Détection et remplacements des valeurs aberrantes

François Chaix

UNIVERSITÉ DE STRASBOURG

OffiSanté

Rapport de Stage

Détection et remplacement de valeurs aberrantes

Start up OFFISANTE SAS

Maître de stage : Nicolas Buglio

François Chaix

M1 Statistique 2013-2014

Remerciements

Je tiens à remercier Maurice Belais et Nicolas Buglio de m'avoir accueilli pour mon stage au sein de leur entreprise.

Je tiens également à remercier Anne-Sophie de la Gournerie, Mathieu Adoutte et François Marcaillou pour m'avoir accueilli chaleureusement et m'avoir conseillé tout au long de mon stage et auprès de qui j'ai beaucoup appris.

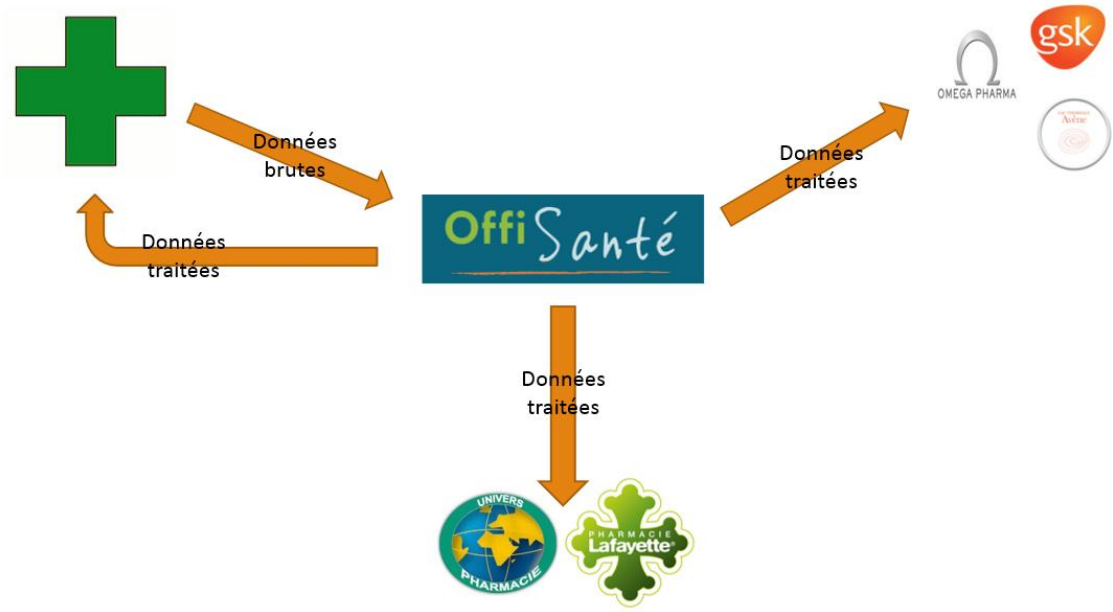
Introduction

A - Offisanté

Offisanté est une start-up jeune et dynamique spécialisée dans les analyses statistiques des données, dans l'aide à la décision et dans la transmission des informations. Créée en 2012, elle recueille aujourd'hui les données de plus de 1000 officines. Ces données sont analysées et mises à disposition pour le pilotage de la pharmacie par le pharmacien titulaire ; le pilotage du groupement auquel appartient la pharmacie ; le pilotage des campagnes marketing des laboratoires. Ces données concernent essentiellement les commandes, les ventes et les stocks de chaque officine. Offisanté propose aux groupements de pharmacies, ainsi qu'aux pharmacies individuelles, des conseils de gestion et de stratégies à adopter pour augmenter leurs marges. La plateforme Offisanté met à disposition un ensemble de tableaux de bords sur les principaux laboratoires fournisseurs selon les catégories de produits, sur les suivis de campagnes pour un produit spécifique, et bien d'autres domaines.

L'équipe d'Offisanté est constitué d'un président qui met en œuvre le développement commercial de l'entreprise. Il fait la présentation, la négociation et la vente du service auprès des clients pharmaciens, groupement et laboratoires.
Son associé (mon maître de stage) assure le développement technique et statistique de la base de données.
La responsable du service clients s'occupe des relations entretenues avec les clients déjà inscrits ou en cours d'inscription. De plus, cette personne conçoit des tableaux de bilans utiles aux dirigeants de groupements de pharmacies et aux pharmaciens pour choisir quel produit acheter et où se fournir.
Deux chefs de projet assurent le développement informatique de l'entreprise. Un chef de projet travaille sur l'élaboration de tableaux de bords statistiques selon les besoin du pharmacien ou du groupement. L'autre chef de projet travaille quant à lui sur la gestion et l'organisation de la base de données.

Chaque pharmacie partenaire dispose dans son LGO (Logiciel de Gestion d'Officine) d'un extracteur de données, envoyant les données quotidiennement et automatiquement. Les trois principaux LGO sont LGPI, Winpharma et Alliadis.


Données brutes
Données traitées
Données traitées
Données traitées
OffiSanté
OMEGA PHARMA
gsk
UNIVERS PHARMACIE
PHARMACIE Lafayette

B - Data Quality sur le prix d'achat des produits

Les enjeux d'une bonne qualité des données sont très importants pour donner du sens aux traitements statistiques des données. Cette étude d'amélioration de data quality est primordiale pour la pertinence des analyses en essayant de redresser les valeurs extrêmement aberrantes autour de la moyenne ou de la médiane selon les types de produits. La valeur d'une entreprise comme Offisanté dépend de la qualité des données qu'elle fournit aux groupements et aux laboratoires. Plus les données sont fiables, plus l'entreprise gagne en valeur et en crédibilité auprès des clients. En effet, lors de l'élaboration d'un tableau de bord sur les marges d'une officine, les résultats sont complètement faussés si on tient compte d'un prix d'achat de plusieurs millions d'euros sur un produit ne valant en réalité que quelques euros. C'est pourquoi j'ai essayé de détecter les valeurs aberrantes pour ensuite les remplacer.

Tout au long de mon stage, j'ai donc travaillé sur l'amélioration de la data quality d'une table d'une base de données.

Tout d'abord, j'ai tenté de détecter les valeurs aberrantes sur les prix d'achats remisés des produits au sein des pharmacies. Ces erreurs peuvent provenir de plusieurs causes (erreurs de saisie des données de la part du pharmacien, mauvaise traduction depuis l'extracteur de données, …).

Dans un second temps, j'ai essayé de remplacer mes valeurs aberrantes par les valeurs les plus adéquates possibles selon les conditions que présente chaque ligne de commande comportant une valeur aberrante.

Enfin, j'ai automatisé ce processus pour qu'il soit affecté sur toutes les données rentrant chaque jour dans les bases d'Offisanté.

I. Détection des valeurs aberrantes

A - Méthode de détection calculatoire des valeurs aberrantes

Pour détecter les valeurs aberrantes au sein du jeu de données étudié, j'ai utilisé une méthode statistique simple basée sur l'étendue des prix d'achats remisés de chaque produit autour de la moyenne. Le prix d'achat remisé est le prix unitaire auquel les pharmaciens achètent un produit après un calcul éventuel suite à une offre promotionnelle faite par le fournisseur. En effet, le pharmacien bénéficie quelques fois d'unité(s) gratuite(s) lorsqu'il commande un produit

J'ai donc effectué ces recherches dans la table répertoriant toutes les lignes de commandes de chaque pharmacie présente dans la base de données.

Pour optimiser la qualité des résultats, j'ai réduit mes recherches à toutes les commandes depuis le 1er janvier 2013. Tout d'abord, j'ai calculé la moyenne ainsi que l'écart-type du prix d'achat remisé pour chaque produit étant répertorié dans cette table des lignes de commandes. J'ai fixé une valeur de rejet inférieure (respectivement une valeur de rejet supérieure) égale à Moyenne-3*Ecart-type (respectivement Moyenne+3*Ecart-type). Cela m'a permis d'extraire un nombre important de valeurs suspectes (plus d'un million soit 2% des lignes de commande étudiées). J'ai donc créé une nouvelle table en y introduisant toutes ces valeurs aberrantes détectées.

En utilisant cette méthode, je me suis rendu compte que plusieurs pharmacies se trouvant dans cette nouvelle table se procuraient un bon nombre de produits à un prix plus élevé que la moyenne mais identique d'une pharmacie à une autre. En investiguant, j'ai appris que ces pharmacies se situaient toutes à la Réunion, ce qui expliquait ce prix supérieur à la moyenne. J'ai donc tenu compte de cet élément important pour réévaluer ma table de valeurs aberrantes en distinguant bien les pharmacies se situant à la Réunion de celles se situant en France métropolitaine.

Ensuite, il m'a fallu introduire une nouvelle notion pour classer ces valeurs de façon plus ou moins aberrante. J'ai donc introduit quatre critères : 'Très aberrant', 'Aberrant', 'Peu aberrant' et 'Non aberrant'. Pour déterminer immédiatement s'il s'agit d'une erreur de saisie de la part du pharmacien (la saisie d'un code à sept chiffres dans le champ du prix du produit entraine des erreurs considérables sur l'analyse des données) j'ai labellisé de 'Très aberrantes' toutes les valeurs étant au moins cinq fois supérieures à la moyenne ainsi que celles inférieures à un cinquième de la moyenne. Ensuite, j'ai qualifié de 'Aberrantes' toutes les valeurs se situant entre deux et cinq fois la moyenne et celles se situant entre un cinquième et la moitié de la moyenne. De plus, j'ai qualifié de 'Non aberrant' toutes les lignes de commande ayant une quantité commandée égale aux UG (unités gratuites) admettant un prix d'achat remisé égal à 0 (ce qui ne présente aucun signe de valeur suspecte). Enfin j'ai nommé de 'Peu aberrantes' toutes les valeurs restantes.

De plus j'ai introduit une nouvelle variable appelée 'Ecart' calculant l'écart entre le prix d'achat et la moyenne du prix du produit.

```sql
SELECT
          A.[ID_LCMD]
          ,A.[ID_Commande]
          ,A.[ID_PHARMACIE]
          ,Z.CP
          ,A.[ID_Date]
          ,A.[ID_PRODUIT]
          ,D.[Libelle Produit]
          ,A.[Quantité_commandé]
          ,convert(float,[Prix_achat_remisé]) as Valeur_aberrante
          ,"Lib_val_ab"=
              case
                  when Prix_achat_remisé=0 and Quantité_commandé=UG then 'Non aberrant'
                  when convert(float, abs([Prix_achat_remisé]))>5*moyenne_prix_produit or (Prix_achat_remisé=0 and Quantité_commandé!=UG) or convert(float,abs([Prix_achat_remisé]))<B.moyenne_prix_produit/5 then 'Très aberrant'
                  when (convert(float,abs([Prix_achat_remisé])) between 2*moyenne_prix_produit and 5*moyenne_prix_produit) or (convert(float,abs([Prix_achat_remisé])) between B.moyenne_prix_produit/5 and B.moyenne_prix_produit/2) then 'Aberrant'
                  else 'Peu aberrant'
              end
          ,convert(float,abs([Prix_achat_remisé]-B.moyenne_prix_produit)) as ecart
          ,A.[ID_Fournisseur]
          ,F.[Lib_Taux_TVA]
          ,B.moyenne_prix_produit
          ,A.[UG]
          ,B.ecart_prix_produit
          ,convert(float,B.val_rejet_inf) as val_rejet_inf
          ,convert(float,B.val_rejet_sup) as val_rejet_sup

FROM [OFFISANTE_DM].[dbo].F_Commande A

inner join [OFFISANTE_DM].[dbo].[D_Produit] D on D.ID_PRODUIT=A.ID_PRODUIT

inner join OFFISANTE_DM.dbo.D_Pharmacie Z on A.ID_PHARMACIE=Z.ID_PHARMACIE

inner join [OFFISANTE_DM].[dbo].[D_TVA] F on F.[Lib_Taux_TVA]=D.[TAUX_TVA_BCB]

inner join(
          SELECT [ID_PRODUIT]
              ,convert(float,avg([Prix_achat_remisé])) as moyenne_prix_produit
              ,isnull(stdev([Prix_achat_remisé]),0) as ecart_prix_produit
              ,convert(float,avg([Prix_achat_remisé])+3*isnull(stdev([Prix_achat_remisé]),0)) as val_rejet_sup
              ,convert(float,avg([Prix_achat_remisé])-3*isnull(stdev([Prix_achat_remisé]),0)) as val_rejet_inf
          FROM [OFFISANTE_DM].[dbo].[F_Commande] X
          inner join OFFISANTE_DM.dbo.D_Pharmacie Y on X.ID_PHARMACIE=Y.ID_PHARMACIE
          WHERE convert(float,[Prix_achat_remisé])>0 and Y.CP like '97%'
          GROUP BY [ID_PRODUIT]
          ) B on B.[ID_PRODUIT]=A.[ID_PRODUIT]

where Quantité_commandé!=0
    and ID_Date>=20130000
    and Z.CP like '97%'
    and (convert(float,[Prix_achat_remisé]) not between val_rejet_inf and val_rejet_sup)
```

Le script ci-dessus permet de détecter les valeurs aberrantes pour les pharmacies se situant à la Réunion. J'ai séparé ces valeurs des autres en introduisant le code postal de la pharmacie (code postal des DOM TOM commençant par 97). J'ai donc sélectionné toutes les commandes aberrantes dont le code postal commence par 97.

La détection des commande aberrantes en métropole s'effectue avec le même script mais en prenant les pharmacies dont le code postal ne commence pas par 97.

B - Représentation graphique des résultats

Afin de visualiser les résultats que j'ai obtenus, j'ai utilisé le logiciel de Business Intelligence MicroStrategy.

