```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

```

```
# Define file path
file_path = "/content/city-populations-to-2035.csv"

# Check if file exists
if not os.path.exists(file_path):
    print("Error: File not found. Please check the file path.")
else:
    # Read the dataset
    df = pd.read_csv(file_path)
    print("Dataset Loaded Successfully!")

```

```
Dataset Loaded Successfully!
```

```
# Display dataset info (Column names, Data types, Missing values)
print("Dataset Info:")
df.info()

```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 540 entries, 0 to 539
Data columns (total 5 columns):
 #   Column                                                     Non-Null Count  Dtype
---  ------                                                     --------------  -----
 0   Entity                                                     540 non-null    object
 1   Code                                                       0 non-null      float64
 2   Year                                                       540 non-null    int64
 3   City population (UN Urbanization Prospects, 2018)          420 non-null    float64
 4   Projected city population (UN Urbanization Prospects, 2018)  120 non-null    float64
dtypes: float64(3), int64(1), object(1)
memory usage: 21.2+ KB
```

```
print(f"Dataset Shape: {df.shape}")
print(f"Total Elements in Dataset: {df.size}")

```

```
Dataset Shape: (540, 5)
Total Elements in Dataset: 2700
```

```
print("Column Names:", df.columns.tolist())

```

```
Column Names: ['Entity', 'Code', 'Year', 'City population (UN Urbanization Prospects, 2018)', 'Projected city population (UN Urbanizatic
```

```
df.index
```

```
RangeIndex(start=0, stop=540, step=1)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
n=int(input("enter the number of rows u want to view from the top"))
df.head(n)
```

```
enter the number of rows u want to view from the top3
```

	Entity	Code	Year	City population (UN Urbanization Prospects, 2018)	Projected city population (UN Urbanization Prospects, 2018)
0	Bangalore	NaN	1950	745999.0	NaN
1	Bangalore	NaN	1955	939396.0	NaN
2	Bangalore	NaN	1960	1165978.0	NaN

```
df.dtypes
```

	0
Entity	object
Code	float64
Year	int64
City population (UN Urbanization Prospects, 2018)	float64
Projected city population (UN Urbanization Prospects, 2018)	float64

dtype: object

```
# Display the first five rows
df.head()

```

	Entity	Code	Year	City population (UN Urbanization Prospects, 2018)	Projected city population (UN Urbanization Prospects, 2018)
0	Bangalore	NaN	1950	745999.0	NaN
1	Bangalore	NaN	1955	939396.0	NaN
2	Bangalore	NaN	1960	1165978.0	NaN
3	Bangalore	NaN	1965	1377314.0	NaN
4	Bangalore	NaN	1970	1614756.0	NaN

```
df.describe()
```

	Code	Year	City population (UN Urbanization Prospects, 2018)	Projected city population (UN Urbanization Prospects, 2018)
count	0.0	540.00000	4.200000e+02	1.200000e+02
mean	NaN	1992.50000	8.449061e+06	1.939155e+07
std	NaN	25.96469	6.535627e+06	6.938304e+06
min	NaN	1950.00000	3.148000e+03	1.077049e+07
25%	NaN	1970.00000	3.394879e+06	1.436186e+07
50%	NaN	1992.50000	7.314234e+06	1.768104e+07
75%	NaN	2015.00000	1.181276e+07	2.281181e+07
max	NaN	2035.00000	3.725611e+07	4.334506e+07

```
# Rename columns for easier access
df.rename(columns={"City population (UN Urbanization Prospects, 2018)": "City Population"}, inplace=True)

# Drop rows where 'City Population' is missing
df_cleaned = df.dropna(subset=["City Population"])
print(f"Missing values handled. Remaining Rows: {df_cleaned.shape[0]}")

```

```
Missing values handled. Remaining Rows: 420
```

```
# Convert 'Year' column to integer
df_cleaned["Year"] = df_cleaned["Year"].astype(int)

# Display updated data types
df_cleaned.dtypes

```

```
<ipython-input-16-96cc6d7f92f2>: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-cc
  df_cleaned["Year"] = df_cleaned["Year"].astype(int)
```

	0
Entity	object
Code	float64
Year	int64
City Population	float64
Projected city population (UN Urbanization Prospects, 2018)	float64

dtype: object

```
# Remove duplicate rows
df_cleaned = df_cleaned.drop_duplicates()
print(f"Duplicate rows removed. Dataset now has {df_cleaned.shape[0]} rows.")
```

```
Duplicate rows removed. Dataset now has 420 rows.
```

```
df_2020 = df.loc[df["Year"] == 2020]
df_2020.head()
```

	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
14	Bangalore	NaN	2020	NaN	12326532.0
32	Beijing	NaN	2020	NaN	20462610.0
50	Bogota	NaN	2020	NaN	10978360.0
68	Buenos Aires	NaN	2020	NaN	15153729.0
86	Cairo	NaN	2020	NaN	20900604.0

```
df_large_cities = df.loc[df["City Population"] > 10_000_000]
df_large_cities.head()
```

	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
13	Bangalore	NaN	2015	10141080.0	NaN
28	Beijing	NaN	2000	10285091.0	NaN
29	Beijing	NaN	2005	12991292.0	NaN
30	Beijing	NaN	2010	16441252.0	NaN
31	Beijing	NaN	2015	18421198.0	NaN

```
df_mumbai = df.loc[df["Entity"] == "Mumbai"]
df_mumbai.head()
```

	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
360	Mumbai	NaN	1950	3088811.0	NaN
361	Mumbai	NaN	1955	3726210.0	NaN
362	Mumbai	NaN	1960	4414904.0	NaN
363	Mumbai	NaN	1965	5314300.0	NaN
364	Mumbai	NaN	1970	6412876.0	NaN

```
df.iloc[:, [0, 2, 3]]  # Retrieves the first, third, and fourth columns
```

	Entity	Year	City Population
0	Bangalore	1950	745999.0
1	Bangalore	1955	939396.0
2	Bangalore	1960	1165978.0
3	Bangalore	1965	1377314.0
4	Bangalore	1970	1614756.0
...	...	...	...
535	Tokyo	2015	37256109.0
536	Tokyo	2020	NaN
537	Tokyo	2025	NaN
538	Tokyo	2030	NaN
539	Tokyo	2035	NaN

540 rows × 3 columns

```
df.iloc[-5:, [0, 3]]  # Retrieves the last 5 rows and only the first and fourth columns

```

	Entity	City Population
535	Tokyo	37256109.0
536	Tokyo	NaN
537	Tokyo	NaN
538	Tokyo	NaN
539	Tokyo	NaN

```
df.loc[100:105]  # Retrieves rows from index 100 to 105 (inclusive)

```

	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
100	Chongqing	NaN	2000	7862976.0	NaN
101	Chongqing	NaN	2005	9454076.0	NaN
102	Chongqing	NaN	2010	11243667.0	NaN
103	Chongqing	NaN	2015	13372015.0	NaN
104	Chongqing	NaN	2020	NaN	15872179.0
105	Chongqing	NaN	2025	NaN	18171198.0

```
df["Entity"].value_counts()
# Returns a count of occurrences for each unique city

```

Entity	count
Bangalore	18
Beijing	18
Bogota	18
Buenos Aires	18
Cairo	18
Chongqing	18
Delhi	18
Dhaka	18
Guangzhou	18
Istanbul	18
Jakarta	18
Karachi	18
Kinshasa	18
Kolkata	18
Lagos	18
Lahore	18
Los Angeles	18
Manila	18
Mexico City	18
Moscow	18
Mumbai	18
New York	18
Osaka	18
Paris	18
Rio de Janeiro	18
Sao Paulo	18
Shanghai	18
Shenzhen	18
Tianjin	18
Tokyo	18

```
df.groupby("Year")["Entity"].nunique()
# Returns the count of unique cities for each year

```

Year	Entity
1950	30
1955	30
1960	30
1965	30
1970	30
1975	30
1980	30
1985	30
1990	30
1995	30
2000	30
2005	30
2010	30
2015	30
2020	30
2025	30
2030	30
2035	30

dtype: int64

```
df["Entity"].value_counts(normalize=True) * 100
```

Entity	proportion
Bangalore	3.333333
Beijing	3.333333
Bogota	3.333333
Buenos Aires	3.333333
Cairo	3.333333
Chongqing	3.333333
Delhi	3.333333
Dhaka	3.333333
Guangzhou	3.333333
Istanbul	3.333333
Jakarta	3.333333
Karachi	3.333333
Kinshasa	3.333333
Kolkata	3.333333
Lagos	3.333333
Lahore	3.333333
Los Angeles	3.333333
Manila	3.333333
Mexico City	3.333333
Moscow	3.333333
Mumbai	3.333333
New York	3.333333
Osaka	3.333333
Paris	3.333333
Rio de Janeiro	3.333333
Sao Paulo	3.333333
Shanghai	3.333333
Shenzhen	3.333333
Tianjin	3.333333
Tokyo	3.333333

```
df_sorted = df.sort_values(by=["Year", "City Population"], ascending=[False, True])
df_sorted.head()

```

	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
17	Bangalore	NaN	2035	NaN	18065541.0
35	Beijing	NaN	2035	NaN	25365920.0
53	Bogota	NaN	2035	NaN	12753324.0
71	Buenos Aires	NaN	2035	NaN	17127741.0
89	Cairo	NaN	2035	NaN	28504351.0

```
# Identify and remove outliers using the Interquartile Range (IQR) method
q1 = df_cleaned["City Population"].quantile(0.25)
q3 = df_cleaned["City Population"].quantile(0.75)
iqr = q3 - q1

# Filter out outliers
df_filtered = df_cleaned[(df_cleaned["City Population"] >= q1 - 1.5 * iqr) &
                         (df_cleaned["City Population"] <= q3 + 1.5 * iqr)]
```

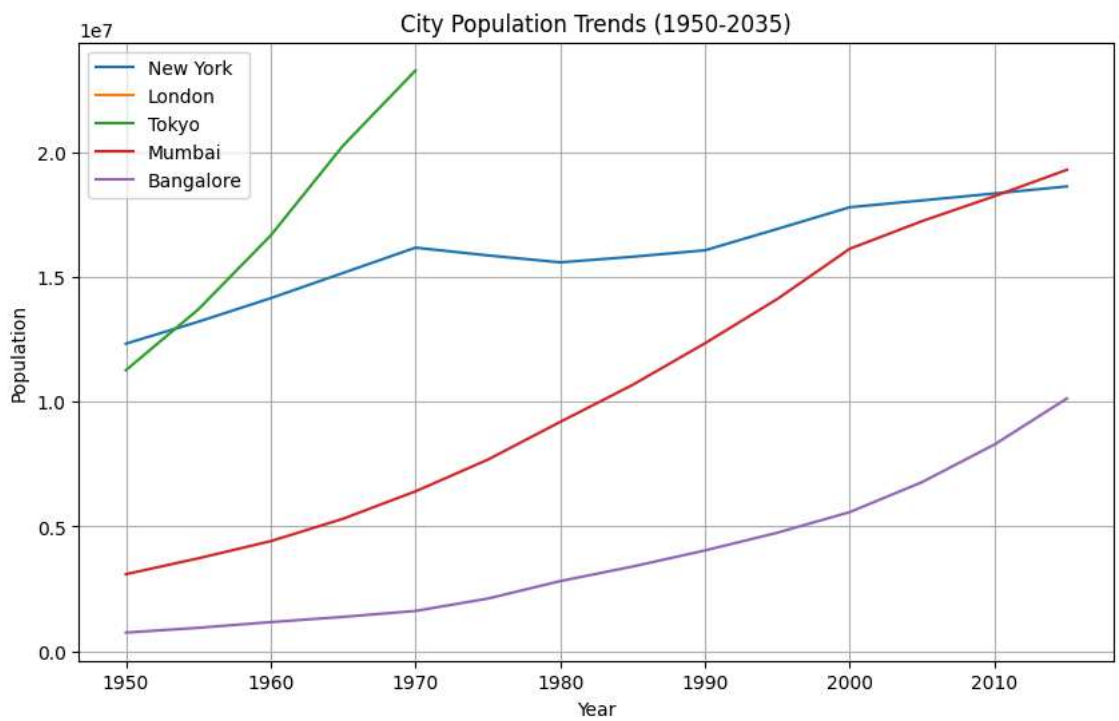
```

