Data and Artificial Intelligence
Cyber Shujaa Program

Week 2 Assignment
Web Scraping and Data Handling in Python

Student Name: Deborah Kwamboka Omae

Student ID: CS-DA02-25075

Table of Contents

Table of figures

Introduction

This week's assignment was on data wrangling. Data wrangling is the process of cleaning, structuring, and transforming raw data into a usable format for analysis. The data source was from Kaggle. I developed hands-on experience on automating web data gathering using Kaggle Data Set and published my work on Kaggle. I practiced the data Wrangling concepts to clean up the Netflix dataset to a well-structured and usable format.

Objectives

The objectives of the assignment were to:

1. Load the Netflix dataset from a CSV file and explore its structure using pandas.
2. Perform data discovery to assess data types, missing values, and quality issues.
3. Clean the dataset by handling duplicates, missing values, and formatting inconsistencies.
4. Transform and enrich the dataset using techniques like filtering, sorting, grouping, and feature extraction.
5. Validate the final dataset by checking consistency, completeness, and logical accuracy.
6. Export the final cleaned dataset to a .csv file ready for analysis or visualization.

Tasks Completed

Step 1

In this step I loaded and inspected the Netflix dataset that I was going to work with. I created a notebook file and attached the dataset.

Code:

```
import pandas as pd
df = pd.read_csv('/kaggle/input/netflix/netflix_titles.csv')
```

Code explanation:

The code imports the python library, pandas, for structuring the data. The csv file is read into a data frame and the result is stored in the variable df. The read_csv function of pandas takes in the file path of the dataset.

Step 2

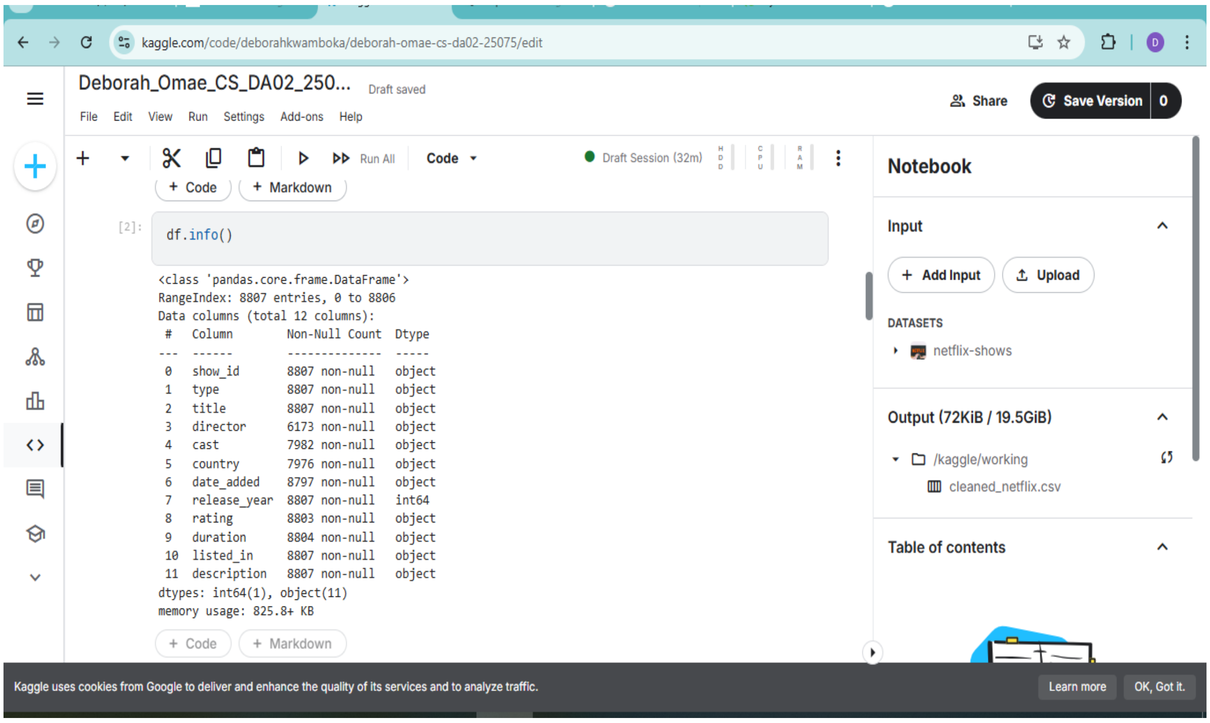
In this step I explored my data to have an overview of it and to better understand it. I checked for its shape, list of column names, data types and missing values.

