Maksimilien DEUTSCH Mathis PAULY Martin GOULET
Pierre TOUET-SI-ABDELHADI Romain CUCHE

ENGINEERING SCHOOL
Creating the future together

Guide Fonctionnement VBA/SQL
2025-2026
Artisan De Confiance
S5 P2028

Maksimilien DEUTSCH Mathis PAULY Martin GOULET
Pierre TOUET-SI-ABDELHADI Romain CUCHE

ENGINEERING SCHOOL
Creating the future together

Sommaire

I. Gestion des artisans et des fournisseurs....2
a. Ajout, modification ou suppression d'un artisan....2
1. Ajout....2
2. Modification....2
3. Suppression....3
b. Affichage d'une fiche artisan (avec toutes ses caractéristiques)....3
c. Affichage d'une fiche fournisseur (avec ses caractéristiques)....4
d. Bilan de stocks (affichage des stocks critiques) – pouvoir saisir le seuil....4
e. Le prix moyen d'un type de travaux donné sur l'ensemble des artisans de la région proposant ce service....5
f. Les artisans déjà affectés sur des travaux (et qui ne sont donc pas disponibles)....6
II. Gestion des commandes clients....7
a. Création d'un compte client qui permet de visualiser ses informations....7
b. Affichage d'une fiche client (avec ses caractéristiques principales) associée avec son historique des commandes et les factures associées. La recherche doit pouvoir se faire sur le nom du client ou sur la première lettre du nom....7
c. La liste des commandes pour un ou plusieurs statut(s) de commande donné(s) avec pour chaque commande le nom des clients associés....8
d. Les commandes avec la réduction de 10% (par rapport aux critères énoncés dans la description de l'entreprise), avec le nom du client....9
e. Définir un(des) indicateur(s) de qualité des délais....10
f. Définir un(des) indicateur(s) de qualité des commandes....10
g. Le nombre de commandes en cours devant se terminer dans le mois....11
h. La liste des commandes en cours avec un indice de confiance sur le respect des délais et la qualité des commandes....11
III. Gestion commerciale....12
a. Bilan des ventes annuelles (chiffre d'affaires) et comparaison de l'année n-1 pour pouvoir faire des prévisions pour les années futures....12
b. Bilan des ventes (chiffre d'affaires) par catégories de travaux afin d'adapter le catalogue du site....13
c. Le nombre de commandes terminées par artisan pour les artisans d'une région donnée pour l'année en cours....13
d. Le bilan des ventes (chiffre d'affaires) pour les travaux d'urgence afin d'en déterminer la part commerciale....14
e. Affichage des travaux en retard pour pouvoir réagir s'il y a de trop nombreux retards 15
f. Mise en place de statistiques sur les causes des retards et sur la satisfaction des clients afin de pouvoir améliorer le service aux clients....15

Maksimilien DEUTSCH Mathis PAULY Martin GOULET
Pierre TOUET-SI-ABDELHADI Romain CUCHE

ENGINEERING SCHOOL
Creating the future together

I. Gestion des artisans et des fournisseurs

a. Ajout, modification ou suppression d'un artisan

1. Ajout

```
Private Sub RetourAjouterAButton_Click()
AjouterAUserform.Hide
GestionAFUserForm.Show
End Sub
```

Lorsqu'on clique sur le bouton retour, on cache la UserForm actuellement affichée et on ouvre le menu principal, menu avec lequel on à affiché cette UserForm.

```
Dim artisan As Worksheet
Dim lastRow As Long
Dim lastID As String ' car
Dim newNum As Long
```

Déclaration des variables :

Pour ma part je prends l'habitude de déclarer mes WorkSheet avec le nom de la feuille Excel pour plus de clarté, avec la première lettre en minuscule.

lastID est le dernier Identifiant qu'on à créé.

et lastRow est la première ligne vide de notre feuille.

```
If NomTextBox.Text = "" Or _
    CodePostalTextBox.Text = "" Or _
    TelTextBox.Text = "" Or _
    EmailTextBox.Text = "" Or _
    TarifTextBox.Text = "" Then
    MsgBox "Veuillez remplir tous les champs avant de valider."
    Exit Sub
End If
```

On vérifie que les TextBox ne sont pas vides, l'utilisateur doit remplir tous les champs pour ajouter un artisan. Le Exit Sub sert à arrêter le programme pour ne pas ajouter l'artisan si les colonnes sont vides

```
Set artisan = ThisWorkbook.Sheets("Artisan")
lastRow = artisan.Cells(artisan.Rows.Count, "A").End(xlUp).Row + 1
lastID = artisan.Cells(artisan.Rows.Count, "A").End(xlUp).Value
newNum = CLng(Mid(lastID, 3)) + 1  ' On prend "AR06" a partir du troisème
artisan.Cells(lastRow, 1).Value = "AR" & Format(newNum, "00")
```

Initialisation des variables :

On commence par initialiser la feuille Artisan. Ensuite pour LastRow on se place tout en bas de la colonne A, c'est-à-dire à la ligne 10^6 environ selon la version de notre Excel, et on remonte jusqu'à la dernière cellule remplie. On ajoute +1 pour aller à la première cellule vide. Pour lastID on récupère le dernier ID_Artisan existant. Pour newNum on extrait la partie numérique de l'ID (pour AR01 on extrait 01), ensuite on le formate en nombre (01 devient 1) et on ajoute 1 pour créer un identifiant unique.

```
artisan.Cells(lastRow, 1).Value = "AR" & Format(newNum, "00")
artisan.Cells(lastRow, 2).Value = NomTextBox.Text
artisan.Cells(lastRow, 3).Value = CodePostalTextBox.Text
artisan.Cells(lastRow, 4).Value = TelTextBox.Text
artisan.Cells(lastRow, 5).Value = EmailTextBox.Text
artisan.Cells(lastRow, 6).Value = TarifTextBox.Text
```

On ajoute les valeurs récupérées dans nos TextBox et on les ajoute dans notre nouvelle ligne.

```
MsgBox "Artisan ajouté !"
NomTextBox.Text = ""
CodePostalTextBox.Text = ""
TelTextBox.Text = ""
EmailTextBox.Text = ""
TarifTextBox.Text = ""
End Sub
```

Un pop up s'affiche pour nous dire si l'artisan à été ajouté ou non, et on vide les champs, comme cela on peut ajouter un deuxième artisan sans avoir à effacer les valeurs dans les TextBox. Ensuite le programme s'arrête.

2. Modification

```
Private Sub ModifierAButton_Click()
ModifierAUserForm.Hide
GestionAFUserForm.Show
End Sub
```

Pour le bouton retour c'est exactement la même méthode à chaque fois, on cache la UserForm actuelle et on affiche la précédente.

```
Dim artisan As Worksheet
Dim id As String
Dim cell As Range
```

Déclaration des variables :

id c'est l'identifiant de l'artisan à modifier
artisan la WorkSheet de la feuille Artisan.
cell c'est la cellule où se trouve l'id dans la feuille.

```
If id = "" Then
    MsgBox "Veuillez entrer l'ID du client à modifier."
    Exit Sub
End If
If NomTextBox.Text = "" Or _
    CodePostalTextBox.Text = "" Or _
    TelTextBox.Text = "" Or _
    EmailTextBox.Text = "" Or _
    TarifTextBox.Text = "" Then
    MsgBox "Veuillez remplir tous les champs."
    Exit Sub
End If
```

On vérifie qu'aucun champ n'est vide.

```
Set cell = artisan.Columns("A").Find(What:=id, LookIn:=xlValues, LookAt:=xlWhole)
```

On cherche la cellule avec comme valeur l'ID saisit dans la textBox.

```
If Not cell Is Nothing Then
    artisan.Cells(cell.Row, 2).Value = NomTextBox.Text
    artisan.Cells(cell.Row, 3).Value = CodePostalTextBox.Text
    artisan.Cells(cell.Row, 4).Value = TelTextBox.Text
    artisan.Cells(cell.Row, 5).Value = EmailTextBox.Text
    artisan.Cells(cell.Row, 6).Value = TarifTextBox.Text
    MsgBox "Client modifié avec succès !"
    IdTextBox.Text = ""
    NomTextBox.Text = ""
    CodePostalTextBox.Text = ""
    TelTextBox.Text = ""
    EmailTextBox.Text = ""
    TarifTextBox.Text = ""
Else
```

La boucle s'exécute si l'ID est trouvé, ainsi on modifie toute la ligne avec les nouvelles informations et on vide les textBox

```
Else
    MsgBox "ID non trouvé, car inexistant"
End If
End Sub
```

Sinon, l'identifiant n'est pas trouvé, on affiche un pop up pour prévenir l'utilisateur.

3. Suppression

C'est exactement pareil que précédemment à une exception près, la boucle :

```
Set cell = artisan.Columns("A").Find(What:=id, LookIn:=xlValues, LookAt:=xlWhole)
If Not cell Is Nothing Then
    cell.EntireRow.Delete
    MsgBox "Artisan supprimé : " & id
    IdArtSupprTextBox.Text = ""
Else
    MsgBox "ID non trouvé : " & id
End If

End Sub
```

Si l'identifiant est trouvé, on supprime la ligne entière et on affiche un pop up de confirmation puis ensuite on vide la textBox.

