103 Pandas Practice Questions & Solutions

```
import pandas as pd
import numpy as np

# Create a sample DataFrame
np.random.seed(42)
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank',
'Grace', 'Hannah', 'Ivy', 'Jack'],
    'Age': np.random.randint(18, 60, size=10),
    'Gender': ['F', 'M', 'M', 'M', 'F', 'M', 'F', 'F', 'F', 'M'],
    'Salary': np.random.randint(30000, 100000, size=10),
    'Department': ['HR', 'IT', 'Finance', 'HR', 'Finance', 'IT', 'HR',
'Finance', 'IT', 'HR'],
    'Join_Date': pd.date_range(start='2020-01-01', periods=10,
freq='M'),
    'Rating': np.random.uniform(1, 5, size=10).round(1)
}

df = pd.DataFrame(data)
df
```

```
C:\Users\91790\AppData\Local\Temp\ipykernel_10404\2156502269.py:12:
FutureWarning: 'M' is deprecated and will be removed in a future
version, please use 'ME' instead.
  'Join_Date': pd.date_range(start='2020-01-01', periods=10,
freq='M'),
```

	Name	Age	Gender	Salary	Department	Join_Date	Rating
0	Alice	56	F	74131	HR	2020-01-31	1.8
1	Bob	46	M	90263	IT	2020-02-29	1.7
2	Charlie	32	M	46023	Finance	2020-03-31	1.7
3	David	25	M	71090	HR	2020-04-30	2.2
4	Eve	38	F	97221	Finance	2020-05-31	3.1
5	Frank	56	M	94820	IT	2020-06-30	2.7
6	Grace	36	F	30769	HR	2020-07-31	2.2
7	Hannah	40	F	89735	Finance	2020-08-31	3.4
8	Ivy	28	F	92955	IT	2020-09-30	1.6
9	Jack	28	M	94925	HR	2020-10-31	2.2

Easy Questions

Q1: Display the first 5 rows and then calculate the mean age of employees.

```
df.head()
```

	Name	Age	Gender	Salary	Department	Join_Date	Rating
0	Alice	56	F	74131	HR	2020-01-31	1.8
1	Bob	46	M	90263	IT	2020-02-29	1.7

```
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

df['Age'].mean()

38.5
```

Q2: Show the total number of unique departments and display a summary of the DataFrame.

```
df['Department'].nunique()

3

df.describe()

             Age       Salary            Join_Date     Rating
count  10.00000     10.00000                   10  10.000000
mean   38.50000  78193.20000  2020-06-15 07:12:00   2.260000
min    25.00000  30769.00000  2020-01-31 00:00:00   1.600000
25%    29.00000  71850.25000  2020-04-07 12:00:00   1.725000
50%    37.00000  89999.00000  2020-06-15 00:00:00   2.200000
75%    44.50000  94353.75000  2020-08-23 06:00:00   2.575000
max    56.00000  97221.00000  2020-10-31 00:00:00   3.400000
std    11.16791  23012.55255                  NaN   0.622183
```

Q3: Select employees whose rating is greater than 3 and show their names and salaries.

```
df[df['Rating'] > 3][['Name', 'Salary']]

     Name  Salary
4     Eve   97221
7  Hannah   89735
```

Q4: Sort the DataFrame by salary in descending order, and display the top 3 records.

```
df.sort_values(by='Salary', ascending=False).nlargest(3, 'Salary')

    Name  Age Gender  Salary Department  Join_Date  Rating
4    Eve   38      F   97221    Finance 2020-05-31     3.1
9   Jack   28      M   94925         HR 2020-10-31     2.2
5  Frank   56      M   94820         IT 2020-06-30     2.7
```

Q5: Display all records where the employee works in the 'IT' or 'HR' department and has a salary above 50,000.

```
df[(df['Department'].isin(['IT', 'HR'])) & (df['Salary'] > 50000)]
```

```
    Name  Age Gender  Salary Department  Join_Date  Rating
0  Alice   56      F   74131         HR 2020-01-31     1.8
1    Bob   46      M   90263         IT 2020-02-29     1.7
3  David   25      M   71090         HR 2020-04-30     2.2
5  Frank   56      M   94820         IT 2020-06-30     2.7
8    Ivy   28      F   92955         IT 2020-09-30     1.6
9   Jack   28      M   94925         HR 2020-10-31     2.2
```

Q6: Filter the DataFrame to show employees with ratings greater than 4, and show the department-wise mean salary.

```
df[df['Rating'] > 4].groupby('Department')['Salary'].mean()
```

```
Series([], Name: Salary, dtype: float64)
```

Q7: Show employees who joined after '2020-05-01' and display their age and department.

```
df[df['Join_Date'] > '2020-05-01'][['Age', 'Department']]
```

```
   Age Department
4   38    Finance
5   56         IT
6   36         HR
7   40    Finance
8   28         IT
9   28         HR
```

Q8: Select rows where the employee's name starts with 'A' or 'B', and display their name, age, and salary.

```
df[df['Name'].str.startswith('A', 'B')][['Name', 'Age', 'Salary']]
```

```
    Name  Age  Salary
0  Alice   56   74131
```

Q9: Create a new column 'Salary_in_Thousands' by dividing salary by 1000, and display the updated DataFrame.

```
df['Salary_in_Thousands'] = df['Salary'] / 1000
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_in_Thousands
```

```
0                74.131
1                90.263
2                46.023
3                71.090
4                97.221
```

Q10: Remove the 'Rating' column and display the remaining columns.

```
df.drop(columns='Rating')
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_in_Thousands
0                74.131
1                90.263
2                46.023
3                71.090
4                97.221
```

Q11: Filter employees with a rating greater than 3 and then display their names, salaries, and departments.

```
df[df['Rating'] > 3][['Name', 'Salary', 'Department']]
```

```
     Name  Salary Department
4     Eve   97221    Finance
7  Hannah   89735    Finance
```

Q12: Display the number of employees in each department, and calculate the mean age of employees.

```
df.groupby('Department').value_counts(), df['Age'].mean()
```

```
(Department  Name     Age  Gender  Salary  Join_Date   Rating
Salary_in_Thousands
 Finance     Charlie  32   M       46023   2020-03-31  1.7     46.023
1
             Eve      38   F       97221   2020-05-31  3.1     97.221
1
             Hannah   40   F       89735   2020-08-31  3.4     89.735
1
 HR          Alice    56   F       74131   2020-01-31  1.8     74.131
1
             David    25   M       71090   2020-04-30  2.2     71.090
```

```
1
             Grace    36   F       30769   2020-07-31  2.2     30.769
1
             Jack     28   M       94925   2020-10-31  2.2     94.925
1
 IT          Bob      46   M       90263   2020-02-29  1.7     90.263
1
             Frank    56   M       94820   2020-06-30  2.7     94.820
1
             Ivy      28   F       92955   2020-09-30  1.6     92.955
1
 Name: count, dtype: int64,
 38.5)
```

Q13: Sort employees by 'Join_Date' in ascending order and display the top 5 rows.

```
df.sort_values(by='Join_Date').head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_in_Thousands
0               74.131
1               90.263
2               46.023
3               71.090
4               97.221
```

Q14: Select employees with a salary higher than 60,000 and who joined after January 2020, displaying their names and ages.

```
df[(df['Salary'] > 60000) & (df['Join_Date'] > '2020-01-31')][['Name',
'Age']]
```

```
     Name  Age
1     Bob   46
3   David   25
4     Eve   38
5   Frank   56
7  Hannah   40
8     Ivy   28
9    Jack   28
```

Q15: Create a new column 'Age_Group' that labels employees as 'Young' if their age is below 30, otherwise 'Experienced'. Then display the first 5 rows.

```
df['Age_Group'] = df['Age'].apply(lambda x: 'Young' if x<30 else
'Experienced')
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_in_Thousands    Age_Group
0               74.131  Experienced
1               90.263  Experienced
2               46.023  Experienced
3               71.090        Young
4               97.221  Experienced
```

Q16: Filter employees in the 'HR' department and calculate the mean and standard deviation of their salaries.

```
df[df['Department'] == 'HR']['Salary'].agg(['mean', 'std'])
```

```
mean    67728.750000
std     26820.054317
Name: Salary, dtype: float64
```

Q17: Display employees whose name ends with the letter 'e' and whose rating is exactly 3.1

```
df[(df['Name'].str.endswith('e')) & (df['Rating']  == 3.1)]
```

```
  Name  Age Gender  Salary Department  Join_Date  Rating
Salary_in_Thousands  \
4  Eve   38      F   97221    Finance 2020-05-31     3.1
97.221

     Age_Group
4  Experienced
```

Q18: Find the median salary and count of employees for each gender.

```
df.groupby('Gender').agg({'Salary':'median', 'Name':'count'})
```

```
         Salary  Name
Gender
F       89735.0     5
M       90263.0     5
```

Q19: Filter employees whose salary is within 40,000 to 70,000, and then display their names and departments.

```
df[df['Salary'].between(40000, 70000)][['Name', 'Department']]
```

```
      Name Department
2  Charlie    Finance
```

Q20: For employees older than 35, display their names, departments, and the difference between their salary and the average salary.

```
df['Salary_Avg_Salary_Diff'] = df['Salary'] - df['Salary'].mean()
df[df['Age'] > 35][['Name', 'Department','Salary_Avg_Salary_Diff']]
```

```
     Name Department  Salary_Avg_Salary_Diff
0   Alice         HR                 -4062.2
1     Bob         IT                 12069.8
4     Eve    Finance                 19027.8
5   Frank         IT                 16626.8
6   Grace         HR                -47424.2
7  Hannah    Finance                 11541.8
```

Q21: Filter employees with a salary greater than 50,000 and display their names and departments. Then sort the results by 'Rating' in descending order.

