Employee ID	Name	Department	Salary	Joining Date	Performance Score
1	Alice	HR	60,000	2020-01-15	3
2	Bob	IT	70,000	2019-06-20	4
3	Charlie	IT	80,000	2018-07-23	2
4	David	HR	65,000	2020-02-10	5
5	Eva	FINANCE	75,000	2021-03-15	3

1. **Calculate the average salary for each department:**

Df.groupby('Department')['Salary'].mean()

Explanation: We use groupby on the "Department" column to group employees by their department. Then, we select the "Salary" column and apply the mean() function to calculate the average salary within each department.

2. **Find the employee with the highest performance score in each department:**

Df.loc[df.groupby('Department')['Performance Score'].idxmax()]

Explanation: We group by the "Department" column and use idxmax() on the "Performance Score" column to get the index of the employee with the highest score in each department. Then, we use .loc to retrieve those rows from the DataFrame, showing details of the top-performing employee in each department.

3. **Add a new column that represents the number of years each employee has been with the company based on the Joining Date:**

From date time import date time

Df['Years with Company'] = df['Joining Date'].apply(lambda x: datetime.now().year – pd.to_datetime(x).year)

Explanation: First, we import datetime to get the current year. We convert the “Joining Date” to a datetime object using pd.to_datetime(), then subtract the joining year from the current year to calculate the number of years each employee has been with the company. We apply this calculation using a lambda function and store the result in a new column called “Years with Company”.

4. **Create a pivot table to display the total salary and average performance score for each department:**

Df.pivot_table(values=[‘Salary’, ‘Performance Score’], index=’Department’, aggfunc={‘Salary’: ‘sum’, ‘Performance Score’: ‘mean’})

Explanation: We create a pivot table where we set “Department” as the index. We specify values=[‘Salary’, ‘Performance Score’] to focus on those columns, and use aggfunc to define how we want to aggregate these values: calculating the sum for “Salary” and the mean for “Performance Score” in each department.

5. **Create a new DataFrame containing only the employees from the IT department who have a performance score greater than 3:**

Df_it_high_perf = df[(df[‘Department’] == ‘IT’) & (df[‘Performance Score’] > 3)]

Explanation: We filter the DataFrame by setting two conditions: employees who belong to the “IT” department and have a “Performance Score” greater than 3. Both conditions are combined using the & operator, and the filtered data is stored in df_it_high_perf.

6. **Perform an inner merge of this DataFrame with another DataFrame containing employee bonus information based on Employee ID:**

Assuming the bonus DataFrame is named `bonus_df` with ‘Employee ID’ and ‘Bonus’ columns

Merged_df = df.merge(bonus_df, on=’Employee ID’, how=’inner’)

Explanation: This performs an inner join between df and bonus_df based on the common column “Employee ID”. Only rows where “Employee ID” matches in both DataFrames are kept in merged_df, allowing us to add bonus information for each employee.

7. **Calculate the cumulative sum of the Performance Score column grouped by Department:**

Df[‘Cumulative Performance Score’] = df.groupby(‘Department’)[‘Performance Score’].cumsum()

Explanation: We use groupby on “Department” to calculate the cumulative sum of the “Performance Score” within each department. Cumsum() adds up performance scores progressively within each group. The results are stored in a new column called “Cumulative Performance Score”.

8. **Rank employees within each department based on their Salary:**

Df[‘Salary Rank’] = df.groupby(‘Department’)[‘Salary’].rank(ascending=False)

Explanation: We rank employees by salary within each department using groupby on “Department”. The rank() function assigns ranks in descending order (ascending=False), so the highest-paid employee in each department gets rank 1. This rank is stored in a new column, “Salary Rank”.

9. **Filter the DataFrame to show only employees who have been with the company for more than 2 years:**

Df[df[‘Years with Company’] > 2]

Explanation: After calculating the “Years with Company” column (as shown in question 3), we filter the DataFrame to keep only employees with more than 2 years in the company. This is done by setting a condition on the “Years with Company” column.