Si l'identifiant n'est pas trouvé un affiche un pop up également.

b. Affichage d'une fiche artisan (avec toutes ses caractéristiques)

```
Private Sub FicheAButton_Click()
Dim approvisionnement As Worksheet, produit As Worksheet, artisan As Worksheet
Dim idProduit As String
Dim lastRowApprovisionnement As Long, i As Long
Dim idArtisan As String, stock As Variant
Dim cellArtisan As Range
Dim cellProduit As Range
Set approvisionnement = ThisWorkbook.Sheets("Approvisionnement")
Set produit = ThisWorkbook.Sheets("Produit")
Set artisan = ThisWorkbook.Sheets("Artisan")
idProduit = IdProduitTextBox.Text
```

Déclaration et initialisation des variables.

```
FicheArtisanListBox.Clear
```

On vide la listBox, car si on change de produit, les anciennes valeurs restent dans le tableau même si elles ne concordent pas avec l'ID produit.

```
If Not cellProduit Is Nothing Then
    stock = produit.Cells(cellProduit.Row, 2).Value
Else
    MsgBox "produit non trouvé."
End If
```

Si le produit existe, on récupère son stock associé.
Si il n'existe pas on affiche un pop up.

```
With FicheArtisanListBox
     .AddItem "ID_Artisan"
     .List(0, 1) = "Nom"
     .List(0, 2) = "Tel"
     .List(0, 3) = "Email"
     .List(0, 4) = "Tarif"
     .List(0, 5) = "Stock"
End With
```

On met les entêtes pour plus de clarté.

```
For i = 2 To lastRowApprovisionnement
    If approvisionnement.Cells(i, "G").Value = idProduit Then
        idArtisan = approvisionnement.Cells(i, "E").Value
        Set cellArtisan = artisan.Columns("A").Find(What:=idArtisan, LookIn:=xlValues, LookAt:=xlWhole)
        If Not cellArtisan Is Nothing Then
            FicheArtisanListBox.AddItem artisan.Cells(cellArtisan.Row, 1).Value    '
            FicheArtisanListBox.List(FicheArtisanListBox.ListCount - 1, 1) = artisan.Cells(cellArtisan.Row, 2).Value
            FicheArtisanListBox.List(FicheArtisanListBox.ListCount - 1, 2) = artisan.Cells(cellArtisan.Row, 4).Value
            FicheArtisanListBox.List(FicheArtisanListBox.ListCount - 1, 3) = artisan.Cells(cellArtisan.Row, 5).Value
            FicheArtisanListBox.List(FicheArtisanListBox.ListCount - 1, 4) = artisan.Cells(cellArtisan.Row, 6).Value
            FicheArtisanListBox.List(FicheArtisanListBox.ListCount - 1, 5) = stock
        End If
    End If
Next i
```

Premièrement on cherche les lignes où le produit correspond à l'ID_Porduit saisit.
Si c'est bon on récupère l'ID de l'artisan associé au produit et on va chercher ses infos dans la feuille Artisan.
Ensuite on affiche ses informations dans la ListBox.

```
If FicheArtisanListBox.ListCount = 0 Then
    MsgBox "Aucun artisan trouvé pour le produit"
End If
End Sub
```

Si aucun artisan n'est associé à ce produit, un pop up s'affiche.

c. Affichage d'une fiche fournisseur (avec ses caractéristiques)

```
Dim fournisseur As Worksheet, approvisionnement As Worksheet
Dim idFournisseur As String
Dim lastRowAppro As Long, i As Long
Dim cellFournisseur As Range


Set fournisseur = ThisWorkbook.Sheets("Fournisseur")
Set approvisionnement = ThisWorkbook.Sheets("Approvisionnement")
idFournisseur = IdFournisseurTextBox.Text


If idFournisseur = "" Then
    MsgBox "Vous devez saisir un ID Fournisseur"
    Exit Sub
End If
FicheFournisseurListBox.Clear
```

Comme d'habitude on vérifie que la voleur dans la textBox n'est pas nulle, sinon on affiche un pop up.

```
Set cellFournisseur = fournisseur.Columns("A").Find(What:=idFournisseur, LookIn:=xlValues, LookAt:=xlWhole)
```

On va chercher dans la fiche fournisseur l'ID correspondant.

```
If cellFournisseur Is Nothing Then
    MsgBox "Fournisseur non trouvé."
    Exit Sub
End If
```

vérification de si on l'a trouvé ou non.

```
lastRowAppro = approvisionnement.Cells(approvisionnement.Rows.Count, "F").End(xlUp).Row
```

Dernière ligne de la feuille approvisionnement.

```
With FicheFournisseurListBox
    .AddItem "ID_Fournisseur"
    .List(0, 1) = "Nom"
    .List(0, 2) = "Ville"
    .List(0, 3) = "Tel"
    .List(0, 4) = "Email"
    .List(0, 5) = "ID_Approvisionnement"
End With
```

On remplit les entêtes.

```
For i = 2 To lastRowAppro
    If approvisionnement.Cells(i, "F").Value = idFournisseur Then
        FicheFournisseurListBox.AddItem fournisseur.Cells(cellFournisseur.Row, 1).Value
        FicheFournisseurListBox.List(FicheFournisseurListBox.ListCount - 1, 1) = fournisseur.Cells(cellFournisseur.Row, 2).Val
        FicheFournisseurListBox.List(FicheFournisseurListBox.ListCount - 1, 2) = fournisseur.Cells(cellFournisseur.Row, 3).Val
        FicheFournisseurListBox.List(FicheFournisseurListBox.ListCount - 1, 3) = fournisseur.Cells(cellFournisseur.Row, 4).Val
        FicheFournisseurListBox.List(FicheFournisseurListBox.ListCount - 1, 4) = fournisseur.Cells(cellFournisseur.Row, 5).Val
        FicheFournisseurListBox.List(FicheFournisseurListBox.ListCount - 1, 5) = approvisionnement.Cells(i, "A").Value
    End If
Next i
```

on regarde chaque ligne de la feuille Approvisionnement ou les approvisionnements sont reliés à l'identifiant du fournisseur entré dans la textBox et on ajoute dans la listBox.

```
If FicheFournisseurListBox.ListCount = 0 Then
    MsgBox "Aucun approvisionnement trouvé pour ce fournisseur."
End If
```

On informe l'utilisateur si son fournisseur n'a fait aucun approvisionnement.

d. Bilan de stocks (affichage des stocks critiques) – pouvoir saisir le seuil

```
Private Sub UserForm_Activate() '
PalierTextBox.Text = "10"
Call MettreAJourStocks
End Sub
```

On commence par remplir la TextBox par 10 qui est le palier par défaut. Ensuite on lance la procédure mettre à jour stocks (On utilise UserForm_Activate()).

```
Private Sub ModifierPalierButton_Click()
Call MettreAJourStocks
End Sub
```

Après avoir modifié le palier, lorsqu'on cliquer sur le bouton Modifier, on appelle la procédure.

```
Dim produit As Worksheet
Dim lastRow As Long, i As Long
Dim palier As Long
```

```
If PalierTextBox.Text = "" Or Not IsNumeric(PalierTextBox.Text) Then
    MsgBox "Veuillez saisir un palier valide."
    Exit Sub
End If
```

On vérifie que le palier n'est pas vide.

```
palier = CLng(PalierTextBox.Text)
```

Transforme le texte en entier.

```
Set produit = ThisWorkbook.Sheets("Produit")
StockListBox.Clear
StockListBox.ColumnCount = 3
StockListBox.ColumnWidths = "90;60;220"
StockListBox.AddItem "ID Produit"
StockListBox.List(0, 1) = "Stock"
StockListBox.List(0, 2) = "Libellé"
```

On paramètre les entêtes et la taille, on pouvait le faire depuis les propriétés mais on varie les méthodes !

```
lastRow = produit.Cells(produit.Rows.Count, "A").End(xlUp).Row
For i = 2 To lastRow
    If produit.Cells(i, "B").Value < palier Then
        StockListBox.AddItem produit.Cells(i, "A").Value
        StockListBox.List(StockListBox.ListCount - 1, 1) = produit.Cells(i, "B").Value
        StockListBox.List(StockListBox.ListCount - 1, 2) = produit.Cells(i, "C").Value
    End If
Next i
```

On ajoute seulement les produits dont le stock est inférieur au palier .

e. Le prix moyen d'un type de travaux donné sur l'ensemble des artisans de la région proposant ce service

```
Private Sub UserForm_Activate()
Dim typeTravaux As Worksheet
Dim lastRow As Long
Dim i As Long
Set typeTravaux = ThisWorkbook.Sheets("Type Travaux")
TypeTravauxComboBox.Clear
With PrixMoyenListBox
    .AddItem "Libellé Travaux"
    .List(0, 1) = "Prix moyen"
End With
lastRow = typeTravaux.Cells(typeTravaux.Rows.Count, "A").End(xlUp).Row
For i = 2 To lastRow
TypeTravauxComboBox.AddItem typeTravaux.Cells(i, "B").Value
Next i
End Sub
```

On paramètre la comboBox.

```
Private Sub ValiderButton_Click()
Dim artisan As Worksheet, devis As Worksheet, typeTravaux As Worksheet
Dim departement As String, libelleTravaux As String, idType As String
Dim total As Double, nbDevis As Long
Dim i As Long, codePostal As Variant
Dim cell As Range


departement = PrixMoyenTextBox.Text
libelleTravaux = TypeTravauxComboBox.Value
If departement = "" Then
    MsgBox "Veuillez saisir un département."
    Exit Sub
End If
```

On initialise le numéro de département et le type de travaux avec les valeurs dans la textBox et dans la comboBox. On vérifie si elles ne sont pas nulles.

```
Set artisan = Sheets("Artisan")
Set devis = Sheets("Devis")
Set typeTravaux = Sheets("Type Travaux")
Set cell = typeTravaux.Columns("B").Find(libelleTravaux, LookIn:=xlValues, LookAt:=xlWhole)
```

On cherche le libellé travaux dans la feuille Type Travaux

```
If cell Is Nothing Then
    MsgBox "Type de travaux introuvable."
    Exit Sub
End If
```

On vérifie si le type de travaux existe bien.

```
idType = typeTravaux.Cells(cell.Row, "A").Value
```

On a le libellé mais on récupère le Type travaux, qui lui est relié à d'autres feuilles.

```
total = 0
nbDevis = 0
```

On initialise de total et le nombre de devis, car moyenne = total/nbDevis.

```
For i = 2 To devis.Cells(devis.Rows.Count, "F").End(xlUp).Row
    codePostal = Application.VLookup(devis.Cells(i, "F").Value, artisan.Range("A:C"), 3, False)
```

A partir de l'ID_Artisant présent dans la feuille Devis, on récupère son code postal dans la feuille Artisan.

```
If Not IsError(codePostal) Then
    If Left(codePostal, 2) = departement And devis.Cells(i, "D").Value = idType Then
        total = total + devis.Cells(i, "C").Value
        nbDevis = nbDevis + 1
    End If
End If
```

On vérifie que l'artisan existe, si oui on vérifie également si l'artisan appartient au bon code postal et si les devis correspondent au type de travaux.

