Employee (rudresh)

import pandas as pd

# Load dataset

df = pd.read\_csv("Extended\_Employee\_Performance\_and\_Productivity\_Data.csv")

df.head()

# Basic info (non-null values, datatypes)

print(df.info())

# Total missing values per column

print(df.isnull().sum())

# Statistical summary of numerical columns

df.describe()

# Fill missing satisfaction scores with the mean

df['Employee\_Satisfaction\_Score'] = df['Employee\_Satisfaction\_Score'].fillna(

df['Employee\_Satisfaction\_Score'].mean()

)

# Convert and encode categorical columns

df['Department'] = df['Department'].astype('category')

df['Department\_encoded'] = df['Department'].cat.codes

print(df[['Department', 'Department\_encoded']])

def evaluate\_performance(training\_hours, kpi\_met, awards\_won):

if training\_hours > 20 and kpi\_met > 80:

return 'Excellent'

elif kpi\_met > 60:

return 'Good'

else:

return 'Needs Improvement'

def is\_eligible\_for\_promotion(performance\_rating, previous\_year\_rating):

if performance\_rating == 'Excellent' and previous\_year\_rating >= 80:

return 'Eligible'

else:

return 'Not Eligible'

def needs\_training(training\_hours):

if training\_hours < 10:

return 'Yes'

else:

return 'No'

def calculate\_salary(monthly\_salary, performance\_rating):

if performance\_rating == 'Excellent':

return monthly\_salary \* 1.20 # 20% increment for excellent performance

elif performance\_rating == 'Good':

return monthly\_salary \* 1.10 # 10% increment for good performance

else:

return monthly\_salary # No increment for needs improvement

# Apply the logic using the DataFrame columns

df['performance\_rating'] = df.apply(

lambda row: evaluate\_performance(

row['Training\_Hours'],

row['Performance\_Score'],

row['Promotions']

),

axis=1

)

df['promotion\_eligibility'] = df.apply(

lambda row: is\_eligible\_for\_promotion(

row['performance\_rating'],

row['Employee\_Satisfaction\_Score']

),

axis=1

)

df['needs\_training'] = df['Training\_Hours'].apply(needs\_training)

df['new\_salary'] = df.apply(

lambda row: calculate\_salary(

row['Monthly\_Salary'],

row['performance\_rating']

),

axis=1

)

# Check the updated DataFrame

print(

df[

[

'Employee\_ID',

'performance\_rating',

'promotion\_eligibility',

'needs\_training',

'new\_salary'

]

].head()

)