Exploring Pandas - Part 2

Module 3: Data Selection & Filtering

- 1. Boolean Indexing & Filtering
- 2. Multiple Conditions (&, |, ~)
- 3. isin(), between()
- 4. query() Method
- 5. Filtering with loc[] (label-based)

```
import pandas as pd

data = {
    'studentId': [101, 102, 103, 104, 105, 106, 107, 108],
    'Name': ['Srinivas', 'Vas', 'Hello', 'Srinivas', 'OK', 'Hai', 'Hell
    'Age': [25, 30, 35, 40, 45, 30, 35, 28],
    'Course': ['ML', 'ML', 'ML', 'Python', 'DL', 'ML', 'DL', 'ML'],
    'City': ['Bangalore', 'Chennai', 'Bangalore', 'Bangalore', 'Delhi',
    'Fee': [20000, 25000, 15000, 18000, 22000, 21000, 17000, 24000]
}
```

```
mydf = pd.DataFrame(data)

mydf
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000
6	107	Hello	35	DL	Pune	17000
7	108	Vas	28	ML	Chennai	24000

Next steps: Generate code with mydf New interactive sheet

1) Boolean Indexing & Filtering

```
# Q1) What is the Boolean mask for students whose age is greater than

mydf['Age'] > 30
```

```
     Age
0  False
1  False
2   True
3   True
4   True
5  False
6   True
7  False

dtype: bool
```

```
# Q2) Get all students whose age is greater than 30.

mydf[mydf['Age'] > 30]
```

	studentId	Name	Age	Course	City	Fee
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000
6	107	Hello	35	DL	Pune	17000

```
# Q3) Get all students who paid a fee of ₹20,000 or more.

mydf[mydf['Fee'] >= 20000]
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000
7	108	Vas	28	ML	Chennai	24000

```
# Q4) Get all students from Bangalore.

mydf[mydf['City'] == "Bangalore"]
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000

```
# Q5) Get all students enrolled in the ML course.

mydf[mydf['Course'] == "ML"]
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
2	103	Hello	35	ML	Bangalore	15000
5	106	Hai	30	ML	Hyderabad	21000
7	108	Vas	28	ML	Chennai	24000

```
# Q5A) Get students Sid, Name and Course who enrolled in the ML course

#a) All rows and All cols
```

```
print(mydf)
print("-"*50)

#b) Filtered rows and All cols
print(mydf[mydf['Course'] == "ML"])
print("-"*50)

#c) All rows and Filtered cols
print(mydf[["studentId","Name","Course"]])
print("-"*50)

#d) Filtered rows and Filtered cols
#mydf[mydf['Course'] == "ML"][["studentId","Name","Course"]]
mydf[["studentId","Name","Course"]][mydf['Course'] == "ML"]
```

```
   studentId       Name  Age  Course       City    Fee
0        101  Srinivas   25      ML  Bangalore  20000
1        102       Vas   30      ML    Chennai  25000
2        103     Hello   35      ML  Bangalore  15000
3        104  Srinivas   40  Python  Bangalore  18000
4        105        OK   45      DL      Delhi  22000
5        106       Hai   30      ML  Hyderabad  21000
6        107     Hello   35      DL       Pune  17000
7        108       Vas   28      ML    Chennai  24000
--------------------------------------------------
   studentId       Name  Age Course       City    Fee
0        101  Srinivas   25     ML  Bangalore  20000
1        102       Vas   30     ML    Chennai  25000
2        103     Hello   35     ML  Bangalore  15000
5        106       Hai   30     ML  Hyderabad  21000
7        108       Vas   28     ML    Chennai  24000
--------------------------------------------------
   studentId       Name  Course
0        101  Srinivas      ML
1        102       Vas      ML
2        103     Hello      ML
3        104  Srinivas  Python
4        105        OK      DL
5        106       Hai      ML
6        107     Hello      DL
7        108       Vas      ML
--------------------------------------------------
```

	studentId	Name	Course
0	101	Srinivas	ML
1	102	Vas	ML
2	103	Hello	ML
5	106	Hai	ML

```
7      108     Vas     ML
```

2) Multiple Conditions (&, |, ~)

```
# Q6) Get all students whose age is greater than 30 and course is "ML'

mydf[(mydf['Age'] < 35) & (mydf['Course'] == "ML")]
```

```
   studentId      Name  Age  Course       City    Fee
0        101  Srinivas   25      ML  Bangalore  20000
1        102       Vas   30      ML    Chennai  25000
5        106       Hai   30      ML  Hyderabad  21000
7        108       Vas   28      ML    Chennai  24000
```

```
# Q7) Get all students whose city is "Chennai" or "Bangalore".

mydf[(mydf['City'] == "Chennai") | (mydf['City'] == "Bangalore")]
```

```
   studentId      Name  Age  Course       City    Fee
0        101  Srinivas   25      ML  Bangalore  20000
1        102       Vas   30      ML    Chennai  25000
2        103     Hello   35      ML  Bangalore  15000
3        104  Srinivas   40  Python  Bangalore  18000
7        108       Vas   28      ML    Chennai  24000
```

```
# Q8) Get all students not from "Bangalore".

mydf[~(mydf['City'] == "Bangalore")]
```

	studentId	Name	Age	Course	City	Fee
1	102	Vas	30	ML	Chennai	25000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000
6	107	Hello	35	DL	Pune	17000
7	108	Vas	28	ML	Chennai	24000

```
# Q9) Get all students whose age is greater than 30 and fee is less tl

mydf[(mydf['Age'] > 30) & (mydf['Fee'] < 20000)]
```

	studentId	Name	Age	Course	City	Fee
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
6	107	Hello	35	DL	Pune	17000

```
# Q10) Get all students whose course is not "Python" and fee is more t

mydf[(mydf['Course'] != "Python") & (mydf['Fee'] > 20000)]
```

	studentId	Name	Age	Course	City	Fee
1	102	Vas	30	ML	Chennai	25000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000
7	108	Vas	28	ML	Chennai	24000

```
# Q11) Get all students whose Age > 25, Course is 'ML', and City is 'E

mydf[
    (mydf["Age"] >= 25) &
    (mydf["Course"] == "ML") &
    (mydf["City"] == "Bangalore")
]
```

```
   studentId      Name  Age  Course       City    Fee
0        101  Srinivas   25      ML  Bangalore  20000
2        103     Hello   35      ML  Bangalore  15000
```

```
# Q12) Get all students whose age >= 25, course is "ML", city is "Hyder









mydf[
    (mydf["Age"] >= 25) &
    (mydf["Course"] == "ML") &
    (mydf["City"] == "Bangalore") &
    (mydf["Name"].str.startswith("S"))
]
```

```
   studentId      Name  Age  Course       City    Fee
0        101  Srinivas   25      ML  Bangalore  20000
```

3) isin(), between()

```
# Q13) Get all students whose city is "Chennai" or "Bangalore" or "Hyc
```

```
mydf[
     (mydf["City"] == "Chennai") |
     (mydf["City"] == "Bangalore") |
     (mydf["City"] == "Hyderabad") |
     (mydf["City"] == "Delhi")
    ]
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000
7	108	Vas	28	ML	Chennai	24000

