

# **VISUALIZING US NATURAL DIASTER DECLARATION – TRENDS AND PATTERNS**

## **Week 2 Documentation**

### **Data Cleaning in Python (Pandas)**

## Data Cleaning in Python (Pandas)

### 1. Introduction

Week 2 focused on replicating the data cleaning workflow from Power BI in Python, using the **Pandas** library. The objective was to automate transformations, and prepare the FEMA Disaster Declarations dataset for analysis. **Python scripts provide flexibility for handling larger datasets and integrating advanced preprocessing steps.**

### 2. Step by Step Procedure

#### 2.1 Importing Data

- **Source:** FEMA Disaster Declarations dataset (CSV format).
- **Code:**

```
import pandas as pd

df = pd.read_csv("fema_disaster_declarations.csv")
```
- **Outcome:** Dataset loaded into a Pandas DataFrame for cleaning.

#### 2.2 Removing Unnecessary Columns

- **To Do:** Drop metadata and unused fields.
- **Procedure:**

```
df = df.drop(columns = [
    "stateCode", "disasterPageUrl", "shapefileUrl",
    "kmzfileUrl", "geoJsonUrl", "id", "hash", "lastRefresh"
])
```
- **Outcome:** Data fields not used in analysis is removed.

#### 2.3 Checking Data Types

- **To Do:** Ensure correct data types for each column.
- **Procedure:**

```
df["disasterNumber"] = df["disasterNumber"].astype(int)

date_cols = ["declarationDate", "incidentBeginDate", "incidentEndDate",
    "entryDate", "closeoutDate", "updateDate"]
```

```
for col in date_cols:
```

```
    df[col] = pd.to_datetime(df[col], errors="coerce")
```

- **Outcome:** Consistent data types across numeric, text, and date fields.

## 2.4 Handling Missing Values

- **Procedure:**

- **Handle missing dates:**

Retain closeoutDate as null to indicate ongoing disasters. Null values in closeoutDate is retained to indicate ongoing disasters. A derived status column is added to categorize records as 'Closed' or 'Open'.

**Add a flag column:**

```
df["status"] = df["closeoutDate"].apply( lambda x: "Closed" if
pd.notnull(x) else "Open" )
```

- **Outcome:** Missing values are handled.

## 2.5 Creating Derived Columns

- **Fiscal Year:**

```
df["fyDeclared"] = df["declarationDate"].apply( lambda x: x.year+1 if
x.month > 9 else x.year )
```

- **Incident Duration:**

```
df["incidentDuration"] = ( df["incidentEndDate"] -
df["incidentBeginDate"]).dt.days )
```

- **Assistance Flags:**

```
flag_cols = ["iaProgramDeclared","ihProgramDeclared",
"paProgramDeclared","hmProgramDeclared"]
```

```
for col in flag_cols:
```

```
    df[col] = df[col].astype(bool)
```

## 2.6 Removing Duplicates

- **Procedure:**

```
df = df.drop_duplicates()
```

- **Outcome:** Ensures unique records, prevents double counting.

## 2.7 Exporting Cleaned Data

- **Procedure:**

```
df.to_csv("fema_cleaned.csv", index=False)
```

- **Outcome:** Cleaned dataset saved for visualization and further analysis.

## 3. Outcome of Week 2

By the end of Week 2:

- The FEMA dataset was cleaned and standardized using Python (Pandas).
- Derived fields (Fiscal Year, Incident Duration, Assistance Flags, Status) were created.
- The workflow is now script-based, and scalable and can be automated for larger datasets.

## 4. Next Steps (Week 3 Preview)

- Begin **Exploratory Data Analysis (EDA)** in Python.
- Generate summary statistics, distributions, and initial trend visualizations.
- Compare outputs with Power BI dashboards for consistency.