Topic	Demographic Filtering	
Class Description	**Students will write a movie recommendation algorithm based on Demographic Filtering.**	
Class	**C139**	
Class time	**45 mins**	
Goal	● Understand the concept of weighted rating ● Write an algorithm for movie recommendation	

Resources Required	● Teacher Resources
	  ○ Google Colab
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	
	 ● Student Resources
	  ○ Google Colab
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen	

Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **15 min** **20 min** **5 min**

CONTEXT ● **Review the concepts learned in the earlier classes**		

Class Steps	Teacher Action	Student Action
Step 1: **Warm Up** **(5 mins)**	Hi <Student Name>! We have prepared our TMDBs data to write movie recommendation algorithms on it!	**ESR:** - We imported data directly from Kaggle into Google Colab and performed terminal commands on

	Can you tell me what all we did in the last class?	Google Colab. We then studied the data and merged it together.
	Yes! Can you also tell me the different types of filters used for movie recommendation systems?	**ESR:** - Demographic Filtering - Content Based Filtering - Collaborative Filtering
	I have an exciting quiz question for you! Are you ready to answer this question? Teacher click on the **Quiz Time** button on the bottom right corner of your screen to start the In-Class Quiz. A quiz will be visible to both you and the student. Encourage the student to answer the quiz question. The student may choose the wrong option, help the student to think correctly about the question and then answer again. After the student selects the correct option, the **End Quiz** button will start appearing on your screen. Click the End quiz to close the quiz pop-up and continue the class.	

	Awesome! Out of all these algorithms we have discussed, we will today perform demographic filtering on our data. Are you excited?	**ESR:** "Yes"
	Let's get started!	

CHALLENGE ● **Make student understand what weighted rating is** ● **Make student understand how is demographic filtering done**		
Step 2: Teacher-led Activity (15 min)	First of all, what does the word demographic mean?	**ESR:** It is referred to a particular sector of a population!
	Great! But when we do demographic filtering, what we want to achieve is to offer generalized recommendations to every user. There are a few things that we need to know before we get started. They are as follows: ● **We need a metric or score to rate movie** ● **We then need to calculate the score for every movie** ● **We finally need to sort the scores and recommend the best rated movie to the users**	**ESR:** varied

	Okay, let's tackle this point by point. We first need a metric or a score to rate the movie. What do you think we can possibly go with?	**ESR:** varied
	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. 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. It goes like this: $$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m} \cdot R\right) + \left(\frac{m}{v+m} \cdot C\right)$$ Here, it says ((v/(v+m))*R)+((m/(v+m))*C) Here, **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	**ESR:** Using Game States

	● **C -** Mean votes across the whole report If we look at our data, we already have v(vote count) and R(vote_average)! We can calculate the C by calculating the mean of all the vote averages. We will do that later!	
	Next, we want to determine an appropriate value for **m, the minimum votes** required to be listed in the chart. Do you know how we can do that?	**ESR:** Varied
	We can take the 90th Percentile as our cutoff. In other words, for a movie to feature in the charts, it must have more votes than at least 90% of the rest of the movies in the list. This is called a quantile. In future you will be writing competitive exams where this quantile concept will come to place. For instance, if you get 90% quantile then it means, you were better than 90% of the other students who appeared for the exam. The way we do this is by using the **quantile()** function to a dataframe. We will pass **0.9** as an argument to this function.	

	Great! Now we are ready to start coding!	-

Teacher Stops Screen Share

	Now it's your turn. Please share your screen with me.	

- **Ask Student to press ESC key to come back to panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

ACTIVITY
- **Students write code to apply demographic filtering on the data!**

	Note: **You need to run all the previous cells of your google colab again!**	

Step 3: Student-Led Activity (20 min)	We will first start by finding the **C, The mean of votes.**
	
	``python
	C= df2['vote_average'].mean()
	print(C)
	``
	
	Here, we are simply calculating the mean of the vote averages of all the movies in our **df2** variable and storing it in a variable **C.**
		Student codes to find the value of C.

```
▾ Demographic Filtering

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

     6.092171559442011
```

	So, now we know that the mean rating for all the movies is approximately 6 out of 10. Now, let's find an appropriate value for **m.** ``` m = df2['vote_count'].quantile(0.9) print(m) ``` Here, we are finding the quantile 0.9 (a value more than that of 90% of the movies in terms of vote count) and storing it in a variable **m.**	*Student codes to find the value of **m.***

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

⊡  1838.4000000000015
```

	Now that we have this number, let's create a new dataframe with all the movies that have more votes than the value of **m**!	*Student codes to create the new dataframe.*

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

Here, we are using a function known as **copy()** to copy the contents of **df2**, we are then applying a condition on the copied contents with the **loc()** function and we are saying that the **vote_count** of the movie shall be equal to or greater than **m**. We are then storing this new dataframe into a variable **q_movies**.

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

     (481, 23)
```

We can see that there are 481 movies which qualify to be in this list. Now, we need to calculate the metric/score for all the movies.

For this, we will define a function **weighted_rating()** and create a new column **"score"** in all the rows.

```
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)
```

The student codes the function and uses it to add a new column **score** in **q_movies**.

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

Here, we have defined a function named **weighted_rating** in which we are passing our individual row, along with m and C values for the formula.

We are then storing the **vote_count** in variable **v** and **vote_average** in variable **R.**

We are finally applying the formula for weighted rating with the values we have and returning the output. We are storing this output in the **score** column of the dataframe.

We have used the **apply** function to use the formula because we are dealing with a DataFrame and we need to pass individual rows to our function. **apply()** function takes care of that.

```
[15] 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)
```

	Great! Now, we have to sort the DataFrame on the **score** column. We will also output the **title, vote_count, vote_average & score/weighted rating** of the top 10 movies. ```\nq_movies =\nq_movies.sort_values('score',\nascending=False)\nq_movies[['title', 'vote_count',\n'vote_average',\n'score']].head(10)\n``` Here, we are just sorting the values using the **sort_values()** function of a Dataframe based on the **score** column that we just created in descending order, by adding the attribute **ascending=False**. We are then displaying the head (10) of our data with the **title, vote_count, vote_average** and **score** columns.	*Student codes to sort the DataFrame based on the **score** column and then display the top 10 movies.*

```
[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

Hurray! We made our first basic recommender!

Many times, you may see a **Trending Now** tab. Now you know how they might be knowing what to display to the user in here and recommend great stuff!

Let's plot a graph on our top 10 movies! We will plot a bar chart (horizontal) with the name of the movie as the Y axis and the score on the X axis.

```
import plotly.express as px

fig =
px.bar((q_movies.head(10).sort_v
```

Student codes to plot the chart.

```
alues('score', ascending=True)),
x="score", y="title",
orientation='h')
fig.show()
```
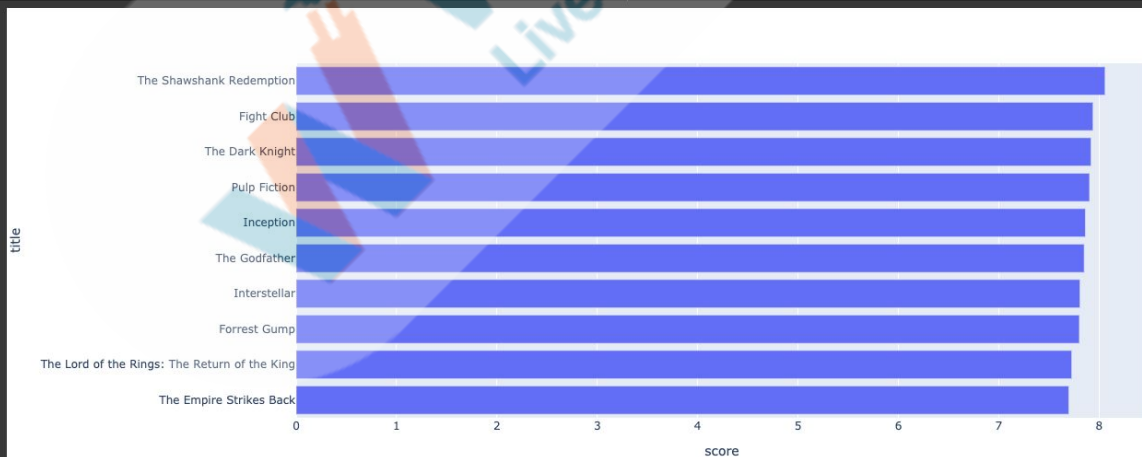
Here, we are using the plotly's library to plot a bar chart.

While passing our dataframe into the **bar()** function, we are taking only the top 10 movies (with the head function) and we are sorting its values in ascending order this time. However, this is not necessary as it only helps in displaying the movie with the highest score at the top of the chart.

We are using **orientation=h** to make the chart horizontal.

```
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()
```

	There, that's how we do demographic filtering!	

FEEDBACK
- **Appreciate the student for their efforts**
- **Identify 2 strengths and 1 area of progress for the student**

Step 4: Wrap-Up (5 min)	So, in this project class we understood how demographic filtering is done and how we can calculate the weighted rating of any kind of data to build a basic recommendation system. Congratulations! You built your first recommendation system! How was your experience?	**ESR:** varied
	Amazing. Now, something to keep in mind is that these demographic recommenders provide a general chart of recommended movies to all the users. They are not sensitive to the interest and tastes of a particular user. This is where a more refined system - **Content Based Filtering** comes into play.	-

	Next class would be based just on that!	
Teacher Clicks	✖ End Class	

Activity	Activity Name	Links
Teacher Activity 1	Solution	https://colab.research.google.com/drive/1vPzsPqXO_Jm0C87BLjX0J1oqRku__VMy?usp=sharing