Pour ce faire, j'ai créé une fonction SQL prenant en entrée deux variables : un identifiant de produit et la pharmacie associée présentant une valeur suspecte au sein des lignes de commande. Pour optimiser la rapidité d'exécution de cette requête, j'ai calculé pour le produit étudié le nombre de commandes par mois effectuées sur l'ensemble de toutes les pharmacies pour chaque prix d'achat. De plus j'ai calculé l'écart entre chaque prix et la moyenne du produit

```
declare @id_produit int
set @id_produit=70395

declare @id_pharmacie int
set @id_pharmacie=263

select *, row_number() over(order by flux) as num_ligne from
(
select distinct A.Prix_achat_remisé
,count(A.Prix_achat_remisé) as nbre_lcmd
,C.Année
,C.Mois
,abs(A.Prix_achat_remisé-D.moyenne) as Ecart
,E.[Libelle Produit]
,-1 as Flux
from OFFISANTE_DW.dbo.tbl_LCMD A
inner join OFFISANTE_DW.dbo.tbl_CMD B on A.ID_Commande=B.ID_Commande
inner join OFFISANTE_DW.dbo.D_PRODUIT E on E.ID_PRODUIT=A.ID_PRODUIT
inner join(select ID_Commande
,left(ID_Date,4) as Année
,right(left(Id_date,6),2) as Mois
  from OFFISANTE_DW.dbo.tbl_CMD) C on C.ID_Commande=A.ID_Commande
inner join(select ID_PRODUIT
,avg(Prix_achat_remisé) as moyenne
  from OFFISANTE_DW.dbo.tbl_LCMD
  group by ID_PRODUIT) D on D.ID_PRODUIT=A.ID_PRODUIT
where A.ID_PRODUIT=@id_produit and A.Quantité!=0 and B.ID_Date is not null and B.ID_Date>20130000
group by E.[Libelle Produit], A.Prix_achat_remisé, C.Année, C.Mois, moyenne

Union

select X.Prix_achat_remisé
,1 as nbre
,left(Y.ID_Date,4) as Année
,right(left(Y.ID_Date,6),2) as mois
,abs(X.prix_achat_remisé-Z.moyenne) as Ecart
,V.[Libelle Produit]
,X.ID_PHARMACIE
from OFFISANTE_DW.dbo.tbl_LCMD X
inner join OFFISANTE_DW.dbo.tbl_CMD Y on X.ID_Commande=Y.ID_Commande
inner join OFFISANTE_DW.dbo.D_PRODUIT V on V.ID_PRODUIT=X.ID_PRODUIT
inner join(select ID_PRODUIT
,avg(Prix_achat_remisé) as moyenne
  from OFFISANTE_DW.dbo.tbl_LCMD
  group by ID_PRODUIT) Z on Z.ID_PRODUIT=X.ID_PRODUIT
where X.Quantité!=0 and Y.ID_Date is not null and Y.ID_Date>20130000
and X.ID_PRODUIT = @id_produit
and X.id_pharmacie = @id_pharmacie
) V_union
```

Pour cet exemple, on va représenter les prix d'achats pour le produit 70935 pour toutes les pharmacies depuis le 1er janvier 2013 et différencier les commandes de la pharmacie 263 des commandes des autres pharmacies grâce à la notion de flux.

Ensuite, j'ai pu exécuter cette requête dans l'interface de MicroStrategy. J'ai donc importé toutes les données utiles à la représentation graphique de la répartition du prix d'achat du produit.

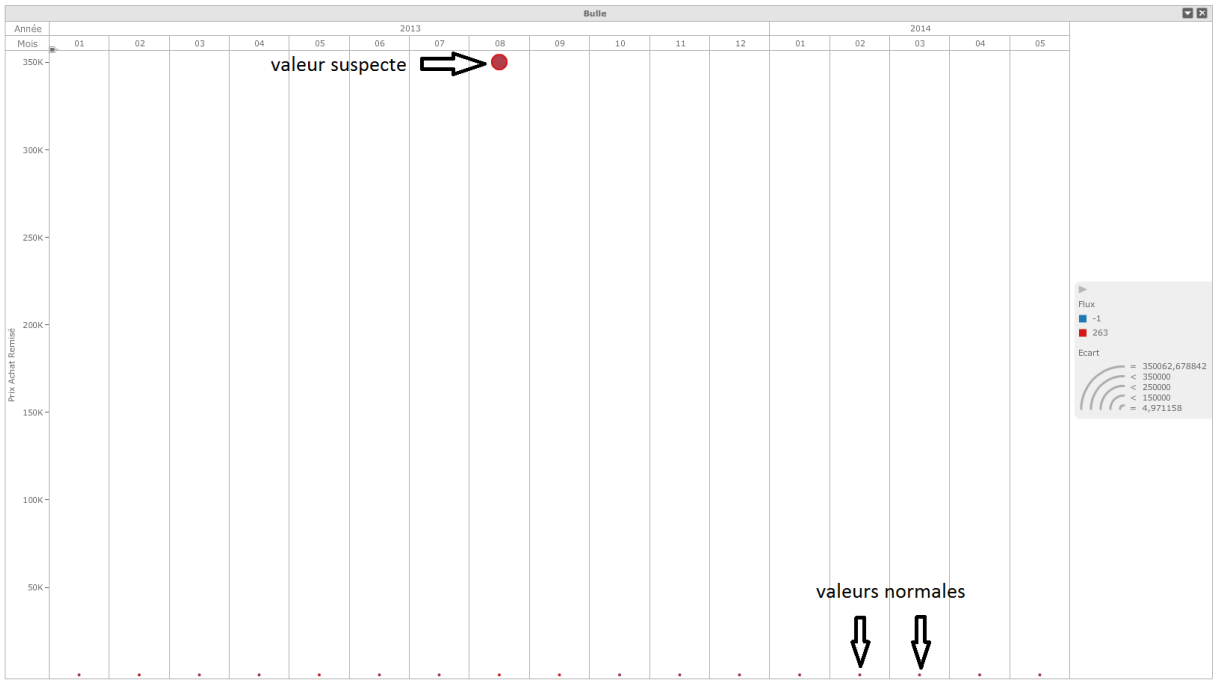
Prix Achat Remisé	Nbre Lcmd	Année	Mois	Ecart	Libelle Produit	Flux	Num Ligne
0	1	2013	12	6	ACIDE FOLIQUE 0,4MG CCD CPR 30	-1	1
0	1	2014	07	53305	ACIDE FOLIQUE 0,4MG CCD CPR 30	-1	2
1	1	2013	08	6	ACIDE FOLIQUE 0,4MG CCD CPR 30	-1	3
1	1	2014	06	53305	ACIDE FOLIQUE 0,4MG CCD CPR 30	-1	4
2	1	2014	04	5	ACIDE FOLIQUE 0,4MG CCD CPR 30	-1	5
1	2	2013	11	53305	ACIDE FOLIQUE 0,4MG CCD CPR 30	-1	6
77	2	2014	02	5	ACIDE FOLIQUE 0,4MG CCD CPR 30	-1	7
1	3	2014	05	33305	ACIDE FOLIQUE 0,4MG CCD CPR 30	-1	8

Ensuite on choisit si les variables importées sont des mesures (colonnes jaunes) ou des attributs (colonnes bleues) :

En exécutant cette requête, MicroStrategy construit un cube et y insert toutes les données importées par la requête, ce qui permet un accès simple et rapide à toutes ces données.

Une fois le cube créé, je peux commencer à construire le graphique de la répartition du prix.

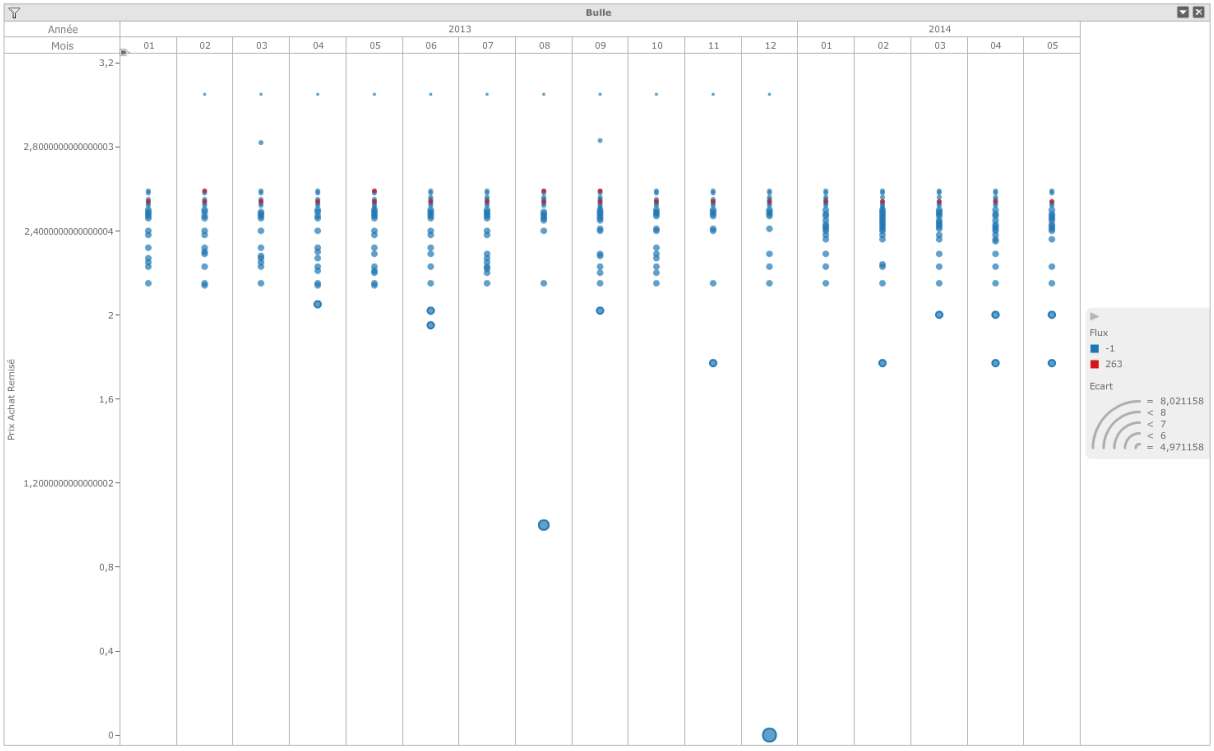
Je trace donc la répartition du prix d'achat en fonction du temps (année et mois) :



Pour faire apparaître clairement la ou les valeur(s) suspecte(s), j'ai décidé d'augmenter la taille des points en fonction de l'écart à la moyenne (plus l'écart est important, plus la taille du point est importante). De plus, j'ai coloré toutes les commandes correspondant à la pharmacie étudiée.

Ainsi, on voit clairement que ce médicament présente une valeur très suspecte pour le mois d'août 2013.

Je décide ensuite d'exclure la valeur suspecte détectée sur le graphique, et MicroStrategy retrace instantanément le nouveau graphe.



Ainsi, on peut voir que les différents prix de ce produit sont concentrés et alignés sur des paliers relativement proches. On remarque de plus qu'il y a à nouveau quelques valeurs qui semblent suspectes.

II. Remplacement des valeurs aberrantes

Toutes les valeurs suspectes étant détectées, il faut maintenant élaborer une technique pour remplacer ces valeurs en suivant des règles bien précises selon les cas spécifiques de chaque ligne de commande. Par exemple, pour tous les produits ayant un taux de TVA égal à 2.1% (TVA correspondant aux médicaments) le prix d'achat et le prix de vente sont fixés à l'échelle nationale. C'est pourquoi le redressement de la valeur aberrante se fera autour de la médiane du prix d'achat. Pour des produits dont le prix d'achat et de vente n'est pas fixé d'une pharmacie à une autre, le redressement s'effectuera plutôt autour de la moyenne du prix d'achat.

A - Conditions sur les lignes de commandes

J'ai donc fixé plusieurs conditions pour essayer de répertorier tous les cas envisageables.

Tout d'abord, j'ai séparé les pharmacies se situant à la Réunion de celles se situant en France métropolitaine :

- A1 : pharmacie en France métropolitaine
- A2 : pharmacie à la Réunion

```
select ID_LCMD
    ,"id_condition"=
        case when CP not like '97%' then 'A1'
             when CP like '97%' then 'A2'
        end

from
 OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande
```

Ensuite, la condition B est portée sur le nombre de commande(s) d'un produit au sein d'une même pharmacie ne présentant pas de valeur aberrante sur une durée précise (entre 2 mois avant et 2 mois après la commande présentant la valeur suspecte) :

- B1 : au moins une commande sans valeur aberrante
- B2 : aucune commande sans valeur aberrante

```sql
select A.ID_LCMD
    ,"id_condition"=
        case when B.Nbre_LCMD_clean>=1 then 'B1'
             when B.Nbre_LCMD_clean=0 then 'B2'
        end

from
 OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande A

 inner join (
            select distinct Z.ID_LCMD
                  ,X.ID_PRODUIT
                  ,X.ID_PHARMACIE
                  ,count(*) as Nbre_LCMD_clean
            from OFFISANTE_DM.dbo.F_Commande X
            inner join OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande Z on X.id_pharmacie=Z.id_pharmacie and X.id_produit=Z.id_produit
            left join offisante_DM.dbo.Tbl_Log_Erreur_Commande Y on X.Id_LCMD=Y.ID_LCMD
            where 1=1
                and Y.ID_LCMD is null
                and X.id_date>20130000
                and X.id_date<Z.id_date+200 and X.id_date>Z.id_date-200
            group by X.id_produit,X.id_pharmacie,Z.ID_LCMD

            union (

            select distinct WW.id_lcmd
                ,W.ID_PRODUIT
                ,W.ID_PHARMACIE
                ,"Nbre_LCMD_clean"='0'
            from OFFISANTE_DM.dbo.F_Commande W
            inner join OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande WW on W.ID_LCMD=WW.Id_LCMD

            left join
                    (select distinct Z.ID_LCMD
                            ,X.ID_PRODUIT
                            ,X.ID_PHARMACIE
                            ,count(*) as Nbre_LCMD_clean
                    from OFFISANTE_DM.dbo.F_Commande X
                    inner join OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande Z on X.id_pharmacie=Z.id_pharmacie and X.id_produit=Z.id_produit
                    left join offisante_DM.dbo.Tbl_Log_Erreur_Commande Y on X.Id_LCMD=Y.ID_LCMD
                    where 1=1
                        and Y.ID_LCMD is null
                        and X.id_date>20130000
                        and X.id_date<Z.id_date+200 and X.id_date>Z.id_date-200
                    group by X.id_produit,X.id_pharmacie,Z.ID_LCMD
                   ) R on WW.ID_lcmd=R.id_lcmd
            where R.ID_LCMD is null
            )

            ) B on A.id_lcmd=B.id_lcmd
```

Dans la condition C, j'ai différencié tous les cas possibles mettant en relation la quantité commandé et le nombre d'UG (unité gratuite) :

- C1 : Quantité_commandée=UG
- C2 : UG=0
- C3 : Quantité_commandée<UG
- C4 : UG>0 et Quantité_commandée>UG

```sql
select ID_LCMD
    ,"id_condition"=
        case when Quantité_commandé=UG then 'C1'
             when UG=0 then 'C2'
             when Quantité_commandé<UG then 'C3'
             when Quantité_commandé>UG and UG>0 then 'C4'
        end

from
 OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande
```

Dans la condition D, j'ai répertorié les différents taux de TVA :

- D0 : Taux de TVA à 0%
- D1 : Taux de TVA à 2,1%
- D2 : Taux de TVA à 5,5%
- D3 : Taux de TVA à 10%
- D4 : Taux de TVA à 20%

```
select ID_LCMD
    ,"id_condition"=
        case when Lib_Taux_TVA=0 then 'D0'
             when Lib_Taux_TVA=2.1 then 'D1'
             when Lib_Taux_TVA=5.5 then 'D2'
             when Lib_Taux_TVA=10 then 'D3'
             when Lib_Taux_TVA=20 then 'D4'
        end

from
 OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande
```

Enfin, dans condition E, j'ai séparé les lignes de commande selon le libellé de la valeur aberrante.

- E1 : 'Non Aberrant'
- E2 : 'Très aberrant'
- E3 : 'Aberrant'
- E4 : 'Peu aberrant'

```
select ID_LCMD
    ,"id_condition"=
        case when Lib_val_ab='Non aberrant' then 'E1'
             when Lib_val_ab='Très aberrant' then 'E2'
             when Lib_val_ab='Aberrant' then 'E3'
             when Lib_val_ab='Peu aberrant' then 'E4'
        end

from
 OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande
```

J'ai donc inséré toutes les lignes obtenues en exécutant chacun des scripts précédents.

Chaque ligne de commande présente donc cinq conditions.

B - Règles mises en place

Une fois les conditions fixées, j'ai établi un système de règles bien précises selon les conditions que présente chaque ligne de commande. Je leur ai donné un identifiant selon leur priorité (plus la priorité est élevée, plus l'identifiant est petit). Les règles sont les suivantes :

- R0 : C1 ∩ (D2 U D3 U D4)
- R1 : A2 ∩ D1
- R2 : A1 ∩ D1
- R3 : A2 ∩ B1 ∩ C4 ∩ (D2 U D3 U D4)
- R4 : A2 ∩ B2 ∩ C4 ∩ (D2 U D3 U D4)
- R5 : A1 ∩ B1 ∩ C4 ∩ (D2 U D3 U D4)
- R6 : A1 ∩ B2 ∩ C4 ∩ (D2 U D3 U D4)
- R7 : A2 ∩ B1 ∩ (C2 U C3) ∩ (D2 U D3 U D4)
- R8 : A2 ∩ B2 ∩ (C2 U C3) ∩ (D2 U D3 U D4)
- R9 : A1 ∩ B1 ∩ (C2 U C3) ∩ (D2 U D3 U D4)
- R10 : A1 ∩ B2 ∩ (C2 U C3) ∩ (D2 U D3 U D4)

```
select A.ID_LCMD
    ,"id_regle"=
            case when A.Quantité_commandé=A.UG and A.Lib_Taux_TVA!=2.1 then '0'
                 when A.CP like '97%' and A.Lib_Taux_TVA=2.1 then '1'
                 when A.CP not like '97%' and A.Lib_Taux_TVA=2.1 then '2'
                 when A.CP like '97%' and B.Nbre_LCMD_clean>=1 and (A.Quantité_commandé>A.UG and A.UG>0) and A.Lib_Taux_TVA!=2.1 then '3'
                 when A.CP like '97%' and B.Nbre_LCMD_clean=0 and (A.Quantité_commandé>A.UG and A.UG>0) and A.Lib_Taux_TVA!=2.1 then '4'
                 when A.CP not like '97%' and B.Nbre_LCMD_clean>=1 and (A.Quantité_commandé>A.UG and A.UG>0) and A.Lib_Taux_TVA!=2.1 then '5'
                 when A.CP not like '97%' and B.Nbre_LCMD_clean=0 and (A.Quantité_commandé>A.UG and A.UG>0) and A.Lib_Taux_TVA!=2.1 then '6'
                 when A.CP like '97%' and B.Nbre_LCMD_clean>=1 and (A.UG=0 or A.Quantité_commandé<A.UG) and A.Lib_Taux_TVA!=2.1 then '7'
                 when A.CP like '97%' and B.Nbre_LCMD_clean=0 and (A.UG=0 or A.Quantité_commandé<A.UG) and A.Lib_Taux_TVA!=2.1 then '8'
                 when A.CP not like '97%' and B.Nbre_LCMD_clean>=1 and (A.UG=0 or A.Quantité_commandé<A.UG) and A.Lib_Taux_TVA!=2.1 then '9'
                 when A.CP not like '97%' and B.Nbre_LCMD_clean=0 and (A.UG=0 or A.Quantité_commandé<A.UG) and A.Lib_Taux_TVA!=2.1 then '10'
            end

from OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande A
```

C - Algorithmes de remplacement des valeurs aberrantes

D'un autre côté, j'ai créé un script de remplacement pour chaque règle fixée.

Pour la règle R0, règle qui s'applique dès que la quantité commandée est égale à l'UG et tous les produits ayant un taux de TVA différent de 2,1% (car les pharmaciens ne peuvent bénéficier d'UG sur les médicaments), j'ai décidé de remplacer toutes les valeurs aberrantes par 0.

```
select id_lcmd
     ,"id_regle"='0'
     ,"valeur_proposee"=convert(numeric(9,2),0)
     ,CURRENT_TIMESTAMP as "timestamp"
from OFFISANTE_DM.dbo.tbl_condition_erreur_lcmd
where id_condition='C1'
```

Pour les produits ayant un taux de TVA de 2.1% (règles 1 et 2), le prix étant fixé d'une pharmacie à une autre, j'ai décidé de remplacer la valeur aberrante par la médiane du prix d'achat du produit concerné remisé sur la période de deux mois avant et après la commande suspecte, sur l'ensemble des pharmacies en France métropolitaine ou à la Réunion selon la géo-localisation de l'officine en question.

```
select distinct X.id_lcmd
    ,"id_regle"='1'
    ,convert(numeric(9,2),b.remp_prix) as valeur_proposee
    ,CURRENT_TIMESTAMP as "timestamp"
from OFFISANTE_DM.dbo.tbl_condition_erreur_lcmd X

inner join OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande Z on X.id_lcmd=Z.id_lcmd
inner join (
            select distinct AD.ID_LCMD, R.remp_prix
            from OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande AD


            inner join(
                        select

                            V.ID_PRODUIT,
                            PERCENTILE_DISC(0.5) within group (order by v.prix_achat_remisé)
                                                 over (partition by v.id_produit) as remp_prix

                        from
                        OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande A
                        right outer join
                        OFFISANTE_DM.dbo.F_Commande V
                        on a.ID_LCMD=v.Id_LCMD
                        inner join
                        OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande A2
                        on v.Id_Produit=A2.id_produit
                        where
                        1=1
                        and A.ID_LCMD is null
                        and A2.CP like '97%'
                        and v.Quantité_commandé !=0
                        and v.id_date>A2.ID_Date-200 and v.ID_Date<A2.ID_Date+200
                      ) R on AD.id_produit=R.id_produit

        ) B on B.ID_LCMD=X.id_lcmd

where Z.CP like '97%'
    and (X.id_condition='A2'
     or X.id_condition='D1')
```

Pour les autres produits ayant un taux de TVA différent de 2,1% et pour les officines à la Réunion, et lorsque la pharmacie présentant l'anomalie admet des lignes de commande non suspectes deux mois avant et après la ligne en question (règles 3), j'ai calculé la moyenne sur cette période au sein de cette officine. Ensuite, j'ai multiplié la valeur obtenue par $\frac{Quantité_{commandée}-UG}{Quantité_{commandée}}$ pour tenir compte des offres du fournisseur.

```
select X.id_lcmd
    ,"id_regle"='3'
    ,convert(numeric(9,2),B.remp_prix*(Z.Quantité_commandé-Z.UG)/Z.Quantité_commandé) as valeur_proposee
    ,CURRENT_TIMESTAMP as "timestamp"
from OFFISANTE_DM.dbo.tbl_condition_erreur_lcmd X

inner join OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande Z on X.id_lcmd=Z.id_lcmd
inner join (select AD.ID_LCMD, R.Remp_prix
            from OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande AD

            inner join(
                        select

                            v.ID_PRODUIT,
                            avg(V.prix_achat_remisé) as Remp_prix

                        from
                        OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande A
                        right outer join
                        OFFISANTE_DM.dbo.F_Commande V
                        on a.ID_LCMD=v.Id_LCMD
                        inner join
                        OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande A2
                        on v.Id_Produit=A2.id_produit
                        where
                        1=1
                        and A.ID_LCMD is null
                        and A2.CP like '97%' and V.id_pharmacie=A2.id_pharmacie
                        and v.Quantité_commandé !=0
                        and v.id_date>A2.ID_Date-200 and v.ID_Date<A2.ID_Date+200
                        group by
                        v.ID_PRODUIT
                      ) R on AD.id_produit=R.id_produit

        ) B on B.ID_LCMD=X.id_lcmd

where Z.CP like '97%'
    and (X.id_condition='A2'
    or X.id_condition='B1'
    or X.id_condition='C4'
    or X.id_condition='D2'
    or X.id_condition='D3'
    or X.id_condition='D4')
```

Lorsque la pharmacie n'admet aucune ligne de commande sans valeur aberrante, je prends la moyenne sur la même période mais sur l'ensemble de la Réunion (règle 4). Je reprends donc le code ci-dessus en enlevant juste la condition V.id_pharmacie=A2.id_pharmacie dans la clause where de l'inner join où je calcule la moyenne.

Les règles 5 et 6 sont traitées de la même façon que les lignes 3 et 4 mais pour la France métropolitaine.

Enfin, les règles 7, 8, 9, 10 sont appliquées de la même manière que les règles précédentes mais sans que je tienne compte des UG (cas où UG=0 ou quantité_commandée<Ug ce qui n'a aucun sens).

J'ai inséré tous mes résultats dans une table. On obtient donc plusieurs valeurs de remplacement selon les règles pour chaque ligne de commande.

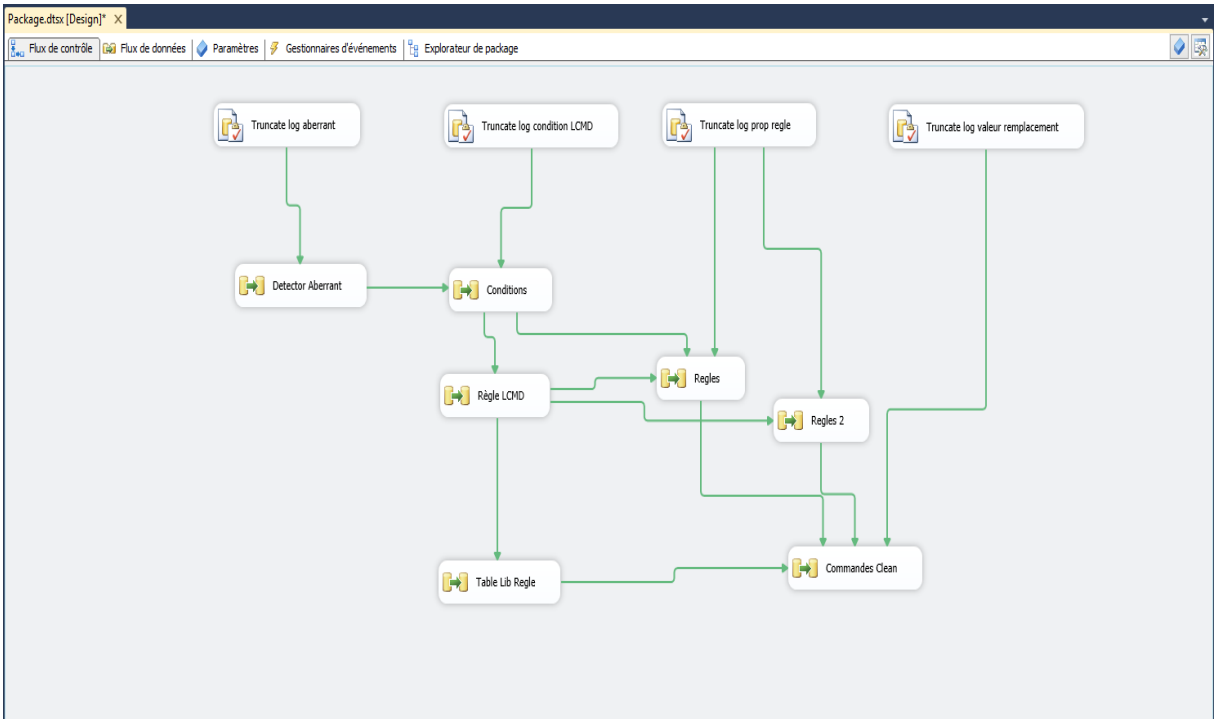
Ensuite, grâce à l'algorithme suivant, on choisit la valeur de remplacement en repérant le minimum de l'identifiant des règles affectées à chaque ligne de commande (les règles étant établies selon leur priorité). Ainsi on obtient une unique valeur de remplacement pour chaque ligne de commande.

```
select distinct t.ID_LCMD
    ,t.ID_Commande
    ,t.ID_PHARMACIE
    ,t.CP
    ,t.ID_Date
    ,t.ID_PRODUIT
    ,t.[Libelle Produit]
    ,t.Quantité_commandé
    ,t.UG
    ,t.Lib_Taux_TVA
    ,t.Valeur_aberrante
    ,t.Lib_val_ab
    ,t.ecart
    ,t.ID_Fournisseur
    ,t.moyenne_prix_produit
    ,t.ecart_prix_produit
    ,t.val_rejet_inf
    ,t.val_rejet_sup
    ,convert(numeric(9,2),V.valeur_proposee) as valeur_proposee
from OFFISANTE_DM.dbo.Tbl_Log_Erreur_Commande T

inner join(
            select prop.*
            from OFFISANTE_DM.dbo.tbl_log_prop_regle Prop
            inner join(
                        select id_lcmd
                            ,min(R.id_regle) as Rmin

                        from OFFISANTE_DM.dbo.tbl_log_prop_regle P
                        inner join [OFFISANTE_DM].[dbo].[tbl_regle] R on p.id_regle=R.id_regle

                        group by id_lcmd
                       ) s on prop.id_lcmd=s.id_lcmd and prop.id_regle=s.Rmin

          ) V on V.id_lcmd=T.ID_LCMD
```

III. Automatisation du processus de détection et de remplacement des valeurs aberrantes

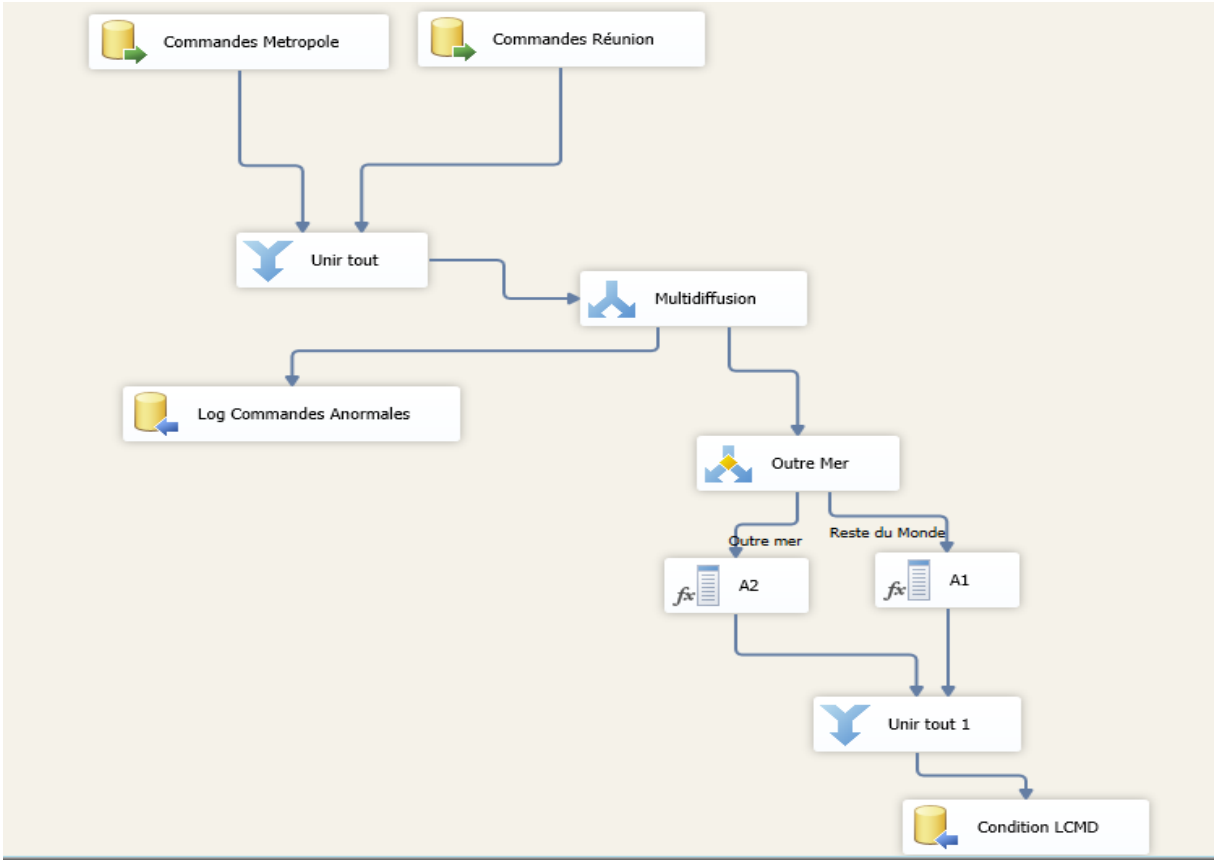
Pour utiliser le travail de détection et de remplacement des valeurs aberrantes, j'ai automatisé le processus pour le mettre en application sur les données qui sont envoyées tous les jours par les officines. Cela représente un gain de temps et de qualité de données pour l'entreprise. Pour cela, j'ai utilisé le logiciel Microsoft Visual Studio.

J'ai donc créé un projet au format SSIS en suivant pas à pas les procédures décrites dans les parties précédentes.



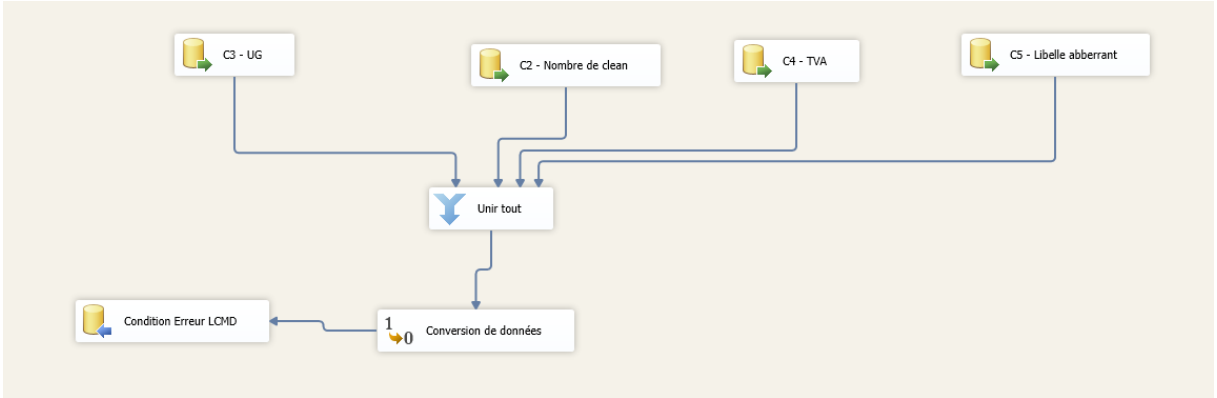
Chaque bloc représente une étape sur le traitement des valeurs aberrantes :

1) Detector Aberrant : Détection des valeurs aberrantes
2) Conditions : Conditions sur chaque ligne de commande
3) Règles LCMD : Règle sur chaque ligne de commande
4) Table Lib Regle : Donne un libellé à chaque règle
5) Regle : Exécution des scripts des règles 0, 1, 3, 4, 5, 6, 7, 8, 9, 10
6) Regle 2 : Exécution du script de la règle 2 (séparé des autres règles car il s'agit du script le plus compliqué, ce qui explique le traitement à part)
7) Commande Clean : Exécution du script choisissant la valeur parmi celles proposées pour chaque ligne de commande



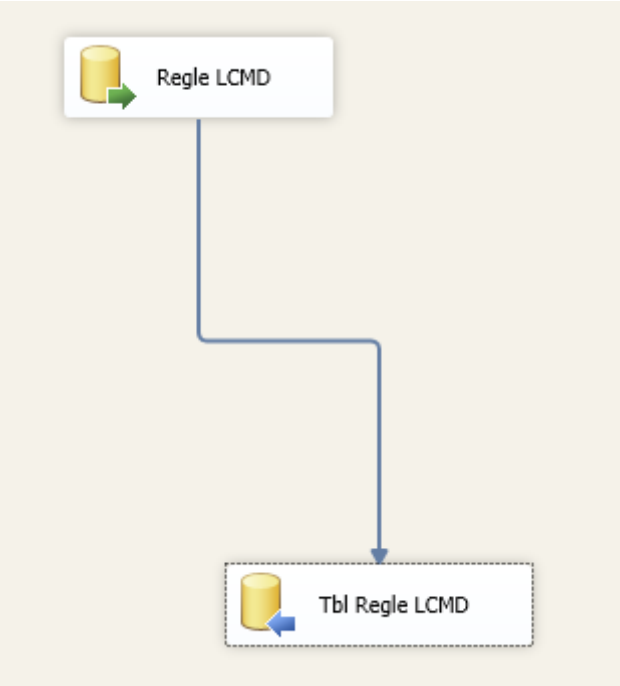
Le schéma ci-dessus représente l'intérieur du bloc Detector Aberrant. On a donc en entrée toutes les lignes de commande. Le script de détection des valeurs aberrantes est appliqué. Les valeurs suspectes partent d'un côté dans la table des valeurs aberrantes ; et de l'autre côté dans la table des conditions et on leur affecte la première condition (la localisation de la pharmacie)

Condition :



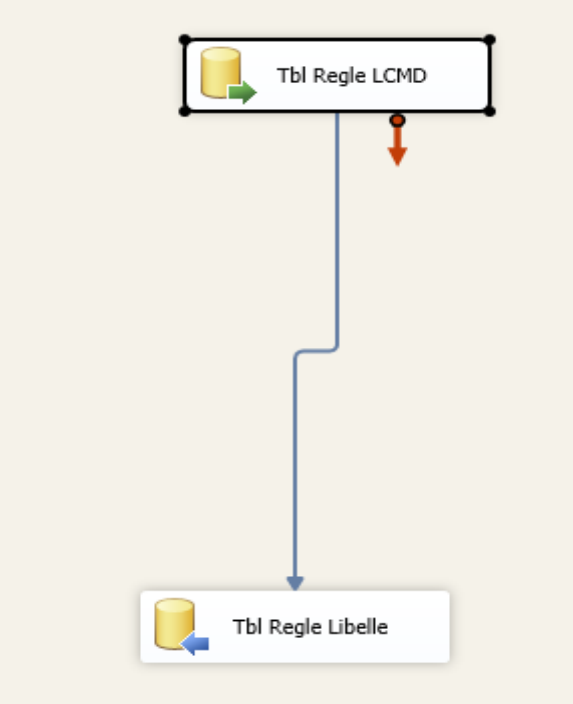
Les autres conditions sont appliquées

Regle LCMD :



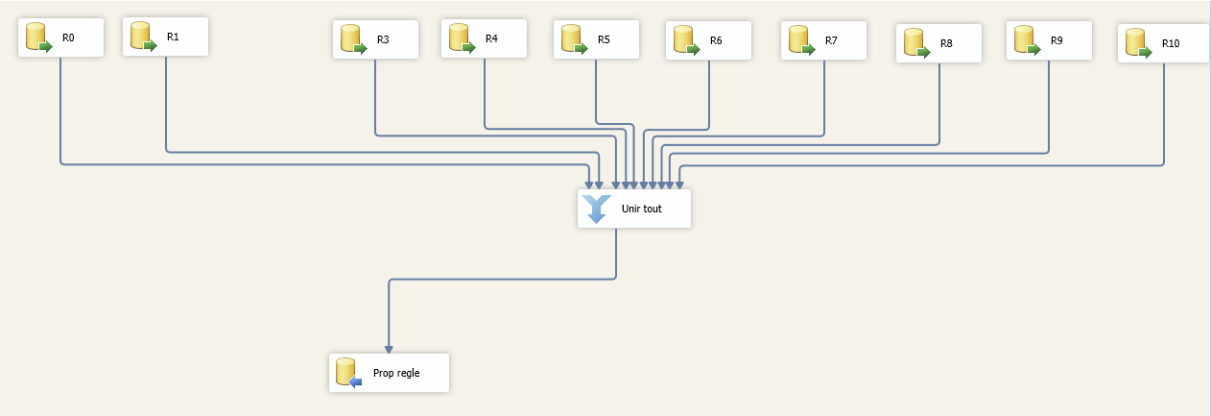
Chaque ligne de commande est ensuite traitée pour l'attribution une règle.

Table Lib Regle :



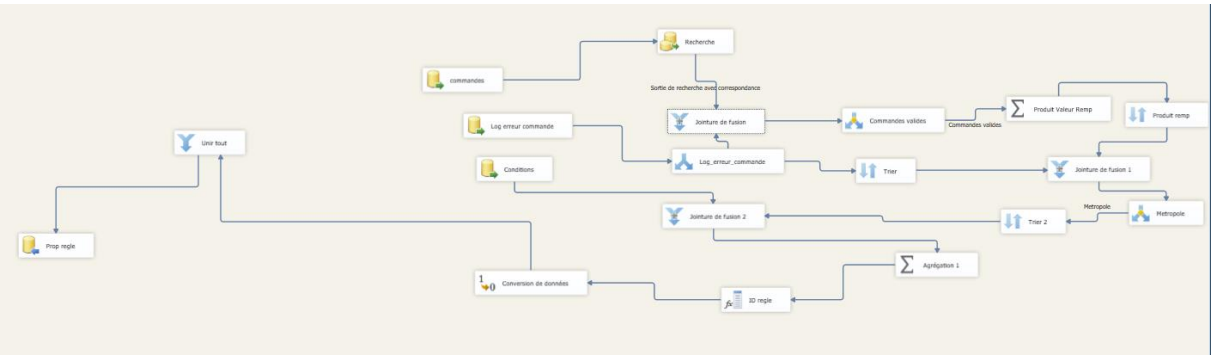
On introduit un libellé pour chaque règle utilisée.

Regle :



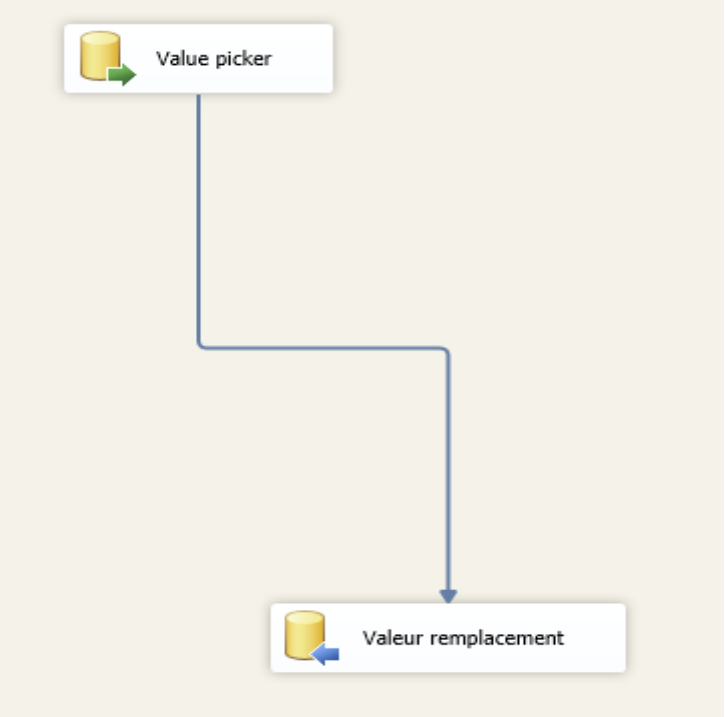
Toutes les règles (sauf la 2 qui est traitée à part pour l'optimisation du temps d'exécution de la requête) sont ensuite appliquées et on obtient une ou plusieurs valeurs de remplacement pour chaque ligne de commande

Regle 2 :



La règle 2 est appliquée.

Commande Clean :



Enfin une valeur de remplacement est choisie pour chaque ligne de commande.

A ce stade, le travail de remplacement des valeurs aberrantes est désormais terminé.

Sur un total de 1 208 728 valeurs aberrantes détectées, on obtient une valeur de remplacement pour 1 161 373 lignes de commande soit plus de 95% des valeurs à remplacer.

Conclusion

Durant ces deux mois de stage, j'ai donc travaillé sur l'amélioration de la data quality d'une table d'une base de données présentant des valeurs aberrantes (environ 2% des données traitées). Détecter les valeurs aberrantes est inévitable si on veut effectuer des analyses statistiques solides et pertinentes.

J'ai donc établi un algorithme pour repérer ces valeurs aberrantes en me basant sur la répartition autour de la moyenne du prix d'achat des produits. J'ai récupérer toutes les données ne se trouvant pas dans l'intervalle [Moyenne – 3*Ecart-type ; Moyenne + 3*Ecart-type]. J'ai pu insérer toutes ces données dans une nouvelle table pour faciliter leur traitement.

Le travail de remplacement des valeurs aberrantes s'est avéré bien plus compliqué que la détection de ces valeurs. J'ai dû créer plusieurs scripts pour proposer des valeurs de remplacement de mes valeurs suspectes détectées selon les règles établies à partir des conditions fixées. Ensuite il a fallu choisir la valeur la plus appropriée selon les caractéristiques de la ligne de commande présentant l'anomalie.

Enfin j'ai pu automatiser ce travail de détection et de remplacement des valeurs aberrantes grâce au logiciel Microsoft Visual Studio. J'ai pu établir un plan d'exécution qui analyse et détecte les valeurs aberrantes et propose aussitôt une valeur de remplacement. Cette automatisation permet l'exécution quotidienne de l'algorithme de détection et de remplacement sur les données rentrantes.

Tout au long de ce stage, j'ai donc appris à manipuler des bases de données pour améliorer la qualité des données. Grâce à ce travail de nettoyage, les analyses portées sur les marges des pharmacies et des groupements gagnent en valeur et en précision.

Table des matières