
Sports Analytics
Second Interactive
OPTIMAL DECISIONS
Sports Analytics
Winter 2023

Topics for Today (20+20+20 minutes)

1. **Pandas Intro**
2. **Case Study - Data Prep for Analysis**

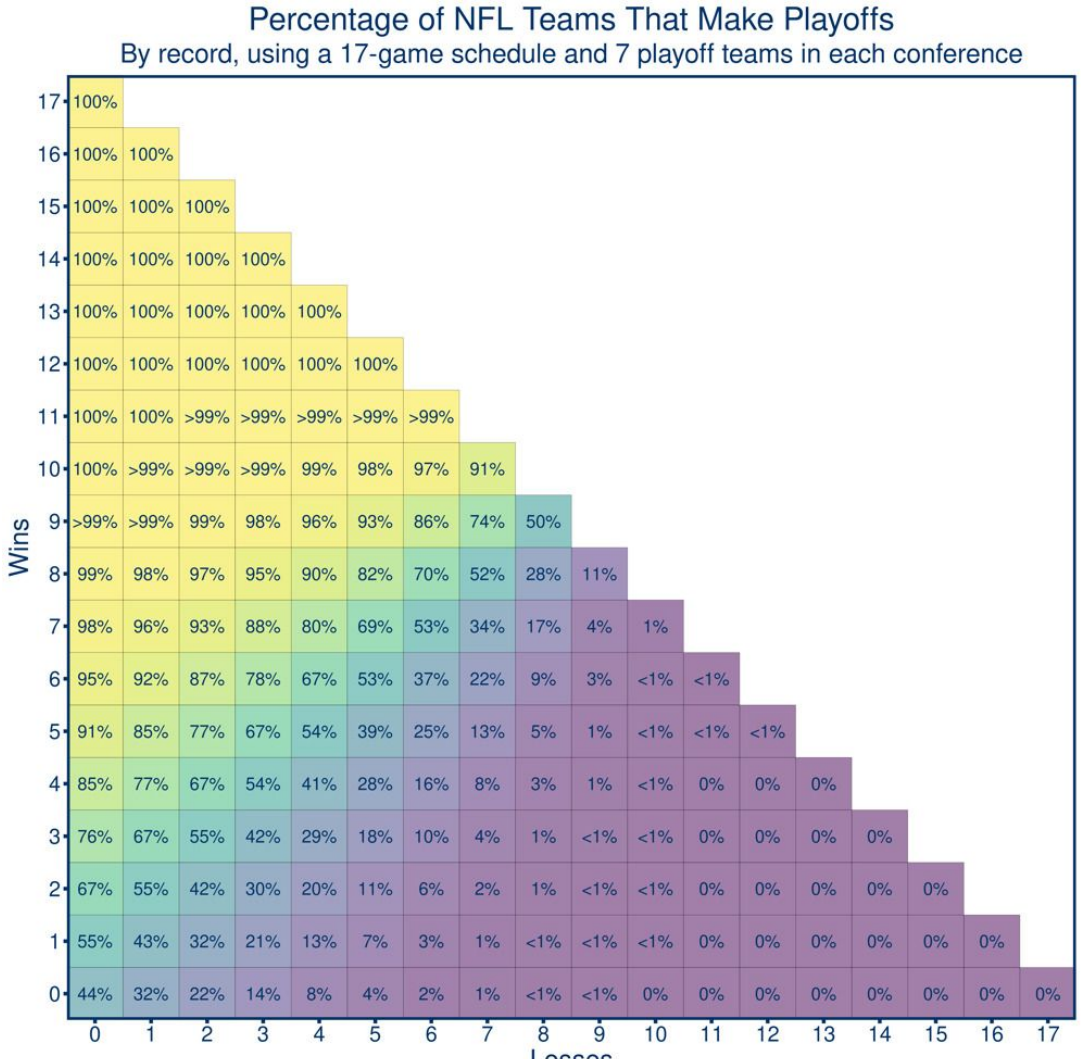
Warm up!

