MIT | Academy of Engineering

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

EDS Assignment

Name: - Aayush Nawale

Class: - ET2

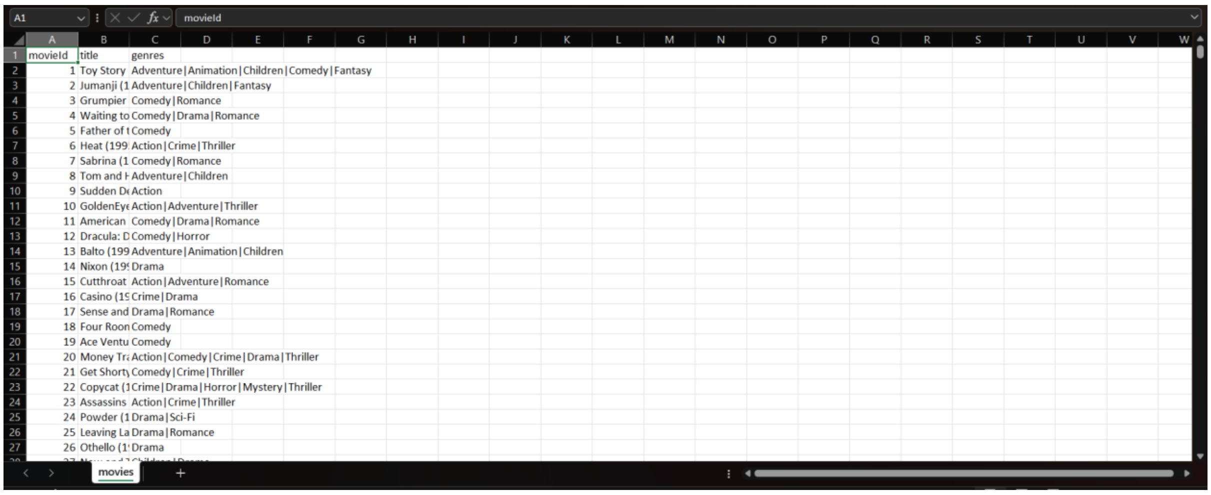
Roll No.: - ET2-34

PRN: - 202401070164

Google Collab Link :-

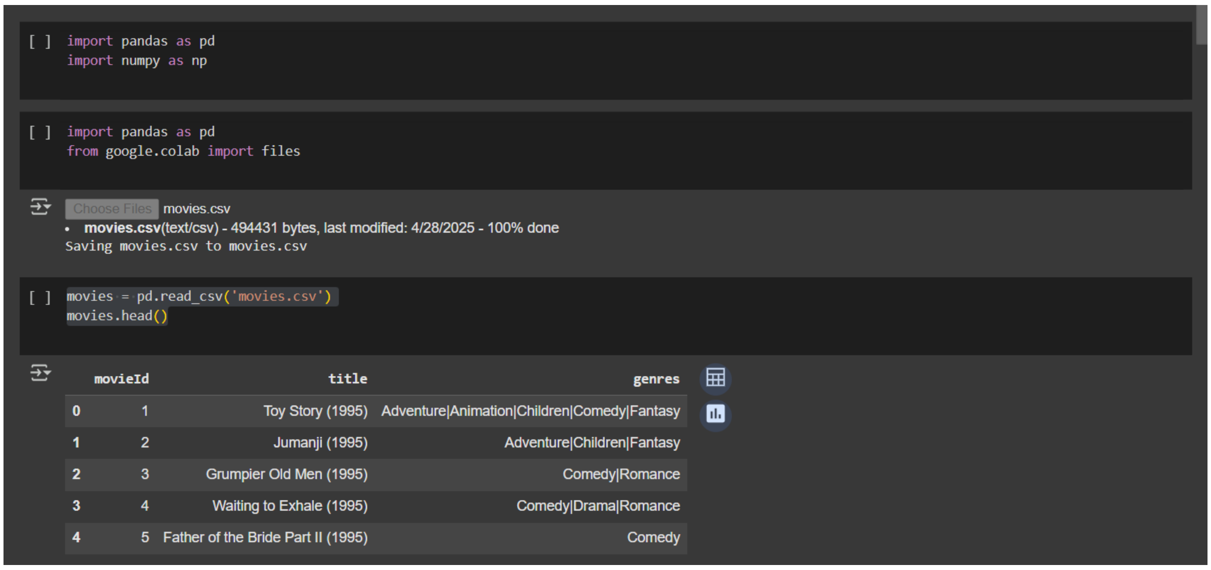
https://colab.research.google.com/drive/1H6jORniSMIJpCbl01K4Di4RleBmAh60b?usp=sharing

Dataset :- Movie Lens



movieId	title	genres
1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1	Adventure|Children|Fantasy
3	Grumpier	Comedy|Romance
4	Waiting to	Comedy|Drama|Romance
5	Father of t	Comedy
6	Heat (199	Action|Crime|Thriller
7	Sabrina (1	Comedy|Romance
8	Tom and H	Adventure|Children
9	Sudden De	Action
10	GoldenEye	Action|Adventure|Thriller
11	American	Comedy|Drama|Romance
12	Dracula: D	Comedy|Horror
13	Balto (199	Adventure|Animation|Children
14	Nixon (19	Drama
15	Cutthroat	Action|Adventure|Romance
16	Casino (19	Crime|Drama
17	Sense and	Drama|Romance
18	Four Roon	Comedy
19	Ace Ventu	Comedy
20	Money Tra	Action|Comedy|Crime|Drama|Thriller
21	Get Shorty	Comedy|Crime|Thriller
22	Copycat (1	Crime|Drama|Horror|Mystery|Thriller
23	Assassins	Action|Crime|Thriller
24	Powder (1	Drama|Sci-Fi
25	Leaving La	Drama|Romance
26	Othello (1	Drama

