': 'PG13', 'tomatoscore': 91},
'The 15:17 To Paris': {'Genre': 'Drama', 'Gross': 'unknown', 'IMDB Metascore': '5
2', 'Popcorn Score': 'unknown', 'Rating': 'unrated', 'Tomato Score': 'unkown'}, 'T
he Commuter': {'popcornscore': 48, 'rating': 'PG13', 'tomatoscore': 58}, 'The Disa
ster Artist': {'popcornscore': 89, 'rating': 'R', 'tomatoscore': 91}, 'The Greates
t Showman': {'Genre': 'Biography', 'Gross': '$627,248', 'IMDB Metascore': '48', 'P
opcorn Score': 90, 'Rating': 'PG', 'Tomato Score': 55}, "The Insult (L'Insulte)":
{'popcornscore': 86, 'rating': 'R', 'tomatoscore': 89}, 'The Post': {'Genre': 'Bio
graphy', 'Gross': '$463,228', 'IMDB Metascore': '83', 'Popcorn Score': 73, 'Ratin
g': 'PG13', 'Tomato Score': 88}, 'The Shape Of Water': {'Genre': 'Adventure', 'Gro
ss': '$448,287', 'IMDB Metascore': '86', 'Popcorn Score': 78, 'Rating': 'R', 'Toma
to Score': 92}, 'Three Billboards Outside Ebbing, Missouri': {'popcornscore': 87,
'rating': 'R', 'tomatoscore': 93}, 'Till The End Of The World': {'popcornscore': -
1, 'rating': 'NR', 'tomatoscore': None}, 'Winchester': {'Genre': 'Biography', 'Gro
ss': '$696,786', 'IMDB Metascore': '28', 'Popcorn Score': 40, 'Rating': 'PG13', 'T
omato Score': 12}}]
```

In [86]:
```python
movies_dict = data[0]  # Extract the inner dictionary
df = pd.DataFrame.from_dict(movies_dict, orient='index')

# Reset the index to have movie titles as a column
df.reset_index(inplace=True)

# Rename columns
df.rename(columns={'index': 'Title', 'Gross': 'Revenue'}, inplace=True)

df.head()
```

Out[86]:

	Title	Genre	Revenue	IMDB Metascore	Popcorn Score	Rating	Tomato Score	popcornscore	rating	tor
0	12 Strong	Action	$453,173	54	72	R	54	NaN	NaN	
1	Den Of Thieves	Action	$491,898	49	69	R	40	NaN	NaN	
2	Fifty Shades Freed	Drama	unknown	34	unknown	unrated	unkown	NaN	NaN	
3	Golden Exits	Drama	unknown	72	unknown	unrated	unkown	NaN	NaN	
4	Hostiles	Adventure	$548,886	65	71	R	72	NaN	NaN	

In [87]:
```
df.tail()
```

Out[87]:

	Title	Genre	Revenue	IMDB Metascore	Popcorn Score	Rating	Tomato Score	popcornscore	rating	tom
33	The Commuter	NaN	NaN	NaN	NaN	NaN	NaN	48.0	PG13	
34	The Disaster Artist	NaN	NaN	NaN	NaN	NaN	NaN	89.0	R	
35	The Insult (L'Insulte)	NaN	NaN	NaN	NaN	NaN	NaN	86.0	R	
36	Three Billboards Outside Ebbing, Missouri	NaN	NaN	NaN	NaN	NaN	NaN	87.0	R	
37	Till The End Of The World	NaN	NaN	NaN	NaN	NaN	NaN	-1.0	NR	

In [88]:
```
df.isna().sum()
```

Out[88]:
```
Title               0
Genre              22
Revenue            22
IMDB Metascore     22
Popcorn Score      22
Rating             22
Tomato Score       22
popcornscore       16
rating             16
tomatoscore        17
dtype: int64
```

In [90]:
```
df['Title'].count()
```

Out[90]:
```
38
```

In [93]:
```
df['Genre'].value_counts()
```

Out[93]:
```
Genre
Action       4
Drama        4
Biography    3
Adventure    2
Animation    2
Comedy       1
Name: count, dtype: int64
```

In [96]:
```
df['IMDB Metascore'].mode()
```

Out[96]:
```
0    65
Name: IMDB Metascore, dtype: object
```

In [101…
```
df.head()
```

Out[101]:

	Title	Genre	Revenue	IMDB Metascore	Popcorn Score	Rating	Tomato Score	popcornscore	rating	tor
0	12 Strong	Action	$453,173	54	72	R	54	NaN	NaN	
1	Den Of Thieves	Action	$491,898	49	69	R	40	NaN	NaN	
2	Fifty Shades Freed	Drama	unknown	34	unknown	unrated	unkown	NaN	NaN	
3	Golden Exits	Drama	unknown	72	unknown	unrated	unkown	NaN	NaN	
4	Hostiles	Adventure	$548,886	65	71	R	72	NaN	NaN	

In [102…
```
df=df.fillna({'Genre': 'Unknown',
    'Revenue': '$0',
    'IMDB Metascore': 'Unknown',
    'Popcorn Score': 0,
    'Tomato Score': 0
          })
```

In [103…
```
df.tail()
```

Out[103]:

	Title	Genre	Revenue	IMDB Metascore	Popcorn Score	Rating	Tomato Score	popcornscore	rating	t
33	The Commuter	Unknown	$0	Unknown	0	NaN	0	48.0	PG13	
34	The Disaster Artist	Unknown	$0	Unknown	0	NaN	0	89.0	R	
35	The Insult (L'Insulte)	Unknown	$0	Unknown	0	NaN	0	86.0	R	
36	Three Billboards Outside Ebbing, Missouri	Unknown	$0	Unknown	0	NaN	0	87.0	R	
37	Till The End Of The World	Unknown	$0	Unknown	0	NaN	0	-1.0	NR	

In [105…

```
df.isna().sum()
```

Out[105]:

```
Title              0
Genre              0
Revenue            0
IMDB Metascore     0
Popcorn Score      0
Rating            22
Tomato Score       0
popcornscore      16
rating            16
tomatoscore       17
dtype: int64
```

In [106…

```
df = df.fillna({
    'Rating': 'NR',                 # For 'Rating', fill with 'NR' (Not Rated)
    'Tomato Score': 0,              # For 'Tomato Score', fill with 0
    'popcornscore': 0,              # For 'popcornscore', fill with 0
    'rating': 'NR',                 # For 'rating', fill with 'NR' (Not Rated)
    'tomatoscore': 0                # For 'tomatoscore', fill with 0
})
```

In [107…

```
df
```

Out[107]:

	Title	Genre	Revenue	IMDB Metascore	Popcorn Score	Rating	Tomato Score	popcornscore	ratin
0	12 Strong	Action	$453,173	54	72	R	54	0.0	N
1	Den Of Thieves	Action	$491,898	49	69	R	40	0.0	N
2	Fifty Shades Freed	Drama	unknown	34	unknown	unrated	unkown	0.0	N
3	Golden Exits	Drama	unknown	72	unknown	unrated	unkown	0.0	N
4	Hostiles	Adventure	$548,886	65	71	R	72	0.0	N
5	Jumanji: Welcome To The Jungle	Action	$760,867	58	89	PG13	76	0.0	N
6	Maze Runner: The Death Cure	Action	$720,463	51	71	PG13	43	0.0	N
7	Paddington 2	Animation	$184,414	88	89	PG	100	0.0	N
8	Permission	Comedy	unknown	53	unknown	unrated	unkown	0.0	N
9	Peter Rabbit	Animation	unknown	56	unknown	unrated	unkown	0.0	N
10	Sanpo Suru Shinryakusha	Drama	unknown	65	unknown	unrated	unkown	0.0	N
11	The 15:17 To Paris	Drama	unknown	52	unknown	unrated	unkown	0.0	N
12	The Greatest Showman	Biography	$627,248	48	90	PG	55	0.0	N
13	The Post	Biography	$463,228	83	73	PG13	88	0.0	N
14	The Shape Of Water	Adventure	$448,287	86	78	R	92	0.0	N
15	Winchester	Biography	$696,786	28	40	PG13	12	0.0	N
16	A Fantastic Woman (Una Mujer Fantástica)	Unknown	$0	Unknown	0	NR	0	83.0	
17	All The Money In The World	Unknown	$0	Unknown	0	NR	0	71.0	
18	Bilal: A New Breed Of Hero	Unknown	$0	Unknown	0	NR	0	91.0	PG1
19	Call Me By Your Name	Unknown	$0	Unknown	0	NR	0	87.0	
20	Darkest Hour	Unknown	$0	Unknown	0	NR	0	84.0	PG1
21	Ferdinand	Unknown	$0	Unknown	0	NR	0	49.0	P
22	Film Stars Don'T Die In Liverpool	Unknown	$0	Unknown	0	NR	0	69.0	

	Title	Genre	Revenue	IMDB Metascore	Popcorn Score	Rating	Tomato Score	popcornscore	ratin
23	Forever My Girl	Unknown	$0	Unknown	0	NR	0	91.0	P
24	I, Tonya	Unknown	$0	Unknown	0	NR	0	89.0	
25	Insidious: The Last Key	Unknown	$0	Unknown	0	NR	0	51.0	PG1
26	Mary And The Witch'S Flower	Unknown	$0	Unknown	0	NR	0	78.0	P
27	Molly'S Game	Unknown	$0	Unknown	0	NR	0	85.0	
28	Padmaavat	Unknown	$0	Unknown	0	NR	0	62.0	N
29	Phantom Thread	Unknown	$0	Unknown	0	NR	0	68.0	
30	Pitch Perfect 3	Unknown	$0	Unknown	0	NR	0	52.0	PG1
31	Proud Mary	Unknown	$0	Unknown	0	NR	0	56.0	
32	Star Wars: The Last Jedi	Unknown	$0	Unknown	0	NR	0	48.0	PG1
33	The Commuter	Unknown	$0	Unknown	0	NR	0	48.0	PG1
34	The Disaster Artist	Unknown	$0	Unknown	0	NR	0	89.0	
35	The Insult (L'Insulte)	Unknown	$0	Unknown	0	NR	0	86.0	
36	Three Billboards Outside Ebbing, Missouri	Unknown	$0	Unknown	0	NR	0	87.0	
37	Till The End Of The	Unknown	$0	Unknown	0	NR	0	-1.0	N

In [108…
```
df['Revenue']=df['Revenue'].replace('unknown','$0')
```

In [109…
```
df
```

Out[109]:

	Title	Genre	Revenue	IMDB Metascore	Popcorn Score	Rating	Tomato Score	popcornscore	ratin
0	12 Strong	Action	$453,173	54	72	R	54	0.0	N
1	Den Of Thieves	Action	$491,898	49	69	R	40	0.0	N
2	Fifty Shades Freed	Drama	$0	34	unknown	unrated	unkown	0.0	N
3	Golden Exits	Drama	$0	72	unknown	unrated	unkown	0.0	N
4	Hostiles	Adventure	$548,886	65	71	R	72	0.0	N
5	Jumanji: Welcome To The Jungle	Action	$760,867	58	89	PG13	76	0.0	N
6	Maze Runner: The Death Cure	Action	$720,463	51	71	PG13	43	0.0	N
7	Paddington 2	Animation	$184,414	88	89	PG	100	0.0	N
8	Permission	Comedy	$0	53	unknown	unrated	unkown	0.0	N
9	Peter Rabbit	Animation	$0	56	unknown	unrated	unkown	0.0	N
10	Sanpo Suru Shinryakusha	Drama	$0	65	unknown	unrated	unkown	0.0	N
11	The 15:17 To Paris	Drama	$0	52	unknown	unrated	unkown	0.0	N
12	The Greatest Showman	Biography	$627,248	48	90	PG	55	0.0	N
13	The Post	Biography	$463,228	83	73	PG13	88	0.0	N
14	The Shape Of Water	Adventure	$448,287	86	78	R	92	0.0	N
15	Winchester	Biography	$696,786	28	40	PG13	12	0.0	N
16	A Fantastic Woman (Una Mujer Fantástica)	Unknown	$0	Unknown	0	NR	0	83.0	
17	All The Money In The World	Unknown	$0	Unknown	0	NR	0	71.0	
18	Bilal: A New Breed Of Hero	Unknown	$0	Unknown	0	NR	0	91.0	PG1
19	Call Me By Your Name	Unknown	$0	Unknown	0	NR	0	87.0	
20	Darkest Hour	Unknown	$0	Unknown	0	NR	0	84.0	PG1
21	Ferdinand	Unknown	$0	Unknown	0	NR	0	49.0	P
22	Film Stars Don'T Die In Liverpool	Unknown	$0	Unknown	0	NR	0	69.0	

	Title	Genre	Revenue	IMDB Metascore	Popcorn Score	Rating	Tomato Score	popcornscore	ratin
23	Forever My Girl	Unknown	$0	Unknown	0	NR	0	91.0	PG
24	I, Tonya	Unknown	$0	Unknown	0	NR	0	89.0	
25	Insidious: The Last Key	Unknown	$0	Unknown	0	NR	0	51.0	PG1
26	Mary And The Witch'S Flower	Unknown	$0	Unknown	0	NR	0	78.0	PG
27	Molly'S Game	Unknown	$0	Unknown	0	NR	0	85.0	
28	Padmaavat	Unknown	$0	Unknown	0	NR	0	62.0	N
29	Phantom Thread	Unknown	$0	Unknown	0	NR	0	68.0	
30	Pitch Perfect 3	Unknown	$0	Unknown	0	NR	0	52.0	PG1
31	Proud Mary	Unknown	$0	Unknown	0	NR	0	56.0	
32	Star Wars: The Last Jedi	Unknown	$0	Unknown	0	NR	0	48.0	PG1
33	The Commuter	Unknown	$0	Unknown	0	NR	0	48.0	PG1
34	The Disaster Artist	Unknown	$0	Unknown	0	NR	0	89.0	
35	The Insult (L'Insulte)	Unknown	$0	Unknown	0	NR	0	86.0	
36	Three Billboards Outside Ebbing, Missouri	Unknown	$0	Unknown	0	NR	0	87.0	
37	Till The End Of The	Unknown	$0	Unknown	0	NR	0	-1.0	N

In [110…

```
df['Popcorn Score'].value_counts()
```

Out[110]:

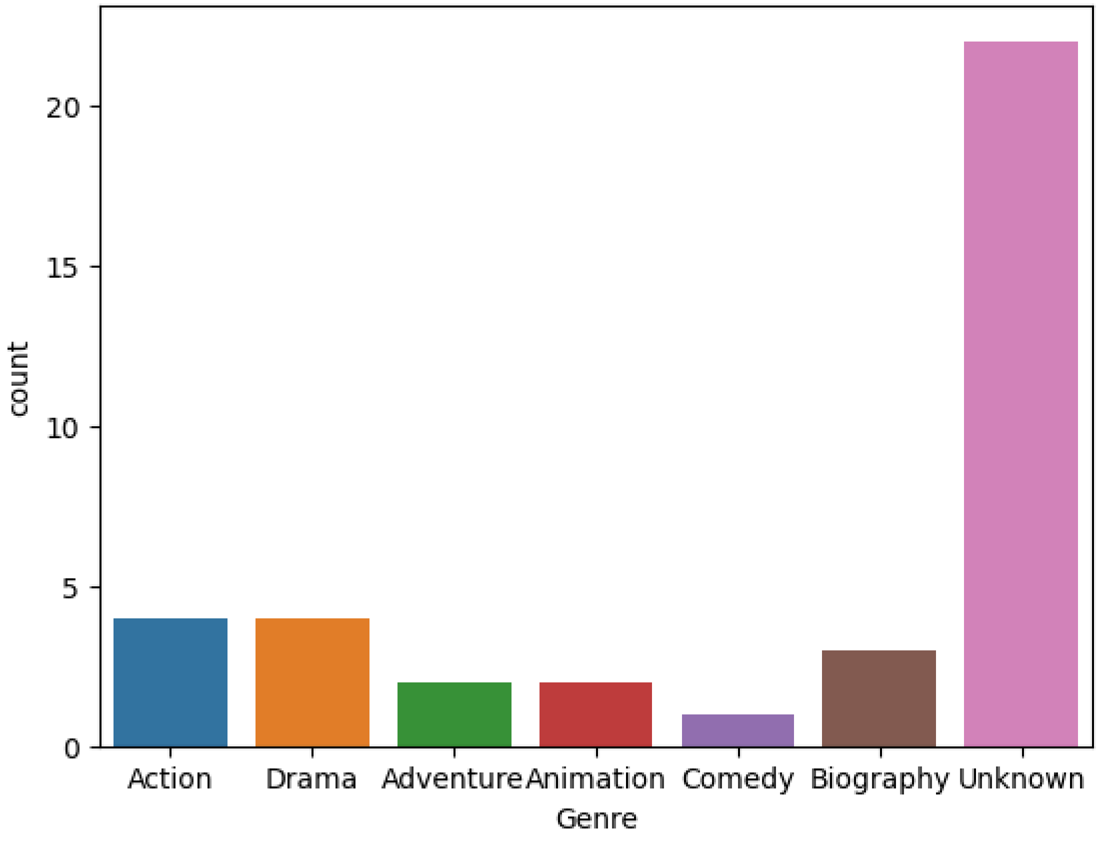
```
Popcorn Score
0          22
unknown     6
71          2
89          2
72          1
69          1
90          1
73          1
78          1
40          1
Name: count, dtype: int64
```

In [114…

```
genre_counts=df['Genre'].value_counts()
genre_counts
```

Out[114]:
```
Genre
Unknown      22
Action        4
Drama         4
Biography     3
Adventure     2
Animation     2
Comedy        1
Name: count, dtype: int64
```

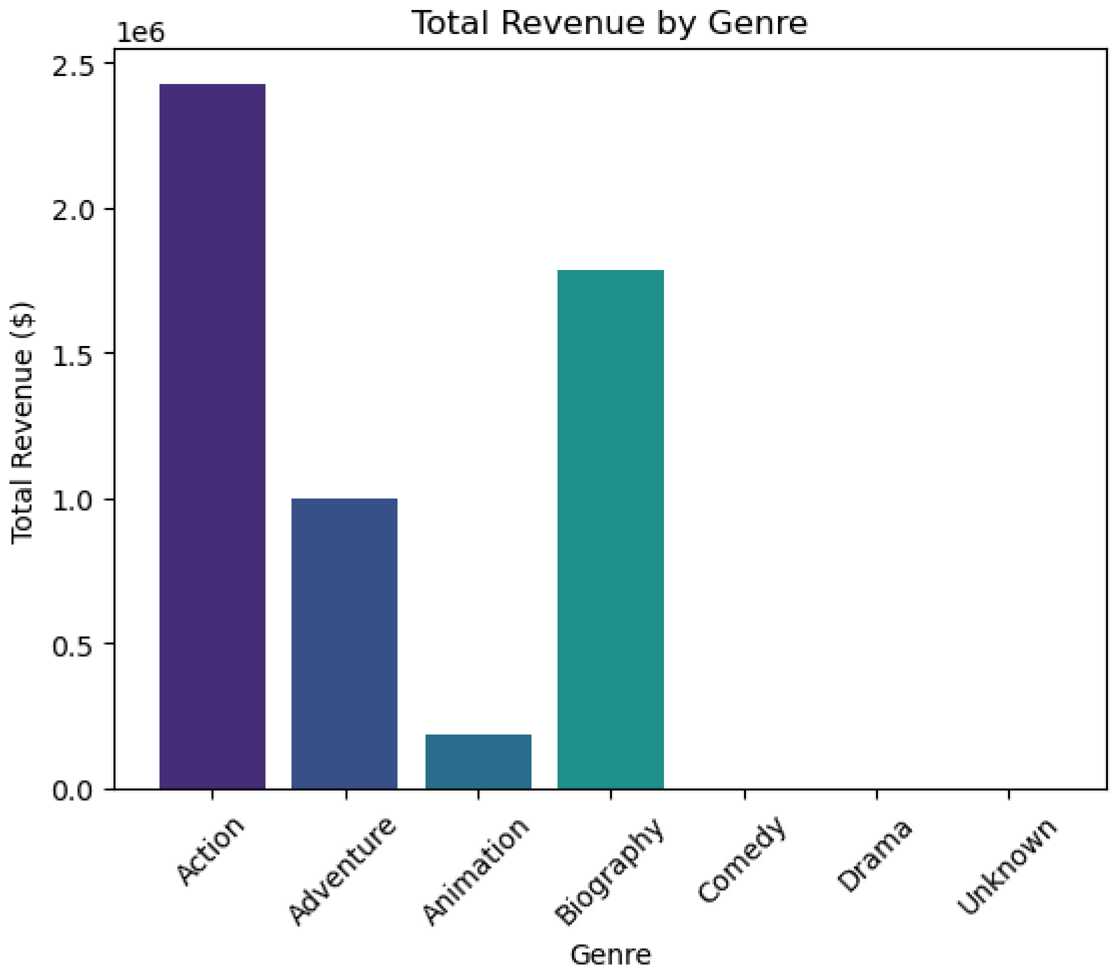
In [113…
```
sns.countplot(data=df,x='Genre')
plt.xlabel("Genre")
plt.ylabel("count")
plt.show()
```



In [116…
```
genre_revenue=df.groupby('Genre')['Revenue'].sum()
genre_revenue
```

Out[116]:
```
Genre
Action                   $453,173$491,898$760,867$720,463
Adventure                                $548,886$448,287
Animation                                     $184,414$0
Biography                        $627,248$463,228$696,786
Comedy                                                $0
Drama                                           $0$0$0$0
Unknown      $0$0$0$0$0$0$0$0$0$0$0$0$0$0$0$0$0$0$0$0$0$0
Name: Revenue, dtype: object
```

In [122…
```
df['Revenue']=df['Revenue'].replace('[\$,]', '', regex=True).astype(float)

genre_rev=df.groupby('Genre')['Revenue'].sum()

colors = sns.color_palette("viridis", len(genre_revenue))

