```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

pd.set_option("display.max_columns", None)


ratings = pd.read_csv("ratings.dat", sep="::", names=["UserID", "MovieID", "Rating",
"Timestamp"], encoding="ISO-8859-1", engine="python")

# 25개 이상 영화를 본 이용자만 추출
filtered_ratings = ratings.groupby("UserID").filter(lambda x: len(x) >= 25)

# training-testing split
ratings_train, ratings_test = train_test_split(filtered_ratings, test_size=0.3,
random_state=77)

seen_movies = ratings_train.loc[:, "MovieID"]
movie_ID, movie_counts = np.unique(seen_movies, return_counts=True)

# 전체 영화 랭킹
rankings = dict(zip(movie_ID, movie_counts))
rankings = sorted(rankings.items(), reverse=True, key=lambda x: x[1])

rankings_df = pd.DataFrame(rankings, columns = ["MovieID", "Counts"])


movies = pd.read_csv("movies.dat", sep="::", names=["MovieID", "Title", "Genres"],
encoding="ISO-8859-1", engine="python")

movies["Genres"] = movies["Genres"].str.split("|")

# movie와raking 데이터프레임join하여 랭킹에 따라movie 특성 정렬
movies = pd.merge(movies, rankings_df, left_on="MovieID", right_on="MovieID",
how="inner")
movies = movies.sort_values("Counts", ascending=False)

movies["Genres_string"] = movies["Genres"].apply(lambda x : (" ").join(x))

# 영화 장르별 랭킹
Action_rankings = movies[movies["Genres_string"].str.contains("Action", case=False)]
```

```
# user와movie 데이터프레임join하여 이용자와 해당 이용자가 본 영화 장르 연관 짓기
user_genres = pd.merge(ratings_train, movies, left_on="MovieID", right_on="MovieID",
how="inner")
user_genres = user_genres.loc[:, ["UserID", "MovieID", "Genres"]]

# 이용자별favorite 장르(가장 많이 본 장르) 찾기
fav_genre = user_genres.groupby("UserID")["Genres"].agg(**{"Genre":lambda
x:x.mode()})
```