**EDS Theory Assignment**

**Name : Akshada Arjun Sonawane**

**Class : ET2**

**Roll no : ET2-30**

**PRN : 202401070123**

**GOODREADS**

**Real life problem statements:**

1. **Identify the author whose books have collectively received the most ratings on Goodreads.**

**author\_ratings = df.groupby('authors')['ratings\_count'].sum().sort\_values(ascending=False)**

**print(author\_ratings.head(1))**

1. **Find the 5 books with the highest average rating on Goodreads (minimum 1000 ratings to avoid obscure books).**

**top\_books = df[df['ratings\_count'] > 1000].sort\_values('average\_rating', ascending=False).head(5)**

**print(top\_books[['title', 'average\_rating', 'ratings\_count']])**

**3. Find all books published between 2000 and 2010 that have an average rating above 4.3.**

**books\_2000s = df[(df['publication\_year'] >= 2000) & (df['publication\_year'] <= 2010) & (df['average\_rating'] > 4.3)]**

**print(books\_2000s[['title', 'publication\_year', 'average\_rating']])**

**4. Calculate the average number of text reviews written per book across Goodreads.**

**avg\_text\_reviews = df['text\_reviews\_count'].mean()**

**print(avg\_text\_reviews)**

**5. List the top 10 most-reviewed books (based on text\_reviews\_count).**

**most\_reviewed\_books = df.sort\_values('text\_reviews\_count', ascending=False).head(10)**

**print(most\_reviewed\_books[['title', 'text\_reviews\_count']])**

**6. Find the most common genre on Goodreads among all listed books.**

**genres\_split = df['genres'].str.split(',').explode()**

**most\_common\_genre = genres\_split.value\_counts().idxmax()**

**print(most\_common\_genre)**

**7. Find books that are considered "hidden gems" — high average rating (>4.5) but less than 500 ratings.**

**hidden\_gems = df[(df['average\_rating'] > 4.5) & (df['ratings\_count'] < 500)]**

**print(hidden\_gems[['title', 'average\_rating', 'ratings\_count']])**

**8. Find the average number of pages for books rated above 4.5 on Goodreads.**

**avg\_pages\_high\_rating = df[df['average\_rating'] > 4.5]['num\_pages'].mean()**

**print(avg\_pages\_high\_rating)**

**9. Which year had the most book publications according to Goodreads data?**

**most\_books\_year = df['publication\_year'].value\_counts().idxmax()**

**print(most\_books\_year)**

**10. Identify books where the number of text reviews is more than 10% of the number of ratings (high engagement).**

**high\_engagement\_books = df[df['text\_reviews\_count'] > 0.1 \* df['ratings\_count']]**

**print(high\_engagement\_books[['title', 'text\_reviews\_count', 'ratings\_count']])**

**11. Find the author who appears most frequently on Goodreads (most books listed).**

**most\_listed\_author = df['authors'].value\_counts().idxmax()**

**print(most\_listed\_author)**

**12. Find the correlation between the number of ratings and the number of text reviews.**

**correlation = df['ratings\_count'].corr(df['text\_reviews\_count'])**

**print(correlation)**

**13. Create a new column showing how popular a book is: ratings\_count / num\_pages.**

**df['popularity\_score'] = df['ratings\_count'] / df['num\_pages']**

**print(df[['title', 'popularity\_score']])**

**14. Find books with less than 100 pages but more than 1000 ratings (short but famous books).**

**short\_popular\_books = df[(df['num\_pages'] < 100) & (df['ratings\_count'] > 1000)]**

**print(short\_popular\_books[['title', 'num\_pages', 'ratings\_count']])**

**15. Analyze the trend: average Goodreads rating by decade.**

**df['decade'] = (df['publication\_year'] // 10) \* 10**

**avg\_rating\_decade = df.groupby('decade')['average\_rating'].mean()**

**print(avg\_rating\_decade)**

**16. Find books whose ratings drastically dropped compared to other books by the same author.**

**author\_mean\_rating = df.groupby('authors')['average\_rating'].transform('mean')**

**df['rating\_gap'] = author\_mean\_rating - df['average\_rating']**

**fallen\_books = df[df['rating\_gap'] > 0.5]**

**print(fallen\_books[['title', 'authors', 'rating\_gap']])**

**17. Find the top 5 books from the 'Fantasy' genre based on ratings.**

**fantasy\_books = df[df['genres'].str.contains('Fantasy', na=False)]**

**top\_fantasy = fantasy\_books.sort\_values('average\_rating', ascending=False).head(5)**

**print(top\_fantasy[['title', 'average\_rating']])**

**18. Identify authors who have books spanning over more than 3 different decades.**

**author\_decades = df.groupby('authors')['decade'].nunique()**

**long\_career\_authors = author\_decades[author\_decades > 3]**

**print(long\_career\_authors)**

**19. Find books that were published in the same year but have very different average ratings (rating spread > 1).**

**rating\_spread = df.groupby('publication\_year')['average\_rating'].agg(lambda x: x.max() - x.min())**

**years\_high\_spread = rating\_spread[rating\_spread > 1]**

**print(years\_high\_spread)**

**20. Identify the most "overrated" books — high ratings but low number of text reviews (maybe fake ratings).**

**overrated\_books = df[(df['average\_rating'] > 4.5) & (df['text\_reviews\_count'] < 50)]**

**print(overrated\_books[['title', 'average\_rating', 'text\_reviews\_count']])**