**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 module for creating visualizations.

**Why**:

* pandas is used for data loading and manipulation.
* matplotlib.pyplot is used to plot data (e.g., bar charts, histograms) for analysis and 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 file and stores it in a DataFrame.

**Why**:

* To load employee data from the file for further processing and analysis.

**Cell 3**

python

CopyEdit

df.head()

**Problem Step**: Preview the data.

**Definition**:

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

**Why**:

* To get a quick look at the dataset structure and verify successful loading.

**Cell 4**

python

CopyEdit

df.info()

**Problem Step**: Check data structure and missing values.

**Definition**:

* df.info(): Outputs summary info including data types and non-null counts.

**Why**:

* Helps in identifying data types and missing values, crucial for cleaning and preprocessing.

**Cell 5**

python

CopyEdit

df.describe()

**Problem Step**: Generate summary statistics.

**Definition**:

* df.describe(): Provides count, mean, std, min, and max for numeric columns.

**Why**:

* Helps understand distributions and potential outliers in numerical data.

**Cell 6**

python

CopyEdit

df['Salary'].median()

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

**Definition**:

* df['Salary'].median(): Returns the middle value of the "Salary" column.

**Why**:

* The median is a robust measure of central tendency, less affected by outliers than the mean.

**Cell 7**

python

CopyEdit

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

df[columns].cov()

**Problem Step**: Analyze relationships between numerical variables.

**Definition**:

* cov(): Calculates the covariance between selected columns.

**Why**:

* Covariance indicates how two variables vary together, useful for correlation analysis.

**Cell 8**

python

CopyEdit

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

grouped

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

**Definition**:

* groupby('Gender'): Groups data by gender.
* ['Salary'].mean(): Calculates average salary for each gender group.

**Why**:

* To compare earnings across genders.

**Cell 9**

python

CopyEdit

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

grouped

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

**Definition**:

* groupby(['Gender','Education']): Groups data by both gender and education.
* agg(['min','max']): Aggregates to find min and max salaries for each group.

**Why**:

* To explore differences in salary based on demographic and educational factors.

**Cell 10**

python

CopyEdit

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

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

**Definition**:

* groupby('Education').size(): Returns the count of records per education group.

**Why**:

* To examine the 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 occurrences of each gender.
* plot(kind='bar'): Plots the counts as a bar chart.

**Why**:

* Bar chart offers a clear visual comparison of gender frequencies.

**Cell 12**

python

CopyEdit

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

**Problem Step**: Plot salary distribution.

**Definition**:

* hist(): Creates a histogram.
* bins=10: Divides data into 10 intervals.
* edgecolor='black': Outlines each bar for clarity.

**Why**:

* Histograms reveal the distribution and spread of salary values.

**Cell 13**

python

CopyEdit

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

**Problem Step**: Visualize city-wise employee distribution.

**Definition**:

* value\_counts(): Counts number of employees in each city.
* plot(kind='bar'): Visualizes it as a bar chart.

**Why**:

* To understand how employees are distributed geographically.