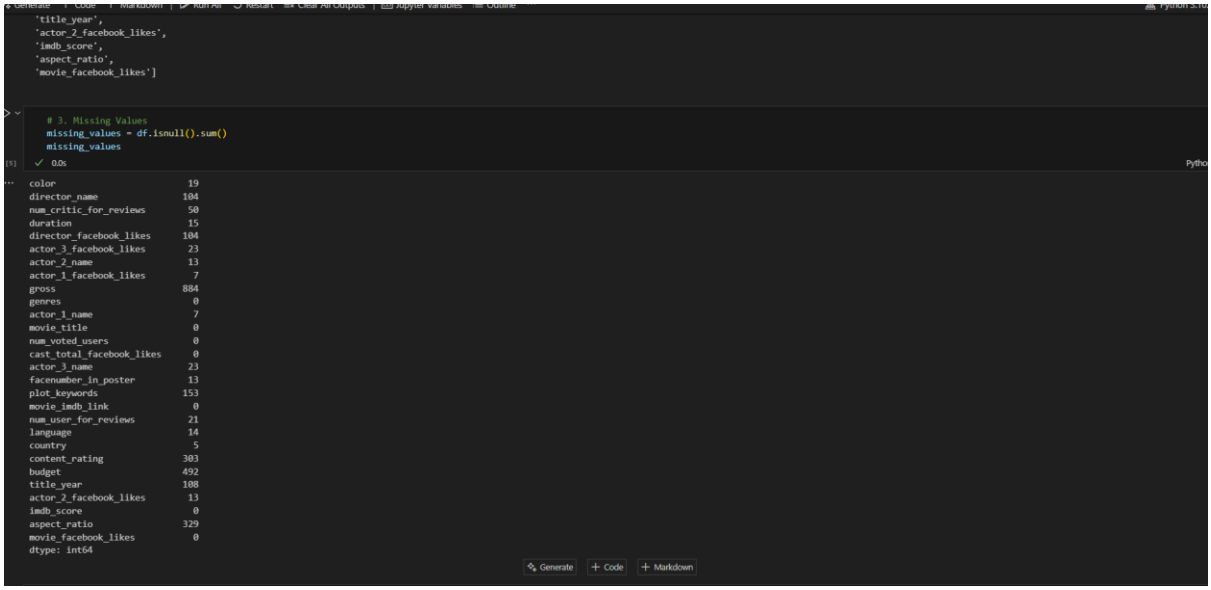
Name: Mohit Sharma

Roll No: CS7-31

Batch: CS72

IMDB Dataset



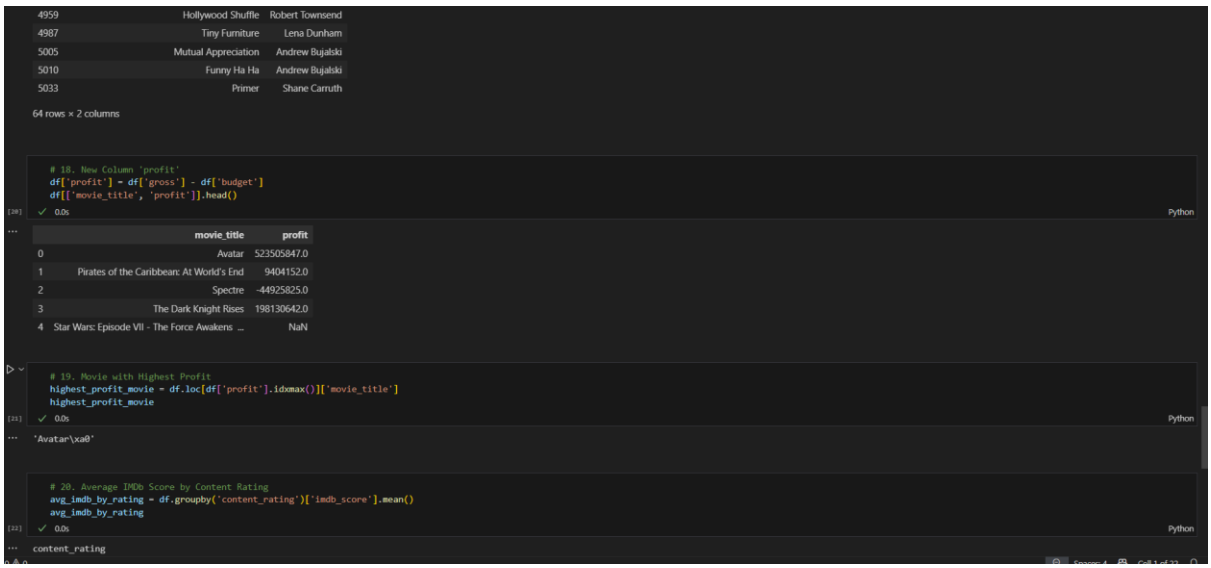
```
'title_year',
'actor_2_facebook_likes',
'imdb_score',
'aspect_ratio',
'movie_facebook_likes']
```

```
# 3. Missing Values
missing_values = df.isnull().sum()
missing_values
```

✓ 0.0s

```
color                      19
director_name             104
num_critic_for_reviews     50
duration                   15
director_facebook_likes   104
actor_3_facebook_likes     23
actor_2_name               13
actor_1_facebook_likes      7
gross                     884
genres                      0
actor_1_name                7
movie_title                 0
num_voted_users             0
cast_total_facebook_likes   0
actor_3_name               23
facenumber_in_poster       13
plot_keywords             153
movie_imdb_link             0
num_user_for_reviews       21
language                   14
country                     5
content_rating            303
budget                    492
title_year                108
actor_2_facebook_likes     13
imdb_score                  0
aspect_ratio              329
movie_facebook_likes        0
dtype: int64
```

Generate + Code + Markdown



4959	Hollywood Shuffle	Robert Townsend
4987	Tiny Furniture	Lena Dunham
5005	Mutual Appreciation	Andrew Bujalski
5010	Funny Ha Ha	Andrew Bujalski
5033	Primer	Shane Carruth

64 rows × 2 columns

```
# 18. New Column 'profit'
df['profit'] = df['gross'] - df['budget']
df[['movie_title', 'profit']].head()
```

✓ 0.0s

	movie_title	profit
0	Avatar	523505847.0
1	Pirates of the Caribbean: At World's End	9404152.0
2	Spectre	-44925825.0
3	The Dark Knight Rises	198130642.0
4	Star Wars: Episode VII - The Force Awakens ...	NaN

```
# 19. Movie with Highest Profit
highest_profit_movie = df.loc[df['profit'].idxmax()]['movie_title']
highest_profit_movie
```

✓ 0.0s

```
'Avatar\xa0'
```

```
# 20. Average IMDb Score by Content Rating
avg_imdb_by_rating = df.groupby('content_rating')['imdb_score'].mean()
avg_imdb_by_rating
```

✓ 0.0s

```
content_rating
```

```python
# Importing required libraries
import pandas as pd
import numpy as np
```

[1] ✓ 1.8s Python

```python
# Reading the dataset
df = pd.read_csv('movie_metadata.csv')
```

[2] ✓ 0.0s Python

```python
# 1. Rows and Columns
rows, cols = df.shape
rows, cols
```

[3] ✓ 0.0s Python

```
(5043, 28)
```

```python
# 2. Column Names
columns = df.columns.tolist()
columns
```

[4] ✓ 0.0s Python

```
['color',
 'director_name',
 'num_critic_for_reviews',
 'duration',
 'director_facebook_likes',
 'actor_3_facebook_likes',
 'actor_2_name',
 'actor_1_facebook_likes',
 'gross',
 'genres',
 'actor_1_name',
 'movie_title',
 'num_voted_users',
 'cast_total_facebook_likes',
 'actor_3_name',
 'facenumber_in_poster',
```

```python
# 20. Average IMDb Score by Content Rating
avg_imdb_by_rating = df.groupby('content_rating')['imdb_score'].mean()
avg_imdb_by_rating
```

[5] ✓ 0.0s Python

```
content_rating
Approved     7.325455
G            6.529464
GP           6.916667
M            6.840000
NC-17        6.542857
Not Rated    6.631034
PG           6.294437
PG-13        6.257495
Passed       7.166667
R            6.527101
TV-14        7.250000
TV-G         6.920000
TV-MA        8.250000
TV-PG        7.353846
TV-Y         7.400000
TV-Y7        7.200000
Unrated      6.920968
X            6.500000
Name: imdb_score, dtype: float64
```

```python
# 4. Unique Genres
if 'genres' in df.columns:
    unique_genres = df['genres'].nunique()
unique_genres
```

[6] ✓ 0.0s Python

```
914
```

```python
# 5. Mean IMDb Score
mean_imdb = df['imdb_score'].mean()
mean_imdb
```

[7] ✓ 0.0s Python

```
6.442137616498116
```

```python
# 6. Movie with Highest IMDb Score
highest_imdb_movie = df.loc[df['imdb_score'].idxmax()]['movie_title']
highest_imdb_movie
```

[8] ✓ 0.0s Python

```
'Towering Inferno\xa0'
```

```python
# 7. Movies Directed by Christopher Nolan
nolan_movies = df[df['director_name'] == 'Christopher Nolan']['movie_title'].tolist()
nolan_movies
```

