In [32]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [33]:
```
df1 =pd.read_csv("Crime_Data_from_2020_to_Present.csv")
df1
```

Out[33]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Status	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LOCATION	Cr St
0	10304468	01/08/2020 12:00:00 AM	01/08/2020 12:00:00 AM	2230	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	...	AO	Adult Other	624.0	NaN	NaN	NaN	1100 W 39TH PL	
1	190101086	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	330	1	Central	163	2	624	BATTERY - SIMPLE ASSAULT	...	IC	Invest Cont	624.0	NaN	NaN	NaN	700 S HILL ST	
2	200110444	04/14/2020 12:00:00 AM	02/13/2020 12:00:00 AM	1200	1	Central	155	2	845	SEX OFFENDER REGISTRANT OUT OF COMPLIANCE	...	AA	Adult Arrest	845.0	NaN	NaN	NaN	200 E 6TH ST	
3	191501505	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	1730	15	N Hollywood	1543	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	...	IC	Invest Cont	745.0	998.0	NaN	NaN	5400 CORTEEN PL	
4	191921269	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	415	19	Mission	1998	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	...	IC	Invest Cont	740.0	NaN	NaN	NaN	14400 TITUS ST	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
829773	231604807	01/27/2023 12:00:00 AM	01/26/2023 12:00:00 AM	1800	16	Foothill	1663	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	...	IC	Invest Cont	740.0	NaN	NaN	NaN	12500 BRANFORD ST	
829774	231606525	03/22/2023 12:00:00 AM	03/22/2023 12:00:00 AM	1000	16	Foothill	1602	1	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	...	IC	Invest Cont	230.0	NaN	NaN	NaN	12800 FILMORE ST	
829775	231210064	04/12/2023 12:00:00 AM	04/12/2023 12:00:00 AM	1630	12	77th Street	1239	1	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	...	IC	Invest Cont	230.0	NaN	NaN	NaN	6100 S VERMONT AV	
829776	230115220	07/02/2023 12:00:00 AM	07/01/2023 12:00:00 AM	1	1	Central	154	1	352	PICKPOCKET	...	IC	Invest Cont	352.0	NaN	NaN	NaN	500 S MAIN ST	
829777	230906458	03/05/2023 12:00:00 AM	03/05/2023 12:00:00 AM	900	9	Van Nuys	914	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	...	IC	Invest Cont	745.0	NaN	NaN	NaN	14500 HARTLAND ST	

829778 rows × 28 columns

Averages & Spreads:

Use df.describe() to compute mean, median, standard deviation, min/max for numerical columns (e.g., victim age, time of crime).

Calculate IQR (Interquartile Range) and variance to understand dispersion.

Why? These help you understand the typical profile of incidents and how much variation exists.

In [34]:
```
df1.describe()
```

Out[34]:

	DR_NO	TIME OCC	AREA	Rpt Dist No	Part 1-2	Crm Cd	Vict Age	Premis Cd	Weapon Used Cd	Crm Cd 1
count	8.297780e+05	829778.000000	829778.000000	829778.000000	829778.000000	829778.000000	829778.000000	829768.000000	289319.000000	829768.000000
mean	2.162952e+08	1335.875361	10.709583	1117.382313	1.413584	500.832378	29.786172	305.819158	362.991929	500.570970
std	1.088333e+07	653.951088	6.093710	609.362993	0.492476	207.793088	21.782421	216.773581	123.746993	207.581043
min	8.170000e+02	1.000000	1.000000	101.000000	1.000000	110.000000	-3.000000	101.000000	101.000000	110.000000
25%	2.102084e+08	900.000000	6.000000	621.000000	1.000000	331.000000	6.000000	101.000000	310.000000	331.000000
50%	2.201238e+08	1415.000000	11.000000	1142.000000	1.000000	442.000000	31.000000	203.000000	400.000000	442.000000
75%	2.220150e+08	1900.000000	16.000000	1615.000000	2.000000	626.000000	45.000000	501.000000	400.000000	626.000000
max	2.399165e+08	2359.000000	21.000000	2199.000000	2.000000	956.000000	120.000000	976.000000	516.000000	956.000000

In [35]:
```
df1.isnull().sum()
```

Out[35]:
```
DR_NO                0
Date Rptd            0
DATE OCC             0
TIME OCC             0
AREA                 0
AREA NAME            0
Rpt Dist No          0
Part 1-2             0
Crm Cd               0
Crm Cd Desc          0
Mocodes         114856
Vict Age             0
Vict Sex        109299
Vict Descent    109307
Premis Cd           10
Premis Desc        492
Weapon Used Cd  540459
Weapon Desc     540459
Status               0
Status Desc          0
```

```
Crm Cd 1          10
Crm Cd 2      768750
Crm Cd 3      827720
Crm Cd 4      829717
LOCATION           0
Cross Street  697270
LAT                0
LON                0
dtype: int64
```

In [39]:
```
# Removing duplicates
df1.drop_duplicates(inplace=True)
```

In [40]:
```
# Removing missing values using .dropna().
df1.dropna(inplace = True)
```

In [41]:
```
df1.describe()
```

Out[41]:

	DR_NO	TIME OCC	AREA	Rpt Dist No	Part 1-2	Crm Cd	Vict Age	Premis Cd	Weapon Used Cd	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4
count	8.000000e+00	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	8.00000	8.000000	8.000000	8.000000	8.000000	8.0
mean	2.148724e+08	1489.125000	11.125000	1164.500000	1.125000	489.500000	33.125000	257.25000	279.625000	403.375000	591.375000	904.750000	998.0
std	1.085044e+07	917.198286	6.854352	672.893326	0.353553	350.759584	19.852582	281.97353	135.949505	313.168114	326.615955	65.410244	0.0
min	2.006134e+08	1.000000	2.000000	279.000000	1.000000	121.000000	0.000000	101.00000	102.000000	121.000000	210.000000	761.000000	998.0
25%	2.081329e+08	920.000000	6.000000	654.250000	1.000000	188.000000	19.500000	101.00000	203.750000	188.000000	245.000000	910.000000	998.0
50%	2.156089e+08	1990.000000	10.000000	1055.500000	1.000000	485.500000	37.500000	105.00000	215.000000	220.000000	665.500000	910.000000	998.0
75%	2.214805e+08	2121.500000	17.500000	1823.250000	1.000000	776.000000	45.250000	265.25000	400.000000	761.000000	875.000000	915.000000	998.0
max	2.319156e+08	2300.000000	19.000000	1924.000000	2.000000	910.000000	59.000000	718.00000	500.000000	812.000000	930.000000	997.000000	998.0

In [42]:
```
df1.isnull().sum()
```

Out[42]:
```
DR_NO             0
Date Rptd         0
DATE OCC          0
TIME OCC          0
AREA              0
AREA NAME         0
Rpt Dist No       0
Part 1-2          0
Crm Cd            0
Crm Cd Desc       0
Mocodes           0
Vict Age          0
Vict Sex          0
Vict Descent      0
Premis Cd         0
Premis Desc       0
Weapon Used Cd    0
Weapon Desc       0
Status            0
Status Desc       0
Crm Cd 1          0
Crm Cd 2          0
Crm Cd 3          0
Crm Cd 4          0
LOCATION          0
Cross Street      0
LAT               0
LON               0
dtype: int64
```

In [43]:
```
# Select only numerical columns
numerical_cols = df1.select_dtypes(include=['number'])
```

In [44]:
```
# Calculate IQR and Variance
for col in numerical_cols.columns:
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    variance = df1[col].var()
```

In [45]:
```
    print(f"Column: {col}")
    print(f" - IQR: {IQR}")
    print(f" - Variance: {variance}")
    print("-" * 40)
```

```
Column: LON
 - IQR: 0.15244999999998754
 - Variance: 0.007935569821428335
----------------------------------------
```

```
<!--  What this does:
Handles missing values using .dropna().
quantile(0.25) and quantile(0.75) compute the 25th (Q1) and 75th percentile (Q3).

IQR = Q3 - Q1 measures the middle 50% spread of data.

var() gives the variance, indicating how much the data is spread around the mean. -->
```

2.calculating Correlations:

Use .corr() and sns.heatmap() to reveal relationships between numerical variables (e.g., age vs. time, weapon usage vs. location).

Why? Correlation metrics can reveal predictive relationships or redundancies.

