```
import pandas as pd
```

```
import pandas as pd

# Specify the file path to the CSV file
file_path = "/content/movies_metadata.csv"

try:
    # Attempt to read the CSV file with pandas
    movies = pd.read_csv(file_path, engine='python')
    
    # If reading is successful, display the first few rows of the DataFrame
    print("Successfully read the CSV file:")
    print(movies.head())
    
except Exception as e:
    # If an error occurs during reading, display the error message
    print("Error:", e)
```

```
movies = pd.read_csv("/content/movies_metadata.csv",
usecols=["id","overview","title","vote_average","vote_count","release_date"])
```

```
movies.head()
```

	id	overview	release_date	title	vote_average	vote_count
0	862	Led by Woody, Andy's toys live happily in his ...	1995-10-30	Toy Story	7.7	5415.0
1	8844	When siblings Judy and Peter discover an encha...	1995-12-15	Jumanji	6.9	2413.0
2	15602	A family wedding reignites the ancient	1995-12-22	Grumpier Old	6.5	92.0

Next steps: Generate code with movies View recommended plots

```
movies.shape
```

```
(45466, 6)
```

```
movies.isnull().sum()
```

```
id                0
overview        954
release_date     87
title             6
vote_average      6
vote_count        6
dtype: int64
```

```
movies = movies.dropna()
```

```
movies.isnull().sum()
```

```
id              0
overview        0
release_date    0
```

```
title           0
vote_average    0
vote_count      0
dtype: int64
```

```
movies.dtypes
```

```
id               object
overview         object
release_date     object
title            object
vote_average    float64
vote_count      float64
dtype: object
```

```
movies.duplicated().sum()
```

```
28
```

```
movies = movies.drop_duplicates()
```

```
movies = movies.reset_index(drop=True)
```

```
movies.shape
```

```
(44407, 6)
```

```
ratings = pd.read_csv("ratings_small.csv")
```

```
ratings
```

```
ratings["date"] = pd.to_datetime(ratings["timestamp"],unit="s")
```

```
ratings
```

	userId	movieId	rating	timestamp	date
0	1	31	2.5	1260759144	2009-12-14 02:52:24
1	1	1029	3.0	1260759179	2009-12-14 02:52:59
2	1	1061	3.0	1260759182	2009-12-14 02:53:02
3	1	1129	2.0	1260759185	2009-12-14 02:53:05
4	1	1172	4.0	1260759205	2009-12-14 02:53:25
...	...	...	...	...	...
99999	671	6268	2.5	1065579370	2003-10-08 02:16:10
100000	671	6269	4.0	1065149201	2003-10-03 02:46:41
100001	671	6365	4.0	1070940363	2003-12-09 03:26:03
100002	671	6385	2.5	1070979663	2003-12-09 14:21:03
100003	671	6565	3.5	1074784724	2004-01-22 15:18:44

100004 rows × 5 columns

```
ratings.isnull().sum()
```

```
userId       0
movieId      0
rating       0
timestamp    0
date         0
dtype: int64
```

```
ratings.duplicated().sum()
```

```
0
```

```
movies["id"].nunique()
```

```
44405
```

```
movies = movies.rename(columns={"id":"movieId"})
```

```
movies
```

	movieId	overview	release_date	title	vote_average	vote_count
0	862	Led by Woody, Andy's toys live happily in his ...	1995-10-30	Toy Story	7.7	5415.0
1	8844	When siblings Judy and Peter discover an encha...	1995-12-15	Jumanji	6.9	2413.0
2	15602	A family wedding reignites the ancient feud be...	1995-12-22	Grumpier Old Men	6.5	92.0
3	31357	Cheated on, mistreated and stepped on, the wom...	1995-12-22	Waiting to Exhale	6.1	34.0
4	11862	Just when George Banks has recovered from his ...	1995-02-10	Father of the Bride Part II	5.7	173.0
...	...	...	...	...	...	...
		Yet another				

Next steps: Generate code with movies View recommended plots

```
movies.dtypes
```

```
movieId          object
overview         object
release_date     object
title            object
vote_average    float64
vote_count      float64
dtype: object
```

```
ratings.dtypes
```

```
userId                int64
movieId               int64
rating              float64
timestamp             int64
date         datetime64[ns]
dtype: object
```

```
movies["movieId"] = movies["movieId"].astype("int64")
```

```
moviemerge_df = pd.merge(movies, ratings, on="movieId", how="inner")
```

```
moviemerge_df
```

1	949	Obsessive master thief, Neil McCauley leads a ...	1995-12-15	Heat	7.7	1886.0	102
2	949	Obsessive master thief, Neil McCauley leads a ...	1995-12-15	Heat	7.7	1886.0	232
3	949	Obsessive master thief, Neil McCauley leads a ...	1995-12-15	Heat	7.7	1886.0	242
4	949	Obsessive master thief, Neil McCauley leads a ...	1995-12-15	Heat	7.7	1886.0	263
...	...	...	...	...	...	...	...
44818	64197	Plucked from an orphanage as a literal love sl...	2007-06-25	Travelling with Pets	6.0	5.0	73
44819	64197	Plucked from an orphanage as a literal love sl...	2007-06-25	Travelling with Pets	6.0	5.0	544
44820	64197	Plucked from an orphanage as a literal love sl...	2007-06-25	Travelling with Pets	6.0	5.0	648
44821	98604	Masha Krapivina - is yet beautiful, and not th...	2012-02-14	Cinderella	4.6	6.0	352
44822	49280	A band-leader has arranged seven chairs for th...	1900-01-01	The One-Man Band	6.5	22.0	187

Next steps: Generate code with `moviemerge_df` View recommended plots

```
moviemerge_df["movieId"].nunique()
```

```
2808
```

```
user_title_df = moviemerge_df.groupby(["userId","movieId"])["rating"].mean().unstack().notnull()
```

```
user_title_df
```

movieId	2	3	5	6	11	12	13	14	15	16	...	132961	13
userId													
1	False	False	False	False	False	False	False	False	False	False	...	False	I
2	False	False	False	False	False	False	False	False	False	False	...	False	I
3	False	False	False	False	False	False	False	False	False	False	...	False	I
4	False	False	False	False	False	False	False	False	False	False	...	False	I
5	False	True	False	False	False	False	False	False	False	False	...	False	I
...	...	...	...	...	...	...	...	...	...	...	...	...	
667	False	False	False	True	True	False	False	False	False	True	...	False	I
668	False	False	False	False	False	False	False	False	False	False	...	False	I
669	False	False	False	False	False	False	False	False	False	False	...	False	I
670	False	False	False	False	False	False	False	False	False	False	...	False	I
671	False	False	False	False	False	False	False	False	False	False	...	False	I

671 rows × 2808 columns

```
def map_ratings(row):
    return (row['movieId'], row['rating'])


def reduce_average_ratings(movie, ratings):
    return (movie, sum(ratings) / len(ratings))


movie_ratings = {}


for index, row in moviemerge_df.iterrows():
    movie_id = row['movieId']
    rating = row['rating']
    if movie_id not in movie_ratings:
        movie_ratings[movie_id] = []
    movie_ratings[movie_id].append(rating)


average_ratings = {}
for movie_id, ratings in movie_ratings.items():
    average_ratings[movie_id] = reduce_average_ratings(movie_id, ratings)[1]


for movie_id, avg_rating in average_ratings.items():
    print(f"Movie {movie_id}: Average Rating {avg_rating}")
```

```
Movie 6182: Average Rating 4.0
Movie 75803: Average Rating 1.5
Movie 49530: Average Rating 4.04054054054054
Movie 3966: Average Rating 4.75
Movie 92751: Average Rating 4.0
Movie 37736: Average Rating 3.0
Movie 47099: Average Rating 3.7916666666666665
Movie 4195: Average Rating 4.0
Movie 4809: Average Rating 3.8
Movie 64983: Average Rating 3.357142857142857
Movie 1254: Average Rating 4.3
Movie 56801: Average Rating 1.875
```