# 🚀 MovieLens Project

🔸 *Azimov Alisher* (Shool21 - `perraseb`)

---

## 🗂️ I. Данные в `ml-latest-small`:

---

### 📄 `ratings.csv`

📌 Содержит оценки фильмов от пользователей
**Столбцы:**

- 👤 `userId` — ID пользователя
- 🎬 `movieId` — ID фильма
- ⭐ `rating` — оценка (от 0.5 до 5.0)
- 🕒 `timestamp` — время, когда пользователь поставил оценку

### 🎞️ `movies.csv`

📌 Список фильмов и их жанры
**Столбцы:**

- 🎬 `movieId` — ID фильма
- 🎥 `title` — название фильма
- 🏷️ `genres` — жанры через `|`

### 🏷️ `tags.csv`

📌 Пользовательские теги к фильмам
**Столбцы:**

- 👤 `userId` — ID пользователя
- 🎬 `movieId` — ID фильма
- 📝 `tag` — текст тега
- 🕒 `timestamp` — время, когда тег был добавлен

### 🌐 `links.csv`

📌 Ссылки на внешние базы фильмов (IMDb, TMDb)

**Столбцы:**

- 🎬 `movieId` — ID фильма
- 🎞️ `imdbId` — ID фильма в IMDb
- 🎟️ `tmdbId` — ID фильма в TMDb

---

# 🚀 II. Как запустить проекта в Linux

Для запуска проекта с аналитикой `MovieLens` необходимо выполнить следующие шаги:

---

## 🛠️ 1. Создание виртуального окружения

```
python3 -m venv jinnjadz
```

## ⚙ 2. Активация окружения

```
source jinnjadz/bin/activate
```

## 📦 3. Установка необходимых библиотек

```
pip install -r requirements.txt
```

## 📥 4. Загрузка `.csv` файлов

Если после клонирования проекта `.csv` файлы в `src/ml-latest-small/` оказались пустыми (это происходит при использовании Git LFS или ссылок), необходимо загрузить сами данные через Git LFS (Large File Storage).

```
git lfs install
git lfs pull
```

## 🧠 5. Создание временного ядра для Jupyter

Для запуска анализа в `movielens_report.ipynb` из виртуального окружения, создадим временное ядро Jupyter:

```
python -m ipykernel install --name temp_jinnjadz --display-name "Temp jinnjadz"
```

🧩 Как выбрать ядро

После выполнения команды:

В данном фале 📓 movielens_report.ipynb, если вы открыли через VS Code.

В правом верхнем углу нажми на кнопку выбора ядра (⚙ Kernel).

Выбери ядро "Temp jinnjadz" из выпадающего списка. (если не выходит перезапустите VS CODE)

🧹 Удаление ядра после анализа

Чтобы не засорять список ядер, по завершении работы можно удалить это ядро:

```
jupyter kernelspec uninstall temp_jinnjadz
```

## ✅ 6. Запуск автотестов

После того как все данные загружены и окружение настроено, можно проверить, что всё работает правильно.

Находясь в директории `src/`, запустите команду:

pytest
🧪 Это выполнит все тесты, написанные в movie_analysis/test_all.py, и выведет информацию об успешных и проваленных проверках.

---

# 📊 III. Анализ данных с помощью созданных нами классов

Перед началом анализа подключим все ранее разработанные классы:

In [83]:
```
import movielens_analysis as ma

print("Модуль movielens_analysis успешно подключён! Готов к анализу данных")
```

```
Модуль movielens_analysis успешно подключён! Готов к анализу данных
```

---

## 🎬 Анализ данных из `movies.csv`

Переходим к исследованию информации о фильмах, жанрах и датах выхода.

