PROJECT 2

Border Crossing Analysis

In [1]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Load the dataset
file_path = "Border_Crossing_Entry_Data.csv"
df = pd.read_csv(file_path)

# Display basic information about the dataset
df.info(), df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398658 entries, 0 to 398657
Data columns (total 10 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   Port Name   398658 non-null  object 
 1   State       398658 non-null  object 
 2   Port Code   398658 non-null  int64  
 3   Border      398658 non-null  object 
 4   Date        398658 non-null  object 
 5   Measure     398658 non-null  object 
 6   Value       398658 non-null  int64  
 7   Latitude    398658 non-null  float64
 8   Longitude   398658 non-null  float64
 9   Point       398658 non-null  object 
dtypes: float64(2), int64(2), object(6)
memory usage: 30.4+ MB
```

Out[1]:
```
(None,
       Port Name      State  Port Code             Border      Date  \
 0       Jackman      Maine        104  US-Canada Border  Jan 2024
 1      Porthill      Idaho       3308  US-Canada Border  Apr 2024
 2      San Luis    Arizona       2608  US-Mexico Border  Apr 2024
 3  Willow Creek    Montana       3325  US-Canada Border  Jan 2024
 4       Warroad  Minnesota       3423  US-Canada Border  Jan 2024

                        Measure  Value  Latitude  Longitude  \
 0                        Trucks   6556    45.806    -70.397
 1                        Trucks     98    49.000   -116.499
 2                         Buses     10    32.485   -114.782
 3                   Pedestrians      2    49.000   -109.731
 4  Personal Vehicle Passengers   9266    48.999    -95.377

                             Point
 0   POINT (-70.396722 45.805661)
 1   POINT (-116.49925 48.999861)
 2    POINT (-114.7822222 32.485)
 3  POINT (-109.731333 48.999972)
 4      POINT (-95.376555 48.999)  )
```

In [2]:
```
# Convert Date column to datetime format
df['Date'] = pd.to_datetime(df['Date'], format='%b %Y')

# Check for missing values
missing_values = df.isnull().sum()

# Check for duplicates
duplicates = df.duplicated().sum()

# Display results
missing_values, duplicates
```

Out[2]:
```
(Port Name    0
 State        0
 Port Code    0
 Border       0
 Date         0
 Measure      0
 Value        0
 Latitude     0
 Longitude    0
 Point        0
 dtype: int64,
 10)
```

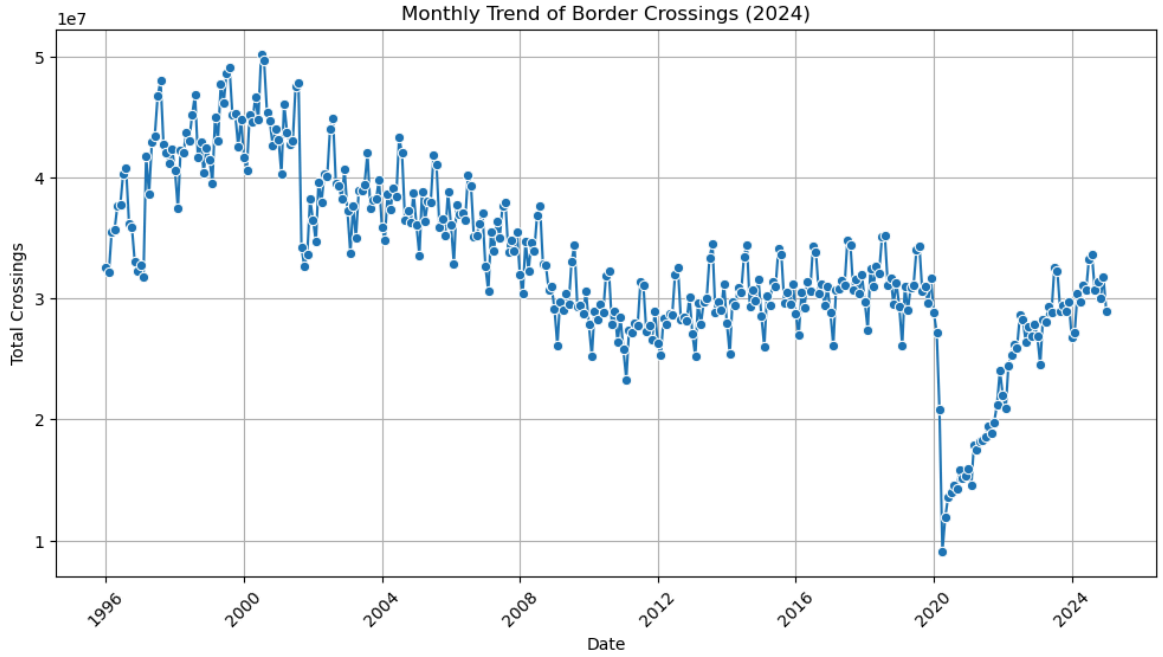
In [3]:
```
# Remove duplicate rows
df_cleaned = df.drop_duplicates().copy()  # Ensure it's a full copy

# Ensure categorical variables are in correct format using .loc[]
categorical_cols = ['Port Name', 'State', 'Border', 'Measure']
df_cleaned.loc[:, categorical_cols] = df_cleaned[categorical_cols].astype('ca

# Verify changes
df_cleaned.info(), df_cleaned.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 398648 entries, 0 to 398657
Data columns (total 10 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Port Name  398648 non-null  object
 1   State      398648 non-null  object
 2   Port Code  398648 non-null  int64
 3   Border     398648 non-null  object
 4   Date       398648 non-null  datetime64[ns]
 5   Measure    398648 non-null  object
 6   Value      398648 non-null  int64
 7   Latitude   398648 non-null  float64
 8   Longitude  398648 non-null  float64
 9   Point      398648 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(2), object(5)
memory usage: 33.5+ MB
```

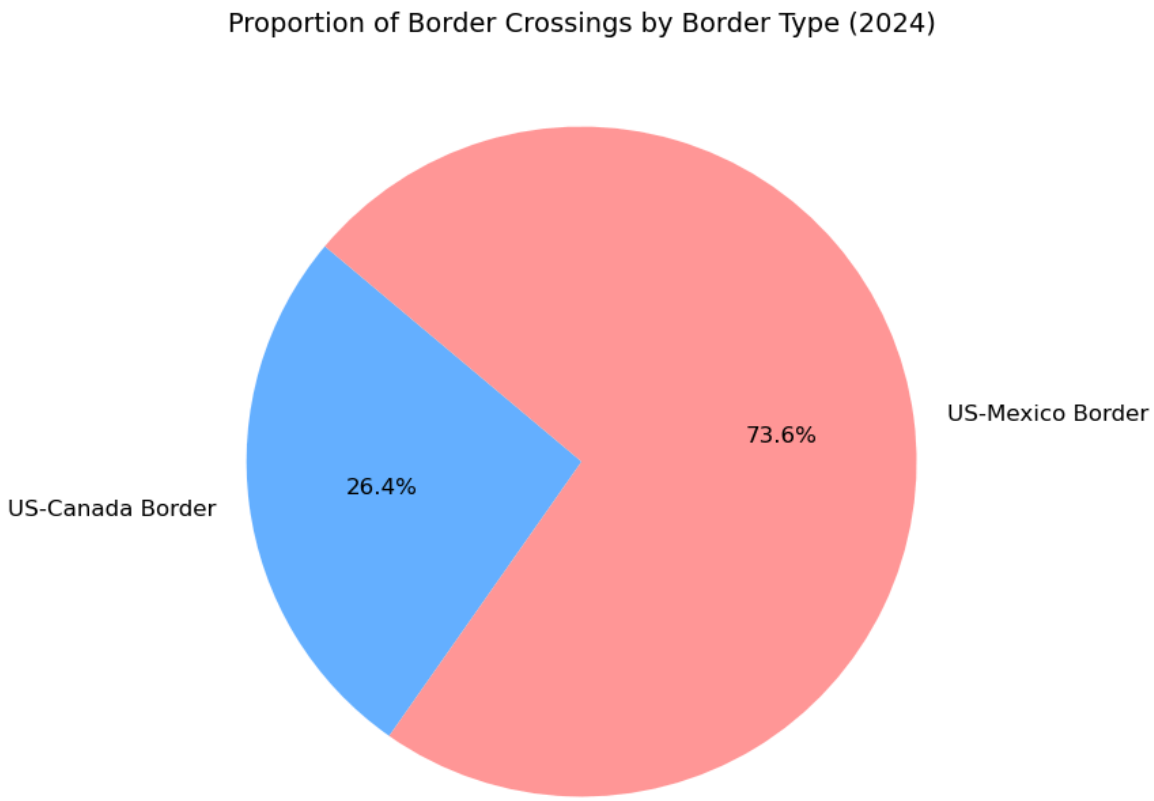
Out[3]:
```
(None,
       Port Name      State  Port Code            Border       Date  \
 0       Jackman      Maine        104  US-Canada Border 2024-01-01
 1      Porthill      Idaho       3308  US-Canada Border 2024-04-01
 2      San Luis    Arizona       2608  US-Canada Border 2024-04-01
 3  Willow Creek    Montana       3325  US-Canada Border 2024-01-01
 4       Warroad  Minnesota       3423  US-Canada Border 2024-01-01

                        Measure  Value  Latitude  Longitude  \
 0                       Trucks   6556    45.806    -70.397
 1                       Trucks     98    49.000   -116.499
 2                        Buses     10    32.485   -114.782
 3                  Pedestrians      2    49.000   -109.731
 4  Personal Vehicle Passengers   9266    48.999    -95.377

                            Point
 0   POINT (-70.396722 45.805661)
 1   POINT (-116.49925 48.999861)
 2    POINT (-114.7822222 32.485)
 3  POINT (-109.731333 48.999972)
 4      POINT (-95.376555 48.999)  )
```

In [4]:
```
save_folder = "border_crossing_visualizations"
os.makedirs(save_folder, exist_ok=True)  # Ensures the folder exists
```

In [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Aggregate data by month
monthly_trend = df_cleaned.groupby('Date')['Value'].sum().reset_index()

# Plot the monthly trend of border crossings
plt.figure(figsize=(12, 6))
sns.lineplot(data=monthly_trend, x='Date', y='Value', marker='o')

plt.title('Monthly Trend of Border Crossings (2024)')
plt.xlabel('Date')
plt.ylabel('Total Crossings')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

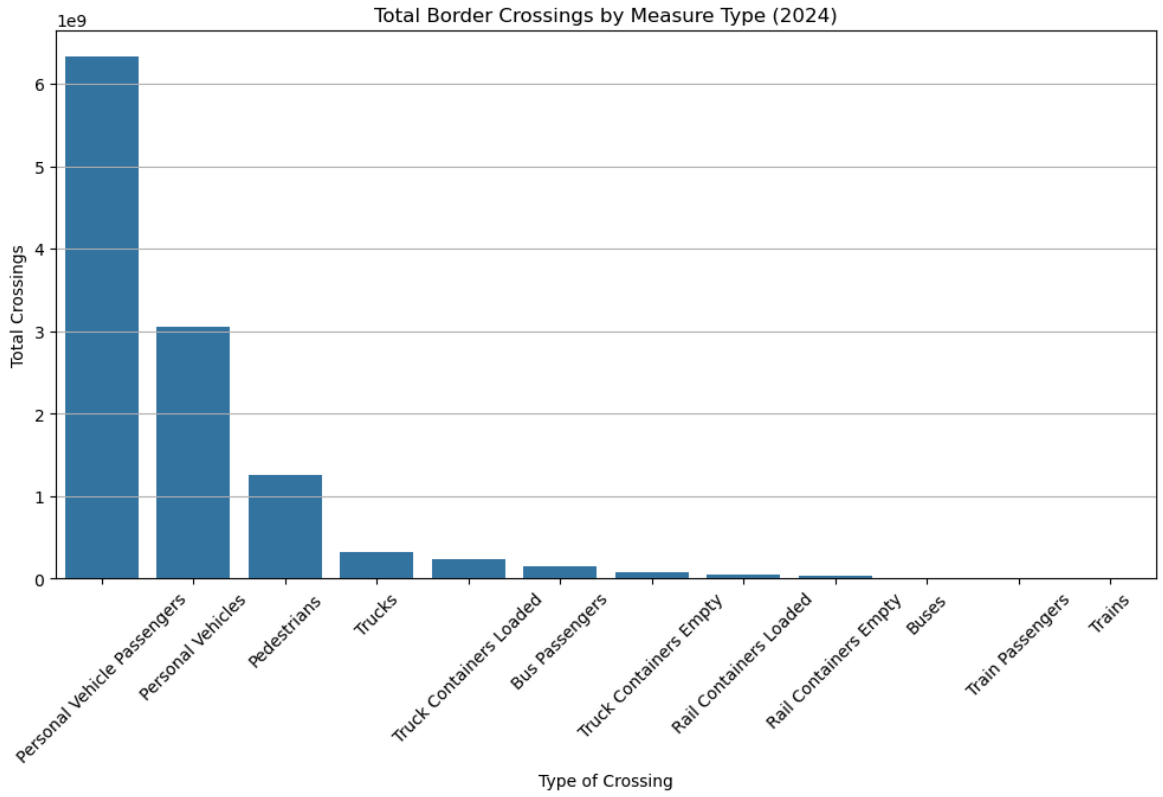


In [6]:
```
# Aggregate data by border type (Fix: Explicitly set `observed=False`)
border_distribution = df_cleaned.groupby('Border', observed=False)['Value'].s

# Plot a pie chart
plt.figure(figsize=(8, 8))
plt.pie(border_distribution, labels=border_distribution.index, autopct='%1.1f
        startangle=140, colors=['#66b3ff','#ff9999'], textprops={'fontsize':

plt.title('Proportion of Border Crossings by Border Type (2024)', fontsize=14

plt.show()
```

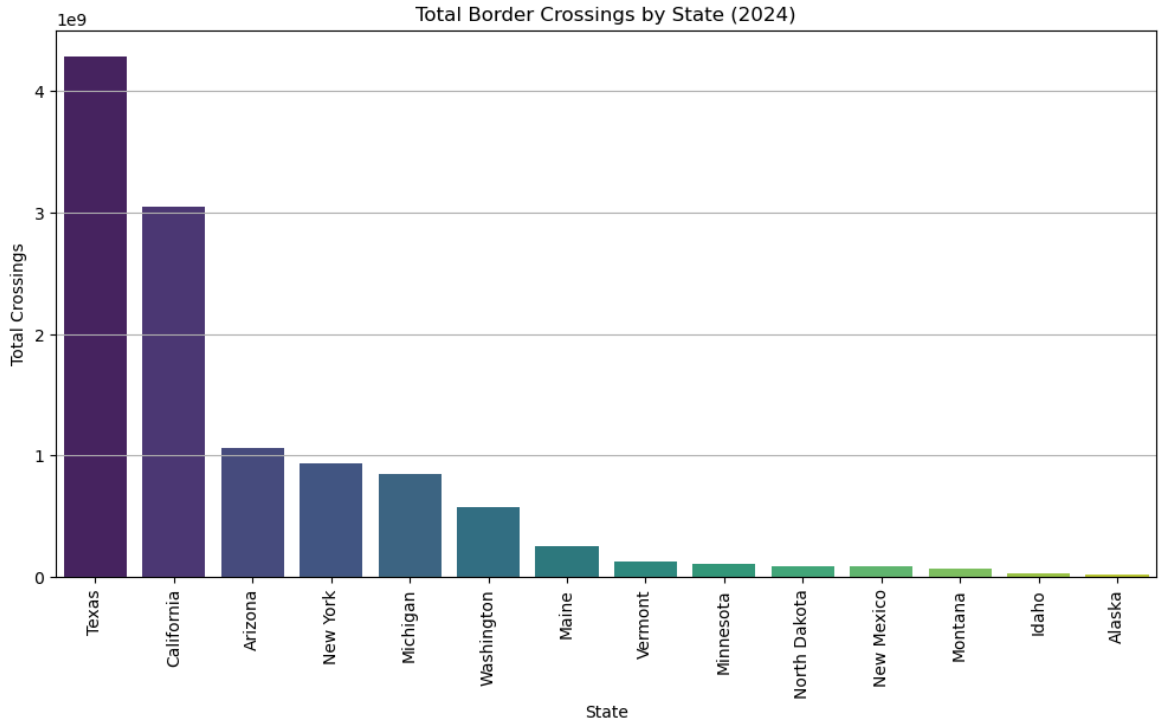


In [7]:

```
# Aggregate data by crossing measure (Fix: Explicitly set `observed=False`)
measure_distribution = df_cleaned.groupby('Measure', observed=False)['Value']

# Sort by total crossings for better visualization
measure_distribution = measure_distribution.sort_values(by='Value', ascending

# Plot the distribution of border crossings by measure type
plt.figure(figsize=(12, 6))
sns.barplot(data=measure_distribution, x='Measure', y='Value')

plt.title('Total Border Crossings by Measure Type (2024)')
plt.xlabel('Type of Crossing')
plt.ylabel('Total Crossings')
plt.xticks(rotation=45)
plt.grid(axis='y')

plt.show()
```

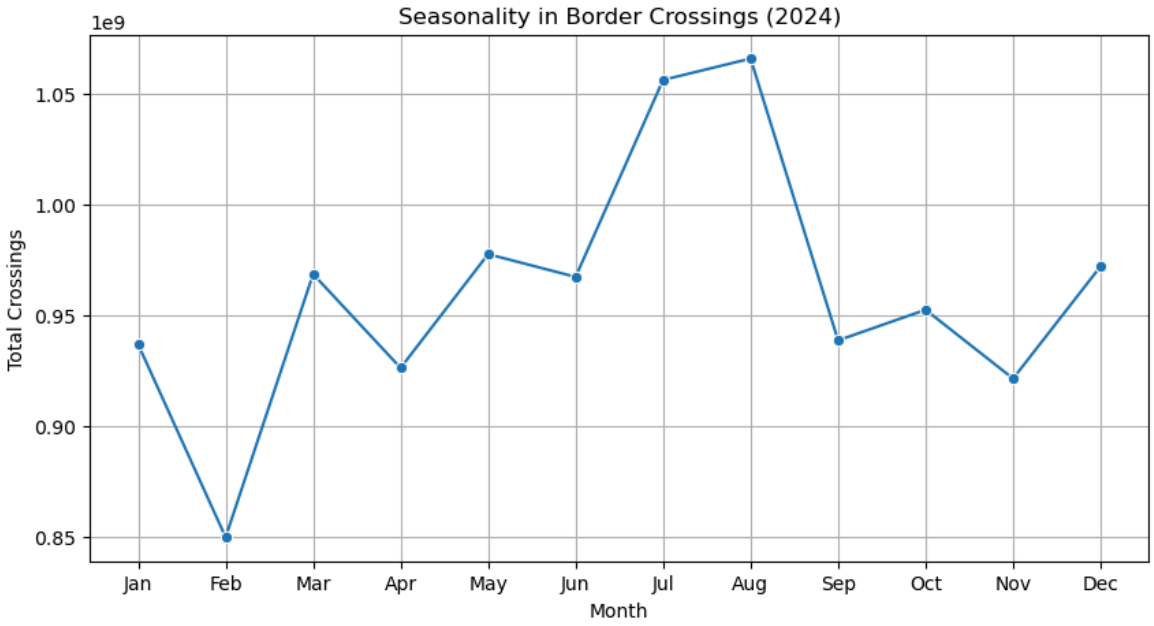


In [8]:
```
# Aggregate data by state (Fix: Explicitly set `observed=False`)
state_distribution = df_cleaned.groupby('State', observed=False)['Value'].sum

# Sort by total crossings for better visualization
state_distribution = state_distribution.sort_values(by='Value', ascending=Fal

# Plot heatmap-style bar chart of border crossings by state
plt.figure(figsize=(12, 6))
sns.barplot(data=state_distribution, x='State', y='Value', hue="State", palet

plt.title('Total Border Crossings by State (2024)')
plt.xlabel('State')
plt.ylabel('Total Crossings')
plt.xticks(rotation=90)
plt.grid(axis='y')

plt.show()
```



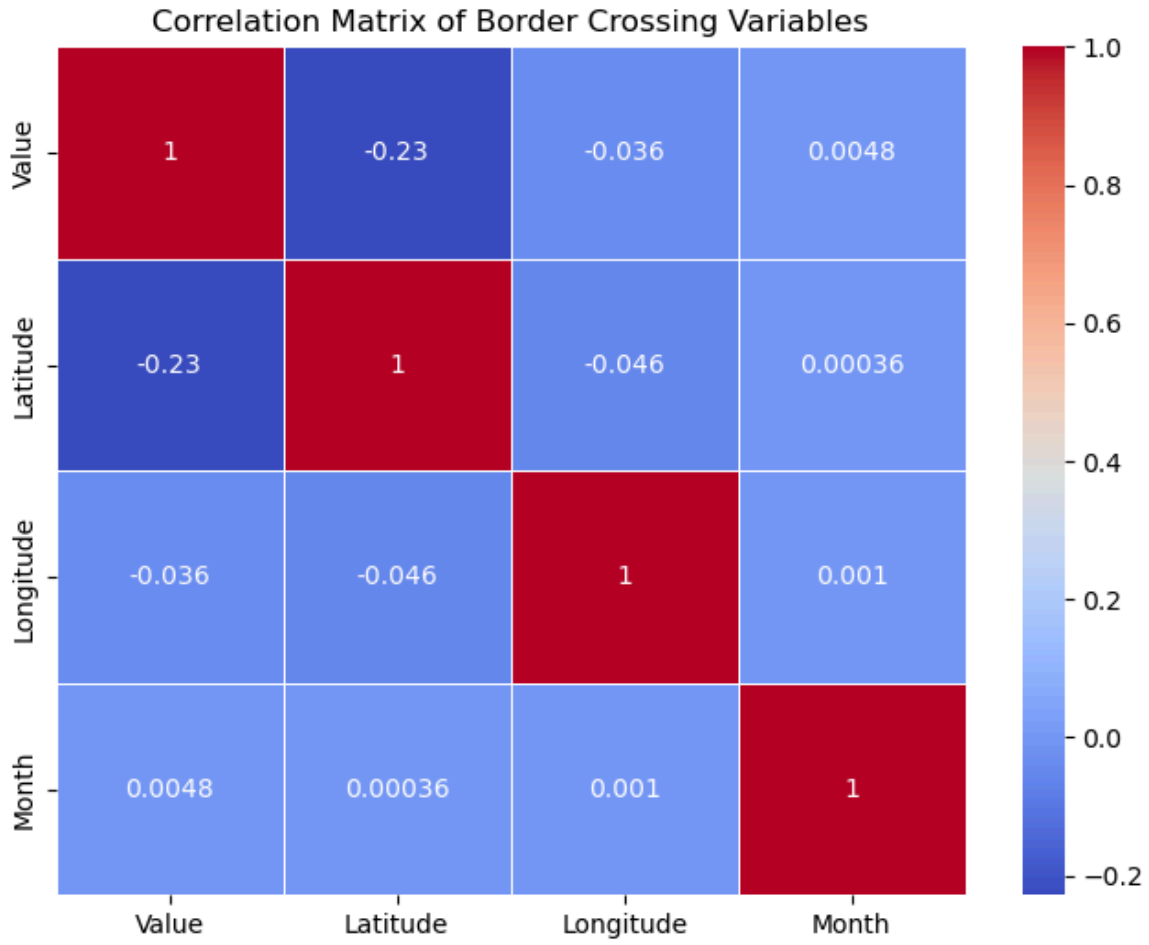
In [9]:

```
# Ensure df_cleaned is a full copy before modifying
df_cleaned = df_cleaned.copy()

# Extract month from Date for seasonality analysis using .loc
df_cleaned.loc[:, 'Month'] = df_cleaned['Date'].dt.month

# Aggregate data by month to analyze seasonality
seasonality_trend = df_cleaned.groupby('Month', observed=False)['Value'].sum(

# Plot seasonality trends
plt.figure(figsize=(10, 5))
sns.lineplot(data=seasonality_trend, x='Month', y='Value', marker='o')

plt.title('Seasonality in Border Crossings (2024)')
plt.xlabel('Month')
plt.ylabel('Total Crossings')
plt.xticks(range(1, 13), [
    'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
    'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'
])
plt.grid(True)

plt.show()
```



In [14]:

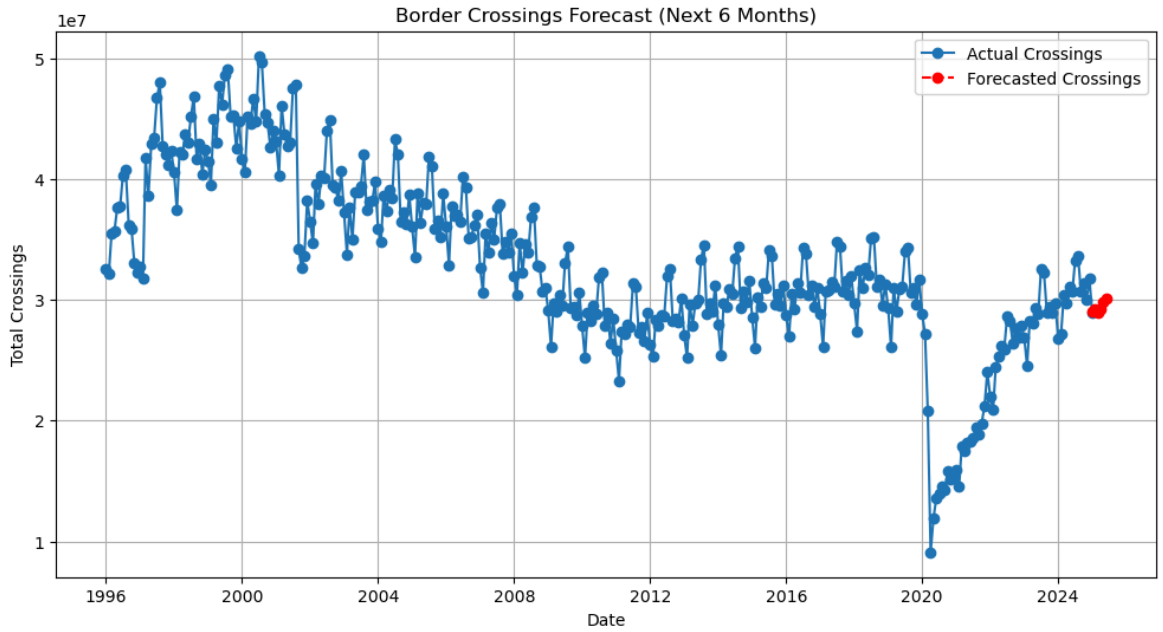
```
# Compute correlation matrix for numerical variables
correlation_matrix = df_cleaned[['Value', 'Latitude', 'Longitude', 'Month']].

# Plot the correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Matrix of Border Crossing Variables')
plt.show()
```



In [13]:

```
# Ensure df_cleaned is a full copy
df_cleaned = df_cleaned.copy()

# Prepare data for forecasting
time_series_data = df_cleaned.groupby('Date', observed=False)['Value'].sum()

# Set the Date column as index and assign frequency to fix the warning
time_series_data.index = pd.DatetimeIndex(time_series_data.index).to_period('

# Try seasonal smoothing first, fallback to non-seasonal if it fails
try:
    model = ExponentialSmoothing(time_series_data, trend='add', seasonal='add
    fitted_model = model.fit()
except:
    print("Optimization failed. Trying a simpler model without seasonality...
    model = ExponentialSmoothing(time_series_data, trend='add')
    fitted_model = model.fit()

# Predict next 6 months
forecast = fitted_model.forecast(steps=6)
forecast.index = pd.date_range(start=time_series_data.index[-1].to_timestamp(

# Plot the actual vs. predicted values
plt.figure(figsize=(12, 6))
plt.plot(time_series_data.index.to_timestamp(), time_series_data, label='Actu
plt.plot(forecast.index, forecast, label='Forecasted Crossings', linestyle='d

plt.title('Border Crossings Forecast (Next 6 Months)')
plt.xlabel('Date')
plt.ylabel('Total Crossings')
plt.legend()
plt.grid(True)

plt.show()
```

```
C:\Users\soumy\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\mode
l.py:915: ConvergenceWarning: Optimization failed to converge. Check mle_re
tvals.
  warnings.warn(
```



In [12]:

```
# Plot histogram for distribution of border crossings
plt.figure(figsize=(10, 5))
sns.histplot(df_cleaned['Value'], bins=50, kde=True, color='blue')

plt.title('Histogram of Border Crossings Distribution')
plt.xlabel('Number of Crossings')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```