ETAPES D'ANALYSE DES DONNEES ADABI CHALLENGE 2

Étape 1 : Importer et normaliser les données Import des fichiers CSV

// Pour chaque fichier (clients.csv, ventes.csv, achats.csv, stock.csv)

CODE : let Source = Csv.Document(File.Contents("C:\chemin\vers\fichier.csv"), [Delimiter=",", Encoding=65001, QuoteStyle=QuoteStyle.None]), PromotedHeaders = Table.PromoteHeaders(Source, [PromoteAllScalars=true]) in PromotedHeaders

Normalisation des dates pour chaque table

Table Clients

let Source = #"Clients importés", // Conversion des colonnes de date DateInscriptionConvertie = Table.TransformColumns(Source, { {"date_inscription", each if _ = null then null else try Date.FromText(_) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " janvier ", "/01/"), " février ", "/02/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " mars ", "/03/"), " avril ", "/04/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " mai ", "/05/"), " juin ", "/06/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " juillet ", "/07/"), " août ", "/08/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " septembre ", "/09/"), " octobre ", "/10/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " novembre ", "/11/"), " décembre ", "/12/")) otherwise null } }), DateDernierAchatConvertie = Table.TransformColumns(DateInscriptionConvertie, { {"date_dernier_achat", each if _ = null then null else try Date.FromText(_) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " janvier ", "/01/"), " février ", "/02/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " mars ", "/03/"), " avril ", "/04/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " mai ", "/05/"), " juin ", "/06/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " juillet ", "/07/"), " août ", "/08/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " septembre ", "/09/"), " octobre ", "/10/")) otherwise try Date.FromText(Text.Replace(Text.Replace(_, " novembre ", "/11/"), " décembre ", "/12/")) otherwise null } }) in DateDernierAchatConvertie

Table Ventes

let Source = #"Ventes importées", DateVenteConvertie = Table.TransformColumns(Source, { {"date_vente", each if _ = null then null else try Date.FromText(_) otherwise null } }) in DateVenteConvertie

Table Achats

```
let Source = #"Achats importés", DateAchatConvertie = Table.TransformColumns(Source, { {"date_achat", each if _ = null then null else try Date.FromText(_) otherwise null } }) in DateAchatConvertie
```

Table Stock

```
let Source = #"Stock importé", DateReappConvertie = Table.TransformColumns(Source, { {"date_réapprovisionnement", each if _ = null then null else try Date.FromText(_) otherwise null } }) in DateReappConvertie
```

<u>Étape 2</u> : **Gestion des valeurs manquantes Table Clients**

```
let Source = #"Clients avec dates normalisées", // Repérer les colonnes avec valeurs manquantes RemplacerValeursNANom = Table.ReplaceValue(Source, null, "Non renseigné", Replacer.ReplaceValue, {"nom"}), RemplacerValeursNAEmail = Table.ReplaceValue(RemplacerValeursNANom, null, "email@manquant.com", Replacer.ReplaceValue, {"email"}), RemplacerValeursNATel = Table.ReplaceValue(RemplacerValeursNAEmail, null, "Non renseigné", Replacer.ReplaceValue, {"téléphone"}), RemplacerValeursNAPays = Table.ReplaceValue(RemplacerValeursNATel, null, "Non renseigné", Replacer.ReplaceValue, {"pays"}), // Pour les dates manquantes, utiliser la médiane // D'abord calculer la médiane des dates d'inscription non-nulles DatesInscriptionNonNulles = List.RemoveNulls(Table.Column(RemplacerValeursNAPays, "date_inscription")), MedianeDateInscription = List.Median(DatesInscriptionNonNulles), RemplacerDateInscriptionNA = Table.ReplaceValue(RemplacerValeursNAPays, null, MedianeDateInscription, Replacer.ReplaceValue, {"date_inscription"}), // Pour date_dernier_achat, on peut utiliser la date d'inscription si null RemplacerDateDernierAchatNA = Table.AddColumn(RemplacerDateInscriptionNA, "date_dernier_achat_temp", each if [date_dernier_achat] = null then [date_inscription] else [date_dernier_achat]), SupprimerAncienneDateDernierAchat = Table.RemoveColumns(RemplacerDateDernierAchatNA, {"date_dernier_achat"}), RenommerNouvelleColonne = Table.RenameColumns(SupprimerAncienneDateDernierAchat, {{"date_dernier_achat_temp", "date_dernier_achat"}}), // Pour total_dépense, utiliser la médiane TotalDepenseNonNulls = List.RemoveNulls(Table.Column(RenommerNouvelleColonne, "total_dépense")), MedianeTotalDepense = List.Median(TotalDepenseNonNulls), RemplacerTotalDepenseNA = Table.ReplaceValue(RenommerNouvelleColonne, null, MedianeTotalDepense, Replacer.ReplaceValue, {"total_dépense"}) in RemplacerTotalDepenseNA
```

