Demographic Filtering

What is our GOAL for this MODULE?

The goal of this module is to understand how demographic filtering is done and then perform it.

What did we ACHIEVE in the class TODAY?

- Understood how demographic filtering is done
- Learned about Weighted Rating
- Performed demographic filtering

Which CONCEPTS/CODING BLOCKS did we cover today?

- Demographic Filtering
- Weighted Rating
- Plotly
- Pandas DataFrame

How did we DO the activities?

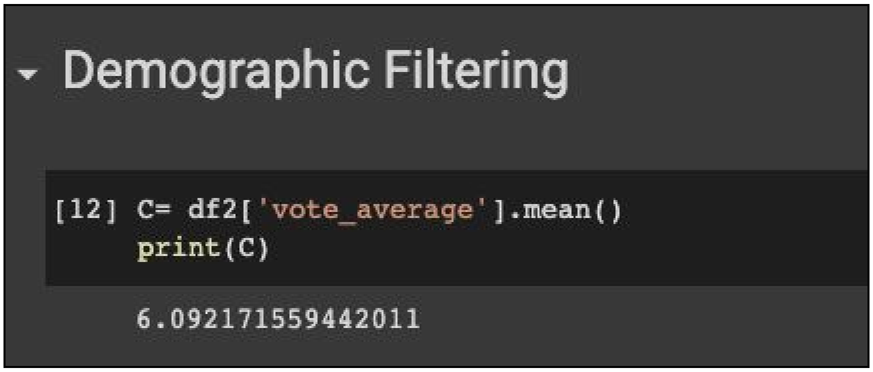
1. Steps to perform demographic filtering
    - **We need a metric or score to rate movies**
    - **We then need to calculate the score of every movie**
    - **We finally need to sort the scores and recommend the best rated movie to the users**
2. It can be said that we can use average ratings of the movie as the score but using this would not be fair. There might be a movie with 8.9 rating and 3 votes but it cannot be considered better than a movie with 7.8 rating and 40 votes.
3. For this, IMDb has created a formula! It is known as weighted rating and it is famously used in the industry for the same thing, to get a score for their products/items.

$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m} . R\right) + \left(\frac{m}{v+m} . C\right)$$

- **v -** The number of votes for the movies (or number of ratings/reviews in case of an amazon product)
- **m -** The minimum votes required to be listed in the chart
- **R -** Average rating of the movie
- **C -** Mean votes across the whole report

4. Calculate the mean of all the vote averages.

```
C= df2['vote_average'].mean()
print(C)
```



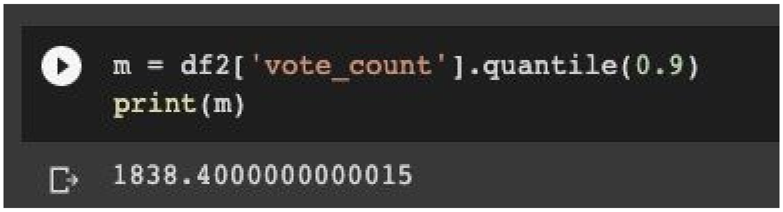
Demographic Filtering

```
[12] C= df2['vote_average'].mean()
     print(C)

     6.092171559442011
```

5. Find out the appropriate formula for **m.**

```
m = df2['vote_count'].quantile(0.9)
print(m)
```

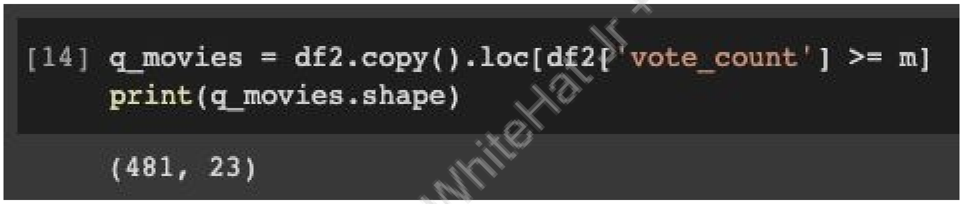


```
m = df2['vote_count'].quantile(0.9)
print(m)

1838.4000000000015
```

6. Create a dataframe that has all the movies whose number of votes are more than **m.**

```
q_movies = df2.copy().loc[df2['vote_count'] >= m]
print(q_movies.shape)
```



```
[14] q_movies = df2.copy().loc[df2['vote_count'] >= m]
     print(q_movies.shape)

     (481, 23)
```

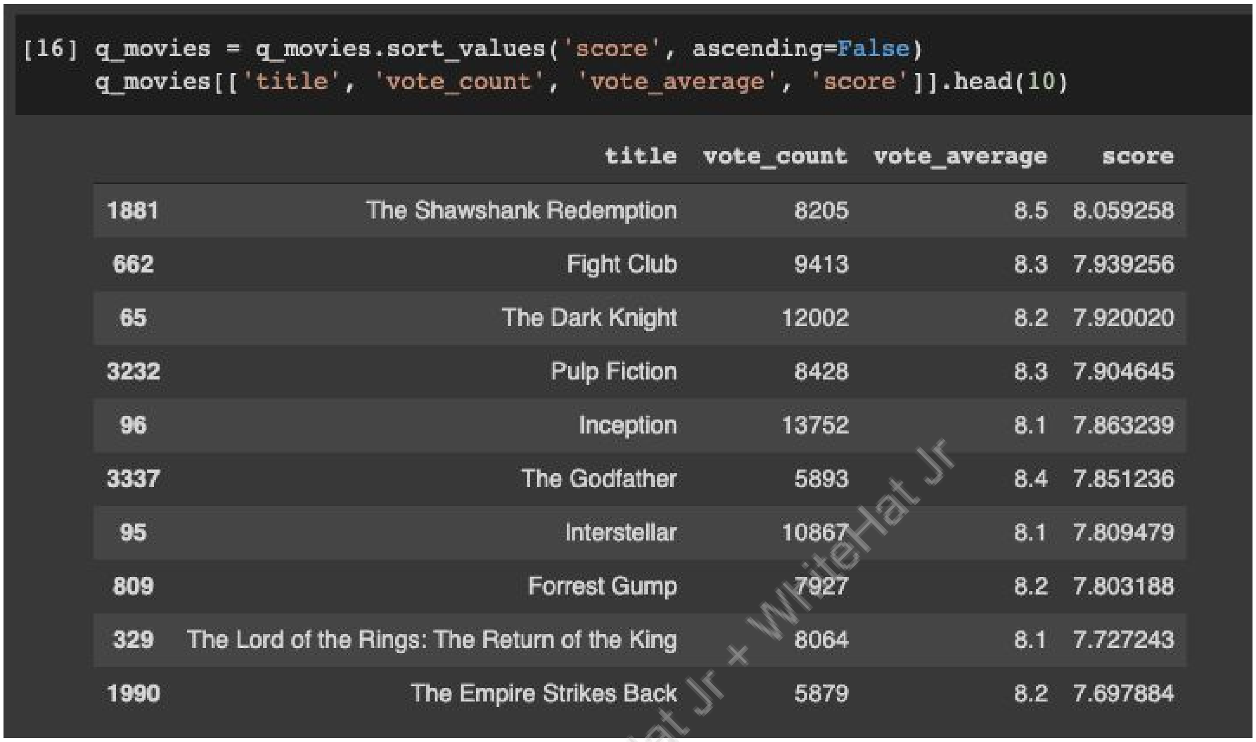
7. Create a function **weighted rating**, calculate the score for all the movies with the formula and add the score into all the rows in the dataframe **q_movies.**

```
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)

q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
```

8. Sort the dataframe based on the score value in descending order.

```
q_movies = q_movies.sort_values('score', ascending=False)
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```



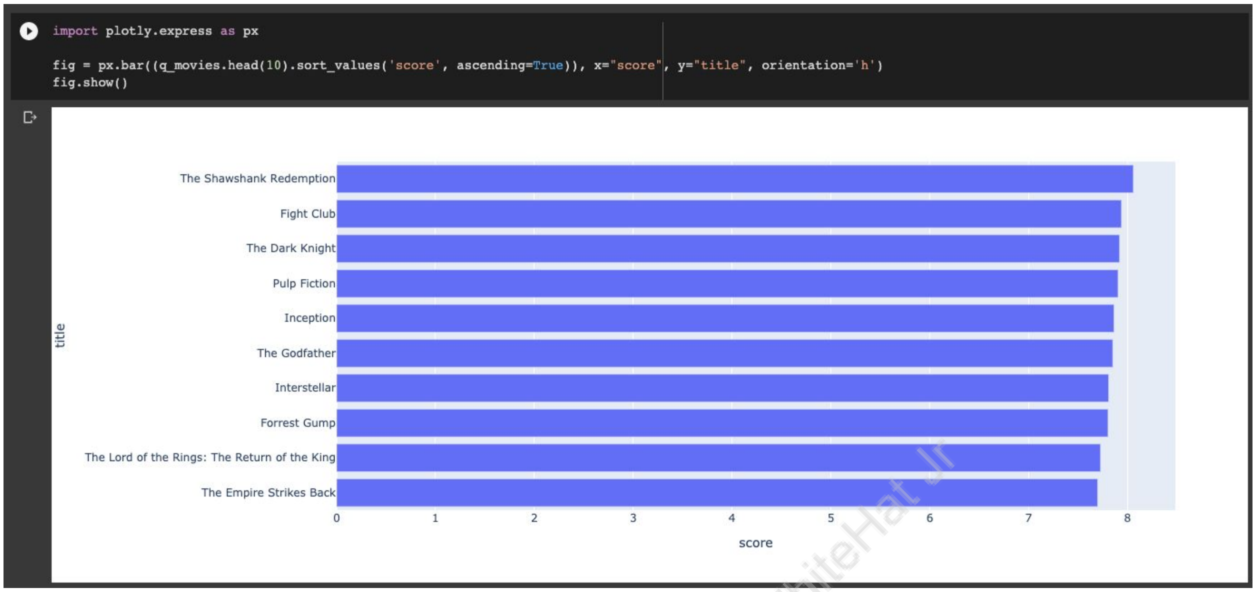
```
[16] q_movies = q_movies.sort_values('score', ascending=False)
     q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

```
                                            title  vote_count  vote_average     score
1881                     The Shawshank Redemption        8205           8.5  8.059258
662                                    Fight Club        9413           8.3  7.939256
65                                The Dark Knight       12002           8.2  7.920020
3232                                 Pulp Fiction        8428           8.3  7.904645
96                                      Inception       13752           8.1  7.863239
3337                                The Godfather        5893           8.4  7.851236
95                                   Interstellar       10867           8.1  7.809479
809                                  Forrest Gump        7927           8.2  7.803188
329   The Lord of the Rings: The Return of the King        8064           8.1  7.727243
1990                      The Empire Strikes Back        5879           8.2  7.697884
```

9. Plot a horizontal bar chart on the top 10 movies.

```
import plotly.express as px

fig = px.bar((q_movies.head(10).sort_values('score',
ascending=True)), x="score", y="title", orientation='h')
fig.show()
```



```
import plotly.express as px

fig = px.bar((q_movies.head(10).sort_values('score', ascending=True)), x="score", y="title", orientation='h')
fig.show()
```

What's NEXT?

In the next class, we will work on Content Based Filtering with this data.