Data and Artificial Intelligence
Cyber Shujaa Program

Week 2 Assignment
Data Wrangling and Analysis (Netflix Dataset)

Student Name: Rodney Roy Gitonga

Student ID: CS-DA03-26025

Introduction

This week's assignment focused on Data Wrangling, a critical step in the data science pipeline where raw data is cleaned and transformed into a format suitable for analysis. The task involved working with the Netflix Movies and TV Shows dataset hosted on Kaggle.

Using Python and the Pandas library within a Kaggle Notebook, I performed data discovery to understand the dataset's structure, applied cleaning techniques to handle missing values and duplicates, and enriched the data by extracting new features.

The objectives of the assignment were:

1. Load the Netflix dataset and perform exploratory data discovery.
2. Clean the dataset by handling duplicates and formatting inconsistencies.
3. Impute missing values for the 'Director' and 'Country' columns using logical relationships with other columns.
4. Transform the 'Duration' column into usable numeric data.
5. Validate the logical consistency of dates (e.g., ensuring content isn't added before it is released).
6. Export the polished dataset to a .csv file.

Tasks Completed

Below is the sequence of tasks completed to clean the dataset, supported by code snippets and evidence of execution.

Step 1: Data Discovery

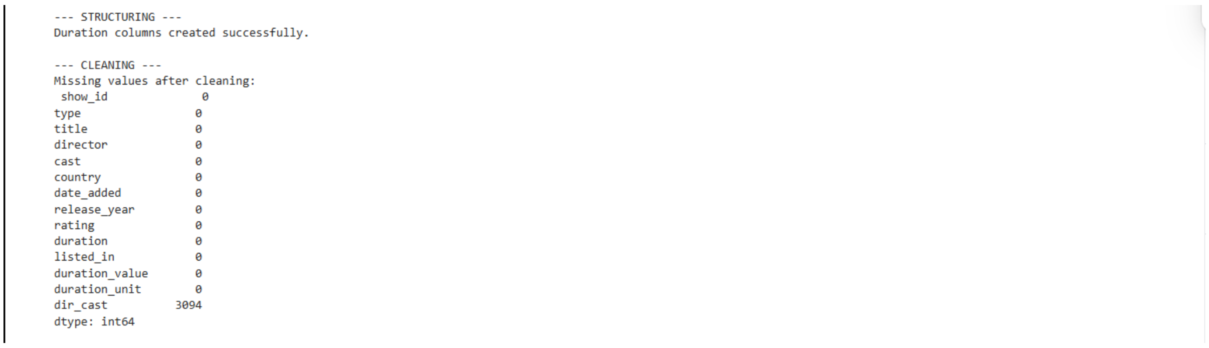
I loaded the dataset using pd.read_csv and inspected its structure using .shape and .isnull().sum(). This revealed missing values in the *Director*, *Cast*, and *Country* columns that needed addressing.

```
Description: This project focuses on cleaning, structuring and validating the Netflix Movies and TV shows Dataset


"""
import pandas as pd
import numpy as np
import datetime as dt

# 1. LOAD DATA
# Load the dataset from the specific Kaggle directory
df = pd.read_csv('/kaggle/input/netflix-shows/netflix_titles.csv')
```

Step 2: Structuring and Feature Extraction

To make the data analysable, I converted the date_added column to a datetime object. I also split the duration column (e.g., "90 min") into two separate columns: duration_value (numeric) and duration_unit (text).

