Employee Data Management with Python

Project Overview: In this project, I worked on managing employee data and performed basic data analysis using Python. The task involved storing employee information, counting employees by department, and displaying unique departments using lists, sets, and dictionaries.

1. Storing Employee Data: I used a list of tuples to store employee information, where each tuple contains the following details:

Name

Department

Years of Experience

Example:

```
employees = [
    ("Alice", "IT", 3),
    ("Bob", "Finance", 5),
    ("Charlie", "IT", 1),
    ("David", "HR", 4),
]
```

2. Unique Departments Using Set: I used a set comprehension to extract and display unique departments from the employee data. The set automatically removes any duplicate entries.

Code:

```
departments = {emp[1] for emp in employees}
print("Departments:", departments)
```

Output:

```
Departments: {'IT', 'Finance', 'HR'}
```

3. Counting Employees by Department: Using a dictionary, I counted the number of employees in each department. The dictionary stored the department name as the key and the count of employees as the value.

Code:

```
employee_count_by_dept = {}
for emp in employees:
    department = emp[1]
    if department in employee_count_by_dept:
        employee_count_by_dept[department] += 1
    else:
        employee_count_by_dept[department] = 1

print("Employee Count by Department:", employee_count_by_dept)
```

Output:

```
Employee Count by Department: {'IT': 2, 'Finance': 1, 'HR': 1}
```

4. Key Takeaways:

- I demonstrated how to use Python's data structures (lists, sets, and dictionaries) to organize and analyze data.
- Set comprehensions provided an efficient way to extract unique values from a list.
- Dictionaries helped in counting and grouping data based on specific criteria (e.g., departments).
- This project enhanced my skills in data handling and processing using Python.

Full Code

```
# 1. Storing employee data - each employee’s data is in a tuple, stored within a list
employees = [
    ("Alice", "IT", 3),  # each tuple contains (name, department, years of experience)
    ("Bob", "Finance", 5),
    ("Charlie", "IT", 1),
    ("David", "HR", 4),
]

# 2. Using a set to show unique departments from the employee list
departments = {emp[1] for emp in employees}
print("Departments:", departments)

# 3. Counting the number of employees in each department using a dictionary
employee_count_by_dept = {}
for emp in employees:
    department = emp[1]
    if department in employee_count_by_dept:
        employee_count_by_dept[department] += 1
    else:
        employee_count_by_dept[department] = 1
```

```
print("Employee Count by Department:", employee_count_by_dept)

# 4. Adding a new employee to the employee list
new_employee = ("Emma", "Marketing", 2)
employees.append(new_employee)
print("Updated Employee List:", employees)

# 5. Finding employees based on years of experience (e.g., employees with 3 years of experience)
target_years = 3
filtered_employees = [emp for emp in employees if emp[2] == target_years]
print(f"Employees with {target_years} years of experience:", filtered_employees)
```