**Cell 1**

python

CopyEdit

import pandas as pd

import matplotlib.pyplot as plt

**Problem Step**: Import all required Python libraries.

**Definition**:

* import pandas as pd: Imports **pandas**, a powerful library for working with structured (tabular) data.
* import matplotlib.pyplot as plt: Imports **matplotlib.pyplot**, a plotting module for visualizing data.

**Why**:

* pandas is essential for reading and analyzing the dataset.
* matplotlib.pyplot is used later for creating plots like bar charts and histograms for data visualization.

**Cell 2**

python

CopyEdit

df = pd.read\_csv('datasets/employee\_2.csv')

**Problem Step**: Load the dataset into a DataFrame.

**Definition**:

* pd.read\_csv(): Reads a CSV (comma-separated values) file into a **pandas DataFrame**.

**Why**:

* This command loads your dataset so you can start analyzing and manipulating the data.

**Cell 3**

python

CopyEdit

df.head()

**Problem Step**: Preview the data.

**Definition**:

* df.head(): Displays the **first five rows** of the DataFrame.

**Why**:

* Helps get a quick look at the data structure, columns, and values to verify successful loading.

**Cell 4**

python

CopyEdit

df.info()

**Problem Step**: Understand data types and missing values.

**Definition**:

* df.info(): Shows **column names**, **data types**, and **non-null counts**.

**Why**:

* Provides a quick summary of the dataset's structure and helps identify columns that may need cleaning.

**Cell 5**

python

CopyEdit

df.describe()

**Problem Step**: Get summary statistics.

**Definition**:

* df.describe(): Returns descriptive statistics (mean, std, min, max, etc.) for numeric columns.

**Why**:

* Helps understand data distribution and detect potential outliers or issues.

**Cell 6**

python

CopyEdit

df['Salary'].median()

**Problem Step**: Find the median salary.

**Definition**:

* df['Salary'].median(): Calculates the **median** of the "Salary" column.

**Why**:

* The median gives a central value of salaries, useful when data has outliers.

**Cell 7**

python

CopyEdit

columns=['Age','Salary','ExperienceInCurrentDomain']

df[columns].cov()

**Problem Step**: Check relationships between numeric variables.

**Definition**:

* df[columns].cov(): Computes the **covariance** between selected numeric columns.

**Why**:

* Covariance helps identify how variables change together—useful for detecting potential relationships.

**Cell 8**

python

CopyEdit

grouped = df.groupby('Gender')['Salary'].mean()

grouped

**Problem Step**: Analyze average salary by gender.

**Definition**:

* groupby('Gender'): Groups the data by the "Gender" column.
* ['Salary'].mean(): Calculates the **mean salary** per gender group.

**Why**:

* To compare salary averages between male and female employees.

**Cell 9**

python

CopyEdit

grouped = df.groupby(['Gender','Education'])['Salary'].agg(['min','max'])

grouped

**Problem Step**: Compare salary range by gender and education.

**Definition**:

* groupby(['Gender','Education']): Groups by both gender and education.
* agg(['min', 'max']): Calculates **minimum** and **maximum** salary per group.

**Why**:

* Useful to analyze salary distribution across combinations of gender and education levels.

**Cell 10**

python

CopyEdit

df.groupby('Education').size()

**Problem Step**: Count records by education level.

**Definition**:

* groupby('Education').size(): Counts the number of entries per education group.

**Why**:

* To understand the frequency distribution of education levels in the dataset.

**Cell 11**

python

CopyEdit

df['Gender'].value\_counts().plot(kind='bar')

**Problem Step**: Visualize gender distribution.

**Definition**:

* value\_counts(): Counts unique values in the "Gender" column.
* plot(kind='bar'): Creates a **bar chart** of the counts.

**Why**:

* To visually compare the number of male and female employees.

**Cell 12**

python

CopyEdit

plt.hist(df['Salary'], bins=10, edgecolor='black')

**Problem Step**: Visualize salary distribution.

**Definition**:

* plt.hist(): Creates a **histogram**, dividing salaries into 10 bins.

**Why**:

* To understand how salary values are distributed (normal, skewed, etc.).

**Cell 13**

python

CopyEdit

df['City'].value\_counts().plot(kind='bar')

**Problem Step**: Visualize number of employees per city.

**Definition**:

* value\_counts() counts occurrences of each city.
* plot(kind='bar') creates a bar chart.

**Why**:

* To visualize geographic distribution of employees.