```
# Alternatively use -  .isin():

mydf[mydf["City"].isin(["Chennai", "Bangalore", "Hyderabad", "Delhi"]
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000
7	108	Vas	28	ML	Chennai	24000

```
# Q14) Get all students who are not staying in "Bangalore" or "Hyderal

mydf[~mydf['City'].isin(['Bangalore', 'Hyderabad'])]
```

	studentId	Name	Age	Course	City	Fee
1	102	Vas	30	ML	Chennai	25000
4	105	OK	45	DL	Delhi	22000
6	107	Hello	35	DL	Pune	17000
7	108	Vas	28	ML	Chennai	24000

```
# Q15) Get all students whose Fee is between 18,000 and 24,000 (inclus

mydf[(mydf['Fee'] >= 18000) & (mydf['Fee'] <= 24000)]
```

```
   studentId      Name  Age  Course       City    Fee
0        101  Srinivas   25      ML  Bangalore  20000
3        104  Srinivas   40  Python  Bangalore  18000
4        105        OK   45      DL      Delhi  22000
5        106       Hai   30      ML  Hyderabad  21000
7        108       Vas   28      ML    Chennai  24000
```

```
# Alternatively use -  .between():

mydf[mydf['Fee'].between(18000, 24000, inclusive="both")]
```

```
   studentId      Name  Age  Course       City    Fee
0        101  Srinivas   25      ML  Bangalore  20000
3        104  Srinivas   40  Python  Bangalore  18000
4        105        OK   45      DL      Delhi  22000
5        106       Hai   30      ML  Hyderabad  21000
7        108       Vas   28      ML    Chennai  24000
```

```
# Q16) Get all students whose Age is between 25 and 35 (exclusive).

mydf[(mydf['Age'] > 25) & (mydf['Age'] < 35)]
```

```
   studentId  Name  Age  Course       City    Fee
1        102   Vas   30      ML    Chennai  25000
5        106   Hai   30      ML  Hyderabad  21000
7        108   Vas   28      ML    Chennai  24000
```

```
# Alternatively using .between():

mydf[mydf['Age'].between(25, 35, inclusive="neither")]
```

	studentId	Name	Age	Course	City	Fee
1	102	Vas	30	ML	Chennai	25000
5	106	Hai	30	ML	Hyderabad	21000
7	108	Vas	28	ML	Chennai	24000

```
# using .between():

mydf[mydf['Age'].between(25, 35, inclusive="left")]
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
5	106	Hai	30	ML	Hyderabad	21000
7	108	Vas	28	ML	Chennai	24000

```
# using .between():




mydf[mydf['Age'].between(25, 35, inclusive="right")]
```

```
   studentId  Name  Age Course       City    Fee
1        102   Vas   30     ML    Chennai  25000
2        103 Hello   35     ML  Bangalore  15000
5        106   Hai   30     ML  Hyderabad  21000
6        107 Hello   35     DL       Pune  17000
7        108   Vas   28     ML    Chennai  24000
```

4) query() Method

```
# Q17) Get all students whose age is greater than 30.

mydf.query("Age > 30")
```

```
   studentId      Name  Age  Course       City    Fee
2        103     Hello   35      ML  Bangalore  15000
3        104  Srinivas   40  Python  Bangalore  18000
4        105        OK   45      DL      Delhi  22000
6        107     Hello   35      DL       Pune  17000
```

```
# Q18) Get all students from Bangalore.

mydf.query("City == 'Bangalore'")
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000

```
# Q19) Get all students whose fee is at least ₹20,000.

mydf.query("Fee >= 20000")
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000
7	108	Vas	28	ML	Chennai	24000

```
# Q20) Get all students whose city is either 'Chennai' or 'Delhi'.

mydf.query("City in ['Chennai', 'Delhi']")
```

	studentId	Name	Age	Course	City	Fee
1	102	Vas	30	ML	Chennai	25000
4	105	OK	45	DL	Delhi	22000
7	108	Vas	28	ML	Chennai	24000

```python
# Q21) Get all students whose course is not 'Python'.

mydf.query("Course != 'Python'")
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
2	103	Hello	35	ML	Bangalore	15000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000
6	107	Hello	35	DL	Pune	17000
7	108	Vas	28	ML	Chennai	24000

```python
# Q22) Get all students whose age is between 25 and 35 (inclusive).

mydf.query("Age >= 25 & Age <= 35")
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
2	103	Hello	35	ML	Bangalore	15000
5	106	Hai	30	ML	Hyderabad	21000
6	107	Hello	35	DL	Pune	17000
7	108	Vas	28	ML	Chennai	24000

```
# Q23) Get all students whose age is less than 30 or fee is more than

mydf.query("Age < 30 | Fee > 23000")
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
1	102	Vas	30	ML	Chennai	25000
7	108	Vas	28	ML	Chennai	24000

```
# Q24) Get all students whose name starts with 'S'.

mydf.query("Name.str.startswith('S')")
```

	studentId	Name	Age	Course	City	Fee
0	101	Srinivas	25	ML	Bangalore	20000
3	104	Srinivas	40	Python	Bangalore	18000

5) Filtering with loc[] (label-based)

- loc[] selects rows and columns by labels (names), not by positions.

Syntax:

```
df.loc[row_labels, column_labels]
```

```
# Q25) Get studentId, Name and City of student with studentId 105

mydf.loc[mydf['studentId'] == 105, ['studentId','Name', 'City']]
```

	studentId	Name	City
4	105	OK	Delhi

```
# Q26) Get Name, Age, 'City' and Fee of students from Bangalore

mydf.loc[mydf['City'] == 'Bangalore', ['Name', 'Age', 'Fee','City']]
```

	Name	Age	Fee	City
0	Srinivas	25	20000	Bangalore
2	Hello	35	15000	Bangalore
3	Srinivas	40	18000	Bangalore

```
# Q27) Get all details of students whose Age > 35

mydf.loc[mydf['Age'] > 35, :]
```

	studentId	Name	Age	Course	City	Fee
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000

Querying the DataFrame

- 3 ways to query the DataFrame
  - Querying the Dataframe with []
  - Querying the Dataframe with query() method
  - Querying the Dataframe loc() method

```
# Querying the Dataframe with []
mydf[mydf['Age'] > 30]
```

	studentId	Name	Age	Course	City	Fee
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000
6	107	Hello	35	DL	Pune	17000

```python
# Querying the Dataframe with query() method
mydf.query("Age > 30")
```

	studentId	Name	Age	Course	City	Fee
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000
6	107	Hello	35	DL	Pune	17000

```python
# Querying the Dataframe loc() method
mydf.loc[mydf['Age'] > 30, :]
```

	studentId	Name	Age	Course	City	Fee
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000
6	107	Hello	35	DL	Pune	17000

```python
mydf.iloc[2:6]
```

	studentId	Name	Age	Course	City	Fee
2	103	Hello	35	ML	Bangalore	15000
3	104	Srinivas	40	Python	Bangalore	18000
4	105	OK	45	DL	Delhi	22000
5	106	Hai	30	ML	Hyderabad	21000

```
mydf.iloc[2:6,1:5]
```

	Name	Age	Course	City
2	Hello	35	ML	Bangalore
3	Srinivas	40	Python	Bangalore
4	OK	45	DL	Delhi
5	Hai	30	ML	Hyderabad

```
mydf.loc[2:6,'Name':'City']
```

	Name	Age	Course	City
2	Hello	35	ML	Bangalore
3	Srinivas	40	Python	Bangalore
4	OK	45	DL	Delhi
5	Hai	30	ML	Hyderabad
6	Hello	35	DL	Pune

Module 4: Data Cleaning and Preprocessing

- Handling Missing Data
- Type Conversion
- String Operations
- Duplicates Handling
- Mapping and Replacing Values

```
import pandas as pd

mydf = pd.read_csv("mystudents_data_1.csv")

mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
1	102	Vas	30	DevOps	Chennai	25000	90
2	103	Hello	NaN	Java	Bangalore	15000	NaN
3	104	Manish	40	Python	Mumbai	18000	78%
4	105	Amit	45	DL	NaN	22000	88.5
5	106	Hai	30	ML	hyderabad	21000	72
6	107	Hello	35	DL	Pune	17000	sixty
7	108	Vas	28	ai	Chennai	24000	95
8	109	RAJ	32	ML	Mumbai	19000	65/100
9	110	Ok	26	NaN	Delhi	16000	68
10	111	Alok	38 years	Python	BANGALORE	23000	88,5
11	112	Super	29	DevOps	Pune	NaN	NaN
12	104	Manish	40	Python	Mumbai	18000	78%
13	108	Vas	28	AI	Chennai	24500	96
14	113	Siri	27	ML	Bangalore	20000	NaN
15	114	Kiran	NaN	NaN	NaN	NaN	absent
16	101	srinivas	25	ML	Bangalore	20k	85
17	109	RAJ	32	ML	Mumbai	19000	65/100
18	115	Hello	35	DL	Pune	17100	sixty
19	108	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with mydf New interactive sheet

A) Handling Missing Data

1) Converting placeholders to real NaN

- a) replace(..., value=pd.NA)

a) replace(..., value=pd.NA)

- Turn “fake” missing markers into true NaN so you can fill/drop consistently.

```
import numpy as np

print(mydf['Marks'].isna().sum()) #3

placeholders = {'na', 'n/a', 'none', 'null', 'missing', 'absent', 'nar

mask = mydf['Marks'].astype(str).str.strip().str.lower().isin(placeho
mydf.loc[mask, 'Marks'] = np.nan;

print(mydf['Marks'].isna().sum()) #4
```

```
3
4
```

```
placeholders = {'na', 'n/a', 'none', 'null', 'missing', 'absent', 'nan'

for mycol in mydf.select_dtypes(include=['object', 'string', 'integer',
    print(mydf[mycol].isna().sum())
    mask = mydf[mycol].astype('string').str.strip().str.lower().isin(pl
    mydf.loc[mask, mycol] = np.nan
    print(mydf[mycol].isna().sum())
```

```
0
0
0
0
2
2
2
2
2
2
2
2
4
4
```

```
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25	ML	Bangalore	20k	85
1	102.0	Vas	30	DevOps	Chennai	25000	90
2	103.0	Hello	NaN	Java	Bangalore	15000	NaN
3	104.0	Manish	40	Python	Mumbai	18000	78%
4	105.0	Amit	45	DL	NaN	22000	88.5
5	106.0	Hai	30	ML	hyderabad	21000	72
6	107.0	Hello	35	DL	Pune	17000	sixty
7	108.0	Vas	28	ai	Chennai	24000	95
8	109.0	RAJ	32	ML	Mumbai	19000	65/100
9	110.0	Ok	26	NaN	Delhi	16000	68
10	111.0	Alok	38 years	Python	BANGALORE	23000	88,5
11	112.0	Super	29	DevOps	Pune	NaN	NaN
12	104.0	Manish	40	Python	Mumbai	18000	78%
13	108.0	Vas	28	AI	Chennai	24500	96
14	113.0	Siri	27	ML	Bangalore	20000	NaN
15	114.0	Kiran	NaN	NaN	NaN	NaN	NaN
16	101.0	srinivas	25	ML	Bangalore	20k	85
17	109.0	RAJ	32	ML	Mumbai	19000	65/100
18	115.0	Hello	35	DL	Pune	17100	sixty
19	108.0	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with mydf New interactive sheet

2) Convert unparseable values to NaN

- a) to_numeric(..., errors='coerce')

a) to_numeric(..., errors='coerce')

- When numbers are stored as text—failed parses become NaN (then fill/drop).

```
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25	ML	Bangalore	20k	85
1	102.0	Vas	30	DevOps	Chennai	25000	90
2	103.0	Hello	NaN	Java	Bangalore	15000	NaN
3	104.0	Manish	40	Python	Mumbai	18000	78%
4	105.0	Amit	45	DL	NaN	22000	88.5
5	106.0	Hai	30	ML	hyderabad	21000	72
6	107.0	Hello	35	DL	Pune	17000	sixty
7	108.0	Vas	28	ai	Chennai	24000	95
8	109.0	RAJ	32	ML	Mumbai	19000	65/100
9	110.0	Ok	26	NaN	Delhi	16000	68
10	111.0	Alok	38 years	Python	BANGALORE	23000	88,5
11	112.0	Super	29	DevOps	Pune	NaN	NaN
12	104.0	Manish	40	Python	Mumbai	18000	78%
13	108.0	Vas	28	AI	Chennai	24500	96
14	113.0	Siri	27	ML	Bangalore	20000	NaN
15	114.0	Kiran	NaN	NaN	NaN	NaN	NaN
16	101.0	srinivas	25	ML	Bangalore	20k	85
17	109.0	RAJ	32	ML	Mumbai	19000	65/100
18	115.0	Hello	35	DL	Pune	17100	sixty
19	108.0	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with mydf New interactive sheet

```
mydf['Age'] = pd.to_numeric(mydf['Age'], errors='coerce')   # '38 yea

mydf['Fee'] = pd.to_numeric(mydf['Fee'], errors='coerce')   # '17000'

mydf['Marks'] = pd.to_numeric(mydf['Marks'], errors='coerce') # '85' -
```

```
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	NaN	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	NaN	Delhi	16000.0	68.0
10	111.0	Alok	NaN	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	NaN	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	NaN	NaN	NaN	NaN	NaN
16	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN
19	108.0	Vas	28.0	AI	Chennai	25000.0	92.0

Next steps: Generate code with mydf New interactive sheet

3) Detecting missing values

- a) isnull() / isna()
- b) notnull() / notna()
- c) Row-wise checks with any() / all()
- d) Check Null percentages

a) isnull() / isna()

- Find missing values (returns True/False).
- They both treat NaN, None, and NaT as missing.
- literal string like "NaN" is not missing.

```python
# nulls per column
mydf.isnull()
```

	studentId	Name	Age	Course	City	Fee	Marks
0	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False
2	False	False	True	False	False	False	True
3	False	False	False	False	False	False	True
4	False	False	False	False	True	False	False
5	False	False	False	False	False	False	False
6	False	False	False	False	False	False	True
7	False	False	False	False	False	False	False
8	False	False	False	False	False	False	True
9	False	False	False	True	False	False	False
10	False	False	True	False	False	False	True
11	False	False	False	False	False	True	True
12	False	False	False	False	False	False	True
13	False	False	False	False	False	False	False
14	False	False	False	False	False	False	True
15	False	False	True	True	True	True	True
16	False	False	False	False	False	True	False
17	False	False	False	False	False	False	True
18	False	False	False	False	False	False	True
19	False	False	False	False	False	False	False

```
# nulls per column
mydf.isnull().sum()
```

```
           0
studentId  0
Name       0
Age        3
Course     2
City       2
Fee        4
Marks     11

dtype: int64
```

```
# nulls per column
mydf.isna().sum()
```

```
           0
studentId  0
Name       0
Age        3
Course     2
City       2
Fee        4
Marks     11

dtype: int64
```

```
mydf['Course'].isna().sum()
```

```
np.int64(2)
```

```
mydf['Marks'].isna().sum()
```

```
np.int64(11)
```

```python
# Rows where Course is null
mydf[mydf['Course'].isna() ]
```

	studentId	Name	Age	Course	City	Fee	Marks
9	110.0	Ok	26.0	NaN	Delhi	16000.0	68.0
15	114.0	Kiran	NaN	NaN	NaN	NaN	NaN

```python
# Rows where Age is null
mydf[ mydf['Age'].isna() ]
```

	studentId	Name	Age	Course	City	Fee	Marks
2	103.0	Hello	NaN	Java	Bangalore	15000.0	NaN
10	111.0	Alok	NaN	Python	BANGALORE	23000.0	NaN
15	114.0	Kiran	NaN	NaN	NaN	NaN	NaN

```python
# Rows where Marks is null
mydf[ mydf['Marks'].isna() ]
```

	studentId	Name	Age	Course	City	Fee	Marks
2	103.0	Hello	NaN	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
10	111.0	Alok	NaN	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	NaN	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	NaN	NaN	NaN	NaN	NaN
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN

b) notnull() / notna()

- Find non-missing values (returns True/False).

```
mydf['City'].notna()
```

```
     City
 0   True
 1   True
 2   True
 3   True
 4  False
 5   True
 6   True
 7   True
 8   True
 9   True
10   True
11   True
12   True
13   True
14   True
15  False
16   True
17   True
18   True
19   True

dtype: bool
```

```python
# Rows where City is present
mydf[ mydf['City'].notna() ]
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	NaN	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	NaN	Delhi	16000.0	68.0
10	111.0	Alok	NaN	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	NaN	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
16	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN
19	108.0	Vas	28.0	AI	Chennai	25000.0	92.0

```
mydf['Marks'].notnull()
```

	Marks
0	True
1	True
2	False
3	False
4	True
5	True
6	False
7	True
8	False
9	True
10	False
11	False
12	False
13	True
14	False
15	False
16	True
17	False
18	False
19	True

dtype: bool

```
# Rows where Marks is present
mydf[ mydf['Marks'].notnull() ]
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
9	110.0	Ok	26.0	NaN	Delhi	16000.0	68.0
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
16	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
19	108.0	Vas	28.0	AI	Chennai	25000.0	92.0

c) Row-wise checks with any() / all()

- Flag rows with any/all nulls.
- **mydf.isnull()** → a DataFrame of booleans (True where the cell is missing; False otherwise).
- **.any(axis=1)** → for each row, checks if any column is True (i.e., at least one missing in that row).
- **.all(axis=1)** → for each row, checks if all columns are True (i.e., the row is entirely missing).

```
# Display the Row if any column is missing
mydf[ mydf.isna().any(axis=1) ]
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
2	103.0	Hello	NaN	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	NaN	Delhi	16000.0	68.0
10	111.0	Alok	NaN	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	NaN	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	NaN	NaN	NaN	NaN	NaN
16	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN

```
# Display the Row if All the Columns are missing
mydf[ mydf.isna().all(axis=1) ]
```

studentId	Name	Age	Course	City	Fee	Marks

```
# Count the Row if any column is missing
num_any = mydf.isna().any(axis=1).sum()

print(num_any)
```

```
15
```

```
# Count the Row if all columns are missing
num_all = mydf.isnull().all(axis=1).sum()

print(num_all)
```

```
0
```

d) Null percentages

- How much of each column is missing (in %).

```
mydf.isna().mean() * 100
```

```
             0
studentId  0.0
     Name  0.0
      Age 15.0
   Course 10.0
     City 10.0
      Fee 20.0
    Marks 55.0

dtype: float64
```

```
(mydf.isna().mean() * 100).sort_values(ascending=False)
```

```
                0
Marks        55.0
Fee          20.0
Age          15.0
City         10.0
Course       10.0
Name          0.0
studentId     0.0

dtype: float64
```

4) Filling missing values

- a) fillna(value)
- b) fillna(method='ffill' | 'bfill', limit=...) - Deprecated
- c) ffill()
- d) bfill()

a) fillna(value)

- Fill nulls with a constant or computed value.

```
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	NaN	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	NaN	Delhi	16000.0	68.0
10	111.0	Alok	NaN	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	NaN	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	NaN	NaN	NaN	NaN	NaN
16	101.0	srinivas	25.0	ML	Bangalore	NaN	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN
19	108.0	Vas	28.0	AI	Chennai	25000.0	92.0

Next steps: Generate code with mydf New interactive sheet

```
mydf['Course'] = mydf['Course'].fillna('Deep Learning')

mydf['Course']
```

```
           Course
 0             ML
 1         DevOps
 2           Java
 3         Python
 4             DL
 5             ML
 6             DL
 7             ai
 8             ML
 9  Deep Learning
10         Python
11         DevOps
12         Python
13             AI
14             ML
15  Deep Learning
16             ML
17             ML
18             DL
19             AI

dtype: object
```

```python
mydf['Fee'] = mydf['Fee'].fillna(mydf['Fee'].median())

mydf['Fee']
```

```
         Fee
 0   19500.0
 1   25000.0
 2   15000.0
 3   18000.0
 4   22000.0
 5   21000.0
 6   17000.0
 7   24000.0
 8   19000.0
 9   16000.0
10   23000.0
11   19500.0
12   18000.0
13   24500.0
14   20000.0
15   19500.0
16   19500.0
17   19000.0
18   17100.0
19   25000.0

dtype: float64
```

```python
mydf['Age'] = mydf['Age'].fillna(mydf['Age'].median())

mydf['Age']
```

```
     Age
 0  25.0
 1  30.0
 2  30.0
 3  40.0
 4  45.0
 5  30.0
 6  35.0
 7  28.0
 8  32.0
 9  26.0
10  30.0
11  29.0
12  40.0
13  28.0
14  27.0
15  30.0
16  25.0
17  32.0
18  35.0
19  28.0

dtype: float64
```

b) ffill(limit=1)

- Forward fill from neighbors (good for time-like data).

```
# check Null Count
null_count = mydf['Age'].isna().sum()
print(null_count)

# Forward-fill only the first NaN in each block
mydf['Age'] = mydf['Age'].ffill(limit=1)

# check Null Count
null_count = mydf['Age'].isna().sum()
print(null_count)
```

```
0
0
```

c) bfill(limit=1)

- Backward fill from the next non-null value (good for time-like data).

```
# check Null Count
null_count = mydf['Fee'].isna().sum()
print(null_count)

# Backward-fill only the first NaN in each block
mydf['Fee'] = mydf['Fee'].bfill(limit=1)

# check Null Count
null_count = mydf['Fee'].isna().sum()
print(null_count)
```

```
0
0
```

```
myseries = pd.Series([10,np.nan,np.nan,40,np.nan,50])
print(myseries)

#myseries = myseries.ffill(limit=1)
#print(myseries)

#myseries = myseries.bfill(limit=1)
#print(myseries)

#myseries = myseries.ffill()
#print(myseries)

myseries = myseries.bfill()
print(myseries)
```

```
0    10.0
1     NaN
2     NaN
3    40.0
4     NaN
5    50.0
dtype: float64
0    10.0
1    40.0
2    40.0
3    40.0
4    50.0
5    50.0
dtype: float64
```

5) Interpolating numeric gaps

- a) interpolate(method=...)

a) interpolate(method=...)

- Estimate numeric nulls from nearby values.

```
mydf['Age'] = mydf['Age'].interpolate(method='linear')
```

```
# mydf['Age'] = mydf['Age'].interpolate(method='linear')
# mydf['Age'] = mydf['Age'].interpolate(method='linear', limit=1)
# mydf['Age'] = mydf['Age'].interpolate(method='nearest')

# mydf['Age'] = mydf['Age'].interpolate(method='pad')   # or method='
# mydf['Age'] = mydf['Age'].interpolate(method='backfill')  # or metho
```

```
import numpy as np
import pandas as pd

myseries = pd.Series([10,np.nan,np.nan,40,np.nan,50])

myseries = myseries.interpolate(method="linear")
print(myseries)

#myseries = myseries.interpolate(method="linear",limit=1)
#print(myseries)

#myseries = myseries.interpolate(method="nearest")
#print(myseries)

#myseries = myseries.interpolate(method="pad") # deprecated ( use ffi
#print(myseries)

#myseries = myseries.interpolate(method="backfill") # deprecated ( use
#print(myseries)
```

```
0    10.0
1    20.0
2    30.0
3    40.0
4    45.0
5    50.0
dtype: float64
```

6) Dropping missing values

- a) dropna(subset=..., how=..., thresh=..., axis=...)

a) dropna(subset=..., how=..., thresh=..., axis=...)

- Remove rows/columns with missing values by rule.
- **axis:** 0 = rows (default), 1 = columns
- **subset:** only look at these columns when deciding to drop rows
- **how:** 'any' (drop if any NA) or 'all' (drop if all NA)
- **thresh:** minimum number of non-NA values required to keep the row/column

```
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	30.0	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0	68.0
10	111.0	Alok	30.0	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	19500.0	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	30.0	Deep Learning	NaN	19500.0	NaN
16	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN

Next steps: Generate code with mydf New interactive sheet

```
 # keep rows where Marks is present
 # Drop the Rows where Marks are NaN
mydf_rows = mydf.dropna(subset=['Marks'])

mydf_rows
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0	68.0
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
16	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
19	108.0	Vas	28.0	AI	Chennai	25000.0	92.0

Next steps: Generate code with `mydf_rows` New interactive sheet

```
# drop rows where all columns are NaN
mydf_rows = mydf.dropna(how='all')

mydf_rows
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	30.0	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0	68.0
10	111.0	Alok	30.0	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	19500.0	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	30.0	Deep Learning	NaN	19500.0	NaN
16	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN

Next steps: Generate code with mydf_rows New interactive sheet

```
# keep rows only if BOTH Fee and Marks are present
mydf.dropna(subset=['Fee','Marks'], how='any')
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0	68.0
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
16	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
19	108.0	Vas	28.0	AI	Chennai	25000.0	92.0

```
# keep rows unless BOTH Fee and Marks are missing
mydf.dropna(subset=['Fee','Marks'], how='all')
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	30.0	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0	68.0
10	111.0	Alok	30.0	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	19500.0	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	30.0	Deep Learning	NaN	19500.0	NaN
16	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN

```
(mydf.isna().mean()*100).round(2).sort_values(ascending=False)
```

```
             0
Marks     55.0
City      10.0
studentId  0.0
Age        0.0
Name       0.0
Course     0.0
Fee        0.0

dtype: float64
```

```
# drop columns with >=50% missing (keep <50% filled)
mydf.dropna(axis=1, thresh=int(0.5*len(mydf)))
```

	studentId	Name	Age	Course	City	Fee
0	101.0	srinivas	25.0	ML	Bangalore	19500.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0
2	103.0	Hello	30.0	Java	Bangalore	15000.0
3	104.0	Manish	40.0	Python	Mumbai	18000.0
4	105.0	Amit	45.0	DL	NaN	22000.0
5	106.0	Hai	30.0	ML	hyderabad	21000.0
6	107.0	Hello	35.0	DL	Pune	17000.0
7	108.0	Vas	28.0	ai	Chennai	24000.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0
10	111.0	Alok	30.0	Python	BANGALORE	23000.0
11	112.0	Super	29.0	DevOps	Pune	19500.0
12	104.0	Manish	40.0	Python	Mumbai	18000.0
13	108.0	Vas	28.0	AI	Chennai	24500.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0
15	114.0	Kiran	30.0	Deep Learning	NaN	19500.0
16	101.0	srinivas	25.0	ML	Bangalore	19500.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0
18	115.0	Hello	35.0	DL	Pune	17100.0
19	108.0	Vas	28.0	AI	Chennai	25000.0

```
# drop columns with >=20% missing (keep <90% filled)
mydf.dropna(axis=1, thresh=int(0.85*len(mydf)))
```

```
    studentId      Name   Age         Course       City      Fee
0       101.0  srinivas  25.0             ML  Bangalore  19500.0
1       102.0       Vas  30.0         DevOps    Chennai  25000.0
2       103.0     Hello  30.0           Java  Bangalore  15000.0
3       104.0    Manish  40.0         Python     Mumbai  18000.0
4       105.0      Amit  45.0             DL        NaN  22000.0
5       106.0       Hai  30.0             ML  hyderabad  21000.0
6       107.0     Hello  35.0             DL       Pune  17000.0
7       108.0       Vas  28.0             ai    Chennai  24000.0
8       109.0       RAJ  32.0             ML     Mumbai  19000.0
9       110.0        Ok  26.0  Deep Learning      Delhi  16000.0
10      111.0      Alok  30.0         Python  BANGALORE  23000.0
11      112.0     Super  29.0         DevOps       Pune  19500.0
12      104.0    Manish  40.0         Python     Mumbai  18000.0
13      108.0       Vas  28.0             AI    Chennai  24500.0
14      113.0      Siri  27.0             ML  Bangalore  20000.0
15      114.0     Kiran  30.0  Deep Learning        NaN  19500.0
16      101.0  srinivas  25.0             ML  Bangalore  19500.0
17      109.0       RAJ  32.0             ML     Mumbai  19000.0
18      115.0     Hello  35.0             DL       Pune  17100.0
19      108.0       Vas  28.0             AI    Chennai  25000.0
```

```
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	30.0	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0	68.0
10	111.0	Alok	30.0	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	19500.0	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	30.0	Deep Learning	NaN	19500.0	NaN
16	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN

Next steps: Generate code with `mydf` New interactive sheet

```
# drop rows that have fewer than 3 non-NA values (across all columns)
mydf.dropna(axis=0, thresh=3,inplace=True)

mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	30.0	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0	68.0
10	111.0	Alok	30.0	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	19500.0	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	30.0	Deep Learning	NaN	19500.0	NaN
16	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN

Next steps: Generate code with mydf New interactive sheet

```
# drop rows that have fewer than 3 non-NA values (across all columns)
mydf.dropna(axis=1, thresh=int(0.42*len(mydf)),inplace=True)

mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
1	102.0	Vas	30.0	DevOps	Chennai	25000.0	90.0
2	103.0	Hello	30.0	Java	Bangalore	15000.0	NaN
3	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
4	105.0	Amit	45.0	DL	NaN	22000.0	88.5
5	106.0	Hai	30.0	ML	hyderabad	21000.0	72.0
6	107.0	Hello	35.0	DL	Pune	17000.0	NaN
7	108.0	Vas	28.0	ai	Chennai	24000.0	95.0
8	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
9	110.0	Ok	26.0	Deep Learning	Delhi	16000.0	68.0
10	111.0	Alok	30.0	Python	BANGALORE	23000.0	NaN
11	112.0	Super	29.0	DevOps	Pune	19500.0	NaN
12	104.0	Manish	40.0	Python	Mumbai	18000.0	NaN
13	108.0	Vas	28.0	AI	Chennai	24500.0	96.0
14	113.0	Siri	27.0	ML	Bangalore	20000.0	NaN
15	114.0	Kiran	30.0	Deep Learning	NaN	19500.0	NaN
16	101.0	srinivas	25.0	ML	Bangalore	19500.0	85.0
17	109.0	RAJ	32.0	ML	Mumbai	19000.0	NaN
18	115.0	Hello	35.0	DL	Pune	17100.0	NaN

Next steps: Generate code with mydf New interactive sheet

B) Type Conversion

- a) pd.to_numeric(..., errors='coerce', downcast=...)
- b) astype(...)
- c) pd.to_datetime(..., errors='coerce')
- d) Downcasting for memory
- e) convert_dtypes()

```python
import pandas as pd

mydf = pd.read_csv("mystudents_data_1.csv")

mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
1	102	Vas	30	DevOps	Chennai	25000	90
2	103	Hello	NaN	Java	Bangalore	15000	NaN
3	104	Manish	40	Python	Mumbai	18000	78%
4	105	Amit	45	DL	NaN	22000	88.5
5	106	Hai	30	ML	hyderabad	21000	72
6	107	Hello	35	DL	Pune	17000	sixty
7	108	Vas	28	ai	Chennai	24000	95
8	109	RAJ	32	ML	Mumbai	19000	65/100
9	110	Ok	26	NaN	Delhi	16000	68
10	111	Alok	38 years	Python	BANGALORE	23000	88,5
11	112	Super	29	DevOps	Pune	NaN	NaN
12	104	Manish	40	Python	Mumbai	18000	78%
13	108	Vas	28	AI	Chennai	24500	96
14	113	Siri	27	ML	Bangalore	20000	NaN
15	114	Kiran	NaN	NaN	NaN	NaN	absent
16	101	srinivas	25	ML	Bangalore	20k	85
17	109	RAJ	32	ML	Mumbai	19000	65/100
18	115	Hello	35	DL	Pune	17100	sixty
19	108	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with `mydf` New interactive sheet

a) pd.to_numeric(..., errors='coerce', downcast=...)

- Safely parse messy numbers; bad parses → NaN (then you can fill/drop).

```
mydf.dtypes
```

```
                0
studentId   int64
     Name  object
      Age  object
   Course  object
     City  object
      Fee  object
    Marks  object

dtype: object
```

```
mydf['Age'] = pd.to_numeric(mydf['Age'], errors='coerce')
mydf['Fee'] = pd.to_numeric(mydf['Fee'], errors='coerce')
mydf['Marks'] = pd.to_numeric(mydf['Marks'], errors='coerce')

mydf.dtypes
```

```
                 0
studentId    int64
     Name   object
      Age  float64
   Course   object
     City   object
      Fee  float64
    Marks  float64

dtype: object
```

b) astype(...)

- Explicitly cast to a target dtype (use nullable types if NaNs exist).

```
# numerics
mydf['Age']   = mydf['Age'].astype('Int64')
mydf['Fee']   = mydf['Fee'].astype('Int64')
mydf['Marks'] = mydf['Marks'].astype('Float64')

# strings
mydf['Name'] = mydf['Name'].astype('string')
mydf['City'] = mydf['City'].astype('string')
mydf['Course'] = mydf['Course'].astype('string')

# mydf[['Name','City','Course']] = mydf[['Name','City','Course']].asty

# boolean
# mydf['HasFee'] = mydf['Fee'].notna().astype('boolean')

mydf['HasFee'] = mydf['Fee'].notna()
mydf['HasFee'] = mydf['HasFee'].astype('boolean')

mydf.dtypes
```

```
                         0
studentId            int64
     Name   string[python]
      Age            Int64
   Course   string[python]
     City   string[python]
      Fee            Int64
    Marks          Float64
   HasFee          boolean

dtype: object
```

```
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks	HasFee
0	101	srinivas	25	ML	Bangalore	<NA>	85.0	False
1	102	Vas	30	DevOps	Chennai	25000	90.0	True
2	103	Hello	<NA>	Java	Bangalore	15000	<NA>	True
3	104	Manish	40	Python	Mumbai	18000	<NA>	True
4	105	Amit	45	DL	<NA>	22000	88.5	True
5	106	Hai	30	ML	hyderabad	21000	72.0	True
6	107	Hello	35	DL	Pune	17000	<NA>	True
7	108	Vas	28	ai	Chennai	24000	95.0	True
8	109	RAJ	32	ML	Mumbai	19000	<NA>	True
9	110	Ok	26	<NA>	Delhi	16000	68.0	True
10	111	Alok	<NA>	Python	BANGALORE	23000	<NA>	True
11	112	Super	29	DevOps	Pune	<NA>	<NA>	False
12	104	Manish	40	Python	Mumbai	18000	<NA>	True
13	108	Vas	28	AI	Chennai	24500	96.0	True
14	113	Siri	27	ML	Bangalore	20000	<NA>	True
15	114	Kiran	<NA>	<NA>	<NA>	<NA>	<NA>	False
16	101	srinivas	25	ML	Bangalore	<NA>	85.0	False
17	109	RAJ	32	ML	Mumbai	19000	<NA>	True
18	115	Hello	35	DL	Pune	17100	<NA>	True
19	108	Vas	28	AI	Chennai	25000	92.0	True

Next steps: Generate code with mydf New interactive sheet

c) pd.to_datetime(..., errors='coerce')

- Parse date strings to datetime (coerce unparseable to NaT).

```
#mydf['Age'] = pd.to_numeric(mydf['Age'], errors='coerce')

# mydf['JoinDate'] = pd.to_datetime(mydf['JoinDate'], errors='coerce')
```

d) convert_dtypes()

- Auto-infer better dtypes (nullable Int64/Float64, string, boolean).

```
import pandas as pd

mydf = pd.read_csv("mystudents_data_1.csv")

mydf.dtypes
```

```
                  0
studentId     int64
     Name    object
      Age    object
   Course    object
     City    object
      Fee    object
    Marks    object

dtype: object
```

```
mydf = mydf.convert_dtypes()

mydf.dtypes
```

	0
studentId	Int64
Name	string[python]
Age	string[python]
Course	string[python]
City	string[python]
Fee	string[python]
Marks	string[python]

dtype: object

C) String Operations

- a) str.strip(), str.lstrip(), str.rstrip()
- b) str.contains(substring, case=False, na=False)
- c) str.startswith() / str.endswith()
- d) str.replace(old, new) (literal substring)
- e) str.split(..., expand=True)

```
mydf.dtypes
```

```
                       0
studentId          Int64
Name      string[python]
Age       string[python]
Course    string[python]
City      string[python]
Fee       string[python]
Marks     string[python]

dtype: object
```

a) Trim & Case - str.strip(), str.lstrip(), str.rstrip()

- Remove spaces at both/left/right ends.

```
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
1	102	Vas	30	DevOps	Chennai	25000	90
2	103	Hello	<NA>	Java	Bangalore	15000	<NA>
3	104	Manish	40	Python	Mumbai	18000	78%
4	105	Amit	45	DL	<NA>	22000	88.5
5	106	Hai	30	ML	hyderabad	21000	72
6	107	Hello	35	DL	Pune	17000	sixty
7	108	Vas	28	ai	Chennai	24000	95
8	109	RAJ	32	ML	Mumbai	19000	65/100
9	110	Ok	26	<NA>	Delhi	16000	68
10	111	Alok	38 years	Python	BANGALORE	23000	88,5
11	112	Super	29	DevOps	Pune	<NA>	<NA>
12	104	Manish	40	Python	Mumbai	18000	78%
13	108	Vas	28	AI	Chennai	24500	96
14	113	Siri	27	ML	Bangalore	20000	<NA>
15	114	Kiran	<NA>	<NA>	<NA>	<NA>	absent
16	101	srinivas	25	ML	Bangalore	20k	85
17	109	RAJ	32	ML	Mumbai	19000	65/100
18	115	Hello	35	DL	Pune	17100	sixty
19	108	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with mydf New interactive sheet

```
# Trim the Spaces and Make it to Title Ccase or Upper Case
mydf['City']   = mydf['City'].str.strip().str.title()   # 'BANGALORE
mydf['Course'] = mydf['Course'].str.strip().str.upper() # ' ai ' → 'A
mydf['Name']   = mydf['Name'].str.strip().str.title()   # '  srinivas

mydf[['Name','Course','City']]
```

	Name	Course	City
0	Srinivas	ML	Bangalore
1	Vas	DEVOPS	Chennai
2	Hello	JAVA	Bangalore
3	Manish	PYTHON	Mumbai
4	Amit	DL	<NA>
5	Hai	ML	Hyderabad
6	Hello	DL	Pune
7	Vas	AI	Chennai
8	Raj	ML	Mumbai
9	Ok	<NA>	Delhi
10	Alok	PYTHON	Bangalore
11	Super	DEVOPS	Pune
12	Manish	PYTHON	Mumbai
13	Vas	AI	Chennai
14	Siri	ML	Bangalore
15	Kiran	<NA>	<NA>
16	Srinivas	ML	Bangalore
17	Raj	ML	Mumbai
18	Hello	DL	Pune
19	Vas	AI	Chennai

b) Find / Filter str.contains(substring, case=False, na=False)

- Filter rows where substring appears (case-insensitive).

str.startswith() / str.endswith()

- Filter by prefix/suffix.

```
mydf['Name'].str.contains('as', case=False, na=False)
```

```
     Name
0    True
1    True
2   False
3   False
4   False
5   False
6   False
7    True
8   False
9   False
10  False
11  False
12  False
13   True
14  False
15  False
16   True
17  False
18  False
19   True

dtype: boolean
```

```
mydf[ mydf['Name'].str.contains('as', case=False, na=False) ]
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	Srinivas	25	ML	Bangalore	20k	85
1	102	Vas	30	DEVOPS	Chennai	25000	90
7	108	Vas	28	AI	Chennai	24000	95
13	108	Vas	28	AI	Chennai	24500	96
16	101	Srinivas	25	ML	Bangalore	20k	85
19	108	Vas	28	AI	Chennai	25000	92

```
mydf[ mydf['Name'].str.startswith('S', na=False) ]
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	Srinivas	25	ML	Bangalore	20k	85
11	112	Super	29	DEVOPS	Pune	<NA>	<NA>
14	113	Siri	27	ML	Bangalore	20000	<NA>
16	101	Srinivas	25	ML	Bangalore	20k	85

```
mydf[ mydf['Course'].str.endswith('N', na=False) ]
```

	studentId	Name	Age	Course	City	Fee	Marks
3	104	Manish	40	PYTHON	Mumbai	18000	78%
10	111	Alok	38 years	PYTHON	Bangalore	23000	88,5
12	104	Manish	40	PYTHON	Mumbai	18000	78%

c) Replace - str.replace(old, new) (literal substring)

- Fix common variants and typos.

```
mydf['City'] = mydf['City'].str.strip().str.replace('Bengaluru', 'Ban
mydf['Course'] = mydf['Course'].str.strip().str.replace('DEV   OPS',

mydf[['Name','Course','City']]
```

	Name	Course	City
0	Srinivas	ML	Bangalore
1	Vas	DEVOPS	Chennai
2	Hello	JAVA	Bangalore
3	Manish	PYTHON	Mumbai
4	Amit	DL	<NA>
5	Hai	ML	Hyderabad
6	Hello	DL	Pune
7	Vas	AI	Chennai
8	Raj	ML	Mumbai
9	Ok	<NA>	Delhi
10	Alok	PYTHON	Bangalore
11	Super	DEVOPS	Pune
12	Manish	PYTHON	Mumbai
13	Vas	AI	Chennai
14	Siri	ML	Bangalore
15	Kiran	<NA>	<NA>
16	Srinivas	ML	Bangalore
17	Raj	ML	Mumbai
18	Hello	DL	Pune
19	Vas	AI	Chennai

```
mydf['City'] = (mydf['City'].str.lower()
                             .str.replace('bangaluru', 'bangalore')
                             .str.title())
```

d) Split / Join - str.split(..., expand=True)

- Split into multiple columns.

```
parts = mydf['Name'].astype('string').str.strip().str.split(n=1)  # Se

mydf['FirstName'] = parts.str[0]   # always present
mydf['Rest']      = parts.str[1]   # becomes NaN when there was no sec

mydf
```

	studentId	Name	Age	Course	City	Fee	Marks	FirstName
0	101	Srinivas	25	ML	Bangalore	20k	85	Srinivas
1	102	Vas	30	DEVOPS	Chennai	25000	90	Vas
2	103	Hello	<NA>	JAVA	Bangalore	15000	<NA>	Hello
3	104	Manish	40	PYTHON	Mumbai	18000	78%	Manish
4	105	Amit	45	DL	<NA>	22000	88.5	Amit
5	106	Hai	30	ML	Hyderabad	21000	72	Hai
6	107	Hello	35	DL	Pune	17000	sixty	Hello
7	108	Vas	28	AI	Chennai	24000	95	Vas
8	109	Raj	32	ML	Mumbai	19000	65/100	Raj
9	110	Ok	26	<NA>	Delhi	16000	68	Ok
10	111	Alok	38 years	PYTHON	Bangalore	23000	88,5	Alok
11	112	Super	29	DEVOPS	Pune	<NA>	<NA>	Super
12	104	Manish	40	PYTHON	Mumbai	18000	78%	Manish
13	108	Vas	28	AI	Chennai	24500	96	Vas
14	113	Siri	27	ML	Bangalore	20000	<NA>	Siri
15	114	Kiran	<NA>	<NA>	<NA>	<NA>	absent	Kiran
16	101	Srinivas	25	ML	Bangalore	20k	85	Srinivas
17	109	Raj	32	ML	Mumbai	19000	65/100	Raj
18	115	Hello	35	DL	Pune	17100	sixty	Hello

Next steps: Generate code with mydf New interactive sheet

```
# Collapse multiple spaces to one via split→join.
mydf['Name'] = mydf['Name'].str.split().str.join(' ')
```

D) Duplicates Handling

- a) Count duplicates: value_counts()
- b) Exact duplicates - duplicated()
- c) Remove Duplicate keys

```python
import pandas as pd

mydf = pd.read_csv("mystudents_data_1.csv")
mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
1	102	Vas	30	DevOps	Chennai	25000	90
2	103	Hello	NaN	Java	Bangalore	15000	NaN
3	104	Manish	40	Python	Mumbai	18000	78%
4	105	Amit	45	DL	NaN	22000	88.5
5	106	Hai	30	ML	hyderabad	21000	72
6	107	Hello	35	DL	Pune	17000	sixty
7	108	Vas	28	ai	Chennai	24000	95
8	109	RAJ	32	ML	Mumbai	19000	65/100
9	110	Ok	26	NaN	Delhi	16000	68
10	111	Alok	38 years	Python	BANGALORE	23000	88,5
11	112	Super	29	DevOps	Pune	NaN	NaN
12	104	Manish	40	Python	Mumbai	18000	78%
13	108	Vas	28	AI	Chennai	24500	96
14	113	Siri	27	ML	Bangalore	20000	NaN
15	114	Kiran	NaN	NaN	NaN	NaN	absent
16	101	srinivas	25	ML	Bangalore	20k	85
17	109	RAJ	32	ML	Mumbai	19000	65/100
18	115	Hello	35	DL	Pune	17100	sixty
19	108	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with mydf New interactive sheet

a) Count duplicates: value_counts()

- Use value_counts() to see how many times each key or pair appears.

```
# counts per studentId
mydf['studentId'].value_counts()
```

```
           count
studentId
      108      3
      104      2
      101      2
      109      2
      105      1
      103      1
      102      1
      107      1
      106      1
      110      1
      111      1
      112      1
      113      1
      114      1
      115      1

dtype: int64
```

```
# counts per City
mydf.value_counts(subset=['City'])
```

```
           count
      City
 Bangalore     4
   Chennai     4
    Mumbai     4
      Pune     3
 BANGALORE     1
     Delhi     1
 hyderabad     1

dtype: int64
```

```
# counts per (Name, City) pair
mydf.value_counts(subset=['Name','City'])
```

```
                        count
   Name       City
   Vas        Chennai       4
   Hello      Pune          2
   srinivas   Bangalore     2
   RAJ        Mumbai        2
   Manish     Mumbai        2
   Hai        hyderabad     1
   Alok       BANGALORE     1
   Ok         Delhi         1
   Hello      Bangalore     1
   Siri       Bangalore     1
   Super      Pune          1

dtype: int64
```

b) Exact duplicates - duplicated()

- Detect the rows that are identical across every column.

```
dup_count = mydf.duplicated().sum()
print(dup_count)

mydf[ mydf.duplicated(keep=False) ]
```

```
3
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
3	104	Manish	40	Python	Mumbai	18000	78%
8	109	RAJ	32	ML	Mumbai	19000	65/100
12	104	Manish	40	Python	Mumbai	18000	78%
16	101	srinivas	25	ML	Bangalore	20k	85
17	109	RAJ	32	ML	Mumbai	19000	65/100

```
dup_count = mydf.duplicated(subset=['studentId'], keep=False).sum()
print(dup_count)

mydf[ mydf.duplicated(subset=['studentId'], keep=False) ]
```

```
9
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
3	104	Manish	40	Python	Mumbai	18000	78%
7	108	Vas	28	ai	Chennai	24000	95
8	109	RAJ	32	ML	Mumbai	19000	65/100
12	104	Manish	40	Python	Mumbai	18000	78%
13	108	Vas	28	AI	Chennai	24500	96
16	101	srinivas	25	ML	Bangalore	20k	85
17	109	RAJ	32	ML	Mumbai	19000	65/100
19	108	Vas	28	AI	Chennai	25000	92

c) Remove Duplicate keys

- drop_duplicates(subset=...).
- Find multiple rows sharing the key and keep the desired one

```
mydf = mydf.drop_duplicates(keep='first')

dup_count = mydf.duplicated().sum()
print(dup_count)

mydf
```

```
0
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
1	102	Vas	30	DevOps	Chennai	25000	90
2	103	Hello	NaN	Java	Bangalore	15000	NaN
3	104	Manish	40	Python	Mumbai	18000	78%
4	105	Amit	45	DL	NaN	22000	88.5
5	106	Hai	30	ML	hyderabad	21000	72
6	107	Hello	35	DL	Pune	17000	sixty
7	108	Vas	28	ai	Chennai	24000	95
8	109	RAJ	32	ML	Mumbai	19000	65/100
9	110	Ok	26	NaN	Delhi	16000	68
10	111	Alok	38 years	Python	BANGALORE	23000	88,5
11	112	Super	29	DevOps	Pune	NaN	NaN
13	108	Vas	28	AI	Chennai	24500	96
14	113	Siri	27	ML	Bangalore	20000	NaN
15	114	Kiran	NaN	NaN	NaN	NaN	absent
18	115	Hello	35	DL	Pune	17100	sixty
19	108	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with mydf New interactive sheet

```
mydf = mydf.drop_duplicates(subset=['studentId'], keep='last')

dup_count = mydf.duplicated(subset=['studentId'], keep=False).sum()
print(dup_count)

mydf
```

```
0
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
1	102	Vas	30	DevOps	Chennai	25000	90
2	103	Hello	NaN	Java	Bangalore	15000	NaN
3	104	Manish	40	Python	Mumbai	18000	78%
4	105	Amit	45	DL	NaN	22000	88.5
5	106	Hai	30	ML	hyderabad	21000	72
6	107	Hello	35	DL	Pune	17000	sixty
8	109	RAJ	32	ML	Mumbai	19000	65/100
9	110	Ok	26	NaN	Delhi	16000	68
10	111	Alok	38 years	Python	BANGALORE	23000	88,5
11	112	Super	29	DevOps	Pune	NaN	NaN
14	113	Siri	27	ML	Bangalore	20000	NaN
15	114	Kiran	NaN	NaN	NaN	NaN	absent
18	115	Hello	35	DL	Pune	17100	sixty
19	108	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with mydf New interactive sheet

E) Mapping and Replacing Values

- a) map(dict)
- b) map(dict)
- c) replace(old→new)
- d) DataFrame.replace({...})
- e) cut()
- f) apply(func)

```python
import pandas as pd

mydf = pd.read_csv("mystudents_data_1.csv")

mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	Bangalore	20k	85
1	102	Vas	30	DevOps	Chennai	25000	90
2	103	Hello	NaN	Java	Bangalore	15000	NaN
3	104	Manish	40	Python	Mumbai	18000	78%
4	105	Amit	45	DL	NaN	22000	88.5
5	106	Hai	30	ML	hyderabad	21000	72
6	107	Hello	35	DL	Pune	17000	sixty
7	108	Vas	28	ai	Chennai	24000	95
8	109	RAJ	32	ML	Mumbai	19000	65/100
9	110	Ok	26	NaN	Delhi	16000	68
10	111	Alok	38 years	Python	BANGALORE	23000	88,5
11	112	Super	29	DevOps	Pune	NaN	NaN
12	104	Manish	40	Python	Mumbai	18000	78%
13	108	Vas	28	AI	Chennai	24500	96
14	113	Siri	27	ML	Bangalore	20000	NaN
15	114	Kiran	NaN	NaN	NaN	NaN	absent
16	101	srinivas	25	ML	Bangalore	20k	85
17	109	RAJ	32	ML	Mumbai	19000	65/100
18	115	Hello	35	DL	Pune	17100	sixty
19	108	Vas	28	AI	Chennai	25000	92

Next steps: Generate code with mydf New interactive sheet

a) map(dict)

- Best for one-column lookups;
- Unmapped → NaN (so often followed by fillna).

```
city2code = {'Bangalore':'BLR','Hyderabad':'HYD','Chennai':'CNA','Mumb

mydf['City'] = mydf['City'].map(city2code).fillna(np.nan)

mydf
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	BLR	20k	85
1	102	Vas	30	DevOps	CNA	25000	90
2	103	Hello	NaN	Java	BLR	15000	NaN
3	104	Manish	40	Python	MUM	18000	78%
4	105	Amit	45	DL	NaN	22000	88.5
5	106	Hai	30	ML	NaN	21000	72
6	107	Hello	35	DL	PUN	17000	sixty
7	108	Vas	28	ai	CNA	24000	95
8	109	RAJ	32	ML	MUM	19000	65/100
9	110	Ok	26	NaN	DEL	16000	68
10	111	Alok	38 years	Python	NaN	23000	88,5
11	112	Super	29	DevOps	PUN	NaN	NaN
12	104	Manish	40	Python	MUM	18000	78%
13	108	Vas	28	AI	CNA	24500	96
14	113	Siri	27	ML	BLR	20000	NaN
15	114	Kiran	NaN	NaN	NaN	NaN	absent
16	101	srinivas	25	ML	BLR	20k	85
17	109	RAJ	32	ML	MUM	19000	65/100
18	115	Hello	35	DL	PUN	17100	sixty
19	108	Vas	28	AI	CNA	25000	92

Next steps: Generate code with `mydf` New interactive sheet

b) map(dict)

- Keep original when no mapping exists

```
course_fix = {'DEV   OPS':'DEVOPS','ai':'AI','PY':'PYTHON'}

mydf['Course'] = mydf['Course'].map(course_fix).fillna(mydf['Course'])

mydf['Course']
```

```
     Course
 0       ML
 1   DevOps
 2     Java
 3   Python
 4       DL
 5       ML
 6       DL
 7       ai
 8       ML
 9      NaN
10   Python
11   DevOps
12   Python
13       AI
14       ML
15      NaN
16       ML
17       ML
18       DL
19       AI

dtype: object
```

c) replace(old→new)

- Change only the matched values; others untouched.

```
mydf['Name'] = mydf['Name'].replace({'Ok':'Okay','Hai':'Hi'})

mydf['Name']
```

```
        Name
 0  srinivas
 1       Vas
 2     Hello
 3    Manish
 4      Amit
 5        Hi
 6     Hello
 7       Vas
 8       RAJ
 9      Okay
10      Alok
11     Super
12    Manish
13       Vas
14      Siri
15     Kiran
16  srinivas
17       RAJ
18     Hello
19       Vas

dtype: object
```

d) DataFrame.replace({...})

- Multi-column replace
- One call to fix multiple columns.

```
mydf = mydf.replace({
    'City':   {'Bengaluru':'Bangalore', 'Hyd':'Hyderabad'},
    'Course': {'Dev Ops':'DEVOPS', 'Js':'JAVA'}
})

mydf.head()
```

	studentId	Name	Age	Course	City	Fee	Marks
0	101	srinivas	25	ML	BLR	20k	85
1	102	Vas	30	DevOps	CNA	25000	90
2	103	Hello	NaN	Java	BLR	15000	NaN
3	104	Manish	40	Python	MUM	18000	78%
4	105	Amit	45	DL	NaN	22000	88.5

Next steps: Generate code with mydf New interactive sheet

e) cut()

- Bins continuous values into labeled intervals and returns an ordered categorical series.

```python
mydf = mydf.copy()
mydf['Marks_num'] = pd.to_numeric(mydf['Marks'], errors='coerce')

mybins   = [0, 70, 80, 90, 100]            # 0–70, 70–80, 80–90, 90–
mylabels = ['D','C','B','A']

mydf['Grade1'] = pd.cut(mydf['Marks_num'], bins=mybins, labels=mylabe

mydf[["Name","Marks_num","Grade1"]]
```

```
        Name  Marks_num  Grade1
 0  srinivas       85.0       B
 1       Vas       90.0       A
 2     Hello        NaN     NaN
 3    Manish        NaN     NaN
 4      Amit       88.5       B
 5        Hi       72.0       C
 6     Hello        NaN     NaN
 7       Vas       95.0       A
 8       RAJ        NaN     NaN
 9      Okay       68.0       D
10      Alok        NaN     NaN
11     Super        NaN     NaN
12    Manish        NaN     NaN
13       Vas       96.0       A
14      Siri        NaN     NaN
15     Kiran        NaN     NaN
16  srinivas       85.0       B
17       RAJ        NaN     NaN
18     Hello        NaN     NaN
19       Vas       92.0       A
```

f) apply(func)

- Runs a custom Python function over a Series (element-wise) or a DataFrame (row/column-wise)

```
def find_grade(marks):
    if pd.isna(marks): return pd.NA
    if marks >= 90: return 'A'
    if marks >= 80: return 'B'
    if marks >= 70: return 'C'
    return 'D'

mydf['Grade2'] = mydf['Marks_num'].apply(find_grade)

mydf[["Name", 'Marks_num','Grade1','Grade2']].head()
```

	Name	Marks_num	Grade1	Grade2
0	srinivas	85.0	B	B
1	Vas	90.0	A	A
2	Hello	NaN	NaN	<NA>
3	Manish	NaN	NaN	<NA>
4	Amit	88.5	B	B

```
def make_label(row):
    # be robust to missing values and mixed cases
    name  = str(row['Name']).strip().title()  if pd.notna(row['Name']]
    course= str(row['Course']).strip().upper() if pd.notna(row['Course
    city  = str(row['City']).strip().upper()  if pd.notna(row['City']]
    return f"{name} | {course} @ {city}"

# Row-wise apply because we use multiple columns
mydf['Label'] = mydf.apply(make_label, axis=1)

mydf[['Name','Course','City','Label']].head()
```

	Name	Course	City	Label
0	srinivas	ML	BLR	Srinivas | ML @ BLR
1	Vas	DevOps	CNA	Vas | DEVOPS @ CNA
2	Hello	Java	BLR	Hello | JAVA @ BLR
3	Manish	Python	MUM	Manish | PYTHON @ MUM
4	Amit	DL	NaN	Amit | DL @ NA

Start coding or generate with AI.

```
mydf[mydf["Marks_num"].notna()][['Name','Marks_num']]
```

	Name	Marks_num
0	srinivas	85.0
1	Vas	90.0
4	Amit	88.5
5	Hi	72.0
7	Vas	95.0
9	Okay	68.0
13	Vas	96.0
16	srinivas	85.0
19	Vas	92.0

```
mydf["Marks_Updated"] = mydf["Marks_num"].apply(lambda x: x+10 if x<9
```

```
mydf[mydf["Marks_num"].notna()][['Name','Marks_num','Marks_Updated']]
```

```
         Name  Marks_num  Marks_Updated
 0   srinivas       85.0           95.0
 1        Vas       90.0           90.0
 4       Amit       88.5           98.5
 5         Hi       72.0           82.0
 7        Vas       95.0           95.0
 9       Okay       68.0           78.0
13        Vas       96.0           96.0
16   srinivas       85.0           95.0
19        Vas       92.0           92.0
```