plt.bar(genre_rev.index,genre_rev.values,color=colors)
plt.xlabel('Genre')
plt.ylabel('Total Revenue ($)')
plt.title('Total Revenue by Genre')
plt.xticks(rotation=45)  # Rotate x-axis labels for readability
plt.show()
```



In [135…
```
df['IMDB Metascore']=df['IMDB Metascore'].replace("Unknown","0").astype(int)
```

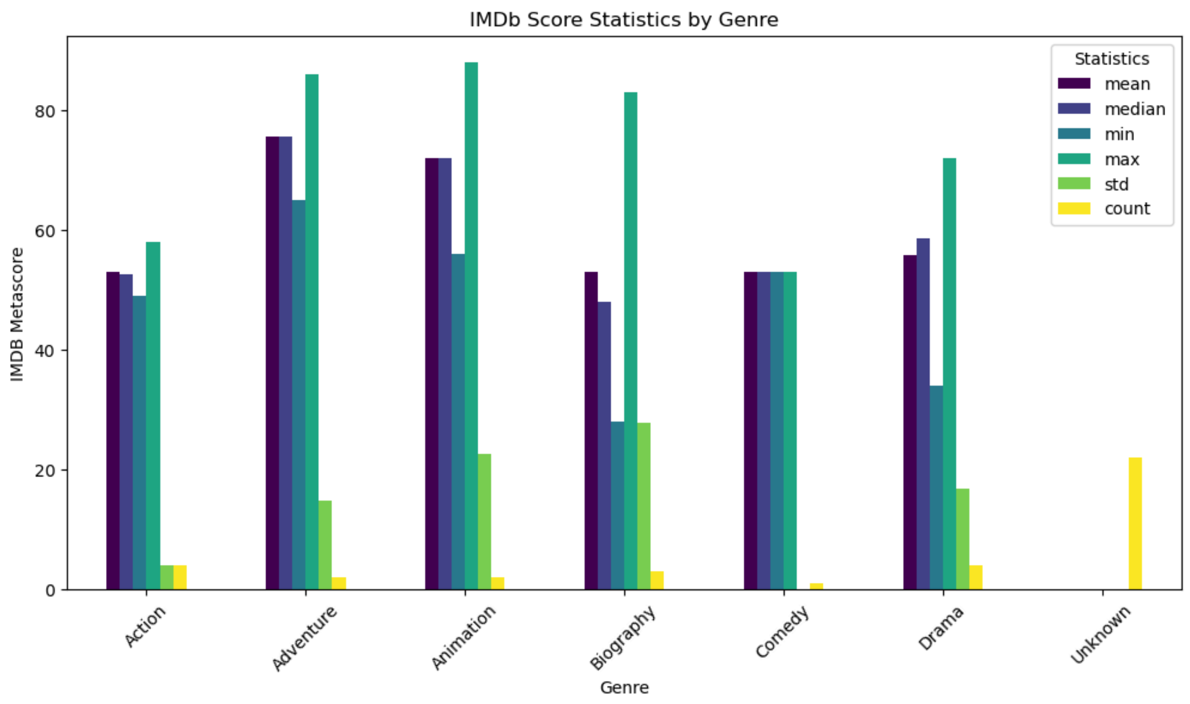
In [145…
```
genre_im=df.groupby('Genre')['IMDB Metascore']
```

In [146…
```
genre_imdb_agg = genre_im.agg(['mean', 'median', 'min', 'max', 'std', 'count'])
genre_imdb_agg
```

Out[146]:

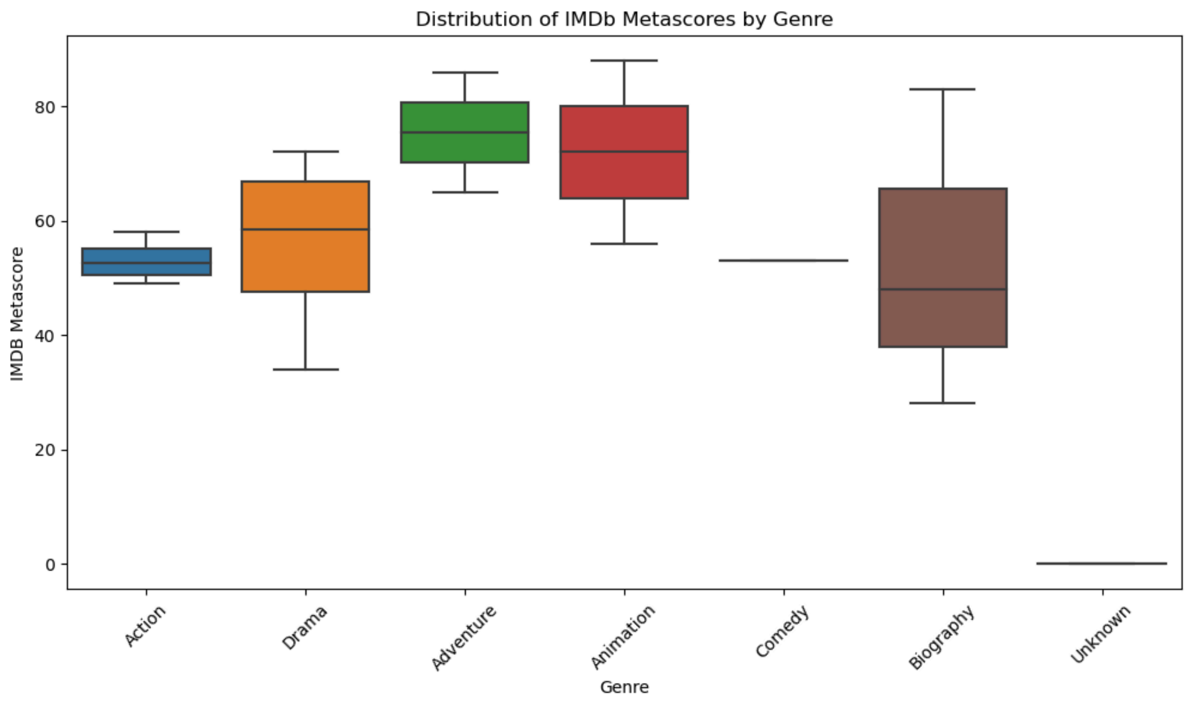
Genre	mean	median	min	max	std	count
Action	53.00	52.5	49	58	3.915780	4
Adventure	75.50	75.5	65	86	14.849242	2
Animation	72.00	72.0	56	88	22.627417	2
Biography	53.00	48.0	28	83	27.838822	3
Comedy	53.00	53.0	53	53	NaN	1
Drama	55.75	58.5	34	72	16.700798	4
Unknown	0.00	0.0	0	0	0.000000	22

In [199…
```
genre_imdb_agg.plot(kind='bar',figsize=(12,6),colormap='viridis')
plt.title("IMDb Score Statistics by Genre")
plt.xlabel("Genre")
plt.ylabel("IMDB Metascore")
plt.xticks(rotation=45)
plt.legend(title="Statistics")
plt.show()
```



In [144…

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Genre', y='IMDB Metascore', data=df)
plt.title("Distribution of IMDb Metascores by Genre")
plt.xlabel("Genre")
plt.ylabel("IMDB Metascore")
plt.xticks(rotation=45)
plt.show()
```



6TH

In [148…

```
with open("List of the Countries and Territories.txt") as file:
    lstdata=file.readlines()
```

In [149…

```
for line in lstdata:
    print(line.strip())
```

```
Afghanistan
Albania
Algeria
American Samoa
Andorra
Angola
Anguilla
Antigua and Barbuda
Argentina
Armenia
Aruba
Australia
Austria
Azerbaijan
Bahamas
Bahrain
Bangladesh
Barbados
Belarus
Belgium
Belize
Benin
Bermuda
Bhutan
Bolivia
Bosnia and Herzegovina
Botswana
Brazil
British Virgin Islands
Brunei
Bulgaria
Burkina Faso
Burundi
Cambodia
Cameroon
Canada
Cape Verde
Cayman Islands
Central African Republic
Chad
Chile
China
Colombia
Comoros
Cook Islands
Costa Rica
Croatia
Cuba
Curacao
Cyprus
Czech Republic
Denmark
Djibouti
Dominica
Dominican Republic
DR Congo
Ecuador
Egypt
El Salvador
Equatorial Guinea
Eritrea
Estonia
Eswatini
Ethiopia
```

```
                   Falkland Islands
                   Faroe Islands
                   Fiji
                   Finland
                   France
                   French Guiana
                   French Polynesia
                   Gabon
                   Gambia
                   Georgia
                   Germany
                   Ghana
                   Gibraltar
                   Greece
                   Greenland
                   Grenada
                   Guadeloupe
                   Guam
                   Guatemala
                   Guernsey
                   Guinea
                   Guinea-Bissau
                   Guyana
                   Haiti
                   Honduras
                   Hong Kong
                   Hungary
                   Iceland
                   India
                   Indonesia
                   Iran
                   Iraq
                   Ireland
                   Isle of Man
                   Israel
                   Italy
                   Ivory Coast
                   Jamaica
                   Japan
                   Jersey
                   Jordan
                   Kazakhstan
                   Kenya
                   Kiribati
                   Kuwait
                   Kyrgyzstan
                   Laos
                   Latvia
                   Lebanon
                   Lesotho
                   Liberia
                   Libya
                   Liechtenstein
                   Lithuania
                   Luxembourg
                   Macau
                   Madagascar
                   Malawi
                   Malaysia
                   Maldives
                   Mali
                   Malta
                   Marshall Islands
                   Martinique
```

```
Mauritania
Mauritius
Mayotte
Mexico
Micronesia
Moldova
Monaco
Mongolia
Montenegro
Montserrat
Morocco
Mozambique
Myanmar
Namibia
Nauru
Nepal
Netherlands
New Caledonia
New Zealand
Nicaragua
Niger
Nigeria
Niue
North Korea
North Macedonia
Northern Mariana Islands
Norway
Oman
Pakistan
Palau
Palestine
Panama
Papua New Guinea
Paraguay
Peru
Philippines
Poland
Portugal
Puerto Rico
Qatar
Republic of the Congo
Reunion
Romania
Russia
Rwanda
Saint Barthelemy
Saint Kitts and Nevis
Saint Lucia
Saint Martin
Saint Pierre and Miquelon
Saint Vincent and the Grenadines
Samoa
San Marino
Sao Tome and Principe
Saudi Arabia
Senegal
Serbia
Seychelles
Sierra Leone
Singapore
Sint Maarten
Slovakia
Slovenia
Solomon Islands
```

```
Somalia
South Africa
South Korea
South Sudan
Spain
Sri Lanka
Sudan
Suriname
Sweden
Switzerland
Syria
Taiwan
Tajikistan
Tanzania
Thailand
Timor-Leste
Togo
Tokelau
Tonga
Trinidad and Tobago
Tunisia
Turkey
Turkmenistan
Turks and Caicos Islands
Tuvalu
Uganda
Ukraine
United Arab Emirates
United Kingdom
United States
United States Virgin Islands
Uruguay
Uzbekistan
Vanuatu
Vatican City
Venezuela
Vietnam
Wallis and Futuna
Western Sahara
Yemen
Zambia
Zimbabwe
```