Q22: Count the number of employees in each department, and then filter departments with more than 3 employees.

```
dept_count = df.groupby('Department').value_counts()
dept_count[dept_count > 3]
```

```
Series([], Name: count, dtype: int64)
```

Q23: Create a new column 'Salary_After_Tax' by reducing 15% tax from the salary and then display the first 5 rows with 'Name' and 'Salary_After_Tax'.

```
df['Salary_After_Tax'] = df['Salary'] - df['Salary'] * 0.15
df[['Name', 'Salary_After_Tax']]
```

```
      Name  Salary_After_Tax
0    Alice          63011.35
1      Bob          76723.55
2  Charlie          39119.55
3    David          60426.50
4      Eve          82637.85
5    Frank          80597.00
6    Grace          26153.65
7   Hannah          76274.75
8      Ivy          79011.75
9     Jack          80686.25
```

Q24: Filter employees in the 'Finance' department and calculate the mean age and median salary for these employees.

```
df[df['Department'] == 'Finance'].agg({'Age':'mean',
'Salary':'median'})
```

```
Age          36.666667
Salary    89735.000000
dtype: float64
```

Q25: Select employees whose names contain the letter 'A' and who are older than 30, displaying their names and ages.

```
df[(df['Name'].str.contains('A')) & (df['Age'] > 30)][['Name', 'Age']]
```

```
    Name  Age
0  Alice   56
```

Q26: Calculate the sum of salaries and count of employees in each department.

```
df.groupby('Department').agg({'Name':'count', 'Salary':'sum'})
```

```
            Name  Salary
Department
Finance        3  232979
HR             4  270915
IT             3  278038
```

Q27: Sort employees by 'Salary' in descending order and then display the top 3 employees along with their names and ratings.

```
sorted_by_salary = df.sort_values(by='Salary',
ascending=False).nlargest(3, 'Salary')
sorted_by_salary[['Name', 'Rating']]
```

```
    Name  Rating
4    Eve     3.1
9   Jack     2.2
5  Frank     2.7
```

Q28: Filter employees who have a rating below the median rating and display their names and departments.

```
df['median_rating'] = df['Rating'].median()
df[df['Rating'] < df['median_rating']][['Name', 'Department']]
```

```
      Name Department
0    Alice         HR
1      Bob         IT
2  Charlie    Finance
8      Ivy         IT
```

Q29: Create a new column 'Experience_Level' to classify employees with less than 2 years of experience as 'Junior', otherwise 'Senior'. Then display the first 5 rows.

```
df['Experience'] = (pd.Timestamp.now() - df['Join_Date']).dt.days //
365

df['Experience_Level'] = df['Experience'].apply(lambda x: 'Junior' if
x < 2 else 'Senior')
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_After_Tax  median_rating  Experience Experience_Level
0          63011.35            2.2           4           Senior
1          76723.55            2.2           4           Senior
2          39119.55            2.2           4           Senior
3          60426.50            2.2           4           Senior
4          82637.85            2.2           4           Senior
```

Q30: For employees in the 'HR' department, display their names and calculate the difference between their salary and the department's average salary.

```
df['Dept_Avg_Salary'] = df.groupby('Department')
['Salary'].transform('mean')
df['Salary_Dept_Avg_Salary_Diff'] = df['Salary'] -
df['Dept_Avg_Salary']
df[(df['Department'] == 'HR')][['Name',
'Salary_Dept_Avg_Salary_Diff']]
```

```
    Name  Salary_Dept_Avg_Salary_Diff
0  Alice                      6402.25
3  David                      3361.25
6  Grace                    -36959.75
9   Jack                     27196.25
```

Q31: Filter employees who joined after the year 2019, then display their names and salaries, sorted by 'Salary' in descending order.

```
df[df['Join_Date'] > '2019-01-31'][['Name',
'Salary']].sort_values(by='Salary', ascending=False)
```

```
      Name  Salary
4      Eve   97221
9     Jack   94925
5    Frank   94820
8      Ivy   92955
```

```
1      Bob   90263
7   Hannah   89735
0    Alice   74131
3    David   71090
2  Charlie   46023
6    Grace   30769
```

Q32: Select employees with salaries greater than 60,000 and whose names contain the letter 'e', and display their names and departments.

```
df[(df['Salary'] > 60000) & (df['Name'].str.contains('e'))][['Name',
'Department']]
```

```
    Name Department
0  Alice         HR
4    Eve    Finance
```

Q33: Create a new column 'Years_In_Company' by subtracting 'Join_Date' from today's date, then display the first 5 rows with this column.

```
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_After_Tax  median_rating  Experience Experience_Level  \
0          63011.35            2.2           4           Senior
1          76723.55            2.2           4           Senior
2          39119.55            2.2           4           Senior
3          60426.50            2.2           4           Senior
4          82637.85            2.2           4           Senior

   Dept_Avg_Salary  Salary_Dept_Avg_Salary_Diff
0     67728.750000                 6402.250000
1     92679.333333                -2416.333333
2     77659.666667               -31636.666667
3     67728.750000                 3361.250000
4     77659.666667                19561.333333
```

Q34: Group employees by 'Department' and calculate the mean and median salaries. Then display departments with a mean salary greater than 55,000.

```
df.groupby('Department')['Salary'].agg(['mean', 'median']).query('mean
> 55000')
```

```
                    mean   median
Department
Finance     77659.666667  89735.0
HR          67728.750000  72610.5
IT          92679.333333  92955.0
```

Q35: Filter employees in the 'HR' department and calculate the average rating and total salary of these employees.

```
df[df['Department'] == 'HR'].agg({'Rating':'mean', 'Salary':'sum'})
```

```
Rating         2.1
Salary    270915.0
dtype: float64
```

Q36: Create a column 'Salary_Tax' by applying a 20% tax deduction on salaries, then filter employees with a net salary greater than 50,000.

```
df['Salary_Tax'] = df['Salary'] - df['Salary']*0.2
df[df['Salary_Tax'] > 50000]
```

```
     Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
0   Alice   56      F   74131         HR 2020-01-31     1.8
63011.35
1     Bob   46      M   90263         IT 2020-02-29     1.7
76723.55
3   David   25      M   71090         HR 2020-04-30     2.2
60426.50
4     Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
5   Frank   56      M   94820         IT 2020-06-30     2.7
80597.00
7  Hannah   40      F   89735    Finance 2020-08-31     3.4
76274.75
8     Ivy   28      F   92955         IT 2020-09-30     1.6
79011.75
9    Jack   28      M   94925         HR 2020-10-31     2.2
80686.25

   median_rating  Experience Experience_Level  Dept_Avg_Salary  \
0            2.2           4           Senior     67728.750000
1            2.2           4           Senior     92679.333333
3            2.2           4           Senior     67728.750000
4            2.2           4           Senior     77659.666667
5            2.2           4           Senior     92679.333333
7            2.2           4           Senior     77659.666667
8            2.2           4           Senior     92679.333333
9            2.2           3           Senior     67728.750000
```

```
   Salary_Dept_Avg_Salary_Diff  Salary_Tax
0                  6402.250000     59304.8
1                 -2416.333333     72210.4
3                  3361.250000     56872.0
4                 19561.333333     77776.8
5                  2140.666667     75856.0
7                 12075.333333     71788.0
8                   275.666667     74364.0
9                 27196.250000     75940.0
```

Q37: Count the number of employees in each department and display only the departments with more than 3 employees.

```
df.groupby('Department').agg({'Name':'count'})
```

```
            Name
Department
Finance        3
HR             4
IT             3
```

```
df['Total_Dept_Employees'] =
df.groupby('Department').agg('Name').transform('count')
df[df['Total_Dept_Employees'] > 3]
```

```
    Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
0  Alice   56      F   74131         HR 2020-01-31     1.8
63011.35
3  David   25      M   71090         HR 2020-04-30     2.2
60426.50
6  Grace   36      F   30769         HR 2020-07-31     2.2
26153.65
9   Jack   28      M   94925         HR 2020-10-31     2.2
80686.25

   median_rating  Experience Experience_Level  Dept_Avg_Salary  \
0            2.2           4           Senior         67728.75
3            2.2           4           Senior         67728.75
6            2.2           4           Senior         67728.75
9            2.2           3           Senior         67728.75

   Salary_Dept_Avg_Salary_Diff  Salary_Tax  Total_Dept_Employees
0                      6402.25     59304.8                     4
3                      3361.25     56872.0                     4
6                    -36959.75     24615.2                     4
9                     27196.25     75940.0                     4
```

Q38: Select employees who are older than 35 and have a rating greater than 3.0, then display their names and ratings.

```
df[(df['Age'] > 35) & (df['Rating'] > 3.0)][['Name', 'Rating']]
```

```
     Name  Rating
4     Eve     3.1
7  Hannah     3.4
```

Q39: Calculate the sum of salaries for each department and sort the results in ascending order of total salary.

```
df['Dept_Total_Salary'] = df.groupby('Department')
['Salary'].transform('sum')
df.sort_values(by='Dept_Total_Salary')
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
4      Eve   38      F   97221    Finance 2020-05-31     3.1
7   Hannah   40      F   89735    Finance 2020-08-31     3.4
0    Alice   56      F   74131         HR 2020-01-31     1.8
3    David   25      M   71090         HR 2020-04-30     2.2
6    Grace   36      F   30769         HR 2020-07-31     2.2
9     Jack   28      M   94925         HR 2020-10-31     2.2
1      Bob   46      M   90263         IT 2020-02-29     1.7
5    Frank   56      M   94820         IT 2020-06-30     2.7
8      Ivy   28      F   92955         IT 2020-09-30     1.6

   Salary_After_Tax  median_rating  Experience Experience_Level  \
2          39119.55            2.2           4           Senior
4          82637.85            2.2           4           Senior
7          76274.75            2.2           4           Senior
0          63011.35            2.2           4           Senior
3          60426.50            2.2           4           Senior
6          26153.65            2.2           4           Senior
9          80686.25            2.2           3           Senior
1          76723.55            2.2           4           Senior
5          80597.00            2.2           4           Senior
8          79011.75            2.2           4           Senior

   Dept_Avg_Salary  Salary_Dept_Avg_Salary_Diff  Salary_Tax  \
2     77659.666667                -31636.666667     36818.4
4     77659.666667                 19561.333333     77776.8
7     77659.666667                 12075.333333     71788.0
0     67728.750000                  6402.250000     59304.8
3     67728.750000                  3361.250000     56872.0
6     67728.750000                -36959.750000     24615.2
9     67728.750000                 27196.250000     75940.0
1     92679.333333                 -2416.333333     72210.4
5     92679.333333                  2140.666667     75856.0
8     92679.333333                   275.666667     74364.0
```

```
   Total_Dept_Employees  Dept_Total_Salary
2                     3             232979
4                     3             232979
7                     3             232979
0                     4             270915
3                     4             270915
6                     4             270915
9                     4             270915
1                     3             278038
5                     3             278038
8                     3             278038
```

Q40: Create a column 'Salary_Diff' that calculates the difference between an employee's salary and the mean salary of their department. Then display the first 5 rows.

```
df['Salary_Dept_Avg_Salary_Diff'].head()
```

```
0     6402.250000
1    -2416.333333
2   -31636.666667
3     3361.250000
4    19561.333333
Name: Salary_Dept_Avg_Salary_Diff, dtype: float64
```

Intermediate Questions

Q41: Group employees by department, and find the maximum salary and minimum age for each department.

```
df.groupby('Department').agg({'Salary':'max', 'Age':'min'})
```

```
            Salary  Age
Department
Finance      97221   32
HR           94925   25
IT           94820   28
```

Q42: Display the first 5 records where employees are male, and their age is less than 40.

```
df[(df['Gender'] == 'M') & (df['Age'] < 40)].head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
9     Jack   28      M   94925         HR 2020-10-31     2.2

   Salary_After_Tax  median_rating  Experience Experience_Level  \
```

```
2          39119.55             2.2           4           Senior
3          60426.50             2.2           4           Senior
9          80686.25             2.2           3           Senior

   Dept_Avg_Salary  Salary_Dept_Avg_Salary_Diff  Salary_Tax  \
2     77659.666667                -31636.666667     36818.4
3     67728.750000                  3361.250000     56872.0
9     67728.750000                 27196.250000     75940.0

   Total_Dept_Employees  Dept_Total_Salary
2                     3             232979
3                     4             270915
9                     4             270915
```

Q43: Create a new column 'Experience' as the difference between the current date and their join date, and show the first 5 rows.

```
df.head()

      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_After_Tax  median_rating  Experience Experience_Level  \
0          63011.35            2.2           4           Senior
1          76723.55            2.2           4           Senior
2          39119.55            2.2           4           Senior
3          60426.50            2.2           4           Senior
4          82637.85            2.2           4           Senior

   Dept_Avg_Salary  Salary_Dept_Avg_Salary_Diff  Salary_Tax  \
0     67728.750000                  6402.250000     59304.8
1     92679.333333                 -2416.333333     72210.4
2     77659.666667                -31636.666667     36818.4
3     67728.750000                  3361.250000     56872.0
4     77659.666667                 19561.333333     77776.8

   Total_Dept_Employees  Dept_Total_Salary
0                     4             270915
1                     3             278038
2                     3             232979
3                     4             270915
4                     3             232979
```

Q44: Find the total number of female employees and the average rating of male employees.

```
df[df['Gender'] == 'F'].shape
```

```
(5, 16)
```

```
df[df['Gender'] == 'M']['Rating'].mean()
```

```
2.1
```

Q45: For employees in the 'Finance' department, calculate the mean age and the total salary.

```
df[df['Department'] == 'Finance'].agg({'Salary':'sum', 'Age':'mean'})
```

```
Salary    232979.000000
Age           36.666667
dtype: float64
```

Q46: Sort employees by age, then reset the index of the sorted DataFrame.

```
df.sort_values(by='Age').reset_index(drop=True)
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    David   25      M   71090         HR 2020-04-30     2.2
1      Ivy   28      F   92955         IT 2020-09-30     1.6
2     Jack   28      M   94925         HR 2020-10-31     2.2
3  Charlie   32      M   46023    Finance 2020-03-31     1.7
4    Grace   36      F   30769         HR 2020-07-31     2.2
5      Eve   38      F   97221    Finance 2020-05-31     3.1
6   Hannah   40      F   89735    Finance 2020-08-31     3.4
7      Bob   46      M   90263         IT 2020-02-29     1.7
8    Alice   56      F   74131         HR 2020-01-31     1.8
9    Frank   56      M   94820         IT 2020-06-30     2.7

   Salary_After_Tax  median_rating  Experience Experience_Level  \
0          60426.50            2.2           4           Senior
1          79011.75            2.2           4           Senior
2          80686.25            2.2           3           Senior
3          39119.55            2.2           4           Senior
4          26153.65            2.2           4           Senior
5          82637.85            2.2           4           Senior
6          76274.75            2.2           4           Senior
7          76723.55            2.2           4           Senior
8          63011.35            2.2           4           Senior
9          80597.00            2.2           4           Senior

   Dept_Avg_Salary  Salary_Dept_Avg_Salary_Diff  Salary_Tax  \
0     67728.750000                  3361.250000     56872.0
1     92679.333333                   275.666667     74364.0
2     67728.750000                 27196.250000     75940.0
```

```
3     77659.666667                -31636.666667     36818.4
4     67728.750000                -36959.750000     24615.2
5     77659.666667                 19561.333333     77776.8
6     77659.666667                 12075.333333     71788.0
7     92679.333333                 -2416.333333     72210.4
8     67728.750000                  6402.250000     59304.8
9     92679.333333                  2140.666667     75856.0

   Total_Dept_Employees  Dept_Total_Salary
0                     4             270915
1                     3             278038
2                     4             270915
3                     3             232979
4                     4             270915
5                     3             232979
6                     3             232979
7                     3             278038
8                     4             270915
9                     3             278038
```

Q47: Filter employees with a salary in the range 40,000 to 90,000, and display their department and rating.

```
df[df['Salary'].between(40000, 90000)][['Department', 'Rating']]

  Department  Rating
0         HR     1.8
2    Finance     1.7
3         HR     2.2
7    Finance     3.4
```

Q48: Find the median salary for each gender and department combination.

```
df.groupby(['Gender', 'Department'])['Salary'].median()

Gender  Department
F       Finance       93478.0
        HR            52450.0
        IT            92955.0
M       Finance       46023.0
        HR            83007.5
        IT            92541.5
Name: Salary, dtype: float64
```

Q49: Filter employees who joined before 2021 and whose salary is above the average salary.

```
df[(df['Join_Date'] < '2021-01-31') & ((df['Salary']) >
df['Salary'].mean())]
```

```
     Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
1     Bob   46      M   90263         IT 2020-02-29     1.7
76723.55
4     Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
5   Frank   56      M   94820         IT 2020-06-30     2.7
80597.00
7  Hannah   40      F   89735    Finance 2020-08-31     3.4
76274.75
8     Ivy   28      F   92955         IT 2020-09-30     1.6
79011.75
9    Jack   28      M   94925         HR 2020-10-31     2.2
80686.25

   median_rating  Experience Experience_Level  Dept_Avg_Salary  \
1            2.2           4           Senior     92679.333333
4            2.2           4           Senior     77659.666667
5            2.2           4           Senior     92679.333333
7            2.2           4           Senior     77659.666667
8            2.2           4           Senior     92679.333333
9            2.2           3           Senior     67728.750000

   Salary_Dept_Avg_Salary_Diff  Salary_Tax  Total_Dept_Employees  \
1                -2416.333333     72210.4                     3
4                19561.333333     77776.8                     3
5                 2140.666667     75856.0                     3
7                12075.333333     71788.0                     3
8                  275.666667     74364.0                     3
9                27196.250000     75940.0                     4

   Dept_Total_Salary
1             278038
4             232979
5             278038
7             232979
8             278038
9             270915
```

Q50: For each department, find the count of employees and the average salary. Sort by average salary in descending order.

```
df.groupby('Department')['Salary'].agg(['count',
'mean']).sort_values(by='mean', ascending=False)

            count          mean
Department
IT              3  92679.333333
Finance         3  77659.666667
HR              4  67728.750000
```

Q51: Group employees by gender and department, then calculate the average rating and total salary for each group.

```
df.groupby(['Department', 'Gender']).agg({'Rating':'mean',
'Salary':'sum'})
```

```
                    Rating  Salary
Department Gender
Finance    F          3.25  186956
           M          1.70   46023
HR         F          2.00  104900
           M          2.20  166015
IT         F          1.60   92955
           M          2.20  185083
```

Q52: Filter employees with a rating above the overall average rating and sort them by salary in descending order.

```
df['avg_rating'] = df['Rating'].mean()
df[df['Rating'] > df['avg_rating']].sort_values(by='Salary',
ascending=False)
```

```
     Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
4     Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
5   Frank   56      M   94820         IT 2020-06-30     2.7
80597.00
7  Hannah   40      F   89735    Finance 2020-08-31     3.4
76274.75

   median_rating  Experience Experience_Level  Dept_Avg_Salary  \
4            2.2           4           Senior     77659.666667
5            2.2           4           Senior     92679.333333
7            2.2           4           Senior     77659.666667

   Salary_Dept_Avg_Salary_Diff  Salary_Tax  Total_Dept_Employees  \
4                 19561.333333     77776.8                     3
5                  2140.666667     75856.0                     3
7                 12075.333333     71788.0                     3

   Dept_Total_Salary  avg_rating
4             232979        2.26
5             278038        2.26
7             232979        2.26
```

Q53: Create a new column 'Years_Since_Joining' that calculates the number of years since an employee joined, and display the first 5 rows.

```
df['Years_Since_Joining'] = (pd.Timestamp.now() -
df['Join_Date']).dt.days//365
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_After_Tax  median_rating  Experience Experience_Level  \
0          63011.35            2.2           4           Senior
1          76723.55            2.2           4           Senior
2          39119.55            2.2           4           Senior
3          60426.50            2.2           4           Senior
4          82637.85            2.2           4           Senior

   Dept_Avg_Salary  Salary_Dept_Avg_Salary_Diff  Salary_Tax  \
0     67728.750000                  6402.250000     59304.8
1     92679.333333                 -2416.333333     72210.4
2     77659.666667                -31636.666667     36818.4
3     67728.750000                  3361.250000     56872.0
4     77659.666667                 19561.333333     77776.8

   Total_Dept_Employees  Dept_Total_Salary  avg_rating
Years_Since_Joining
0                     4             270915        2.26
4
1                     3             278038        2.26
4
2                     3             232979        2.26
4
3                     4             270915        2.26
4
4                     3             232979        2.26
4
```

Q54: Display employees with a salary in the top 10%, showing their name, salary, and department.

```
Salary_Cut_Off = df['Salary'].quantile(0.9)
df[df['Salary'] > Salary_Cut_Off][['Name', 'Salary', 'Department']]
```

```
  Name  Salary Department
4  Eve   97221    Finance
```

Q55: Calculate the average rating of employees for each combination of department and gender, and filter those with an average rating above 2.

```
df.groupby(['Gender', 'Department'])
['Rating'].mean().reset_index().query('Rating > 2')
```

```
  Gender Department  Rating
0      F    Finance    3.25
4      M         HR    2.20
5      M         IT    2.20
```

Q56: For each department, calculate the sum of salaries and the count of employees, and filter departments with more than 3 employees.

```
df.groupby('Department').agg({'Salary':'sum',
'Name':'count'}).query('Name > 3')
```

```
            Salary  Name
Department
HR          270915     4
```

Q57: Find the youngest and oldest employees in each department and display their names, age, and department.

Q58: Display the names and salaries of employees whose salary is more than the average salary of their respective department.

```
df['Dept_Avg_Salary'] = df.groupby('Department')
['Salary'].transform('mean')
df[df['Salary'] > df['Dept_Avg_Salary']][['Name', 'Salary',
'Dept_Avg_Salary']]
```

```
     Name  Salary  Dept_Avg_Salary
0   Alice   74131     67728.750000
3   David   71090     67728.750000
4     Eve   97221     77659.666667
5   Frank   94820     92679.333333
7  Hannah   89735     77659.666667
8     Ivy   92955     92679.333333
9    Jack   94925     67728.750000
```

Q59: Create a new column 'Salary_Per_Year_Of_Experience' by dividing the salary by the number of years since joining, and then filter employees with a salary per year greater than 15,000.

```
df['Salary_Per_Year_Of_Experience'] = df['Salary'] / df['Experience']
df[df['Salary_Per_Year_Of_Experience'] > 15000]
```

```
     Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
0   Alice   56      F   74131         HR 2020-01-31     1.8
```

```
63011.35
1     Bob   46      M   90263         IT 2020-02-29     1.7
76723.55
3   David   25      M   71090         HR 2020-04-30     2.2
60426.50
4     Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
5   Frank   56      M   94820         IT 2020-06-30     2.7
80597.00
7  Hannah   40      F   89735    Finance 2020-08-31     3.4
76274.75
8     Ivy   28      F   92955         IT 2020-09-30     1.6
79011.75
9    Jack   28      M   94925         HR 2020-10-31     2.2
80686.25

   median_rating  Experience Experience_Level  Dept_Avg_Salary  \
0            2.2           4           Senior     67728.750000
1            2.2           4           Senior     92679.333333
3            2.2           4           Senior     67728.750000
4            2.2           4           Senior     77659.666667
5            2.2           4           Senior     92679.333333
7            2.2           4           Senior     77659.666667
8            2.2           4           Senior     92679.333333
9            2.2           3           Senior     67728.750000

   Salary_Dept_Avg_Salary_Diff  Salary_Tax  Total_Dept_Employees  \
0                  6402.250000     59304.8                     4
1                 -2416.333333     72210.4                     3
3                  3361.250000     56872.0                     4
4                 19561.333333     77776.8                     3
5                  2140.666667     75856.0                     3
7                 12075.333333     71788.0                     3
8                   275.666667     74364.0                     3
9                 27196.250000     75940.0                     4

   Dept_Total_Salary  avg_rating  Years_Since_Joining  \
0             270915        2.26                    4
1             278038        2.26                    4
3             270915        2.26                    4
4             232979        2.26                    4
5             278038        2.26                    4
7             232979        2.26                    4
8             278038        2.26                    4
9             270915        2.26                    3

   Salary_Per_Year_Of_Experience
0                   18532.750000
1                   22565.750000
3                   17772.500000
```

```
4                    24305.250000
5                    23705.000000
7                    22433.750000
8                    23238.750000
9                    31641.666667
```

Q60: Find employees who have been in the company for more than 3 years and whose salary is below the median salary of their department.

```
df['dept_median_salary'] = df.groupby('Department')
['Salary'].transform('median')
df[(df['Experience'] > 3) & (df['Salary'] < df['dept_median_salary'])]
[['Name', 'Salary', 'Experience','dept_median_salary']]
```

```
      Name  Salary  Experience  dept_median_salary
1      Bob   90263           4             92955.0
2  Charlie   46023           4             89735.0
3    David   71090           4             72610.5
6    Grace   30769           4             72610.5
```

Q61: Group employees by gender and department, calculate the average salary and the count of employees in each group, and filter groups with an average salary above 60,000.

```
df.groupby(['Gender', 'Department']).agg({'Salary':'mean',
'Name':'count'}).query('Salary > 60000')
```

```
                    Salary  Name
Gender Department
F      Finance     93478.0     2
       IT          92955.0     1
M      HR          83007.5     2
       IT          92541.5     2
```

Q62: Create a new column 'Years_Till_Retirement' that calculates how many years remain until the employee reaches 65 years of age. Then filter employees who will retire in less than 10 years.

```
df['Years_Till_Retirement'] = 65 - df['Age']
df[df['Years_Till_Retirement'] < 10]
```

```
    Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
0  Alice   56      F   74131         HR 2020-01-31     1.8
63011.35
5  Frank   56      M   94820         IT 2020-06-30     2.7
80597.00

   median_rating  Experience  ... Dept_Avg_Salary  \
0            2.2           4  ...    67728.750000
```

```
5            2.2           4  ...    92679.333333

   Salary_Dept_Avg_Salary_Diff  Salary_Tax  Total_Dept_Employees  \
0                  6402.250000     59304.8                     4
5                  2140.666667     75856.0                     3

   Dept_Total_Salary  avg_rating  Years_Since_Joining  \
0             270915        2.26                    4
5             278038        2.26                    4

   Salary_Per_Year_Of_Experience  dept_median_salary
Years_Till_Retirement
0                       18532.75             72610.5
9
5                       23705.00             92955.0
9

[2 rows x 21 columns]
```

Q63: Sort employees by 'Join_Date' and calculate the cumulative sum of salaries within each department.

```
df.sort_values(by='Join_Date').groupby('Department')
['Salary'].cumsum()

0     74131
1     90263
2     46023
3    145221
4    143244
5    185083
6    175990
7    232979
8    278038
9    270915
Name: Salary, dtype: int32
```

Q64: Display employees with the top 5 highest ratings, showing their names, salaries, and the difference between their salary and the overall average salary.

```
df['Salary_Avg_Salary_Diff'] = df['Salary'] - df['Salary'].mean()
df.nlargest(5, 'Rating')[['Name', 'Salary', 'Salary_Avg_Salary_Diff']]

     Name  Salary  Salary_Avg_Salary_Diff
7  Hannah   89735                 11541.8
4     Eve   97221                 19027.8
5   Frank   94820                 16626.8
3   David   71090                 -7103.2
6   Grace   30769                -47424.2
```

Q65: Find employees with a rating higher than the mean rating in their department and display their names and ratings.

```
df['Dept_Mean_Rating'] = df.groupby('Department')
['Rating'].transform('mean')
df[df['Rating'] > df['Dept_Mean_Rating']][['Name', 'Rating']]
```

```
     Name  Rating
3   David     2.2
4     Eve     3.1
5   Frank     2.7
6   Grace     2.2
7  Hannah     3.4
9    Jack     2.2
```

Q66: Group employees by department and calculate the minimum and maximum salary for each department, then filter departments where the maximum salary exceeds 95,000.

```
df.groupby('Department')['Salary'].agg(['min', 'max'])
```

```
               min    max
Department
Finance     46023  97221
HR          30769  94925
IT          90263  94820
```

Q67: Create a new column 'Age_Range' to categorize employees as 'Young', 'Mid-Age', or 'Senior' based on their age, and then display the count of employees in each category.

```
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_After_Tax  median_rating  Experience  ... Salary_Tax  \
0          63011.35            2.2           4  ...    59304.8
1          76723.55            2.2           4  ...    72210.4
2          39119.55            2.2           4  ...    36818.4
3          60426.50            2.2           4  ...    56872.0
4          82637.85            2.2           4  ...    77776.8

   Total_Dept_Employees  Dept_Total_Salary  avg_rating
Years_Since_Joining  \
0                     4             270915        2.26
4
```

```
1                       3             278038        2.26
4
2                       3             232979        2.26
4
3                       4             270915        2.26
4
4                       3             232979        2.26
4

   Salary_Per_Year_Of_Experience  dept_median_salary
Years_Till_Retirement  \
0                        18532.75             72610.5
9
1                        22565.75             92955.0
19
2                        11505.75             89735.0
33
3                        17772.50             72610.5
40
4                        24305.25             89735.0
27

   Salary_Avg_Salary_Diff  Dept_Mean_Rating
0                 -4062.2          2.100000
1                 12069.8          2.000000
2                -32170.2          2.733333
3                 -7103.2          2.100000
4                 19027.8          2.733333

[5 rows x 23 columns]
```

```
df['Age_Range'] = pd.cut(df['Experience'], bins=[0, 30, 50, np.inf],
labels=['Young', 'Mid_Age', 'Senior'])
df['Age_Range'].value_counts()
```

```
Age_Range
Young      10
Mid_Age     0
Senior      0
Name: count, dtype: int64
```

Q68: Calculate the correlation between 'Salary' and 'Rating' and between 'Age' and 'Years_Since_Joining'.

```
df[['Salary', 'Rating', 'Age', 'Experience']].corr()
```

```
              Salary    Rating       Age  Experience
Salary      1.000000  0.326757  0.177724   -0.255467
Rating      0.326757  1.000000  0.142317    0.033884
```

```
Age         0.177724  0.142317  1.000000    0.330350
Experience -0.255467  0.033884  0.330350    1.000000
```

Q69: For each gender and department, calculate the percentage of employees with a salary above 50,000.

Q70: For each employee, calculate the ratio of their salary to their years since joining, and filter those with a ratio above 20,000.

```
df['Salary_To_Experience_Ratio'] = df['Salary'] / df['Experience']
df[df['Salary_To_Experience_Ratio'] > 20000]
```

```
     Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
1     Bob   46      M   90263         IT 2020-02-29     1.7
76723.55
4     Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
5   Frank   56      M   94820         IT 2020-06-30     2.7
80597.00
7  Hannah   40      F   89735    Finance 2020-08-31     3.4
76274.75
8     Ivy   28      F   92955         IT 2020-09-30     1.6
79011.75
9    Jack   28      M   94925         HR 2020-10-31     2.2
80686.25

   median_rating  Experience  ... Dept_Total_Salary  avg_rating  \
1            2.2           4  ...             278038        2.26
4            2.2           4  ...             232979        2.26
5            2.2           4  ...             278038        2.26
7            2.2           4  ...             232979        2.26
8            2.2           4  ...             278038        2.26
9            2.2           3  ...             270915        2.26

   Years_Since_Joining  Salary_Per_Year_Of_Experience
dept_median_salary  \
1                    4                   22565.750000
92955.0
4                    4                   24305.250000
89735.0
5                    4                   23705.000000
92955.0
7                    4                   22433.750000
89735.0
8                    4                   23238.750000
92955.0
9                    3                   31641.666667
72610.5
```

```
   Years_Till_Retirement  Salary_Avg_Salary_Diff  Dept_Mean_Rating
Age_Range  \
1                     19                 12069.8          2.000000
Young
4                     27                 19027.8          2.733333
Young
5                      9                 16626.8          2.000000
Young
7                     25                 11541.8          2.733333
Young
8                     37                 14761.8          2.000000
Young
9                     37                 16731.8          2.100000
Young

   Salary_To_Experience_Ratio
1                22565.750000
4                24305.250000
5                23705.000000
7                22433.750000
8                23238.750000
9                31641.666667

[6 rows x 25 columns]
```

Q71: Group employees by 'Gender' and 'Department' to calculate the average salary, total salary, and count of employees. Then filter groups where the average salary exceeds 70,000.

```
df.groupby(['Gender', 'Department'])['Salary'].agg(['mean', 'sum',
'count']).query('mean > 70000')
```

```
                       mean     sum  count
Gender Department
F      Finance     93478.0  186956      2
       IT          92955.0   92955      1
M      HR          83007.5  166015      2
       IT          92541.5  185083      2
```

Q72: Create a column 'Experience_Level' based on 'Years_Since_Joining', categorizing employees with less than 3 years as 'Junior', between 3-7 years as 'Mid', and more than 7 years as 'Senior'. Then display the count of employees in each category.

```
df['Experience_Level'] = pd.cut(df['Experience'], bins=[0, 3, 7,
np.inf], labels=['Junior', 'Mid', 'Senior'])
df['Experience_Level'].value_counts()
```

```
Experience_Level
Mid       9
```

```
Junior    1
Senior    0
Name: count, dtype: int64
```

Q73: Sort employees by their 'Join_Date' and calculate the cumulative salary for each department, then display the first 5 rows.

```
df.sort_values('Join_Date').groupby('Department')
['Salary'].cumsum().head()
```

```
0     74131
1     90263
2     46023
3    145221
4    143244
Name: Salary, dtype: int32
```

Q74: Display employees with the top 5 highest 'Salary_Per_Year' (salary divided by years since joining) and show their names, salaries, and ratios.

```
df['Salary_Per_Year'] = df['Salary'] / df['Experience']
df.nlargest(5, 'Salary_Per_Year')[['Name', 'Salary',
'Salary_Per_Year']]
```

```
    Name  Salary  Salary_Per_Year
9   Jack   94925     31641.666667
4    Eve   97221     24305.250000
5  Frank   94820     23705.000000
8    Ivy   92955     23238.750000
1    Bob   90263     22565.750000
```

Q75: Calculate the mean rating for each department and filter employees whose rating is greater than their department's average rating.

```
df[df['Rating'] > df['Dept_Mean_Rating']]
```

```
     Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
3   David   25      M   71090         HR 2020-04-30     2.2
60426.50
4     Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
5   Frank   56      M   94820         IT 2020-06-30     2.7
80597.00
6   Grace   36      F   30769         HR 2020-07-31     2.2
26153.65
7  Hannah   40      F   89735    Finance 2020-08-31     3.4
76274.75
9    Jack   28      M   94925         HR 2020-10-31     2.2
80686.25
```

```
   median_rating  Experience  ... avg_rating  Years_Since_Joining  \
3            2.2           4  ...       2.26                    4
4            2.2           4  ...       2.26                    4
5            2.2           4  ...       2.26                    4
6            2.2           4  ...       2.26                    4
7            2.2           4  ...       2.26                    4
9            2.2           3  ...       2.26                    3

   Salary_Per_Year_Of_Experience  dept_median_salary
Years_Till_Retirement  \
3                   17772.500000             72610.5
40
4                   24305.250000             89735.0
27
5                   23705.000000             92955.0
9
6                    7692.250000             72610.5
29
7                   22433.750000             89735.0
25
9                   31641.666667             72610.5
37

   Salary_Avg_Salary_Diff  Dept_Mean_Rating  Age_Range  \
3                 -7103.2          2.100000      Young
4                 19027.8          2.733333      Young
5                 16626.8          2.000000      Young
6                -47424.2          2.100000      Young
7                 11541.8          2.733333      Young
9                 16731.8          2.100000      Young

   Salary_To_Experience_Ratio  Salary_Per_Year
3                17772.500000     17772.500000
4                24305.250000     24305.250000
5                23705.000000     23705.000000
6                 7692.250000      7692.250000
7                22433.750000     22433.750000
9                31641.666667     31641.666667

[6 rows x 26 columns]
```

Q76: Group employees by 'Department' and calculate the maximum, minimum, and mean salary. Then display departments where the maximum salary is greater than 90,000.

```
df.groupby(['Department'])['Salary'].agg(['max', 'min',
'mean']).rename(columns={'max':'max_salary'}).query("max_salary >
90000")
```

```
            max_salary    min          mean
Department
Finance          97221  46023  77659.666667
HR               94925  30769  67728.750000
IT               94820  90263  92679.333333
```

Q77: Create a new column 'Salary_Per_Age' by dividing salary by age, then find and display the top 5 employees with the highest values.

```
df['Salary_Per_Age'] = df['Salary'] / df['Age']
df.nlargest(5, 'Salary_Per_Age')
```

```
     Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
9    Jack   28      M   94925         HR 2020-10-31     2.2
80686.25
8     Ivy   28      F   92955         IT 2020-09-30     1.6
79011.75
3   David   25      M   71090         HR 2020-04-30     2.2
60426.50
4     Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
7  Hannah   40      F   89735    Finance 2020-08-31     3.4
76274.75

   median_rating  Experience  ... Years_Since_Joining  \
9            2.2           3  ...                   3
8            2.2           4  ...                   4
3            2.2           4  ...                   4
4            2.2           4  ...                   4
7            2.2           4  ...                   4

   Salary_Per_Year_Of_Experience  dept_median_salary
Years_Till_Retirement  \
9                    31641.666667             72610.5
37
8                    23238.750000             92955.0
37
3                    17772.500000             72610.5
40
4                    24305.250000             89735.0
27
7                    22433.750000             89735.0
25

   Salary_Avg_Salary_Diff  Dept_Mean_Rating  Age_Range  \
9                 16731.8          2.100000      Young
8                 14761.8          2.000000      Young
3                 -7103.2          2.100000      Young
4                 19027.8          2.733333      Young
```

```
7                  11541.8          2.733333      Young

   Salary_To_Experience_Ratio  Salary_Per_Year  Salary_Per_Age
9                31641.666667     31641.666667     3390.178571
8                23238.750000     23238.750000     3319.821429
3                17772.500000     17772.500000     2843.600000
4                24305.250000     24305.250000     2558.447368
7                22433.750000     22433.750000     2243.375000

[5 rows x 27 columns]
```

Q78: Create a pivot table to show the average salary and total salary for each combination of 'Department' and 'Gender'.

```
pd.pivot_table(df, index='Gender', columns='Department',
values='Salary', aggfunc=['mean', 'sum'])

               mean                          sum
Department  Finance       HR       IT Finance      HR      IT
Gender
F           93478.0  52450.0  92955.0  186956  104900   92955
M           46023.0  83007.5  92541.5   46023  166015  185083
```

Q79: Calculate the rolling mean salary over a window of 4 employees sorted by 'Join_Date', then display the first 10 results.

```
df['Rolling_mean']=df.sort_values(by='Join_Date')
['Salary'].rolling(window=4).mean()
df.nlargest(10, 'Rolling_mean')

      Name  Age Gender  Salary Department  Join_Date  Rating  \
7   Hannah   40      F   89735    Finance 2020-08-31     3.4
5    Frank   56      M   94820         IT 2020-06-30     2.7
9     Jack   28      M   94925         HR 2020-10-31     2.2
8      Ivy   28      F   92955         IT 2020-09-30     1.6
4      Eve   38      F   97221    Finance 2020-05-31     3.1
6    Grace   36      F   30769         HR 2020-07-31     2.2
3    David   25      M   71090         HR 2020-04-30     2.2
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7

   Salary_After_Tax  median_rating  Experience  ...  \
7          76274.75            2.2           4  ...
5          80597.00            2.2           4  ...
9          80686.25            2.2           3  ...
8          79011.75            2.2           4  ...
4          82637.85            2.2           4  ...
6          26153.65            2.2           4  ...
3          60426.50            2.2           4  ...
```

```
0          63011.35            2.2            4  ...
1          76723.55            2.2            4  ...
2          39119.55            2.2            4  ...

  Salary_Per_Year_Of_Experience  dept_median_salary
Years_Till_Retirement  \
7                   22433.750000             89735.0
25
5                   23705.000000             92955.0
9
9                   31641.666667             72610.5
37
8                   23238.750000             92955.0
37
4                   24305.250000             89735.0
27
6                    7692.250000             72610.5
29
3                   17772.500000             72610.5
40
0                   18532.750000             72610.5
9
1                   22565.750000             92955.0
19
2                   11505.750000             89735.0
33

   Salary_Avg_Salary_Diff  Dept_Mean_Rating  Age_Range  \
7                 11541.8          2.733333      Young
5                 16626.8          2.000000      Young
9                 16731.8          2.100000      Young
8                 14761.8          2.000000      Young
4                 19027.8          2.733333      Young
6                -47424.2          2.100000      Young
3                 -7103.2          2.100000      Young
0                 -4062.2          2.100000      Young
1                 12069.8          2.000000      Young
2                -32170.2          2.733333      Young

   Salary_To_Experience_Ratio  Salary_Per_Year  Salary_Per_Age
Rolling_mean
7                22433.750000     22433.750000     2243.375000
78136.25
5                23705.000000     23705.000000     1693.214286
77288.50
9                31641.666667     31641.666667     3390.178571
77096.00
8                23238.750000     23238.750000     3319.821429
77069.75
4                24305.250000     24305.250000     2558.447368
```

```
76149.25
6                 7692.250000      7692.250000      854.694444
73475.00
3                17772.500000     17772.500000     2843.600000
70376.75
0                18532.750000     18532.750000     1323.767857
NaN
1                22565.750000     22565.750000     1962.239130
NaN
2                11505.750000     11505.750000     1438.218750
NaN

[10 rows x 28 columns]
```

Q80: Create a new column 'Rating_Age_Ratio' as the ratio of 'Rating' to 'Age', then find the employee with the highest ratio and display their name and ratio.

```
df['Rating_Age_Ratio'] = df['Rating'] / df['Age']
df.nlargest(1, 'Rating_Age_Ratio')[['Name', 'Rating_Age_Ratio']]
```

```
    Name  Rating_Age_Ratio
3  David             0.088
```

Hard Questions

Q81: Find the top 3 highest-paid employees from each department.

```
df.groupby('Department').apply(lambda x:x.nlargest(3, 'Salary'))
```

```
C:\Users\91790\AppData\Local\Temp\ipykernel_10404\1168016206.py:1:
DeprecationWarning: DataFrameGroupBy.apply operated on the grouping
columns. This behavior is deprecated, and in a future version of
pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or
explicitly select the grouping columns after groupby to silence this
warning.
  df.groupby('Department').apply(lambda x:x.nlargest(3, 'Salary'))

                  Name  Age Gender  Salary Department  Join_Date
Rating  \
Department

Finance    4      Eve   38      F   97221    Finance 2020-05-31
3.1
           7   Hannah   40      F   89735    Finance 2020-08-31
3.4
           2  Charlie   32      M   46023    Finance 2020-03-31
1.7
HR         9     Jack   28      M   94925         HR 2020-10-31
2.2
```

```
            0    Alice   56      F   74131          HR 2020-01-31
1.8
            3    David   25      M   71090          HR 2020-04-30
2.2
IT          5    Frank   56      M   94820          IT 2020-06-30
2.7
            8      Ivy   28      F   92955          IT 2020-09-30
1.6
            1      Bob   46      M   90263          IT 2020-02-29
1.7

               Salary_After_Tax  median_rating  Experience  ...  \
Department                                                  ...
Finance    4          82637.85           2.2           4  ...
           7          76274.75           2.2           4  ...
           2          39119.55           2.2           4  ...
HR         9          80686.25           2.2           3  ...
           0          63011.35           2.2           4  ...
           3          60426.50           2.2           4  ...
IT         5          80597.00           2.2           4  ...
           8          79011.75           2.2           4  ...
           1          76723.55           2.2           4  ...

              Years_Till_Retirement  Salary_Avg_Salary_Diff
Dept_Mean_Rating  \
Department

Finance    4                     27                 19027.8
2.733333
           7                     25                 11541.8
2.733333
           2                     33                -32170.2
2.733333
HR         9                     37                 16731.8
2.100000
           0                      9                 -4062.2
2.100000
           3                     40                 -7103.2
2.100000
IT         5                      9                 16626.8
2.000000
           8                     37                 14761.8
2.000000
           1                     19                 12069.8
2.000000

              Age_Range  Salary_To_Experience_Ratio
Salary_Per_Year  \
Department
```

```
Finance    4      Young                 24305.250000     24305.250000
           7      Young                 22433.750000     22433.750000
           2      Young                 11505.750000     11505.750000
HR         9      Young                 31641.666667     31641.666667
           0      Young                 18532.750000     18532.750000
           3      Young                 17772.500000     17772.500000
IT         5      Young                 23705.000000     23705.000000
           8      Young                 23238.750000     23238.750000
           1      Young                 22565.750000     22565.750000


               Salary_Per_Age  Rolling_mean  Rating_Age_Ratio
Dept_Max_Salary
Department

Finance    4     2558.447368      76149.25          0.081579
NaN
           7     2243.375000      78136.25          0.085000
NaN
           2     1438.218750           NaN          0.053125
NaN
HR         9     3390.178571      77096.00          0.078571
NaN
           0     1323.767857           NaN          0.032143
NaN
           3     2843.600000      70376.75          0.088000
NaN
IT         5     1693.214286      77288.50          0.048214
NaN
           8     3319.821429      77069.75          0.057143
NaN
           1     1962.239130           NaN          0.036957
NaN

[9 rows x 30 columns]
```

Q82: Create a pivot table showing the mean salary and mean rating for each combination of department and gender.

```
pd.pivot_table(df, index='Gender', columns='Department',
values=['Salary', 'Rating'], aggfunc='mean')
```

```
            Rating              Salary
Department Finance   HR   IT  Finance       HR       IT
Gender
F             3.25  2.0  1.6  93478.0  52450.0  92955.0
M             1.70  2.2  2.2  46023.0  83007.5  92541.5
```

Q83: Calculate the cumulative salary of employees in the DataFrame and add it as a new column.

```
df['Cumulative_Salary'] = df['Salary'].cumsum()
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_After_Tax  median_rating  Experience  ...
Years_Till_Retirement  \
0          63011.35            2.2           4  ...
9
1          76723.55            2.2           4  ...
19
2          39119.55            2.2           4  ...
33
3          60426.50            2.2           4  ...
40
4          82637.85            2.2           4  ...
27

   Salary_Avg_Salary_Diff  Dept_Mean_Rating  Age_Range  \
0                 -4062.2          2.100000      Young
1                 12069.8          2.000000      Young
2                -32170.2          2.733333      Young
3                 -7103.2          2.100000      Young
4                 19027.8          2.733333      Young

   Salary_To_Experience_Ratio  Salary_Per_Year  Salary_Per_Age
Rolling_mean  \
0                    18532.75         18532.75     1323.767857
NaN
1                    22565.75         22565.75     1962.239130
NaN
2                    11505.75         11505.75     1438.218750
NaN
3                    17772.50         17772.50     2843.600000
70376.75
4                    24305.25         24305.25     2558.447368
```

```
76149.25

   Rating_Age_Ratio  Cumulative_Salary
0          0.032143              74131
1          0.036957             164394
2          0.053125             210417
3          0.088000             281507
4          0.081579             378728

[5 rows x 30 columns]
```

Q84: Create a new column 'Salary_Per_Year' by dividing the salary by the number of years since joining.

```
df['Salary_Per_Year'] = df['Salary'] / df['Experience']
```

Q85: Filter employees whose name contains the letter 'a' and who joined after 2020, then sort by age.

```
df[(df['Name'].str.contains('a')) & (df['Join_Date'] > '2020-01-
31')].sort_values(by='Age')

      Name  Age Gender  Salary Department  Join_Date  Rating  \
3    David   25      M   71090         HR 2020-04-30     2.2
9     Jack   28      M   94925         HR 2020-10-31     2.2
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
6    Grace   36      F   30769         HR 2020-07-31     2.2
7   Hannah   40      F   89735    Finance 2020-08-31     3.4
5    Frank   56      M   94820         IT 2020-06-30     2.7

   Salary_After_Tax  median_rating  Experience  ...
Years_Till_Retirement  \
3          60426.50            2.2           4  ...
40
9          80686.25            2.2           3  ...
37
2          39119.55            2.2           4  ...
33
6          26153.65            2.2           4  ...
29
7          76274.75            2.2           4  ...
25
5          80597.00            2.2           4  ...
9

   Salary_Avg_Salary_Diff  Dept_Mean_Rating  Age_Range  \
3                 -7103.2          2.100000      Young
9                 16731.8          2.100000      Young
2                -32170.2          2.733333      Young
6                -47424.2          2.100000      Young
```

```
7                 11541.8          2.733333      Young
5                 16626.8          2.000000      Young

   Salary_To_Experience_Ratio  Salary_Per_Year  Salary_Per_Age
Rolling_mean  \
3                17772.500000     17772.500000     2843.600000
70376.75
9                31641.666667     31641.666667     3390.178571
77096.00
2                11505.750000     11505.750000     1438.218750
NaN
6                 7692.250000      7692.250000      854.694444
73475.00
7                22433.750000     22433.750000     2243.375000
78136.25
5                23705.000000     23705.000000     1693.214286
77288.50

   Rating_Age_Ratio  Cumulative_Salary
3          0.088000             281507
9          0.078571             781932
2          0.053125             210417
6          0.061111             504317
7          0.085000             594052
5          0.048214             473548

[6 rows x 30 columns]
```

Q86: For each employee, calculate their percentage contribution to the total salary in their department, and display the first 5 rows.

```
df['Salary_Contribution'] = (df['Salary'] / df['Dept_Total_Salary']) *
100
df[['Name', 'Department', 'Salary', 'Dept_Total_Salary',
'Salary_Contribution']]

      Name Department  Salary  Dept_Total_Salary  Salary_Contribution
0    Alice         HR   74131             270915            27.363195
1      Bob         IT   90263             278038            32.464267
2  Charlie    Finance   46023             232979            19.754141
3    David         HR   71090             270915            26.240703
4      Eve    Finance   97221             232979            41.729512
5    Frank         IT   94820             278038            34.103252
6    Grace         HR   30769             270915            11.357437
7   Hannah    Finance   89735             232979            38.516347
8      Ivy         IT   92955             278038            33.432480
9     Jack         HR   94925             270915            35.038665
```

Q87: Create a pivot table that shows the average salary and rating for each combination of department and gender.

```
pd.pivot_table(df, index='Gender', columns='Department',
values=['Salary','Rating'], aggfunc='mean')
```

```
            Rating                Salary
Department Finance   HR   IT  Finance       HR       IT
Gender
F             3.25  2.0  1.6  93478.0  52450.0  92955.0
M             1.70  2.2  2.2  46023.0  83007.5  92541.5
```

Q88: Display the top 3 employees in terms of salary for each department, including their names and salaries.

```
df.groupby('Department').apply(lambda x:x.nlargest(3, 'Salary'))
[['Name', 'Salary']]
```

```
C:\Users\91790\AppData\Local\Temp\ipykernel_10404\1334172920.py:1:
DeprecationWarning: DataFrameGroupBy.apply operated on the grouping
columns. This behavior is deprecated, and in a future version of
pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or
explicitly select the grouping columns after groupby to silence this
warning.
  df.groupby('Department').apply(lambda x:x.nlargest(3, 'Salary'))
[['Name', 'Salary']]
```

```
                  Name  Salary
Department
Finance    4       Eve   97221
           7    Hannah   89735
           2   Charlie   46023
HR         9      Jack   94925
           0     Alice   74131
           3     David   71090
IT         5     Frank   94820
           8       Ivy   92955
           1       Bob   90263
```

Q84: Calculate the cumulative sum of salaries within each department and create a new column 'Cumulative_Salary'. Display the first 5 rows.

```
df['Cumulative_Salary'] = df.groupby('Department')['Salary'].cumsum()
df.head()
```

```
      Name  Age Gender  Salary Department  Join_Date  Rating  \
0    Alice   56      F   74131         HR 2020-01-31     1.8
1      Bob   46      M   90263         IT 2020-02-29     1.7
2  Charlie   32      M   46023    Finance 2020-03-31     1.7
3    David   25      M   71090         HR 2020-04-30     2.2
```

```
4      Eve   38      F   97221    Finance 2020-05-31     3.1

   Salary_After_Tax  median_rating  Experience  ...
Salary_Avg_Salary_Diff  \
0          63011.35            2.2           4  ...                -
4062.2
1          76723.55            2.2           4  ...
12069.8
2          39119.55            2.2           4  ...                -
32170.2
3          60426.50            2.2           4  ...                -
7103.2
4          82637.85            2.2           4  ...
19027.8

   Dept_Mean_Rating  Age_Range  Salary_To_Experience_Ratio
Salary_Per_Year  \
0          2.100000      Young                     18532.75
18532.75
1          2.000000      Young                     22565.75
22565.75
2          2.733333      Young                     11505.75
11505.75
3          2.100000      Young                     17772.50
17772.50
4          2.733333      Young                     24305.25
24305.25

   Salary_Per_Age  Rolling_mean  Rating_Age_Ratio
Cumulative_Salary  \
0     1323.767857           NaN          0.032143               74131

1     1962.239130           NaN          0.036957               90263

2     1438.218750           NaN          0.053125               46023

3     2843.600000      70376.75          0.088000              145221

4     2558.447368      76149.25          0.081579              143244


   Salary_Contribution
0            27.363195
1            32.464267
2            19.754141
3            26.240703
4            41.729512

[5 rows x 31 columns]
```

Q85: Display employees who are in the top 20% by age and have a rating above 2.5.

```
df['Top_20%_By_Age'] = df['Age'].quantile(0.8)
df[(df['Rating'] > 2.5) & (df['Top_20%_By_Age'])]
```

```
     Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
4     Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
5   Frank   56      M   94820         IT 2020-06-30     2.7
80597.00
7  Hannah   40      F   89735    Finance 2020-08-31     3.4
76274.75

   median_rating  Experience  ... Dept_Mean_Rating  Age_Range  \
4            2.2           4  ...         2.733333      Young
5            2.2           4  ...         2.000000      Young
7            2.2           4  ...         2.733333      Young

   Salary_To_Experience_Ratio  Salary_Per_Year  Salary_Per_Age
Rolling_mean  \
4                    24305.25         24305.25     2558.447368
76149.25
5                    23705.00         23705.00     1693.214286
77288.50
7                    22433.75         22433.75     2243.375000
78136.25

   Rating_Age_Ratio  Cumulative_Salary  Salary_Contribution
Top_20%_By_Age
4          0.081579             143244            41.729512
48.0
5          0.048214             185083            34.103252
48.0
7          0.085000             232979            38.516347
48.0

[3 rows x 32 columns]
```

Q86: For each department, calculate the mean salary, standard deviation of salaries, and the number of employees. Then filter departments where the mean salary exceeds 60,000 and the standard deviation is less than 10,000.

```
df.groupby('Department')['Salary'].agg(['mean', 'std',
'count']).query('mean > 60000 & std < 10000')
```

```
                    mean          std  count
Department
IT          92679.333333  2290.972792      3
```

Q87: Find the employee(s) with the highest 'Salary_Per_Year' ratio for each department and display their name, salary, and ratio.

```
df['Salary_Per_Year'] = df['Salary'] / df['Experience']
df.groupby('Department').apply(lambda x:x.nlargest(1,
'Salary_Per_Year'))[['Name', 'Salary', 'Salary_Per_Year']]
```

```
C:\Users\91790\AppData\Local\Temp\ipykernel_10404\1252141479.py:2:
DeprecationWarning: DataFrameGroupBy.apply operated on the grouping
columns. This behavior is deprecated, and in a future version of
pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or
explicitly select the grouping columns after groupby to silence this
warning.
  df.groupby('Department').apply(lambda x:x.nlargest(1,
'Salary_Per_Year'))[['Name', 'Salary', 'Salary_Per_Year']]
```

		Name	Salary	Salary_Per_Year
Department				
Finance	4	Eve	97221	24305.250000
HR	9	Jack	94925	31641.666667
IT	5	Frank	94820	23705.000000

Q88: Create a pivot table showing the average salary and total salary for each combination of department and gender.

```
pd.pivot_table(df, index='Gender', columns='Department',
values='Salary', aggfunc=['mean', 'count'])
```

	mean			count		
Department	Finance	HR	IT	Finance	HR	IT
Gender						
F	93478.0	52450.0	92955.0	2	2	1
M	46023.0	83007.5	92541.5	1	2	2

Q89: Calculate the rolling mean of salaries over a window of 5 employees, sorted by 'Join_Date', and display the first 10 results.

```
df.sort_values(by='Join_Date')
['Salary'].rolling(window=4).mean().head(10)
```

```
0         NaN
1         NaN
2         NaN
3    70376.75
4    76149.25
5    77288.50
6    73475.00
7    78136.25
8    77069.75
```

```
9    77096.00
Name: Salary, dtype: float64
```

Q90: Create a new column 'Salary_to_Age_Ratio' by dividing the salary by age, and then find the employee with the highest ratio.

```
df['Salary_to_Age_Ratio'] = df['Salary'] / df['Age']
df.nlargest(1, 'Salary_to_Age_Ratio')
```

```
   Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
9  Jack   28      M   94925         HR 2020-10-31     2.2
80686.25

   median_rating  Experience  ... Age_Range
Salary_To_Experience_Ratio  \
9            2.2           3  ...     Young
31641.666667

   Salary_Per_Year  Salary_Per_Age  Rolling_mean  Rating_Age_Ratio  \
9     31641.666667     3390.178571       77096.0          0.078571

   Cumulative_Salary  Salary_Contribution  Top_20%_By_Age
Salary_to_Age_Ratio
9             270915            35.038665            48.0
3390.178571

[1 rows x 33 columns]
```

Q91: Display the names and salaries of the bottom 10% of employees by salary, and calculate the total salary of these employees.

```
Salary_Threshold = df['Salary'].quantile(0.1)
df[df['Salary'] < Salary_Threshold][['Salary', 'Name']]
```

```
   Salary   Name
6   30769  Grace
```

Q92: For each employee, calculate their percentage contribution to the total salary in their department and display the first 10 rows with 'Name', 'Department', and 'Salary_Contribution'.

```
df[['Name', 'Department', 'Salary', 'Salary_Contribution',]]
```

```
      Name Department  Salary  Salary_Contribution
0    Alice         HR   74131            27.363195
1      Bob         IT   90263            32.464267
2  Charlie    Finance   46023            19.754141
3    David         HR   71090            26.240703
4      Eve    Finance   97221            41.729512
```

```
5    Frank         IT   94820            34.103252
6    Grace         HR   30769            11.357437
7   Hannah    Finance   89735            38.516347
8      Ivy         IT   92955            33.432480
9     Jack         HR   94925            35.038665
```

Q93: Calculate the ratio of each employee's salary to the average salary in their department, and filter employees with a ratio greater than 1.2.

```
df['Dept_Avg_Salary'] = df.groupby('Department')
['Salary'].transform('mean')
Salary_Dept_Avg_Salary_Ratio = df['Salary'] / df['Dept_Avg_Salary']
Salary_Dept_Avg_Salary_Ratio[Salary_Dept_Avg_Salary_Ratio > 1.2]
```

```
4    1.251885
9    1.401547
dtype: float64
```

Q94: Create a rolling window of 5 employees based on their 'Join_Date', and for each window, calculate the average age and display the results.

```
df.sort_values('Join_Date')['Age'].rolling(window=5).mean()
```

```
0     NaN
1     NaN
2     NaN
3     NaN
4    39.4
5    39.4
6    37.4
7    39.0
8    39.6
9    37.6
Name: Age, dtype: float64
```

Q95: Group employees by 'Department' and calculate the mean, standard deviation, and total salary. Filter departments where the mean salary is greater than 60,000 and the standard deviation is less than 12,000.

```
df.groupby('Department')['Salary'].agg(['mean', 'std',
'sum']).query('mean > 60000 & std > 12000')
```

```
                    mean           std     sum
Department
Finance     77659.666667  27652.650096  232979
HR          67728.750000  26820.054317  270915
```

Q96: For each department, find the employee with the highest 'Salary_Per_Year' ratio and display their name, department, and salary.

```
df.nlargest(1, 'Salary_Per_Year')[['Name', 'Department', 'Salary']]
```

```
   Name Department  Salary
9  Jack         HR   94925
```

Q97: Calculate the percentage of employees in each gender within each department who earn more than 50,000.

Q98: Calculate the rolling mean of 'Rating' over a window of 5 employees sorted by 'Join_Date', and display the top 5 results.

```
df.sort_values(by='Join_Date')
['Rating'].rolling(window=5).mean().head()
```

```
0    NaN
1    NaN
2    NaN
3    NaN
4    2.1
Name: Rating, dtype: float64
```

Q99: For each employee, calculate the percentage contribution of their salary to the total salary of their department, and display the first 10 rows.

```
df['Salary_Contribution']
```

```
0    27.363195
1    32.464267
2    19.754141
3    26.240703
4    41.729512
5    34.103252
6    11.357437
7    38.516347
8    33.432480
9    35.038665
Name: Salary_Contribution, dtype: float64
```

Q100: Identify the top 2 departments with the highest total salary, and display their total salary and employee count.

```
df.groupby('Department').agg({'Salary':'sum',
'Name':'count'}).nlargest(2, 'Salary')
```

```
            Salary  Name
Department
IT          278038     3
HR          270915     4
```

Q101: Calculate the ratio of each employee's salary to the average salary in their department, and filter employees with a ratio greater than 1.2.

```
df['Salary_Dept_Avg_Salary_Ratio'] = df['Salary'] /
df['Dept_Avg_Salary']
df[df['Salary_Dept_Avg_Salary_Ratio'] > 1.2]
```

```
   Name  Age Gender  Salary Department  Join_Date  Rating
Salary_After_Tax  \
4   Eve   38      F   97221    Finance 2020-05-31     3.1
82637.85
9  Jack   28      M   94925         HR 2020-10-31     2.2
80686.25

   median_rating  Experience  ... Salary_To_Experience_Ratio
Salary_Per_Year  \
4            2.2           4  ...                24305.250000
24305.250000
9            2.2           3  ...                31641.666667
31641.666667

   Salary_Per_Age  Rolling_mean  Rating_Age_Ratio
Cumulative_Salary  \
4     2558.447368      76149.25          0.081579              143244

9     3390.178571      77096.00          0.078571              270915


   Salary_Contribution  Top_20%_By_Age  Salary_to_Age_Ratio  \
4            41.729512            48.0          2558.447368
9            35.038665            48.0          3390.178571

   Salary_Dept_Avg_Salary_Ratio
4                      1.251885
9                      1.401547

[2 rows x 34 columns]
```

Q102: Create a pivot table that shows the count of employees and the mean salary, grouped by both 'Department' and 'Gender'.

```
pd.pivot_table(df, index='Gender', columns='Department',
values='Salary', aggfunc=['mean', 'count'])
```

```
               mean                     count
Department  Finance       HR       IT Finance HR IT
Gender
F           93478.0  52450.0  92955.0       2  2  1
M           46023.0  83007.5  92541.5       1  2  2
```

Q103: Calculate the rolling mean of 'Age' over a window of 5 employees based on their 'Join_Date', and display the results.

```
df.sort_values(by='Join_Date')['Age'].rolling(window=5).mean()
```

```
0     NaN
1     NaN
2     NaN
3     NaN
4    39.4
5    39.4
6    37.4
7    39.0
8    39.6
9    37.6
Name: Age, dtype: float64
```