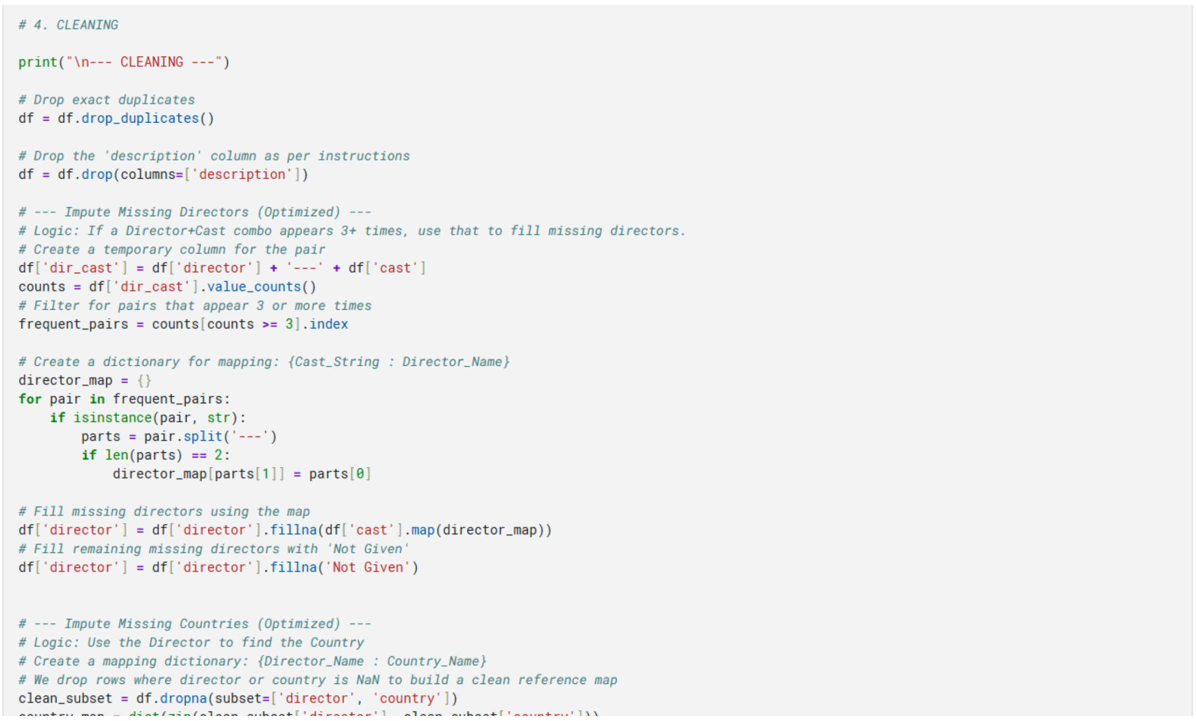
```
# 2. DISCOVERY

print("--- DISCOVERY ---")
print("Shape of the dataset:", df.shape)
print("\nMissing values per column:\n", df.isnull().sum())
print("\nDuplicate rows:", df.duplicated().sum())


# 3. STRUCTURING

print("\n--- STRUCTURING ---")

# Convert 'date_added' to datetime format
# errors='coerce' turns invalid dates into NaT (Not a Time) to prevent crashes
df['date_added'] = pd.to_datetime(df['date_added'].str.strip(), format='mixed', errors='coerce')

# Separate 'duration' into value and unit
# We use regex to find the numbers (\d+) and the text (\w+)
df[['duration_value', 'duration_unit']] = df['duration'].str.extract(r'(\d+)\s*(\w+)')

# Convert duration_value to numeric
df['duration_value'] = pd.to_numeric(df['duration_value'])

print("Duration columns created successfully.")
```



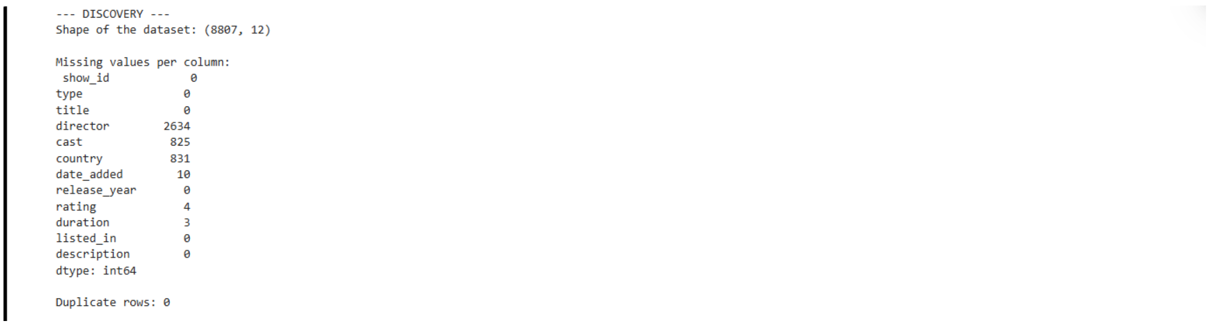
```
--- STRUCTURING ---
Duration columns created successfully.

--- CLEANING ---
Missing values after cleaning:
 show_id              0
type                 0
title                0
director             0
cast                 0
country              0
date_added           0
release_year         0
rating               0
duration             0
listed_in            0
duration_value       0
duration_unit        0
dir_cast          3094
dtype: int64
```

