LOAD THE NECESSARY LIBRARIES

```
import pandas as pd
```

```
movie = pd.read_csv('movie_rating_avg.csv')
```

```
movie.head()
```

	tconst	primaryTitle	startYear	runtimeMinutes	genres	movie_rating	avg_cast_rating	director_rating
0	tt19034332	The Mystery of Marilyn Monroe: The Unheard Tapes	2022	101	Biography,Crime,Documentary	6.2	6.20	6.20
1	tt6893836	They'll Love Me When I'm Dead	2018	98	Biography,Documentary	7.4	NaN	7.70
2	tt4566758	Mulan	2020	115	Action,Adventure,Drama	5.8	6.04	5.75
3	tt7131622	Once Upon a Time... in Hollywood	2019	161	Comedy,Drama	7.6	7.35	7.60
4	tt21279806	Scoop	2024	102	Biography,Drama	6.5	6.50	6.50

```
#converting the names into a list that i can use
movie_list = movie['primaryTitle'].tolist()
```

Importing titles from tmdb (that have similar titles).

```
import requests
import pandas as pd
import time

# 🔑 Replace this with your actual TMDb API key
TMDB_API_KEY = "***"

# Sample list of movie titles (replace this with your real list)
movies = movie_list.copy()

def tmdb_search_movie(title):
    """Search TMDb by title and return the best match (if any)."""
    url = "https://api.themoviedb.org/3/search/movie"
    params = {"api_key": TMDB_API_KEY, "query": title}
    response = requests.get(url, params=params)
    response.raise_for_status()
    results = response.json().get("results", [])
    if results:
        return results[0]  # best match
    return None

def tmdb_get_movie_details(tmdb_id):
    """Get full details of a movie by TMDb ID."""
    url = f"https://api.themoviedb.org/3/movie/{tmdb_id}"
    params = {"api_key": TMDB_API_KEY}
    response = requests.get(url, params=params)
    response.raise_for_status()
    return response.json()

# Collect results
data = []

for title in movies:
    print(f"Searching: {title}")
    try:
        result = tmdb_search_movie(title)
        time.sleep(0.25)  # to avoid hitting rate limits

        if not result:
            print(f"❌ No results for '{title}'")
            continue

        tmdb_id = result["id"]
```

```
            details = tmdb_get_movie_details(tmdb_id)
            time.sleep(0.25)

            data.append({
                "title": title,
                "tmdb_title": details.get("title"),
                "release_date": details.get("release_date"),
                "runtime": details.get("runtime"),
                "budget": details.get("budget"),
                "revenue": details.get("revenue"),
                "vote_average": details.get("vote_average"),
                "vote_count": details.get("vote_count"),
                "tmdb_id": tmdb_id,
                "imdb_id": details.get("imdb_id"),
                "genres": ", ".join([g['name'] for g in details.get("genres", [])]),
                "tmdb_url": f"https://www.themoviedb.org/movie/{tmdb_id}"
            })

        except Exception as e:
            print(f"⚠️ Error for '{title}': {e}")

    # Convert to DataFrame
    df = pd.DataFrame(data)

    # Compute profit and profit margin
    df["budget"] = pd.to_numeric(df["budget"], errors="coerce")
    df["revenue"] = pd.to_numeric(df["revenue"], errors="coerce")
    df["profit"] = df["revenue"] - df["budget"]
    df["profit_margin"] = df["profit"] / df["budget"]

    # Save to CSV
    df.to_csv("tmdb_movie_data.csv", index=False)
    print("\n✅ Data saved to tmdb_movie_data.csv")
    print(df.head())
```

```
Searching: Batman: Gotham by Gaslight
```

```
Searching: Housefull 5
Searching: The Woman King
Searching: After
Searching: The Oath
Searching: Leave No Trace
Searching: Last the Night
Searching: The Secret: Dare to Dream
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4127 entries, 0 to 4126
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   title          4127 non-null   object
 1   tmdb_title     4127 non-null   object
 2   release_date   4127 non-null   object
 3   runtime        4127 non-null   int64
 4   budget         4127 non-null   int64
 5   revenue        4127 non-null   int64
 6   vote_average   4127 non-null   float64
 7   vote_count     4127 non-null   int64
 8   tmdb_id        4127 non-null   int64
 9   imdb_id        4037 non-null   object
 10  genres         4127 non-null   object
 11  tmdb_url       4127 non-null   object
 12  profit         4127 non-null   int64
 13  profit_margin  1989 non-null   float64
dtypes: float64(2), int64(6), object(6)
memory usage: 451.5+ KB
```

The result from tmdb only contains titles from out imdb dataset.

```
df.head()
```

	title	tmdb_title	release_date	runtime	budget	revenue	vote_average	vote_count	tmdb_id	imdb_id	genres	h
0	The Mystery of Marilyn Monroe: The Unheard Tapes	The Mystery of Marilyn Monroe: The Unheard Tapes	2022-04-27	101	0	0	6.352	145	953300	tt19034332	Documentary	h
1	They'll Love Me When I'm Dead	They'll Love Me When I'm Dead	2018-08-31	98	0	0	7.100	142	538002	tt6893836	Documentary	h
2	Mulan	Mulan	1998-06-18	88	90000000	304320254	7.903	10132	10674	tt0120762	Animation, Family, Adventure	
3	Once Upon a Time... in Hollywood	Once Upon a Time... in Hollywood	2019-07-24	162	95000000	392105159	7.426	14234	466272	tt7131622	Comedy, Drama, Thriller	h
4	Scoop	Scoop	1996-01-02	100	0	0	6.000	1	334904	tt0274805	Comedy, Crime, Drama	h

```
Start coding or generate with AI.
```