1. Notified Assessments Team to add more practice questions for Numpy and Pandas
2. Please Please do not miss post-read for today (handling missing values in Pandas), its very very important https://colab.research.google.com/drive/1-uYNdWVDElSeDn4iVukBQuIDYm8_8tTX?usp=sharing

```
!gdown 1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
```

```
Downloading...
From: https://drive.google.com/uc?id=1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
To: /content/movies.csv
100% 112k/112k [00:00<00:00, 66.6MB/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, 62.1MB/s]
```

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

```
movies = pd.read_csv("movies.csv")
movies.head()
```

	Unnamed: 0	id	budget	popularity	revenue	title	vote_average	vc
0	0	43597	237000000	150	2787965087	Avatar	7.2	
1	1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	
2	2	43599	245000000	107	880674609	Spectre	6.3	
3	3	43600	250000000	112	1084939099	The Dark Knight	7.6	

Saved successfully!

```
directors = pd.read_csv('directors.csv')
directors.head()
```

	Unnamed: 0	director_name	id	gender
0	0	James Cameron	4762	Male
2	2	Sam Mendes	4764	Male

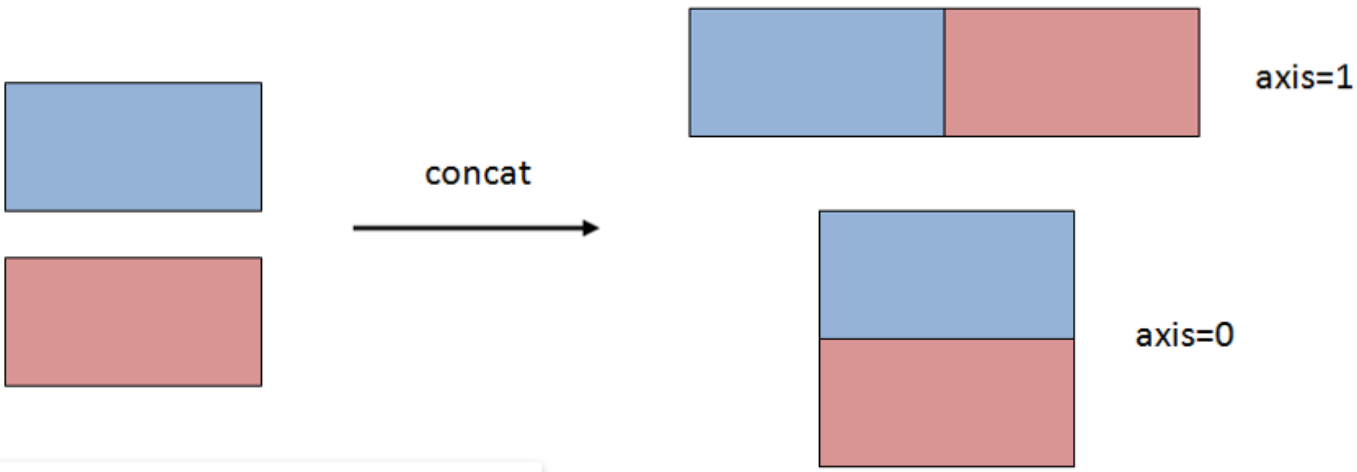
```
a = pd.DataFrame({'A':[10,30], 'B':[20,40]})
b = pd.DataFrame({'A':[10,30], 'C':[20,40]})
```

a

	A	B
0	10	20
1	30	40

b

	A	C
0	10	20
1	30	40



Saved successfully!

```
pd.concat([a, b], axis=0)
```

	A	B	C
0	10	20.0	NaN
1	30	40.0	NaN
0	10	NaN	20.0
1	30	NaN	40.0

```
pd.concat([a, b], axis=1)
```

	A	B	A	C
0	10	20	10	20
1	30	40	30	40

```
pd.concat([a, b], axis=0).loc[0]
```

	A	B	C
0	10	20.0	NaN
0	10	NaN	20.0

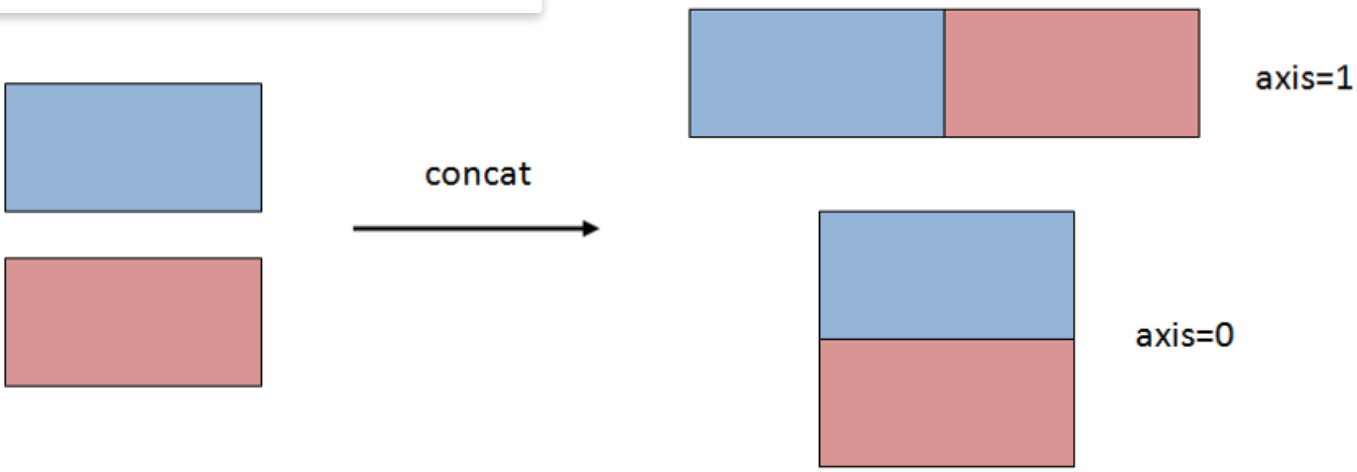
```
pd.concat([a, b], axis=0, ignore_index=True)
```

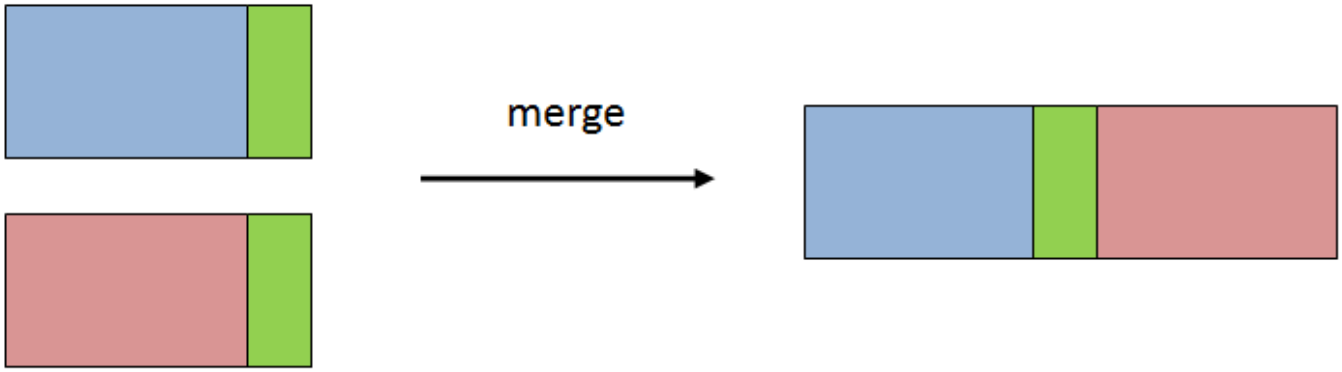
	A	B	C
0	10	20.0	NaN
1	30	40.0	NaN
2	10	NaN	20.0
3	30	NaN	40.0

```
pd.concat([a, b], keys=["x", "y"])
```

		A	B	C
x	**0**	10	20.0	NaN
	1	30	40.0	NaN
y	**0**	10	NaN	20.0
	1	30	NaN	40.0

Saved successfully!






```
# inner concatenation
pd.concat([a, b], axis=0, join="inner")
```

	A
0	10
1	30
0	10
1	30

```
pd.concat([a, b], axis=0, join="outer")
```

	A	B	C
0	10	20.0	NaN
1	30	40.0	NaN
0	10	NaN	20.0
1	30	NaN	40.0

Saved successfully!

```
# datframe-1 --> users
# dataframe-2 ---> msgs

users = pd.DataFrame({'userid':[1, 2, 3], 'name':['A', 'B', 'C']})
users
```

	userid	name
0	1	A
1	2	B
2	3	C

```
msgs = pd.DataFrame({'userid':[1, 1, 2], 'msg':['hello', 'bye', 'hi']})
msgs
```

	userid	msg
0	1	hello
1	1	bye
2	2	hi

```
pd.concat([users, msgs], axis=1) # doesnt make any sense
```

	userid	name	userid	msg
0	1	A	1	hello
1	2	B	1	bye
2	3	C	2	hi

```
msgs.merge(users, on="userid")
```

	userid	msg	name
0	1	hello	A
1	1	bye	A
2	2	hi	B

```
users.rename(columns={"userid":"id"}, inplace=True)
```

```
users
```

	id	name
1	2	B
2	3	C

Saved successfully!

```
msgs
```

	userid	msg
0	1	hello
1	1	bye
2	2	hi

```
users.merge(msgs, left_on="id", right_on="userid")
```

	id	name	userid	msg
0	1	A	1	hello
1	1	A	1	bye
2	2	B	2	hi

```
# inner, outer, left, right join
```

```
users.merge(msgs, left_on="id", right_on="userid", how="inner")
```

	id	name	userid	msg
0	1	A	1	hello
1	1	A	1	bye
2	2	B	2	hi

```
users.merge(msgs, left_on="id", right_on="userid", how="outer")
```

	id	name	userid	msg
0	1	A	1.0	hello
1	1	A	1.0	bye
2	2	B	2.0	hi
3	3	C	NaN	NaN

```
# left, right
```

Saved successfully!

```
ight_on="userid", how="left")
```

	id	name	userid	msg
0	1	A	1.0	hello
1	1	A	1.0	bye
2	2	B	2.0	hi
3	3	C	NaN	NaN

```
users.merge(msgs, left_on="id", right_on="userid", how="right")
```

	id	name	userid	msg
0	1	A	1	hello
1	1	A	1	bye

```
movies = pd.read_csv("movies.csv", index_col=0)
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

```
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)
```

Saved successfully!

```
(2349, 3)
```

```
# movies directors -- left outer join
```

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

```
199
```

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

```
2349
```

```
movies["director_id"].isin(directors["id"])
```

```
0       True
1       True
2       True
3       True
5       True
        ...
4736    True
4743    True
4748    True
4749    True
4768    True
Name: director_id, Length: 1465, dtype: bool
```

```
np.all(movies["director_id"].isin(directors["id"]))
```

```
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
			15	890871626	Spider-Man 3	5.9	3576

Saved successfully!

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

```
data # merged dataset
```

	id_x	budget	popularity	revenue	title	vote_average	vote_cour
0	43597	237000000	150	2787965087	Avatar	7.2	118(
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	45(
2	43599	245000000	107	880674609	Spectre	6.3	44(
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	91(

```
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()
```

Saved successfully!

	et	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
top	Avatar	Dec	Friday	Steven Spielberg	Male
freq	1	193	654	26	1309

```
data["revenue"] = data["revenue"]/1000000
```

```
data["budget"] = data["budget"]/1000000
```

```
# Quering dataframe to fetch the data
```

```
data.loc[data["vote_average"] >= 8].head()
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count
45	43662	185.0	187	1004.558444	The Dark Knight	8.2	12002
58	43692	165.0	724	675.120017	Interstellar	8.1	10867
59	43693	160.0	167	825.532764	Inception	8.1	13752
156	43859	93.0	138	871.368364	The Lord of the Rings: The Fellowship of the Ring	8.0	8705

Saved successfully!

```
data.loc[data["vote_average"] >= 8].head()
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count
45	43662	185.0	187	1004.558444	The Dark Knight	8.2	12002

```
data.loc[data["vote_average"] >= 8, ["title", "vote_average"]].head()
```

	title	vote_average
45	The Dark Knight	8.2
58	Interstellar	8.1
59	Inception	8.1
156	The Lord of the Rings: The Fellowship of the Ring	8.0
199	The Lord of the Rings: The Return of the King	8.1

```
data.loc[(data['vote_average'] >=7 ) & (data['year'] >= 2015)].head()
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count
30	43641	190.0	102	1506.249360	Furious 7	7.3	4176
78	43724	150.0	434	378.858340	Mad Max: Fury Road	7.2	9427
106	43773	135.0	100	532.950503	The Revenant	7.3	6396
162	43867	108.0	167	630.161890	The Martian	7.6	7268

```
# Strings methods - startswith, contains
```

```
data.loc[data["title"].str.contains("Batman")]
```

Saved successfully!

				y	revenue	title	vote_average	vote_count	y
5	43606	250.0		155	873.260194	Batman v Superman: Dawn of Justice	5.7	7004	2
74	43716	150.0		115	374.218673	Batman Begins	7.5	7359	2
128	43807	125.0		50	238.207122	Batman & Robin	4.2	1418	1
184	43896	100.0		48	336.529144	Batman Forever	5.2	1498	1

```
data.loc[data["title"].str.startswith("The")]
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count
3	43600	250.00	112	1084.939099	The Dark Knight Rises	7.6	9106
9	43610	255.00	49	89.289910	The Lone Ranger	5.9	2311
11	43612	225.00	53	419.651413	The Chronicles of Narnia: Prince Caspian	6.3	1630
14	43616	250.00	120	956.019788	The Hobbit: The Battle of the Five Armies	7.1	4760
16	43619	250.00	94	958.400000	The Hobbit: The Desolation of Smaug	7.6	4524
...	...	...	...	...	...	...	...

```
data.sort_values("popularity", ascending=False).head(5)
```

	id_x	budget	popularity	revenue	title	vote_average	vote_count
58	43692	165.0	724	675.120017	Interstellar	8.1	10867
78	43724	150.0	434	378.858340	Mad Max: Fury Road	7.2	9427
			71	655.011224	Pirates of the Caribbean: The Curse of the Bla...	7.5	6985
120	43797	125.0	206	752.100229	The Hunger Games:	6.6	5584

Saved successfully!

```
data.loc[data["director_name"] == "Christopher Nolan", "title"]
```

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

```
1341                    Memento
Name: title, dtype: object
```

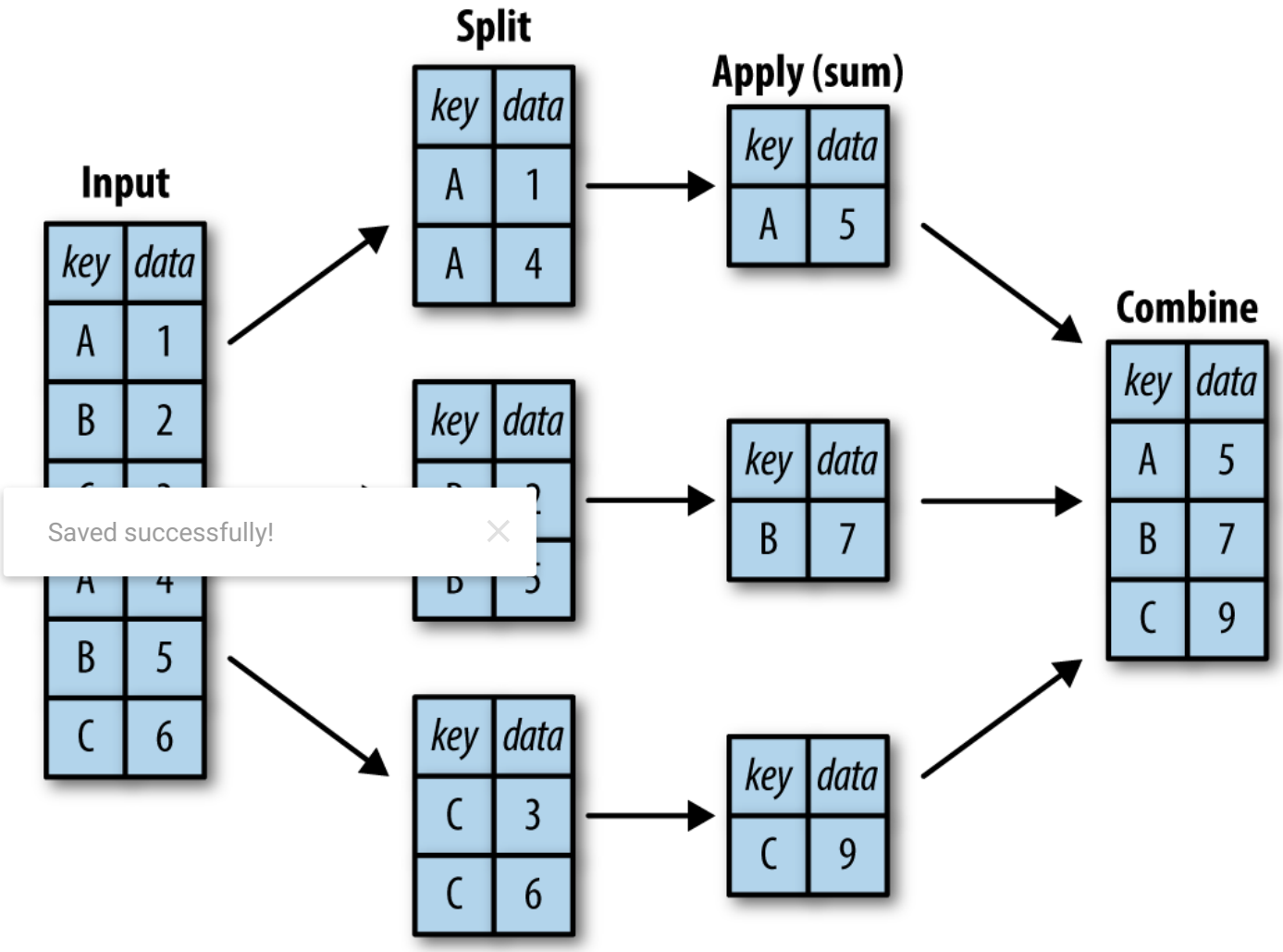
```
data.loc[data["director_name"] == "Christopher Nolan", "title"].count()
```

```
8
```

```
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
```

```
# finding the highest budget movie of every director?
```




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

```
director_name
Adam McKay                      100.0
Adam Shankman                    80.0
Alejandro González Iñárritu     135.0
Alex Proyas                     140.0
Alexander Payne                  30.0
                                ...
Wes Craven                       40.0
Wolfgang Petersen               175.0
Woody Allen                      30.0
Zack Snyder                     250.0
Zhang Yimou                      94.0
Name: budget, Length: 199, dtype: float64
```

```
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
```

```
# which director is the most productive director?
```

```
# number of movies
# quality - vote_average
```

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



```
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
```

```
#number of movies directed per year
```

```
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
```

```
df_agg = data.groupby("director_name")[["year", "title"]].aggregate(
    {"year": ["min", "max"], "title":"count"})
df_agg
```

	year		title
	min	max	count
director_name			

```
df_agg.columns
```

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

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

...	...	...	...

```
df_agg.columns
```

```
Index(['year_min', 'year_max', 'title_count'], dtype='object')
```

Woody Allen	1977	2013	18

```
df_agg
```

director_name	year_min	year_max	title_count
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
...	...	...	...
Wes Craven	1984	2011	10
Wolfgang Petersen	1981	2006	7
Woody Allen	1977	2013	18
Zack Snyder	2004	2016	7
Zhang Yimou	2002	2014	6

199 rows × 3 columns

Saved successfully!

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

```
df_agg["active_yrs"] = df_agg["year_max"] - df_agg["year_min"]
```

```
df_agg
```

	director_name	year_min	year_max	title_count	active_yrs
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

```
df_agg["movie_per_yr"] = df_agg["title_count"] / df_agg["active_yrs"]
```

```
df_agg.sort_values("movie_per_yr", ascending=False).head(5)
```

	director_name	year_min	year_max	title_count	active_yrs	movie_per_yr
190	Tyler Perry	2006	2013	9	7	1.285714
73	Jason Friedberg	2006	2010	5	4	1.250000
169	Shawn Levy	2002	2014	11	12	0.916667
158	Robert Rodriguez	1992	2014	16	22	0.727273
1	Adam Shankman	2001	2012	8	11	0.727273

Saved successfully!

✓ 0s completed at 23:49

Saved successfully!