Student Exam Data Processing and Report Generation

Scenario

A university system stores raw student exam data as unstructured strings.
The system must clean, structure, and analyze this data to identify high performers and generate a formatted summary report.

Requirements & Logic

1. **Data Structure Setup**
    - Initialize a List `Raw_Student_Data` containing unstructured student records. Example:

```
['John: Math, 92: pass', 'Alice: Science, 78: fail', 'Bob: Math, 89: pass']
```

2. **Structuring Data (Loop/String Methods)**
    - Initialize an empty list `Structured_Student_List`
    - For each record in `Raw_Student_Data`:
        - Split the string by ":" and "," to extract Name, Subject, Marks, and Status.
        - Use `.strip()` to clean whitespace.
        - Create a tuple `(Name, Subject, Marks, Status)` and append to the list.

5. **Analysis and Reporting (Loop/Slicing/Condition)**
    - Create a Set `High_Scorers` that stores tuples of students with Marks ≥ 85.
    - Use slicing to print the first two structured student records.
    - Generate a report for the top student using an f-string, converting the name to uppercase using `.upper()`.

Python Code Implementation

```
# 1. Data Setup
Raw_Student_Data = [
    'John: Math, 92: pass',
    'Alice: Science, 78: fail',
    'Bob: Math, 89: pass'
]

# 2. Structuring Data
```

```
Structured_Student_List = []

for record in Raw_Student_Data:
    parts = record.replace(',', ':').split(':')
    Name = parts[0].strip()
    Subject = parts[1].strip()
    Marks = int(parts[2].strip())
    Status = parts[3].strip()
    Structured_Student_List.append((Name, Subject, Marks, Status))

# 3. Analysis and Reporting
High_Scorers = set()
for student in Structured_Student_List:
    if student[2] >= 85:
        High_Scorers.add(student)

# Print last two records using slicing
print("Last two student records:", Structured_Student_List[-2:])

# Generate formatted report
first_student = Structured_Student_List[0]
report = f"STUDENT REPORT: {first_student[0].upper()} scored
{first_student[2]} in {first_student[1]}"
print(report)

# Summary
print("High Scorers Set:", High_Scorers)
```

Expected Output

```
Last two student records: [('Alice', 'Science', 78, 'fail'),
('Bob', 'Math', 89, 'pass')]
STUDENT REPORT: JOHN scored 92 in Math
High Scorers Set: {('Bob', 'Math', 89, 'pass'), ('John',
'Math', 92, 'pass')}
```

Employee Task Monitoring and Performance Summary

Scenario

An HR department tracks employees' work updates as text strings.
The system needs to clean the text, structure it into tuples, and generate a report of pending tasks and top performers.

Requirements & Logic

1. **Data Structure Setup**

Initialize a List `Raw_Employee_Data` containing unstructured strings.
Example:

['E101: Dev, completed 5 tasks: active',

```
'E102: QA, completed 2 tasks: pending',
'E103: Dev, completed 8 tasks: active']
```

2. **Structuring Data (Loop/String Methods)**
    - Initialize an empty list `Structured_Employee_List`.
    - Split each string using ":" and "," to extract Employee ID, Role, Tasks Completed, and Status.
    - Clean data using `.strip()` and create tuples like `(EmpID, Role, Tasks, Status)`.

7. **Analysis and Reporting (Loop/Slicing/Condition)**
    - Create a Set `Pending_Employees` to store all employees whose status contains "pending".
    - Use slicing to print the first two structured employee entries.
    - Use an f-string with `.upper()` to print a summary report for the last employee in uppercase.

Python Code Implementation

```
# 1. Data Setup
Raw_Employee_Data = [
    'E101: Dev, completed 5 tasks: active',
    'E102: QA, completed 2 tasks: pending',
    'E103: Dev, completed 8 tasks: active'
]

# 2. Structuring Data
Structured_Employee_List = []

for entry in Raw_Employee_Data:
    parts = entry.replace(',', ':').split(':')
    EmpID = parts[0].strip()
    Role = parts[1].strip()
    Tasks = parts[2].replace('completed', '').replace('tasks',
'').strip()
    Status = parts[3].strip()
    Structured_Employee_List.append((EmpID, Role, int(Tasks),
Status))
```

```python
# 3. Analysis and Reporting
Pending_Employees = set()
for emp in Structured_Employee_List:
    if 'pending' in emp[3].lower():
        Pending_Employees.add(emp)

# Print first two employees using slicing
print("First two employee records:", Structured_Employee_List[:2])

# Generate formatted report for last employee
last_emp = Structured_Employee_List[-1]
report = f"EMPLOYEE SUMMARY: {last_emp[0].upper()} from
{last_emp[1]} team completed {last_emp[2]} tasks."
print(report)

# Summary Output
print("Pending Employees Set:", Pending_Employees)
```

Expected Output

```
First two employee records: [('E101', 'Dev', 5, 'active'),
('E102', 'QA', 2, 'pending')]
EMPLOYEE SUMMARY: E103 from Dev team completed 8 tasks.
Pending Employees Set: {('E102', 'QA', 2, 'pending')}
```