Créer et utiliser une fonction en langage M (Power Query)

Objectif :

- Apprendre **à créer une fonction** dans Power Query (langage M).
- Apprendre **à appeler cette fonction** dans un cas concret.
- Travailler avec le fichier réel ECS_Donnees_Entreprise.xlsx.

Exemple 1 : Créer une fonction qui classe les clients selon la quantité

Étape 1 – Charger les données

Ouvrir Power BI ou Excel > Power Query
Charger le fichier ECS_Donnees_Entreprise.xlsx

Nom de la requête : ECS_Donnees

Étape 2 – Créer une fonction personnalisée

1. Dans Power Query, cliquer sur Accueil > Avancé > Nouvelle requête > Requête vide
2. Aller dans Éditeur avancé
3. Coller ce code :

```
let
   ClientGros = (quantite as number) as text =>
      if quantite > 300 then "Gros Client"
      else if quantite >= 100 then "Client Moyen"
      else "Petit Client"
in
   ClientGros
```

4. Renommer la requête : fnTypeClient

Cette requête est une fonction qui prend quantite comme paramètre et retourne un **texte**.

Étape 3 – Tester la fonction

Créer une requête vide, coller :

```
let
   Test = fnTypeClient(150)
in
   Test
```

Résultat = "Client Moyen"

Étape 4 – Utiliser la fonction dans les données ECS_Donnees

1. Retourner à la requête ECS_Donnees

2. Ajouter une colonne personnalisée :
    - Aller dans Ajouter une colonne > Colonne personnalisée
    - Nom : Type Client
    - Formule :

fnTypeClient([Quantité])

Cela va appeler la fonction **pour chaque ligne** avec la valeur de la colonne Quantité.

Résultat attendu :

Produit	Quantité	Type Client
Lait entier	400	Gros Client
Yaourt fraise	120	Client Moyen
Poussin démarrage	80	Petit Client

EXEMPLE 2 : Fonction qui identifie si un produit est un produit laitier

Objectif :

Créer une fonction qui dit :

- "Produit Laitier" si le nom contient "lait" ou "yaourt"
- Sinon → "Autre Produit"

Étape 1 – Créer la fonction

1. Dans Power Query, cliquer sur :
   Accueil > Nouvelle source > Requête vide
2. Aller dans :
   Accueil > Éditeur avancé
3. Copier-coller ce code :

```
let
   EstProduitLaitier = (nomProduit as text) as text =>
      let
         nomMinuscule = Text.Lower(nomProduit),
         resultat = if Text.Contains(nomMinuscule, "lait") or Text.Contains(nomMinuscule,
"yaourt") then
            "Produit Laitier"
         else
            "Autre Produit"
      in
         resultat
in
```

EstProduitLaitier

4. Renommer la requête : fnProduitLaitier

Étape 2 – Tester la fonction

Créer une **requête vide** pour tester, puis coller ceci :

```
let
   Test = fnProduitLaitier("Yaourt fraise")
in
   Test
```

Résultat : "Produit Laitier"

Essayer aussi :

```
let
   Test = fnProduitLaitier("Poussin démarrage")
in
   Test
```

Résultat : "Autre Produit"

Étape 3 – Utiliser la fonction dans la table principale

1. Retourner dans la requête ECS_Donnees
2. Aller dans Ajouter une colonne > Colonne personnalisée
3. Nom de la colonne : Catégorie Produit
4. Formule :

```
fnProduitLaitier([Produit])
```

Cela va appliquer la fonction à chaque ligne avec la valeur de la colonne Produit.

Résultat attendu :

Produit	Catégorie Produit
Yaourt nature	Produit Laitier
Lait entier	Produit Laitier
Engrais NPK	Autre Produit
Poussin croissance	Autre Produit

Grâce à cette **fonction réutilisable**, on peut facilement :

- Réutiliser la même logique ailleurs
- Modifier une seule fois la fonction (au lieu de changer plusieurs colonnes)
- Travailler plus vite et plus proprement qu'avec l'interface graphique

EXEMPLE 3 : Dire si une quantité est "Grande" ou "Petite"

Objectif :

Créer une fonction qui retourne :

- "Grande Quantité" si Quantité > 100
- Sinon "Petite Quantité"

ÉTAPE 1 – Créer la fonction dans Power Query

1. Ouvrir Power BI ou Excel > Aller dans Power Query

Cliquer sur :
Accueil > Nouvelle source > Requête vide

2. Ouvrir l'Éditeur avancé :

Cliquer sur Accueil > Éditeur avancé

3. Coller ce code :

```
let
   ClasserQuantite = (qte as number) as text =>
      if qte > 100 then
         "Grande Quantité"
      else
         "Petite Quantité"
in
   ClasserQuantite
```

Puis cliquer sur Terminer

Renommer la requête en **fnClasserQuantite**

ÉTAPE 2 – Tester la fonction (facultatif, mais utile)

1. Créer une **nouvelle requête vide**
2. Ouvrir l'Éditeur avancé
3. Coller ce test :

```
let
```

```
    Test = fnClasserQuantite(150)
in
    Test
```

Résultat : "Grande Quantité"

Refaire avec 50, résultat attendu : "Petite Quantité"

ÉTAPE 3 – Utiliser la fonction dans vos données

1. Revenir à votre table principale : ECS_Donnees
2. Cliquer sur :
   Ajouter une colonne > Colonne personnalisée
3. Nommer la colonne : **"Type Quantité"**
4. Écrire la formule suivante :

```
fnClasserQuantite([Quantité])
```

Quantité doit être bien écrit comme dans votre table.

Résultat attendu :

Quantité	Type Quantité
50	Petite Quantité
120	Grande Quantité
95	Petite Quantité
400	Grande Quantité

Etapes :

Étape	Action	Exemple
1	Créer une fonction	let NomFonction = (x) => if x > 100 then "Gros" else "Petit"
2	Tester la fonction	NomFonction(150) donne "Gros"
3	Utiliser dans une table	NomFonction([Colonne])

Les types de données en langage M (avec exemples réels)

Source utilisée : ECS_Donnees_Entreprise.xlsx

Type M	Description	Exemple concret dans ECS_Donnees_Entreprise
null	Valeur vide ou manquante	Une cellule vide dans la colonne Pays
function	Fonction définie dans le code	Une logique pour calculer un rang
duration	Période entre deux dates	Délai entre Date Commande et Date Livraison
date	Une date sans heure	Date Commande : #date(2024, 4, 7)
time	Heure uniquement	Création manuelle d'une colonne avec l'heure
binary	Donnée binaire (fichier)	Charger une image/logo via File.Contents()
type	Type comme valeur (métadonnée)	Vérifier si une colonne est type number
any	Type générique, accepte tout	Une fonction personnalisée avec tout type

1. null – Valeur vide

= Table.SelectRows(Source, each [Pays] = null)

Cela renvoie toutes les lignes où la colonne Pays est vide. Très utile pour vérifier les données manquantes.

2. function – Créer une fonction personnalisée

```
// Fonction pour vérifier si produit est un "Produit laitier"
ProduitLaitier = (produit as text) as text =>
    if Text.Contains(produit, "lait") then "Oui" else "Non"
```

On peut ensuite l'utiliser pour créer une nouvelle colonne :

= Table.AddColumn(Source, "Est Produit Laitier", each ProduitLaitier([Produit]))

3. duration – Calculer la durée entre deux dates

= Table.AddColumn(Source, "Délai Livraison", each [Date Livraison] - [Date Commande], type duration)

Cela donne une durée du type 3.00:00:00 (3 jours)

4. date – Forcer une date précise (pour filtrer ou tester)

= Table.SelectRows(Source, each [Date Commande] = #date(2024, 4, 7))

5. time – Créer une colonne avec une heure fixe

= Table.AddColumn(Source, "Heure Fixe", each #time(10, 0, 0), type time)

On ajoute une heure fictive à chaque ligne (utile si on veut simuler des horaires de livraison)

6. binary – Charger un fichier binaire

= File.Contents("C:\Users\Mention Info 2\Pictures\logo_ecs.png")

Cela retourne les données binaires de l'image/logo de l'entreprise. Utilisable pour charger un fichier externe dans Power BI.

7. type – Vérification ou définition de types

= Value.Type([Quantité])

Cette commande retourne type number, utile pour détecter dynamiquement le type d'une colonne ou d'une valeur.

8. any – Utilisé dans des fonctions flexibles

```
TypeFlexible = (valeur as any) =>
    if valeur is number then "C'est un nombre"
    else if valeur is text then "C'est du texte"
    else "Autre"
```

Fonction générique capable de traiter plusieurs types sans erreur.