Technical Training:

Table of Contents

[Day 1 2](#_Toc114042178)

[1.Create Table 2](#_Toc114042179)

[2.Create List 4](#_Toc114042180)

[3.Create Card 6](#_Toc114042181)

[Day 2: 10](#_Toc114042182)

[1. Linking no. series in a particular field. 10](#_Toc114042183)

[2. Creating table relation 12](#_Toc114042184)

[3. Using SetRange and FindFirst inbuilt function: 13](#_Toc114042185)

[4. Using get function to get the data from another table. 15](#_Toc114042186)

[Day 3(Sales Billing System) 16](#_Toc114042187)

[1Create table for Billing Header table. 16](#_Toc114042188)

[2. Create List 19](#_Toc114042189)

[3. Create Card Page 21](#_Toc114042190)

[4. Create table for Listpart 23](#_Toc114042191)

[5. Create page for ListPart 25](#_Toc114042192)

[Day 4: 28](#_Toc114042193)

[1.Create Posting Tables 28](#_Toc114042194)

[2.Write codeunit for posting 33](#_Toc114042195)

[3.Create Procedure for tax calculation 35](#_Toc114042196)

[4.Create Ledger Table. 35](#_Toc114042197)

[5.Create Procedure for Inserting into Ledger 38](#_Toc114042198)

[Day 5: 39](#_Toc114042199)

[1.Item-No Duplication 39](#_Toc114042200)

[2. Inserting Data and CustomerName To Posted Billing Line Ledger 41](#_Toc114042201)

[3.Type 42](#_Toc114042202)

[4. General Journal 46](#_Toc114042203)

[5. Adding Page extension for adding “Training” Field in “General Journal” 50](#_Toc114042204)

[6. Event Subscription 50](#_Toc114042205)

[Day 6: 51](#_Toc114042206)

[1. Passing the Tax Amount to General Journal from Sales Line. 51](#_Toc114042207)

[2. Passing Tax Amount from Sales Line to Item Journal Line 52](#_Toc114042208)

[3. Passing the Line Tax Amount from Item Journal Line to Item Ledger Entry. 53](#_Toc114042209)

[4. Passing the Tax Amount from Item Journal Line to Value Entry. 54](#_Toc114042210)

[5. Set filter function: 55](#_Toc114042211)

[6. CodeUnit 56](#_Toc114042212)

[7. Event Subscription: 56](#_Toc114042213)

[Day 7 57](#_Toc114042214)

[1. Adding a Field in User Setup Table and Showing in User Card. 57](#_Toc114042215)

[2. Creating a procedure to call a function that to check if a user is allow to post bill or not. 59](#_Toc114042216)

[3. Creating a Bill Post to update the report . 60](#_Toc114042217)

# Day 1

## 1.Create Table

First we create a table and add the required fields and logic.

table 50111 LoanHeader

{

    DataClassification = ToBeClassified;

    fields

    {

        field(1; EntryNo; Integer)

        {

            DataClassification = ToBeClassified;

            AutoIncrement = true;

        }

        field(2; No; Code[50])

        {

            DataClassification = ToBeClassified;

        }

        field(3; Name; Text[50])

        {

            DataClassification = ToBeClassified;

        }

        field(4; Description; Text[50])

        {

            DataClassification = ToBeClassified;

        }

        field(5; Amount; Integer)

        {

            DataClassification = ToBeClassified;

            trigger OnValidate()

            begin

                "Current Remaining Amount" := "Amount" - "Remaining Amount";

            end;

        }

        field(6; Blocked; Boolean)

        {

            DataClassification = ToBeClassified;

        }

        field(7; Date; Date)

        {

            DataClassification = ToBeClassified;

            Editable = false;

        }

        field(8; "Remaining Amount"; Decimal)

        {

            DataClassification = ToBeClassified;

            trigger OnValidate()

            begin

                "Current Remaining Amount" := "Amount" - "Remaining Amount";

            end;

        }

        field(9; "Current Remaining Amount"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

    }

    keys

    {

        key(PK; EntryNo)

        {

            Clustered = true;

        }

    }

    var

        myInt: Integer;

    trigger OnInsert()

    begin

        Date := Today;

    end;

    trigger OnModify()

    begin

    end;

    trigger OnDelete()

    begin

    end;

    trigger OnRename()

    begin

    end;

}

*Figure 1: Table Code*

**Step:**

* Select table template
* Add fields
* Add triggers
* Add conditions

## 2.Create List

We then create a list to display the values in the table created above.

Graphical user interface, application, table

Description automatically generated

*Figure 2: List in Business Central*

page 50114 "Loan List"

{

    PageType = List;

    ApplicationArea = All;

    UsageCategory = Lists;

    SourceTable = LoanHeader;

    CardPageId = LoanHeader;

    Editable = false;

    layout

    {

        area(Content)

        {

            repeater(Details)

            {

                field(EntryNo; Rec.EntryNo)

                {

                    ApplicationArea = All;

                }

                field(No; Rec.No)

                {

                    ApplicationArea = All;

                }

                field(Name; Rec.Name)

                {

                    ApplicationArea = All;

                }

                field(Description; Rec.Description)

                {

                    ApplicationArea = All;

                }

                field(Amount; Rec.Amount)

                {

                    ApplicationArea = All;

                }

                field(Blocked; Rec.Blocked)

                {

                    ApplicationArea = All;

                }

                field(Date; Rec.Date)

                {

                    ApplicationArea = All;

                }

                field("Remaining Amount"; Rec."Remaining Amount")

                {

                    ApplicationArea = All;

                }

                field("Current Remaining Amount"; Rec."Current Remaining Amount")

                {

                    ApplicationArea = All;

                }

            }

        }

        area(Factboxes)

        {

        }

    }

    actions

    {

        area(Processing)

        {

            action(ActionName)

            {

                ApplicationArea = All;

                trigger OnAction();

                begin

                end;

            }

        }

    }

}

*Figure 3: List Code*

**Step:**

* Select list template
* Add fields
* Add conditions

## 3.Create Card

Lastly, we create a card to input values in the table.

Graphical user interface, text, application, email

Description automatically generated

*Figure 4: Card in Business Central*

page 50112 LoanHeader

{

    PageType = Card;

    ApplicationArea = All;

    UsageCategory = Administration;

    SourceTable = LoanHeader;

    layout

    {

        area(Content)

        {

            group(Details)

            {

                field(EntryNo; Rec.EntryNo)

                {

                    ApplicationArea = All;

                }

                field(No; Rec.No)

                {

                    ApplicationArea = All;

                }

                field(Name; Rec.Name)

                {

                    ApplicationArea = All;

                }

                field(Description; Rec.Description)

                {

                    ApplicationArea = All;

                }

                field(Amount; Rec.Amount)

                {

                    ApplicationArea = All;

                }

                field(Blocked; Rec.Blocked)

                {

                    ApplicationArea = All;

                }

                field(Date; Rec.Date)

                {

                    ApplicationArea = All;

                }

                field("Remaining Amount"; Rec."Remaining Amount")

                {

                    ApplicationArea = All;

                }

                field("Current Remaining Amount"; Rec."Current Remaining Amount")

                {

                    ApplicationArea = All;

                }

            }

        }

    }

    actions

    {

        area(Processing)

        {

            action(ActionName)

            {

                ApplicationArea = All;

                trigger OnAction()

                begin

                end;

            }

        }

    }

    var

        myInt: Integer;

}

*Figure 5: Card Code*

**Step:**

* Select card template
* Add fields

We also need to link the card and the list page so that this card design will be displayed when we select a value from the list. We do so by adding the following line to the list page.

    CardPageId = LoanHeader;

# Day 2:

## Linking no. series in a particular field.

First, we assigned a particular field as no. series by writing following code as shown in figure 1. Then we write number series management code inside the onInsert() method as shown in figure 2. At last, we manually insert number series in Microsoft BC.

Text

Description automatically generated

*Figure 1: No. series code.*

A screenshot of a computer screen

Description automatically generated with medium confidence

*Figure 2: No. series code*

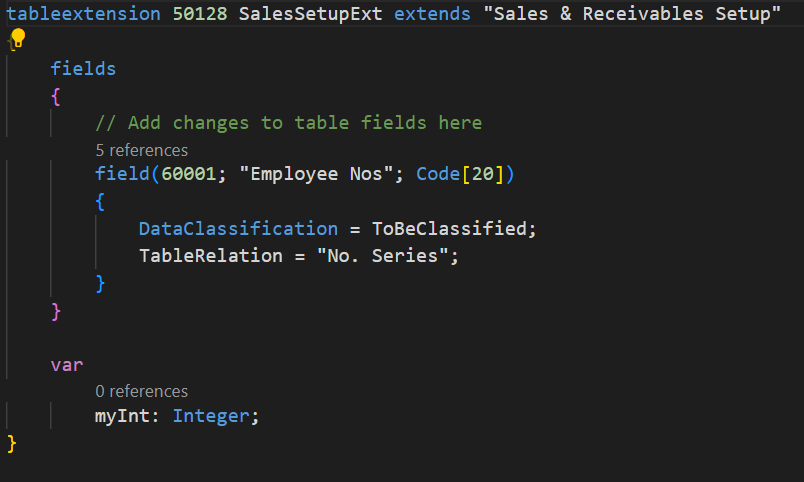
Graphical user interface, application

Description automatically generated

*Figure 3: Manually creating no. series in Business central.*

## Creating table relation

Here, table relation is created between no. series table and SalesSetupExt table. The SalesSetupExt extends “Sales & Receivables Setup”. The “Employee Nos” fields is linked with “No. Series” table.



*Figure 4: Creating Table relation*

## Using SetRange and FindFirst inbuilt function:

Here, we used SetRange() function to filter the necessary data and FindFirst() function to locate the data needed and inserted into fields accordingly.

field(10; "Emp Code"; Code[20])

        {

            DataClassification = ToBeClassified;

            TableRelation = Employees where(Blocked = const(false));

            trigger OnValidate()

            var

                Emp: Record Employees;

            begin

                //get

                if Emp.Get("Emp Code") then

                    validate("Employee Name", Emp.Name);

            end;

        }

        field(11; "Employee Name"; Text[100])

        {

            DataClassification = ToBeClassified;

            trigger OnValidate()

            var

                Emp: Record Employees;

            begin

                //codes

                Emp.Reset();

                Emp.SetRange(Name, "Employee Name");

                //Emp.SetFilter(Name,"Employee Name");

                if Emp.FindFirst() then

                    Address := Emp.Address;

            end;

        }

        field(12; Address; Text[100])

        {

            DataClassification = ToBeClassified;

        }

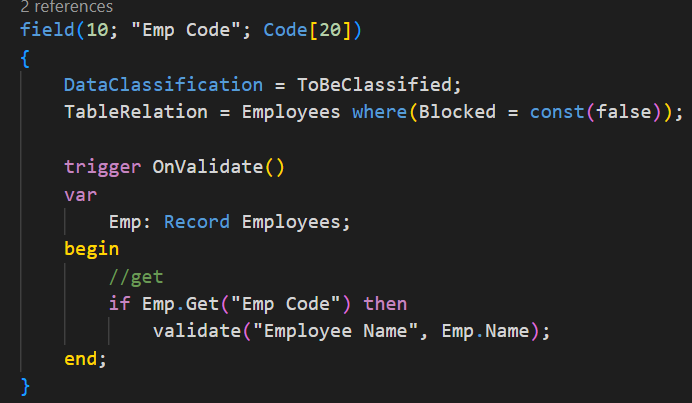
Steps:

* Created three fields: “Emp code”, “Employee Name” and “Address”
* Inside the OnValidate() method of “Employee Name” field a variable is initialized and which is of type record.
* Then SetRange() method is used on the variable to extract the employee name.
* Then FindFirst() method is used to get the first matching record of the entered employee name and if the data matches, the address field of this page automatically by tracking the data from database.

## Using get function to get the data from another table.

Here, we used the get function to get the data from Employees table depending upon the entered “Emp Code” by creating table relation. Here, table relation is created between Employees and LoanHeader table.

Note: Get function can only be used on primary keys fields.



Steps:

* First, the table relation is created.
* Then a variable is initialized of type Record (Employees table’s record).
* Then get function is used and the data is validated.

# Day 3(Sales Billing System)

## 1Create table for Billing Header table.

table 50150 "Billing Header"

{

    DataClassification = ToBeClassified;

    fields

    {

        field(1; No; Code[20])

        {

            DataClassification = ToBeClassified;

            trigger OnValidate()

            var

                SalesRecieve: Record "Sales & Receivables Setup";

                NoSeriesMgt: Codeunit "NoSeriesManagement";

            begin

                IF rec.No <> xRec.No THEN BEGIN

                    SalesRecieve.GET;

                    SalesRecieve.TestField(SalesRecieve."BillingNos");

                    NoSeriesMgt.SetSeries("No");

                END;

            end;

        }

        field(2; "Posting Date"; Date)

        {

            DataClassification = ToBeClassified;

        }

        field(3; "User ID"; Code[20])

        {

            DataClassification = ToBeClassified;

        }

        field(4; "Sell To Customer"; Code[50])

        {

            TableRelation = Customer;

            DataClassification = ToBeClassified;

            trigger OnValidate()

            var

                Cus: Record Customer;

            begin

                if Cus.Get("Sell To Customer") then

                    Validate("Address", Cus.Address);

            end;

        }

        field(5; Address; Text[100])

        {

            DataClassification = ToBeClassified;

        }

        field(6; "Billing No series"; Code[20])

        {

            DataClassification = ToBeClassified;

        }

        field(7; "Total Amount of All Lines"; Decimal)

        {

            FieldClass = FlowField;

            CalcFormula = sum("Billing Line List"."Line Total" where("Document No" = field(No)));

        }

    }

    keys

    {

        key(PK; No)

        {

            Clustered = true;

        }

    }

    var

        myInt: Integer;

    trigger OnInsert()

    var

        SalesSetup: Record "Sales & Receivables Setup";

        NoSeriesMgmt: Codeunit "NoSeriesManagement";

    begin

        IF "No" = '' THEN BEGIN

            SalesSetup.GET;

            SalesSetup.TESTFIELD(SalesSetup."BillingNos");

            CLEAR(NoSeriesMgmt);

            NoSeriesMgmt.InitSeries(SalesSetup."BillingNos", xRec."Billing No series", 0D, "No", Rec."Billing No series");

        END;

        "Posting Date":=Today;

        "User ID":=UserID;

    end;

    trigger OnModify()

    begin

    end;

    trigger OnDelete()

    begin

    end;

    trigger OnRename()

    begin

    end;

}

Steps:

* Select table template
* Add fields with No series management in first field “No.”
* Add Trigger in Sell to customer field to get customer’s Address
* Add OnInsert Trigger
* Add FieldClass Property and its value as FlowField in Total Amount of All Lines to calculate sum of all lines.

## 2. Create List

Graphical user interface, application

Description automatically generated with medium confidence

page 50150 "Billing Header"

{

    PageType = List;

    ApplicationArea = All;

    UsageCategory = Lists;

    SourceTable = "Billing Header";

    CardPageId=BillingHeaderCard;

    layout

    {

        area(Content)

        {

            repeater(Group)

            {

                field(No; Rec.No)

                {

                    ApplicationArea = All;

                }

                field("Posting Date"; Rec."Posting Date")

                {

                    ApplicationArea = All;

                }

                field(UserID; Rec."User ID")

                {

                    ApplicationArea = All;

                }

                field("Sell To Customer"; Rec."Sell To Customer")

                {

                    ApplicationArea = All;

                }

                field(Address; Rec.Address)

                {

                    ApplicationArea = All;

                }

            }

        }

    }

    actions

    {

        area(Processing)

        {

            action(ActionName)

            {

                ApplicationArea = All;

                trigger OnAction()

                begin

                end;

            }

        }

    }

    var

        myInt: Integer;

}

Steps

* Select List template.
* Add fields
* Add Conditions

## 3. Create Card Page

Graphical user interface, text, application, email

Description automatically generated

page 50156 "BillingHeaderCard"

{

    PageType = Card;

    ApplicationArea = All;

    UsageCategory = Administration;

    SourceTable = "Billing Header";

    layout

    {

        area(Content)

        {

            group(GroupName)

            {

                field(No; Rec.No)

                {

                    ApplicationArea = All;

                }

                field("Posting Date"; Rec."Posting Date")

                {

                    ApplicationArea = All;

                }

                field(UserID; Rec."User ID")

                {

                    ApplicationArea = All;

                }

                field("Sell To Customer"; Rec."Sell To Customer")

                {

                    ApplicationArea = All;

                }

                field(Address; Rec.Address)

                {

                    ApplicationArea = All;

                }

                field("Total Amount of All Lines"; Rec."Total Amount of All Lines")

                {

                    ApplicationArea=All;

                }

            }

            group("Lines")

            {

                part(List; ListPart)

                {

                    SubPageLink = "Document No" = field(No);

                    ApplicationArea = All;

                    UpdatePropagation = Both;

                }

            }

        }

    }

    actions

    {

        area(Processing)

        {

            action(ActionName)

            {

                ApplicationArea = All;

                trigger OnAction()

                begin

                end;

            }

        }

    }

    var

        myInt: Integer;

}

Steps

* Select card template
* Add fields
* Add another group for lines to connect Listpart with Card

## 4. Create table for Listpart

table 50180 "Billing Line List"

{

    DataClassification = ToBeClassified;

    fields

    {

        field(1; "Document No"; Code[20])

        {

            DataClassification = ToBeClassified;

        }

        field(2; "Item No"; Code[50])

        {

            TableRelation = Item;

            DataClassification = ToBeClassified;

            trigger OnValidate()

            var

                Itms: Record Item;

            begin

                if Itms.Get("Item No") then begin

                    Validate("Description", Itms.Description);

                    Validate("Unit Price", Itms."Unit Price");

                end;

            end;

        }

        field(3; "Line No."; Integer)

        {

            DataClassification = ToBeClassified;

         //   AutoIncrement = true;

        }

        field(7; Description; Text[100])

        {

            DataClassification = ToBeClassified;

        }

        field(4; "Quantity"; Decimal)

        {

            trigger OnValidate()

            var

                myInt: Integer;

            begin

                Rec."Line Total" := Rec.Quantity \* Rec."Unit Price";

            end;

        }

        field(5; "Unit Price"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

        field(6; "Line Total"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

    }

    keys

    {

        key(PK; "Document No", "Line No.")

        {

            Clustered = true;

        }

    }

    var

        myInt: Integer;

    trigger OnInsert()

    begin

    end;

    trigger OnModify()

    begin

    end;

    trigger OnDelete()

    begin

    end;

    trigger OnRename()

    begin

    end;

}

Steps

* Select table template
* Add fields
* Add conditions

## 5. Create page for ListPart

page 50191 "ListPart"

{

    PageType = ListPart;

    SourceTable = "Billing Line List";

    AutoSplitKey=true;

    // Filter on the sales orders that are pending completion.

    layout

    {

        area(Content)

        {

            repeater(General)

            {

                field("Document No"; rec."Document No")

                {

                    TableRelation = "No. Series";

                    ApplicationArea = All;

                }

                field("Item No"; rec."Item No")

                {

                    ApplicationArea = All;

                    TableRelation = Item;

                }

                field("Line No."; rec."Line No.")

