1)DATA CLEANING STEPS:

Steps Involved in Cleaning Netflix Data

Step 1: Import Required Libraries

- Imported essential Python libraries such as:
  - **Pandas** – for data handling and cleaning.
  - **NumPy** – for numerical operations.
- These libraries provide powerful tools for managing and transforming datasets.

Step 2: Load the Dataset

- Used pd.read_csv('netflix_titles.csv') to load the dataset into a DataFrame.

Step 3: Explore the Dataset

- Displayed the first few rows using df.head() and examined the structure with df.info() and df.describe().
- Helps identify:
  - Data types of each column
  - Missing values

Step 4: Handle Missing Data

- Checked for missing values using df.isnull().sum().
- Replaced *'Unknown'* entries with NaN for consistency.
- Filled or dropped missing data depending on the column importance:
  - Filled director, cast, country, rating, date_added, and duration columns with appropriate values.
  - Dropped rows or columns with excessive missing data (>50%).

Step 5: Remove Duplicate Records

- Removed duplicates using df.drop_duplicates() to maintain data quality and prevent repetition in analysis.

Step 6: Convert Data Types

- Converted **'date_added'** column from string to datetime format using pd.to_datetime().
- Ensured numeric columns like **duration** were properly typed.

Step 7: Clean String Columns

- Stripped extra whitespaces using .str.strip().
- Converted text to lowercase (for columns like **listed_in**) to standardize categorical values.

- Normalized **rating** values (uppercase, consistent spacing).

Step 8: Extract and Normalize Duration

- Split **duration** (e.g., “90 min” or “2 Seasons”) into:
    - **Numeric value** (duration_value)
    - **Unit** (duration_unit)

Step 10: Handle Categorical Values

- Standardized categories (like “Movies” and “TV Shows”) by converting them to lowercase or title case.

Step 11: Validate Data Consistency

- Ensured that no missing or inconsistent values remained using df.isnull().sum().

Step 12: Save the Cleaned Dataset

- Saved the cleaned data using df.to_csv('netflix_cleaned.csv', index=False).
- This final cleaned dataset can then be used for analysis, visualization, and modeling.

Steps Involved in Data Normalization

Step 1: Identify Numerical Columns

- Used methods like df.select_dtypes(include=['int64', 'float64']) to identify numeric columns.
- Normalization applies **only to numerical features**, not categorical ones.

Step 2: Handle Missing or Invalid Values (Pre-Normalization)

- Checked for missing values using df.isnull().sum() and handled them appropriately:
    - Filled missing numeric values with mean/median.
    - Dropped rows if they contained invalid data points.

Step 3: Choose Normalization Technique

Two main normalization techniques are usually applied:

1. **Min-Max Normalization (Rescaling)**
    - Formula:
      $$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$
    - Scales all values between **0 and 1**.
    - Suitable when data has a fixed range.
2. **Z-Score Normalization (Standardization)**
    - Formula:
      [

X_{std} = \frac{X - \mu}{\sigma}
]

   - Centers data around **mean = 0** and **standard deviation = 1**.
   - Preferred when data has varying ranges or outliers.

Step 4: Apply the Normalization

- Applied normalization to the selected numeric columns using:
   - **Manual formulas**, or
   - **Built-in methods** such as sklearn.preprocessing.MinMaxScaler() or StandardScaler().
- Example using pandas:
- df['normalized_column'] = (df['column'] - df['column'].min()) / (df['column'].max() - df['column'].min())

Step 5: Save the Normalized Dataset

- Saved the normalized dataset for later analysis or modeling using:
- df.to_csv('normalized_data.csv', index=False)
- This ensures the preprocessed data is ready for machine learning or visualization tasks.

EDA PROCESS

1. Convert Date Column

- Converted the date_added column from **object** to **datetime** type:
- df['date_added'] = pd.to_datetime(df['date_added'], format='%B %d, %Y', errors='coerce')

2. Check Dataset Information

- Displayed dataset info and shape:
- df.info()
- print(f"Rows: {df.shape[0]}, Columns: {df.shape[1]}")
- Helps understand the number of features and datatypes.

3. Handle Missing Values

- Filled all null or missing values with 0:
- df = df.fillna(0)
- df.isnull().sum()

4. Summary Statistics

- Generated **descriptive statistics** for both numerical and categorical data:

- df.describe()
- df.describe(include='object')

5. Movie vs TV Show Count

- Counted how many entries are **Movies** vs **TV Shows**:
- df['type'].value_counts()

6. Country Distribution

- Split multiple countries per entry, cleaned them, and counted the top 10:
- countries = df['country'].dropna().str.split(',').explode().str.strip()
- top_countries = countries.value_counts().head(10)

7. Genre Analysis

- Split and counted **top 10 genres**:
- genres = df['listed_in'].dropna().str.split(',').explode().str.strip()
- top_genres = genres.value_counts().head(10)

8. Titles Released per Year

- Counted the number of titles released each year:
- release_trend = df['release_year'].value_counts().sort_index()

9. Data Visualization

- Used **Matplotlib** and **Seaborn** for charts:
  - **Movies vs TV Shows (Bar chart)**
  - **Top 10 Countries with Most Content**
  - **Top 10 Genres on Netflix**
- type_count.plot(kind='bar')
- top_countries.plot(kind='bar')
- top_genres.plot(kind='barh')

✅ **Summary of EDA Workflow**

1. Load dataset
2. Convert datatypes
3. Explore dataset info

4. Handle missing values
5. Statistical summary
6. Analyze content type, country, and genre
7. Study release year trend
8. Visualize insights
9. Univariate and Bivariate analysis
10. Summary Statistics