```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
books = pd.read_csv('/content/drive/MyDrive/Predictive/Dataset/books.csv')
ratings = pd.read_csv('/content/drive/MyDrive/Predictive/Dataset/ratings.csv')
book_tags = pd.read_csv('/content/drive/MyDrive/Predictive/Dataset/book_tags.csv')
tags = pd.read_csv('/content/drive/MyDrive/Predictive/Dataset/tags.csv')
```

Clean the dataset

```
books['original_publication_year'] = books['original_publication_year'].fillna(-1).apply(lambda x: int(x) if x != -1 else -1)
```

```
ratings_rmv_duplicates = ratings.drop_duplicates()
unwanted_users = ratings_rmv_duplicates.groupby('user_id')['user_id'].count()
unwanted_users = unwanted_users[unwanted_users < 3]
unwanted_ratings = ratings_rmv_duplicates[ratings_rmv_duplicates.user_id.isin(unwanted_users.index)]
new_ratings = ratings_rmv_duplicates.drop(unwanted_ratings.index)
```

```
new_ratings['title'] = books.set_index('id').title.loc[new_ratings.book_id].values
new_ratings.head()
```

	book_id	user_id	rating	title
0	1	314	5	The Hunger Games (The Hunger Games, #1)

Simple Recommender

```
v = books['ratings_count']
m = books['ratings_count'].quantile(0.95)
R = books['average_rating']
C = books['average_rating'].mean()
W = (R*v + C*m) / (v + m)
```

```
books['weighted_rating'] = W
```

```
qualified  = books.sort_values('weighted_rating', ascending=False).head(250)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
qualified[['title', 'authors', 'average_rating', 'weighted_rating']].head(15)
```

	title	authors	average_rating	weighted_rating
24	Harry Potter and the Deathly Hallows (Harry Po...	J.K. Rowling, Mary GrandPré	4.61	4.555956
26	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling, Mary GrandPré	4.54	4.490428
17	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling, Mary GrandPré, Rufus Beck	4.53	4.485090
23	Harry Potter and the Goblet of Fire (Harry Pot...	J.K. Rowling, Mary GrandPré	4.53	4.483227
1	Harry Potter and the Sorcerer's Stone (Harry P...	J.K. Rowling, Mary GrandPré	4.44	4.424365

Unsupported Cell Type. Double-Click to inspect/edit the content.

30	The Help	Kathryn Stockett	4.45	4.405158

```
book_tags.head()
```

	goodreads_book_id	tag_id	count
0	1	30574	167697
1	1	11305	37174
2	1	11557	34173
3	1	8717	12986
4	1	33114	12716

```
tags.head()
```

	tag_id	tag_name
0	0	-
1	1	--1-
2	2	--10-
3	3	--12-
4	4	--122-

```
genres = ["Art", "Biography", "Business", "Chick Lit", "Children's", "Christian", "Classics",
          "Comics", "Contemporary", "Cookbooks", "Crime", "Ebooks", "Fantasy", "Fiction",
          "Gay and Lesbian", "Graphic Novels", "Historical Fiction", "History", "Horror",
          "Humor and Comedy", "Manga", "Memoir", "Music", "Mystery", "Nonfiction", "Paranormal",
          "Philosophy", "Poetry", "Psychology", "Religion", "Romance", "Science", "Science Fiction",
          "Self Help", "Suspense", "Spirituality", "Sports", "Thriller", "Travel", "Young Adult"]
```

```
genres = list(map(str.lower, genres))
genres[:4]
```

```
['art', 'biography', 'business', 'chick lit']
```

```
available_genres = tags.loc[tags.tag_name.str.lower().isin(genres)]
```

```
available_genres.head()
```

	tag_id	tag_name
2938	2938	art
4605	4605	biography
5951	5951	business
7077	7077	christian
7457	7457	classics

```
available_genres_books = book_tags[book_tags.tag_id.isin(available_genres.tag_id)]
```

```
print('There are {} books that are tagged with above genres'.format(available_genres_books.shape[0]))
```

```
There are 60573 books that are tagged with above genres
```

```
available_genres_books.head()
```

	goodreads_book_id	tag_id	count
1	1	11305	37174
5	1	11743	9954
25	1	7457	958

```
available_genres_books['genre'] = available_genres.tag_name.loc[available_genres_books.tag_id].values
available_genres_books.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/inde
  """Entry point for launching an IPython kernel.
```