SI oui on calcul le total,en parcourant la boucle on ajoute le montant du devis à chaque itération.
On ajoute + 1 à nb Devis également si le if est respecté.

```
PrixMoyenListBox.Clear
With PrixMoyenListBox
    .AddItem "Libellé Travaux"
    .List(0, 1) = "Prix moyen"
End With
```

On clear et on ajoute les entêtes.

```
PrixMoyenListBox.AddItem libelleTravaux
If nbDevis > 0 Then
    PrixMoyenListBox.List(l, 1) = Format(total / nbDevis, "0.00")
Else
    PrixMoyenListBox.List(l, 1) = "Aucun devis"
End If
End Sub
```

Si il y a au moins 1 devis, on affiche le prix moyen, sinon on met dans la listBox : aucun devis.

f. Les artisans déjà affectés sur des travaux (et qui ne sont donc pas disponibles)

```
Private Sub UserForm_Activate()
Dim artisan As Worksheet, disponibilité As Worksheet
Dim lastRowDispo As Long
Dim i As Long, idArtisan As String
Dim nomArtisan As Variant
Set artisan = ThisWorkbook.Sheets("Artisan")
Set disponibilité = ThisWorkbook.Sheets("Disponibilité")
IndispoListBox.Clear
IndispoListBox.ColumnCount = 1
```

Déclaration des variables + initialisation. Et on clear la listBox au cas où.

```
IndispoListBox.AddItem "Artisans non disponibles"
```

On crée l'entête.

```
lastRowDispo = disponibilité.Cells(disponibilité.Rows.Count, "B").End(xlUp).Row
```

dernière ligne de la feuille Disponibilité.

```
For i = 2 To lastRowDispo
    If Trim(UCase(disponibilité.Cells(i, "A").Value)) = "NON" Then
        idArtisan = disponibilité.Cells(i, "B").Value
        nomArtisan = Application.VLookup(idArtisan, artisan.Range("A:B"), 2, False)
        If Not IsError(nomArtisan) Then
            IndispoListBox.AddItem nomArtisan
        End If
    End If
Next i
End Sub
```

On parcours chaque ligne, si la valeur de la cellule dans la colonne B est "NON" :

On commence par récupérer l'identifiant de l'artisan dans la feuille Disponibilité, ensuite on va chercher son nom dans la Feuille Artisan à l'aide de la formule VLookUp.

Et enfin on l'ajoute dans la liste.

II. Gestion des commandes clients

a. Création d'un compte client qui permet de visualiser ses informations

Maksimilien DEUTSCH Mathis PAULY Martin GOULET
Pierre TOUET-SI-ABDELHADI Romain CUCHE

Explication du fonctionnement de la procédure VBA “ValiderButton_Click”. Cette procédure est déclenchée lorsqu’on clique sur le bouton « Valider » dans un formulaire (UserForm) dédié à la création d’un nouveau client. Elle permet d’ajouter un client dans la feuille Excel nommée "Client" .

Le code commence par pointer vers la feuille Excel "Client" où sont stockés tous les clients (une ligne par client).

```
Dim client As Worksheet
Set client = ThisWorkbook.Sheets("Client")
```

Il contrôle que les six zones de texte du formulaire (Nom, Prénom, Email, Mot de passe, Code Postal, Téléphone) ne sont pas vides. Si l’un des champs manque, un message d’erreur s’affiche (« Tous les champs sont obligatoires ! ») et l’ajout est annulé.

```
If Trim(TextBox1.Text) = "" Or _
   Trim(TextBox2.Text) = "" Or _
   Trim(TextBox3.Text) = "" Or _
   Trim(TextBox4.Text) = "" Or _
   Trim(TextBox5.Text) = "" Or _
   Trim(TextBox6.Text) = "" Then
   MsgBox "Tous les champs sont obligatoires !"
   Exit Sub
End If
```

Le code trouve la première ligne vide en bas de la liste existante (dans la colonne B, généralement celle du Nom) pour y insérer le nouveau client sans écraser les données précédentes.

```
derligne = client.Cells(client.Rows.Count, "B").End(xlUp).Row + 1
```

Si la feuille est vide (premier client), l’ID créé est "C1". Sinon, il récupère le dernier ID existant (ex. : "C10"), augmente le numéro de 1 et crée le nouvel ID (ex. : "C11"). Cela garantit que chaque client a un identifiant unique au format "Cxx".

```
If derligne = 2 And client.Cells(2, "A").Value = "" Then
        client.Cells(derligne, "A").Value = "C1"
Else: derID = client.Cells(client.Rows.Count, "A").End(xlUp).Value
        newID = CLng(Mid(derID, 2)) + 1
        client.Cells(derligne, "A").Value = "C" & newID
End If
```

Les informations saisies dans les zones de texte sont copiées dans les colonnes correspondantes de la nouvelle ligne :

- Colonne A : ID généré automatiquement
- Colonne B : Nom
- Colonne C : Prénom
- Colonne D : Email
- Colonne E : Mot de passe
- Colonne F : Code Postal
- Colonne G : Téléphone

- Colonne H : Statut fidélité ("Oui" ou "Non", selon le bouton radio coché)

```
With client
        .Cells(derligne, 2).Value = Trim(TextBox1.Text)
        .Cells(derligne, 3).Value = Trim(TextBox2.Text)
        .Cells(derligne, 4).Value = Trim(TextBox3.Text)
        .Cells(derligne, 5).Value = Trim(TextBox4.Text)
        .Cells(derligne, 6).Value = Trim(TextBox5.Text)
        .Cells(derligne, 7).Value = Trim(TextBox6.Text)
If OptionButton1.Value = True Then
            .Cells(derligne, 8).Value = OptionButton1.Caption
ElseIf OptionButton2.Value = True Then
            .Cells(derligne, 8).Value = OptionButton2.Caption
Else
            .Cells(derligne, 8).Value = ""
End If
End With
```

Une zone de liste (ListBox) sur le formulaire est remplie avec un résumé clair des données juste ajoutées (ex. : "Nom : Paris", "Fidélité : Oui"). Cela permet à l'utilisateur de vérifier visuellement que tout est correct.

```
With ListBox1
        .AddItem "FICHE CLIENT AJOUTÉE"
        .AddItem "Nom         : " & Trim(TextBox1.Text)
        .AddItem "Prénom      : " & Trim(TextBox2.Text)
        .AddItem "Email       : " & Trim(TextBox3.Text)
        .AddItem "Mot de passe : " & Trim(TextBox4.Text)
        .AddItem "Code Postal: " & Trim(TextBox5.Text)
        .AddItem "Téléphone   : " & Trim(TextBox6.Text)
        .AddItem "Fidélité        : " & client.Cells(derligne, 8).Value
End With
```

Tous les champs texte sont vidés, les boutons radio sont décochés, et le curseur revient sur le premier champ (Nom), prêt pour ajouter un nouveau client.

```
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""
TextBox5.Text = ""
TextBox6.Text = ""
OptionButton1.Value = False
OptionButton2.Value = False
TextBox1.SetFocus
```

À la fin, une boîte de message indique « Client ajouté avec succès ! », confirmant que l'opération s'est bien déroulée.

```
MsgBox "Client ajouté avec succès !"
```

b. Affichage d'une fiche client (avec ses caractéristiques principales) associée avec son historique des commandes et les factures associées. La recherche doit pouvoir se faire sur le nom du client ou sur la première lettre du nom

Le code pointe vers les quatre feuilles Excel contenant les données : "Client", "Commande", "Facture" et "Statut Commande".

```
Dim client As Worksheet
Set client = ThisWorkbook.Sheets("Client")
Dim commande As Worksheet
Set commande = ThisWorkbook.Sheets("Commande")
Dim facture As Worksheet
Set facture = ThisWorkbook.Sheets("Facture")
Dim statutCmd As Worksheet
Set statutCmd = ThisWorkbook.Sheets("Statut Commande")
```

Le code récupère le texte saisi dans la zone TextBox1 et vérifie qu'il n'est pas vide. Sinon, un message d'erreur s'affiche et la recherche s'arrête.

```
NomSaisi = Trim(TextBox1.Text)
If NomSaisi = "" Then
   MsgBox "Entre au moins le nom ou la première lettre du nom !"
   Exit Sub
End If
```

Il extrait la première lettre (en majuscule) pour la recherche partielle et vide la ListBox qui affichera les résultats.

```
PremiereLettre = UCase(Left(NomSaisi, 1))
ListBox1.Clear
```

