Here are the converted MS SQL queries and updated controller functions based on your model:

**1. Converted INSERT Statement:**

sql

Copy

INSERT INTO RbMasterlist (

LastModifiedBy, DateModified, JobNumber, CoreCharge, RebuiltStockNum,

CorePartNum, Keyword, DetailedDesc, MmsStockCode, EstimatedCost,

SopNumber, BuyNewCost, RemanCost, ExternalCost, Active

) VALUES (

UPPER(ISNULL(@LastModifiedBy, ' ')),

GETDATE(),

UPPER(ISNULL(@JobNumber, ' ')),

ISNULL(@CoreCharge, ' '),

ISNULL(@RebuiltStockNum, ' '),

ISNULL(@CorePartNum, ' '),

UPPER(ISNULL(@Keyword, ' ')),

UPPER(ISNULL(@DetailedDesc, ' ')),

ISNULL(@MmsStockCode, ' '),

0,

ISNULL(@SopNumber, ' '),

ISNULL(@BuyNewCost, ''),

ISNULL(@RemanCost, ''),

ISNULL(@ExternalCost, ''),

@Active

)

**2. Converted SELECT Statement:**

sql

Copy

SELECT

UPPER(LinkCode) AS LinkCode,

UPPER(LinkType) AS LinkType

FROM ScPartsUsed

WHERE

RebuiltPart = 'Y'

AND MmsRebuiltCode = @MmsStockCode

**Updated Controller Functions:**

csharp

Copy

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> UpdateRebuiltPartsView(

[DataSourceRequest] DataSourceRequest request,

[FromForm] RebuiltPartsViewModel model)

{

try

{

// Update the RebuiltPart

var updatePartSql = @"

UPDATE SBCES.RbMasterlist

SET

LastModifiedBy = UPPER(ISNULL(@LastModifiedBy, ' ')),

DateModified = GETDATE(),

JobNumber = UPPER(ISNULL(@JobNumber, ' ')),

CoreCharge = ISNULL(@CoreCharge, ' '),

CorePartNum = ISNULL(@CorePartNum, ' '),

Keyword = UPPER(ISNULL(@Keyword, ' ')),

DetailedDesc = UPPER(ISNULL(@DetailedDesc, ' ')),

MmsStockCode = ISNULL(@MmsStockCode, ' '),

SopNumber = ISNULL(@SopNumber, ' '),

BuyNewCost = ISNULL(@BuyNewCost, ''),

RemanCost = ISNULL(@RemanCost, ''),

ExternalCost = ISNULL(@ExternalCost, ''),

Active = @IsActive

WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(updatePartSql, new {

model.LastModifiedBy,

model.JobNumber,

model.CoreCharge,

model.CorePartNum,

model.Keyword,

model.DetailedDesc,

model.MmsStockCode,

model.SopNumber,

model.BuyNewCost,

model.RemanCost,

ExternalCost = "", // Add to model if needed

IsActive = model.IsActive ? 1 : 0,

model.RebuiltStockNum

});

// Update buses

var deleteOldBusesSql = "DELETE FROM SBCES.RbListOfBuses WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(deleteOldBusesSql, new { model.RebuiltStockNum });

if (model.ListOfBusIds?.Any() == true)

{

var insertNewBusesSql = @"

INSERT INTO SBCES.RbListOfBuses (RebuiltStockNum, ListId)

VALUES (@RebuiltStockNum, @ListId)";

foreach (var listId in model.ListOfBusIds.Where(id => id.HasValue))

{

await \_dbConnection.ExecuteAsync(insertNewBusesSql, new {

model.RebuiltStockNum,

ListId = listId.Value

});

}

}

return Json(new { success = true });

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> AddRebuiltPartsView(

[DataSourceRequest] DataSourceRequest request,

RebuiltPartsViewModel model)

{

try

{

var insertSql = @"

INSERT INTO RbMasterlist (...)

VALUES (...);

SELECT SCOPE\_IDENTITY();";

// Execute insert and get ID

var id = await \_dbConnection.ExecuteScalarAsync<int>(insertSql, new {

// Map parameters from model

});

return Ok(new { Id = id });

}

catch (Exception ex)

{

return StatusCode(500, "An error occurred while adding Rebuilt Part.");

}

}

**Important Notes:**

1. **Model Updates Needed:**
   * Add ExternalCost property to your model
   * Consider adding EstimatedCost if needed
   * Convert boolean IsActive to bit using model.IsActive ? 1 : 0
2. **SQL Server Specifics:**
   * SYSDATE → GETDATE()
   * NVL() → ISNULL()
   * Column names use PascalCase without underscores
   * Use SCOPE\_IDENTITY() to get inserted ID