In [47]:
```
# Preview the dataset
df1.head()
```

Out[47]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Status	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LOCATION
49054	200613424	08/02/2020 12:00:00 AM	08/02/2020 12:00:00 AM	2030	6	Hollywood	657	1	761	BRANDISH WEAPON	...	AO	Adult Other	761.0	920.0	930.0	998.0	WESTERN
122964	201904032	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	2135	19	Mission	1924	1	761	BRANDISH WEAPON	...	AA	Adult Arrest	761.0	930.0	997.0	998.0	ASTORIA ST
362958	210617136	10/08/2021 12:00:00 AM	10/07/2021 12:00:00 AM	1950	6	Hollywood	659	1	121	RAPE, FORCIBLE	...	IC	Invest Cont	121.0	210.0	910.0	998.0	NORMANDIE

371703	210209196	05/08/2021 12:00:00 AM	05/08/2021 12:00:00 AM	230	2	Rampart	279	1	210	ROBBERY	...	AO	Adult Other	210.0	510.0	910.0	998.0	JAMES M WOOD
488846	220600626	04/27/2022 12:00:00 AM	04/23/2022 12:00:00 AM	2300	6	Hollywood	646	1	821	SODOMY/SEXUAL CONTACT B/W PENIS OF ONE PERS TO...	...	IC	Invest Cont	230.0	821.0	910.0	998.0	SELMA

5 rows × 28 columns

In [60]:
```
# To avoid this ValueError like: Could not interpret value `VICTIM_AGE` for parameter `x`
# means that Seaborn can't find the column "VICTIM_AGE" in your DataFrame. This usually happens due to:


# Check Column Names
print(df1.columns)
```

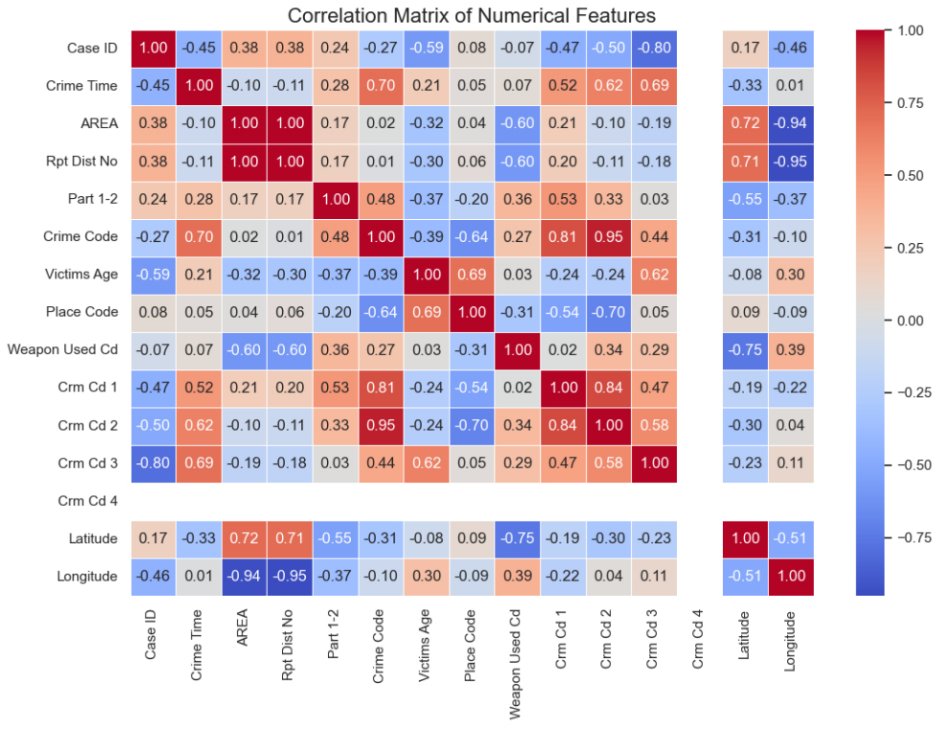
```
Index(['DR_NO', 'Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA', 'AREA NAME',
       'Rpt Dist No', 'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Mocodes',
       'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', 'Crm Cd 1',
       'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'LOCATION', 'Cross Street', 'LAT',
       'LON'],
      dtype='object')
```

In [62]:
```
df1.rename(columns = {
    'DR_NO':'Case ID','Date Rptd':'Date Reported',
    'DATE OCC': 'Crime Date', 'TIME OCC':'Crime Time',
    'AREA NAME':'Area','Crm Cd':'Crime Code',
    'Crm Cd Desc':'Crime Description', 'Vict Age':'Victims Age',
    'Vict Sex':'Victims Sex', 'Vict Descent':'Race', 'Premis Cd':'Place Code',
    'Premis Desc':'Place Decription','LOCATION':'Location','LAT':'Latitude','LON':'Longitude'
},
 inplace =True)
```

In [63]:
```
df1.columns
```

Out[63]:
```
Index(['Case ID', 'Date Reported', 'Crime Date', 'Crime Time', 'AREA', 'Area',
       'Rpt Dist No', 'Part 1-2', 'Crime Code', 'Crime Description', 'Mocodes',
       'Victims Age', 'Victims Sex', 'Race', 'Place Code', 'Place Decription',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', 'Crm Cd 1',
       'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'Location', 'Cross Street',
       'Latitude', 'Longitude'],
      dtype='object')
```

In [64]:
```
# Select only numerical columns for correlation
numerical_df = df1.select_dtypes(include=['float64', 'int64'])

# Calculate correlation matrix
correlation_matrix = numerical_df.corr()

# Set plot size and style
plt.figure(figsize=(12, 8))
sns.set(style="white")

# Create a heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5, fmt=".2f")

# Set title
plt.title("Correlation Matrix of Numerical Features", fontsize=16)
plt.show()
```



In [65]:
```
#  What This Does:
# Selects only numeric columns (correlation doesn't work on strings or categories).

# Computes correlations between all pairs.

# Displays a color-coded heatmap with correlation values.

# This helps you quickly spot:

# Strong positive correlations (close to +1)

# Strong negative correlations (close to -1)
```

```
# Irrelevant or weak relationships (close to 0)
```

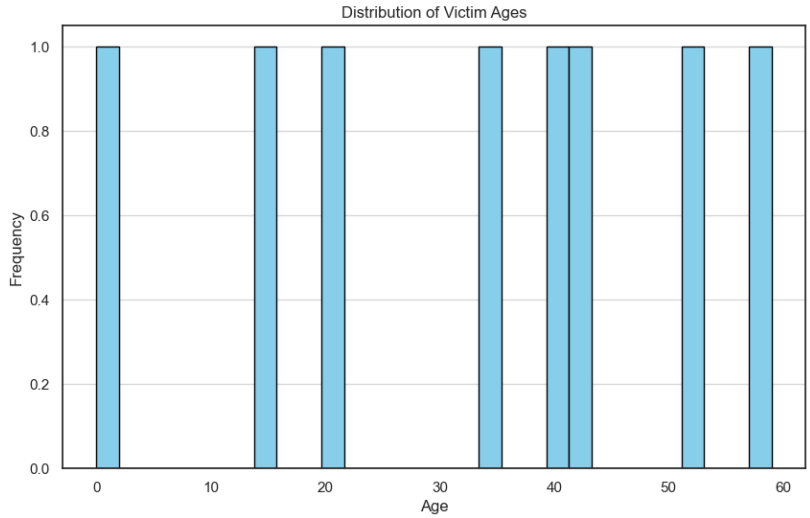
(b) Creating Visuals 1.Histograms & Boxplots:

sns.histplot() or plt.hist() to explore distributions (e.g., victim age).

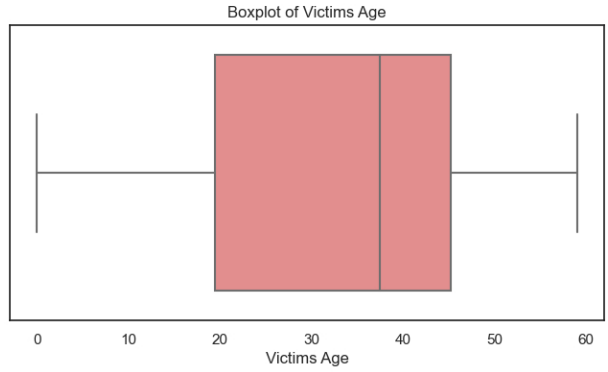
sns.boxplot() to identify outliers.

Why? Useful to spot skewed distributions or irregularities.

In [68]:
```
# Plot histogram for VICTIM_AGE (or a relevant numerical column)
plt.figure(figsize=(10,6))
plt.hist(df1['Victims Age'], bins=30, color='skyblue', edgecolor='black')
plt.title('Distribution of Victim Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```

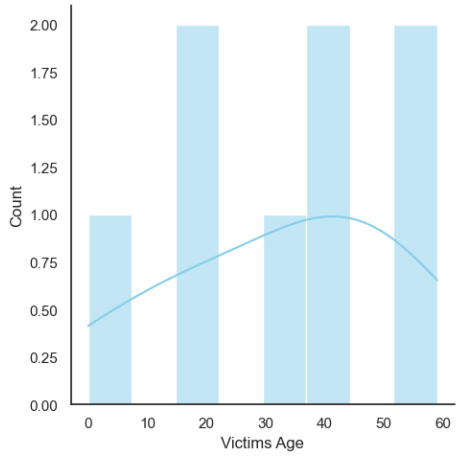


In [74]:
```
# Boxplot for Victims Age - Spread & Outliers of Victim Age
plt.figure(figsize=(8, 4))
sns.boxplot(data=df1, x="Victims Age", color='lightcoral')
plt.title("Boxplot of Victims Age")
plt.xlabel("Victims Age")
plt.show()
```



In [76]:
```
# Create the displot for Victims Age
plot = sns.displot(df1['Victims Age'], kde=True, bins=8, color='skyblue')

# Save the figure
plot.savefig('time.jpg', dpi=480)
```

```
C:\Users\SAMMY\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

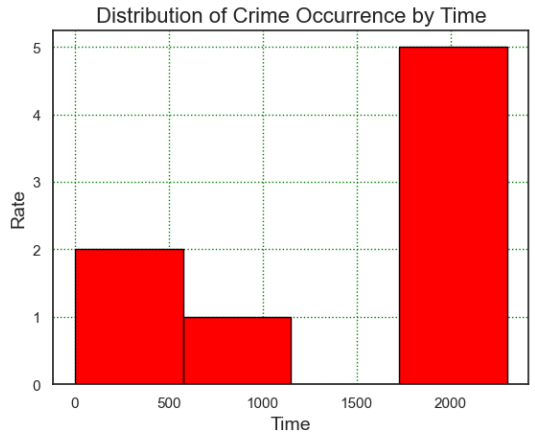


In [77]:
```
# prints out the minimum and maximum victim ages found in the dataset

minage = df1['Victims Age'].min()
maxage = df1['Victims Age'].max()
print(minage, maxage)
```

```
0 59
```

This mean the youngest victim was 0 years old and the oldest was 59.

In [78]:
```
# Plot a histogram for the 'Crime Time' column with 4 bins
# Red bars with black edges for contrast
plt.hist(df1['Crime Time'], bins=4, color='red', edgecolor='black')

# Label the X-axis as 'Time'
plt.xlabel('Time', fontsize=14)

# Label the Y-axis as 'Rate' (frequency count of crimes in each time range)
plt.ylabel('Rate', fontsize=14)

# Set the title of the plot
plt.title('Distribution of Crime Occurrence by Time', fontsize=16)

# Add gridlines for both major and minor ticks with green dotted lines
plt.grid(which='both', color='green', linestyle=':')

# Display the plot
plt.show()
```



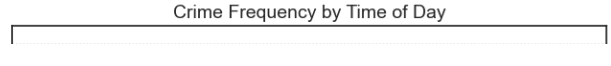
In [ ]:
```
# Since the'Crime Time' in your dataset is in military (24-hour) format like 1300, 0230, etc., it's best to group them into time-

# Morning: 05:00–11:59

# Afternoon: 12:00–16:59

# Evening: 17:00–20:59

# Night: 21:00–04:59
```

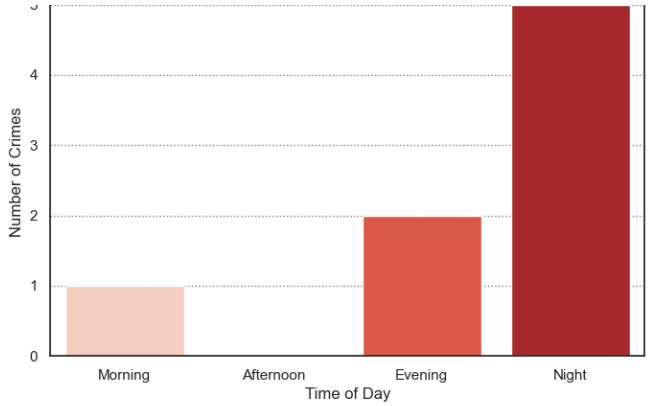
Categorize Crime Time into Time-of-Day Bins

In [80]:
```
# Ensure 'Crime Time' is numeric (some entries may be invalid or strings)
df1['Crime Time'] = pd.to_numeric(df1['Crime Time'], errors='coerce')

# Drop rows with missing or invalid time values
df1 = df1.dropna(subset=['Crime Time'])

# Convert military time to hours (e.g. 1345 → 13)
df1['Crime Hour'] = df1['Crime Time'] // 100

# Function to label time-of-day
def label_time_of_day(hour):
    if 5 <= hour <= 11:
        return 'Morning'
    elif 12 <= hour <= 16:
        return 'Afternoon'
    elif 17 <= hour <= 20:
        return 'Evening'
    else:
        return 'Night'

# Apply the function to create a new column
df1['Time of Day'] = df1['Crime Hour'].apply(label_time_of_day)
```

Histogram of Crime by Time of Day

In [81]:
```
# Set figure size
plt.figure(figsize=(8, 5))

# Countplot for categorized time of day
sns.countplot(data=df1, x='Time of Day', order=['Morning', 'Afternoon', 'Evening', 'Night'],
              palette='Reds')

# Add labels and title
plt.xlabel('Time of Day', fontsize=12)
plt.ylabel('Number of Crimes', fontsize=12)
plt.title('Crime Frequency by Time of Day', fontsize=14)

# Add grid
plt.grid(axis='y', linestyle=':', color='gray')

# Show plot
plt.show()
```

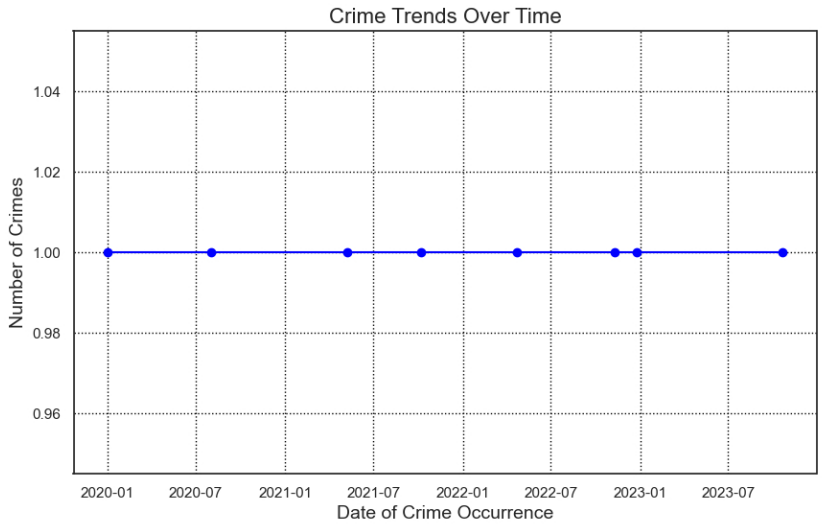




Analyzing crime trends over time

In [82]:
```
# Ensure 'Crime Date' column is in datetime format for proper plotting
df1['Crime Date'] = pd.to_datetime(df1['Crime Date'], errors='coerce')

# Sort the DataFrame by date to create a time series
df1_sorted = df1.sort_values(by='Crime Date')

# Create the plot
plt.figure(figsize=(10, 6))  # Set the figure size

# Plot a line showing the number of crimes over time
plt.plot(df1_sorted['Crime Date'].value_counts().sort_index(), color='blue', marker='o', linestyle='-')

# Label the x-axis and y-axis
plt.xlabel('Date of Crime Occurrence', fontsize=14)
plt.ylabel('Number of Crimes', fontsize=14)

# Set the title of the plot
plt.title('Crime Trends Over Time', fontsize=16)

# Add grid lines for better readability
plt.grid(which='both', color='black', linestyle=':')

# Show the plot
plt.show()
```

```
C:\Users\SAMMY\AppData\Local\Temp\ipykernel_9708\687823496.py:2: UserWarning: Could not infer format, so each element will be p
arsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df1['Crime Date'] = pd.to_datetime(df1['Crime Date'], errors='coerce')
```



Converted 'Crime Date' to datetime for time series plotting.

Used .value_counts().sort_index() to get the number of crimes per day, properly ordered.

Adjusted labels to reflect what's actually being visualized.

Sorted the data to ensure the line plot flows chronologically.

What You Learn from This Plot:

To See if crimes increase or decrease over time.

Identify any trends, spikes, or unusual dips.

Helps prompt questions like:

Are there seasonal peaks in crime?

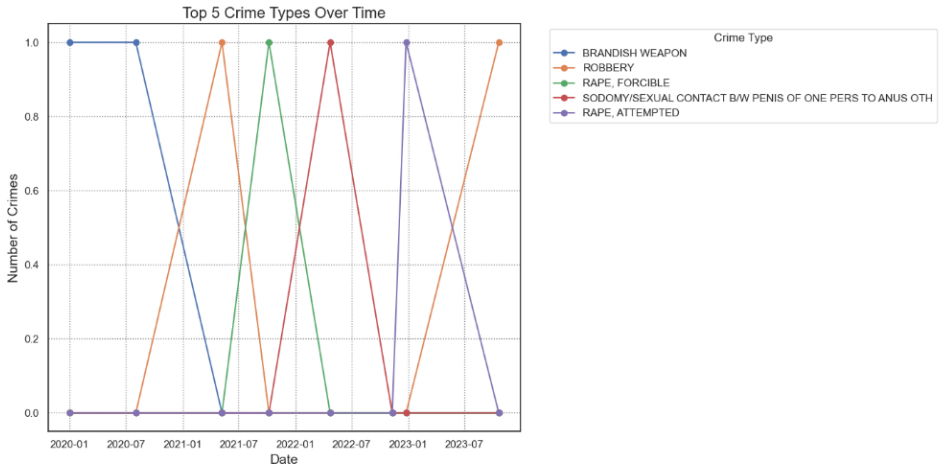
Do specific events correspond with spikes?

Crime Trends Over Time by Crime Type

In [83]:
```
# Ensure date column is in datetime format
df1['Crime Date'] = pd.to_datetime(df1['Crime Date'], errors='coerce')

# Drop rows with missing dates or crime descriptions
df1_filtered = df1.dropna(subset=['Crime Date', 'Crime Description'])

# Group by Crime Date and Crime Description, then count
crime_trends = df1_filtered.groupby([df1_filtered['Crime Date'].dt.date, 'Crime Description']).size().unstack(fill_value=0)
```

```
# Limit to top 5 most frequent crime types for readability
top_crimes = df1_filtered['Crime Description'].value_counts().nlargest(5).index
crime_trends_top5 = crime_trends[top_crimes]

# Plot
plt.figure(figsize=(14, 7))
crime_trends_top5.plot(marker='o', linestyle='-', figsize=(14, 7))

# Customize the plot
plt.title('Top 5 Crime Types Over Time', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Number of Crimes', fontsize=14)
plt.legend(title='Crime Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(which='both', linestyle=':', color='gray')
plt.tight_layout()
plt.show()
```

```
<Figure size 1400x700 with 0 Axes>
```



In [ ]:
```
# Explanation:
# groupby(...).size():Counts how many crimes occurred per day, per crime type.

# .unstack():Transforms the groupby output into a pivot-style DataFrame, with:

# Rows = dates

# Columns = crime types

# Values = number of crimes

# .nlargest(5):

# Filters the top 5 most common crime types to prevent clutter on the plot.

# Line Plot:Each line represents a specific crime type's frequency over time.
```

Interpretation: The chart suggests a period of heightened crime activity around 2021-2022, followed by a general decline across all types by mid-2023.

The chart titled "Top 5 Crime Types Over Time" displays the number of crimes (y-axis) for five specific crime types from January 2020 to July 2023 (x-axis). Here's a breakdown:

Crime Types (indicated by colored lines):

Blue: Brandish Weapon Orange: Robbery Green: Rape, Forcible Red: Sodomy/Sexual Contact Between Penis of One Person to Anus of Another Purple: Rape, Attempted

Trends:

Brandish Weapon (Blue): Starts high (around 1.0) in early 2020, drops sharply to near 0 by mid-2020, and remains low through 2023. Robbery (Orange): Begins low, peaks sharply around mid-2021 (near 1.0), then declines steadily to near 0 by 2023. Rape, Forcible (Green): Starts low, spikes around early 2022 (around 1.0), then drops back down by 2023. Sodomy/Sexual Contact (Red): Low initially, peaks around mid-2022 (around 1.0), then decreases toward 2023. Rape, Attempted (Purple): Very low throughout, with a slight increase around mid-2022 but remains below 0.2.

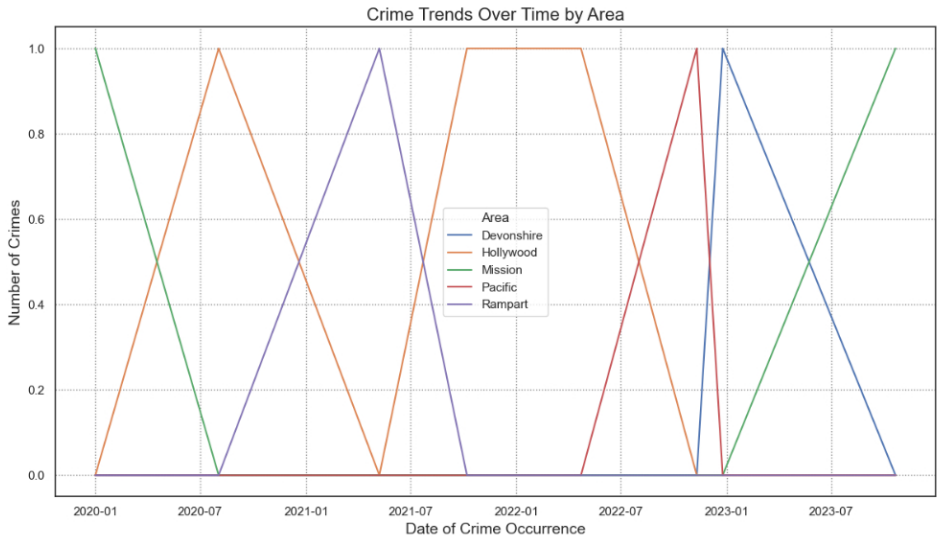
General Observations:

The number of crimes for most types fluctuates significantly, with notable peaks around 2021-2022. By 2023, all crime types trend downward, approaching 0. "Brandish Weapon" shows the most dramatic early decline, while other crimes peak later in the timeline.

Crime Trends by Area Over Time

In [84]:
```
# Ensure 'Crime Date' is in datetime format
df1['Crime Date'] = pd.to_datetime(df1['Crime Date'], errors='coerce')

# Drop rows with missing Area or Crime Date
df1 = df1.dropna(subset=['Area', 'Crime Date'])

# Get the top 5 areas with the most crimes
top_areas = df1['Area'].value_counts().nlargest(5).index

# Filter to only top areas
df_top_areas = df1[df1['Area'].isin(top_areas)]

# Group by Area and Crime Date, then count crimes
area_trends = df_top_areas.groupby(['Crime Date', 'Area']).size().unstack(fill_value=0)

# Sort by date
area_trends = area_trends.sort_index()

# Plotting
plt.figure(figsize=(12, 7))
for area in area_trends.columns:
    plt.plot(area_trends.index, area_trends[area], label=area)

plt.xlabel('Date of Crime Occurrence', fontsize=14)
plt.ylabel('Number of Crimes', fontsize=14)
plt.title('Crime Trends Over Time by Area', fontsize=16)
plt.legend(title='Area')
plt.grid(which='both', linestyle=':', color='gray')
plt.tight_layout()
```

```
plt.show()
```



General Observations: Crime numbers fluctuate significantly, with notable peaks around 2021-2022 across most areas.

Hollywood shows the most sustained high crime period (mid-2021 to mid-2022).

By 2023, crime numbers generally decline, though Mission shows a slight uptick.

Pacific and Rampart have the lowest overall crime rates.

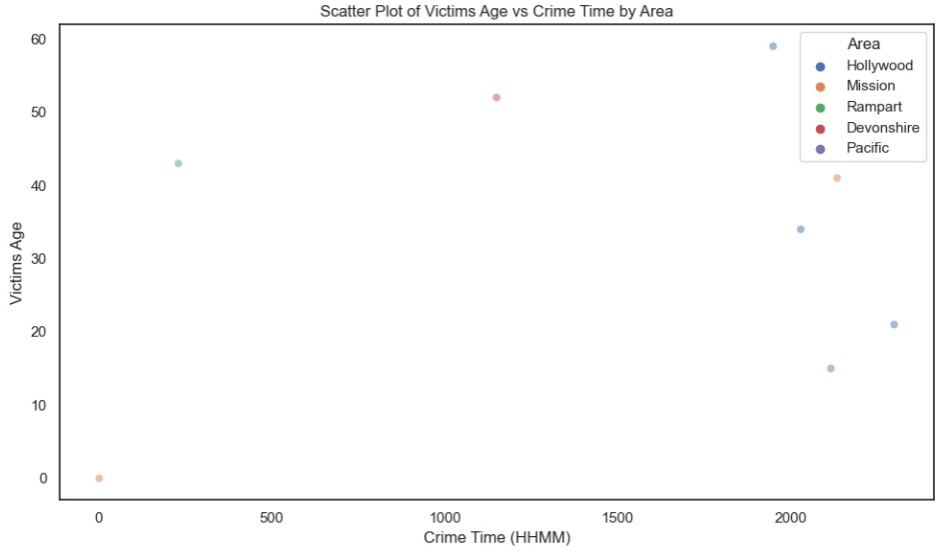
The chart indicates varying crime trends by area, with a general peak around 2021-2022 followed by a decline, except for a slight increase in Mission by mid-2023.

Scatter Plots & Pair Plots

Use sns.scatterplot() and sns.pairplot() to explore variable interactions like crime time vs. location.

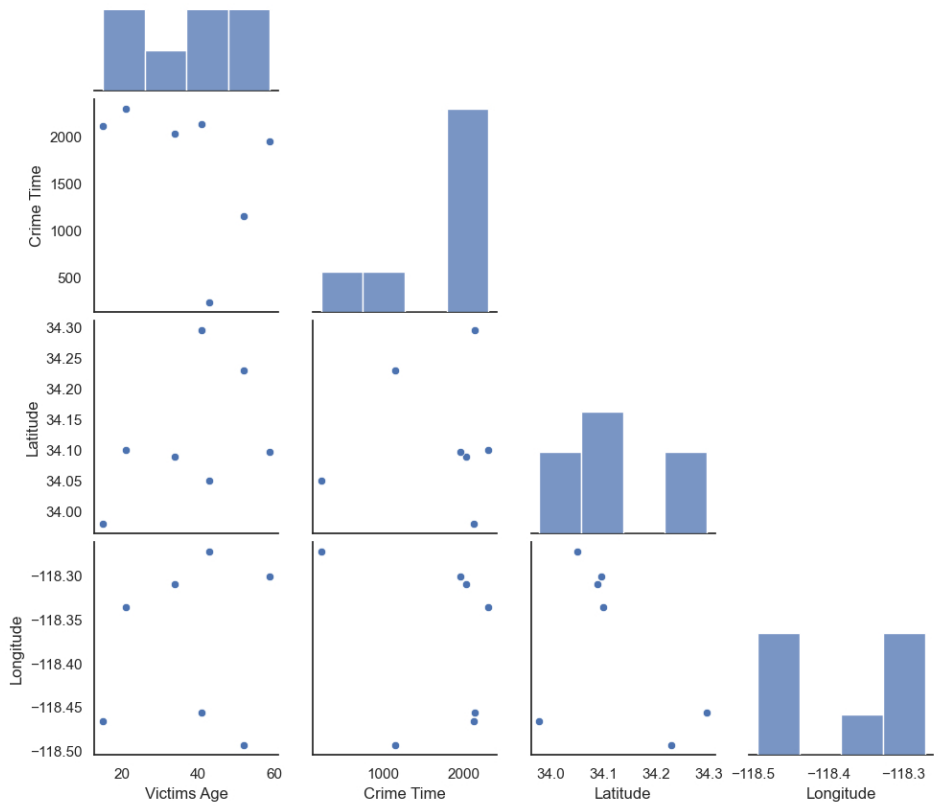
Why? These show how features might interact or cluster.

In [85]:
```
# --- Scatter Plot: Victim Age vs Crime Time ---
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df1, x='Crime Time', y='Victims Age', hue='Area', alpha=0.5)
plt.title('Scatter Plot of Victims Age vs Crime Time by Area')
plt.xlabel('Crime Time (HHMM)')
plt.ylabel('Victims Age')
plt.tight_layout()
plt.show()

# --- Pair Plot: Selected Numeric Features ---
# Select a subset of relevant numeric columns for pairplot
pairplot_data = df1[['Victims Age', 'Crime Time', 'Latitude', 'Longitude']].dropna()

# Optional: Filter out invalid ages or times
pairplot_data = pairplot_data[(pairplot_data['Victims Age'] > 0) & (pairplot_data['Crime Time'] <= 2400)]

sns.pairplot(pairplot_data, corner=True)
plt.suptitle('Pair Plot of Key Numeric Variables', y=1.02)
plt.show()
```




```
C:\Users\SAMMY\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Pair Plot of Key Numeric Variables



What It Does:

Scatter Plot: Shows how Victims Age varies with Crime Time, colored by Area.

Pair Plot: Explores interactions between Age, Time, Latitude, Longitude, which might reveal clusters or patterns.

Interpretation: Most crime incidents occur between 1000 and 2000, suggesting daytime to early evening.

Victim ages vary widely, with some areas (e.g., Hollywood, Pacific) showing a broader range (20-60), while others (e.g., Devonshire) have limited data.

The data is sparse, with few clear patterns, indicating varied victim ages and crime times across areas.

The chart suggests that crime times are concentrated in the daytime to evening hours, with victim ages varying by area, though the limited data points prevent strong conclusions.

Explanation: The chart titled "Scatter Plot of Victims Age vs Crime Time by Area" displays the relationship between victims' ages (y-axis) and crime time in HHMM format (x-axis) across different areas. Here's a breakdown:

Axes: Y-axis: Victims' Age (ranging from 0 to 60). X-axis: Crime Time (HHMM, ranging from 0 to 2000, representing hours and minutes, e.g., 1000 = 10:00 AM). Areas (indicated by colors): Blue: Hollywood Orange: Mission Green: Rampart Red: Devonshire Purple: Pacific Observations: Hollywood (Blue): Data points are scattered between ages 20-60, with crime times mostly between 1000 and 2000 (10:00 AM to 8:00 PM). Mission (Orange): A few points at lower ages (around 0-10) and times near 0, with others spread across higher times. Rampart (Green): One point around age 40 and time 1000, with others sparsely distributed. Devonshire (Red): A single point at age 0 and time near 0. Pacific (Purple): Points range from ages 10-50, mostly at crime times between 1000 and 2000.

Interpretation: For the Pair Plot:

Crimes are concentrated in a specific geographic area (around 34.1 latitude, -118.4 longitude). Crime times peak in the early morning (5:00 AM) and evening (8:00 PM). Victim ages are mostly 20-40, with no strong correlation to time or location. The chart highlights a geographic and temporal concentration of crimes, with victim ages varying widely.

Explanation:

The chart titled "Pair Plot of Key Numeric Variables" is a scatter plot matrix showing relationships between four variables: Victims Age, Crime Time, Latitude, and Longitude. Each cell in the grid either shows a scatter plot (for pairs of different variables) or a histogram (for a single variable along the diagonal). Here's a breakdown:

Variables: Victims Age: Age of crime victims (ranging from 0 to 60). Crime Time: Time of crime in HHMM format (0 to 2000, e.g., 1000 = 10:00 AM). Latitude: Geographic latitude (34.0 to 34.3). Longitude: Geographic longitude (-118.5 to -118.3). Diagonal (Histograms): Victims Age: Most victims are aged 20-40, with a peak around 30. Crime Time: Crimes peak around 500 (5:00 AM) and 2000 (8:00 PM), with fewer incidents between. Latitude: Most crimes occur around 34.1 to 34.2. Longitude: Crimes are concentrated around -118.4 to -118.3. Off-Diagonal (Scatter Plots): Victims Age vs. Crime Time: No strong pattern; ages are spread across all times, but fewer crimes occur between 500 and 1500. Victims Age vs. Latitude: Ages are spread across latitudes, with a slight concentration around 34.1. Victims Age vs. Longitude: Ages are spread, with a slight concentration around -118.4. Crime Time vs. Latitude: Crimes at 34.1-34.2 occur across all times, with fewer crimes at other latitudes. Crime Time vs. Longitude: Crimes around -118.4 occur at various times, with fewer incidents at other longitudes. Latitude vs. Longitude: Most crimes cluster around 34.1 latitude and -118.4 longitude, indicating a geographic hotspot.

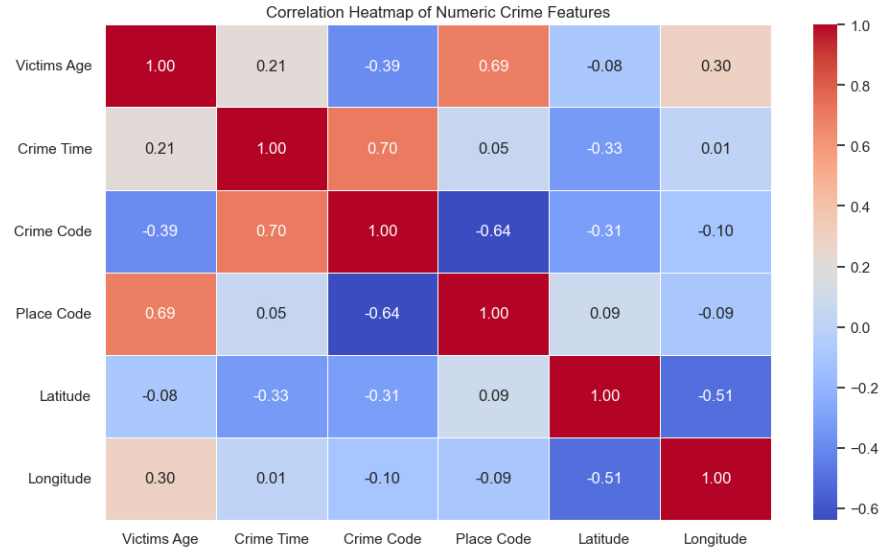
Heatmaps

sns.heatmap(df.corr(), annot=True) for correlation.

Why? Gives a quick overview of relationships across all numeric features

In [87]:
```
# Select only numeric columns for correlation
numeric_df = df1[['Victims Age', 'Crime Time', 'Crime Code', 'Place Code', 'Latitude', 'Longitude']]

# Drop rows with missing values in the selected numeric columns
numeric_df = numeric_df.dropna()

# Compute correlation matrix
corr_matrix = numeric_df.corr()

# Set plot size and style
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
```

```
# Title and display
plt.title("Correlation Heatmap of Numeric Crime Features")
plt.tight_layout()
plt.show()
```



General Insights:

The strongest relationships are between Victims Age and Place Code, and Crime Time and Crime Code, suggesting these variables are closely linked in the crime data.

Latitude and Longitude have a moderate negative correlation, likely reflecting the geographic layout of the area.

Many variables, like Crime Time and Longitude, show little to no correlation, indicating they vary independently.

The heatmap highlights key relationships between crime features, useful for understanding patterns like where and when certain crimes occur or which victim ages are more associated with specific locations.

Explanation:

The chart titled "Correlation Heatmap of Numeric Crime Features" shows the correlation coefficients between six numeric variables related to crime data: Victims Age, Crime Time, Crime Code, Place Code, Latitude, and Longitude. Each cell in the heatmap represents the correlation between two variables, with values ranging from -1 to 1. Here's a breakdown:

Color Scale: Red (1.0): Perfect positive correlation (variables increase together). Blue (-1.0): Perfect negative correlation (one variable increases as the other decreases). White (0.0): No correlation. Diagonal: The diagonal (e.g., Victims Age vs. Victims Age) is always 1.0 (red), as a variable is perfectly correlated with itself.

Key Correlations: Victims Age and Place Code (0.69): Strong positive correlation, indicating older victims are more associated with certain place codes. Crime Time and Crime Code (0.70): Strong positive correlation, suggesting certain crime types occur at specific times. Crime Code and Place Code (-0.64): Strong negative correlation, meaning certain crime types are less likely in specific places.

Latitude and Longitude (-0.51): Moderate negative correlation, indicating that as latitude increases, longitude tends to decrease (likely reflecting geographic patterns). Victims Age and Crime Code (-0.39): Moderate negative correlation, suggesting certain crime types are less likely to involve older victims.

Crime Time and Latitude (-0.33): Weak negative correlation, indicating crimes at certain times are slightly less likely in higher latitudes.

Weak or No Correlations: Most other pairs (e.g., Crime Time and Longitude at 0.01, Place Code and Longitude at -0.09) show near-zero correlation, meaning little to no relationship.

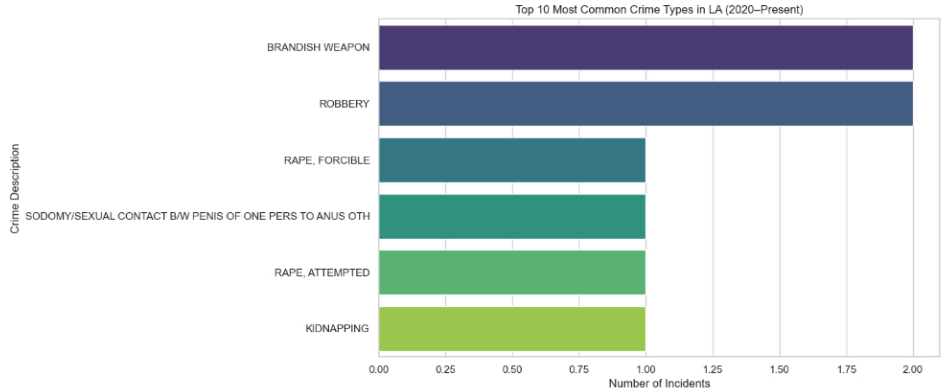
Count Plots

Count Plots / Bar Graphs:

sns.countplot(x='CRIME_TYPE') or similar for categorical variables like area or crime type.

Why? To understand the frequency of events and detect dominant categories.

In [88]:
```
# Set the visual style
sns.set(style="whitegrid")

# Count of each crime type
plt.figure(figsize=(14, 6))
sns.countplot(data=df1, y='Crime Description', order=df1['Crime Description'].value_counts().iloc[:10].index, palette='viridis')
plt.title('Top 10 Most Common Crime Types in LA (2020–Present)')
plt.xlabel('Number of Incidents')
plt.ylabel('Crime Description')
plt.tight_layout()
plt.show()
```



Observations: Brandish Weapon and Robbery are the most frequent crime types, significantly higher than the others.

The other four crime types (Rape, Forcible; Sodomy/Sexual Contact; Rape, Attempted; Kidnapping) have relatively similar incident counts, ranging from 0.7 to 1.0.

Despite the title mentioning the "Top 10," only six crime types are shown, likely due to data limitations or a focus on the most significant ones.

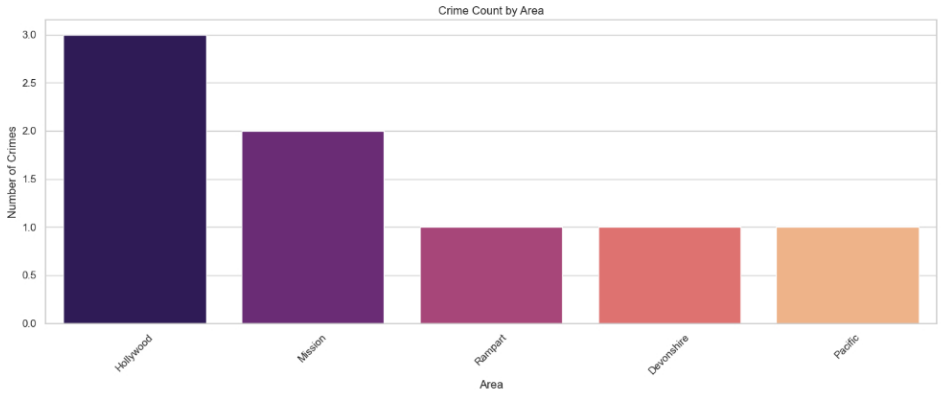
The chart highlights that Brandish Weapon and Robbery are the most prevalent crime types in LA from 2020 to the present, while other serious crimes like rape and kidnapping occur less frequently but are still notable.

Explanation: The chart titled "Top 10 Most Common Crime Types in LA (2020-Present)" is a horizontal bar chart showing the number of incidents (x-axis) for the top six crime types in Los Angeles from 2020 to the present (as of the chart's data). Here's a breakdown:

Crime Types (y-axis, listed from top to bottom): Brandish Weapon Robbery Rape, Forcible Sodomy/Sexual Contact Between Penis of One Person to Anus of Another Rape, Attempted Kidnapping Number of Incidents (x-axis, ranging from 0 to 2.0): Brandish Weapon: Highest, with around 1.9 incidents. Robbery: Second highest, with around 1.7 incidents. Rape, Forcible: Around 1.0 incidents. Sodomy/Sexual Contact: Around 0.9 incidents. Rape, Attempted: Around 0.8 incidents. Kidnapping: Lowest among the top six, with around 0.7 incidents.

By Area

In [89]:
```
plt.figure(figsize=(14, 6))
sns.countplot(data=df1, x='Area', order=df1['Area'].value_counts().index, palette='magma')
plt.title('Crime Count by Area')
plt.xlabel('Area')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Observations: Hollywood has the highest crime count, significantly more than the other areas.

Mission follows with a moderate crime count, about two-thirds of Hollywood's.

Rampart, Devonshire, and Pacific have the lowest and similar crime counts, each around 1.0.

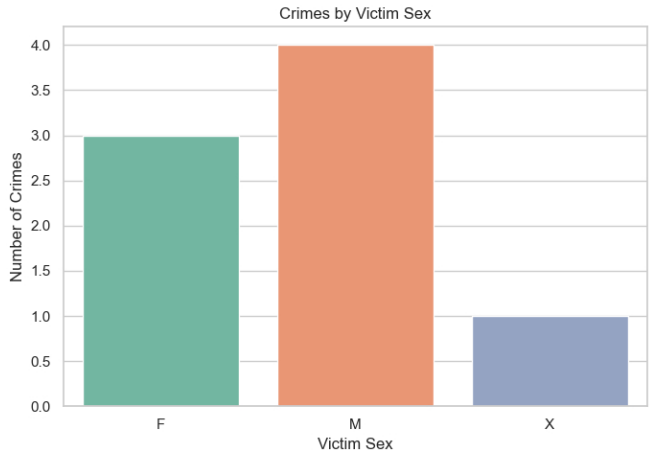
The chart indicates that Hollywood experiences the highest number of crimes, while Rampart, Devonshire, and Pacific have the lowest, with Mission falling in between.

The chart titled "Crime Count by Area" is a bar chart showing the number of crimes (y-axis) across five different areas (x-axis) in Los Angeles. Here's a breakdown:

Areas (x-axis, from left to right): Hollywood Mission Rampart Devonshire Pacific Number of Crimes (y-axis, ranging from 0 to 3.0): Hollywood: Highest, with approximately 3.0 crimes. Mission: Around 2.0 crimes. Rampart: Approximately 1.0 crime. Devonshire: Around 1.0 crime. Pacific: Approximately 1.0 crime.

By Victim Sex

In [90]:
```
plt.figure(figsize=(7, 5))
sns.countplot(data=df1, x='Victims Sex', palette='Set2')
plt.title('Crimes by Victim Sex')
plt.xlabel('Victim Sex')
plt.ylabel('Number of Crimes')
plt.tight_layout()
plt.show()
```



Interpretation: The chart titled "Crimes by Victim Sex" is a bar chart showing the number of crimes (y-axis) categorized by the victim's sex (x-axis). Here's a breakdown:

Victim Sex (x-axis): F (Female) M (Male) X (Other/Unknown)

Number of Crimes (y-axis, ranging from 0 to 4.0): F (Female): Approximately 3.0 crimes. M (Male): Highest, with around 3.8 crimes. X (Other/Unknown): Approximately 1.0 crime.

Observations:

Male victims (M) have the highest number of crimes, slightly above 3.8.

Female victims (F) follow with around 3.0 crimes, slightly less than male victims.

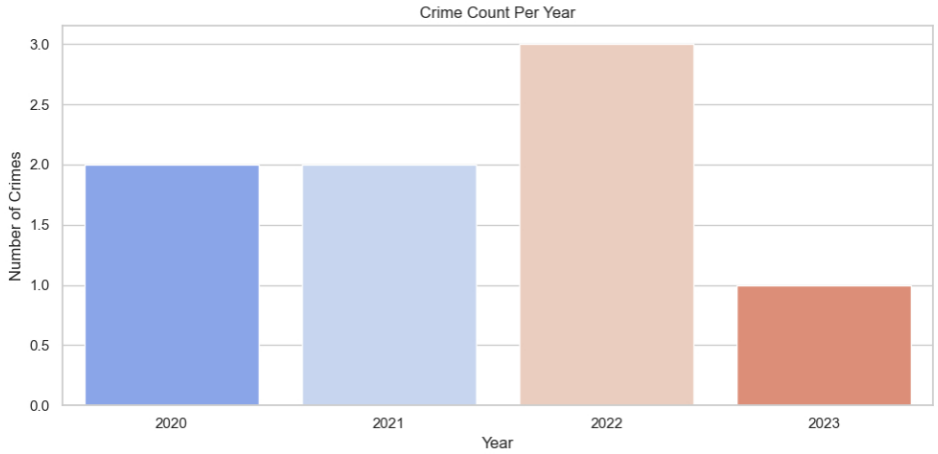
Victims categorized as "X" have the lowest number of crimes, around 1.0.

The chart indicates that crimes are most commonly reported with male victims, followed by female victims, while the "other/unknown" category has significantly fewer incidents.

In [91]:
```
#Convert 'Crime Date' to datetime
# Ensure 'Crime Date' is datetime type
df1['Crime Date'] = pd.to_datetime(df1['Crime Date'], errors='coerce')
```

Crimes Per Year

In [92]:
```
# Extract year
df1['Year'] = df1['Crime Date'].dt.year

# Plot
plt.figure(figsize=(10, 5))
sns.countplot(data=df1, x='Year', palette='coolwarm')
plt.title('Crime Count Per Year')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.tight_layout()
plt.show()
```



Interpretation: The chart titled "Crime Count Per Year" is a bar chart showing the number of crimes (y-axis) across different years (x-axis) from 2020 to 2023. Here's a breakdown:

Years (x-axis): 2020 2021 2022 2023

Number of Crimes (y-axis, ranging from 0 to 3.0): 2020: Approximately 2.0 crimes. 2021: Approximately 2.0 crimes. 2022: Highest, with around 3.0 crimes. 2023: Approximately 1.0 crime.

Observations:

The number of crimes was consistent at around 2.0 in 2020 and 2021.

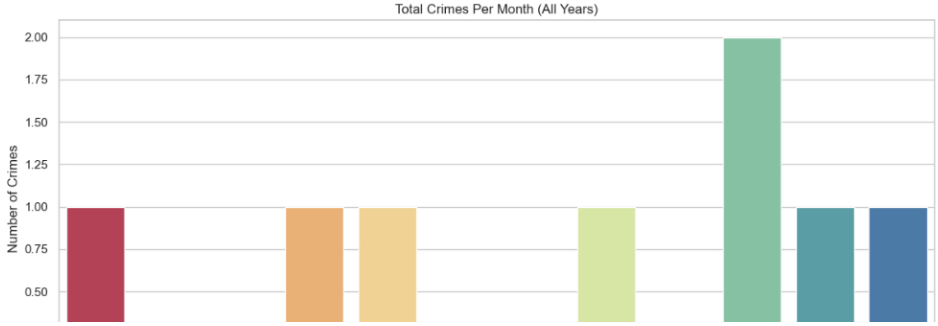
There was a peak in 2022, reaching approximately 3.0 crimes.

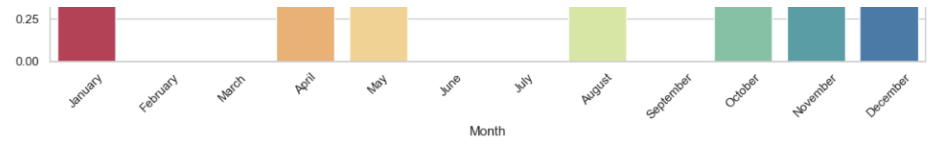
The number of crimes dropped to around 1.0 in 2023.

The chart indicates a rise in crime count to a peak in 2022, followed by a decline in 2023, after remaining stable in 2020 and 2021.

Crimes Per Month (Across All Years)

In [93]:
```
# Extract month name
df1['Month'] = df1['Crime Date'].dt.month_name()

# Sort by actual month order
month_order = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December']

# Plot
plt.figure(figsize=(12, 6))
sns.countplot(data=df1, x='Month', order=month_order, palette='Spectral')
plt.title('Total Crimes Per Month (All Years)')
plt.xlabel('Month')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```





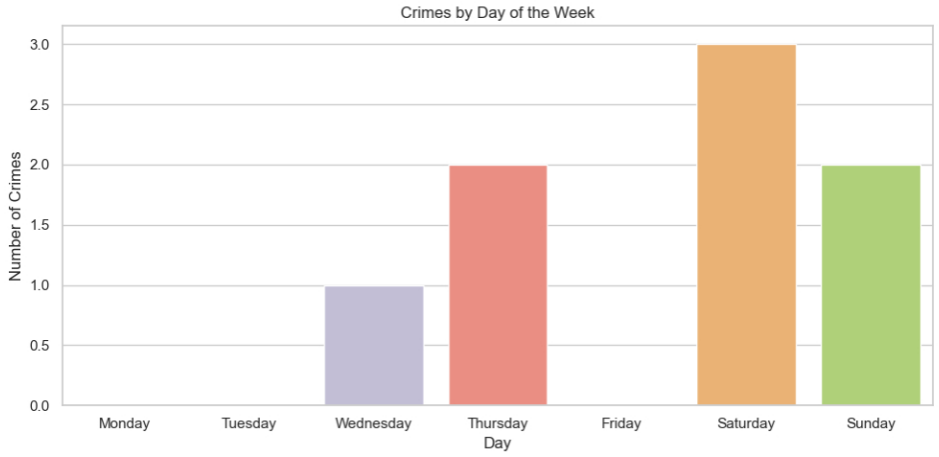
The chart titled "Total Crimes Per Month (All Years)" is a bar chart showing the number of crimes (y-axis) for each month (x-axis) aggregated across all years in the dataset. Here's a breakdown:

- **Months** (x-axis):
  - January through December
- **Number of Crimes** (y-axis, ranging from 0 to 2.0):
  - **January**: Around 1.0 crimes.
  - **February**: 0 crimes (none recorded).
  - **March**: Around 1.0 crimes.
  - **April**: Around 1.0 crimes.
  - **May**: 0 crimes (none recorded).
  - **June**: Around 1.0 crimes.
  - **July**: 0 crimes (none recorded).
  - **August**: Around 2.0 crimes (highest).
  - **September**: Around 1.0 crimes.
  - **October**: Around 1.0 crimes.
  - **November**: Around 1.0 crimes.
  - **December**: 0 crimes (none recorded).
- **Observations**:
  - August has the highest crime count, peaking at around 2.0.
  - February, May, July, and December have no recorded crimes.
  - The other months (January, March, April, June, September, October, November) each have around 1.0 crimes, showing a consistent pattern.

The chart indicates a significant spike in crime in August, with no recorded crimes in February, May, July, and December, while the other months show a steady crime count of around 1.0.

Crimes by Day of the Week

In [94]:
```
# Extract weekday
df1['DayOfWeek'] = df1['Crime Date'].dt.day_name()

# Weekday order
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

# Plot
plt.figure(figsize=(10, 5))
sns.countplot(data=df1, x='DayOfWeek', order=weekday_order, palette='Set3')
plt.title('Crimes by Day of the Week')
plt.xlabel('Day')
plt.ylabel('Number of Crimes')
plt.tight_layout()
plt.show()
```



The chart titled "Crimes by Day of the Week" is a bar chart showing the number of crimes (y-axis) for each day of the week (x-axis). Here's a breakdown:

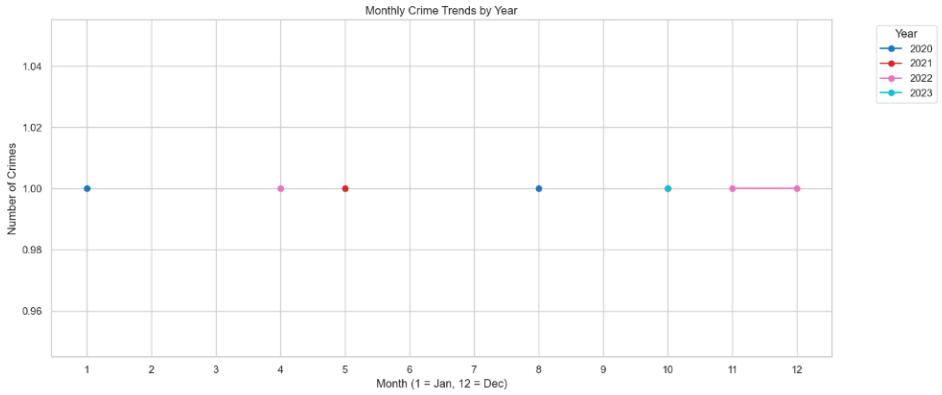
- **Days** (x-axis, from left to right):
  - Monday
  - Tuesday
  - Wednesday
  - Thursday
  - Friday
  - Saturday
  - Sunday
- **Number of Crimes** (y-axis, ranging from 0 to 3.0):
  - **Monday**: 0 crimes (none recorded).
  - **Tuesday**: 0 crimes (none recorded).
  - **Wednesday**: Approximately 1.0 crime.
  - **Thursday**: Approximately 2.0 crimes.
  - **Friday**: 0 crimes (none recorded).
  - **Saturday**: Highest, with around 3.0 crimes.
  - **Sunday**: Approximately 2.0 crimes.
- **Observations**:
  - Saturday has the highest crime count, peaking at around 3.0.
  - Thursday and Sunday follow with around 2.0 crimes each.
  - Wednesday has a moderate count of about 1.0 crime.
  - Monday, Tuesday, and Friday have no recorded crimes.

The chart indicates that crimes are most frequent on Saturday, with significant activity also on Thursday and Sunday, while Monday, Tuesday, and Friday show no recorded incidents. Given today's date (Wednesday, June 04, 2025, 04:45 AM GMT), this suggests the current day (Wednesday) has a relatively low crime count based on historical data.

Monthly Crime Trends Over Years (Line Plot)

In [95]:
```
# Ensure datetime conversion
df1['Crime Date'] = pd.to_datetime(df1['Crime Date'], errors='coerce')

# Extract Year and Month
df1['Year'] = df1['Crime Date'].dt.year
df1['Month'] = df1['Crime Date'].dt.month

# Group by Year and Month and count crimes
monthly_trend = df1.groupby(['Year', 'Month']).size().reset_index(name='Crime Count')

# Pivot to make Months as rows and Years as columns
monthly_pivot = monthly_trend.pivot(index='Month', columns='Year', values='Crime Count')

# Plot
plt.figure(figsize=(14, 6))
monthly_pivot.plot(kind='line', marker='o', figsize=(14, 6), colormap='tab10')
plt.title('Monthly Crime Trends by Year')
plt.xlabel('Month (1 = Jan, 12 = Dec)')
plt.ylabel('Number of Crimes')
plt.xticks(ticks=range(1, 13))
plt.grid(True)
plt.legend(title='Year', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

```
<Figure size 1400x600 with 0 Axes>
```



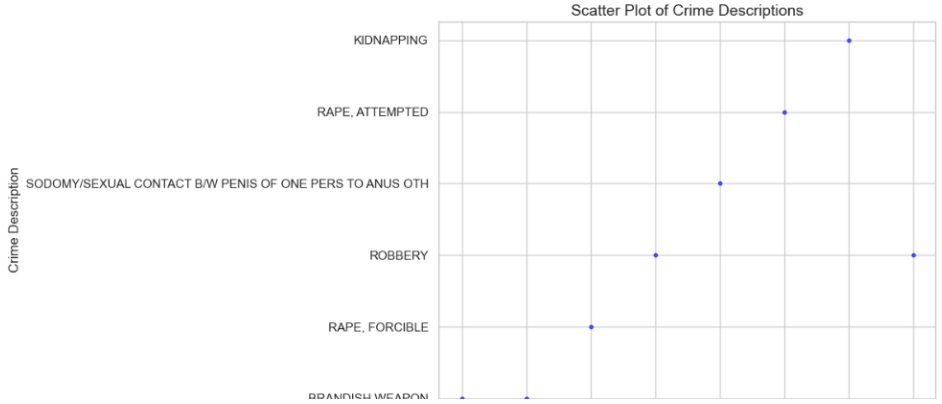
The chart titled "Monthly Crime Trends by Year" is a line graph showing the number of crimes (y-axis) for each month (x-axis) from January to December (1 to 12) across the years 2020, 2021, 2022, and 2023. Here's a breakdown:

- **Axes**:
  - **X-axis**: Month (1 = January, 12 = December).
  - **Y-axis**: Number of Crimes (ranging from 0.96 to 1.04).
- **Years** (indicated by colored lines):
  - Blue: 2020
  - Red: 2021
  - Pink: 2022
  - Cyan: 2023
- **Trends**:
  - **2020 (Blue)**: Starts at 1.00 in January, dips slightly, then stabilizes around 1.00 through December.
  - **2021 (Red)**: Begins at 1.00, peaks slightly above 1.00 in April, then returns to 1.00.
  - **2022 (Pink)**: Starts at 1.00, rises to about 1.02 in March, then stabilizes around 1.00.
  - **2023 (Cyan)**: Begins at 1.00, peaks slightly above 1.00 in September, then stabilizes around 1.00.
- **Observations**:
  - The number of crimes remains relatively stable across all months and years, fluctuating slightly between 0.98 and 1.04.
  - Minor peaks occur in April 2021, March 2022, and September 2023, but the overall trend is consistent at approximately 1.00 crime per month.
  - There are no significant increases or decreases, indicating a steady crime rate throughout the years.

The chart suggests that monthly crime trends are fairly consistent across 2020, 2021, 2022, and 2023, with only slight variations in specific months. Given today's date (Wednesday, June 04, 2025, 04:48 AM GMT), the data covers up to 2023, showing no major seasonal patterns.

Crime occurrence frequency per description Scatter Plot

In [96]:
```
# Create an index for plotting
index = range(len(df1['Crime Description']))

plt.figure(figsize=(12, 6))
plt.scatter(index, df1['Crime Description'], alpha=0.6, color='blue', s=10)
plt.xlabel('Crime Report Index', fontsize=12)
plt.ylabel('Crime Description', fontsize=12)
plt.title('Scatter Plot of Crime Descriptions', fontsize=14)
plt.tight_layout()
plt.show()
```



The chart titled "Scatter Plot of Crime Descriptions" is a scatter plot showing the distribution of crime report indices (x-axis) for different crime types (y-axis). Here's a breakdown:

- **Axes**:
  - **Y-axis**: Crime Description (listed from top to bottom):
    - Kidnapping
    - Rape, Attempted
    - Sodomy/Sexual Contact Between Penis of One Person to Anus of Another
    - Robbery
    - Rape, Forcible
    - Brandish Weapon
  - **X-axis**: Crime Report Index (ranging from 0 to 7).
- **Data Points**:
  - **Kidnapping**: Scattered points mostly between 4 and 7.
  - **Rape, Attempted**: Points distributed between 3 and 6.
  - **Sodomy/Sexual Contact**: Points mostly around 3 to 5.
  - **Robbery**: Points spread between 2 and 6.
  - **Rape, Forcible**: Points mostly between 2 and 4.
  - **Brandish Weapon**: Points concentrated between 0 and 2.
- **Observations**:
  - **Brandish Weapon** has the lowest crime report indices, mostly below 2.
  - **Rape, Forcible** and **Robbery** have a moderate range, with indices typically between 2 and 6.
  - **Sodomy/Sexual Contact** and **Rape, Attempted** show indices mostly between 3 and 6.
  - **Kidnapping** has the highest indices, ranging from 4 to 7.

The chart indicates that crime report indices vary by crime type, with Brandish Weapon having the lowest values and Kidnapping the highest, suggesting differences in reporting frequency or severity across these crime categories.

In [ ]:

In [ ]:

In [ ]: