McKesson RxO Programming Test

**Question 1**

Using C#, write a code snippet for the following:

class User

{

public int UserID {get;set;}

public string Username {get;set;}

public string FirstName {get;set;}

public string LastName {get;set;}

}

List<User> AllUsers = GetUsers();

private List<User> GetUsers()

{

//Returns a list of users from database

}

/\*

TODO: Create a hierarchical list of users by the first letter of their first name

E.g.

List<User> contains

John Adams, Catherine Johnson, Jessica Smith, Allan Davies, Carla Lewis, Ann Clark, Mary Webb

Output should be:

A

Allan Davies

Ann Clark

C

Catherine Johnson

Carla Lewis

J

John Adams

Jessica Smith

M

Mary Webb

\*/

**Question 2**

In the code snippet below, the values in "StringIDs" are (potentially empty) comma-delimited lists of ID Numbers.

For example, StringIDs could contain the following strings:

* "3, 4, 5, 82, 19"
* ""
* "18,4,103,3"
* "7"
* "1, 82"

Write a method using C# which returns the distinct ID numbers from this input as a List of integers. Assume that all inputs are valid (e.g. the numbers will always be whole numbers).

public List<int> GetIDsAsIntegers(List<string> StringIDs)

{

}

**Question 3**

We have a table that stores the drug costs based on an effective date. Effective date is the date when the price for that drug takes into effect until the date when it is changed.

|  |  |  |
| --- | --- | --- |
| DrugID | EffectiveDate | Cost |
| 101 | 1/1/2015 | 135 |
| 101 | 3/15/2015 | 167 |
| 101 | 4/17/2015 | 185 |
| 101 | 9/8/2015 | 212 |
| 102 | 2/3/2015 | 1245 |
| 102 | 7/7/2015 | 1550 |

Write a SQL Server function that accepts a DrugID and a Date and returns the Cost. The cost of the drug at a given date is defined as the cost of the drug from the table where the date is less than or equal to the maximum effective date for that drug.

E.g. In the above data, the cost of DrugID 101 when the date is 2/5/2015 is 135 and on 6/20/2015 is 185.

USE [McKessonDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SelectDrugCostByEffectiveDate] Script Date: 9/5/2022 7:32:13 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[SelectDrugCostByEffectiveDate]

@DrugID INT,

@InputDate DATETIME

AS

-- All Drug Cost Records

SELECT \*

FROM [dbo].[DrugCost]

-- Effective Drug Cost

SELECT TOP 1 d.DrugID,

d.EffectiveDate,

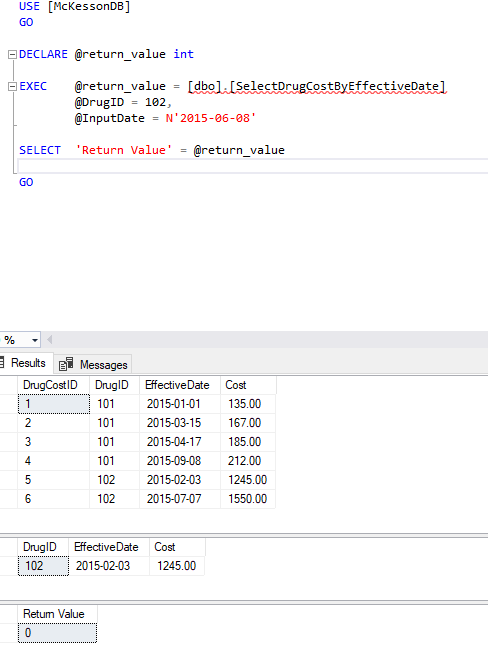
d.Cost

FROM [dbo].[DrugCost] d

WHERE @DrugID = d.DrugID AND

@InputDate >= D.EffectiveDate

ORDER BY d.EffectiveDate DESC

****

**Question 4**

Consider the following two tables:

**Employee**

|  |  |
| --- | --- |
| EmployeeId | EmployeeName |
| 1001 | Allan Davies |
| 1002 | John Adams |
| 1003 | Catherine Johnson |
| 1004 | Jessica Smith |
| 1005 | Carla Lewis |

**PartTimeEmployee**

|  |
| --- |
| EmployeeId |
| 1001 |
| 1004 |
| 1005 |

Write a SQL Query to return all employees who are not part time employees using:

1. LEFT JOIN
2. NOT EXISTS
3. NOT IN
4. USE [McKessonDB]
5. GO
6. /\*\*\*\*\*\* Object: StoredProcedure [dbo].[SelectFullTimeEmployees] Script Date: 9/5/2022 7:20:33 PM \*\*\*\*\*\*/
7. SET ANSI\_NULLS ON
8. GO
9. SET QUOTED\_IDENTIFIER ON
10. GO
11. ALTER PROCEDURE [dbo].[SelectFullTimeEmployees]
12. AS
13. -- Gets all records form Employee
14. SELECT TOP (1000) [EmployeeId]
15. ,[EmployeeName]
16. FROM [McKessonDB].[dbo].[Employee]
17. -- NOT EXISTS
18. SELECT e.EmployeeId,
19. e.EmployeeName
20. FROM [dbo].[Employee] e
21. LEFT OUTER JOIN [dbo].[PartTimeEmployee] p ON
22. E.EmployeeId = P.EmployeeId
23. WHERE NOT EXISTS (SELECT EmployeeId e2
24. FROM [dbo].[PartTimeEmployee] e2
25. WHERE e.EmployeeId = e2.EmployeeId)
26. -- NOT IN
27. SELECT e.EmployeeId,
28. e.EmployeeName
29. FROM [dbo].[Employee] e
30. LEFT OUTER JOIN [dbo].[PartTimeEmployee] p ON
31. E.EmployeeId = P.EmployeeId
32. WHERE E.EmployeeId NOT IN (SELECT EmployeeId e2
33. FROM [dbo].[PartTimeEmployee] e2
34. WHERE e.EmployeeId = e2.EmployeeId)

Application

Description automatically generated with low confidence

Explain which option is preferred and why.

**Question 5**

Consider the following Revenue table which stores the revenues generated for a given facility by month and year

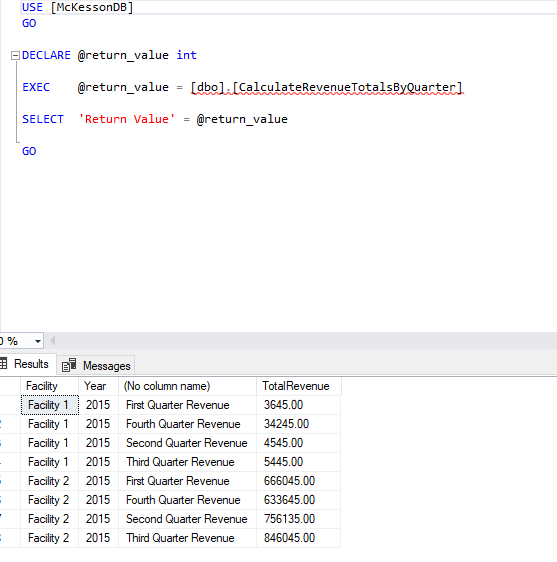
|  |  |  |  |
| --- | --- | --- | --- |
| Facility | Month | Year | Revenue |
| Facility 1 | 1 | 2015 | $ |
| Facility 1 | 2 | 2015 | $ |
| Facility 1 | 3 | 2015 | $ |
| Facility 1 | 4 | 2015 | $ |
| Facility 1 | 5 | 2015 | $ |
| Facility 1 | 6 | 2015 | $ |
| Facility 1 | 7 | 2015 | $ |
| Facility 1 | 8 | 2015 | $ |
| Facility 1 | 9 | 2015 | $ |
| Facility 1 | 10 | 2015 | $ |
| Facility 1 | 11 | 2015 | $ |
| Facility 1 | 12 | 2015 | $ |
| Facility 2 | 1 | 2015 | $ |
| Facility 2 | 2 | 2015 | $ |
| Facility 2 | 3 | 2015 | $ |
| Facility 2 | 4 | 2015 | $ |
| Facility 2 | 5 | 2015 | $ |
| Facility 2 | 6 | 2015 | $ |
| Facility 2 | 7 | 2015 | $ |
| Facility 2 | 8 | 2015 | $ |
| Facility 2 | 9 | 2015 | $ |
| Facility 2 | 10 | 2015 | $ |
| Facility 2 | 11 | 2015 | $ |
| Facility 2 | 12 | 2015 | $ |

1. Write a query to return the revenues by Facility and Year.
2. USE [McKessonDB]
3. GO
4. /\*\*\*\*\*\* Object: StoredProcedure [dbo].[CalculateRevenueTotals] Script Date: 9/5/2022 7:52:48 PM \*\*\*\*\*\*/
5. SET ANSI\_NULLS ON
6. GO
7. SET QUOTED\_IDENTIFIER ON
8. GO
9. ALTER PROCEDURE [dbo].[CalculateRevenueTotals]
10. AS
11. SELECT r.Facility,
12. r.[Year],
13. SUM(r.Revenue) AS TotalRevenue
14. FROM [dbo].[Revenue] r
15. GROUP BY r.Facility,
16. r.[Year]

Graphical user interface, text, application

Description automatically generated

1. Write a query to return the revenues by Facility, Quarter and Year.
2. USE [McKessonDB]
3. GO
4. /\*\*\*\*\*\* Object: StoredProcedure [dbo].[CalculateRevenueTotalsByQuarter] Script Date: 9/5/2022 9:11:27 PM \*\*\*\*\*\*/
5. SET ANSI\_NULLS ON
6. GO
7. SET QUOTED\_IDENTIFIER ON
8. GO
9. ALTER PROCEDURE [dbo].[CalculateRevenueTotalsByQuarter]
10. AS
11. SELECT r.Facility,
12. r.[Year],
13. '1st Quarter Revenue' AS Period,
14. SUM(r.Revenue) AS TotalRevenue
15. FROM [dbo].[Revenue] r
16. WHERE r.Month BETWEEN 1 AND 3
17. GROUP BY r.Facility,
18. r.[Year]
19. UNION
20. SELECT r.Facility,
21. r.[Year],
22. '2nd Quarter Revenue' AS Period,
23. SUM(r.Revenue) AS TotalRevenue
24. FROM [dbo].[Revenue] r
25. WHERE r.Month BETWEEN 4 AND 6
26. GROUP BY r.Facility,
27. r.[Year]
28. UNION
29. SELECT r.Facility,
30. r.[Year],
31. '3rd Quarter Revenue' AS Period,
32. SUM(r.Revenue) AS TotalRevenue
33. FROM [dbo].[Revenue] r
34. WHERE r.Month BETWEEN 7 AND 9
35. GROUP BY r.Facility,
36. r.[Year]
37. UNION
38. SELECT r.Facility,
39. r.[Year],
40. '4th Quarter Revenue' AS Period,
41. SUM(r.Revenue) AS TotalRevenue
42. FROM [dbo].[Revenue] r
43. WHERE r.Month BETWEEN 10 AND 12
44. GROUP BY r.Facility,
45. r.[Year]



1. Write a query to return the facilities whose total revenues for a year are more than 1 million dollars.

USE [McKessonDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CalculateRevenueOverOneMillion] Script Date: 9/5/2022 9:34:21 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[CalculateRevenueOverOneMillion]

AS

SELECT r.Facility,

r.[Year],

SUM(r.Revenue) AS TotalRevenue

FROM [dbo].[Revenue] r

GROUP BY r.Facility,

r.[Year]

HAVING SUM(r.Revenue) >= 1000000

Graphical user interface, text, application

Description automatically generated