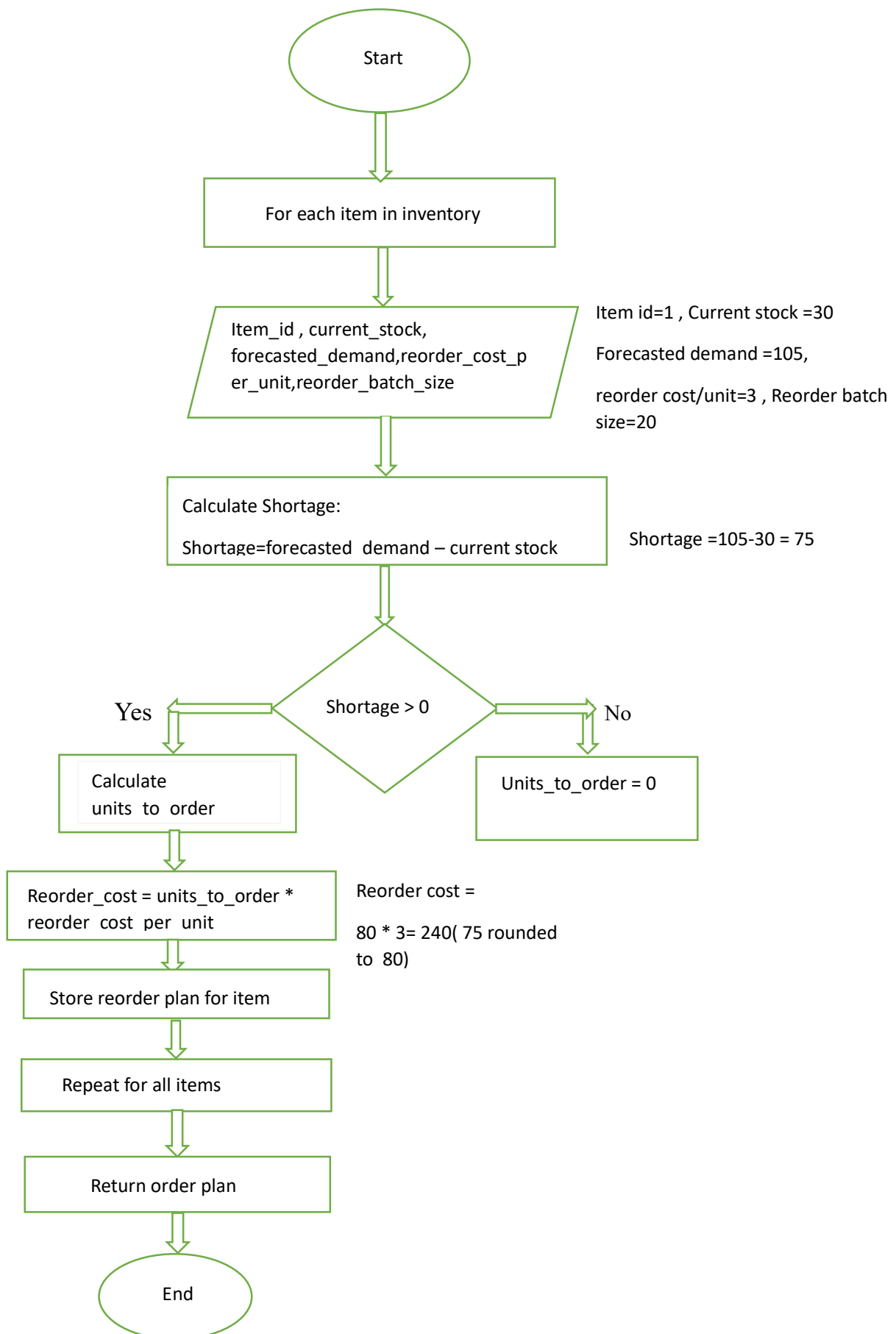


Task 1: Algorithm Development Task: Inventory Recording System



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 -...-D10907EI\HP (51))			LAPTOP-D10907EI\S...ns - dbo.Salaries	
	Column Name	Data Type	Allow Nulls	
▼	EmployeeID	int	<input type="checkbox"/>	
	Name	nvarchar(50)	<input type="checkbox"/>	
	DepartmentID	int	<input type="checkbox"/>	
	HireDate	date	<input type="checkbox"/>	
			<input type="checkbox"/>	

