* **Grouping and Aggregating:-** Compining multiple pieces of data into a single result

* **Median Function**

→ df['Salary'].median() → 57287.0

→ df.median() → تحسب ال median لكل column يحمل قيم عددية

→ Salary 57287
Age 29
Work Week Hrs 40.0

* **describe function**

→ df.describe() → for a quick overview for the data

→ (count, mean, std, min, 25%, 50%, 75%, max) تحسب

* **Count function**

→ df['Salary'].count() → عدد العناصر الموجوده في ال column
بمعنى عدد الناس اللى جاوبت على السؤال.

* **value_counts()**

→ df['Hobbyist'].value_counts() → Yes 71257
No 17626

عدد الناس اللى جاوبوا بـ Yes أو No ←
مثلا عدد الناس اللى بتستخدم insta أو FB أو YT ←

→ df['Hobbyist'].value_count(normalize=True)

→ To break it down by percentage

→ Yes 0.8017 No 0.1983

Toeach group independently

DataFrames into groups

Grouping :- 1- split 2- Apply function 3- Combine Results

groupby() function

→ country_grp = df.groupby(['country'])

country_grp.get_group('India') → تطبع البيانات الأشخاص اللى ال country بتاعتهم India زى الفلتر على ال country

filt = df['country'] == 'India'

df.loc[filt]

→ country_grp['Social Media'].value_counts() → تحسب ال columns الخاصة بال Social media لكل country

→ country_grp['Social Media'].value_counts().loc['india'] → نفس الحوار لكن مع India فقط

→ country_grp['Social Media'].value_counts(normalize=True).loc['india'] → نفس الحوار لكن لطباعة الأرقام كنسبة

→ country_grp['Salary'].median().loc['Germany'] → للحصول على ال median الخاص بال Salary في Germany

* agg function for multiple function

→ country_grp['salary'].agg(['median','mean']).loc['India']

لطباعة ال mean والmedian الخاص بـ India

* using filtering to get python users by country

→ filt = df['country']=='India'

df.loc[filt]['Language'].str.contains('Python').sum()

→3105 → India في Python لطباعة عدد مستخدمي

* لا يمكن استبدال df.loc[filt] بـ country_grp فقط يلزم نستخدم apply *

→ country_grp['Language'].apply(lambda x: x.str.contains('Python').sum())

* % of developers using python in each country

→ country_respondents = df['country'].value_counts()

country_uses_python = country_grp['Language'].apply(lambda x: x.str.contains('Python').sum())

python_df = pd.concat([country_responds, country_uses_pythn], axis='columns', sort=False)

python_df['PctKnowsPython'] = (python_df['country_uses_pythn'] / python_df['country_respond']) * 100

python_df.sort_values(by='PctKnowsPython', ascending=False, inplace=True)

حل آخر مختصر → country_grp['Language'].apply(lambda x: x.str.contains('Python').sum() / len(x) * 100)

* When to use "groupby" in pandas?

→ Anytime you wanna Analyse some pandas sysles by some category.

→ when using the word 'For each....', what is something
ex. continent ex. mean, max, ...

※ To display a groupby in visual Form

→ %matplotlib inline

```
drinks.groupby('continent').mean().plot(kind='bar')
```

لي لعرض البيانات على شكل barplot

※ To groupby multiple columns

→ df.groupby(['Salary','Language'])

* Working with Dates and Time Series Data:-

※ convert to datetime using to_datetime:-

→ df['Date']=pd.to_datetime(df['Date'])

← ممكن تدى error بـ Unknown Format لكن ده معتمد على الـ Format

مثال ← 08-PM 2020-03-13

→ df['Date']=pd.to_datetime(df['Date'], Format= %Y-%m-%d %I-%p')

→ df.Loc[0,'Date'].day_name() → لجلب اليوم في هذا التاريخ
→ 'Friday

※ Parse dates while loading data from csv

→ d_parser=lambda x: pd.datetime.strptime(x,'%Y-%m-%d %I-%p')
df= pd.read_csv('DFname.csv',parse_dates=['Date'], date_parser = d_parser)

※ using day_name on whole series using dt class

→ df['Date'].dt.day_name() → يجلب اسم اليوم في Date كلها

* create column of dayname

→ df['DayOfWeek'] = df['Date'].dt.day_name()

* min & max methods on datetime series

→ df['Date'].min() → Timestamp('2017-07-01 11:00:00')

→ df['Date'].max() → Timestamp('2020-03-13 20:00:00')

time delta → df['Date'].max() - df['Date'].min()

→ Timedelta('986 days 09:00:00')

* Filtering by dates as string

→ Filt(df['Date'] >= '2020')

df.Loc[Filt] → الطلبات الـ rows اللي بعد سنة 2020 وما بعدها

→ Filt(df['Date'] >= '2019') & (df['Date' < '2020')

الطلبات الـ rows الخاصة بـ 2019 فقط

* Filter by to_datetime

→ Filt = (df['Date'] >= pd.to_datetime('2019-01-01'))
& (df['Date'] < pd.to_datetime('2020-01-01'))

Set date as index

→ df.set_index('Date', inplace=True)

Filtering by just passing the date in brackets

→ df['2019'] → الطريقة دي أسهل من الفلتر

Using Slices to get specific dates data

→ df['2020-01':'2020-02']

Calculating average of a slice (time Line)

→ df['2020-1':'2020-02']['Close'].mean() → 195.16559

Getting max value of a column on a given day

→ df['2020-01-01']['High'].max()

resampling (breaking down by days) a whole column into a new variable

→ df['High'].resample['D'] → Daily basis

2 days ← 2D 3D W → Weekly

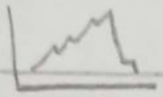
موجودة في الـ documentation

→ highs = df['High'].resample['D'].max() → يعرض أعلى سعر لكل يوم

→ highs['2020-01-01'] → لعرض أعلى سعر في يوم معين

Plotting with matplotLib in pandas

→ % matplotLib inline

high.plot() → 

resampling df with multiple columns

→ df.resample('W').mean() → يطابع الـ mean لكل الـ columns

using agg to apply different function on different columns while resampling

→ df.resamp('w').agg({'Close':'mean', 'High':'max', 'Low':'min', 'Volume':'sum'})

→ ({'اسم الـ column' : 'نوع الـ Method المستخدمة'})