Code:

```
df.info()
print("Shape of the dataset (R x C):", df.shape)
print("Columns in the dataset:\n", df.columns.tolist())
print("Data types:\n", df.dtypes)
print("Missing values per column:\n", df.isnull().sum())
print("Number of duplicate rows:", df.duplicated().sum())
```

Code explanation:

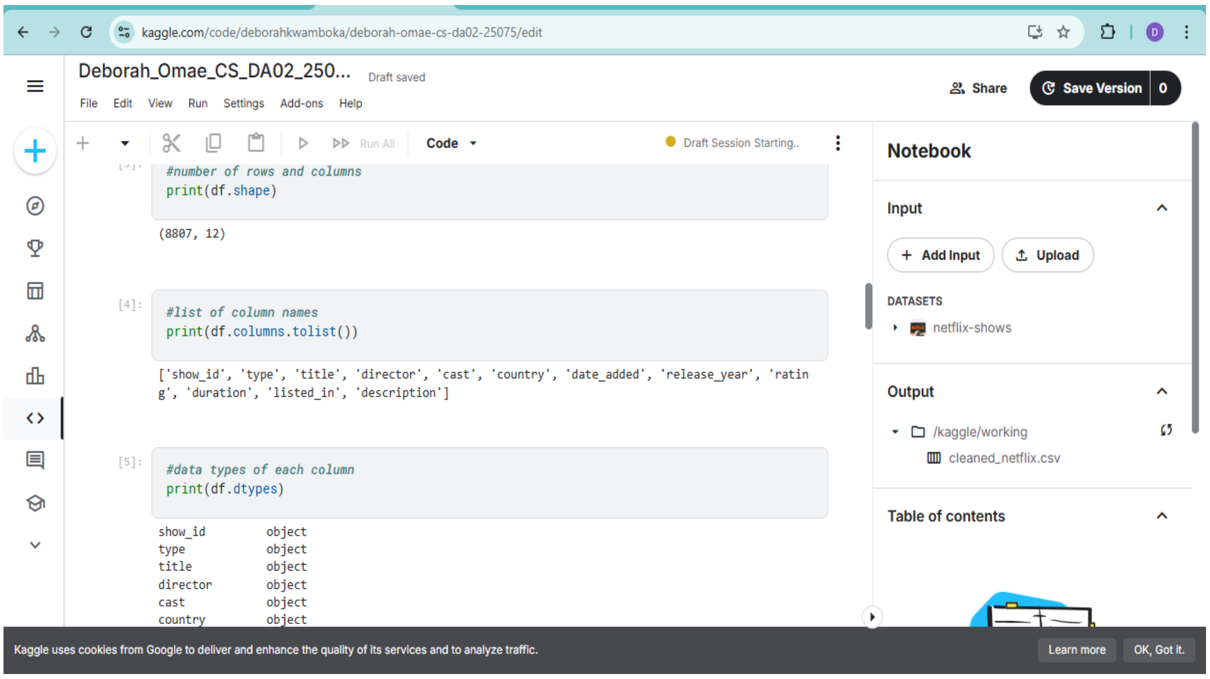
df.info() gives an overview of the data. df.shape() gives the shape of the data set which is the number of rows and column it has. df.columns.tolist() lists all column names in the dataset. df.dtypes gives the data types of each column. df.isnull().sum() groups and counts missing null values in each column. df.duplicated().sum() counts all duplicate rows.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

Figure 1: screenshot showing the result of df.info ()



```
#number of rows and columns
print(df.shape)
```

```
(8807, 12)
```

```
#list of column names
print(df.columns.tolist())
```

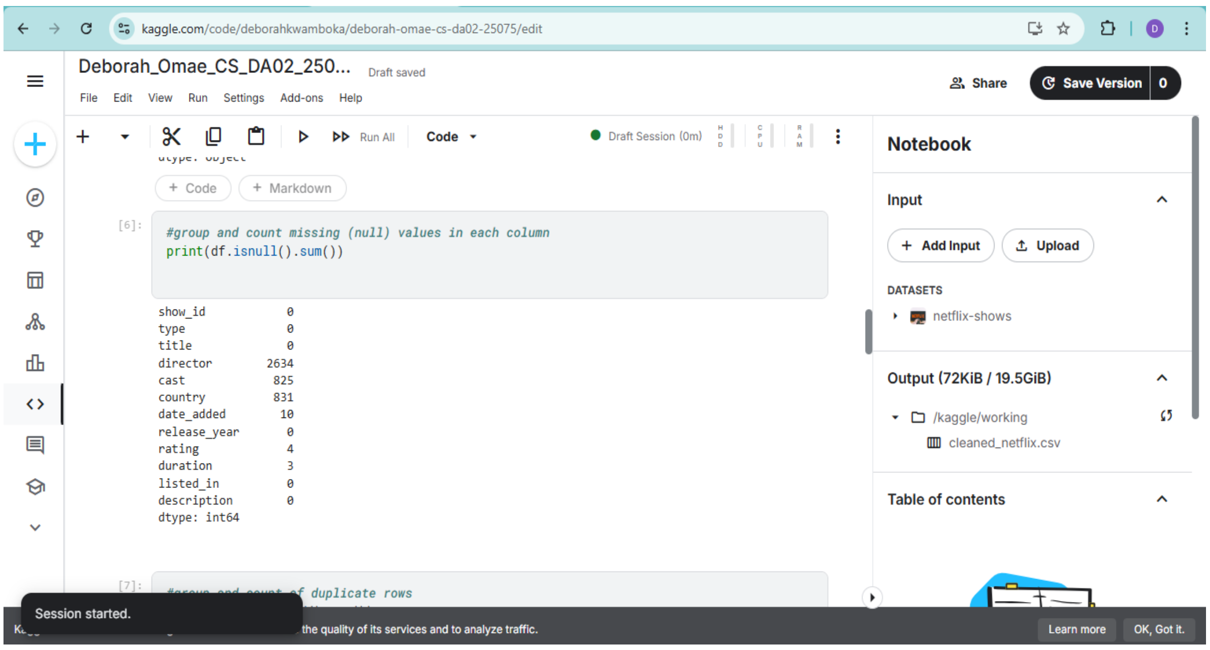
```
['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added', 'release_year', 'ratin
g', 'duration', 'listed_in', 'description']
```

```
#data types of each column
print(df.dtypes)
```

```
show_id         object
type            object
title           object
director        object
cast            object
country         object
```

kaggle.com/code/deborahkwamboka/deborah-omae-cs-da02-25075/edit
Deborah_Omae_CS_DA02_250...
Draft saved
Share
Save Version
Draft Session Starting..
Notebook
Input
Add Input
Upload
DATASETS
netflix-shows
Output
/kaggle/working
cleaned_netflix.csv
Table of contents
Kaggle uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic.
Learn more
OK, Got it.

Figure 2: screenshot showing number of rows and columns and the list of column names



```
[6]:
#group and count missing (null) values in each column
print(df.isnull().sum())
```

```
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
```

```
[7]:
#group and count of duplicate rows
```

Figure 3: screenshot showing missing null values of each column

Step3

In this step, I did structure and formatting of some columns. I structured the date added to datetime, the duration column into value and unit.

Code:

df['date_added'] = pd.to_datetime(df['date_added'],format='mixed')

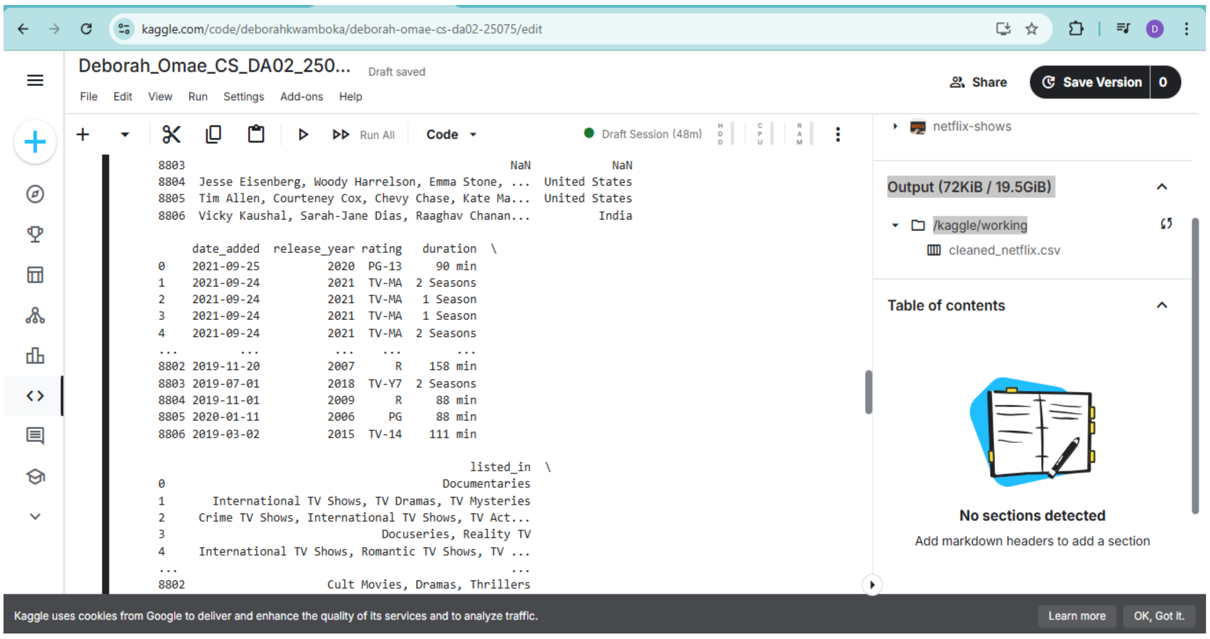
df[['duration_value', 'duration_unit']] = df['duration'].str.extract(r'(\d+)\s(\w+)')*

df['duration_value'] = pd.to_numeric(df['duration_value'])

print(df[['duration_value', 'duration_unit']])

code explanation:

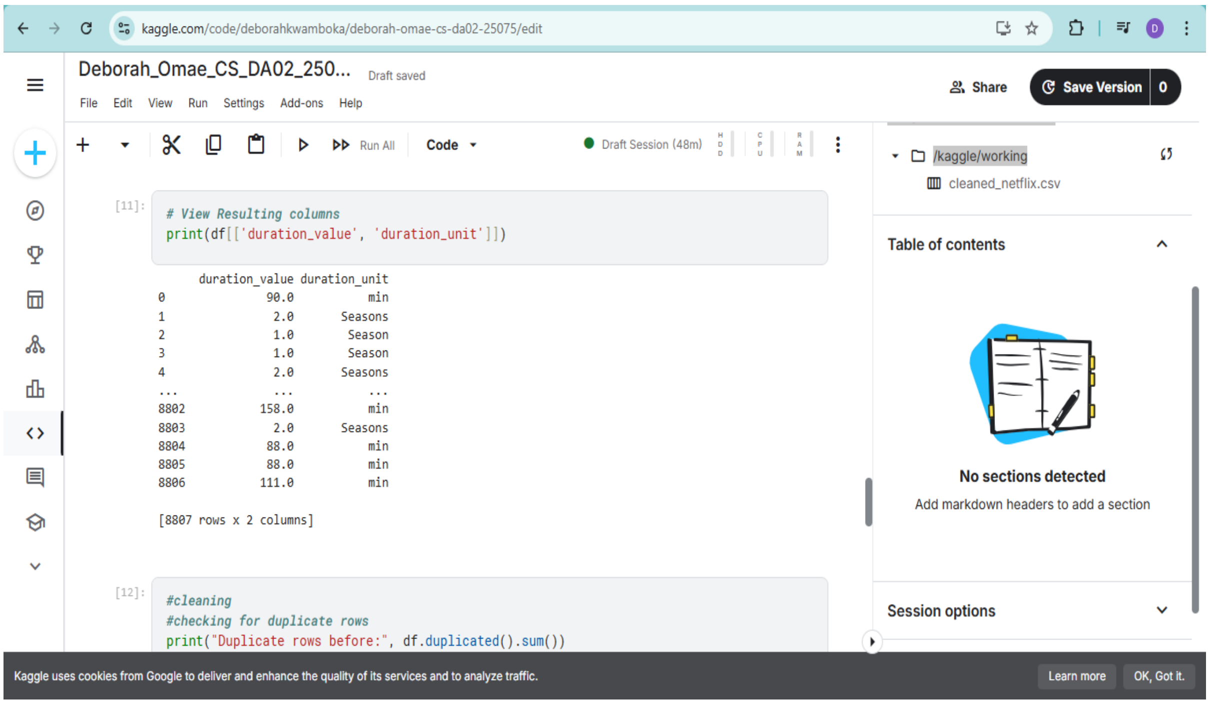
The datetime function of pandas converts the date added to date time format. To separate the duration column to numerical value and unit we use this regex expression; str.extract(r'(\d+)\s*(\w+)'). (\d+) captures the number, (\w+) captures the word. The space is matched and discarded since its not in parentheses. The duration value is converted to numeric value through the pandas to_numeric() function.


```
8803                                                    NaN            NaN
8804  Jesse Eisenberg, Woody Harrelson, Emma Stone, ...  United States
8805  Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...  United States
8806  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...          India

     date_added  release_year rating   duration  \
0    2021-09-25          2020  PG-13     90 min
1    2021-09-24          2021  TV-MA  2 Seasons
2    2021-09-24          2021  TV-MA   1 Season
3    2021-09-24          2021  TV-MA   1 Season
4    2021-09-24          2021  TV-MA  2 Seasons
...         ...           ...    ...        ...
8802 2019-11-20          2007      R    158 min
8803 2019-07-01          2018  TV-Y7  2 Seasons
8804 2019-11-01          2009      R     88 min
8805 2020-01-11          2006     PG     88 min
8806 2019-03-02          2015  TV-14    111 min

                                              listed_in  \
0                                         Documentaries
1       International TV Shows, TV Dramas, TV Mysteries
2     Crime TV Shows, International TV Shows, TV Act...
3                                Docuseries, Reality TV
4     International TV Shows, Romantic TV Shows, TV ...
...                                                 ...
8802                         Cult Movies, Dramas, Thrillers
```

Figure 4: screenshot showing the conversion of date added to date time format


```
# View Resulting columns
print(df[['duration_value', 'duration_unit']])
```

```
      duration_value duration_unit
0               90.0           min
1                2.0       Seasons
2                1.0        Season
3                1.0        Season
4                2.0       Seasons
...              ...           ...
8802           158.0           min
8803             2.0       Seasons
8804            88.0           min
8805            88.0           min
8806           111.0           min

[8807 rows x 2 columns]
```

```
#cleaning
#checking for duplicate rows
print("Duplicate rows before:", df.duplicated().sum())
```

Figure 5: screenshot showing the duration column separated into duration value and unit. This is useful in case one wants to perform calculations on the duration value column.

Step 4

In this step I performed cleaning of the data. I checked for duplicate rows and dropped them if any.

Code:

```
print("Duplicate rows before:", df.duplicated().sum())

df = df.drop_duplicates()
```

code explanation

The duplicated() function checks for duplicated rows and used with the sum() method it returns the number of rows that are duplicates. df.drop_duplicates() function drops the duplicated rows if any.

Note: For this data set there were no duplicated rows.

Step 5

In this step I dropped the description column since its not useful.

code

```
df = df.drop(columns=['description'])
```

step 6

In this step I Impute missing values in the director column by using repeated director–cast relationships found elsewhere in the dataset. The assumption is: if a particular director and cast string appear together frequently, we can trust that relationship and use it to infer missing directors when we see the same cast again.

