Задание: Раписать алгоритм формирования значения в поле **IngoingTnsEspk**

Структура **MaterialRouteStep** для справки ниже.

/// <summary>

/// Формирование отчета "Анализ посада"

/// </summary>

/// <param name="millUnitCode">Код стана</param>

/// <param name="beginDate">Дата начала</param>

/// <param name="endDate">Дата окончания</param>

/// <param name="brigade">Смена</param>

/// <param name="diameter">Диаметр</param>

/// <param name="steelGrade">Марка стали</param>

public List<FurnaceChargingSteelGradeDTO> GetFurnaceChargingAnalysis(Int64? millUnitCode, DateTime beginDate, DateTime endDate,

String brigade, Decimal? diameter, String steelGrade)

{

var report = new List<FurnaceChargingSteelGradeDTO>();

// выбор посада

var ingoingMaterials = DAOContainer.MaterialRouteStepDao.Linq()

.Where(x => x.Shift.ShiftDate >= beginDate &&

x.Shift.ShiftDate <= endDate.Date.AddDays(1).AddMilliseconds(-1) &&

(String.IsNullOrEmpty(brigade) || x.Shift.Brigade == brigade) &&

(x.TechOperation != null &&

x.TechOperation.Code == OperationConst.FurnaceCharging))

.ToList()

.Where(x => (millUnitCode == null || x.Unit.AllParents.Any(y => y.Code == millUnitCode)) &&

x.OrderFactRule != null &&

x.OrderFactRule.OperationFactType == OperationFactType.Ingoing &&

!x.IsCanceled && (x.Operation == null || x.Operation.Code != OperationConst.Cancel) &&

(diameter == null || x.Order.GetNumberDescription(DescriptionTypeConst.Diameter) == diameter) &&

(steelGrade == "0" || x.Material.GetRefDescriptionID(DescriptionTypeConst.SteelGrade) == long.Parse(steelGrade))).ToList();

var ingoingMaterialIDs = ingoingMaterials.ConvertAll(x => x.Material.ID);

// выбор брака (горелый открой)

var rejected = DAOContainer.MaterialRouteStepDao.Linq()

.Where(x => x.Shift.ShiftDate >= beginDate &&

x.Shift.ShiftDate <= endDate.Date.AddDays(1).AddMilliseconds(-1) &&

(String.IsNullOrEmpty(brigade) || x.Shift.Brigade == brigade) &&

x.TechOperation != null &&

x.TechOperation.Code == OperationConst.HotScissorsCutting)

.Where(x => x.Unit != null && (x.Unit.Code == UnitConst.HotScissorsDefectCradle || x.Unit.Code == UnitConst.HotScissorsFinalDefectCradle))

.ToList()

.Where(x => (millUnitCode == null || x.Unit.AllParents.Any(y => y.Code == millUnitCode)) &&

x.OrderFactRule != null &&

x.OrderFactRule.OperationFactType == OperationFactType.Rejected &&

!x.IsCanceled && (x.Operation == null || x.Operation.Code != OperationConst.Cancel) &&

(diameter == null || x.Order.GetNumberDescription(DescriptionTypeConst.Diameter) == diameter) &&

(steelGrade == "0" || x.Material.GetRefDescriptionID(DescriptionTypeConst.SteelGrade) == long.Parse(steelGrade)))

.ToList()

.GroupBy(x => new { x.Material.ID }).ToList()

.ConvertAll(x => new

{

x.OrderBy(y => y.Material.ID).First().Material,

Sum = x.Sum(y => y.ResultQuantity),

ParentMaterialsID = DAOContainer.MaterialDao.GetAllParentsID(x.OrderBy(y => y.Material.ID).First().Material.ID)

.Where(ingoingMaterialIDs.Contains)

.OrderBy(y => y).ToList()

});

var ingoing = ingoingMaterials

.ConvertAll(p => new

{

QuantityTns = p.Material.Quantity.Tns,

BurnDefectTns = rejected

.FindAll(x => x.ParentMaterialsID != null &&

x.ParentMaterialsID.Contains(p.Material.ID) &&

x.ParentMaterialsID.IndexOf(

x.ParentMaterialsID.Find(y => y == p.Material.ID)) == 0)

.Sum(x => x.Sum).Tns,

Mill = p.Unit.GetParentUnitByCategory(UnitCategoryConst.Mill),

RollingOrder = p.Order,

SteelGradeID = p.Material.GetRefDescriptionID(DescriptionTypeConst.SteelGrade),

SteelGradeName = p.Material.GetRefDescriptionName(DescriptionTypeConst.SteelGrade),

Diameter = p.Order.GetNumberDescription(DescriptionTypeConst.Diameter),

SupplierCode = p.Material.GetRefDescriptionCode(DescriptionTypeConst.Supplier)

})

.GroupBy(y => new

{

y.Mill,

y.RollingOrder,

y.SteelGradeID,

y.SteelGradeName,

y.Diameter,

y.SupplierCode

})

.Select(y => new

{

y.Key.Mill,

y.Key.SteelGradeID,

y.Key.SteelGradeName,

y.Key.Diameter,

ResultQuantityEspk = (y.Key.SupplierCode == DescriptionValueConst.espk ? y.Sum(t => t.QuantityTns) : 0),

ResultQuantity = (y.Key.SupplierCode != DescriptionValueConst.espk ? y.Sum(t => t.QuantityTns) : 0),

BurnDefectEspk = (y.Key.SupplierCode == DescriptionValueConst.espk ? y.Sum(t => t.BurnDefectTns) : 0),

BurnDefect = (y.Key.SupplierCode != DescriptionValueConst.espk ? y.Sum(t => t.BurnDefectTns) : 0),

})

.GroupBy(y => new

{

y.Mill,

y.SteelGradeID,

y.SteelGradeName,

y.Diameter

}).ToList()

.ConvertAll(y => new

{

MillCode = (y.Key.Mill != null ? y.Key.Mill.Code : (Int64?)null),

MillName = (y.Key.Mill != null ? y.Key.Mill.ShortName : ""),

y.Key.SteelGradeID,

y.Key.SteelGradeName,

y.Key.Diameter,

IngoingTnsEspk = y.Sum(t => t.ResultQuantityEspk) - y.Sum(t => t.BurnDefectEspk),

IngoingTns = y.Sum(t => t.ResultQuantity) - y.Sum(t => t.BurnDefect)

})

.OrderBy(x => x.Diameter).ThenBy(x => x.SteelGradeName).ToList();

foreach (var item in ingoing)

{

var record = new FurnaceChargingSteelGradeDTO();

record.MillCode = item.MillCode;

record.MillName = item.MillName;

record.SteelGradeID = item.SteelGradeID;

record.SteelGradeName = item.SteelGradeName;

record.Diameter = item.Diameter;

record.IngoingTnsEspk = item.IngoingTnsEspk;

record.IngoingTns = item.IngoingTns;

report.Add(record);

}

return report;

}