[9] ✓ 0.0s Python

```
['The Dark Knight Rises\xa0',
 'The Dark Knight\xa0',
 'Interstellar\xa0',
 'Inception\xa0',
 'Batman Begins\xa0',
 'Insomnia\xa0',
 'The Prestige\xa0',
 'Memento\xa0']
```

```python
# 14. Average Movie Duration
avg_duration = df['duration'].mean()
avg_duration
```

[16] ✓ 0.0s Python

```
107.201073985802
```

```python
# 15. Movies with Missing IMDb Score
missing_imdb_movies = df[df['imdb_score'].isnull()]['movie_title'].tolist()
missing_imdb_movies
```

[17] ✓ 0.0s Python

```
[]
```

```python
# 16. Total Gross Revenue
total_gross = df['gross'].sum()
total_gross
```

[18] ✓ 0.0s Python

```
201580106904.0
```

```python
# 17. Movies where Actor and Director are Same
same_actor_director = df[df['actor_1_name'] == df['director_name']][['movie_title', 'director_name']]
same_actor_director
```

[19] ✓ 0.0s Python

	movie_title	director_name
4	Star Wars: Episode VII - The Force Awakens ...	Doug Walker
664	Space Cowboys	Clint Eastwood
973	Absolute Power	Clint Eastwood
1004	Blood Work	Clint Eastwood
1178	Animal Kingdom: Let's go Ape	Jamel Debbouze
...	...	...
4959	Hollywood Shuffle	Robert Townsend

```python
# 10. Top 5 Actors Appearing Most
top_actors = df['actor_1_name'].value_counts().head(5)
top_actors
```

[12] ✓ 0.0s Python

```
actor_1_name
Robert De Niro    49
Johnny Depp       41
Nicolas Cage      33
J.K. Simmons      31
Bruce Willis      30
Name: count, dtype: int64
```

```python
# 11. Correlation between Budget and IMDb Score
correlation = df['budget'].corr(df['imdb_score'])
correlation
```

[13] ✓ 0.0s Python

```
0.030687727112957558
```

```python
# 12. Most Common Language
most_common_language = df['language'].mode()[0]
most_common_language
```

[14] ✓ 0.0s Python

```
'English'
```

```python
# 13. Movies with Budget > $100 Million
big_budget_movies = df[df['budget'] > 100_000_000]['movie_title'].tolist()
big_budget_movies
```

[45] ✓ 0.0s Python

```
['Avatar\xa0',
 "Pirates of the Caribbean: At World's End\xa0",
 'Spectre\xa0',
 'The Dark Knight Rises\xa0',
 'John Carter\xa0',
 'Spider-Man 3\xa0',
 'Tangled\xa0',
 'Avengers: Age of Ultron\xa0',
 'Harry Potter and the Half-Blood Prince\xa0',
 'Batman v Superman: Dawn of Justice\xa0',
 'Superman Returns\xa0',
 'Quantum of Solace\xa0',
 "Pirates of the Caribbean: Dead Man's Chest\xa0",
 'The Lone Ranger\xa0',
 'Man of Steel\xa0',
 'The Chronicles of Narnia: Prince Caspian\xa0',
 'The Avengers\xa0',
 'Pirates of the Caribbean: On Stranger Tides\xa0',
 'Men in Black 3\xa0',
 'The Hobbit: The Battle of the Five Armies\xa0',
 'The Amazing Spider-Man\xa0',
 'Robin Hood\xa0',
 'The Hobbit: The Desolation of Smaug\xa0',
 'The Golden Compass\xa0',
 'King Kong\xa0',
 ...
 'Vaalu\xa0',
 'Godzilla 2000\xa0',
 'King Kong\xa0',
 'Home\xa0',
 'Oz the Great and Powerful\xa0']
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
# 8. Movies Released Each Year
movies_per_year = df['title_year'].value_counts().sort_index()
movies_per_year
```

[30] ✓ 0.0s Python

```
title_year
1916.0      1
1920.0      1
1925.0      1
1927.0      1
1929.0      2
          ...
2012.0    221
2013.0    237
2014.0    252
2015.0    226
2016.0    106
Name: count, Length: 91, dtype: int64
```

```python
# 9. Average Budget per Decade
df['decade'] = (df['title_year'] // 10) * 10
average_budget_decade = df.groupby('decade')['budget'].mean()
average_budget_decade
```

[31] ✓ 0.0s Python

```
decade
1910.0    3.859070e+05
1920.0    1.681000e+06
1930.0    1.638910e+06
1940.0    2.388827e+06
1950.0    2.970456e+06
1960.0    5.415441e+06
1970.0    7.913097e+06
1980.0    1.822726e+07
1990.0    3.741215e+07
2000.0    4.761055e+07
2010.0    4.075647e+07
Name: budget, dtype: float64
```