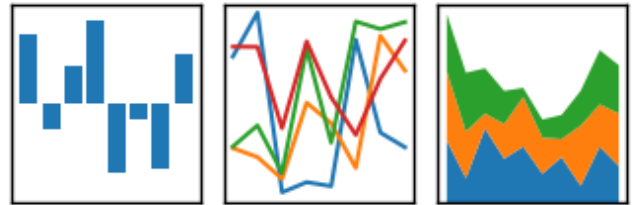


# Pandas groupby

groupby in pandas is a function that lets you group data in a DataFrame based on specific criteria, and then apply aggregate functions to each group. It's a powerful tool for data analysis that allows you to quickly and easily calculate summary statistics for your data.

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$


```
In [1]: import numpy as np
import pandas as pd
```

## Groupby (Applies on Categorical data)

```
In [2]: movies = pd.read_csv("imdb-top-1000.csv")
movies.head(3)
```

```
Out[2]:
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes
0	The Shawshank Redemption	1994	142	Drama	9.3	Frank Darabont	Tim Robbins	2343110
1	The Godfather	1972	175	Crime	9.2	Francis Ford Coppola	Marlon Brando	1620367
2	The Dark Knight	2008	152	Action	9.0	Christopher Nolan	Christian Bale	2303232

```
In [3]: movies.groupby('Genre')
```

```
Out[3]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000024A502D5550>
```

```
In [4]: genres = movies.groupby('Genre')
```

```
In [5]: # Applying builtin aggregation functions on groupby objects
genres.sum().head(2)
```

```
Out[5]:
```

	Runtime	IMDB_Rating	No_of_Votes	Gross	Metascore
<b>Genre</b>					
<b>Action</b>	22196	1367.3	72282412	3.263226e+10	10499.0
<b>Adventure</b>	9656	571.5	22576163	9.496922e+09	5020.0

```
In [6]: #Second Approach
movies.groupby('Genre')['Gross'].sum().sort_values()
```

```
Out[6]: Genre
Thriller      1.755074e+07
Western       5.822151e+07
Film-Noir     1.259105e+08
Family        4.391106e+08
Fantasy       7.827267e+08
Horror        1.034649e+09
Mystery       1.256417e+09
Biography     8.276358e+09
Crime         8.452632e+09
Adventure     9.496922e+09
Animation     1.463147e+10
Comedy        1.566387e+10
Action        3.263226e+10
Drama         3.540997e+10
Name: Gross, dtype: float64
```

```
In [7]: # find the genre with highest avg IMDB rating
movies.groupby('Genre')['IMDB_Rating'].mean().sort_values(ascending=False).head(1)
```

```
Out[7]: Genre
Western      8.35
Name: IMDB_Rating, dtype: float64
```

```
In [8]: # find director with most popularity
movies.groupby('Director')['No_of_Votes'].sum().sort_values(ascending=False).head(1)
```

```
Out[8]: Director
Christopher Nolan    11578345
Name: No_of_Votes, dtype: int64
```

```
In [9]: # find the highest rated movie of each genre
movies.head(1)
```

```
Out[9]:
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes
0	The Shawshank Redemption	1994	142	Drama	9.3	Frank Darabont	Tim Robbins	2343110

```
In [10]: # find number of movies done by each actor
#movies['Star1'].value_counts() # first method
movies.groupby('Star1')['Series_Title'].count().sort_values(ascending=False)
```

```
Out[10]: Star1
Tom Hanks          12
Robert De Niro     11
Clint Eastwood     10
Al Pacino          10
Leonardo DiCaprio  9
..
Glen Hansard        1
Giuseppe Battiston  1
Giulietta Masina    1
Gerardo Taracena    1
Ömer Faruk Sorak    1
Name: Series_Title, Length: 660, dtype: int64
```

## GroupBy Attributes and Methods

```
In [12]: # find total number of groups -> len
len(movies.groupby('Genre')) # 14 different groups
```

```
Out[12]: 14
```

```
In [13]: movies['Genre'].nunique() # second method
```

```
Out[13]: 14
```

```
In [14]: # find items in each group -> size
movies.groupby('Genre').size() # index based
```

```
Out[14]: 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
```

```
In [15]: movies['Genre'].value_counts() # second method
```

```
Out[15]: Drama      289
Action      172
Comedy       155
Crime        107
Biography     88
Animation     82
Adventure     72
Mystery       12
Horror        11
Western        4
Film-Noir     3
Fantasy       2
Family        2
Thriller       1
Name: Genre, dtype: int64
```

```
In [18]: # first()/last() -> nth item
#movies.groupby('Genre').first()
#movies.groupby('Genre').last()
movies.groupby('Genre').nth(6) ----> #gives 7th movie
```

Out[18]:

