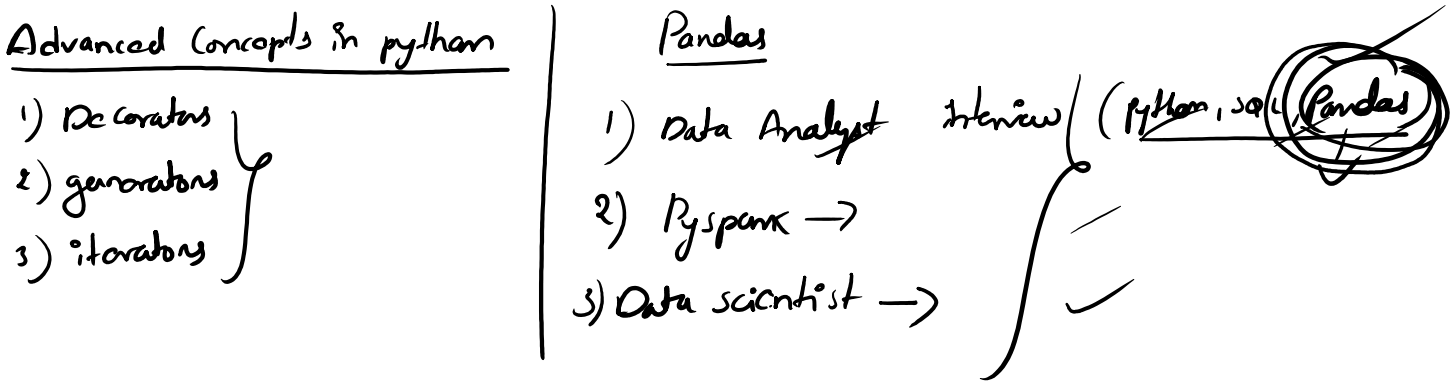
Pandas

18 February 2025 07:36



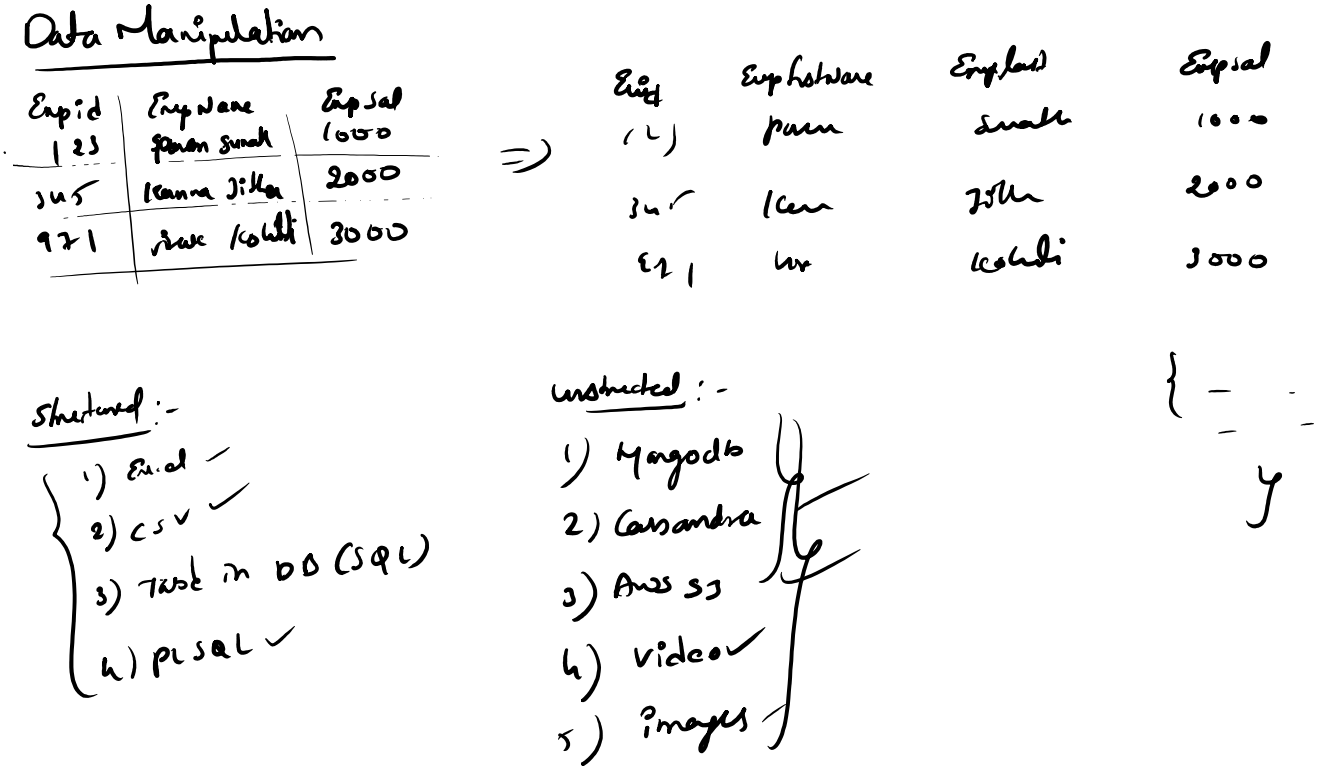
What is Pandas?

Pandas is an **open-source Python library** used for data manipulation and analysis. It provides **data structures** like **Series** (1D) and **DataFrame** (2D) to handle structured data efficiently.

Where is Pandas Used?

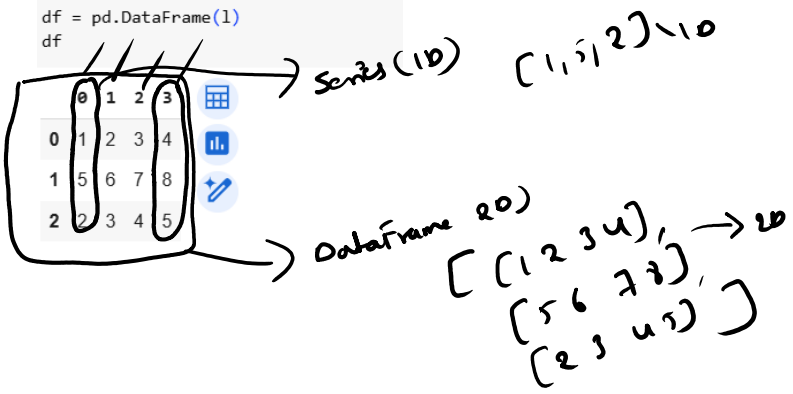
Pandas is widely used in **data science, analytics, and machine learning** for:

- ☑ **Data Cleaning & Preprocessing** – Handling missing values, duplicates, and formatting data.
- ☑ **Data Analysis** – Performing statistical calculations and data summarization.
- ☑ **Data Visualization** – Plotting data using built-in functions and integration with Matplotlib & Seaborn.
- ☑ **Data Transformation** – Merging, reshaping, and filtering large datasets.
- ☑ **Time Series Analysis** – Handling time-indexed data like stock prices or IoT data.
- ☑ **Big Data Handling** – Works well with CSV, Excel, SQL databases, and JSON files.



```
l=[[1,2,3,4],[5,6,7,8],[2,3,4,5]]

df = pd.DataFrame(l)
df
```



	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	2	3	4	5

Few Important Methods

19 February 2025 07:28

.loc & .iloc

.apply()
----> Modify the Runtime column
----> Modify the Gross column

.rename()

.De-nesting Genre for top 10 movies based on Meta_score

.duplicated()

.isnull()

.notnull()

.fillna()

.dropna()

.drop_duplicates()

.drop()

.set_index()

.reset_index()

.sort_values()

.value_counts()

.

.loc & .iloc

-) Both are used to filter / select / update rows / columns in a dataframe.

diff .loc & .iloc

-) .loc uses labels

-) .iloc uses indexes / positions

Syntax

df.loc[,]
rows columns

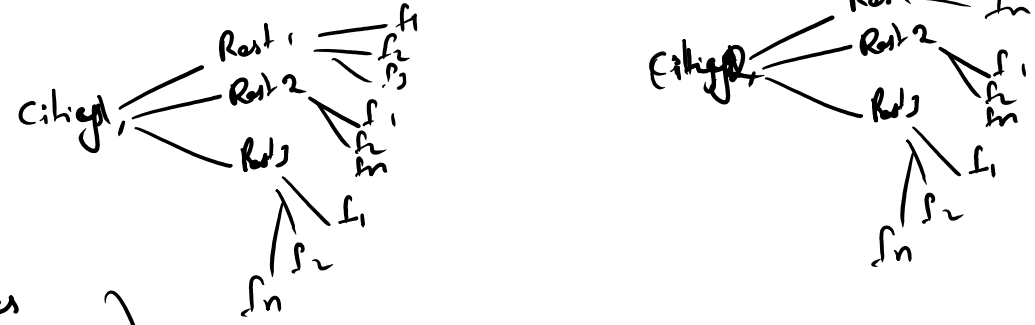
df.iloc[,]
rows columns

groupby

24 February 2025 08:02

→ helps in dividing dataset into groups, on top of it you can perform transformations / operations on each group.

Ex:-



Categories

- Biryani
- beverages
- Snacks
- breakfast

Swiggy task

Sample dataset:-

5 categories

S.No	food item	RestName	Rating	CityName	Category	Amount
1	f1	R1	9.5	Kolkata	Biryani	250
2	f2	R1	8.5	Chennai	breakfast	90
.						
.						
.						
.						
.						