Code:

```
# List of Director-Cast pairs and the number of times they appear
df['dir_cast'] = df['director'] + '---' + df['cast']
counts = df['dir_cast'].value_counts() #counts unique values
filtered_counts = counts[counts >= 3] #checks if repeated 3 or more times
filtered_values = filtered_counts.index #gets the values i.e. names
lst_dir_cast = list(filtered_values) #convert to list
dict_direcast = dict()
for i in lst_dir_cast :
    director,cast = i.split('---’)
```

dict_direcast[director]=cast

for i in range(len(dict_direcast)):

df.loc[(df['director'].isna()) & (df['cast'] == list(dict_direcast.items())[i][1]),'director'] = list(dict_direcast.items())[i][0]

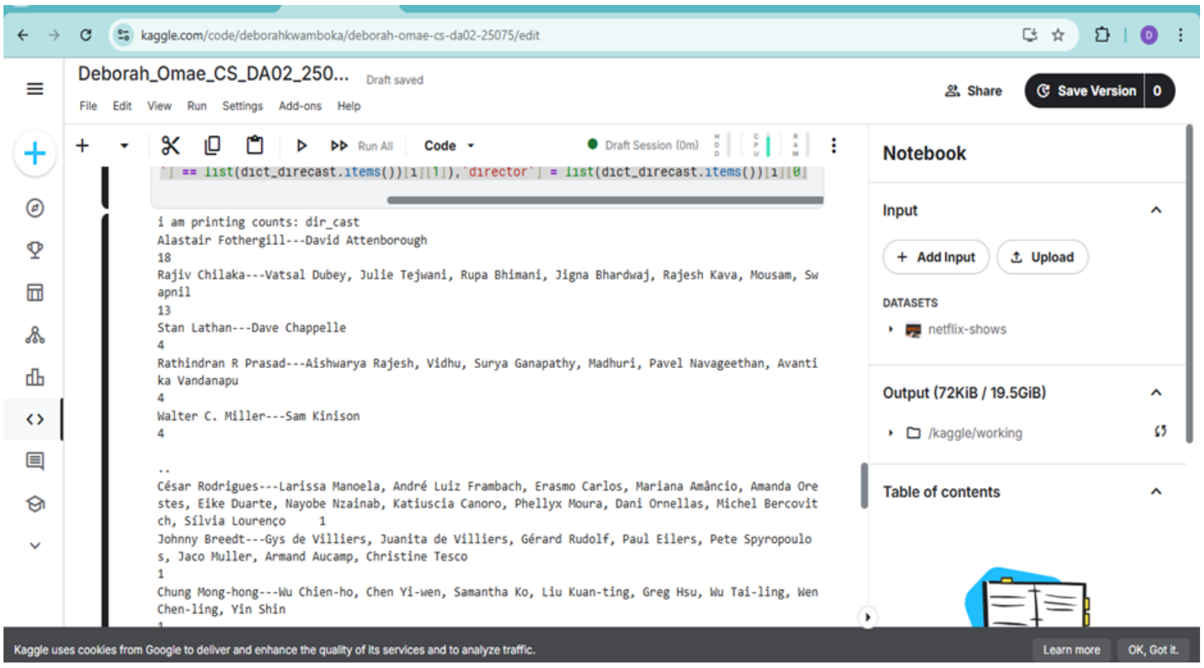
Code explanation:

- df['dir_cast'] = df['director'] + '---' + df['cast']

  This code creates a combined identifier by merging director and cast into a single string to represent each pair uniquely.

- counts = df['dir_cast'].value_counts()

  This code counts how many times each unique director–cast pair appears in the dataset.


kaggle.com/code/deborahkwamboka/deborah-omae-cs-da02-25075/edit
Deborah_Omae_CS_DA02_250...
Draft saved
File Edit View Run Settings Add-ons Help
Share
Save Version 0
Run All
Code
Draft Session (0m)
Notebook
Input
Add Input
Upload
DATASETS
netflix-shows
Output (72KiB / 19.5GiB)
/kaggle/working
Table of contents
Kaggle uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic.
Learn more
OK, Got it.

```
'] == list(dict_direcast.items())[i][1]),'director'] = list(dict_direcast.items())[i][0]
```

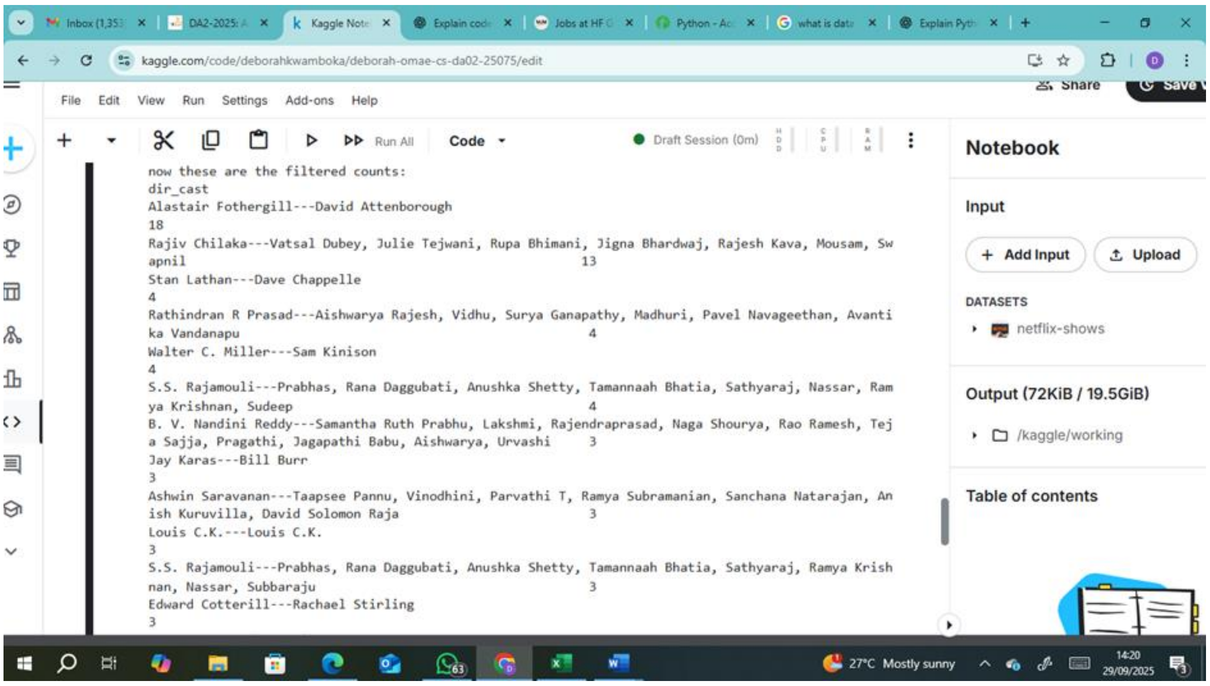
```
i am printing counts: dir_cast
Alastair Fothergill---David Attenborough
18
Rajiv Chilaka---Vatsal Dubey, Julie Tejwani, Rupa Bhimani, Jigna Bhardwaj, Rajesh Kava, Mousam, Sw
apnil
13
Stan Lathan---Dave Chappelle
4
Rathindran R Prasad---Aishwarya Rajesh, Vidhu, Surya Ganapathy, Madhuri, Pavel Navageethan, Avanti
ka Vandanapu
4
Walter C. Miller---Sam Kinison
4

..
César Rodrigues---Larissa Manoela, André Luiz Frambach, Erasmo Carlos, Mariana Amâncio, Amanda Ore
stes, Eike Duarte, Nayobe Nzainab, Katiuscia Canoro, Phellyx Moura, Dani Ornellas, Michel Bercovit
ch, Sílvia Lourenço     1
Johnny Breedt---Gys de Villiers, Juanita de Villiers, Gérard Rudolf, Paul Eilers, Pete Spyropoulo
s, Jaco Muller, Armand Aucamp, Christine Tesco
1
Chung Mong-hong---Wu Chien-ho, Chen Yi-wen, Samantha Ko, Liu Kuan-ting, Greg Hsu, Wu Tai-ling, Wen
Chen-ling, Yin Shin
```

Figure 6: This screenshot shows the dir_cast column and counts of the unique pairs.

- filtered_counts = counts[counts >= 3]

  This code filters the results to keep only pairs that appear three or more times, assuming frequent collaborations are more reliable.



```
now these are the filtered counts:
dir_cast
Alastair Fothergill---David Attenborough
18
Rajiv Chilaka---Vatsal Dubey, Julie Tejwani, Rupa Bhimani, Jigna Bhardwaj, Rajesh Kava, Mousam, Sw
apnil                                                                         13
Stan Lathan---Dave Chappelle
4
Rathindran R Prasad---Aishwarya Rajesh, Vidhu, Surya Ganapathy, Madhuri, Pavel Navageethan, Avanti
ka Vandanapu                                                                   4
Walter C. Miller---Sam Kinison
4
S.S. Rajamouli---Prabhas, Rana Daggubati, Anushka Shetty, Tamannaah Bhatia, Sathyaraj, Nassar, Ram
ya Krishnan, Sudeep                                                            4
B. V. Nandini Reddy---Samantha Ruth Prabhu, Lakshmi, Rajendraprasad, Naga Shourya, Rao Ramesh, Tej
a Sajja, Pragathi, Jagapathi Babu, Aishwarya, Urvashi                          3
Jay Karas---Bill Burr
3
Ashwin Saravanan---Taapsee Pannu, Vinodhini, Parvathi T, Ramya Subramanian, Sanchana Natarajan, An
ish Kuruvilla, David Solomon Raja                                              3
Louis C.K.---Louis C.K.
3
S.S. Rajamouli---Prabhas, Rana Daggubati, Anushka Shetty, Tamannaah Bhatia, Sathyaraj, Ramya Krish
nan, Nassar, Subbaraju                                                         3
Edward Cotterill---Rachael Stirling
3
```

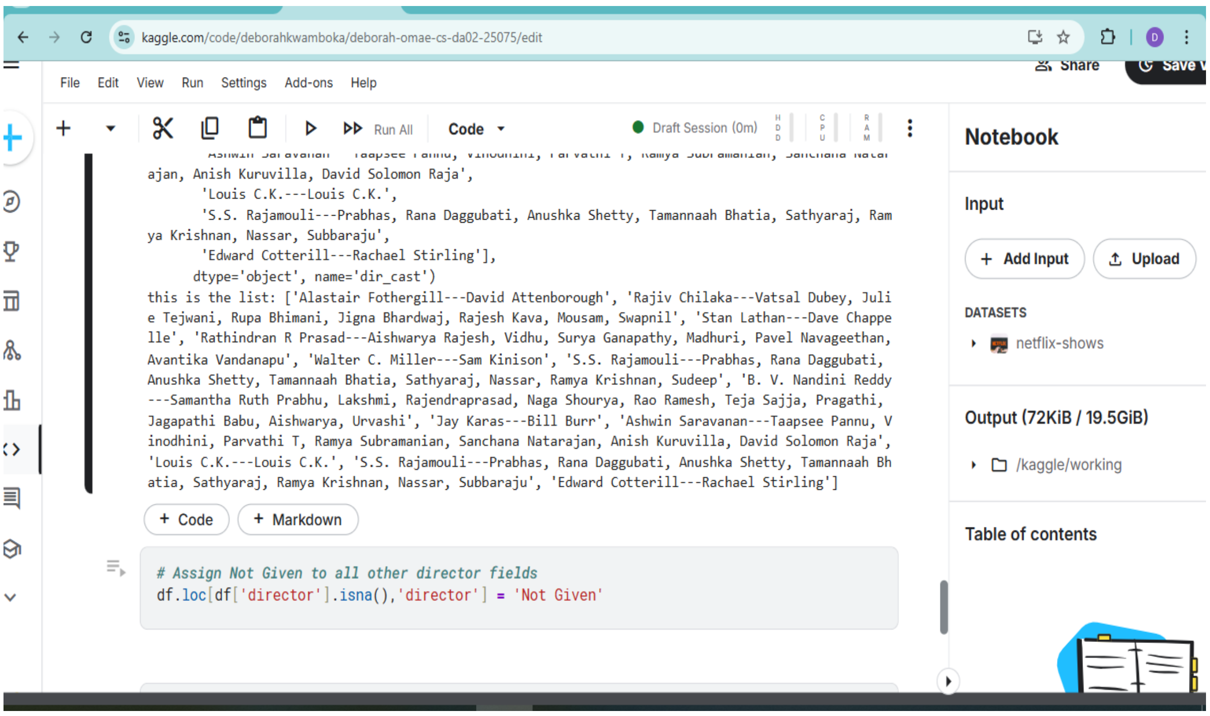
Figure 7: This filter shows the filtered counts which are the ones with counts above 3

- filtered_values = filtered_counts.index

  This code extracts the values (the combined strings) of the filtered pairs.

- lst_dir_cast = list(filtered_values)

  This code converts the filtered pair labels into a list for iteration.



```
       'Ashwin Saravanan---Taapsee Pannu, Vinodhini, Parvathi T, Ramya Subramanian, Sanchana Natar
ajan, Anish Kuruvilla, David Solomon Raja',
       'Louis C.K.---Louis C.K.',
       'S.S. Rajamouli---Prabhas, Rana Daggubati, Anushka Shetty, Tamannaah Bhatia, Sathyaraj, Ram
ya Krishnan, Nassar, Subbaraju',
       'Edward Cotterill---Rachael Stirling'],
      dtype='object', name='dir_cast')
this is the list: ['Alastair Fothergill---David Attenborough', 'Rajiv Chilaka---Vatsal Dubey, Juli
e Tejwani, Rupa Bhimani, Jigna Bhardwaj, Rajesh Kava, Mousam, Swapnil', 'Stan Lathan---Dave Chappe
lle', 'Rathindran R Prasad---Aishwarya Rajesh, Vidhu, Surya Ganapathy, Madhuri, Pavel Navageethan,
Avantika Vandanapu', 'Walter C. Miller---Sam Kinison', 'S.S. Rajamouli---Prabhas, Rana Daggubati,
Anushka Shetty, Tamannaah Bhatia, Sathyaraj, Nassar, Ramya Krishnan, Sudeep', 'B. V. Nandini Reddy
---Samantha Ruth Prabhu, Lakshmi, Rajendraprasad, Naga Shourya, Rao Ramesh, Teja Sajja, Pragathi,
Jagapathi Babu, Aishwarya, Urvashi', 'Jay Karas---Bill Burr', 'Ashwin Saravanan---Taapsee Pannu, V
inodhini, Parvathi T, Ramya Subramanian, Sanchana Natarajan, Anish Kuruvilla, David Solomon Raja',
'Louis C.K.---Louis C.K.', 'S.S. Rajamouli---Prabhas, Rana Daggubati, Anushka Shetty, Tamannaah Bh
atia, Sathyaraj, Ramya Krishnan, Nassar, Subbaraju', 'Edward Cotterill---Rachael Stirling']
```

```
# Assign Not Given to all other director fields
df.loc[df['director'].isna(),'director'] = 'Not Given'
```

Figure 8: This screenshot shows the filtered pair values converted to a list

- for i in lst_dir_cast: director, cast = i.split('---'); dict_direcast[director] = cast
  This code Splits each string at '---' and builds a dictionary mapping each director to their cast string.

- for i in range(len(dict_direcast)): df.loc[(df['director'].isna()) & (df['cast'] == list(dict_direcast.items())[i][1]), 'director'] = list(dict_direcast.items())[i][0]
  This code block Iterates through the dictionary and fills in the missing director values for rows with missing director values but matching cast.