	Series_Title	Released_Year	Runtime	IMDB_Rating	Director	Star1	No_of_Vote
Genre							
<b>Action</b>	Star Wars: Episode V - The Empire Strikes Back	1980	124	8.7	Irvin Kershner	Mark Hamill	115931
<b>Adventure</b>	North by Northwest	1959	136	8.3	Alfred Hitchcock	Cary Grant	29915
<b>Animation</b>	WALL·E	2008	98	8.4	Andrew Stanton	Ben Burtt	99975
<b>Biography</b>	Braveheart	1995	178	8.3	Mel Gibson	Mel Gibson	95915
<b>Comedy</b>	The Great Dictator	1940	125	8.4	Charles Chaplin	Charles Chaplin	20315
<b>Crime</b>	Se7en	1995	127	8.6	David Fincher	Morgan Freeman	144505
<b>Drama</b>	It's a Wonderful Life	1946	130	8.6	Frank Capra	James Stewart	40580
<b>Horror</b>	Get Out	2017	104	7.7	Jordan Peele	Daniel Kaluuya	49285
<b>Mystery</b>	Sleuth	1972	138	8.0	Joseph L. Mankiewicz	Laurence Olivier	4474

```
In [19]: # get_group -> vs filtering
movies.groupby('Genre').get_group('Horror')
```

Out[19]:

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Vo
49	Psycho	1960	109	Horror	8.5	Alfred Hitchcock	Anthony Perkins	6042
75	Alien	1979	117	Horror	8.4	Ridley Scott	Sigourney Weaver	7878
271	The Thing	1982	109	Horror	8.1	John Carpenter	Kurt Russell	3712
419	The Exorcist	1973	122	Horror	8.0	William Friedkin	Ellen Burstyn	3623
544	Night of the Living Dead	1968	96	Horror	7.9	George A. Romero	Duane Jones	1165
707	The Innocents	1961	100	Horror	7.8	Jack Clayton	Deborah Kerr	270
724	Get Out	2017	104	Horror	7.7	Jordan Peele	Daniel Kaluuya	4928
844	Halloween	1978	91	Horror	7.7	John Carpenter	Donald Pleasence	2331
876	The Invisible Man	1933	71	Horror	7.7	James Whale	Claude Rains	306
932	Saw	2004	103	Horror	7.6	James Wan	Cary Elwes	3790
948	The Others	2001	101	Horror	7.6	Alejandro Amenábar	Nicole Kidman	3376

```
In [20]: movies.groupby('Genre').get_group('Fantasy')
```

Out[20]:

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes
321	Das Cabinet des Dr. Caligari	1920	76	Fantasy	8.1	Robert Wiene	Werner Krauss	57428
568	Nosferatu	1922	94	Fantasy	7.9	F.W. Murnau	Max Schreck	88794

```
In [21]: movies[movies['Genre']=='Fantasy'] #Second Method
```

Out[21]:

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes
321	Das Cabinet des Dr. Caligari	1920	76	Fantasy	8.1	Robert Wiene	Werner Krauss	57428
568	Nosferatu	1922	94	Fantasy	7.9	F.W. Murnau	Max Schreck	88794

```
In [24]: # groups
movies.groupby('Genre').groups ---> #Dictionary= gives index postion
```

```
Out[24]: {'Action': [2, 5, 8, 10, 13, 14, 16, 29, 30, 31, 39, 42, 44, 55, 57, 59, 60, 63, 68, 72, 106, 109, 129, 130, 134, 140, 142, 144, 152, 155, 160, 161, 166, 168, 171, 172, 177, 181, 194, 201, 202, 216, 217, 223, 224, 236, 241, 262, 275, 294, 308, 320, 325, 326, 331, 337, 339, 340, 343, 345, 348, 351, 353, 356, 357, 362, 368, 369, 375, 376, 390, 410, 431, 436, 473, 477, 479, 482, 488, 493, 496, 502, 507, 511, 532, 535, 540, 543, 564, 569, 570, 573, 577, 582, 583, 602, 605, 608, 615, 623, ...], 'Adventure': [21, 47, 93, 110, 114, 116, 118, 137, 178, 179, 191, 193, 209, 226, 231, 247, 267, 273, 281, 300, 301, 304, 306, 323, 329, 361, 366, 377, 402, 406, 415, 426, 458, 470, 497, 498, 506, 513, 514, 537, 549, 552, 553, 566, 576, 604, 609, 618, 638, 647, 675, 681, 686, 692, 711, 713, 739, 755, 781, 797, 798, 851, 873, 884, 912, 919, 947, 957, 964, 966, 984, 991], 'Animation': [23, 43, 46, 56, 58, 61, 66, 70, 101, 135, 146, 151, 158, 170, 197, 205, 211, 213, 219, 229, 230, 242, 245, 246, 270, 330, 332, 358, 367, 378, 386, 389, 394, 395, 399, 401, 405, 409, 469, 499, 510, 516, 518, 522, 578, 586, 592, 595, 596, 599, 633, 640, 643, 651, 665, 672, 694, 728, 740, 741, 744, 756, 758, 761, 771, 783, 796, 799, 822, 828, 843, 875, 891, 892, 902, 906, 920, 956, 971, 976, 986, 992], 'Biography': [7, 15, 18, 35, 38, 54, 102, 107, 131, 139, 147, 157, 159, 173, 176, 212, 215, 218, 228, 235, 243, 263, 276, 282, 290, 298, 317, 328, 338, 342, 346, 359, 360, 365, 372, 373, 385, 411, 416, 418, 424, 429, 484, 525, 536, 542, 545, 575, 579, 587, 600, 606, 614, 622, 632, 635, 644, 649, 650, 657, 671, 673, 684, 729, 748, 753, 757, 759, 766, 770, 779, 809, 810, 815, 820, 831, 849, 858, 877, 882, 897, 910, 915, 923, 940, 949, 952, 987], 'Comedy': [19, 26, 51, 52, 64, 78, 83, 95, 96, 112, 117, 120, 127, 128, 132, 153, 169, 183, 192, 204, 207, 208, 214, 221, 233, 238, 240, 250, 251, 252, 256, 261, 266, 277, 284, 311, 313, 316, 318, 322, 327, 374, 379, 381, 392, 396, 403, 413, 414, 417, 427, 435, 445, 446, 449, 455, 459, 460, 463, 464, 466, 471, 472, 475, 481, 490, 494, 500, 503, 509, 526, 528, 530, 531, 533, 538, 539, 541, 547, 557, 558, 562, 563, 565, 574, 591, 593, 594, 598, 613, 626, 630, 660, 662, 667, 679, 680, 683, 687, 701, ...], 'Crime': [1, 3, 4, 6, 22, 25, 27, 28, 33, 37, 41, 71, 77, 79, 86, 87, 103, 108, 111, 113, 123, 125, 133, 136, 162, 163, 164, 165, 180, 186, 187, 189, 198, 222, 232, 239, 255, 257, 287, 288, 299, 305, 335, 363, 364, 380, 384, 397, 437, 438, 441, 442, 444, 450, 451, 465, 474, 480, 485, 487, 505, 512, 519, 520, 523, 527, 546, 556, 560, 584, 597, 603, 607, 611, 621, 639, 653, 664, 669, 676, 695, 708, 723, 762, 763, 767, 775, 791, 795, 802, 811, 823, 827, 833, 885, 895, 921, 922, 926, 938, ...], 'Drama': [0, 9, 11, 17, 20, 24, 32, 34, 36, 40, 45, 50, 53, 62, 65, 67, 73, 74, 76, 80, 82, 84, 85, 88, 89, 90, 91, 92, 94, 97, 98, 99, 100, 104, 105, 121, 122, 124, 126, 138, 141, 143, 148, 149, 150, 154, 156, 167, 174, 175, 182, 184, 185, 188, 190, 195, 196, 199, 200, 203, 206, 210, 225, 227, 234, 237, 244, 248, 249, 253, 254, 258, 259, 260, 264, 265, 268, 269, 272, 274, 278, 279, 280, 283, 285, 286, 289, 291, 292, 293, 295, 296, 297, 302, 303, 307, 310, 312, 314, 315, ...], 'Family': [688, 698], 'Fantasy': [321, 568], 'Film-Noir': [309, 456, 712], 'Horror': [49, 75, 271, 419, 544, 707, 724, 844, 876, 932, 948], 'Mystery': [69, 81, 119, 145, 220, 393, 420, 714, 829, 899, 959, 961], 'Thriller': [700], 'Western': [12, 48, 115, 691]}
```

```
In [25]: # describe
movies.groupby('Genre').describe()
```

```
Out[25]:
```

	Runtime								IMDB_Rating		
	count	mean	std	min	25%	50%	75%	max	count	mean	
Genre											
Action	172.0	129.046512	28.500706	45.0	110.75	127.5	143.25	321.0	172.0	7.949419	.
Adventure	72.0	134.111111	33.317320	88.0	109.00	127.0	149.00	228.0	72.0	7.937500	.
Animation	82.0	99.585366	14.530471	71.0	90.00	99.5	106.75	137.0	82.0	7.930488	.
Biography	88.0	136.022727	25.514466	93.0	120.00	129.0	146.25	209.0	88.0	7.938636	.
Comedy	155.0	112.129032	22.946213	68.0	96.00	106.0	124.50	188.0	155.0	7.901290	.
Crime	107.0	126.392523	27.689231	80.0	106.50	122.0	141.50	229.0	107.0	8.016822	.
Drama	289.0	124.737024	27.740490	64.0	105.00	121.0	137.00	242.0	289.0	7.957439	.
Family	2.0	107.500000	10.606602	100.0	103.75	107.5	111.25	115.0	2.0	7.800000	.
Fantasy	2.0	85.000000	12.727922	76.0	80.50	85.0	89.50	94.0	2.0	8.000000	.
Film-Noir	3.0	104.000000	4.000000	100.0	102.00	104.0	106.00	108.0	3.0	7.966667	.
Horror	11.0	102.090909	13.604812	71.0	98.00	103.0	109.00	122.0	11.0	7.909091	.
Mystery	12.0	119.083333	14.475423	96.0	110.75	117.5	130.25	138.0	12.0	7.975000	.
Thriller	1.0	108.000000	NaN	108.0	108.00	108.0	108.00	108.0	1.0	7.800000	.
Western	4.0	148.250000	17.153717	132.0	134.25	148.0	162.00	165.0	4.0	8.350000	.

14 rows × 40 columns





```
In [30]: # sample  
movies.groupby('Genre').sample(2,replace=True)
```

Out[30]:

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_
308	White Heat	1949	114	Action	8.1	Raoul Walsh	James Cagney	
171	Die Hard	1988	132	Action	8.2	John McTiernan	Bruce Willis	
300	Ben-Hur	1959	212	Adventure	8.1	William Wyler	Charlton Heston	
47	Back to the Future	1985	116	Adventure	8.5	Robert Zemeckis	Michael J. Fox	
640	Les triplettes de Belleville	2003	80	Animation	7.8	Sylvain Chomet	Michèle Caucheteux	
799	South Park: Bigger, Longer & Uncut	1999	81	Animation	7.7	Trey Parker	Trey Parker	
632	The World's Fastest Indian	2005	127	Biography	7.8	Roger Donaldson	Anthony Hopkins	
372	Mar adentro	2014	126	Biography	8.0	Alejandro Amenábar	Javier Bardem	
846	Love and Death	1975	85	Comedy	7.7	Woody Allen	Woody Allen	
660	The Sandlot	1993	101	Comedy	7.8	David Mickey Evans	Tom Guiry	
397	Bound by Honor	1993	180	Crime	8.0	Taylor Hackford	Damian Chapa	
108	Scarface	1983	170	Crime	8.3	Brian De Palma	Al Pacino	
53	Capharnaüm	2018	126	Drama	8.4	Nadine Labaki	Zain Al Rafeea	
894	Creed	2015	133	Drama	7.6	Ryan Coogler	Michael B. Jordan	
688	E.T. the Extra-Terrestrial	1982	115	Family	7.8	Steven Spielberg	Henry Thomas	
688	E.T. the Extra-Terrestrial	1982	115	Family	7.8	Steven Spielberg	Henry Thomas	
568	Nosferatu	1922	94	Fantasy	7.9	F.W. Murnau	Max Schreck	
321	Das Cabinet des Dr. Caligari	1920	76	Fantasy	8.1	Robert Wiene	Werner Krauss	
309	The Third Man	1949	104	Film-Noir	8.1	Carol Reed	Orson Welles	
712	Shadow of a Doubt	1943	108	Film-Noir	7.8	Alfred Hitchcock	Teresa Wright	
49	Psycho	1960	109	Horror	8.5	Alfred Hitchcock	Anthony Perkins	

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_
948	The Others	2001	101	Horror	7.6	Alejandro Amenábar	Nicole Kidman	
81	Rear Window	1954	112	Mystery	8.4	Alfred Hitchcock	James Stewart	
959	Dark City	1998	100	Mystery	7.6	Alex Proyas	Rufus Sewell	
700	Wait Until Dark	1967	108	Thriller	7.8	Terence Young	Audrey Hepburn	
700	Wait Until Dark	1967	108	Thriller	7.8	Terence Young	Audrey Hepburn	
691	The Outlaw Josey Wales	1976	135	Western	7.8	Clint Eastwood	Clint Eastwood	
48	Once Upon a Time in the West	1968	165	Western	8.5	Sergio Leone	Henry Fonda	

In [31]: `# nunique`  
`movies.groupby('Genre').nunique() # unique --> unique items , nunique--> gives`

Out[31]:

	Series_Title	Released_Year	Runtime	IMDB_Rating	Director	Star1	No_of_Votes	Gr
Genre								
Action		172	61	78	15	123	121	172
Adventure		72	49	58	10	59	59	72
Animation		82	35	41	11	51	77	82
Biography		88	44	56	13	76	72	88
Comedy		155	72	70	11	113	133	155
Crime		106	56	65	14	86	85	107
Drama		289	83	95	14	211	250	288
Family		2	2	2	1	2	2	2
Fantasy		2	2	2	2	2	2	2
Film-Noir		3	3	3	3	3	3	3
Horror		11	11	10	8	10	11	11
Mystery		12	11	10	8	10	11	12
Thriller		1	1	1	1	1	1	1
Western		4	4	4	4	2	2	4

## aggregate method

In [33]:

```
# passing dict
movies.groupby('Genre').agg(
    {
        'Runtime': 'mean',
        'IMDB_Rating': 'mean',
        'No_of_Votes': 'sum',
        'Gross': 'sum',
        'Metascore': 'min'
    }
)
```

Out[33]:

	Runtime	IMDB_Rating	No_of_Votes	Gross	Metascore
Genre					
<b>Action</b>	129.046512	7.949419	72282412	3.263226e+10	33.0
<b>Adventure</b>	134.111111	7.937500	22576163	9.496922e+09	41.0
<b>Animation</b>	99.585366	7.930488	21978630	1.463147e+10	61.0
<b>Biography</b>	136.022727	7.938636	24006844	8.276358e+09	48.0
<b>Comedy</b>	112.129032	7.901290	27620327	1.566387e+10	45.0
<b>Crime</b>	126.392523	8.016822	33533615	8.452632e+09	47.0
<b>Drama</b>	124.737024	7.957439	61367304	3.540997e+10	28.0
<b>Family</b>	107.500000	7.800000	551221	4.391106e+08	67.0
<b>Fantasy</b>	85.000000	8.000000	146222	7.827267e+08	NaN
<b>Film-Noir</b>	104.000000	7.966667	367215	1.259105e+08	94.0
<b>Horror</b>	102.090909	7.909091	3742556	1.034649e+09	46.0
<b>Mystery</b>	119.083333	7.975000	4203004	1.256417e+09	52.0
<b>Thriller</b>	108.000000	7.800000	27733	1.755074e+07	81.0
<b>Western</b>	148.250000	8.350000	1289665	5.822151e+07	69.0

In [37]:

```
#Passsing List
movies.groupby('Genre').agg(['min','max','mean','sum','median'])
```

Out[37]:

Genre	Runtime					IMDB_Rating					...	
	min	max	mean	sum	median	min	max	mean	sum	median		
Action	45	321	129.046512	22196	127.5	7.6	9.0	7.949419	1367.3	7.9	...	
Adventure	88	228	134.111111	9656	127.0	7.6	8.6	7.937500	571.5	7.9	...	
Animation	71	137	99.585366	8166	99.5	7.6	8.6	7.930488	650.3	7.9	...	
Biography	93	209	136.022727	11970	129.0	7.6	8.9	7.938636	698.6	7.9	...	
Comedy	68	188	112.129032	17380	106.0	7.6	8.6	7.901290	1224.7	7.9	...	
Crime	80	229	126.392523	13524	122.0	7.6	9.2	8.016822	857.8	8.0	...	
Drama	64	242	124.737024	36049	121.0	7.6	9.3	7.957439	2299.7	8.0	...	
Family	100	115	107.500000	215	107.5	7.8	7.8	7.800000	15.6	7.8	...	4
Fantasy	76	94	85.000000	170	85.0	7.9	8.1	8.000000	16.0	8.0	...	337
Film-Noir	100	108	104.000000	312	104.0	7.8	8.1	7.966667	23.9	8.0	...	
Horror	71	122	102.090909	1123	103.0	7.6	8.5	7.909091	87.0	7.8	...	
Mystery	96	138	119.083333	1429	117.5	7.6	8.4	7.975000	95.7	8.0	...	1
Thriller	108	108	108.000000	108	108.0	7.8	7.8	7.800000	7.8	7.8	...	17
Western	132	165	148.250000	593	148.0	7.8	8.8	8.350000	33.4	8.4	...	5

14 rows × 25 columns

```
In [39]: # Adding both the syntax
movies.groupby('Genre').agg(
    {
        'Runtime':['min','mean'],
        'IMDB_Rating':'mean',
        'No_of_Votes':['sum','max'],
        'Gross':'sum',
        'Metascore':'min'
    }
)
```

Out[39]:

		Runtime		IMDB_Rating		No_of_Votes		Gross	Metascore
		min	mean	mean	sum	max	sum	min	
Genre									
<b>Action</b>	45	129.046512	7.949419	72282412	2303232	3.263226e+10		33.0	
<b>Adventure</b>	88	134.111111	7.937500	22576163	1512360	9.496922e+09		41.0	
<b>Animation</b>	71	99.585366	7.930488	21978630	999790	1.463147e+10		61.0	
<b>Biography</b>	93	136.022727	7.938636	24006844	1213505	8.276358e+09		48.0	
<b>Comedy</b>	68	112.129032	7.901290	27620327	939631	1.566387e+10		45.0	
<b>Crime</b>	80	126.392523	8.016822	33533615	1826188	8.452632e+09		47.0	
<b>Drama</b>	64	124.737024	7.957439	61367304	2343110	3.540997e+10		28.0	
<b>Family</b>	100	107.500000	7.800000	551221	372490	4.391106e+08		67.0	
<b>Fantasy</b>	76	85.000000	8.000000	146222	88794	7.827267e+08		NaN	
<b>Film-Noir</b>	100	104.000000	7.966667	367215	158731	1.259105e+08		94.0	
<b>Horror</b>	71	102.090909	7.909091	3742556	787806	1.034649e+09		46.0	
<b>Mystery</b>	96	119.083333	7.975000	4203004	1129894	1.256417e+09		52.0	
<b>Thriller</b>	108	108.000000	7.800000	27733	27733	1.755074e+07		81.0	
<b>Western</b>	132	148.250000	8.350000	1289665	688390	5.822151e+07		69.0	

```
In [40]: # Looping on groups
for group, data in movies.groupby('Genre'):
    print(data)
```

	Series_Title	Released_Year	Runtime
2	The Dark Knight	2008	
5	The Lord of the Rings: The Return of the King	2003	
8	Inception	2010	
10	The Lord of the Rings: The Fellowship of the Ring	2001	
13	The Lord of the Rings: The Two Towers	2002	
..	...	...	
968	Falling Down	1993	
979	Lethal Weapon	1987	
982	Mad Max 2	1981	

```
In [41]: # find the highest rated movie of each genre

df = pd.DataFrame(columns=movies.columns)

for group , data in movies.groupby('Genre'):
    df= df.append(data[data['IMDB_Rating'] == data['IMDB_Rating'].max()])
df
```

Out[41]:

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1
2	The Dark Knight	2008	152	Action	9.0	Christopher Nolan	Christian Bale
21	Interstellar	2014	169	Adventure	8.6	Christopher Nolan	Matthew McConaughey
23	Sen to Chihiro no kamikakushi	2001	125	Animation	8.6	Hayao Miyazaki	Daveigh Chase
7	Schindler's List	1993	195	Biography	8.9	Steven Spielberg	Liam Neeson
19	Gisaengchung	2019	132	Comedy	8.6	Bong Joon Ho	Kang-ho Song
26	La vita è bella	1997	116	Comedy	8.6	Roberto Benigni	Roberto Benigni
1	The Godfather	1972	175	Crime	9.2	Francis Ford Coppola	Marlon Brando
0	The Shawshank Redemption	1994	142	Drama	9.3	Frank Darabont	Tim Robbins
688	E.T. the Extra-Terrestrial	1982	115	Family	7.8	Steven Spielberg	Henry Thomas
698	Willy Wonka & the Chocolate Factory	1971	100	Family	7.8	Mel Stuart	Gene Wilder
321	Das Cabinet des Dr. Caligari	1920	76	Fantasy	8.1	Robert Wiene	Werner Krauss
309	The Third Man	1949	104	Film-Noir	8.1	Carol Reed	Orson Welles
49	Psycho	1960	109	Horror	8.5	Alfred Hitchcock	Anthony Perkins
69	Memento	2000	113	Mystery	8.4	Christopher Nolan	Guy Pearce
81	Rear Window	1954	112	Mystery	8.4	Alfred Hitchcock	James Stewart
700	Wait Until Dark	1967	108	Thriller	7.8	Terence Young	Audrey Hepburn
12	Il buono, il brutto, il cattivo	1966	161	Western	8.8	Sergio Leone	Clint Eastwood



## split (apply) combine

```
In [42]: # apply -> builtin function
movies.groupby('Genre').apply(min)
```

Out[42]:

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1
Genre							
<b>Action</b>	300	1924	45	Action	7.6	Abhishek Chaubey	Aamir Khan
<b>Adventure</b>	2001: A Space Odyssey	1925	88	Adventure	7.6	Akira Kurosawa	Aamir Khan
<b>Animation</b>	Akira	1940	71	Animation	7.6	Adam Elliot	Adrian Molina
<b>Biography</b>	12 Years a Slave	1928	93	Biography	7.6	Adam McKay	Adrien Brody
<b>Comedy</b>	(500) Days of Summer	1921	68	Comedy	7.6	Alejandro G. Iñárritu	Aamir Khan
<b>Crime</b>	12 Angry Men	1931	80	Crime	7.6	Akira Kurosawa	Ajay Devgn
<b>Drama</b>	1917	1925	64	Drama	7.6	Aamir Khan	Abhay Deol
<b>Family</b>	E.T. the Extra-Terrestrial	1971	100	Family	7.8	Mel Stuart	Gene Wilder
<b>Fantasy</b>	Das Cabinet des Dr. Caligari	1920	76	Fantasy	7.9	F.W. Murnau	Max Schreck
<b>Film-Noir</b>	Shadow of a Doubt	1941	100	Film-Noir	7.8	Alfred Hitchcock	Humphrey Bogart
<b>Horror</b>	Alien	1933	71	Horror	7.6	Alejandro Amenábar	Anthony Perkins
<b>Mystery</b>	Dark City	1938	96	Mystery	7.6	Alex Proyas	Bernard-Pierre Donnadieu
<b>Thriller</b>	Wait Until Dark	1967	108	Thriller	7.8	Terence Young	Audrey Hepburn
<b>Western</b>	Il buono, il brutto, il cattivo	1965	132	Western	7.8	Clint Eastwood	Clint Eastwood

```
In [43]: # find number of movies starting with A for each group
def foo(group):
    print(group) # type = Dataframe
    return group
```

```
In [44]: movies.groupby('Genre').apply(foo)
```

	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

```
In [49]: #Custom Logic
def foo(group):
    return group['Series_Title'].str.startswith('A').sum()
```

```
In [50]: movies.groupby('Genre').apply(foo)
```

```
Out[50]: 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
```

```
In [51]: # find ranking of each movie in the group according to IMDB score
def rank_movie(group):
    group['genre_rank']=group['IMDB_Rating'].rank(ascending=False)
    return group
```

```
In [52]: movies.groupby('Genre').apply(rank_movie)
```

```
Out[52]:
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_
0	The Shawshank Redemption	1994	142	Drama	9.3	Frank Darabont	Tim Robbins	23
1	The Godfather	1972	175	Crime	9.2	Francis Ford Coppola	Marlon Brando	16
2	The Dark Knight	2008	152	Action	9.0	Christopher Nolan	Christian Bale	23
3	The Godfather: Part II	1974	202	Crime	9.0	Francis Ford Coppola	Al Pacino	11
4	12 Angry Men	1957	96	Crime	9.0	Sidney Lumet	Henry Fonda	6
...	...	...	...	...	...	...	...	...
995	Breakfast at Tiffany's	1961	115	Comedy	7.6	Blake Edwards	Audrey Hepburn	1
996	Giant	1956	201	Drama	7.6	George Stevens	Elizabeth Taylor	1
997	From Here to Eternity	1953	118	Drama	7.6	Fred Zinnemann	Burt Lancaster	1
998	Lifeboat	1944	97	Drama	7.6	Alfred Hitchcock	Tallulah Bankhead	1
999	The 39 Steps	1935	86	Crime	7.6	Alfred Hitchcock	Robert Donat	1

1000 rows × 11 columns



```
In [55]: # find normalized IMDB rating group wise
#x normalized = (x - x minimum) / (x maximum - x minimum)
def normal(group):
    group['normal_rating'] = (group['IMDB_Rating'] - group['IMDB_Rating'].min())
    return group

movies.groupby('Genre').apply(normal)
```

```
Out[55]:
```

	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_
0	The Shawshank Redemption	1994	142	Drama	9.3	Frank Darabont	Tim Robbins	23
1	The Godfather	1972	175	Crime	9.2	Francis Ford Coppola	Marlon Brando	16
2	The Dark Knight	2008	152	Action	9.0	Christopher Nolan	Christian Bale	23
3	The Godfather: Part II	1974	202	Crime	9.0	Francis Ford Coppola	Al Pacino	11
4	12 Angry Men	1957	96	Crime	9.0	Sidney Lumet	Henry Fonda	6
...	...	...	...	...	...	...	...	...
995	Breakfast at Tiffany's	1961	115	Comedy	7.6	Blake Edwards	Audrey Hepburn	16
996	Giant	1956	201	Drama	7.6	George Stevens	Elizabeth Taylor	1
997	From Here to Eternity	1953	118	Drama	7.6	Fred Zinnemann	Burt Lancaster	1
998	Lifeboat	1944	97	Drama	7.6	Alfred Hitchcock	Tallulah Bankhead	1
999	The 39 Steps	1935	86	Crime	7.6	Alfred Hitchcock	Robert Donat	1

1000 rows × 11 columns

## groupby on multiple cols

```
In [57]: combo = movies.groupby(['Director', 'Star1'])
#size
combo.size()
```

```
Out[57]: 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
```

```
In [59]: #if we want particular combo--> get_group
combo.get_group(('Aamir Khan', 'Amole Gupte'))
```

```
Out[59]:
```

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

```
In [62]: # find the most earning actor-> director combo
combo['Gross'].sum().sort_values(ascending=False).head(2)
```

```
Out[62]: Director      Star1
Akira Kurosawa  Toshirô Mifune      2.999877e+09
Anthony Russo   Joe Russo          2.205039e+09
Name: Gross, dtype: float64
```

```
In [75]: # find the best(in-terms of metascore(avg)) actor->genre combo
movies.groupby(['Star1', 'Genre'])['Metascore'].mean().reset_index().sort_value
```

```
Out[75]:
```

	Star1	Genre	Metascore
230	Ellar Coltrane	Drama	100.0

```
In [77]: # agg on multiple groupby
        combo.agg(['min', 'max', 'mean'])
```

Out[77]:

		Runtime			IMDB_Rating			No_of_Votes			
		min	max	mean	min	max	mean	min	max	mean	n
Director	Star1										
Aamir Khan	Amole Gupte	165	165	165.0	8.4	8.4	8.4	168895	168895	168895.0	1223861
Aaron Sorkin	Eddie Redmayne	129	129	129.0	7.8	7.8	7.8	89896	89896	89896.0	853090411
Abdellatif Kechiche	Léa Seydoux	180	180	180.0	7.7	7.7	7.7	138741	138741	138741.0	2199671
Abhishek Chaubey	Shahid Kapoor	148	148	148.0	7.8	7.8	7.8	27175	27175	27175.0	218428301
Abhishek Kapoor	Amit Sadh	130	130	130.0	7.7	7.7	7.7	32628	32628	32628.0	1122521
...	...	...	...	...	...	...	...	...	...	...	...
Zaza Urushadze	Lembit Ulfesak	87	87	87.0	8.2	8.2	8.2	40382	40382	40382.0	144501
Zoya Akhtar	Hrithik Roshan	155	155	155.0	8.1	8.1	8.1	67927	67927	67927.0	3108481
	Vijay Varma	154	154	154.0	8.0	8.0	8.0	31886	31886	31886.0	5566531
Çagan Irmak	Çetin Tekindor	112	112	112.0	8.3	8.3	8.3	78925	78925	78925.0	461855361
Ömer Faruk Sorak	Cem Yilmaz	127	127	127.0	8.0	8.0	8.0	56960	56960	56960.0	196206071

898 rows × 15 columns

# Excercise

```
In [78]: ipl = pd.read_csv("deliveries.csv")
ipl.head(2)
```

```
Out[78]:
```

	match_id	inning	battling_team	bowling_team	over	ball	batsman	non_striker	bowler	is_s
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	

2 rows × 21 columns

```
In [80]: ipl.shape # ball by ball dataset
```

```
Out[80]: (179078, 21)
```

```
In [87]: # find the top 10 batsman in terms of runs
ipl.groupby('batsman')['batsman_runs'].sum().sort_values(ascending=False).reset_index()
```

```
Out[87]:
```

	batsman	batsman_runs
0	V Kohli	5434
1	SK Raina	5415
2	RG Sharma	4914
3	DA Warner	4741
4	S Dhawan	4632
5	CH Gayle	4560
6	MS Dhoni	4477
7	RV Uthappa	4446
8	AB de Villiers	4428
9	G Gambhir	4223

```
In [94]: # find the batsman with max no of sixes
six = ipl[ipl['batsman_runs'] == 6]

six.groupby('batsman')['batsman'].count().sort_values(ascending=False).head(2)
```

```
Out[94]: batsman
CH Gayle      327
AB de Villiers 214
Name: batsman, dtype: int64
```

```
In [105]: # find batsman with most number of 4's and 6's in last 5 overs

temp = ipl[ipl['over']>15]
temp = temp[(temp['batsman_runs'] == 4) | (temp['batsman_runs'] == 6)]

temp.groupby('batsman')['batsman'].count().sort_values(ascending=False).head(1)
```

Out[105]: 'MS Dhoni'

```
In [110]: # find V Kohli's record against all teams

temp = ipl[ipl['batsman'] == 'V Kohli']

temp.groupby('bowling_team')['batsman_runs'].sum().sort_values(ascending=False)
```

Out[110]:

	bowling_team	batsman_runs
0	Delhi Daredevils	763
1	Chennai Super Kings	749

```
In [118]: # Create a function that can return the highest score of any batsman

temp = ipl[ipl['batsman'] == 'V Kohli']

temp.groupby('match_id')['batsman_runs'].sum().sort_values(ascending=False).head(1)
```

Out[118]: 113

```
In [122]: def highest(batsman):
    temp = ipl[ipl['batsman'] == batsman]
    return temp.groupby('match_id')['batsman_runs'].sum().sort_values(ascending=False).head(1)
```

```
In [123]: highest('DA Warner')
```

Out[123]: 126

In [ ]: