Group by (In Detail)

https://colab.research.google.com/drive/1JZwTCZp2kbiTACzcuXmWq_FL-GRNo9ym?usp=sharing#scrollTo=_NNmeC7kALcE

https://www.youtube.com/watch?v=LPBjF4_gZnI

- In `groupby`, we form groups on basis of a column
- There are 2 types of columns
  - Numerical
  - Categorical
- `groupby` is applied on **Categorical Column.**
- We have 2 dataset - Movies & IPL

```
import pandas as pd
import numpy as np
movies = pd.read_csv(r"C:\Users\Jeevan\Downloads\imdb-top-1000 - imdb-top-
movies.head()
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
0	The Shawshank Redemption	1994	142	Drama	9.3	Frank Darabont	Tim Robbins	2343110	28341469	80.0
1	The Godfather	1972	175	Crime	9.2	Francis Ford Coppola	Marlon Brando	1620367	134966411	100.0
2	The Dark Knight	2008	152	Action	9.0	Christopher Nolan	Christian Bale	2303232	534858444	84.0
3	The Godfather: Part II	1974	202	Crime	9.0	Francis Ford Coppola	Al Pacino	1129952	57300000	90.0
4	12 Angry Men	1957	96	Crime	9.0	Sidney Lumet	Henry Fonda	689845	4360000	96.0

- We applied `sum()` function on the df.

```
genres= movies.groupby('Genre')
genres.sum()
```

Genre	Series_Title	Released_Year	Runtime	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
Action	The Dark KnightThe Lord of the Rings: The Retu...	200820032010200120021999198019771962195420001 9...	22196	1367.3	Christopher NolanPeter JacksonChristopher Nola...	Christian BaleElijah WoodLeonardo DiCaprioElij...	72282412	32632261314	10499.0
Adventure	InterstellarBack to the FutureInglourious Bast...	201419852009198119681962195920131975196319481 9...	9656	571.5	Christopher NolanRobert ZemeckisQuentin Tarant...	Matthew McConaugheyMichael J. FoxBrad PittJürg...	22576163	9496922464	5020.0
Animation	Sen to Chihiro no kamikakushiThe Lion KingHota...	200119941988201620182017200819971995201920092 0...	8166	650.3	Hayao MiyazakiRoger AllersIsao TakahataMakoto ...	Daveigh ChaseRob MinkoffTsutomu TatsumiRyûnosu...	21978630	14631473048	6082.0
Biography	Schindler's ListGoodfellasHamiltonThe	199319902020201120022017199519842018201320132 0...	11970	698.6	Steven SpielbergMartin ScorseseThomas	Liam NeesonRobert De NiroLin-Manuel	24006844	8276357606	6023.0

- It summed up all the numerical columns.
- You can apply `min(), max()` as they work on strings
- But `mean()` won't work

Solution:

```
genres= movies.groupby('Genre')
numeric_columns = ['Runtime', 'IMDB_Rating', 'No_of_Votes', 'Gross', 'Metascore
movies[numeric_columns].dtypes
```

```
Runtime          int64
IMDB_Rating    float64
No_of_Votes      int64
Gross            int64
Metascore      float64
dtype: object
```

- We select the numeric columns & find out their type

```
genre_means = genres[numeric_columns].mean()

genre_means
```

Genre	Runtime	IMDB_Rating	No_of_Votes	Gross	Metascore
Action	129.046512	7.949419	420246.581395	1.897224e+08	73.419580
Adventure	134.111111	7.937500	313557.819444	1.319017e+08	78.437500
Animation	99.585366	7.930488	268032.073171	1.784326e+08	81.093333
Biography	136.022727	7.938636	272805.045455	9.404952e+07	76.240506
Comedy	112.129032	7.901290	178195.658065	1.010572e+08	78.720000
Crime	126.392523	8.016822	313398.271028	7.899656e+07	77.080460
Drama	124.737024	7.957439	212343.612457	1.225259e+08	79.701245
Family	107.500000	7.800000	275610.500000	2.195553e+08	79.000000
Fantasy	85.000000	8.000000	73111.000000	3.913633e+08	NaN
Film-Noir	104.000000	7.966667	122405.000000	4.197018e+07	95.666667
Horror	102.090909	7.909091	340232.363636	9.405902e+07	80.000000
Mystery	119.083333	7.975000	350250.333333	1.047014e+08	79.125000
Thriller	108.000000	7.800000	27733.000000	1.755074e+07	81.000000
Western	148.250000	8.350000	322416.250000	1.455538e+07	78.250000

- We find out their mean.

QUESTION

Q. find the top 3 genres by total earning

- We will apply `sum()`
  - We had already run `genres= movies.groupby('Genre')`

`genres.sum()`

- And single out the “Gross” column

```
genres.sum()['Gross']
```

```
Genre
Action        32632261314
Adventure      9496922464
Animation     14631473048
Biography      8276357606
Comedy        15663868165
Crime          8452631908
Drama         35409974041
Family          439110554
Fantasy         782726696
Film-Noir       125910543
Horror         1034649238
Mystery        1256417015
Thriller         17550741
Western          58221508
Name: Gross, dtype: int64
```

- now, we'll sort this→ `sort_values(ascending=False)`

```
genres.sum()['Gross'].sort_values(ascending=False)
```

```
Genre
Drama        35409974041
Action       32632261314
Comedy       15663868165
Animation    14631473048
Adventure     9496922464
Crime         8452631908
Biography     8276357606
Mystery       1256417015
Horror        1034649238
Fantasy        782726696
Family         439110554
Film-Noir      125910543
Western         58221508
Thriller        17550741
Name: Gross, dtype: int64
```

Alternative way:

```
movies.groupby('Genre')['Gross'].sum().sort_values(ascending=False).head(3)
```

```
Genre
Drama      35409974041
Action     32632261314
Comedy     15663868165
Name: Gross, dtype: int64
```

- Here, we first we separate the "Gross" column & apply `sum()` function to a single column.

- The 2nd (Alternative way) is faster cuz in this, you're applying sum to only 1 column.

Sort by **Mean**

```
genres= movies.groupby('Genre')
numeric_columns = ['Runtime', 'IMDB_Rating', 'No_of_Votes', 'Gross', 'Metascore
mean_values= genres[numeric_columns].mean()
mean_values['Gross']
```

```
Genre
Action        1.897224e+08
Adventure     1.319017e+08
Animation     1.784326e+08
Biography     9.404952e+07
Comedy        1.010572e+08
Crime         7.899656e+07
Drama         1.225259e+08
Family        2.195553e+08
Fantasy       3.913633e+08
Film-Noir     4.197018e+07
Horror        9.405902e+07
Mystery       1.047014e+08
Thriller      1.755074e+07
Western       1.455538e+07
Name: Gross, dtype: float64
```

Q. find the genre with highest avg IMDB rating

```
genres = movies.groupby('Genre')
genres['IMDB_Rating'].mean().sort_values(ascending=False)
```

```
Genre
Western      8.350000
Crime        8.016822
Fantasy      8.000000
Mystery      7.975000
Film-Noir    7.966667
Drama        7.957439
Action       7.949419
Biography    7.938636
Adventure    7.937500
Animation    7.930488
```

Q. find director with most popularity

```
movies.groupby('Director')['No_of_Votes'].sum().sort_values(ascending=False).h

Output:
Director
Christopher Nolan    11578345
Name: No_of_Votes, dtype: int64
```

- Here, we sort with “Director”
- Took No of votes column to determine popularity
- Applied sum fn as we have to calculate total number of votes

Q. find number of movies done by each actor

- Apply value count on column “Star1"

```
movies['Star1'].value_counts()
```

```
Star1
Tom Hanks          12
Robert De Niro     11
Al Pacino          10
Clint Eastwood     10
Humphrey Bogart     9
```

- This will tell you which actor did how many movies

Alternative Method:

- We will use `groupby`

```
movies.groupby('Star1')['Series_Title'].count().sort_values(ascending=False)
```

- `count()` will enter into each group and count the number of rows.
- We can take any column instead of *Series_Title*

Alt Method:

```
star_movie_count = movies.groupby('Star1').size().sort_values(ascending=False)
star_movie_count
```

- `.size()`: Counts the total number of rows (movies) for each group (actor), which gives the number of movies each actor (`Star1`) has been in.
  - The `size()` function in **Pandas** is used to count the **number of rows** in each group when you apply `groupby()` to a DataFrame
  - **Counts all rows** in each group, regardless of whether any of the column values are missing (null).

Q. find total number of groups formed

`len` function

```
len (movies.groupby('Genre'))
Output: 14
```

There are 14 groups on basis of genre.

Alt way: `nunique()`

```
unique= movies['Genre'].nunique()
unique
Output: 14
```

find the highest rated movie of each genre

```
movies.groupby('Genre')[["Series_Title", 'IMDB_Rating']].max().sort_values(by
='IMDB_Rating', ascending=False).reset_index()
```

	Genre	Series_Title	IMDB_Rating
0	Drama	Zwartboek	9.3
1	Crime	À bout de souffle	9.2
2	Action	Yôjinbô	9.0
3	Biography	Zerkalo	8.9
4	Western	The Outlaw Josey Wales	8.8
5	Adventure	Zombieland	8.6
6	Animation	Ôkami kodomo no Ame to Yuki	8.6
7	Comedy	Zindagi Na Milegi Dobara	8.6
8	Horror	The Thing	8.5
9	Mystery	Vertigo	8.4
10	Fantasy	Nosferatu	8.1
11	Film-Noir	The Third Man	8.1
12	Family	Willy Wonka & the Chocolate Factory	7.8
13	Thriller	Wait Until Dark	7.8

Find out Rows in every group

```
movies.groupby('Genre').size()
```

```
Genre
Action        172
Adventure      72
Animation      82
Biography      88
Comedy        155
Crime         107
Drama         289
Family          2
Fantasy         2
Film-Noir       3
Horror         11
Mystery        12
Thriller        1
Western         4
dtype: int64
```

- Sorted on basis of index

Alt method:

```
movies['Genre'].value_counts()
```

- Sorted by values

First movie of a group

```
genres= movies.groupby('Genre')
genres.first()
```

Genre	Series_Title	Released_Year	Runtime	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
Action	The Dark Knight	2008	152	9.0	Christopher Nolan	Christian Bale	2303232	534858444	84.0
Adventure	Interstellar	2014	169	8.6	Christopher Nolan	Matthew McConaughey	1512360	188020017	74.0
Animation	Sen to Chihiro no kamikakushi	2001	125	8.6	Hayao Miyazaki	Daveigh Chase	651376	10055859	96.0
Biography	Schindler's List	1993	195	8.9	Steven Spielberg	Liam Neeson	1213505	96898818	94.0
Comedy	Gisaengchung	2019	132	8.6	Bong Joon Ho	Kang-ho Song	552778	53367844	96.0
Crime	The Godfather	1972	175	9.2	Francis Ford Coppola	Marlon Brando	1620367	134966411	100.0
Drama	The Shawshank Redemption	1994	142	9.3	Frank Darabont	Tim Robbins	2343110	28341469	80.0
Family	E.T. the Extra-Terrestrial	1982	115	7.8	Steven Spielberg	Henry Thomas	372490	435110554	91.0

7th movie in a group

```
genres.nth(6)
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
28	The Silence of the Lambs	1991	118	Crime	8.6	Jonathan Demme	Jodie Foster	1270197	130742922	85.0
29	Star Wars	1977	121	Action	8.6	George Lucas	Mark Hamill	1231473	322740140	90.0
34	Whiplash	2014	106	Drama	8.5	Damien Chazelle	Miles Teller	717585	13092000	88.0
70	Mononoke-hime	1997	134	Animation	8.4	Hayao Miyazaki	Yôji Matsuda	343171	2375308	76.0
95	Amélie	2001	122	Comedy	8.3	Jean-Pierre Jeunet	Audrey Tautou	703810	33225499	69.0
107	Amadeus	1984	160	Biography	8.3	Milos Forman	F. Murray Abraham	369007	51973029	88.0
137	Queen	2013	146	Adventure	8.2	Vikas Bahl	Kangana Ranaut	60701	1429534	NaN
714	The Lady Vanishes	1938	96	Mystery	7.8	Alfred Hitchcock	Margaret Lockwood	47400	474203697	98.0

💡 **We gave 6 because index starts from 0.**

get_group

- Shows all the movies in a particular group.

```
genres.get_group('Horror')
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
49	Psycho	1960	109	Horror	8.5	Alfred Hitchcock	Anthony Perkins	604211	32000000	97.0
75	Alien	1979	117	Horror	8.4	Ridley Scott	Sigourney Weaver	787806	78900000	89.0
271	The Thing	1982	109	Horror	8.1	John Carpenter	Kurt Russell	371271	13782838	57.0
419	The Exorcist	1973	122	Horror	8.0	William Friedkin	Ellen Burstyn	362393	232906145	81.0
544	Night of the Living Dead	1968	96	Horror	7.9	George A. Romero	Duane Jones	116557	89029	89.0
707	The Innocents	1961	100	Horror	7.8	Jack Clayton	Deborah Kerr	27007	2616000	88.0
724	Get Out	2017	104	Horror	7.7	Jordan Peele	Daniel Kaluuya	492851	176040665	85.0
844	Halloween	1978	91	Horror	7.7	John Carpenter	Donald Pleasence	233106	47000000	87.0

All details

```
genres.describe()
```

	Runtime								IMDB_Rating		...	Gross		Metascore						
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%	75%	max
Genre																				
Action	172.0	129.046512	28.500706	45.0	110.75	127.5	143.25	321.0	172.0	7.949419	...	2.674437e+08	936662225.0	143.0	73.419580	12.421252	33.0	65.00	74.0	82.00	98.0
Adventure	72.0	134.111111	33.317320	88.0	109.00	127.0	149.00	228.0	72.0	7.937500	...	1.998070e+08	874211619.0	64.0	78.437500	12.345393	41.0	69.75	80.5	87.25	100.0
Animation	82.0	99.585366	14.530471	71.0	90.00	99.5	106.75	137.0	82.0	7.930488	...	2.520612e+08	873839108.0	75.0	81.093333	8.813646	61.0	75.00	82.0	87.50	96.0
Biography	88.0	136.022727	25.514466	93.0	120.00	129.0	146.25	209.0	88.0	7.938636	...	9.829924e+07	753585104.0	79.0	76.240506	11.028187	48.0	70.50	76.0	84.50	97.0
Comedy	155.0	112.129032	22.946213	68.0	96.00	106.0	124.50	188.0	155.0	7.901290	...	8.107809e+07	886752933.0	125.0	78.720000	11.829160	45.0	72.00	79.0	88.00	99.0
Crime	107.0	126.392523	27.689231	80.0	106.50	122.0	141.50	229.0	107.0	8.016822	...	7.102163e+07	790482117.0	87.0	77.080460	13.099102	47.0	69.50	77.0	87.00	100.0
Drama	289.0	124.737024	27.740490	64.0	105.00	121.0	137.00	242.0	289.0	7.957439	...	1.164461e+08	924558264.0	241.0	79.701245	12.744687	28.0	72.00	82.0	89.00	100.0

1 Random Movie from group