print(f"Outliers removed. Remaining Rows: {df_filtered.shape[0]}")

```

```
Outliers removed. Remaining Rows: 410
```

```
# Select specific cities for visualization
selected_cities = ["New York", "London", "Tokyo", "Mumbai", "Bangalore"]
df_selected = df_filtered[df_filtered["Entity"].isin(selected_cities)]

print(f"Filtered Data for Selected Cities: {df_selected.shape[0]} rows")

```

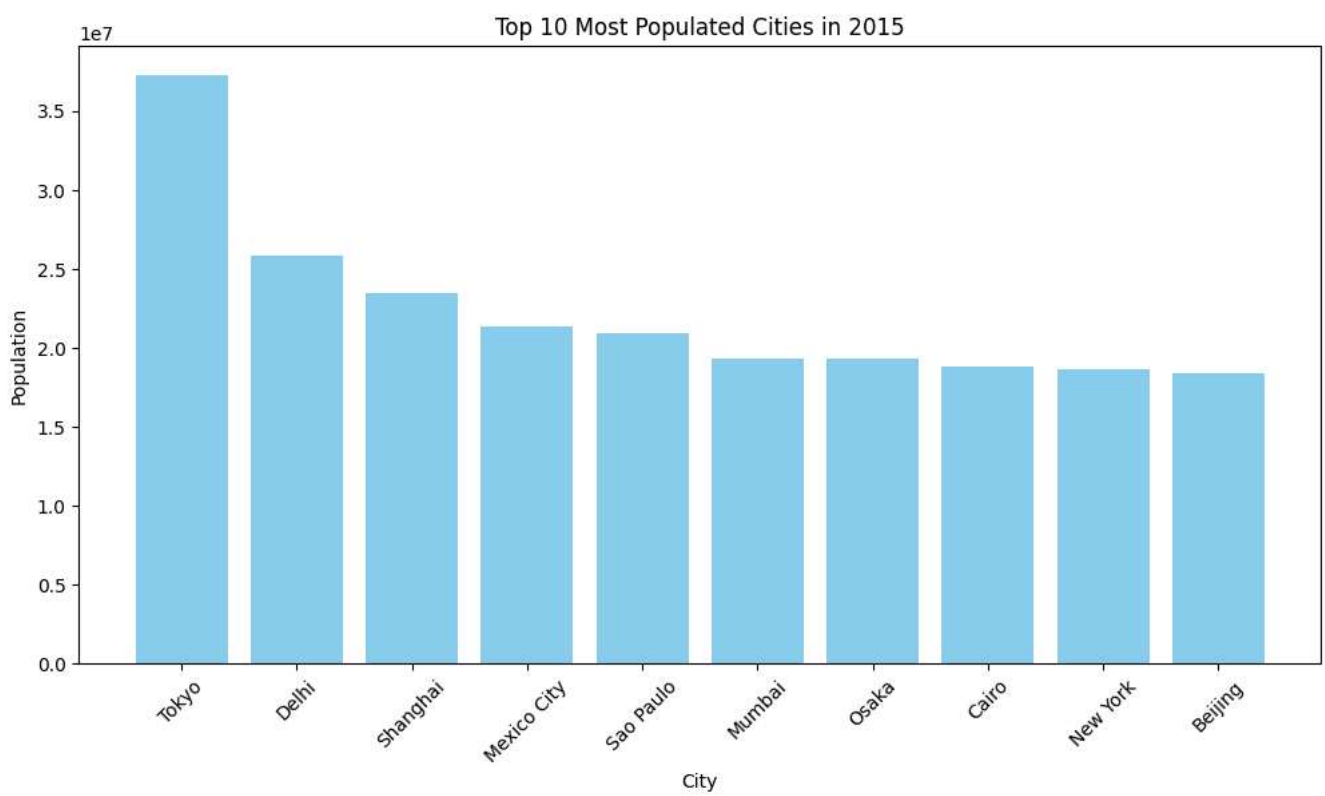
```
Filtered Data for Selected Cities: 47 rows
```

