In [32]:
```
import numpy as np
import pandas as pd
```

In [33]:
```
# read the dataset
df = pd.read_csv('pandas_practice_dataset.csv')
```

In [34]:
```
# show the first 5 rows
df.head()
```

Out[34]:

	id	name	age	department	salary
0	1	Ali	25	IT	5000
1	2	Sara	30	HR	6000
2	3	John	22	IT	4500
3	4	Mona	28	Finance	7000
4	5	Omar	35	IT	8000

In [35]:
```
# show the last 5 rows
df.tail()
```

Out[35]:

	id	name	age	department	salary
15	16	Nada	32	IT	6200
16	17	Karim	27	HR	5600
17	18	Huda	36	Sales	5900
18	19	Samir	34	IT	6100
19	20	Rania	28	Finance	7500

In [36]:
```
# show random 5 rows
df.sample(5)
```

Out[36]:

	id	name	age	department	salary
13	14	Salma	21	Sales	4700
3	4	Mona	28	Finance	7000
7	8	Nour	29	Finance	7200
16	17	Karim	27	HR	5600
0	1	Ali	25	IT	5000

In [37]:
```
# get the shape of the dataset shape = (rows, columns)
df.shape
```

Out[37]:
```
(20, 5)
```

In [38]:
```
# show the column names
df.columns
```

Out[38]:
```
Index(['id', 'name', 'age', 'department', 'salary'], dtype='object')
```

In [39]:
```
# show the row names
df.index
```

Out[39]:
```
RangeIndex(start=0, stop=20, step=1)
```

In [40]:
```
# See data types + null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id          20 non-null     int64 
 1   name        20 non-null     object
 2   age         20 non-null     int64 
 3   department  20 non-null     object
 4   salary      20 non-null     int64 
dtypes: int64(3), object(2)
memory usage: 932.0+ bytes
```

In [41]:
```
# Select ONE column
df['name'].head()
```

Out[41]:
```
0     Ali
1    Sara
2    John
3    Mona
4    Omar
Name: name, dtype: object
```

In [42]:
```
# select multiple column
df[["name", "salary"]].head()
```

Out[42]:

	name	salary
0	Ali	5000
1	Sara	6000
2	John	4500
3	Mona	7000
4	Omar	8000

In [43]:
```
# Select row by index
df.iloc[0]
```

Out[43]:
```
id               1
name           Ali
age             25
department      IT
salary        5000
Name: 0, dtype: object
```

In [44]:
```
# Select specific row & column
df.loc[0, "name"]
```

Out[44]:
```
'Ali'
```

In [45]:

```
# Employees older than 30
df[df["age"] > 30]
```

Out[45]:

	id	name	age	department	salary
4	5	Omar	35	IT	8000
8	9	Khaled	31	Sales	5100
10	11	Hassan	40	Management	9000
12	13	Tarek	33	Finance	6800
14	15	Mostafa	38	Management	8500
15	16	Nada	32	IT	6200
17	18	Huda	36	Sales	5900
18	19	Samir	34	IT	6100

In [46]:

```
# Employees in IT department
df[df["department"] == "IT"]
```

Out[46]:

	id	name	age	department	salary
0	1	Ali	25	IT	5000
2	3	John	22	IT	4500
4	5	Omar	35	IT	8000
9	10	Dina	26	IT	5300
15	16	Nada	32	IT	6200
18	19	Samir	34	IT	6100

In [47]:
```
# Multiple conditions
df[(df["age"] > 25) & (df["salary"] > 6000)]
```

Out[47]:

	id	name	age	department	salary
3	4	Mona	28	Finance	7000
4	5	Omar	35	IT	8000
7	8	Nour	29	Finance	7200
10	11	Hassan	40	Management	9000
12	13	Tarek	33	Finance	6800
14	15	Mostafa	38	Management	8500
15	16	Nada	32	IT	6200
18	19	Samir	34	IT	6100
19	20	Rania	28	Finance	7500

In [48]:

```
# Sort by salary
df.sort_values("salary")
```

Out[48]:

	id	name	age	department	salary
2	3	John	22	IT	4500
11	12	Yara	23	HR	4600
13	14	Salma	21	Sales	4700
6	7	Ahmed	24	Sales	4800
0	1	Ali	25	IT	5000
8	9	Khaled	31	Sales	5100
9	10	Dina	26	IT	5300
5	6	Lina	27	HR	5500
16	17	Karim	27	HR	5600
17	18	Huda	36	Sales	5900
1	2	Sara	30	HR	6000
18	19	Samir	34	IT	6100
15	16	Nada	32	IT	6200
12	13	Tarek	33	Finance	6800
3	4	Mona	28	Finance	7000
7	8	Nour	29	Finance	7200
19	20	Rania	28	Finance	7500
4	5	Omar	35	IT	8000
14	15	Mostafa	38	Management	8500
10	11	Hassan	40	Management	9000

In [49]:
```
# Sort descending
df.sort_values("salary", ascending=False)
```

Out[49]:

	id	name	age	department	salary
10	11	Hassan	40	Management	9000
14	15	Mostafa	38	Management	8500
4	5	Omar	35	IT	8000
19	20	Rania	28	Finance	7500
7	8	Nour	29	Finance	7200
3	4	Mona	28	Finance	7000
12	13	Tarek	33	Finance	6800
15	16	Nada	32	IT	6200
18	19	Samir	34	IT	6100
1	2	Sara	30	HR	6000
17	18	Huda	36	Sales	5900
16	17	Karim	27	HR	5600
5	6	Lina	27	HR	5500
9	10	Dina	26	IT	5300
8	9	Khaled	31	Sales	5100
0	1	Ali	25	IT	5000
6	7	Ahmed	24	Sales	4800
13	14	Salma	21	Sales	4700
11	12	Yara	23	HR	4600
2	3	John	22	IT	4500

In [50]:
```
# Average salary
df["salary"].mean()
```

Out[50]:
```
np.float64(6165.0)
```

In [51]:
```
# Maximum salary
df["salary"].max()
```

Out[51]:
```
9000
```

In [52]:
```
# Minimum salary
df["salary"].min()
```

Out[52]:
```
4500
```

In [56]:
```
# Add new column
df["bonus"] = df["salary"] * 0.10
df.head()
```

Out[56]:

	id	name	age	department	salary	bonus
0	1	Ali	25	IT	5000	500.0
1	2	Sara	30	HR	6000	600.0
2	3	John	22	IT	4500	450.0
3	4	Mona	28	Finance	7000	700.0
4	5	Omar	35	IT	8000	800.0

In [57]:
```
# Increase age by 1 year
df["age"] = df["age"] + 1
df.head()
```

Out[57]:

	id	name	age	department	salary	bonus
0	1	Ali	26	IT	5000	500.0
1	2	Sara	31	HR	6000	600.0
2	3	John	23	IT	4500	450.0
3	4	Mona	29	Finance	7000	700.0
4	5	Omar	36	IT	8000	800.0

In [58]:
```
# Average salary per department
df.groupby("department")["salary"].mean()
```

Out[58]:
```
department
Finance       7125.0
HR            5425.0
IT            5850.0
Management    8750.0
Sales         5125.0
Name: salary, dtype: float64
```

In [59]:
```
# Count employees per department
df.groupby("department")["id"].count()
```

Out[59]:
```
department
Finance       4
HR            4
IT            6
Management    2
Sales         4
Name: id, dtype: int64
```

In [60]:
```
# Save to new CSV
df.to_csv("output.csv", index=False)
```

In [61]:
```
# Show employees from IT department
df[df["department"] == "IT"]
```

Out[61]:

	id	name	age	department	salary	bonus
0	1	Ali	26	IT	5000	500.0
2	3	John	23	IT	4500	450.0
4	5	Omar	36	IT	8000	800.0
9	10	Dina	27	IT	5300	530.0
15	16	Nada	33	IT	6200	620.0
18	19	Samir	35	IT	6100	610.0

In [62]:
```
# Find the highest salary
df[df["salary"] == df["salary"].max()]
```

Out[62]:

	id	name	age	department	salary	bonus
10	11	Hassan	41	Management	9000	900.0

In [63]:
```
# Find employees older than 30
df[df["age"] > 30]
```

Out[63]:

	id	name	age	department	salary	bonus
1	2	Sara	31	HR	6000	600.0
4	5	Omar	36	IT	8000	800.0
8	9	Khaled	32	Sales	5100	510.0
10	11	Hassan	41	Management	9000	900.0
12	13	Tarek	34	Finance	6800	680.0
14	15	Mostafa	39	Management	8500	850.0
15	16	Nada	33	IT	6200	620.0
17	18	Huda	37	Sales	5900	590.0
18	19	Samir	35	IT	6100	610.0

In [64]:
```
# Calculate average salary
df["salary"].mean()
```

Out[64]:
```
np.float64(6165.0)
```

In [65]:
```
# Sort employees by age (descending)
df.sort_values("age", ascending=False)
```

Out[65]:

	id	name	age	department	salary	bonus
10	11	Hassan	41	Management	9000	900.0
14	15	Mostafa	39	Management	8500	850.0
17	18	Huda	37	Sales	5900	590.0
4	5	Omar	36	IT	8000	800.0
18	19	Samir	35	IT	6100	610.0
12	13	Tarek	34	Finance	6800	680.0
15	16	Nada	33	IT	6200	620.0
8	9	Khaled	32	Sales	5100	510.0
1	2	Sara	31	HR	6000	600.0
7	8	Nour	30	Finance	7200	720.0
3	4	Mona	29	Finance	7000	700.0
19	20	Rania	29	Finance	7500	750.0
16	17	Karim	28	HR	5600	560.0
5	6	Lina	28	HR	5500	550.0
9	10	Dina	27	IT	5300	530.0
0	1	Ali	26	IT	5000	500.0
6	7	Ahmed	25	Sales	4800	480.0
11	12	Yara	24	HR	4600	460.0
2	3	John	23	IT	4500	450.0
13	14	Salma	22	Sales	4700	470.0

In [66]:
```
# Highest salary per department
df.groupby("department")["salary"].max()
```

Out[66]:
```
department
Finance       7500
HR            6000
IT            8000
Management    9000
Sales         5900
Name: salary, dtype: int64
```

In [67]:
```
# Count employees in each department
df["department"].value_counts()
```

Out[67]:
```
department
IT            6
HR            4
Finance       4
Sales         4
Management    2
Name: count, dtype: int64
```

In [68]:
```
# Show employees earning above average
df[df["salary"] > df["salary"].mean()]
```

Out[68]:

	id	name	age	department	salary	bonus
3	4	Mona	29	Finance	7000	700.0
4	5	Omar	36	IT	8000	800.0
7	8	Nour	30	Finance	7200	720.0
10	11	Hassan	41	Management	9000	900.0
12	13	Tarek	34	Finance	6800	680.0
14	15	Mostafa	39	Management	8500	850.0
15	16	Nada	33	IT	6200	620.0
19	20	Rania	29	Finance	7500	750.0

In [70]:
```
# Check if missing exists, this will show True/False values
df.isnull()
```

Out[70]:

	id	name	age	department	salary	bonus
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False
8	False	False	False	False	False	False
9	False	False	False	False	False	False
10	False	False	False	False	False	False
11	False	False	False	False	False	False
12	False	False	False	False	False	False
13	False	False	False	False	False	False
14	False	False	False	False	False	False
15	False	False	False	False	False	False
16	False	False	False	False	False	False
17	False	False	False	False	False	False
18	False	False	False	False	False	False
19	False	False	False	False	False	False

In [71]:
```
# Count missing values per column
df.isnull().sum()
```

Out[71]:
```
id            0
name          0
age           0
department    0
salary        0
bonus         0
dtype: int64
```

In [72]:
```
# Drop rows with missing data
df.dropna()
```

Out[72]:

	id	name	age	department	salary	bonus
0	1	Ali	26	IT	5000	500.0
1	2	Sara	31	HR	6000	600.0
2	3	John	23	IT	4500	450.0
3	4	Mona	29	Finance	7000	700.0
4	5	Omar	36	IT	8000	800.0
5	6	Lina	28	HR	5500	550.0
6	7	Ahmed	25	Sales	4800	480.0
7	8	Nour	30	Finance	7200	720.0
8	9	Khaled	32	Sales	5100	510.0
9	10	Dina	27	IT	5300	530.0
10	11	Hassan	41	Management	9000	900.0
11	12	Yara	24	HR	4600	460.0
12	13	Tarek	34	Finance	6800	680.0
13	14	Salma	22	Sales	4700	470.0
14	15	Mostafa	39	Management	8500	850.0
15	16	Nada	33	IT	6200	620.0
16	17	Karim	28	HR	5600	560.0
17	18	Huda	37	Sales	5900	590.0
18	19	Samir	35	IT	6100	610.0
19	20	Rania	29	Finance	7500	750.0

In [73]:
```
# Drop only if specific column is missing
df.dropna(subset=["salary"])
```

Out[73]:

	id	name	age	department	salary	bonus
0	1	Ali	26	IT	5000	500.0
1	2	Sara	31	HR	6000	600.0
2	3	John	23	IT	4500	450.0
3	4	Mona	29	Finance	7000	700.0
4	5	Omar	36	IT	8000	800.0
5	6	Lina	28	HR	5500	550.0
6	7	Ahmed	25	Sales	4800	480.0
7	8	Nour	30	Finance	7200	720.0
8	9	Khaled	32	Sales	5100	510.0
9	10	Dina	27	IT	5300	530.0
10	11	Hassan	41	Management	9000	900.0
11	12	Yara	24	HR	4600	460.0
12	13	Tarek	34	Finance	6800	680.0
13	14	Salma	22	Sales	4700	470.0
14	15	Mostafa	39	Management	8500	850.0
15	16	Nada	33	IT	6200	620.0
16	17	Karim	28	HR	5600	560.0
17	18	Huda	37	Sales	5900	590.0
18	19	Samir	35	IT	6100	610.0
19	20	Rania	29	Finance	7500	750.0

In [74]:
```
# Drop columns with missing data
df.dropna(axis=1)
```

Out[74]:

	id	name	age	department	salary	bonus
0	1	Ali	26	IT	5000	500.0
1	2	Sara	31	HR	6000	600.0
2	3	John	23	IT	4500	450.0
3	4	Mona	29	Finance	7000	700.0
4	5	Omar	36	IT	8000	800.0
5	6	Lina	28	HR	5500	550.0
6	7	Ahmed	25	Sales	4800	480.0
7	8	Nour	30	Finance	7200	720.0
8	9	Khaled	32	Sales	5100	510.0
9	10	Dina	27	IT	5300	530.0
10	11	Hassan	41	Management	9000	900.0
11	12	Yara	24	HR	4600	460.0
12	13	Tarek	34	Finance	6800	680.0
13	14	Salma	22	Sales	4700	470.0
14	15	Mostafa	39	Management	8500	850.0
15	16	Nada	33	IT	6200	620.0
16	17	Karim	28	HR	5600	560.0
17	18	Huda	37	Sales	5900	590.0
18	19	Samir	35	IT	6100	610.0
19	20	Rania	29	Finance	7500	750.0

In [75]:
```
# Fill with a fixed value
df.fillna(0)
```

Out[75]:

	id	name	age	department	salary	bonus
0	1	Ali	26	IT	5000	500.0
1	2	Sara	31	HR	6000	600.0
2	3	John	23	IT	4500	450.0
3	4	Mona	29	Finance	7000	700.0
4	5	Omar	36	IT	8000	800.0
5	6	Lina	28	HR	5500	550.0
6	7	Ahmed	25	Sales	4800	480.0
7	8	Nour	30	Finance	7200	720.0
8	9	Khaled	32	Sales	5100	510.0
9	10	Dina	27	IT	5300	530.0
10	11	Hassan	41	Management	9000	900.0
11	12	Yara	24	HR	4600	460.0
12	13	Tarek	34	Finance	6800	680.0
13	14	Salma	22	Sales	4700	470.0
14	15	Mostafa	39	Management	8500	850.0
15	16	Nada	33	IT	6200	620.0
16	17	Karim	28	HR	5600	560.0
17	18	Huda	37	Sales	5900	590.0
18	19	Samir	35	IT	6100	610.0
19	20	Rania	29	Finance	7500	750.0

In [78]:
```
# Fill with mean (very common)
df["salary"].fillna(df["salary"].mean())
```

Out[78]:
```
0     5000
1     6000
2     4500
3     7000
4     8000
5     5500
6     4800
7     7200
8     5100
9     5300
10    9000
11    4600
12    6800
13    4700
14    8500
15    6200
16    5600
17    5900
18    6100
19    7500
Name: salary, dtype: int64
```

In [80]:
```
# Fill with most frequent value
df["department"].fillna(df["department"].mode()[0])
```

Out[80]:
```
0             IT
1             HR
2             IT
3        Finance
4             IT
5             HR
6          Sales
7        Finance
8          Sales
9             IT
10    Management
11            HR
12       Finance
13         Sales
14    Management
15            IT
16            HR
17         Sales
18            IT
19       Finance
Name: department, dtype: object
```

In [81]:
```
# Check duplicates
df.duplicated()
```

Out[81]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
dtype: bool
```

In [82]:
```
# Count duplicates
df.duplicated().sum()
```

Out[82]:
```
np.int64(0)
```

In [83]:
```
# Remove duplicates
df.drop_duplicates(inplace=True)
```

In [84]:
```
# Remove duplicates based on column
df.drop_duplicates(subset=["id"])
```

Out[84]:

	id	name	age	department	salary	bonus
0	1	Ali	26	IT	5000	500.0
1	2	Sara	31	HR	6000	600.0
2	3	John	23	IT	4500	450.0
3	4	Mona	29	Finance	7000	700.0
4	5	Omar	36	IT	8000	800.0
5	6	Lina	28	HR	5500	550.0
6	7	Ahmed	25	Sales	4800	480.0
7	8	Nour	30	Finance	7200	720.0
8	9	Khaled	32	Sales	5100	510.0
9	10	Dina	27	IT	5300	530.0
10	11	Hassan	41	Management	9000	900.0
11	12	Yara	24	HR	4600	460.0
12	13	Tarek	34	Finance	6800	680.0
13	14	Salma	22	Sales	4700	470.0
14	15	Mostafa	39	Management	8500	850.0
15	16	Nada	33	IT	6200	620.0
16	17	Karim	28	HR	5600	560.0
17	18	Huda	37	Sales	5900	590.0
18	19	Samir	35	IT	6100	610.0
19	20	Rania	29	Finance	7500	750.0

In [85]:
```
# Remove extra spaces
df["name"] = df["name"].str.strip()
```

In [86]:
```
# Convert to lowercase
df["department"] = df["department"].str.lower()
```

In [87]:
```
# Convert to uppercase
df["department"] = df["department"].str.upper()
```

In [88]:
```
# Check data types
df.dtypes
```

Out[88]:
```
id              int64
name           object
age             int64
department     object
salary          int64
bonus         float64
dtype: object
```

In [89]:
```
# Convert to integer
df["age"] = df["age"].astype(int)
```

In [91]:
```
# Convert to datetime (VERY IMPORTANT)
# df["date"] = pd.to_datetime(df["date"])
```

In [93]:
```
# Replace values
df["department"].replace("IT Dept", "IT")
```

Out[93]:
```
0             IT
1             HR
2             IT
3        FINANCE
4             IT
5             HR
6          SALES
7        FINANCE
8          SALES
9             IT
10    MANAGEMENT
11            HR
12       FINANCE
13         SALES
14    MANAGEMENT
15            IT
16            HR
17         SALES
18            IT
19       FINANCE
Name: department, dtype: object
```

In [95]:
```
# Replace multiple values
df["department"].replace(["HR Dept", "Human Resources"], "HR")
```

Out[95]:
```
0             IT
1             HR
2             IT
3        FINANCE
4             IT
5             HR
6          SALES
7        FINANCE
8          SALES
9             IT
10    MANAGEMENT
11            HR
12       FINANCE
13         SALES
14    MANAGEMENT
15            IT
16            HR
17         SALES
18            IT
19       FINANCE
Name: department, dtype: object
```

In [97]:
```
# Example: remove extremely high salaries
df = df[df["salary"] < 100000]
df.head()
```

Out[97]:

	id	name	age	department	salary	bonus
0	1	Ali	26	IT	5000	500.0
1	2	Sara	31	HR	6000	600.0
2	3	John	23	IT	4500	450.0
3	4	Mona	29	FINANCE	7000	700.0
4	5	Omar	36	IT	8000	800.0

In [99]:
```
# Save Cleaned Data
df.to_csv("cleaned_data.csv", index=False)
```

In [100]:
```
# Check how many missing values exist
df.isnull().sum()
```

Out[100]:
```
id            0
name          0
age           0
department    0
salary        0
bonus         0
dtype: int64
```

In [102]:
```
# Fill missing salaries with the average salary
df["salary"].fillna(df["salary"].mean())
```

Out[102]:
```
0     5000
1     6000
2     4500
3     7000
4     8000
5     5500
6     4800
7     7200
8     5100
9     5300
10    9000
11    4600
12    6800
13    4700
14    8500
15    6200
16    5600
17    5900
18    6100
19    7500
Name: salary, dtype: int64
```

In [103]:
```
# Remove duplicate rows
df.drop_duplicates(inplace=True)
```

In [104]:
```
# Convert department names to uppercase
df["department"] = df["department"].str.upper()
```

In [105]:
```
# Remove extra spaces from names
df["name"] = df["name"].str.strip()
```

In [107]:

```
# Full Cleaning Pipeline (REAL ETL STYLE)
df = pd.read_csv("pandas_practice_dataset.csv")

# Check data
df.info()

# Handle missing values
df["salary"].fillna(df["salary"].mean())

# Remove duplicates
df.drop_duplicates(inplace=True)

# Clean text
df["name"] = df["name"].str.strip()
df["department"] = df["department"].str.upper()

# Save cleaned data
df.to_csv("cleaned_data.csv", index=False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id          20 non-null     int64 
 1   name        20 non-null     object
 2   age         20 non-null     int64 
 3   department  20 non-null     object
 4   salary      20 non-null     int64 
dtypes: int64(3), object(2)
memory usage: 932.0+ bytes
```

In [ ]: