hbase-ex2

June 3, 2025

0.1 Crime Data Analysis (2020 - Present)

0.1.1 Part 1: Data Understanding

Objectives

- Load crime dataset (`Crime_Data_from_2020_to_Present.csv`) using pandas.
- Explore dataset structure (rows, columns, missing values).
- Analyze crime types & area distribution.
- Perform a **temporal analysis** of crime trends.
- Visualize findings using **charts & plots**.

[ ]:

```python
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Configure visual settings
sns.set_style("whitegrid")
plt.rcParams["figure.figsize"] = (12, 6)

print("Libraries imported successfully.")
```

```
Libraries imported successfully.
```

0.2 Load Crime Dataset

We load the dataset using pandas and display basic information.

[ ]:

```python
# Load dataset
file_path = "data/Crime_Data_from_2020_to_Present.csv"
df = pd.read_csv(file_path)

# Display dataset info
print("Dataset Loaded Successfully!")
df.info()
```

```
Dataset Loaded Successfully!
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005091 entries, 0 to 1005090
```

```
Data columns (total 28 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   DR_NO           1005091 non-null  int64
 1   Date Rptd       1005091 non-null  object
 2   DATE OCC        1005091 non-null  object
 3   TIME OCC        1005091 non-null  int64
 4   AREA            1005091 non-null  int64
 5   AREA NAME       1005091 non-null  object
 6   Rpt Dist No     1005091 non-null  int64
 7   Part 1-2        1005091 non-null  int64
 8   Crm Cd          1005091 non-null  int64
 9   Crm Cd Desc     1005091 non-null  object
 10  Mocodes         853386 non-null   object
 11  Vict Age        1005091 non-null  int64
 12  Vict Sex        860362 non-null   object
 13  Vict Descent    860350 non-null   object
 14  Premis Cd       1005075 non-null  float64
 15  Premis Desc     1004503 non-null  object
 16  Weapon Used Cd  327250 non-null   float64
 17  Weapon Desc     327250 non-null   object
 18  Status          1005090 non-null  object
 19  Status Desc     1005091 non-null  object
 20  Crm Cd 1        1005080 non-null  float64
 21  Crm Cd 2        69157 non-null    float64
 22  Crm Cd 3        2314 non-null     float64
 23  Crm Cd 4        64 non-null       float64
 24  LOCATION        1005091 non-null  object
 25  Cross Street    154237 non-null   object
 26  LAT             1005091 non-null  float64
 27  LON             1005091 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 214.7+ MB
```

0.3 Dataset Overview

- **Show Number of Rows & Columns**
- **Column Data Types**
- **Missing Values Check**

[4]:

```
# Show number of rows & columns
print(f"Total Rows: {df.shape[0]}")
print(f"Total Columns: {df.shape[1]}")

# Check for missing values
missing_values = df.isnull().sum()
print("\nMissing Values:")
print(missing_values[missing_values > 0])
```

```
Total Rows: 1005091
Total Columns: 28

Missing Values:
Mocodes            151705
Vict Sex           144729
Vict Descent       144741
Premis Cd              16
Premis Desc           588
Weapon Used Cd     677841
Weapon Desc        677841
Status                  1
Crm Cd 1               11
Crm Cd 2           935934
Crm Cd 3          1002777
Crm Cd 4          1005027
Cross Street       850854
dtype: int64
```

0.4 Crime Types & Area Categories

We analyze: - **Unique Crime Categories** - **Top 10 Most Frequent Crimes** - **Top 10 Areas with Highest Crime Reports**

[ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns
# Unique crime categories
print("Unique Crime Categories:")
print(df["Crm Cd Desc"].unique())

# Top 10 most frequent crimes
crime_counts = df["Crm Cd Desc"].value_counts().head(10)

plt.figure(figsize=(12, 6))
sns.barplot(x=crime_counts.values, y=crime_counts.index, palette="coolwarm")
plt.title("  Top 10 Most Reported Crimes")
plt.xlabel("Number of Incidents")
plt.ylabel("Crime Type")
plt.show()
```

```
Unique Crime Categories:
['VEHICLE - STOLEN' 'BURGLARY FROM VEHICLE' 'BIKE - STOLEN'
 'SHOPLIFTING-GRAND THEFT ($950.01 & OVER)' 'ARSON' 'BURGLARY' 'PIMPING'
 'PANDERING' 'OTHER MISCELLANEOUS CRIME'
 'VANDALISM - MISDEAMEANOR ($399 OR UNDER)'
 'INTIMATE PARTNER - SIMPLE ASSAULT' 'ROBBERY'
 'THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD'
 'ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT' 'THEFT OF IDENTITY'
```

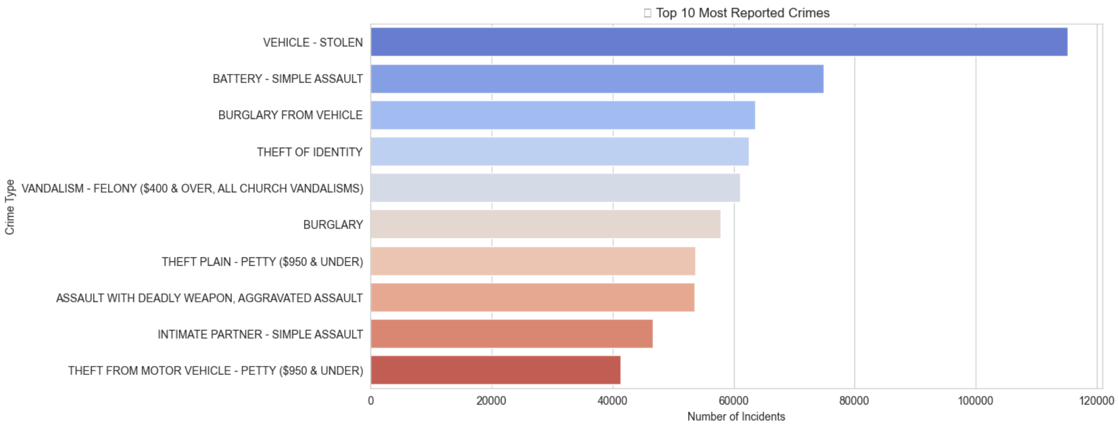
```
'BATTERY - SIMPLE ASSAULT' 'SHOPLIFTING - PETTY THEFT ($950 & UNDER)'
'BUNCO, GRAND THEFT' 'VIOLATION OF COURT ORDER'
'VIOLATION OF RESTRAINING ORDER' 'THEFT PLAIN - PETTY ($950 & UNDER)'
'VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)'
'RAPE, FORCIBLE' 'THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)'
'TRESPASSING' 'VEHICLE - ATTEMPT STOLEN' 'RESISTING ARREST'
'EMBEZZLEMENT, GRAND THEFT ($950.01 & OVER)'
'BURGLARY FROM VEHICLE, ATTEMPTED'
'LETTERS, LEWD  -  TELEPHONE CALLS, LEWD'
'CRIMINAL THREATS - NO WEAPON DISPLAYED'
'SEX OFFENDER REGISTRANT OUT OF COMPLIANCE'
'UNAUTHORIZED COMPUTER ACCESS'
'THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)'
'CRM AGNST CHLD (13 OR UNDER) (14-15 & SUSP 10 YRS OLDER)'
'BRANDISH WEAPON' 'BURGLARY, ATTEMPTED' 'DISCHARGE FIREARMS/SHOTS FIRED'
'BATTERY POLICE (SIMPLE)'
'VEHICLE, STOLEN - OTHER (MOTORIZED SCOOTERS, BIKES, ETC)'
'ORAL COPULATION' 'INDECENT EXPOSURE' 'THEFT FROM PERSON - ATTEMPT'
'CHILD ABUSE (PHYSICAL) - SIMPLE ASSAULT' 'OTHER ASSAULT'
'DISTURBING THE PEACE' 'INTIMATE PARTNER - AGGRAVATED ASSAULT'
'BOMB SCARE' 'FAILURE TO YIELD' 'CONTEMPT OF COURT' 'ATTEMPTED ROBBERY'
'ASSAULT WITH DEADLY WEAPON ON POLICE OFFICER'
'DOCUMENT FORGERY / STOLEN FELONY' 'BUNCO, PETTY THEFT'
'SEXUAL PENETRATION W/FOREIGN OBJECT' 'SHOTS FIRED AT INHABITED DWELLING'
'CHILD STEALING' 'DEFRAUDING INNKEEPER/THEFT OF SERVICES, $950 & UNDER'
'KIDNAPPING - GRAND ATTEMPT'
'SHOTS FIRED AT MOVING VEHICLE, TRAIN OR AIRCRAFT' 'THEFT, PERSON'
'CHILD ABUSE (PHYSICAL) - AGGRAVATED ASSAULT' 'EXTORTION'
'CHILD NEGLECT (SEE 300 W.I.C.)'
'TILL TAP - GRAND THEFT ($950.01 & OVER)'
'SEX,UNLAWFUL(INC MUTUAL CONSENT, PENETRATION W/ FRGN OBJ'
'BATTERY WITH SEXUAL CONTACT' 'HUMAN TRAFFICKING - COMMERCIAL SEX ACTS'
'CHILD ANNOYING (17YRS & UNDER)' 'DOCUMENT WORTHLESS ($200.01 & OVER)'
'RAPE, ATTEMPTED' 'FALSE IMPRISONMENT'
'THROWING OBJECT AT MOVING VEHICLE' 'LEWD CONDUCT' 'PEEPING TOM'
'KIDNAPPING' 'CRIMINAL HOMICIDE' 'STALKING' 'THEFT PLAIN - ATTEMPT'
'SODOMY/SEXUAL CONTACT B/W PENIS OF ONE PERS TO ANUS OTH'
'VIOLATION OF TEMPORARY RESTRAINING ORDER' 'CHILD PORNOGRAPHY'
'WEAPONS POSSESSION/BOMBING' 'DRIVING WITHOUT OWNER CONSENT (DWOC)'
'THEFT FROM MOTOR VEHICLE - ATTEMPT' 'PICKPOCKET' 'SHOPLIFTING - ATTEMPT'
'COUNTERFEIT' 'BUNCO, ATTEMPT'
'DEFRAUDING INNKEEPER/THEFT OF SERVICES, OVER $950.01'
'CRUELTY TO ANIMALS' 'FALSE POLICE REPORT' 'PROWLER'
'DISHONEST EMPLOYEE - GRAND THEFT' 'THREATENING PHONE CALLS/LETTERS'
'PURSE SNATCHING' 'EMBEZZLEMENT, PETTY THEFT ($950 & UNDER)'
'DOCUMENT WORTHLESS ($200 & UNDER)' 'ILLEGAL DUMPING'
'LEWD/LASCIVIOUS ACTS WITH CHILD' 'BATTERY ON A FIREFIGHTER'
'PETTY THEFT - AUTO REPAIR' 'MANSLAUGHTER, NEGLIGENT' 'RECKLESS DRIVING'
```