In [156…

```
lstdf=pd.DataFrame(lstdata)
lstdf
```

Out[156]:

	0
0	Afghanistan\n
1	Albania\n
2	Algeria\n
3	American Samoa\n
4	Andorra\n
...	...
229	Wallis and Futuna\n
230	Western Sahara\n
231	Yemen\n
232	Zambia\n
233	Zimbabwe

234 rows × 1 columns

In [159…
```
lstdf.columns=['Country']
```

In [161…
```
lstdf['Country']=lstdf['Country'].str.strip()
```

In [162…
```
lstdf
```

Out[162]:

	Country
0	Afghanistan
1	Albania
2	Algeria
3	American Samoa
4	Andorra
...	...
229	Wallis and Futuna
230	Western Sahara
231	Yemen
232	Zambia
233	Zimbabwe

234 rows × 1 columns

In [163…
```
lstdf.count()
```

Out[163]:
```
Country    234
dtype: int64
```

In [165…
```
lstdf['Country'].value_counts().sum()
```

Out[165]:
```
234
```

In [166…
```
lstdf.isna().sum()
```
Out[166]:
```
Country    0
dtype: int64
```

7TH

In [167…
```
lotrdf=pd.read_csv("LOTR.csv")
lotrdf
```
Out[167]:

	FellowshipID	FirstName	Skills
0	1001	Frodo	Hiding
1	1002	Samwise	Gardening
2	1003	Gandalf	Spells
3	1004	Pippin	Fireworks

In [169…
```
lotrdf['FellowshipID'].value_counts().sum()
```
Out[169]:
```
4
```

8TH

In [170…
```
lort2=pd.read_csv("LOTR 2.csv")
lort2
```
Out[170]:

	FellowshipID	FirstName	Age
0	1001	Frodo	50
1	1002	Samwise	39
2	1006	Legolas	2931
3	1007	Elrond	6520
4	1008	Barromir	51

In [171…
```
lort2['FellowshipID'].value_counts()
```
Out[171]:
```
FellowshipID
1001    1
1002    1
1006    1
1007    1
1008    1
Name: count, dtype: int64
```

In [172…
```
lort2['FellowshipID'].value_counts().sum()
```
Out[172]:
```
5
```

9TH

In [173…
```
worlddf=pd.read_csv("world_population.csv")
worlddf
```

Out[173]:

	Rank	CCA3	Country	Capital	Continent	2022 Population	2020 Population	2015 Population	201 Populatio
0	36	AFG	Afghanistan	Kabul	Asia	41128771.0	38972230.0	33753499.0	28189672.
1	138	ALB	Albania	Tirana	Europe	2842321.0	2866849.0	2882481.0	2913399.
2	34	DZA	Algeria	Algiers	Africa	44903225.0	43451666.0	39543154.0	35856344.
3	213	ASM	American Samoa	Pago Pago	Oceania	44273.0	46189.0	51368.0	54849.
4	203	AND	Andorra	Andorra la Vella	Europe	79824.0	77700.0	71746.0	71519.
...	...	...	...	...	...	...	...	...	.
229	226	WLF	Wallis and Futuna	Mata-Utu	Oceania	11572.0	11655.0	12182.0	13142.
230	172	ESH	Western Sahara	El Aaiún	Africa	575986.0	556048.0	491824.0	413296.
231	46	YEM	Yemen	Sanaa	Asia	33696614.0	32284046.0	28516545.0	24743946.
232	63	ZMB	Zambia	Lusaka	Africa	20017675.0	18927715.0	NaN	13792086.
233	74	ZWE	Zimbabwe	Harare	Africa	16320537.0	15669666.0	14154937.0	12839771.

234 rows × 17 columns

In [174…
```
worlddf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Rank                         234 non-null    int64  
 1   CCA3                         234 non-null    object 
 2   Country                      234 non-null    object 
 3   Capital                      234 non-null    object 
 4   Continent                    234 non-null    object 
 5   2022 Population              230 non-null    float64
 6   2020 Population              233 non-null    float64
 7   2015 Population              230 non-null    float64
 8   2010 Population              227 non-null    float64
 9   2000 Population              227 non-null    float64
 10  1990 Population              229 non-null    float64
 11  1980 Population              229 non-null    float64
 12  1970 Population              230 non-null    float64
 13  Area (km²)                   232 non-null    float64
 14  Density (per km²)            230 non-null    float64
 15  Growth Rate                  232 non-null    float64
 16  World Population Percentage  234 non-null    float64
dtypes: float64(12), int64(1), object(4)
memory usage: 31.2+ KB
```

In [176…
```
worlddf.shape
```

Out[176]:
```
(234, 17)
```

In [178…
```
worlddf.isna().sum()
```

Out[178]:
```
Rank                           0
CCA3                           0
Country                        0
Capital                        0
Continent                      0
2022 Population                4
2020 Population                1
2015 Population                4
2010 Population                7
2000 Population                7
1990 Population                5
1980 Population                5
1970 Population                4
Area (km²)                     2
Density (per km²)              4
Growth Rate                    2
World Population Percentage    0
dtype: int64
```

In [182…
```
worlddf[worlddf['2022 Population'].isna()]
```

Out[182]:

	Rank	CCA3	Country	Capital	Continent	2022 Population	2020 Population	2015 Population	201 Populatio
62	159	SWZ	Eswatini	Mbabane	Africa	NaN	1180655.0	1133936.0	1099920.
154	120	NOR	Norway	Oslo	Europe	NaN	5379839.0	NaN	4889741.
157	222	PLW	Palau	Ngerulmud	Oceania	NaN	17972.0	17794.0	18540.
207	155	TLS	Timor-Leste	Dili	Asia	NaN	1299995.0	1205813.0	1088486.

In [183…
```
mean22=worlddf['2022 Population'].mean()
mean22
```

Out[183]:
```
34632250.87826087
```

In [184…
```
worlddf['2022 Population']=worlddf['2022 Population'].fillna(mean22)
```

In [185…
```
worlddf['2022 Population'].isna().sum()
```

Out[185]:
```
0
```

In [186…
```
mean20=worlddf['2022 Population'].mean()
mean20
```

Out[186]:
```
34632250.878260866
```

In [187…
```
worlddf['2020 Population']=worlddf['2020 Population'].fillna(mean20)
```

In [188…
```
columns = ['2015 Population', '2010 Population', '2000 Population', '1990 Populatic

# Fill missing values for each column with the column's mean
for col in columns:
    mean_value = worlddf[col].mean()
    worlddf[col] = worlddf[col].fillna(mean_value)
```

In [189…

```
worlddf.isna().sum()
```

Out[189]:

```
Rank                           0
CCA3                           0
Country                        0
Capital                        0
Continent                      0
2022 Population                0
2020 Population                0
2015 Population                0
2010 Population                0
2000 Population                0
1990 Population                0
1980 Population                0
1970 Population                0
Area (km²)                     2
Density (per km²)              4
Growth Rate                    2
World Population Percentage    0
dtype: int64
```

In [194…

```
worlddf.dropna(inplace=True)
```

In [195…

```
worlddf.isna().sum()
```

Out[195]:

```
Rank                           0
CCA3                           0
Country                        0
Capital                        0
Continent                      0
2022 Population                0
2020 Population                0
2015 Population                0
2010 Population                0
2000 Population                0
1990 Population                0
1980 Population                0
1970 Population                0
Area (km²)                     0
Density (per km²)              0
Growth Rate                    0
World Population Percentage    0
dtype: int64
```

In [196…

```
worlddf.head()
```

Out[196]:

	Rank	CCA3	Country	Capital	Continent	2022 Population	2020 Population	2015 Population	2010 Population
0	36	AFG	Afghanistan	Kabul	Asia	41128771.0	38972230.0	33753499.0	28189672.0
1	138	ALB	Albania	Tirana	Europe	2842321.0	2866849.0	2882481.0	2913399.0
2	34	DZA	Algeria	Algiers	Africa	44903225.0	43451666.0	39543154.0	35856344.0
3	213	ASM	American Samoa	Pago Pago	Oceania	44273.0	46189.0	51368.0	54849.0
4	203	AND	Andorra	Andorra la Vella	Europe	79824.0	77700.0	71746.0	71519.0

