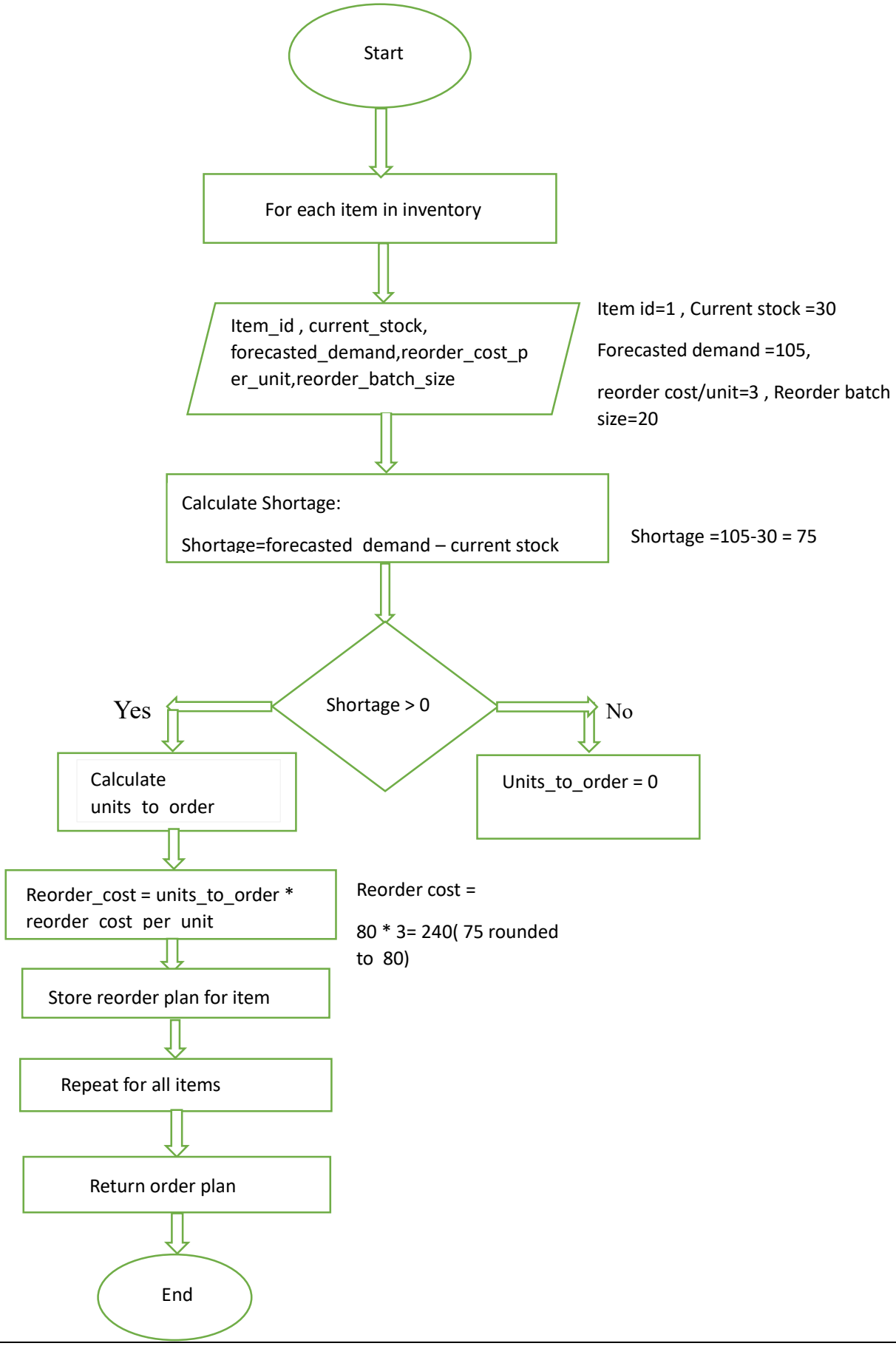
Task 1: Algorithm Development Task: Inventory Recording System



Algorithm Explanation

Let explain the algorithm with an example

Begin the process for analyzing the inventory to calculate reorder quantities.

Each item in the inventory

item_id

current_stock (number of items currently in stock)

forecasted_demand (estimated demand for the item)

reorder_cost_per_unit (cost per unit to reorder)

reorder_batch_size (minimum units that must be ordered in one batch).

Here,

Item id=1 , Current stock =30

Forecasted demand =105,

reorder cost/unit=3 , Reorder batch size=20

To Calculate shortage:

Shortage = Forecasted_demand – Current stock

Shortage = 105 - 30 = 75

Units to Order = Round 75 to next multiple of 20(Reorder batch size) → 80

If units_to_order > 0:

Use the formula:
reorder_cost = units_to_order * reorder_cost_per_unit

Reorder Cost = 80 * 3 = 240

End
Terminate the process after processing all inventory items.

Database Task: Employee Management System

1.Database Schema :

Employee Table

SQLQuery12.sql -...-D1O9O7EI\HP (51)) LAPTOP-D1O9O7EI\S...ns - dbo.Salaries

Column Name	Data Type	Allow Nulls
EmployeeID	int	☐
Name	nvarchar(50)	☐
DepartmentID	int	☐
HireDate	date	☐
		☐

SQLQuery13.sql -...-D1O9O7EI\HP (52))* SQLQuery12.sql -...-D1O9O7EI\H

```
insert into Employees values('Sunil',3,'2021-07-03')
select * from Employees
```

100 %

Results Messages

	EmployeeID	Name	DepartmentID	HireDate
1	1	Manu	2	2022-10-23
2	2	Renu	1	2022-03-02
3	3	Jasmine	4	2021-09-19
4	4	Sunil	3	2021-07-03

Departments Table

SQLQuery13.sql -...-D1O9O7EI\HP (52))* SQLQuery12.sql -...-D1O9O7EI\HP (51))

Column Name	Data Type	Allow Nulls
DepartmentID	int	☐
DepartmentName	nvarchar(50)	☐
		☐

SQLQuery13.sql -...-D1O9O7EI\HP (52))* SQLQuery12.sql -...-D

```
insert into Departments values(4,'Sales')
select * from Departments
```

100 %

Results Messages

	DepartmentID	DepartmentName
1	1	Finance
2	2	HR
3	3	Communication
4	4	Sales

Salaries Table

SQLQuery13.sql -...-D1O9O7EI\HP (52))* SQLQuery12.sql -...-D1O9O7EI\HP (51))* LAPT(

Column Name	Data Type	Allow Nulls
EmployeeID	int	☐
BaseSalary	int	☐
Bonus	int	☐
Deductions	int	☐
		☐

SQLQuery13.sql -...-D1O9O7EI\HP (52))* SQLQuery12.sql -...-D1O9(

```
insert into Salaries values(4,75000,6000,3000)
select * from Salaries
```

100 %

Results Messages

	EmployeeID	BaseSalary	Bonus	Deductions
1	1	50000	2000	1500
2	2	65000	5000	2500
3	3	40000	2000	1000
4	4	75000	6000	3000

2.SQL Queries

- List all employees along with their department names

SQLQuery13.sql -...-D1O9O7EI\HP (52))* | SQLQuery12.sql -...-D1O9O7EI\HP (

```sql
SELECT e.EmployeeID, e.Name, d.DepartmentName
FROM Employees e
JOIN Departments d ON e.DepartmentID = d.DepartmentID;
```

100 %

Results | Messages

	EmployeeID	Name	DepartmentName
1	1	Manu	HR
2	2	Renu	Finance
3	3	Jasmine	Sales
4	4	Sunil	Communication

- Calculate the net salary for each employee using:
  Net Salary = Base salary + Bonus -Deductions

```sql
SELECT e.EmployeeID, e.Name, s.BaseSalary, s.Bonus, s.Deductions,
       (s.BaseSalary + s.Bonus - s.Deductions) AS NetSalary
FROM Employees e
JOIN Salaries s ON e.EmployeeID = s.EmployeeID;
```

) %

Results | Messages

EmployeeID	Name	BaseSalary	Bonus	Deductions	NetSalary
1	Manu	50000	2000	1500	50500
2	Renu	65000	5000	2500	67500
3	Jasmine	40000	2000	1000	41000
4	Sunil	75000	6000	3000	78000

- Identify the department with the highest average salary.

```sql
SELECT d.DepartmentName, AVG(s.BaseSalary + s.Bonus - s.Deductions) AS AverageSalary
FROM Employees e
JOIN Salaries s ON e.EmployeeID = s.EmployeeID
JOIN Departments d ON e.DepartmentID = d.DepartmentID
GROUP BY d.DepartmentName
ORDER BY AverageSalary DESC;
```

100 %

Results | Messages

	DepartmentName	AverageSalary
1	Communication	78000
2	Finance	67500
3	HR	50500
4	Sales	41000

3.Stored Procedure

A procedure to insert a new employee into the Employees table ,ensuring valid DepartmentID and other constarints

```
SQLQuery16.sql -...-D1O9O7EI\HP (66))*        SQLQuery15.sql -...-D1O9O7EI\HP (67))*        SQLQuery
create procedure InsertEmployee
    @Name nvarchar(50),
    @DepartmentID int,
    @HireDate date,
    @BaseSalary int,
    @Bonus int,
    @Deductions int

    as
    begin

    begin try
        begin transaction;

     insert into Employees (Name, DepartmentID, HireDate)
        values (@Name, @DepartmentID, @HireDate);

        declare @NewEmployeeID int = SCOPE_IDENTITY();

        insert into Salaries (EmployeeID, BaseSalary, Bonus, Deductions)
       values (@NewEmployeeID, @BaseSalary, @Bonus, @Deductions);

       commit transaction;
       end try
       begin catch

        if @@TRANCOUNT > 0
            rollback transaction;
            throw;
    end catch;
    end;
```

```
SQLQuery16.sql -...-D1O9O7EI\HP (66))*        SQLQu
EXEC InsertEmployee
    @Name = 'Angel',
    @DepartmentID = 4,
    @HireDate = '2022-10-17',
    @BaseSalary = 50000,
    @Bonus = 5000,
    @Deductions = 2000;

100 %
Messages

(1 row affected)

(1 row affected)
```

- A procedure to update the salary details of an employee

SQLQuery17.sql -...-D1O9O7EI\HP (68))* | SQLQuery16.sql -...-D1O9O7EI\HP (66))* | SQLQuery15.sql -...-D1O9O7EI\HP (67))* | SQLQuer

```sql
CREATE procedure UpdateSalary
@EmployeeID int,
@BaseSalary int,
@Bonus int,
@Deductions int
as
begin

begin try
        begin transaction;
update Salaries set BaseSalary=@BaseSalary,Bonus=@Bonus,Deductions=@Deductions where EmployeeId=@EmployeeID;
commit transaction;
    end try
    begin catch

        if @@TRANCOUNT > 0
            rollback transaction;
        throw;
    end catch;
end;
```

SQLQuery18.sql -...-D1O9O7EI\HP (59))* | SQLQuery

```sql
EXEC UpdateSalary

    @EmployeeID = '1',
    @BaseSalary= 66000,
    @Bonus = 3000,
    @Deductions = 2000;

    select * from Salaries
```

100 %

Results | Messages

	EmployeeID	BaseSalary	Bonus	Deductions
1	1	66000	3000	2000
2	2	65000	5000	2500
3	3	40000	2000	1000
4	4	75000	6000	3000
5	5	50000	5000	2000

4.Views:

A view that combines Employees ,Departments, and Salaries to provide a detailed report of employee salaries with department name and net salaries

SQLQuery18.sql -...-D1O9O7EI\HP (59))* SQLQuery17.sql -...-D1O9O7EI\HP (68))*

```sql
create view SalaryReport as
select
    e.EmployeeID,
    e.Name AS EmployeeName,
    d.DepartmentName,
    s.BaseSalary,
    s.Bonus,
    s.Deductions,
    (s.BaseSalary + s.Bonus - s.Deductions) as NetSalary
from
    Employees e
join
    Departments d on e.DepartmentID = d.DepartmentID
join
    Salaries s on e.EmployeeID = s.EmployeeID;

    select * from SalaryReport
```

100 %

Results | Messages

	EmployeeID	EmployeeName	DepartmentName	BaseSalary	Bonus	Deductions	NetSalary
1	1	Manu	HR	66000	3000	2000	67000
2	2	Renu	Finance	65000	5000	2500	67500
3	3	Jasmine	Sales	40000	2000	1000	41000
4	4	Sunil	Communication	75000	6000	3000	78000
5	5	Angel	Sales	50000	5000	2000	53000

- A view that lists employees earning above a certain threshold

SQLQuery18.sql -...-D1O9O7EI\HP (59))* SQLQuery17.sql -...-D1O9O7EI\HP (68))*

```sql
create view Highearning as
select
    e.EmployeeID,
    e.Name AS EmployeeName,
    d.DepartmentName,
    s.BaseSalary,
    s.Bonus,
    s.Deductions,
    (s.BaseSalary + s.Bonus - s.Deductions) as NetSalary
from
    Employees e
join
    Departments d on e.DepartmentID = d.DepartmentID
join
    Salaries s on e.EmployeeID = s.EmployeeID;

    select * from Highearning where NetSalary >65000;
```

100 %

Results | Messages

	EmployeeID	EmployeeName	DepartmentName	BaseSalary	Bonus	Deductions	NetSalary
1	1	Manu	HR	66000	3000	2000	67000
2	2	Renu	Finance	65000	5000	2500	67500
3	4	Sunil	Communication	75000	6000	3000	78000

Programming Task: Employee Payroll System (C# Console Application)

Employee_Payroll_System | Employee_Payroll_System.BaseEmployee | CalculateSalary(int dedu

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Employee_Payroll_System
{
    5 references
    class BaseEmployee
    {
        1 reference
        public int ID{ get; set; }
        1 reference
        public string Name { get; set; }
        1 reference
        public string Role { get; set; }
        2 references
        public int Basic_pay { get; set; }
        2 references
        public int Allowances { get; set; }

        6 references
        public virtual int CalculateSalary(int deductions)
        {
            return Basic_pay + Allowances - deductions;

        }
        0 references
        public override string ToString()
        {
            return $"ID: {ID}, Name: {Name}, Role: {Role}, Basic Pay: {Basic_pay}, Allowances: {Allowances}";
        }

    }
    0 references
    class Manager : BaseEmployee
    {
        4 references
        public override int CalculateSalary(int deductions)
        {
            return base.CalculateSalary(deductions);
        }
```

```
        }

    }
    0 references
    class Manager : BaseEmployee
    {
        4 references
        public override int CalculateSalary(int deductions)
        {
            return base.CalculateSalary(deductions);
        }
    }
    0 references
    class Developer : BaseEmployee
    {
        4 references
        public override int CalculateSalary(int deductions)
        {
            return base.CalculateSalary(deductions);
        }

    }
    0 references
    class Intern : BaseEmployee
    {
        4 references
        public override int CalculateSalary(int deductions)
        {
            return base.CalculateSalary(deductions);
        }
    }
```

```
}
0 references
class Emp
{
    public static List<BaseEmployee> employees = new List<BaseEmployee>();

    0 references
    public static void Main(string[] args)
    {
        int i = 3;
        int choice;
        choice = int.Parse(Console.ReadLine());

        while (i > 0)
        {
            Console.WriteLine("1.Add new Employee");
            Console.WriteLine("2.Display all employees ");
            Console.WriteLine("3.Calculate and display individual salaries");

            Console.WriteLine("Enter the choice");

            switch (choice)
            {
                case 1:
                    AddNewEmployee();
                    break;

                case 2:
                    DisplayAllEmployees();
                    break;

                case 3:
                    CalculateSalaries();
                    break;

                default:
                    break;

            }

        }
```

```
            }

        }
    }
    1 reference
    public static void AddNewEmployee()
    {
        Console.WriteLine("Enter the employee Name:");
        String Name = Console.ReadLine();


        Console.WriteLine("Enter the employee role:");
        String Role = Console.ReadLine();

        Console.WriteLine("Enter the employee Base_pay:");
        int Base_pay = int.Parse(Console.ReadLine());

        Console.WriteLine("Enter the employee Allowances:");
        int Allowances = int.Parse(Console.ReadLine());
    }
    1 reference
    public static void DisplayAllEmployees()
    {
    }

    1 reference
    public static void CalculateSalaries()
    {
    }

}
```