Traitements similaires appliquées pour les autres tables

<u>Étape 3</u> : Création du modèle de données

1. Création des relations Dans Power BI Desktop:
2. Création de mesures DAX pour les KPIs de relation client

```
// Nombre de clients
Nombre de clients = COUNTROWS(clients)
// Nombre de ventes
Nombre de ventes = COUNTROWS(ventes)
// Chiffre d'affaires total
CA Total = SUM(ventes[montant_total])
// Panier moyen
Panier Moyen = DIVIDE([CA Total], [Nombre de ventes], 0)
// Fréquence d'achat (nombre moyen d'achats par client)
Fréquence d'achat = DIVIDE([Nombre de ventes], [Nombre de clients], 0)
// Valeur client moyenne (CLV simplifié)
CLV = DIVIDE([CA Total], [Nombre de clients], 0)
// Moyenne des jours depuis le dernier achat
Récence Moyenne =
AVERAGEX(
  clients,
  DATEDIFF(clients[date_dernier_achat], TODAY(), DAY)
)
// Taux de rétention (clients ayant acheté dans les 90 derniers jours)
Taux de rétention =
DIVIDE(
  CALCULATE(
    COUNTROWS(clients),
    FILTER(
```

```
        clients,
        DATEDIFF(clients[date_dernier_achat], TODAY(), DAY) <= 90
    )
  ),
  [Nombre de clients],
  0
) * 100
// Taux de clients dormants (pas d'achat depuis 180 jours)
Taux de clients dormants =
DIVIDE(
  CALCULATE(
    COUNTROWS(clients),
    FILTER(
      clients,
DATEDIFF(clients[date_dernier_achat], TODAY(), DAY) > 180 ) ), [Nombre de clients], 0 ) * 100
// Segmentation RFM // Pour cela, J'ai créé une table calculée RFM en DAX
Table Segmentation RFM (Récence, Fréquence, Montant)
Table RFM =
VAR TableRecence =
  SUMMARIZE(
    clients,
    clients[id_client],
    "Date Dernier Achat",
      CALCULATE(
        MAX(ventes[date_vente]),
        FILTER(ventes, VALUE(ventes[id_client]) = VALUE(clients[id_client]))
      ),
    "Récence",
```

```
        DATEDIFF(
            CALCULATE(
                MAX(ventes[date_vente]),
                FILTER(ventes, VALUE(ventes[id_client]) = VALUE(clients[id_client]))
            ),
            TODAY(),
            DAY
        ),
    "Score R",
        VAR DerniereDate = CALCULATE(
            MAX(ventes[date_vente]),
            FILTER(ventes, VALUE(ventes[id_client]) = VALUE(clients[id_client]))
        )
        VAR JoursDepuisDernierAchat = DATEDIFF(DerniereDate, TODAY(), DAY)
        RETURN
            SWITCH(
                TRUE(),
                JoursDepuisDernierAchat <= 30, 5,
                JoursDepuisDernierAchat <= 60, 4,
                JoursDepuisDernierAchat <= 90, 3,
                JoursDepuisDernierAchat <= 180, 2,
                1
            )
)
VAR TableAvecFrequence =
    ADDCOLUMNS(
        TableRecence,
        "Fréquence",
```

```
        CALCULATE(
            COUNTROWS(ventes),
            FILTER(ventes, VALUE(ventes[id_client]) = VALUE([id_client]))
        ),
    "Score F",
        VAR FrequenceClient = CALCULATE(
            COUNTROWS(ventes),
            FILTER(ventes, VALUE(ventes[id_client]) = VALUE([id_client]))
        )
        RETURN
            SWITCH(
                TRUE(),
                FrequenceClient >= 10, 5,
                FrequenceClient >= 7, 4,
                FrequenceClient >= 5, 3,
                FrequenceClient >= 3, 2,
                1
            )
  )
VAR TableComplete =
  ADDCOLUMNS(
    TableAvecFrequence,
    "Montant",
        CALCULATE(
            SUM(ventes[montant_total]),
            FILTER(ventes, VALUE(ventes[id_client]) = VALUE([id_client]))
        ),
    "Score M",
```

```
        VAR MontantClient = CALCULATE(
            SUM(ventes[montant_total]),
            FILTER(ventes, VALUE(ventes[id_client]) = VALUE([id_client]))
        )
        RETURN
            SWITCH(
                TRUE(),
                MontantClient >= 10000, 5,
                MontantClient >= 5000, 4,
                MontantClient >= 2000, 3,
                MontantClient >= 1000, 2,
                1
            )
    )
VAR TableAvecSegment =
    ADDCOLUMNS(
        TableComplete,
        "Segment RFM",
            [Score R] & [Score F] & [Score M],
        "Score RFM Numerique",
            [Score R] * 100 + [Score F] * 10 + [Score M]
    )
RETURN
    ADDCOLUMNS(
        TableAvecSegment,
        "Catégorie Client",
            VAR ScoreRFM = [Score RFM Numerique]
            RETURN
```

```
        SWITCH(
            TRUE(),
            ScoreRFM >= 444, "Champions",
            ScoreRFM >= 400 && [Score R] >= 4, "Clients Fidèles",
            ScoreRFM <= 200 && [Score R] <= 2, "Clients Perdus",
            ScoreRFM <= 300 && [Score R] >= 4, "Nouveaux Clients à Potentiel",
            "Clients Standards"
        )
    )
```

CREATION TABLES:

TABLE DE DATE : Date1 = CALENDARAUTO(12)

TABLE METRIQUES MAGASIN :

```
Métriques Magasins =
SUMMARIZE(
 ventes,
 ventes[id_magasin],
 "Nombre de Ventes", COUNTROWS(ventes),
 "CA Total", SUM(ventes[montant_total]),
 "Nombre de Clients Uniques", DISTINCTCOUNT(ventes[id_client]),
 "Panier Moyen", DIVIDE(SUM(ventes[montant_total]), COUNTROWS(ventes), 0)
 )
```

Par Désiré TRA,

Data Analyste