El Rhilani Hassan

SAÉ 2.01

Conception et implémentation d’une base de données

Une image contenant obscurité

Description générée automatiquementUne image contenant Graphique, logo, symbole, Police

Description générée automatiquement

Table des matières

[Introduction 2](#_Toc168303652)

[Normalisation des Données 3](#_Toc168303653)

[Analyse Initiale 3](#_Toc168303654)

[Relation universelle 3](#_Toc168303655)

[Types des variables 3](#_Toc168303656)

[Dépendances fonctionnelles 4](#_Toc168303657)

[Première Forme Normale (1NF) 4](#_Toc168303658)

[Deuxième Forme Normale (2NF) 5](#_Toc168303659)

[Troisième Forme Normale (3NF) 6](#_Toc168303660)

[Forme Normale de Boyce-Codd (BCNF) 6](#_Toc168303661)

[Quatrième Forme Normale (4NF) 6](#_Toc168303662)

[Requêtes et BDR 7](#_Toc168303663)

[Base de données relationnel 7](#_Toc168303664)

[Requêtes 8](#_Toc168303665)

[Datavisualisation 16](#_Toc168303666)

# Introduction

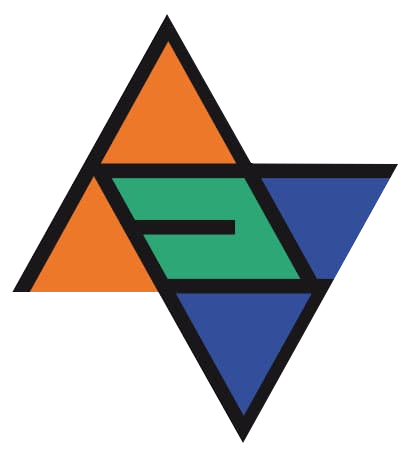
Dans le cadre de cette SAE, nous avons pour objectif de collaborer avec l'association AVEC, une organisation d'intérêt général créée en janvier 2015. AVEC favorise l'émancipation de tous et lutte contre toutes formes d'exclusion en développant un projet d'accompagnement global : culture, éducation, social. Cette association intervient dans trois dimensions : développement social, territorial et culturel, afin de permettre aux habitants d'accéder à l'autonomie et à l'émancipation.

L'association AVEC est partiellement subventionnée par les sommes versées par les adhérents sous forme d'adhésion ou de don. Chaque année, ces informations sont enregistrées dans un fichier Excel. Le travail demandé dans cette SAE consiste à :

Créer une base de données permettant le suivi des adhésions et des dons au fil des années, avec un niveau de granularité plus fin que celui existant aujourd'hui.

Présenter un tableau de bord basé sur les résultats des requêtes formulées par l'association.

Cette tâche nous permettra d'appliquer nos compétences en gestion de données et en analyse, tout en contribuant au développement d'un outil essentiel pour l'association AVEC.



# Normalisation des Données

## Analyse Initiale

Les informations incluent :  
- ID, nom, prénom, date de naissance, adresse (rue, longitude…), téléphone, état.  
- Informations sur les adhésions pour chaque année (date d'adhésion, montant, don, moyen de paiement).  
- Informations sur le bénévolat.

## Relation universelle

R ( ID\*\*, Nom , Prenom , DateNaissance , Rue , Ville , Latitude , Longitude , Etat , Telephone , DateAdhesion2015 , Montant2015 , Don 2015 , MoyenDePaiement2015 , DateAdhesion2016 , Montant2016 , Don 2016 , MoyenDePaiement2016, DateAdhesion2017 , Montant2017 , Don 2017 , MoyenDePaiement2017, DateAdhesion2018 , Montant2018 , Don 2018 , MoyenDePaiement2018, DateAdhesion2019 , Montant2019 , Don 2019 , MoyenDePaiement2019, DateAdhesion2020 , Montant2020 , Don 2020 , MoyenDePaiement2020 , DateAdhesion2021 , Montant2021 , Don 2021 , MoyenDePaiement2021, Bénévole )

On a ID\*\* comme clé primaire car elle a une dépendance totale avec les autres attributs

Choix de supprimer certaines colonnes (expr1034), car elle est vide.