Для анализа используется созданный нами класс:

In [ ]:
```
dM=ma.Movies(limit=1000)
dM.get_all()[0:2]
```

Out[ ]:
```
[{'movieID': 1,
  'title': 'Toy Story',
  'release': 1995,
  'genres': ['Adventure', 'Animation', 'Children', 'Comedy', 'Fantasy']},
 {'movieID': 2,
  'title': 'Jumanji',
  'release': 1995,
  'genres': ['Adventure', 'Children', 'Fantasy']}]
```

Возвращает список всех фильмов (ID, название, жанры, год)

---

📅 Возвращает распределение фильмов по годам выпуска

In [12]:
```
dM.dist_by_release()
```

Out[12]:
```
{1995: 224,
 1994: 184,
 1996: 181,
 1993: 101,
 1992: 23,
 1990: 15,
 1991: 15,
 1989: 14,
 1986: 9,
 1982: 8,
 1940: 8,
 1957: 8,
 1987: 8,
 1980: 8,
 1981: 7,
 1988: 7,
 1979: 7,
 1955: 6,
 1959: 6,
 1968: 6,
 1997: 6,
 1939: 6,
 1985: 6,
 1967: 5,
 1965: 5,
 1951: 5,
 1958: 5,
 1944: 5,
 1941: 5,
 1975: 5,
 1971: 5,
 1984: 5,
 1964: 4,
 1973: 4,
 1954: 4,
 1934: 4,
 1960: 4,
 1963: 4,
 1950: 4,
 1974: 4,
 1983: 4,
 1977: 3,
 1937: 3,
 1972: 3,
 1952: 3,
 1961: 3,
 1953: 3,
 1946: 3,
 1938: 3,
 1956: 3,
 1962: 3,
 1976: 2,
 1969: 2,
 1970: 2,
```

```
 1942: 2,
 1945: 2,
 1947: 2,
 1935: 2,
 1936: 2,
 1949: 2,
 1978: 2,
 1943: 1,
 1932: 1,
 1966: 1,
 1948: 1,
 1933: 1,
 1931: 1}
```

---

📊 Возвращает распределение фильмов по жанрам

In [13]:
```
dM.dist_by_genres()
```

Out[13]:
```
{'Drama': 507,
 'Comedy': 365,
 'Romance': 208,
 'Thriller': 179,
 'Action': 158,
 'Adventure': 126,
 'Crime': 122,
 'Children': 100,
 'Fantasy': 69,
 'Sci-Fi': 69,
 'Mystery': 58,
 'Musical': 53,
 'Horror': 51,
 'War': 48,
 'Animation': 37,
 'Documentary': 25,
 'Western': 23,
 'Film-Noir': 18,
 'IMAX': 3}
```

In [24]:
```
# Покажем в процента
allGenre=sum(dM.dist_by_genres().values())

genreProtsent={}

for k,v in dM.dist_by_genres().items():
    genreProtsent[k]=float((v/allGenre)*100)

genreProtsent
```

Out[24]:
```
{'Drama': 22.84812978819288,
 'Comedy': 16.448850833708878,
 'Romance': 9.373591707976566,
 'Thriller': 8.066696710229833,
 'Action': 7.120324470482199,
 'Adventure': 5.678233438485805,
 'Crime': 5.497972059486255,
 'Children': 4.506534474988734,
 'Fantasy': 3.1095087877422265,
 'Sci-Fi': 3.1095087877422265,
 'Mystery': 2.6137899954934656,
 'Musical': 2.388463271744029,
 'Horror': 2.298332582244254,
 'War': 2.163136547994592,
 'Animation': 1.6674177557458314,
 'Documentary': 1.1266336187471835,
 'Western': 1.0365029292474088,
 'Film-Noir': 0.811176205497972,
 'IMAX': 0.135196034249662}
```

---

🏆 Возвращает топ-N фильмов с наибольшим количеством жанров

In [14]:
```
dM.most_genres(10)
```

Out[14]:
```
{'Strange Days': 6,
 'Lion King, The': 6,
 'Getaway, The': 6,
 'Super Mario Bros.': 6,
 'Beauty and the Beast': 6,
 'All Dogs Go to Heaven 2': 6,
 'Space Jam': 6,
 'Aladdin and the King of Thieves': 6,
 'Toy Story': 5,
 'Money Train': 5}
```

In [ ]:
```
# Давайте посмотрим какие жанры у данных фильмах

for i in dM.get_all():
    if i['title'] in dM.most_genres(10).keys():
        print(i['title'],' - ', i['genres'])
```

```
Toy Story  -  ['Adventure', 'Animation', 'Children', 'Comedy', 'Fantasy']
Money Train  -  ['Action', 'Comedy', 'Crime', 'Drama', 'Thriller']
Strange Days  -  ['Action', 'Crime', 'Drama', 'Mystery', 'Sci-Fi', 'Thriller']
Lion King, The  -  ['Adventure', 'Animation', 'Children', 'Drama', 'Musical',
'IMAX']
Getaway, The  -  ['Action', 'Adventure', 'Crime', 'Drama', 'Romance', 'Thrille
r']
Super Mario Bros.  -  ['Action', 'Adventure', 'Children', 'Comedy', 'Fantasy',
'Sci-Fi']
Beauty and the Beast  -  ['Animation', 'Children', 'Fantasy', 'Musical', 'Roman
ce', 'IMAX']
All Dogs Go to Heaven 2  -  ['Adventure', 'Animation', 'Children', 'Fantasy',
'Musical', 'Romance']
Space Jam  -  ['Adventure', 'Animation', 'Children', 'Comedy', 'Fantasy', 'Sci-
Fi']
Aladdin and the King of Thieves  -  ['Animation', 'Children', 'Comedy', 'Fantas
y', 'Musical', 'Romance']
```

---

## 📈 Вывод по данным из `movies.csv`

На основе анализа первых **1000 фильмов** из датасета можно сделать следующие наблюдения:

- 📅 **Основной период выпуска фильмов — 1993–1996 годы**
- 🎭 Среди жанров наибольшую популярность имели:
  - **Драма** — `22.84%`
  - **Комедия** — `16.44%`

Это отражает интересы аудитории 90-х годов.

---

## ⭐ Анализ данных из `ratings.csv`

In [33]:
```
dR=ma.Ratings(limit=1000)

dR.data[:2]
```

Out[33]:
```
[{'userId': '1', 'movieId': '1', 'rating': '4.0', 'timestamp': '964982703'},
 {'userId': '1', 'movieId': '3', 'rating': '4.0', 'timestamp': '964981247'}]
```

---

🗓️ Распределение оценок по годам

In [36]:
```
dRm=dR.Movies(dR) #Передаем данные внутренному классу Movies
dRm.data[:2]
```

Out[36]:
```
[{'userId': '1', 'movieId': '1', 'rating': '4.0', 'timestamp': '964982703'},
 {'userId': '1', 'movieId': '3', 'rating': '4.0', 'timestamp': '964981247'}]
```

In [37]:
```
dRm.dist_by_year()
```

Out[37]:
```
{1996: 358,
 2000: 296,
 2005: 121,
 1999: 82,
 2001: 70,
 2011: 39,
 2015: 29,
 2006: 4,
 2007: 1}
```

---

🔢 `dist_by_rating()` — распределение по значениям оценок

In [38]:
```
dRm.dist_by_rating()
```

Out[38]:
```
{'4.0': 292,
 '5.0': 267,
 '3.0': 253,
 '2.0': 57,
 '1.0': 39,
 '4.5': 33,
 '0.5': 24,
 '3.5': 17,
 '1.5': 11,
 '2.5': 7}
```

---

🏆 `top_by_num_of_ratings(n)` — топ-N фильмов по количеству оценок

In [39]:
```
dRm.top_by_num_of_ratings(10)
```

Out[39]:
```
{'Usual Suspects, The': 4,
 'Pulp Fiction': 4,
 'Fugitive, The': 4,
 "Schindler's List": 4,
 'Batman': 4,
 'Silence of the Lambs, The': 4,
 'Fargo': 4,
 'Aladdin': 4,
 'Beauty and the Beast': 4,
 'Toy Story': 3}
```

---

📈 `top_by_ratings(n, metric)` — топ-N фильмов по средним или медианным оценкам

In [40]:
```
dRm.top_by_ratings(10,'average')
```

Out[40]:
```
{'Bottle Rocket': 5.0,
 'Canadian Bacon': 5.0,
 'Star Wars: Episode IV - A New Hope': 5.0,
 'James and the Giant Peach': 5.0,
 'Wizard of Oz, The': 5.0,
 'Citizen Kane': 5.0,
 'Adventures of Robin Hood, The': 5.0,
 'Mr. Smith Goes to Washington': 5.0,
 'Winnie the Pooh and the Blustery Day': 5.0,
 'Three Caballeros, The': 5.0}
```

In [41]:
```
dRm.top_by_ratings(10,'median')
```

Out[41]:
```
{'Bottle Rocket': 5.0,
 'Canadian Bacon': 5.0,
 'Star Wars: Episode IV - A New Hope': 5.0,
 'Tommy Boy': 5.0,
 'Forrest Gump': 5.0,
 'Fugitive, The': 5.0,
 'Jurassic Park': 5.0,
 'Tombstone': 5.0,
 'Dances with Wolves': 5.0,
 'Pinocchio': 5.0}
```