Step 7

In this step I assigned the string value not given' to all other director fields

Code:

# Assign Not Given to all other director fields

df.loc[df['director'].isna(),'director'] = 'Not Given'

Step 8

In this step I used the director's column to fill in missing countries such that if a director's country is known in one row, but missing in another row, the code uses that known relationship (director - country) to fill in the missing value.

Code:

```
directors = df['director']

countries = df['country']

#pair each director with their country use zip() to get an iterator of tuples

pairs = zip(directors, countries)

# Convert the list of tuples into a dictionary

dir_cntry = dict(list(pairs))

# Director matched to Country values used to fill in null country values

for i in range(len(dir_cntry)):

df.loc[(df['country'].isna()) & (df['director'] == list(dir_cntry.items())[i][0]),'country'] = list(dir_cntry.items())[i][1]

# Assign Not Given to all other country fields
df.loc[df['country'].isna(),'country'] = 'Not Given'

# Assign Not Given to all other fields

df.loc[df['cast'].isna(),'cast'] = 'Not Given'
```

code explanation:

- directors = df['director']
  This code selects the director column from the dataset.
- countries = df['country']
  This code selects the country column from the dataset.
- pairs = zip(directors, countries)

This code pairs each director with their corresponding country using zip(), producing an iterator of tuples such as ("Christopher Nolan", "USA").

- dir_cntry = dict(list(pairs))

  This code converts the list of tuples into a dictionary, mapping each director to a country.
- for i in range(len(dir_cntry)): df.loc[(df['country'].isna()) & (df['director'] == list(dir_cntry.items())[i][0]), 'country'] = list(dir_cntry.items())[i][1]

  This code loops through the dictionary. For rows where the country value is missing but the director matches, it fills in the country from the dictionary.
- df.loc[df['country'].isna(), 'country'] = 'Not Given'

  This code assigns the value "Not Given" to any rows where the country is still missing after the dictionary-based filling.
- df.loc[df['cast'].isna(), 'cast'] = 'Not Given'

  This code assigns the value "Not Given" to any rows where the cast information is missing.

step 9

In this step I dropped all rows that have null values

code

df.drop(df[df['date_added'].isna()].index,axis=0,inplace=True)

df.drop(df[df['rating'].isna()].index,axis=0,inplace=True)

df.drop(df[df['duration'].isna()].index,axis=0,inplace=True)

code explanation:

df.drop(df[df['date_added'].isna()].index, axis=0, inplace=True)

This code drops all rows from the dataset where the date_added column has missing (NaN) values.

df.drop(df[df['rating'].isna()].index, axis=0, inplace=True).

This code drops all rows from the dataset where the rating column has missing values.

df.drop(df[df['duration'].isna()].index, axis=0, inplace=True).

This code drops all rows from the dataset where the duration column has missing values.

Step 10

In this step I checked for errors if any and confirmed there were inconsistencies in my cleaned data

Code

import datetime as dt

sum(df['date_added'].dt.year < df['release_year'])

df.loc[(df['date_added'].dt.year < df['release_year']),['date_added','release_year']]

# sample some of the records and check that they have been accurately replaced

df.iloc[[1551,1696,2920,3168]]

#Confirm that no more release_year inconsistencies

sum(df['date_added'].dt.year < df['release_year'])

code explanation:

This code was used to check and resolve inconsistencies between the date_added and release_year columns. First, the datetime library was imported to work with date values. The code then checked whether there were any records where the year of date_added was earlier than the release_year using sum(df['date_added'].dt.year < df['release_year']). Any such mismatched rows were retrieved and displayed with df.loc[(df['date_added'].dt.year < df['release_year']), ['date_added','release_year']] for inspection. To further verify the corrections, a few specific rows were sampled with df.iloc[[1551,1696,2920,3168]]. Finally, the check was repeated with sum(df['date_added'].dt.year < df['release_year']) to confirm that no more inconsistencies remained after the cleaning process.

Step 11

In this step, I validated the dataset to ensure accuracy, consistency, and completeness. I removed any temporary columns that I had created during the wrangling process, such as *dir_cast*, so that only the necessary fields remained. **The code**: *df.drop(columns=['dir_cast'], inplace=True)*

I then checked that each column had the correct data type, for example confirming that *date added* was in datetime format and *duration_value* was numeric. To identify potential anomalies, I applied business logic and sanity rules, such as flagging records that appeared before 1997. I also ensured that no important fields were left missing. To further confirm the quality of the data, I sampled a few rows for visual inspection, and finally I reset the index to provide a clean and properly ordered dataset for analysis.

Step 12

In this final step I converted my cleaned data to a csv file.

Code:

df.to_csv('/kaggle/working/cleaned_netflix.csv', index=False)

Link to code

Link to Code: https://www.kaggle.com/code/deborahkwamboka/deborah-omae-cs-da02-25075

Conclusion

This week I gained hands on experience on the techniques of data wrangling. I restructured, formatted and cleaned my data set to a usable format. This assignment project was an eye opener to the use of many pandas' function and python data structures such as lists and dictionaries.