Le code parcourt toutes les lignes de la feuille "Client" et recherche un client dont :

- le nom complet correspond exactement,
- ou la première lettre du nom correspond

```
For i = 2 To client.Cells(Rows.Count, "B").End(xlUp).Row
    If UCase(Trim(client.Cells(i, "B").Value)) = UCase(NomSaisi) Or _
       UCase(Trim(client.Cells(i, "C").Value)) = UCase(NomSaisi) Or _
       UCase(Left(Trim(client.Cells(i, "B").Value), 1)) = UCase(Left(NomSaisi, 1)) Then
```

Si un client est trouvé, ses données (nom, prénom, email, etc.) sont ajoutées dans la ListBox pour constituer la fiche client.

```
With ListBox1
          .AddItem "FICHE CLIENT TROUVÉE"
          .AddItem "Nom         : " & client.Cells(i, "B").Value
          .AddItem "Prénom      : " & client.Cells(i, "C").Value
          .AddItem "Email       : " & client.Cells(i, "D").Value
          .AddItem "Mot de passe: " & client.Cells(i, "E").Value
          .AddItem "Code Postal : " & client.Cells(i, "F").Value
          .AddItem "Téléphone   : " & client.Cells(i, "G").Value
          .AddItem "Fidélité    : " & client.Cells(i, "H").Value
          .AddItem ""

End With
```

Pour le client trouvé (via son ID), le code parcourt la feuille "Commande". Pour chaque commande correspondante :

- Il récupère le libellé du statut en consultant la feuille "Statut Commande".
- Il ajoute les informations de la commande (date, montant, statut) dans la ListBox.

```
For j = 2 To commande.Cells(Rows.Count, "B").End(xlUp).Row
          If commande.Cells(j, "B").Value = ID_Client Then
              AucuneCommande = False
              ID_Statut = commande.Cells(j, "I").Value
              LibelleStatut = "Inconnu"
              For m = 2 To statutCmd.Cells(Rows.Count, "A").End(xlUp).Row
                  If statutCmd.Cells(m, "A").Value = ID_Statut Then
                      LibelleStatut = statutCmd.Cells(m, "B").Value
                      Exit For
                  End If
              Next m

              With ListBox1
                  .AddItem " HISTORIQUE DES COMMANDES "
                  .AddItem " Date : " & commande.Cells(j, "C").Value
                  .AddItem " Montant HT : " & commande.Cells(j, "E").Value & " $ "
                  .AddItem " Statut : " & LibelleStatut
                  .AddItem ""
              End With
```

Pour chaque commande trouvée, le code recherche dans la feuille "Facture" les lignes correspondant à l'ID de la commande et ajoute les détails (date, montant TTC, paiement).

```
For k = 2 To facture.Cells(Rows.Count, "B").End(xlUp).Row
    If facture.Cells(k, "B").Value = ID_Commande Then
        With ListBox1
            .AddItem " FACTURE"
            .AddItem " Date : " & facture.Cells(k, "C").Value
            .AddItem " Montant TTC : " & facture.Cells(k, "D").Value & " $ "
            .AddItem " Payée : " & facture.Cells(k, "E").Value
        End With
    End If
Next k
```

Si le client n'a aucune commande → message "Aucune commande enregistrée".

Si aucun client ne correspond à la recherche → message "Aucun client trouvé" avec conseil de vérification.

```
          If AucuneCommande Then
              ListBox1.AddItem "Aucune commande enregistrée"
              ListBox1.AddItem ""
          End If
      End If
  Next i

  If Not Trouve Then
      With ListBox1
          .AddItem "Aucun client trouvé"
          .AddItem ""
          .AddItem "Vérifie le nom saisi"
      End With
  End If
```

c. La liste des commandes pour un ou plusieurs statut(s) de commande donné(s) avec pour chaque commande le nom des clients associés

Le code pointe vers les trois feuilles Excel nécessaires : "Commande" (liste des commandes), "Client" (informations clients) et "Statut Commande" (table de correspondance

entre ID statut et libellé).

```
Dim commande As Worksheet
Set commande = ThisWorkbook.Sheets("Commande")
Dim client As Worksheet
Set client = ThisWorkbook.Sheets("Client")
Dim statutCmd As Worksheet
Set statutCmd = ThisWorkbook.Sheets("Statut Commande")
```

Le code vérifie si au moins une des quatre cases à cocher (CheckBox1 à CheckBox4, correspondant aux différents statuts) est sélectionnée. Sinon, un message d'erreur s'affiche et la procédure s'arrête.

```
AuMoinsUnCoche = CheckBox1.Value Or CheckBox2.Value Or CheckBox3.Value Or CheckBox4.Value
If Not AuMoinsUnCoche Then
    MsgBox "Coche au moins un statut !"
    Exit Sub
End If
```

La ListBox est vidée et un titre général est ajouté.

```
ListBox1.Clear
ListBox1.AddItem "COMMANDES SÉLECTIONNÉES"
```

Le code examine chaque ligne de la feuille "Commande". Pour chaque commande, il récupère l'ID du statut.

```
For j = 2 To commande.Cells(Rows.Count, "A").End(xlUp).Row
    ID_Statut = Trim(commande.Cells(j, "I").Value)
```

En consultant la feuille "Statut Commande", le code traduit l'ID du statut en texte lisible (ex. : "Terminée", "En cours", "En retard", "Devis accepté").

```
For m = 2 To statutCmd.Cells(Rows.Count, "A").End(xlUp).Row
    If Trim(statutCmd.Cells(m, "A").Value) = ID_Statut Then
        LibelleStatut = Trim(statutCmd.Cells(m, "B").Value)
        Exit For
    End If
Next m
```

La commande n'est affichée que si son libellé correspond à l'un des statuts cochés dans les CheckBox (CheckBox1 = Terminée, CheckBox2 = En cours, etc.).

```
If LibelleStatut = "Terminée" And CheckBox1.Value Then AfficherCetteCommande = True
If LibelleStatut = "En cours" And CheckBox2.Value Then AfficherCetteCommande = True
If LibelleStatut = "En retard" And CheckBox3.Value Then AfficherCetteCommande = True
If LibelleStatut = "Devis accepté" And CheckBox4.Value Then AfficherCetteCommande = True
If AfficherCetteCommande Then
```

Si la commande doit être affichée, le code recherche dans la feuille "Client" le nom et prénom correspondant à l'ID client de la commande.

```
ID_Client = commande.Cells(j, "B").Value
NomClient = ""
PrenomClient = ""
Dim i As Long
For i = 2 To client.Cells(Rows.Count, "A").End(xlUp).Row
    If client.Cells(i, "A").Value = ID_Client Then
        NomClient = client.Cells(i, "B").Value
        PrenomClient = client.Cells(i, "C").Value
        Exit For
    End If
Next i
```

Une ligne complète est ajoutée avec : numéro de commande, nom/prénom du client, date, montant et statut.

```
ListBox1.AddItem "Commande : " & commande.Cells(j, "A").Value & _
                 " | Client : " & NomClient & " " & PrenomClient & _
                 " | Date : " & commande.Cells(j, "C").Value & _
                 " | Montant : " & commande.Cells(j, "E").Value & " $ " & _
                 " | Statut : " & LibelleStatut
```

Si aucune commande ne satisfait les critères, un message informatif est affiché dans la ListBox.

```
If Not Trouve Then
    ListBox1.AddItem "Aucune commande ne correspond aux statuts sélectionnés."
End If
```

d. Les commandes avec la réduction de 10% (par rapport aux critères énoncés dans la description de l'entreprise), avec le nom du client

Le code pointe vers les deux feuilles Excel nécessaires : "Commande" (liste des commandes) et "Client" (informations clients).

```
Dim commande As Worksheet
Set commande = ThisWorkbook.Sheets("Commande")
Dim client As Worksheet
Set client = ThisWorkbook.Sheets("Client")
```

Le code examine chaque ligne de la feuille "Commande" (à partir de la ligne 2).

```
For j = 2 To commande.Cells(Rows.Count, "A").End(xlUp).Row
```

La réduction est stockée dans la colonne G de la feuille "Commande" sous forme de valeur décimale (0.1 pour 10 %). Le code ne retient que les commandes où cette valeur est exactement 0.1.

```
If commande.Cells(j, "G").Value = 0.1 Then
```

Pour chaque commande éligible, le code récupère l'ID client (colonne B), puis recherche dans la feuille "Client" le nom et prénom correspondant. Si le client n'est pas trouvé, il affiche "Inconnu".

```
ID_Client = Trim(commande.Cells(j, "B").Value)
NomClient = "Inconnu"
PrenomClient = ""
For i = 2 To client.Cells(Rows.Count, "A").End(xlUp).Row
    If Trim(client.Cells(i, "A").Value) = ID_Client Then
        NomClient = Trim(client.Cells(i, "B").Value)
        PrenomClient = Trim(client.Cells(i, "C").Value)
        Exit For
    End If
Next i
```

Une ligne complète est ajoutée avec : numéro de commande, nom/prénom du client, date, montant HT et mention de la réduction de 10 %.

```
ListBox1.AddItem "Commande : " & commande.Cells(j, "A").Value & _
                 " | Client : " & NomClient & " " & PrenomClient & _
                 " | Date : " & commande.Cells(j, "C").Value & _
                 " | Montant HT : " & commande.Cells(j, "D").Value & " $ " & _
                 " | Réduction : 10%"
```

