1.Student Grades Analysis Using NumPy Arrays

```
#Create a 2D NumPy array for student marks in 3 subjects
import numpy as np
student_marks=np.array([[60,70,90],[85,90,92],[89,95,90],[89,91,95],[94,78,88]])
print(student_marks)
```

```
[[60 70 90]
 [85 90 92]
 [89 95 90]
 [89 91 95]
 [94 78 88]]
```

```
#Calculate average, minimum, and maximum marks
average_marks = np.mean(student_marks, axis=0)
min_marks = np.min(student_marks, axis=0)
max_marks = np.max(student_marks, axis=0)
print(f"\nAverage Marks: {average_marks}")
print(f"Minimum Marks: {min_marks}")
print(f"Maximum Marks: {max_marks}")
```

```

Average Marks: [83.4 84.8 91. ]
Minimum Marks: [60 70 88]
Maximum Marks: [94 95 95]
```

```
#Use slicing to retrieve marks of specific students
first_student_marks = student_marks[0, :]
print(f"\nMarks for the first student: {first_student_marks}")
second_third_students_marks = student_marks[1:3, :]
print(f"Marks for the second and third students:\n{second_third_students_marks}")
last_student_marks = student_marks[-1, :]
print(f"Marks for the last student: {last_student_marks}")
```

```

Marks for the first student: [60 70 90]
Marks for the second and third students:
[[85 90 92]
 [89 95 90]]
Marks for the last student: [94 78 88]
```

```
#Use boolean indexing to find students scoring above 80
students_above_80 = student_marks[student_marks > 80]
print(f"\nmarks above 80:\n{students_above_80}")
```

```

marks above 80:
[90 85 90 92 89 95 90 89 91 95 94 88]
```

```
#Calculate average, minimum, and maximum marks for each subject
import numpy as np
#Reshape the array to observe subject-wise performance
subject_wise_performance = student_marks.T
average_subject_marks = np.mean(subject_wise_performance, axis=1)
min_subject_marks = np.min(subject_wise_performance, axis=1)
max_subject_marks = np.max(subject_wise_performance, axis=1)
print(f"\nAverage Marks per Subject: {average_subject_marks}")
print(f"Minimum Marks per Subject: {min_subject_marks}")
print(f"Maximum Marks per Subject: {max_subject_marks}")
```

```

Average Marks per Subject: [83.4 84.8 91. ]
Minimum Marks per Subject: [60 70 88]
Maximum Marks per Subject: [94 95 95]
```

```
#Comment on the observed results
print("Commentary on Subject Performance:")
print(f"Average Marks per Subject: {average_subject_marks}")
print(f"Minimum Marks per Subject: {min_subject_marks}")
print(f"Maximum Marks per Subject: {max_subject_marks}")
print("\nObservations:")
print("- The third subject has the highest average mark.")
print("- The first subject has the lowest minimum mark, suggesting a wider spread of scores in that subject.")
print("- All subjects have high maximum marks, indicating strong performance at the top end.")
```

```
Commentary on Subject Performance:
Average Marks per Subject: [83.4 84.8 91. ]
Minimum Marks per Subject: [60 70 88]
Maximum Marks per Subject: [94 95 95]
```

```
Observations:
- The third subject has the highest average mark.
- The first subject has the lowest minimum mark, suggesting a wider spread of scores in that subject.
- All subjects have high maximum marks, indicating strong performance at the top end.
```

2.Employee Salary Analysis with pandas DataFrame

```
#Create a pandas DataFrame with employee Name, ID,Department, and Salary
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'ID': [101, 102, 103, 104, 105],
        'Department': ['HR', 'IT', 'Finance', 'Marketing', 'IT'],
        'Salary': [60000, 75000, 80000, 65000, 70000]}

employee_df = pd.DataFrame(data)
display(employee_df)
```

	Name	ID	Department	Salary
0	Alice	101	HR	60000
1	Bob	102	IT	75000
2	Charlie	103	Finance	80000
3	David	104	Marketing	65000
4	Eve	105	IT	70000

```
#Filter employees with salary > 50,000
employees_above_50k = employee_df[employee_df['Salary'] > 50000]
display(employees_above_50k)
```

	Name	ID	Department	Salary
0	Alice	101	HR	60000
1	Bob	102	IT	75000
2	Charlie	103	Finance	80000
3	David	104	Marketing	65000
4	Eve	105	IT	70000

```
#Sort the DataFrame by salary in descending order
sorted_employee_df = employee_df.sort_values(by='Salary', ascending=False)
display(sorted_employee_df)
```

	Name	ID	Department	Salary
2	Charlie	103	Finance	80000
1	Bob	102	IT	75000
4	Eve	105	IT	70000
3	David	104	Marketing	65000
0	Alice	101	HR	60000

```
#Add a new column for Bonus (10% of salary)
sorted_employee_df['Bonus'] = sorted_employee_df['Salary'] * 0.10
display(sorted_employee_df)
```

	Name	ID	Department	Salary	Bonus
2	Charlie	103	Finance	80000	8000.0
1	Bob	102	IT	75000	7500.0
4	Eve	105	IT	70000	7000.0
3	David	104	Marketing	65000	6500.0
0	Alice	101	HR	60000	6000.0

```
#Calculate total salary expense including bonuses
total_salary_expense = sorted_employee_df['Salary'].sum() + sorted_employee_df['Bonus'].sum()
print(f"Total salary expense including bonuses: {total_salary_expense}")
```

```
Total salary expense including bonuses: 385000.0
```

Start coding or generate with AI.

```
# Save the DataFrame to a CSV file
sorted_employee_df.to_csv('employee_data_with_bonus.csv', index=False)
print("DataFrame saved to employee_data_with_bonus.csv")
```

```
DataFrame saved to employee_data_with_bonus.csv
```