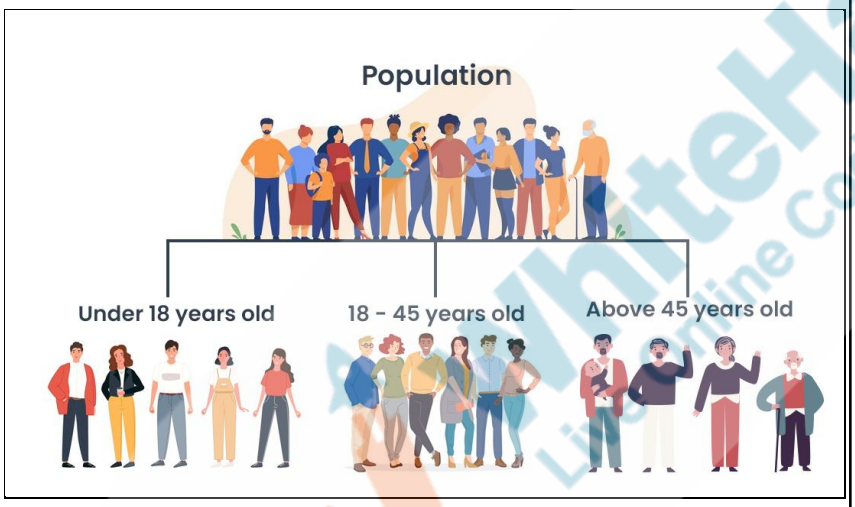
Topic	DEMOGRAPHIC FILTERING	
Class Description	**The student will clean the dataset and write a movie recommendation algorithm based on Demographic Filtering.**	
Class	**PRO C139**	
Class time	**45 mins**	
Goal	• Clean the dataset. • Understand the concept of weighted rating. • Write an algorithm for movie recommendation.	
Resources Required	• Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone • Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher-Led Activity 1** **Student-Led Activity 1** **Wrap-Up**	**5 mins** **15 mins** **20 mins** **5 mins**
WARM-UP SESSION - 5 mins		
Teacher Starts Slideshow  **Slide # to #** <**Note**: Only Applicable for Classes with VA> Refer to speaker notes and follow the instructions on each slide.		

Teacher Action	Student Action
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes, I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide # to #** <**Note**: Only Applicable for Classes with VA>	
Activity Details **Following are the session deliverables:** • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
Teacher Action	**Student Action**
What do you understand by the term **demographic filtering**? ***Note:*** *Encourage the student to give answers and connect the answer with today's topic.* Great, if you look through my perspective, the term **demographic filtering** means grouping or segregating	**ESR:** Varied.

population based on **demographic variables** like **age, gender, education, income**, etc.

A very prominent example of segregating population based on the demographic variable **'age'** is, when the vaccination campaign against the pandemic caused by the coronavirus, was started, the whole country was segregated into 3 categories :

a) Under 18 years old.
b) 18-45 years old.
c) Above 45.



Can you tell an example for the same?

Note: Appreciate the student if he tries to come up with an example.

If we talk in the context of movies, a demographic filtering system recommends similar kinds of movies to people with similar demographic backgrounds.
This system recommends movies that are popular and

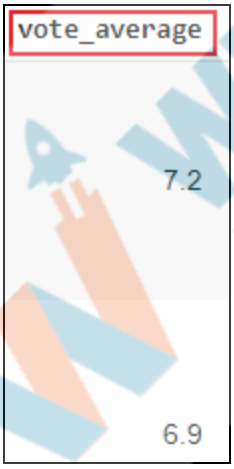
ESR: Varied.

well-rated, regardless of the genre, cast, or any other factor. **This system does not consider the individual taste of each person**, hence, it is very simple and easy to implement.
Now, if we want to create a demographic recommendation system, we need:

a) A common **scoring or metric system** to rate the movie.
b) We need to calculate that **score** for each movie.
c) We need to sort or evaluate the movie based on that **score** and recommend the best movie according to it.

Let's consider an example to understand it. In the given image, we have rated 4 cricketers using the metric as **'total number of runs which they have made in one day international cricket matches, throughout their career'.** Based on this metric can you recommend or tell, who is the top rated cricketer or the most popular cricketer?

ESR: SR Tendulkar.

Great, you are right! Just like we need a metric or scoring system to find out the top rated or most popular cricketers, similarly we need a scoring system to rate movies. Okay, so let's start by thinking of a **common scoring** or **metric** system to rate the movies. Can you think of one? *Note: Let the student think and if he doesn't come up with anything, give him a hint.* *HINT: THE ANSWER IS CLOSE TO ONE OF THE COLUMNS IN THE DATASET.* If you look at the dataset carefully, we have a column named **"vote_average",** which scores each movie between **1 and 10.**  First of all, tell me what you understand by the term voting average? And second, is it a good enough score to rate movies? *Note: Let the student think. If he says yes, then why yes? And if no, why no?*	**ESR:** Varied. **ESR**: Varied.

So to answer the first question, the **voting average** for a movie can be defined as the **ratio of sum of all the ratings given to a movie to total number of viewers**.

Mathematically, it can be expressed as,

$$\text{VOTING AVERAGE} = \frac{\text{Sum of scores given by the users}}{\text{Total number of users}}$$

Now, moving on to the second question, from my perspective I don't think it is a good enough score to rate movies. Let me explain it to you with an example.

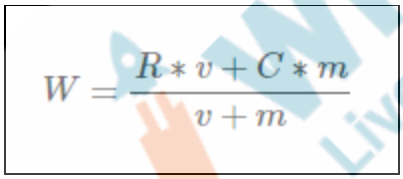
Movie	Rating	Viewers
Avatar	Each of them gave a rating of 8	10000
Inception	Each of them gave a rating of 10	2

Can you try and calculate the average rating for both of the movies?

Note: Let the student do some calculations and come up with an answer. If they need help, guide them, don't solve for them.

For **Avatar,** average rating = (8*10000) / (10000) = **8**
For **Inception,** average rating = (10*2) / (2) = **10**

ESR: Sure

So what do you think, is **Avatar**, a movie with **10000 viewers** rating it as 8, is not as good as **Inception,** a movie with only 2 viewers rating it as 10? True, because only 2 people have rated Inception as 10, while 10000 people have rated Avatar as 8. So we can conclude that **average rating** is not a fair method to score movies. Now it takes us back to our problem, to find an appropriate scoring system, which not only takes **average rating** into count, but also takes **number of viewers** or **vote_count** into consideration as well. 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:  $$W = \frac{R * v + C * m}{v + m}$$ Where, **W**: Weighted rating. **R**: Average number of the movie's ratings **[0 - 10].** **v**: Number of votes for the movie. **C**: The mean vote from overall data. **m**: Minimum number of necessary votes required for a movie to be considered as a recommendation. Great! now we are ready to start coding!	**ESR:** It is not a fair rating!

Teacher Ends Slideshow 	
TEACHER-LED ACTIVITY - 15 mins	
Teacher Initiates Screen Share	
ACTIVITY • **Make students understand what weighted rating is.** • **Make students understand how demographic filtering is done.**	
Teacher Action	**Student Action**
Before calculating a weighted average for each movie, let's clean our dataset. There is a lot of data in our dataset which is not necessary. Let's print all the column headers in our dataset. For that, open this Teacher Activity 1 : Boilerplate, and in a new cell, write the command, **result_df.info(),** which will print all the column headers in our dataset.	

```
result_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4803 entries, 0 to 4802
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4803 non-null   int64  
 1   genres                4803 non-null   object 
 2   homepage              1712 non-null   object 
 3   id                    4803 non-null   int64  
 4   keywords              4803 non-null   object 
 5   original_language     4803 non-null   object 
 6   original_title        4803 non-null   object 
 7   overview              4800 non-null   object 
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object 
 10  production_countries  4803 non-null   object 
 11  release_date          4802 non-null   object 
 12  revenue               4803 non-null   int64  
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object 
 15  status                4803 non-null   object 
 16  tagline               3959 non-null   object 
 17  title_x               4803 non-null   object 
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64  
 20  title_y               4803 non-null   object 
 21  cast                  4803 non-null   object 
 22  crew                  4803 non-null   object 
dtypes: float64(3), int64(4), object(16)
memory usage: 900.6+ KB
```

Can you spot some of the columns which might not be useful to us for getting insights from our data?

Note: Let the student try!

Some of the columns like **budget**, **homepage, title_x, title_y , production_companies** etc, might not be that useful to us.

So we can clean our dataset or remove some of the

ESR: Varied.

unwanted columns by using the **.drop()** method as, **dataframe.drop([column names] , axis = 1 , inplace=True)** *Note: Here axis = 1 means we are trying to remove columns, not rows.* Can you try and frame this command for our DataFrame? So let's clean our dataset as, **result_df.drop(['homepage' , 'title_x' , 'title_y' , 'production_companies'] , axis = 1 , inplace = True)**	**ESR:** Sure.
``# dropping columns : cleaning dataset result_df.drop(['homepage', 'title_x', 'title_y', 'production_companies'] , axis = 1 , inplace = True)```	
To verify, let's print the column headers, using the **.info()** method, as **result_df.info()**	

```
result_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 4803 entries, 0 to 4802
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   id                    4803 non-null   int64
 3   keywords              4803 non-null   object
 4   original_language     4803 non-null   object
 5   original_title        4803 non-null   object
 6   overview              4799 non-null   object
 7   popularity            4803 non-null   float64
 8   production_countries  4803 non-null   object
 9   release_date          4802 non-null   object
 10  revenue               4803 non-null   int64
 11  runtime               4801 non-null   float64
 12  spoken_languages      4803 non-null   object
 13  status                4803 non-null   object
 14  tagline               3959 non-null   object
 15  vote_average          4803 non-null   float64
 16  vote_count            4803 non-null   int64
 17  cast                  4803 non-null   object
 18  crew                  4802 non-null   object
```

If we observe, we will see that some of the columns have null values stored in them. For instance, the **overview** column has **4 null values** (4803 - 4799 = 4), the **tagline** column has **845 null values** (4803 - 3959 = 845).

Certain operations are not applicable to **null** values which might cause an error later, so it would be a wise decision to drop the rows with the null values.

For that, let's use the **dropna()** method as,
result_df.dropna(inplace = True)

```
result_df.dropna(inplace = True)
```

To verify, let's print the dataframe information, using the **info()** method.

We will see that rows with any null values are dropped.

```
result_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 3958 entries, 0 to 4801
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                3958 non-null   int64  
 1   genres                3958 non-null   object 
 2   id                    3958 non-null   int64  
 3   keywords              3958 non-null   object 
 4   original_language     3958 non-null   object 
 5   original_title        3958 non-null   object 
 6   overview              3958 non-null   object 
 7   popularity            3958 non-null   float64
 8   production_countries  3958 non-null   object 
 9   release_date          3958 non-null   object 
 10  revenue               3958 non-null   int64  
 11  runtime               3958 non-null   float64
 12  spoken_languages      3958 non-null   object 
 13  status                3958 non-null   object 
 14  tagline               3958 non-null   object 
 15  vote_average          3958 non-null   float64
 16  vote_count            3958 non-null   int64  
 17  cast                  3958 non-null   object 
 18  crew                  3958 non-null   object 
```

Teacher Stops Screen Share

So now it's your turn.
Please share your screen with me.

Teacher Starts Slideshow  **Slide # to #** <**Note**: Only Applicable for Classes with VA> Refer to speaker notes and follow the instructions on each slide.	
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.	
Teacher Ends Slideshow 	
STUDENT-LED ACTIVITY - 20 mins	
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start Screen Share.** • **The teacher gets into Full Screen.**	
Student Initiates Screen Share	
ACTIVITY • **Students write code to apply demographic filtering on the data!**	
Teacher Action	**Student Action**
Now that our dataset is clean, let's find the **weighted rating** for each movie. To do that, we need to calculate four variables, **R, v, C, and m.** Let's start by finding the value of **R,** which is the **average rating** for a movie **[0 to 10]**. If we closely look at our dataset, we already have a column named **vote_average.** To extract the data from **vote_average** column, open the Student boilerplate link : Boilerplated, and in a new cell, run this line of code:	

R = result_df[‘ vote_average ‘]	
`R = result_df['vote_average']`	
Great. Now we need **v,** which is the number of votes for a movie. Can you look at the dataset and find something, which gives you the number of votes for a movie? *Note: Let the student try first.* It’s good that you tried. If we look at the dataset carefully, we have a column named **vote_count** which will give us the value of **v.** To extract **vote_count**, run the following line of code: **v = result_df[‘ vote_count ‘]**	**ESR:** Varied.
`v = result_df['vote_count']`	
Now comes the part where we need to find the value of **C.** So, if we look at the definition of **C,** which is **“The mean vote value from overall data”,** which means we have to find the **mean** of all the values listed in the **vote_average** column. To find that value, use the **.mean()** method as, **C = result_df[‘vote_average’].mean()**. We can clearly see that the value of C or the mean value comes out to be **6.0921**.	
`C = result_df['vote_average'].mean()` `print(C)` `6.092171559442011`	
Now finally, to find the value of **‘m’,** we need to look at its definition, which says, **“Minimum number of necessary votes required for a movie to be considered as a**	

recommendation."

To understand this statement, we need to learn about a mathematical term known as **'quantile'**.

Have you ever heard about this term? Can you explain what it means?

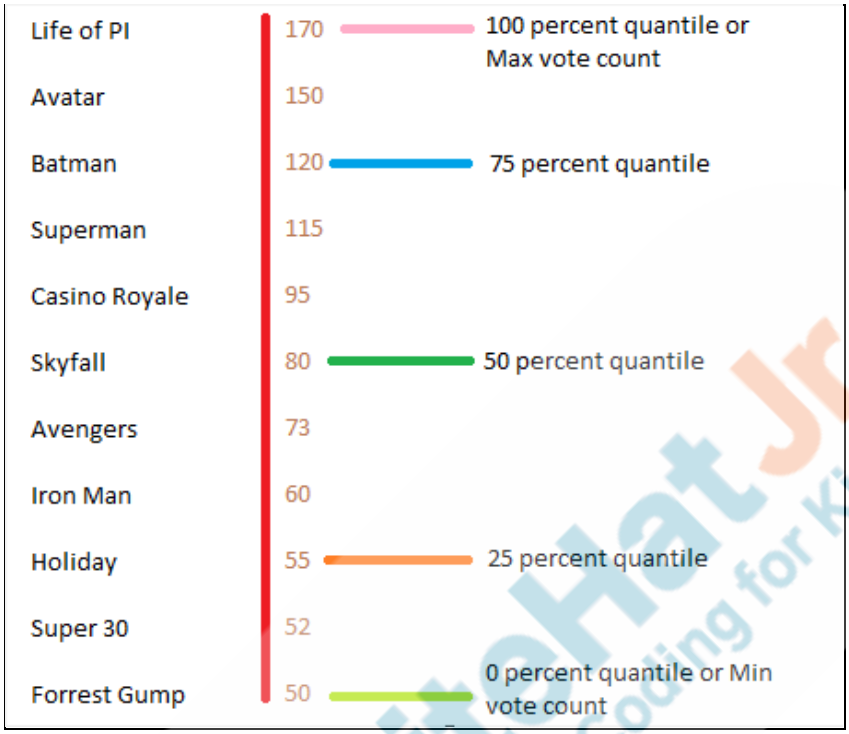
Quantile is a point, where a sample data is divided into equal sized subgroups.

Let's understand it with the help of an example. Consider the table below, where we have **10 movies, sorted** in **descending order** based on their **vote count**.

Movie	Vote count
Life of PI	170
Avatar	150
Batman	120
Superman	115
Casino Royale	95
Skyfall	80
Avengers	73
Iron Man	60
Holiday	55
Super 30	52
Forrest Gump	50

ESR: Varied.

Now, according to the definition, **quantile is defined as a point, where a sample data is divided into equal sized subgroups.** For example, if we talk about **50 percent quantile**, it will be a point in our table, where the data will be divided into 2 equal categories. If we closely look at the table, we can say that the 6th movie, **Skyfall** with a **vote count** of **80** is at **50 percent quantile**. Let's try to find a 25 percent quantile mark in our data. Can you spot the vote count at the 25 percent quantile mark? It will be that point in our vote count where 25 percent of the movie lies below that point and 75 percent of the movie lies above it. If you have spotted the vote count value of **55 (Holiday movie)**, you are right. If we present the above data pictorially, it will look somewhat like this,	**ESR**: Varied.



Now in our original dataframe, let's consider only the top 10 percent of the movies for recommendation, which means we need to find the **90 percent quantile** mark in the **vote_count** column.

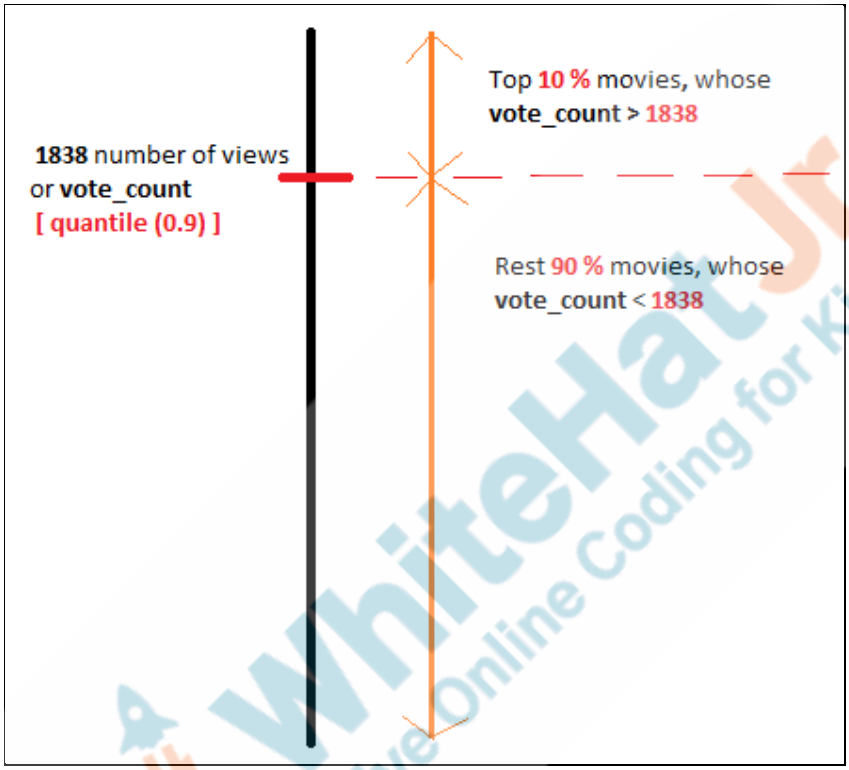
To do so, use the command,
m = result_df['vote_count'].quantile(0.9)

Also, let's print the value of **m** as well, using the command, **print(m)**.

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

```
1838.4000000000015
```

If we try to present the above result graphically, we can say that 10 percent of the movies lie above the **vote_count** value of **1838 (90 percent quantile)** and 90 percent of the movies lie below it.



Now, we have all the variables, it's time to calculate the **weighted rating** for each movie.
Let's add a column in our dataset named as **weighted_rating**,
result_df['weighted_rating'] = (R*v + C*m) / (v + m)

```
result_df['weighted_rating'] = (R*v + C*m) / (v + m)
```

To see the output properly, let's print the first 5 rows for 2 columns only using the command,
result_df[['original_title' , 'weighted_rating']].head(),

```
result_df[['original_title' , 'weighted_rating']].head()
```

	original_title	weighted_rating
0	Avatar	7.050669
1	Pirates of the Caribbean: At World's End	6.665696
2	Spectre	6.239396
3	The Dark Knight Rises	7.346721
4	John Carter	6.096368

Perfect, now we have a scoring system with us and we have rated each movie according to that score. Let's move on to our last step, which is to **sort** the movies according to their **weighted_rating** in **descending order,** so that we can get the most popular movies listed at the top of our dataframe.

To do that, we can use the **.sort_values()** method as,

result_df.sort_values('weighted_rating' , ascending = False , inplace = True)

Which will sort our dataset in **descending** order using the **'weighted_rating'** column as reference.

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

To analyze the data properly, let's print top 10 rows of some selected columns from our dataframe using,

result_df['original_title' , 'vote_average' , 'vote_count' , 'weighted_rating' , 'popularity'].head(10).

```
result_df[['original_title' , 'vote_average' , 'vote_count' , 'weighted_rating' , 'popularity']].head(10)
```

	original_title	vote_average	vote_count	weighted_rating	popularity
1881	The Shawshank Redemption	8.5	8205	8.059258	136.747729
662	Fight Club	8.3	9413	7.939256	146.757391
65	The Dark Knight	8.2	12002	7.920020	187.322927
3232	Pulp Fiction	8.3	8428	7.904645	121.463076
96	Inception	8.1	13752	7.863239	167.583710
3337	The Godfather	8.4	5893	7.851236	143.659698
95	Interstellar	8.1	10867	7.809479	724.247784
809	Forrest Gump	8.2	7927	7.803188	138.133331
329	The Lord of the Rings: The Return of the King	8.1	8064	7.727243	123.630332
1990	The Empire Strikes Back	8.2	5879	7.697884	78.517830

Hurray! We made our first basic recommendation system! 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.

Here, we are using the **express** module in 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. Let's write the command as,

import plotly.express as px

bar_plot = px.bar(result_df.head(10).sort_values('weighted_rating' , x = 'weighted_rating' , y = 'original_title' , orientation = 'h'))

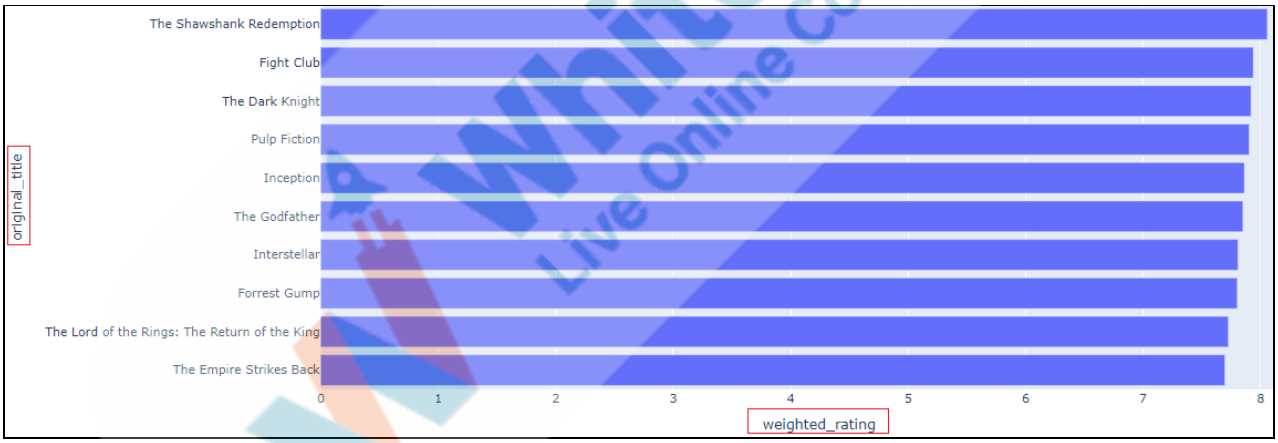
bar_plot.show()

```
import plotly.express as px
bar_plot = px.bar(result_df.head(10).sort_values('weighted_rating') , x = 'weighted_rating' , y = 'original_title' , orientation = 'h')
bar_plot.show()
```

The output will look like the below screenshot. Can you recommend or suggest the top rated movie by looking at this bar graph?

Great!

ESR: The Shawshank Redemption



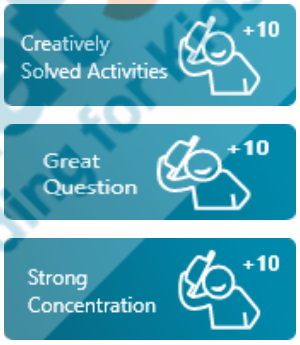
That's how we make the demographic **filtering recommendation system.**

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.	
Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 05 mins	
Teacher Starts Slideshow  **Slide # to #** <**Note**: Only Applicable for Classes with VA>	
Activity details **Following are the WRAP-UP session deliverables:** • Appreciate the student. • Revise the current class activities. • Discuss the quizzes.	
WRAP-UP QUIZ Click on In-Class Quiz	
Continue WRAP-UP Session  **Slide # to #** <**Note**: Only Applicable for Classes with VA>	
Activity Details **Following are the session deliverables:** • Explain the facts and trivia. • Next class challenge. • Project for the day. • Additional Activity (Optional).	

FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	Student Action
You get “hats-off” for your excellent work! In the next class, we’ll be making a content-based recommender system.	*Make sure you have given at least 2 hats-off during the class for:* 
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections	
Teacher Clicks ✖ End Class	

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Activity 1	Boilerplate Code	https://colab.research.google.com/drive/1dtfRmroXYK2RIT5zB7UUFcR994HDEZfp?usp=sharing
Teacher Activity 2	Reference Code	https://colab.research.google.com/drive/1kNL4wsEhVC0sJ-ClQ4lDeVpQGFJXqHT7?usp=sharing
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/8a0b55b1-709c-4b26-86d8-1c3196aab202.pdf
Teacher Reference 2	Project Solution	https://colab.research.google.com/drive/1viaP93hbMsuE9cLhq6sObbA6s0D8rLCH?usp=sharing
Teacher Reference 3	Visual-Aid	Will be added after VA creation
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/535ab8b6-d2b5-45bd-b6e5-9c76618035ad.pdf
Student Activity 1	Boilerplate Code	https://colab.research.google.com/drive/1kxjwW7_FMgU7tIX4JJ2fuF09q1CD2xj4?usp=sharing