	goodreads_book_id	tag_id	count	genre
1	1	11305	37174	fantasy
5	1	11743	9954	fiction
25	1	7457	958	classics
38	1	22973	673	paranormal
52	1	20939	465	mystery

```
def build_chart(genre, percentile=0.85):
    df = available_genres_books[available_genres_books['genre'] == genre.lower()]
    qualified = books.set_index('book_id').loc[df.goodreads_book_id]

    v = qualified['ratings_count']
    m = qualified['ratings_count'].quantile(percentile)
    R = qualified['average_rating']
    C = qualified['average_rating'].mean()
    qualified['weighted_rating'] = (R*v + C*m) / (v + m)

    qualified.sort_values('weighted_rating', ascending=False, inplace=True)
    return qualified
```

```
cols = ['title','authors','original_publication_year','average_rating','ratings_count','work_text_reviews_count','weighted_rating']
```

```
genre = 'Fiction'
build_chart(genre)[cols].head(15)
```

	title	authors	original_publication_year	average_rating	ratings_count	work_text
book_id						
	Harry Potter and the	J.K.				

```
list(enumerate(available_genres.tag_name))
```

```
[(0, 'art'),
 (1, 'biography'),
 (2, 'business'),
 (3, 'christian'),
 (4, 'classics'),
 (5, 'comics'),
 (6, 'contemporary'),
 (7, 'cookbooks'),
 (8, 'crime'),
 (9, 'ebooks'),
 (10, 'fantasy'),
 (11, 'fiction'),
 (12, 'history'),
 (13, 'horror'),
 (14, 'manga'),
 (15, 'memoir'),
 (16, 'music'),
 (17, 'mystery'),
 (18, 'nonfiction'),
 (19, 'paranormal'),
 (20, 'philosophy'),
 (21, 'poetry'),
 (22, 'psychology'),
 (23, 'religion'),
 (24, 'romance'),
 (25, 'science'),
 (26, 'spirituality'),
 (27, 'sports'),
 (28, 'suspense'),
 (29, 'thriller'),
 (30, 'travel')]
```

The Name

```
idx = 24  # romance
build_chart(list(available_genres.tag_name)[idx])[cols].head(15)
```

	title	authors	original_publication_year	average_rating	ratings_count	work_text
book_id						
136251	Harry Potter and the Deathly Hallows (Harry Po...	J.K. Rowling, Mary GrandPré	2007	4.61	1746574	
862041	Harry Potter Boxset (Harry Potter, #1-7)	J.K. Rowling	1998	4.74	190050	
1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling, Mary GrandPré	2005	4.54	1678823	
62291	A Storm of Swords (A Song of Ice and Fire, #3)	George R.R. Martin	2000	4.54	469022	
186074	The Name of the Wind (The Kingkiller Chronicle...	Patrick Rothfuss	2007	4.55	400101	
	The Wise Man's Fear					

Content Based Recommender

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
```

```
books['authors'] = books['authors'].apply(lambda x: [str.lower(i.replace(" ", "")) for i in x.split(', ')])
```

```
def get_genres(x):
    t = book_tags[book_tags.goodreads_book_id==x]
    return [i.lower().replace(" ", "") for i in tags.tag_name.loc[t.tag_id].values]
```

```
books['genres'] = books.book_id.apply(get_genres)
```

```
books['soup'] = books.apply(lambda x: ' '.join([x['title']] + x['authors'] + x['genres']), axis=1)
```

```
books.soup.head()
```

```
0    The Hunger Games (The Hunger Games, #1) suzann...
1    Harry Potter and the Sorcerer's Stone (Harry P...
2    Twilight (Twilight, #1) stepheniemeyer young-a...
3    To Kill a Mockingbird harperlee classics favor...
4    The Great Gatsby f.scottfitzgerald classics fa...
Name: soup, dtype: object
```

```
count = CountVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english')
count_matrix = count.fit_transform(books['soup'])
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
cosine_sim = cosine_similarity(count_matrix, count_matrix)
```

```
indices = pd.Series(books.index, index=books['title'])
titles = books['title']
```

```
def get_recommendations(title, n=10):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:31]
    book_indices = [i[0] for i in sim_scores]
    return list(titles.iloc[book_indices].values)[:n]
```