1000 rows

100 x 7

1000 x 7
#rows #cols

200 x 7
#rows #cols

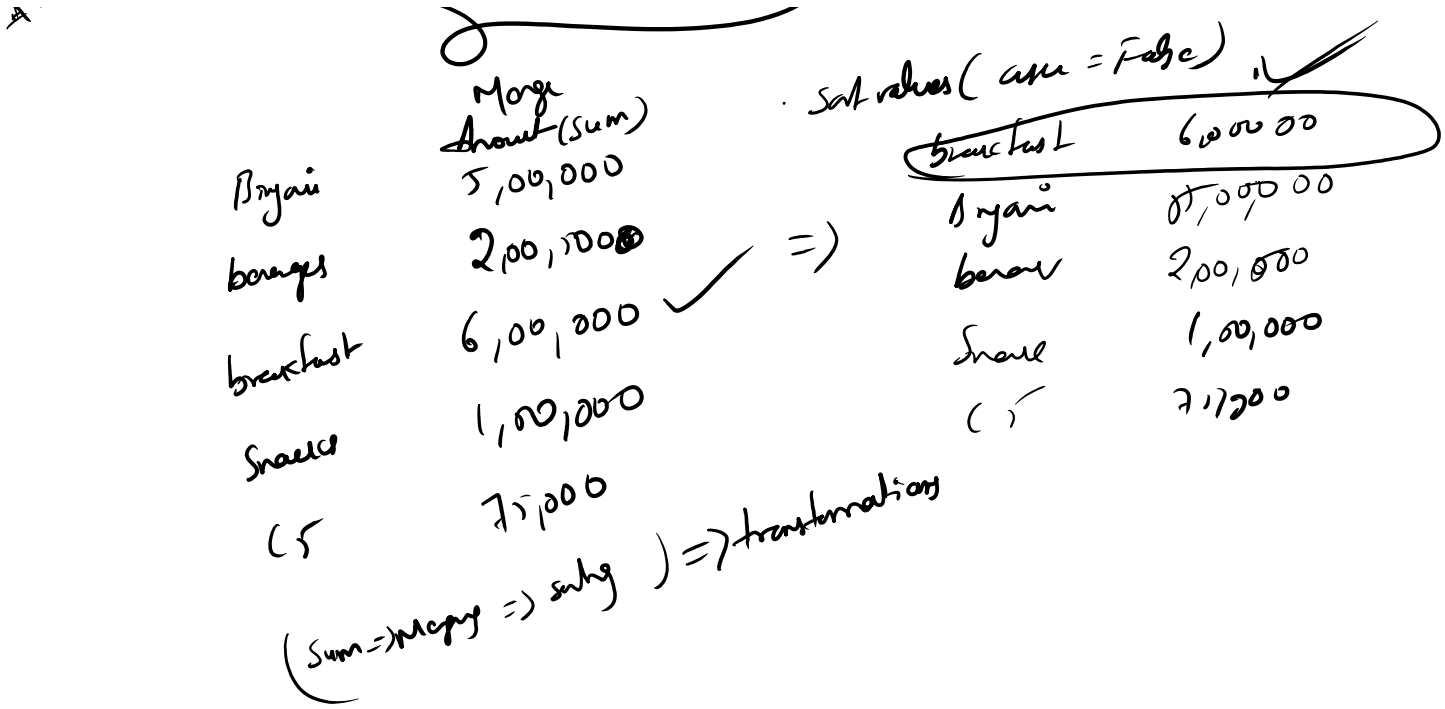
100 x 7

300 x 7

Sample data

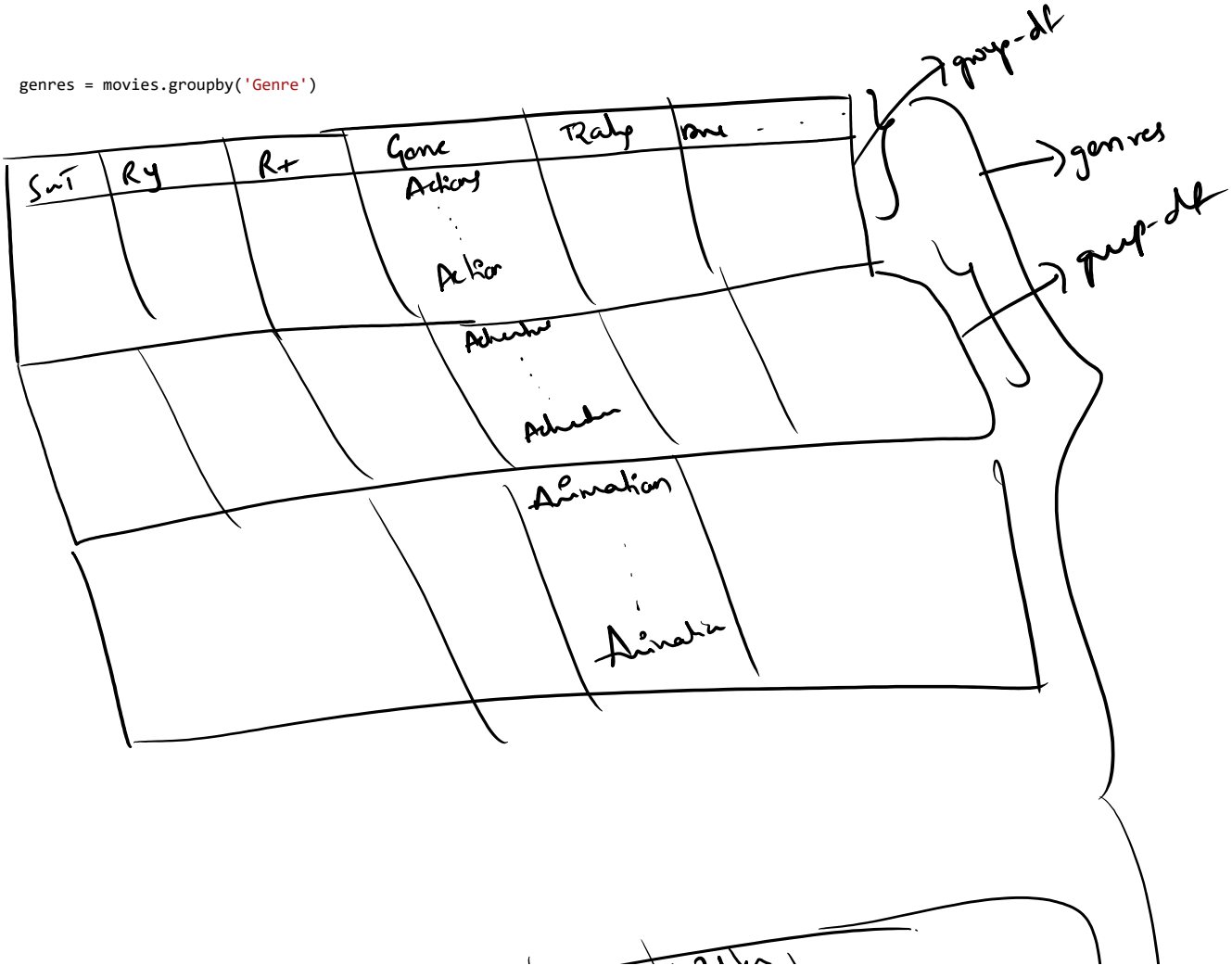
find cost of food item with highest sales!

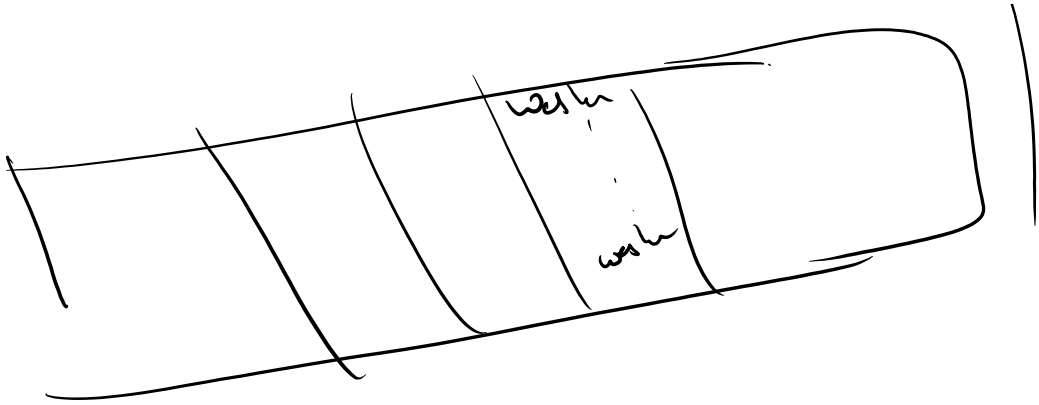
Merge ... sort values(case = False)

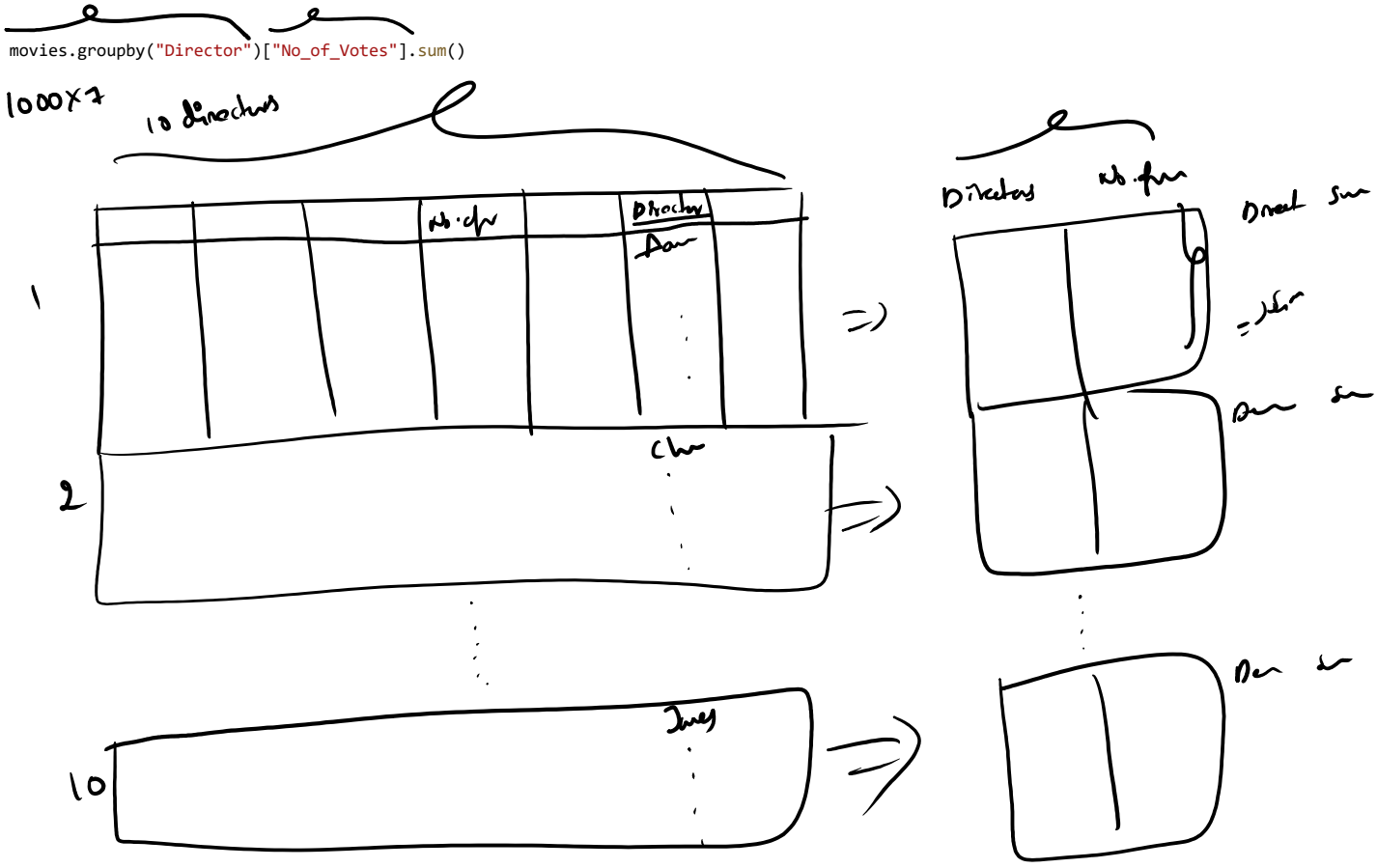


	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Director	Star1	No_of_Votes	Gross	Metascore
0	The Shawshank Redemption	1994	142	Drama	9.3	Frank Darabont	Tim Robbins	2343110	28341469.0	80.0
1	The Godfather	1972	175	Crime	9.2	Francis Ford Coppola	Marlon Brando	1620367	134966411.0	100.0
2	The Dark Knight	2008	152	Action	9.0	Christopher Nolan	Christian Bale	2303232	534858444.0	84.0
3	The Godfather: Part II	1974	202	Crime	9.0	Francis Ford Coppola	Al Pacino	1129952	57300000.0	90.0
4	12 Angry Men	1957	96	Crime	9.0	Sidney Lumet	Henry Fonda	689845	4360000.0	96.0

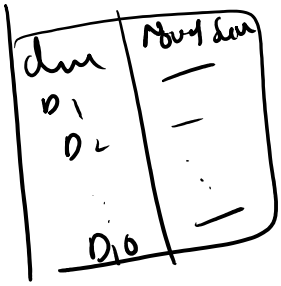
```
genres = movies.groupby('Genre')
```

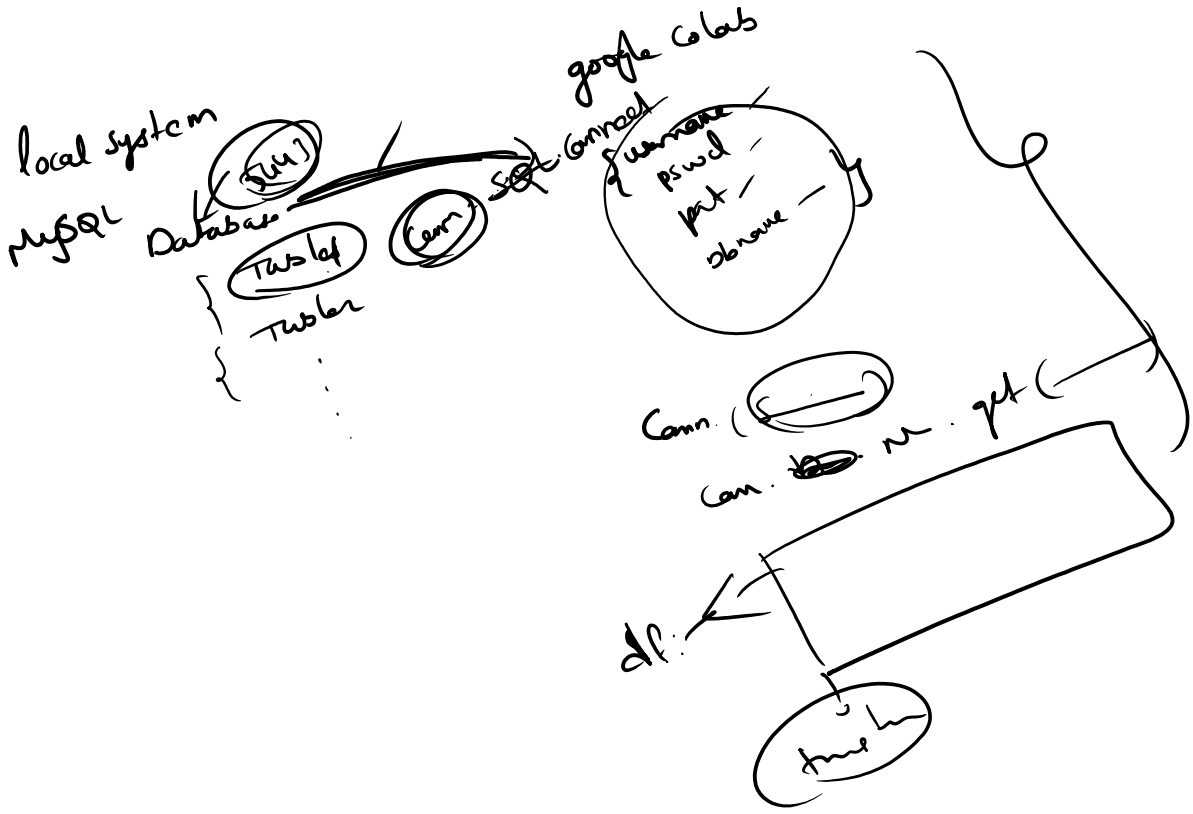





movies.groupby("Director")["No_of_Votes"].sum()
1000X7
10 Directors
1
2
10
=>

10X1





=====agg
- max
- min
- sum
- mean
- mode
- std
- var
- count
- size - group wise count

first - first row in every group

last - last row in every group

nth(num) - if num = 5, 5th row in every group

get_group("Horror") - returns Horror group related data

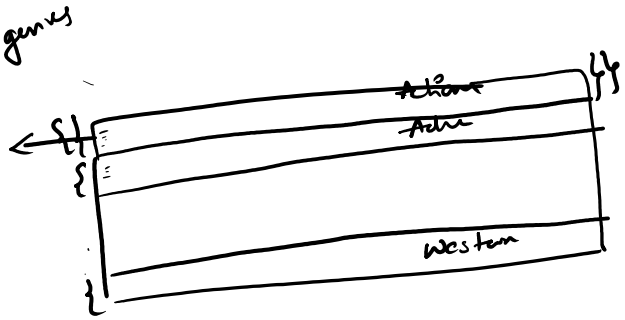
groups - returns dict with keys as group names and values as index positions

describe() - similarly like pandas df on groups it will summarize every column(count, mean, std, 25% etc)

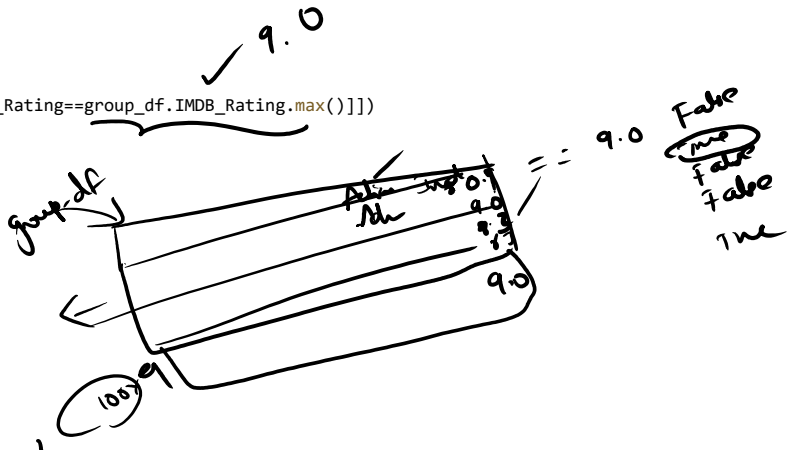
sample(2) - randomly picks 2 rows on each group

nunique() - gives column wise unique count

```
for group, group_df in genres:
```




```
# looping on groups
#find highest rated movie from each genre
df=pd.DataFrame(columns=movies.columns)
for group, group_df in genres:
  df = pd.concat([df,group_df[group_df.IMDB_Rating==group_df.IMDB_Rating.max()]])
```



Divide & conquer

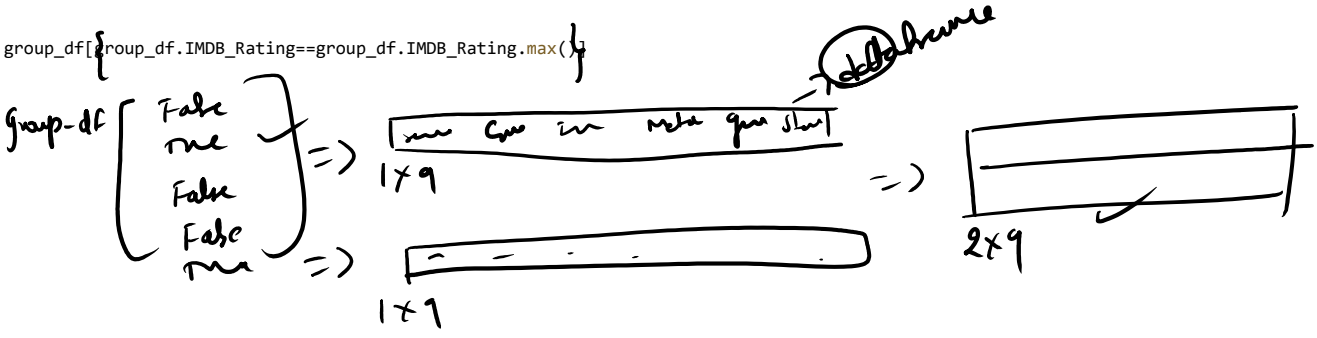
```
group_df.IMDB_Rating.max()
```

9.0

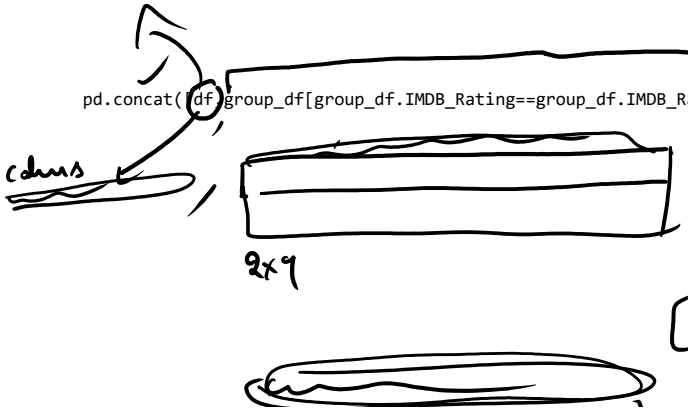
```
group_df.IMDB_Rating==group_df.IMDB_Rating.max()
```

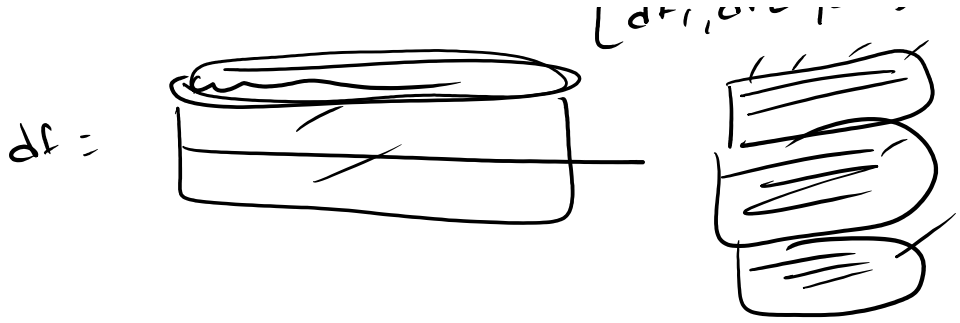
False
True
False
False

```
group_df[group_df.IMDB_Rating==group_df.IMDB_Rating.max()]
```



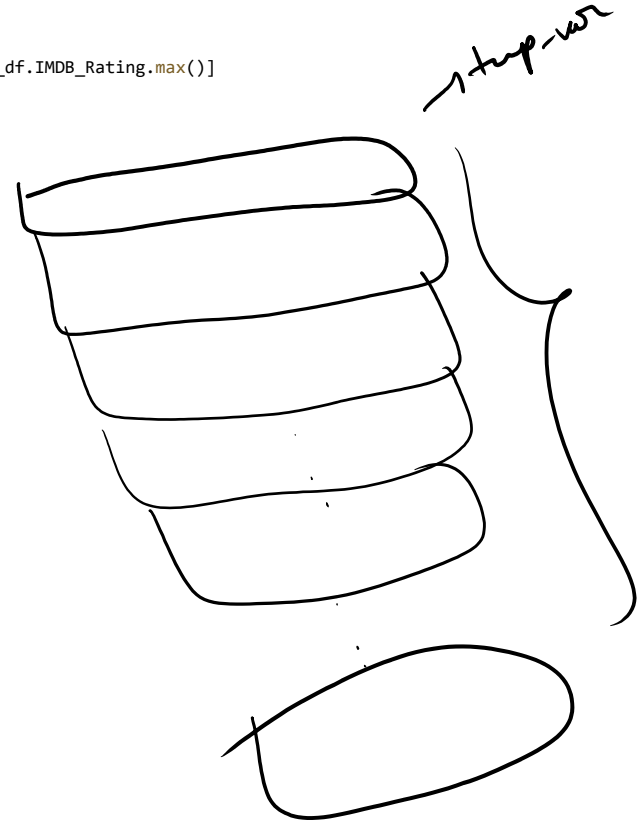

```
pd.concat([df,group_df[group_df.IMDB_Rating==group_df.IMDB_Rating.max()]])
```





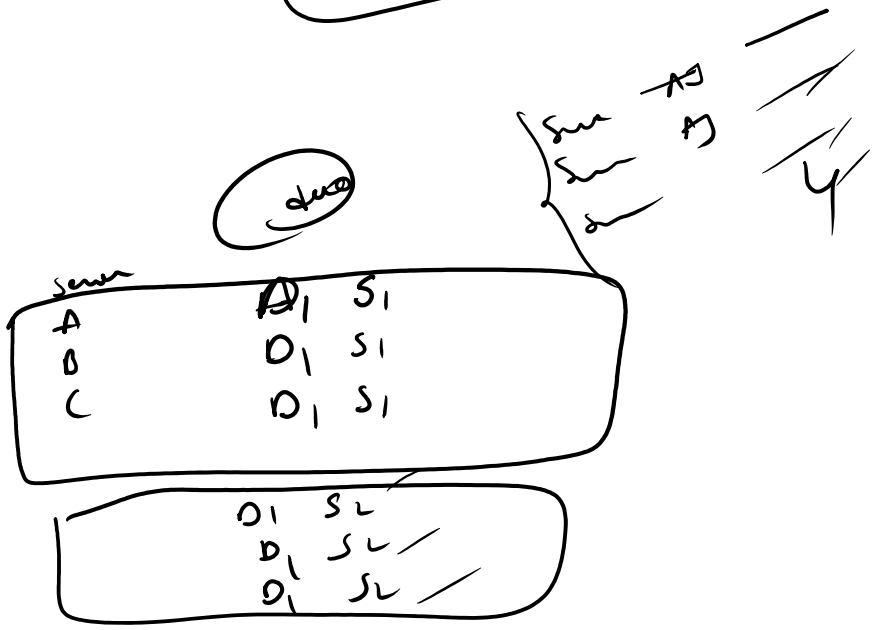
```
#using apply method
def genre_highest_rated(group_df):
  return group_df[group_df.IMDB_Rating==group_df.IMDB_Rating.max()]

genres.apply(genre_highest_rated)
```



```
# groupby on multiple cols

movies.groupby(["Director","Star1"])
```

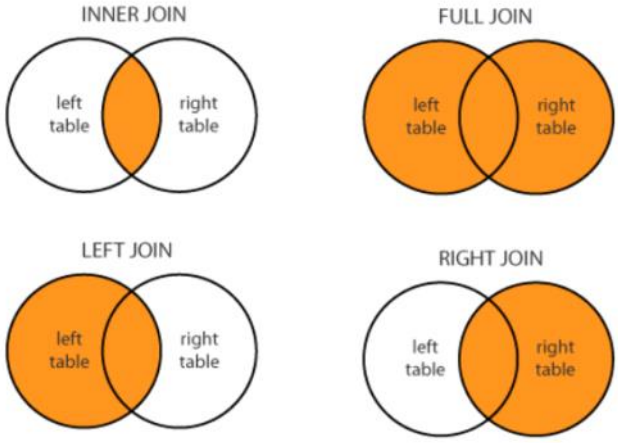


concat, merge, visualization

28 February 2025 07:24

Concat - It allows us to combine 2 or more data frames along horizontal and vertical

Merge - It is also another way to combine data frames



Difference Between concat() and merge() in Pandas

Feature	concat()	merge()
Purpose	Combines DataFrames along rows (axis=0) or columns (axis=1)	Joins DataFrames based on key columns (like SQL JOIN)
How it Works	Simply stacks DataFrames (row-wise or column-wise)	Matches data based on common columns or indexes
Default Behavior	Does not match values, just appends	Matches rows based on key columns
Best Use Case	When combining similar DataFrames (stacking datasets)	When you need relational joins (similar to SQL joins)

Data Visualization(Using Matplotlib, Seaborn):

What is Matplotlib?

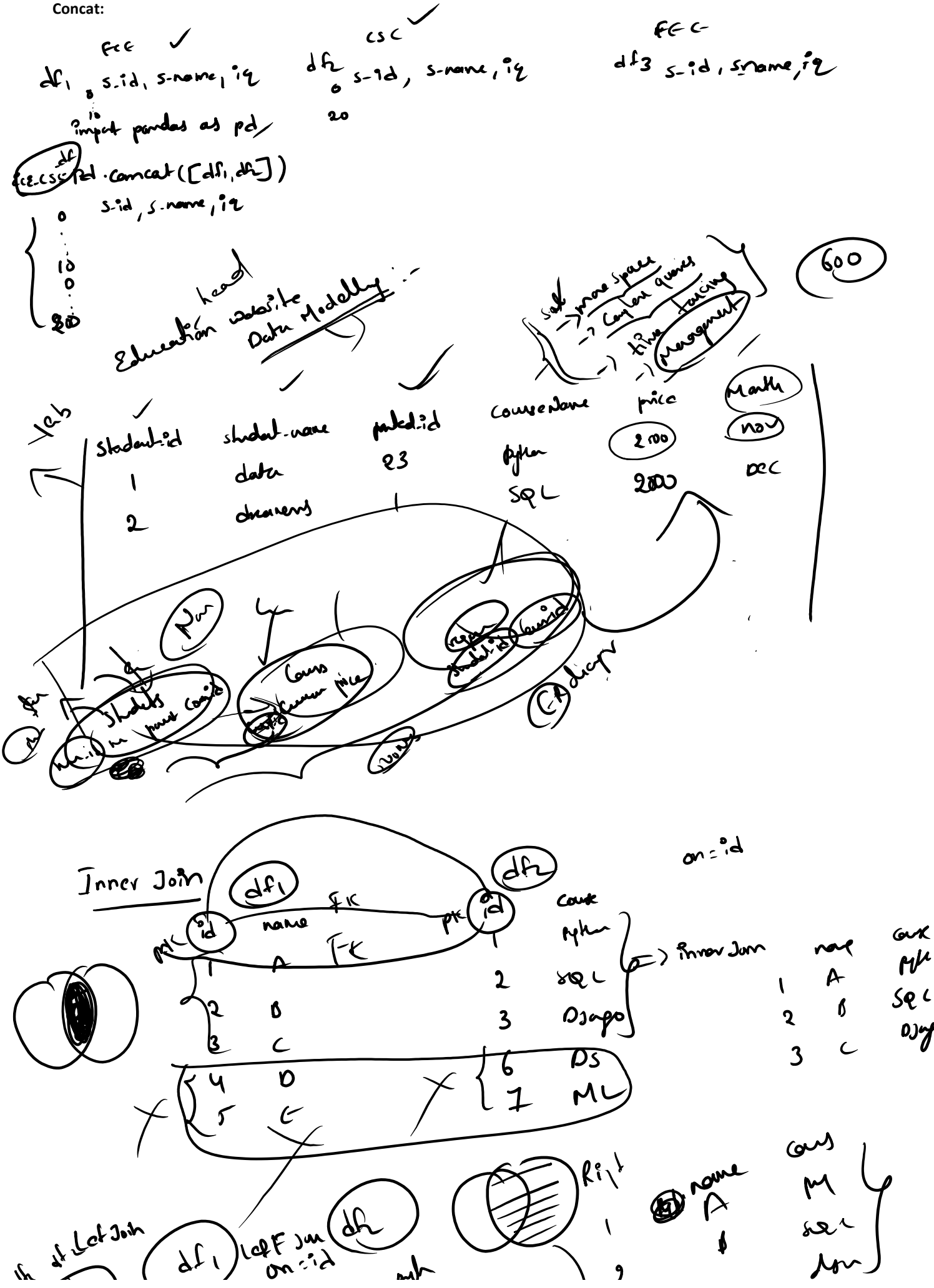
Matplotlib is a powerful Python library for creating static, animated, and interactive visualizations. It is often used for plotting line charts, bar charts, scatter plots, and histograms.

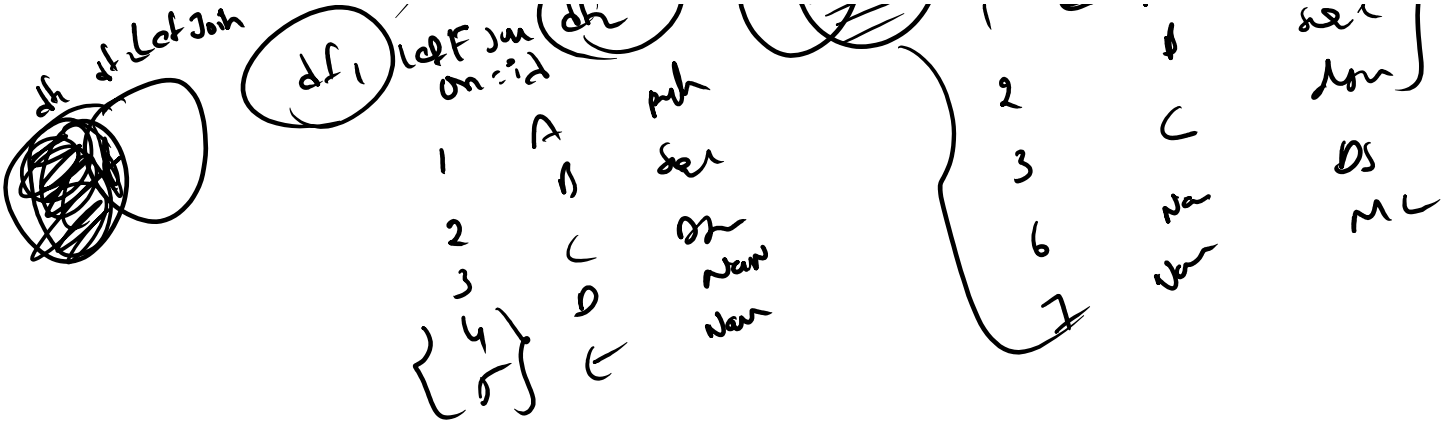
What is Seaborn?

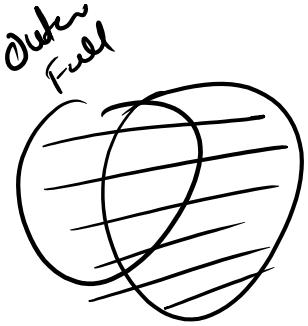
Seaborn is built on top of Matplotlib and provides a more visually appealing and statistical way to create plots. It simplifies complex visualizations like heatmaps, box plots, and violin plots.

Concat:

Concat:


FCC
CSC
FCC
df1 s_id, s_name, iq
df2 s_id, s_name, iq
df3 s_id, s_name, iq
import pandas as pd
fcc_csc_df
pd.concat([df1, df2])
s_id, s_name, iq
0
10
0
200
Education website
Data Modelling
600
Skill
more space
Complex queries
time tracing
management
Math
nov
dec
price
2000
2000
Lab
Student_id
student_name
parked_id
course_name
python
SQL
1
2
data
dreamers
23
Inner Join
df1
df2
id
name
FK
PK
PK
A
B
C
D
E
1
2
3
4
5
6
7
DS
ML
python
SQL
Django
on=id
Inner Join
Left Join
Right
df1
df2
left Join
on=id
name
Cours
A
B
M
SQL
Dja


df Left Join
df1
Left Join
on = id


Outer
Full

Matplotlib & Seaborn

02 March 2025 19:49

Matplotlib: A Low-Level Plotting Library

- Matplotlib is a **low-level** library that gives **full control** over every aspect of a plot.
- It works like a **canvas**, where you can modify everything, including:
  - Colors
  - Line styles
  - Axes properties
  - Legends
  - Grid styles
  - Annotations
- Since it provides raw access to plot elements, you can create **highly customized visualizations**.

Seaborn: A High-Level Statistical Library

- Seaborn is built **on top of Matplotlib** but is designed to work with **structured datasets (DataFrames)**.
- It **automates** many visualizations, making plots **easier to create and visually appealing by default**.
- However, this automation **limits deep customization** because Seaborn **abstracts away** some of Matplotlib's low-level details.

1) Bar plot

2) Scatter plot

3) Box plot

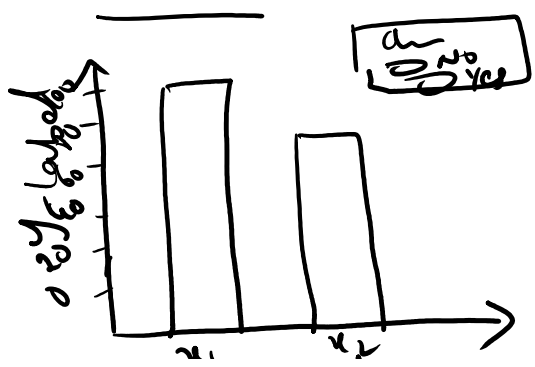
4) histogram

5) heat map

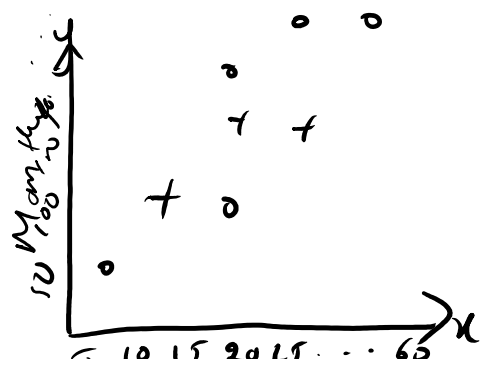
6) Violin plot

7) pie chart

1) Bar plot :-



2) Scatter plot



3) Box plot

Monthly-charges = [50, 75, 120, 25, 35, 100 ...]

= [25, 35, 50, 75, 100, 120 ...]

IQR (Inter quartile range) = 75% − 25%

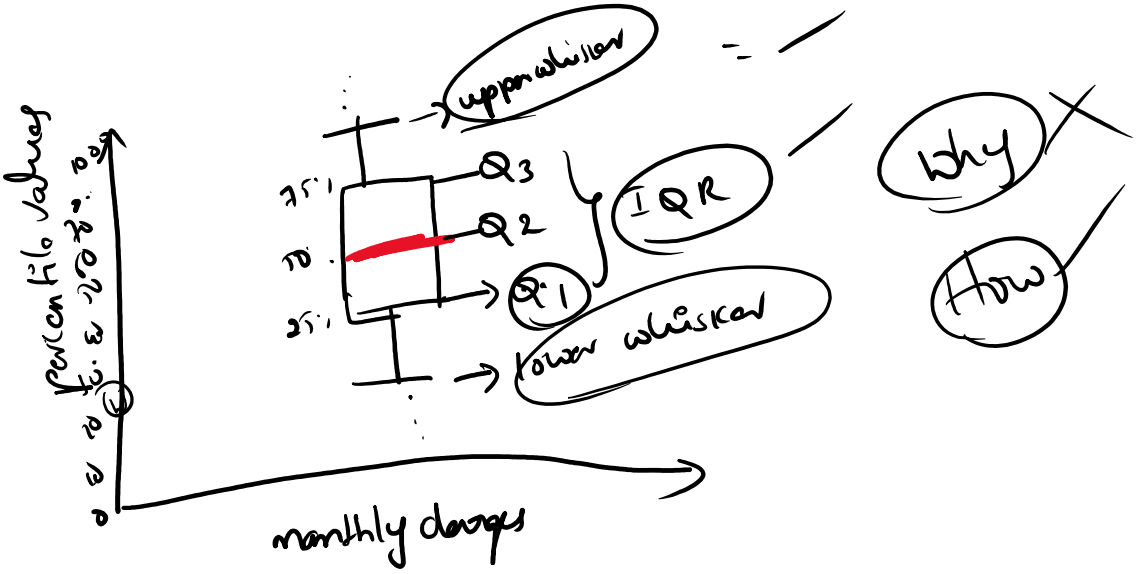
Q1 = 25%
Q2 = 50%
Q3 = 75%

upper-whisker = Q3 + 1.5 IQR

= 75% + 1.5 * (75% − 25%)

lower-whisker = Q1 − 1.5 (IQR)

= 25% − 1.5 (75% − 25%)



Why

How