Crime Against on women

This dataset provides a comprehensive overview of crime statistics reported across various Indian states over a series of years, beginning from 2001. Each record represents the number of reported cases for a specific type of crime in a particular state and year. The structured format of this data allows for in-depth analysis of crime trends over time, regional comparisons, and insights into the prevalence of different types of criminal activities. This dataset is particularly useful for data visualization, policy assessment, and criminological research.

State: The name of the Indian state or union territory where the crimes were reported. - **Year**: The year in which the crimes were recorded. - **Kidnap And Assault**:Unlawful taking or abduction of a person against their will, typically to demand ransom or exert pressure. - **Dowry Deaths**:the death of a woman caused by harassment or violence related to dowry demands. - **assault against women**:criminal acts involving physical or sexual violence, threats, or force directed specifically at women - **Assault against modesty of women**:a legal and statistical term used in many crime records (especially in India) to refer to actions that violate the personal dignity, privacy, and decency of a woman, without necessarily involving physical violence like in rape or grievous assault. - **Domestic violence**:physical, emotional, sexual, or economic abuse that occurs within a domestic setting — typically between intimate partners, family members, or individuals living in the same household. - **Women Trafficking**:

```
#Importing Libraries
import os
import pandas as pd
import numpy as np
```

Define and Create Directory Paths

To ensure reproducibility andorganized storage, we programmatically create directories for:

- **raw data**
- **processed data**
- **results**
- **documentation**

These directories will store intermediate and final outputs for reproducibility.

```
#get working directories
current_dir = os.getcwd()
#Go one directory up to the root directory
project_root_dir = os.path.dirname(current_dir)
project_root_dir
# Define paths to the data folders
data_dir = os.path.join(project_root_dir, "Data")
raw_dir = os.path.join(data_dir, "raw")
processed_dir = os.path.join(data_dir, "processed")
# Define paths to results folder
results_dir = os.path.join(project_root_dir, "results")
#define paths to the docs folder
docs_dir = os.path.join(project_root_dir,"docs")

# Creates directories if they do not exist
os.makedirs(raw_dir, exist_ok = True)
os.makedirs(processed_dir, exist_ok = True)
os.makedirs(results_dir, exist_ok = True)
os.makedirs(docs_dir, exist_ok = True)
```

Loading the Dataset

We load the **crime against woman data.csv** as a CSV file.

-we load the **description.csv**

Key considerations here are: we create some columns name from short into long columns name

```
crimes_df = pd.read_csv(r"C:\Users\user\Downloads\Crime Against Woman\CrimesOnWomenData.csv"
description_df = pd.read_csv(r"C:\Users\user\Downloads\Crime Against Woman\description.csv")
crimes_df.head(), description_df.head()
```

```
(   Unnamed: 0              State  Year  Rape   K&A   DD   AoW   AoM    DV  WT
 0           0     ANDHRA PRADESH  2001   871   765  420  3544  2271  5791   7
 1           1  ARUNACHAL PRADESH  2001    33    55    0    78     3    11   0
 2           2              ASSAM  2001   817  1070   59   850     4  1248   0
 3           3              BIHAR  2001   888   518  859   562    21  1558  83
 4           4       CHHATTISGARH  2001   959   171   70  1763   161   840   0,
    Unnamed: 0 Column Names         Explanation
 0           0        State               State
 1           1         Year                Year
 2           2         Rape   No. of Rape cases
 3           3          K&A  Kidnap And Assault
 4           4           DD        Dowry Deaths)
```

We also inspect the dataset's shape. We see that the data has 736 rows and 10 columns.

```
crimes_df.shape
```

```
(736, 10)
```

In addition, we check also the data types using .info.

```
crimes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 736 entries, 0 to 735
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  736 non-null    int64
 1   State       736 non-null    object
 2   Year        736 non-null    int64
 3   Rape        736 non-null    int64
 4   K&A         736 non-null    int64
 5   DD          736 non-null    int64
 6   AoW         736 non-null    int64
 7   AoM         736 non-null    int64
 8   DV          736 non-null    int64
 9   WT          736 non-null    int64
dtypes: int64(9), object(1)
memory usage: 57.6+ KB
```

```
crimes_df.columns
```

```
Index(['Unnamed: 0', 'State', 'Year', 'Rape', 'K&A', 'DD', 'AoW', 'AoM', 'DV',
       'WT'],
      dtype='object')
```

Data Cleaning and Transformation Process

1. Initial Raw Dataset

We started with a raw dataset containing information on various crimes reported against women across Indian states and union territories from 2001 to 2021. However, this dataset was not ready for analysis because:

- It contained **missing or inconsistent values** (e.g., `NaN`)
- Some **text columns** had extra spaces, inconsistent casing, or misspellings
- The data was presented in a **wide format** (each year as a separate column)

---

2. Cleaning Steps Applied

a. Handled Missing Values

- We identified and filled or removed rows with missing values depending on their impact.
- For important columns like `State/UT` or `Crime Type`, missing values were filled with `"unknown"` or dropped using `dropna()` if needed.

b. Standardized Text Data

- Removed extra spaces using `.str.strip()`
- Converted all categorical text to lowercase using `.str.lower()` to maintain consistency
- Renamed similar values to unified categories (e.g., `"Andhra Pradesh "` → `"andhra pradesh"`)

c. Renamed Columns

- Renamed columns to more descriptive names using:

~~~
df.columns = ['state_ut', 'crime_type', '2001', '2002', ..., '2021']

::: {.cell execution_count=16}
``` {.python .cell-code}
# Checking if there are missing values
crimes_df.isnull().sum().sum()
~~~

```
0
```

:::

```
# Checking for duplicate
crimes_df.duplicated().sum()
```

```
0
```

Manually define the short and long column names

Raw data column names can be very long, complicated, or not user-friendly. Manually defining short column names makes it easier to reference them in code or visualizations without confusion.

```
# Manually define the short and long column names as lists
short_names = ['State', 'Year', 'Rape', 'K&A', 'DD', 'AoM', 'AoW', 'DV', 'WT']
long_names = ['State', 'Year', 'No. of Rape cases', 'Kidnap And Assault', 'Dowry Deaths',
              'Assault against modesty of women', 'Assault against women',
              'Domestic violence', 'Women Trafficking']

# Create a mapping using zip()
column_mapping = dict(zip(short_names, long_names))

# Apply the mapping to rename columns
crimes_df.rename(columns=column_mapping, inplace=True)
crimes_df
```

	Unnamed: 0	State	Year	No. of Rape cases	Kidnap And Assault	Dowry De
0	0	ANDHRA PRADESH	2001	871	765	420
1	1	ARUNACHAL PRADESH	2001	33	55	0
2	2	ASSAM	2001	817	1070	59
3	3	BIHAR	2001	888	518	859
4	4	CHHATTISGARH	2001	959	171	70
...	...	...	...	...	...	...
731	731	D&N Haveli	2021	1250	4083	141
732	732	Daman & Diu	2021	315	904	16
733	733	Delhi UT	2021	2	1	0
734	734	Lakshadweep	2021	0	0	0
735	735	Puducherry	2021	2	0	2

	Unnamed: 0	State	Year	No. of Rape cases	Kidnap And Assault	Dowry De

```python
crimes_df.to_csv('Cleaned_crimes_on_women.csv', index=False)
```

```python
# Convert all state names to lowercase
crimes_df['State'] = crimes_df['State'].str.lower()

# Display unique states to verify transformation
crimes_df['State'].unique()
```

```
array(['andhra pradesh', 'arunachal pradesh', 'assam', 'bihar',
       'chhattisgarh', 'goa', 'gujarat', 'haryana', 'himachal pradesh',
       'jammu & kashmir', 'jharkhand', 'karnataka', 'kerala',
       'madhya pradesh', 'maharashtra', 'manipur', 'meghalaya', 'mizoram',
       'nagaland', 'odisha', 'punjab', 'rajasthan', 'sikkim',
       'tamil nadu', 'tripura', 'uttar pradesh', 'uttarakhand',
       'west bengal', 'a & n islands', 'chandigarh', 'd & n haveli',
       'daman & diu', 'lakshadweep', 'puducherry', 'telangana',
       'd&n haveli', 'delhi ut'], dtype=object)
```

```python
crimes_df
```

	Unnamed: 0	State	Year	No. of Rape cases	Kidnap And Assault	Dowry Deaths	A
0	0	andhra pradesh	2001	871	765	420	35
1	1	arunachal pradesh	2001	33	55	0	78
2	2	assam	2001	817	1070	59	85
3	3	bihar	2001	888	518	859	56
4	4	chhattisgarh	2001	959	171	70	17
...	...	...	...	...	...	...	...
731	731	d&n haveli	2021	1250	4083	141	20
732	732	daman & diu	2021	315	904	16	18
733	733	delhi ut	2021	2	1	0	5
734	734	lakshadweep	2021	0	0	0	1
735	735	puducherry	2021	2	0	2	31

Checking duplicate

Finding duplicates is an essential part of data cleaning and preprocessing before doing any analysis.

and there is no duplicate

```
crimes_df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
731    False
732    False
733    False
734    False
735    False
Length: 736, dtype: bool
```

Remove Unnecessary Column

During the cleaning process, we identified a column that was not relevant to our analysis and did not add meaningful value. To maintain a clean and focused dataset, we decided to remove this column.

- Reduces noise in the data
- Improves performance during analysis
- Makes visualizations and summaries more clear

```
crimes_df.drop(columns=['total_crimes'], inplace=True)
```

```
crimes_df
```

	State	Year	No. of Rape cases	Kidnap And Assault	Dowry Deaths	Assault against
0	andhra pradesh	2001	871	765	420	3544
1	arunachal pradesh	2001	33	55	0	78
2	assam	2001	817	1070	59	850
3	bihar	2001	888	518	859	562
4	chhattisgarh	2001	959	171	70	1763
…	…	…	…	…	…	…

	State	Year	No. of Rape cases	Kidnap And Assault	Dowry Deaths	Assault against
731	d&n haveli	2021	1250	4083	141	2068
732	daman & diu	2021	315	904	16	1851
733	delhi ut	2021	2	1	0	5
734	lakshadweep	2021	0	0	0	1
735	puducherry	2021	2	0	2	31

Reshape Data from Wide Format into Long Format

The original dataset was in **wide format**, where each year (e.g., 2001, 2002, …, 2021) was represented as a separate column. This made it difficult to visualize and analyze time-based trends.

We reshaped the dataset into **long format** using the `pd.melt()` function.

This transformation was important because it:

- Makes it easier to create time-series visualizations
- Helps us compare values across different years more effectively
- Simplifies grouping and filtering by year or crime type
- Organizes the data in a more analysis-friendly structure

```
# Melt (reshape) the data
df_long = crimes_df.melt(id_vars=["State", "Year"],
                  var_name="Crime Type",
                  value_name="Value")

# Save reshaped data
final_file = os.path.join(processed_dir, 'reshaped_data.csv')
df_long.to_csv("reshaped_data.csv", index=False)
```

Save the Reshaped Dataset to CSV

After successfully cleaning and reshaping the dataset from wide to long format, we saved the final version to a CSV file named `reshaped.csv`.

This step ensures that the cleaned and structured data is:

- Stored for future use without needing to repeat the cleaning steps

- Ready for further analysis, visualizations, or modeling
- Easily shareable with others or usable in other tools like Excel, Tableau, or Power BI

```
reshaped_df=pd.read_csv(r"C:\Users\user\Downloads\Crime\crime_against_on_womens\Data\process
reshaped_df
```

	State	Year	Crime Type	Value
0	andhra pradesh	2001	No. of Rape cases	871
1	arunachal pradesh	2001	No. of Rape cases	33
2	assam	2001	No. of Rape cases	817
3	bihar	2001	No. of Rape cases	888
4	chhattisgarh	2001	No. of Rape cases	959
...	...	...	...	...
5147	d&n haveli	2021	Women Trafficking	4
5148	daman & diu	2021	Women Trafficking	1
5149	delhi ut	2021	Women Trafficking	0
5150	lakshadweep	2021	Women Trafficking	0
5151	puducherry	2021	Women Trafficking	0