Movielens dataset

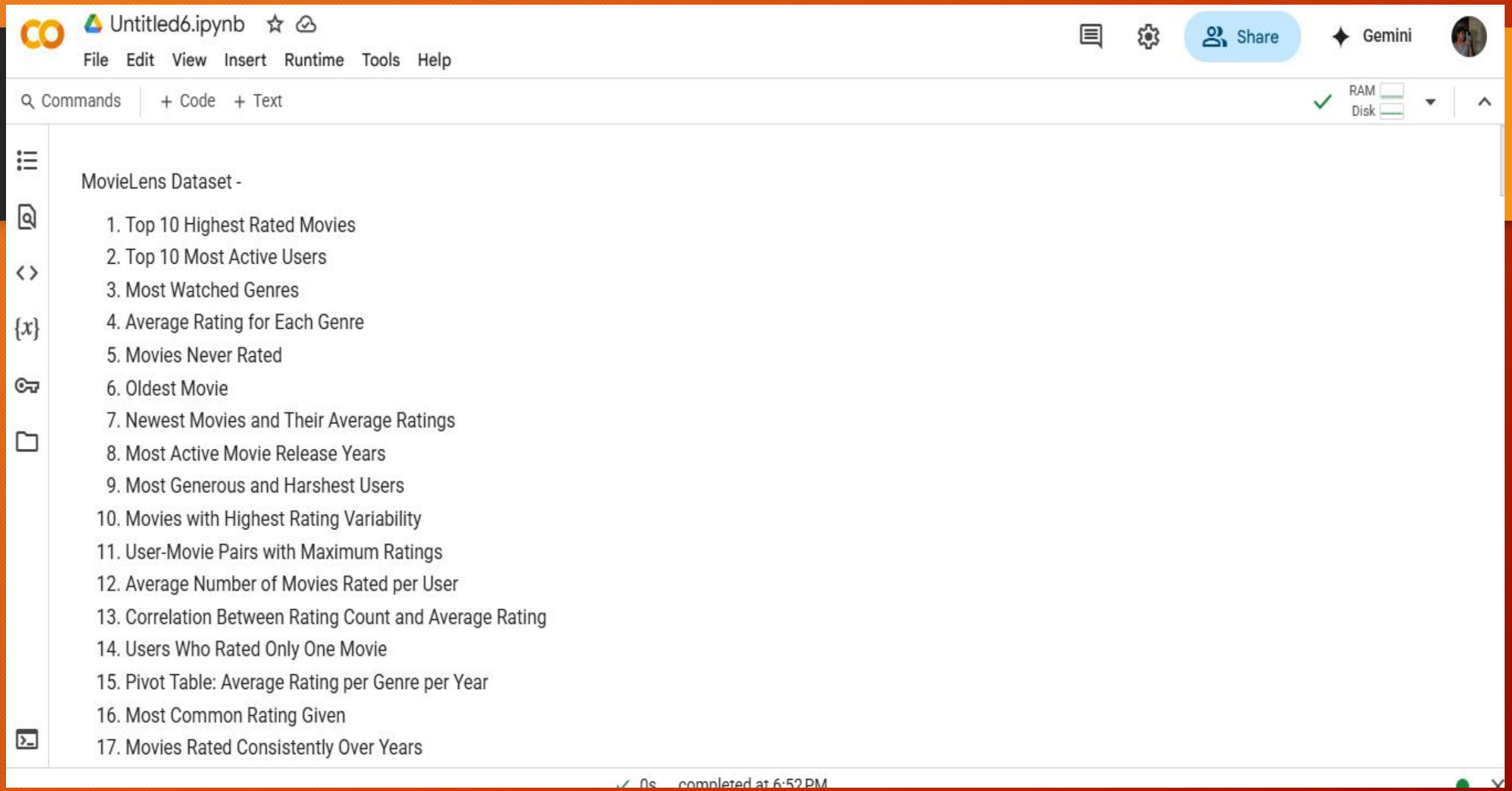
NAME - YASH KEDARE

ROLL NO.- ET1-76

Prn- 202401070155

GOOGLE COLAB LINK –

https://drive.google.com/file/d/1G1qOKL8tHnKAzWO2OA6pjEDWvvilPqGY/view?usp=drivesdk


Untitled6.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
RAM
Disk
MovieLens Dataset -
1. Top 10 Highest Rated Movies
2. Top 10 Most Active Users
3. Most Watched Genres
4. Average Rating for Each Genre
5. Movies Never Rated
6. Oldest Movie
7. Newest Movies and Their Average Ratings
8. Most Active Movie Release Years
9. Most Generous and Harshest Users
10. Movies with Highest Rating Variability
11. User-Movie Pairs with Maximum Ratings
12. Average Number of Movies Rated per User
13. Correlation Between Rating Count and Average Rating
14. Users Who Rated Only One Movie
15. Pivot Table: Average Rating per Genre per Year
16. Most Common Rating Given
17. Movies Rated Consistently Over Years
0s completed at 6:52 PM