```
get_recommendations("The One Minute Manager")
```

```
["Good to Great: Why Some Companies Make the Leap... and Others Don't",
 "First, Break All the Rules: What the World's Greatest Managers Do Differently",
 'Execution: The Discipline of Getting Things Done',
 "What Got You Here Won't Get You There: How Successful People Become Even More Successful",
 'Start with Why: How Great Leaders Inspire Everyone to Take Action',
 'Great by Choice: Uncertainty, Chaos, and Luck--Why Some Thrive Despite Them All',
 'The 21 Irrefutable Laws of Leadership: Follow Them and People Will Follow You',
 'The Speed of Trust: The One Thing that Changes Everything',
 'Fish: A Proven Way to Boost Morale and Improve Results',
 'Leadership and Self-Deception: Getting Out of the Box']
```

```
def get_name_from_partial(title):
    return list(books.title[books.title.str.lower().str.contains(title) == True].values)
```

```
title = "business"
l = get_name_from_partial(title)
list(enumerate(l))
```

```
[(0, 'The Power of Habit: Why We Do What We Do in Life and Business'),
 (1,
  "The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses"),
 (2,
  'Caps for Sale: A Tale of a Peddler, Some Monkeys and Their Monkey Business'),
 (3,
  "The E-Myth Revisited: Why Most Small Businesses Don't Work and What to Do About It"),
 (4, 'The Snowball: Warren Buffett and the Business of Life'),
 (5,
  "The Innovator's Dilemma: The Revolutionary Book that Will Change the Way You Do Business (Collins Business Essentials)"),
 (6, 'The Intelligent Investor (Collins Business Essentials)'),
 (7, 'Purple Cow: Transform Your Business by Being Remarkable'),
 (8, 'Business Model Generation'),
 (9, 'The Long Tail: Why the Future of Business is Selling Less of More'),
 (10,
  "Losing My Virginity: How I've Survived, Had Fun, and Made a Fortune Doing Business My Way"),
 (11,
  'The Hard Thing About Hard Things: Building a Business When There Are No Easy Answers'),
 (12, 'Wicked Business (Lizzy & Diesel, #2)'),
 (13,
  'The Effective Executive: The Definitive Guide to Getting the Right Things Done (Harperbusiness Essentials)'),
 (14,
  'The Google Story: Inside the Hottest Business, Media, and Technology Success of Our Time'),
 (15, 'The Personal MBA: Master the Art of Business'),
 (16, 'Fifth Business'),
 (17,
  'The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win'),
 (18, 'Trouble Is My Business'),
 (19,
  'Amusing Ourselves to Death: Public Discourse in the Age of Show Business'),
 (20,
  'EntreLeadership: 20 Years of Practical Business Wisdom from the Trenches')]
```

```
get_recommendations(l[1])
```

```
['Rework',
 'The Hard Thing About Hard Things: Building a Business When There Are No Easy Answers',
 'Blue Ocean Strategy: How To Create Uncontested Market Space And Make The Competition Irrelevant',
 'The Art of the Start: The Time-Tested, Battle-Hardened Guide for Anyone Starting Anything',
 "Good to Great: Why Some Companies Make the Leap... and Others Don't",
 'Start with Why: How Great Leaders Inspire Everyone to Take Action',
 'Zero to One: Notes on Startups, or How to Build the Future',
 "The E-Myth Revisited: Why Most Small Businesses Don't Work and What to Do About It",
 'How Google Works',
 'Delivering Happiness: A Path to Profits, Passion, and Purpose']
```

Popularity and Ratings

```
def improved_recommendations(title, n=10):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:31]
    book_indices = [i[0] for i in sim_scores]
    df = books.iloc[book_indices][['title', 'ratings_count', 'average_rating', 'weighted_rating']]

    v = df['ratings_count']
    m = df['ratings_count'].quantile(0.60)
    R = df['average_rating']
    C = df['average_rating'].mean()
    df['weighted_rating'] = (R*v + C*m) / (v + m)

    qualified = df[df['ratings_count'] >= m]
    qualified = qualified.sort_values('weighted_rating', ascending=False)
    return qualified.head(n)

improved_recommendations("The One Minute Manager")
```

	title	ratings_count	average_rating	weighted_rating
2559	The 21 Irrefutable Laws of Leadership: Follow ...	30255	4.12	4.060190
246	The 7 Habits of Highly Effective People: Power...	314700	4.05	4.045478
3234	Start with Why: How Great Leaders Inspire Ever...	32899	4.07	4.035066
931	Good to Great: Why Some Companies Make the Lea...	85277	4.04	4.028535
2326	The Five Dysfunctions of a Team: A Leadership ...	40239	4.01	4.002625
2387	Delivering Happiness: A Path to Profits, Passi...	37601	4.01	4.002321
2413	The E-Myth Revisited: Why Most Small Businesse...	37671	3.98	3.984657
2219	Built to Last: Successful Habits of Visionary ...	39618	3.98	3.984520
3360	First, Break All the Rules: What the World's G...	27207	3.92	3.955049
989	Rework	88626	3.93	3.944028

```
improved_recommendations(l[1])
```

	title	ratings_count	average_rating	weighted_rating
2165	Zero to One: Notes on Startups, or How to Buil...	47807	4.17	4.097532

Collaborative Filtering

User Based

2219	Built to Last: Successful Habits of Visionary ...	39618	3.98	3.974785

```
!pip install surprise
```

```
Collecting surprise
  Downloading surprise-0.1-py2.py3-none-any.whl (1.8 kB)
Collecting scikit-surprise
  Downloading scikit-surprise-1.1.1.tar.gz (11.8 MB)
     |████████████████████████████████| 11.8 MB 28.2 MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.1.0)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.21.5)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.15.0)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp37-cp37m-linux_x86_64.whl size=1630169 sha256=ad486b84290c5694
  Stored in directory: /root/.cache/pip/wheels/76/44/74/b498c42be47b2406bd27994e16c5188e337c657025ab400c1c
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.1 surprise-0.1
```

```
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
```

```
reader = Reader()
data = Dataset.load_from_df(new_ratings[['user_id', 'book_id', 'rating']], reader)
```

```
svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'])
```

```
{'fit_time': (46.24783658981323,
 46.86831188201904,
 47.28285264968872,
 47.20978260040283,
 47.790398597717285),
'test_mae': array([0.65817603, 0.659029  , 0.65843088, 0.65842624, 0.65790762]),
'test_rmse': array([0.84179547, 0.84260203, 0.84182249, 0.8423519 , 0.840984  ]),
'test_time': (2.677412509918213,
 2.176800489425659,
 2.5482187271118164,
 2.661578893661499,
 2.601630926132202)}
```

```
trainset = data.build_full_trainset()
svd.fit(trainset);
```

```
new_ratings[new_ratings['user_id'] == 10]
```

	book_id	user_id	rating	title
150478	1506	10	4	The Zahir
282986	2833	10	4	The Prisoner of Heaven (The Cemetery of Forgot...
452158	4531	10	4	The Joke

```
svd.predict(10, 1506)
```

```
Prediction(uid=10, iid=1506, r_ui=None, est=3.4994336643958683, details={'was_impossible': False})
```

Item Based

```
bookmat = new_ratings.pivot_table(index='user_id', columns='title', values='rating')
bookmat.head()
```

10%
Happier:

```
def get_similar(title, mat):
    title_user_ratings = mat[title]
    similar_to_title = mat.corrwith(title_user_ratings)
    corr_title = pd.DataFrame(similar_to_title, columns=['correlation'])
    corr_title.dropna(inplace=True)
    corr_title.sort_values('correlation', ascending=False, inplace=True)
    return corr_title
```

#3) مع كف #2) You Losing Shot, Sonnets

```
title = "Twilight (Twilight, #1)"
smlr = get_similar(title, bookmat)
```

```
/usr/local/lib/python3.7/dist-packages/numpy/lib/function_base.py:2683: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.7/dist-packages/numpy/lib/function_base.py:2542: RuntimeWarning: divide by zero encountered in true_divide
  c *= np.true_divide(1, fact)
```

```
smlr.head(10)
```

```
smlr = smlr.join(books.set_index('title')['ratings_count'])
smlr.head()
```

title	correlation	ratings_count
'Salem's Lot	0.275938	228680
'Salem's Lot	0.275938	72797
11/22/63	0.431331	258464
13 Little Blue Envelopes (Little Blue Envelope, #1)	-0.500000	66950
1776	0.301511	130293

```
smlr[smlr.ratings_count > 5e5].sort_values('correlation', ascending=False).head(10)
```

title	correlation	ratings_count
Twilight (Twilight, #1)	1.000000	3866839
New Moon (Twilight, #2)	0.885400	1149630
The Selection (The Selection, #1)	0.866025	505340
Eclipse (Twilight, #3)	0.857845	1134511
Me Before You (Me Before You, #1)	0.771845	587647
Matched (Matched, #1)	0.707029	511815
Breaking Dawn (Twilight, #4)	0.689029	1070245
Bossypants	0.669954	506250
City of Bones (The Mortal Instruments, #1)	0.654081	1154031
The Perks of Being a Wallflower	0.574701	888806

Hybrid Recommender

```python
def hybrid(user_id, title, n=10):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:51]
    book_indices = [i[0] for i in sim_scores]

    df = books.iloc[book_indices][['book_id', 'title', 'original_publication_year', 'ratings_count', 'average_rating']]
    df['est'] = df['book_id'].apply(lambda x: svd.predict(user_id, x).est)
    df = df.sort_values('est', ascending=False)
    return df.head(n)
```

```python
hybrid(4, 'Eat, Pray, Love')
```

	book_id	title	original_publication_year	ratings_count	average_rating	est
382	1241	A Million Little Pieces	2003	184241	3.62	4.198591
4038	6365221	Mennonite in a Little Black Dress: A Memoir of...	2009	23096	3.17	3.914656
604	40173	Are You There, Vodka? It's Me, Chelsea	2007	127096	3.85	3.914656
3984	46190	Love Is a Mix Tape	2007	21971	3.83	3.914656
744	12868761	Let's Pretend This Never Happened: A Mostly Tr...	2012	118475	3.90	3.914656
4724	13642929	My Beloved World	2013	17742	4.03	3.914656
		Kabul Beauty				

```python
hybrid(10, 'Eat, Pray, Love')
```

	book_id	title	original_publication_year	ratings_count	average_rating	est
382	1241	A Million Little Pieces	2003	184241	3.62	4.102207
4038	6365221	Mennonite in a Little Black Dress: A Memoir of...	2009	23096	3.17	3.866401
604	40173	Are You There, Vodka? It's Me, Chelsea	2007	127096	3.85	3.866401
3984	46190	Love Is a Mix Tape	2007	21971	3.83	3.866401
744	12868761	Let's Pretend This Never Happened: A Mostly Tr...	2012	118475	3.90	3.866401

improved_hybrid = (content based+user based)/2

Kabul Beauty

```
def improved_hybrid(user_id, title, n=10):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:51]
    book_indices = [i[0] for i in sim_scores]

    df = books.iloc[book_indices][['book_id', 'title', 'ratings_count', 'average_rating', 'original_publication_year']]
    v = df['ratings_count']
    m = df['ratings_count'].quantile(0.60)
    R = df['average_rating']
    C = df['average_rating'].mean()
    df['weighted_rating'] = (R*v + C*m) / (v + m)

    df['est'] = df['book_id'].apply(lambda x: svd.predict(user_id, x).est)

    df['score'] = (df['est'] + df['weighted_rating']) / 2
    df = df.sort_values('score', ascending=False)
    return df[['book_id', 'title', 'original_publication_year', 'ratings_count', 'average_rating', 'score']].head(n)
```

```
improved_hybrid(4, 'Eat, Pray, Love')
```

	book_id	title	original_publication_year	ratings_count	average_rating	score
328	2318271	The Last Lecture	2008	241869	4.25	4.056910
80	7445	The Glass Castle	2005	621099	4.24	4.014463
198	12691	Marley and Me: Life and Love With the World's ...	2005	367304	4.12	4.005892
1669	104189	Same Kind of Different as Me	2005	52964	4.21	3.992597
2803	18039963	A House in the Sky	2013	29369	4.20	3.966061
753	6366437	Half Broke Horses	2008	110597	4.05	3.960136
1067	29209	The Color of Water: A Black Man's Tribute to H...	1996	80906	4.06	3.958000
6286	8564644	Little Princes: One Man's Promise to Bring Hom...	2010	14765	4.25	3.947853
2701	6114607	The Midwife: A Memoir of Birth, Joy, and Hard ...	2002	19176	4.17	3.944820
4593	31845516	Love Warrior	2016	20094	4.10	3.934285

```
improved_hybrid(10, 'Eat, Pray, Love')
```

	book_id	title	original_publication_year	ratings_count	average_rating	score
328	2318271	The Last Lecture	2008	241869	4.25	4.032783
80	7445	The Glass Castle	2005	621099	4.24	4.026636
198	12691	Marley and Me: Life and Love With the World's ...	2005	367304	4.12	3.981764
1669	104189	Same Kind of Different as Me	2005	52964	4.21	3.968470
2803	18039963	A House in the Sky	2013	29369	4.20	3.941933
753	6366437	Half Broke Horses	2008	110597	4.05	3.936009
1067	29209	The Color of Water: A Black Man's Tribute to	1996	80906	4.06	3.933873