Si aucune commande ne satisfait le critère, un message informatif est affiché dans la ListBox.

```
If Not Trouve Then
    ListBox1.AddItem "Aucune commande avec réduction de 10% trouvée."
End If
```

e. Définir un(des) indicateur(s) de qualité des délais

Le code pointe vers les deux feuilles Excel nécessaires : "Commande" (liste des commandes) et "Statut Commande" (table de correspondance entre ID statut et libellé).

```
Dim commande As Worksheet
Set commande = ThisWorkbook.Sheets("Commande")
Dim statutCmd As Worksheet
Set statutCmd = ThisWorkbook.Sheets("Statut Commande")
```

Plusieurs variables sont mises à zéro pour compter : le total des commandes, les terminées, celles à temps, en retard et en cours.

```
TotalCmd = 0
Terminees = 0
A_Temps = 0
EnRetard = 0
EnCours = 0
```

Le code examine chaque ligne de la feuille "Commande" (à partir de la ligne 2) et incrémente le compteur total à chaque fois.

```
For j = 2 To commande.Cells(Rows.Count, "A").End(xlUp).Row
    TotalCmd = TotalCmd + 1
```

Pour chaque commande, le code récupère l'ID du statut (colonne I) et traduit cet ID en texte lisible (ex. : "en cours", "terminée") en consultant la feuille "Statut Commande".

```
ID_Statut = Trim(commande.Cells(j, "I").Value)
Dim LibelleStatut As String
LibelleStatut = ""
For m = 2 To statutCmd.Cells(Rows.Count, "A").End(xlUp).Row
    If Trim(statutCmd.Cells(m, "A").Value) = ID_Statut Then
        LibelleStatut = Trim(statutCmd.Cells(m, "B").Value)
        Exit For
    End If
Next m
```

Si le statut est "en cours" → incrémente le compteur des commandes en cours.
Si le statut est "terminée" → incrémente le compteur des terminées, puis compare les dates (prévue en colonne D, réelle en colonne K) pour déterminer si la commande était à temps ou en retard.

```
If LCase(LibelleStatut) = "en cours" Then
    EnCours = EnCours + 1
ElseIf LCase(LibelleStatut) = "terminée" Then
    Terminees = Terminees + 1
    Dim DatePrevue As Date
    Dim DateReelle As Date
    If IsDate(commande.Cells(j, "D").Value) And IsDate(commande.Cells(j, "K").Value) Then
        DatePrevue = commande.Cells(j, "D").Value
        DateReelle = commande.Cells(j, "K").Value
        If DateReelle <= DatePrevue Then
            A_Temps = A_Temps + 1
        Else
            EnRetard = EnRetard + 1
        End If
    End If
End If
```

Si au moins une commande est terminée, le code calcule les pourcentages à temps et en retard (arrondis à une décimale). Sinon, il affiche 0 %.

```
If Terminees > 0 Then
    PourcA_Temps = Round((A_Temps / Terminees) * 100, 1) & " %"
    PourcEnRetard = Round((EnRetard / Terminees) * 100, 1) & " %"
Else
    PourcA_Temps = "0 %"
    PourcEnRetard = "0 %"
End If
```

Les résultats sont affichés dans quatre Labels du formulaire pour une lecture immédiate : pourcentage à temps, pourcentage en retard, total des commandes et nombre en cours.

```
Label1.Caption = "Commandes livrées à temps : " & PourcA_Temps
Label2.Caption = "Commandes en retard : " & PourcEnRetard
Label3.Caption = "Total des commandes : " & TotalCmd
Label4.Caption = "Commandes en cours : " & EnCours
```

f. Définir un(des) indicateur(s) de qualité des commandes

Le code pointe vers la feuille Excel "Commande" qui contient la liste complète des commandes.

```
Dim commande As Worksheet
Set commande = ThisWorkbook.Sheets("Commande")
```

Plusieurs variables sont mises à zéro pour accumuler : le nombre total de commandes, le nombre avec réduction, et le montant total HT de toutes les commandes.

```
TotalCmd = 0
CmdAvecReduction = 0
MontantTotal = 0
```

Le code examine chaque ligne de la feuille "Commande" (à partir de la ligne 2). Pour chaque commande, il incrémente le compteur total.

```
For j = 2 To commande.Cells(Rows.Count, "A").End(xlUp).Row
    TotalCmd = TotalCmd + 1
```

Si la valeur du montant HT (colonne E) est numérique, elle est ajoutée au total général.

```
If IsNumeric(commande.Cells(j, "E").Value) Then
    MontantTotal = MontantTotal + commande.Cells(j, "E").Value
End If
```

La réduction est stockée en colonne G sous forme décimale (0.1) ou parfois texte ("0,1"). Le code détecte ces deux formats pour compter les commandes éligibles.

```
If commande.Cells(j, "G").Value = 0.1 Or commande.Cells(j, "G").Value = "0,1" Then
    CmdAvecReduction = CmdAvecReduction + 1
End If
```

Si au moins une commande existe :

- Calcul du pourcentage de commandes avec réduction (arrondi à 1 décimale).
- Calcul du montant moyen par commande (arrondi à 2 décimales). Sinon, les valeurs sont mises à 0.

```
If TotalCmd > 0 Then
    PourcReduction = Round((CmdAvecReduction / TotalCmd) * 100, 1) & " %"
    MontantMoyen = Round(MontantTotal / TotalCmd, 2)
Else
    PourcReduction = "0 %"
    MontantMoyen = 0
End If
```

Les résultats sont affichés dans quatre Labels du formulaire avec un formatage clair (séparateurs de milliers et devise).

```
Label1.Caption = "Commandes avec réduction 10% : " & PourcReduction
Label2.Caption = "Montant total des commandes : " & Format(MontantTotal, "#,##0.00") & " $ "
Label3.Caption = "Nombre total de commandes : " & TotalCmd
Label4.Caption = "Montant moyen par commande : " & Format(MontantMoyen, "#,##0.00") & " $ "
```

g. Le nombre de commandes en cours devant se terminer dans le mois

Le code pointe vers les deux feuilles Excel nécessaires : "Commande" (liste des commandes) et "Statut Commande" (table de correspondance entre ID statut et libellé).

```
Dim commande As Worksheet
Set commande = ThisWorkbook.Sheets("Commande")
Dim statutCmd As Worksheet
Set statutCmd = ThisWorkbook.Sheets("Statut Commande")
```

Une variable est mise à zéro pour compter le nombre de commandes en cours à terminer ce mois.

```
Dim NbCommandesCeMois As Long
NbCommandesCeMois = 0
```

Le code examine chaque ligne de la feuille "Commande" (à partir de la ligne 2).

```
For j = 2 To commande.Cells(Rows.Count, "A").End(xlUp).Row
```

Pour chaque commande, le code récupère l'ID du statut (colonne I) et traduit cet ID en texte lisible (ex. : "en cours") en consultant la feuille "Statut Commande".

```
ID_Statut = Trim(commande.Cells(j, "I").Value)
LibelleStatut = ""
For m = 2 To statutCmd.Cells(Rows.Count, "A").End(xlUp).Row
    If Trim(statutCmd.Cells(m, "A").Value) = ID_Statut Then
        LibelleStatut = Trim(statutCmd.Cells(m, "B").Value)
        Exit For
    End If
Next m
```

Seules les commandes dont le statut est exactement "en cours" (insensible à la casse) sont prises en compte.

```
If LCase(LibelleStatut) = "en cours" Then
```

Si la date prévue de fin (colonne D) est une date valide, le code compare l'année et le mois de cette date à ceux de la date actuelle (fonction Date). Si ils correspondent (même mois et même année), le compteur est incrémenté.

```
If IsDate(commande.Cells(j, "D").Value) Then
    DatePrevue = commande.Cells(j, "D").Value

    If Year(DatePrevue) = Year(Date) And Month(DatePrevue) = Month(Date) Then
        NbCommandesCeMois = NbCommandesCeMois + 1
    End If
End If
```

À la fin du parcours, le nombre total est affiché dans un Label pour une lecture immédiate.

```
Label1.Caption = "Nombre de commandes en cours à terminer ce mois : " & NbCommandesCeMois
```

h. La liste des commandes en cours avec un indice de confiance sur le respect des délais et la qualité des commandes

Le code pointe vers les trois feuilles Excel nécessaires : "Commande" (liste des commandes), "Client" (informations clients) et "Statut Commande" (table de correspondance entre ID statut et libellé).

```
Dim commande As Worksheet
Set commande = ThisWorkbook.Sheets("Commande")
Dim client As Worksheet
Set client = ThisWorkbook.Sheets("Client")
Dim statutCmd As Worksheet
Set statutCmd = ThisWorkbook.Sheets("Statut Commande")
```

Le code examine chaque ligne de la feuille "Commande" (à partir de la ligne 2).

```
For j = 2 To commande.Cells(Rows.Count, "A").End(xlUp).Row
```

Pour chaque commande, le code récupère l'ID du statut (colonne I) et traduit cet ID en texte lisible (ex. : "en cours") en consultant la feuille "Statut Commande".

```
ID_Statut = Trim(commande.Cells(j, "I").Value)
LibelleStatut = ""
For m = 2 To statutCmd.Cells(Rows.Count, "A").End(xlUp).Row
    If Trim(statutCmd.Cells(m, "A").Value) = ID_Statut Then
        LibelleStatut = Trim(statutCmd.Cells(m, "B").Value)
        Exit For
    End If
Next m
```

Seules les commandes dont le statut est exactement "en cours" (insensible à la casse) sont prises en compte.

```
If LCase(LibelleStatut) = "en cours" Then
```

Calcul de l'indice de confiance

- Si la date prévue de fin (colonne D) est valide, le code calcule les jours restants jusqu'à cette date (DatePrevue - Date).