In [213…

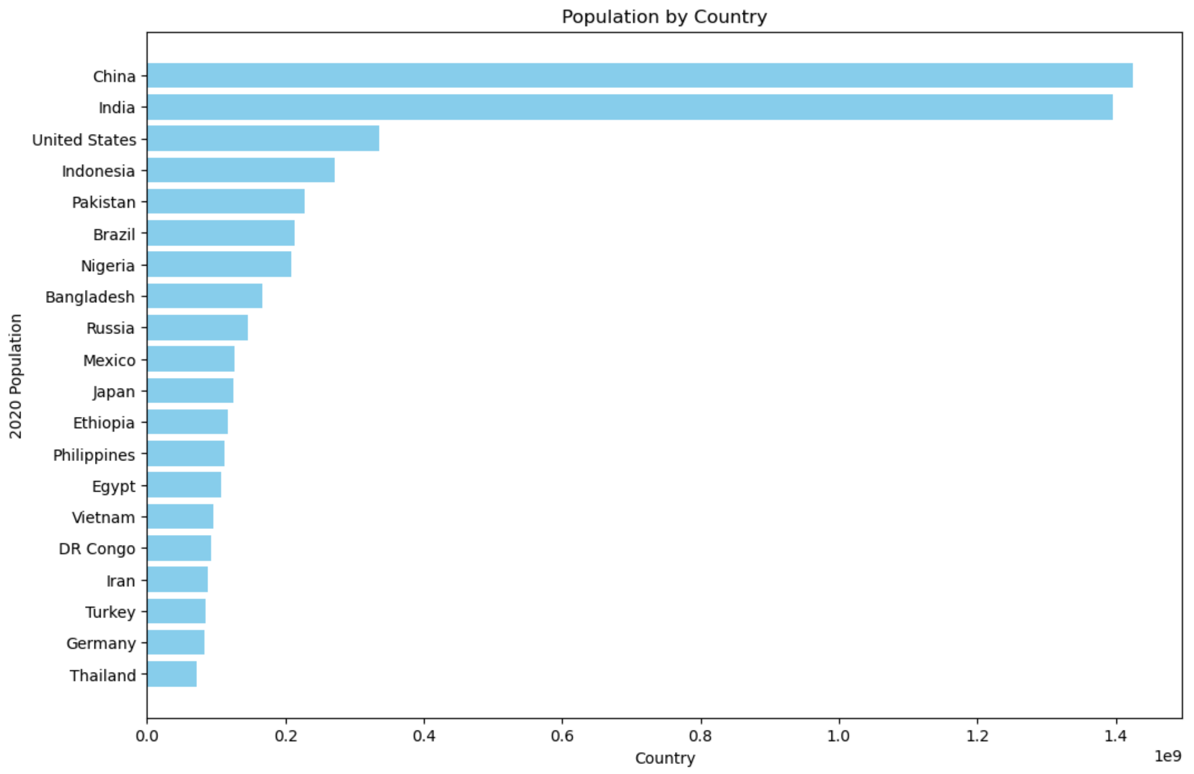
```
worlddf['Country'].count()
```

Out[213]:

```
229
```

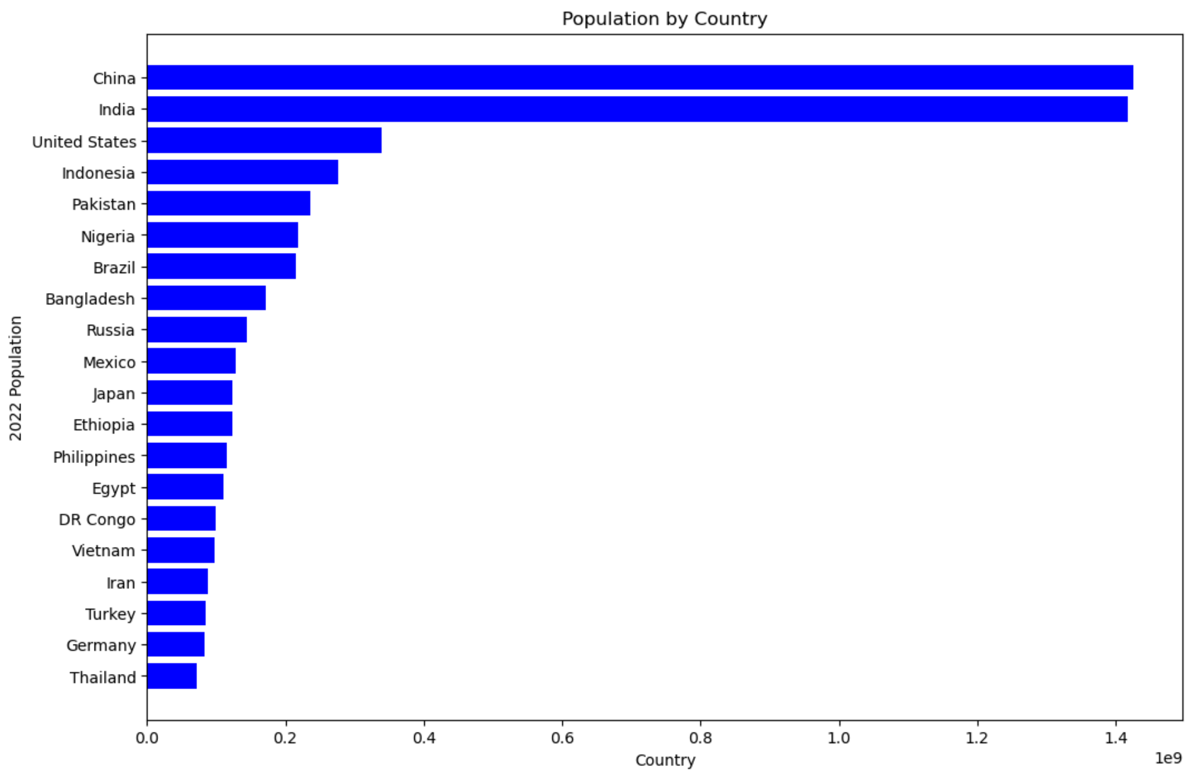
In [217…

```
sorted_df_20 = worlddf.sort_values('2020 Population', ascending=False).head(20)

plt.figure(figsize=(12, 8))
plt.barh(sorted_df_20['Country'], sorted_df_20['2020 Population'], color='skyblue')
plt.title('Population by Country')
plt.ylabel("2020 Population")
plt.xlabel("Country")
plt.gca().invert_yaxis()  # Largest bar on top
plt.show()
```



In [223…

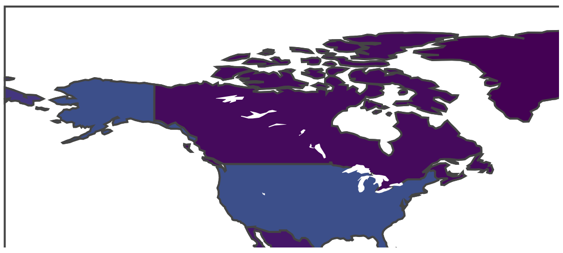
```
sorted_df_22 = worlddf.sort_values('2022 Population', ascending=False).head(20)

plt.figure(figsize=(12, 8))
plt.barh(sorted_df_22['Country'], sorted_df_22['2022 Population'], color='blue')
plt.title('Population by Country')
plt.ylabel("2022 Population")
plt.xlabel("Country")
plt.gca().invert_yaxis()  # Largest bar on top
plt.show()
```



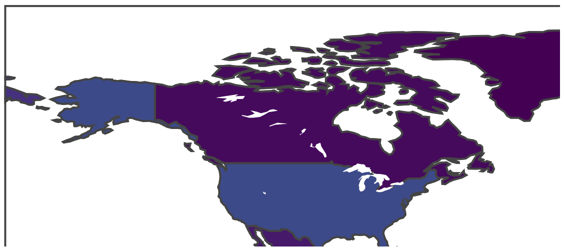
In [226…

```
years = ['1970 Population', '1980 Population', '1990 Population', '2000 Population'
import plotly.express as px

# Loop through each year, now including 2015
for year in years:
    fig = px.choropleth(worlddf, locations="Country", locationmode="country names",
                        color=year, color_continuous_scale="Viridis",
                        title=f"World Population by Country in {year.split()[0]}")
    fig.show()
```