Importing Pandas and num.py In python



```
[ ] import pandas as pd
    import numpy as np
```

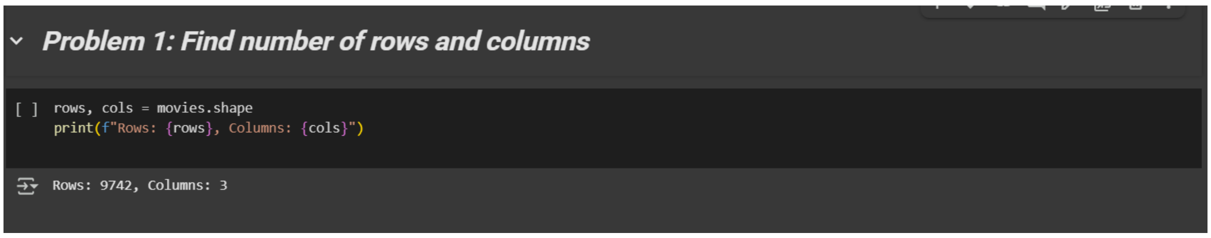
```
[ ] import pandas as pd
    from google.colab import files
```

```
Choose Files movies.csv
• movies.csv(text/csv) - 494431 bytes, last modified: 4/28/2025 - 100% done
Saving movies.csv to movies.csv
```

```
[ ] movies = pd.read_csv('movies.csv')
    movies.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

1. ***Find number of rows and columns***

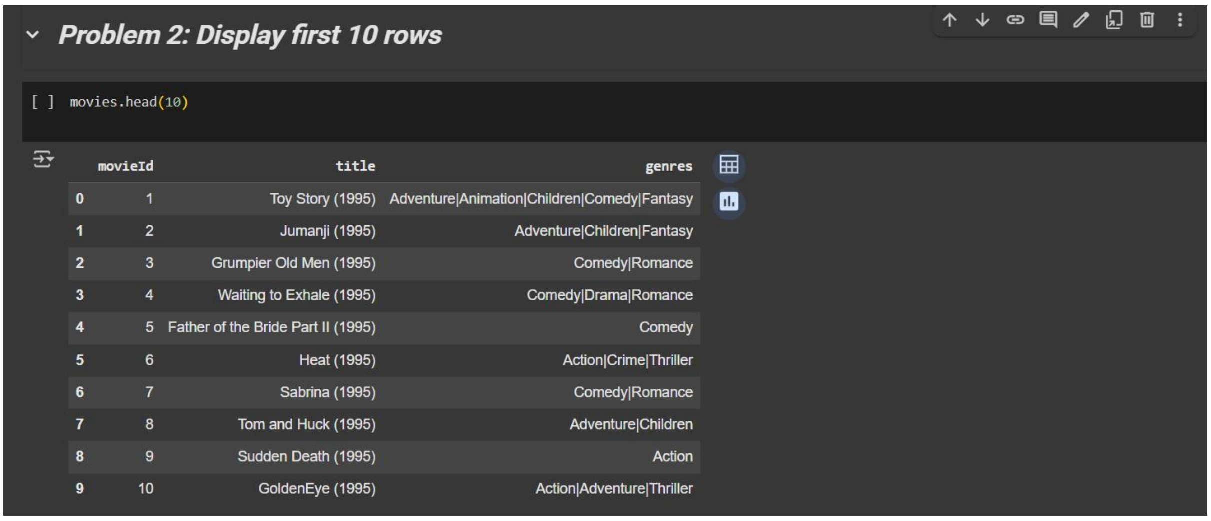


Problem 1: Find number of rows and columns

```
[ ] rows, cols = movies.shape
    print(f"Rows: {rows}, Columns: {cols}")
```

```
Rows: 9742, Columns: 3
```

2. ***Display first 10 rows***

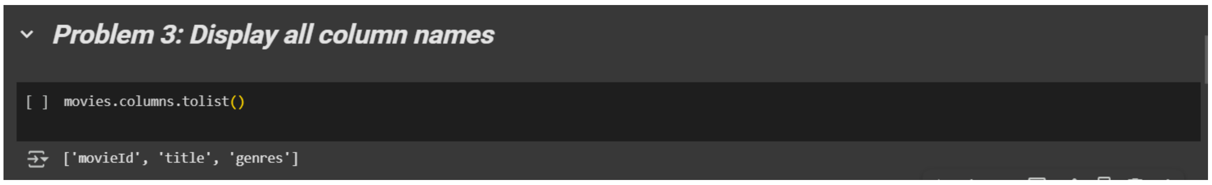


Problem 2: Display first 10 rows

```
[ ] movies.head(10)
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller

3. ***Display all column names***

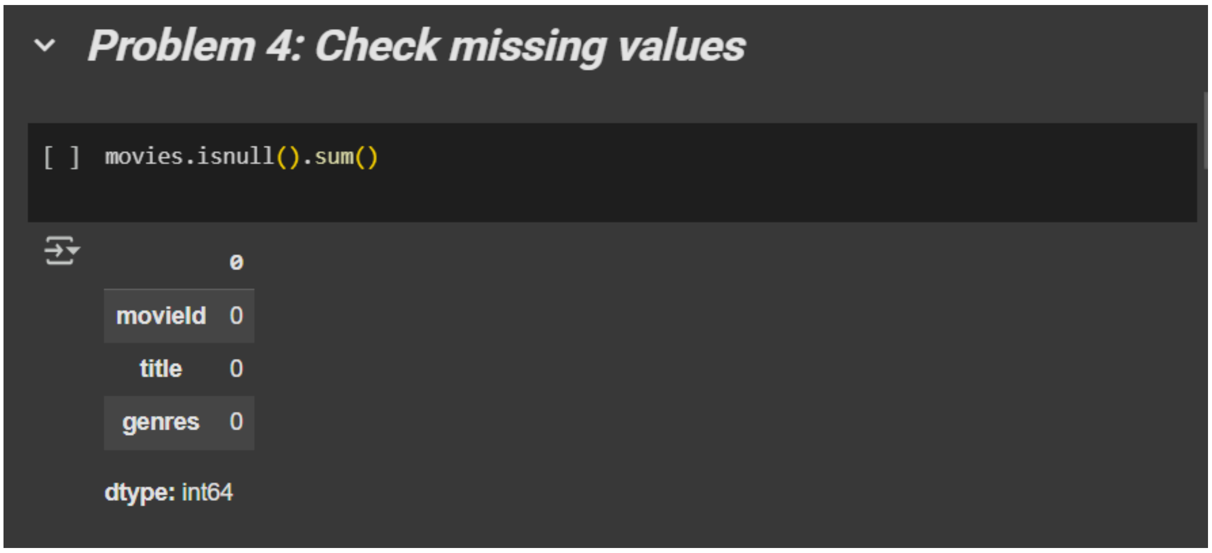


Problem 3: Display all column names

```
[ ] movies.columns.tolist()

['movieId', 'title', 'genres']
```

4. ***Check missing values***



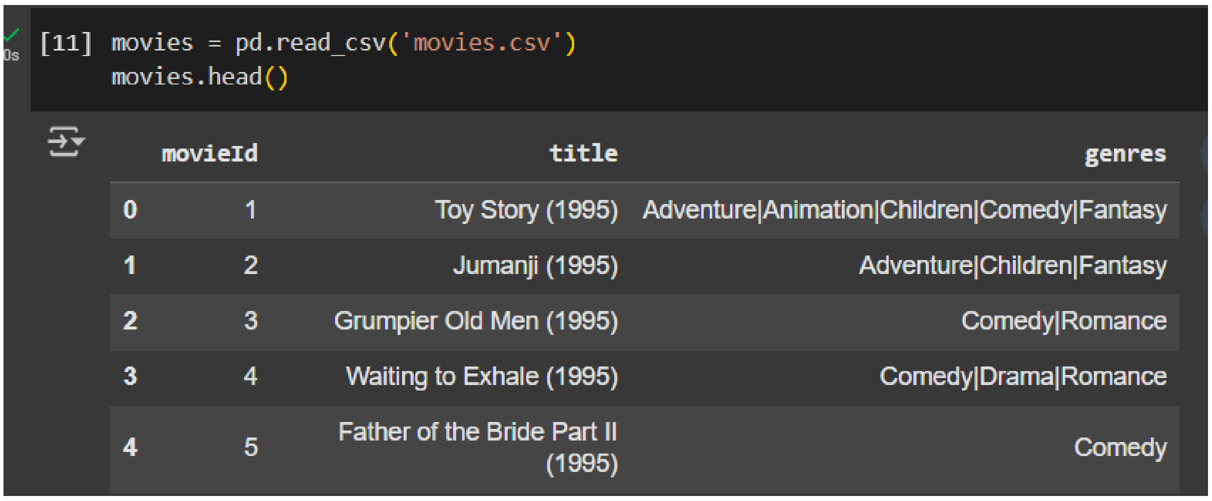
Problem 4: Check missing values

```
[ ] movies.isnull().sum()
```