Step 3: Data Cleaning and Imputation

I removed exact duplicate rows to ensure data integrity. For missing 'Director' and 'Country' values, I implemented a logic to fill them based on relationships (e.g., if a specific Cast member frequently works with a Director, I used that to fill missing Director names). Remaining nulls were labelled as "Not Given".



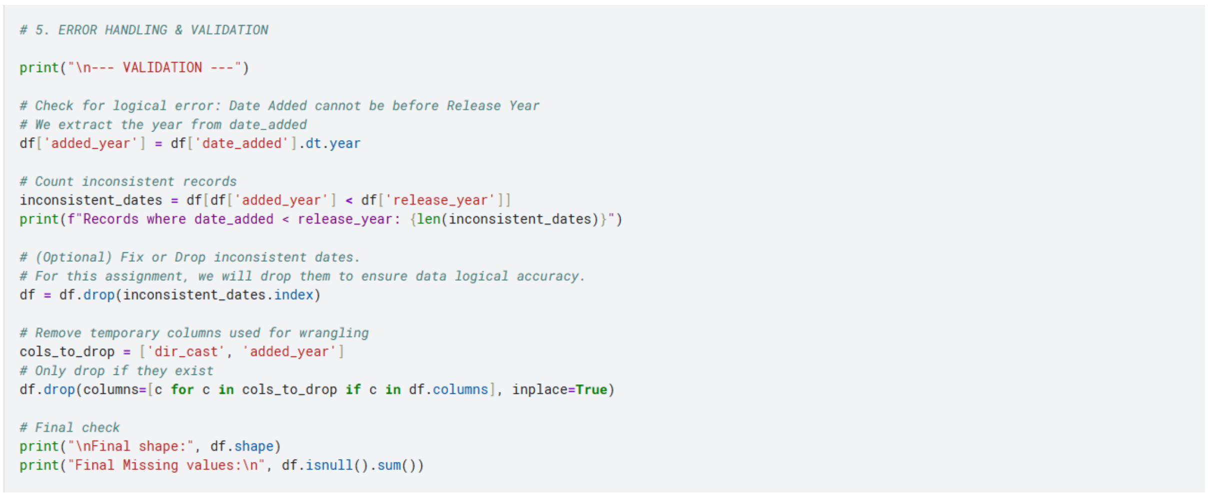
```python
# 4. CLEANING

print("\n--- CLEANING ---")

# Drop exact duplicates
df = df.drop_duplicates()

# Drop the 'description' column as per instructions
df = df.drop(columns=['description'])

# --- Impute Missing Directors (Optimized) ---
# Logic: If a Director+Cast combo appears 3+ times, use that to fill missing directors.
# Create a temporary column for the pair
df['dir_cast'] = df['director'] + '---' + df['cast']
counts = df['dir_cast'].value_counts()
# Filter for pairs that appear 3 or more times
frequent_pairs = counts[counts >= 3].index

# Create a dictionary for mapping: {Cast_String : Director_Name}
director_map = {}
for pair in frequent_pairs:
    if isinstance(pair, str):
        parts = pair.split('---')
        if len(parts) == 2:
            director_map[parts[1]] = parts[0]

# Fill missing directors using the map
df['director'] = df['director'].fillna(df['cast'].map(director_map))
# Fill remaining missing directors with 'Not Given'
df['director'] = df['director'].fillna('Not Given')


# --- Impute Missing Countries (Optimized) ---
# Logic: Use the Director to find the Country
# Create a mapping dictionary: {Director_Name : Country_Name}
# We drop rows where director or country is NaN to build a clean reference map
clean_subset = df.dropna(subset=['director', 'country'])
```