---

⚖ `top_controversial(n)` — топ-N самых противоречивых фильмов (по дисперсии)

In [42]:
```
dRm.top_controversial(10)
```

Out[42]:
```
{'My Fair Lady': 5.06,
 "Schindler's List": 3.42,
 'Courage Under Fire': 3.06,
 'Usual Suspects, The': 2.42,
 'Dazed and Confused': 2.25,
 'Circle of Friends': 2.25,
 'Hot Shots! Part Deux': 2.25,
 'Ghost': 2.25,
 'Pulp Fiction': 2.19,
 'Tombstone': 2.0}
```

---

📊 `dist_by_num_of_rating()` — активность пользователей по количеству оценок

In [44]:
```
dRu=dR.Users(dR) # Переходим внутренний на класс users который наследуеться от

dRu.dist_by_num_of_rating()
```

Out[44]: {'6': 314, '1': 232, '4': 216, '7': 126, '5': 44, '3': 39, '2': 29}

---

📏 `dist_by_rating_values(metric)` — рейтинг пользователей по средней или медианной оценке

In [46]:
```
dRu.dist_by_rating_values('median')
```

Out[46]: {'1': 5.0, '2': 4.0, '4': 4.0, '5': 4.0, '7': 4.0, '6': 3.0, '3': 0.5}

In [47]:
```
dRu.dist_by_rating_values()
```

Out[47]: {'1': 4.37, '2': 3.95, '5': 3.64, '4': 3.56, '6': 3.49, '7': 3.35, '3': 2.44}

---

💥 `top_controversial_users(n)` — топ-N пользователей с наибольшей вариативностью оценок

In [48]:
```
dRu.top_controversial_users(10)
```

Out[48]: {'3': 4.26, '4': 1.72, '7': 1.65, '5': 0.96, '6': 0.72, '1': 0.64, '2': 0.63}

---

## 📈 Вывод по данным из `ratings.csv`

На основе анализа оценок пользователей можно выделить следующие наблюдения:

- 🕰️ **Пик активности** пришёлся на период **1996–2000 гг.**, особенно в 1996 году (358 оценок).
- ⭐ Наиболее часто встречающиеся оценки:
  - `4.0` — 292 оценок
  - `5.0` — 267 оценок
  - `3.0` — 253 оценок

  → Пользователи склонны ставить **высокие оценки**.
- 🎞️ Топ фильмов по количеству оценок включает культовые картины:
  *Usual Suspects, Pulp Fiction, Schindler's List, Batman, Fargo* и др.
- 📊 Средняя и медианная оценка у ряда фильмов достигает **максимума — 5.0**, например: *Star Wars*, *Forrest Gump*, *Citizen Kane*,

*Tommy Boy*

- ⚖️ Самыми противоречивыми (по дисперсии) фильмами стали:
    - *My Fair Lady* (5.06)
    - *Schindler's List* (3.42)
    - *Courage Under Fire* (3.06)
- 👤 В разрезе пользователей:
    - Пользователь `6` — самый активный (314 оценок)
    - Пользователь `3` — наиболее противоречивый (дисперсия 4.26)
    - Средняя оценка у большинства пользователей — **выше 3.5**, что говорит о позитивном восприятии фильмов.

---

## 🏷️ Анализ данных из `tags.csv`

In [49]:
```
dT=ma.Tags(limit=1000)

dT.get_tags()[:2]
```

Out[49]:
```
[{'userId': 2, 'movieId': 60756, 'tag': 'funny', 'timestamp': 1445714994},
 {'userId': 2,
  'movieId': 60756,
  'tag': 'Highly quotable',
  'timestamp': 1445714996}]
```

---

📝 `most_words(n)` — теги с наибольшим числом слов

In [50]:
```
dT.most_words(10)
```

Out[50]:
```
{'something for everyone in this one... saw it without and plan on seeing it
with kids!': 16,
 'the catholic church is the most corrupt organization in history': 10,
 'oscar (best music - original score)': 6,
 'based on a true story': 5,
 'everything you want is here': 5,
 'based on a play': 4,
 'end of the world': 4,
 'guardians of the galaxy': 4,
 'a clever chef rat': 4,
 'based on a book': 4}
```