```
genres.sample()
```

```
genres.sample(2)
```

- Will give 2 movies

```
genres.sample(2, replace=True)
```

- If a genre has only 1 movie, `replace=True` will duplicate that movie.

Unique Movies in a group

```
genres.nunique()
```

	Series_Title	Released_Year	Runtime	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
Genre									
Action	172	61	78	15	123	121	172	172	50
Adventure	72	49	58	10	59	59	72	72	33
Animation	82	35	41	11	51	77	82	82	29
Biography	88	44	56	13	76	72	88	88	40
Comedy	155	72	70	11	113	133	155	155	44
Crime	106	56	65	14	86	85	107	107	39
Drama	289	83	95	14	211	250	288	287	52

- There are 72 unique movies in action
- In action, there are 61 unique year values, so on.

AGGREGATE FUNCTIONS

- sum, min, ,max, etc.
- You can apply multiple fns at one time.
- You write `agg` and pass a **dictionary.**

```
genres.agg(
    {
        'Runtime':'mean',
        'IMDB_Rating':'mean',
        'No_of_Votes':'sum',
        'Gross':'sum',
        'Metascore':'min'
    }
)
```

Genre	Runtime	IMDB_Rating	No_of_Votes	Gross	Metascore
Action	129.046512	7.949419	72282412	32632261314	33.0
Adventure	134.111111	7.937500	22576163	9496922464	41.0
Animation	99.585366	7.930488	21978630	14631473048	61.0
Biography	136.022727	7.938636	24006844	8276357606	48.0
Comedy	112.129032	7.901290	27620327	15663868165	45.0
Crime	126.392523	8.016822	33533615	8452631908	47.0
Drama	124.737024	7.957439	61367304	35409974041	28.0
Family	107.500000	7.800000	551221	439110554	67.0
Fantasy	85.000000	8.000000	146222	782726696	NaN

Agg + Rename

```
result = genres.agg(Total_Runtime=('Runtime', 'sum'),
                    Max_Votes=('No_of_Votes', 'max'))
result
```

Genre	Total_Runtime	Max_Votes
Action	22196	2303232
Adventure	9656	1512360
Animation	8166	999790
Biography	11970	1213505
Comedy	17380	939631
Crime	13524	1826188
Drama	36049	2343110
Family	215	372490

Multiple function on a same column

- You will pass a list

```
genres.agg(['min','max','sum'])
```

	Series_Title			Released_Year			Runtime			IMDB_Rating	...	Star1	No_of_Votes	
	min	max	sum	min	max	sum	min	max	sum	min	...	sum	min	max
Genre														
Action	300	Yôjinbô	The Dark KnightThe Lord of the Rings: The Retu...	1924	2019	200820032010200120021999198019771962195420001 9...	45	321	22196	7.6	...	Christian BaleElijah WoodLeonardo DiCaprioElij...	25312	230323
Adventure	2001: A Space Odyssey	Zombieland	InterstellarBack to the FutureInglourious Bast...	1925	PG	201419852009198119681962195920131975196319481 9...	88	228	9656	7.6	...	Matthew McConaugheyMichael J. FoxBrad PittJürg...	29999	151236
Animation	Akira	Ôkami kodomo no Ame to Yuki	Sen to Chihiro no kamikakushiThe Lion KingHota...	1940	2020	200119941988201620182017200819971995201920092 0...	71	137	8166	7.6	...	Daveigh ChaseRob MinkoffTsutomu TatsumiRyûnosu...	25229	99979
Biography	12 Years a Slave	Zerkalo	Schindler's ListGoodfellasHamiltonThe Intoucha...	1928	2020	199319902020201120022017199519842018201320132 0...	93	209	11970	7.6	...	Liam NeesonRobert De NiroLin-Manuel MirandaÉri...	27254	121350
Comedy	(500) Days of Summer	Zindagi Na Milegi Dobara	GisaengchungLa vita è bellaModern TimesCity Li...	1921	2020	201919971936193120091964194020012000197319601 9...	68	188	17380	7.6	...	Kang-ho SongRoberto BenigniCharles	26337	93963

- It will calculate these 3 for all columns.

Merge the above 2 syntaxes

```
# Adding both the syntax
genres.agg(
    {
        'Runtime':['min','mean'],
        'IMDB_Rating':'mean',
        'No_of_Votes':['sum','max'],
        'Gross':'sum',
        'Metascore':'min'
    }
)
```

	Runtime		IMDB_Rating	No_of_Votes		Gross	Metascore
	min	mean	mean	sum	max	sum	min
Genre							
Action	45	129.046512	7.949419	72282412	2303232	32632261314	33.0
Adventure	88	134.111111	7.937500	22576163	1512360	9496922464	41.0
Animation	71	99.585366	7.930488	21978630	999790	14631473048	61.0
Biography	93	136.022727	7.938636	24006844	1213505	8276357606	48.0
Comedy	68	112.129032	7.901290	27620327	939631	15663868165	45.0
Crime	80	126.392523	8.016822	33533615	1826188	8452631908	47.0
Drama	64	124.737024	7.957439	61367304	2343110	35409974041	28.0
Family	100	107.500000	7.800000	551221	372490	439110554	67.0
Fantasy	76	85.000000	8.000000	146222	88794	782726696	NaN
Film-Noir	100	104.000000	7.966667	367215	158731	125910543	94.0
Horror	71	102.090909	7.909091	3742556	787806	1034649238	46.0
Mystery	96	119.083333	7.975000	4203004	1129894	1256417015	52.0

Loop

- We get 2 things:
  - Group : ‘str’
  - Data: ‘Dataframe’

