DATA CLEANING WITH PYTHON: A WALKTHROUGH 🧹

In this notebook, I have shown in details how to clean data using the Pandas library in Python. The data we're cleaning contains information about the trends of movies in the IMDb dataset. We'll clean the data in Python and then visualize on Tableau.

In [1]:
```
#Importing the neccesary libraries
import pandas as pd
```

In [2]:
```
#Loading the dataset into a pandas dataframe
imdb_df = pd.read_excel("Desktop/IMDb Data.xlsx")
```

In [3]:
```
#Data Overview
imdb_df.head()
```

Out[3]:

	Movie Title	Rating	User Rating	Genres	Overview	Plot Keyword	Director	Top 5 Casts	Writer
0	Top Gun: Maverick	8.6	187000	Action| Drama	After more than thirty years of service as one...	fighter jet, sequel, u.s. navy, fighter aircra...	Joseph Kosinski	Jack Epps Jr., Peter Craig, Tom Cruise, Jennif...	Jim Cash
1	Jurassic World Dominion	6	56000	Action| Adventure| Sci-Fi	Four years after the destruction of Isla Nubla...	dinosaur, jurassic park, tyrannosaurus rex, ve...	Colin Trevorrow	Colin Trevorrow, Derek Connolly, Chris Pratt, ...	Emily Carmichael
2	Top Gun	6.9	380000	Action| Drama	As students at the United States Navys elite f...	pilot, male camaraderie, u.s. navy, grumman f ...	Tony Scott	Jack Epps Jr., Ehud Yonay, Tom Cruise, Tim Rob...	Jim Cash
3	Lightyear	5.2	32000	Animation| Action| Adventure	While spending years attempting to return home...	galaxy, spaceship, robot, rocket, space advent...	Angus MacLane	Jason Headley, Matthew Aldrich, Chris Evans, K...	Angus MacLane
4	Spiderhead	5.4	23000	Action| Crime| Drama	In the near future, convicts are offered the c...	discover, medical, test, reality, fictional dr...	Joseph Kosinski	Rhett Reese, Paul Wernick, Chris Hemsworth, Mi...	George Saunders

In [4]:
```
imdb_df.shape
```

Out[4]:
```
(24402, 11)
```

The dataset has 24,402 rows and 11 columns.

In [5]:
```
imdb_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24402 entries, 0 to 24401
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Movie Title   24402 non-null  object
 1   Rating        24402 non-null  object
 2   User Rating   24402 non-null  int64 
 3   Genres        24402 non-null  object
 4   Overview      24157 non-null  object
 5   Plot Keyword  22706 non-null  object
 6   Director      24402 non-null  object
 7   Top 5 Casts   24401 non-null  object
 8   Writer        24402 non-null  object
 9   Year          23433 non-null  float64
 10  Path          24402 non-null  object
dtypes: float64(1), int64(1), object(9)
memory usage: 2.0+ MB
```

Let's look at the different columns in the dataset, with focus on columns like Rating, User Rating, Genres and Year which will be useful for analysis.

- RATING COLUMN

  The rating column contains the rating given to a movie by different people who have seen the movie. The column should be a "float" data type and not "object". Let's check why it is so.

In [20]:
```
#Data Cleaning
#Rating Column : checking the unique values
imdb_df["Rating"].unique()
```

Out[20]:
```
array([8.6, 6. , 6.9, 5.2, 5.4, 8.3, 4.4, 3.6, 0. , 7.2, 8.2, 6.2, 7.1,
       7.3, 7.9, 6.8, 6.1, 8.8, 5.1, 5.6, 6.6, 6.4, 5.3, 6.5, 8. , 6.3,
       5.9, 8.4, 5.7, 8.1, 6.7, 9. , 7.8, 7.7, 7.6, 8.7, 5.8, 7.4, 7. ,
       7.5, 8.5, 5. , 4.7, 5.5, 4.3, 3.7, 2.6, 4.6, 4.9, 4.5, 4. , 4.8,
       3.4, 3.9, 3.3, 1.6, 4.2, 1.7, 3.8, 3.5, 2.8, 2.9, 1.8, 2.7, 1.9,
       2.5, 3. , 3.2, 4.1, 3.1, 2.1, 2.2, 2.4, 9.3, 1.3, 1.4, 2.3, 1.1,
       9.6, 2. , 1.2, 9.8, 8.9, 1.5, 1. , 9.1, 9.2, 9.5])
```

Change the values "no-rating" to 0.

In [7]:
```
imdb_df["Rating"] = imdb_df["Rating"].replace("no-rating", 0)
imdb_df["Rating"].unique()
```

Out[7]:
```
array([8.6, 6. , 6.9, 5.2, 5.4, 8.3, 4.4, 3.6, 0. , 7.2, 8.2, 6.2, 7.1,
       7.3, 7.9, 6.8, 6.1, 8.8, 5.1, 5.6, 6.6, 6.4, 5.3, 6.5, 8. , 6.3,
       5.9, 8.4, 5.7, 8.1, 6.7, 9. , 7.8, 7.7, 7.6, 8.7, 5.8, 7.4, 7. ,
       7.5, 8.5, 5. , 4.7, 5.5, 4.3, 3.7, 2.6, 3.9, 4.6, 4.9, 4.5, 4. ,
       2.9, 4.8, 3.4, 3.3, 1.6, 4.2, 1.7, 3.8, 3.5, 2.8, 1.8, 2.7, 1.9,
       2.5, 3. , 3.2, 4.1, 3.1, 2.1, 2.2, 2.4, 9.3, 1.3, 1.4, 2.3, 1.1,
       8.9, 9.6, 2. , 9.2, 9.9, 1.2, 9.8, 9.4, 9.7, 9.1, 1.5, 1. , 9.5])
```

Perfect! Now, let's change the data type of the column to float.

In [8]:
```
imdb_df["Rating"] = imdb_df["Rating"].astype(float)
imdb_df["Rating"].dtypes
```

Out[8]:
```
dtype('float64')
```

- USER RATING

  This columns contains the total number of rating a movie has gotten.

In [9]:
```
#User rating: Checking the unique values in the column
imdb_df["User Rating"].unique()
```

Out[9]:
```
array([187000,  56000, 380000, ..., 441000, 485000, 347000], dtype=int64)
```

In [11]:
```
#Checking the sum of null values
imdb_df.shape
```

Out[11]:
```
(24402, 11)
```

In [93]:
```
#Checking the frequency of the values in the column
imdb_df.groupby("User Rating").size()
```

Out[93]:
```
User Rating
0          1740
5             3
6             8
7            20
8            14
           ...
1800000       3
1900000       1
2000000       3
2300000       1
2600000       2
Length: 1684, dtype: int64
```

- YEAR COLUMN

  This column contains the year that the movies were released. Let's check it out.

In [94]:
```
#Year: Checking the unique values
imdb_df["Year"].unique()
```

Out[94]:
```
array([2022., 1986., 1993., 2021., 2015., 2018., 1997., 1984., 2001.,
       1977., 2019., 2003., 1996., 1999., 2008., 2017., 2010., 2005.,
       2023., 1989., 2011., 1995., 2009., 2020., 2014., 2000., 2002.,
       1983., 2012., 2013., 1988., 2006., 2016., 1982., 1987., 2007.,
       1991., 2004., 1994., 1998., 1990., 1992., 1978., 1962., 1976.,
       1979., 1980.,   nan, 1985., 1964., 1981., 1968., 1971., 1974.,
       1959., 1973., 1963., 1965., 1969., 1967., 2024., 1972., 1961.,
       1915., 1966., 1953., 1975., 1970., 1937., 1952., 1938., 1960.,
       1932., 2025., 1954., 1949., 1926., 1948., 1958., 1950., 1957.,
       1951., 2027., 1940., 1924., 2028., 1941., 1934., 1927., 1942.,
       1955., 1935., 2026., 1956., 1923., 1943., 1939., 1945., 1928.,
```

```
       1931., 1944., 1916., 1936., 1918., 1933., 1946., 1947., 1914.,
       1930., 1906., 1929., 1925., 1921., 1919., 1920., 1913., 1922.])
```

You would notice that there are null values there, which means the dataset didn't contain the year those movies were released. Let's check the sum of null values there and do away with them.

In [95]:
```
#Sum of null values
imdb_df["Year"].isnull().sum()
```

Out[95]:
```
969
```

In [13]:
```
#Removing all the null values from the dataframe
imdb_df = imdb_df.dropna()
imdb_df.shape
```

Out[13]:
```
(22034, 11)
```

In [17]:
```
imdb_df["Year"].isnull().sum()
```

Out[17]:
```
0
```

The null values have been removed. We need to change the data type to integer since we won't be needing year values with decimal points.

In [18]:
```
imdb_df["Year"] = imdb_df["Year"].astype(int)
imdb_df["Year"].unique()
```

Out[18]:
```
array([2022, 1986, 1993, 2021, 2015, 2018, 1997, 1984, 2001, 1977, 2019,
       2003, 1996, 1999, 2008, 2017, 2010, 2005, 2023, 1989, 2011, 1995,
       2009, 2020, 2014, 2000, 2002, 1983, 2012, 2013, 1988, 2006, 2016,
       1982, 1987, 2007, 1991, 2004, 1994, 1998, 1990, 1992, 1978, 1962,
       1976, 1979, 1980, 1985, 1964, 1981, 1968, 1971, 1974, 1959, 1973,
       1963, 1965, 1969, 1967, 2024, 1972, 1961, 1915, 1966, 1953, 1975,
       1970, 1937, 1952, 1938, 1960, 1932, 2025, 1954, 1949, 1926, 1948,
       1958, 1950, 1957, 1951, 2027, 1940, 1924, 2028, 1941, 1934, 1927,
       1942, 1955, 1935, 2026, 1956, 1923, 1943, 1939, 1945, 1928, 1931,
       1944, 1916, 1936, 1918, 1933, 1946, 1947, 1914, 1930, 1906, 1929,
       1925, 1921, 1919, 1920, 1913, 1922])
```

- GENRES

  This columns contains the genre of the different movies in the dataset.

In [19]:
```
#Genre: checking the frequency of the values in descending order.
imdb_df.groupby("Genres").size().sort_values(ascending = False)
```

Out[19]:
```
Genres
Action| Crime| Drama           803
Drama                          796
Comedy| Drama| Romance         597
Crime| Drama| Thriller         576
Comedy| Drama                  504
                              ...
Adventure| Crime| Film-Noir      1
Drama| Music| War                1
```

```
Action| Family| Mystery          1
Drama| Mystery| Crime            1
Western| Crime| Thriller         1
Length: 719, dtype: int64
```

Looking at the top five genre of movies, it seems to me that IMDb movies are mostly Dramas😈 .
We'll uncover more in the visualization.

In [27]:
```
#Final Data Overview
#Number of rows and columns
imdb_df.shape
```

Out[27]:
```
(22034, 11)
```

In [26]:
```
#Sum of null values
imdb_df.isnull().sum()
```

Out[26]:
```
Movie Title     0
Rating          0
User Rating     0
Genres          0
Overview        0
Plot Keyword    0
Director        0
Top 5 Casts     0
Writer          0
Year            0
Path            0
dtype: int64
```

In [32]:
```
#Unique values
imdb_df.nunique()
```

Out[32]:
```
Movie Title     21623
Rating             88
User Rating      1683
Genres            719
Overview        22016
Plot Keyword    21099
Director        10296
Top 5 Casts     21986
Writer          14031
Year              116
Path            21624
dtype: int64
```

In [33]:
```
#Checking the information about the colums
imdb_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22034 entries, 0 to 24401
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Movie Title   22034 non-null  object
 1   Rating        22034 non-null  float64
 2   User Rating   22034 non-null  int64
 3   Genres        22034 non-null  object
 4   Overview      22034 non-null  object
```

```
 5   Plot Keyword  22034 non-null  object
 6   Director      22034 non-null  object
 7   Top 5 Casts   22034 non-null  object
 8   Writer        22034 non-null  object
 9   Year          22034 non-null  int32
 10  Path          22034 non-null  object
dtypes: float64(1), int32(1), int64(1), object(8)
memory usage: 1.9+ MB
```

In [29]:

```
#First 5 rows
imdb_df.head()
```

Out[29]:

	Movie Title	Rating	User Rating	Genres	Overview	Plot Keyword	Director	Top 5 Casts	Writer
0	Top Gun: Maverick	8.6	187000	Action| Drama	After more than thirty years of service as one...	fighter jet, sequel, u.s. navy, fighter aircra...	Joseph Kosinski	Jack Epps Jr., Peter Craig, Tom Cruise, Jennif...	Jim Cash
1	Jurassic World Dominion	6.0	56000	Action| Adventure| Sci-Fi	Four years after the destruction of Isla Nubla...	dinosaur, jurassic park, tyrannosaurus rex, ve...	Colin Trevorrow	Colin Trevorrow, Derek Connolly, Chris Pratt, ...	Emily Carmichael
2	Top Gun	6.9	380000	Action| Drama	As students at the United States Navys elite f...	pilot, male camaraderie, u.s. navy, grumman f ...	Tony Scott	Jack Epps Jr., Ehud Yonay, Tom Cruise, Tim Rob...	Jim Cash
3	Lightyear	5.2	32000	Animation| Action| Adventure	While spending years attempting to return home...	galaxy, spaceship, robot, rocket, space advent...	Angus MacLane	Jason Headley, Matthew Aldrich, Chris Evans, K...	Angus MacLane
4	Spiderhead	5.4	23000	Action| Crime| Drama	In the near future, convicts are offered the c...	discover, medical, test, reality, fictional dr...	Joseph Kosinski	Rhett Reese, Paul Wernick, Chris Hemsworth, Mi...	George Saunders

In [34]:

```
#Transferring the cleaned data to an excel file
writer = pd.ExcelWriter("Desktop/Cleaned IMDb Data.xlsx")
imdb_df.to_excel(writer)
writer.save()
print('DataFrame has been transferred successfully to Excel Workbook')
```

```
DataFrame has been transferred successfully to Excel Workbook
```

The visualization would be done on Tableau. Here is the link to view it.

Tableau Dashboard

In [ ]: