

Agenda

- (Introduction)
- (Teaching)
 - Introduction to IMDB use case
 - Merging **movies** & **directors** datasets
 - IMDB data exploration (Post-read)
 - `apply()` \rightarrow Proccally
 - `groupby()`
 - Group based Aggregation
 - Group based Filtering
 - Group based Apply

v.v imp { 2 of my interviews }

Robert Jindal \rightarrow Sr Data Scientist at Tech (Retail)

Civil Engineer \rightarrow IISc Bangalore (Masters in D.S)

Mastercard A.I Garage (Financial ML)

Shakti Street Global Advisors

A-Manager at Indian Railways

University of Cambridge \rightarrow (6 month visit) (EP & Sustainable) Data Science

High Level

High Level

Primary key \rightarrow movies (Table 1)

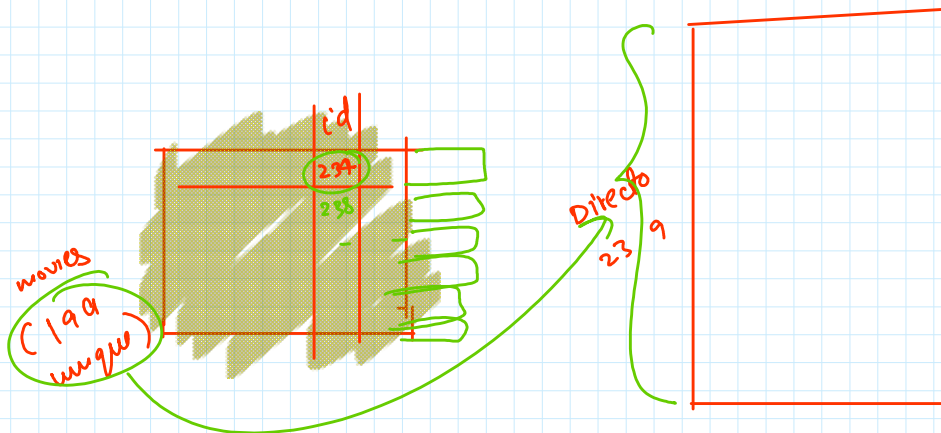
Movies ? Foreign key

Unnamed: 0	id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	Now	Good	
0	0	43597	237000000	150	2787965087	Avatar	7.2	11800	4762	2009	Dec	Thursday	—	—
1	1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500	4763	2007	May	Saturday	—	—
2	2	43599	245000000	107	880674609	Spectre	6.3	4466	4764	2015	Oct	Monday	—	—
3	3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106	4765	2012	Jul	Monday	—	—
4	5	43602	258000000	115	890871626	Spider-Man 3	5.9	3576	4767	2007	May	Tuesday	—	—

Code { Proccess }

Directors ? director ID (Primary key in directors table)

	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



After Joining we get id-x, id-y

Notice the two strange id columns - `id_x` and `id_y`.

What do you think these newly created columns are?

Since the columns with name `id` are present in both the dataframes,

- `id_x` represents id values from movie df
- `id_y` represents id values from directors df

after joining we get $10 - x$ & $10 - y$

Notice the two strange id columns - `id_x` and `id_y`.

What do you think these newly created columns are?

Since the columns with name `id` are present in both the dataframes,

- `id_x` represents id values from movie df
- `id_y` represents id values from directors df

Do you think any column is redundant here and can be dropped?

- `id_y` is redundant as it is the same as `director_id`
- But we don't require the `director_id` any further.

So we can simply drop these features -

Apply function in Pandas

	id_x	budget	popularity	revenue	title	vote_average	vote_count	year	month	day	director_name	gender	encoded
0	43597	237000000	150	2787965087	Avatar	7.2	11800	2009	Dec	Thursday	James Cameron	Male	0
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500	2007	May	Saturday	Gore Verbinski	Male	0
2	43599	245000000	107	880674609	Spectre	6.3	4466	2015	Oct	Monday	Sam Mendes	Male	0
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106	2012	Jul	Monday	Christopher Nolan	Male	0
4	43602	258000000	115	890871626	Spider-Man 3	5.9	3576	2007	May	Tuesday	Sam Raimi	Male	0
...
1460	48363	0	3	321952	The Last Waltz	7.9	64	1978	May	Monday	Martin Scorsese	Male	1
1461	48370	27000	19	3151130	Clerks	7.4	755	1994	Sep	Tuesday	Kevin Smith	Male	1
1462	48375	0	7	0	Rampage	6.0	131	2009	Aug	Friday	Uwe Boll	Male	0
1463	48376	0	3	0	Slacker	6.4	77	1990	Jul	Friday	Richard Linklater	Male	0
1464	48395	220000	14	2040920	El Mariachi	6.6	238	1992	Sep	Friday	Robert Rodriguez	NaN	1

(Forget other Rows)

```
def encode(data):
    if data == "Male":
        return 0
    else:
        return 1
```

one entry

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

Group By
`df.groupby("key").split`

Input

key	data
A	1
B	2
C	3
A	4
B	5
C	6

A	1
A	4

B	2
B	5

C	3
C	6

Apply(Sum)

A	5
---	---

B	7
---	---

C	9
---	---

Combine

A	5
B	7
C	9

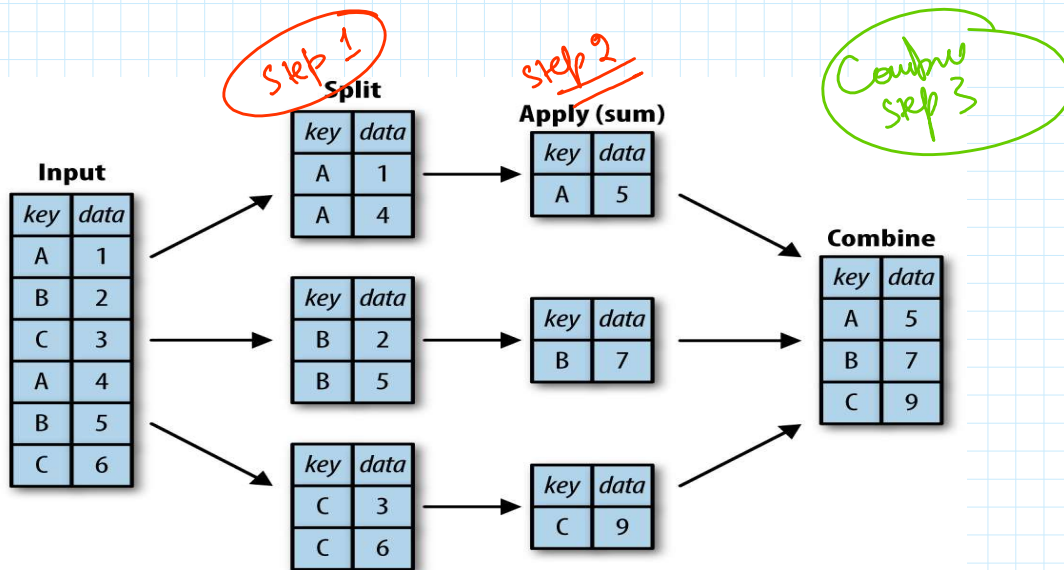
B	5
C	6

C	3
C	6

C	9
---	---

df.groupby("key")

• apply("sum")



1. **Split**: Breaking up and grouping a DataFrame depending on the value of the specified key.

2. **Apply**: Computing some function, usually an aggregate, transformation, or filtering, within the individual groups.

3. **Combine**: Merging the results of these operations into an output array.

5 mins

⇒ 22:09 (10:09 PM)

Quiz Ended!

Given a dataframe of employee rating info (`emp_id`, `monthly_rating`). How will you calculate the average rating of each employee?

56 users have participated

- A `data.groupby('emp_id')` 4%
- B `data.groupby('monthly_rating')` 2%
- ☒ C `data.groupby('emp_id').mean()` 52%
- D `data.groupby('monthly_rating').mean()` 43%

children

directors

df.groupby(["direct", "child"]).agg([fn1, fn2...])
• agg(["min", "max"])

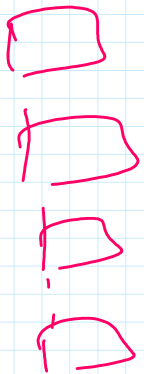
2

1/

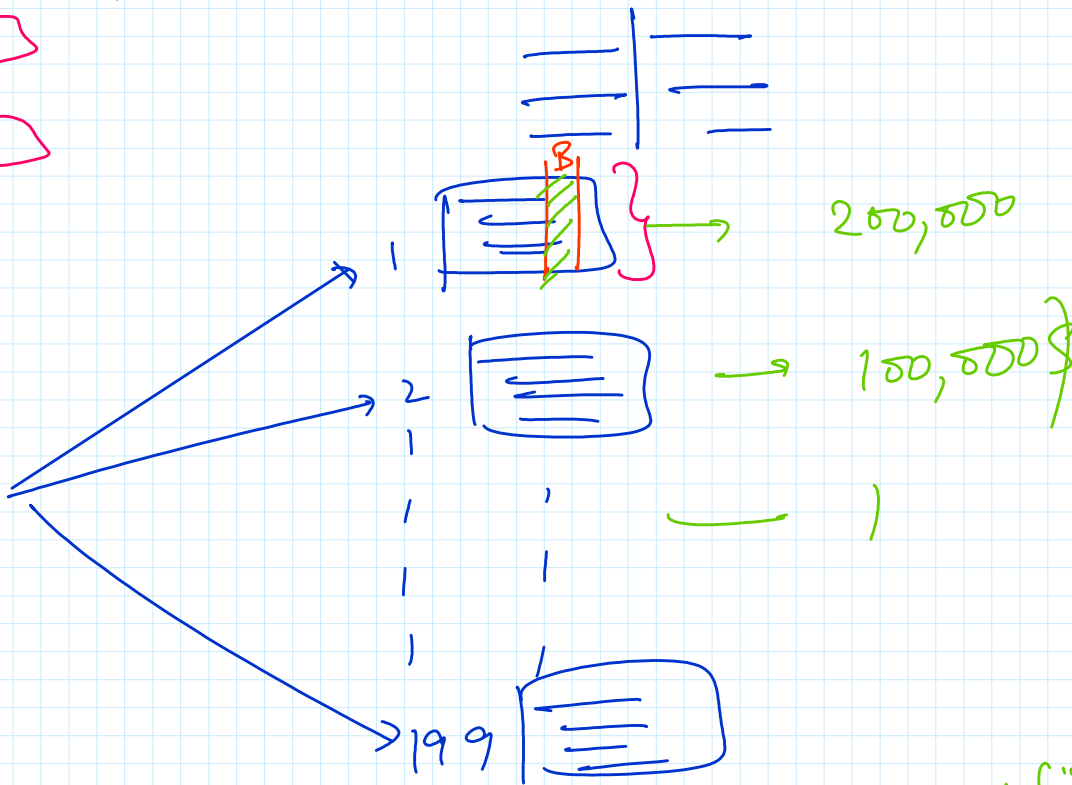
agg(["min", "max"])

Group Based Filtering

high Budget Director → (director whose at least one movie has budget > 100 M)



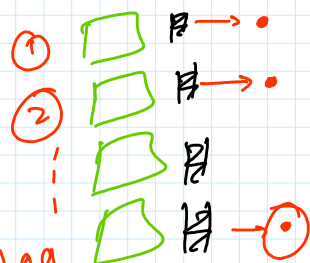
```
data.groupby("director")["budget"].max()
```



group["budget"]

Here "groups will be filtered"

```
data.groupby("director-name").filter(lambda x: x["budget"].max() >= 100 million)
```



J.C
J.C

Budget
100M
200M
0
keep it

199 100 100 → 0

data [data.budget > 100M]

JC
JC
JC

200M
0
0
-100
keep the "group"

(Group Based Apply)

Let's say, we want to filter the risky movies whose budget was even higher than the average revenue of the director from his other movies.

$$x(\text{risky}) = x(\text{budget}) - x(\text{revenue}) \cdot \text{mean}() > 0$$

df.apply → S1 → write fn for a single row
→ S2 → use df.apply(fn)

group Based apply → S1 → write a fn for one group
→ S2 → df.groupby(" ") . apply (fn)

Given the following data frame:

```
import pandas as pd
data = {'Group': ['G1', 'G2', 'G2', 'G3', 'G3', 'G3'],
        'Points': [2, 4, 6, 8, 10, 12]}
df = pd.DataFrame(data)
```

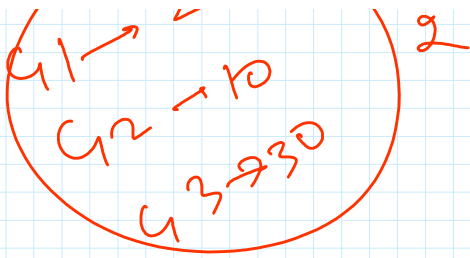
What will be output of the code given below?

```
result = df.groupby('Group')['Points'].apply(lambda x: x.sum())
print(result)
```

G1 → 2
→ 10

Points
G1
G2
G2

Group	Points
G1	2
G2	4
G2	6
G3	8
G3	10
G3	12



For each year, how you will get the average, lowest, and highest value of movie ratings?

3 options

Active Duration (Most preferred: 30 seconds)

Appears for 60 Secs

A `data.groupby('year').agg({'vote_average': ['mean', 'min', 'max']})`

B `data.groupby('year').agg({'vote_average': ['mean', 'min', 'max']})`

C `data.groupby('year').agg({'vote_average': ['mean', 'min', 'max']})`

`agg(["min", "max", "count"])`
`{ "mean": "mean", "min": "min", "max": "max" }`

`df.groupby("year").agg({'vote_avg': ['mean', 'min', 'max']})`

20 mins *

	vote_avg	mean	min	max
Year (2012)		-	-	-
2014		-	-	-

2012
2014

`df.agg(lambda x: x.max() - x.min())`

① `import pandas`

`pandas.merge(df1, df2, ...)`

II

df1.merge(df2, on="id", how="left")

1st 1st

→

→

skill based

.ipynb
scrubbed not