Below is the implementation of the UpdateRebuiltPartsAsync function with exception handling and transaction support. Additionally, I've included a CreateRebuiltPartsAsync function for creating new records.

**Update Function**

csharp

Copy

[AllowAnonymous]

private async Task UpdateRebuiltPartsAsync(RebuiltPartsViewModel rebuiltPartsViewModel)

{

using (var transaction = await \_dbConnection.BeginTransactionAsync())

{

try

{

// Update the main RbMasterlist table

var updateRbMasterlistQuery = @"

UPDATE SBCES.RbMasterlist

SET

MmsStockCode = @MmsStockCode,

DetailedDesc = @DetailedDesc,

Keyword = @Keyword,

JobNumber = @JobNumber,

CoreCharge = @CoreCharge,

CorePartNum = @CorePartNum,

BuyNewCost = @BuyNewCost,

RemanCost = @RemanCost,

LastModifiedBy = @LastModifiedBy,

LastModifiedDate = @LastModifiedDate,

IsActive = @IsActive

WHERE RbMasterlistId = @RbMasterlistId;

";

await \_dbConnection.ExecuteAsync(updateRbMasterlistQuery, new

{

rebuiltPartsViewModel.MmsStockCode,

rebuiltPartsViewModel.DetailedDesc,

rebuiltPartsViewModel.Keyword,

rebuiltPartsViewModel.JobNumber,

rebuiltPartsViewModel.CoreCharge,

rebuiltPartsViewModel.CorePartNum,

rebuiltPartsViewModel.BuyNewCost,

rebuiltPartsViewModel.RemanCost,

rebuiltPartsViewModel.LastModifiedBy,

rebuiltPartsViewModel.LastModifiedDate,

rebuiltPartsViewModel.IsActive,

rebuiltPartsViewModel.RbMasterlistId

}, transaction);

// Delete existing bus associations for the rebuilt part

var deleteBusesQuery = @"

DELETE FROM SBCES.RBLISTOFBUSES

WHERE RebuiltStockNum = @RebuiltStockNum;

";

await \_dbConnection.ExecuteAsync(deleteBusesQuery, new

{

rebuiltPartsViewModel.RebuiltStockNum

}, transaction);

// Insert new bus associations

var insertBusesQuery = @"

INSERT INTO SBCES.RBLISTOFBUSES (RebuiltStockNum, listid)

VALUES (@RebuiltStockNum, @BusId);

";

foreach (var bus in rebuiltPartsViewModel.Buses)

{

await \_dbConnection.ExecuteAsync(insertBusesQuery, new

{

rebuiltPartsViewModel.RebuiltStockNum,

BusId = bus.BusId

}, transaction);

}

// Commit the transaction

await transaction.CommitAsync();

}

catch (Exception ex)

{

// Rollback the transaction in case of an error

await transaction.RollbackAsync();

Console.WriteLine($"Error updating rebuilt parts data: {ex.Message}");

throw; // Re-throw the exception to propagate it

}

}

}

**Create Function**

csharp

Copy

[AllowAnonymous]

private async Task<int> CreateRebuiltPartsAsync(RebuiltPartsViewModel rebuiltPartsViewModel)

{

using (var transaction = await \_dbConnection.BeginTransactionAsync())

{

try

{

// Insert into the main RbMasterlist table

var insertRbMasterlistQuery = @"

INSERT INTO SBCES.RbMasterlist (

RebuiltStockNum, MmsStockCode, DetailedDesc, Keyword, JobNumber,

CoreCharge, CorePartNum, BuyNewCost, RemanCost, LastModifiedBy,

LastModifiedDate, IsActive

)

VALUES (

@RebuiltStockNum, @MmsStockCode, @DetailedDesc, @Keyword, @JobNumber,

@CoreCharge, @CorePartNum, @BuyNewCost, @RemanCost, @LastModifiedBy,

@LastModifiedDate, @IsActive

);

SELECT CAST(SCOPE\_IDENTITY() AS INT);

";

var rbMasterlistId = await \_dbConnection.QuerySingleAsync<int>(insertRbMasterlistQuery, new

{

rebuiltPartsViewModel.RebuiltStockNum,

rebuiltPartsViewModel.MmsStockCode,

rebuiltPartsViewModel.DetailedDesc,

rebuiltPartsViewModel.Keyword,

rebuiltPartsViewModel.JobNumber,

rebuiltPartsViewModel.CoreCharge,

rebuiltPartsViewModel.CorePartNum,

rebuiltPartsViewModel.BuyNewCost,

rebuiltPartsViewModel.RemanCost,

rebuiltPartsViewModel.LastModifiedBy,

rebuiltPartsViewModel.LastModifiedDate,

rebuiltPartsViewModel.IsActive

}, transaction);

// Insert bus associations

var insertBusesQuery = @"

INSERT INTO SBCES.RBLISTOFBUSES (RebuiltStockNum, listid)

VALUES (@RebuiltStockNum, @BusId);

";

foreach (var bus in rebuiltPartsViewModel.Buses)

{

await \_dbConnection.ExecuteAsync(insertBusesQuery, new

{

rebuiltPartsViewModel.RebuiltStockNum,

BusId = bus.BusId

}, transaction);

}

// Commit the transaction

await transaction.CommitAsync();

return rbMasterlistId;

}

catch (Exception ex)

{

// Rollback the transaction in case of an error

await transaction.RollbackAsync();

Console.WriteLine($"Error creating rebuilt parts data: {ex.Message}");

throw; // Re-throw the exception to propagate it

}

}

}