Untitled6.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

17. Movies Rated Consistently Over Years
18. Genre with Most Improved Rating Over Years
19. Cluster-like Grouping: Movies by Average Rating and Rating Count
20. Top 5 Directors with Highest Average Rating

```python
import pandas as pd

# Example data for movies and ratings
movies = pd.DataFrame({
    'movieId': [1, 2, 3, 4],
    'title': ['Toy Story (1995)', 'Jumanji (1995)', 'Grumpier Old Men (1995)', 'Waiting to Exhale (1995)'],
    'genres': ['Adventure|Animation|Children|Comedy|Fantasy', 'Adventure|Children|Fantasy', 'Comedy|Romance', 'Comedy|Drama|Romance']
})

ratings = pd.DataFrame({
    'userId': [1, 2, 3, 1, 2, 3, 4],
    'movieId': [1, 1, 1, 2, 2, 3, 4],
    'rating': [4.0, 5.0, 4.5, 3.0, 3.5, 2.0, 4.0],
    'timestamp': [964982703, 964981247, 964982224, 964983815, 964982931, 964982400, 964982176]
})

# 1. Top 10 Highest Rated Movies
avg_ratings = ratings.groupby('movieId')['rating'].mean()
rating_counts = ratings.groupby('movieId')['rating'].count()
movies['avg_rating'] = movies['movieId'].map(avg_ratings)
```

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
movies['avg_rating'] = movies['movieId'].map(avg_ratings)
movies['rating_count'] = movies['movieId'].map(rating_counts)
top10_movies = movies[movies['rating_count'] >= 100].sort_values('avg_rating', ascending=False).head(10)
print("Top 10 Highest Rated Movies:\n", top10_movies)

# 2. Top 10 Most Active Users
top_users = ratings.groupby('userId')['rating'].count().sort_values(ascending=False).head(10)
print("Top 10 Most Active Users:\n", top_users)

# 3. Most Watched Genres
movies['genres'] = movies['genres'].str.split('|')
movies_genre = movies.explode('genres')
merged = pd.merge(ratings, movies_genre, on='movieId')
most_watched_genres = merged['genres'].value_counts()
print("Most Watched Genres:\n", most_watched_genres)

# 4. Average Rating for Each Genre
avg_genre_rating = merged.groupby('genres')['rating'].mean().sort_values(ascending=False)
print("Average Rating for Each Genre:\n", avg_genre_rating)

# 5. Movies Never Rated
rated_movie_ids = ratings['movieId'].unique()
never_rated_movies = movies[~movies['movieId'].isin(rated_movie_ids)]
print("Movies Never Rated:\n", never_rated_movies)

# 6. Oldest Movie
```

0s completed at 6:52 PM

Untitled6.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
# 6. Oldest Movie
movies['year'] = movies['title'].str.extract(r'(\d{4})')
oldest_movie = movies.sort_values('year').head(1)
print("Oldest Movie:\n", oldest_movie)

# 7. Newest Movies and Their Average Ratings
newest_movies = movies.sort_values('year', ascending=False).head(10)
newest_movie_avg_ratings = ratings.groupby('movieId')['rating'].mean()
newest_movies['avg_rating'] = newest_movies['movieId'].map(newest_movie_avg_ratings)
print("Newest Movies and Their Average Ratings:\n", newest_movies)

# 8. Most Active Movie Release Years
year_counts = movies['year'].value_counts().sort_values(ascending=False)
print("Most Active Movie Release Years:\n", year_counts)

# 9. Most Generous and Harshest Users
user_avg_rating = ratings.groupby('userId')['rating'].mean()
most_generous_users = user_avg_rating.sort_values(ascending=False).head(5)
most_harsh_users = user_avg_rating.sort_values().head(5)
print("Most Generous Users:\n", most_generous_users)
print("Most Harsh Users:\n", most_harsh_users)

# 10. Movies with Highest Rating Variability
rating_std = ratings.groupby('movieId')['rating'].std()
movies['rating_std'] = movies['movieId'].map(rating_std)
```

Untitled6.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
highest_variability_movies = movies.sort_values('rating_std', ascending=False).head(10)
print("Movies with Highest Rating Variability:\n", highest_variability_movies)

# 11. User-Movie Pairs with Maximum Ratings
max_rating = ratings['rating'].max()
max_rating_pairs = pd.merge(ratings[ratings['rating'] == max_rating], movies, on='movieId')
print("User-Movie Pairs with Maximum Ratings:\n", max_rating_pairs)

# 12. Average Number of Movies Rated per User
avg_movies_per_user = ratings.groupby('userId')['movieId'].count().mean()
print("Average Number of Movies Rated per User:", avg_movies_per_user)

# 13. Correlation Between Rating Count and Average Rating
movie_rating_count = ratings.groupby('movieId')['rating'].count()
movie_avg_rating = ratings.groupby('movieId')['rating'].mean()
movies['rating_count'] = movies['movieId'].map(movie_rating_count)
movies['avg_rating'] = movies['movieId'].map(movie_avg_rating)
correlation = movies['rating_count'].corr(movies['avg_rating'])
print("Correlation Between Rating Count and Average Rating:", correlation)

# 14. Users Who Rated Only One Movie
user_rating_count = ratings.groupby('userId')['rating'].count()
single_rating_users = user_rating_count[user_rating_count == 1]
print("Users Who Rated Only One Movie:\n", single_rating_users)

# 15. Pivot Table: Average Rating per Genre per Year
```

Untitled6.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
# 15. Pivot Table: Average Rating per Genre per Year
pivot_data = merged.copy()
pivot_data['year'] = pivot_data['title'].str.extract(r'(\d{4})')
pivot_table = pivot_data.pivot_table(values='rating', index='genres', columns='year', aggfunc='mean')
print("Pivot Table: Average Rating per Genre per Year:\n", pivot_table)

# 16. Most Common Rating Given
most_common_rating = ratings['rating'].value_counts().idxmax()
print("Most Common Rating Given:", most_common_rating)

# 17. Movies Rated Consistently Over Years
ratings['year'] = pd.to_datetime(ratings['timestamp'], unit='s').dt.year
ratings_per_year = ratings.groupby(['movieId', 'year']).size().unstack(fill_value=0)
consistent_movies = ratings_per_year[ratings_per_year.gt(0).sum(axis=1) >= 5]
print("Movies Rated Consistently Over Years:\n", consistent_movies)

# 18. Genre with Most Improved Rating Over Years
genre_year_pivot = pivot_data.pivot_table(values='rating', index='genres', columns='year', aggfunc='mean')
genre_improvement = genre_year_pivot.iloc[:, -1] - genre_year_pivot.iloc[:, 0]
most_improved_genre = genre_improvement.sort_values(ascending=False).head(1)
print("Genre with Most Improved Rating Over Years:\n", most_improved_genre)

# 19. Cluster-like Grouping: Movies by Average Rating and Rating Count
movie_stats = ratings.groupby('movieId').agg({'rating': ['mean', 'count']})
movie_stats.columns = ['avg_rating', 'rating_count']
movies = movies.merge(movie_stats, on='movieId')
print("Cluster-like Grouping: Movies by Average Rating and Rating Count:\n", movies)
```

Untitled6.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
# 20. Top 5 Directors with Highest Average Rating (assuming 'director' column exists)
# Assuming the 'director' column exists
# For now, I'll comment it out since no director data is available
# director_avg_rating = merged.groupby('director')['rating'].mean().sort_values(ascending=False).head(5)
# print("Top 5 Directors with Highest Average Rating:\n", director_avg_rating)
```

```
   rating_count  year  rating_std
0             3  1995    0.500000
1             2  1995    0.353553
2             1  1995         NaN
3             1  1995         NaN
User-Movie Pairs with Maximum Ratings:
    userId  movieId  rating  timestamp            title  \
0       2        1     5.0  964981247  Toy Story (1995)

                                        genres  avg_rating  \
0  [Adventure, Animation, Children, Comedy, Fantasy]         4.5

   rating_count  year  rating_std
0             3  1995         0.5
Average Number of Movies Rated per User: 1.75
Correlation Between Rating Count and Average Rating: 0.6203723351351519
Users Who Rated Only One Movie:
 userId
4    1
Name: rating, dtype: int64
Pivot Table: Average Rating per Genre per Year:
```

0s completed at 6:52 PM

Untitled6.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
Pivot Table: Average Rating per Genre per Year:
 year       1995
genres
Adventure   4.0
Animation   4.5
Children    4.0
Comedy      3.9
Drama       4.0
Fantasy     4.0
Romance     3.0
Most Common Rating Given: 4.0
Movies Rated Consistently Over Years:
 Empty DataFrame
Columns: [2000]
Index: []
Genre with Most Improved Rating Over Years:
 genres
Adventure    0.0
Name: 1995, dtype: float64
Cluster-like Grouping: Movies by Average Rating and Rating Count:
    movieId                    title  \
0        1          Toy Story (1995)
1        2            Jumanji (1995)
2        3   Grumpier Old Men (1995)
3        4  Waiting to Exhale (1995)

                                       genres  avg_rating_x  \
0  [Adventure, Animation, Children, Comedy, Fantasy]          4.50
1                    [Adventure, Children, Fantasy]          3.25
```