```
# Plot city population trends over time
plt.figure(figsize=(10, 6))

for city in selected_cities:
    city_data = df_selected[df_selected["Entity"] == city]
    plt.plot(city_data["Year"], city_data["City Population"], label=city)

# Customize the plot
plt.xlabel("Year")
plt.ylabel("Population")
plt.title("City Population Trends (1950-2035)")
plt.legend()
plt.grid(True)

# Show the plot
plt.show()

```

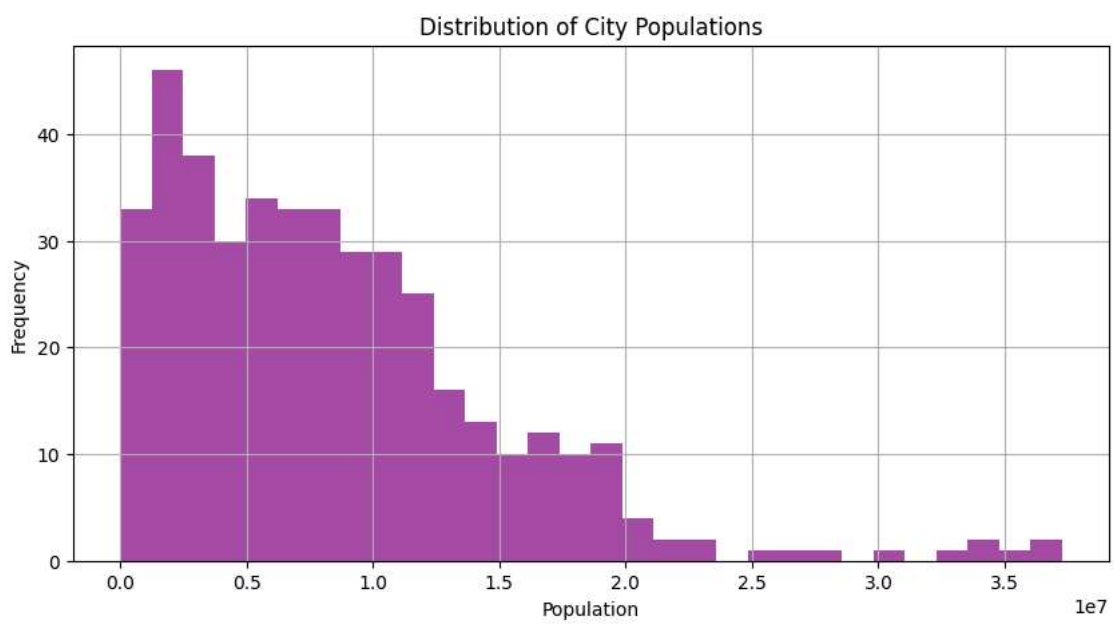



```
# Ensure valid data for the latest available year
latest_year = df["Year"].max()
df_latest = df[df["Year"] == latest_year].dropna(subset=["City Population"])

# If df_latest is still empty, use a previous year with valid data
if df_latest.empty:
    valid_years = df.dropna(subset=["City Population"])["Year"].unique()
    latest_year = max(valid_years)  # Find the last available year with data
    df_latest = df[df["Year"] == latest_year]

plt.figure(figsize=(12, 6))
top_cities = df_latest.nlargest(10, "City Population")
```

```
plt.bar(top_cities["Entity"], top_cities["City Population"], color='skyblue')
plt.xlabel("City")
plt.ylabel("Population")
plt.title(f"Top 10 Most Populated Cities in {latest_year}")
plt.xticks(rotation=45)
plt.show()
```

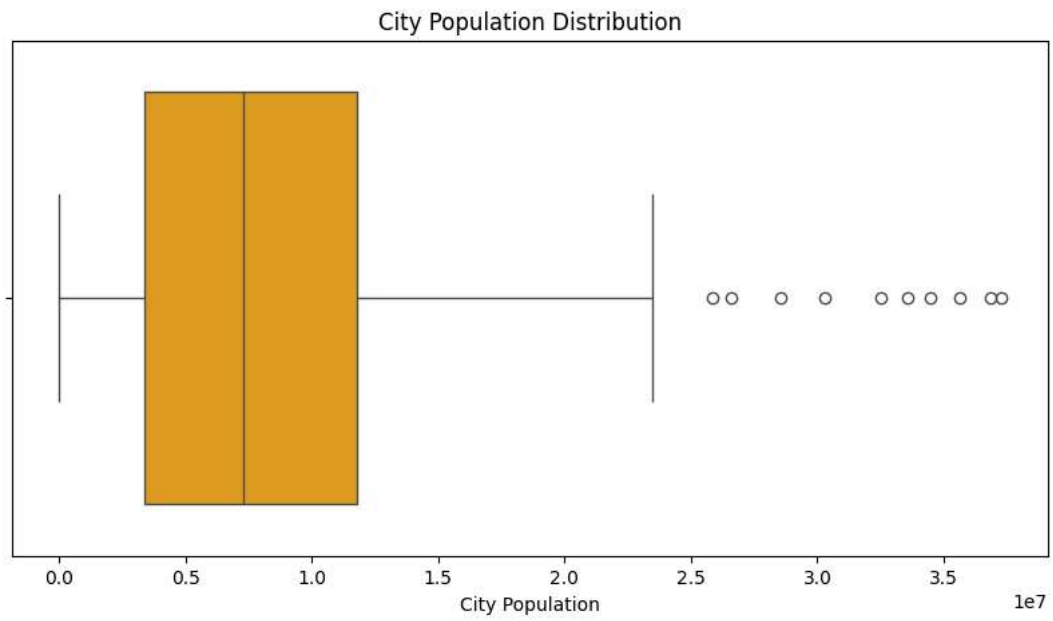



```
plt.figure(figsize=(10, 5))
plt.hist(df["City Population"], bins=30, color='purple', alpha=0.7)
plt.xlabel("Population")
plt.ylabel("Frequency")
plt.title("Distribution of City Populations")
plt.grid(True)
plt.show()
```

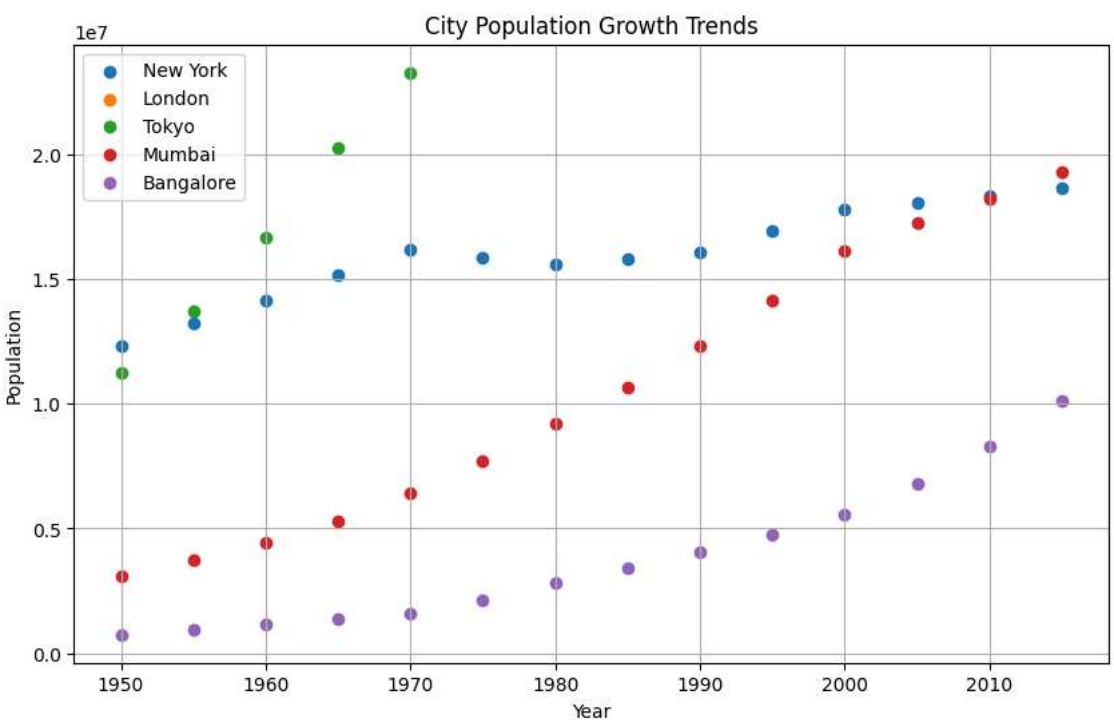



```
plt.figure(figsize=(10, 5))
sns.boxplot(x=df["City Population"], color='orange')
```

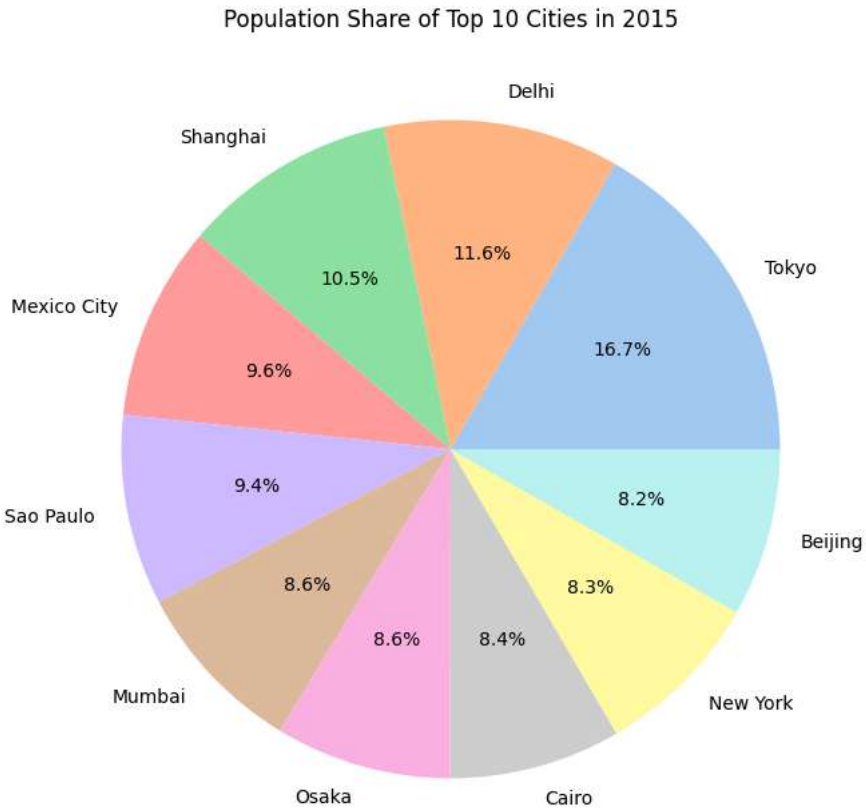
```
plt.title("City Population Distribution")
plt.show()
```



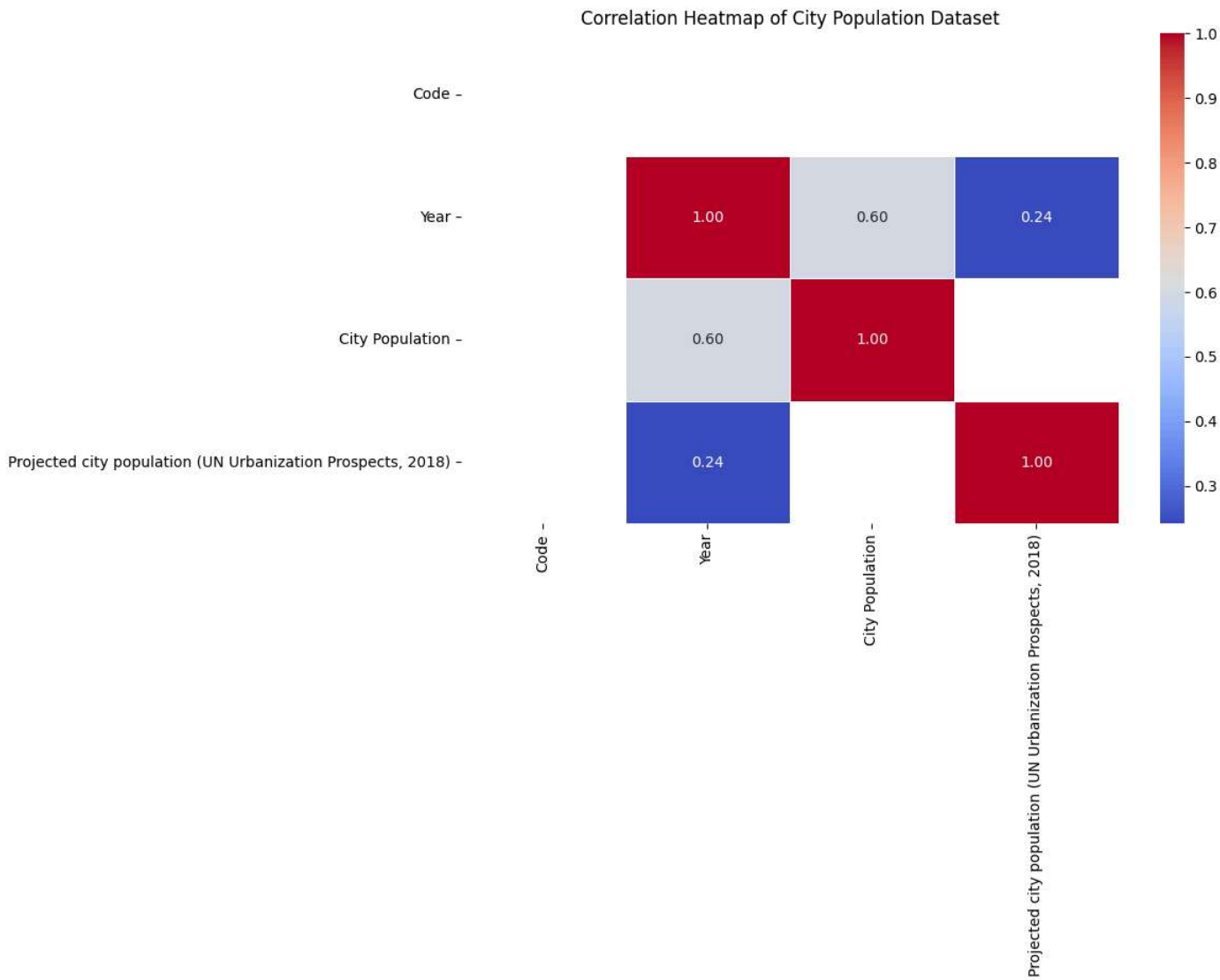

```
plt.figure(figsize=(10, 6))
for city in selected_cities:
    city_data = df_selected[df_selected["Entity"] == city]
    plt.scatter(city_data["Year"], city_data["City Population"], label=city)
plt.xlabel("Year")
plt.ylabel("Population")
plt.title("City Population Growth Trends")
plt.legend()
plt.grid()
plt.show()
```



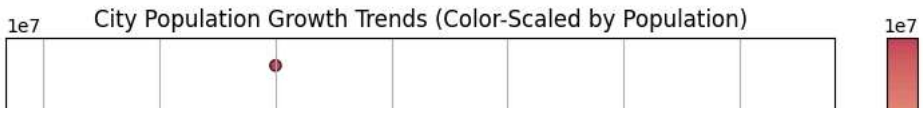

```
plt.figure(figsize=(8, 8))
plt.pie(top_cities["City Population"], labels=top_cities["Entity"], autopct='%1.1f%%', colors=sns.color_palette("pastel"))
plt.title(f"Population Share of Top 10 Cities in {latest_year}")
plt.show()
```




```
import seaborn as sns
import numpy as np


correlation_matrix = df.corr(numeric_only=True)

plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap of City Population Dataset")
plt.show()

```




```
plt.figure(figsize=(10, 6))


scatter = plt.scatter(df_selected["Year"], df_selected["City Population"],
                      c=df_selected["City Population"], cmap='coolwarm', alpha=0.75, edgecolors='k')

cbar = plt.colorbar(scatter)
cbar.set_label("Population")

plt.xlabel("Year")
plt.ylabel("Population")
plt.title("City Population Growth Trends (Color-Scaled by Population)")

plt.grid(True)
plt.show()
```


City Population Growth Trends (Color-Scaled by Population)
1e7
1e7