```
 'TILL TAP - PETTY ($950 & UNDER)' 'PURSE SNATCHING - ATTEMPT'
 'LYNCHING - ATTEMPTED' 'CREDIT CARDS, FRAUD USE ($950.01 & OVER)'
 'CREDIT CARDS, FRAUD USE ($950 & UNDER'
 'THEFT, COIN MACHINE - PETTY ($950 & UNDER)'
 'HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE' 'BIKE - ATTEMPTED STOLEN'
 'CONTRIBUTING' 'BRIBERY' 'BOAT - STOLEN' 'CONSPIRACY'
 'GRAND THEFT / INSURANCE FRAUD' 'DRUGS, TO A MINOR' 'CHILD ABANDONMENT'
 'THEFT, COIN MACHINE - GRAND ($950.01 & OVER)' 'DISRUPT SCHOOL'
 'THEFT, COIN MACHINE - ATTEMPT' 'DISHONEST EMPLOYEE - PETTY THEFT'
 'LYNCHING' 'FIREARMS RESTRAINING ORDER (FIREARMS RO)'
 'REPLICA FIREARMS(SALE,DISPLAY,MANUFACTURE OR DISTRIBUTE)'
 'GRAND THEFT / AUTO REPAIR' 'DRUNK ROLL' 'PICKPOCKET, ATTEMPT'
 'TELEPHONE PROPERTY - DAMAGE'
 'BEASTIALITY, CRIME AGAINST NATURE SEXUAL ASSLT WITH ANIM' 'BIGAMY'
 'FAILURE TO DISPERSE'
 'FIREARMS EMERGENCY PROTECTIVE ORDER (FIREARMS EPO)'
 'INCEST (SEXUAL ACTS BETWEEN BLOOD RELATIVES)'
 'BLOCKING DOOR INDUCTION CENTER' 'INCITING A RIOT'
 'DISHONEST EMPLOYEE ATTEMPTED THEFT' 'TRAIN WRECKING'
 'DRUNK ROLL - ATTEMPT']
```

```
C:\Users\Gaurav Chugh\AppData\Local\Temp\ipykernel_32576\1428609675.py:9:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=crime_counts.values, y=crime_counts.index, palette="coolwarm")
C:\Users\Gaurav Chugh\AppData\Roaming\Python\Python312\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128269 (\N{LEFT-
POINTING MAGNIFYING GLASS}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



[8]:

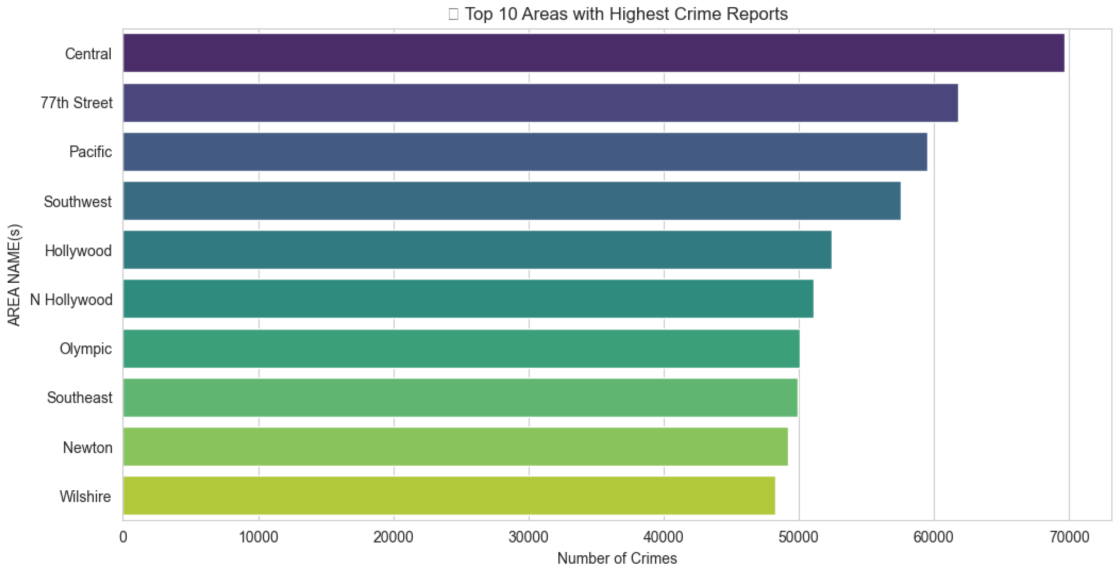
```python
import matplotlib.pyplot as plt
import seaborn as sns
# Top 10 areas with highest crime reports
area_counts = df["AREA NAME"].value_counts().head(10)

plt.figure(figsize=(12, 6))
sns.barplot(x=area_counts.values, y=area_counts.index, palette="viridis")
plt.title("  Top 10 Areas with Highest Crime Reports")
plt.xlabel("Number of Crimes")
plt.ylabel("AREA NAME(s)")
plt.show()
```

```
C:\Users\Gaurav Chugh\AppData\Local\Temp\ipykernel_32576\2063246029.py:7:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=area_counts.values, y=area_counts.index, palette="viridis")
C:\Users\Gaurav Chugh\AppData\Roaming\Python\Python312\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 127750 (\N{CITYSCAPE
AT DUSK}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



0.5 Temporal Analysis

We analyze: - **Crimes by Year** - **Crimes by Month** Charts will help visualize crime trends over time.

[10]:

```
# Convert date column to pandas datetime format
df["Date"] = pd.to_datetime(df["DATE OCC"])

# Extract year and month
df["Year"] = df["Date"].dt.year
df["Month"] = df["Date"].dt.month
```

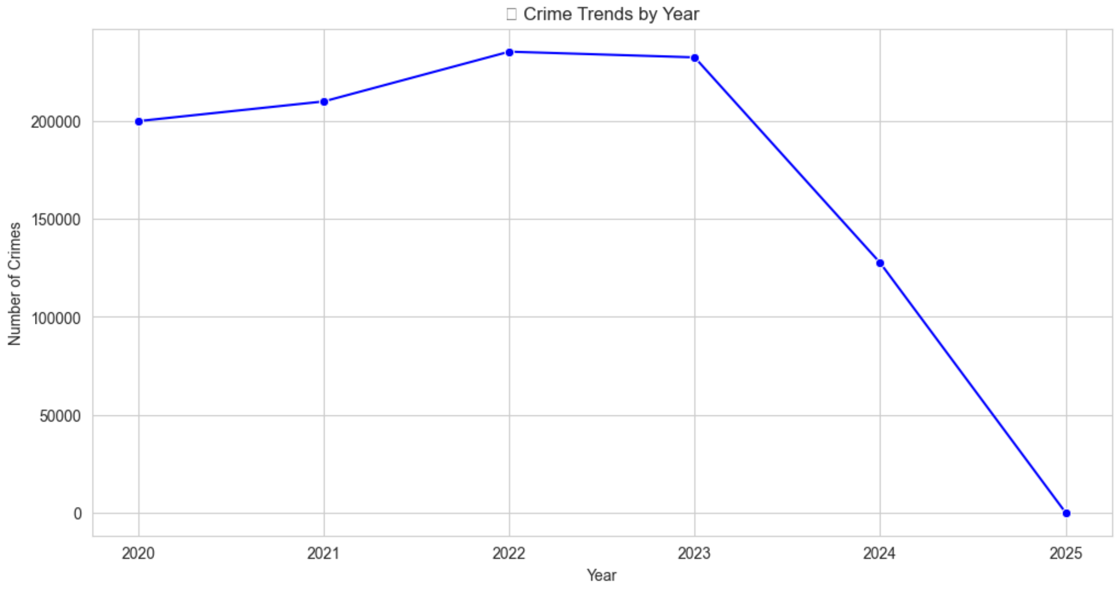
```
C:\Users\Gaurav Chugh\AppData\Local\Temp\ipykernel_32576\1132031127.py:2:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  df["Date"] = pd.to_datetime(df["DATE OCC"])
```

[11]:

```
# Crimes per Year
yearly_crimes = df["Year"].value_counts().sort_index()

plt.figure(figsize=(12, 6))
sns.lineplot(x=yearly_crimes.index, y=yearly_crimes.values, marker="o",␣
  ↪color="blue")
plt.title("  Crime Trends by Year")
plt.xlabel("Year")
plt.ylabel("Number of Crimes")
plt.show()
```

```
C:\Users\Gaurav Chugh\AppData\Roaming\Python\Python312\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128198 (\N{TEAR-OFF
CALENDAR}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



[12]:

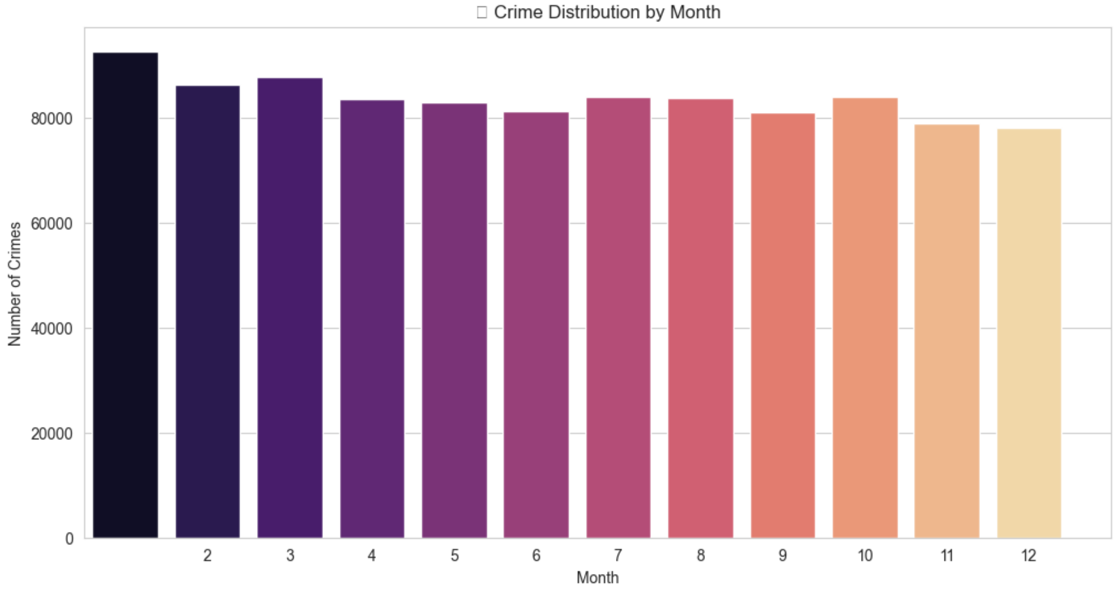
```
# Crimes per Month
monthly_crimes = df["Month"].value_counts().sort_index()

plt.figure(figsize=(12, 6))
sns.barplot(x=monthly_crimes.index, y=monthly_crimes.values, palette="magma")
plt.title("  Crime Distribution by Month")
plt.xlabel("Month")
plt.ylabel("Number of Crimes")
plt.xticks(range(1, 13))
plt.show()
```

```
C:\Users\Gaurav Chugh\AppData\Local\Temp\ipykernel_32576\1320578419.py:5:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=monthly_crimes.index, y=monthly_crimes.values, palette="magma")
C:\Users\Gaurav Chugh\AppData\Roaming\Python\Python312\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128202 (\N{BAR
CHART}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



0.5.1 3 Data Preprocessing

- Use Loaded dataset
- Rename columns (**convert to lowercase, replace spaces with _**)

- Convert date columns (`DATE OCC`, `Date Rptd`) to string format (`YYYYMMDD`).

[13]:
```python
# Rename columns for consistency
df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")

# Convert date columns to 'YYYYMMDD' format
df['date_occurred'] = pd.to_datetime(df['date_occ']).dt.strftime("%Y%m%d")
df['date_reported'] = pd.to_datetime(df['date_rptd']).dt.strftime("%Y%m%d")

print("  Data cleaned successfully!")
```

```
C:\Users\Gaurav Chugh\AppData\Local\Temp\ipykernel_32576\2624637903.py:5:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  df['date_occurred'] = pd.to_datetime(df['date_occ']).dt.strftime("%Y%m%d")
C:\Users\Gaurav Chugh\AppData\Local\Temp\ipykernel_32576\2624637903.py:6:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  df['date_reported'] = pd.to_datetime(df['date_rptd']).dt.strftime("%Y%m%d")
```

```
 Data cleaned successfully!
```

0.5.2 4 Map Columns to Column Families

Assign each column to its respective column family (`location`, `crime_info`).

[14]:
```python
COLUMN_FAMILIES = {
    'location': ['location', 'cross_street', 'lat', 'lon'],
    'crime_info': ['dr_no', 'date_reported', 'date_occurred', 'area', 'crm_cd',␣
 ↪'crm_cd_desc', 'vict_age', 'vict_sex']
}

print("  Column mappings set!")
```

```
 Column mappings set!
```

0.5.3 5 Implement RowKey Strategy

Create an optimized rowkey format: `YYYYMMDD_DR_NO` → Example: `20200301_190326475`

[15]:
```python
# Create efficient rowkeys
df['rowkey'] = df['date_occurred'] + "_" + df['dr_no'].astype(str)
print("  RowKeys generated successfully!")
```

```
 RowKeys generated successfully!
```

0.5.4 6 Efficient Data Insertion into HBase

Insert data with batching (`batch_size=1000`) & skip null values (`NA`).

[16]:

```
import happybase

# Connect to HBase
connection = happybase.Connection('localhost')  # Use 'hbase' if inside Docker
connection.open()

# Access the practice:crimes table
table = connection.table('practice2:crimes')

print("  Connected to HBase successfully!")
def push_to_hbase(table, df):
    batch = table.batch(batch_size=1000)  # Batch processing

    for _, row in df.iterrows():
        rowkey = row['rowkey']
        hbase_data = {}

        for cf, cols in COLUMN_FAMILIES.items():
            for col in cols:
                if pd.notna(row[col]):  # Only insert non-null values
                    hbase_data[f"{cf}:{col}"] = str(row[col])

        batch.put(rowkey, hbase_data)

    batch.send()
    print("  Data inserted into HBase successfully!")

# Push the first 500,000 rows to HBase
push_to_hbase(table, df.head(500000))
```

```
 Connected to HBase successfully!
 Data inserted into HBase successfully!
```

[17]:

```
!echo "count 'practice:crimes'" | hbase shell
```

```
'hbase' is not recognized as an internal or external command,
operable program or batch file.
```