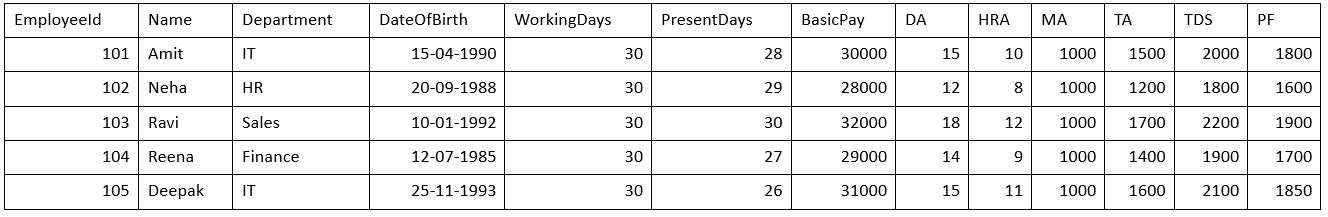
Jupyter notebook:Employee Salary Analysis

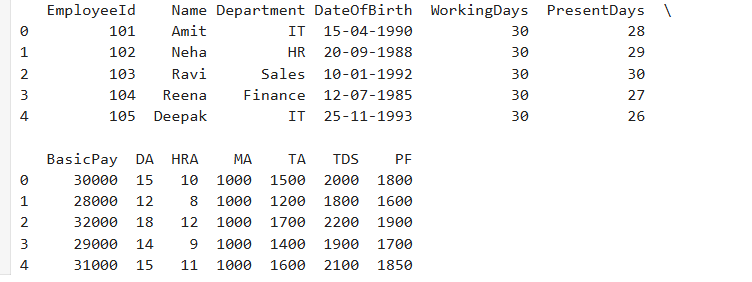


import pandas as pd

**1.Load the CSV data**

df = pd.read\_csv("employee\_salary.csv")

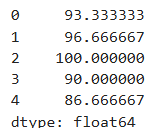
print(df)



**2. Calculate attendance percentage**

a=df['Attendance%'] = (df['PresentDays'] / df['WorkingDays']) \* 100

print(a)



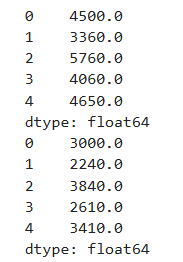
**3. Calculate DA and HRA amounts**

s=df['DA\_Amount'] = (df['BasicPay'] \* df['DA']) / 100

f=df['HRA\_Amount'] = (df['BasicPay'] \* df['HRA']) / 100

print(s)

print(f)

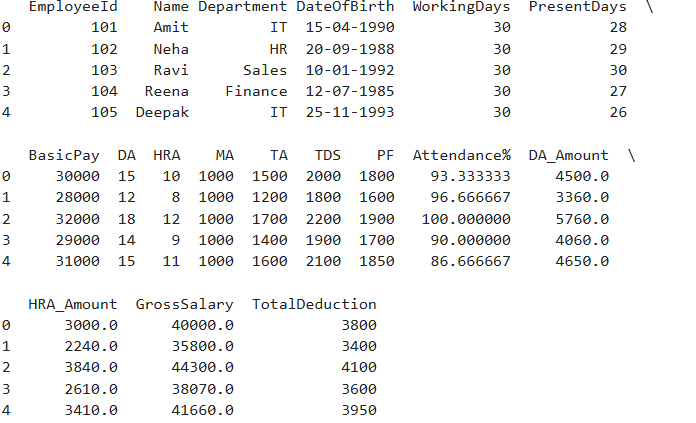


**4. Calculate Gross Salary**

df['GrossSalary'] = df['BasicPay'] + df['DA\_Amount'] + df['HRA\_Amount'] + df['MA'] + df['TA']

df['TotalDeduction'] = df['TDS'] + df['PF']