Just think. Don’t shout out the answer

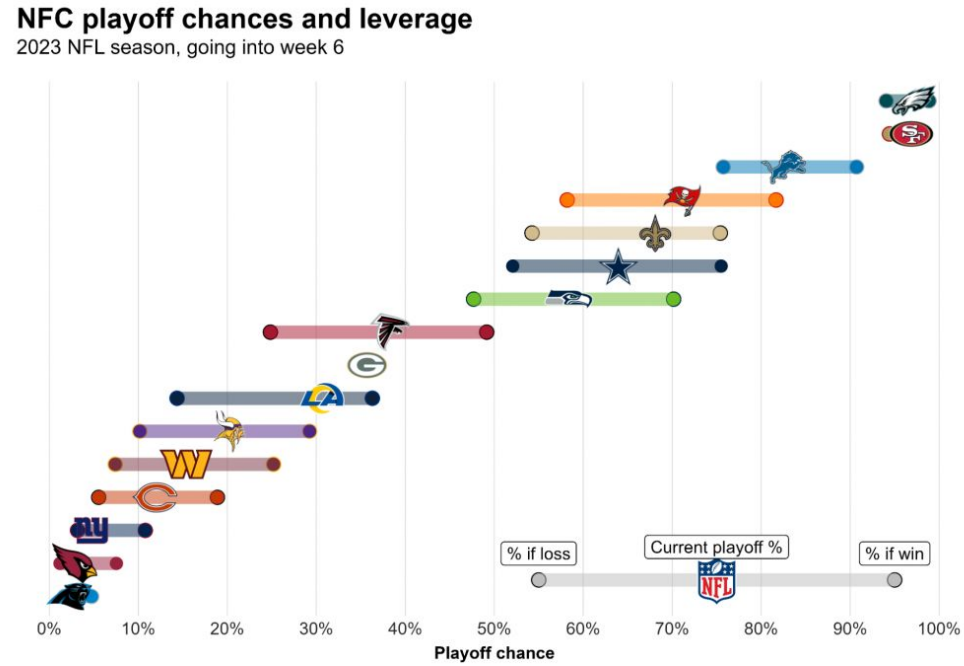
What type of Plot is this?
How to read this Plot?

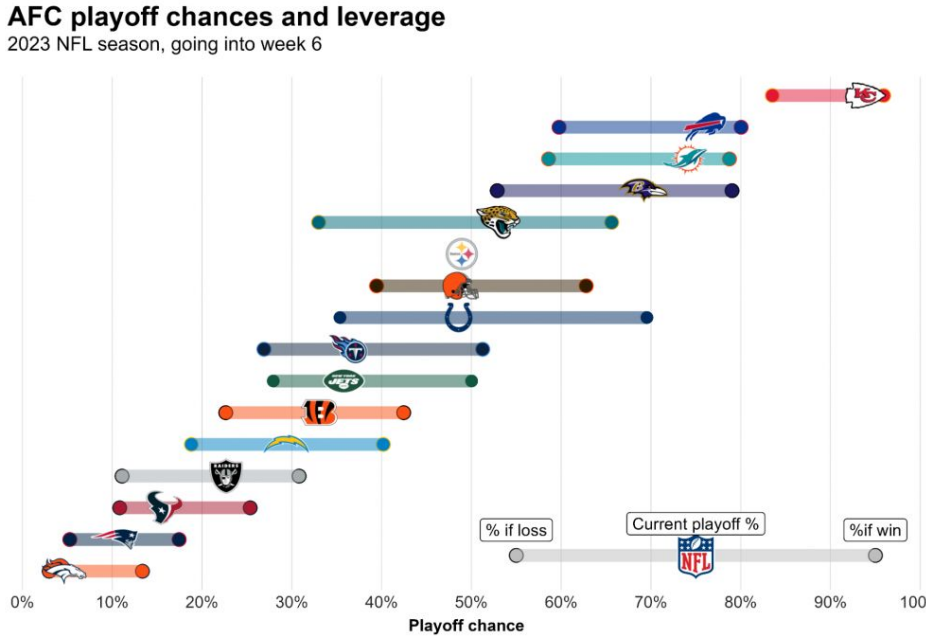
If we had 20 years of NFL data,
Can we create this ourselves?

Looking for ideas…



Playoff chances - back in Week 6





The Sports Analytics “Journey”

1. Ask some (what-if) questions
2. Frame as a “Story”
3. Gather Data
4. Descriptive
5. Visualization
6. Comparative Analysis
7. Predictive
8. Simulations
9. Communicate the Story /Presentation

If you do not know how to ask the right question,
you discover nothing.

(W. Edwards Deming)

Pythagorean Expectation

OPTIMAL DECISIONS

Sports Analytics

Winter 2023

$$\text{Win \%} = \frac{(\text{Runs Scored})^2}{(\text{Runs Scored})^2 + (\text{Runs Against})^2}$$

Pythagorean Expectation in Baseball

A really simple way to predict Win Percentages!

What's the big deal with PE - Pythagorean Expectation?

$$Win\% = \frac{Runs\,Scored^2}{Runs\,Scored^2 + Runs\,Conceded^2}$$

- Mid-season, future outlook
- "Deserve to win" vs Actual wins
- Reversion to the Mean - powerful mathematical concept
- Applies to Baseball, Basketball, NFL
- But not as such to Ice Hockey. Why would that be?!
- Can play around with Exponents

- Go over to Colab
- Idea: For any team, calculate "actual win percentage" and calculate "PE" and plot the two to see if the Expectation is being met

Pandas Basics

Someone dumps a Dataset on us.

What could we *possibly* look for?

What is the very first thing we could look for?

Size of Dataset

We can try and
“Describe” our dataset

Basic Stats (Measures)

Sum
Minimum / Maximum
Average
Median / Mode
Duplicates
Quantiles

What if there a few columns that are **text**? (Not numerical)

What can you do with text columns?

Text columns could be ways to group “stats” into useful categories

Name	Country	Operating System	Sales in Q2
Andy	USA	Mac OS	35,000
Beverly	Canada	Windows	21,000
Charlie	Canada	Windows	17,000
Donna	Mexico	Windows	28,500
Eliza	USA	Mac OS	77,400
Farooq	Mexico	Windows	9,800
George	USA	Mac OS	39,200
Harry	Mexico	Mac OS	27,200

What is Pandas?

pandas is a Python package providing fast and flexible data structures designed to make working with data both easy and intuitive.

It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

Pandas has 2 fundamental Structures

Series

and

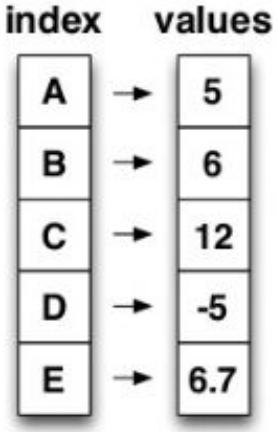
Data Frame

- Series = 1-dimensional
- Data Frame = 2 (or more) Dimensions

Pandas Series

- How many columns are in this Pandas Series?

1	‘A’
2	‘B’
3	‘C’
4	‘D’
5	‘E’

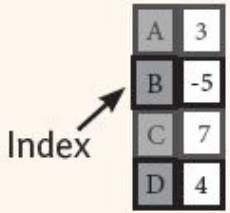


Pandas Series is very similar to a List

Pandas Data Structures

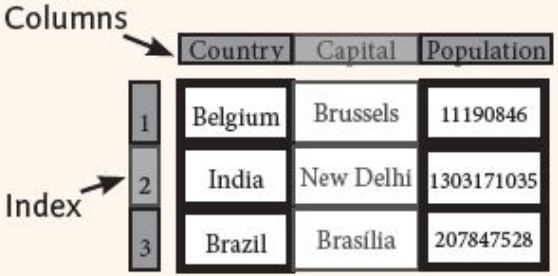
Series

A **one-dimensional** labeled array capable of holding any data type




```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

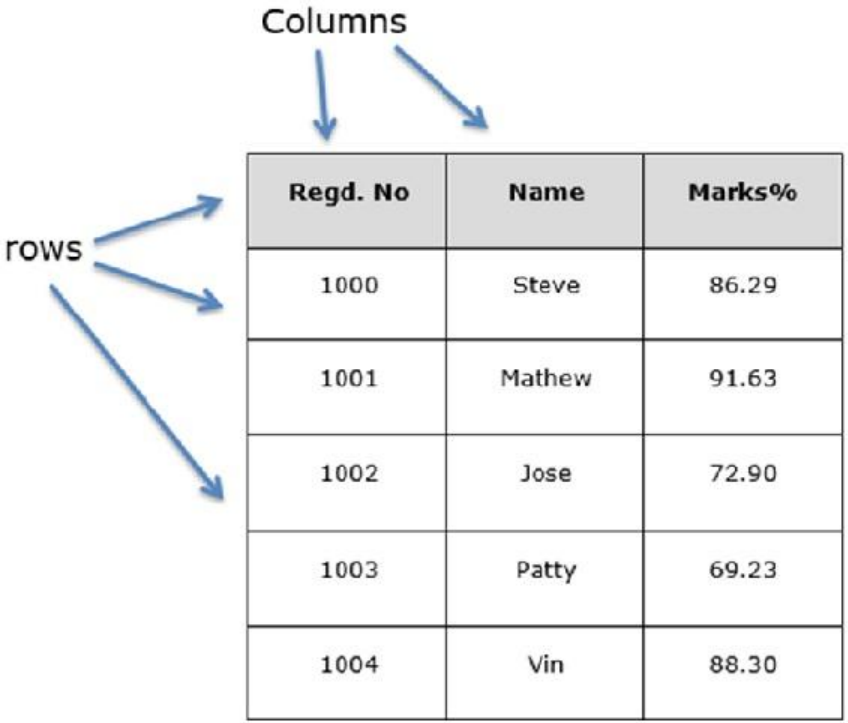


A **two-dimensional** labeled data structure with columns of potentially different types

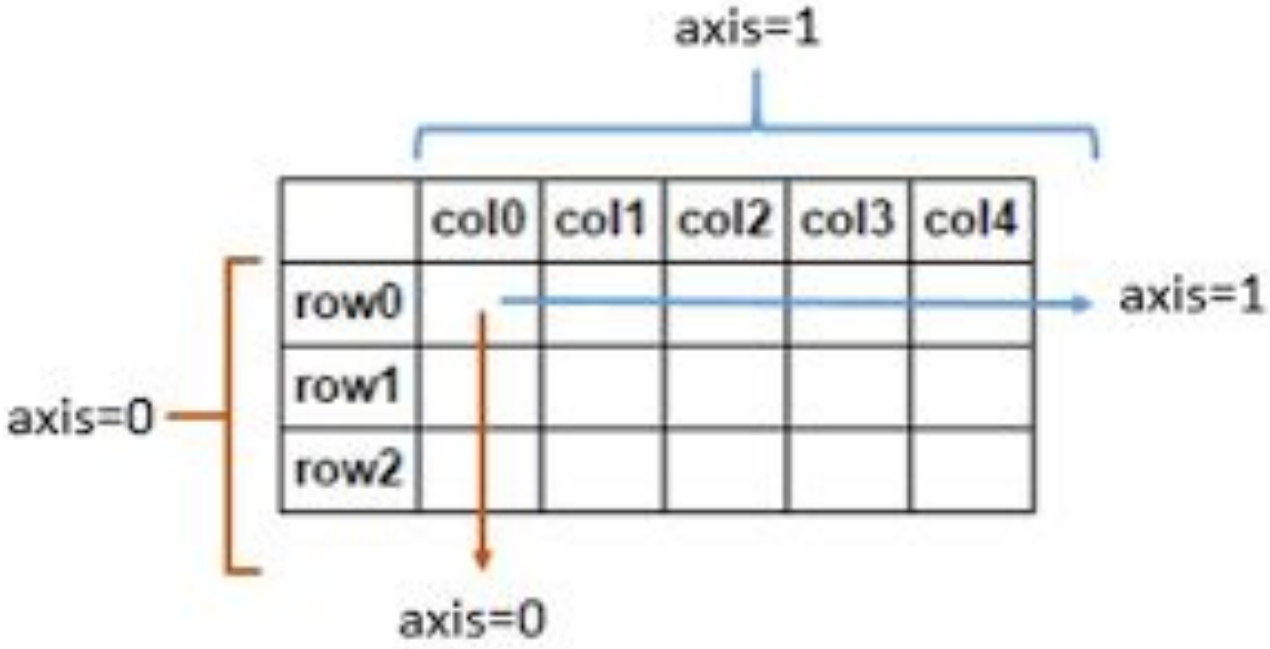
```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Pandas Data Frame

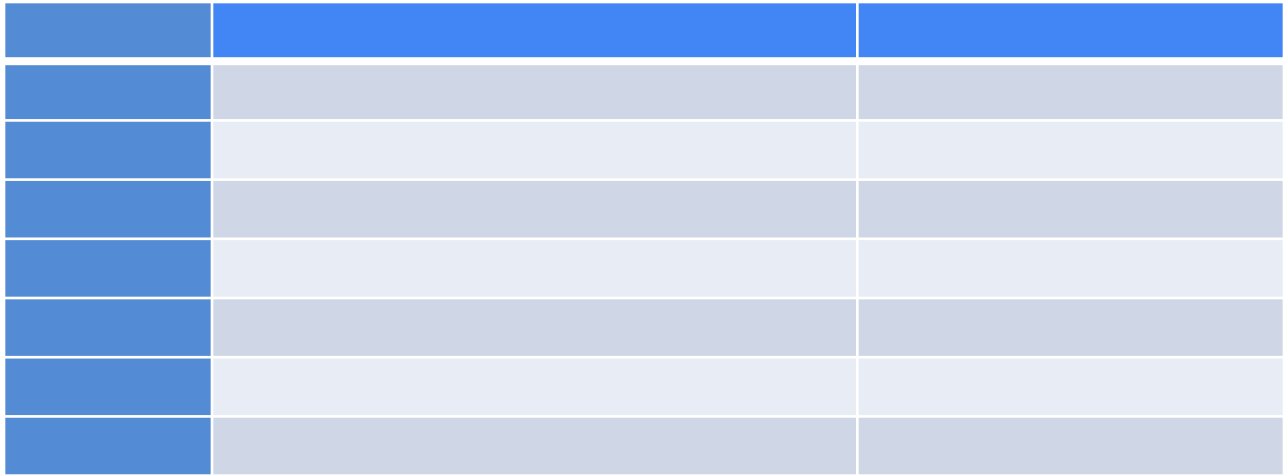


Regd. No	Name	Marks%
1000	Steve	86.29
1001	Mathew	91.63
1002	Jose	72.90
1003	Patty	69.23
1004	Vin	88.30



For simplicity, you can think of
INDEX = ROW NAMES (ROW LABELS)
COLUMNS = HEADER (COLUMN NAMES)

Pandas Dataframe



Shape = (# rows, # columns)

```
In [37]: df1
```

Out[37]:

	GEOID	State	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	04000US01	Alabama	37150	37952	42212	44476	39980	40933	42590	43464	41381
1	04000US02	Alaska	55891	56418	62993	63989	61604	57848	57431	63648	61137
2	04000US04	Arizona	45245	46657	47215	46914	45739	46896	48621	47044	50602
3	04000US05	Arkansas	36658	37057	40795	39586	36538	38587	41302	39018	39919
4	04000US06	California	51755	55319	55734	57014	56134	54283	53367	57020	57528

How many columns are there?
How Many Rows?

In this data frame, what are we doing in each column?

FinancialYear	2014/2015	2015/2016	2016/2017	2017/2018
Month				
April	8.7%	6.3%	6.3%	23.6%
May	7.3%	6.8%	6.0%	29.1%
June	4.7%	10.4%	6.3%	24.0%
July	5.9%	10.2%	5.3%	23.3%
August	9.1%	7.9%	9.9%	0.0%
September	8.9%	6.1%	12.1%	0.0%
October	9.2%	9.9%	7.8%	0.0%
November	9.7%	8.3%	8.9%	0.0%
December	9.2%	10.9%	6.6%	0.0%
January	8.7%	8.0%	10.4%	0.0%
February	9.3%	7.0%	10.8%	0.0%
March	9.2%	8.1%	9.7%	0.0%

What can you do with Columns?

- Number of Unique values
- Counts (Frequency of each unique value)
- Sort By Column (Ascending, Descending)
- Count Number of Missing Values
- Calculate Means for each column
- Add two columns

Some Advanced Things we can do…

- SELECT only rows that follow a particular condition
  - Very powerful!
- Remove Duplicate Values
- Apply any function to any column

You don’t have to remember all these commands.

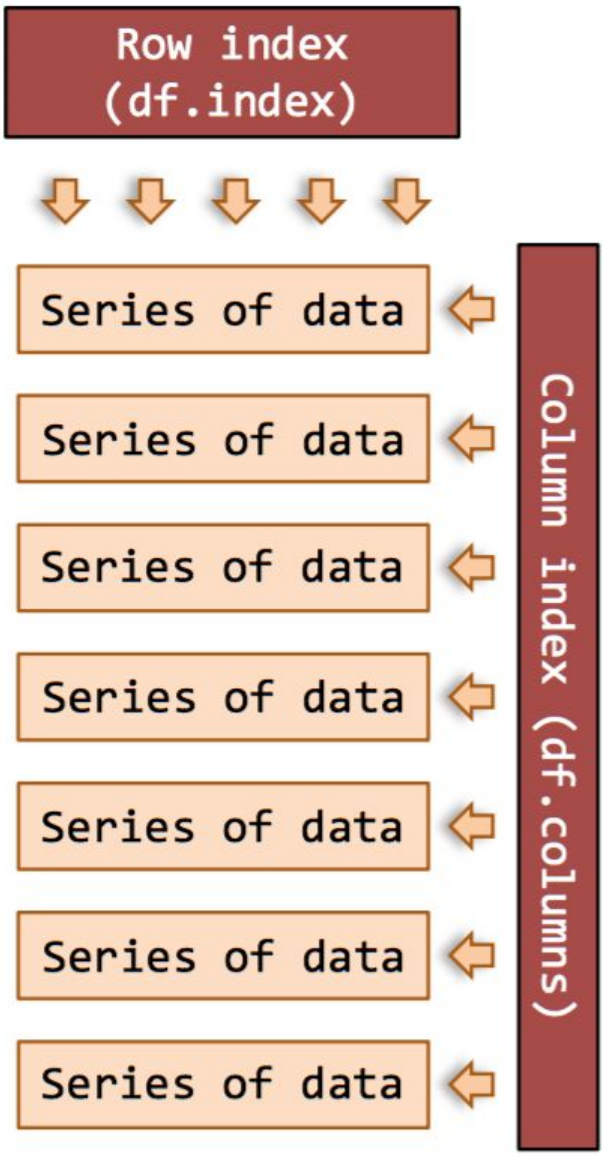
I use them a lot, but I don’t remember all the commands!

I just look them up!

What can you do with A DATA FRAME?

df = data frame

1. # Rows and columns
2. Examine (Head and Tail) of the df
3. Describe() each of the columns
    1. Statistics about the column
4. Statistics [mean, count, max, min, median]
5. Combine (concatenate) two data frames
6. Merge two data frames
7. Filter, Sort and GroupBy


Row index
(df.index)
Series of data
Series of data
Series of data
Series of data
Series of data
Series of data
Series of data
Column index (df.columns)

An Important Idea: Boolean Masks

- In Python & Pandas, **True** has the value of 1
- **False** has a value of 0

[True, True, False, True].sum() is 3

temps = [10, 7, 3, 9]

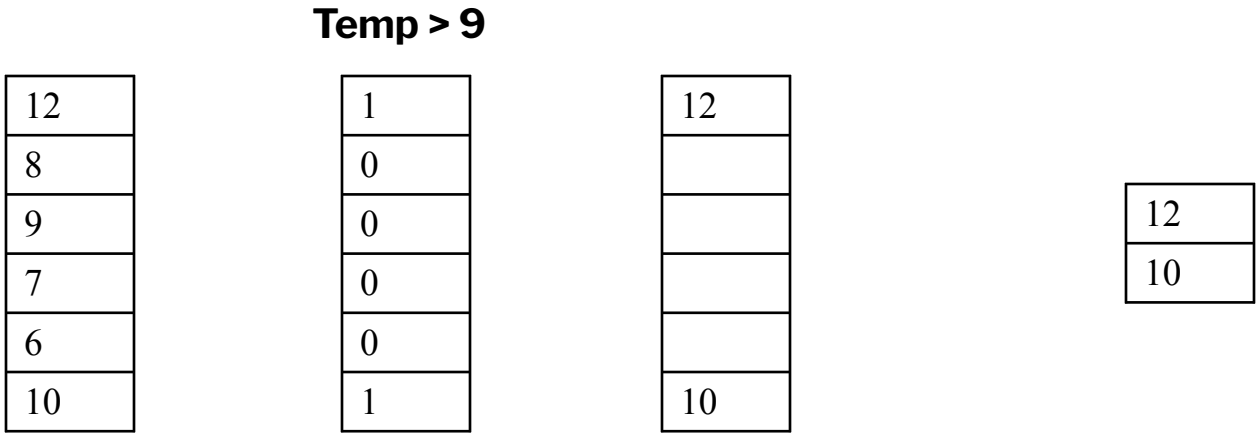
We can apply a “condition” to any column.

The result will be a True/False column.

`temps > 7` will give **[True, False, False, True]**

Understanding Pandas conditional filtering

- Boolean Mask



- True means keep the row
- False means “I don’t want the row”

temps = [10, 7, 3, 9]

condition = temps > 7

`temps > 7` will give **[True, False, False, True]**

temps[condition]

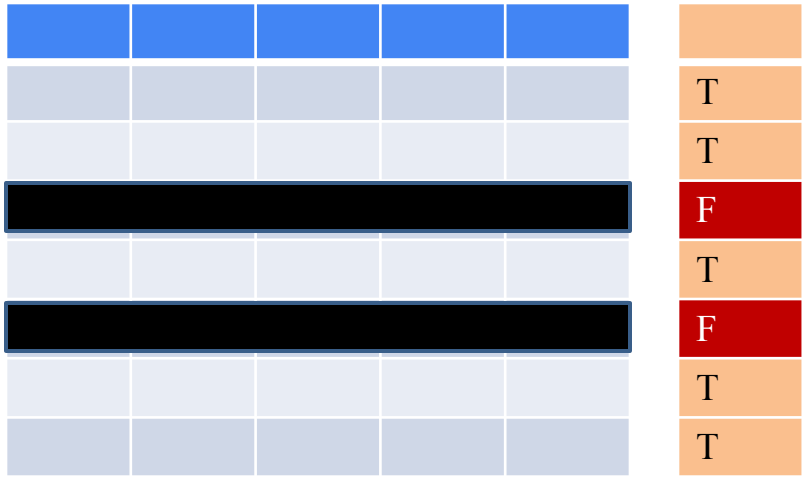
temps[temps > 7]

True means Keep it, False means leave it out

Data Frame + Boolean Mask

Apply the mask

Create the mask



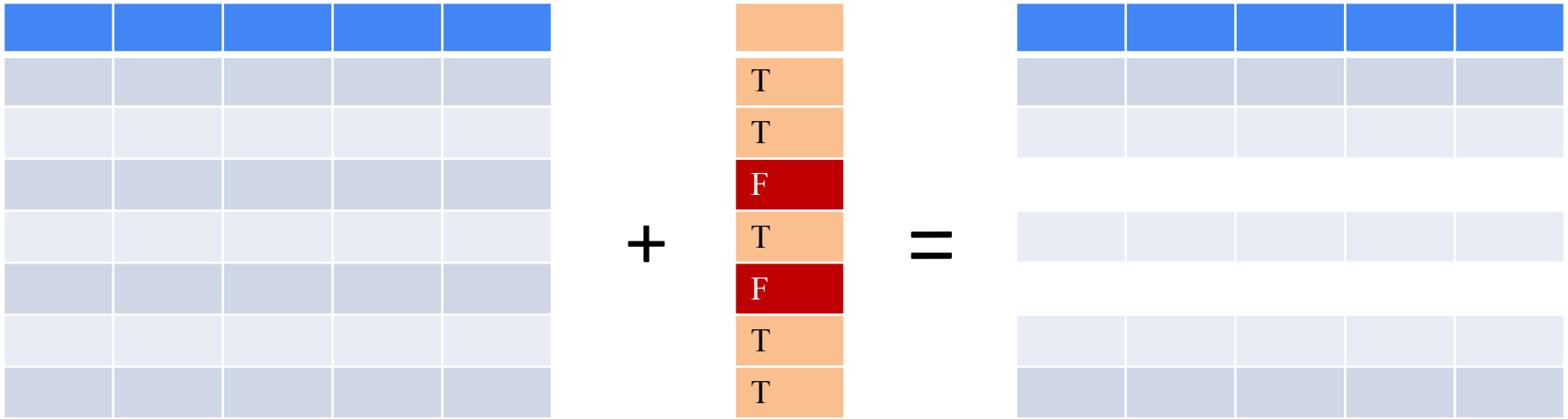
Get the result. Store it with
A new name

Keep only the rows
that match condition.
Mask (black out) all
other rows

End up with fewer
rows

Data Frame + Boolean Mask

df + condition = Boolean Column



df

Boolean Column

= Smaller_df

Grouping Operations

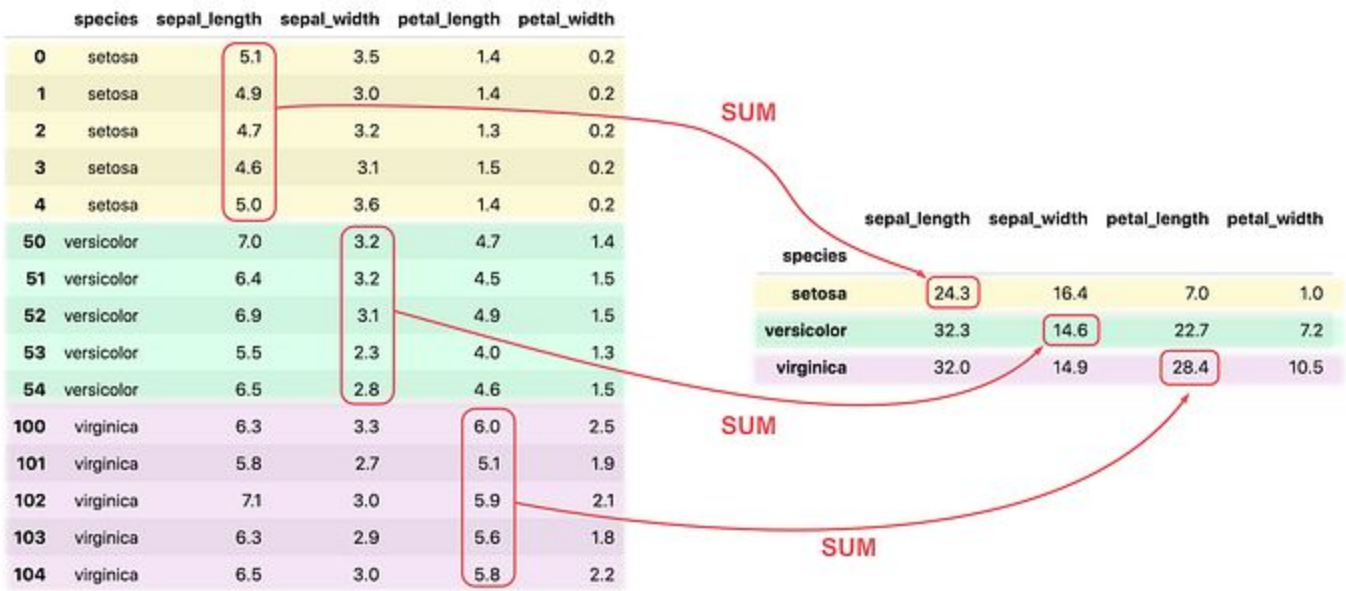
Sub-group Insights

Getting insights at each subgroup level is a core part of exploratory sports data analysis

To understand how various sub-categories are performing

1. Win Percentages by Region | Conference | League
2. “Production” by Player
3. High performing Venues or Stadiums for a League
4. Chronic reasons for failures. (Group by type of unforced errors)

Image Source: Sunscrapers



Sort into groups based on “Grouping Column(s)”
For each group, take some OTHER Column (numeric)
Apply some statistic function. (SUM, Count, Average etc.)
Combine and present the results

Data Prep Case Study

OPTIMAL DECISIONS

Sports Analytics

Winter 2023

Real Life Case Stude: Soccer Players Dataset - Physical attributes, Financial, Clubs

	Name	Age	Nationality	Club	Loaned From	Value	Wage	Height	Weight
0	L. Messi	31	Argentina	FC Barcelona	NaN	€110.5M	€565K	5'7	159lbs
1	Cristiano Ronaldo	33	Portugal	Juventus	NaN	€77M	€405K	6'2	183lbs
2	Neymar Jr	26	Brazil	Paris Saint-Germain	NaN	€118.5M	€290K	5'9	150lbs
3	De Gea	27	Spain	Manchester United	NaN	€72M	€260K	6'4	168lbs
4	K. De Bruyne	27	Belgium	Manchester City	NaN	€102M	€355K	5'11	154lbs
...	...	...	...	...	...	...	...	...	...
17316	R. Akbari	18	Australia	Melbourne Victory	NaN	€140K	€1K	5'11	150lbs
17317	A. Reyes Ambuila	18	Colombia	Atlético Nacional	NaN	€130K	€1K	5'10	165lbs
17318	I. Sadiq	18	Ghana	FC Nordsjælland	NaN	€140K	€1K	5'9	154lbs
17319	C. Merrie	19	England	Wigan Athletic	NaN	€140K	€3K	5'11	150lbs
17320	L. Morrison	19	Scotland	Sligo Rovers	NaN	€130K	€1K	NaN	NaN

	Loa ed From	Club
12269	NaN	Independiente Medellín
10834	NaN	Envigado FC
12811	NaN	NaN
5746	NaN	SK Slavia Praha
10514	SV Da nstadt 98	TSV 1860 München
5747	NaN	Gimnasia y Esgrima La Plata
2345	NaN	Stoke City
11282	NaN	Independiente Medellín
6856	NaN	Portimonense SC
14757	NaN	GIF Sundsvall

	Name	Value	Wage	Releas
4287	D. Andrade	€2M	€2K	
12182	Yang Xiaotian	€290K	€4K	
4233	J. Quiñones	€3.9M	€25K	
5781	Roldão Riso	€925K	€12K	
16321	B. Vera	€200K	€1K	
3770	M. Abeid	€3.2M	€10K	
5912	K. Mączyński	€700K	€5K	
10387	N. Mezquida	€650K	€2K	
6098	M. Petković	€1.1M	€1K	
2066	S. Sinclair	€6.5M	€48K	

	Name	Height	Weight
2361	S. Deli	6'4	185lbs
11472	P. Taylor	5'7	157lbs
11370	S. Salinas	5'10	150lbs
13780	J. Kotzke	6'0	165lbs
9962	M. Bakker	6'0	159lbs
6720	R. Gall	5'9	154lbs
957	Wilson Eduardo	5'10	161lbs
10490	Álvaro Traver	6'2	168lbs
653	C. Wilson	5'11	146lbs
4366	Bruno Alves	6'2	179lbs

€11.7M

Data Clean up activities Needed

1. **Drop** `Loaned From` **Column**: Not relevant to the analysis.
2. If a player has no Club affiliation, we want to **Drop that row**
3. **Remove** (strip) the **'€'** symbol, and
4. There are 3 financial columns -- `Wages` `Value` and `Release Clause`. These are in M, or K formats –we want to **Convert them to proper integers**.
5. Convert Dollar values **strings to Integers**
6. If a Player's "value" is missing, we want to drop that player **(Drop Row)**
7. The Heights are given as a string `5'11"` to be **Converted to Integer** – `71` inches.
8. **Drop the rows** where Player Height is missing
9. Each player's weight has `lbs` attached to it. **Strip “lbs”**, **Convert into an integer**.

Let’s Go to Colab
And do it step by step

Case Study: Interactive