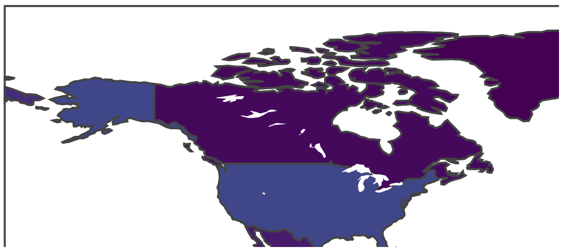
World Population by Country in 1970



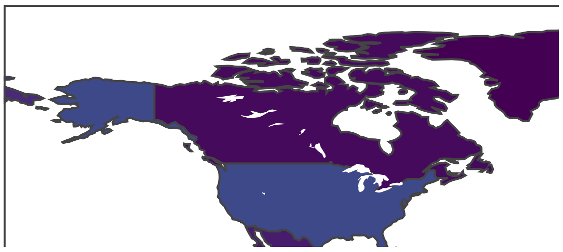
World Population by Country in 1980



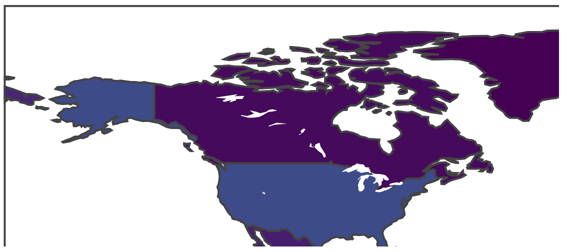
World Population by Country in 1990



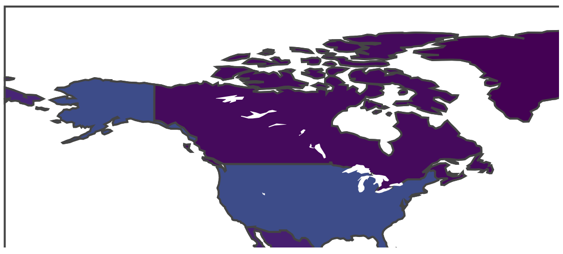
World Population by Country in 2000



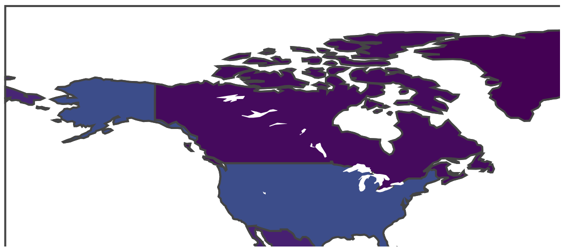
World Population by Country in 2010



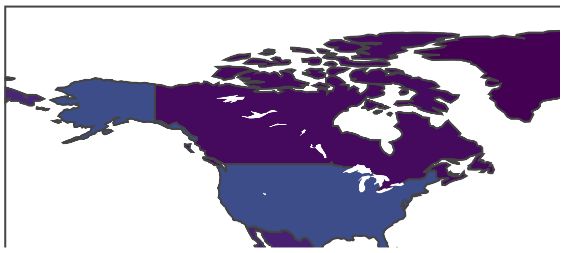
World Population by Country in 2015



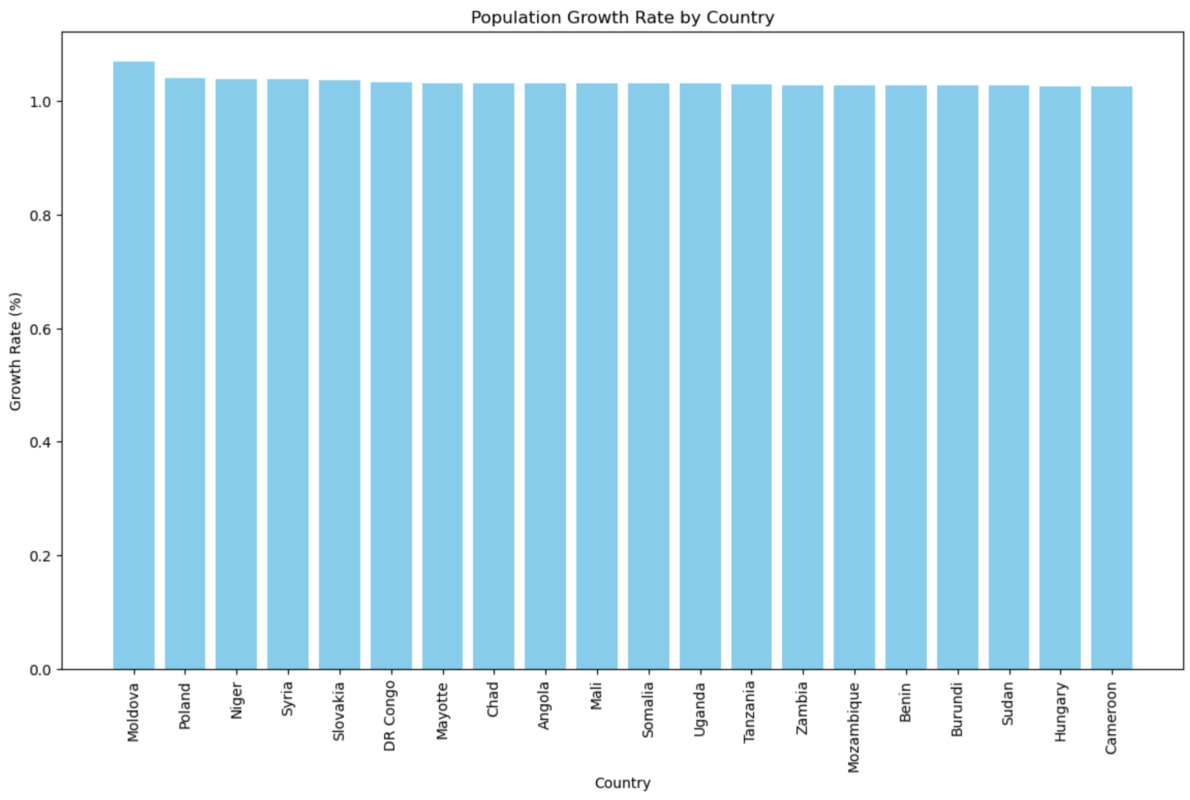
World Population by Country in 2020



World Population by Country in 2022



```
In [234…
sorted_worlddf = worlddf.sort_values(by='Growth Rate', ascending=False).head(20)

# Plot
plt.figure(figsize=(14, 8))
plt.bar(sorted_worlddf['Country'], sorted_worlddf['Growth Rate'], color='skyblue')
plt.xlabel('Country')
plt.ylabel('Growth Rate (%)')
plt.title('Population Growth Rate by Country')
plt.xticks(rotation=90)
plt.show()
```



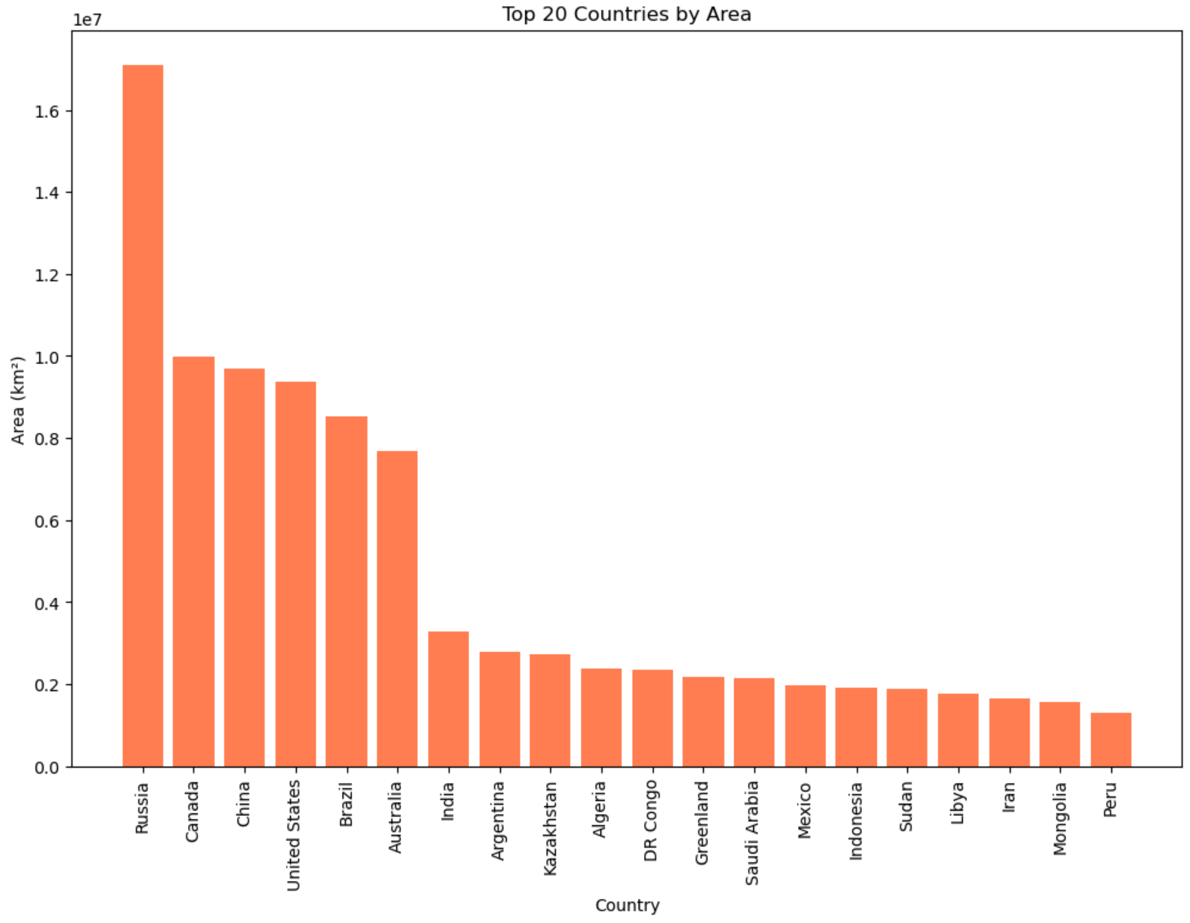
In [239…

```
worlddf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 229 entries, 0 to 233
Data columns (total 17 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Rank                          229 non-null    int64  
 1   CCA3                          229 non-null    object 
 2   Country                       229 non-null    object 
 3   Capital                       229 non-null    object 
 4   Continent                     229 non-null    object 
 5   2022 Population               229 non-null    float64
 6   2020 Population               229 non-null    float64
 7   2015 Population               229 non-null    float64
 8   2010 Population               229 non-null    float64
 9   2000 Population               229 non-null    float64
 10  1990 Population               229 non-null    float64
 11  1980 Population               229 non-null    float64
 12  1970 Population               229 non-null    float64
 13  Area (km²)                    229 non-null    float64
 14  Density (per km²)             229 non-null    float64
 15  Growth Rate                   229 non-null    float64
 16  World Population Percentage  229 non-null    float64
dtypes: float64(12), int64(1), object(4)
memory usage: 32.2+ KB
```

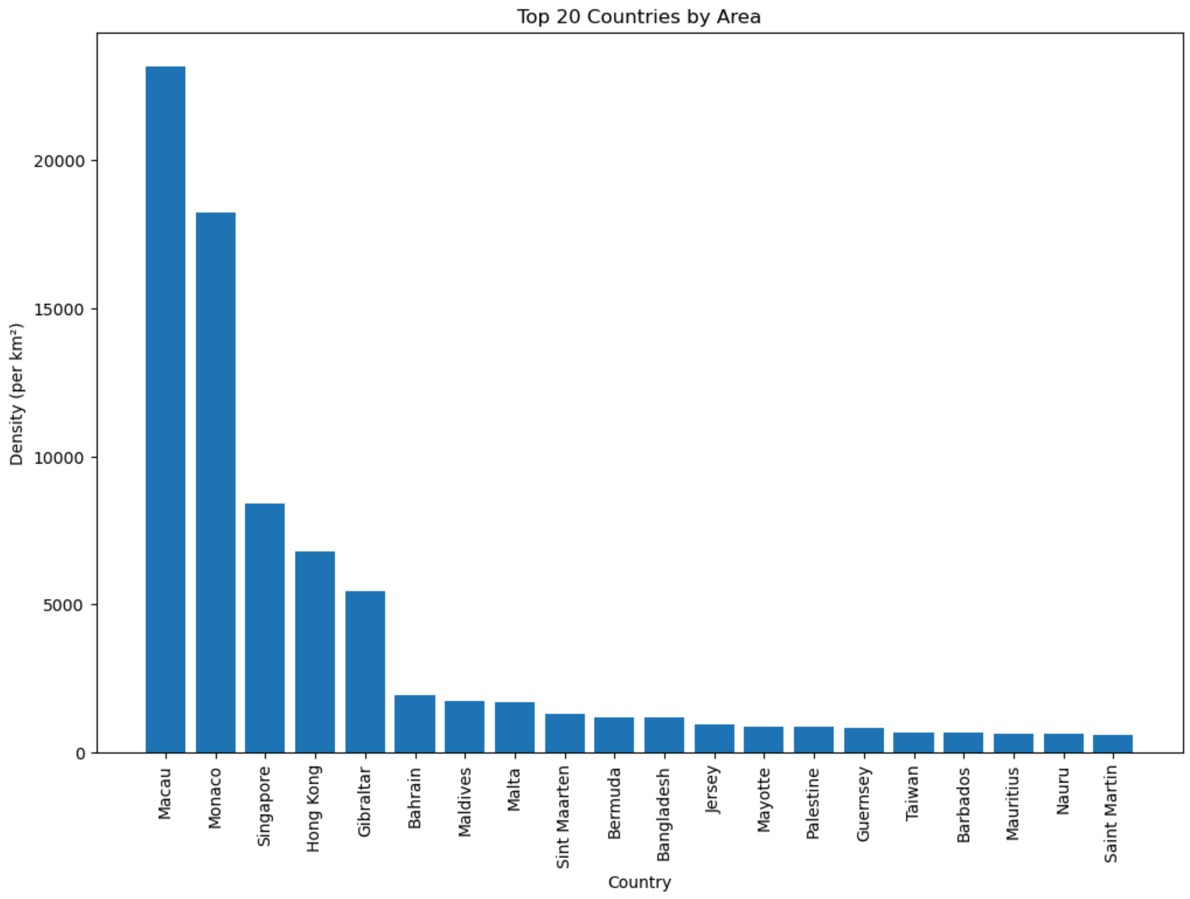
In [244…

```
sortedarea = worlddf.sort_values('Area (km²)', ascending=False).head(20)

# Plotting
plt.figure(figsize=(12, 8))
plt.bar(sortedarea['Country'], sortedarea['Area (km²)'], color='coral')
plt.xlabel('Country')
plt.ylabel('Area (km²)')
plt.title('Top 20 Countries by Area')
plt.xticks(rotation=90)
plt.show()
```



In [246…

```
sortedarea = worlddf.sort_values('Density (per km²)', ascending=False).head(20)

# Plotting
plt.figure(figsize=(12, 8))
plt.bar(sortedarea['Country'], sortedarea['Density (per km²)'], color='#1f77b4')
plt.xlabel('Country')
plt.ylabel('Density (per km²)')
plt.title('Top 20 Countries by Area')
plt.xticks(rotation=90)
plt.show()
```



10TH

In [235…

```
worldxl=pd.read_excel("world_population_excel_workbook.xlsx")
worldxl.head()
```

Out[235]:

	Rank	CCA3	Country	Capital	Continent	2022 Population	2020 Population	2015 Population	2010 Population
0	36	AFG	Afghanistan	Kabul	Asia	41128771	38972230	33753499	28189672
1	138	ALB	Albania	Tirana	Europe	2842321	2866849	2882481	2913399
2	34	DZA	Algeria	Algiers	Africa	44903225	43451666	39543154	35856344
3	213	ASM	American Samoa	Pago Pago	Oceania	44273	46189	51368	54849
4	203	AND	Andorra	Andorra la Vella	Europe	79824	77700	71746	71519

In [ ]: