Codes

# General trends in traffic accidents, fatalities, and serious injuries

accidents\_over\_time = df.groupby('Year')['Total Victims'].sum()

fatalities\_over\_time = df.groupby('Year')['Fatalities'].sum()

serious\_injuries\_over\_time = df.groupby('Year')['Serious Injuries'].sum()

# Plot trends

plt.figure(figsize=(12, 6))

plt.plot(accidents\_over\_time, label='Total Accidents')

plt.plot(fatalities\_over\_time, label='Fatalities')

plt.plot(serious\_injuries\_over\_time, label='Serious Injuries')

plt.title('General Trends in Traffic Accidents (2010-2021)')

plt.xlabel('Year')

plt.ylabel('Count')

plt.legend()

plt.show()

# Common characteristics in severe accidents

severe\_accidents = df[df['Severity of Accident'] == 'Severe']

# Explore characteristics

severe\_accidents\_by\_time = severe\_accidents.groupby('Hour of Day')['Total Victims'].mean()

# Plot

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

sns.barplot(x=severe\_accidents\_by\_time.index, y=severe\_accidents\_by\_time.values)

plt.title('Average Total Victims in Severe Accidents by Time of Day')

plt.xlabel('Hour of Day')

plt.ylabel('Average Total Victims')

plt.show()

# Municipalities or counties with the highest incidence of traffic accidents

accidents\_by\_municipality = df.groupby('Municipality Name')['Total Victims'].sum()

# Visualize

plt.figure(figsize=(14, 6))

accidents\_by\_municipality.sort\_values(ascending=False)[:10].plot(kind='bar')

plt.title('Top 10 Municipalities with Highest Traffic Accidents')

plt.xlabel('Municipality')

plt.ylabel('Total Victims')

plt.show()

# Yearly trends in traffic accident patterns

yearly\_trends = df.groupby('Year').agg({'Total Victims': 'sum', 'Fatalities': 'sum', 'Serious Injuries': 'sum'})

# Plot yearly trends

yearly\_trends.plot(kind='bar', stacked=True, figsize=(12, 6))

plt.title('Yearly Trends in Traffic Accidents (2010-2021)')

plt.xlabel('Year')

plt.ylabel('Count')

plt.show()

# Day and time patterns of accidents

day\_time\_patterns = df.groupby(['Day of the Week Grouping', 'Time of Day Grouping']).size().unstack()

# Plot heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(day\_time\_patterns, cmap='Blues', annot=True, fmt='d', cbar\_kws={'label': 'Number of Accidents'})

plt.title('Day and Time Patterns of Traffic Accidents')

plt.show()

# Environmental impact on accidents

environmental\_impact = df.groupby('Weather Conditions').agg({'Total Victims': 'sum', 'Serious Injuries': 'sum', 'Fatalities': 'sum'})

# Plot environmental impact

environmental\_impact.plot(kind='bar', stacked=True, figsize=(12, 6))

plt.title('Environmental Impact on Traffic Accidents')

plt.xlabel('Weather Conditions')

plt.ylabel('Count')

plt.show()

# Impact of road features and traffic density

road\_traffic\_impact = df.groupby(['Speed Limit', 'Road Type']).agg({'Total Victims': 'sum', 'Serious Injuries': 'sum', 'Fatalities': 'sum'})

# Plot impact

road\_traffic\_impact.plot(kind='bar', stacked=True, figsize=(12, 6))

plt.title('Impact of Road Features and Traffic Density on Traffic Accidents')

plt.xlabel('Speed Limit and Road Type')

plt.ylabel('Count')

plt.show()

# Impact of vehicle types on accident severity

vehicle\_severity\_impact = df.groupby('Type of Accident').agg({'Total Victims': 'sum', 'Serious Injuries': 'sum', 'Fatalities': 'sum'})

# Plot impact

vehicle\_severity\_impact.plot(kind='bar', stacked=True, figsize=(12, 6))

plt.title('Impact of Vehicle Types on Accident Severity')

plt.xlabel('Type of Accident')

plt.ylabel('Count')

plt.show()

# Temporal clustering analysis

temporal\_clusters = df.groupby(['Year', 'Month']).size().unstack()

# Plot heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(temporal\_clusters, cmap='Blues', annot=True, fmt='d', cbar\_kws={'label': 'Number of Accidents'})

plt.title('Temporal Clustering of Traffic Accidents')

plt.show()

# Extract relevant time series for forecasting (e.g., Total Victims)

time\_series = df.resample('M').sum()['Total Victims']

# Train-test split

train\_size = int(len(time\_series) \* 0.8)

train, test = time\_series[:train\_size], time\_series[train\_size:]

# Exponential Smoothing model

model = ExponentialSmoothing(train, seasonal='add', seasonal\_periods=12)

fit\_model = model.fit()

# Forecasting

forecast = fit\_model.forecast(len(test))

# Evaluate the model

mse = mean\_squared\_error(test, forecast)

print(f'Mean Squared Error: {mse}')

# Plot the results

plt.figure(figsize=(12, 6))

plt.plot(train.index, train, label='Train')

plt.plot(test.index, test, label='Test')

plt.plot(test.index, forecast, label='Forecast')

plt.title('Time-Series Forecasting of Traffic Accidents')

plt.xlabel('Year')

plt.ylabel('Total Victims')

plt.legend()

plt.show()