---

🔠 `longest(n)` — теги с наибольшей длиной (в символах)

In [51]:
```
dT.longest(10)
```

Out[51]:
```
[('something for everyone in this one... saw it without and plan on seeing it
with kids!',
  85),
 ('the catholic church is the most corrupt organization in history', 63),
 ('audience intelligence underestimated', 36),
 ('oscar (best music - original score)', 35),
 ('assassin-in-training (scene)', 28),
 ('everything you want is here', 27),
 ('political right versus left', 27),
 ('oscar (best cinematography)', 27),
 ('representation of children', 26),
 ('guardians of the galaxy', 23)]
```

---

🧩 `most_words_and_longest(n)` — пересечение длинных и многословных тегов

In [52]:
```
dT.most_words_and_longest(10)
```

Out[52]:
```
['everything you want is here',
 'guardians of the galaxy',
 'oscar (best music - original score)',
 'something for everyone in this one... saw it without and plan on seeing it
with kids!',
 'the catholic church is the most corrupt organization in history']
```

---

🔝 `most_popular(n)` — самые часто встречающиеся теги

In [53]:
```
dT.most_popular(10)
```

Out[53]:
```
{'funny': 15,
 'sci-fi': 15,
 'twist ending': 13,
 'action': 13,
 'dark comedy': 12,
 'atmospheric': 12,
 'comedy': 12,
 'suspense': 11,
 'superhero': 10,
 'will ferrell': 9}
```

---

🔎 `tags_with(word)` — все теги, содержащие указанное слово

In [54]:
```
dT.tags_with('sci')
```

Out[54]:
```
['classic sci-fi',
 'mad scientist',
 'sci-fi',
 'science fiction',
 'scifi',
 'scifi cult',
 'sexy female scientist']
```

In [ ]:
```
dT.tags_with('good')
```

Out[ ]:
```
['feel-good',
 'good dialogue',
 'good soundtrack',
 'good writing',
 'oldie but goodie']
```

---

📅 `dist_by_year()` — распределение тегов по годам (по timestamp)

In [60]:
```
dT.dist_by_year()
```

Out[60]:
```
{2018: 412,
 2016: 355,
 2015: 79,
 2012: 45,
 2006: 42,
 2009: 13,
 2010: 13,
 2013: 9,
 2007: 8,
 2014: 7,
 2008: 7,
 2011: 7,
 2017: 3}
```

---

📅 `dist_by_month()` — распределение тегов по месяцам (по timestamp)

In [ ]:
```
dT.dist_by_month() # месяц это число от 1 до 12
```

Out[ ]:
```
{3: 302,
 5: 253,
 9: 153,
 6: 106,
 7: 57,
 2: 37,
 1: 33,
 10: 16,
 4: 16,
 11: 10,
 12: 9,
 8: 8}
```

---

## 📈 Вывод по данным из `tags.csv`

На основе анализа пользовательских тегов можно выделить следующие наблюдения:

- 📝 **Самый многословный тег** содержит **16 слов** и выглядит как мини-рецензия:

  > "*something for everyone in this one... saw it without and plan on seeing it with kids!*"

- 🔠 **Самый длинный по символам тег** — тот же, что и самый многословный — **85 символов**
  Также в топ входят теги с критикой, описаниями жанров и сюжетов (*e.g.,* "*the catholic church is the most corrupt organization in history*")

- 🧩 Пересечение между **топ-10 по длине** и **по количеству слов** показывает содержательные, развернутые пользовательские комментарии.

- 🔝 **Самые популярные теги** по частоте упоминания:

  - `funny`, `sci-fi`, `twist ending`, `action`, `dark comedy` → Это указывает на интерес пользователей к определённым жанрам и эмоциональной окраске фильмов.

- 📆 **Год наибольшей активности тегирования — 2018** (412 тегов), за ним следуют 2016 и 2015
  Это может быть связано с активным использованием платформы MovieLens в те годы.

- 📅 **Самые "тегированные" месяцы** — март (302) и май (253), что может говорить о сезонной активности пользователей (например, весной перед отпускным сезоном).