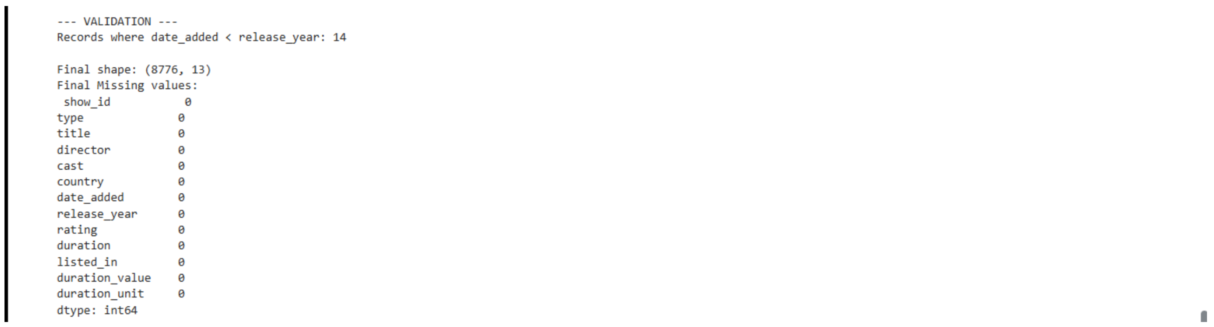
```
--- DISCOVERY ---
Shape of the dataset: (8807, 12)

Missing values per column:
 show_id           0
type              0
title             0
director       2634
cast            825
country         831
date_added       10
release_year      0
rating            4
duration          3
listed_in         0
description       0
dtype: int64

Duplicate rows: 0
```

Step 4: Validation

I performed a logical check to ensure that the date_added was not earlier than the release_year. Inconsistent records were identified and removed to maintain accuracy.



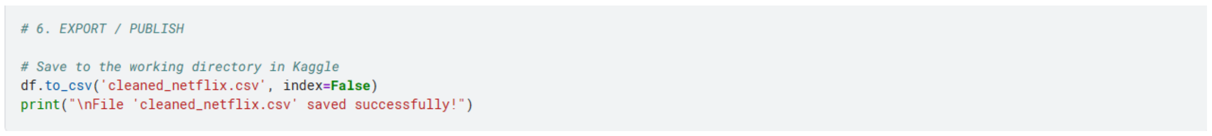
```python
# 5. ERROR HANDLING & VALIDATION

print("\n--- VALIDATION ---")

# Check for logical error: Date Added cannot be before Release Year
# We extract the year from date_added
df['added_year'] = df['date_added'].dt.year

# Count inconsistent records
inconsistent_dates = df[df['added_year'] < df['release_year']]
print(f"Records where date_added < release_year: {len(inconsistent_dates)}")

# (Optional) Fix or Drop inconsistent dates.
# For this assignment, we will drop them to ensure data logical accuracy.
df = df.drop(inconsistent_dates.index)

# Remove temporary columns used for wrangling
cols_to_drop = ['dir_cast', 'added_year']
# Only drop if they exist
df.drop(columns=[c for c in cols_to_drop if c in df.columns], inplace=True)

# Final check
print("\nFinal shape:", df.shape)
print("Final Missing values:\n", df.isnull().sum())
```



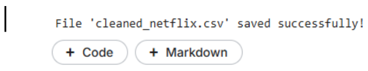
```
--- VALIDATION ---
Records where date_added < release_year: 14

Final shape: (8776, 13)
Final Missing values:
 show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
duration_value  0
duration_unit   0
dtype: int64
```

Step 5: Publishing

The final cleaned dataset was exported to the Kaggle working directory.



```python
# 6. EXPORT / PUBLISH

# Save to the working directory in Kaggle
df.to_csv('cleaned_netflix.csv', index=False)
print("\nFile 'cleaned_netflix.csv' saved successfully!")
```



```
File 'cleaned_netflix.csv' saved successfully!
```

Link to Code:

https://www.kaggle.com/code/fytroy/rodney-roy-gitonga-data-wrangling

Conclusion

Through this assignment, I learned that real-world data is rarely ready for immediate analysis. I gained hands-on experience with advanced Pandas functions, including regex for string extraction and mapping dictionaries for data imputation. I also understood the importance of validating data logic (such as date consistency) before finalizing a dataset. This project has strengthened my ability to transform raw, messy data into a structured format, a crucial skill for my future work in Data and AI.