SQLQuery13.sql -...-D10907EI\HP (52))*

SQLQuery12.sql -...-D10907EI\HP (51))*

```
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 -...-D10907EI\HP (52))*			SQLQuery12.sql -...-D10907EI\HP (51))		
	Column Name	Data Type		Allow Nulls	
	DepartmentID	int		<input type="checkbox"/>	
	DepartmentName	nvarchar(50)		<input type="checkbox"/>	
				<input type="checkbox"/>	

SQLQuery13.sql -...-D10907EI\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

Column Name	Data Type	Allow Nulls
EmployeeID	int	<input type="checkbox"/>
BaseSalary	int	<input type="checkbox"/>
Bonus	int	<input type="checkbox"/>
Deductions	int	<input type="checkbox"/>

SQLQuery13.sql -...-D10907EI\HP (52))* SQLQuery12.sql -...-D10907EI\HP (51))* LAPTO

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

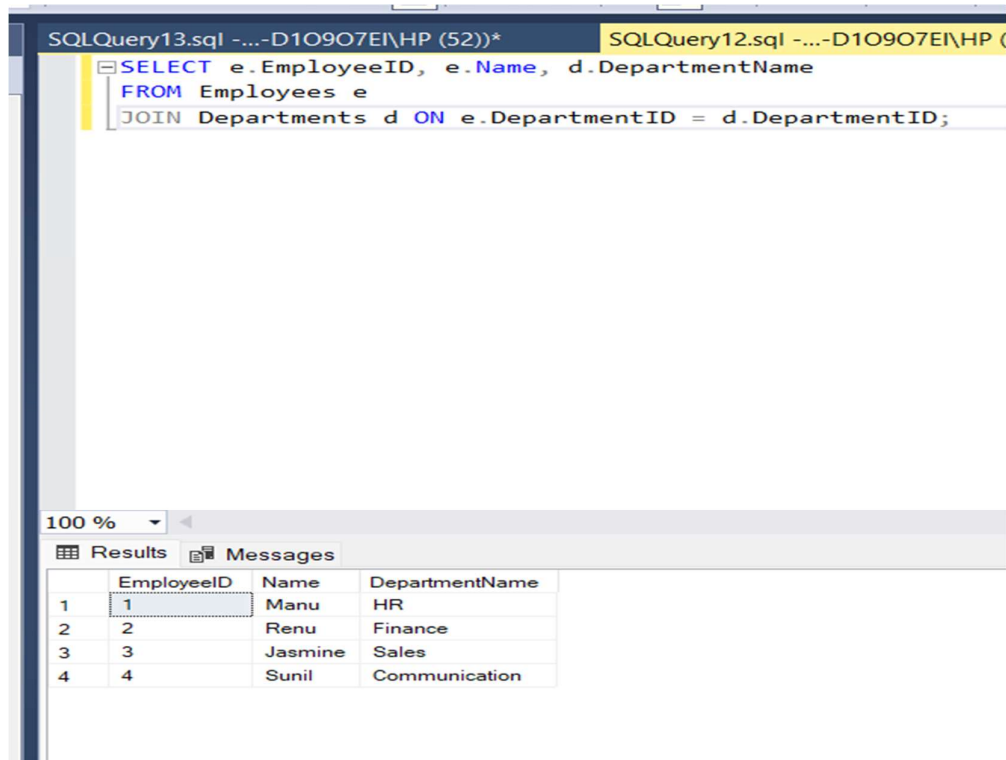
100 %

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



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery13.sql' and 'SQLQuery12.sql'. The 'SQLQuery12.sql' tab is active, displaying the following SQL query:

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

Below the query editor, the 'Results' tab is selected, showing a table with 4 rows and 4 columns: EmployeeID, Name, and DepartmentName. The data is as follows:

	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:
 $\text{Net Salary} = \text{Base salary} + \text{Bonus} - \text{Deductions}$

```
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;
```



The screenshot shows the 'Results' tab of a SQL query, displaying a table with 7 columns: EmployeeID, Name, BaseSalary, Bonus, Deductions, and NetSalary. The data is as follows:

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

- Identify the department with the highest average salary.

```
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 -...-D10907EI\HP (66))*    SQLQuery15.sql -...-D10907EI\HP (67))*    SQLQuery16.sql -...-D10907EI\HP (66))*
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 -...-D10907EI\HP (66))*    SQLQuery16.sql -...-D10907EI\HP (66))*
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 -...-D10907EI\HP (68))* X SQLQuery16.sql -...-D10907EI\HP (66))* SQLQuery15.sql -...-D10907EI\HP (67))* SQLQuer

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 -...-D10907EI\HP (59))* X SQLQuery

```
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

The screenshot displays the SQL Server Enterprise Manager interface. At the top, two tabs are visible: 'SQLQuery18.sql -...-D1O9O7EI\HP (59))*' and 'SQLQuery17.sql -...-D1O9O7EI\HP (68))*'. The active tab shows the following SQL code:

```
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
```

Below the code editor, the 'Results' tab is selected, showing a table with 8 columns: EmployeeID, EmployeeName, DepartmentName, BaseSalary, Bonus, Deductions, and NetSalary. The table contains 5 rows of data:

	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 -...-D10907EI\HP (59))* SQLQuery17.sql -...-D10907EI\HP (68))*

```

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