---

## 🌐 Анализ данных из `links.csv` и IMDb

In [ ]:
```
dL=ma.Links('./ml-latest-small/links.csv',100)
dL.get_links() # imovieId и imdbId
```

Out[ ]: {1: 114709,
2: 113497,
3: 113228,
4: 114885,
5: 113041,
6: 113277,
7: 114319,
8: 112302,
9: 114576,
10: 113189,
11: 112346,
12: 112896,
13: 112453,
14: 113987,
15: 112760,
16: 112641,
17: 114388,
18: 113101,
19: 112281,
20: 113845,
21: 113161,
22: 112722,
23: 112401,
24: 114168,
25: 113627,
26: 114057,
27: 114011,
28: 114117,
29: 112682,
30: 115012,
31: 112792,
32: 114746,
34: 112431,
36: 112818,
38: 113442,
39: 112697,
40: 112749,
41: 114279,
42: 112819,
43: 114272,
44: 113855,
45: 114681,
46: 113347,
47: 114369,
48: 114148,
49: 114916,
50: 114814,
52: 113819,
53: 110299,
54: 112499,
55: 113158,
57: 113321,
58: 110877,
60: 113419,

```
 61: 116260,
 62: 113862,
 63: 116126,
 64: 118002,
 65: 115683,
 66: 116839,
 68: 113149,
 69: 113118,
 70: 116367,
 71: 113010,
 72: 113537,
 73: 113828,
 74: 115644,
 75: 115676,
 76: 114367,
 77: 113973,
 78: 112744,
 79: 116731,
 80: 112445,
 81: 114660,
 82: 112379,
 83: 114039,
 85: 112365,
 86: 118158,
 87: 116151,
 88: 115697,
 89: 113972,
 92: 117002,
 93: 114825,
 94: 115639,
 95: 115759,
 96: 113403,
 97: 113247,
 99: 113283,
 100: 115907,
 101: 115734,
 102: 117102,
 103: 118040,
 104: 116483,
 105: 112579,
 106: 110251,
 107: 117110,
 108: 112646,
 110: 112573,
 111: 75314,
 112: 113326}
```

---

📝 Получаем данные из сайта imdb

In [ ]:
```
id_list=list(dL.get_links().keys())
```

```
dL.get_imdb(id_list[19:99:4]) # Из за медленной работы библиотеки requests огр
```

```
Запрашиваем: https://www.imdb.com/title/tt0113845/
Запрашиваем: https://www.imdb.com/title/tt0114168/
Запрашиваем: https://www.imdb.com/title/tt0114117/
Запрашиваем: https://www.imdb.com/title/tt0114746/
Запрашиваем: https://www.imdb.com/title/tt0112697/
Запрашиваем: https://www.imdb.com/title/tt0114272/
Запрашиваем: https://www.imdb.com/title/tt0114369/
Запрашиваем: https://www.imdb.com/title/tt0113819/
Запрашиваем: https://www.imdb.com/title/tt0113321/
Запрашиваем: https://www.imdb.com/title/tt0113862/
Запрашиваем: https://www.imdb.com/title/tt0116839/
Запрашиваем: https://www.imdb.com/title/tt0113010/
Запрашиваем: https://www.imdb.com/title/tt0115676/
Запрашиваем: https://www.imdb.com/title/tt0116731/
Запрашиваем: https://www.imdb.com/title/tt0114039/
Запрашиваем: https://www.imdb.com/title/tt0115697/
Запрашиваем: https://www.imdb.com/title/tt0115639/
Запрашиваем: https://www.imdb.com/title/tt0113283/
Запрашиваем: https://www.imdb.com/title/tt0118040/
Запрашиваем: https://www.imdb.com/title/tt0117110/
```

Out[ ]:

```
[{'movie_id': 107,
  'Director': 'Brian Henson',
  'Budget': None,
  'Cumulative Worldwide Gross': '$34,327,766',
  'Runtime': '1 hour 39 minutes'},
 {'movie_id': 103,
  'Director': 'John Dahl',
  'Budget': '$18,000,000 (estimated)',
  'Cumulative Worldwide Gross': '$2,821,671',
  'Runtime': '1 hour 57 minutes'},
 {'movie_id': 99,
  'Director': 'Nick Broomfield',
  'Budget': None,
  'Cumulative Worldwide Gross': '$34,402',
  'Runtime': '1 hour 46 minutes'},
 {'movie_id': 94,
  'Director': 'Ted Demme',
  'Budget': None,
  'Cumulative Worldwide Gross': '$10,597,759',
  'Runtime': '1 hour 52 minutes'},
 {'movie_id': 88,
  'Director': 'Penelope Spheeris',
  'Budget': None,
  'Cumulative Worldwide Gross': '$32,417,995',
  'Runtime': '1 hour 27 minutes'},
 {'movie_id': 83,
  'Director': 'Tim Reid',
  'Budget': None,
  'Cumulative Worldwide Gross': '$2,291,255',
  'Runtime': '1 hour 55 minutes'},
 {'movie_id': 79,
  'Director': 'Brian Gibson',
  'Budget': '$44,000,000 (estimated)',
  'Cumulative Worldwide Gross': '$22,754,725',
  'Runtime': '1 hour 58 minutes'},
 {'movie_id': 75,
  'Director': 'Steve Miner',
  'Budget': '$15,000,000 (estimated)',
  'Cumulative Worldwide Gross': '$2,042,530',
  'Runtime': '1 hour 30 minutes'},
 {'movie_id': 71,
  'Director': 'Andrew Sipes',
  'Budget': '$50,000,000 (estimated)',
  'Cumulative Worldwide Gross': '$11,534,477',
  'Runtime': '1 hour 31 minutes'},
 {'movie_id': 66,
  'Director': 'Farhad Mann',
  'Budget': '$15,000,000 (estimated)',
  'Cumulative Worldwide Gross': '$2,409,225',
  'Runtime': '1 hour 33 minutes'},
 {'movie_id': 62,
  'Director': 'Stephen Herek',
  'Budget': '$23,000,000 (estimated)',
  'Cumulative Worldwide Gross': '$106,269,971',
```

```
 'Runtime': '2 hours 24 minutes'},
{'movie_id': 57,
 'Director': 'Jodie Foster',
 'Budget': '$20,000,000 (estimated)',
 'Cumulative Worldwide Gross': '$17,519,169',
 'Runtime': '1 hour 43 minutes'},
{'movie_id': 52,
 'Director': 'Woody Allen',
 'Budget': '$15,000,000 (estimated)',
 'Cumulative Worldwide Gross': '$6,468,498',
 'Runtime': '1 hour 35 minutes'},
{'movie_id': 47,
 'Director': 'David Fincher',
 'Budget': '$33,000,000 (estimated)',
 'Cumulative Worldwide Gross': '$328,981,827',
 'Runtime': '2 hours 7 minutes'},
{'movie_id': 43,
 'Director': 'Michael Hoffman',
 'Budget': '$19,000,000 (estimated)',
 'Cumulative Worldwide Gross': '$4,005,941',
 'Runtime': '1 hour 57 minutes'},
{'movie_id': 39,
 'Director': 'Amy Heckerling',
 'Budget': '$12,000,000 (estimated)',
 'Cumulative Worldwide Gross': '$56,688,409',
 'Runtime': '1 hour 37 minutes'},
{'movie_id': 32,
 'Director': 'Terry Gilliam',
 'Budget': '$29,000,000 (estimated)',
 'Cumulative Worldwide Gross': '$168,839,459',
 'Runtime': '2 hours 9 minutes'},
{'movie_id': 28,
 'Director': 'Roger Michell',
 'Budget': '£1,000,000 (estimated)',
 'Cumulative Worldwide Gross': '$5,269,757',
 'Runtime': '1 hour 43 minutes'},
{'movie_id': 24,
 'Director': 'Victor Salva',
 'Budget': '$9,500,000 (estimated)',
 'Cumulative Worldwide Gross': '$30,862,156',
 'Runtime': '1 hour 51 minutes'},
{'movie_id': 20,
 'Director': 'Joseph Ruben',
 'Budget': '$68,000,000 (estimated)',
 'Cumulative Worldwide Gross': '$35,431,113',
 'Runtime': '1 hour 50 minutes'}]
```


---

### 🎬 `top_directors(n)`

Топ-N самых часто встречающихся режиссёров среди просмотренных фильмов.

In [94]:
```
dL.top_directors(10)
```

Out[94]:
```
{'Brian Henson': 1,
 'John Dahl': 1,
 'Nick Broomfield': 1,
 'Ted Demme': 1,
 'Penelope Spheeris': 1,
 'Tim Reid': 1,
 'Brian Gibson': 1,
 'Steve Miner': 1,
 'Andrew Sipes': 1,
 'Farhad Mann': 1}
```

---

- 💰 **`most_expensive(n)`**
  Фильмы с наибольшим бюджетом.

In [95]:
```
dL.most_expensive(10)
```

Out[95]:
```
{20: 68000000,
 71: 50000000,
 79: 44000000,
 47: 33000000,
 32: 29000000,
 62: 23000000,
 57: 20000000,
 43: 19000000,
 103: 18000000,
 75: 15000000}
```

---

🌍 **`most_profitable(n)`**
Фильмы с наибольшей разницей между сборами и бюджетом.

In [96]:
```
dL.most_profitable(10)
```

Out[96]:
```
{47: 295981827,
 32: 139839459,
 62: 83269971,
 39: 44688409,
 24: 21362156,
 57: -2480831,
 52: -8531502,
 66: -12590775,
 75: -12957470,
 43: -14994059}
```

---

- ⏱️ **`longest(n)`**
  Самые длинные фильмы по продолжительности (в минутах).

In [97]:
```
dL.longest(10)
```

Out[97]:
```
{62: 144,
 32: 129,
 47: 127,
 79: 118,
 103: 117,
 43: 117,
 83: 115,
 94: 112,
 24: 111,
 20: 110}
```

---

- 💸 **`top_cost_per_minute(n)`**
  Фильмы с самой дорогой стоимостью одной минуты хронометража.

In [98]:
```
dL.top_cost_per_minute(10)
```

Out[98]:
```
{20: 618181.82,
 71: 549450.55,
 79: 372881.36,
 47: 259842.52,
 32: 224806.2,
 57: 194174.76,
 75: 166666.67,
 43: 162393.16,
 66: 161290.32,
 62: 159722.22}
```

---

## 📈 Вывод по данным из `links.csv` и IMDb

На основе расширенного анализа фильмов с использованием IMDb-данных можно выделить следующее:

- 🎬 **Режиссёры**:
  Все 10 фильмов из текущей выборки были сняты разными режиссёрами.
  ➤ Это может указывать на разнообразие жанров и стилей.

- 💰 **Бюджет**:
  Самый дорогой фильм из анализируемых — с бюджетом **68 млн долларов**,
  далее следуют фильмы с бюджетами от **50 до 15 млн долларов**.

- 🌍 **Прибыльность**:
  Наиболее прибыльный фильм заработал почти **296 млн долларов**,
  однако в выборке также есть фильмы с **убытками от -2.5 до -15 млн долларов**,
  что указывает на коммерческие провалы несмотря на затраты.

- ⏱️ **Продолжительность**:
  Самый длинный фильм длится **144 минуты** (~2 часа 24 минуты), большинство других — в диапазоне **110–130 минут**.

- 💸 **Стоимость одной минуты фильма**: Стоимость производства одной минуты достигает:
  - до **618,000 $/мин** — впечатляющий показатель для высокобюджетных картин
  - минимальные значения в топе — от **150,000 до 200,000 $/мин**

---

# ✅ Финальный итог анализа

В ходе выполнения проекта мы:

- 📦 Проанализировали 4 основных источника данных:
  - `movies.csv` — жанры и годы выпуска
  - `ratings.csv` — оценки пользователей и активность
  - `tags.csv` — смысловые теги и поведенческие паттерны
  - `links.csv` + IMDb — бюджеты, кассовые сборы, режиссёры и продолжительность
- 🧠 Использовали **собственные классы** `Movies`, `Ratings`, `Tags`, `Links`
  для структурированной и расширяемой обработки данных

- 📊 Получили инсайты:

- 🎬 Основной массив фильмов — 1993–1996 гг., с доминированием жанров драма и комедия
- ⭐ Пользователи склонны ставить высокие оценки (4.0–5.0)
- 🏷️ Популярные теги — `funny`, `sci-fi`, `action`
- 🌍 Существуют как высокодоходные фильмы, так и провальные по бюджету
- 💸 Стоимость производства одной минуты фильма может достигать сотен тысяч долларов

🔍 Анализ позволил глубже понять структуру и свойства датасета `MovieLens`, а также развил навыки объектно-ориентированного проектирования и веб-скрейпинга.

---