**DATA AND ARTIFICIAL INTELLIGENCE**

**CYBER SHUJAA PROGRAM**

**WEEK 2 ASSIGNMENT : DATA WRANGLING**

**STUDENT NAME:DOROTHY AWINO ONGONGA**

**STUDENT ID: CS-DA01-25075**

**KAGGLE JUPYTER NOTEBOOK LINK:** [**https://www.kaggle.com/code/awino614/netflix-movies-datasets-data-wrangling-project**](https://www.kaggle.com/code/awino614/netflix-movies-datasets-data-wrangling-project)

**GOOGLE COLAB LINK:** [**https://colab.research.google.com/#fileId=https%3A//storage.googleapis.com/kaggle-colab-exported-notebooks/awino614/netflix-movies-datasets-data-wrangling-project.badba3af-be24-4ac4-a199-d35534593125.ipynb%3FX-Goog-Algorithm%3DGOOG4-RSA-SHA256%26X-Goog-Credential%3Dgcp-kaggle-com%2540kaggle-161607.iam.gserviceaccount.com/20250609/auto/storage/goog4\_request%26X-Goog-Date%3D20250609T193210Z%26X-Goog-Expires%3D259200%26X-Goog-SignedHeaders%3Dhost%26X-Goog-Signature%**](https://colab.research.google.com/#fileId=https%3A//storage.googleapis.com/kaggle-colab-exported-notebooks/awino614/netflix-movies-datasets-data-wrangling-project.badba3af-be24-4ac4-a199-d35534593125.ipynb%3FX-Goog-Algorithm%3DGOOG4-RSA-SHA256%26X-Goog-Credential%3Dgcp-kaggle-com%2540kaggl)

**GITHUB LINK:** [**www.githubaccount.dorothyawino**](http://www.githubaccount.dorothyawino)

**INTRODUCTION**

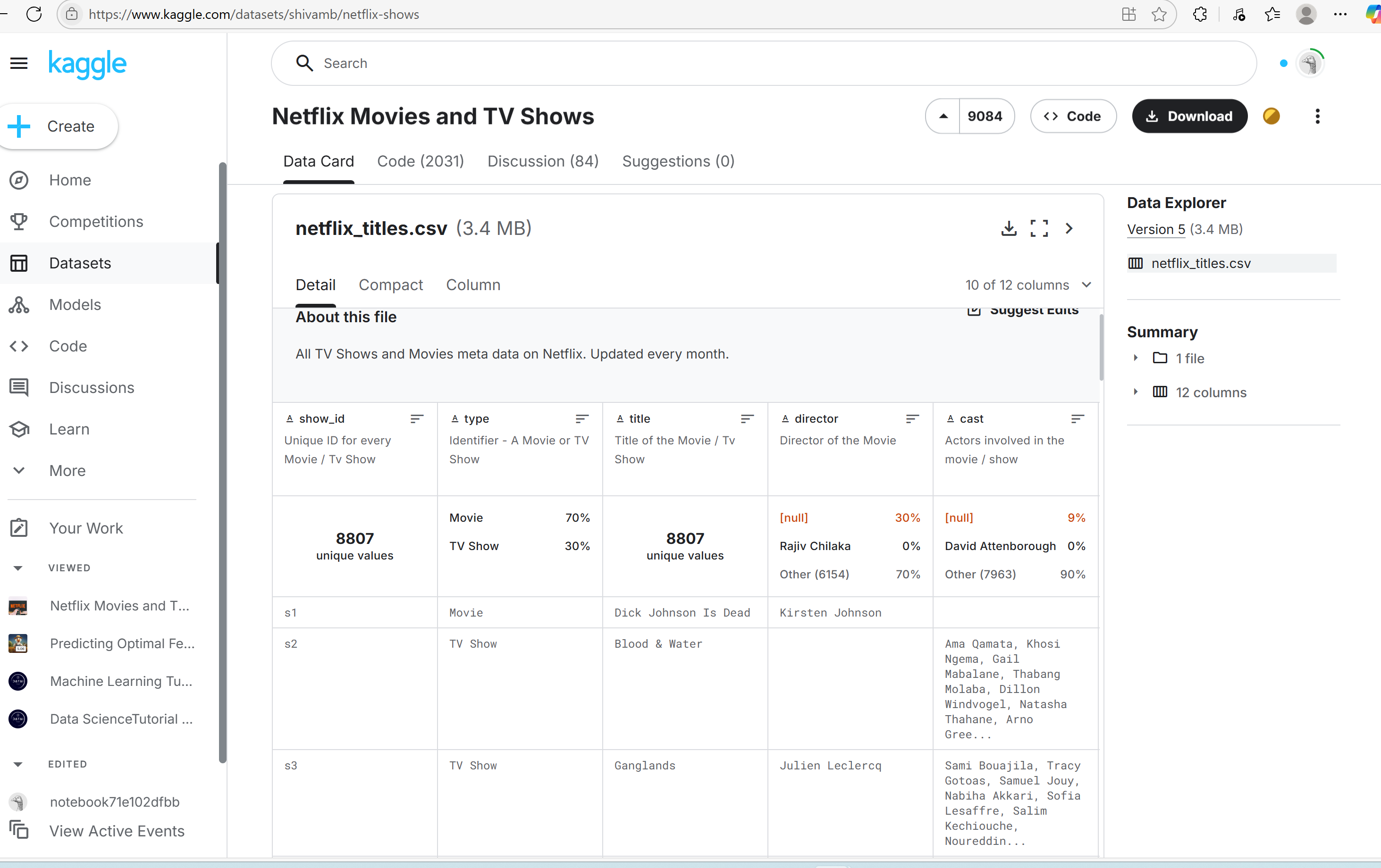
This week’s assignment will develop hands-on experience automating web data wrangling using Kaggle Data Set and publishing finished project on Kaggle.

Required to practice the Data Wrangling concepts to clean up the Netflix dataset available on Kaggle

Link:  <https://www.kaggle.com/datasets/shivamb/netflix-shows>

The objectives of the assignment are 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.



-It Displays the Data Card with an overview of the columns that make up the Data Set. There are 8807

- Unique Shows and out of that there are two groups made up of 70% Movies and 30% TV Shows.

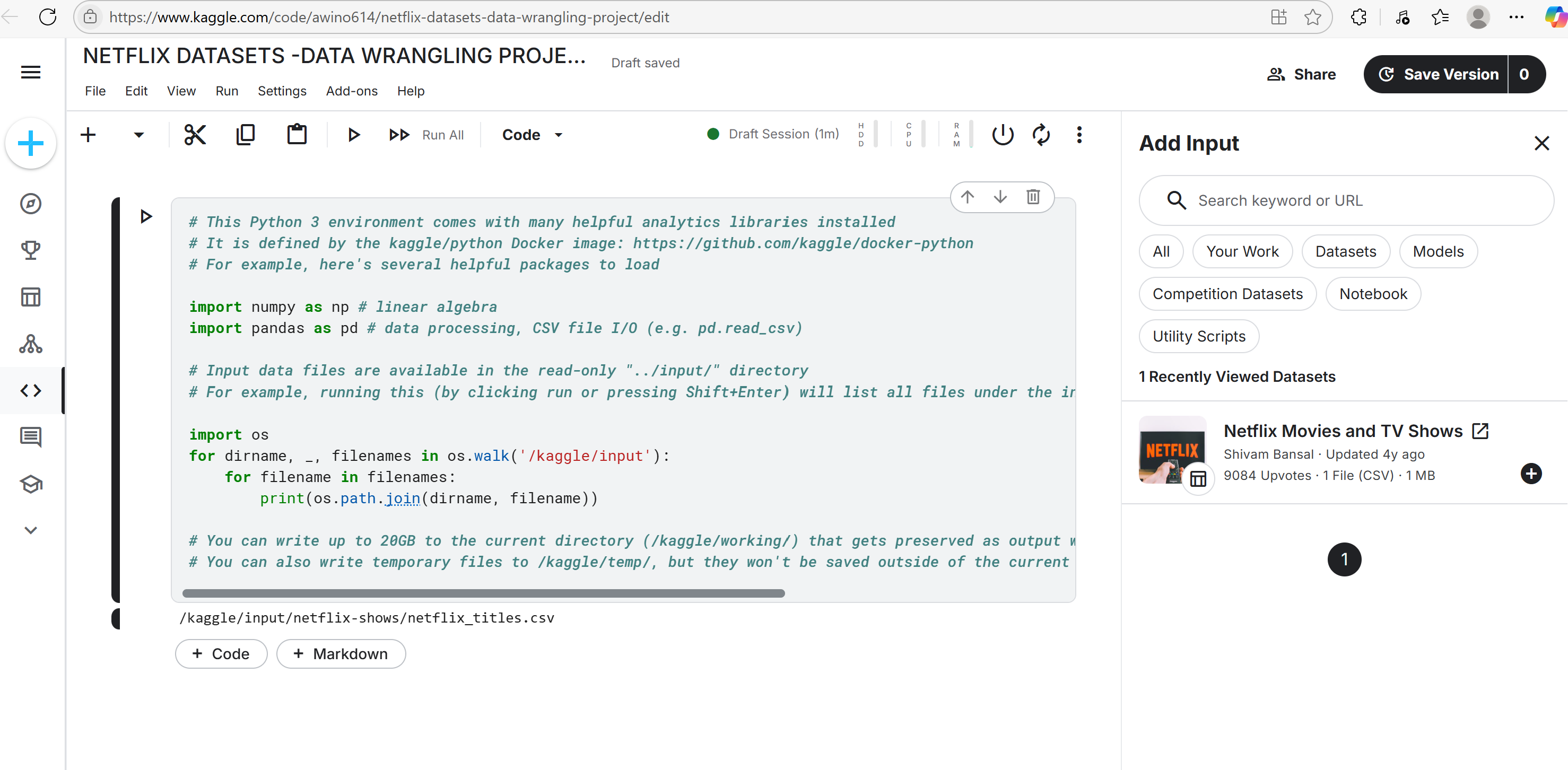
-Another observed attribute is that there are no duplicates but for the Column Titled Director there are quite a number of nulls,30% of the movies don’t have an indicated Director.

-Additionally,9% of the movies don’t have a cast which is a major influence on the viewership.

-The two most highly influenced areas for production are India 11% and United States with 32%.

-Additionally,you can see the Date and Time,Release Year and the Duration in minutes and the number of seasons produced.

-I created a new JupyterNotebook on Kaggle Titled ‘NETFLIX DATASETS-DATA WRANGLING PROJECT’ and Added The Dataset



The code displayed is default and once I run the current cell I get to observe the location of the

file.

Using The Markdown Feature,I gave a short description of the Steps I followed for the Data Wrangling Project.



I Imported the Data to a Pandas DataFrame -/kaggle/input/netflix-shows/netflix\_titles.csv



**STEP 1. Discovery**

#Have a quick overview of the data

df.info()

# Number of rows and columns

print("Shape of the dataset (R x C):", df.shape)

# List of all column names

print("Columns in the dataset:\n", df.columns.tolist())

# Data types of each column

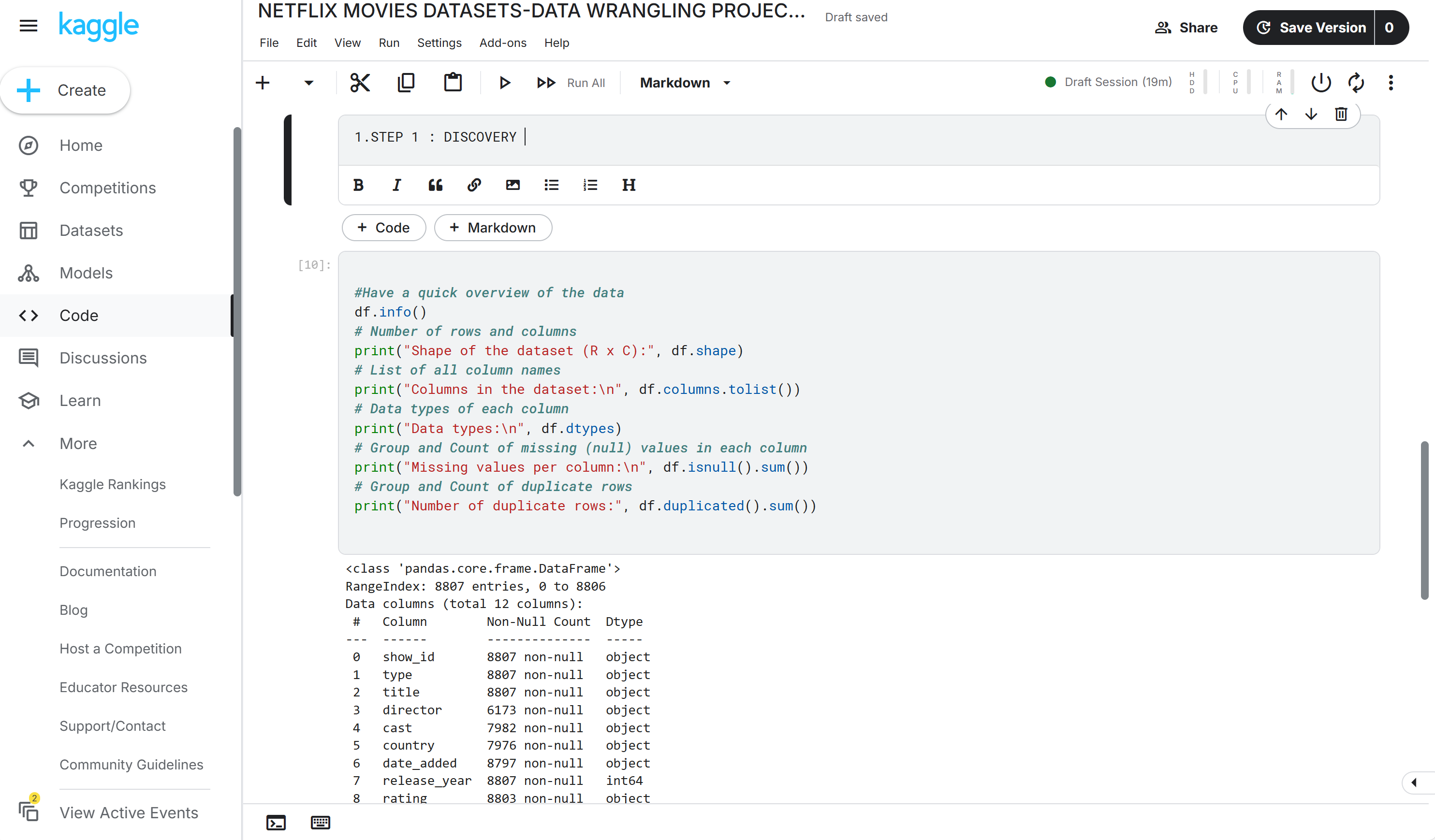
print("Data types:\n", df.dtypes)

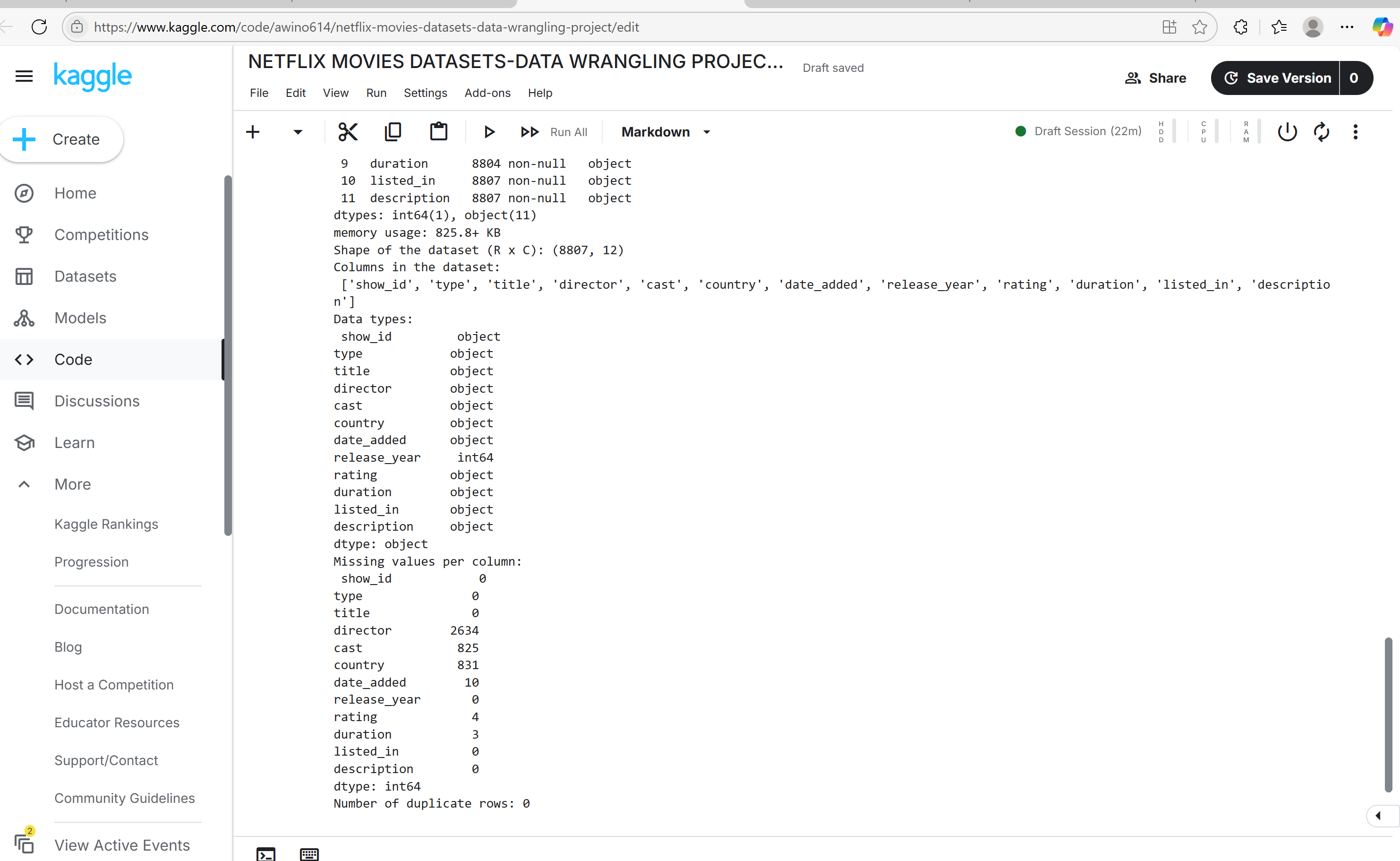
# Group and Count of missing (null) values in each column

print("Missing values per column:\n", df.isnull().sum())

# Group and Count of duplicate rows

print("Number of duplicate rows:", df.duplicated().sum())





**STEP 2. Structuring**

# Convert 'date\_added' to datetime

df['date\_added'] = pd.to\_datetime(df['date\_added'],format='mixed')

# Separate 'duration' into numeric value and unit (e.g., '90 min' → 90, 'min')

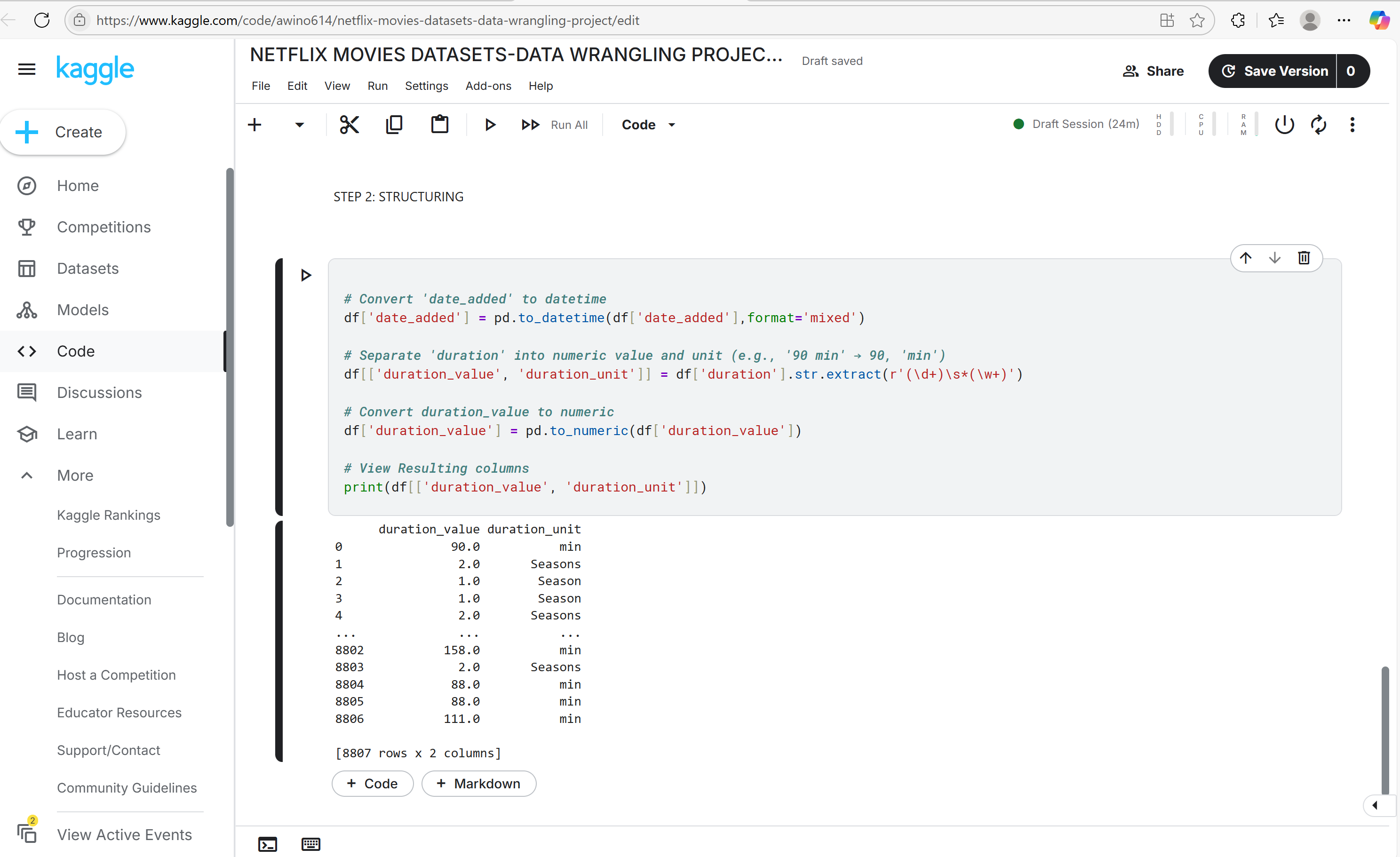
df[['duration\_value', 'duration\_unit']] = df['duration'].str.extract(r'(\d+)\s\*(\w+)')

# Convert duration\_value to numeric

df['duration\_value'] = pd.to\_numeric(df['duration\_value'])

# View Resulting columns

print(df[['duration\_value', 'duration\_unit']])



**STEP 3 : CLEANING**

# Check for duplicate rows

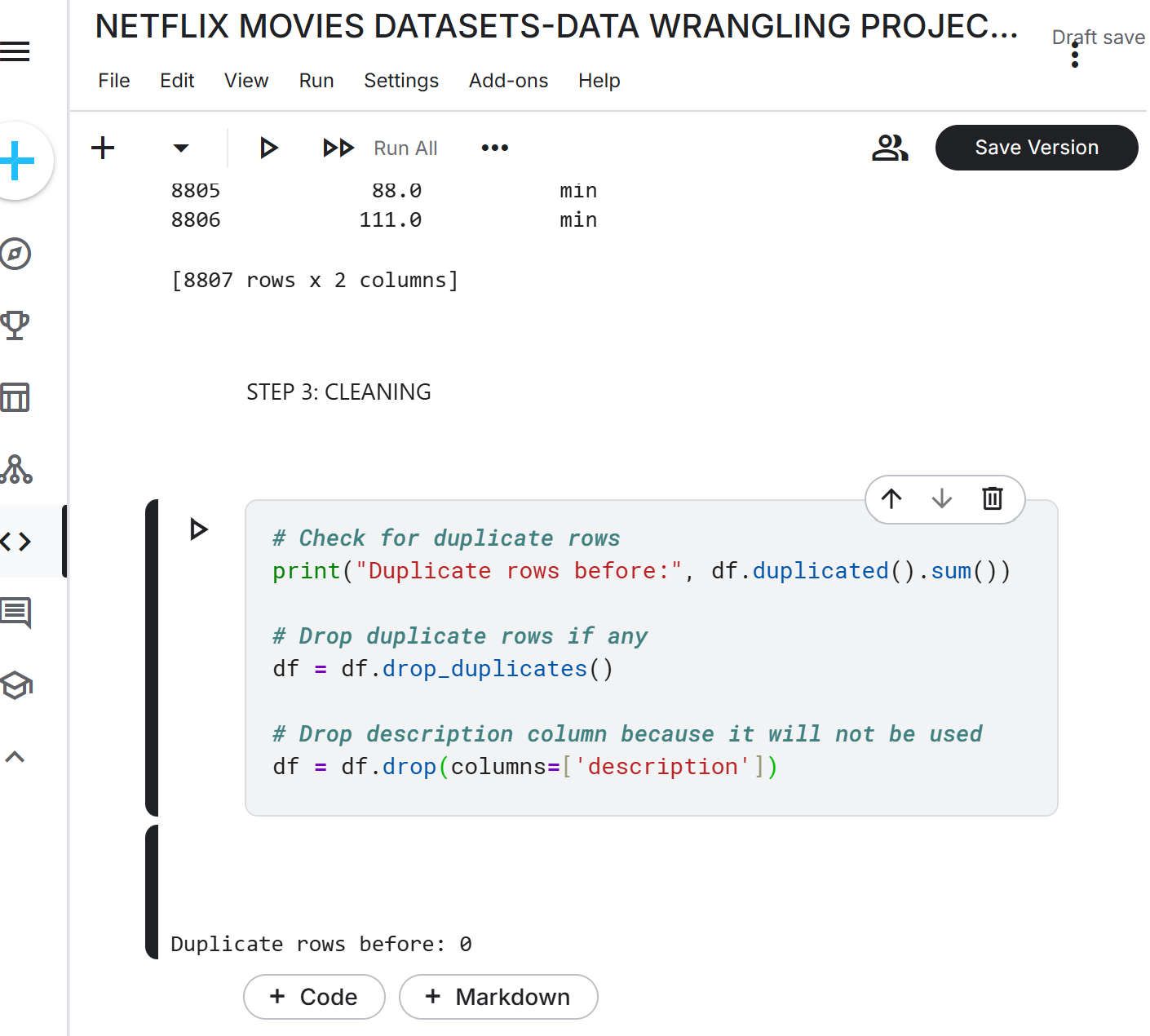
print("Duplicate rows before:", df.duplicated().sum())

# Drop duplicate rows if any

df = df.drop\_duplicates()

# Drop description column because it will not be used

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



# Impute Director values by using relationship between cast and director

# Create a new column with combined 'director' and 'cast' values

df['dir\_cast'] = df['director'] + '---' + df['cast']

# Count the occurrences of each pair

counts = df['dir\_cast'].value\_counts()

# Filter pairs that appear at least 3 times

filtered\_counts = counts[counts >= 3]

filtered\_values = filtered\_counts.index

# Convert to list of valid director-cast combinations

lst\_dir\_cast = list(filtered\_values)

# Create dictionary to map cast to director

dict\_direcast = dict()

for pair in lst\_dir\_cast:

director, cast = pair.split('---') # split the pair into director and cast

dict\_direcast[cast] = director # map cast to director

# Fill in missing directors based on cast

for cast, director in dict\_direcast.items():

df.loc[(df['director'].isna()) & (df['cast'] == cast), 'director'] = director

# Assign 'Not Given' to all missing director fields

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

# Use directors to fill missing countries

directors = df['director']

countries = df['country']

# Pair each director with their country using zip() and convert to dictionary

pairs = zip(directors, countries)

dir\_cntry = dict(pairs)

# Fill in missing country values using director-country mapping

for director, country in dir\_cntry.items():

df.loc[(df['country'].isna()) & (df['director'] == director), 'country'] = country

# Assign 'Not Given' to all other missing country fields

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

# Assign 'Not Given' to all missing cast fields

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

# Drop rows with missing values in certain fields

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)

# Check for inconsistencies: date\_added earlier than release\_year

import datetime as dt

sum(df['date\_added'].dt.year < df['release\_year'])

# Display records with inconsistency

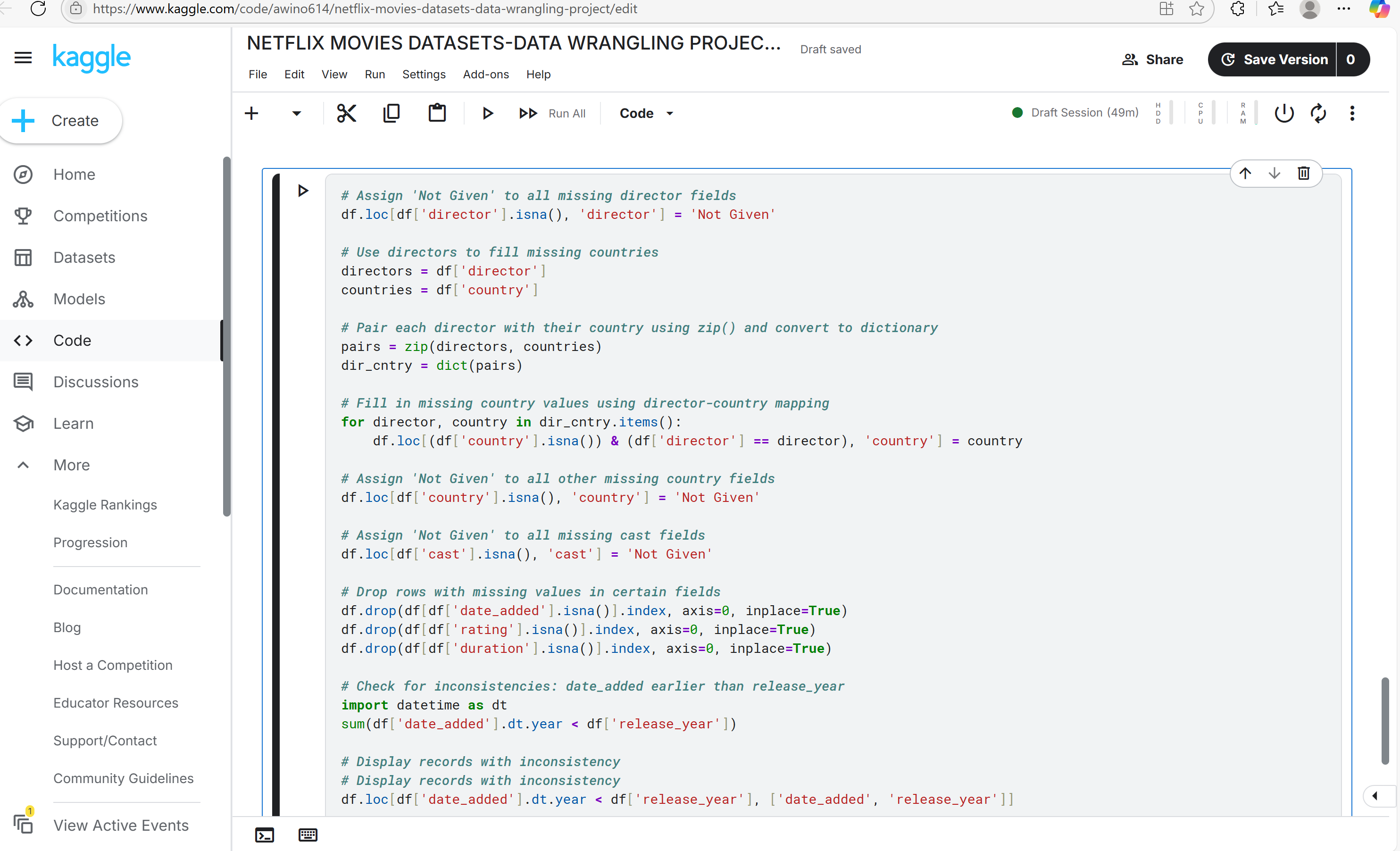
df.loc[df['date\_added'].dt.year < df['release\_year'], ['date\_added', 'release\_year']]

# Sample some rows to verify corrections

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

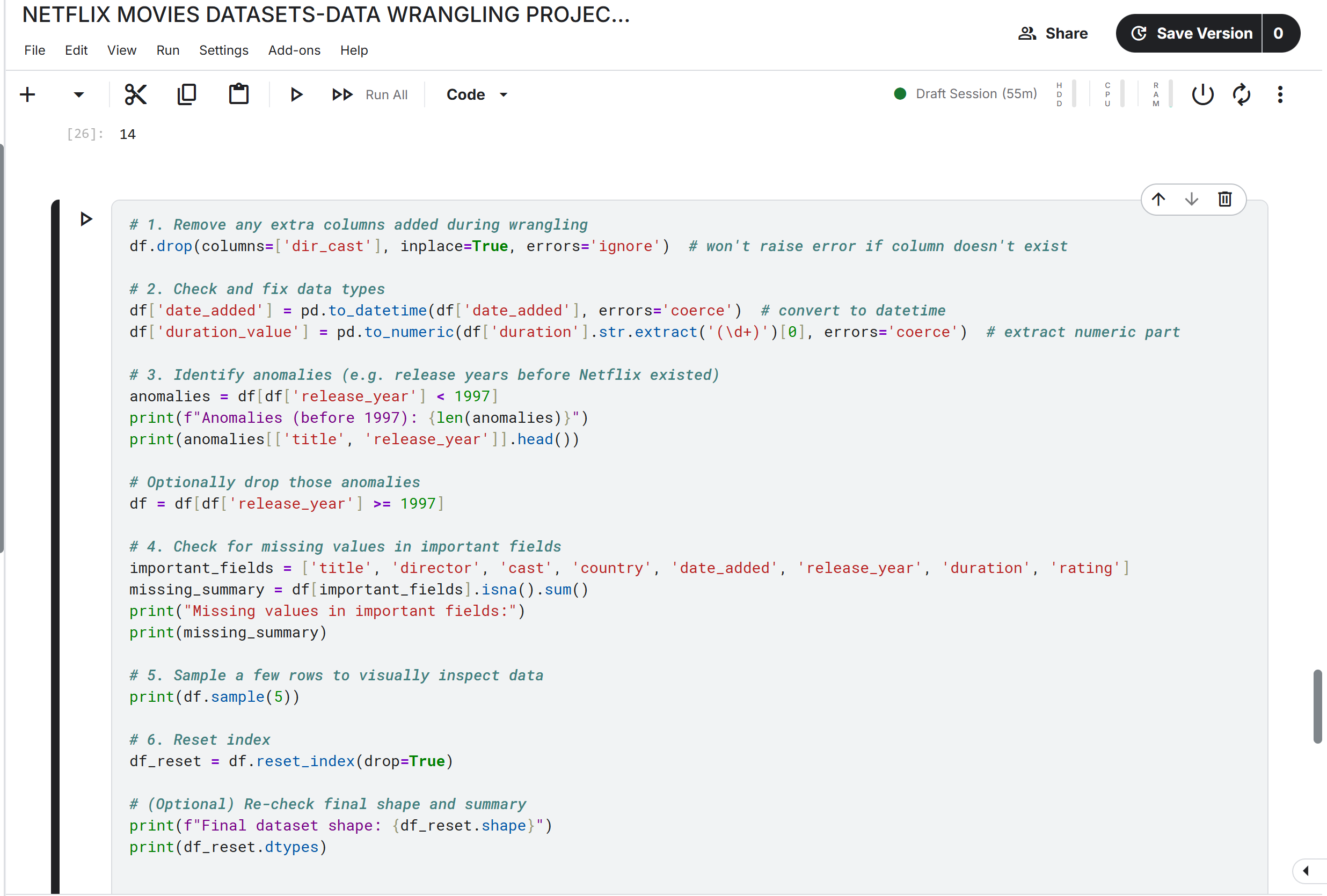
# Confirm no remaining release\_year inconsistencies

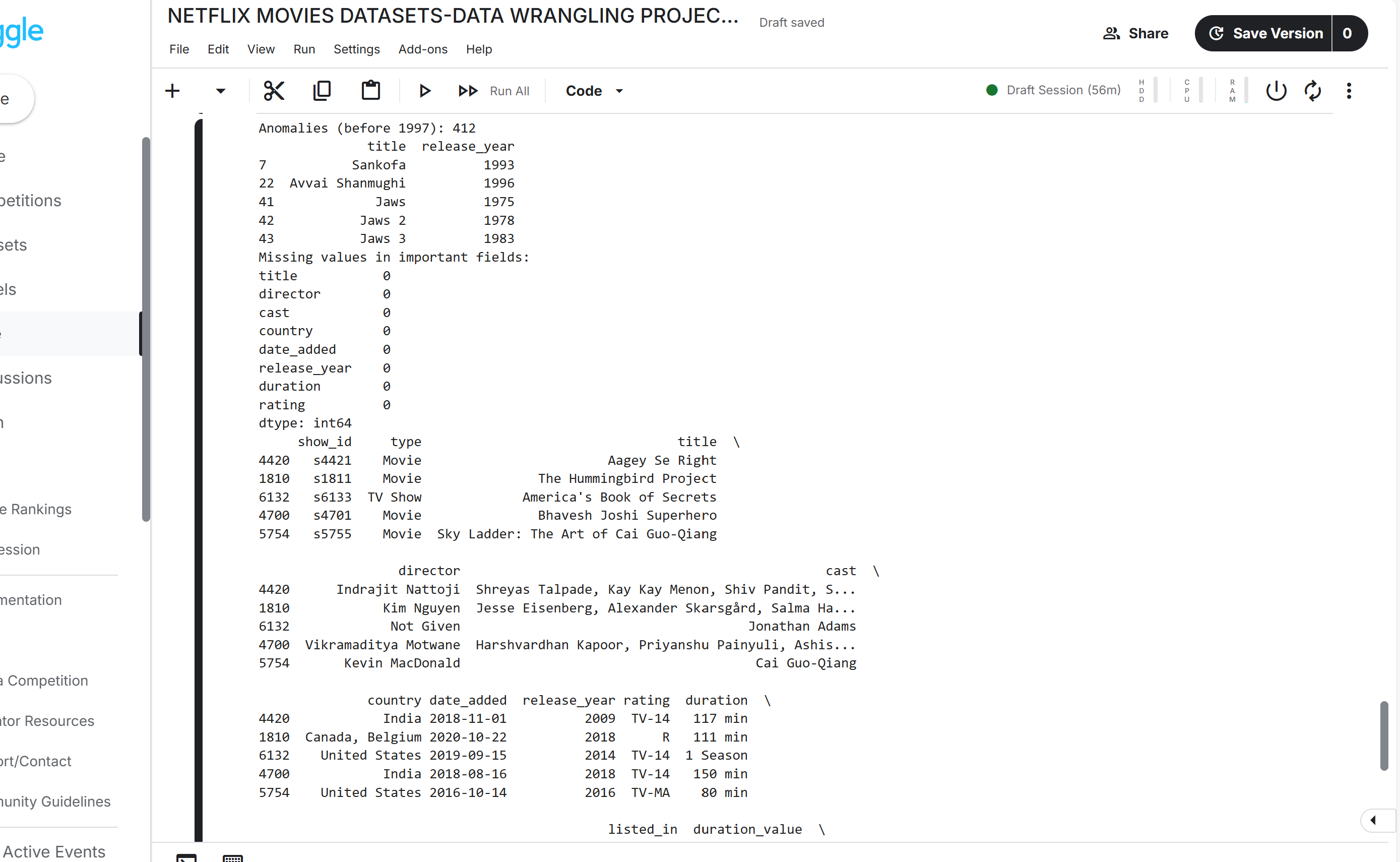
sum(df['date\_added'].dt.year < df['release\_year'])

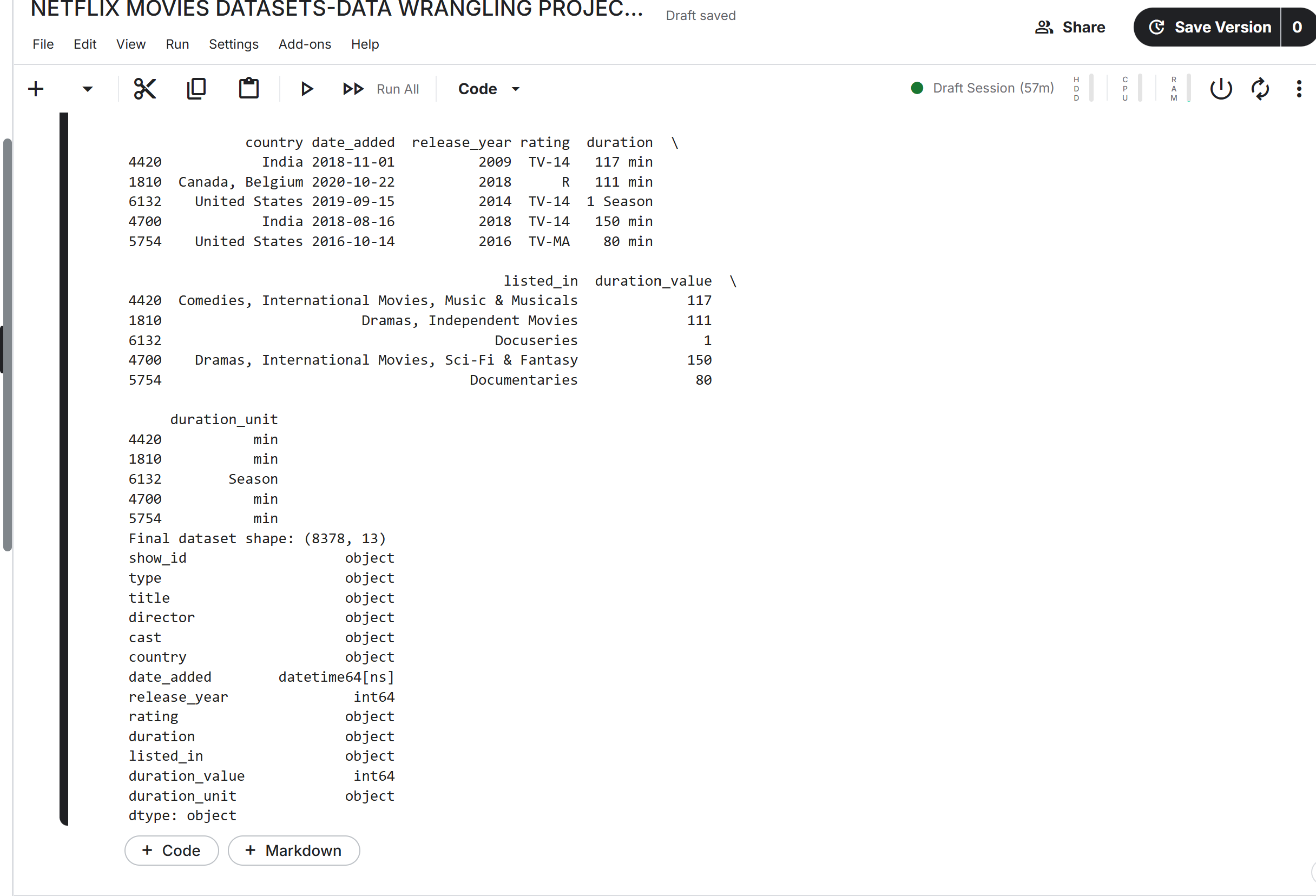


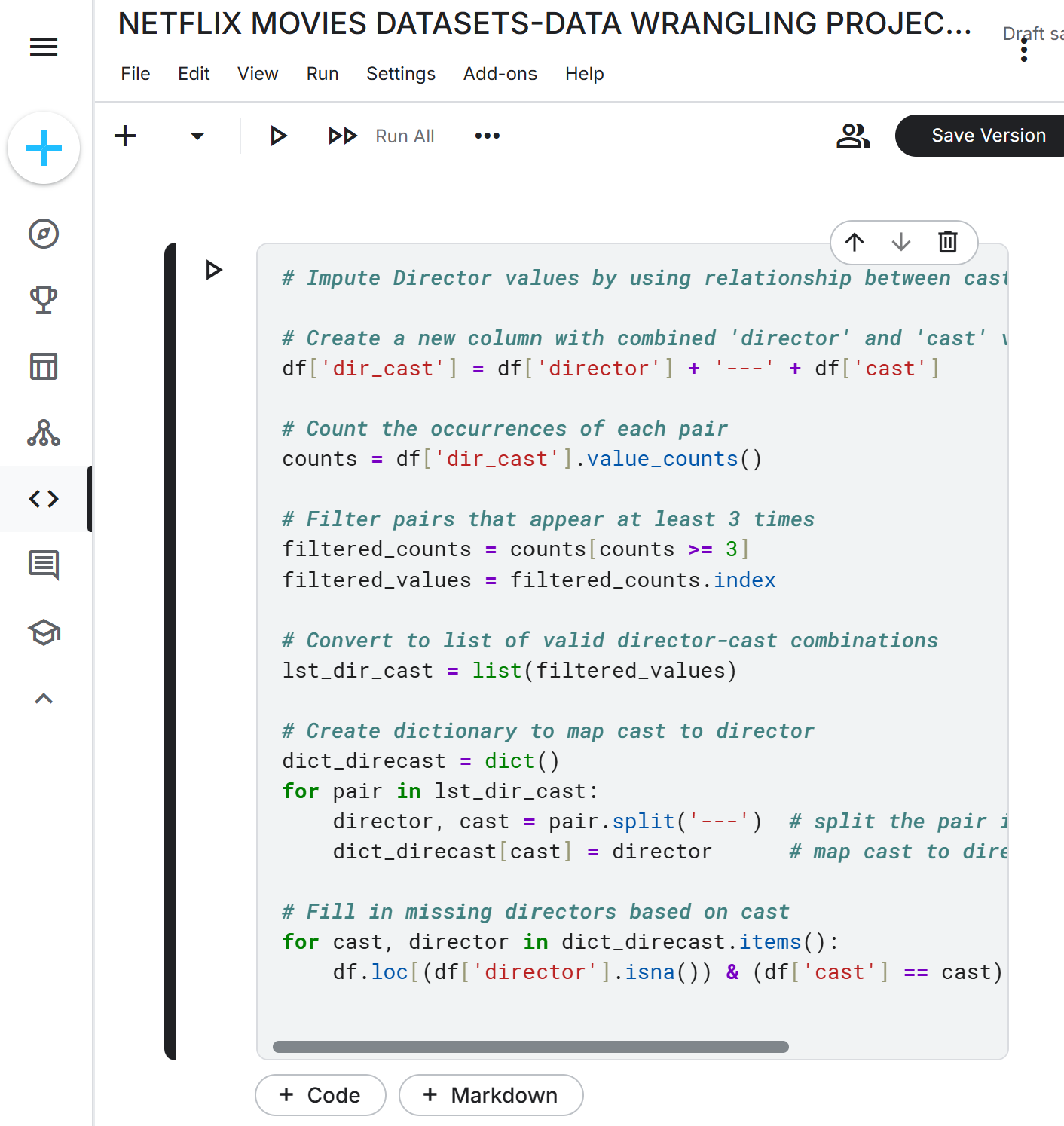
**STEP 4: VALIDATION**

* Remove any columns you may have added during wrangling e.g.
* df.drop(columns=['dir\_cast'], inplace=True)
* Check the consistency, accuracy, and completeness of the data
* Ensure each column has the correct data type e.g. verify that date\_added is datetime and duration\_value is numeric.
* Use business logic or sanity rules to identify anomalies e.g. records before 1997
* Ensure no important fields are still missing
* Sample a few rows to check visually e.g. df.sample(5)
* Reset the Index e.g. df\_reset = df.reset\_index(drop=True)

****

****

****



VALIDATION CODE: # 1. Remove any extra columns added during wrangling

df.drop(columns=['dir\_cast'], inplace=True, errors='ignore') # won't raise error if column doesn't exist

# 2. Check and fix data types

df['date\_added'] = pd.to\_datetime(df['date\_added'], errors='coerce') # convert to datetime

df['duration\_value'] = pd.to\_numeric(df['duration'].str.extract('(\d+)')[0], errors='coerce') # extract numeric part

# 3. Identify anomalies (e.g. release years before Netflix existed)

anomalies = df[df['release\_year'] < 1997]

print(f"Anomalies (before 1997): {len(anomalies)}")

print(anomalies[['title', 'release\_year']].head())

# Optionally drop those anomalies

df = df[df['release\_year'] >= 1997]

# 4. Check for missing values in important fields

important\_fields = ['title', 'director', 'cast', 'country', 'date\_added', 'release\_year', 'duration', 'rating']

missing\_summary = df[important\_fields].isna().sum()

print("Missing values in important fields:")

print(missing\_summary)

# 5. Sample a few rows to visually inspect data

print(df.sample(5))

# 6. Reset index

df\_reset = df.reset\_index(drop=True)

# (Optional) Re-check final shape and summary

print(f"Final dataset shape: {df\_reset.shape}")

print(df\_reset.dtypes)

### 🧠 Notes:

* The errors='ignore' in drop() ensures the code won't crash if 'dir\_cast' is already gone.
* duration\_value is extracted using regex to pull numbers from strings like "90 min".
* reset\_index(drop=True) avoids adding the old index as a new column.

**FINAL STEP:Publish**

* # Save as CSV   
  df.to\_csv('/kaggle/working/cleaned\_netflix.csv', index=False