Events task

1. Task: Validate and Restrict Discount on Sales Order

Objective:

Write a **Codeunit** that ensures the "Line Discount %" on a Sales Line cannot exceed **10%**. If it does, reset it to 10% and show a warning message.

Solutions ::

```
// Write a Codeunit that ensures the "Line Discount %" on a Sales Line cannot exceed 10%. If it does, reset i
[EventSubscriber(ObjectType::Table, Database::"Sales Line", OnAfterValidateEvent, "Line Discount %", true, tr
0 references
procedure RestrictLinepercentDiscount(var Rec: Record "Sales Line"; var xRec: Record "Sales Line"; CurrFieldN
var
    Discount: Decimal;

begin
    // Discount:=
    if Rec."Line Discount %" > 10 then begin
        Error('Line Discount can not be Grater than 10');
    end;
    ;
end;
```

2. Task: Auto-Calculate Total Weight on Sales Order

Objective:

Create a **Codeunit** that calculates the **Total Weight** of all Sales Lines and updates it on the Sales Header. The "Weight" is a custom field in the **Item** table.

Solutions :

```
field(50107; TotalWeight; Integer)
{
    // DataClassification = ToBeClassified;
    FieldClass = FlowField;
    CalcFormula = Sum("Sales Line".Weight where("Document No." = field("No.")));
}
```

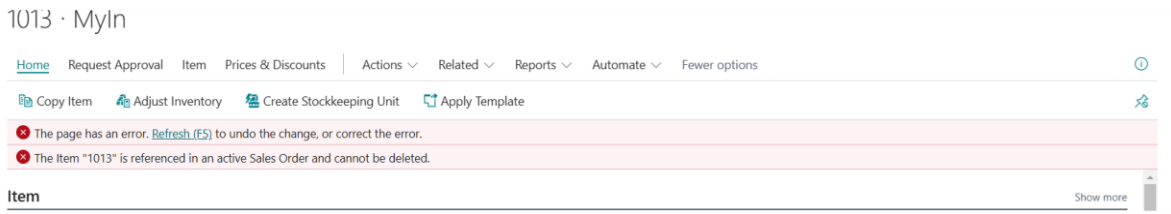
Tsk 3:

Write a Codeunit that prevents the deletion of an Item if it is referenced in an active Sales Order. Key

Requirements: Use an Event Subscriber on the **OnBeforeDeleteEvent** trigger of the Item table. Check if the Item exists in any unposted Sales Lines. Prevent deletion and display a message if the condition is met.

```
//    Codeunit to Prevent Item Deletion
[EventSubscriber(ObjectType::Table, Database::Item, OnBeforeDeleteEvent, '', true, true)]
0 references
procedure GetRecordfromItem(var Rec: Record Item; RunTrigger: Boolean)
var
    SalesLine: Record "Sales Line";
begin
    SalesLine.Reset();
    SalesLine.SetRange("Document Type", SalesLine."Document Type"::Order);
    SalesLine.SetRange("No.", SalesLine."No.");
    if SalesLine.FindFirst() then begin

        Error('The Item "%1" is referenced in an active Sales Order and cannot be deleted.', Rec."No.");
    end
end;
```

o/p:



Flow Data From item to Sales Line by Events Subscriber

```
59
60      [EventSubscriber(ObjectType::Table, Database::"Sales Line", OnAfterValidateEvent, "No.", true, true)]
        0 references
61      local procedure ItemtoSalesLine(var Rec: Record "Sales Line"; CurrFieldNo: Integer; var xRec: Record "Sales L
62      var
63          ItemRec: Record Item;
64
65      begin
66          if ItemRec.Get(Rec."No.") then begin
67              Rec."Custom Line" := ItemRec."Custom Field";
68          end;
69      end;
70
71  }
```

Flow Data From Vendor to Purchase Header :" When Select the Buy-from Vendor No.

```
// Vendor to Purchase Header While Selection Vendor No
[EventSubscriber(ObjectType::Table, Database::"Purchase Header", OnAfterValidateEvent, "Buy-from Vendor No.",
0 references
local procedure VendorToPurchaseHeader(var Rec: Record "Purchase Header"; CurrFieldNo: Integer; var xRec: Rec
var
    VendorRec: Record Vendor;

begin
    Rec.SetRange("Document Type", Rec."Document Type"::Order);
    if VendorRec.get(Rec."Buy-from Vendor No.") then begin
        Rec.CustExtDoc := VendorRec."Custom Vendor No";
    end;

end;
```

Data Flow **From Purchase Header to Gen Jorunal line then Detailed vendor Ledger entry**

```
// Purchase Header to Gen Journal Line then Detailed Vendor Ledger entry

[EventSubscriber(ObjectType::Codeunit, Codeunit::"Gen. Jnl.-Post Line", OnAfterInsertDtldVendLedgEntry, '', fa
0 references
local procedure OnAfterInsertDetailedVendorLedgEntry(DtldCVLedgEntryBuffer: Record "Detailed CV Ledg. Entry Bu
begin
    DtldVendLedgEntry."Is AEG U" := GenJournalLine."Is AEG";
    DtldVendLedgEntry.Modify();
end;
```

Data Flow **From Purchase header to Vendor Ledger** entry without Intermediate Tables

```
// Flow Data Purchase Header to Vendor Ledger entries:
[EventSubscriber(ObjectType::Codeunit, Codeunit::"Purch.-Post", OnAfterPostInvoice, '', false, false)]
0 references
local procedure PurchaseHeaderToVendorLedgernetry(TotalPurchLine: Record "Purchase Line"; TotalPurchLineLCY: Re
var

begin

    VendorLedgerEntry."Is AEG U" := PurchHeader."IS AEGU";
    VendorLedgerEntry.Modify();
end;
```

SETRANGE AND SETFILTER USE

```
1 reference
procedure Balancecount()
var
    CustomerRec: Record Customer;
    TOtalCustomer: Integer;

begin
    TOtalCustomer := 0;
    CustomerRec.Reset();
    // CustomerRec.SetRange(Balance, 10000);// filter Balance=10000
    // CustomerRec.SetFilter(Balance, '<=%1', 10000);// filter Balance<=10000
    CustomerRec.SetFilter(Balance, '>=%1', 10000);// filter Balance>=10000
 // TOtalCustomer := CustomerRec.count();//based on filter count
//  or without count By Findset
    if CustomerRec.FindSet() then begin
        repeat
            TOtalCustomer += 1;
        Until CustomerRec.Next() = 0;
        Message('total customer %1', TOtalCustomer);
    end;
end;
```

Scenario:
Create a Codeunit to process customer ledger entries. Fetch all customer ledger entries where the "Remaining Amount" is greater than 10,000, calculate the total "Remaining Amount," and display it.

Hints:

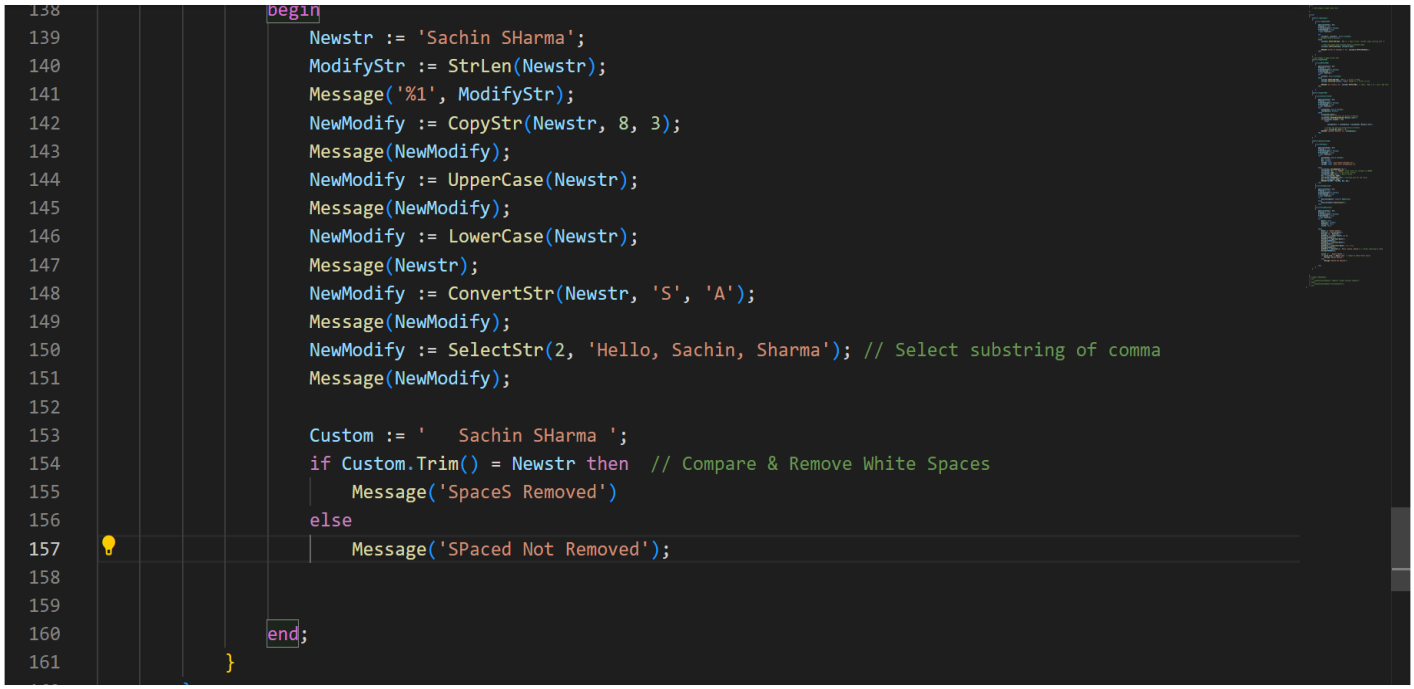
- **Use SETRANGE or SETFILTER to filter records.**
- **Use CALCSUM to calculate the total.**
- **Use CALCFIELDS for the "Remaining Amount" FlowField.**

```
// Create a Codeunit to process customer ledger entries. Fetch all customer ledger entries where the "Remaining Amount" is greater than 10,000,
   // calculate the total "Remaining Amount," and display it.
   [EventSubscriber(ObjectType::Page, Page::"Customer Ledger Entries", OnOpenPageEvent, '', true, true)]

   procedure FetchCustomerledgerentrybasedonRemainingAmt()
   var
       CustomerLedgerEntry_rec: Record "Cust. Ledger Entry";
       TotalRemainingAMt: Decimal;

   begin
       TotalRemainingAMt := 0;
       CustomerLedgerEntry_rec.Reset();
       CustomerLedgerEntry_rec.SetFilter("Remaining Amount", '>%1', 10000);
       Message('Total Customer > 10000 Remaining Amt are %1', CustomerLedgerEntry_rec.Count());
       if CustomerLedgerEntry_rec.FindSet() then begin
           repeat
               CustomerLedgerEntry_rec.CalcFields("Remaining Amount");
               TotalRemainingAMt := TotalRemainingAMt + CustomerLedgerEntry_rec."Remaining Amount";
           until CustomerLedgerEntry_rec.Next() = 0;
           Message('The Sum of Remaining Amt >10000 %1 is ', TotalRemainingAMt);

       end;

   end;
```

Create a function that fetches **all Sales Orders where the "Order Date" is within the last 30 days. Display these records in a message.**

```
// Create a function that fetches all Sales Orders where the "Order Date" is within the last 30 days.
// Display these records in a message
[EventSubscriber(ObjectType::Page, Page::"Sales Order List", OnOpenPageEvent, '', false, false)]
0 references
procedure FetchSaleOrderBasedonOrderDate()
var
    SalesOrder: Record "Sales Header";
    TotalSalesOrder: Integer;
    StartDate: Date;
    EndDate: Date;
begin
    StartDate := Today();
    EndDate := CalcDate('-30D', StartDate); // 30 Day Before Today
    Message('End Date %1', EndDate);
    SalesOrder.SetRange("Order Date", EndDate, StartDate);
    TotalSalesOrder := SalesOrder.Count();
    Message('total Sales Order Based On Order Date Before 30 Day %1', TotalSalesOrder);

end;
```

String Functions Logic Practice:ce:



```
138             begin
139                 Newstr := 'Sachin SHarma';
140                 ModifyStr := StrLen(Newstr);
141                 Message('%1', ModifyStr);
142                 NewModify := CopyStr(Newstr, 8, 3);
143                 Message(NewModify);
144                 NewModify := UpperCase(Newstr);
145                 Message(NewModify);
146                 NewModify := LowerCase(Newstr);
147                 Message(Newstr);
148                 NewModify := ConvertStr(Newstr, 'S', 'A');
149                 Message(NewModify);
150                 NewModify := SelectStr(2, 'Hello, Sachin, Sharma'); // Select substring of comma
151                 Message(NewModify);
152
153                 Custom := '   Sachin SHarma ';
154                 if Custom.Trim() = Newstr then  // Compare & Remove White Spaces
155                     Message('SpaceS Removed')
156                 else
157                     Message('SPaced Not Removed');
158
159
160             end;
161         }
```

Amount <10000 Change color of the Sale order List

```
// Add changes to page layout here
        modify(Amount)
        {
            ApplicationArea = All;
            Style = Attention;
            StyleExpr = Changecolor_gln;
        }

    }
    actions
    {
        // Add changes to page actions here
    }
    // Change color Based on AMount
    local procedure Changecolor()
    begin
        if Rec.Amount > 10000 then
            Changecolor_gln := false // Global Var & called at Field also
        else
            Changecolor_gln := true;
    end;


    trigger OnAfterGetRecord()
    begin
        Changecolor();
    end;

    var
        Changecolor_gln: Boolean;
```

Value	Description
None	None
Standard	Standard
StandardAccent	Blue
Strong	Bold
StrongAccent	Blue + Bold
Attention	Red + Italic
AttentionAccent	Blue + Italic
Favorable	Bold + Green
Unfavorable	Bold + Italic + Red
Ambiguous	Yellow
Subordinate	Grey