Introduction

Our dataset on global blood group distributions by country presents an intriguing puzzle: the percentages of blood types vary across regions in ways that prompt many questions about genetics, migration, and environmental influences. If you find these insights helpful, feel free to upvote this notebook.

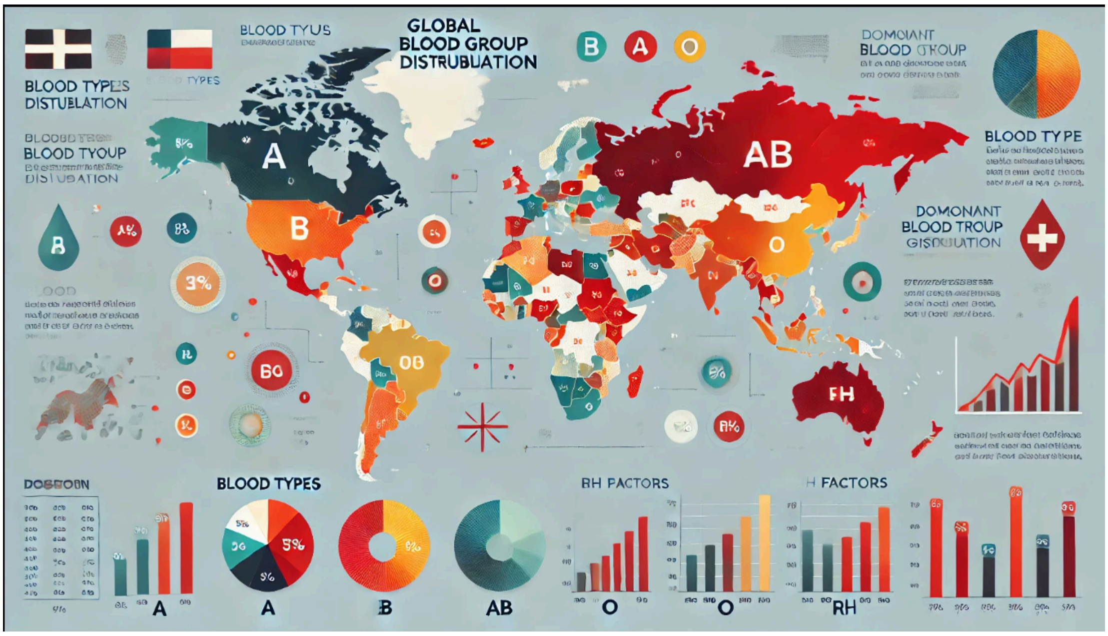


Table of Contents

In [4]:
```python
# Suppress warnings for cleaner notebook output
import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Loading

In this section, we load the cleaned blood group distribution dataset. We use the cleaned_blood_type_distribution_by_country.csv file which has resolved some of the obvious issues found in other versions.

In [9]:
```
df=pd.read_csv(r"C:\Users\chitt\Downloads\cleaned_blood_type_distribution_by_cou
df
```

Out[9]:

	Country/Dependency	Population	O+	A+	B+	AB+	O-	A-
0	Albania	3,074,579	34.10%	31.20%	14.50%	5.20%	6.00%	5.50%
1	Algeria	43,576,691	40.00%	30.00%	15.00%	4.25%	6.60%	2.30%
2	Argentina	45,479,118	50.34%	31.09%	8.20%	2.16%	4.29%	2.98%
3	Armenia	3,021,324	29.00%	46.30%	12.00%	5.60%	2.00%	3.70%
4	Australia	25,466,459	38.00%	32.00%	12.00%	4.00%	7.00%	6.00%
...	...	...	...	...	...	...	...	...
121	Venezuela	28,644,603	58.30%	28.20%	5.60%	1.90%	4.00%	1.50%
122	Vietnam	98,721,275	41.70%	20.90%	30.80%	4.98%	0.30%	0.10%
123	Yemen	29,884,405	47.84%	27.50%	15.32%	2.14%	3.66%	2.10%
124	Zimbabwe	14,546,314	36.40%	29.30%	8.10%	2.00%	14.10%	8.10%
125	World	7,772,850,805	38.40%	27.30%	8.10%	2.00%	14.10%	8.10%

126 rows × 10 columns

In [11]:
```
df.head()
```

Out[11]:

	Country/Dependency	Population	O+	A+	B+	AB+	O-	A-	
0	Albania	3,074,579	34.10%	31.20%	14.50%	5.20%	6.00%	5.50%	2.6(
1	Algeria	43,576,691	40.00%	30.00%	15.00%	4.25%	6.60%	2.30%	1.1(
2	Argentina	45,479,118	50.34%	31.09%	8.20%	2.16%	4.29%	2.98%	0.7
3	Armenia	3,021,324	29.00%	46.30%	12.00%	5.60%	2.00%	3.70%	1.0(
4	Australia	25,466,459	38.00%	32.00%	12.00%	4.00%	7.00%	6.00%	2.0(

In [13]:
```
df.tail()
```

Out[13]:

	Country/Dependency	Population	O+	A+	B+	AB+	O-	A-
121	Venezuela	28,644,603	58.30%	28.20%	5.60%	1.90%	4.00%	1.50%
122	Vietnam	98,721,275	41.70%	20.90%	30.80%	4.98%	0.30%	0.10%
123	Yemen	29,884,405	47.84%	27.50%	15.32%	2.14%	3.66%	2.10%
124	Zimbabwe	14,546,314	36.40%	29.30%	8.10%	2.00%	14.10%	8.10%
125	World	7,772,850,805	38.40%	27.30%	8.10%	2.00%	14.10%	8.10%

In [17]:
```
df.shape
```

Out[17]:
```
(126, 10)
```

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126 entries, 0 to 125
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Country/Dependency  126 non-null    object
 1   Population          126 non-null    object
 2   O+                  126 non-null    object
 3   A+                  126 non-null    object
 4   B+                  126 non-null    object
 5   AB+                 125 non-null    object
 6   O-                  125 non-null    object
 7   A-                  125 non-null    object
 8   B-                  125 non-null    object
 9   AB-                 125 non-null    object
dtypes: object(10)
memory usage: 10.0+ KB
```

In [23]:
```
df.describe()
```

Out[23]:

	Country/Dependency	Population	O+	A+	B+	AB+	O-	A-
count	126	126	126	126	126	125	125	125
unique	126	126	99	94	94	77	80	72
top	Albania	3,074,579	35.00%	37.00%	15.00%	4.00%	5.00%	6.00%
freq	1	1	5	7	6	13	13	17

Data Cleaning and Preprocessing

The data contains string representations of numerical values. Population values use commas as thousand separators, and blood group percentages include the percent symbol. We will remove these characters so that we can convert these strings into floats. Note the improvement over previous attempts: we address errors such as ValueError: could not convert string to float: '34.10%' by stripping both commas and percent signs.

In [28]:
```
# Create a copy of the cleaned dataframe for our analysis
df = df.copy()

# List of columns to convert to numeric. 'Population' has commas and the blood g
cols_to_convert = ['Population', 'O+', 'A+', 'B+', 'AB+', 'O-', 'A-', 'B-', 'AB-

for col in cols_to_convert:
    # Remove commas and the percent sign, then convert to float
    df[col] = df[col].str.replace(',', '', regex=False).str.replace('%', '', reg
# Display data types after conversion to verify successful transformation
print('Data types after conversion:')
```

```
print(df.dtypes)

# Check for missing values in each column
print('\nMissing values in each column:')
print(df.isna().sum())
```

```
Data types after conversion:
Country/Dependency     object
Population            float64
O+                    float64
A+                    float64
B+                    float64
AB+                   float64
O-                    float64
A-                    float64
B-                    float64
AB-                   float64
dtype: object

Missing values in each column:
Country/Dependency    0
Population            0
O+                    0
A+                    0
B+                    0
AB+                   1
O-                    1
A-                    1
B-                    1
AB-                   1
dtype: int64
```

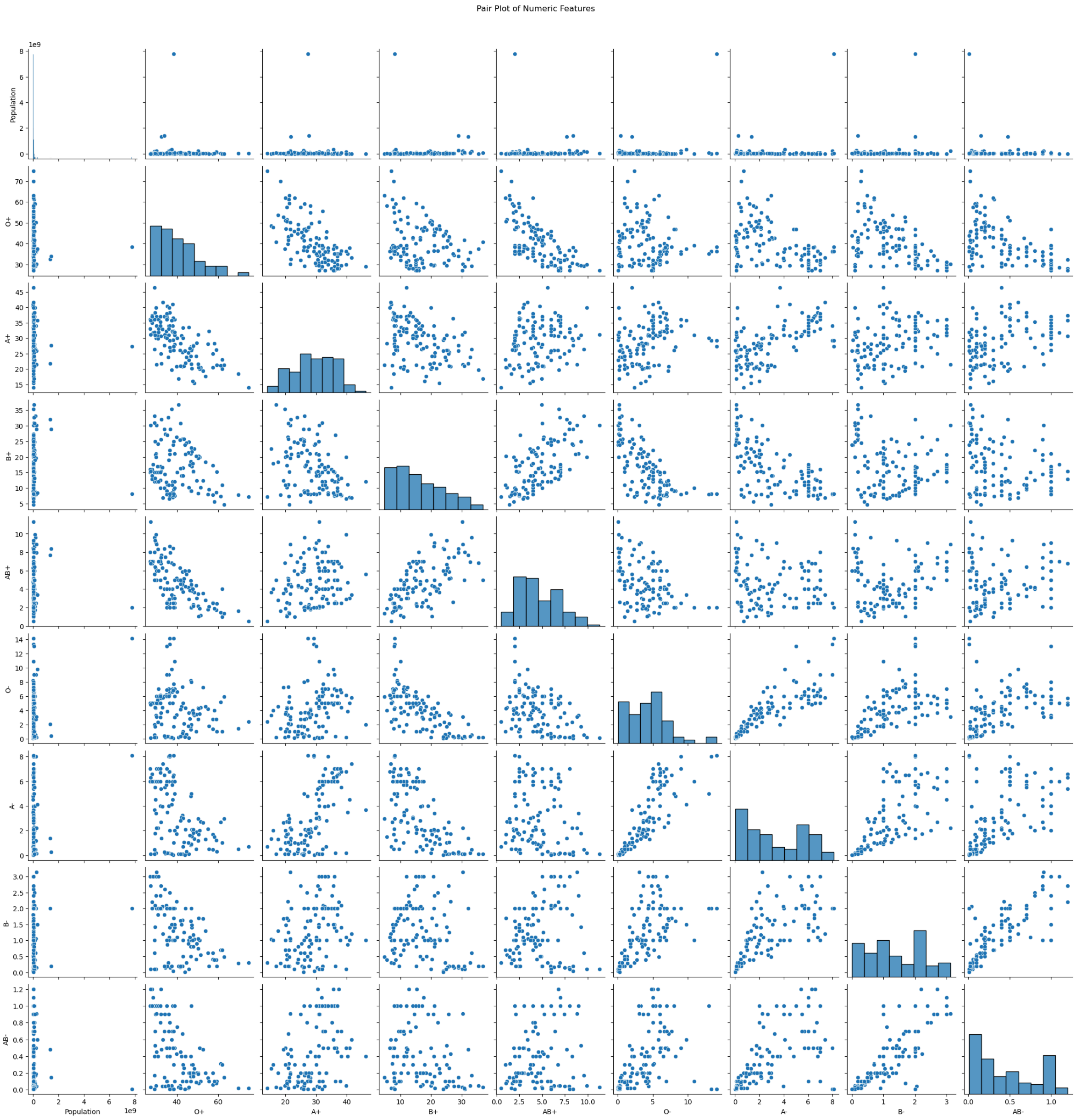
Exploratory Data Analysis

Below are several visualizations to understand our data better.

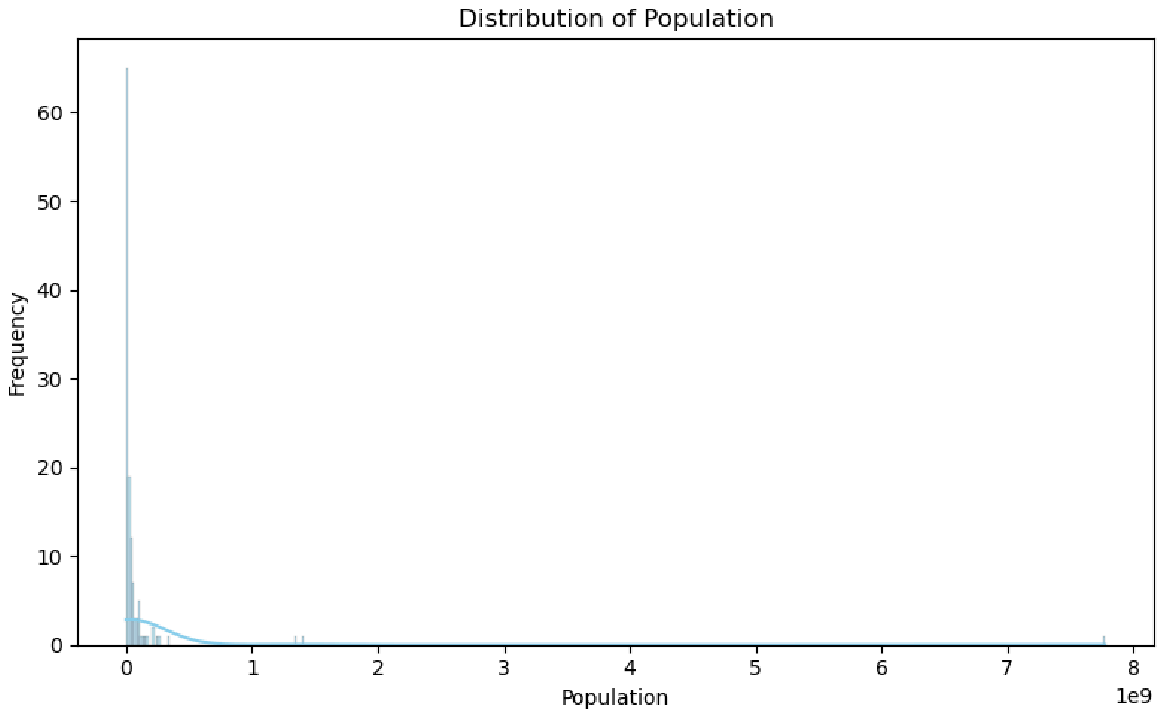
1. A correlation heatmap will help us see how the blood type percentages and population correlate.
2. A pair plot to inspect pairwise relationships among numeric variables.
3. Histograms, box plots, and bar plots to explore the distributions of the features.

This multifaceted approach ensures we explore different angles of the dataset.

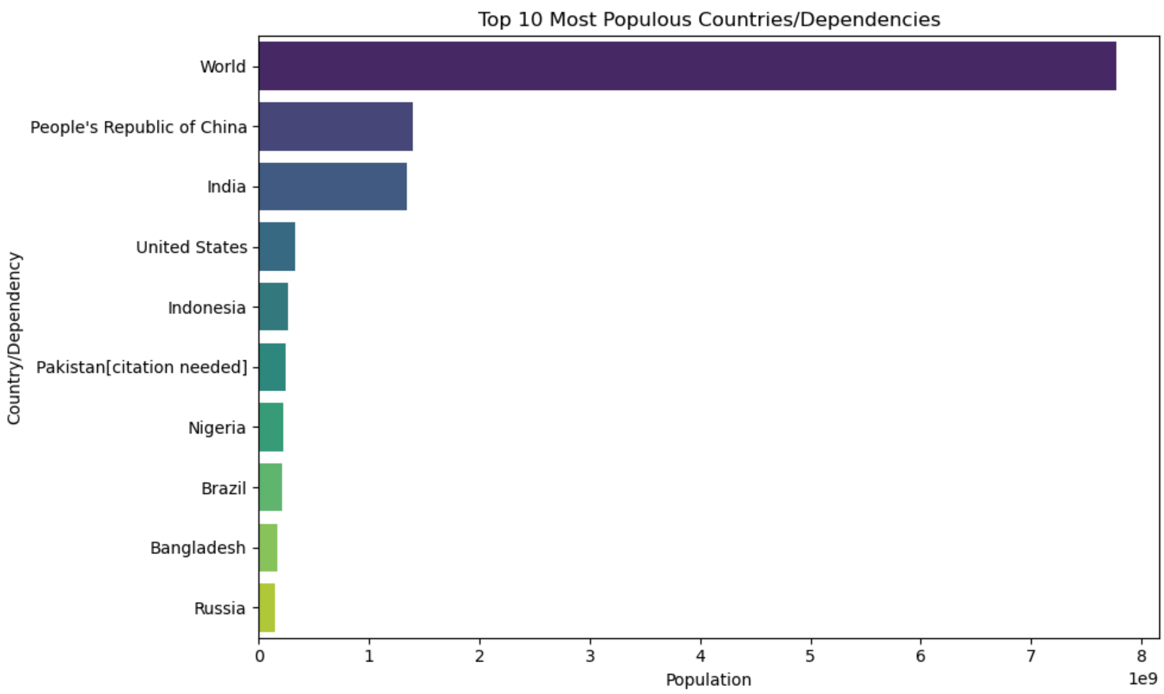
In [31]:
```
# Create pair plots
sns.pairplot(df)
plt.suptitle('Pair Plot of Numeric Features', y=1.02)
plt.show()
```



In [33]:
```
# Histogram of Population
plt.figure(figsize=(8, 5))
sns.histplot(df['Population'], kde=True, color='skyblue')
plt.title('Distribution of Population')
plt.xlabel('Population')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

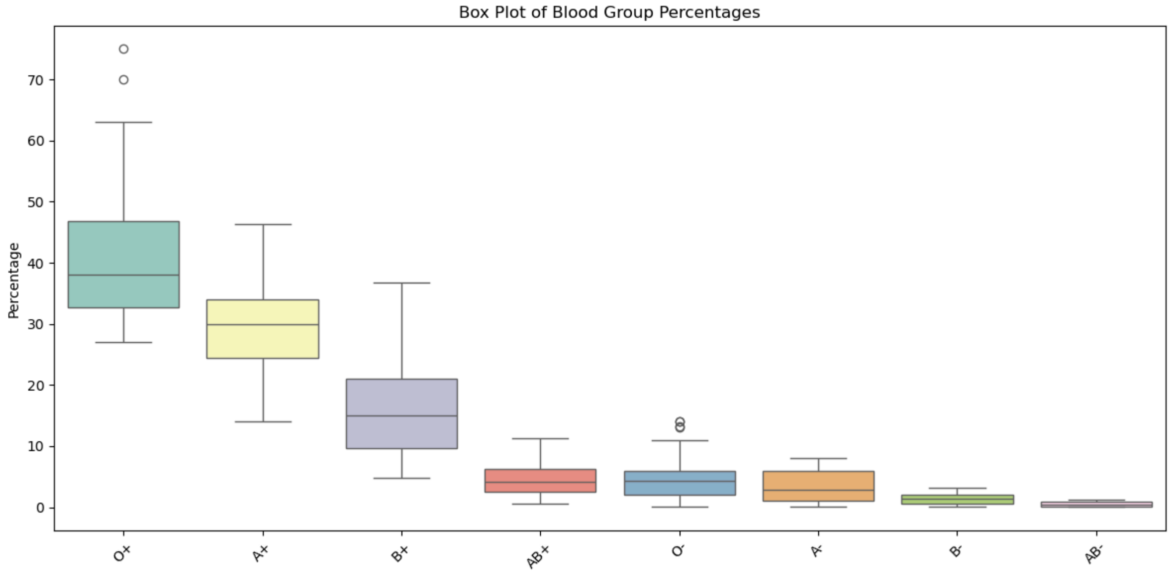


In [35]:
```
# Bar plot for the top 10 most populous countries
top10 = df.sort_values('Population', ascending=False).head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x='Population', y='Country/Dependency', data=top10, palette='viridis
plt.title('Top 10 Most Populous Countries/Dependencies')
plt.xlabel('Population')
plt.ylabel('Country/Dependency')
plt.tight_layout()
plt.show()
```



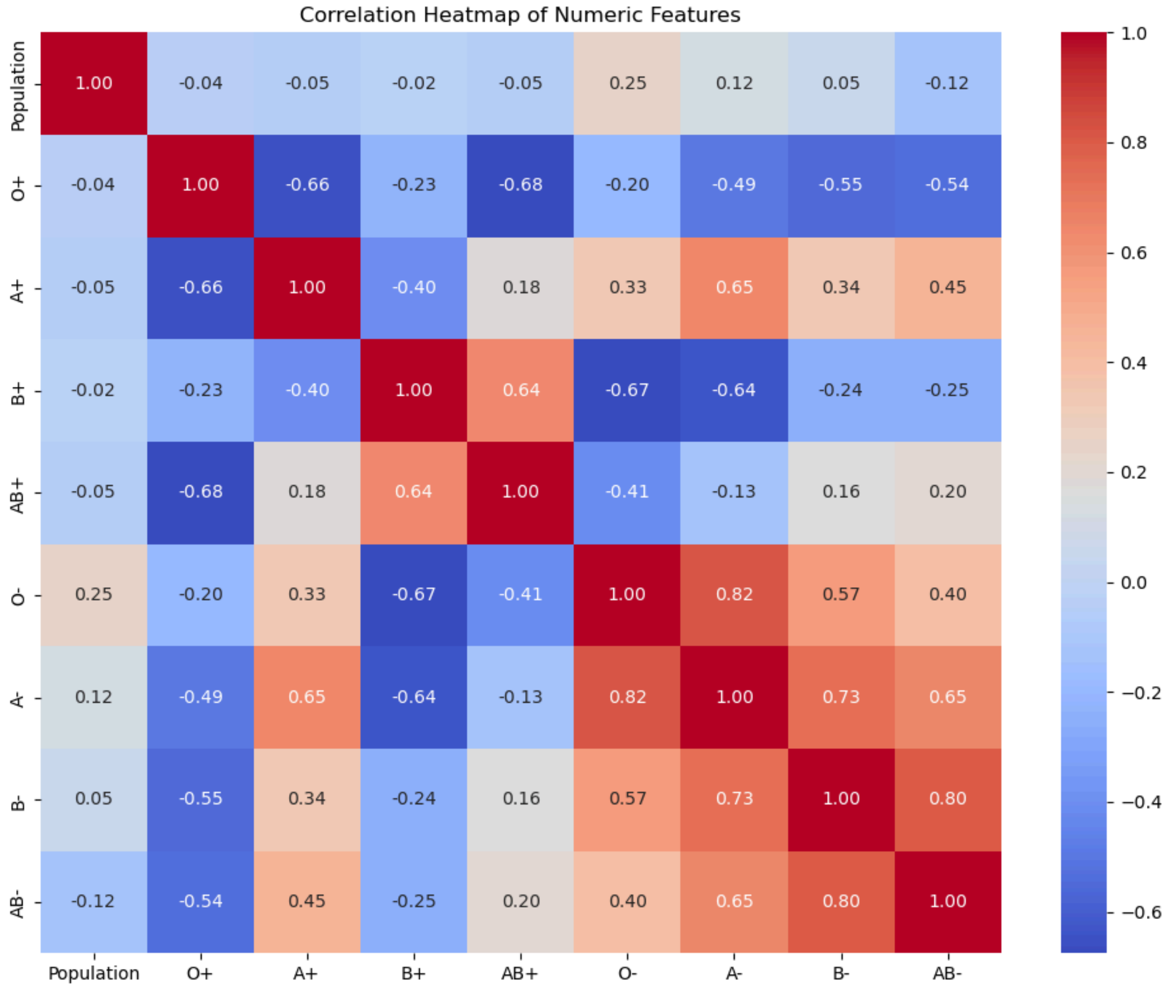
In [37]:
```
# Box plot for blood group percentages
blood_groups = ['O+', 'A+', 'B+', 'AB+', 'O-', 'A-', 'B-', 'AB-']
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[blood_groups], palette='Set3')
plt.title('Box Plot of Blood Group Percentages')
plt.ylabel('Percentage')
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [41]:
```
numeric_df = df.select_dtypes(include=[np.number])

if numeric_df.shape[1] >= 4:
    plt.figure(figsize=(10, 8))
    corr = numeric_df.corr()
    sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
    plt.title('Correlation Heatmap of Numeric Features')
    plt.tight_layout()
    plt.show()
```



Predictor Development

While our analysis has already unearthed interesting relationships, we can also try our hand at prediction. In this notebook, we attempt to predict the O+ blood group percentage for each country from the remaining blood group percentages and the population. We use a simple linear regression model for this purpose and report the R^2 score as a measure of prediction accuracy.

Multicollinearity and the fact that blood group percentages add up to nearly 100 may restrict the power of our predictor, but this exercise is useful for demonstrating regression modeling on real-world data.

In [44]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [46]:
```
# First, drop rows with missing values (if any)
df_model = df.dropna()
```

In [48]:
```
# Define features (exclude 'O+' as it is our target) and target
feature_cols = ['Population', 'A+', 'B+', 'AB+', 'O-', 'A-', 'B-', 'AB-']
X = df_model[feature_cols]
y = df_model['O+']
```

In [50]:
```
# Split the data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [52]:
```
# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

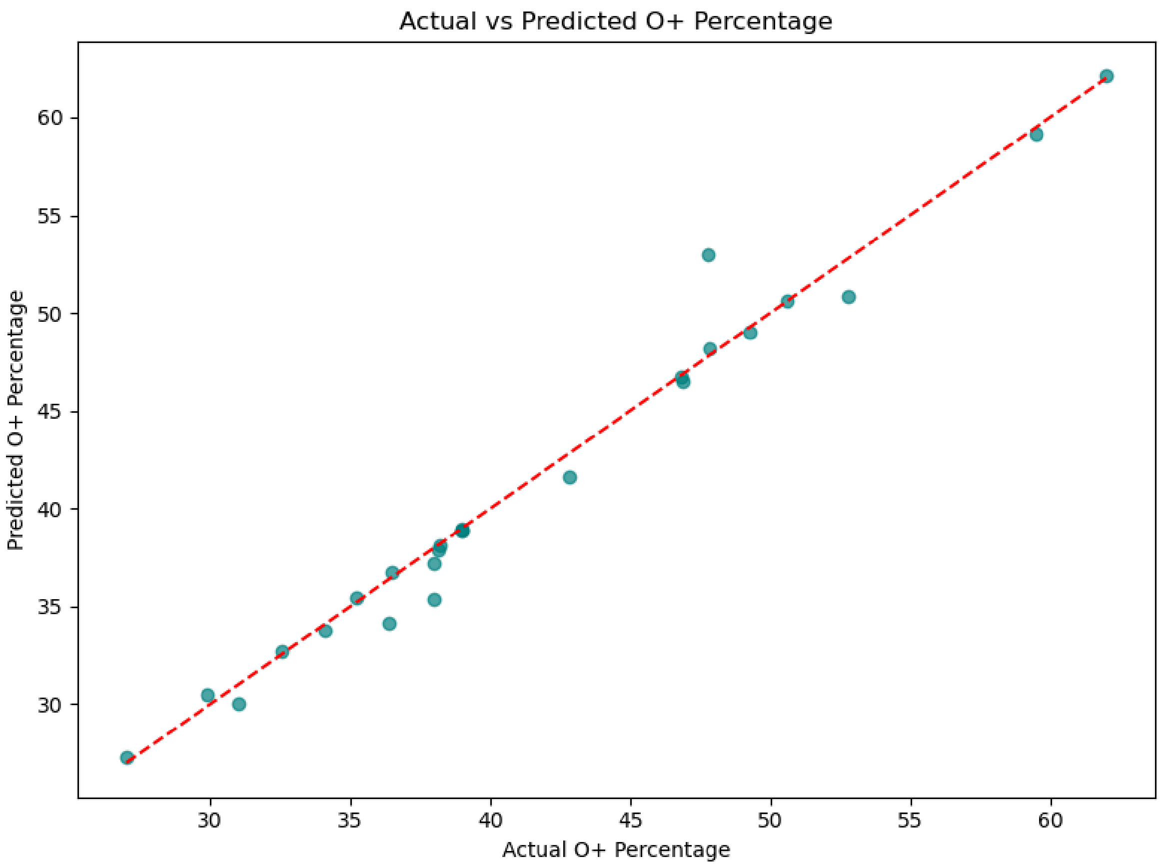
Out[52]:
```
LinearRegression
LinearRegression()
```

In [54]:
```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

In [56]:
```
# Evaluate the model using R^2 score
score = r2_score(y_test, y_pred)
print('R^2 Score for predicting O+ percentage:', score)
```

```
R^2 Score for predicting O+ percentage: 0.9750297831024386
```

In [58]:
```
# Optional: Plotting the predicted vs actual values
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.7, color='teal')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', li
plt.xlabel('Actual O+ Percentage')
plt.ylabel('Predicted O+ Percentage')
plt.title('Actual vs Predicted O+ Percentage')
plt.tight_layout()
plt.show()
```



Summary and Future Work

In this notebook, we took a deep dive into the global blood group distribution dataset. Our analysis involved data cleaning—especially the careful removal of commas and percentage symbols to correctly convert values to numerical types—and a suite of exploratory visualizations that laid bare the relationships among the various blood group percentages and population.

A simple linear regression model was built to predict the O+ blood group percentage, achieving an R^2 score that provides a preliminary gauge of its predictive power. Future explorations could include:

Trying more complex models or regularization techniques to account for multicollinearity among blood type percentages. Exploring geospatial visualizations and clustering to discover regional groupings. Leveraging domain-specific knowledge to inform feature engineering and improve model performance.

This notebook demonstrates the versatility of applied data science, from thorough data cleaning to model development. If you found this analysis insightful, please consider upvoting it.

In [ ]:

```

```