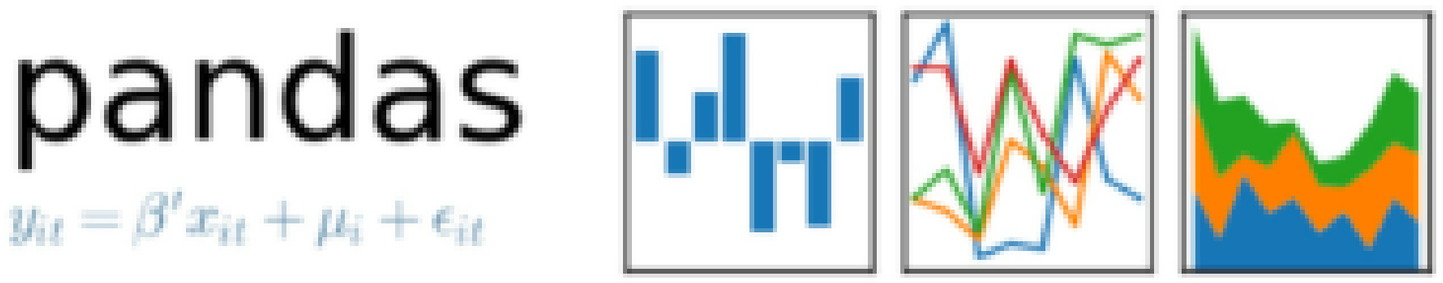
Pandas - Essential Concepts and Guide

A concise guide with code, explanations and visuals



AUTHOR: POOJA JAIN

WHAT IS PANDAS?

- **Pandas** is a Python library, widely used for data manipulation, data analysis, data cleaning.
- It provides 2 Primary data structure:
  - **SERIES** : **One-dimensional array** like object that can hold any data type [Usually holds data of single data type --> HOMOGENEOUS DATA]. It is similar to a **column** in a table.
  - **DATAFRAME** : **Two-dimensional** heterogeneous tabular data structure with labelled axes (Rows and Columns). It is similar to a **database or Excel**.

season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin
2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0
2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0
2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0
2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0
2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0

SERIES

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margi
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.

DATAFRAME

WAYS OF CREATING A DATAFRAME

```
df = pd.read_csv('matches.csv')
```

```
df = pd.read_excel('matches.xlsx', sheet_name = 'Sheet 1')
```

```
data = {'Roll_No':[1,2,3,4,5],
        'marks':[98,67,89,72,79]}
df = pd.DataFrame(data)
```

```
df
```

	Roll_No	marks
0	1	98
1	2	67
2	3	89
3	4	72
4	5	79

(Using dictionary)

Pandas is usually not installed by default. You need to install it, if you want to use it.

- In Jupyter Notebook you can write this command in the cell and execute : **!pip install pandas**
- In other IDE'S, you can go to **terminal** and type : **pip install pandas**
- If it is already installed you get **'Requirement already satisfied...'**

For further learning Pandas functions, we will be using the IPL Match dataset

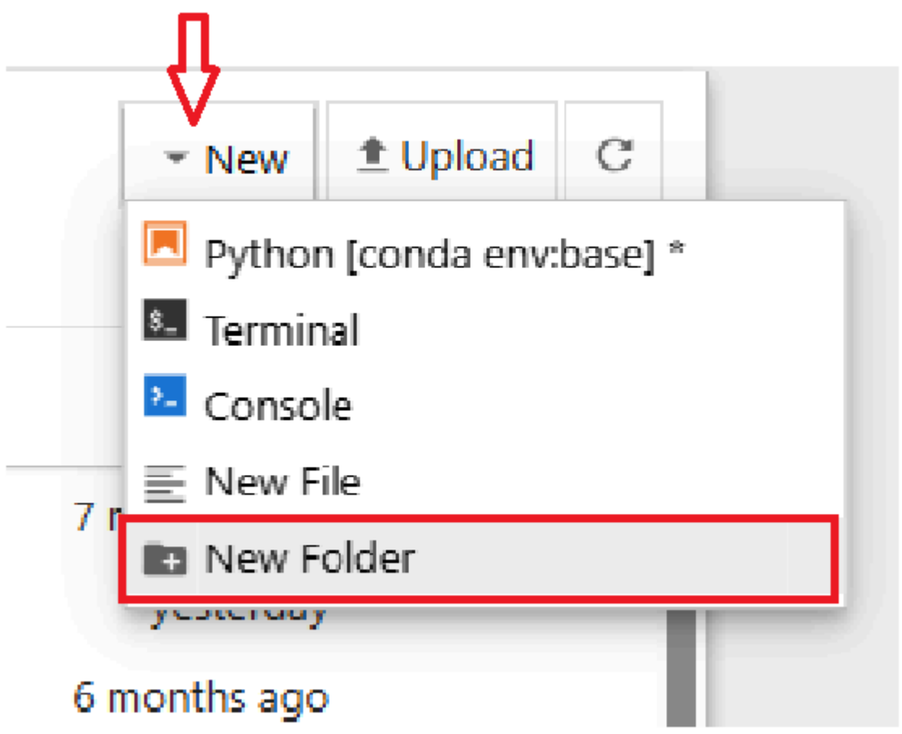
- Each row represents a **single IPL match**.
- Each columns describes various aspects of that match.

Column Name	Description
`id`	Unique identifier for the match.
`season`	The IPL season
`city`	City where the match was played.
`date`	Date of the match in YYYY-MM-DD format.
`match_type`	Type of match – It is “League” unless playoffs or finals are included.
`player_of_match`	The player who was awarded "Player of the Match."
`venue`	Stadium where the match was held.

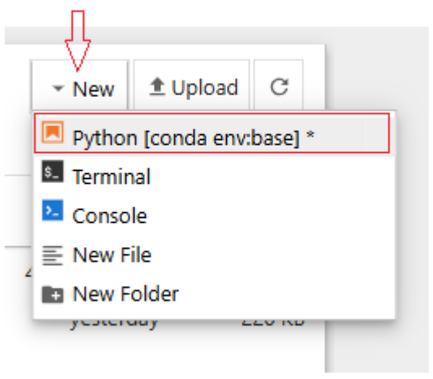
Column Name	Description
`team1`	One of the two teams that played.
`team2`	The other team that played.
`toss_winner`	Team that won the toss.
`toss_decision`	Decision made by toss winner: "bat" or "field."
`winner`	Team that won the match.
`result`	Indicates how the match was won: e.g., by “runs” or “wickets.”
`result_margin`	Margin of victory – number of runs or wickets.
`target_runs`	Target set for the team batting second.
`target_overs`	Number of overs the chasing team had (usually 20 in T20).
`super_over`	Indicates if a **Super Over** was played (Y/N).
`method`	If the match was decided using a special method (e.g., **DLS**), this would be filled, else it will be NA.
`umpire1`	Name of the first on-field umpire.
`umpire2`	Name of the second on-field umpire.

Notebook Setup

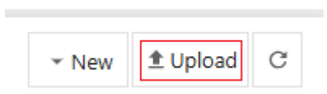
1. Go to Jupyter Notebook --> Click on 'New' --> Click on 'New Folder' --> Make a new folder and give it a name.



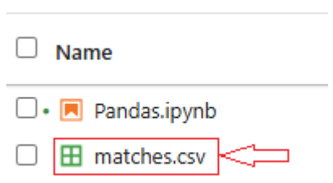
2. To create a new notebook (.ipynb) --> Open the folder you just created --> Click on 'New' --> Click on 'Python [conda env:base] *' --> Name the .ipynb file accordingly



3. If you want to work with any dataset/ file during your analysis you need to add those files in this folder --> Click on 'Upload' --> Choose the file --> Click 'Open'



4. Since we are working on IPL Match dataset, the 'matches.csv' file is added in this folder.



In [1]:
```
# Importing pandas and giving it an alias
# Why alias ? --> 1. You can write pd everytime you want to use it, instead of pandas which is relatively long 2. Also the code is more readable
import pandas as pd
```

Reading a file

- Reading the csv file (dataset) and storing the data in 'df' variable so that it can used later
- The read_csv() function in pandas is used to read a CSV (Comma Seperated Values) file
- You need to pass the exact name of the file you want to read as parameter

In [2]:
```
df = pd.read_csv('matches.csv')
```

In [3]:
```
# This is how our data looks like
# What is this df? --> It is a variable that stores the data and stands for dataframe [since it is 2-D and Tabular data]
df
```

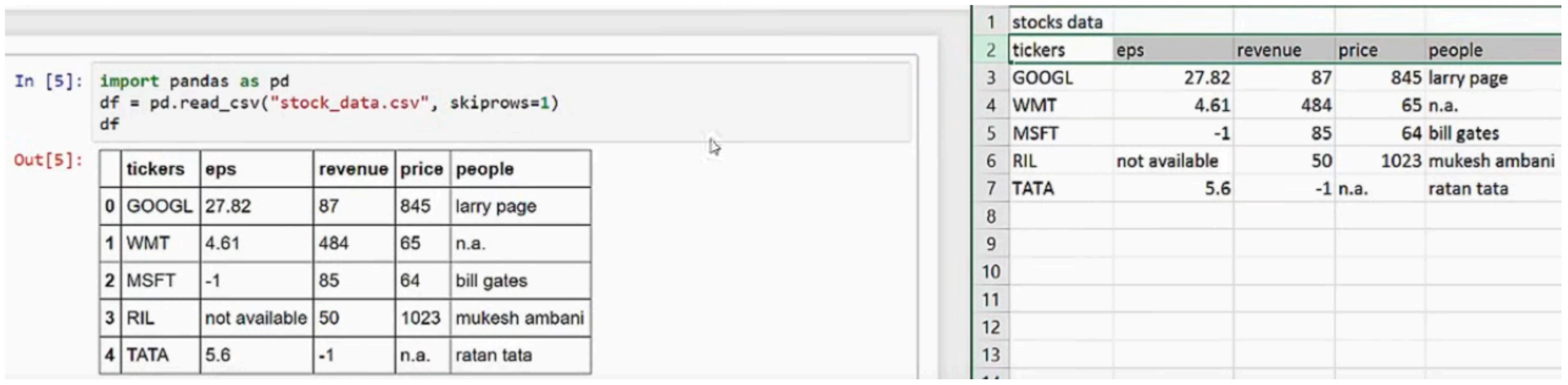
Out[3]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	NaN
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1090	1426307	2024	Hyderabad	2024-05-19	League	Abhishek Sharma	Rajiv Gandhi International Stadium, Uppal, Hyd...	Punjab Kings	Sunrisers Hyderabad	Punjab Kings	bat	Sunrisers Hyderabad	wickets	4.0	215.0	20.0	N	NaN
1091	1426309	2024	Ahmedabad	2024-05-21	Qualifier 1	MA Starc	Narendra Modi Stadium, Ahmedabad	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	160.0	20.0	N	NaN
1092	1426310	2024	Ahmedabad	2024-05-22	Eliminator	R Ashwin	Narendra Modi Stadium, Ahmedabad	Royal Challengers Bengaluru	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	4.0	173.0	20.0	N	NaN
1093	1426311	2024	Chennai	2024-05-24	Qualifier 2	Shahbaz Ahmed	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Rajasthan Royals	Rajasthan Royals	field	Sunrisers Hyderabad	runs	36.0	176.0	20.0	N	NaN
1094	1426312	2024	Chennai	2024-05-26	Final	MA Starc	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	114.0	20.0	N	NaN

1095 rows × 20 columns

ADDITIONAL PARAMETERS THAT THE READ_CSV() FUNCTION TAKES:

- These parameters are OPTIONAL to use, however they are very useful:
  - **skiprows**: This function is used to skip specified number of rows and display the remaining (If skiprows = 5, then first 5 rows will be skipped and the very next row will be made the header/column names). This function is often used where there are extra rows above the header that you don't want to display in your dataframe



In [5]:
```
import pandas as pd
df = pd.read_csv("stock_data.csv", skiprows=1)
df
```

Out[5]:

	tickers	eps	revenue	price	people
0	GOOGL	27.82	87	845	larry page
1	WMT	4.61	484	65	n.a.
2	MSFT	-1	85	64	bill gates
3	RIL	not available	50	1023	mukesh ambani
4	TATA	5.6	-1	n.a.	ratan tata

stock_data.csv

1	stocks data				
2	tickers	eps	revenue	price	people
3	GOOGL	27.82	87	845	larry page
4	WMT	4.61	484	65	n.a.
5	MSFT	-1	85	64	bill gates
6	RIL	not available	50	1023	mukesh ambani
7	TATA	5.6	-1	n.a.	ratan tata
8					
9					
10					
11					
12					
13					

- **header**: The ROW number that is specified will be the header/ column name for the dataset (If header = 2, then row 2 will be made as header).

```
df = pd.read_csv('matches.csv', header = 3)
df.head()
```

	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals.1	bat	Delhi Daredevils.1	wickets	9	130	20
0	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0
1	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0
2	335987	2007/08	Jaipur	2008-04-21	League	SR Watson	Sawai Mansingh Stadium	Rajasthan Royals	Kings XI Punjab	Kings XI Punjab	bat	Rajasthan Royals	wickets	6.0	167.0	20.0

- If header=None then column names/header is removed and replaced with Pandas self-generated Index numbers

```
df = pd.read_csv('matches.csv', header = None)
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10
0	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision
1	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field
2	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat

- **nrows**: The number of ROWS to be displayed (If nrows=3, then only first 3 rows will be displayed)

```
df = pd.read_csv('matches.csv', nrows=3)
df.head()
```

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat

- **na_values**: Usually the empty values are represented as '?','n.a.','not available',etc. in raw data. We specify these values that should be recognized and converted to 'NaN'.

```
In [14]: import pandas as pd
         df = pd.read_csv("stock_data.csv", na_values=["not available","n.a."])
         df
```

Out[14]:

	tickers	eps	revenue	price	people
0	GOOGL	27.82	87	845.0	larry page
1	WMT	4.61	484	65.0	NaN
2	MSFT	-1.00	85	64.0	bill gates
3	RIL	NaN	50	1023.0	mukesh ambani
4	TATA	5.60	-1	NaN	ratan tata

	tickers	eps	revenue	price	people
2	GOOGL	27.82	87	845	larry page
3	WMT	4.61	484	65	n.a.
4	MSFT	-1	85	64	bill gates
5	RIL	not available	50	1023	mukesh ambani
6	TATA	5.6	-1	n.a.	ratan tata

- For each of the columns we can specify which values should be recognized and converted to 'NaN'

```
: import pandas as pd
  df = pd.read_csv("stock_data.csv", na_values={
          'eps': ["not available","n.a."],
          'revenue': ["not available","n.a.", -1],
          'people': ["not available","n.a."]
      })
  df
```

:

	tickers	eps	revenue	price	people
0	GOOGL	27.82	87.0	845	larry page
1	WMT	4.61	484.0	65	NaN
2	MSFT	-1.00	85.0	64	bill gates
3	RIL	NaN	50.0	1023	mukesh ambani
4	TATA	5.60	NaN	n.a.	ratan tata

	tickers	eps	revenue	price	people
2	GOOGL	27.82	87	845	larry page
3	WMT	4.61	484	65	n.a.
4	MSFT	-1	85	64	bill gates
5	RIL	not available	50	1023	mukesh ambani
6	TATA	5.6	-1	n.a.	ratan tata

```
# If you made some changes to your Dataframe and now you want that modified dataframe to be saved on your system as csv then you can use to_csv()
# The to_csv() function takes the following arguments: 'new_match.csv' --> Name of the new csv file
# index=False --> We don't want the Pandas self-genearted ROW Index
# columns=['season','winner'] ---> We wish to include only these columns in our new csv
# header=False --> We don't want the header/column names to be displayed
df.to_csv('new_match.csv', index=False,columns=['season','winner'],header=False)
```

	2007/08	Kolkata Knight Riders
1	2007/08	Chennai Super Kings
2	2007/08	Delhi Daredevils

- If the dataset is a csv file (.csv) then we use read_csv()
- If the dataset is an excel file (.xlsx) then we use read_excel()
- **df = pd.read_excel('your_file.xlsx', sheet_name='Sheet1')** # sheet_name is a crucial parameter used to specify which sheet or sheets within an Excel workbook should be loaded into a DataFrame.
- If you have modified your dataset and want to store it on your system as excel, then you can use **to_excel()**
- **df.to_excel("output.xlsx", sheet_name='Sheet1', index=False)**
- The following code is used to write 2 different dataframes in a single excel sheet
- Here df_stocks and df_weather are 2 dataframes
- 'stocks_weather.xlsx' is the file name and 'stocks' and 'weather' are sheet names in which the respective dataframes will be written

```
df_stocks = pd.DataFrame({
    'tickers': ['GOOGL', 'WMT', 'MSFT'],
    'price': [845, 65, 64 ],
    'pe': [30.37, 14.26, 30.97],
    'eps': [27.82, 4.61, 2.12]
})

df_weather =  pd.DataFrame({
    'day': ['1/1/2017','1/2/2017','1/3/2017'],
    'temperature': [32,35,28],
    'event': ['Rain', 'Sunny', 'Snow']
})
```

```
with pd.ExcelWriter('stocks_weather.xlsx') as writer:
    df_stocks.to_excel(writer, sheet_name="stocks")
    df_weather.to_excel(writer, sheet_name="weather")
```

```
In [4]: # Want to check the data structure/data type of this data?
        # The type() function comes to rescue and checks the data type
        type(df)

        # Ah, I was right – I knew it was a DataFrame! 😊
```

```
Out[4]: pandas.core.frame.DataFrame
```

```
In [5]: # The 'columns' attribute returns the column labels of the DataFrame
```

```
In [5]: df.columns
```

```
Out[5]: Index(['id', 'season', 'city', 'date', 'match_type', 'player_of_match',
       'venue', 'team1', 'team2', 'toss_winner', 'toss_decision', 'winner',
       'result', 'result_margin', 'target_runs', 'target_overs', 'super_over',
       'method', 'umpire1', 'umpire2'],
      dtype='object')
```

```
In [6]: # The 'index' attribute returns the range between which our df lies (Basically the ROW index)
        # Starts from 0, Stops at 1095, howvever 1095 is not included (only till 1094 it is included) and step size is 1
        df.index
```

```
Out[6]: RangeIndex(start=0, stop=1095, step=1)
```

Head Function

- The head() function in pandas is used to display a specified number of rows from the TOP. (can be used on both DataFrames and Series.)
- It is used to preview/inspect the structure and content of your data, especially when the datasets is large.
- If no argument is passed then by default it displays top 5 rows

```
In [7]: df.head()
```

Out[7]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	umpi
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	NaN	A F
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN	Ben
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN	Ale
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN	SJ D
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	NaN	Bow

```
In [8]: # If an integer argument is passed, then it displays those number of rows
        # 3 is passed as argument, hence top 3 rows are fetched
        df.head(3)
```

Out[8]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	umpire1
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	NaN	Asad Rauf
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN	MR Benson
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN	Aleem Dar

```
In [9]: # If a negative integer 'n' is passed as argument, head() returns all rows except the last n rows.
        # For example, df.head(-2) would return all rows except the last two.
```

Tail Function

- The tail() function in pandas is complimentary to head() function
- Used to display a specified number of rows from the BOTTOM or last 'n' rows of DataFrame or Series
- If no argument is passed then by default it displays last 5 rows

```
In [10]: df.tail()
```

Out[10]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	
1090	1426307	2024	Hyderabad	2024-05-19	League	Abhishek Sharma	Rajiv Gandhi International Stadium, Uppal, Hyd...	Punjab Kings	Sunrisers Hyderabad	Punjab Kings	bat	Sunrisers Hyderabad	wickets	4.0	215.0	20.0	N	NaN	
1091	1426309	2024	Ahmedabad	2024-05-21	Qualifier 1	MA Starc	Narendra Modi Stadium, Ahmedabad	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	160.0	20.0	N	NaN	
1092	1426310	2024	Ahmedabad	2024-05-22	Eliminator	R Ashwin	Narendra Modi Stadium, Ahmedabad	Royal Challengers Bengaluru	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	4.0	173.0	20.0	N	NaN	A
1093	1426311	2024	Chennai	2024-05-24	Qualifier 2	Shahbaz Ahmed	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Rajasthan Royals	Rajasthan Royals	field	Sunrisers Hyderabad	runs	36.0	176.0	20.0	N	NaN	
1094	1426312	2024	Chennai	2024-05-26	Final	MA Starc	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	114.0	20.0	N	NaN	

```
In [11]: # If an integer argument is passed, then it displays those number of rows
         # 2 is passed as argument, hence last 2 rows are fetched
         df.tail(2)
```

Out[11]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	um
1093	1426311	2024	Chennai	2024-05-24	Qualifier 2	Shahbaz Ahmed	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Rajasthan Royals	Rajasthan Royals	field	Sunrisers Hyderabad	runs	36.0	176.0	20.0	N	NaN	Nitin N
1094	1426312	2024	Chennai	2024-05-26	Final	MA Starc	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	114.0	20.0	N	NaN	Madana

In [12]:
```
# If a negative value n is passed, tail() returns all rows except the first n rows.
# For example, df.tail(-2) will return all rows except the first two.
```

Shape Attribute

- The .shape attribute returns a tuple representing the size of the dataframe/series
- Output = (No. of ROWS, No. of COLUMNS)
- .shape is an attribute hence we don't use parenthesis (If we use function we use parenthesis ())

In [13]:
```
df.shape
```

Out[13]:
```
(1095, 20)
```

WHAT IS THE DIFFERENCE BETWEEN ATTRIBUTE AND FUNCTION?

ATTRIBUTE:

- An attribute basically does not perform any computation it just retrieves information.
- It behaves like a metadata (data about data) of an object
- Eg: Here our data is the actual matches dataframe and the .shape provides us with extra information about the data which is the number of rows/columns
- It is not performing any computation just returning information

FUNCTION:

- On the other hand function is used to perform an action or operation
- Eg: We have describe() function which returns statistics summary for numerical columns --> It performs mathematical operations and then returns the result.

In [14]:
```
print("Number of Rows:",df.shape[0])
print("Number of columns:",df.shape[1])
```

```
Number of Rows: 1095
Number of columns: 20
```

INFO Method

- The info() method provides a concise summary of a DataFrame.
- It is typically useful to understand the structure and content of DataFrame at a glance. It provides information such as:
  - Type of Data (Series or DataFrame)
  - Number of Rows and Columns
  - Each column's data type and non-null count (Useful in understanding if there are any null values)
  - Memory Usage: Memory consumed by the dataset (Offers insight about memory usage, which can be useful for optimizing performance with large datasets.)

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1095 entries, 0 to 1094
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               1095 non-null   int64  
 1   season           1095 non-null   object 
 2   city             1044 non-null   object 
 3   date             1095 non-null   object 
 4   match_type       1095 non-null   object 
 5   player_of_match  1090 non-null   object 
 6   venue            1095 non-null   object 
 7   team1            1095 non-null   object 
 8   team2            1095 non-null   object 
 9   toss_winner      1095 non-null   object 
 10  toss_decision    1095 non-null   object 
 11  winner           1090 non-null   object 
 12  result           1095 non-null   object 
 13  result_margin    1076 non-null   float64
 14  target_runs      1092 non-null   float64
 15  target_overs     1092 non-null   float64
 16  super_over       1095 non-null   object 
 17  method           21 non-null     object 
 18  umpire1          1095 non-null   object 
 19  umpire2          1095 non-null   object 
dtypes: float64(3), int64(1), object(16)
memory usage: 171.2+ KB
```

DESCRIBE Method

- The describe() function returns statistical summary only for numerical columns. It provides information such as:
  - count: Number of non-null entries.
  - mean: Average value.
  - std: Standard deviation.
  - min: Minimum value.
  - 25%: First quartile (25th percentile) --> 25% of data in dataset falls below this values --> Eg: If a test score of 1400 is at the 25th percentile, it means 25% of the students scored 1400 or below, and 75% scored 1400 or above.
  - 50%: Median (50th percentile) --> 50% of data in dataset falls below this values --> Eg: If a test score of 1400 is at the 50th percentile, it means 50% of the students scored 1400 or below, and 50% scored 1400 or above.
  - 75%: Third quartile (75th percentile) --> 75% of data in dataset falls below this values --> Eg: If a test score of 1400 is at the 75th percentile, it means 75% of the students scored 1400 or below, and 25% scored 1400 or above.
  - max: Maximum value.

In [16]:
```
df.describe()
```

Out[16]:

	id	result_margin	target_runs	target_overs
count	1.095000e+03	1076.000000	1092.000000	1092.000000
mean	9.048283e+05	17.259294	165.684066	19.759341
std	3.677402e+05	21.787444	33.427048	1.581108
min	3.359820e+05	1.000000	43.000000	5.000000
25%	5.483315e+05	6.000000	146.000000	20.000000
50%	9.809610e+05	8.000000	166.000000	20.000000
75%	1.254062e+06	20.000000	187.000000	20.000000
max	1.426312e+06	146.000000	288.000000	20.000000

FETCHING ROWS AND COLUMNS

- Fetching rows and columns is useful when you want to do:
  - Feature Selection: Choose only relevant columns and discard the irrelevant ones
  - Performing Analysis
  - Visualizing
  - Exploration and understanding
  - Updation

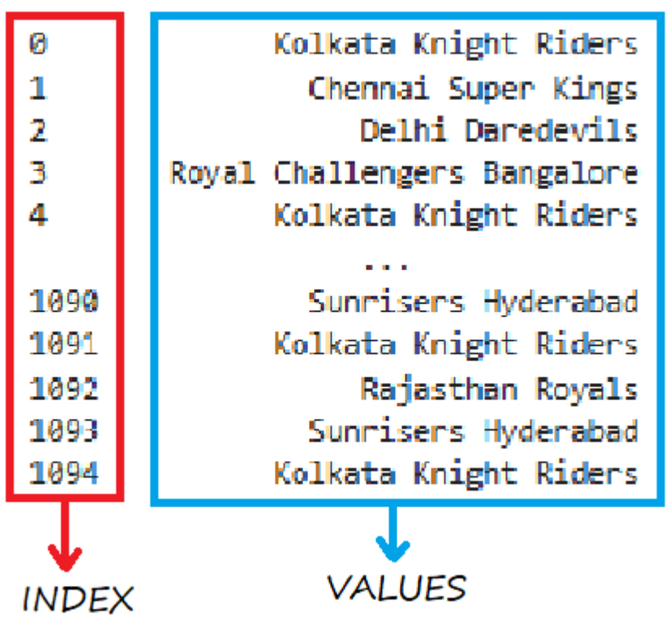
FETCHING A SINGLE COLUMN

- There are 2 ways to fetch a single column

USING BRACKET NOTATION

Syntax: df ['column name']

- We write the column name within single [ ] square brackets
- In the output we see the INDEX (which is usually self generated) and VALUES (from dataframe)



USING DOT NOTATION:

SYNATX: df.column_name

- This method is suitable only if there are no gaps between the column names (Eg: 'df.team winner' will not work here because there is a space in the column name, in such cases we use the bracket notation)

In [17]:
```
df['winner']
```

Out[17]:
```
0             Kolkata Knight Riders
1               Chennai Super Kings
2                  Delhi Daredevils
3       Royal Challengers Bangalore
4             Kolkata Knight Riders
                    ...
1090            Sunrisers Hyderabad
1091          Kolkata Knight Riders
1092               Rajasthan Royals
1093            Sunrisers Hyderabad
1094          Kolkata Knight Riders
Name: winner, Length: 1095, dtype: object
```

In [18]:
```
df.winner
```

Out[18]:
```
0             Kolkata Knight Riders
1               Chennai Super Kings
2                  Delhi Daredevils
3       Royal Challengers Bangalore
4             Kolkata Knight Riders
                    ...
1090            Sunrisers Hyderabad
1091          Kolkata Knight Riders
1092               Rajasthan Royals
1093            Sunrisers Hyderabad
1094          Kolkata Knight Riders
Name: winner, Length: 1095, dtype: object
```

In [19]:
```
# Since this is a single column (1-D) it's type is Series
type(df['winner'])
```

Out[19]:
```
pandas.core.series.Series
```

In [20]:
```
# For series data type the .shape attribute returns a tuple with single integer representing number of rows [Because we know Number of columns is = 1]
df['winner'].shape
```

Out[20]:
```
(1095,)
```

FETCHING MULTIPLE COLUMNS

Syntax: df [['column_1','column_2','column_3'...]]

- To fetch multiple columns we need to write the column names within '' seperated by commas within 2 [ ] brackets --> Passing list of column names within [ ], hence 2 [ ] brackets

In [21]:
```
df[['team1','team2','winner']]
```

Out[21]:

	team1	team2	winner
0	Royal Challengers Bangalore	Kolkata Knight Riders	Kolkata Knight Riders
1	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings
2	Delhi Daredevils	Rajasthan Royals	Delhi Daredevils
3	Mumbai Indians	Royal Challengers Bangalore	Royal Challengers Bangalore
4	Kolkata Knight Riders	Deccan Chargers	Kolkata Knight Riders
...	...	...	...
1090	Punjab Kings	Sunrisers Hyderabad	Sunrisers Hyderabad
1091	Sunrisers Hyderabad	Kolkata Knight Riders	Kolkata Knight Riders
1092	Royal Challengers Bengaluru	Rajasthan Royals	Rajasthan Royals
1093	Sunrisers Hyderabad	Rajasthan Royals	Sunrisers Hyderabad
1094	Sunrisers Hyderabad	Kolkata Knight Riders	Kolkata Knight Riders

1095 rows × 3 columns

In [22]:
```
# Although it is a subset of main data, it is 2-D, hence it is a DataFrame
type(df[['team1','team2','winner']])
```

Out[22]:
```
pandas.core.frame.DataFrame
```

APPLYING SOME STATISTICAL OPERATIONS ON COLUMNS

- These statistical operations can only be applied on numerical columns.

In [23]:
```
df['target_runs'].mean() # Using the bracket notation
```

Out[23]:
```
165.68406593406593
```

In [24]:
```
df.target_runs.max() # Using the dot notation
```

Out[24]:
```
288.0
```

In [25]:
```
df[['target_runs','target_overs']].min() # Applying the statistical operation on multiple columns
# Whenever we use multiple columns always use double []
```

Out[25]:
```
target_runs     43.0
target_overs     5.0
dtype: float64
```

In [26]:
```
# To fetch the row where target_runs were minimum we use the following code
# df['target_runs'].idxmin() --> returns the index where target_runs were minimum
# df.loc[the index where target_runs were minimum] --> returns that particular row and all columns
df.loc[df['target_runs'].idxmin()]
```

Out[26]:
```
id                            733993
season                          2014
city                           Delhi
date                      2014-05-10
match_type                    League
player_of_match             DW Steyn
venue                Feroz Shah Kotla
team1                Delhi Daredevils
team2              Sunrisers Hyderabad
toss_winner        Sunrisers Hyderabad
toss_decision                    field
winner             Sunrisers Hyderabad
result                         wickets
result_margin                      8.0
target_runs                       43.0
target_overs                       5.0
super_over                           N
method                             D/L
umpire1                 RM Deshpande
umpire2                 BNJ Oxenford
Name: 429, dtype: object
```

LOC AND ILOC

- Pandas provides 2 primary methods to access data from the dataframe.

.loc (Label-based indexing):

- This method selects data based on the labels (names) of rows and columns.
- It is **inclusive** of both the start and end labels when slicing.

.iloc (Integer-position-based indexing):

- This method selects data based on the integer positions of rows and columns, similar to standard Python list indexing.
- It is **exclusive** of the end position when slicing, meaning the element at the end position is not included.

In [27]:
```
# Using .loc to fetch single element --> In the dataframe each row is assigned an index that act as label
# df.loc[row_label,column_label]
df.loc[0,'winner']
```

Out[27]:
```
'Kolkata Knight Riders'
```

In [28]:
```
# Using .loc to fetch multiple elements
df.loc[[0,1,3,4,5,7],['season','winner']]
```

Out[28]:

	season	winner
0	2007/08	Kolkata Knight Riders
1	2007/08	Chennai Super Kings
3	2007/08	Royal Challengers Bangalore
4	2007/08	Kolkata Knight Riders
5	2007/08	Rajasthan Royals
7	2007/08	Chennai Super Kings

In [29]:
```
# Using .loc to fetch multiple elements using slicing --> start and end labels are included
df.loc[0:11,['season','winner']]
```

Out[29]:

	season	winner
0	2007/08	Kolkata Knight Riders
1	2007/08	Chennai Super Kings
2	2007/08	Delhi Daredevils
3	2007/08	Royal Challengers Bangalore
4	2007/08	Kolkata Knight Riders
5	2007/08	Rajasthan Royals
6	2007/08	Delhi Daredevils
7	2007/08	Chennai Super Kings
8	2007/08	Rajasthan Royals
9	2007/08	Kings XI Punjab
10	2007/08	Rajasthan Royals
11	2007/08	Chennai Super Kings

In [30]:
```
# Using .iloc to fetch single element --> Labels are not used here, indexes are used
# df.iloc[row_index, column_index]
df.iloc[0,11]
```

Out[30]:
```
'Kolkata Knight Riders'
```

In [31]:
```
# Using .iloc to fetch multiple elements
df.iloc[[0,2,6,7],[1,11]]
```

Out[31]:

	season	winner
0	2007/08	Kolkata Knight Riders
2	2007/08	Delhi Daredevils
6	2007/08	Delhi Daredevils
7	2007/08	Chennai Super Kings

In [32]:
```
# Using .iloc to fetch multiple elements using slicing --> end indexes are not included
df.iloc[0:11,0:5]
```

Out[32]:

	id	season	city	date	match_type
0	335982	2007/08	Bangalore	2008-04-18	League
1	335983	2007/08	Chandigarh	2008-04-19	League
2	335984	2007/08	Delhi	2008-04-19	League
3	335985	2007/08	Mumbai	2008-04-20	League
4	335986	2007/08	Kolkata	2008-04-20	League
5	335987	2007/08	Jaipur	2008-04-21	League
6	335988	2007/08	Hyderabad	2008-04-22	League
7	335989	2007/08	Chennai	2008-04-23	League
8	335990	2007/08	Hyderabad	2008-04-24	League
9	335991	2007/08	Chandigarh	2008-04-25	League
10	335992	2007/08	Bangalore	2008-04-26	League

Filtering DataFrame Based on a Condition

- Often called as **Masking**, this method is used to selectively extract data within a DataFrame or Series based on a specific condition.

In [33]:
```
# We want the data of only those matches that took place in 'Wankhede Stadium'
# But this returns a BOOLEAN SERIES [Series with TRUE and FALSE values] --> TRUE when venue = 'Wankhede Stadium' and FALSE when venue = Other location
df['venue'] == 'Wankhede Stadium'
```

Out[33]:
```
0       False
1       False
2       False
3        True
4       False
        ...
1090    False
1091    False
1092    False
1093    False
1094    False
Name: venue, Length: 1095, dtype: bool
```

In [34]:
```
# Using 1 condition
# Writing the condition within df will filter and return data of matches played in 'Wankhede Stadium'
df[df['venue'] == 'Wankhede Stadium']
```

Out[34]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	umpire
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN	SJ Davi
35	336018	2007/08	Mumbai	2008-05-14	League	ST Jayasuriya	Wankhede Stadium	Mumbai Indians	Chennai Super Kings	Mumbai Indians	field	Mumbai Indians	wickets	9.0	157.0	20.0	N	NaN	B Doctrov
37	336021	2007/08	Mumbai	2008-05-16	League	SM Pollock	Wankhede Stadium	Mumbai Indians	Kolkata Knight Riders	Mumbai Indians	field	Mumbai Indians	wickets	8.0	68.0	20.0	N	NaN	B Doctrov
44	336028	2007/08	Mumbai	2008-05-21	League	SE Marsh	Wankhede Stadium	Mumbai Indians	Kings XI Punjab	Mumbai Indians	field	Kings XI Punjab	runs	1.0	190.0	20.0	N	NaN	B Bowde
55	336038	2007/08	Mumbai	2008-05-30	Semi Final	SR Watson	Wankhede Stadium	Delhi Daredevils	Rajasthan Royals	Delhi Daredevils	field	Rajasthan Royals	runs	105.0	193.0	20.0	N	NaN	B Bowde
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
719	1178399	2019	Mumbai	2019-04-10	League	KA Pollard	Wankhede Stadium	Kings XI Punjab	Mumbai Indians	Mumbai Indians	field	Mumbai Indians	wickets	3.0	198.0	20.0	N	NaN	S Rav
722	1178402	2019	Mumbai	2019-04-13	League	JC Buttler	Wankhede Stadium	Mumbai Indians	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	4.0	188.0	20.0	N	NaN	A Nan Kishor
726	1178406	2019	Mumbai	2019-04-15	League	SL Malinga	Wankhede Stadium	Royal Challengers Bangalore	Mumbai Indians	Mumbai Indians	field	Mumbai Indians	wickets	5.0	172.0	20.0	N	NaN	N Erasmu
746	1178426	2019	Mumbai	2019-05-02	League	JJ Bumrah	Wankhede Stadium	Mumbai Indians	Sunrisers Hyderabad	Mumbai Indians	bat	Mumbai Indians	tie	NaN	163.0	20.0	Y	NaN	C Nanda
751	1178431	2019	Mumbai	2019-05-05	League	HH Pandya	Wankhede Stadium	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	Mumbai Indians	wickets	9.0	134.0	20.0	N	NaN	A Nan Kishor

73 rows × 20 columns

In [35]:
```
# Using the .shape attribute helps us know the number of matches played in 'Wankhede Stadium'
# Number of matches played = Number of rows in which 'Wankhede Stadium' appeared, hence shape[0]
df[df['venue'] == 'Wankhede Stadium'].shape[0]
```

Out[35]: 73

Using more than one condition:

- When combining more than one conditions to filter data in pandas we use BITWISE operators such as:
  - BITWISE AND --> &
  - BITWISE OR ---> |
  - BITWISE NOT --> ~
- But why use BITWISE operators and not keywords like AND, OR, NOT?
- That's because operators like AND, OR, NOT expect two **single boolean values**, not Series. But df['winner'] == 'Mumbai Indians' and df['match_type'] == 'Final' in the below code returns a **Series of booleans**, not a single True or False.
- Single Boolean Values on both side --> USE AND, OR, NOT between them
- Series of Boolean Values on both side --> USE &, |, ~ between them

It is incorrect to use BITWISE OPERATORS without proper parentheses:

- In pandas, when you're using bitwise operators like &, |, ~ for combining boolean conditions, each condition must be wrapped in parentheses.
- But why the need of wrapping conditions within parentheses?
- Because '&','|','~' has higher precedence than comparison operators (==), so Python gets confused and will raise an error if conditions are not wrapped in ( )

In [36]:
```
# Finding out matches data when 'Mumbai Indians' won in Finals
# This will return us a dataframe with all columns
df[(df['winner'] == 'Mumbai Indians') & (df['match_type'] == 'Final')]
```

Out[36]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	umpire
397	598073	2013	Kolkata	2013-05-26	Final	KA Pollard	Eden Gardens	Chennai Super Kings	Mumbai Indians	Mumbai Indians	bat	Mumbai Indians	runs	23.0	149.0	20.0	N	NaN	HDF Dharmaser
516	829823	2015	Kolkata	2015-05-24	Final	RG Sharma	Eden Gardens	Mumbai Indians	Chennai Super Kings	Chennai Super Kings	field	Mumbai Indians	runs	41.0	203.0	20.0	N	NaN	HDF Dharmaser
635	1082650	2017	Hyderabad	2017-05-21	Final	KH Pandya	Rajiv Gandhi International Stadium, Uppal	Mumbai Indians	Rising Pune Supergiant	Mumbai Indians	bat	Mumbai Indians	runs	1.0	130.0	20.0	N	NaN	NJ Llor
755	1181768	2019	Hyderabad	2019-05-12	Final	JJ Bumrah	Rajiv Gandhi International Stadium	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	Mumbai Indians	runs	1.0	150.0	20.0	N	NaN	IJ Gou
815	1237181	2020/21	NaN	2020-11-10	Final	TA Boult	Dubai International Cricket Stadium	Delhi Capitals	Mumbai Indians	Delhi Capitals	bat	Mumbai Indians	wickets	5.0	157.0	20.0	N	NaN	CB Gaffane

In [37]:
```
# But what if you wanted to see only specific columns along with the conditioning
# Between condition and df we just mention the columns we need to see
df[['season','winner']][(df['winner'] == 'Mumbai Indians') & (df['match_type'] == 'Final')]
```

Out[37]:

	season	winner
397	2013	Mumbai Indians
516	2015	Mumbai Indians
635	2017	Mumbai Indians
755	2019	Mumbai Indians
815	2020/21	Mumbai Indians

value_counts( ) function:

- The value_counts( ) method is used to count the occurrences of unique values within a Series or a column in a DataFrame.
- It returns a new Series where the index represents the unique values and the corresponding values represent their frequencies (counts).
- By default, the results are sorted in descending order of frequency, and missing values (NaN) are excluded.
- Useful if the values in a column are repeatative.
- Not Useful if values are unique and don't repeat.

In [38]:
```
df['winner'].value_counts()
```

Out[38]:
```
winner
Mumbai Indians                 144
Chennai Super Kings            138
Kolkata Knight Riders          131
Royal Challengers Bangalore    116
Rajasthan Royals               112
Kings XI Punjab                 88
Sunrisers Hyderabad             88
Delhi Daredevils                67
Delhi Capitals                  48
Deccan Chargers                 29
Gujarat Titans                  28
Lucknow Super Giants            24
Punjab Kings                    24
Gujarat Lions                   13
Pune Warriors                   12
Rising Pune Supergiant          10
Royal Challengers Bengaluru      7
Kochi Tuskers Kerala             6
Rising Pune Supergiants          5
Name: count, dtype: int64
```

In [39]:
```
# By default the results are sorted in descending order
# If you set ascending = True, the results will be sorted in ascending order
df['toss_decision'].value_counts(ascending=True)
```

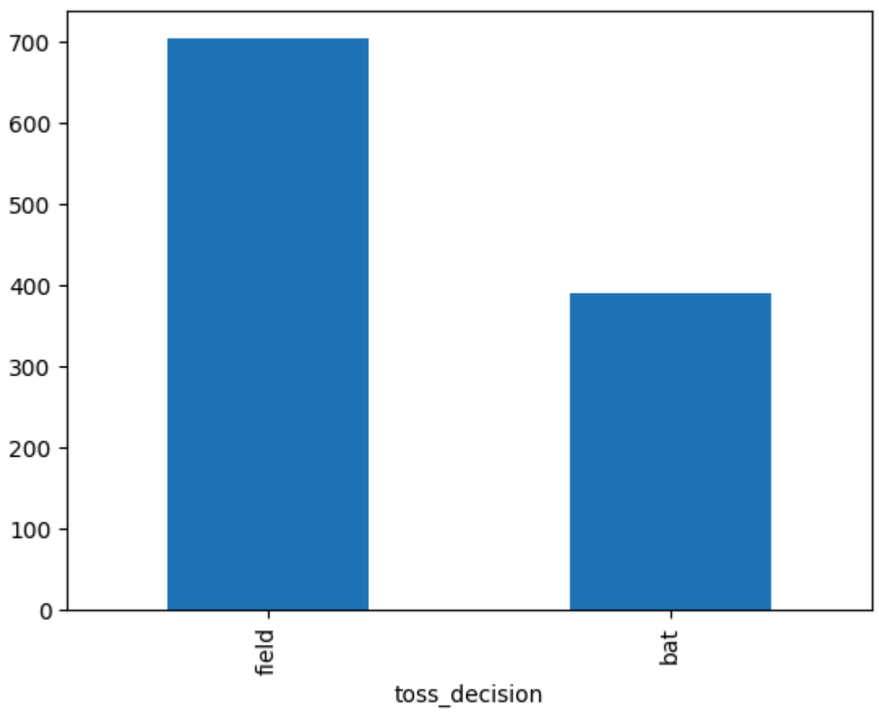
Out[39]:
```
toss_decision
bat      391
field    704
Name: count, dtype: int64
```

PLOT( ) FUNCTION:

- The plot( ) function takes in data as an input and helps generate various types of plots based on that data.
- This function takes an argument called **'kind'** that allows you to specify the desired plot type.
- Some of the common values that can be provided to the 'kind' argument is:
  - 'line' (default) --> When plot( ) is called without specifying kind argument, plot() defaults to creating a line plot.
  - 'bar' (vertical bar chart)
  - 'barh' (horizontal bar chart)
  - 'hist' (histogram)
  - 'box' (box plot)
  - 'kde' or 'density' (kernel density estimation plot)
  - 'area' (area plot)
  - 'pie' (pie chart)
  - 'scatter' (scatter plot, requires x and y arguments)
- Usually the Categorical data is kept on the x-axis and numerical data/count/frequency is kept on y-axis.

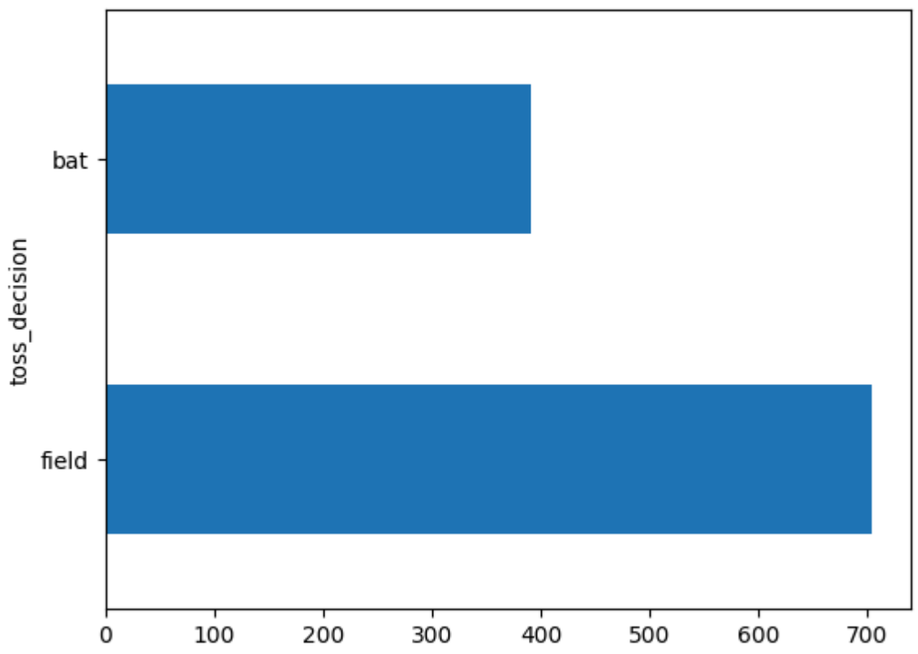
In [40]:
```
df['toss_decision'].value_counts().plot(kind='bar')  # Most of the toss decisions taken upon winning the toss is to field
```

Out[40]:
```
<Axes: xlabel='toss_decision'>
```



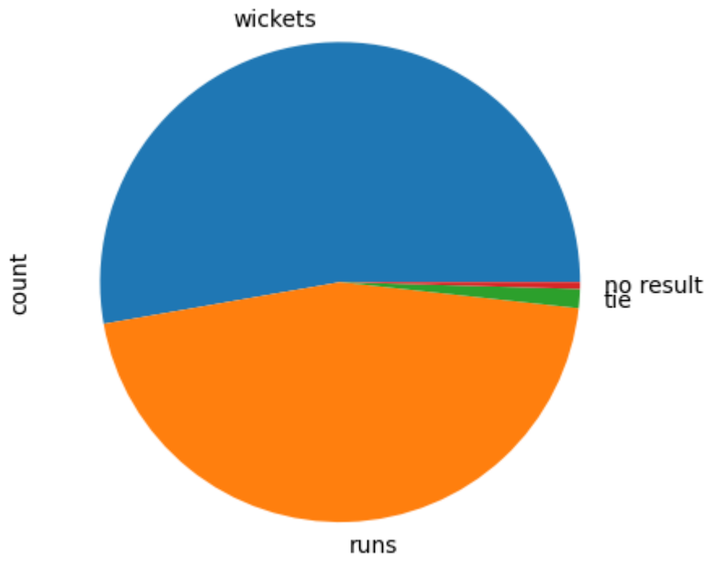
In [41]:
```
df['toss_decision'].value_counts().plot(kind='barh')
```

Out[41]:
```
<Axes: ylabel='toss_decision'>
```



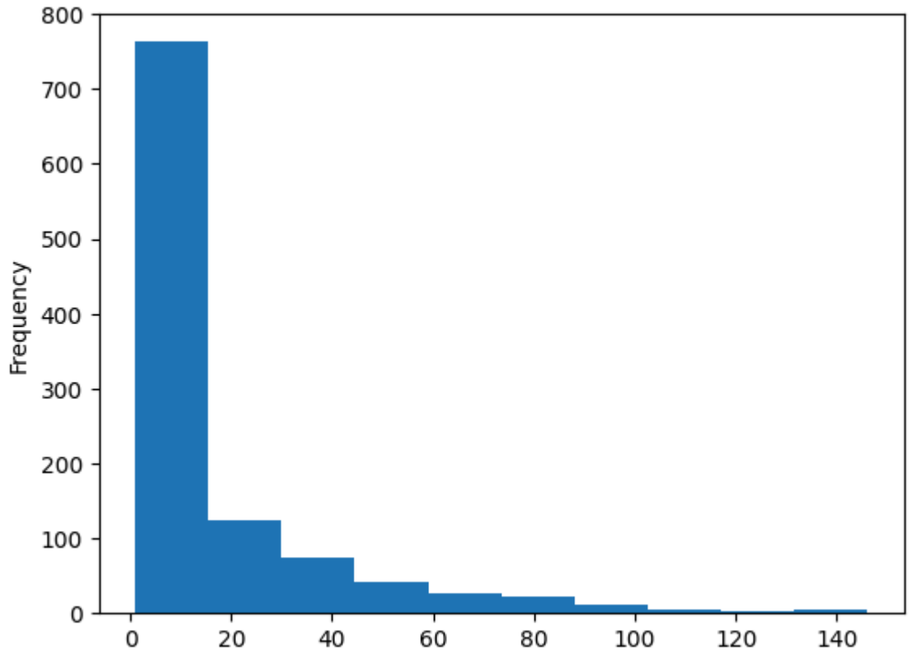
In [42]:
```
df['result'].value_counts().plot(kind='pie')  # Win by Wickets are more - which means chasing team has won more matches in comparison to defending team
```

Out[42]:
```
<Axes: ylabel='count'>
```



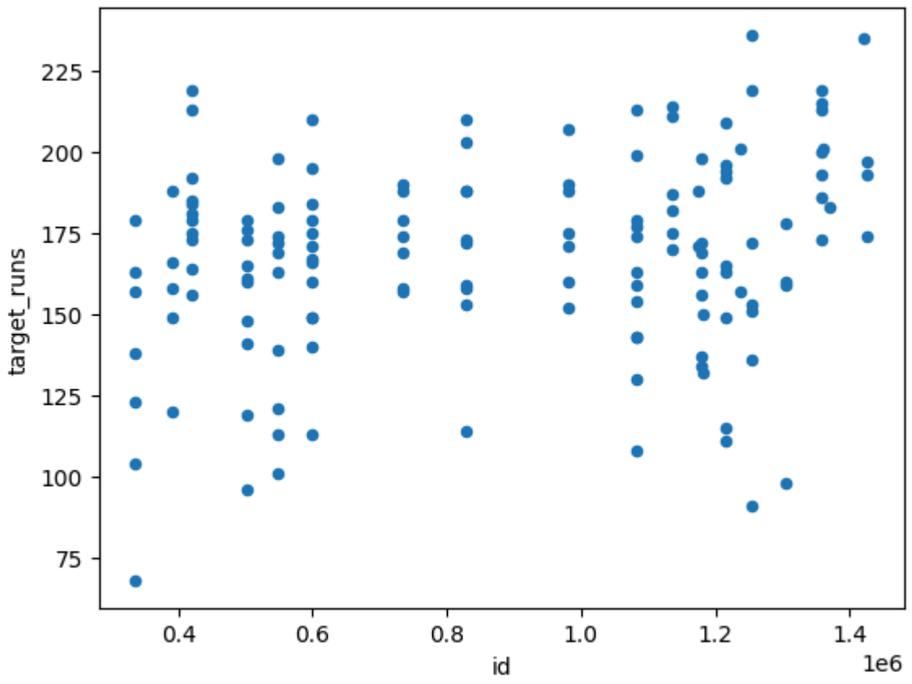
In [43]:
```
df['result_margin'].plot(kind='hist')  # Most of the matches won by runs/wickets are in range of 0-15
```

Out[43]: <Axes: ylabel='Frequency'>



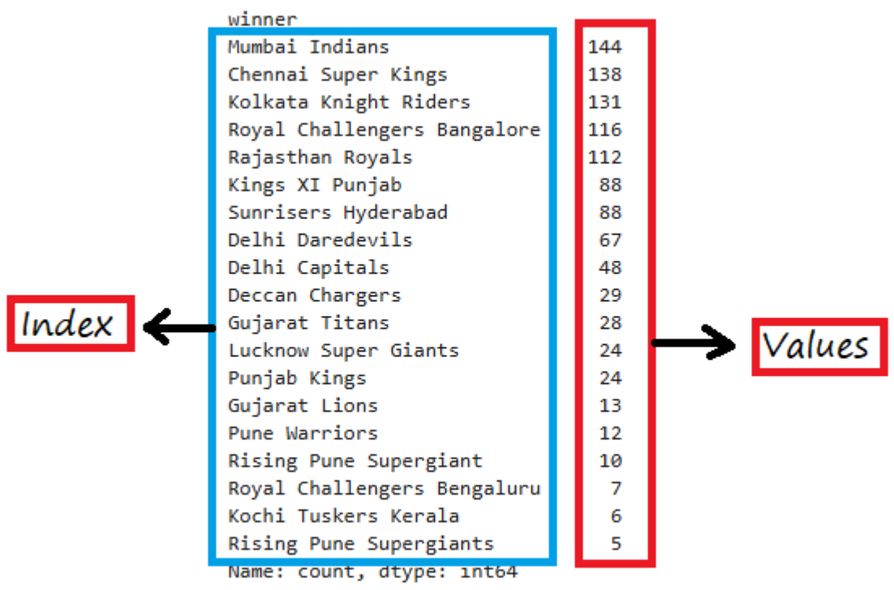
In [44]:
```
# To check if there is any trend in 'target_runs' when matches were won by Mumbai Indians, such as:
# 1. Are they winning more often when the target is high or low.
# 2. Do target runs change over time?
# 3. Any outliers (where they chased very high scores) ?
filter_df = df[df['winner'] == 'Mumbai Indians']
filter_df.plot(x='id', y='target_runs',kind='scatter') # For scatter plots you need to always provide 'x' and 'y' arguments
```

Out[44]: <Axes: xlabel='id', ylabel='target_runs'>

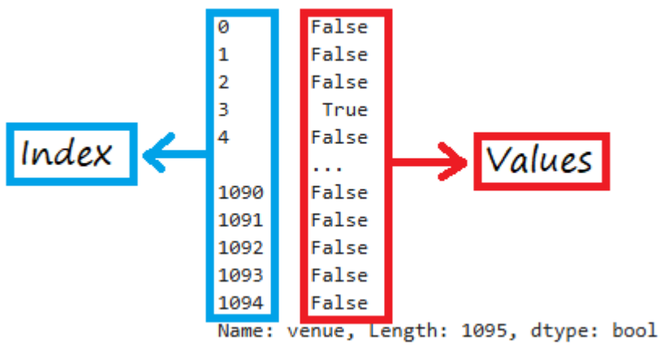


KNOWING SERIES

- Some operations like value_counts( ) and conditions returns SERIES as output.
- The elements on the left side of the SERIES are called **INDEX**
- **INDEX** must be **UNIQUE** (Cannot be repetitive)
- The elements on the right side of the SERIES are called **VALUES**



- **VALUES** can be repititive



- We also have attributes such as .index and .values that can be used on Series

In [45]:
```
winner_series = df['winner'].value_counts()
```

In [46]:
```
winner_series.index
```

Out[46]:
```
Index(['Mumbai Indians', 'Chennai Super Kings', 'Kolkata Knight Riders',
       'Royal Challengers Bangalore', 'Rajasthan Royals', 'Kings XI Punjab',
       'Sunrisers Hyderabad', 'Delhi Daredevils', 'Delhi Capitals',
       'Deccan Chargers', 'Gujarat Titans', 'Lucknow Super Giants',
       'Punjab Kings', 'Gujarat Lions', 'Pune Warriors',
       'Rising Pune Supergiant', 'Royal Challengers Bengaluru',
       'Kochi Tuskers Kerala', 'Rising Pune Supergiants'],
      dtype='object', name='winner')
```

In [47]:
```
winner_series.values
```

Out[47]:
```
array([144, 138, 131, 116, 112,  88,  88,  67,  48,  29,  28,  24,  24,
        13,  12,  10,   7,   6,   5], dtype=int64)
```

- To get corresponding value of a particular index:
  - SYNTAX : df['INDEX'] --> This will give the corresponding value

In [48]:
```
winner_series['Royal Challengers Bangalore']
```

Out[48]:
```
116
```

In [49]:
```
# We want to know the total number of matches played by each team
# But the teams are distributed in team1 and team2 columns
# So we take both the column's value_counts() and add them --> This is relevant because both the columns contain same indexes/groups
total_matches_played = df['team1'].value_counts() + df['team2'].value_counts()
```

In [50]:
```
total_matches_played
```

Out[50]:
```
Chennai Super Kings           238
Deccan Chargers                75
Delhi Capitals                 91
Delhi Daredevils              161
Gujarat Lions                  30
Gujarat Titans                 45
Kings XI Punjab               190
Kochi Tuskers Kerala           14
Kolkata Knight Riders         251
Lucknow Super Giants           44
Mumbai Indians                261
Pune Warriors                  46
Punjab Kings                   56
Rajasthan Royals              221
Rising Pune Supergiant         16
Rising Pune Supergiants        14
Royal Challengers Bangalore   240
Royal Challengers Bengaluru    15
Sunrisers Hyderabad           182
Name: count, dtype: int64
```

SORT_VALUES( )

- The sort_values( ) method is used to sort the values within a Series/DataFrame object.
- This method allows for sorting in either ascending or descending order.
- By default the data will be sorted in **ascending** order.

In [51]:
```
# Applying sort_values() on a SERIES
df['winner'].value_counts().sort_values()
```

Out[51]:
```
winner
Rising Pune Supergiants         5
Kochi Tuskers Kerala            6
Royal Challengers Bengaluru     7
Rising Pune Supergiant         10
Pune Warriors                  12
Gujarat Lions                  13
Punjab Kings                   24
Lucknow Super Giants           24
Gujarat Titans                 28
Deccan Chargers                29
Delhi Capitals                 48
Delhi Daredevils               67
Sunrisers Hyderabad            88
Kings XI Punjab                88
Rajasthan Royals              112
Royal Challengers Bangalore   116
Kolkata Knight Riders         131
Chennai Super Kings           138
Mumbai Indians                144
Name: count, dtype: int64
```

In [52]:
```
# Sorting in descending order
df['winner'].value_counts().sort_values(ascending=False)
```

Out[52]:
```
winner
Mumbai Indians                144
Chennai Super Kings           138
Kolkata Knight Riders         131
Royal Challengers Bangalore   116
Rajasthan Royals              112
Kings XI Punjab                88
Sunrisers Hyderabad            88
Delhi Daredevils               67
Delhi Capitals                 48
Deccan Chargers                29
Gujarat Titans                 28
Lucknow Super Giants           24
Punjab Kings                   24
Gujarat Lions                  13
Pune Warriors                  12
Rising Pune Supergiant         10
Royal Challengers Bengaluru     7
Kochi Tuskers Kerala            6
Rising Pune Supergiants         5
Name: count, dtype: int64
```

In [53]:
```
# Applying sort_values() on a DATAFRAME
# Entire dataframe will be sorted based on the column you have provided
df.sort_values('venue')
```

Out[53]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	
680	1136605	2018	Delhi	2018-05-12	League	AB de Villiers	Arun Jaitley Stadium	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Royal Challengers Bangalore	wickets	5.0	182.0	20.0	N	NaN	Ananthapadm
732	1178412	2019	Delhi	2019-04-20	League	SS Iyer	Arun Jaitley Stadium	Kings XI Punjab	Delhi Capitals	Delhi Capitals	field	Delhi Capitals	wickets	5.0	164.0	20.0	N	NaN	C Shan
748	1178428	2019	Delhi	2019-05-04	League	A Mishra	Arun Jaitley Stadium	Rajasthan Royals	Delhi Capitals	Rajasthan Royals	bat	Delhi Capitals	wickets	5.0	116.0	20.0	N	NaN	AY D
729	1178409	2019	Delhi	2019-04-18	League	HH Pandya	Arun Jaitley Stadium	Mumbai Indians	Delhi Capitals	Mumbai Indians	bat	Mumbai Indians	runs	40.0	169.0	20.0	N	NaN	BNJ C
711	1175371	2019	Delhi	2019-04-04	League	JM Bairstow	Arun Jaitley Stadium	Delhi Capitals	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Sunrisers Hyderabad	wickets	5.0	130.0	20.0	N	NaN	C Shan
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
857	1254099	2021	Abu Dhabi	2021-09-28	League	KA Pollard	Zayed Cricket Stadium, Abu Dhabi	Punjab Kings	Mumbai Indians	Mumbai Indians	field	Mumbai Indians	wickets	6.0	136.0	20.0	N	NaN	
861	1254089	2021	Abu Dhabi	2021-10-02	League	RD Gaikwad	Zayed Cricket Stadium, Abu Dhabi	Chennai Super Kings	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	7.0	190.0	20.0	N	NaN	CB
870	1254088	2021	Abu Dhabi	2021-10-08	League	Ishan Kishan	Zayed Cricket Stadium, Abu Dhabi	Mumbai Indians	Sunrisers Hyderabad	Mumbai Indians	bat	Mumbai Indians	runs	42.0	236.0	20.0	N	NaN	Tapar
853	1254098	2021	Abu Dhabi	2021-09-26	League	RA Jadeja	Zayed Cricket Stadium, Abu Dhabi	Kolkata Knight Riders	Chennai Super Kings	Kolkata Knight Riders	bat	Chennai Super Kings	wickets	2.0	172.0	20.0	N	NaN	CB
867	1254095	2021	Abu Dhabi	2021-10-06	League	KS Williamson	Zayed Cricket Stadium, Abu Dhabi	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Sunrisers Hyderabad	runs	4.0	142.0	20.0	N	NaN	

1095 rows × 20 columns

In [54]:
```
df.sort_values('venue',ascending=False) # Sorting in descending order
```

Out[54]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	
861	1254089	2021	Abu Dhabi	2021-10-02	League	RD Gaikwad	Zayed Cricket Stadium, Abu Dhabi	Chennai Super Kings	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	7.0	190.0	20.0	N	NaN	CB
851	1254097	2021	Abu Dhabi	2021-09-25	League	SS Iyer	Zayed Cricket Stadium, Abu Dhabi	Delhi Capitals	Rajasthan Royals	Rajasthan Royals	field	Delhi Capitals	runs	33.0	155.0	20.0	N	NaN	CB
846	1254087	2021	Abu Dhabi	2021-09-20	League	CV Varun	Zayed Cricket Stadium, Abu Dhabi	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	bat	Kolkata Knight Riders	wickets	9.0	93.0	20.0	N	NaN	CB
849	1254096	2021	Abu Dhabi	2021-09-23	League	SP Narine	Zayed Cricket Stadium, Abu Dhabi	Mumbai Indians	Kolkata Knight Riders	Kolkata Knight Riders	field	Kolkata Knight Riders	wickets	7.0	156.0	20.0	N	NaN	
870	1254088	2021	Abu Dhabi	2021-10-08	League	Ishan Kishan	Zayed Cricket Stadium, Abu Dhabi	Mumbai Indians	Sunrisers Hyderabad	Mumbai Indians	bat	Mumbai Indians	runs	42.0	236.0	20.0	N	NaN	Tapa
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
677	1136602	2018	Delhi	2018-05-10	League	S Dhawan	Arun Jaitley Stadium	Delhi Daredevils	Sunrisers Hyderabad	Delhi Daredevils	bat	Sunrisers Hyderabad	wickets	9.0	188.0	20.0	N	NaN	AY
657	1136582	2018	Delhi	2018-04-23	League	AS Rajpoot	Arun Jaitley Stadium	Kings XI Punjab	Delhi Daredevils	Delhi Daredevils	field	Kings XI Punjab	runs	4.0	144.0	20.0	N	NaN	A Nan
690	1136615	2018	Delhi	2018-05-20	League	A Mishra	Arun Jaitley Stadium	Delhi Daredevils	Mumbai Indians	Delhi Daredevils	bat	Delhi Daredevils	runs	11.0	175.0	20.0	N	NaN	HDPK Dh
748	1178428	2019	Delhi	2019-05-04	League	A Mishra	Arun Jaitley Stadium	Rajasthan Royals	Delhi Capitals	Rajasthan Royals	bat	Delhi Capitals	wickets	5.0	116.0	20.0	N	NaN	AY
680	1136605	2018	Delhi	2018-05-12	League	AB de Villiers	Arun Jaitley Stadium	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Royal Challengers Bangalore	wickets	5.0	182.0	20.0	N	NaN	Ananthapadn

1095 rows × 20 columns

In [55]:
```
# Sorting the entire dataframe based on 2 columns
df.sort_values(['team1','team2']) # Here it is sorted in ascending order for both the columns
```

Out[55]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method
25	336007	2007/08	Chennai	2008-05-06	League	AC Gilchrist	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	Deccan Chargers	Deccan Chargers	field	Deccan Chargers	wickets	7.0	145.0	20.0	N	NaN
71	392196	2009	Durban	2009-04-27	League	HH Gibbs	Kingsmead	Chennai Super Kings	Deccan Chargers	Deccan Chargers	field	Deccan Chargers	wickets	6.0	166.0	20.0	N	NaN
84	392209	2009	East London	2009-05-04	League	MS Dhoni	Buffalo Park	Chennai Super Kings	Deccan Chargers	Chennai Super Kings	bat	Chennai Super Kings	runs	78.0	179.0	20.0	N	NaN
119	419110	2009/10	Chennai	2010-03-14	League	WPUJC Vaas	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	Deccan Chargers	Deccan Chargers	bat	Deccan Chargers	runs	31.0	191.0	20.0	N	NaN
172	419163	2009/10	Mumbai	2010-04-22	Semi Final	DE Bollinger	Dr DY Patil Sports Academy	Chennai Super Kings	Deccan Chargers	Chennai Super Kings	bat	Chennai Super Kings	runs	38.0	143.0	20.0	N	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
706	1175366	2019	Hyderabad	2019-03-31	League	JM Bairstow	Rajiv Gandhi International Stadium	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Sunrisers Hyderabad	runs	118.0	232.0	20.0	N	NaN
749	1178429	2019	Bengaluru	2019-05-04	League	SO Hetmyer	M.Chinnaswamy Stadium	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Royal Challengers Bangalore	wickets	4.0	176.0	20.0	N	NaN
867	1254095	2021	Abu Dhabi	2021-10-06	League	KS Williamson	Zayed Cricket Stadium, Abu Dhabi	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Sunrisers Hyderabad	runs	4.0	142.0	20.0	N	NaN
1014	1359539	2023	Hyderabad	2023-05-18	League	V Kohli	Rajiv Gandhi International Stadium, Uppal, Hyd...	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Royal Challengers Bangalore	wickets	8.0	187.0	20.0	N	NaN
1053	1426268	2024	Bengaluru	2024-04-15	League	TM Head	M Chinnaswamy Stadium, Bengaluru	Sunrisers Hyderabad	Royal Challengers Bengaluru	Royal Challengers Bengaluru	field	Sunrisers Hyderabad	runs	25.0	288.0	20.0	N	NaN

1095 rows × 20 columns

In [56]:
```
# If you want to sort one column is ascending and the other in descending, you can pass a list as follows
df.sort_values(['team1','team2'],ascending=[True,False]) # Here team1 will be sorted in ascending and team2 in descending
```

Out[56]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	
355	598030	2013	Chennai	2013-04-25	League	MS Dhoni	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	Sunrisers Hyderabad	Sunrisers Hyderabad	bat	Chennai Super Kings	wickets	5.0	160.0	20.0	N	NaN	A
447	734029	2014	Ranchi	2014-05-22	League	DA Warner	JSCA International Stadium Complex	Chennai Super Kings	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Sunrisers Hyderabad	wickets	6.0	186.0	20.0	N	NaN	
461	829711	2015	Chennai	2015-04-11	League	BB McCullum	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	Sunrisers Hyderabad	Chennai Super Kings	bat	Chennai Super Kings	runs	45.0	210.0	20.0	N	NaN	II
655	1136580	2018	Hyderabad	2018-04-22	League	AT Rayudu	Rajiv Gandhi International Stadium	Chennai Super Kings	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Chennai Super Kings	runs	4.0	183.0	20.0	N	NaN	C
728	1178408	2019	Hyderabad	2019-04-17	League	DA Warner	Rajiv Gandhi International Stadium	Chennai Super Kings	Sunrisers Hyderabad	Chennai Super Kings	bat	Sunrisers Hyderabad	wickets	6.0	133.0	20.0	N	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
736	1178416	2019	Chennai	2019-04-23	League	SR Watson	MA Chidambaram Stadium	Sunrisers Hyderabad	Chennai Super Kings	Chennai Super Kings	field	Chennai Super Kings	wickets	6.0	176.0	20.0	N	NaN	C
769	1216516	2020/21	NaN	2020-10-02	League	PK Garg	Dubai International Cricket Stadium	Sunrisers Hyderabad	Chennai Super Kings	Sunrisers Hyderabad	bat	Sunrisers Hyderabad	runs	7.0	165.0	20.0	N	NaN	C
838	1254080	2021	Delhi	2021-04-28	League	RD Gaikwad	Arun Jaitley Stadium, Delhi	Sunrisers Hyderabad	Chennai Super Kings	Sunrisers Hyderabad	bat	Chennai Super Kings	wickets	7.0	172.0	20.0	N	NaN	Sha
859	1254091	2021	Sharjah	2021-09-30	League	JR Hazlewood	Sharjah Cricket Stadium	Sunrisers Hyderabad	Chennai Super Kings	Chennai Super Kings	field	Chennai Super Kings	wickets	6.0	135.0	20.0	N	NaN	Nit
978	1359503	2023	Chennai	2023-04-21	League	RA Jadeja	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Chennai Super Kings	Chennai Super Kings	field	Chennai Super Kings	wickets	7.0	135.0	20.0	N	NaN	H

1095 rows × 20 columns

- Also, note that all of these sorting operations are not permanent.
- To make these changes permanent, we need to use a parameter called **inplace** and by default **inplace=False**
- Set **inplace=True** to make the changes permanent.

DUPLICATED( ):

- The **duplicated( )** method is used to identify duplicate rows within a DataFrame or duplicate values within a Series. It returns a Boolean Series where True indicates a duplicate row/value and False indicates a unique one.
- We have a parameter called **'subset'** --> This parameter is OPTIONAL --> Takes a list of column names to consider when checking for duplicates.
- If this parameter is not provided then by default 'subset' = None --> all columns are considered for checking duplicates --> If any row values matches exactly only then those rows are returned.
- If for example 'subset' = ['city'] --> Then this column will be considered for checking duplicates and only unique city values will be retained.
- We also have a parameter called **'keep'** --> This parameter is OPTIONAL.
- If 'keep' = 'first' (default value) --> Then first occurences of the values are retained.
- If 'keep' = 'last' --> Then last occurences of the values are retained.

In [57]:
```
# Checking for duplicate rows in a DataFrame
df.duplicated()
```

Out[57]:
```
0       False
1       False
2       False
3       False
4       False
        ...
1090    False
1091    False
1092    False
1093    False
1094    False
Length: 1095, dtype: bool
```

In [58]:
```
# Writing the function within df will return the row values if there are any duplicates --> Considers only 'True' and returns its corresponding values
df[df.duplicated()]
# Since there are no duplicates the output is empty
```

Out[58]:

id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	umpire1	umpire2

In [59]:
```
# Checking for duplicates based on city column only
df[df.duplicated(subset=['city'])]
# The output returned here is duplicated values
```

Out[59]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method
8	335990	2007/08	Hyderabad	2008-04-24	League	YK Pathan	Rajiv Gandhi International Stadium, Uppal	Deccan Chargers	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	3.0	215.0	20.0	N	NaN
9	335991	2007/08	Chandigarh	2008-04-25	League	KC Sangakkara	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Mumbai Indians	Mumbai Indians	field	Kings XI Punjab	runs	66.0	183.0	20.0	N	NaN
10	335992	2007/08	Bangalore	2008-04-26	League	SR Watson	M Chinnaswamy Stadium	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	7.0	136.0	20.0	N	NaN
11	335993	2007/08	Chennai	2008-04-26	League	JDP Oram	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	Kolkata Knight Riders	Kolkata Knight Riders	bat	Chennai Super Kings	wickets	9.0	148.0	20.0	N	NaN
12	335994	2007/08	Mumbai	2008-04-27	League	AC Gilchrist	Dr DY Patil Sports Academy	Mumbai Indians	Deccan Chargers	Deccan Chargers	field	Deccan Chargers	wickets	10.0	155.0	20.0	N	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1090	1426307	2024	Hyderabad	2024-05-19	League	Abhishek Sharma	Rajiv Gandhi International Stadium, Uppal, Hyd...	Punjab Kings	Sunrisers Hyderabad	Punjab Kings	bat	Sunrisers Hyderabad	wickets	4.0	215.0	20.0	N	NaN
1091	1426309	2024	Ahmedabad	2024-05-21	Qualifier 1	MA Starc	Narendra Modi Stadium, Ahmedabad	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	160.0	20.0	N	NaN
1092	1426310	2024	Ahmedabad	2024-05-22	Eliminator	R Ashwin	Narendra Modi Stadium, Ahmedabad	Royal Challengers Bengaluru	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	4.0	173.0	20.0	N	NaN
1093	1426311	2024	Chennai	2024-05-24	Qualifier 2	Shahbaz Ahmed	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Rajasthan Royals	Rajasthan Royals	field	Sunrisers Hyderabad	runs	36.0	176.0	20.0	N	NaN
1094	1426312	2024	Chennai	2024-05-26	Final	MA Starc	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	114.0	20.0	N	NaN

1058 rows × 20 columns

In [60]:
```
df[df.duplicated(subset=['city'],keep='last')]  # Last occurences of the city is retained
```

Out[60]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	NaN
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1082	1426297	2024	Ahmedabad	2024-05-10	League	Shubman Gill	Narendra Modi Stadium, Ahmedabad	Gujarat Titans	Chennai Super Kings	Chennai Super Kings	field	Gujarat Titans	runs	35.0	232.0	20.0	N	NaN
1084	1426299	2024	Chennai	2024-05-12	League	Simarjeet Singh	MA Chidambaram Stadium, Chepauk, Chennai	Rajasthan Royals	Chennai Super Kings	Rajasthan Royals	bat	Chennai Super Kings	wickets	5.0	142.0	20.0	N	NaN
1085	1426300	2024	Bengaluru	2024-05-12	League	C Green	M Chinnaswamy Stadium, Bengaluru	Royal Challengers Bengaluru	Delhi Capitals	Delhi Capitals	field	Royal Challengers Bengaluru	runs	47.0	188.0	20.0	N	NaN
1091	1426309	2024	Ahmedabad	2024-05-21	Qualifier 1	MA Starc	Narendra Modi Stadium, Ahmedabad	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	160.0	20.0	N	NaN
1093	1426311	2024	Chennai	2024-05-24	Qualifier 2	Shahbaz Ahmed	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Rajasthan Royals	Rajasthan Royals	field	Sunrisers Hyderabad	runs	36.0	176.0	20.0	N	NaN

1058 rows × 20 columns

DROP_DUPLICATES( ):

- Using the duplicated( ) method we found out the duplicate values.
- Now to drop these duplicate values we can use drop_duplicates( ).
- This method also takes parameters such as **'subset'** and **'keep'** whose functionality is the same as duplicated( )
- Also, dropping of duplicates is not a permanent operation --> To make it permanent you can use **'inplace'** parameter
- If 'inplace' = False (default) --> Drop operation is not permanent
- If 'inplace' = True --> Drop operation is permanent

In [61]:
```
df.drop_duplicates()
# All the columns are considered and only those rows are dropped that matches EXACTLY
# Since no rows in our dataset matches exactly, no rows are dropped.
```

Out[61]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	NaN
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1090	1426307	2024	Hyderabad	2024-05-19	League	Abhishek Sharma	Rajiv Gandhi International Stadium, Uppal, Hyd...	Punjab Kings	Sunrisers Hyderabad	Punjab Kings	bat	Sunrisers Hyderabad	wickets	4.0	215.0	20.0	N	NaN
1091	1426309	2024	Ahmedabad	2024-05-21	Qualifier 1	MA Starc	Narendra Modi Stadium, Ahmedabad	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	160.0	20.0	N	NaN
1092	1426310	2024	Ahmedabad	2024-05-22	Eliminator	R Ashwin	Narendra Modi Stadium, Ahmedabad	Royal Challengers Bengaluru	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	4.0	173.0	20.0	N	NaN
1093	1426311	2024	Chennai	2024-05-24	Qualifier 2	Shahbaz Ahmed	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Rajasthan Royals	Rajasthan Royals	field	Sunrisers Hyderabad	runs	36.0	176.0	20.0	N	NaN
1094	1426312	2024	Chennai	2024-05-26	Final	MA Starc	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	114.0	20.0	N	NaN

1095 rows × 20 columns

In [62]:
```
df.drop_duplicates(subset=['city'],keep='last')
# Only the last occurence of the city names are kept and recurring ones are dropped
```

Out[62]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	metho
70	392195	2009	Cape Town	2009-04-26	League	KC Sangakkara	Newlands	Kings XI Punjab	Rajasthan Royals	Kings XI Punjab	bat	Kings XI Punjab	runs	27.0	140.0	20.0	N	Na
90	392215	2009	East London	2009-05-08	League	A Nehra	Buffalo Park	Delhi Daredevils	Mumbai Indians	Mumbai Indians	bat	Delhi Daredevils	wickets	7.0	117.0	20.0	N	Na
95	392220	2009	Kimberley	2009-05-11	League	DR Smith	De Beers Diamond Oval	Deccan Chargers	Rajasthan Royals	Deccan Chargers	bat	Deccan Chargers	runs	53.0	167.0	20.0	N	Na
102	392227	2009	Port Elizabeth	2009-05-16	League	ML Hayden	St George's Park	Chennai Super Kings	Mumbai Indians	Mumbai Indians	bat	Chennai Super Kings	wickets	7.0	148.0	20.0	N	Na
105	392230	2009	Bloemfontein	2009-05-17	League	AB de Villiers	OUTsurance Oval	Delhi Daredevils	Rajasthan Royals	Delhi Daredevils	bat	Delhi Daredevils	runs	14.0	151.0	20.0	N	Na
109	392234	2009	Durban	2009-05-20	League	M Muralitharan	Kingsmead	Chennai Super Kings	Kings XI Punjab	Chennai Super Kings	bat	Chennai Super Kings	runs	24.0	117.0	20.0	N	Na
112	392237	2009	Centurion	2009-05-22	Semi Final	AC Gilchrist	SuperSport Park	Delhi Daredevils	Deccan Chargers	Deccan Chargers	field	Deccan Chargers	wickets	6.0	154.0	20.0	N	Na
114	392239	2009	Johannesburg	2009-05-24	Final	A Kumble	New Wanderers Stadium	Royal Challengers Bangalore	Deccan Chargers	Royal Challengers Bangalore	field	Deccan Chargers	runs	6.0	144.0	20.0	N	Na
160	419151	2009/10	Nagpur	2010-04-12	League	Harmeet Singh	Vidarbha Cricket Association Stadium, Jamtha	Deccan Chargers	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Deccan Chargers	runs	13.0	152.0	20.0	N	Na
218	501242	2011	Kochi	2011-05-05	League	BJ Hodge	Nehru Stadium	Kochi Tuskers Kerala	Kolkata Knight Riders	Kolkata Knight Riders	field	Kochi Tuskers Kerala	runs	17.0	157.0	20.0	N	Na
437	734009	2014	Cuttack	2014-05-14	League	RV Uthappa	Barabati Stadium	Kolkata Knight Riders	Mumbai Indians	Kolkata Knight Riders	field	Kolkata Knight Riders	wickets	6.0	142.0	20.0	N	Na
515	829821	2015	Ranchi	2015-05-22	Qualifier 2	A Nehra	JSCA International Stadium Complex	Chennai Super Kings	Royal Challengers Bangalore	Chennai Super Kings	field	Chennai Super Kings	wickets	3.0	140.0	20.0	N	Na
572	981011	2016	Raipur	2016-05-22	League	V Kohli	Shaheed Veer Narayan Singh International Stadium	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Royal Challengers Bangalore	wickets	6.0	139.0	20.0	N	Na
610	1082625	2017	Rajkot	2017-04-29	League	KH Pandya	Saurashtra Cricket Association Stadium	Gujarat Lions	Mumbai Indians	Gujarat Lions	bat	Mumbai Indians	tie	NaN	154.0	20.0	Y	Na
628	1082643	2017	Kanpur	2017-05-13	League	Mohammed Siraj	Green Park	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Sunrisers Hyderabad	wickets	8.0	155.0	20.0	N	Na
634	1082649	2017	Bangalore	2017-05-19	Qualifier 2	KV Sharma	M Chinnaswamy Stadium	Mumbai Indians	Kolkata Knight Riders	Mumbai Indians	field	Mumbai Indians	wickets	6.0	108.0	20.0	N	Na
683	1136608	2018	Indore	2018-05-14	League	UT Yadav	Holkar Cricket Stadium	Kings XI Punjab	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Royal Challengers Bangalore	wickets	10.0	89.0	20.0	N	Na
815	1237181	2020/21	NaN	2020-11-10	Final	TA Boult	Dubai International Cricket Stadium	Delhi Capitals	Mumbai Indians	Delhi Capitals	bat	Mumbai Indians	wickets	5.0	157.0	20.0	N	Na
870	1254088	2021	Abu Dhabi	2021-10-08	League	Ishan Kishan	Zayed Cricket Stadium, Abu Dhabi	Mumbai Indians	Sunrisers Hyderabad	Mumbai Indians	bat	Mumbai Indians	runs	42.0	236.0	20.0	N	Na
874	1254116	2021	Sharjah	2021-10-13	Qualifier 2	VR Iyer	Sharjah Cricket Stadium	Delhi Capitals	Kolkata Knight Riders	Kolkata Knight Riders	field	Kolkata Knight Riders	wickets	3.0	136.0	20.0	N	Na
875	1254117	2021	Dubai	2021-10-15	Final	F du Plessis	Dubai International Cricket Stadium	Chennai Super Kings	Kolkata Knight Riders	Kolkata Knight Riders	field	Chennai Super Kings	runs	27.0	193.0	20.0	N	Na
936	1304107	2022	Pune	2022-05-14	League	AD Russell	Maharashtra Cricket Association Stadium, Pune	Kolkata Knight Riders	Sunrisers Hyderabad	Kolkata Knight Riders	bat	Kolkata Knight Riders	runs	54.0	178.0	20.0	N	Na
941	1304112	2022	Navi Mumbai	2022-05-18	League	Q de Kock	Dr DY Patil Sports Academy, Mumbai	Lucknow Super Giants	Kolkata Knight Riders	Lucknow Super Giants	bat	Lucknow Super Giants	runs	2.0	211.0	20.0	N	Na
995	1359520	2023	Chandigarh	2023-05-03	League	Ishan Kishan	Punjab Cricket Association IS Bindra Stadium, ...	Punjab Kings	Mumbai Indians	Mumbai Indians	field	Mumbai Indians	wickets	6.0	215.0	20.0	N	Na
1039	1422134	2024	Visakhapatnam	2024-04-03	League	SP Narine	Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket St...	Kolkata Knight Riders	Delhi Capitals	Kolkata Knight Riders	bat	Kolkata Knight Riders	runs	106.0	273.0	20.0	N	Na
1060	1426275	2024	Mohali	2024-04-21	League	R Sai Kishore	Maharaja Yadavindra Singh International Cricke...	Punjab Kings	Gujarat Titans	Punjab Kings	bat	Gujarat Titans	wickets	3.0	143.0	20.0	N	Na
1061	1426276	2024	Jaipur	2024-04-22	League	Sandeep Sharma	Sawai Mansingh Stadium, Jaipur	Mumbai Indians	Rajasthan Royals	Mumbai Indians	bat	Rajasthan Royals	wickets	9.0	180.0	20.0	N	Na
1077	1426292	2024	Lucknow	2024-05-05	League	SP Narine	Bharat Ratna Shri Atal Bihari Vajpayee Ekana C...	Kolkata Knight Riders	Lucknow Super Giants	Lucknow Super Giants	field	Kolkata Knight Riders	runs	98.0	236.0	20.0	N	Na
1081	1426296	2024	Dharamsala	2024-05-09	League	V Kohli	Himachal Pradesh Cricket Association Stadium, ...	Royal Challengers Bengaluru	Punjab Kings	Punjab Kings	field	Royal Challengers Bengaluru	runs	60.0	242.0	20.0	N	Na

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	metho
1083	1426298	2024	Kolkata	2024-05-11	League	CV Varun	Eden Gardens, Kolkata	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	Kolkata Knight Riders	runs	18.0	158.0	16.0	N	Na
1086	1426302	2024	Delhi	2024-05-14	League	I Sharma	Arun Jaitley Stadium, Delhi	Delhi Capitals	Lucknow Super Giants	Lucknow Super Giants	field	Delhi Capitals	runs	19.0	209.0	20.0	N	Na
1087	1426303	2024	Guwahati	2024-05-15	League	SM Curran	Barsapara Cricket Stadium, Guwahati	Rajasthan Royals	Punjab Kings	Rajasthan Royals	bat	Punjab Kings	wickets	5.0	145.0	20.0	N	Na
1088	1426305	2024	Mumbai	2024-05-17	League	N Pooran	Wankhede Stadium, Mumbai	Lucknow Super Giants	Mumbai Indians	Mumbai Indians	field	Lucknow Super Giants	runs	18.0	215.0	20.0	N	Na
1089	1426306	2024	Bengaluru	2024-05-18	League	F du Plessis	M Chinnaswamy Stadium, Bengaluru	Royal Challengers Bengaluru	Chennai Super Kings	Chennai Super Kings	field	Royal Challengers Bengaluru	runs	27.0	219.0	20.0	N	Na
1090	1426307	2024	Hyderabad	2024-05-19	League	Abhishek Sharma	Rajiv Gandhi International Stadium, Uppal, Hyd...	Punjab Kings	Sunrisers Hyderabad	Punjab Kings	bat	Sunrisers Hyderabad	wickets	4.0	215.0	20.0	N	Na
1092	1426310	2024	Ahmedabad	2024-05-22	Eliminator	R Ashwin	Narendra Modi Stadium, Ahmedabad	Royal Challengers Bengaluru	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	4.0	173.0	20.0	N	Na
1094	1426312	2024	Chennai	2024-05-26	Final	MA Starc	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	114.0	20.0	N	Na

In [63]:
```
# Here in 'subset' parameter we have provided a list of 2 column names
# This operation means --> CITY AND SEASON (BOTH) combination must be Unique
# Any row that has same city and season value will be removed
# Since 'keep' parameter is not specified --> First occurence is retained and other duplicate values are dropped
df.drop_duplicates(['city','season'])
```

Out[63]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	metho
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	Nal
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	Nal
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	Nal
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	Nal
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	Nal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1036	1422131	2024	Visakhapatnam	2024-03-31	League	KK Ahmed	Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket St...	Delhi Capitals	Chennai Super Kings	Delhi Capitals	bat	Delhi Capitals	runs	20.0	192.0	20.0	N	Nal
1037	1422132	2024	Mumbai	2024-04-01	League	TA Boult	Wankhede Stadium, Mumbai	Mumbai Indians	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals	wickets	6.0	126.0	20.0	N	Nal
1058	1426273	2024	Delhi	2024-04-20	League	TM Head	Arun Jaitley Stadium, Delhi	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	Sunrisers Hyderabad	runs	67.0	267.0	20.0	N	Nal
1076	1426291	2024	Dharamsala	2024-05-05	League	RA Jadeja	Himachal Pradesh Cricket Association Stadium, ...	Chennai Super Kings	Punjab Kings	Punjab Kings	field	Chennai Super Kings	runs	28.0	168.0	20.0	N	Nal
1087	1426303	2024	Guwahati	2024-05-15	League	SM Curran	Barsapara Cricket Stadium, Guwahati	Rajasthan Royals	Punjab Kings	Rajasthan Royals	bat	Punjab Kings	wickets	5.0	145.0	20.0	N	Nal

165 rows × 20 columns

FIND THE SEASON WINNERS FOR EACH SEASON

- There are 2 ways to do this:
  - Using drop_duplicates( )
  - Using condition

In [64]:
```
# Based on season we are dropping the duplicates
# For each season the last match is always a final --> So keeping the last occurence and dropping other values
df.drop_duplicates('season',keep='last')[['season','winner']]
```

Out[64]:

	season	winner
57	2007/08	Rajasthan Royals
114	2009	Deccan Chargers
174	2009/10	Chennai Super Kings
247	2011	Chennai Super Kings
321	2012	Kolkata Knight Riders
397	2013	Mumbai Indians
457	2014	Kolkata Knight Riders
516	2015	Mumbai Indians
576	2016	Sunrisers Hyderabad
635	2017	Mumbai Indians
695	2018	Chennai Super Kings
755	2019	Mumbai Indians
815	2020/21	Mumbai Indians
875	2021	Chennai Super Kings
949	2022	Gujarat Titans
1023	2023	Chennai Super Kings
1094	2024	Kolkata Knight Riders

In [65]:
```
# First we are applying condition that the match must be 'Final' and then selecting the columns we need
df[df['match_type'] == 'Final'][['season', 'winner']]
```

Out[65]:

	season	winner
57	2007/08	Rajasthan Royals
114	2009	Deccan Chargers
174	2009/10	Chennai Super Kings
247	2011	Chennai Super Kings
321	2012	Kolkata Knight Riders
397	2013	Mumbai Indians
457	2014	Kolkata Knight Riders
516	2015	Mumbai Indians
576	2016	Sunrisers Hyderabad
635	2017	Mumbai Indians
695	2018	Chennai Super Kings
755	2019	Mumbai Indians
815	2020/21	Mumbai Indians
875	2021	Chennai Super Kings
949	2022	Gujarat Titans
1023	2023	Chennai Super Kings
1094	2024	Kolkata Knight Riders

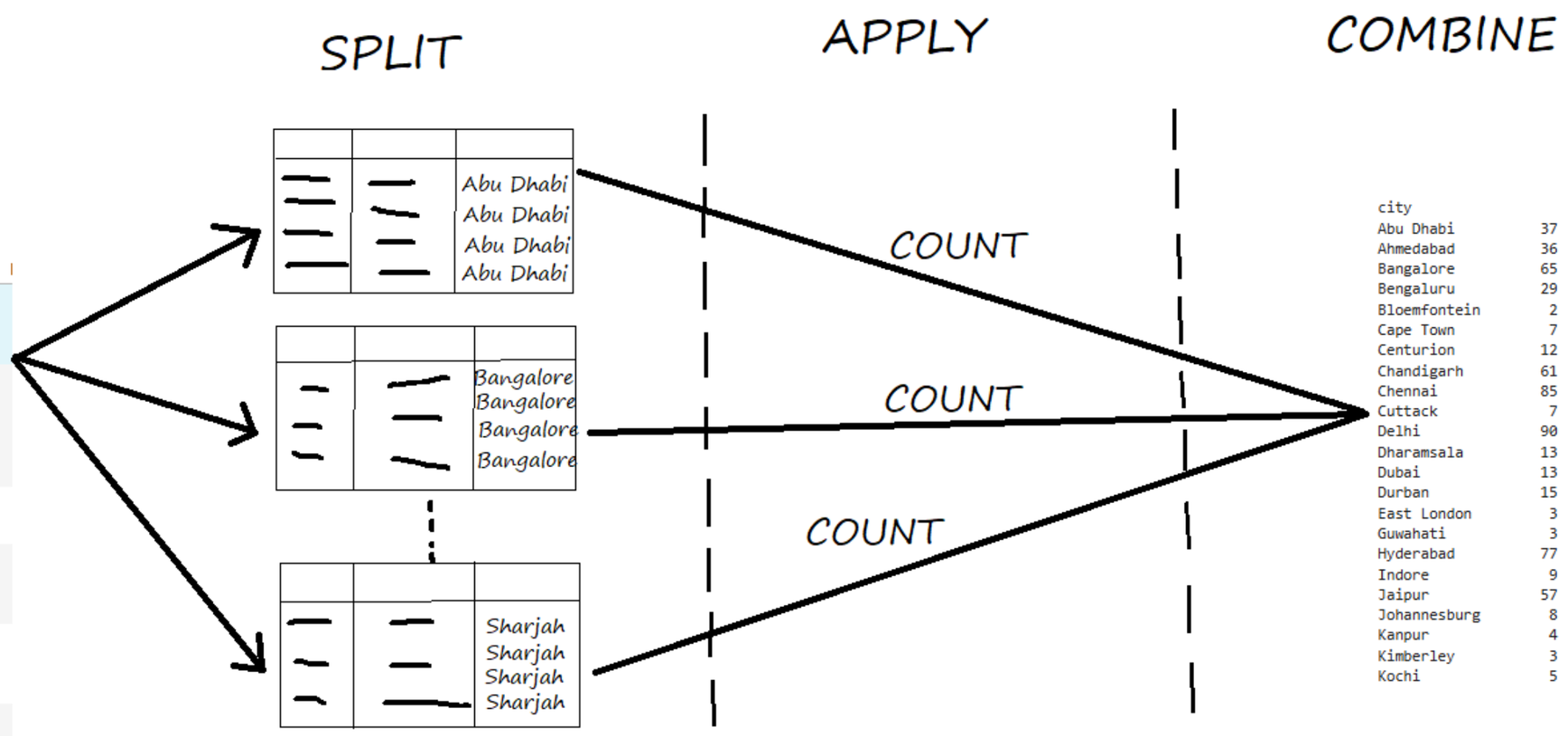
GROUPBY( ):

- The groupby( ) function is used to **group data** based on a single column or more.
- Basic Syntax: groupby('column_name')
- It usually returns a DataFrameGroupBy object.
- To access each of these groups we use LOOPS/get_group('group_name') method
- The groupby() works on the concept of SPLIT-APPLY-COMBINE

In [66]:
```
# Group the dataset on the basis of 'City'
# Choosing only the 'city' column to be displayed in the output
# Applying count() to calculate the number of matches played in each city
# The output is sorted alphabetically, to sort it numerically we can use sort_values(ascending=False)
grp_city = df.groupby('city')['city']
grp_city.count()
```

Out[66]:
```
city
Abu Dhabi          37
Ahmedabad          36
Bangalore          65
Bengaluru          29
Bloemfontein        2
Cape Town           7
Centurion          12
Chandigarh         61
Chennai            85
Cuttack             7
Delhi              90
Dharamsala         13
Dubai              13
Durban             15
East London         3
Guwahati            3
Hyderabad          77
Indore              9
Jaipur             57
Johannesburg        8
Kanpur              4
Kimberley           3
Kochi               5
Kolkata            93
Lucknow            14
Mohali              5
Mumbai            173
Nagpur              3
Navi Mumbai         9
Port Elizabeth      7
Pune               51
Raipur              6
Rajkot             10
Ranchi              7
Sharjah            10
Visakhapatnam      15
Name: city, dtype: int64
```

	id	season	city	date	match_type
0	335982	2007/08	Bangalore	2008-04-18	League
1	335983	2007/08	Chandigarh	2008-04-19	League
2	335984	2007/08	Delhi	2008-04-19	League
3	335985	2007/08	Mumbai	2008-04-20	League
4	335986	2007/08	Kolkata	2008-04-20	League
...	...	...	...	...	...



city	
Abu Dhabi	37
Ahmedabad	36
Bangalore	65
Bengaluru	29
Bloemfontein	2
Cape Town	7
Centurion	12
Chandigarh	61
Chennai	85
Cuttack	7
Delhi	90
Dharamsala	13
Dubai	13
Durban	15
East London	3
Guwahati	3
Hyderabad	77
Indore	9
Jaipur	57
Johannesburg	8
Kanpur	4
Kimberley	3
Kochi	5

In [67]:
```
grouped_data = df.groupby('toss_decision')
```

In [68]:
```
grouped_data
```

Out[68]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000027EDBBE1D90>
```

In [69]:
```
for key, grp_df in grouped_data:
    print(key,grp_df)
```

```
bat            id   season        city        date   match_type  player_of_match  \
1      335983  2007/08  Chandigarh  2008-04-19       League       MEK Hussey
2      335984  2007/08       Delhi  2008-04-19       League      MF Maharoof
3      335985  2007/08      Mumbai  2008-04-20       League       MV Boucher
4      335986  2007/08     Kolkata  2008-04-20       League        DJ Hussey
5      335987  2007/08      Jaipur  2008-04-21       League        SR Watson
...       ...      ...         ...         ...          ...              ...
1084  1426299     2024     Chennai  2024-05-12       League  Simarjeet Singh
1087  1426303     2024    Guwahati  2024-05-15       League        SM Curran
1090  1426307     2024   Hyderabad  2024-05-19       League  Abhishek Sharma
1091  1426309     2024   Ahmedabad  2024-05-21  Qualifier 1         MA Starc
1094  1426312     2024     Chennai  2024-05-26        Final         MA Starc

                                          venue  \
1            Punjab Cricket Association Stadium, Mohali
2                                    Feroz Shah Kotla
3                                    Wankhede Stadium
4                                        Eden Gardens
5                              Sawai Mansingh Stadium
...                                               ...
1084           MA Chidambaram Stadium, Chepauk, Chennai
1087                Barsapara Cricket Stadium, Guwahati
1090  Rajiv Gandhi International Stadium, Uppal, Hyd...
1091                  Narendra Modi Stadium, Ahmedabad
1094           MA Chidambaram Stadium, Chepauk, Chennai

                     team1                       team2          toss_winner  \
1           Kings XI Punjab          Chennai Super Kings  Chennai Super Kings
2          Delhi Daredevils             Rajasthan Royals     Rajasthan Royals
3            Mumbai Indians  Royal Challengers Bangalore       Mumbai Indians
4     Kolkata Knight Riders              Deccan Chargers      Deccan Chargers
5          Rajasthan Royals              Kings XI Punjab      Kings XI Punjab
...                     ...                          ...                  ...
1084       Rajasthan Royals          Chennai Super Kings     Rajasthan Royals
1087       Rajasthan Royals                 Punjab Kings     Rajasthan Royals
1090           Punjab Kings          Sunrisers Hyderabad         Punjab Kings
1091    Sunrisers Hyderabad        Kolkata Knight Riders  Sunrisers Hyderabad
1094    Sunrisers Hyderabad        Kolkata Knight Riders  Sunrisers Hyderabad

     toss_decision                      winner   result  result_margin  \
1              bat          Chennai Super Kings     runs           33.0
2              bat             Delhi Daredevils  wickets            9.0
3              bat  Royal Challengers Bangalore  wickets            5.0
4              bat        Kolkata Knight Riders  wickets            5.0
5              bat             Rajasthan Royals  wickets            6.0
...            ...                          ...      ...            ...
1084           bat          Chennai Super Kings  wickets            5.0
1087           bat                Punjab Kings  wickets            5.0
1090           bat          Sunrisers Hyderabad  wickets            4.0
1091           bat        Kolkata Knight Riders  wickets            8.0
1094           bat        Kolkata Knight Riders  wickets            8.0

      target_runs  target_overs super_over method        umpire1  \
1           241.0          20.0          N    NaN      MR Benson
2           130.0          20.0          N    NaN      Aleem Dar
3           166.0          20.0          N    NaN       SJ Davis
4           111.0          20.0          N    NaN      BF Bowden
5           167.0          20.0          N    NaN      Aleem Dar
...           ...           ...        ...    ...            ...
1084        142.0          20.0          N    NaN       R Pandit
1087        145.0          20.0          N    NaN       R Pandit
1090        215.0          20.0          N    NaN    Nitin Menon
1091        160.0          20.0          N    NaN   AK Chaudhary
1094        114.0          20.0          N    NaN  J Madanagopal

                  umpire2
1              SL Shastri
2           GA Pratapkumar
3                DJ Harper
4              K Hariharan
5                RB Tiffin
...                    ...
1084             YC Barde
1087  MV Saidharshan Kumar
1090             VK Sharma
1091              R Pandit
1094           Nitin Menon

[391 rows x 20 columns]
field            id   season        city        date   match_type player_of_match  \
0      335982  2007/08   Bangalore  2008-04-18       League     BB McCullum
7      335989  2007/08     Chennai  2008-04-23       League       ML Hayden
8      335990  2007/08   Hyderabad  2008-04-24       League       YK Pathan
9      335991  2007/08  Chandigarh  2008-04-25       League   KC Sangakkara
10     335992  2007/08   Bangalore  2008-04-26       League       SR Watson
...       ...      ...         ...         ...          ...             ...
1086  1426302     2024       Delhi  2024-05-14       League        I Sharma
1088  1426305     2024      Mumbai  2024-05-17       League        N Pooran
1089  1426306     2024   Bengaluru  2024-05-18       League    F du Plessis
1092  1426310     2024   Ahmedabad  2024-05-22   Eliminator        R Ashwin
1093  1426311     2024     Chennai  2024-05-24  Qualifier 2   Shahbaz Ahmed

                                      venue                       team1  \
0                     M Chinnaswamy Stadium  Royal Challengers Bangalore
7             MA Chidambaram Stadium, Chepauk          Chennai Super Kings
8      Rajiv Gandhi International Stadium, Uppal              Deccan Chargers
9     Punjab Cricket Association Stadium, Mohali              Kings XI Punjab
10                    M Chinnaswamy Stadium  Royal Challengers Bangalore
...                                     ...                         ...
1086            Arun Jaitley Stadium, Delhi               Delhi Capitals
1088              Wankhede Stadium, Mumbai         Lucknow Super Giants
1089         M Chinnaswamy Stadium, Bengaluru  Royal Challengers Bengaluru
1092         Narendra Modi Stadium, Ahmedabad  Royal Challengers Bengaluru
1093   MA Chidambaram Stadium, Chepauk, Chennai          Sunrisers Hyderabad

                    team2                 toss_winner toss_decision  \
0     Kolkata Knight Riders  Royal Challengers Bangalore         field
7           Mumbai Indians               Mumbai Indians         field
8        Rajasthan Royals             Rajasthan Royals         field
9           Mumbai Indians               Mumbai Indians         field
10       Rajasthan Royals             Rajasthan Royals         field
...                   ...                          ...           ...
1086  Lucknow Super Giants         Lucknow Super Giants         field
1088        Mumbai Indians               Mumbai Indians         field
1089   Chennai Super Kings          Chennai Super Kings         field
1092      Rajasthan Royals             Rajasthan Royals         field
1093      Rajasthan Royals             Rajasthan Royals         field

                         winner   result  result_margin  target_runs  \
0          Kolkata Knight Riders     runs          140.0        223.0
7            Chennai Super Kings     runs            6.0        209.0
8               Rajasthan Royals  wickets            3.0        215.0
9                Kings XI Punjab     runs           66.0        183.0
10              Rajasthan Royals  wickets            7.0        136.0
...                          ...      ...            ...          ...
1086              Delhi Capitals     runs           19.0        209.0
1088        Lucknow Super Giants     runs           18.0        215.0
1089  Royal Challengers Bengaluru     runs           27.0        219.0
1092            Rajasthan Royals  wickets            4.0        173.0
```

```
1093          Sunrisers Hyderabad     runs           36.0        176.0

      target_overs super_over method                umpire1  \
0             20.0          N    NaN              Asad Rauf
7             20.0          N    NaN              DJ Harper
8             20.0          N    NaN              Asad Rauf
9             20.0          N    NaN              Aleem Dar
10            20.0          N    NaN              MR Benson
...            ...        ...    ...                    ...
1086          20.0          N    NaN               A Totre
1088          20.0          N    NaN          Navdeep Singh
1089          20.0          N    NaN               A Totre
1092          20.0          N    NaN  KN Ananthapadmanabhan
1093          20.0          N    NaN            Nitin Menon

                    umpire2
0              RE Koertzen
7            GA Pratapkumar
8                MR Benson
9                AM Saheba
10               IL Howell
...                    ...
1086           Vinod Seshan
1088              R Pandit
1089  KN Ananthapadmanabhan
1092   MV Saidharshan Kumar
1093             VK Sharma

[704 rows x 20 columns]
```

DATAFRAME

team1	team2	toss_winner	toss_decision	winner
hallengers Bangalore	Kolkata Knight Riders	al Challengers Bangalore	field	Kolkata Knight Riders
Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings
Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils
Mumbai Indians	al Challengers Bangalore	Mumbai Indians	bat	al Challengers Bangalore
Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders
Rajasthan Royals	Kings XI Punjab	Kings XI Punjab	bat	Rajasthan Royals
Deccan Chargers	Delhi Daredevils	Deccan Chargers	bat	Delhi Daredevils
Chennai Super Kings	Mumbai Indians	Mumbai Indians	field	Chennai Super Kings
Deccan Chargers	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals
Kings XI Punjab	Mumbai Indians	Mumbai Indians	field	Kings XI Punjab
hallengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	Rajasthan Royals

GROUP 1 --> BAT

toss_decision	team1	team2	toss_winner
bat	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings
bat	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals
bat	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians
bat	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers
bat	Rajasthan Royals	Kings XI Punjab	Kings XI Punjab
...	...	...	...
bat	Rajasthan Royals	Chennai Super Kings	Rajasthan Royals
bat	Rajasthan Royals	Punjab Kings	Rajasthan Royals
bat	Punjab Kings	Sunrisers Hyderabad	Punjab Kings
bat	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad
bat	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad

GROUP 2 --> FIELD

toss_decision	team1	team2	toss_winner
field	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore
field	Chennai Super Kings	Mumbai Indians	Mumbai Indians
field	Deccan Chargers	Rajasthan Royals	Rajasthan Royals
field	Kings XI Punjab	Mumbai Indians	Mumbai Indians
field	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals
...	...	...	...
field	Delhi Capitals	Lucknow Super Giants	Lucknow Super Giants
field	Lucknow Super Giants	Mumbai Indians	Mumbai Indians
field	Royal Challengers Bengaluru	Chennai Super Kings	Chennai Super Kings
field	Royal Challengers Bengaluru	Rajasthan Royals	Rajasthan Royals
field	Sunrisers Hyderabad	Rajasthan Royals	Rajasthan Royals

KEY — GRP_DF

In [70]:
```
# When len() method is applied on the DataFrameGroupBy object --> it returns the number of groups created
len(grouped_data)
```

Out[70]: 2

In [71]:
```
# The size() method returns the groups created and count of elements present/are a part of that particular group
# It is sorted alphabetically
# To sort it numerically you can use .sort_values(ascending=False)
grouped_data.size()
```

Out[71]:
```
toss_decision
bat      391
field    704
dtype: int64
```

In [72]:
```
# The first() method returns the first occuring row/element of each of the groups
# Basically like --> the first row which belonged to 'bat' and 'field' groups will be shown
grouped_data.first()
```

Out[72]:

toss_decision	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	winner	result	result_margin	target_runs	target_overs	super_over	method	umpire1	umpi
bat	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	Chennai Super Kings	runs	33.0	241.0	20.0	N	D/L	MR Benson	Sha
field	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	D/L	Asad Rauf	Koertz

In [73]:
```
# The last() method returns the last occuring row/element of each of the groups
grouped_data.last()
```

Out[73]:

toss_decision	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	winner	result	result_margin	target_runs	target_overs	super_over	method	umpire1
bat	1426312	2024	Chennai	2024-05-26	Final	MA Starc	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	Kolkata Knight Riders	wickets	8.0	114.0	20.0	N	D/L	J Madanagopal
field	1426311	2024	Chennai	2024-05-24	Qualifier 2	Shahbaz Ahmed	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Rajasthan Royals	Rajasthan Royals	Sunrisers Hyderabad	runs	36.0	176.0	20.0	N	D/L	Nitin Menon

In [74]:
```
# The get_group('group_name') retrieves a new DataFrame containing only the rows belonging to a particular group.
# Here a DataFrame is returned containing rows only where toss_decision = 'bat'
grouped_data.get_group('bat')
```

Out[74]:

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	NaN
5	335987	2007/08	Jaipur	2008-04-21	League	SR Watson	Sawai Mansingh Stadium	Rajasthan Royals	Kings XI Punjab	Kings XI Punjab	bat	Rajasthan Royals	wickets	6.0	167.0	20.0	N	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1084	1426299	2024	Chennai	2024-05-12	League	Simarjeet Singh	MA Chidambaram Stadium, Chepauk, Chennai	Rajasthan Royals	Chennai Super Kings	Rajasthan Royals	bat	Chennai Super Kings	wickets	5.0	142.0	20.0	N	NaN
1087	1426303	2024	Guwahati	2024-05-15	League	SM Curran	Barsapara Cricket Stadium, Guwahati	Rajasthan Royals	Punjab Kings	Rajasthan Royals	bat	Punjab Kings	wickets	5.0	145.0	20.0	N	NaN
1090	1426307	2024	Hyderabad	2024-05-19	League	Abhishek Sharma	Rajiv Gandhi International Stadium, Uppal, Hyd...	Punjab Kings	Sunrisers Hyderabad	Punjab Kings	bat	Sunrisers Hyderabad	wickets	4.0	215.0	20.0	N	NaN
1091	1426309	2024	Ahmedabad	2024-05-21	Qualifier 1	MA Starc	Narendra Modi Stadium, Ahmedabad	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	160.0	20.0	N	NaN
1094	1426312	2024	Chennai	2024-05-26	Final	MA Starc	MA Chidambaram Stadium, Chepauk, Chennai	Sunrisers Hyderabad	Kolkata Knight Riders	Sunrisers Hyderabad	bat	Kolkata Knight Riders	wickets	8.0	114.0	20.0	N	NaN

391 rows × 20 columns

BRAINSTORM EXERCISE 1: FIND OUT TOP 5 BATSMAN TO SCORE MOST NUMBER OF **RUNS** IN THE IPL

- We will be using the 'deliveries.csv' dataset.
- In this dataset, each row represents a ball played by a batsman, along with additional information like:
  - 'match_id' which is unique for every match
  - 'inning' --> when inning = 1, first half of match and when inning = 2, second half of match
  - The team he belongs to (batting_team)
  - The team against whom he is playing (bowling_team)
  - The current over (over)
  - Which ball of an over (ball)
  - The batsman name ('batter')
  - The bowler name ('bowler')
  - 'non_striker' --> Batsman not facing the current delivery --> The one standing opposite to the batsman facing delivery
  - 'batsman_runs' --> Runs scored by the batsman
  - 'extra_runs' --> In case of wides/No ball/Byes/legbyes --> extra runs scored are specified, else it is marked as 0
  - 'total_runs' --> batsman_runs + extra_runs (if any)
  - 'extras_type' --> Defines whether it is a legbyes/ wide/ byes/ no ball
  - 'is_wicket' --> 0 when no wicket and 1 when wicket is taken
  - 'player_dismissed' --> Name of the player whose wicket was taken/ if no wicket taken then NaN
  - 'dismissal_kind' --> How the wicket was taken --> bowled/caught/stump out. If no wicket was taken then NaN
  - 'fielder' --> The name of the fielder who caught/stump out the batsman

In [75]:
```
delivery = pd.read_csv('deliveries.csv')
```

In [76]:
```
delivery.head()
```

Out[76]:

	match_id	inning	batting_team	bowling_team	over	ball	batter	bowler	non_striker	batsman_runs	extra_runs	total_runs	extras_type	is_wicket	player_dismissed	dismissal_kind	fielder
0	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	1	SC Ganguly	P Kumar	BB McCullum	0	1	1	legbyes	0	NaN	NaN	NaN
1	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	2	BB McCullum	P Kumar	SC Ganguly	0	0	0	NaN	0	NaN	NaN	NaN
2	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	3	BB McCullum	P Kumar	SC Ganguly	0	1	1	wides	0	NaN	NaN	NaN
3	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	4	BB McCullum	P Kumar	SC Ganguly	0	0	0	NaN	0	NaN	NaN	NaN
4	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	5	BB McCullum	P Kumar	SC Ganguly	0	0	0	NaN	0	NaN	NaN	NaN

In [77]:
```
delivery.shape
```

Out[77]:
```
(260920, 17)
```

In [78]:
```
# First do a group by on the basis of batters, so each group will have one batsman name
batter_group = delivery.groupby('batter')
# Now for each batters group go to 'batsman_runs' and apply sum() to sum up all the runs he has scored till date
# Then to show top scores we sort the values and keep ascending = False, meaning the most top scorer will appear first
# To see top 5 we use the head() function
batter_group['batsman_runs'].sum().sort_values(ascending=False).head()
```

Out[78]:
```
batter
V Kohli      8014
S Dhawan     6769
RG Sharma    6630
DA Warner    6567
SK Raina     5536
Name: batsman_runs, dtype: int64
```

BRAINSTORM EXERCISE 2: FIND OUT TOP 5 BATSMAN TO SCORE MOST NUMBER OF **FOURS** IN THE IPL

In [79]:
```
# Filtering out dataset, so that it contains only the balls on which a 4 was hit
# four_runs --> Dataset in which the 'batsman_runs' is 4 only
four_runs = delivery[delivery['batsman_runs'] == 4]
```

In [80]:
```
# Now we do a group by on the basis of batter where each group contains one batsman name
four_group = four_runs.groupby('batter')
# In this group go to the 'batsman_runs' column and apply count() to count the number of fours he has hit
```

```
# Then sorting the values to display top number first and applying head to see top 5
four_group['batsman_runs'].count().sort_values(ascending=False).head()
```

Out[80]:
```
batter
S Dhawan      768
V Kohli       708
DA Warner     663
RG Sharma     599
SK Raina      506
Name: batsman_runs, dtype: int64
```

BRAINSTORM EXERCISE 3: GIVEN A BATSMAN NAME RETURN THE TOP 3 TEAMS AGAINST WHICH HE HAS SCORED MAXIMUM NUMBER OF RUNS

In [81]:
```
# We are trying to find out top 3 teams against whom, Virat Kohli has scored maximum numbers of runs in IPL
# vk --> Dataset that contains rows only where the batter is Virat Kohli
vk = delivery[delivery['batter'] == 'V Kohli']
```

In [82]:
```
vk
```

Out[82]:

	match_id	inning	batting_team	bowling_team	over	ball	batter	bowler	non_striker	batsman_runs	extra_runs	total_runs	extras_type	is_wicket	player_dismissed	dismissal_kind	fielder
132	335982	2	Royal Challengers Bangalore	Kolkata Knight Riders	1	2	V Kohli	I Sharma	W Jaffer	0	0	0	NaN	0	NaN	NaN	NaN
133	335982	2	Royal Challengers Bangalore	Kolkata Knight Riders	1	3	V Kohli	I Sharma	W Jaffer	0	4	4	legbyes	0	NaN	NaN	NaN
134	335982	2	Royal Challengers Bangalore	Kolkata Knight Riders	1	4	V Kohli	I Sharma	W Jaffer	1	0	1	NaN	0	NaN	NaN	NaN
137	335982	2	Royal Challengers Bangalore	Kolkata Knight Riders	2	1	V Kohli	AB Dinda	W Jaffer	0	0	0	NaN	0	NaN	NaN	NaN
138	335982	2	Royal Challengers Bangalore	Kolkata Knight Riders	2	2	V Kohli	AB Dinda	W Jaffer	0	0	0	NaN	1	V Kohli	bowled	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
260280	1426310	1	Royal Challengers Bengaluru	Rajasthan Royals	6	1	V Kohli	R Ashwin	C Green	1	0	1	NaN	0	NaN	NaN	NaN
260282	1426310	1	Royal Challengers Bengaluru	Rajasthan Royals	6	3	V Kohli	R Ashwin	C Green	1	0	1	NaN	0	NaN	NaN	NaN
260286	1426310	1	Royal Challengers Bengaluru	Rajasthan Royals	6	7	V Kohli	R Ashwin	C Green	1	0	1	NaN	0	NaN	NaN	NaN
260287	1426310	1	Royal Challengers Bengaluru	Rajasthan Royals	7	1	V Kohli	YS Chahal	C Green	0	0	0	NaN	0	NaN	NaN	NaN
260288	1426310	1	Royal Challengers Bengaluru	Rajasthan Royals	7	2	V Kohli	YS Chahal	C Green	0	0	0	NaN	1	V Kohli	caught	D Ferreira

6236 rows × 17 columns

In [83]:
```
# First we apply a group by on 'bowling_team' which is basically the opposite team/ team against whom virat kohli is playing
# Each group will consist of a team name like CSK, KKR...
# For each of these groups go to the 'batsman_runs' column and apply sum() to calculate number of runs scored by Virat against that team
# Sort the values to show the highest first
vk_max_runs = vk.groupby('bowling_team')['batsman_runs'].sum().sort_values(ascending=False)
```

In [84]:
```
# Apply head() to see top 3
vk_max_runs.head(3)

# From this we infer that against CSK, Virat has scored 1053 runs, against KKR he has scored 962 runs and so on...
```

Out[84]:
```
bowling_team
Chennai Super Kings      1053
Kolkata Knight Riders     962
Mumbai Indians            860
Name: batsman_runs, dtype: int64
```

HANDLING MISSING VALUES IN A DATASET

- Since it is not logical to fill the missing values here in the deliveries.csv, we will consider 'titanic.csv' dataset temporarily, only to show how to handle missing values.
- There are 4 ways to handle the missing values:
  - dropna( ) --> Drop the missing values
  - fillna( ) --> Fill the missing values with some new values
  - interpolate( ) --> Fill the missing values by estimating them based on existing data points.
  - replace( ) --> replaces the specified value with another specified value.

In [85]:
```
# The load_dataset() function loads an example dataset from the online repository
# This function provides quick access to small number of example datasets that are useful for learning purpose
import seaborn as sns
titanic = sns.load_dataset('titanic')
```

In [86]:
```
titanic.head() # We see that there are missing values represented by Nan (Not a Number)
```

Out[86]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [87]:
```
# STEP 1: Identifying how many missing values we have
titanic.isnull().sum() # Returns the number of missing values in each column
```

Out[87]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [88]:
```
titanic.shape # 891 ROWS and 15 COLUMNS
```

Out[88]:
```
(891, 15)
```

In [89]:
```
# METHOD 1: Using dropna() to drop the missing values
# Here we have a parameter called 'how' and you can provide 2 values to it --> 'any' and 'all'
```

```
# When how='any' (default) is provided then the entire row is dropped even if there is one missing value
titanic.dropna(how='any')
```

Out[89]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False	G	Southampton	yes	False
11	1	1	female	58.0	0	0	26.5500	S	First	woman	False	C	Southampton	yes	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
871	1	1	female	47.0	1	1	52.5542	S	First	woman	False	D	Southampton	yes	False
872	0	1	male	33.0	0	0	5.0000	S	First	man	True	B	Southampton	no	True
879	1	1	female	56.0	0	1	83.1583	C	First	woman	False	C	Cherbourg	yes	False
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True

182 rows × 15 columns

In [90]:
```
# When how='all' then the entire row is dropped only if all the values in the row are NaN
titanic.dropna(how='all')
```

Out[90]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

In [91]:
```
# We also have a parameter called 'axis' which takes 2 values: 0 for ROW (default) and 1 for COLUMN
# In the previous code we did not specify 'axis' parameter so it was 0 by default and would drop ROWS on basis of how the 'how' parameter is defined
# Now let's sepcify 'axis' parameter and keep it to 1 which means COLUMNS
# The below code means that a column will be dropped even if there is one missing value (NaN)
titanic.dropna(how='any',axis=1) # 4 columns are dropped because they had missing values
```

Out[91]:

	survived	pclass	sex	sibsp	parch	fare	class	who	adult_male	alive	alone
0	0	3	male	1	0	7.2500	Third	man	True	no	False
1	1	1	female	1	0	71.2833	First	woman	False	yes	False
2	1	3	female	0	0	7.9250	Third	woman	False	yes	True
3	1	1	female	1	0	53.1000	First	woman	False	yes	False
4	0	3	male	0	0	8.0500	Third	man	True	no	True
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	0	0	13.0000	Second	man	True	no	True
887	1	1	female	0	0	30.0000	First	woman	False	yes	True
888	0	3	female	1	2	23.4500	Third	woman	False	no	False
889	1	1	male	0	0	30.0000	First	man	True	yes	True
890	0	3	male	0	0	7.7500	Third	man	True	no	True

891 rows × 11 columns

In [92]:
```
# A column will only be dropped if all the values in the column are missing
titanic.dropna(how='all',axis=1) # No columns are dropped, because there are few missing values, not all missing values
```

Out[92]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

In [93]:
```
# We have another parameter called 'subset' in which you provide your 'column_names' --> The dropna() will then only work considering those columns
# By default subset=none
# Note that 'subset' parameter works only when 'axis' = 0
# The below code will remove only those rows where age is missing or NaN
titanic.dropna(how='any',axis=0,subset=['age'])
```

Out[93]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False	NaN	Queenstown	no	False
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

714 rows × 15 columns

In [94]:
```
# The 'subset' parameter can take one/multiple column_names
# But when you provide multiple column names, how does the dropna() work here? Does it want either or both of them to have NaN in order to drop the rows
# This is dependent on the 'how' parameter
# When how='any' then a row is dropped if either of these columns have NaN
# Basically when how='any' --> OR operation
titanic.dropna(how='any', subset=['age','deck']) # If either age or deck has NaN then row is dropped
```

Out[94]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False	G	Southampton	yes	False
11	1	1	female	58.0	0	0	26.5500	S	First	woman	False	C	Southampton	yes	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
871	1	1	female	47.0	1	1	52.5542	S	First	woman	False	D	Southampton	yes	False
872	0	1	male	33.0	0	0	5.0000	S	First	man	True	B	Southampton	no	True
879	1	1	female	56.0	0	1	83.1583	C	First	woman	False	C	Cherbourg	yes	False
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True

184 rows × 15 columns

In [95]:
```
# When how='all' then all the columns provided in subset must have NaN to drop the row
# Basically when how='all' --> AND operation
titanic.dropna(how='all', subset=['age','deck']) # Only if age and deck have NaN then row is dropped
```

Out[95]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False	NaN	Queenstown	no	False
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

733 rows × 15 columns

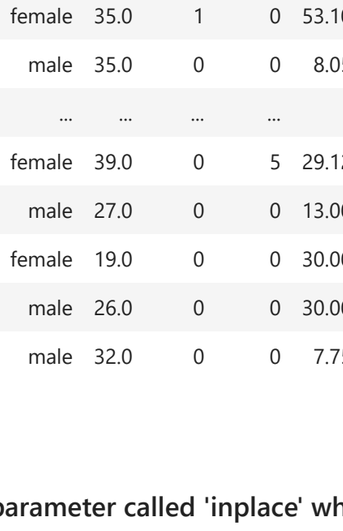
We also have one parameter called 'inplace' which is False by default

To make the changes permanent, set inplace=True

The thresh parameter in dropna() specifies the minimum number of non-missing values required for a row or column to be kept. If a row or column has fewer non-missing values than the thresh value, it will be dropped.

```
# thresh = 1 --> atleast 1 non-null value must be present in order to keep the row
df.dropna(thresh=1)
```

	value	temp
0	1.0	NaN
1	2.0	32.0
2	3.0	31.0
3	NaN	NaN
4	NaN	29.0
5	NaN	30.0
6	7.0	NaN
7	8.0	29.0
8	9.0	30.0



	value	temp
0	1.0	NaN
1	2.0	32.0
2	3.0	31.0
4	NaN	29.0
5	NaN	30.0
6	7.0	NaN
7	8.0	29.0
8	9.0	30.0

```
# # thresh = 2 --> atleast 2 non-null value must be present in order to keep the row
df.dropna(thresh=2)
```

	value	temp
1	2.0	32.0
2	3.0	31.0
7	8.0	29.0
8	9.0	30.0

FILLNA( ):

- dataframe.fillna(value) --> Replaces missing values with a specified value in the entire dataframe

In [96]:
```
# Replacing NaN values in a particular column with a specified value
titanic['age'].fillna(25) # The NaN values in the age column will be replaced with 25
```

Out[96]:
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    25.0
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

In [97]:
```
# You can also replace the missing values with the mean of that column
titanic['age'].fillna(titanic['age'].mean()) # NaN values in age column is replaced with 29 which is the mean of this column
```

Out[97]:
```
0      22.000000
1      38.000000
2      26.000000
3      35.000000
4      35.000000
         ...
886    27.000000
887    19.000000
888    29.699118
889    26.000000
890    32.000000
Name: age, Length: 891, dtype: float64
```

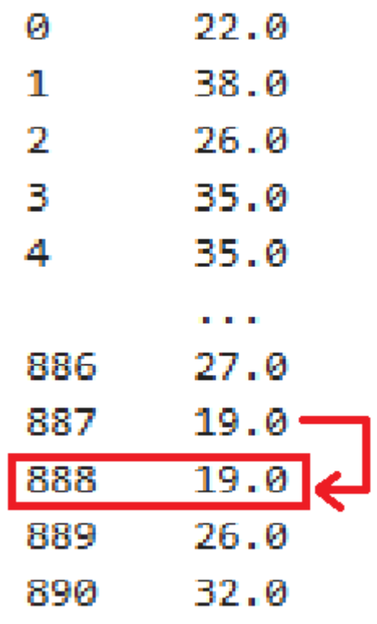
In [98]:
```
# # You can also replace the missing values with the median of that column
titanic['age'].fillna(titanic['age'].median()) # # NaN values in age column is replaced with 28 which is the median of this column
```

Out[98]:
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    28.0
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

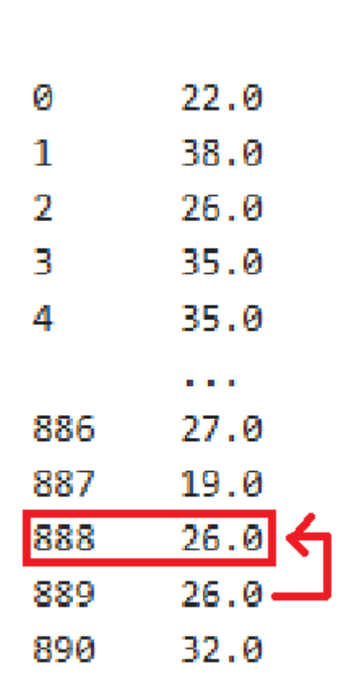
In [99]:
```
# You can also replace the missing values with the mode of that column
# Since titanic['age'].mode() returns a series, we use indexing [0] to access the first element which is our mode
titanic['age'].fillna(titanic['age'].mode()[0]) # # NaN values in age column is replaced with 24 which is the mode of this column
```

Out[99]:
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    24.0
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

- The fillna() also takes the 'method' parameter
- The method parameter can take 2 values either 'ffill' or 'bfill'
- When method='ffill' (short form for FORWARD FILL) the value previous to NaN is selected and replaced in place of NaN



- Here the row highlited in red had NaN value and when 'ffill' was applied the previous value (19) was selected and replaced in place of NaN
- When method='bfill' (short form for BACKWARD FILL) the value ahead of NaN is selected and replaced in place of NaN



- Here the row highlited in red had NaN value and when 'bfill' was applied the forward value (26) was selected and replaced in place of NaN

In [100…
```
titanic['age'].fillna(method='ffill')
```

```
C:\Users\Pooja\AppData\Local\Temp\ipykernel_1900\1768166572.py:1: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  titanic['age'].fillna(method='ffill')
```

Out[100…
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    19.0
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

In [101…
```
# Since the 'method' parameter is deprecated we have a function called ffill() that should be used
titanic['age'].ffill()
```

Out[101…
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    19.0
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

In [102…
```
titanic['age'].fillna(method='bfill')
```

```
C:\Users\Pooja\AppData\Local\Temp\ipykernel_1900\3844541350.py:1: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  titanic['age'].fillna(method='bfill')
```

Out[102…
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    26.0
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

In [103…
```
# Since the 'method' parameter is deprecated we have a function called bfill() that should be used
titanic['age'].bfill()
```

Out[103…
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    26.0
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

INTERPOLATE()

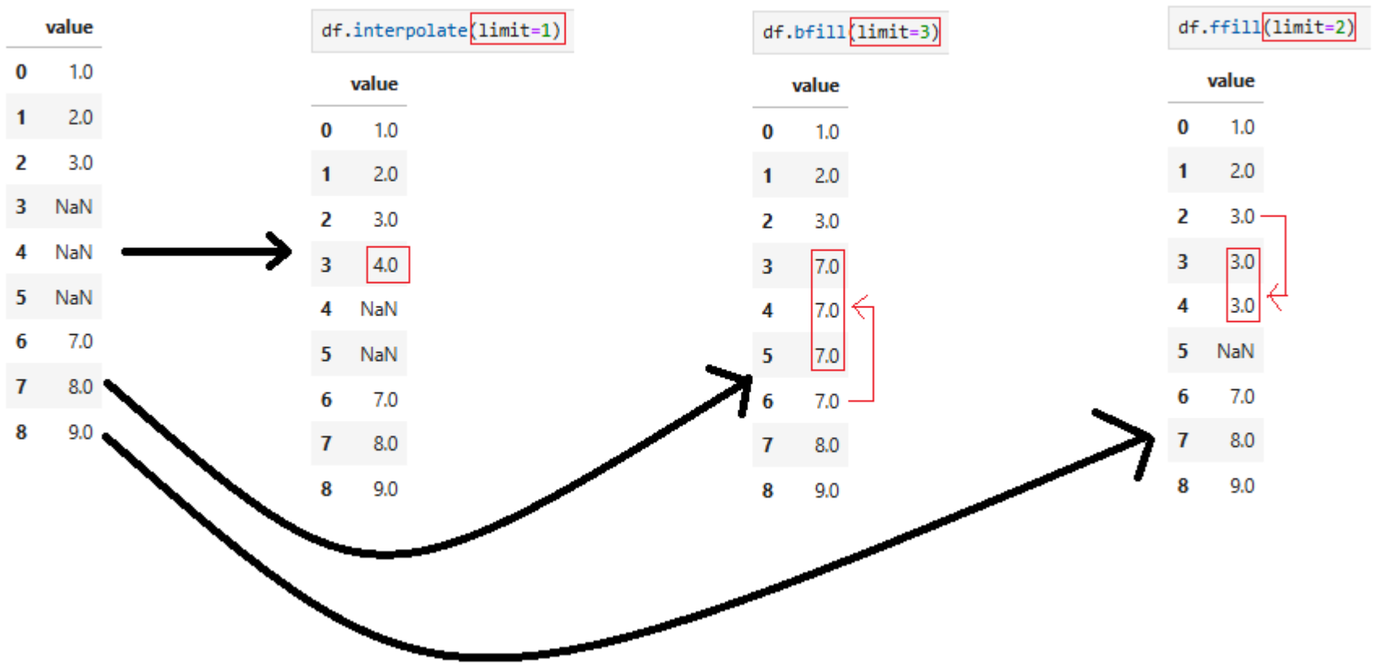
- The interpolate() method is used to fill missing values by estimating them based on existing data points. This is a common technique for handling missing data during the data cleaning process.
- It primarily works with numerical data. For categorical data, other methods like fillna() are typically used.

In [104…
```
# To make the operation permanent use inplace=True
# By default the interpolate() function is applied along rows, so axis = 0. If you want to apply across columns, use axis = 1
titanic['age'].interpolate()
```

Out[104…
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    22.5
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

THE LIMIT PARAMETER:

- The **limit** parameter is primarily used with ffill(), bfill() and interpolate().
- Purpose: Controlling the extent of fillingg- : When using ffil, or bfil or interpolate()ll) to replace er) values, the limit parameter specifies the maximum number of consecutive NaN values that should be fill
- Crucial for preventing the over-imputation of missing data, especially in cases where a long sequence of NaN values are present. ed.



REPLACE( ) FUNCTION:

- In Raw dataset, sometimes if there are missing values they are either kept blank or denoted by special character like '?' or number like -999
- We want to replace these special characters/ numbers with NaN so that we can easily use other data handling methods to fill missing values
- **SYNTAX: df.replace(old_value, new_value)**

REPLACING SINGLE VALUE

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

```
df = pd.DataFrame({
    'temp':[26,'?',32,34],
    'windspeed':[80,70,'?',65]
})
df
```

	temp	windspeed
0	26	80
1	?	70
2	32	?
3	34	65

```
df.replace('?',np.nan)
# If you execute df to see the dataframe we will see that '?' values still persist
# To make this operation of replacement permanent, use inplace=True
```

	temp	windspeed
0	26.0	80.0
1	NaN	70.0
2	32.0	NaN
3	34.0	65.0

REPLACING A LIST OF VALUES

	temp	windspeed
0	26	80
1	?	70
2	32	?
3	34	-999

```
# We had 2 special values '?' and -999 which are invalid
# We provide a LIST of these invalid values to the replace() function to be replaced with NaN values
df.replace(['?',-999],np.nan)
```

	temp	windspeed
0	26.0	80.0
1	NaN	70.0
2	32.0	NaN
3	34.0	NaN

REPLACING EACH MISSING VALUE WITH A DIFFERENT VALUE

	temp	windspeed
0	26	80
1	?	70
2	32	?
3	34	65

```
# Now assume that you found out the values of missing 'temp' and 'windspeed'
# If you use df.replace('?',30) then '?' will be replaced with 30 in both Columns, which is not what we want
# We want '?' to be placed with 30 in 'temp' column and '?' to be placed with 72 in 'windspeed' column
# We use a nested Dictionary
df.replace({
    'temp':{'?':30},
    'windspeed':{'?':72}
})
```

	temp	windspeed
0	26	80
1	30	70
2	32	72
3	34	65

REPLACING LIST OF VALUES WITH ANOTHER LIST OF VALUES AND DICTIONARY MAPPING

```
df = pd.DataFrame({
    'temp':[26,'?',32,34],
    'windspeed':[80,70,-999,65],
    'event':['Cloudy','no event','no event','Sunny']
})
df
```

	temp	windspeed	event
0	26	80	Cloudy
1	?	70	no event
2	32	-999	no event
3	34	65	Sunny

```
# Replacing List of Values with another list of values
# ? is replaced with 30, -999 is replaced with 72 and 'no event' is replaced with 'Windy'
df.replace(['?',-999,'no event'],[30,72,'Windy'])
```

	temp	windspeed	event
0	26	80	Cloudy
1	30	70	Windy
2	32	72	Windy
3	34	65	Sunny

REPLACING LIST OF VALUES WITH ANOTHER LIST OF VALUES

```
# Replacing values using DICTIONARY MAPPING
# ? is replaced with 30, -999 is replaced with 72 and 'no event' is replaced with 'Windy'
df.replace({'?':30, -999:72, 'no event':'Windy'})
```

	temp	windspeed	event
0	26	80	Cloudy
1	30	70	Windy
2	32	72	Windy
3	34	65	Sunny

REPLACING VALUES USING DICTIONARY MAPPING

REPLACING VALUES USING REGEX

	temp	windspeed
0	26	80
1	Not Available	70
2	32	Not Available
3	34	65

```
df.replace({r'[A-Za-z]+'},29.5,regex=True)
```

	temp	windspeed
0	26.0	80.0
1	29.5	70.0
2	32.0	29.5
3	34.0	65.0

REPLACING 'NOT AVAILABLE' IN ALL COLUMNS WITH 29.5

```
df.replace({'temp':{r'[A-Za-z]+':29.5},
            'windspeed':{r'[A-Za-z]+':72}},regex=True)
```

	temp	windspeed
0	26.0	80
1	29.5	70
2	32.0	72
3	34.0	65

REPLACING 'NOT AVAILABLE' IN TEMP --> 29.5 AND WINDSPEED --> 72

FORMATTING THE DATA:

- The **parse_dates** parameter, is used with read_csv(), is a crucial argument for converting columns containing date and time information into proper datetime objects during data loading.
- In the below example we see that the date column is actually of type string, which is not what we need.

```
df = pd.read_csv('weather.csv')
df
```

	date	temp
0	01/04/2025	32
1	02/04/2025	33
2	03/04/2025	31

```
type(df['date'][0])
```

```
str
```

- We then provide **parse_dates** parameter to read_csv() and provide the column name which you want to convert to proper datetime object

```
df = pd.read_csv('weather.csv', parse_dates=['date'])
df
```

	date	temp
0	2025-01-04	32
1	2025-02-04	33
2	2025-03-04	31

```
type(df['date'][0])
```

```
pandas._libs.tslibs.timestamps.Timestamp
```

SET_INDEX() AND RESET_INDEX():

- When you do dataframe.head(), you usually see on the extreme left that we have numbers starting from 0 for each row. This is Pandas self generated index to uniquely identify each row. However if you don't want this self-generated index and you have a column that can act as index, uniquely identifying the rows, then we can use set_index( )

SYNTAX: df.set_index('column_name')

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers

Pandas self generated Index

- If you feel that Noo..! The pandas generated one was better and you want to revert this operation, then you can use reset_index( ).

SYNTAX: df.reset_index()

In [105…

```
df.head()
```

Out[105…

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	umpi
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	NaN	A R
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN	Ben
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN	Ale
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN	SJ D
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	NaN	Bow

In [106…
```
df.loc[353] # Here 353 is row index using which I can access the corresponding ROW (This index is pandas self generated one)
```

Out[106…
```
id                          598059
season                        2013
city                         Delhi
date                    2013-04-23
match_type                  League
player_of_match      Harmeet Singh
venue              Feroz Shah Kotla
team1              Delhi Daredevils
team2               Kings XI Punjab
toss_winner         Kings XI Punjab
toss_decision                 field
winner              Kings XI Punjab
result                        wickets
result_margin                     5.0
target_runs                     121.0
target_overs                     20.0
super_over                          N
method                            NaN
umpire1                 VA Kulkarni
umpire2                   K Srinath
Name: 353, dtype: object
```

In [107…
```
# Since 'id' column is unique for each row we can make the 'id' column as index
# This operation is not permanent, to make it permanent we can use the inplace parameter and set it to True
df.set_index('id',inplace=True)
```

In [108…
```
df.head() # We see that now the 'id' column acts as index
```

Out[108…

id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	umpire1
335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	NaN	Asac Rau
335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN	MR Benson
335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN	Aleem Dar
335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN	SJ Davis
335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	NaN	BF Bowden

In [109…
```
# You can use loc and pass an id then it will return a correspondig row
# This works when you have make permanent changes by setting inplace=True
# If inplace=False and changes are not permanent then the following code raises an error
df.loc[1426312]
```

Out[109…
```
season                                       2024
city                                      Chennai
date                                   2024-05-26
match_type                                  Final
player_of_match                        MA Starc
venue              MA Chidambaram Stadium, Chepauk, Chennai
team1                          Sunrisers Hyderabad
team2                        Kolkata Knight Riders
toss_winner                    Sunrisers Hyderabad
toss_decision                                  bat
winner                       Kolkata Knight Riders
result                                     wickets
result_margin                                  8.0
target_runs                                  114.0
target_overs                                  20.0
super_over                                       N
method                                         NaN
umpire1                              J Madanagopal
umpire2                                Nitin Menon
Name: 1426312, dtype: object
```

In [110…
```
# Now I don't want the 'id' column to be the index and want the same old pandas self-generated index
# This is also not permanenet operation, to make it permanent you can set inplace=True
df.reset_index(inplace=True)
```

In [111…
```
df.head() # We see that now the 'id' column is no more the index
```

Out[111...

	id	season	city	date	match_type	player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	target_overs	super_over	method	umpi
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	M Chinnaswamy Stadium	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders	runs	140.0	223.0	20.0	N	NaN	A R
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Punjab Cricket Association Stadium, Mohali	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings	runs	33.0	241.0	20.0	N	NaN	Ben
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Shah Kotla	Delhi Daredevils	Rajasthan Royals	Rajasthan Royals	bat	Delhi Daredevils	wickets	9.0	130.0	20.0	N	NaN	Ale
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhede Stadium	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore	wickets	5.0	166.0	20.0	N	NaN	SJ D
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Eden Gardens	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders	wickets	5.0	111.0	20.0	N	NaN	Bow

COMBINING DATAFRAMES

- Sometimes you will have data from multiple sources, which will result in different dataframes/datasets.
- Now, you want to combine all of these dataframes into one for convinience.
- There are 2 ways using which you can combine your data:
  - CONCAT
  - MERGE

CONCAT

- The purpose of concat() is to **combine or stack** the data horizontally (along ROWS) or vertically (along COLUMNS), adding new data at the bottom or side-by-side, without combining KEYS.
- The concat() function is usually used, when 2 or more Dataframes have same columns and you just want to combine additional data.
- SYNTAX: pd.concat([df1,df2....])
- The dataframes you want to combine is to be provided within [ ] seperated by commas

In [112...
```
df1 = pd.DataFrame({"A": [1, 2], "B": [3, 4], "C": [5, 6]})
df2 = pd.DataFrame({"A": [5, 6], "B": [7, 8]})
```

In [113...
```
df1
```

Out[113...

	A	B	C
0	1	3	5
1	2	4	6

In [114...
```
df2
```

Out[114...

	A	B
0	5	7
1	6	8

In [115...
```
pd.concat([df1,df2]) # By default axis=0, so data is combined along ROWS
```

Out[115...

	A	B	C
0	1	3	5.0
1	2	4	6.0
0	5	7	NaN
1	6	8	NaN

In [116...
```
# We see that in the output, the index is not continous, It is still using it's old index values
# To keep the index continous we use a parameter called 'ignore_index' and set it to True
pd.concat([df1,df2],ignore_index=True)
```

Out[116...

	A	B	C
0	1	3	5.0
1	2	4	6.0
2	5	7	NaN
3	6	8	NaN

In [117...
```
# If you want to combine the data along columns/Vertically, you can set axis = 1
# This will combine data side-by-side, but will not combine the KEYS (A,B,C)
pd.concat([df1,df2],axis=1)
```

Out[117...

	A	B	C	A	B
0	1	3	5	5	7
1	2	4	6	6	8

In [118...
```
# The 'keys' parameter is used to create a hierarchical (MultiIndex) in the resulting concatenated DataFrame.
# Note: The 'keys' parameter does not work with 'ignore_index' parameter
data = pd.concat([df1,df2],keys=['Alice','Bob'])
```

In [119...
```
data
```

Out[119...

		A	B	C
Alice	**0**	1	3	5.0
	1	2	4	6.0
Bob	**0**	5	7	NaN
	1	6	8	NaN

In [120...
```
# You can access a part of this entire dataframe using a particular key
data.loc['Bob']
```

Out[120...

	A	B	C
0	5	7	NaN
1	6	8	NaN

MERGE

- Unlike concat() that just stacked the rows/columns, merge() is used to combine the dataframes based on common columns or indexes.
- It is similar to **SQL JOIN OPERATIONS**
- There are 4 ways in which you can combine your dataframes [using the 'how' parameter]:
  - INNER
  - OUTER
  - LEFT
  - RIGHT
- We also have a parameter called 'on' --> Considering which column you want to perform the above merge operations
- 'indicator' parameter --> Shows that the row came from which table

In [121…
```
df1 = pd.DataFrame({'Names':['Alice','Bob','Charlie'], 'Scores':[89,72,86]})
df2 = pd.DataFrame({'Names':['Alice','Bob','Don'], 'Scores':[79,72,96]})
```

In [122…
```
df1
```

Out[122…

	Names	Scores
0	Alice	89
1	Bob	72
2	Charlie	86

In [123…
```
df2
```

Out[123…

	Names	Scores
0	Alice	79
1	Bob	72
2	Don	96

In [124…
```
# By default it will apply INNER operation --> Extracts common data from both the columns in both the tables
pd.merge(df1,df2)
```

Out[124…

	Names	Scores
0	Bob	72

In [125…
```
# It will return only the NAMES that are present in both the tables and their corresponding scores
# The 'indicator' signifies from which table did the scores populate
# Here there are 2 scores and it is populated from each of the tables, hence 'both' is written
pd.merge(df1,df2,on='Names',how='inner',indicator=True)
```

Out[125…

	Names	Scores_x	Scores_y	_merge
0	Alice	89	79	both
1	Bob	72	72	both

In [126…
```
# Since it is an outer join, all the NAMES from both the tables and their corresponding scores are populated
# Here df1 --> LEFT TABLE and df2 --> RIGHT TABLE
# Since 'Charlie' was present only in df1 and 'Don' was present only in df2, the indicator shows from which table the values are populated
pd.merge(df1,df2,on='Names',how='outer',indicator=True)
```

Out[126…

	Names	Scores_x	Scores_y	_merge
0	Alice	89.0	79.0	both
1	Bob	72.0	72.0	both
2	Charlie	86.0	NaN	left_only
3	Don	NaN	96.0	right_only

In [127…
```
# Since it is a LEFT JOIN, the common NAMES and their corresponding scores from both the tables are taken and the remaining name 'Charlie' from LEFT TABLE is populated
pd.merge(df1,df2,on='Names',how='left',indicator=True)
```

Out[127…

	Names	Scores_x	Scores_y	_merge
0	Alice	89	79.0	both
1	Bob	72	72.0	both
2	Charlie	86	NaN	left_only

In [128…
```
# Since it is a RIGHT JOIN, the common NAMES and their corresponding scores from both the tables are taken and the remaining name 'DON' from RIGHT TABLE is populated
pd.merge(df1,df2,on='Names',how='right',indicator=True)
```

Out[128…

	Names	Scores_x	Scores_y	_merge
0	Alice	89.0	79	both
1	Bob	72.0	72	both
2	Don	NaN	96	right_only

BRAINSTORM EXERCISE 4: FIND ORANGE CAP HOLDERS FOR EACH SEASON

- To find Orange Cap Holders we need **season** data from 'df' and **batter name** and **batsman_runs** data from 'delivery'
- Since we need data from 2 tables, we need to combine it
- We merge the 2 tables on the basis of common column that they have, which is 'id' in df and 'match_id' in delivery
- To merge 2 tables with common columns, their column_names must be same, hence we are renaming 'match_id' in delivery dataframe to 'id'

In [129…
```
# Renaming the column names
# df.rename(columns={old_name:'new_name'})
delivery.rename(columns={'match_id':'id'},inplace=True)
```

In [130…
```
merged_data = pd.merge(df,delivery,on='id',how='outer')
```

In [131…
```
# Grouping the data on basis of season and batters for each season ---> merged_data.groupby(['season','batter'])
# Calculating the total scores scored by them --> ['batsman_runs'].sum()
# Sorting to get the highest runs first ---> sort_values(ascending=False)
# Converting this MultiIndex Series into a Dataframe --> reset_index()
# Dropping the duplicate seasons and keeping first occurence because the highest scores are present first ---> drop_duplicates(subset = 'season',keep='first')
# Sorting the season in order --> sort_values('season')
merged_data.groupby(['season','batter'])['batsman_runs'].sum().sort_values(ascending=False).reset_index().drop_duplicates(subset = 'season',keep='first').sort_values('season')
```

Out[131…

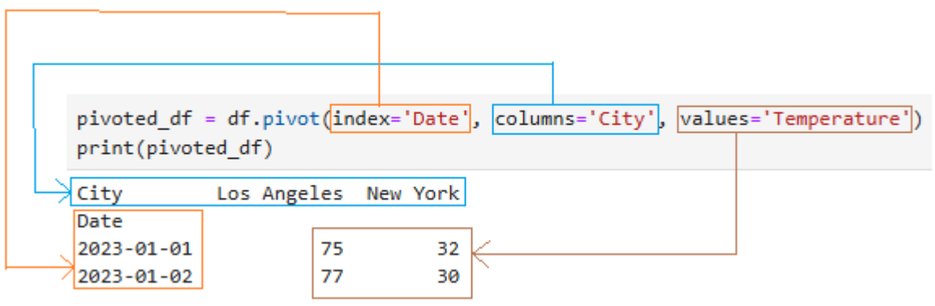
	season	batter	batsman_runs
26	2007/08	SE Marsh	616
38	2009	ML Hayden	572
25	2009/10	SR Tendulkar	618
28	2011	CH Gayle	608
6	2012	CH Gayle	733
7	2013	MEK Hussey	733
15	2014	RV Uthappa	660
42	2015	DA Warner	562
0	2016	V Kohli	973
17	2017	DA Warner	641
5	2018	KS Williamson	735
10	2019	DA Warner	692
13	2020/21	KL Rahul	676
20	2021	RD Gaikwad	635
2	2022	JC Buttler	863
1	2023	Shubman Gill	890
4	2024	V Kohli	741

PIVOT( )

- The pivot() function is used to reshape a DataFrame. This function is particularly useful for converting data from a "long" format to a "wide" format, making it easier to analyze and compare data across different categories.
- pivot() only works when your **index + columns** combination is **unique**
- If duplicate combinations exist, a ValueError will be raised. In such cases, pivot_table() should be used instead, as it can handle aggregation of duplicate values.

PARAMETERS:

- index: Specifies the colums) whose unique values will form the new(ro) of the reshaped DataFrame [BASICALLY THE COLUMN THAT WILL BE Y-AXIS]
- columns: Specifies the columns whose unique values will form the new column headers of the reshaped DataFrame. [BASICALLY THE COLUMN THAT WILL BE X-AXIS]
- values: Specifies the columns whose values will populate the cells of the reshaped DataFrame.



In [132…

```
data = pd.DataFrame({'Date': ['2023-01-01', '2023-01-01', '2023-01-02', '2023-01-02'],
        'City': ['New York', 'Los Angeles', 'New York', 'Los Angeles'],
        'Temperature': [32, 75, 30, 77]})
pivoted_df = data.pivot(index='Date', columns='City', values='Temperature')
print(pivoted_df)
```

```
City        Los Angeles  New York
Date
2023-01-01           75        32
2023-01-02           77        30
```

PIVOT_TABLE( )

- The pivot_table( ) is a powerful data summarization tool that allows you to reorganize and aggregate data from a DataFrame.

Parameters:

- data: The DataFrame you want to pivot.
- The 'index', 'columns' and 'values' parameter serve the same purpose as that in pivot( ) function.
- aggfunc: The aggregation function to apply to the values. [Default function that will be applied is 'mean', you can provide functions like sum, count and others]gregates.

In [133…

```
data = pd.DataFrame({'Date': ['2023-01-01', '2023-01-01', '2023-01-02', '2023-01-02', '2023-01-01'],
        'Region': ['East', 'West', 'East', 'West', 'East'],
        'Product': ['A', 'B', 'A', 'A', 'B'],
        'Sales': [100, 150, 120, 180, 200]})

# Create a pivot table to show sum of sales by Region and Product
pivot_df = pd.pivot_table(data, values='Sales', index='Region', columns='Product', aggfunc='sum')
print(pivot_df)
```

```
Product    A    B
Region
East     220  200
West     180  150
```

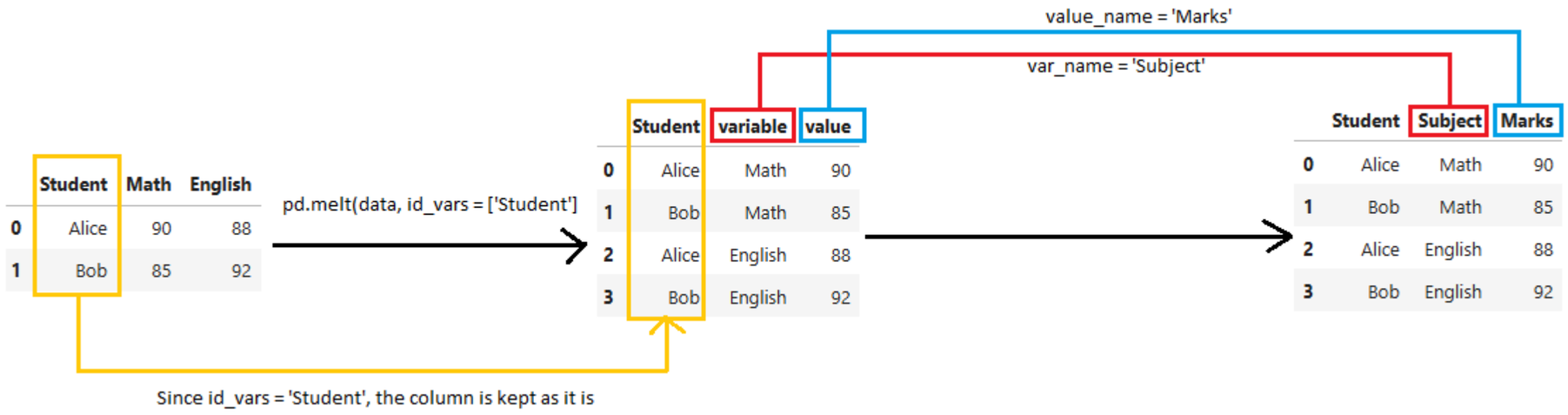
DIFFERENCE BETWEEN PIVOT( ) AND PIVOT_TABLE( ):

Both pivot( ) and pivot_table( ) are used for **reshaping data**, but they differ in their capabilities:

- pivot( ):
  - Does not perform any aggregation. It simply reshapes the data.
  - index + column pair must be unique, If there are duplicate values, ValueError is raised
- pivot_table( ):
  - Allows for aggregation of data. It can calculate statistics like sum, mean, count, min, max, etc., for values. This is controlled by the aggfunc argument.
  - Can handle duplicate values in the index and columns
- pivot() is suitable for straightforward reshaping where each combination of index and column values is unique and no aggregation is needed. pivot_table() is the more general and robust function, providing the ability to handle duplicates through aggregation and offering greater flexibility for advanced data analysis tasks.

MELT( ) FUNCTION:

- The melt() function is used to transform a DataFrame from a "wide" format to a "long" format. This process is also known as **"unpivoting"** or "reshaping" data.
- Parameters:
  - id_vars: A list of column names to be used as identifier variables. These columns will be kept as-is in the melted DataFrame
  - var_name: A string to rename the 'variable' column (default is 'variable').
  - value_name: A string to rename the 'value' column (default is 'value').



In [134…
```
data = pd.DataFrame({'Student': ['Alice', 'Bob'],
        'Math': [90, 85],
        'English': [88, 92]})
data
```

Out[134…

	Student	Math	English
0	Alice	90	88
1	Bob	85	92

In [135…
```
melted_df = pd.melt(data, id_vars = ['Student'], var_name = 'Subject',value_name = 'Marks')
melted_df
```

Out[135…

	Student	Subject	Marks
0	Alice	Math	90
1	Bob	Math	85
2	Alice	English	88
3	Bob	English	92

Handling large datasets and optimizing memory usage in Pandas:

Handling large datasets and optimizing memory usage in Pandas is crucial for efficient **data processing**. Several strategies include:

1. **Data Type Optimization:**
   - **Downcasting Numeric Types:** Pandas often defaults to int64 and float64, which consume more memory than necessary for many datasets. Downcast to smaller integer types (e.g., int8, int16, int32) or float types (e.g., float32) if the data range and precision allow.
   - **Categorical Data Types:** Convert columns with a limited number of unique values (categorical data) to the category dtype. This significantly reduces memory usage compared to object (string) dtype. [Eg. df["season"] = df["season"].astype("category")]
2. **Selective Data Loading:**
   - **usecols Parameter:** When reading CSV files with pd.read_csv(), use the **usecols** parameter to load only the necessary columns, avoiding loading irrelevant data into memory.
   - **low_memory Parameter:** For large CSV files, setting **low_memory=True** in pd.read_csv() can help Pandas process the file in chunks, reducing memory spikes during parsing.
3. **Efficient Operations:**
   - **Vectorized Operations:** Leverage Pandas' vectorized operations (e.g., df.sum(), df.mean(), df.apply()) instead of explicit Python loops, as they are significantly faster and more memory-efficient.

---------- END ----------