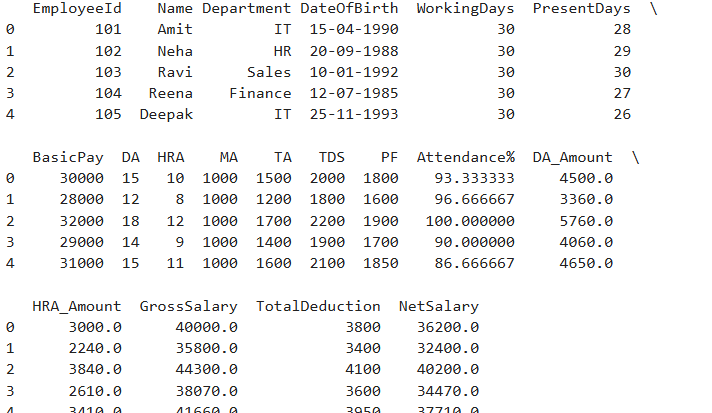
print(df)



**5. Calculate Net Salary**

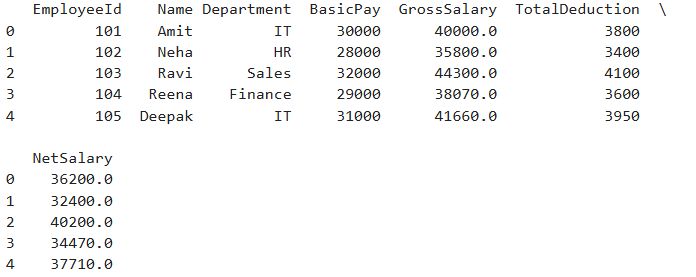
s=df['NetSalary'] = df['GrossSalary'] - df['TotalDeduction']

print(s)



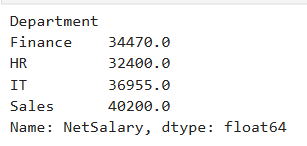
**6.Show final salary sheet**

print(df[['EmployeeId', 'Name', 'Department', 'BasicPay', 'GrossSalary', 'TotalDeduction', 'NetSalary']])



**7. Average Net Salary per Department**

print(df.groupby('Department')['NetSalary'].mean())



**8.Bar Plot of Net Salary per Employee**

import matplotlib.pyplot as plt

plt.figure(figsize=(10,5))

plt.bar(df['Name'], df['NetSalary'], color='orange')

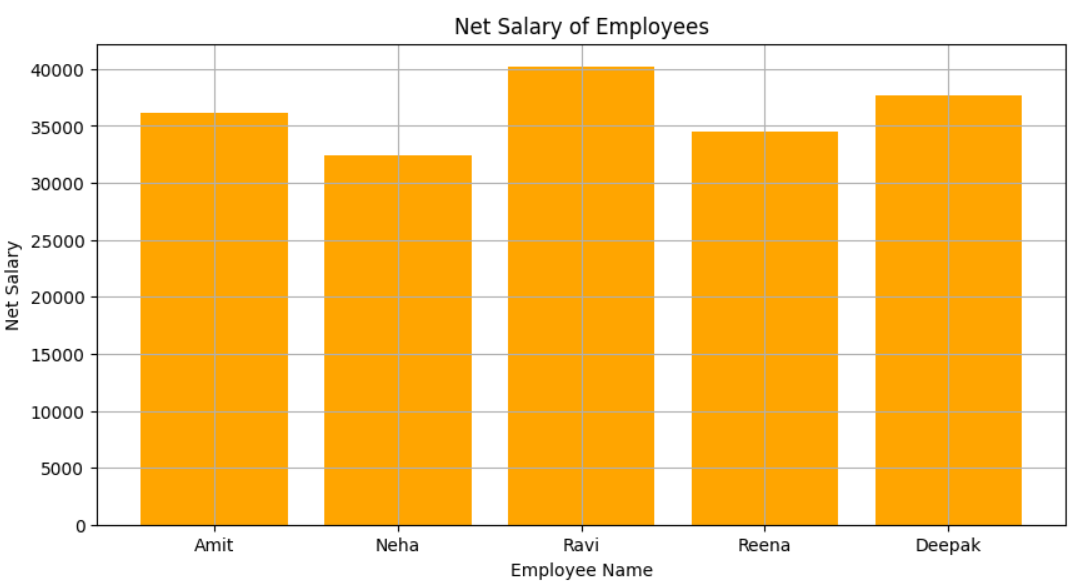
plt.title("Net Salary of Employees")

plt.xlabel("Employee Name")

plt.ylabel("Net Salary")

plt.grid(True)

