### Initial Observations:

1. **EducationLevel.csv** appears to have tab-separated values instead of commas.
2. **Employee.csv** contains demographic, job-related, and compensation information.
3. **PerformanceRating.csv** includes satisfaction ratings, training opportunities, work-life balance, and performance evaluations.
4. **RatingLevel.csv** maps numerical ratings to descriptive categories.
5. **SatisfiedLevel.csv** maps numerical satisfaction levels to descriptive categories.

### Next Steps:

**Data Cleaning** – Handle missing values, standardize categorical values, and check for duplicates.

* **Pre-Merge Cleaning:**
  + Trims whitespace in key columns to ensure proper merging.
* **Merging:**
  + Combines the datasets using EmployeeID and aligns educational details via the Education column.
* **Categorical Cleaning:**
  + Replaces missing values in categorical columns with "Unknown".
  + Standardizes text by trimming whitespace.
* **Numerical Cleaning:**
  + Converts relevant columns to numeric.
  + Fills missing numerical values with the median of each column.
* **Date Conversion:**
  + Converts the HireDate column to datetime format.
* **Duplicates and Outliers:**
  + Removes duplicate rows.

### Merge:

merged\_df = (

    employee\_df.merge(performance\_rating\_df, on="EmployeeID", how="left")

               .merge(education\_level\_df, left\_on="Education", right\_on="EducationLevelID", how="left")

               .merge(rating\_level\_df, left\_on="SelfRating", right\_on="RatingID", how="left")

               .merge(satisfied\_level\_df, left\_on="JobSatisfaction", right\_on="SatisfactionID", how="left")

)

#Drop redundant ID columns if they exist

drop\_cols = ["EducationLevelID", "RatingID", "SatisfactionID"]

merged\_df.drop(columns=[col for col in drop\_cols if col in merged\_df.columns], inplace=True)

### Review the data types:

print(merged\_df.info())

print(merged\_df.dtypes)

print("\nMissing values per column:")

print(merged\_df.isnull().sum())

Data reviewing Insights:

1. No duplicate rows exist.
2. **190 missing values** in performance-related columns (PerformanceID, ReviewDate, JobSatisfaction, etc.), likely due to employees without performance reviews.

### Cleaning:

#Replace missing categorical values with "Unknown"

categorical\_cols = ["RatingLevel", "SatisfactionLevel"]

merged\_df[categorical\_cols] = merged\_df[categorical\_cols].fillna("Unknown")

#Fill missing numerical values with median for performance-related columns

numerical\_cols = [

    "EnvironmentSatisfaction", "JobSatisfaction", "RelationshipSatisfaction",

    "TrainingOpportunitiesWithinYear", "TrainingOpportunitiesTaken", "WorkLifeBalance",

    "SelfRating", "ManagerRating"

]

merged\_df[numerical\_cols] = merged\_df[numerical\_cols].fillna(merged\_df[numerical\_cols].median())

#Convert HireDate to datetime

merged\_df["HireDate"] = pd.to\_datetime(merged\_df["HireDate"], errors='coerce')

#Remove duplicate rows

merged\_df.drop\_duplicates(inplace=True)

#Display the cleaned data

print("Cleaned Data Preview:")

print(merged\_df.head())

print("\nMissing values per column:")

print(merged\_df.isnull().sum())

### Correlation:

I'll perform **Exploratory Data Analysis**  to identify trends and correlations related to the six key questions. Let's start with **performance ratings vs. satisfaction, tenure, and department**

import seaborn as sns

import matplotlib.pyplot as plt

# Convert necessary columns to numeric types

merged\_df["SelfRating"] = pd.to\_numeric(merged\_df["SelfRating"], errors="coerce")

merged\_df["JobSatisfaction"] = pd.to\_numeric(merged\_df["JobSatisfaction"], errors="coerce")

merged\_df["YearsAtCompany"] = pd.to\_numeric(merged\_df["YearsAtCompany"], errors="coerce")

# Plot correlation heatmap for performance ratings, satisfaction, and tenure

plt.figure(figsize=(8, 6))

sns.heatmap(merged\_df[["SelfRating", "JobSatisfaction", "YearsAtCompany"]].corr(), annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Correlation: Performance Ratings vs. Satisfaction & Tenure")

plt.show()

### Findings:

* **Performance ratings (SelfRating) and Job Satisfaction** show a moderate positive correlation.
* **Years at Company has a weaker correlation with SelfRating**, indicating tenure alone does not strongly impact performance ratings.

correlation heatmap for performance ratings, satisfaction, Work Life Balance, Environment Satisfaction, Relationship Satisfaction & Salary

plt.figure(figsize=(8, 6))

sns.heatmap(merged\_df[["SelfRating", "JobSatisfaction", "WorkLifeBalance" , "EnvironmentSatisfaction" , "RelationshipSatisfaction" , "Salary"]].corr(), annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Correlation: Performance Ratings vs. Job Satisfaction vs. Work Life Balance vs. Environment Satisfaction vs. RelationshipSatisfaction & Salary")

plt.show()

show a moderate positive correlation.

Plot correlation heatmap for Age & Salary

plt.figure(figsize=(8, 6))

sns.heatmap(merged\_df[["Age" , "Salary"]].corr(), annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Correlation: Age & Salary")

plt.show()

show a moderate positive correlation.

### Saving the sheet:

#Save the merged dataframe to a CSV file

merged\_df.to\_csv("HR Dataset.csv", index=False)

print("merged data saved as 'HR Dataset.csv'.")

# Download the CSV file

from google.colab import files

files.download("HR Dataset.csv")

Power BI

Cleaning

* 1. Drop unknown ratings
  2. Change type for columns = Table.TransformColumnTypes(#"PromotedHeaders",{{"EmployeeID", type text}, {"FirstName", type text}, {"LastName", type text}, {"Gender", type text}, {"Age", Int64.Type}, {"BusinessTravel", type text}, {"Department", type text}, {"DistanceFromHome (KM)", Int64.Type}, {"State", type text}, {"Ethnicity", type text}, {"Education", Int64.Type}, {"EducationField", type text}, {"JobRole", type text}, {"MaritalStatus", type text}, {"Salary", Int64.Type}, {"StockOptionLevel", Int64.Type}, {"OverTime", type text}, {"HireDate", type date}, {"Attrition", type text}, {"YearsAtCompany", Int64.Type}, {"YearsInMostRecentRole", Int64.Type}, {"YearsSinceLastPromotion", Int64.Type}, {"YearsWithCurrManager", Int64.Type}, {"PerformanceID", type text}, {"ReviewDate", type date}, {"EnvironmentSatisfaction", Int64.Type}, {"JobSatisfaction", Int64.Type}, {"RelationshipSatisfaction", Int64.Type}, {"TrainingOpportunitiesWithinYear", Int64.Type}, {"TrainingOpportunitiesTaken", Int64.Type}, {"WorkLifeBalance", Int64.Type}, {"SelfRating", Int64.Type}, {"ManagerRating", Int64.Type}, {"EducationLevel", type text}, {"RatingLevel", type text}, {"SatisfactionLevel", type text}})

1. Merge First Name + Last Name= Table.CombineColumns(#"Changed Type",{"FirstName", "LastName"},Combiner.CombineTextByDelimiter(" ", QuoteStyle.None),"Full Name")
2. Add Custom Columns

* Age Range=Table.AddColumn(#"Merged Columns", "Age Range", each if [Age] <= 29 then "18-29" else if [Age] <= 39 then "30-39" else if [Age] <= 49 then "40-49" else "Above 50")
* Salary Range= Table.AddColumn(#"Filtered Rows", "Salary Range", each if [Salary] < 20000 then "<20K" else if [Salary] <= 30000 then "20K-30K" else if [Salary] <= 40000 then "30K-40K" else if [Salary] <= 50000 then "40K-50K" else if [Salary] <= 60000 then "50K-60K" else if [Salary] <= 70000 then "60K-70K" else if [Salary] <= 80000 then "70K-80K" else if [Salary] <= 90000 then "80K-90K" else if [Salary] <= 100000 then "90K-100K" else ">100K")
* Distance Category= Table.AddColumn(#"Filtered Rows1", "Custom", each if [#"DistanceFromHome (KM)"] <= 5 then "Very Close" else if [#"DistanceFromHome (KM)"] <= 15 then "Short Distance)" else if [#"DistanceFromHome (KM)"] <= 30 then "Moderate Distance" else if [#"DistanceFromHome (KM)"] <= 50 then "Long Distance" else "Very Long Distance")

1. Replace Values

* Training Taken= Table.ReplaceValue(#"Replaced Value","1","Some Training Taken",Replacer.ReplaceText,{"TrainingOpportunitiesTaken"})
* Table.ReplaceValue(#"Replaced Value1","2","Some Training Taken",Replacer.ReplaceText,{"TrainingOpportunitiesTaken"})
* = Table.ReplaceValue(#"Replaced Value2","3","Extensive Training Taken",Replacer.ReplaceText,{"TrainingOpportunitiesTaken"})

1. Dax

* AVG Manager rate = AVERAGE('HR Dataset (1)'[ManagerRating )
* AVG Self Rating = AVERAGE('HR Dataset (1)'[SelfRating] )
* Employee Retention = [Employees working]/[Employees count]

])

* Employees count = DISTINCTCOUNT('HR Dataset (1)'[EmployeeID )
* Employees Left = CALCULATE(DISTINCTCOUNT('HR Dataset (1)'[EmployeeID]), 'HR Dataset (1)'[Attrition] = "Yes")
* Employees working = CALCULATE(DISTINCTCOUNT('HR Dataset (1)'[EmployeeID]), 'HR Dataset (1)'[Attrition] = "No")