## Types des variables

ID : INT – entier, car c’est un nombre distinct et me sert de clé primaire

DateNaissance : DATE – car c’est plus simple pour les requêtes qu’une chaine de char

Rue : VARCHAR – une rue, c’est une chaine de charactères

Ville : VARCHAR – une ville, c’est aussi une chaine de charactères

Latitude : FLOAT – comme les coordonnées sont en format « 12,12313 »

Longitude : FLOAT - comme les coordonnées sont en format « 12,12313 »

Etat : VARCHAR – un état, c’est également une chaine de charactères

Telephone : VARCHAR – en effet, j’ai préféré choisir une chaine de charactères, car je pense que c’est plus adapté que des entiers et c’est plus simple après pour les requêtes, et les numéros de téléphone commencent par 0 et ce n’est pas faisable pour des entiers

DateAdhesion[« année »] : DATE - car c’est plus simple pour les requêtes qu’une chaine de char

Montant[« année »] : FLOAT – car c’est un montant d’une monnaie

Don[« année »] : FLOAT – car c’est un montant d’une monnaie

MoyenDePaiement[« année »] : VARCHAR – car les moyens de paiements sont représentés par des chaines de charactères

Avec [« année »] : 2015, 2016, 2017, 2018, 2019, 2020, 2021

Bénévole : VARCHAR – J’ai choisi une chaine de charactères pour « OUI » et « NON », dans les faits c’est faisable aussi pour des booléens, mais j’ai préféré OUI et NON car c’est plus clair

## Dépendances fonctionnelles

Dépendance fonctionnelle quand les attributs sont déterminés de façon unique par les déterminants

DF1 : ID 🡪 Nom, Prenom, DateNaissance , Rue , Ville, Telephone

DF2 : ID 🡪 DateAdhesion2015, Montant2015, Don2015, MoyenDepaiement2015

DF3 : ID 🡪 DateAdhesion2016, Montant2016, Don2016, MoyenDepaiement2016

DF4 : ID 🡪 DateAdhesion2015, Montant2017, Don2017, MoyenDepaiement2017

DF5 : ID 🡪 DateAdhesion2015, Montant2018, Don2018, MoyenDepaiement2018

DF6 : ID 🡪 DateAdhesion2019, Montant2019, Don2019, MoyenDepaiement2019

DF7 : ID 🡪 DateAdhesion2020, Montant2020, Don2020, MoyenDepaiement2020

DF8 : ID 🡪 DateAdhesion2021, Montant2021, Don2021, MoyenDepaiement2021

DF9 : ID 🡪 Bénévole

## Première Forme Normale (1NF)

Pour respecter la 1NF, chaque colonne doit contenir des valeurs atomiques et chaque enregistrement doit être unique.

Du coup, j’ai supprimé le numéro de téléphone additionnel pour la rendre 1NF.

## Deuxième Forme Normale (2NF)

Pour passer à la 2NF, la relation doit être en 1NF et Non respectée si dans une DF tous les attributs non clés doivent dépendre de la clé primaire.

2Nf POUR LA DF1 :

2nf non respectée, décomposition r en 2 tables

adh(ID\*\*, Nom, Prenom, DateNaissance , Rue , Ville, Telephone)  
R2 ( ID\*\* , Latitude , Longitude , Etat , DateAdhesion2015 , Montant2015 , Don 2015 , MoyenDePaiement2015 , DateAdhesion2016 , Montant2016 , Don 2016 , MoyenDePaiement2016, DateAdhesion2017 , Montant2017 , Don 2017 , MoyenDePaiement2017, DateAdhesion2018 , Montant2018 , Don 2018 , MoyenDePaiement2018, DateAdhesion2019 , Montant2019 , Don 2019 , MoyenDePaiement2019, DateAdhesion2020 , Montant2020 , Don 2020 , MoyenDePaiement2020 , DateAdhesion2021 , Montant2021 , Don 2021 , MoyenDePaiement2021, Bénévole )

2NF pour la DF2, DF3, DF4, DF5, DF6, DF7 et DF8 : (comme c’est quasiment pareil, y’a juste les années qui changent) :

[« année »] : 2015, 2016, 2017, 2018, 2019, 2020, 2021

2nf non respectée

An(ID\*\*, DateAdhesion[« année »], Montant[« année »], Don[« année »], Moyen de paiement[« année »])

R3(ID\*\*, Latitude, Longitude, Etat, Bénévole)

2NF pour la DF9 :

2nf non respectée

Ben(ID\*\*, Bénévole)

R4(ID\*\*, Latitude, Longitude, Etat)

Adhérents (ID\*\*, Nom, Prenom, DateNaissance, Rue, Ville, Telephone)

[« Année »] (ID\*\*, DateAdhesion, Montant, Don, MoyenDePaiement)

[« année »] : 2015, 2016, 2017, 2018, 2019, 2020, 2021

Bénévoles (ID\*\*, Bénévole)

## Troisième Forme Normale (3NF)

Pour être en 3NF, une relation doit être en 2NF et que ce n’est pas un déterminant primaire vers un déterminé non primaire

Dans notre cas, tous les DF sont en 3NF

## Forme Normale de Boyce-Codd (BCNF)

Pour que la DF respecte la BCNF, ne faut pas qu’un déterminant non primaire détermine un déterminé primaire

Dans notre cas, tous les DF sont en BCNF.

## Quatrième Forme Normale (4NF)

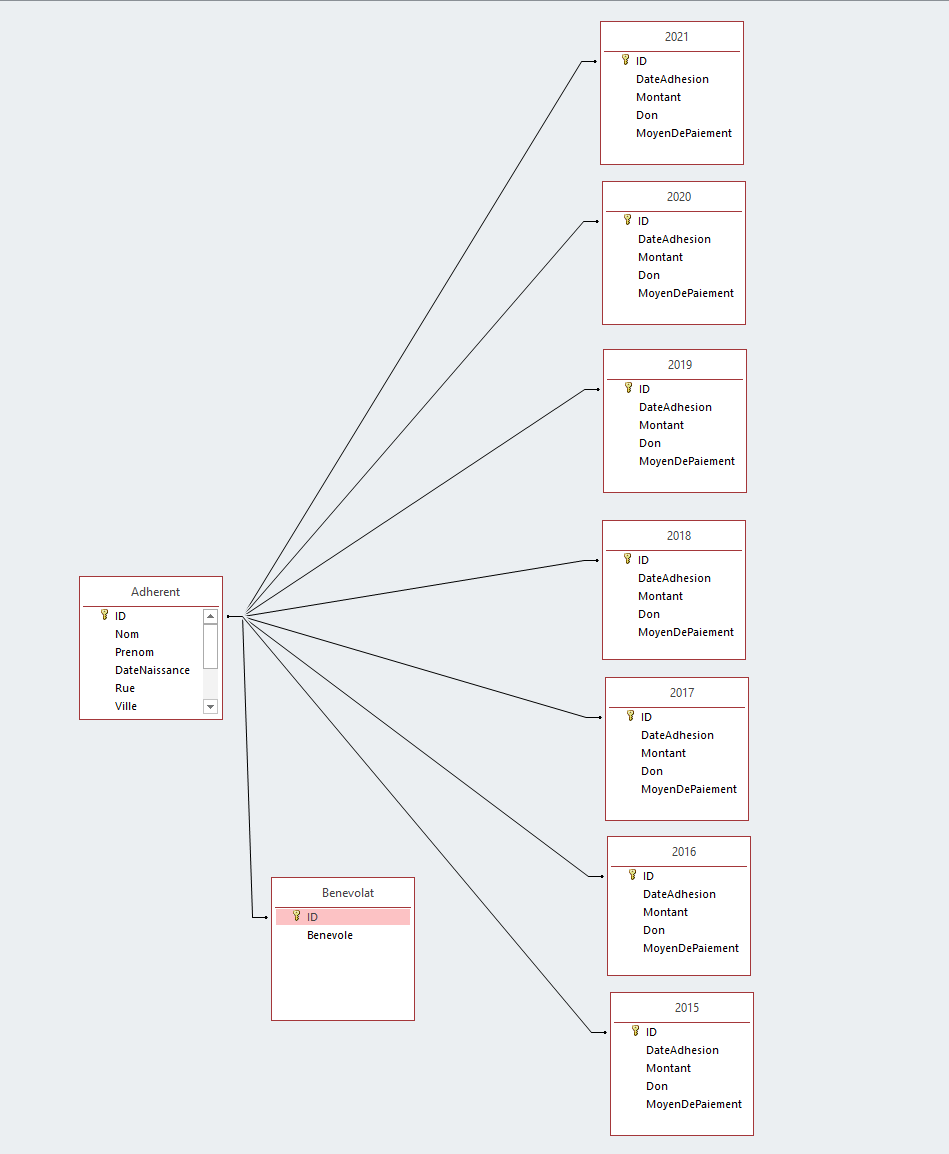
Pour que la DF respecte la 4NF, ne faut pas qu’un déterminant primaire détermine un déterminé primaire.

Dans notre cas, la 4NF est également respecté dans les DF

# Requêtes et BDR

## Base de données relationnel

J’ai choisi de travailler sur ACCESS, car c’est un logiciel complet et simple d’utilisation.

J’ai créé mes tables issues de la normalisation avec la clé primaire « ID\*\* »

## Requêtes

#### Nombre d’adhérents par années

SELECT YEAR(DateAdhesion) AS Annee, COUNT(ID) AS NombreAdherents

FROM (SELECT DateAdhesion, ID FROM [2015]

        UNION ALL

        SELECT DateAdhesion, ID FROM [2016]

        UNION ALL

        SELECT DateAdhesion, ID FROM [2017]

        UNION ALL

        SELECT DateAdhesion, ID FROM [2018]

        UNION ALL

        SELECT DateAdhesion, ID FROM [2019]

        UNION ALL

        SELECT DateAdhesion, ID FROM [2020]

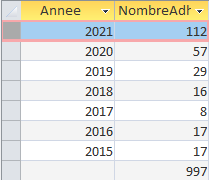
        UNION ALL

        SELECT DateAdhesion, ID FROM [2021]

    )  AS AllYears

GROUP BY YEAR(DateAdhesion)

ORDER BY Annee;

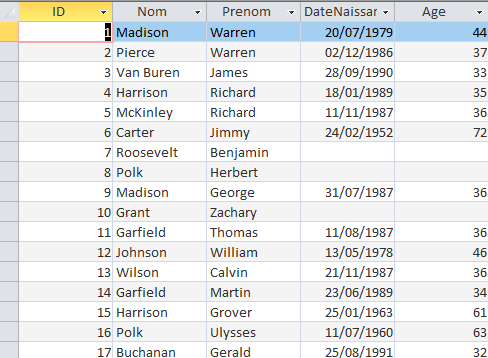


Le code utilise UNION ALL pour combiner les colonnes DateAdhesion et ID des tables de 2015 à 2021 en une seule table temporaire nommée AllYears. Il regroupe les données par année (YEAR(DateAdhesion)) et compte le nombre d'adhérents (COUNT(ID)) pour chaque année. Enfin, les résultats sont triés par année (ORDER BY Annee).

#### L’âge des adhérents

SELECT Adherent.ID, Adherent.Nom, Adherent.Prenom, Adherent.DateNaissance, DateDiff("yyyy",[DateNaissance],Date())-IIf(Format([DateNaissance],"mmdd")>Format(Date(),"mmdd"),1,0) AS Age

FROM Adherent;



Je sélectionne les colonnes ID, Nom, Prenom et DateNaissance de la table Adherent, et calcule l'âge de chaque adhérent en années. Il utilise DateDiff("yyyy", [DateNaissance], Date()) pour obtenir la différence en années entre la date de naissance et la date actuelle. Ensuite, il ajuste l'âge en soustrayant 1 si l'anniversaire de l'adhérent est après la date actuelle (IIf(Format([DateNaissance], "mmdd") > Format(Date(), "mmdd"), 1, 0)).

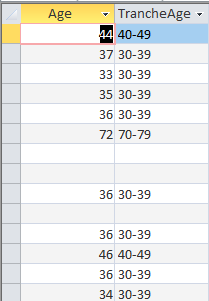
#### Création des tranches d’âge

SELECT AgesAdherents.Age, IIf(Age Is Null,Null,IIf(Age<10,"0-9",IIf(Age<20,"10-19",IIf(Age<30,"20-29",IIf(Age<40,"30-39",IIf(Age<50,"40-49",IIf(Age<60,"50-59",IIf(Age<70,"60-69",IIf(Age<80,"70-79",IIf(Age<90,"80-89","90+")))))))))) AS TrancheAge

FROM AgesAdherents;

Comme vous pouvez voir dans le code, mes tranches d’âge sont :

[0-9], [10-19], [20-29], [30-39], [40-49], [50-59], [60-69], [70-79], [80-89] et 90+

Le code sélectionne l'âge des adhérents depuis la table AgesAdherents et crée une nouvelle colonne TrancheAge qui classe chaque adhérent dans une tranche d'âge. Si l'âge est nul, la tranche d'âge sera également nulle. Sinon, l'âge est évalué pour déterminer à quelle tranche d'âge il appartient : 0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89 ou 90+

#### Nombre d’adhérents par tranche d’âge

Faisable à l’aide de la requête juste au-dessus.

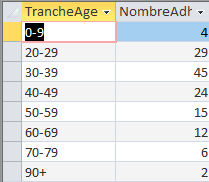
SELECT TrancheAge, COUNT(TrancheAge) AS NombreAdherents

FROM TranchesAge

WHERE TrancheAge IS NOT NULL

GROUP BY TrancheAge

ORDER BY TrancheAge;

 Le code sélectionne les tranches d'âge (TrancheAge) et compte le nombre d'adhérents (COUNT(TrancheAge)) pour chaque tranche d'âge à partir de la table TranchesAge. Il exclut les tranches d'âge nulles (WHERE TrancheAge IS NOT NULL), regroupe les résultats par tranche d'âge (GROUP BY TrancheAge), et les trie par tranche d'âge (ORDER BY TrancheAge).

#### Les adhérents ayant adhérer au moins une fois (année) et leur ville

SELECT DISTINCT Adherent.ID, Adherent.Ville

FROM ((((((Adherent LEFT JOIN 2015 ON Adherent.ID = [2015].ID) LEFT JOIN 2016 ON Adherent.ID = [2016].ID) LEFT JOIN 2017 ON Adherent.ID = [2017].ID) LEFT JOIN 2018 ON Adherent.ID = [2018].ID) LEFT JOIN 2019 ON Adherent.ID = [2019].ID) LEFT JOIN 2020 ON Adherent.ID = [2020].ID) LEFT JOIN 2021 ON Adherent.ID = [2021].ID

WHERE [2015].ID IS NOT NULL OR

    [2016].ID IS NOT NULL OR

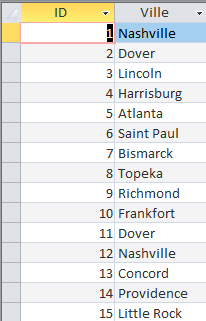
    [2017].ID IS NOT NULL OR

    [2018].ID IS NOT NULL OR

    [2019].ID IS NOT NULL OR

    [2020].ID IS NOT NULL OR

    [2021].ID IS NOT NULL;

 Le code utilise des jointures LEFT JOIN pour associer la table Adherent avec les tables annuelles de 2015 à 2021. Il sélectionne les adhérents distincts (DISTINCT) et leurs villes (Adherent.Ville) pour lesquels une adhésion existe dans au moins une des tables annuelles (vérifié par WHERE [year].ID IS NOT NULL).

#### Nombre d’adhérents par ville

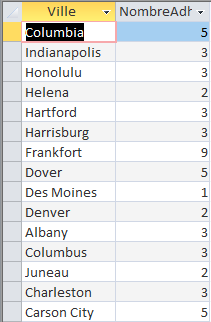
Faisable à l’aide de la requête au-dessus.

SELECT Ville, COUNT(ID) AS NombreAdherents

FROM AdherentsUniques

GROUP BY Ville

ORDER BY NombreAdherents DESC;

Le code sélectionne les villes (Ville) et compte le nombre d'adhérents (COUNT(ID)) pour chaque ville à partir de la table AdherentsUniques. Les résultats sont regroupés par ville (GROUP BY Ville) et triés par le nombre d'adhérents en ordre décroissant (ORDER BY NombreAdherents DESC).

#### Moyenne d’âge des adhérents

Faisable à l’aide de L’âge des adhérents

SELECT Avg(Age) AS MoyenneAge

FROM AgesAdherents;

Le code sélectionne la moyenne des âges (Avg(Age)) des adhérents à partir de la table AgesAdherents et renomme cette moyenne en MoyenneAge.

#### Les montants d’adhésions

SELECT Montant FROM 2015

UNION ALL

SELECT Montant FROM 2016

UNION ALL

SELECT Montant FROM 2017

UNION ALL

SELECT Montant FROM 2018

UNION ALL

SELECT Montant FROM 2019

UNION ALL

SELECT Montant FROM 2020

UNION ALL SELECT Montant FROM 2021;

C’est une jointure qui va nous aider à déterminer le montant moyen dépensé en général

#### Moyenne des montants dépensés

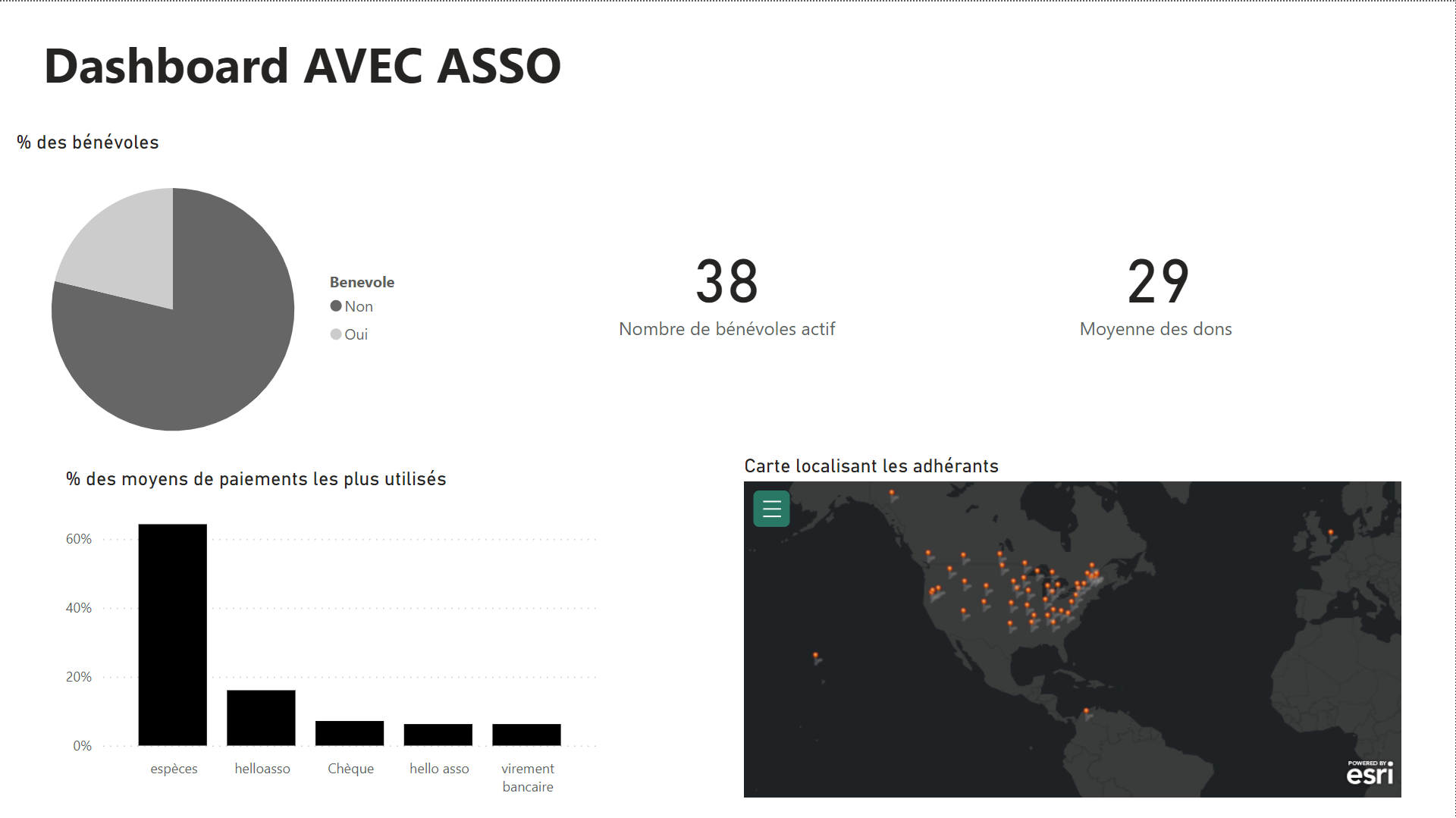
Faisable à l’aide de la requête au-dessus.

SELECT Avg(Montant) AS MontantMoyen

FROM MontantsCombinés;

 Le code sélectionne la moyenne des montants (Avg(Montant)) de la table MontantsCombinés et renomme cette moyenne en MontantMoyen.

# Datavisualisation

Pour tous ce qui est interactif, et pour plus de détails : <dashboard201.pbix>