```
for group,data in genres:
    print (type(group), type(data))
```

```
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
```

```
for group,data in genres:
    print(group)
```

```
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
Film-Noir
Horror
Mystery
Thriller
Western
```

```
for group,data in genres:
    print (data)
```

```
                                       Series_Title Released_Year  Runtime  \
2                                   The Dark Knight          2008      152
5        The Lord of the Rings: The Return of the King          2003      201
8                                         Inception          2010      148
10   The Lord of the Rings: The Fellowship of the Ring          2001      178
13              The Lord of the Rings: The Two Towers          2002      179
..                                                ...           ...      ...
968                                     Falling Down          1993      113
979                                    Lethal Weapon          1987      109
982                                        Mad Max 2          1981       96
983                                     The Warriors          1979       92
985                              Escape from Alcatraz          1979      112

      Genre  IMDB_Rating           Director              Star1  No_of_Votes  \
2    Action          9.0  Christopher Nolan     Christian Bale      2303232
5    Action          8.9      Peter Jackson        Elijah Wood      1642758
8    Action          8.8  Christopher Nolan  Leonardo DiCaprio      2067042
10   Action          8.8      Peter Jackson        Elijah Wood      1661481
13   Action          8.7      Peter Jackson        Elijah Wood      1485555
..      ...          ...                ...                ...          ...
968  Action          7.6    Joel Schumacher    Michael Douglas       171640
979  Action          7.6     Richard Donner         Mel Gibson       236894
982  Action          7.6      George Miller         Mel Gibson       166588
983  Action          7.6        Walter Hill       Michael Beck        93878
985  Action          7.6         Don Siegel     Clint Eastwood       121731
```

DF1

DF2

> **DF1= Action, DF2= Adenture...so on**

Highest rated movie in each genre

- We will run a loop on a group
- We will get a df
- You can apply `max()` fn to IMDB rating to that df

```
for group,data in genres:
    print(group)
    print (data['IMDB_Rating'].max())
```

```
Action
9.0
Adventure
8.6
Animation
8.6
Biography
8.9
Comedy
8.6
Crime
9.2
Drama
9.3
```

```
for group,data in genres:
    print(group)
    print(data[data['IMDB_Rating']==data['IMDB_Rating'].max()])
```

```
Action
      Series_Title Released_Year  Runtime   Genre  IMDB_Rating  \
2  The Dark Knight          2008      152  Action          9.0

            Director           Star1  No_of_Votes      Gross  Metascore
2  Christopher Nolan  Christian Bale      2303232  534858444       84.0
Adventure
    Series_Title Released_Year  Runtime      Genre  IMDB_Rating  \
21  Interstellar          2014      169  Adventure          8.6

             Director                Star1  No_of_Votes      Gross  Metascore
21  Christopher Nolan  Matthew McConaughey      1512360  188020017       74.0
Animation
                     Series_Title Released_Year  Runtime      Genre  \
23  Sen to Chihiro no kamikakushi          2001      125  Animation

    IMDB_Rating        Director          Star1  No_of_Votes     Gross  \
23          8.6  Hayao Miyazaki  Daveigh Chase       651376  10055859

    Metascore
23       96.0
```

- This will give movie name as well
- `data[data['IMDB_Rating'] == data['IMDB_Rating'].max()]` filters the rows where the `IMDB_Rating` equals the maximum rating in that genre.
  - `data['IMDB_Rating']` is just the column of ratings for that group.
- `data[data['IMDB_Rating']`
  - first `data` is the **entire group**
  - second `data` is the **filtering operation**

If we remove 1 `data`:

`print(data['IMDB_Rating']==data['IMDB_Rating'].max())`

```
Action
2       True
5      False
8      False
10     False
13     False
       ...
968    False
979    False
982    False
983    False
985    False
Name: IMDB_Rating, Length: 172, dtype: bool
Adventure
21      True
47     False
93     False
110    False
```

Apply (Split-Apply-Combine)

```
genres.apply(min)
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
Genre										
Action	300	1924	45	Action	7.6	Abhishek Chaubey	Aamir Khan	25312	3296	NaN
Adventure	2001: A Space Odyssey	1925	88	Adventure	7.6	Akira Kurosawa	Aamir Khan	29999	61001	NaN
Animation	Akira	1940	71	Animation	7.6	Adam Elliot	Adrian Molina	25229	128985	NaN
Biography	12 Years a Slave	1928	93	Biography	7.6	Adam McKay	Adrien Brody	27254	21877	NaN
Comedy	(500) Days of Summer	1921	68	Comedy	7.6	Alejandro G. Iñárritu	Aamir Khan	26337	1305	NaN
Crime	12 Angry Men	1931	80	Crime	7.6	Akira Kurosawa	Ajay Devgn	27712	6013	NaN

- **Working same as a normal min fn →** `genres.min()`

Warning:

▼

- In the future version of Pandas, `min()` will behave differently. Instead of internally using `np.minimum.reduce`, Pandas will directly apply the `min` function as you provided.
- To keep the current behavior and avoid this warning, you can explicitly use `np.minimum.reduce` in your code.

```
import numpy as np

genres.apply(np.minimum.reduce)
```

- You're using `apply()` on a `groupby` object (`genres` in this case). The warning tells you that, currently, when you apply a function like `min()`, the grouping columns (in your case, "Genre") are included in the operation.
- However, in future versions of Pandas, the grouping columns will **not** be included by default. If you want to exclude them, you'll need to explicitly pass `include_groups=False` or select the grouping columns yourself.

Pass `include_groups=False` explicitly to exclude the grouping columns from the operation:

```
genres.apply(min, include_groups=False)
```

AVOID BOTH WARNINGS

```
import numpy as np
genres.apply(np.minimum.reduce, include_groups=False)
```

- The beauty of the apply function is that you can insert your custom logic in it.

Q. Find out the number of movies starting with letter A.

```
def mova (group):
    print(group)
    return group
genres.apply(mova)
```

- `apply()` is used on the `genres` groupby object. It applies the `mova` function to each group (i.e., each genre).
- For each genre, the `mova` function is called, it prints the genre's data, and then it returns the same data back.

💡 **The function receives only the DataFrame of that group, not the group name.**

```
def foo(group):
  return group['Series_Title'].str.startswith('A').sum()

genres.apply(foo)

Output:
Genre
Action       10
Adventure     2
Animation     2
Biography     9
Comedy       14
Crime         4
Drama        21
Family        0
Fantasy       0
Film-Noir     0
Horror        1
Mystery       0
```

```
Thriller      0
Western       0
dtype: int64
```

Rank movies on basis of IMDB Rating

```
def rank(group):
  group ['genre_rank'] = group['IMDB_Rating'].rank(ascending=False)
  return group
genres.apply(rank)
```

		Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore	genre_rank
Genre												
Action	2	The Dark Knight	2008	152	Action	9.0	Christopher Nolan	Christian Bale	2303232	534858444	84.0	1.0
	5	The Lord of the Rings: The Return of the King	2003	201	Action	8.9	Peter Jackson	Elijah Wood	1642758	377845905	94.0	2.0
	8	Inception	2010	148	Action	8.8	Christopher Nolan	Leonardo DiCaprio	2067042	292576195	74.0	3.5
	10	The Lord of the Rings: The Fellowship of the Ring	2001	178	Action	8.8	Peter Jackson	Elijah Wood	1661481	315544750	92.0	3.5
	13	The Lord of the Rings: The Two Towers	2002	179	Action	8.7	Peter Jackson	Elijah Wood	1485555	342551365	87.0	6.0
...	...	...	...	...	...	...	...	...	...	...	...	...
Thriller	700	Wait Until Dark	1967	108	Thriller	7.8	Terence Young	Audrey Hepburn	27733	17550741	81.0	1.0
Western	12	Il buono, il brutto, il cattivo	1966	161	Western	8.8	Sergio Leone	Clint Eastwood	688390	6100000	90.0	1.0

`group ['genre_rank']` → We added a new column “genre_rank”

`group['IMDB_Rating'].rank(ascending=False)` → We ranked the IMDB rating in descending order

`return group` → will return the entire group

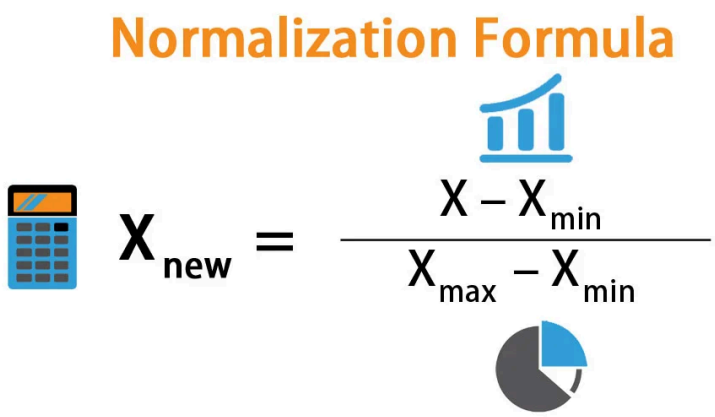
Print only specific columns:

```
def rank(group):
  group ['genre_rank'] = group['IMDB_Rating'].rank(ascending=False)
  return group[['genre_rank', 'Series_Title', 'IMDB_Rating']]
genres.apply(rank)
```

Genre		genre_rank	Series_Title	IMDB_Rating
Action	2	1.0	The Dark Knight	9.0
	5	2.0	The Lord of the Rings: The Return of the King	8.9
	8	3.5	Inception	8.8
	10	3.5	The Lord of the Rings: The Fellowship of the Ring	8.8
	13	6.0	The Lord of the Rings: The Two Towers	8.7
...	...	...	...	...
Thriller	700	1.0	Wait Until Dark	7.8
Western	12	1.0	Il buono, il brutto, il cattivo	8.8
	48	2.0	Once Upon a Time in the West	8.5
	115	3.0	Per qualche dollaro in più	8.3
	691	4.0	The Outlaw Josey Wales	7.8

Q. find normalized IMDB rating group wise

normalization formula: **a + (((x - xminimum) * (b - a)) / range of x**




```
def normal (group):
    group['norm_rating']= (group['IMDB_Rating'] - group['IMDB_Rating'].min()) / (g
```

```
    return group
genres.apply(normal)
```

Genre		Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore	norm_rating
Action	2	The Dark Knight	2008	152	Action	9.0	Christopher Nolan	Christian Bale	2303232	534858444	84.0	1.000000
	5	The Lord of the Rings: The Return of the King	2003	201	Action	8.9	Peter Jackson	Elijah Wood	1642758	377845905	94.0	0.928571
	8	Inception	2010	148	Action	8.8	Christopher Nolan	Leonardo DiCaprio	2067042	292576195	74.0	0.857143
	10	The Lord of the Rings: The Fellowship of the Ring	2001	178	Action	8.8	Peter Jackson	Elijah Wood	1661481	315544750	92.0	0.857143
	13	The Lord of the Rings: The Two Towers	2002	179	Action	8.7	Peter Jackson	Elijah Wood	1485555	342551365	87.0	0.785714
...	...	...	...	...	...	...	...	...	...	...	...	...
Thriller	700	Wait Until Dark	1967	108	Thriller	7.8	Terence Young	Audrey Hepburn	27733	17550741	81.0	NaN
Western	12	Il buono, il brutto, il cattivo	1966	161	Western	8.8	Sergio Leone	Clint Eastwood	688390	6100000	90.0	1.000000
	48	Once Upon a Time in the West	1968	165	Western	8.5	Sergio Leone	Henry Fonda	302844	5321508	80.0	0.700000
	115	Per qualche dollaro in più	1965	132	Western	8.3	Sergio Leone	Clint Eastwood	232772	15000000	74.0	0.500000

Groupby on Multiple Columns

```
duo = movies.groupby(['Director','Star1'])
duo
duo.size()
```

```
Director              Star1
Aamir Khan            Amole Gupte        1
Aaron Sorkin          Eddie Redmayne     1
Abdellatif Kechiche   Léa Seydoux        1
Abhishek Chaubey      Shahid Kapoor      1
Abhishek Kapoor       Amit Sadh          1
                                         ..
Zaza Urushadze        Lembit Ulfsak      1
Zoya Akhtar           Hrithik Roshan     1
                      Vijay Varma        1
Çagan Irmak           Çetin Tekindor     1
Ömer Faruk Sorak      Cem Yilmaz         1
Length: 898, dtype: int64
```

Know the movie

```
duo.get_group(('Aamir Khan','Amole Gupte'))
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
65	Taare Zameen Par	2007	165	Drama	8.4	Aamir Khan	Amole Gupte	168895	1223869	NaN

Q. find the most earning actor→director combo

```
duo['Gross'].sum().sort_values(ascending=False).head(1)
```

```
Director        Star1
Akira Kurosawa  Toshirô Mifune    2999876948
Name: Gross, dtype: int64
```

Q. Find the best(in-terms of metascore(avg)) actor→genre combo

```
movies.groupby(['Star1','Genre'])['Metascore'].mean().reset_index()
```

	Star1	Genre	Metascore
0	Aamir Khan	Action	NaN
1	Aamir Khan	Adventure	84.0
2	Aamir Khan	Comedy	67.0
3	Aaron Taylor-Johnson	Action	66.0
4	Abhay Deol	Drama	NaN
...	...	...	...
824	Zbigniew Zamachowski	Comedy	88.0
825	Zooey Deschanel	Comedy	76.0
826	Çetin Tekindor	Drama	NaN
827	Éric Toledano	Biography	57.0

- When you reset index, series is converted into df.

```
movies.groupby(['Star1','Genre'])['Metascore'].mean().reset_index().sort_values(
```

	Star1	Genre	Metascore
230	Ellar Coltrane	Drama	100.0

Apply aggregate

```
duo.agg(['min','max'])
```

		Series_Title		Released_Year		Runtime		Genre	
		min	max	min	max	min	max	min	max
Director	Star1								
Aamir Khan	Amole Gupte	Taare Zameen Par	Taare Zameen Par	2007	2007	165	165	Drama	Drama
Aaron Sorkin	Eddie Redmayne	The Trial of the Chicago 7	The Trial of the Chicago 7	2020	2020	129	129	Drama	Drama
Abdellatif Kechiche	Léa Seydoux	La vie d'Adèle	La vie d'Adèle	2013	2013	180	180	Drama	Drama
Abhishek Chaubey	Shahid Kapoor	Udta Punjab	Udta Punjab	2016	2016	148	148	Action	Action
Abhishek Kapoor	Amit Sadh	Kai po che!	Kai po che!	2013	2013	130	130	Drama	Drama

IPL DATASET

```
import pandas as pd
import numpy as np
ipl = pd.read_csv(r"C:\Users\Jeevan\Downloads\deliveries.csv")
ipl.shape
ipl.head()
```

shape =(179078, 21)

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...	bye_runs	legbye_runs	noball_runs	penalty_runs	batsman_runs	extra_runs	total_runs	p
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0	0	0	0	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0	0	0	0	
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0	4	0	4	
3	1	1	Sunrisers	Royal	1	4	DA	S Dhawan	TS	0		0	0	0	0	0	0	0	

- It’s a ball by ball data in IPL

Q. find the top 10 batsman in terms of runs scored

- Group by batsman
- Apply `sum` fn to **`batsman_runs`**

```
ipl.groupby('batsman')['batsman_runs'].sum().sort_values(ascending=False).hea
```

```
batsman
V Kohli           5434
SK Raina          5415
RG Sharma         4914
DA Warner         4741
S Dhawan          4632
CH Gayle          4560
MS Dhoni          4477
RV Uthappa        4446
AB de Villiers    4428
G Gambhir         4223
Name: batsman_runs, dtype: int64
```

Q. find the batsman with max no of sixes

- You will filter out the balls which were hit six

```
ipl[ipl['batsman_runs']==6][['batsman','batsman_runs']]
```

	batsman	batsman_runs
10	DA Warner	6
47	MC Henriques	6
75	Yuvraj Singh	6
89	Yuvraj Singh	6
91	MC Henriques	6
...	...	...
178987	SR Watson	6
179048	DJ Bravo	6
179061	SR Watson	6
179062	SR Watson	6
179063	SR Watson	6

8170 rows × 2 columns

`[['batsman','batsman_runs']]` → Gives us only these 2 columns

- We will store this in a new variable `six`

```
six= ipl[ipl['batsman_runs']==6][['batsman','batsman_runs']]
six.groupby('batsman')['batsman'].count().sort_values(ascending=False).head(1)

Output:
batsman
CH Gayle    327
Name: batsman, dtype: int64
```

This can also be done with valuecount

```
six= ipl[ipl['batsman_runs']==6][['batsman','batsman_runs']]
six.groupby('batsman').value_counts().sort_values(ascending=False).head(1)
```

`index[0]` will print the batsman name

```
six= ipl[ipl['batsman_runs']==6][['batsman','batsman_runs']]
six.groupby('batsman').value_counts().sort_values(ascending=False).head().inde

Output:('CH Gayle', 6)
```

Q. find batsman with most number of 4's and 6's in last 5 overs

- Filter the data with over 16-20

```
temp_df= ipl[ipl['over']>15]
```

- Now we will filter `batsman_runs` 4 & 6

```
temp_df= temp_df[(temp_df['batsman_runs']==4) | (temp_df['batsman_runs']==
```

- Now group `temp_df` with batsman.

```
temp_df = ipl[ipl['over'] > 15]
temp_df = temp_df[(temp_df['batsman_runs'] == 4) | (temp_df['batsman_runs'] =
temp_df.groupby('batsman')['batsman'].count().sort_values(ascending=False).he
```

```
batsman
MS Dhoni          340
AB de Villiers    208
RG Sharma         208
KA Pollard        206
V Kohli           160
Name: batsman, dtype: int64
```

Q. find V Kohli's record against all teams

- We have to find out how many runs he scored against each team.
- We will filter with batsman kohli

```
temp_df = ipl[ipl['batsman'] == 'V Kohli']
```

- We will do `groupby` on top of `bowling_team`

```
temp_df = ipl[ipl['batsman'] == 'V Kohli']
temp_df.groupby('bowling_team')['batsman_runs'].sum().reset_index()
```

	bowling_team	batsman_runs
0	Chennai Super Kings	749
1	Deccan Chargers	306
2	Delhi Capitals	66
3	Delhi Daredevils	763
4	Gujarat Lions	283
5	Kings XI Punjab	636
6	Kochi Tuskers Kerala	50
7	Kolkata Knight Riders	675
8	Mumbai Indians	628
9	Pune Warriors	128
10	Rajasthan Royals	370
11	Rising Pune Supergiant	83
12	Rising Pune Supergiants	188
13	Sunrisers Hyderabad	509

Q. Create a function that can return the highest score of any batsman

- Yo will input a name and it will tell his highest score.

```
temp_df = ipl[ipl['batsman'] == 'V Kohli']
temp_df.groupby('match_id')['batsman_runs'].sum().sort_values(ascending=Fals

Output:
match_id
626    113
Name: batsman_runs, dtype: int64
```

- We will create a fn and paste the above code in it

```
def highest(batsman):
  temp_df = ipl[ipl['batsman'] == batsman]
  return temp_df.groupby('match_id')['batsman_runs'].sum().sort_values(ascendi

highest('DA Warner')

Output: 126
```

Q. Virat Most no. of 6s

```
temp_df = ipl[(ipl['batsman'] == 'V Kohli') & (ipl['batsman_runs'] == 6)]

temp_df.groupby('bowling_team')['batsman_runs'].count().reset_index()
```

	bowling_team	batsman_runs
0	Chennai Super Kings	30
1	Deccan Chargers	14
2	Delhi Capitals	3
3	Delhi Daredevils	22
4	Gujarat Lions	11
5	Kings XI Punjab	18
6	Kolkata Knight Riders	22
7	Mumbai Indians	24
8	Pune Warriors	5
9	Rajasthan Royals	10
10	Rising Pune Supergiant	2
11	Rising Pune Supergiants	9
12	Sunrisers Hyderabad	21