# Dataset Description:

The dataset describes a lot of movies such as the year of show, the stars, its

type(adventure,romance,action,etc),rating for the movie from the audience,and its runtime.

# Problem Definition :

The dataset contains nulls, duplicates records, outliers,NANS)

Methodology:

**Cleaning:**

1. The first cell reads the CSV file named "movies.csv" and loads it into a pandas DataFrame called data.
2. The second cell Identifies duplicate rows in the DataFrame.
3. The third cell removes duplicate rows from the DataFrame. It returns a new DataFrame without duplicates.
4. The fourth cell counts the number of missing values (NaNs) in each column of the DataFrame data and then prints the count of missing values for each column in the DataFrame.
5. The fifth cell calculates the total number of cells in the DataFrame (the product of its shape, which is the number of rows times the number of columns), sums up the counts of missing values across all columns, giving the total number of missing values in the DataFrame,then calculates the percentage of missing values in the DataFrame.
6. Drops rows with any missing values from the DataFrame.
7. Creates a subset DataFrame containing only the columns from 'VOTES' to 'Gross'.
8. Fills missing values (NaNs) in the subset DataFrame with zeros.
9. Creates a box plot for the numerical columns specified in numerical columns. plt.ylim(500,1000) Sets the y-axis limits for the box plot to control the range of values displayed.
10. Calculates the first quartile (25th percentile) for the numerical columns specified in numerical columns.Calculates the third quartile (75th percentile) for the numerical columns. IQR=Q3-Q1 Calculates the interquartile range (IQR) for the numerical columns then prints the IQR scores for the numerical columns.

**Validation:**

1. First cell in validation code reads a CSV file containing movie data, defines a schema specifying the expected structure of the data, validates the actual data against this schema, and prints the validated DataFrame if it meets the specified criteria.
2. Second cell in validation code uses Pandera to automatically infer the schema of a DataFrame, converts the inferred schema into a script format, and prints the script to facilitate schema definition and customization.

validator.expect\_column\_values\_to\_not\_be\_null('MOVIES')

1. to tell Great Expectations to expect that the column should not contain any null values. If any null values are found during the validation process, Great Expectations will raise an exception to indicate that the expectation has failed.

validator.expect\_column\_values\_to\_be\_between("RATING", min\_value=0, max\_value=100)

1. to tell Great Expectations to expect that all values in the Rating column should be between 0 and 100 .If any values fall outside of this range during the validation process, Great Expectations will raise an exception to indicate that the expectation has failed.

validator.expect\_column\_values\_to\_be\_unique(column='MOVIES')

1. to tell Great Expectations to expect that all values in the movies column should be unique. If there are any duplicate values in this column during the validation process, Great Expectations will raise an exception to indicate that the expectation has failed.

validator.expect\_column\_values\_to\_be\_between('RunTime', 0, 100)

1. It defines an expectation for the column named RunTime in the dataset, ensuring that all values in this column fall within the specified range of 0 to 100.

x = ['10000', '20000','30000']

validator.expect\_column\_values\_to\_be\_in\_set('VOTES', value\_set= x)

1. To tell Great Expectations to expect that all values in the votes column should be present in the set[1000,2000,3000]. If any value in this column is not found in the set during the validation process, Great Expectations will raise an exception to indicate that the expectation has failed.

x = ['100', '200','300']

validator.expect\_column\_values\_to\_be\_in\_set('Gross', value\_set= x)

1. The same is made for Gross column but the set is [100,200,300]

context.add\_expectation\_suite("my\_expectation\_suite")

validator.save\_expectation\_suite()

1. These lines of code are used in the Great Expectations library to add an expectation suite to a context and then save the expectation suite.

context.view\_validation\_result(checkpoint\_result)

checkpoint\_result

1. The function is used to view the validation result of a checkpoint in the Great Expectations context. It displays the validation results in a user-friendly format, allows to inspect any issues or discrepancies found during the validation process.

**3) Security Privacy:**

import string

def randomize\_values(df):

for column in df.columns:

if df[column].dtype == 'O':

df[column] = [''.join(random.choices(string.ascii\_letters + string.digits, k=10))

for \_ in range(len(df))] return df

11)The function is designed to randomize the values in string columns of a DataFrame while adding a new column of unique identifiers to it.

from anonymizedf.anonymizedf import anonymize

an = anonymize(df)

an.fake\_names("MOVIES")df

12)This code is using the anonymized library to anonymize the 'MOVIES' column in the DataFrame. The anonymize function from the library is applied to the DataFrame, and then the fake\_names method is called with the column name 'MOVIES' as an argument to anonymize the values in that column. Finally, the DataFrame df is returned with the 'MOVIES' column anonymized.

df\_swp = df.copy()

df\_swp['RunTime\_swap'] = df['RunTime'].sample(frac=1).reset\_index(drop=True)

df\_swp

13)This code creates a copy of the DataFrame df called df\_swp. Then, it adds a new column named 'RunTime\_swap' to df\_swp, which contains the values from the 'RunTime' column of df in a random order.

from faker import Faker

faker = Faker()

synthetic\_data = {

'MOVIES': [faker.name() for \_ in range(len(df))],

'VOTES': [faker.random\_int(5000,10000) for \_ in range(len(df))]

}

df\_synthetic = pd.DataFrame(synthetic\_data)

print(df\_synthetic)

df

14)This code generates synthetic data for the 'MOVIES' and 'VOTES' columns using the Faker library. For the 'MOVIES' column, it generates random names using faker.name(), and for the 'VOTES' column, it generates random integers between 5000 and 10000.

import nltk

from nltk.tokenize import word\_tokenize

df\_token= df

nltk.download('punkt')

def tokenize\_data(data):

return word\_tokenize(data)

df\_token['Tokenized\_ONE-LINE'] = df\_token['ONE-LINE'].apply(tokenize\_data)

df\_token

15)This code snippet tokenizes the text in the 'ONE-LINE' column of the DataFrame df using NLTK's word\_tokonize function. First, it imports the necessary modules from NLTK. Then, it defines a function to tokenize data that takes a string input and tokenizes it. Finally, it applies this function to the 'ONE-LINE' column of the DataFrame df and stores the tokenized results in a new column called 'Tokenized\_ONE-LINE' in the DataFrame.

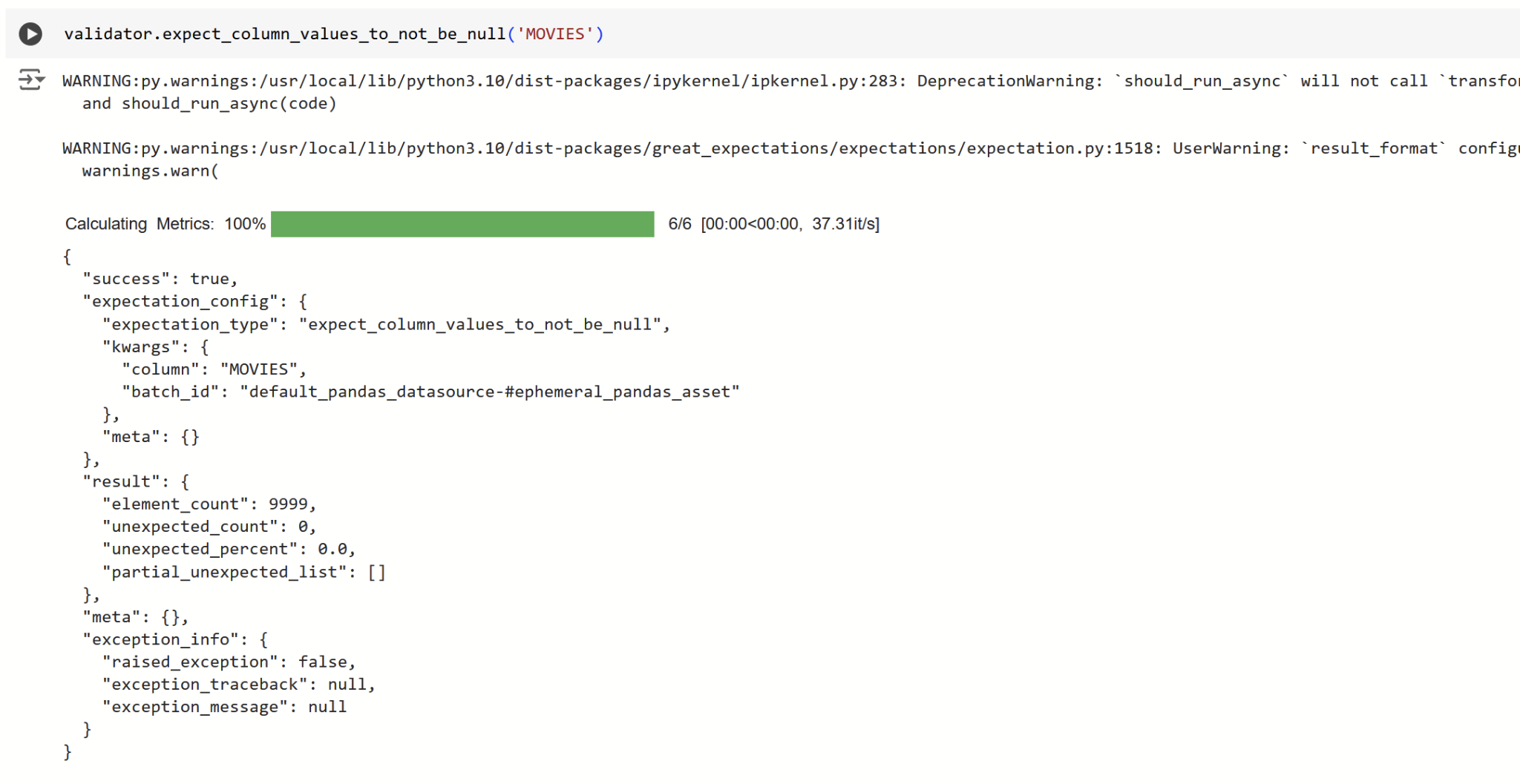
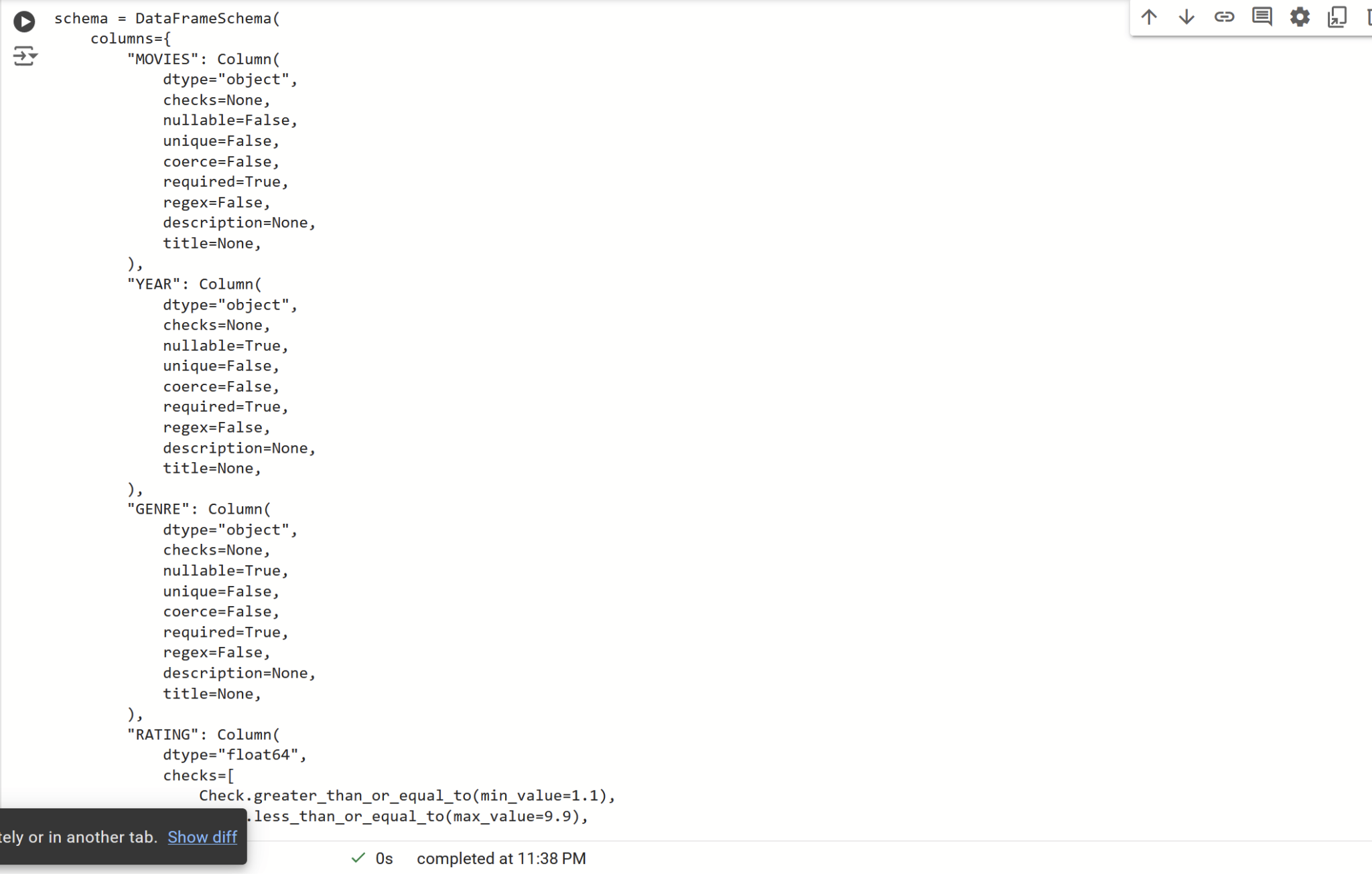
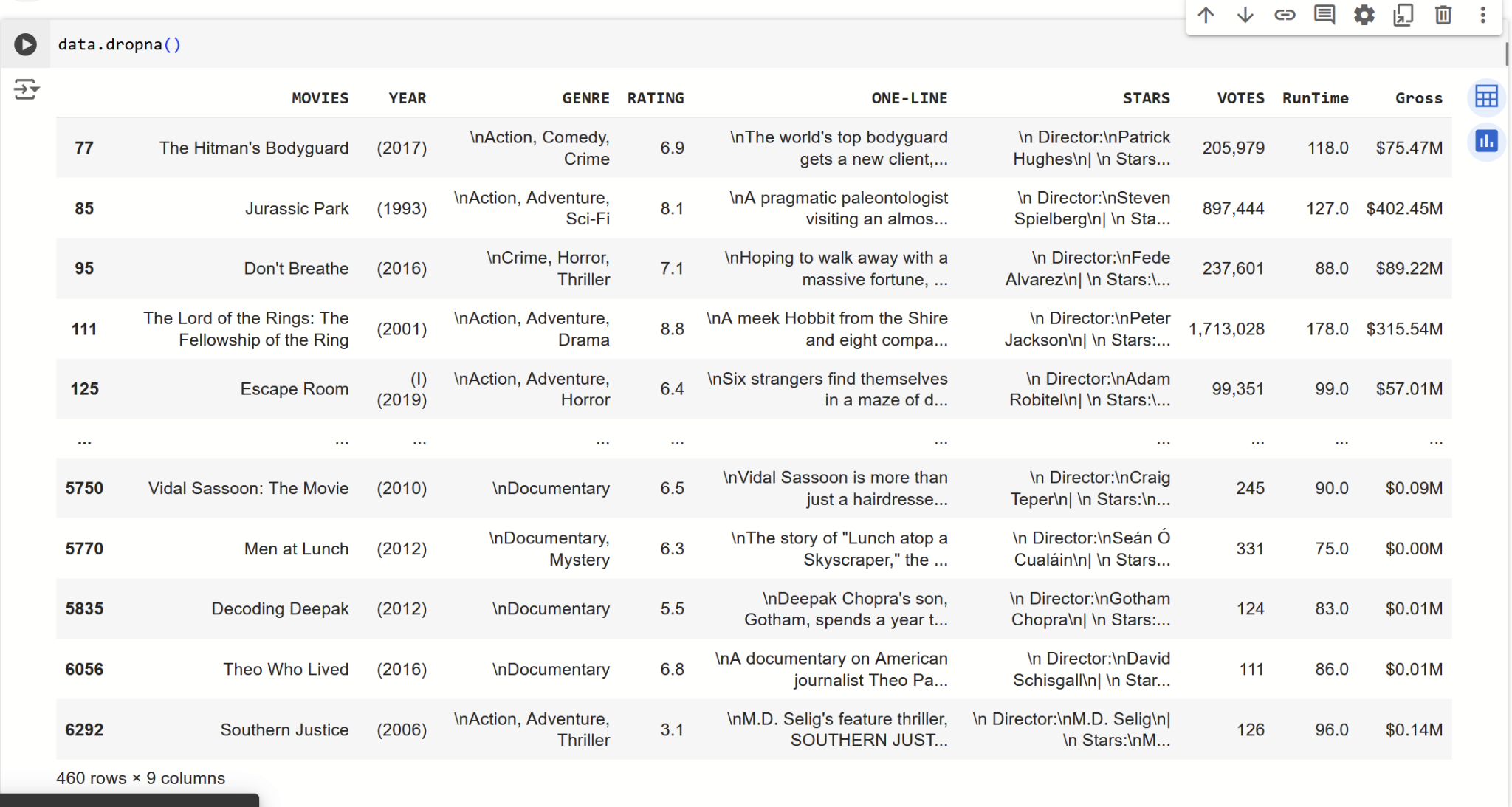
import hashlib

df['Hashed\_ONE-LINE'] = df['ONE-LINE'].apply(lambda x: hashlib.sha256(x.encode()).hexdigest()) # Secure Hash Algorithm 256-bit

df

16)This code applies the cryptographic hash function to the text in the 'ONE-LINE' column of the DataFrame. It converts each text entry into its corresponding hash value using the this function.The resulting hash values are hexadecimal representations of the original texts, and they are stored in a new column called 'Hashed\_ONE-LINE' in the DataFrame

# Results



# Conclusion

1. Cleaning involved handling duplicates, missing values, and data integrity issues to ensure the quality of the dataset.
2. Validation used the Great Expectations library to enforce data integrity rules and expectations, ensuring the dataset's consistency and reliability.
3. Security and Privacy measures included data anonymization, value randomization, and cryptographic hashing to protect sensitive information while maintaining data utility and integrity.

These functions and techniques collectively ensure data quality, integrity, and privacy compliance, fostering trust in the dataset's reliability and confidentiality.

Note: You didn’t need to insert screenshots for the whole code but use it when it needed to clarify an issue or a result.