Output :-

```
{'id': 2, 'name': 'bob', 'department': 'engineering'}
```

5. Implementation of various Searching and sorting operations in python programming

5.1 A company stores employee records in a list of dictionaries, where each dictionary contains id, name, department. Write a function find_employee_by_id that takes this list and a target employee ID as arguments and returns the dictionary of the employee with the matching ID, or None if no such employee is found.

Aim :- To implement the details of the employee using various Searching and Sorting Operations in

Algorithm :-

1. Input Definition:
2. Define the function find_employee_by_id that take two parameters:
   a. A list of dictionaries (employees), where each dictionary represents an employee record with keys id, name and department.
   b. An integer (target_id) representing the employee ID to be searched.
3. Iterate Through the list:
   Use a for loop to iterate through each dictionary in the employees list.
4. Check for Matching ID:
   Within the loop, check if the id field of the current dictionary matches the target_Id.
5. Return Matching Record:
   If a match is found, return the current dictionary.
6. Handle No match:
   If the loop completes without finding a match, return None

Program :-

```
def find_employee_by_id(employees, target_id):
    for employee in employees:
        if employee['id'] == target_id:
            return employee
    return None
# Test the function
employees = [
    {'id': 1, 'name': 'Alice', 'department': 'HR'},
    {'id': 2, 'name': 'Bob', 'department': 'Engineering'},
    {'id': 3, 'name': 'Charlie', 'department': 'Sales'},
]
```

Output:-

```
Before Sorting
{'name': 'alice', 'score': 88}
{'name': 'bob', 'score': 95}
{'name': 'Charlie', 'score': 75}

{'name': 'diana', 'score': 85}
~~After~~ Sorting
[{'name': 'alice', 'score': 88}
 {'name': 'bob', 'score': 95}
 {'name': 'charlie', 'score': 75}
 {'name': 'diana', 'score': 85}]
```

5.2 You are developing a grade management system for a School. The system maintains a list of student records, where each record is represented as a dictionary contain-
-ning a student's name and score. The school needs to generate a report that display students' scores in ascending order.
a report that display Student s' Scores in ascending order. a report that displays
Your task is to generate a report that displays Sorts the students records by their scores using Bubble sort algorithm.

Aim:- To implement a feature that sorts the student records by their ascending scores using the Bubble sort algorithm.

Algorithm:- 1. Initialization:
• Get the length of the students list and store it in n.
2. Outer loop:
• Iteration from i=0 to n-1. This loop represents the number of passes through the list.
3. Track swaps:
• Initialize a boolean variable swapped to False. This loop compares adjacent emplements in the list and per-
-forms swaps if necessary.
4. Inner loop:
• Iterate from j=0 to n-i-2 (inclusive). This loop compares adjacent elements in the list and performs swaps if necessary.
5. Compare and Swap:
• For each pair of adjacent elements (i.e., Students[j] and Students[j+1]):
• Compare their score values.
• If students[j]['score'] > students[j+1]['score'], swap the two elements.
• Set Swapped to True to indicate that a swap was made.
6. Early Termination:
• After each pass of the inner loop, check if swapped is False. If no swaps were made during the pass, the list is already sorted, and you can break out of the outer loop early.

7. Completion:
- The function modifies the students list in place, sorting it by score

Program:

```
def bubble-sort-scores(students):
  n= len(students)
  for i in range(n):
    # Track if any swap is made in this pass.
    swapped = False
    for j in range(0,n-i-1):
      if students[j]['score'] >students[j+1]['score']:
        # Swap if the score of the current student is grea-
              -ter than the next
        students[j], students[j+1] = students[j+1], students[j]
        swapped = True
    # if no two elements were swapped, the list is already
      sorted.
    if not swapped:
      break
  # Example usage
  students = [
    {'name': 'Alic', score': 88},
    {'name': 'Bob', 'score': 95},
    {'name': 'Charlie', 'score':75},
    {name': 'Diana', 'Score': 85},
  ]
  print("Before Sorting:")
  for student in students:
    print(student)
  bubble_sort_scores(students)
  print("\nAfter Sorting:")
  for student in students:
    print(student)
```

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
DATE	

Result:- Thus the implementation of various searching and sorting operations in python programs was executed.