                {

                    ApplicationArea = All;

                }

                field("Description"; rec."Description")

                {

                    ApplicationArea = All;

                }

                field("Quantity"; rec.Quantity)

                {

                    ApplicationArea = All;

                     trigger OnValidate()

                    var

                        myInt: Integer;

                    begin

                        CurrPage.UPDATE(true)

                    end;

                }

                field("Unit Price"; rec."Unit Price")

                {

                    ApplicationArea = All;

                    trigger OnValidate()

                    var

                        myInt: Integer;

                    begin

                        CurrPage.UPDATE(true)

                    end;

                }

                field("Line Total"; rec."Line Total")

                {

                    ApplicationArea = All;

                }

            }

        }

    }

}

Text

Description automatically generated with low confidence

Steps:

* Select page template
* Add AutosplitKey property as true for this page to increase the line no. by 10000 on each insert because it is primary key.
* Add fields
* Add CurrPage.Update(true) for Unit price and quantity fields to auto update the field’s value without refreshing the page.

Final Output

Table

Description automatically generated

Day 4:

1.Create Posting Tables

First we create posting tables for Header and Line to store the posted data.

table 60110 "PostedPurchaseHeader"

{

    DataClassification = ToBeClassified;

    fields

    {

        field(1; "No."; Code[20])

        {

            DataClassification = ToBeClassified;

        }

        field(2; "Posting Date"; Date)

        {

            DataClassification = ToBeClassified;

        }

        field(3; "User ID"; Text[100])

        {

            DataClassification = ToBeClassified;

        }

        field(4; "Buy From Vendor"; Code[20])

        {

            DataClassification = ToBeClassified;

        }

        field(12; "Vendor Name"; Text[100])

        {

            DataClassification = ToBeClassified;

        }

        field(5; Address; Text[200])

        {

            DataClassification = ToBeClassified;

        }

        field(6; "Total Amount"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

        field(27; "Received By"; Code[20])

        {

            DataClassification = ToBeClassified;

        }

        field(99; "Amount After Tax"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

    }

    keys

    {

        key(Key1; "No.")

        {

            Clustered = true;

        }

    }

    var

        myInt: Integer;

    trigger OnInsert()

    begin

    end;

    trigger OnModify()

    begin

    end;

    trigger OnDelete()

    begin

    end;

    trigger OnRename()

    begin

    end;

}

*Figure 1: Header Table*

table 60103 "PostedPurchaseLine"

{

    DataClassification = ToBeClassified;

    fields

    {

        field(1; "Document No."; Code[20])

        {

            DataClassification = ToBeClassified;

        }

        field(2; "Item No."; Code[20])

        {

            DataClassification = ToBeClassified;

        }

        field(3; "Line No."; Integer)

        {

            DataClassification = ToBeClassified;

        }

        field(4; Qty; Integer)

        {

            DataClassification = ToBeClassified;

        }

        field(5; "Unit Price"; Integer)

        {

            DataClassification = ToBeClassified;

        }

        field(6; "Line Total"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

        field(7; "Item Name"; Text[200])

        {

            DataClassification = ToBeClassified;

        }

        field(21; "Amount after Tax"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

    }

    keys

    {

        key(Key1; "Document No.", "Line No.")

        {

            Clustered = true;

        }

    }

    var

        myInt: Integer;

    trigger OnInsert()

    begin

    end;

    trigger OnModify()

    begin

    end;

    trigger OnDelete()

    begin

    end;

    trigger OnRename()

    begin

    end;

}

*Figure 2: List Part Table*

**Steps:**

* Select table template
* Add fields

We also create lists and cards for the tables above.

Table

Description automatically generated

*Figure 3: Header List*

Graphical user interface, text, table, email

Description automatically generated

*Figure 4: Header Card and Purchase Lines*

2.Write codeunit for posting

Then we create a codeunit and add procedures to post the data into the posted tables.

    procedure PostPurchToPostedPurch(PurchCode: Code[20])

    var

        PurchaseHeader: Record PurchaseHeader;

        PurchLine: Record PurchaseLine;

        PostedPurchaseHeader: Record "PostedPurchaseHeader";

        PostedPurchaseLine: Record "PostedPurchaseLine";

        LocalAmtAfterTax: Decimal;

        totalAmount: Decimal;

        PurchAndRec: Record "Purchases & Payables Setup";

        Item: record Item;

    begin

        PurchAndRec.Get();

        if PurchaseHeader.Get(PurchCode) then begin

            PurchaseHeader.TestField("Buy From Vendor");

            PostedPurchaseHeader.Init();

            PostedPurchaseHeader.TransferFields(PurchaseHeader);

            PostedPurchaseHeader.Insert(true);

            PostedPurchaseHeader."Received By" := PurchaseHeader."Received By";

            PostedPurchaseHeader.Modify();

            Clear(totalAmount);

            PurchLine.Reset();

            PurchLine.SetRange("Document No.", PurchaseHeader."No.");

            if PurchLine.FindSet() then

                repeat

                    PostedPurchaseLine.Init();

                    PostedPurchaseLine.TransferFields(PurchLine);

                    TaxCalculation(LocalAmtAfterTax, PurchLine."Line Total");

                    PostedPurchaseLine."Amount after Tax" := LocalAmtAfterTax;

                    PostedPurchaseLine.Insert();

                    InsertIntoPurchLedger(PurchLine, LocalAmtAfterTax);

                    totalAmount += PurchLine."Line Total";

                until PurchLine.Next() = 0;

            PurchLine.SetRange("Item No.", Item."No.");

            PostedPurchaseHeader."Total Amount" := totalAmount;

            PostedPurchaseHeader."Amount After Tax" := totalAmount + (totalAmount \* (PurchAndRec."Purchase Tax %"));

            PostedPurchaseHeader.Modify();

            PurchLine.DeleteAll();

            PurchaseHeader.Delete(true);

        end;

    end;

*Figure 5: Procedure to Post Data*

**Steps:**

* Select codeunit template
* Create procedure to post data
* Declare the required variables
* Write the logics and conditions.

We also need to write an action in the Header card to call the post procedure.

actions

    {

        area(Processing)

        {

            action("Post")

            {

                ApplicationArea = All;

                Promoted = true;

                PromotedIsBig = true;

                Image = Post;

                trigger OnAction()

                var

                    BillMgt: Codeunit "Purchase Management";

                begin

                    if not Confirm('Do you want to post?', false) then

                        exit;

                    BillMgt.PostPurchToPostedPurch(Rec."No.");

                    Message('Document has been Posted');

                end;

            }

        }

    }

*Figure 6: Procedure to Post Data*

**Steps:**

* Add an action on the header card page
* Call the procedure if the required conditions are met
* Pass the required parameters

3.Create Procedure for tax calculation

We then create a procedure to calculate the total amount after tax.

procedure TaxCalculation(var AmtAfterTax: Decimal; lineAmt: Decimal)

    var

        PurchAndPay: Record "Purchases & Payables Setup";

    begin

        PurchAndPay.Get();

        AmtAfterTax := (PurchAndPay."Purchase Tax %" / 100) \* lineAmt + lineAmt;

    end;

*Figure 7: Procedure to calculate tax*

**Steps:**

* Create a procedure
* Add the required parameters and variables
* Write the code to calculate tax

## 4.Create Ledger Table.

We then create a table for the ledger.

table 60115 "Purchase Ledger Entry"

{

    DataClassification = ToBeClassified;

    fields

    {

        field(1; "Entry No."; Integer)

        {

            DataClassification = ToBeClassified;

        }

        field(2; "Document No."; Code[50])

        {

            DataClassification = ToBeClassified;

        }

        field(3; "Item No."; Code[50])

        {

            DataClassification = ToBeClassified;

        }

        field(4; Quantity; Integer)

        {

            DataClassification = ToBeClassified;

        }

        field(5; "Unit Price"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

        field(6; "Line Total"; Decimal)

        {

            DataClassification = ToBeClassified;

        }

        field(18; "Customer No."; Code[20])

        {

            DataClassification = ToBeClassified;

        }

    }

    keys

    {

        key(Key1; "Entry No.")

        {

            Clustered = true;

        }

    }

    var

        myInt: Integer;

    trigger OnInsert()

    begin

    end;

    trigger OnModify()

    begin

    end;

    trigger OnDelete()

    begin

    end;

    trigger OnRename()

    begin

    end;

}

*Figure 8: Ledger Table*

Table

Description automatically generated

*Figure 9: Ledger List*

**Steps:**

* Create a table for ledger
* Create List

## 5.Create Procedure for Inserting into Ledger

Lastly, we create a procedure to insert data into the billing Ledger.

  local procedure InsertIntoPurchLedger(PurchLine: Record "PurchaseLine"; taxAmt: Decimal)

    var

        PurchLedger: Record "Purchase Ledger Entry";

        PurchHeader: Record PurchaseHeader;

        CusNo: Code[20];

    begin

        PurchLedger.Init();

        if PurchLedger.FindLast() then

            PurchLedger."Entry No." += 1

        else

            PurchLedger."Entry No." := 1;

        PurchLedger."Document No." := PurchLine."Document No.";

        PurchLedger."Item No." := PurchLine."Item No.";

        PurchLedger."Line Total" := taxAmt;

        PurchLedger.Quantity := PurchLine.Quantity;

        PurchLedger."Customer No." := GetCustomerNoFromBillHeader(PurchLine."Document No.");

        ;

        PurchLedger.Insert();

    end;

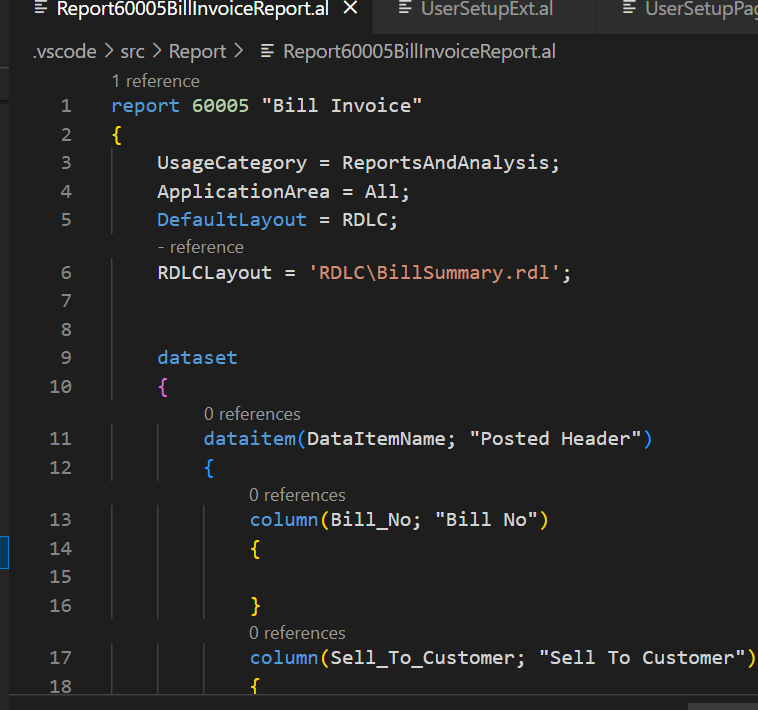
*Figure 10: Procedure to add data to Ledger*

**Steps:**

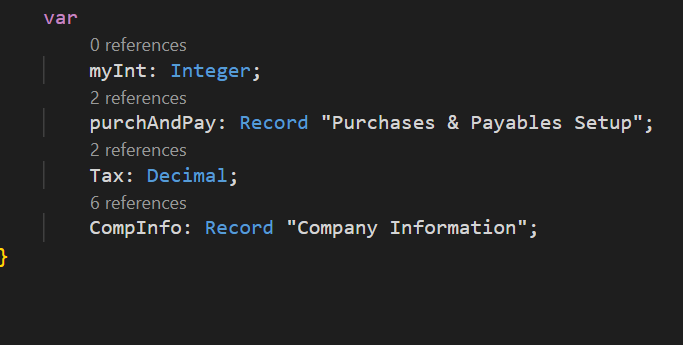
* Create the required procedure
* Pass the parameters and create variables
* Write the code
* Pass the procedure in the Post Management Procedure above.

# Day 5:

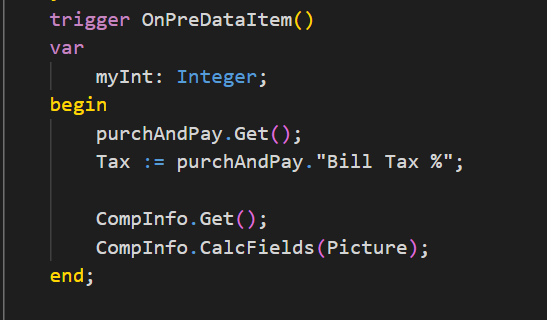
## Making Report in Business Central

First we create a folder named Report in the VSCode and inside the folder we create a file. Then we create a report of named **“Bill Invo”**. As the report is made only for the posted document, in the dataitem we link the posted Billing Header table named **“Posted Header”.** Inside the dataitem, we add the column as per our needs.   

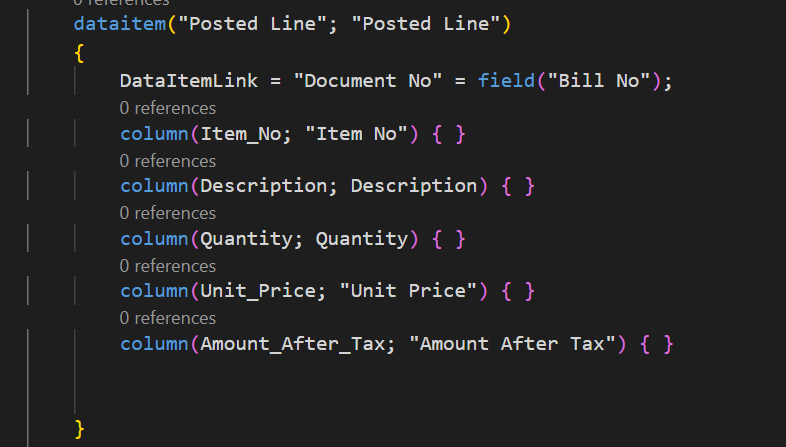

We also added the column that contains the company information. For this firstly, we declare a variable named **CompInfo** at the end of the Bill Invoice Report which contains all the data related to the specific Company.



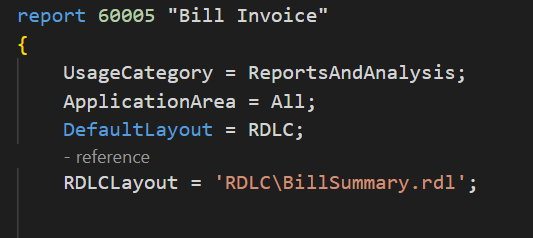
We declare a variable for Company Information, now we have to call it to get data. So we made a trigger named **OnPreDataitem()** inside the Dataitem set of Posted Header and we get the company information as follow.



Now, we also need the field from Posted Billing Line. So we created another dataitem set for Posted Billing Line where we link the Posted Billing Table named “**Posted Line**”. Previously, we have linked the Posted Billing Line with Posted Billing Header, so here we also placed the subpage link which shows the relationship between Posted Billing Header and Posted Billing Line. Then we define column as per our needs.

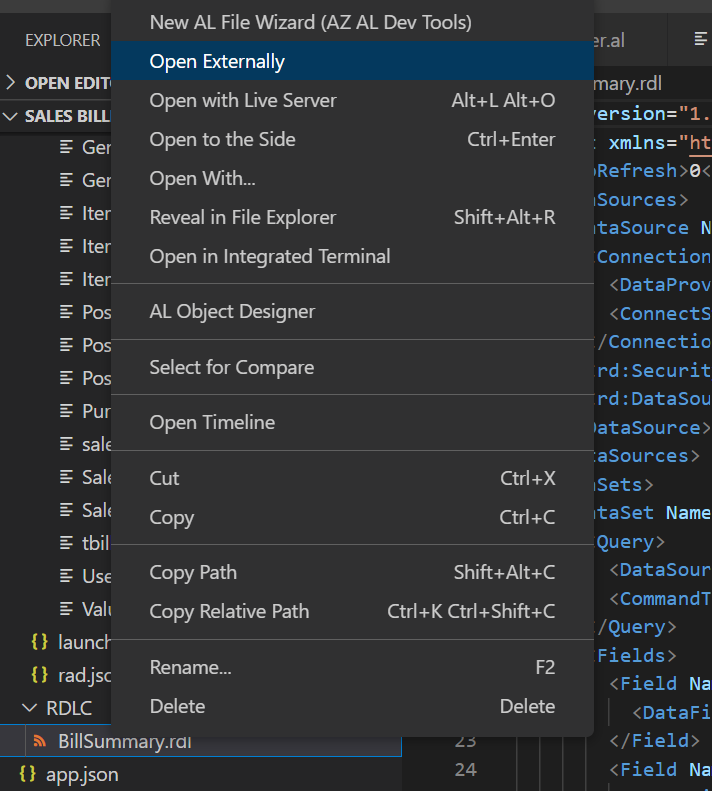


Now, we need to build the report but before building the report we need to assign path to the report. So, in the property of the **Bill Invoice report** we added following Properties.

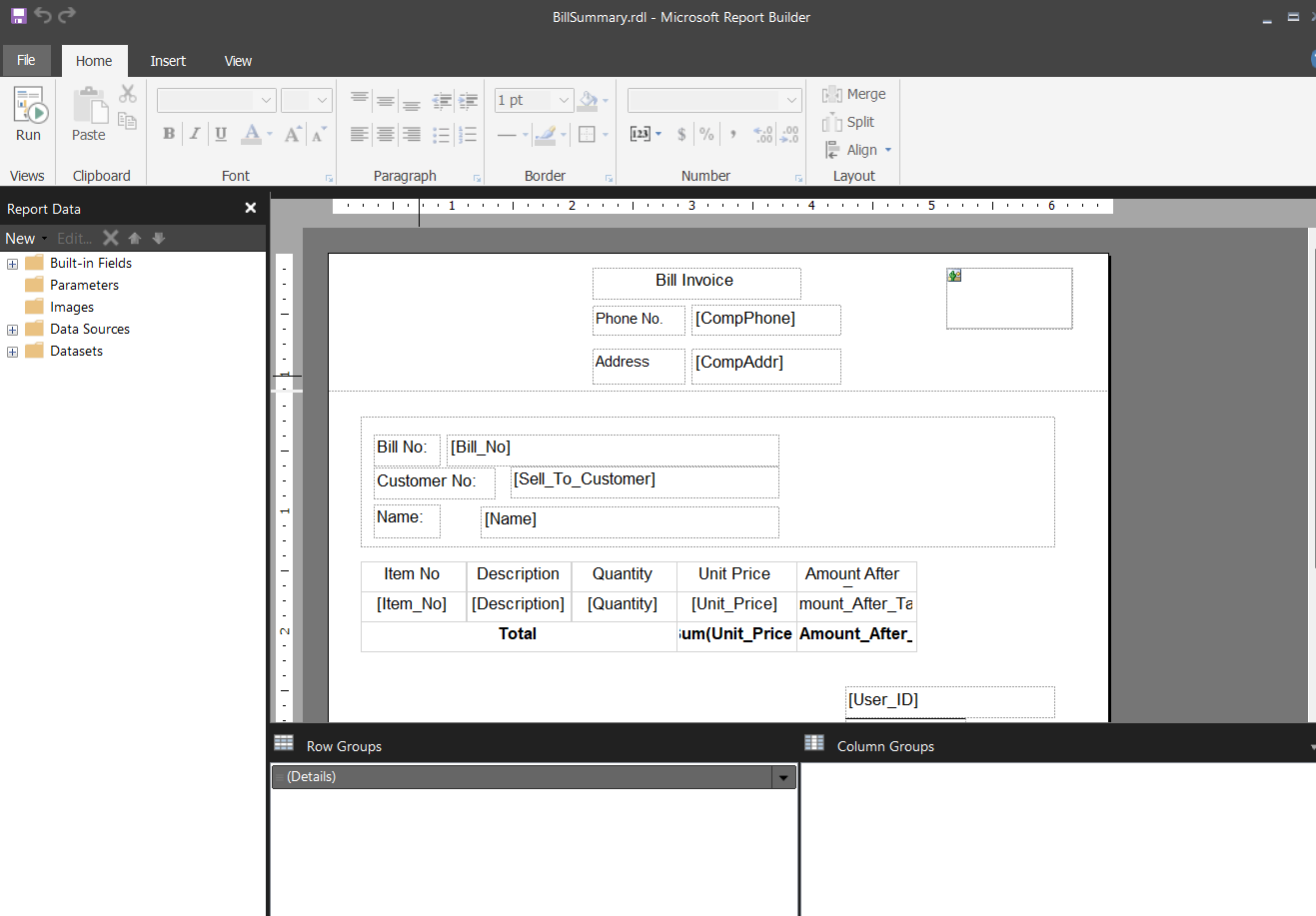


Here we have added the property **DefaultLayout = RDLC** which specifies the layout is RDLC. And the other property RDLCLayout = **‘RDLC\BillSummary.rdl ’** assign a path to the layout that is being generated. When we build this report this will automatically create a folder RDLC and place the file BillSummary.rdl inside of it.

After we build this report using key **(Ctrl + Shift + B),** the BillSummary will be build inside the RDLC file. Now we right click on the BillSummary and goes to Open Externally.



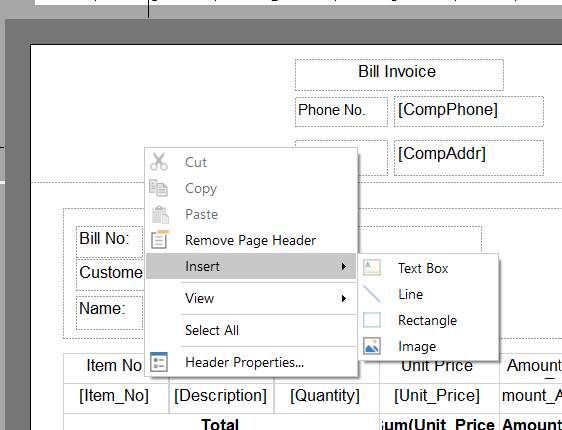
This will open a Report Builder where we can design our report as we want.



We have design our report in the report builder. Here we first create page header and page footer by right clicking on the white space and selecting insert: page header, insert: page footer.

**Inserting Text Box in Report Builder**

Then we insert Text box in the same way and inside the text box we write the text we want.



To get the information from the posted Header and posted Line, we write a formula. For example:

To get the company phone number, we have the formula **=fields!CompPhone.value** which we placed inside the textbox near to Phone No. textbox. Here, the CompPhone indicates the columnname that we have defined while adding the column in dataitem set of report Bill Invoice. In same way, we can get the Address, Bill No, Customer No, Name and User ID.

**Inserting Image in Report Builder**

**Step 1:**

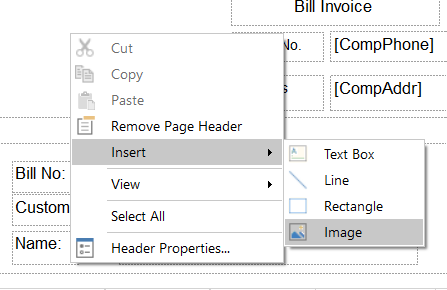


Figure 1

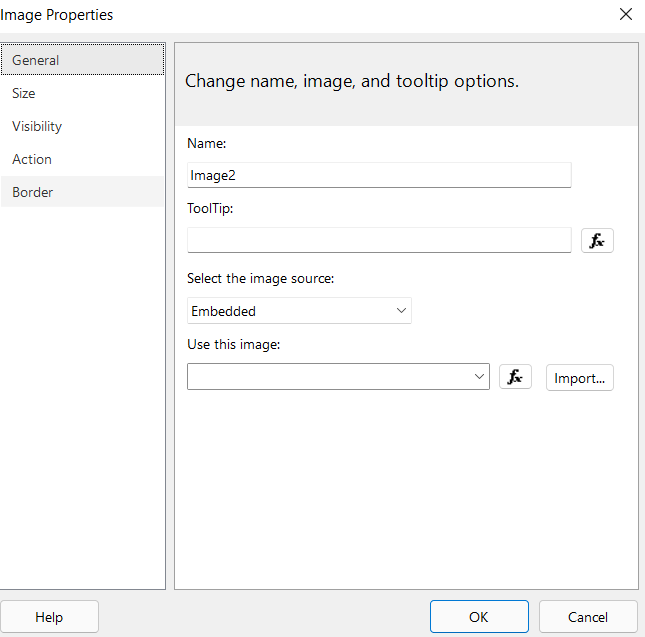
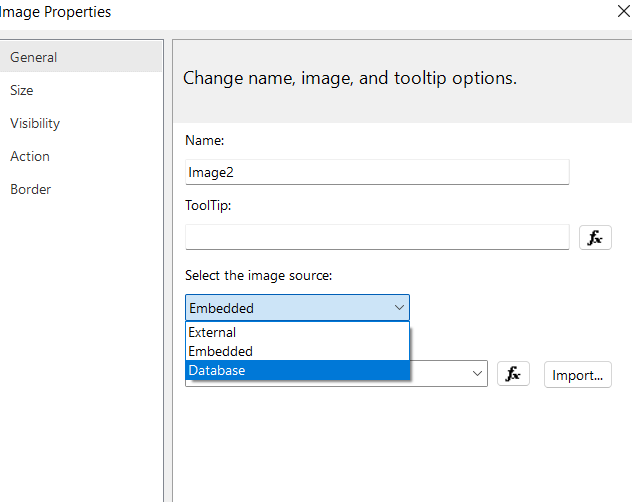


Figure 2

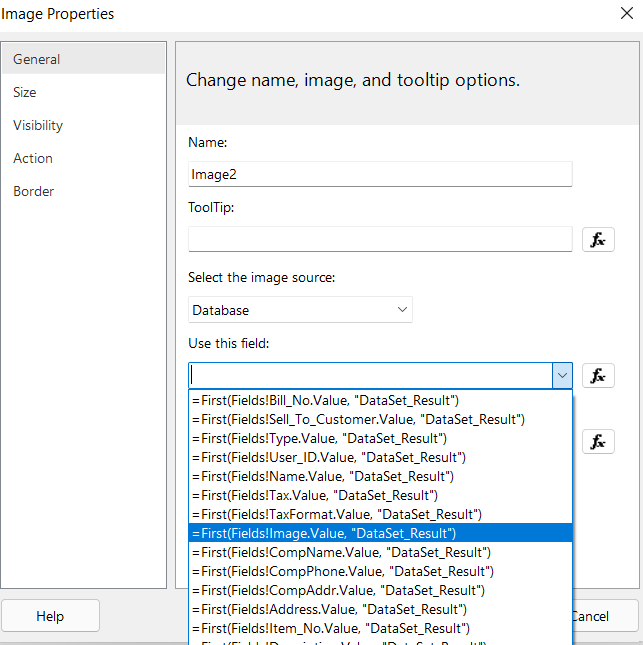
Here, at first we insert an image in a way shown in figure 1 which will open up a dialog box as shown in figure 2.

**Step 2:**



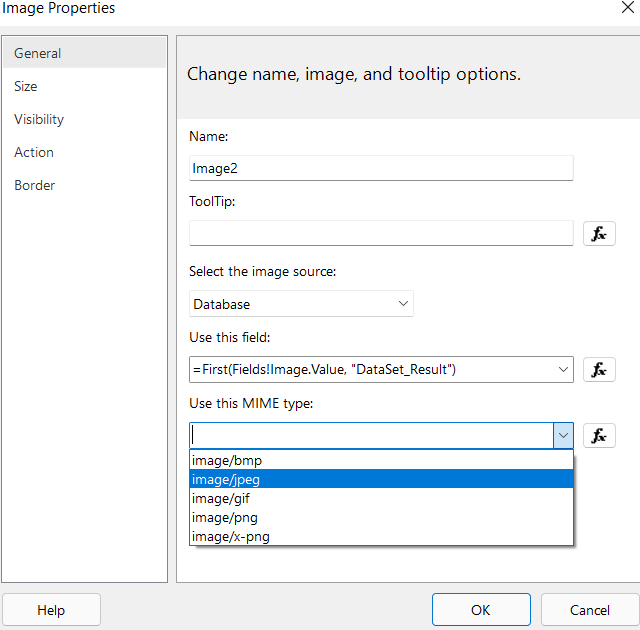
We then select Database in the select the image source field because we are getting the image from the database.

**Step3:**



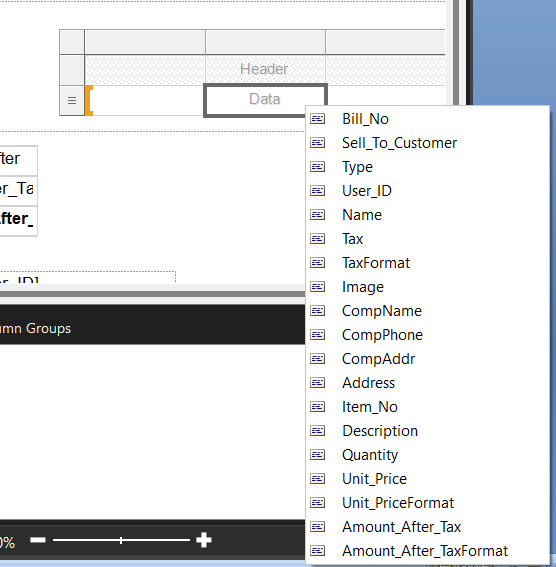
We then select **=First(Fields!Image.value,”Dataset\_Result”)** in the Use this Field section.

**Step 4:**

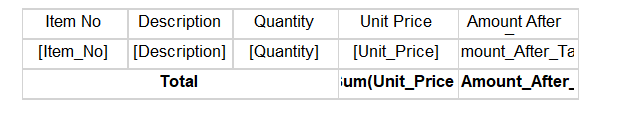


Then we select the MIME type as jpeg.

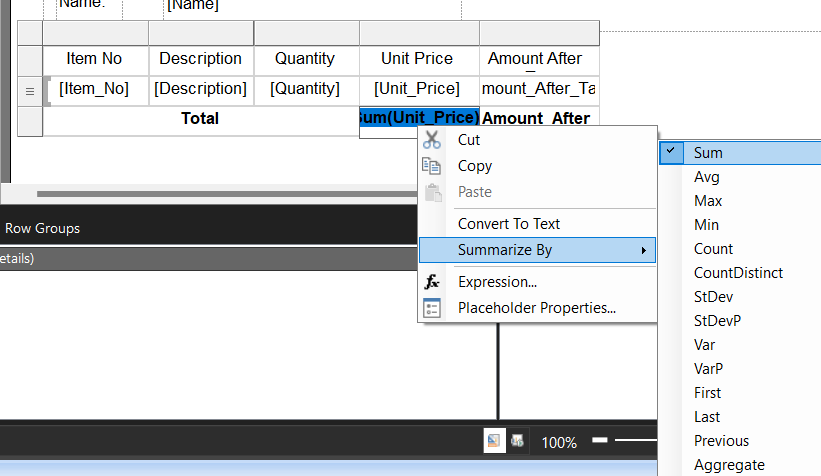
**Inserting table in Report Builder**



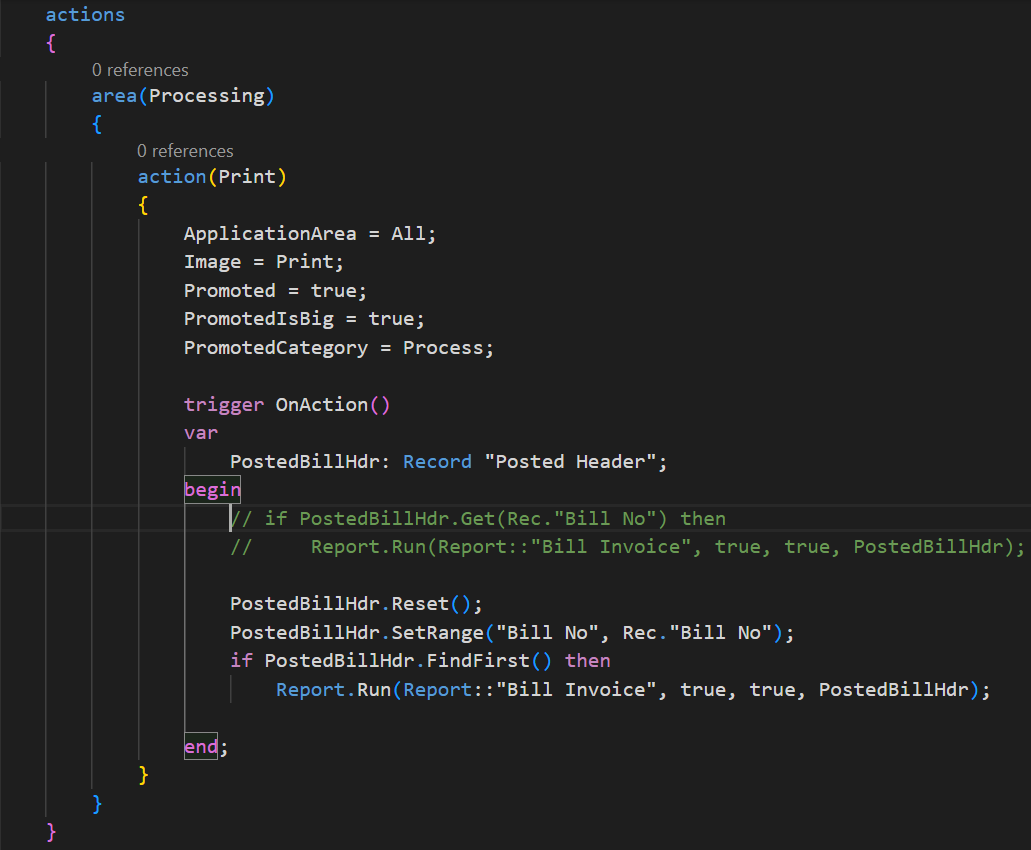
We insert a table in the same way as we insert image and text. In the table we can select the field we want to add. After that we created a table as follow:



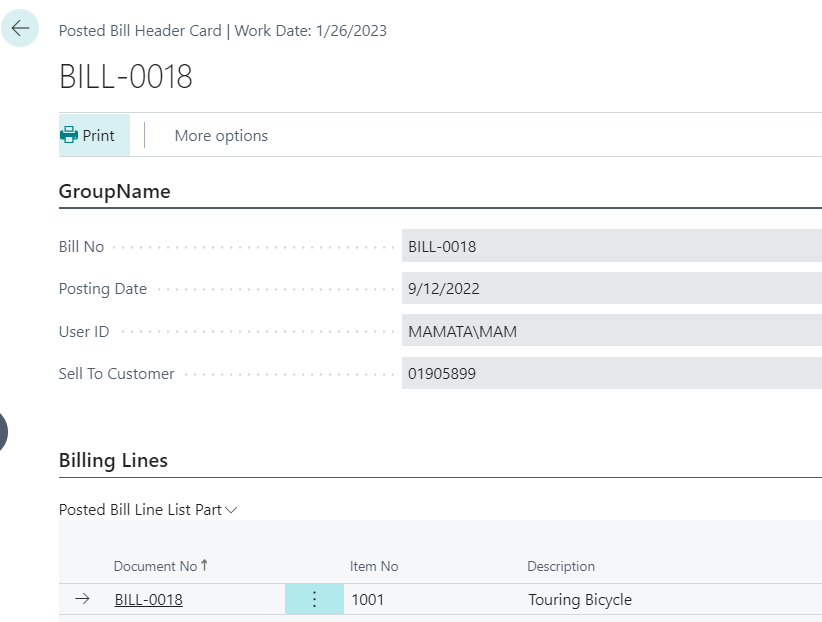
To calculate the sum of Unit price and Amount After tax, we summarize these fields as shown below:



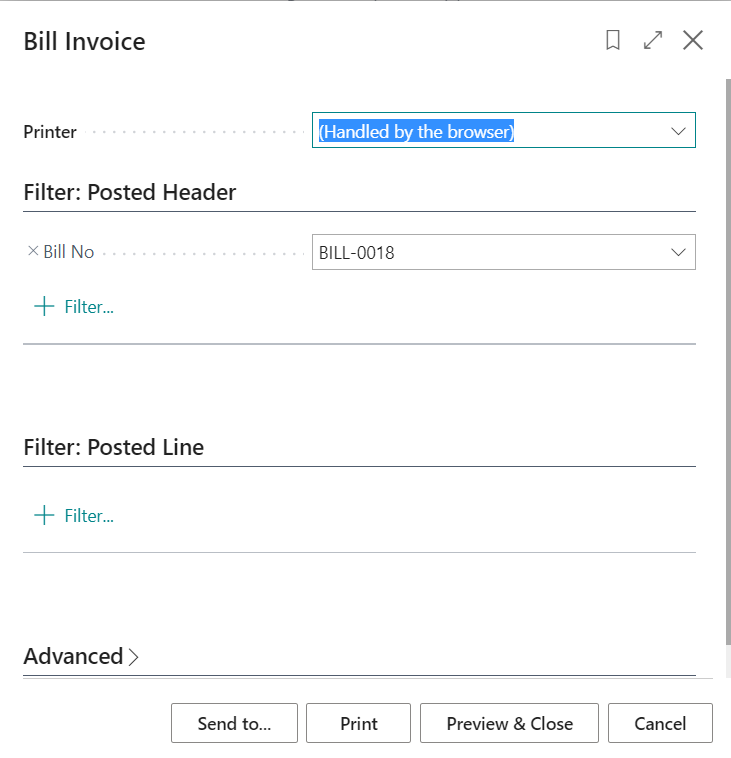
After this we need to define an action to print this report. So in the Posted Billing Header Card Page we define an action.



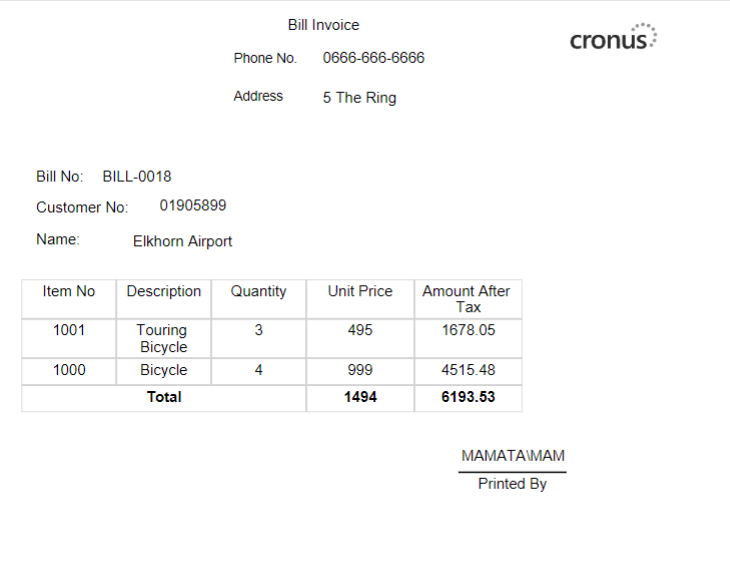
When we publish this, it will create an action **Print** in the Posted Billing Header card as:



When we click on the print it will open up a dialog box as:



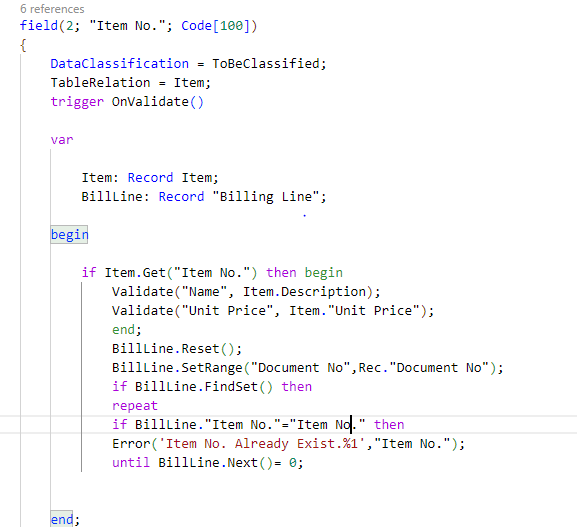
Now when we press on Print it will print the report. To preview what our final report looks like we can go to the Preview & Close option and see the final report that can be printed. The final report looks like:



# Day 5:

## 1.Item-No Duplication

Firstly, we reset the bill line. Then we compare the document no. We are Inserting to the previously inserted Document number using Set Range property in Bill Line. Then we started the loop in bill line using Find set where if the item No. we are inserting match with the Previously entered item no then it will throw an error saying “The item already exist” along with the item no.



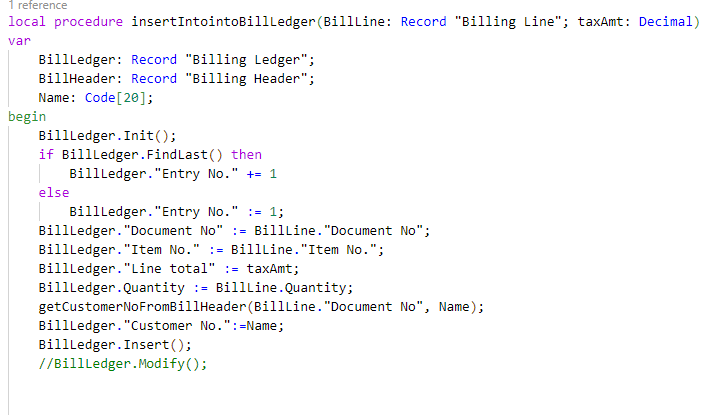
**Table

Description automatically generated**

## 2. Inserting Data and CustomerName To Posted Billing Line Ledger

Firstly, we have manually inserted data to Bill Ledger from Billing line then using get function we have inserted Customer Name in to Billing Ledger.

Manually Transfering data from Bill Line to Billing Ledger



Adding Customer Name Field from Bill Header to Billing Ledger

Graphical user interface, text, application

Description automatically generated

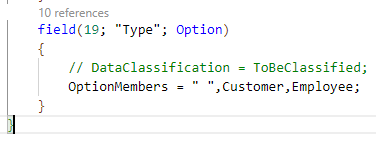
**Graphical user interface, application

Description automatically generated with medium confidence**

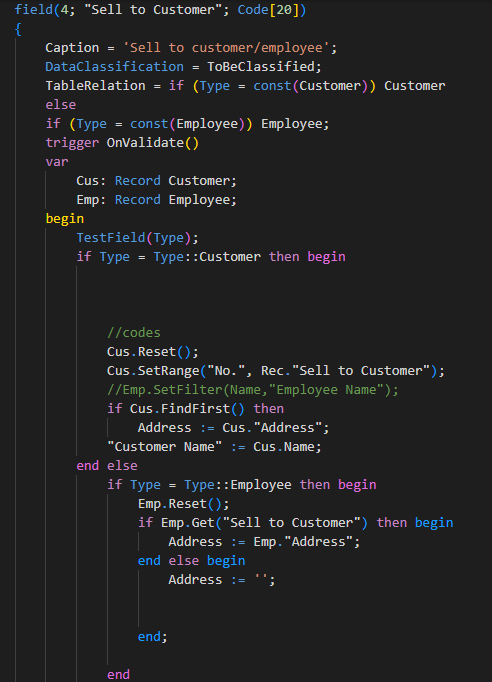
## 3.Type

We Have created a Field called “Type” In Billing Header Card with a option Data type with an option of Customer and Employee then we have declared a relation of type with customer and employee with a feature where when we select customer it will show data of customer and when we select employee it will show data of Employee. We have also Declared type as Mandatory Field with a function where if we don’t select type and try to post a data then it will throw an exception saying, “Type is Mandatory to Select before posting”.

Adding Field to Billing header



Declaring Relation of type with customer and employee in “Sell to customer” field from billing header table



Function for throwing exception in Code unit ”billing management”

Text

Description automatically generated

Adding field in Table Posted Billing header



Text

Description automatically generated

Adding trigger in Billing header List Page

Graphical user interface, text, application, email

Description automatically generated

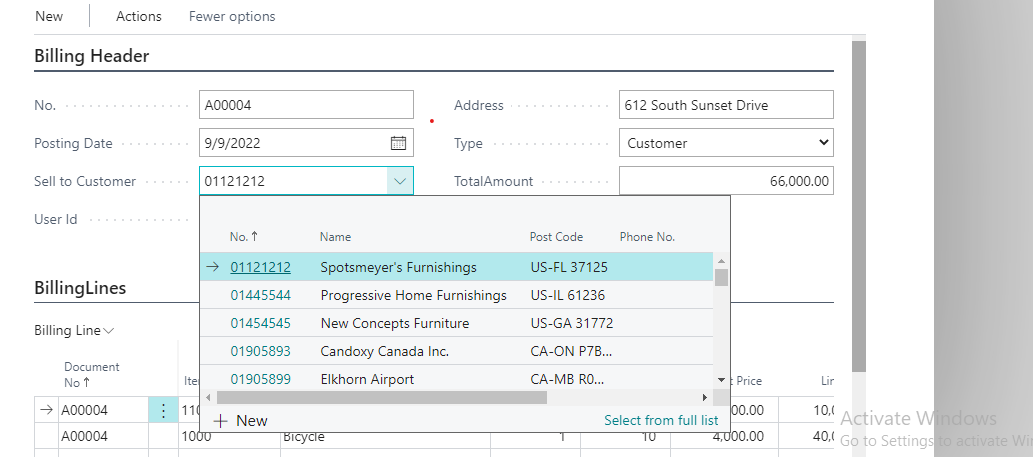
Adding SourceTableView in Billing Header List Employee

Text

Description automatically generated

**Graphical user interface, text, application, email

Description automatically generated**

****

**Adding type into BillingHeaderCardPage**

Graphical user interface, text, application, email

Description automatically generated

## 4. General Journal

We have added field “Training” in General Journal and G/L Entry Extension Table

Graphical user interface, text, application

Description automatically generated

**Table

Description automatically generated**

Graphical user interface, text, application, email

Description automatically generated

## 5. Adding Page extension for adding “Training” Field in “General Journal”

Graphical user interface, text, application, email

Description automatically generated

## 6. Event Subscription

We have created a File called “Event Subscription” in Code unit then We have added an Event to transfer General Journal Line to G/l Entry

Text

Description automatically generated

# Day 6:

## Passing the Tax Amount to General Journal from Sales Line.

Extending new field in Sales Line Table and using it to calculate and store Sum Amount of each line and pass it into the general journal with the help of event subscription by using OnAfterCopyGenJnlLineFromSalesHeader event.

Since there are multiple line in Sales line we have to use loop and calculate the sum of all tax amount and store it in a single variable. In this case, the Total Sum of tax is stored in SumAmt variable as shown in figure 2.

Text

Description automatically generated

Text

Description automatically generated

## Passing Tax Amount from Sales Line to Item Journal Line

Extending new field in Item Journal Line table then with the help of Event subscription using the OnAfterCopyItemJnlLineFromSalesLine event to pass the value of Tax amount from SalesLine to Item Journal Line.

Text

Description automatically generated

A screenshot of a computer

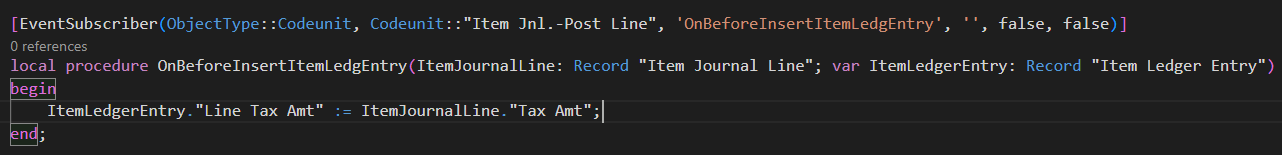
Description automatically generated with medium confidence

## Passing the Line Tax Amount from Item Journal Line to Item Ledger Entry.

Extending new field in Item Ledger Entry table then with the help of Event subscription using the OnBeforeInsertItemLedEntry event to pass the value of Tax amount from Item Journal line to Item Ledger Line.

A screenshot of a computer

Description automatically generated with medium confidence



## Passing the Tax Amount from Item Journal Line to Value Entry.

Extending new field in Value Entry table then with the help of Event subscription using the OnBeforeInsertValueEntry event to pass the value of Tax amount from Item Journal Line to Value Entry.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

## Set filter function:

Syntax:

Record.SetFilter(Field: Any, String: Text [, Value: Any,...])

Record

 Type: **Record**

An instance of the Record data type.

Field

 Type: **Any**

The field that you want to filter.

String

 Type: **Text**

The filter expression. A valid expression consists of alphanumeric characters and one or more of the following operators: <, >, \, &, |, and =. You can use replacement fields (%1, %2, and so on) to insert values at run-time.

[Optional] Value

 Type: **Any**

Replacement values to insert in replacement fields in the filter expression. The data type of Value must match the data type of Field

Example:

1. SetFilter(Amt, ‘%1..%2’, 1000, 2000 )
2. SetFilter(Data, ‘>%1’,Today-30)

In the first example, the SetFilter filters data Record which has the matching key word “xyz”.

In the second example, the SetFilter filters the data from past one month.

## CodeUnit

CodeUnit is a container for AL code which can be used in multiple different application object. CodeUnit contains the business logic and can be called in application area where there is need of that specific logic.

CodeUnit Contains Procedures, EventSubscribers, etc. CodeUnit can be used both as a direct call to **codeunit.run(customer)** or as a call to the procedure inside the codeunit **createcustomer.CheckSize(customer).**

## Event Subscription:

An event subscriber is an AL method that listens to a specific event and determines which action to take when that event is raised. Sometimes you need to subscribe to an event, but you need to control when that method code must be run

# Day 7

***Documentation on User Setup and Report Update by Mr. Bishwas Khatri Poudel on 12th September.***

***Setting a user setup –Weather to provide post bill access to user or not.***

## Adding a Field in User Setup Table and Showing in User Card.

* Step-01: Open the VS Code to edit the User Setup Table.
* Step-02: Firstly, Create a Table Extension for User Setup because we are working on existing table with “TTableExt”.
* Graphical user interface, text

  Description automatically generatedStep-03: Add a table field name “Allow Bill Post” with a Boolean Data Type.

*Figure -01****:*** User Setup Table Extension with added Field.

* Step-04: Now, After we have created a table extension, we need to create a page Extension for the table to display in User Card.
* Step-05: Use (TPageExt) to create a Page Extension for the User Setup Table we already created in Step-01 .
* Step-06 : In Table Page Extension we need show the field after “Allow Posting From”. Here ,we use a function name “addafter” that will the place the field after “Allow Posting From” as shown in the figure-02.

Graphical user interface, text, application

Description automatically generated

*Figure-02*: User Setup Page Extension with a Field to display in Card.

## Creating a procedure to call a function that to check if a user is allow to post bill or not.

* Step-01: Open the VS Code to start editing .
* Text

  Description automatically generatedStep-02: We have build Table and Page Extension for the User Setup. Now, we pursue to build a procedure in Bill management inside ‘CodeUnit’ that Check If a “User Can Post a Bill or Not” code is displayed in Figure-03.

*Figure-03*: Procedure Section to Check If User Can Post Bill.

* Step-04: Similarly, after building a procedure we need to check from “Purchase and Payable” procedure in “CodeUnit” inside Bill Management and code is displayed in Figure-04.

Text

Description automatically generated

*Figure-04* : Error response while User cannot post the Bill.

***Update on Posted Bill (Report).***

## Creating a Bill Post to update the report .

* Step-01: Firstly, we create two field in the report update page.
* Step-02: Create two variables of type record to instiate those variables. Shown in Figure-05.

Text

Description automatically generated

*Figure-05*: Added to field in Update report Page.

* Step-03: Then, we apply the setRange() function on PostedBillHeader to compare between PostedBillHeader and PostedBill. Shown in Figure-06.
* Step-04: Report property must have a value of “Processing only”=true.

Text

Description automatically generated

*Figure-06*: Shown the comparison between PostedBillHeader with PostedBill .

# .Here a Screen Shot On Allow Bill Post from Business Central.

Table

Description automatically generated

*Figure-07*: Allow Posting Section shown with a checkbox Below.

Where we can tick for Allowing the post to happen.

# Day 8

## Add name from header to line

First, we learnt how to take data from header and add it to its list part.

    trigger OnInsert()

    var

        purchhead: record PurchaseHeader;

    begin

        if purchhead.get("Document No.") then

            "Vendor Name" := purchhead."Vendor Name";

    end;

*Figure 1: Code in trigger of the List Part*

PurchLine.Reset();

if PurchLine.FindSet() then

                    repeat

                        PurchLine."Vendor Name" := "Vendor Name";

                        PurchLine.Modify();

                    until PurchLine.Next() = 0;

*Figure 2: Code in trigger of the Header table*

**Steps:**

* Add the insert code into the trigger of the list part table
* Then add the insert code into a trigger of the header table

## 2.Filter Group

We also added Filter Group to prevent the user from removing particular filters.

Graphical user interface, text, application

Description automatically generated

*Figure 3: FilterGroup in Posted List*

**Steps:**

* Add the filter group to the code in the trigger applying the filter.

## 3.Change Base Report

We also learnt how to change a base report.

Text

Description automatically generated

*Figure 4: Copy of Base Report*

Table

Description automatically generated

*Figure 5: Copied RDLC Layout*

**Steps:**

* Copy the code of the base report from the Microsoft base application code in alpackages
* Modify the contents of the code as required.
* Copy the RDLC file and paste it in your rdlc folder
* Change RDLC to RDL extension
* Open externally and change format as required.

## 4.Calcfields

Furthermore, we learnt how to use Calcfields in pages.

action("Total Amount by calcsum")

            {

                ApplicationArea = All;

                Promoted = true;

                PromotedIsBig = true;

                Image = Calculate;

                trigger OnAction()

                var

                    purchLine: Record PurchaseLine;

                    sumamt: Decimal;

                begin

                    sumamt := 0;

                    purchLine.Reset();

                    purchLine.CalcSums("Line Total");

                    Message(Format(purchline."Line Total"));

                end;

            }

            action("Total Amount")

            {

                ApplicationArea = All;

                Promoted = true;

                PromotedIsBig = true;

                Image = AmountByPeriod;

                trigger OnAction()

                var

                    purchhead: Record PurchaseHeader;

                    sumamt: Decimal;

                begin

                    sumamt := 0;

                    purchhead.Reset();

                    if purchhead.FindSet() then

                        repeat

                            purchhead.CalcFields("Total Amount");

                            sumamt += purchhead."Total Amount";

                        until purchhead.Next() = 0;

                    Message(Format(sumamt));

                end;

            }

*Figure 6: Actions in Header Card*

**Steps:**

* Create the actions in the header card.
* Write the required logic and run.

## 5.Remote Connection Setup

We also learnt how to setup a remote connection using mstsc.

Graphical user interface, text, application

Description automatically generated

*Figure 7: Run mstsc*

Graphical user interface, text, application

Description automatically generated

*Figure 8: Remote Desktop Connection Window*

**Steps:**

* Open the Run Window using Win+R.
* Input the mstsc command in
* Input the IP of the computer needed in the remote desktop connection window.

## 6.API

We also learnt how to post and get data from a table or a function using API in Navision.

## 7. Adding Boolean in Insert or modify fields

We also learned how to run a code upon inserting, validating or modifying data using the Boolean value in the function.

 PurchLine.SetRange("Document No.", "No.");

                PurchLine.Reset();

                if PurchLine.FindSet() then

                    repeat

                        PurchLine."Vendor Name" := "Vendor Name";

                        PurchLine.Modify(true);

                    until PurchLine.Next() = 0;

*Figure 9: Adding Boolean Value in Modify*

trigger OnModify()

    begin

        Message('Vendor Selected');

    end;

*Figure 10: Adding code for modify*

Graphical user interface, application, Word

Description automatically generated

*Figure 11: Message displayed*

**Steps:**

* Add ‘True’ into the specified function.
* Write code into whichever trigger the function is relevant on.