3.1 DATA CLEANING :-

Data Cleaning :

Data cleansing or **data cleaning** is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. Data cleansing may be performed interactively with data wrangling tools, oras batch processing through scripting.

After cleansing, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores. Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.

➢ Missing Value

Before we dive into code, it's important to understand the sources of missing data. Here some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

➢ Why missing values treatment is required?

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behaviour and relationship with other variables correctly. It can lead to wrong prediction or classification.

➢ Types of Missing Value

1. **Standard Missing Value :**
   Standard missing values are missing data that pandas can detect

2. **Non Standard Missing Value:**
   Sometimes it might be the case where there's missing values that have different formats.

3. **Unexpected missing Values:**
   It means that data are in not in actual format. For example it is expected to be a string, but there's a numeric type, then technically it is also a missing value

Methods of Missing Data

Identify the missing values:

1. Find missing values

Command:

```
print(cc.isnull())
```

Output:

```
      STATE/UT  CRIME HEAD  ...  Above 60 Years  Total
0        False       False  ...           False   True
1        False       False  ...           False  False
2        False       False  ...           False  False
3        False       False  ...           False  False
4        False       False  ...           False  False
5        False       False  ...           False  False
6        False       False  ...           False  False
7        False       False  ...           False  False
8        False       False  ...           False  False
9        False       False  ...           False  False
10       False       False  ...           False  False
```

2. **Count missing values**

Command:

```
print(cc.isnull().sum())
```

Output:

```
STATE/UT                      0
CRIME HEAD                    0
YEAR                          0
Below 18 Years                0
Between 18-30 Years           0
Between 30-45 Years           0
Between 45-60 Years           0
Above 60 Years                0
Total                         1
dtype: int64
```

3. **Fill the missing data**

Command:

```
fil=cc.fillna('Total')

print(fil.head())
```

Output:

```
            STATE/UT  ...    Total
0     ANDHRA PRADESH   ...    Total
1  ARUNACHAL PRADESH   ...        0
2               ASSAM  ...        0
3               BIHAR  ...        0
4        CHHATTISGARH  ...        0

[5 rows x 9 columns]
```

4. **Replace the missing value**

Command:

```
cc['Total'].fillna('100', inplace=True)

print(cc)
```

Output:

```
                STATE/UT   ...   Total
0         ANDHRA PRADESH   ...     100
1      ARUNACHAL PRADESH   ...       0
2                  ASSAM   ...       0
3                  BIHAR   ...       0
4           CHHATTISGARH   ...       0
5                    GOA   ...       0
6                GUJARAT   ...       5
7                HARYANA   ...       0
8       HIMACHAL PRADESH   ...       0
9        JAMMU & KASHMIR   ...       0
10             JHARKHAND   ...       0
```

3.2 DATA TRANSFORMATION :-

Data Transformation :

Data transformation is the process of converting data from one format or structure into another format or structure. Data transformation is critical to activities such as data integration and data management. Data transformation can include a range of activities: you might convert data types, cleanse data by removing nulls or duplicate data, enrich the data, or perform aggregations, depending on the needs of your project.

Typically, the process involves two stages.

In the first stage, you:

- Perform data discovery where you identify the sources and data types.
- Determine the structure and data transformations that need to occur.
- Perform data mapping to define how individual fields are mapped, modified, joined, filtered, and aggregated.

In the second stage, you:

- Extract data from the original source. The range of sources can vary, including structured sources, like databases, or streaming sources, such as telemetry from connected devices, or log files from customers using your web applications.
- Perform transformations. You transform the data, such as aggregating sales data or converting date formats, editing text strings or joining rows and columns.
- Send the data to the target store. The target might be a database or a data warehouse that handles structured and unstructured data.

➢ **Drop Duplicate Record :**

- **Duplicate Record :**

  Duplicate record is a data which is same as other row

Command:

```
clean=cc.dropna()
print(clean)
```

Output:

	STATE/UT	...	Total
1	ARUNACHAL PRADESH	...	0.0
2	ASSAM	...	0.0
3	BIHAR	...	0.0
4	CHHATTISGARH	...	0.0
5	GOA	...	0.0
6	GUJARAT	...	5.0
7	HARYANA	...	0.0
8	HIMACHAL PRADESH	...	0.0
9	JAMMU & KASHMIR	...	0.0
10	JHARKHAND	...	0.0

➢ **Drop field :**

- Droping unnecessary field (unwanted columns)

Command:

```
to_drop = ['Total']
cc.drop(to_drop, inplace=True, axis=1)
print(cc)
```

Output:

	STATE/UT	...	Above 60 Years
0	ANDHRA PRADESH	...	0
1	ARUNACHAL PRADESH	...	0
2	ASSAM	...	0
3	BIHAR	...	0
4	CHHATTISGARH	...	0
5	GOA	...	0
6	GUJARAT	...	0
7	HARYANA	...	0
8	HIMACHAL PRADESH	...	0
9	JAMMU & KASHMIR	...	0
10	JHARKHAND	...	0

3.3 DATA DISCRIMINATION :-

Data discrimination, also called discrimination by algorithm, is bias that occurs when predefined data types or data sources are intentionally or unintentionally treated differently than others.

Data discrimination includes the censorship of lawful information by an internet service provider (ISP). Throughout the last decade, ISPs have been criticized for allegedly quashing competition, promoting or discouraging particular political ideologies or religious beliefs and blocking union websites during employee labour disputes.

As per our dataset cyber crime our data is present in nominal data only so, we don't have to transform the data set.

- **Convert data set in to Lower case**

Command :

```
cc["STATE/UT"]=cc["STATE/UT"].str.lower()

print(cc)
```

Output:

```
              STATE/UT  ...  Total
0       andhra pradesh  ...      0
1    arunachal pradesh  ...      0
2                assam  ...      0
3                bihar  ...      0
4         chhattisgarh  ...      0
5                  goa  ...      0
6              gujarat  ...      5
7              haryana  ...      0
8     himachal pradesh  ...      0
9      jammu & kashmir  ...      0
10           jharkhand  ...      0
```

- **Convert data set in to Upper case**

Command :

```
cc["STATE/UT"]=cc["STATE/UT"].str.upper()
print(cc)
```

Output:

	STATE/UT	...	Total
0	ANDHRA PRADESH	...	0
1	ARUNACHAL PRADESH	...	0
2	ASSAM	...	0
3	BIHAR	...	0
4	CHHATTISGARH	...	0
5	GOA	...	0
6	GUJARAT	...	5
7	HARYANA	...	0
8	HIMACHAL PRADESH	...	0
9	JAMMU & KASHMIR	...	0
10	JHARKHAND	...	0

- **Access data set in Ascending Order**

Command:

```
Asending=cc.sort_values(by ='STATE/UT',ascending=True)[1:11]
print(Asending)
```

Output:

```
           STATE/UT  ...  Total
2271  A & N ISLANDS  ...      0
2917  A & N ISLANDS  ...      0
143   A & N ISLANDS  ...      0
3981  A & N ISLANDS  ...      0
1093  A & N ISLANDS  ...      0
1359  A & N ISLANDS  ...      0
333   A & N ISLANDS  ...      0
3145  A & N ISLANDS  ...      0
1435  A & N ISLANDS  ...      0
2233  A & N ISLANDS  ...      0

[10 rows x 9 columns]
```

- **Access data set in Descending Order**

Command :

```
stat=cc.sort_values(by ='STATE/UT',ascending=False)[1:11]
print(stat)
```

Output:

```
         STATE/UT  ...  Total
2687  WEST BENGAL  ...      0
2383  WEST BENGAL  ...      0
3485  WEST BENGAL  ...     27
3219  WEST BENGAL  ...      0
1547  WEST BENGAL  ...      0
4169  WEST BENGAL  ...     39
1661  WEST BENGAL  ...     21
1167  WEST BENGAL  ...      0
2041  WEST BENGAL  ...      0
1395  WEST BENGAL  ...      0

[10 rows x 9 columns]
```