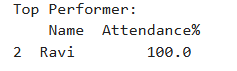
plt.show()



**9. Top Performer (Based on Attendance)**

top\_attendance = df[df['Attendance%'] == df['Attendance%'].max()]

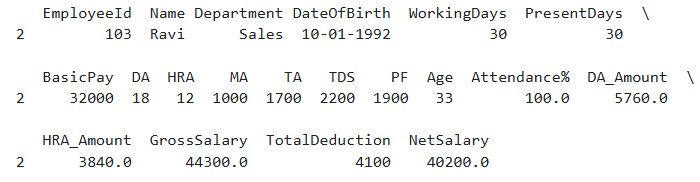
print("Top Performer:\n", top\_attendance[['Name', 'Attendance%’]]



**10.Employee which has the highest net salary**

d=df[df['NetSalary'] == df['NetSalary'].max()]

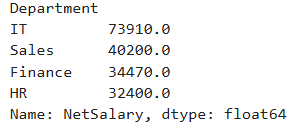
print(d)



**11.Department which has the highest total salary expense**

s=df.groupby('Department')['NetSalary'].sum().sort\_values(ascending=False)

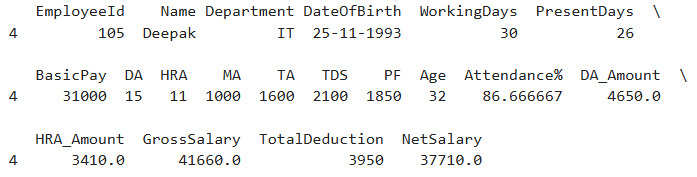
print(s)



**12.Employee which has the lowest attendance percentage**

df=df[df['Attendance%'] == df['Attendance%'].min()]

print(df)



**13.Employees which has less than 90% attendance**

df=df[df['Attendance%'] < 90].shape[0]

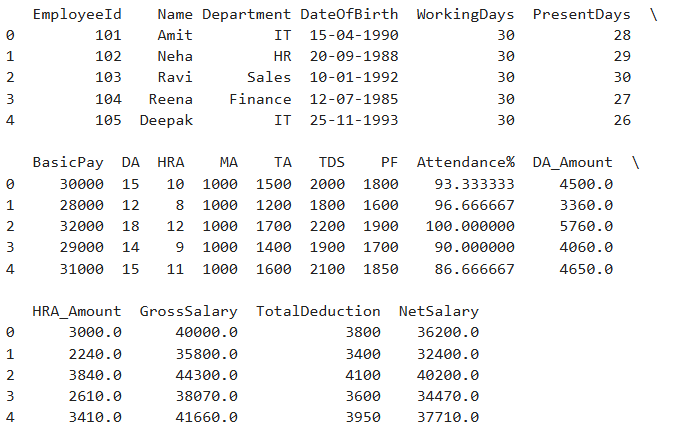
print(df)



**14.Average gross salary per department**

df=df.groupby('Department')['GrossSalary'].mean()

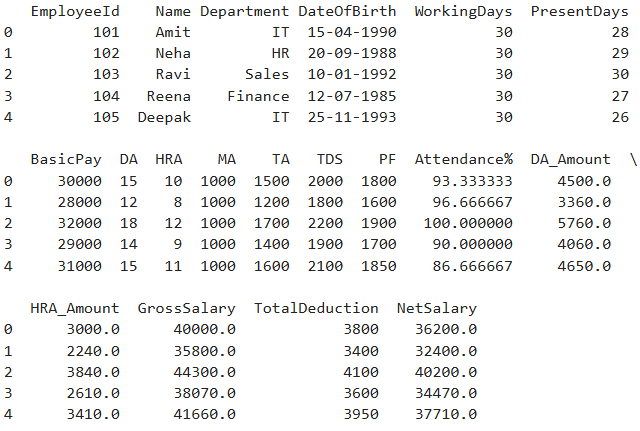
print(df)



