Colab: https://colab.research.google.com/drive/1WJfI4t7xlS76sWnVb_J842a_8_mu0Nj8?usp=sharing

```
import pandas as pd
import numpy as np
```

```
!gdown 1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
```

```
Downloading...
From: https://drive.google.com/uc?id=1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
To: /content/movies.csv
100% 112k/112k [00:00<00:00, 79.3MB/s]
```

```
!gdown 1Ws-_s1fHZ9nHfGLVUQurbHDvStePlEJm
```

```
Downloading...
From: https://drive.google.com/uc?id=1Ws-_s1fHZ9nHfGLVUQurbHDvStePlEJm
To: /content/directors.csv
100% 65.4k/65.4k [00:00<00:00, 51.9MB/s]
```

```
movies = pd.read_csv("movies.csv", index_col=0)
movies.reset_index(drop=True, inplace=True)
movies.head()
```

	id	budget	popularity	revenue	title	vote_average	vote_count
0	43597	237000000	150	2787965087	Avatar	7.2	11800
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500
2	43599	245000000	107	880674609	Spectre	6.3	4466
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106
4	43602	258000000	115	890871626	Spider-Man 3	5.9	3576

```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
```

```
 ---  ------        --------------  -----
  0   id            1465 non-null   int64
  1   budget        1465 non-null   int64
  2   popularity    1465 non-null   int64
  3   revenue       1465 non-null   int64
  4   title         1465 non-null   object
  5   vote_average  1465 non-null   float64
  6   vote_count    1465 non-null   int64
  7   director_id   1465 non-null   int64
  8   year          1465 non-null   int64
  9   month         1465 non-null   object
  10  day           1465 non-null   object
dtypes: float64(1), int64(7), object(3)
memory usage: 126.0+ KB
```

```
directors = pd.read_csv('directors.csv',index_col=0)
directors.head()
```

	director_name	id	gender
0	James Cameron	4762	Male
1	Gore Verbinski	4763	Male
2	Sam Mendes	4764	Male
3	Christopher Nolan	4765	Male
4	Andrew Stanton	4766	Male

```
movies.shape
```

```
(1465, 11)
```

```
directors.shape
```

```
(2349, 3)
```

```
directors["director_name"].nunique()
```

```
2349
```

```
movies["director_id"].nunique()
```

```
199
```

```
movies["director_id"]
```

```
0       4762
1       4763
2       4764
3       4765
4       4767
        ...
```

```
1460    4809
1461    5369
1462    5148
1463    5535
1464    5097
Name: director_id, Length: 1465, dtype: int64
```

```
4762 in directors["id"].to_list() # one operation for checking
```

```
True
```

```
np.all(movies["director_id"].isin(directors["id"])) # vectorised op for checking
```

```
True
```

```
data = movies.merge(directors, how="left", left_on="director_id", right_on="id")
data.head()
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count
0	43597	237000000	150	2787965087	Avatar	7.2	11800
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500
2	43599	245000000	107	880674609	Spectre	6.3	4466
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106
4	43602	258000000	115	890871626	Spider-Man 3	5.9	3576

```
data.drop(["director_id", "id_y"], axis=1, inplace=True)
data.head()
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count
0	43597	237000000	150	2787965087	Avatar	7.2	11800
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1465 entries, 0 to 1464
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id_x           1465 non-null   int64  
 1   budget         1465 non-null   int64  
 2   popularity     1465 non-null   int64  
 3   revenue        1465 non-null   int64  
 4   title          1465 non-null   object 
 5   vote_average   1465 non-null   float64
 6   vote_count     1465 non-null   int64  
 7   year           1465 non-null   int64  
 8   month          1465 non-null   object 
 9   day            1465 non-null   object 
 10  director_name  1465 non-null   object 
 11  gender         1341 non-null   object 
dtypes: float64(1), int64(6), object(5)
memory usage: 148.8+ KB
```

```
data.describe()
```

	id_x	budget	popularity	revenue	vote_average	vote_co
count	1465.000000	1.465000e+03	1465.000000	1.465000e+03	1465.000000	1465.000
mean	45225.191126	4.802295e+07	30.855973	1.432539e+08	6.368191	1146.396
std	1189.096396	4.935541e+07	34.845214	2.064918e+08	0.818033	1578.077
min	43597.000000	0.000000e+00	0.000000	0.000000e+00	3.000000	1.000
25%	44236.000000	1.400000e+07	11.000000	1.738013e+07	5.900000	216.000
50%	45022.000000	3.300000e+07	23.000000	7.578164e+07	6.400000	571.000
75%	45990.000000	6.600000e+07	41.000000	1.792469e+08	6.900000	1387.000
max	48395.000000	3.800000e+08	724.000000	2.787965e+09	8.300000	13752.000

```
data.describe(include=object)
```

	title	month	day	director_name	gender
count	1465	1465	1465	1465	1341
unique	1465	12	7	199	2

```
data.describe(include="all")
```

	id_x	budget	popularity	revenue	title	vote_average
count	1465.000000	1.465000e+03	1465.000000	1.465000e+03	1465	1465.000000
unique	NaN	NaN	NaN	NaN	1465	NaN
top	NaN	NaN	NaN	NaN	Avatar	NaN
freq	NaN	NaN	NaN	NaN	1	NaN
mean	45225.191126	4.802295e+07	30.855973	1.432539e+08	NaN	6.368191
std	1189.096396	4.935541e+07	34.845214	2.064918e+08	NaN	0.818033
min	43597.000000	0.000000e+00	0.000000	0.000000e+00	NaN	3.000000
25%	44236.000000	1.400000e+07	11.000000	1.738013e+07	NaN	5.900000
50%	45022.000000	3.300000e+07	23.000000	7.578164e+07	NaN	6.400000
75%	45990.000000	6.600000e+07	41.000000	1.792469e+08	NaN	6.900000
max	48395.000000	3.800000e+08	724.000000	2.787965e+09	NaN	8.300000

```
data["revenue"] = (data["revenue"]/1000000).round(2)

data["budget"] = (data["budget"]/1000000).round(2)

data.head()
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count	year
0	43597	237.0	150	2787.97	Avatar	7.2	11800	2009
					Pirates of the			

```
# Give me the rows which have movies ratings > 7?
# SELECT * FROM movies WHERE vote_average > 7
```

```
data.loc[data["vote_average"] > 7]
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count	y
0	43597	237.00	150	2787.97	Avatar	7.2	11800	2
3	43600	250.00	112	1084.94	The Dark Knight Rises	7.6	9106	2
14	43616	250.00	120	956.02	The Hobbit: The Battle of the Five Armies	7.1	4760	2
16	43619	250.00	94	958.40	The Hobbit: The Desolation of Smaug	7.6	4524	2
19	43622	200.00	100	1845.03	Titanic	7.5	7562	1
...	...	...	...	...	...	...	...	
1456	48321	0.01	20	7.00	Eraserhead	7.5	485	1
1457	48323	0.00	5	0.00	The Mighty	7.1	51	1
1458	48335	0.06	27	3.22	Pi	7.1	586	1
1460	48363	0.00	3	0.32	The Last Waltz	7.9	64	1
1461	48370	0.03	19	3.15	Clerks	7.4	755	1

301 rows × 12 columns

```
data[data["vote_average"] > 7]
# personally, I dont like to use this,
# as it confuses me if iloc/loc is used here
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count	y
0	43597	237.00	150	2787.97	Avatar	7.2	11800	2
3	43600	250.00	112	1084.94	The Dark Knight Rises	7.6	9106	2
14	43616	250.00	120	956.02	The Hobbit: The Battle of the Five Armies	7.1	4760	2
16	43619	250.00	94	958.40	The Hobbit: The Desolation of Smaug	7.6	4524	2
19	43622	200.00	100	1845.03	Titanic	7.5	7562	1
...	...	...	...	...	...	...	...	
1456	48321	0.01	20	7.00	Eraserhead	7.5	485	1
1457	48323	0.00	5	0.00	The Mighty	7.1	51	1
1458	48335	0.06	27	3.22	Pi	7.1	586	1
1460	48363	0.00	3	0.32	The Last Waltz	7.9	64	1
1461	48370	0.03	19	3.15	Clerks	7.4	755	1

301 rows × 12 columns

```
data.loc[data["vote_average"] > 7, ["title", "director_name"]]
```

	title	director_name
0	Avatar	James Cameron
3	The Dark Knight Rises	Christopher Nolan
14	The Hobbit: The Battle of the Five Armies	Peter Jackson
16	The Hobbit: The Desolation of Smaug	Peter Jackson
19	Titanic	James Cameron
...	...	...
1456	Eraserhead	David Lynch
1457	The Mighty	Peter Chelsom
1458	Pi	Darren Aronofsky
1460	The Last Waltz	Martin Scorsese
1461	Clerks	Kevin Smith

301 rows × 2 columns

```
data[data["vote_average"] > 7][["title", "director_name"]] #2-step process
```

	title	director_name
0	Avatar	James Cameron
3	The Dark Knight Rises	Christopher Nolan
14	The Hobbit: The Battle of the Five Armies	Peter Jackson
16	The Hobbit: The Desolation of Smaug	Peter Jackson
19	Titanic	James Cameron
...	...	...
1456	Eraserhead	David Lynch
1457	The Mighty	Peter Chelsom
1458	Pi	Darren Aronofsky
1460	The Last Waltz	Martin Scorsese
1461	Clerks	Kevin Smith

301 rows × 2 columns

```
# Filter highly rated movies (vote_average > 7) but are also latest (>=2015)
data.loc[(data["vote_average"] > 7) & (data["year"] >= 2015)]
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count	ye
30	43641	190.0	102	1506.25	Furious 7	7.3	4176	2(
78	43724	150.0	434	378.86	Mad Max: Fury Road	7.2	9427	2(
106	43773	135.0	100	532.95	The Revenant	7.3	6396	2(

```
# Filter all the movies which were released in the weekend (Friday, Saturday, Sunda
data.loc[(data["day"] == "Friday") | (data["day"] == "Saturday") | (data["day"] ==
```

	id_x	budget	popularity	revenue	title	vot
1	43598	300.00	139	961.00	Pirates of the Caribbean: At World's End	
12	43614	380.00	135	1045.71	Pirates of the Caribbean: On Stranger Tides	
22	43627	200.00	35	783.77	Spider-Man 2	
25	43632	150.00	21	836.30	Transformers: Revenge of the Fallen	
40	43656	200.00	45	769.65	2012	
...	...	...	...	...	...	
1458	48335	0.06	27	3.22	Pi	
1459	48359	0.00	2	0.00	George Washington	
1462	48375	0.00	7	0.00	Rampage	
1463	48376	0.00	3	0.00	Slacker	
1464	48395	0.22	14	2.04	El Mariachi	

747 rows × 12 columns

```
data.loc[data["day"].isin(["Friday", "Saturday", "Sunday"])]
```

	id_x	budget	popularity	revenue	title	vot
1	43598	300.00	139	961.00	Pirates of the Caribbean: At World's End	
12	43614	380.00	135	1045.71	Pirates of the Caribbean: On Stranger Tides	
22	43627	200.00	35	783.77	Spider-Man 2	
25	43632	150.00	21	836.30	Transformers: Revenge of the Fallen	
40	43656	200.00	45	769.65	2012	
...	...	...	...	...	...	
1458	48335	0.06	27	3.22	Pi	
1459	48359	0.00	2	0.00	George Washington	
1462	48375	0.00	7	0.00	Rampage	
1463	48376	0.00	3	0.00	Slacker	
1464	48395	0.22	14	2.04	El Mariachi	

747 rows × 12 columns

```
# Give me details top-5 most popular movies?
data.sort_values(["popularity"], ascending=False).head(5)
```

	id_x	budget	popularity	revenue	title
58	43692	165.0	724	675.12	Interstellar

```
# Give me details of all the movie titles directed by "Christopher Nolan"?
data.loc[data["director_name"] == "Christopher Nolan", ["title"]]
```

	title
3	The Dark Knight Rises
45	The Dark Knight
58	Interstellar
59	Inception
74	Batman Begins
565	Insomnia
641	The Prestige
1341	Memento

```
# Apply
# gender, male-->0, female-->1, NaN is kept same
def encode(x):
  if x == "Male":
    return 0
  elif x == "Female":
    return 1
  else:
    return x
```

```
data["gender"] = data["gender"].apply(encode)
```

```
def profit(x):
  return x["revenue"] - x["budget"]
```

```
data[["revenue", "budget"]].apply(profit, axis=1)
```

```
0       2550.97
1        661.00
2        635.67
3        834.94
4        632.87
         ...
1460       0.32
1461       3.12
1462       0.00
1463       0.00
1464       1.82
Length: 1465, dtype: float64
```

```
data[["revenue", "budget"]].apply(profit, axis=0) # default, axis=0 for apply funct
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.8/dist-packages/pandas/core/indexes/base.py in get_loc(
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

                              9 frames
pandas/_libs/index_class_helper.pxi in pandas._libs.index.Int64Engine._check_t

pandas/_libs/index_class_helper.pxi in pandas._libs.index.Int64Engine._check_t

KeyError: 'revenue'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.8/dist-packages/pandas/core/indexes/base.py in get_loc(
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364
   3365         if is_scalar(key) and isna(key) and not self.hasnans:

KeyError: 'revenue'
```

SEARCH STACK OVERFLOW

```
data[["revenue", "budget"]].apply(np.sum, axis=0)
```

```
revenue    209867.04
budget      70353.62
dtype: float64
```

```
data[["revenue", "budget"]].apply(np.sum, axis=1)
```

```
0       3024.97
1       1261.00
2       1125.67
3       1334.94
4       1148.87
         ...
1460       0.32
1461       3.18
1462       0.00
1463       0.00
1464       2.26
Length: 1465, dtype: float64
```

```
# Groupby
```

```
# The count of movies directed by Christopher Nolan?
data.loc[data["director_name"]=="Christopher Nolan", "title"].count()
```

```
8
```

```
# The count of movies directed each director?
data["director_name"].value_counts()
```

```
Steven Spielberg      26
Martin Scorsese       19
Clint Eastwood        19
Woody Allen           18
Ridley Scott          16
                      ..
Tim Hill               5
Jonathan Liebesman     5
Roman Polanski         5
Larry Charles          5
Nicole Holofcener      5
Name: director_name, Length: 199, dtype: int64
```

```
# Average popularity of each director?
```

```
data.groupby("director_name")
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f6568b6d940>
```

```
data.groupby("director_name").ngroups
```

```
199
```

```
data.groupby("director_name").groups
```

```
{'Adam McKay': [176, 323, 366, 505, 839, 916], 'Adam Shankman': [265, 300,
350, 404, 458, 843, 999, 1231], 'Alejandro González Iñárritu': [106, 749,
1015, 1034, 1077, 1405], 'Alex Proyas': [95, 159, 514, 671, 873], 'Alexander
Payne': [793, 1006, 1101, 1211, 1281], 'Andrew Adamson': [11, 43, 328, 501,
947], 'Andrew Niccol': [533, 603, 701, 722, 1439], 'Andrzej Bartkowiak':
[349, 549, 754, 911, 924], 'Andy Fickman': [517, 681, 909, 926, 973, 1023],
'Andy Tennant': [314, 320, 464, 593, 676, 885], 'Ang Lee': [99, 134, 748,
840, 1089, 1110, 1132, 1184], 'Anne Fletcher': [610, 650, 736, 789, 1206],
'Antoine Fuqua': [310, 338, 424, 467, 576, 808, 818, 1105], 'Atom Egoyan':
[946, 1128, 1164, 1194, 1347, 1416], 'Barry Levinson': [313, 319, 471, 594,
878, 898, 1013, 1037, 1082, 1143, 1185, 1345, 1378], 'Barry Sonnenfeld': [13,
48, 90, 205, 591, 778, 783], 'Ben Stiller': [209, 212, 547, 562, 850], 'Bill
Condon': [102, 307, 902, 1233, 1381], 'Bobby Farrelly': [352, 356, 481, 498,
624, 630, 654, 806, 928, 972, 1111], 'Brad Anderson': [1163, 1197, 1350,
1419, 1430], 'Brett Ratner': [24, 39, 188, 207, 238, 292, 405, 456, 920],
'Brian De Palma': [228, 255, 318, 439, 747, 905, 919, 1088, 1232, 1261, 1317,
1354], 'Brian Helgeland': [512, 607, 623, 742, 933], 'Brian Levant': [418,
449, 568, 761, 860, 1003], 'Brian Robbins': [416, 441, 669, 962, 988, 1115],
'Bryan Singer': [6, 32, 33, 44, 122, 216, 297, 1326], 'Cameron Crowe': [335,
434, 488, 503, 513, 698], 'Catherine Hardwicke': [602, 695, 724, 937, 1406,
```

```
1412], 'Chris Columbus': [117, 167, 204, 218, 229, 509, 656, 897, 996, 1086,
1129], 'Chris Weitz': [17, 500, 794, 869, 1202, 1267], 'Christopher Nolan':
[3, 45, 58, 59, 74, 565, 641, 1341], 'Chuck Russell': [177, 410, 657, 1069,
1097, 1339], 'Clint Eastwood': [369, 426, 447, 482, 490, 520, 530, 535, 645,
727, 731, 786, 787, 899, 974, 986, 1167, 1190, 1313], 'Curtis Hanson': [494,
579, 606, 711, 733, 1057, 1310], 'Danny Boyle': [527, 668, 1083, 1085, 1126,
1168, 1287, 1385], 'Darren Aronofsky': [113, 751, 1187, 1328, 1363, 1458],
'Darren Lynn Bousman': [1241, 1243, 1283, 1338, 1440], 'David Ayer': [50,
273, 741, 1024, 1146, 1407], 'David Cronenberg': [541, 767, 994, 1055, 1254,
1268, 1334], 'David Fincher': [62, 213, 253, 383, 398, 478, 522, 555, 618,
785], 'David Gordon Green': [543, 862, 884, 927, 1376, 1418, 1432, 1459],
'David Koepp': [443, 644, 735, 1041, 1209], 'David Lynch': [583, 1161, 1264,
1340, 1456], 'David O. Russell': [422, 556, 609, 896, 982, 989, 1229, 1304],
'David R. Ellis': [582, 634, 756, 888, 934], 'David Zucker': [569, 619, 965,
1052, 1175], 'Dennis Dugan': [217, 260, 267, 293, 303, 718, 780, 977, 1247],
'Donald Petrie': [427, 507, 570, 649, 858, 894, 1106, 1331], 'Doug Liman':
[52, 148, 251, 399, 544, 1318, 1451], 'Edward Zwick': [92, 182, 346, 566,
791, 819, 825], 'F. Gary Gray': [308, 402, 491, 523, 697, 833, 1272, 1380],
'Francis Ford Coppola': [487, 559, 622, 646, 772, 1076, 1155, 1253, 1312],
'Francis Lawrence': [63, 72, 109, 120, 679], 'Frank Coraci': [157, 249, 275,
451, 577, 599, 963], 'Frank Oz': [193, 355, 473, 580, 712, 813, 987], 'Garry
Marshall': [329, 496, 528, 571, 784, 893, 1029, 1169], 'Gary Fleder': [518,
667, 689, 867, 981, 1165], 'Gary Winick': [258, 797, 798, 804, 1454], 'Gavin
O'Connor': [820, 841, 939, 953, 1444], 'George A. Romero': [250, 1066, 1096,
1278, 1367, 1396], 'George Clooney': [343, 450, 831, 966, 1302], 'George
Miller': [78, 103, 233, 287, 1250, 1403, 1450], 'Gore Verbinski': [1, 8, 9,
107, 119, 633, 1040], 'Guillermo del Toro': [35, 252, 419, 486, 1118], 'Gus
Van Sant': [595, 1018, 1027, 1159, 1240, 1311, 1398], 'Guy Ritchie': [124,
215, 312, 1093, 1225, 1269, 1420], 'Harold Ramis': [425, 431, 558, 586, 788,
1137, 1166, 1325], 'Ivan Reitman': [274, 643, 816, 883, 910, 935, 1134,
1242], 'James Cameron': [0, 19, 170, 173, 344, 1100, 1320], 'James Ivory':
[1125, 1152, 1180, 1291, 1293, 1390, 1397], 'James Mangold': [140, 141, 557,
560, 829, 845, 958, 1145], 'James Wan': [30, 617, 1002, 1047, 1337, 1417,
1424], 'Jan de Bont': [155, 224, 231, 270, 781], 'Jason Friedberg': [812,
1010, 1012, 1014, 1036], 'Jason Reitman': [792, 1092, 1213, 1295, 1299],
'Jaume Collet-Serra': [516, 540, 640, 725, 1011, 1189], 'Jay Roach': [195,
359, 389, 397, 461, 703, 859, 1072], 'Jean-Pierre Jeunet': [423, 485, 605,
664, 765], 'Joe Dante': [284, 525, 638, 1226, 1298, 1428], 'Joe Wright': [85
```

```
data.groupby("director_name").get_group("Kenny Ortega")
```

	id_x	budget	popularity	revenue	title	vote_aver
412	44316	60.0	15	0.00	This Is It	
852	45315	28.0	18	39.51	Hocus Pocus	
1228	46513	0.0	21	0.00	High School Musical 3: Senior Year	

```
data.groupby('director_name')["title"].count()
```

```
director_name
Adam McKay                      6
Adam Shankman                   8
Alejandro González Iñárritu     6
Alex Proyas                     5
Alexander Payne                 5
                               ..
Wes Craven                     10
Wolfgang Petersen               7
Woody Allen                    18
Zack Snyder                     7
Zhang Yimou                     6
Name: title, Length: 199, dtype: int64
```

```
# Average popularity of each director?
data.groupby('director_name')["popularity"].mean()
# Groupby Aggregation - Group based Aggregates
```

```
director_name
Adam McKay                     30.333333
Adam Shankman                  23.125000
Alejandro González Iñárritu    47.000000
Alex Proyas                    53.200000
Alexander Payne                24.800000
                                 ...
Wes Craven                     22.300000
Wolfgang Petersen              35.857143
Woody Allen                    17.722222
Zack Snyder                    71.857143
Zhang Yimou                    12.000000
Name: popularity, Length: 199, dtype: float64
```

```
# First and the last active year of "every" director?
data.groupby("director_name")["year"].aggregate([np.min, np.max])
```

director_name	amin	amax
Adam McKay	2004	2015
Adam Shankman	2001	2012
Alejandro González Iñárritu	2000	2015
Alex Proyas	1994	2016
Alexander Payne	1999	2013
...	...	...

```
data.groupby("director_name")["year"].min()
```

```
director_name
Adam McKay                      2004
Adam Shankman                   2001
Alejandro González Iñárritu     2000
Alex Proyas                     1994
Alexander Payne                 1999
                                ...
Wes Craven                      1984
Wolfgang Petersen               1981
Woody Allen                     1977
Zack Snyder                     2004
Zhang Yimou                     2002
Name: year, Length: 199, dtype: int64
```

```
data.groupby("director_name")["year"].max()
```

```
director_name
Adam McKay                      2015
Adam Shankman                   2012
Alejandro González Iñárritu     2015
Alex Proyas                     2016
Alexander Payne                 2013
                                ...
Wes Craven                      2011
Wolfgang Petersen               2006
Woody Allen                     2013
Zack Snyder                     2016
Zhang Yimou                     2014
Name: year, Length: 199, dtype: int64
```

```
# Filter all the rows (movies) which are directed by a "high budget director"
# high budget director is one whose average budget is >=100
# budget.max() >= 100


# in detailed notes -
# Q1 - Filter all the rows (movies) which are directed by a "high budget director"
# Q2 - Filter all the rows which are of high budget (not a groupby based)
data.groupby("director_name").filter(lambda x: x["budget"].mean() >= 100)
# Group based filtering - filtering the data based on group level characteristics
```

	id_x	budget	popularity	revenue	title	vote_
0	43597	237.0	150	2787.97	Avatar	
1	43598	300.0	139	961.00	Pirates of the Caribbean: At World's End	
3	43600	250.0	112	1084.94	The Dark Knight Rises	
5	43606	250.0	155	873.26	Batman v Superman: Dawn of Justice	
6	43607	270.0	57	391.08	Superman Returns	
...	...	...	...	...	...	
1341	47170	9.0	60	39.72	Memento	
1346	47220	5.0	4	5.48	Made	
1348	47228	5.0	8	3.05	Heavenly Creatures	
1362	47297	4.5	15	7.01	Bound	
1410	47719	13.5	5	0.19	Stonewall	

105 rows × 12 columns

```
data.apply(profit, axis=1) # learner's opw
```

```
0       2550.97
1        661.00
2        635.67
3        834.94
4        632.87
         ...
1460       0.32
1461       3.12
1462       0.00
1463       0.00
1464       1.82
Length: 1465, dtype: float64
```

```
# Group based transform/apply
```

```
# risky movies - 1 --> risky 0 --> not risky
# risky movie - whose budget is greater than the average revenue by a director of t
```

```
def is_risky(x):
  x["risky"] = x["budget"] - x["revenue"].mean() >= 0
  return x
```

```
data_risky = data.groupby("director_name").apply(is_risky)
```

```
# post-read -- transform is applicable for one column
```

```
data_risky.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1465 entries, 0 to 1464
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id_x           1465 non-null   int64
 1   budget         1465 non-null   float64
 2   popularity     1465 non-null   int64
 3   revenue        1465 non-null   float64
 4   title          1465 non-null   object
 5   vote_average   1465 non-null   float64
 6   vote_count     1465 non-null   int64
 7   year           1465 non-null   int64
 8   month          1465 non-null   object
 9   day            1465 non-null   object
 10  director_name  1465 non-null   object
 11  gender         1341 non-null   float64
 12  risky          1465 non-null   bool
dtypes: bool(1), float64(4), int64(4), object(4)
memory usage: 182.5+ KB
```

```
data.loc[data_risky["risky"]]
```

	id_x	budget	popularity	revenue	title	vot
7	43608	200.0	107	586.09	Quantum of Solace	
12	43614	380.0	135	1045.71	Pirates of the Caribbean: On Stranger Tides	
15	43618	200.0	37	310.67	Robin Hood	
20	43624	209.0	64	303.03	Battleship	
24	43630	210.0	3	459.36	X-Men: The Last Stand	
...	...	...	...	...	...	
1347	47224	5.0	7	3.26	The Sweet Hereafter	
1349	47229	5.0	3	4.84	90 Minutes in Heaven	
1351	47233	5.0	6	0.00	Light Sleeper	
1356	47263	15.0	10	0.00	Dying of the Light	
1383	47453	3.5	4	0.00	In the Name of the King III	

131 rows × 12 columns

```
# multi-indexing
# which director is the most productive director
# list of directors in the order of their productivity


# number of movies directed - quantity
# quality/ratings - critical review
# earnings of the movies - box office succes


data.groupby("director_name")["title"].count().sort_values(ascending=False)
```

```
director_name
Steven Spielberg    26
Clint Eastwood      19
Martin Scorsese     19
Woody Allen         18
Robert Rodriguez    16
                    ..
Paul Weitz           5
John Madden          5
Paul Verhoeven       5
John Whitesell       5
Kevin Reynolds       5
Name: title, Length: 199, dtype: int64
```

```
data_agg = data.groupby("director_name")[["year", "title"]].aggregate({"year": ["mi


data_agg.head()
```

	year		title
	min	max	count
director_name			
Adam McKay	2004	2015	6
Adam Shankman	2001	2012	8
Alejandro González Iñárritu	2000	2015	6
Alex Proyas	1994	2016	5
Alexander Payne	1999	2013	5

```
data_agg.columns
```

```
MultiIndex([( 'year',   'min'),
            ( 'year',   'max'),
            ('title', 'count')],
           )
```

```
data_agg["year"]
```

	min	max
director_name		
Adam McKay	2004	2015
Adam Shankman	2001	2012
Alejandro González Iñárritu	2000	2015
Alex Proyas	1994	2016
Alexander Payne	1999	2013
...	...	...
Wes Craven	1984	2011
Wolfgang Petersen	1981	2006
Woody Allen	1977	2013
Zack Snyder	2004	2016
Zhang Yimou	2002	2014

199 rows × 2 columns

```
data_agg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 199 entries, Adam McKay to Zhang Yimou
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   (year, min)     199 non-null    int64
```

```
 1   (year, max)     199 non-null    int64
 2   (title, count)  199 non-null    int64
dtypes: int64(3)
memory usage: 6.2+ KB
```

```
["_".join(col) for col in data_agg.columns]
```

```
['year_min', 'year_max', 'title_count']
```

```
data_agg.columns = ["_".join(col) for col in data_agg.columns]
```

```
data_agg.reset_index(inplace=True)
```

```
data_agg
```

	director_name	year_min	year_max	title_count
0	Adam McKay	2004	2015	6
1	Adam Shankman	2001	2012	8
2	Alejandro González Iñárritu	2000	2015	6
3	Alex Proyas	1994	2016	5
4	Alexander Payne	1999	2013	5
...	...	...	...	...
194	Wes Craven	1984	2011	10
195	Wolfgang Petersen	1981	2006	7
196	Woody Allen	1977	2013	18
197	Zack Snyder	2004	2016	7
198	Zhang Yimou	2002	2014	6

199 rows × 4 columns

```
data.groupby('director_name')[['year', 'title']].agg(
    year_max=('year','max'),
    year_min=('year','min'),
    title_count=('title','count')
)
```

director_name	year_max	year_min	title_count
Adam McKay	2015	2004	6
Adam Shankman	2012	2001	8
Alejandro González Iñárritu	2015	2000	6
Alex Proyas	2016	1994	5
Alexander Payne	2013	1999	5
...	...	...	...
Wes Craven	2011	1984	10

```
data_agg["yrs_active"] = data_agg["year_max"] - data_agg["year_min"]
data_agg
```

	director_name	year_min	year_max	title_count	yrs_active
0	Adam McKay	2004	2015	6	11
1	Adam Shankman	2001	2012	8	11
2	Alejandro González Iñárritu	2000	2015	6	15
3	Alex Proyas	1994	2016	5	22
4	Alexander Payne	1999	2013	5	14
...	...	...	...	...	...
194	Wes Craven	1984	2011	10	27
195	Wolfgang Petersen	1981	2006	7	25
196	Woody Allen	1977	2013	18	36
197	Zack Snyder	2004	2016	7	12
198	Zhang Yimou	2002	2014	6	12

199 rows × 5 columns

```
data_agg["movies_per_year"] = data_agg["title_count"]/data_agg["yrs_active"]
```

```
data_agg
```

	director_name	year_min	year_max	title_count	yrs_active	movies_
0	Adam McKay	2004	2015	6	11	
1	Adam Shankman	2001	2012	8	11	
2	Alejandro González Iñárritu	2000	2015	6	15	
3	Alex Proyas	1994	2016	5	22	
4	Alexander Payne	1999	2013	5	14	
...	...	...	...	...	...	
194	Wes Craven	1984	2011	10	27	

```
data_agg.sort_values("movies_per_year", ascending=False)
```

	director_name	year_min	year_max	title_count	yrs_active	movies_per_ye
190	Tyler Perry	2006	2013	9	7	1.2857
73	Jason Friedberg	2006	2010	5	4	1.2500
169	Shawn Levy	2002	2014	11	12	0.9166
158	Robert Rodriguez	1992	2014	16	22	0.7272
1	Adam Shankman	2001	2012	8	11	0.7272
...	...	...	...	...	...	
104	Lawrence Kasdan	1985	2012	5	27	0.1851
109	Luc Besson	1985	2014	5	29	0.1724
157	Robert Redford	1980	2010	5	30	0.1666
170	Sidney Lumet	1976	2006	5	30	0.1666
117	Michael Apted	1980	2010	5	30	0.1666

199 rows × 6 columns

✓ 0s completed at 21:44