- L'indice est déterminé selon des règles simples :
  - "Élevé" si plus de 10 jours restants et réduction de 10 % appliquée (colonne G = 0.1).
  - "Moyen" si plus de 5 jours restants.
  - "Faible" sinon.
- Si la date est invalide → "Date inconnue".

```
If IsDate(commande.Cells(j, "D").Value) Then
    DatePrevue = commande.Cells(j, "D").Value
    JoursRestants = DatePrevue - Date
    Reduction = commande.Cells(j, "G").Value
    If JoursRestants > 10 And Reduction = 0.1 Then
        IndiceConfiance = "Élevé"
    ElseIf JoursRestants > 5 Then
        IndiceConfiance = "Moyen"
    Else
        IndiceConfiance = "Faible"
    End If
Else
    IndiceConfiance = "Date inconnue"
End If
```

Pour chaque commande en cours, le code récupère l'ID client (colonne B), puis recherche dans la feuille "Client" le nom et prénom correspondant. Si non trouvé, affiche "Inconnu".

```
ID_Client = Trim(commande.Cells(j, "B").Value)
NomClient = "Inconnu"
PrenomClient = ""
For i = 2 To client.Cells(Rows.Count, "A").End(xlUp).Row
    If Trim(client.Cells(i, "A").Value) = ID_Client Then
        NomClient = Trim(client.Cells(i, "B").Value)
        PrenomClient = Trim(client.Cells(i, "C").Value)
        Exit For
    End If
Next i
```

Une ligne complète est ajoutée avec : numéro de commande, nom/prénom du client, date prévue formatée, jours restants et indice de confiance.

```
ListBox1.AddItem "Commande : " & commande.Cells(j, "A").Value & _
                 " | Client : " & NomClient & " " & PrenomClient & _
                 " | Prévue : " & Format(DatePrevue, "dd/mm/yyyy") & _
                 " | Jours restants : " & JoursRestants & _
                 " | Confiance : " & IndiceConfiance
```

Un message informatif est affiché dans la ListBox si aucune commande ne correspond.

```
If Not Trouve Then
    ListBox1.AddItem "Aucune commande en cours trouvée."
End If
```

III. Gestion commerciale

Avant la présentation du des parties Gestion Commerciale, nous introduisons deux fonctions:

ChargerDonnees :

```
Sub ChargerDonnees()
    Dim ws As Worksheet
    On Error Resume Next
    Set ws = ThisWorkbook.Sheets("commande")
    On Error GoTo 0

    If ws Is Nothing Then
        MsgBox "ERREUR : Je ne trouve pas la feuille 'commande'.", vbCritical
        End
    End If

    Dim DerLigne As Long
    DerLigne = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
    If DerLigne < 2 Then Exit Sub
    DataCmd = ws.Range("A2:L" & DerLigne).Value
End Sub
```

Le code commence par définir la feuille de travail cible ("Commande"). Une sécurité est mise en place avec `On Error Resume Next` pour éviter que le programme ne plante si la feuille n'existe pas.

Ensuite, il identifie la dernière ligne remplie de la colonne A (`derLigne`). Si le tableau contient des données, l'intégralité de la plage (de A2 jusqu'à la dernière ligne de la colonne L) est chargée d'un seul coup dans la variable tableau `DataCmd`. Cela évite de lire les cellules une par une, ce qui serait beaucoup plus lent.

De plus, nous avions rencontré beaucoup de problèmes de chargement, ce pourquoi nous nous sommes documentés et accorder pour utiliser cette fonction.

ChargerRef :

```
Function ChargerRef(nomFeuille As String, colCle As String, colValeur As String) As Object
    Dim dict As Object
    Set dict = CreateObject("Scripting.Dictionary")
    Dim ws As Worksheet

    On Error Resume Next
    Set ws = ThisWorkbook.Sheets(nomFeuille)
    On Error GoTo 0

    If ws Is Nothing Then Set ChargerRef = dict: Exit Function

    Dim DerLigne As Long
    DerLigne = ws.Cells(ws.Rows.Count, colCle).End(xlUp).Row
    If DerLigne < 2 Then Set ChargerRef = dict: Exit Function

    Dim rCle As Variant, rVal As Variant
    rCle = ws.Range(colCle & "2:" & colCle & DerLigne).Value
    rVal = ws.Range(colValeur & "2:" & colValeur & DerLigne).Value

    Dim i As Long
    For i = 1 To UBound(rCle, 1)
        If Not IsEmpty(rCle(i, 1)) Then
            Dim k As String: k = UCase(Trim(CStr(rCle(i, 1))))
            If Not dict.Exists(k) Then dict.Add k, rVal(i, 1)
        End If
    Next i
    Set ChargerRef = dict
End Function
```

Le code charge les colonnes demandées en mémoire. Il parcourt ensuite chaque ligne :

Il nettoie la clé (suppression des espaces inutiles avec Trim et mise en majuscule avec UCase) pour éviter les erreurs de format, vérifie si la clé existe déjà dans le dictionnaire (If Not dict.Exists...). Si elle est unique, il l'ajoute au dictionnaire associée à sa valeur.

À la fin, la fonction renvoie l'objet dictionnaire rempli, prêt à être utilisé par les autres macros d'analyse.

Ces deux fonctions étaient essentielles pour le fonctionnement de notre code. Sans elles, nous n'arrivions pas à gérer les bugs de la 3ème partie, notamment l'affichage des graphiques avec les bonnes données.

a. Bilan des ventes annuelles (chiffre d'affaires) et comparaison de l'année n-1 pour pouvoir faire des prévisions pour les années futures

```
Public Sub BilanVentesAnnuelles()
    Dim wsData As Worksheet
    Dim dataArr As Variant
    Dim salesDict As Object
    Dim i As Long, lastRow As Long
    Dim dateReelle As Variant
    Dim montantCmd As Double
    Dim reducCmd As Double
    Dim anneeCmd As Long
    Dim netAmount As Double
    Dim anneeCible As Long
    Dim caN As Double, caN_1 As Double
    Dim variation As Double

    Set wsData = ThisWorkbook.Sheets("Commande")
    Set salesDict = CreateObject("Scripting.Dictionary")
    Application.ScreenUpdating = False
    lastRow = wsData.Cells(wsData.Rows.Count, "A").End(xlUp).Row
    If lastRow < 2 Then Exit Sub
    dataArr = wsData.Range("A2:J" & lastRow).Value
```

Le code commence par pointer vers la feuille Excel "Commande" contenant l'historique des ventes. Il initialise un dictionnaire(salesDict) qui servira à stocker le cumul du Chiffre d'Affaires (CA) pour chaque année trouvée.

Remarque : Ici, nous n'avons pas utiliser ChargerDonnee et ChargerRef, car nous n'avons pas rencontrer de bugs sur ce code.

```
  For i = LBound(dataArr, 1) To UBound(dataArr, 1)

      dateReelle = dataArr(i, 4)

      If IsDate(dateReelle) Then

          anneeCmd = Year(dateReelle)
          montantCmd = CDbl(dataArr(i, 5))

          If IsNumeric(dataArr(i, 7)) And Not IsEmpty(dataArr(i, 7)) Then
              reducCmd = CDbl(dataArr(i, 7))
          Else
              reducCmd = 0
          End If


          netAmount = montantCmd - reducCmd

          If salesDict.Exists(anneeCmd) Then
              salesDict(anneeCmd) = salesDict(anneeCmd) + netAmount
          Else
              salesDict.Add anneeCmd, netAmount
          End If
      End If
  Next i
```

Le code parcourt chaque ligne du tableau des commandes : Il vérifie si la **Date Réelle** est

valide pour extraire l'année, il récupère le **Montant Commande** et la **Réduction** éventuelle, il calcule le **Montant Net** (NetAmount = montantCmd - reducCmd).

SalesDict(AnneeCmd) représente le montant total du Chiffre d'Affaires cumulé pour cette année précise.

Si l'année existe déjà dans le dictionnaire, il ajoute le montant au total existant ; sinon, il crée une nouvelle entrée pour cette année.

```
    Dim userInput As String
    userInput = InputBox("Entrez l'année à analyser (ex: 2025) :", "Bilan Commercial", Year(Date))

    If Not IsNumeric(userInput) Then Exit Sub
    anneeCible = CLng(userInput)


    If salesDict.Exists(anneeCible) Then caN = salesDict(anneeCible) Else caN = 0
    If salesDict.Exists(anneeCible - 1) Then caN_1 = salesDict(anneeCible - 1) Else caN_1 = 0

    Dim strVariation As String
    If caN_1 <> 0 Then
        variation = (caN - caN_1) / caN_1
        strVariation = Format(variation, "+0.00%;-0.00%")
    Else
        strVariation = "N/A (N-1 vide)"
    End If

    MsgBox "Analyse des Ventes pour l'année " & anneeCible & vbCrLf & _
           "CA Annuel N   : " & Format(caN, "#,##0.00 €") & vbCrLf & _
           "CA Annuel N-1 : " & Format(caN_1, "#,##0.00 €") & vbCrLf & _
           "Variation     : " & strVariation, vbInformation, "KPI Commercial"

    Application.ScreenUpdating = True
End Sub
```

Une boîte de dialogue (InputBox) demande à l'utilisateur l'année cible à analyser (par défaut l'année en cours). Le code récupère ensuite le CA de l'année **N** et de l'année **N-1** dans le dictionnaire. Il calcule la variation en pourcentage et affiche le bilan complet dans une MsgBox.

---

b. Bilan des ventes (chiffre d'affaires) par catégories de travaux afin d'adapter le catalogue du site

```
' 3)b.
Sub BilanCategories()
    Call ChargerDonnees
    If IsEmpty(DataCmd) Then Exit Sub

    Dim dictType As Object: Set dictType = ChargerRef("Type Travaux", "A", "B")
    Dim dictRes As Object: Set dictRes = CreateObject("Scripting.Dictionary")

    Dim i As Long, code As String, lbl As String, net As Double
    For i = 1 To UBound(DataCmd, 1)
        If IsDate(DataCmd(i, 3)) Then
            code = UCase(Trim(DataCmd(i, 6)))
            If dictType.Exists(code) Then lbl = dictType(code) Else lbl = code
            net = CDbl(DataCmd(i, 5)) * (1 - CDbl(DataCmd(i, 7)))
            If dictRes.Exists(lbl) Then dictRes(lbl) = dictRes(lbl) + net Else dictRes.Add lbl, net
        End If
    Next i
```

Le code charge deux dictionnaires :

1. dictType : Pour faire la correspondance entre le code (ex: "T01") et le libellé complet (ex: "Peinture").
2. dictRes : Pour stocker le CA cumulé par catégorie.

```
    Application.DisplayAlerts = False: On Error Resume Next: Sheets("Analyse_Cat").Delete: On Error GoTo 0: Application.DisplayAlerts = True
    Dim ws As Worksheet: Set ws = Sheets.Add: ws.Name = "Analyse_Cat"
    ws.Range("A1:B1").Value = Array("Catégorie", "CA")
    Dim k, r As Integer: r = 2
    For Each k In dictRes.Keys
        ws.Cells(r, 1).Value = k: ws.Cells(r, 2).Value = dictRes(k): r = r + 1
    Next k
    ws.Shapes.AddChart2(251, xlPie).Chart.SetSourceData Source:=ws.Range("A1:B" & r - 1)
    MsgBox "Graphique créé!", vbInformation
End Sub
```

Il parcourt le tableau des commandes, calcule le montant net de chaque ligne, identifie la catégorie de travaux et additionne ce montant dans dictRes.

Le code crée une nouvelle feuille nommée "Analyse_Cat". Il y écrit un tableau à deux colonnes (Catégorie, CA) à partir des données calculées. Enfin, il génère automatiquement un **graphique Camembert** pour visualiser la répartition du chiffre d'affaires.

c. Le nombre de commandes terminées par artisan pour les artisans d'une région donnée pour l'année en cours

```
'3)c.
Sub CommandeParArtisan()
    Call ChargerDonnees
    If IsEmpty(DataCmd) Then Exit Sub
    ' Table Devis
    Dim dictDevis As Object
    Set dictDevis = ChargerRef("Devis", "A", "F")
    ' Table Artisan
    Dim dictArtisan As Object
    Set dictArtisan = ChargerRef("Artisan", "A", "B")

    If dictDevis.Count = 0 Or dictArtisan.Count = 0 Then
        MsgBox "Erreur : Vérifie que tes feuilles s'appellent bien 'Devis' et 'Artisan'.", vbCritical
        Exit Sub
    End If

    Dim dictCpt As Object: Set dictCpt = CreateObject("Scripting.Dictionary")
    Dim i As Long, idDev As String, idArt As String, nomArt As String
```

Le code charge les tables de correspondance nécessaires :

- La feuille "Devis" pour lier une commande à un ID Artisan.
- La feuille "Artisan" pour obtenir le Nom de l'artisan à partir de son ID.

Avec les problèmes précédents, nous avons ajouté un moyen de vérifier que les feuilles prises sont les bonnes, donc une vérification par un If dictDevis.Count.

```
    For i = 1 To UBound(DataCmd, 1)
        If IsDate(DataCmd(i, 3)) Then
            idDev = UCase(Trim(DataCmd(i, 8)))
            If dictDevis.Exists(idDev) Then
                idArt = UCase(Trim(dictDevis(idDev)))
                If dictArtisan.Exists(idArt) Then
                    nomArt = dictArtisan(idArt)
                Else
                    nomArt = "Artisan inconnu (" & idArt & ")"
                End If
            Else
                nomArt = "Devis inconnu (" & idDev & ")"
            End If

            If dictCpt.Exists(nomArt) Then dictCpt(nomArt) = dictCpt(nomArt) + 1 Else dictCpt.Add nomArt, 1
        End If
    Next i
```

Pour chaque commande, le script remonte la chaîne d'information : **Commande -> ID Devis -> ID Artisan -> Nom Artisan**. Il utilise un dictionnaire compteur (`dictCpt`) qui s'incrémente (+1) à chaque fois qu'une commande est attribuée à un artisan spécifique. Le

résultat final est affiché dans une boîte de messages listant chaque artisan(ayant été amené à travailler uniquement) et son volume de commandes.

d. Le bilan des ventes (chiffre d'affaires) pour les travaux d'urgence afin d'en déterminer la part commerciale

```
Sub AnalyseUrgences()
    Call ChargerDonnees

    If IsEmpty(DataCmd) Then Exit Sub

    If UBound(DataCmd, 2) < 8 Then
        MsgBox "Erreur : La plage de données chargée est trop petite (moins de 8 colonnes).", vbCritical
        Exit Sub
    End If
```

Ce sont des sécurités: on doit vérifier si le tableau est vide puis vérifier qu'on a assez de colonnes (jusqu'à H = 8). Nous avions eu des soucis de tableau déjà rempli ou partiellement, nous avons donc été contraint d'ajouter ces deux fonctions afin de sécuriser au maximum le code, pour qu'il soit le plus fiable possible et cibler nos réelles erreurs.

```
Dim dictDevisUrg As Object
Set dictDevisUrg = ChargerRef("Devis", "A", "G")

Dim i As Long
Dim caTotal As Double, caUrg As Double
Dim montantBrut As Double, reduc As Double, net As Double
Dim idDevis As String
Dim estUrgent As Boolean
```

Chargement du tableau, des données à récupérer et des variables.

```
For i = LBound(DataCmd, 1) To UBound(DataCmd, 1)

    If DataCmd(i, 1) <> "" Then

        montantBrut = 0
        If IsNumeric(DataCmd(i, 5)) Then montantBrut = CDbl(DataCmd(i, 5))
        reduc = 0
        If IsNumeric(DataCmd(i, 7)) Then reduc = CDbl(DataCmd(i, 7))

        net = montantBrut * (1 - reduc)

        caTotal = caTotal + net

        idDevis = Trim(CStr(DataCmd(i, 8)))

        estUrgent = False
        If dictDevisUrg.Exists(idDevis) Then
            If UCase(Trim(dictDevisUrg(idDevis))) = "OUI" Then estUrgent = True
        End If

        If estUrgent Then caUrg = caUrg + net
    End If
Next i
```

On commence par vérifier simplement qu'il y a un ID commande (Col 1) pour traiter la ligne avec l'instruction de la boucle If.

On récupère ensuite les montants et les réductions à appliquer sur les montants correspondants. On calcul le montant net de chaque cellule excel.

On calcule ensuite le Chiffre d'Affaire (CA) net c'est-à-dire avec les réductions appliquées dessus de l'année en cours.

Le code charge les informations d'urgence depuis la colonne G de la feuille "Devis" (OUI/NON). Il parcourt toutes les commandes et calcule le montant net. À l'aide de l'ID Devis, il vérifie si la commande était urgente :

Si **OUI**, le montant est ajouté à la variable caUrg (CA Urgent) en plus du caTotal

Le code calcule la part commerciale (caUrg / caTotal) et affiche un rapport financier formaté comparant le CA global et le CA généré spécifiquement par les urgences.

---

e. Affichage des travaux en retard pour pouvoir réagir s'il y a de trop nombreux retards

```
Dim dictDevis As Object, dictArtisan As Object
Set dictDevis = ChargerRef("Devis", "A", "F")
Set dictArtisan = ChargerRef("Artisan", "A", "B")

Dim dictCause As Object
Set dictCause = ChargerRef("Retard", "A", "C")   '
```

Chargement des dictionnaires pour retrouver les noms et les causes des retards

```
Dim wsReport As Worksheet
On Error Resume Next
Application.DisplayAlerts = False
Sheets("Alerte_Retards").Delete '
Application.DisplayAlerts = True
On Error GoTo 0

Set wsReport = Sheets.Add
wsReport.Name = "Alerte_Retards"
```

On prépare la feuille de résultat Alerte_Retard. On supprime l'ancienne avec Sheets("Alerte_Retards").Delete.

Le code détermine si une commande est en retard selon deux cas de figure :

La commande est terminée (IsDate(dReelle)), mais la date réelle est supérieure à la date prévue. La commande est en cours (pas de date réelle), mais la date du jour a dépassé la date prévue.

```
With wsReport
    .Range("A1").Value = "ID Client"
    .Range("B1").Value = "Artisan Responsable"
    .Range("C1").Value = "Date Prévue"
    .Range("D1").Value = "Date Réelle / État"
    .Range("E1").Value = "Jours de Retard"
    .Range("F1").Value = "Cause Identifiée"
    .Range("G1").Value = "Action Requise"
    .Range("A1:G1").Font.Bold = True
    .Range("A1:G1").Interior.Color = RGB(200, 50, 50)
    .Range("A1:G1").Font.Color = vbWhite
End With
```

Voici la mise en forme et la création des en-têtes de notre tableau. (Couleur cherchée sur internet). Le code marche ainsi :

```
ligne = 2

For i = LBound(DataCmd, 1) To UBound(DataCmd, 1)
    estEnRetard = False
    jours = 0


    If IsDate(DataCmd(i, 4)) Then
        dPrevue = DataCmd(i, 4)
        If IsDate(DataCmd(i, 11)) Then
            dReelle = DataCmd(i, 11)
            If dReelle > dPrevue Then
                estEnRetard = True
                jours = DateDiff("d", dPrevue, dReelle)
            End If
        ElseIf Date > dPrevue Then
            estEnRetard = True
            jours = DateDiff("d", dPrevue, Date)
        End If
```

On récupère la date prévue dPrevue et la date réelle de livraison (dReelle). Avec une boucle if/else if : on compare la date prévue avec la date réelle et la date (si jamais la commande n'est pas encore arrivée). La valeur est stockée.

```
            If DataCmd(i, 9) = "S03" Then estEnRetard = True
            If estEnRetard Then
                idDev = UCase(Trim(DataCmd(i, 8)))
                If dictDevis.Exists(idDev) Then
                    idArt = dictDevis(idDev)
                    If dictArtisan.Exists(idArt) Then
                        nomArt = dictArtisan(idArt)
                    Else
                        nomArt = idArt & " (?)"
                    End If
                Else
                    nomArt = "Inconnu"
                End If
                idRet = UCase(Trim(DataCmd(i, 12)))
                If dictCause.Exists(idRet) Then
                    lblCause = dictCause(idRet)
                Else
                    lblCause = "Non précisée"
                End If
                If idRet = "R00" Then lblCause = "Aucune (Retard inexpliqué)"
```

Ensuite, le programme va regarder le nombre de jours de retard et présenter un message dans la colonne G en fonction de la durée du retard. Si le retard est trop long, plus de 10 jours dans notre cas, la case en G passe d'une couleur rouge et présente de message : URGENT : Appeler l'Artisan. Sinon, il est marqué : à surveiller.

```
                With wsReport
                   .Cells(ligne, 1).Value = DataCmd(i, 2)
                   .Cells(ligne, 2).Value = nomArt
                   .Cells(ligne, 3).Value = dPrevue
                   If IsDate(DataCmd(i, 11)) Then
                       .Cells(ligne, 4).Value = dReelle
                   Else
                      .Cells(ligne, 4).Value = "EN COURS"
                      .Cells(ligne, 4).Font.Color = vbRed
                      .Cells(ligne, 4).Font.Bold = True
                   End If

                   .Cells(ligne, 5).Value = jours
                   .Cells(ligne, 6).Value = lblCause

                   If jours > 10 Then
                       .Cells(ligne, 7).Value = "URGENT : Appeler l'artisan"
                      .Cells(ligne, 7).Interior.Color = RGB(255, 200, 200)
                   Else
                       .Cells(ligne, 7).Value = "À surveiller"
                   End If
                End With
                ligne = ligne + 1
            End If
        End If
    Next i
    If ligne > 2 Then
       wsReport.Columns("A:G").AutoFit
       MsgBox "Analyse terminée : " & (ligne - 2) & " retards détectés." & vbCrLf & _
              "Consultez l'onglet 'Alerte_Retards' qui vient d'être créé.", vbExclamation
       wsReport.Activate
    Else
        MsgBox "Bonne nouvelle : Aucun retard détecté !", vbInformation
        Application.DisplayAlerts = False
        wsReport.Delete
        Application.DisplayAlerts = True
    End If
End Sub
```

Si un retard est détecté, le code crée une feuille "Alerte_Retards". Il y inscrit les détails (Client, Artisan, Jours de retard).

---

f. Mise en place de statistiques sur les causes des retards et sur la satisfaction

des clients afin de pouvoir améliorer le service aux clients

```
Sub StatCauses()
    Call ChargerDonnees

    If IsEmpty(DataCmd) Then Exit Sub
    If UBound(DataCmd, 2) < 12 Then
        MsgBox "Erreur : La plage de données chargée ne contient pas la colonne L (ID_Retard). Vérifiez la macro ChargerDonnees.", vbCritical
        Exit Sub
    End If

    Dim wsAvis As Worksheet, wsRetard As Worksheet, wsBilan As Worksheet
    Dim dictCauses As Object, dictRefCauses As Object

    On Error Resume Next
    Set wsAvis = ThisWorkbook.Sheets("Avis")
    Set wsRetard = ThisWorkbook.Sheets("Retard")
    On Error GoTo 0

    If wsAvis Is Nothing Or wsRetard Is Nothing Then
        MsgBox "Erreur : Vérifie que tu as bien les feuilles 'Avis' et 'Retard'.", vbCritical
        Exit Sub
    End If

    Dim moyenneNotes As Double
    On Error Resume Next
    moyenneNotes = Application.WorksheetFunction.Average(wsAvis.Range("B:B"))
    On Error GoTo 0
    Set dictCauses = CreateObject("Scripting.Dictionary")
    Set dictRefCauses = ChargerRef("Retard", "A", "C")

    Dim i As Long
    Dim nbCmd As Long, nbRetards As Long
    Dim codeRetard As String, libelleCause As String

    nbCmd = 0
    nbRetards = 0
    For i = LBound(DataCmd, 1) To UBound(DataCmd, 1)
        nbCmd = nbCmd + 1
        codeRetard = Trim(CStr(DataCmd(i, 12)))
```

Le code commence par charger les données des commandes et vérifie que la colonne contenant les codes de retard est bien présente pour éviter les erreurs.

Ensuite, il calcule la note moyenne de satisfaction des clients en lisant la colonne des notes dans la feuille Avis. Il prépare aussi un système de comptage pour identifier les causes des retards en se basant sur les descriptions stockées dans la feuille Retard.

Le programme parcourt ensuite toutes les commandes une par une. Si une commande possède un code de retard valide (différent de R00), il traduit ce code en une description compréhensible et l'ajoute au compteur des causes.

```
Application.ScreenUpdating = False
On Error Resume Next
Application.DisplayAlerts = False
Sheets("Bilan_Qualite").Delete
Application.DisplayAlerts = True
On Error GoTo 0

Set wsBilan = Sheets.Add
wsBilan.Name = "Bilan_Qualite"

With wsBilan
  .Range("B2").Value = "SATISFACTION CLIENT"
  .Range("B2").Font.Size = 16
  .Range("B2").Font.Bold = True

  .Range("B4").Value = "Note Moyenne :"
  .Range("C4").Value = moyenneNotes
  .Range("C4").NumberFormat = "0.0 ""/ 5"""
  .Range("C4").Font.Size = 20
  .Range("C4").Font.Bold = True

  .Range("B5").Value = "Verdict :"
    If moyenneNotes >= 4 Then
      .Range("C5").Value = "Excellent - Tout va bien!"
      .Range("C5").Interior.Color = RGB(146, 208, 80)
    ElseIf moyenneNotes >= 3 Then
      .Range("C5").Value = "Moyen - À surveiller"
      .Range("C5").Interior.Color = RGB(255, 192, 0)
    Else
      .Range("C5").Value = "Critique - Action requise"
      .Range("C5").Interior.Color = RGB(255, 0, 0)
      .Range("C5").Font.Color = vbWhite
    End If
  .Range("C5").HorizontalAlignment = xlCenter
  .Range("C5").Font.Bold = True
  .Range("B8").Value = "PERFORMANCE DÉLAIS"
  .Range("B8").Font.Size = 16
  .Range("B8").Font.Bold = True

  .Range("B10").Value = "Commandes Totales"
  .Range("C10").Value = nbCmd
```

Une fois les calculs terminés, le code supprime l'ancienne feuille de bilan si elle existe et en crée une nouvelle nommée Bilan_Qualite.

```
        .Range("B11").Value = "Commandes en Retard"
        .Range("C11").Value = nbRetards
        .Range("B12").Value = "Taux de Retard"
         If nbCmd > 0 Then
            .Range("C12").Value = nbRetards / nbCmd
         Else
            .Range("C12").Value = 0
         End If
        .Range("C12").NumberFormat = "0.00%"
        .Range("E2").Value = "TOP CAUSES DES RETARDS"
        .Range("E2").Font.Size = 14
        .Range("E2").Font.Bold = True
        .Range("E4").Value = "Cause"
        .Range("F4").Value = "Volume"
        .Range("E4:F4").Interior.Color = RGB(220, 220, 220)
    End With
    Dim lig As Long: lig = 5
    Dim k As Variant
    If dictCauses.Count > 0 Then
        For Each k In dictCauses.Keys
            wsBilan.Cells(lig, 5).Value = k
            wsBilan.Cells(lig, 6).Value = dictCauses(k)
            lig = lig + 1
        Next k
        Dim graph As Shape
        Set graph = wsBilan.Shapes.AddChart2(201, xlColumnClustered)
        graph.Left = wsBilan.Range("E" & lig + 2).Left
        graph.Top = wsBilan.Range("E" & lig + 2).Top
        graph.Width = 350
        graph.Height = 200

        graph.Chart.SetSourceData Source:=wsBilan.Range("E4:F" & lig - 1)
        graph.Chart.ChartTitle.Text = "Répartition des Causes"
    Else
        wsBilan.Range("E5").Value = "Aucun retard enregistré (Bravo!)"
    End If
    wsBilan.Columns("B:F").AutoFit
    Application.ScreenUpdating = True
    MsgBox "Analyse Qualité terminée.", vbInformation
End Sub
```

Sur cette feuille, il affiche d'abord la note de satisfaction client accompagnée d'un commentaire et d'une couleur (vert, orange ou rouge) selon le résultat. Il inscrit ensuite les statistiques de performance, comme le nombre total de commandes, le nombre de retards et le pourcentage global de retard. Enfin, il liste les différentes causes identifiées dans un tableau et génère automatiquement un graphique en barres pour visualiser la répartition de ces causes de retard.

Maksimilien DEUTSCH Mathis PAULY Martin GOULET
Pierre TOUET-SI-ABDELHADI Romain CUCHE

ENGINEERING SCHOOL
Creating the future together