**15.Employees whose TDS is more than ₹2000**

df=df[df['TDS'] > 2000]

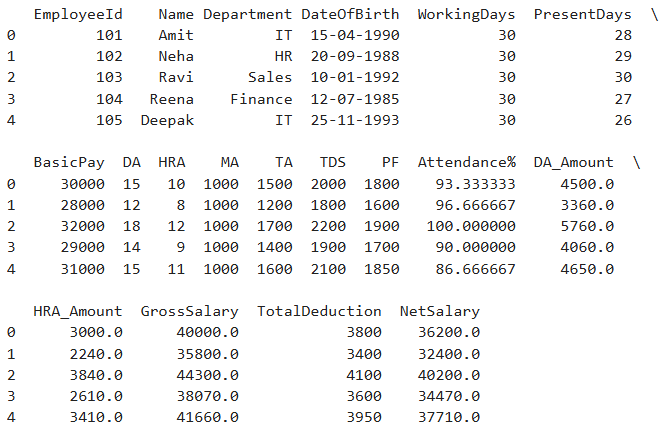
print(df)



**16.Correlation between Basic Pay and Net Salary**

S=df[['BasicPay', 'NetSalary']].corr()

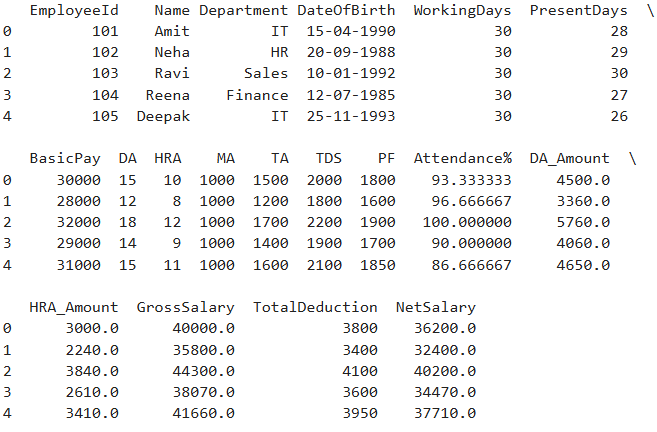
print(S)



**17.List employees who have DA greater than 15%.**

df[df['DA'] > 15]

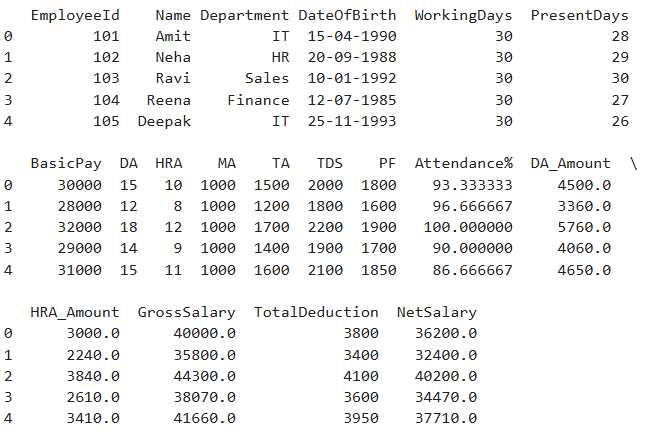
print(df)



**18. List employees who have DA greater than 15%**

df[df['PF'] > df['BasicPay'] \* 0.06]

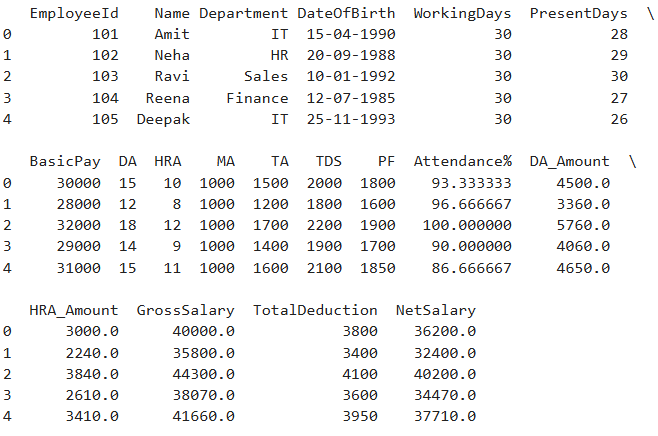
print(df)



**19. Rank employees by their Net Salary**

df.sort\_values(by='NetSalary', ascending=False)

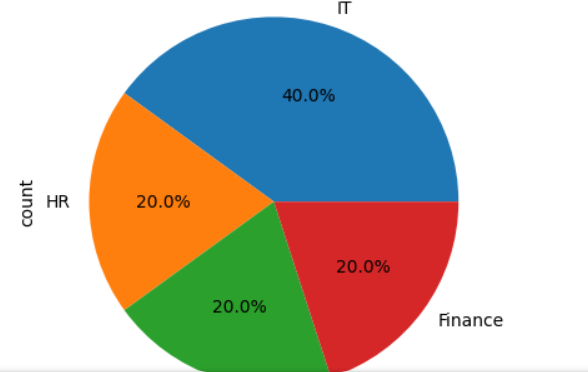
print(df)



**20.Plot Pie chart of total employees by Department**

df['Department'].value\_counts().plot.pie(autopct='%1.1f%%')

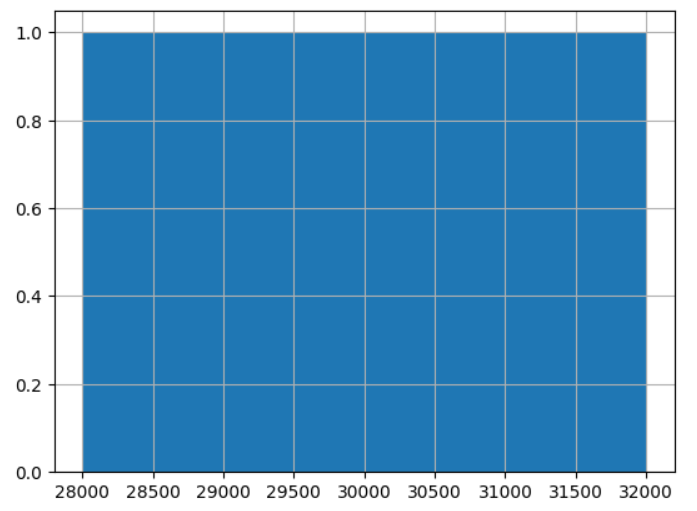
**plt.show()**

****

**21. Plot Histogram of Basic Pay**

df['BasicPay'].hist(bins=5)

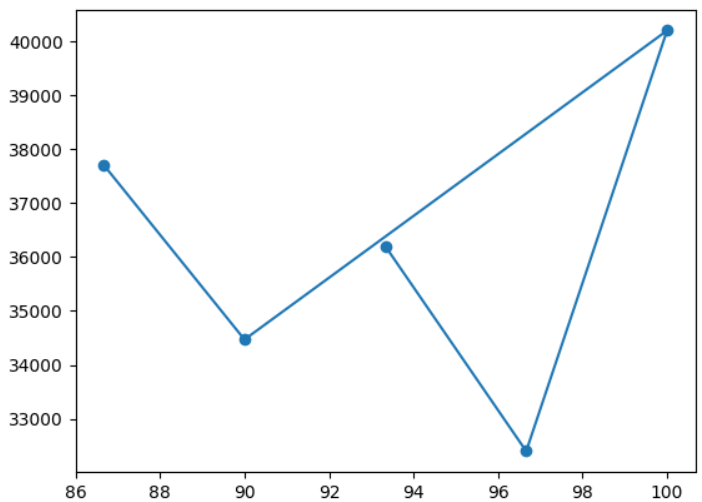
plt.show()



**22.Plot Line Graph: Attendance % vs Net Salary**

plt.plot(df['Attendance%'], df['NetSalary'], marker='o')

plt.show()



23.**Heatmap of salary components (BasicPay, DA, HRA, NetSalary).**

import seaborn as sns

sns.heatmap(df[['BasicPay','DA\_Amount','HRA\_Amount','NetSalary']].corr(), annot=True)

plt.show()



**24. Plot stacked bar chart: Gross Salary vs Deductions (TDS + PF).**

import pandas as pd

import matplotlib.pyplot as plt

# Sample dataset data

data = {

'Name': ['Amit', 'Neha', 'Ravi', 'Reena', 'Deepak'],

'GrossSalary': [45000, 41000, 50000, 43000, 47000],

'TDS': [2000, 1800, 2200, 1900, 2100],

'PF': [1800, 1600, 1900, 1700, 1850]

}

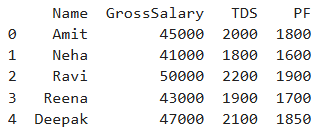
print(data)



# Create DataFrame

df = pd.DataFrame(data)

print(df)

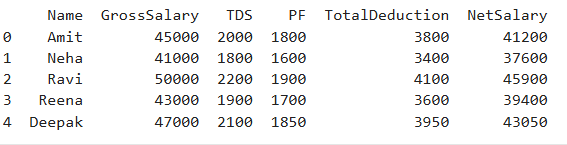


# Calculate Total Deduction and Net Salary

df['TotalDeduction'] = df['TDS'] + df['PF']

df['NetSalary'] = df['GrossSalary'] - df['TotalDeduction']

print(df)



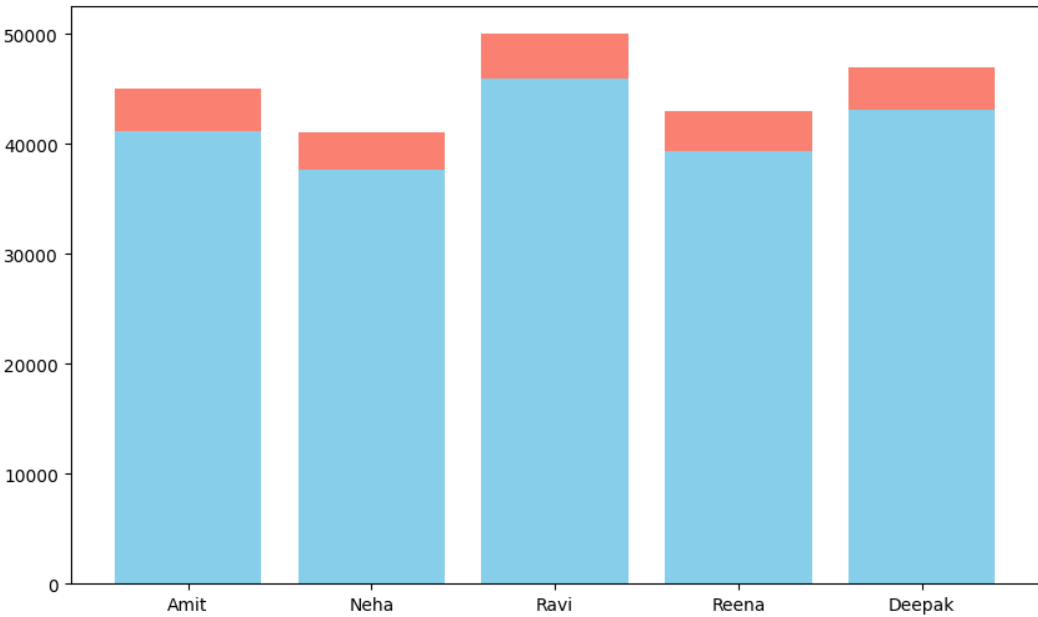
# Plot stacked bar chart

plt.figure(figsize=(10, 6))

plt.bar(df['Name'], df['NetSalary'], label='Net Salary', color='skyblue')

plt.bar(df['Name'], df['TotalDeduction'], bottom=df['NetSalary'], label='Deductions (TDS + PF)', color='salmon')

plt.show()

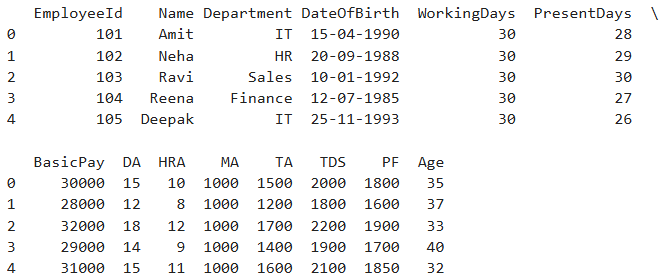


**25. Calculate Age of each employee (from Date of Birth)**

from datetime import datetime

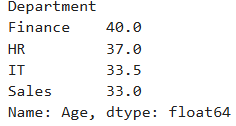
df['Age'] = pd.to\_datetime('today').year- pd.to\_datetime(df['DateOfBirth']).dt.year

print(df)



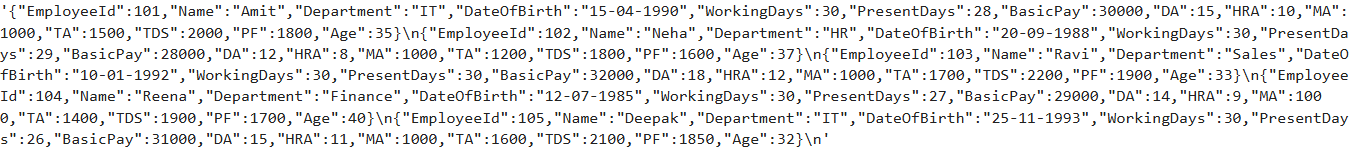
**26. Find average age by department**

df.groupby('Department')['Age'].mean()



**27. Display all employees as JSON**

df.to\_json(orient='records', lines=True)



**28. Add a 'Grade' column based on Net Salary**

def grade(sal):

    if sal > 50000:

        return 'A'

    elif sal > 40000:

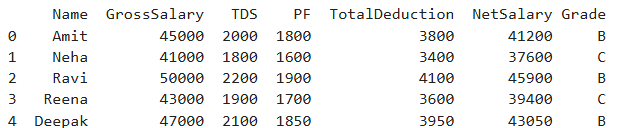
        return 'B'

    else:

        return 'C'

df['Grade'] = df['NetSalary'].apply(grade)

print(df)



**29.Highlight outliers in Net Salary using IQR**

Q1 = df['NetSalary'].quantile(0.25)

Q3 = df['NetSalary'].quantile(0.75)

IQR = Q3 - Q1

print(IQR)

outliers = df[(df['NetSalary'] < Q1 - 1.5\*IQR) | (df['NetSalary'] > Q3 + 1.5\*IQR)]

****

**30. Calculating the age of all employees using their Date of Birth**

from datetime import datetime

import pandas as pd

df = pd.read\_csv("employee\_salary.csv")

df['Age'] = pd.to\_datetime('today').year - pd.to\_datetime(df['DateOfBirth']).dt.year

df['AgeGroup'] = df['Age'].apply(lambda x: '<30' if x < 30 else

'30-40' if x <= 40 else '>40')

print(df['AgeGroup'].value\_counts())



**31.Create a department-wise report showing total gross salary, total deduction, and average net salary. Sort the result by highest average net salary.**

report = df.groupby('Department').agg({

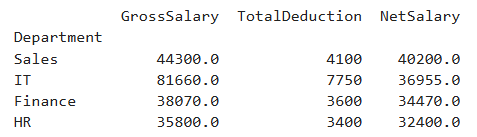
    'GrossSalary': 'sum',

    'TotalDeduction': 'sum',

    'NetSalary': 'mean'

}).sort\_values(by='NetSalary', ascending=False)

print(report)



**32.Create a salary slip generator function that takes an employee ID and returns a formatted string with all salary components and net salary.**

def generate\_salary\_slip(emp\_id):

    record = df[df['EmployeeId'] == emp\_id].iloc[0]

    slip = f"""Salary Slip for {record['Name']} (ID: {emp\_id})

    Department: {record['Department']}

    Basic Pay: ₹{record['BasicPay']}

    DA: ₹{record['DA\_Amount']}

    HRA: ₹{record['HRA\_Amount']}

    MA: ₹{record['MA']}

    TA: ₹{record['TA']}

    Gross Salary: ₹{record['GrossSalary']}

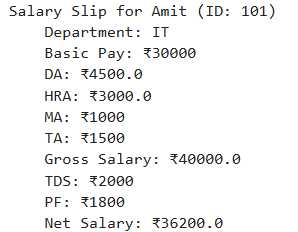
    TDS: ₹{record['TDS']}

    PF: ₹{record['PF']}

    Net Salary: ₹{record['NetSalary']}"""

    return slip

print(generate\_salary\_slip(101))



**33.Assigning performance ratings based on attendance percentage: A (≥95%), B (90–94.9%), C (<90%). Add this as a new column.**

def rating(att):

    if att >= 95:

        return 'A'

    elif att >= 90:

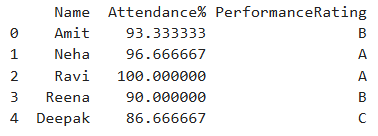
        return 'B'

    else:

        return 'C'

df['PerformanceRating'] = df['Attendance%'].apply(rating)

print(df[['Name', 'Attendance%', 'PerformanceRating']])

****

**34.Identifying employees who are outliers in terms of Net Salary using the IQR method, and print their names and salaries.**

Q1 = df['NetSalary'].quantile(0.25)

Q3 = df['NetSalary'].quantile(0.75)

IQR = Q3 - Q1

outliers = df[(df['NetSalary'] < Q1 - 1.5 \* IQR) |

(df['NetSalary'] > Q3 + 1.5 \* IQR)]

print(outliers[['Name', 'NetSalary']])



**35.Generating a list of employees who have perfect attendance and also receive the highest net salary in their department.**

perfect\_attendance = df[df['PresentDays'] == df['WorkingDays']]

result = perfect\_attendance.loc[

    perfect\_attendance.groupby('Department')['NetSalary'].idxmax()

]

print(result[['Name', 'Department', 'NetSalary']])



**36.Export department-wise salary summary to separate CSV files.**

departments = df['Department'].unique()

for dept in departments:

    df[df['Department'] == dept].to\_csv(f"{dept}\_salary\_summary.csv", index=False)

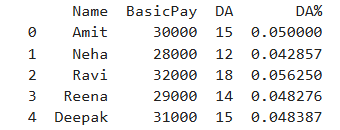
print(departments)



**37.Create a column DA% that represents the percentage of DA over Basic Pay.**

df['DA%'] = (df['DA'] / df['BasicPay']) \* 100

print(df[['Name', 'BasicPay', 'DA', 'DA%']])



**38.Creating a pivot table showing the average net salary per department and per performance rating.**

pivot = df.pivot\_table(

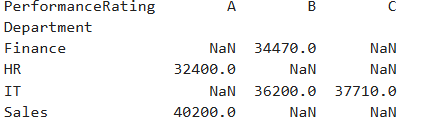
    values='NetSalary',

    index='Department',

    columns='PerformanceRating',

    aggfunc='mean' )

print(pivot)



**39.Adding a column LeaveDays and calculate it from WorkingDays - PresentDays.**

df['LeaveDays'] = df['WorkingDays'] - df['PresentDays']

print(df[['Name', 'WorkingDays', 'PresentDays', 'LeaveDays']])

