```
from google.colab import files

uploaded = files.upload()
```

Choose Files 5 files

- **movies.csv**(text/csv) - 494431 bytes, last modified: 4/28/2025 - 100% done
- **ratings.csv**(text/csv) - 2483723 bytes, last modified: 4/28/2025 - 100% done
- **README.txt**(text/plain) - 8342 bytes, last modified: 4/28/2025 - 100% done
- **tags.csv**(text/csv) - 118660 bytes, last modified: 4/28/2025 - 100% done
- **links.csv**(text/csv) - 197979 bytes, last modified: 4/28/2025 - 100% done

```
Saving movies.csv to movies (3).csv
Saving ratings.csv to ratings (3).csv
Saving README.txt to README (3).txt
Saving tags.csv to tags (3).csv
Saving links.csv to links (3).csv
```

```
import pandas as pd
import numpy as np
movies = pd.read_csv('movies.csv')
ratings = pd.read_csv('ratings.csv')
tags = pd.read_csv('tags.csv')
links = pd.read_csv('links.csv')
```

1. Find the total number of movies

```
total_movies = movies.shape[0]
print(f"Total movies: {total_movies}")
```

```
Total movies: 9742
```

2. Identify the movie with the maximum number of ratings

```
most_rated_movie_id = ratings['movieId'].value_counts().idxmax()
most_rated_movie = movies[movies['movieId'] == most_rated_movie_id]['title'].values[0]
print(f"Most rated movie: {most_rated_movie}")
```

```
Most rated movie: Forrest Gump (1994)
```

3. Calculate the average rating given to all movies

```
average_rating = ratings['rating'].mean()
print(f"Average rating: {average_rating:.2f}")
```

```
Average rating: 3.50
```

4. Find the standard deviation of all ratings

```
rating_std = ratings['rating'].std()
print(f"Standard deviation of ratings: {rating_std:.2f}")
```

```
Standard deviation of ratings: 1.04
```

5. Top 10 movies with highest average ratings (with at least 50 ratings)

```
movie_ratings = ratings.groupby('movieId').agg({'rating': ['mean', 'count']})
movie_ratings.columns = ['avg_rating', 'rating_count']
filtered_movies = movie_ratings[movie_ratings['rating_count'] >= 50]
top_10_movies = filtered_movies.sort_values('avg_rating', ascending=False).head(10)
top_10_movies = top_10_movies.merge(movies, on='movieId')
print(top_10_movies[['title', 'avg_rating']])
```

```
                                               title  avg_rating
0                   Shawshank Redemption, The (1994)    4.429022
1                              Godfather, The (1972)    4.289062
2                                  Fight Club (1999)    4.272936
3                              Cool Hand Luke (1967)    4.271930
4  Dr. Strangelove or: How I Learned to Stop Worr...    4.268041
5                                 Rear Window (1954)    4.261905
6                     Godfather: Part II, The (1974)    4.259690
7                                Departed, The (2006)    4.252336
8                                  Goodfellas (1990)    4.250000
9                                  Casablanca (1942)    4.240000
```

6. Identify the number of unique genres listed across all movies

```
# Split the genres
unique_genres = movies['genres'].str.split('|').explode().nunique()
print(f"Total unique genres: {unique_genres}")
```

```
Total unique genres: 20
```

7. Find the genre that appears most frequently among all movies

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

```
Most common genre: Drama
```

8. Calculate the number of movies released each year

```
# Extract year from the title
movies['year'] = movies['title'].str.extract(r'\((\d{4})\)')
movies_per_year = movies['year'].value_counts().sort_index()
print(movies_per_year)
```

```
year
1902      1
1903      1
1908      1
1915      1
1916      4
       ...
2014    278
2015    274
2016    218
2017    147
2018     41
Name: count, Length: 106, dtype: int64
```

9. Find which user has rated the most movies

```
top_user = ratings['userId'].value_counts().idxmax()
num_movies_rated = ratings['userId'].value_counts().max()
print(f"User {top_user} rated the most movies: {num_movies_rated} movies")
```

```
User 414 rated the most movies: 2698 movies
```

10. Compute the average rating per genre (consider multi-genre movies)

```
# Expand genres
movie_genres = movies[['movieId', 'genres']].copy()
movie_genres = movie_genres.assign(genres=movie_genres['genres'].str.split('|')).explode('genres')

# Merge with ratings
ratings_with_genre = ratings.merge(movie_genres, on='movieId')

# Group by genre and calculate average rating
avg_rating_per_genre = ratings_with_genre.groupby('genres')['rating'].mean().sort_values(ascending=False)
```

```
print(avg_rating_per_genre)
```

```
genres
Film-Noir             3.920115
War                   3.808294
Documentary           3.797785
Crime                 3.658294
Drama                 3.656184
Mystery               3.632460
Animation             3.629937
IMAX                  3.618335
Western               3.583938
Musical               3.563678
Adventure             3.508609
Romance               3.506511
Thriller              3.493706
Fantasy               3.491001
(no genres listed)    3.489362
Sci-Fi                3.455721
Action                3.447984
Children              3.412956
Comedy                3.384721
Horror                3.258195
Name: rating, dtype: float64
```

11. Identify the earliest and latest rating timestamps

```
earliest_rating = pd.to_datetime(ratings['timestamp'], unit='s').min()
latest_rating = pd.to_datetime(ratings['timestamp'], unit='s').max()
print(f"Earliest rating: {earliest_rating}, Latest rating: {latest_rating}")
```

```
Earliest rating: 1996-03-29 18:36:55, Latest rating: 2018-09-24 14:27:30
```

12. Find how many tags were given per user on average

```
tags_per_user_avg = tags.groupby('userId').size().mean()
print(f"Average number of tags per user: {tags_per_user_avg:.2f}")
```

```
Average number of tags per user: 63.50
```

13. Check which movie has the maximum number of different tags

```
# Count unique tags per movie
movie_tag_counts = tags.groupby('movieId')['tag'].nunique()
most_tagged_movie_id = movie_tag_counts.idxmax()
most_tagged_movie = movies[movies['movieId'] == most_tagged_movie_id]['title'].values[0]
print(f"Movie with most different tags: {most_tagged_movie}")
```

```
Movie with most different tags: Pulp Fiction (1994)
```

14. Identify users who have given both a rating and a tag for the same movie

```
# Merge ratings and tags
merged = pd.merge(ratings[['userId', 'movieId']], tags[['userId', 'movieId']], on=['userId', 'movieId'])
users_both = merged['userId'].nunique()
print(f"Users who rated and tagged the same movie: {users_both}")
```

```
Users who rated and tagged the same movie: 54
```

15. Calculate the rating distribution (number of ratings per score)

```
rating_distribution = ratings['rating'].value_counts().sort_index()
print(rating_distribution)
```

```
rating
0.5     1370
```

```
1.0     2811
1.5     1791
2.0     7551
2.5     5550
3.0    20047
3.5    13136
4.0    26818
4.5     8551
5.0    13211
Name: count, dtype: int64
```

16. Find movies with no genre information

```
movies_no_genre = movies[movies['genres'] == '(no genres listed)']
print(f"Movies with no genres listed: {movies_no_genre.shape[0]}")
print(movies_no_genre[['title']])
```

```
Movies with no genres listed: 34
                                                  title
8517                                La cravate (1957)
8684                                  Ben-hur (2016)
8687  Pirates of the Caribbean: Dead Men Tell No Tal...
8782                                Superfast! (2015)
8836                                Let It Be Me (1995)
8902              Trevor Noah: African American (2013)
9033                                Guardians (2016)
9053                                Green Room (2015)
9070                      The Brand New Testament (2015)
9091                                        Hyena Road
9138  The Adventures of Sherlock Holmes and Doctor W...
9178                          A Cosmic Christmas (1977)
9217                                Grease Live (2016)
9248                            Noin 7 veljestä (1968)
9259                                          Paterson
9307                        Ali Wong: Baby Cobra (2016)
9316                   A Midsummer Night's Dream (2016)
9348                        The Forbidden Dance (1990)
9413                              Ethel & Ernest (2016)
9426                                  Whiplash (2013)
9448                                            The OA
9478                                  Lemonade (2016)
9514                                            Cosmos
9515                            Maria Bamford: Old Baby
9518                   Death Note: Desu nôto (2006–2007)
9525                                Generation Iron 2
9534                   T2 3-D: Battle Across Time (1996)
9541               The Godfather Trilogy: 1972-1990 (1992)
9562  The Adventures of Sherlock Holmes and Doctor W...
9573                        The Putin Interviews (2017)
9611                                      Black Mirror
9661  Too Funny to Fail: The Life and Death of The D...
9663  Serving in Silence: The Margarethe Cammermeyer...
9669                      A Christmas Story Live! (2017)
```

17. Analyze the average number of ratings per movie

```
avg_ratings_per_movie = ratings.groupby('movieId').size().mean()
print(f"Average number of ratings per movie: {avg_ratings_per_movie:.2f}")
```

```
Average number of ratings per movie: 10.37
```

18. Find the IMDb link for the highest-rated movie

```
# Find highest rated movie (with sufficient ratings)
movie_ratings = ratings.groupby('movieId').agg({'rating': ['mean', 'count']})
movie_ratings.columns = ['avg_rating', 'rating_count']
filtered_movies = movie_ratings[movie_ratings['rating_count'] >= 50]
highest_rated_movie_id = filtered_movies['avg_rating'].idxmax()

# Get IMDb ID
imdb_id = links[links['movieId'] == highest_rated_movie_id]['imdbId'].values[0]
imdb_link = f"http://www.imdb.com/title/tt{int(imdb_id):07d}/"
print(f"IMDb Link: {imdb_link}")
```

```
IMDb Link: http://www.imdb.com/title/tt0111161/
```

19. List all movies tagged with the word "thriller"

```
thriller_tags = tags[tags['tag'].str.contains('thriller', case=False)]
thriller_movies = thriller_tags.merge(movies, on='movieId')
print(thriller_movies[['title', 'tag']].drop_duplicates())
```

```
                        title                    tag
0     Maze Runner, The (2014)               thriller
1  Usual Suspects, The (1995)               thriller
2               Misery (1990)               thriller
3       Shutter Island (2010)  Psychological Thriller
4            Prisoners (2013)               thriller
5          Pulp Fiction (1994)              thriller
6           Fight Club (1999)  psychological thriller
```

20. Calculate the average rating given by each user and find users with avg rating > 4.5

```
user_avg_ratings = ratings.groupby('userId')['rating'].mean()
high_avg_users = user_avg_ratings[user_avg_ratings > 4.5]
print(f"Users with average rating > 4.5:\n{high_avg_users}")
```

```
Users with average rating > 4.5:
userId
25     4.807692
30     4.735294
43     4.552632
53     5.000000
122    4.546233
171    4.634146
251    4.869565
348    4.672727
371    4.548780
400    4.511628
441    4.522222
452    4.556931
515    4.846154
523    4.693333
Name: rating, dtype: float64
```