	0
movieId	0
title	0
genres	0

```
dtype: int64
```

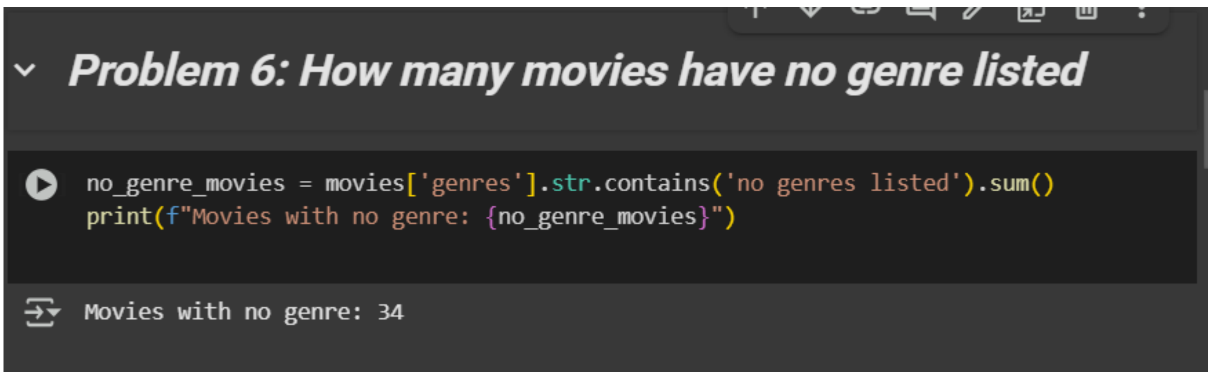
5. ***Find unique genres***



```
[11] movies = pd.read_csv('movies.csv')
     movies.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

6. ***How many movies have no genre listed***

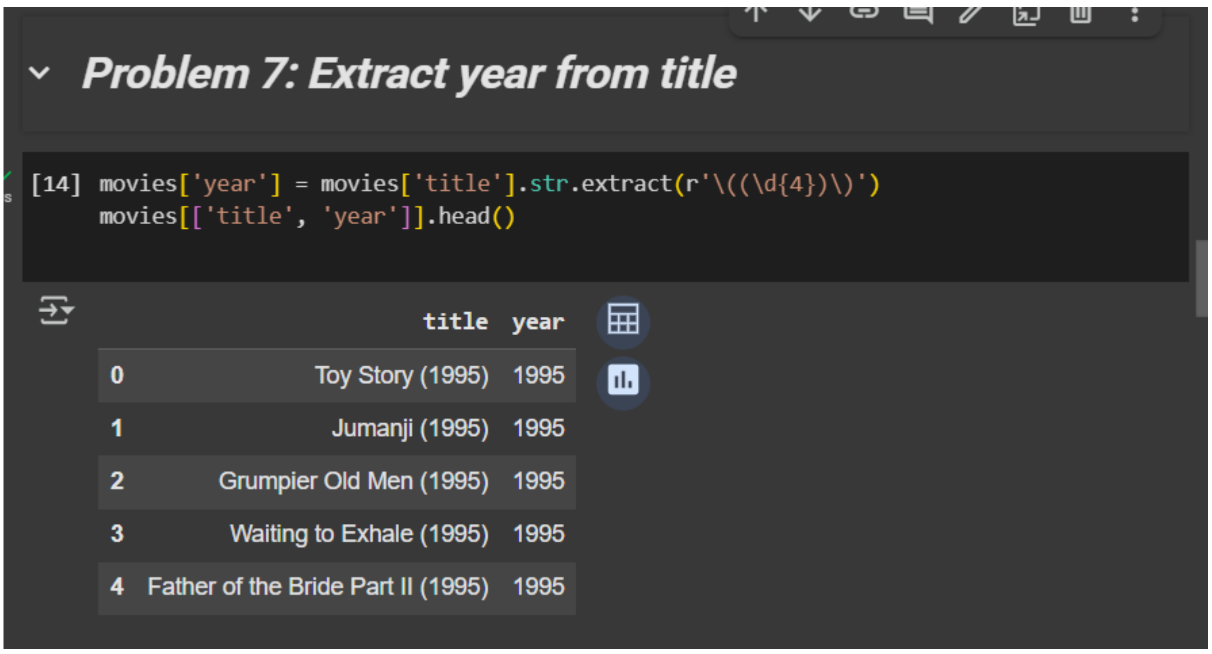


Problem 6: How many movies have no genre listed

```
no_genre_movies = movies['genres'].str.contains('no genres listed').sum()
print(f"Movies with no genre: {no_genre_movies}")
```

```
Movies with no genre: 34
```

7. ***Extract year from title***

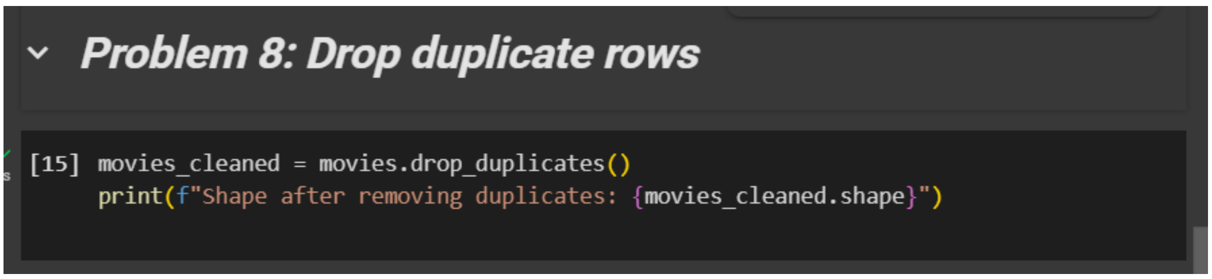


Problem 7: Extract year from title

```
[14] movies['year'] = movies['title'].str.extract(r'\((\d{4})\)')
     movies[['title', 'year']].head()
```

	title	year
0	Toy Story (1995)	1995
1	Jumanji (1995)	1995
2	Grumpier Old Men (1995)	1995
3	Waiting to Exhale (1995)	1995
4	Father of the Bride Part II (1995)	1995

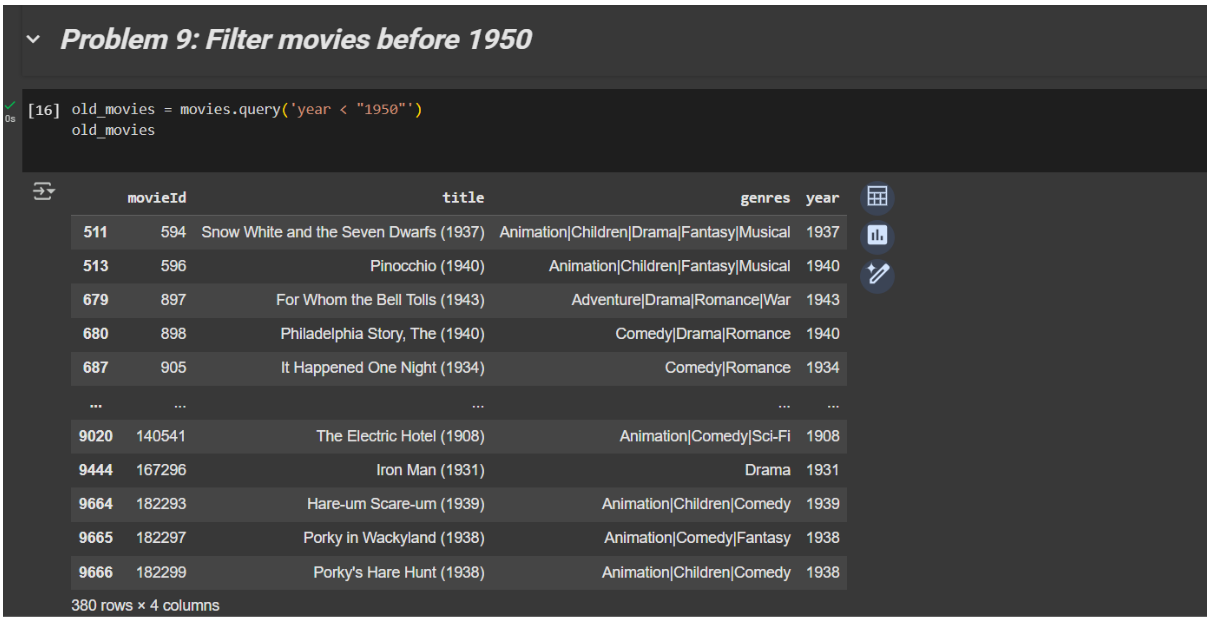
8. ***Drop duplicate rows***



Problem 8: Drop duplicate rows

```
[15] movies_cleaned = movies.drop_duplicates()
     print(f"Shape after removing duplicates: {movies_cleaned.shape}")
```

9. ***Filter movies before 1950***



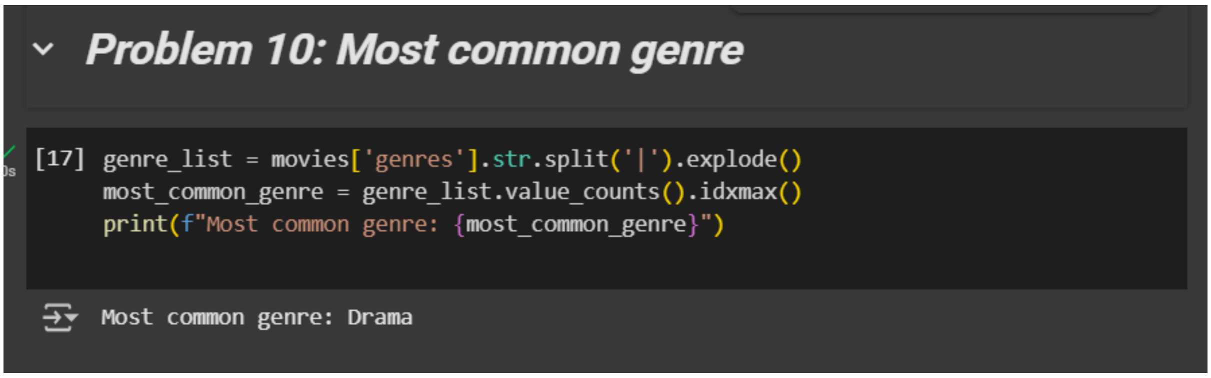
Problem 9: Filter movies before 1950

```
[16] old_movies = movies.query('year < "1950"')
     old_movies
```

	movieId	title	genres	year
511	594	Snow White and the Seven Dwarfs (1937)	Animation|Children|Drama|Fantasy|Musical	1937
513	596	Pinocchio (1940)	Animation|Children|Fantasy|Musical	1940
679	897	For Whom the Bell Tolls (1943)	Adventure|Drama|Romance|War	1943
680	898	Philadelphia Story, The (1940)	Comedy|Drama|Romance	1940
687	905	It Happened One Night (1934)	Comedy|Romance	1934
...	...	...	...	...
9020	140541	The Electric Hotel (1908)	Animation|Comedy|Sci-Fi	1908
9444	167296	Iron Man (1931)	Drama	1931
9664	182293	Hare-um Scare-um (1939)	Animation|Children|Comedy	1939
9665	182297	Porky in Wackyland (1938)	Animation|Comedy|Fantasy	1938
9666	182299	Porky's Hare Hunt (1938)	Animation|Children|Comedy	1938

380 rows × 4 columns

10. ***Most common genre***

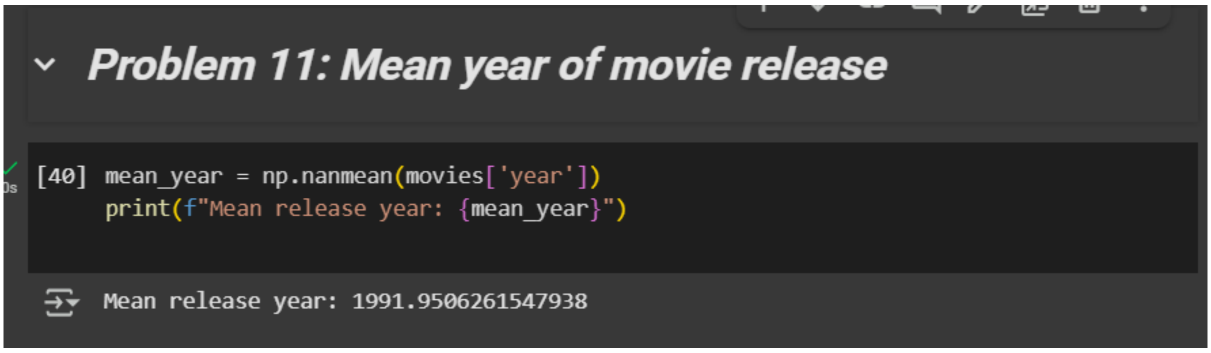


Problem 10: Most common genre

```
[17] genre_list = movies['genres'].str.split('|').explode()
     most_common_genre = genre_list.value_counts().idxmax()
     print(f"Most common genre: {most_common_genre}")
```

```
Most common genre: Drama
```

11. *Mean year of movie release*

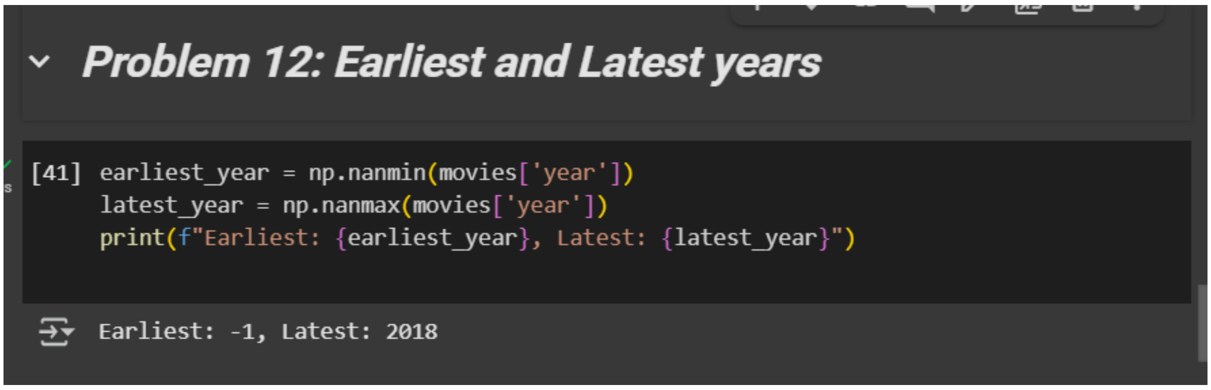


Problem 11: Mean year of movie release

```
[40] mean_year = np.nanmean(movies['year'])
     print(f"Mean release year: {mean_year}")
```

```
Mean release year: 1991.9506261547938
```

12. *Earliest and Latest years*

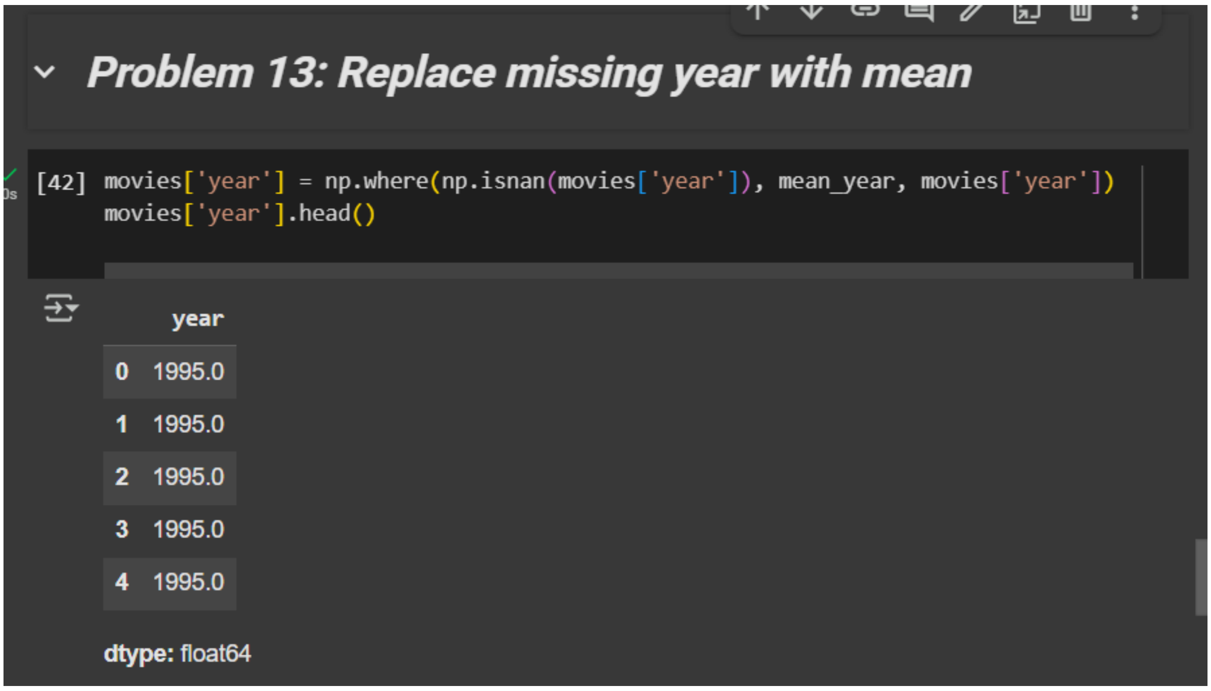


Problem 12: Earliest and Latest years

```
[41] earliest_year = np.nanmin(movies['year'])
     latest_year = np.nanmax(movies['year'])
     print(f"Earliest: {earliest_year}, Latest: {latest_year}")
```

```
Earliest: -1, Latest: 2018
```

13. *Replace missing year with mean*



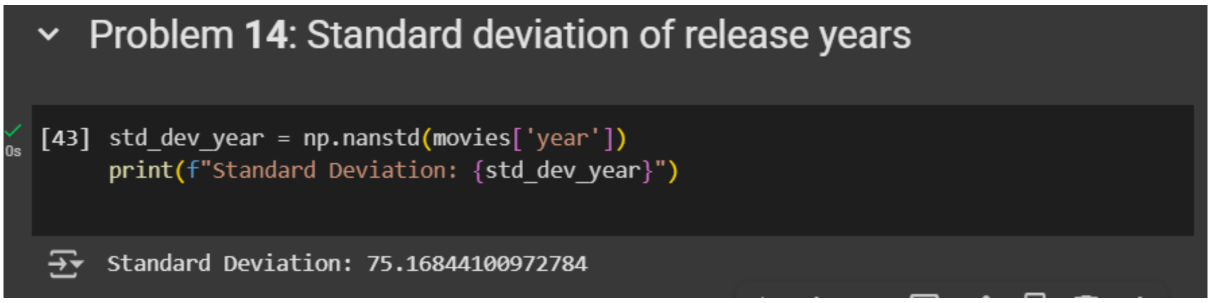
Problem 13: Replace missing year with mean

```
[42] movies['year'] = np.where(np.isnan(movies['year']), mean_year, movies['year'])
     movies['year'].head()
```

	year
0	1995.0
1	1995.0
2	1995.0
3	1995.0
4	1995.0

dtype: float64

14. Standard deviation of release years

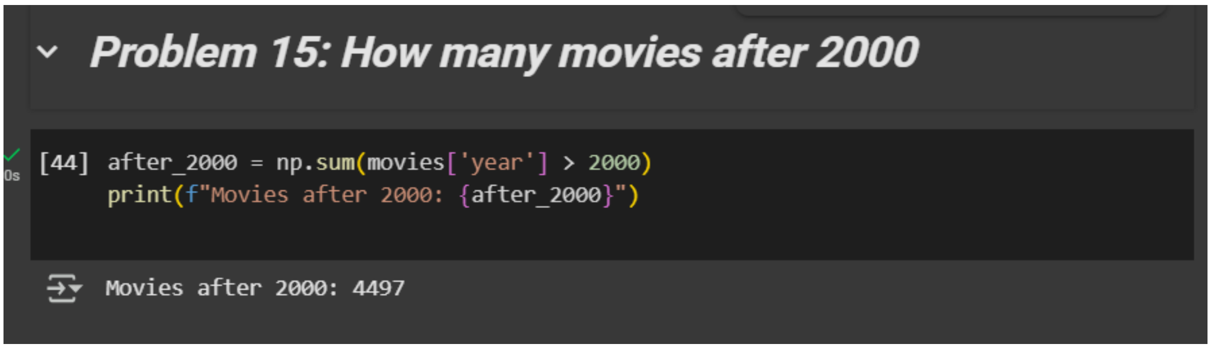


Problem 14: Standard deviation of release years

```
[43] std_dev_year = np.nanstd(movies['year'])
     print(f"Standard Deviation: {std_dev_year}")
```

```
Standard Deviation: 75.16844100972784
```

15. *How many movies after 2000*

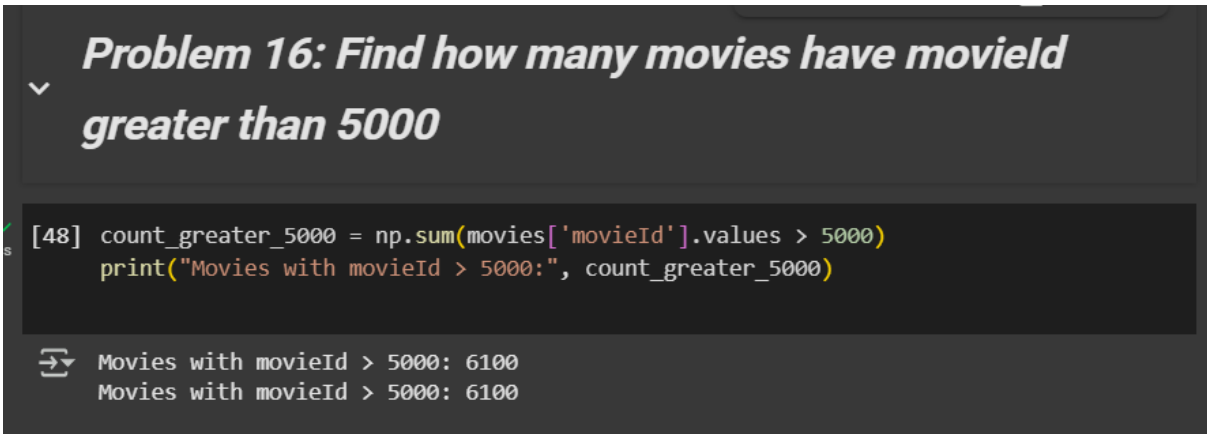


Problem 15: How many movies after 2000

```
[44] after_2000 = np.sum(movies['year'] > 2000)
     print(f"Movies after 2000: {after_2000}")
```

```
Movies after 2000: 4497
```

16. *Find how many movies have movieId greater than 5000*

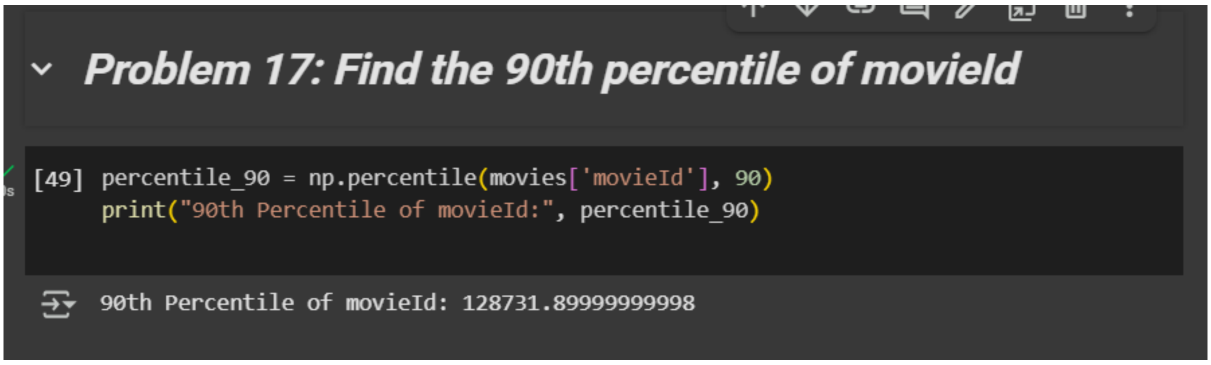


Problem 16: Find how many movies have movieId greater than 5000

```
[48] count_greater_5000 = np.sum(movies['movieId'].values > 5000)
     print("Movies with movieId > 5000:", count_greater_5000)
```

```
Movies with movieId > 5000: 6100
Movies with movieId > 5000: 6100
```

17. *Find the 90th percentile of movieId*

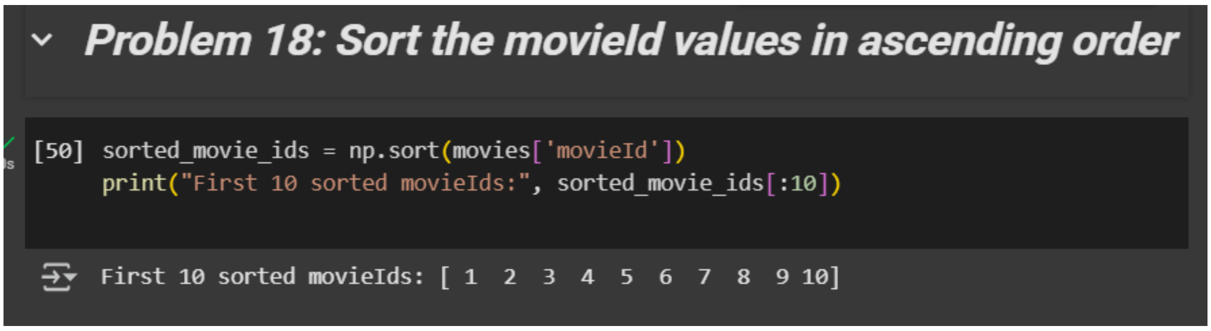


Problem 17: Find the 90th percentile of movieId

```
[49] percentile_90 = np.percentile(movies['movieId'], 90)
     print("90th Percentile of movieId:", percentile_90)
```

```
90th Percentile of movieId: 128731.89999999998
```

18. *Sort the movieId values in ascending order*

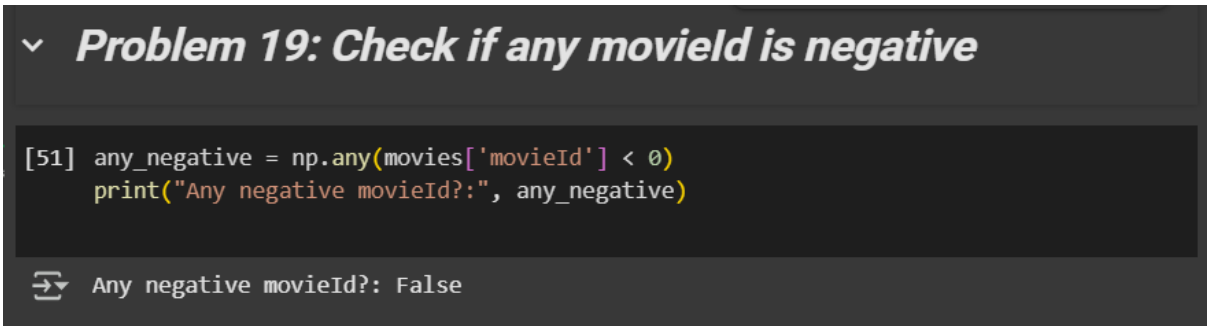


Problem 18: Sort the movieId values in ascending order

```
[50] sorted_movie_ids = np.sort(movies['movieId'])
     print("First 10 sorted movieIds:", sorted_movie_ids[:10])
```

```
First 10 sorted movieIds: [ 1  2  3  4  5  6  7  8  9 10]
```

19. *Check if any movieId is negative*

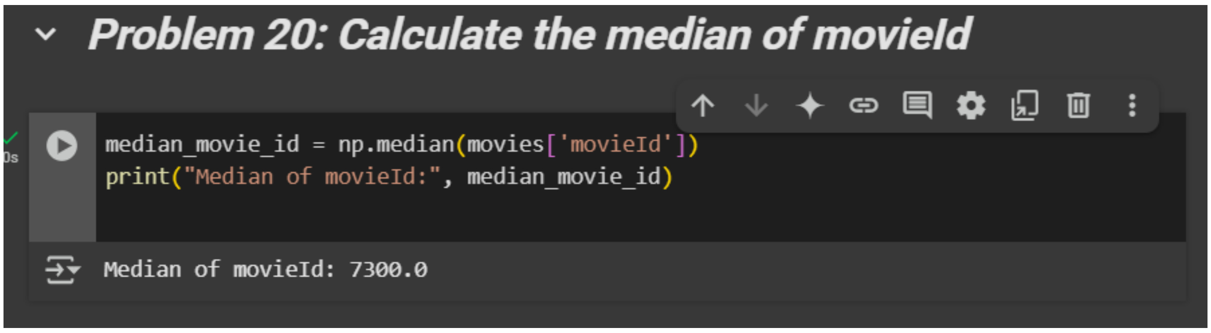


Problem 19: Check if any movieId is negative

```
[51] any_negative = np.any(movies['movieId'] < 0)
     print("Any negative movieId?:", any_negative)
```

```
Any negative movieId?: False
```

20. *Calculate the median of movieId*



Problem 20: Calculate the median of movieId

```
median_movie_id = np.median(movies['movieId'])
print("Median of movieId:", median_movie_id)
```

```
Median of movieId: 7300.0
```