BD51-A18:
Business Intelligence
&
Data Warehouse

Sommaire

Dans le cadre de l'UV BD51, nous devions développer un système décisionnel pour la gestion des ventes par magasin pour la base de données EMODE.

Ce rapport aura pour objectif d'expliquer en détail, les différentes étapes de la conception de ce projet, à savoir la mise en œuvre des fonctions d'ETL, l'optimisation du data warehouse et enfin, le reporting.

Ce projet a été réalisé sur une machine virtuelle de type Microsoft Virtual PC sous Windows 2012. pour toute la partie ETL et pour l'optimisation du data warehouse.

Le but de ce projet est d'avoir des rapports permettant un suivi des ventes des différents magasins, dans les différents pays...

Part 1 : ETL functions implementation

1. Qualité des données

Les scripts de qualité de données se trouve en annexe 1.

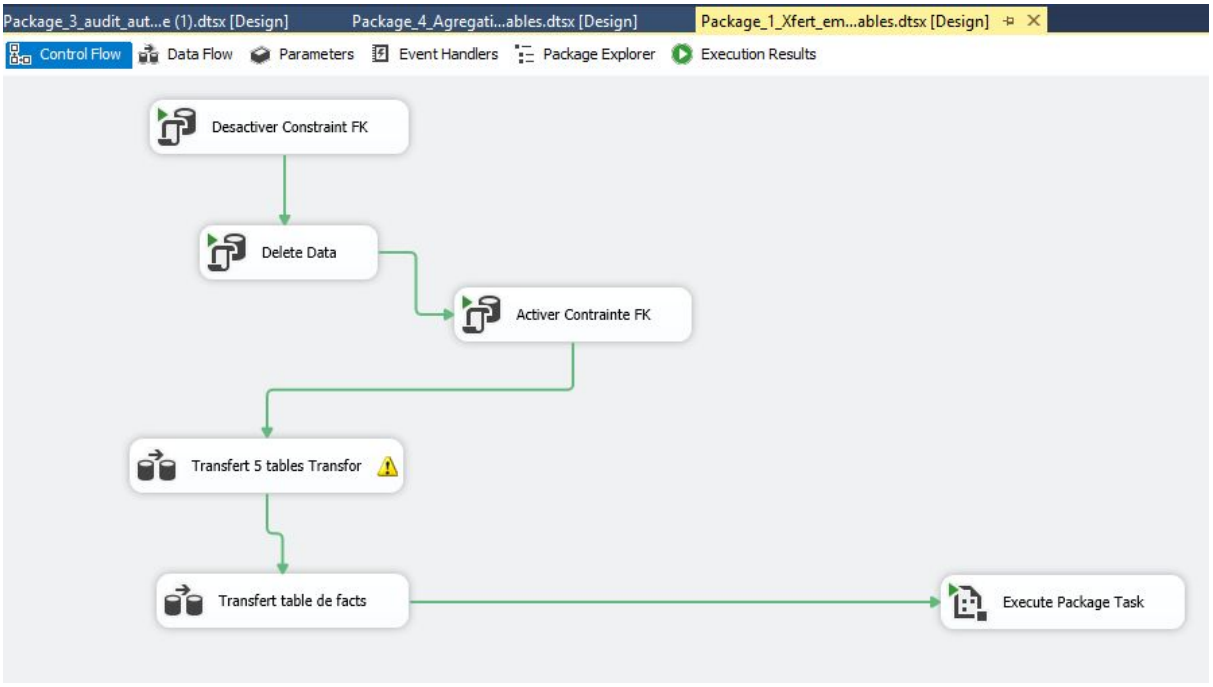
		Clé primaire en double	Clé étrangère manquante
1) Article_color_lookup	678 lignes	article_code="170016" 177264	0
2) Article_lookup	211	0	0
3) CALENDAR_YEAR_LOOKUP	262	0	0
4) OUTLET_LOOKUP	13	0	0
5)SHOP_FACTS	89171	0	0

Nous avons par la suite créer des tables de rejet, pour chaque table, contenant les données invalides.

2. Développement des packages SSIS

Package 1

Ce premier package permet le transfert de toutes les données des tables nécessaires du serveur Oracle au serveur SQL server.



Le Control Flow de ce premier page est composé de différentes parties : il y a d'abord la suppression des contraintes de clé étrangère dans toutes les tables d'Emode sur SQL Server avant de vider les données, supprimer les données de SQL Server puis réactiver les contraintes. Les données sont par la suite transférées dans les tables de dimensions puis dans les tables de faits.

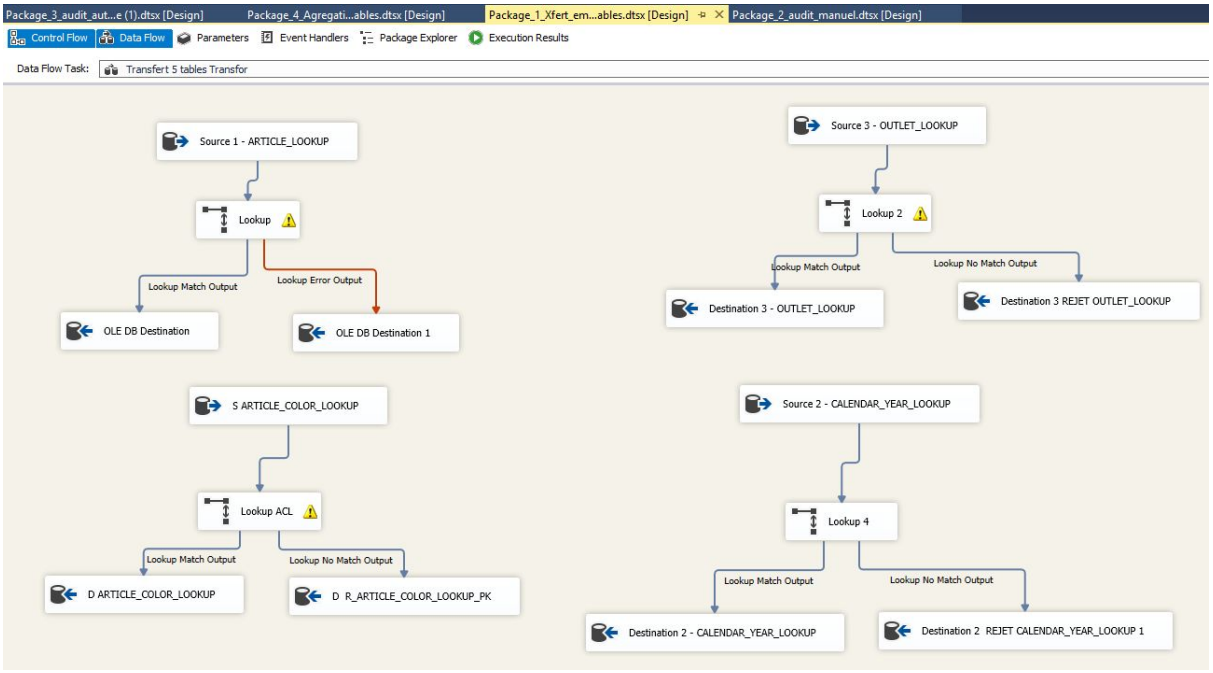


Figure - Data Flow package 1

Dans le Data Flow, pour chaque table, les données sont récupérées dans la base Emode d'Oracle et le lookup permet d'effectuer le tri dans les tables de rejet si les données

non conformes et dans la bonne table si les données sont conformes. Ce tri est effectué à l'aide d'un script SQL sur chaque lookup qui vérifie la conformité des données.

Package 2

Ce second package est le transfert incrémental des données avec un audit lorsqu'il y a une modification, une insertion ou une suppression dans la base Emode d'Oracle.

Pour ce package, nous avons dû créer Emode_inc qui contient les même tables que la base Emode mais avec une colonne supplémentaire dans chaque table : une colonne permettant de connaître l'opération effectué. Si c'est une mise à jour : "u", si c'est une insertion : "i", si c'est une suppression : "d".

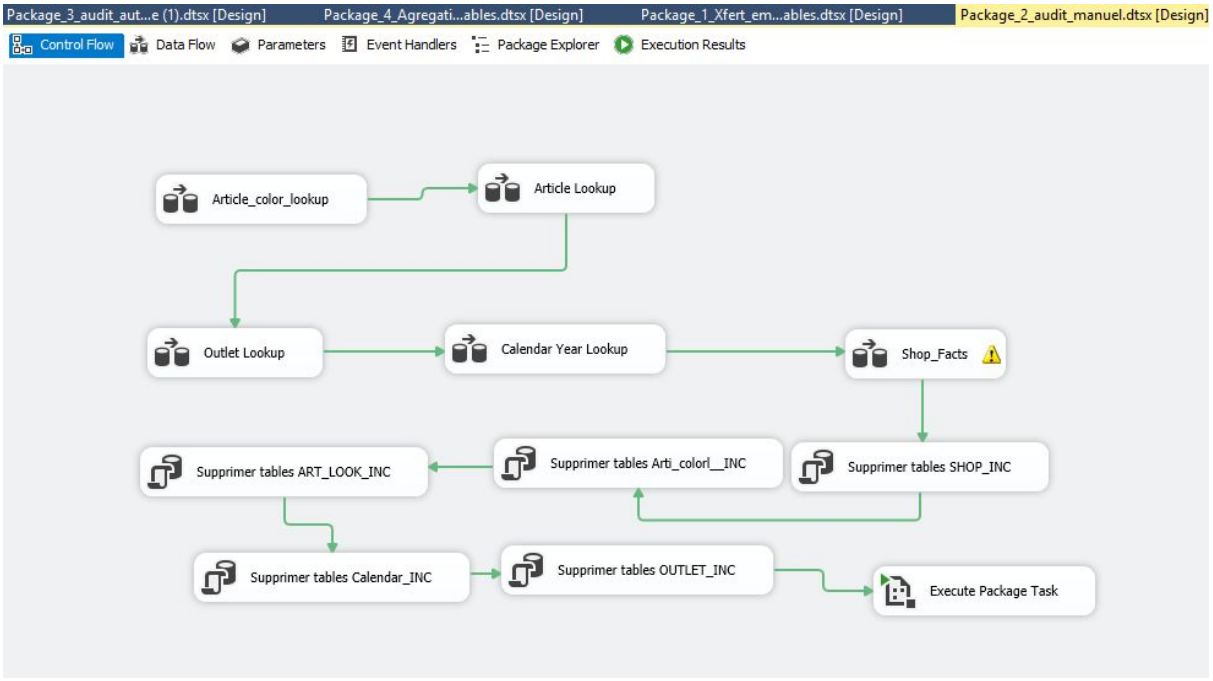


Figure - Control Flow package 2

Des triggers ont été créés pour détecter les insertions, les modifications ou les suppressions dans la base Emode pour chaque table de la base, lorsque c'est le cas : la ligne modifié est ajouté dans la base EMODE_INC avec le type d'opération effectué (u,i ou d).

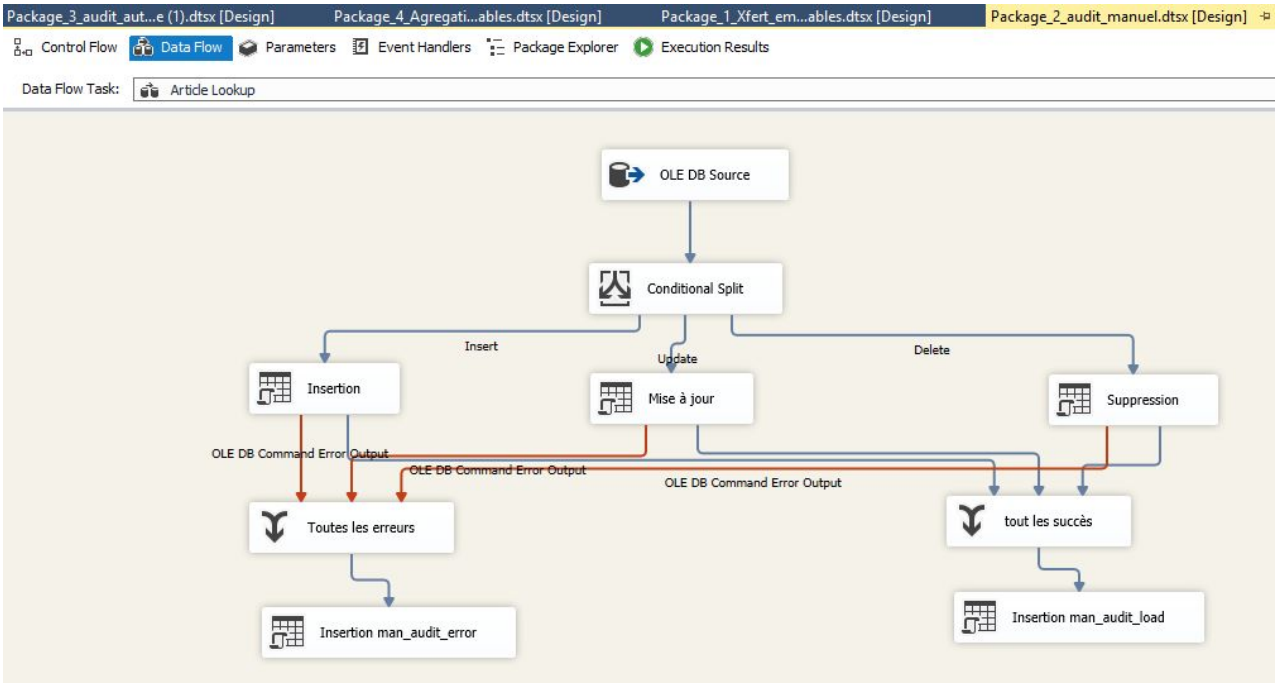


Figure - Data Flow package 2

Les données sont récupérées de la base Emode_inc, il y a ensuite un Conditional Split qui permet le tri en fonction du type d'opération effectué sur la ligne (insertion, mise à jour ou suppression).

Deux tables d'audit ont été créés : l'une permettant de tracer les opérations de chargements (avec le numéro de transfert, la date et l'heure de transfert et le statut de l'opération) et l'autre table est une table détaillé des erreurs (avec le nom de la table comprenant l'erreur, la valeur de la clé primaire mais aussi les informations à propos de la source du problème.

Dans le control Flow, les tables d'EMODE_INC sont ensuite vidées.

Package 3

Le troisième package est le même que le précédent sauf qu'ici les fonctions d'audit sont celles fournies par Integration Services.

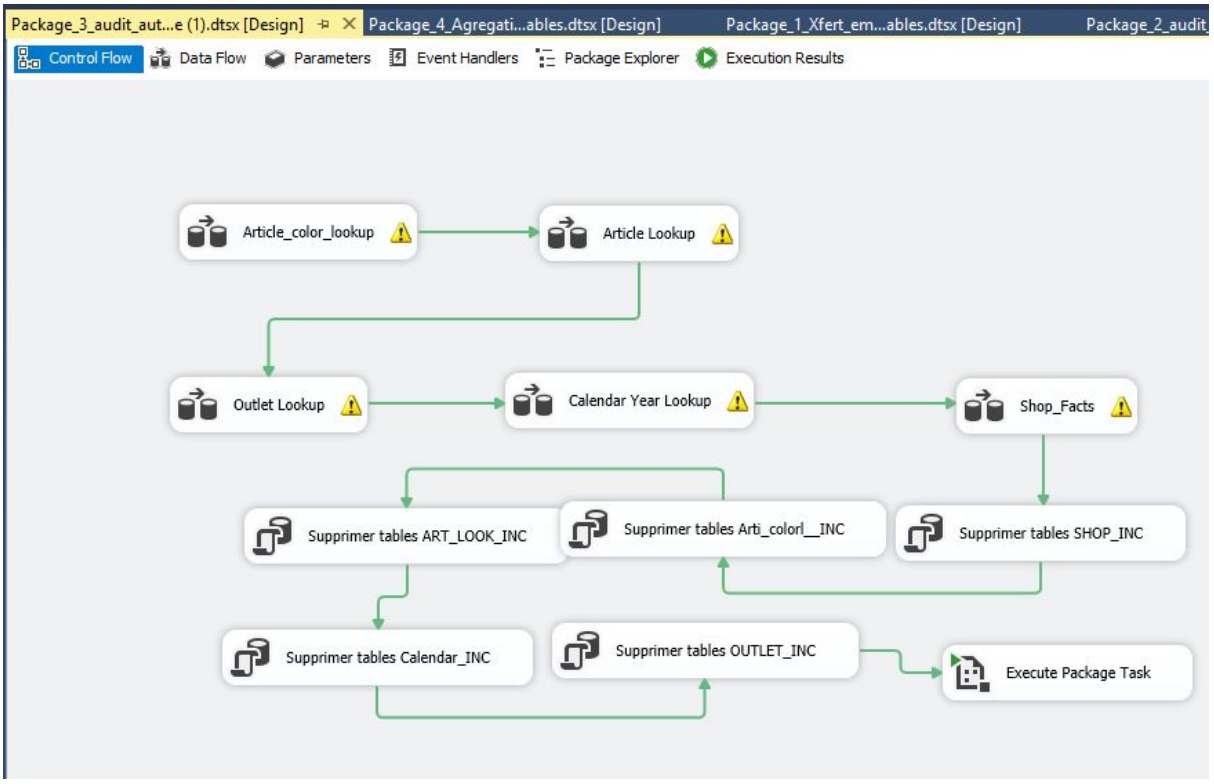
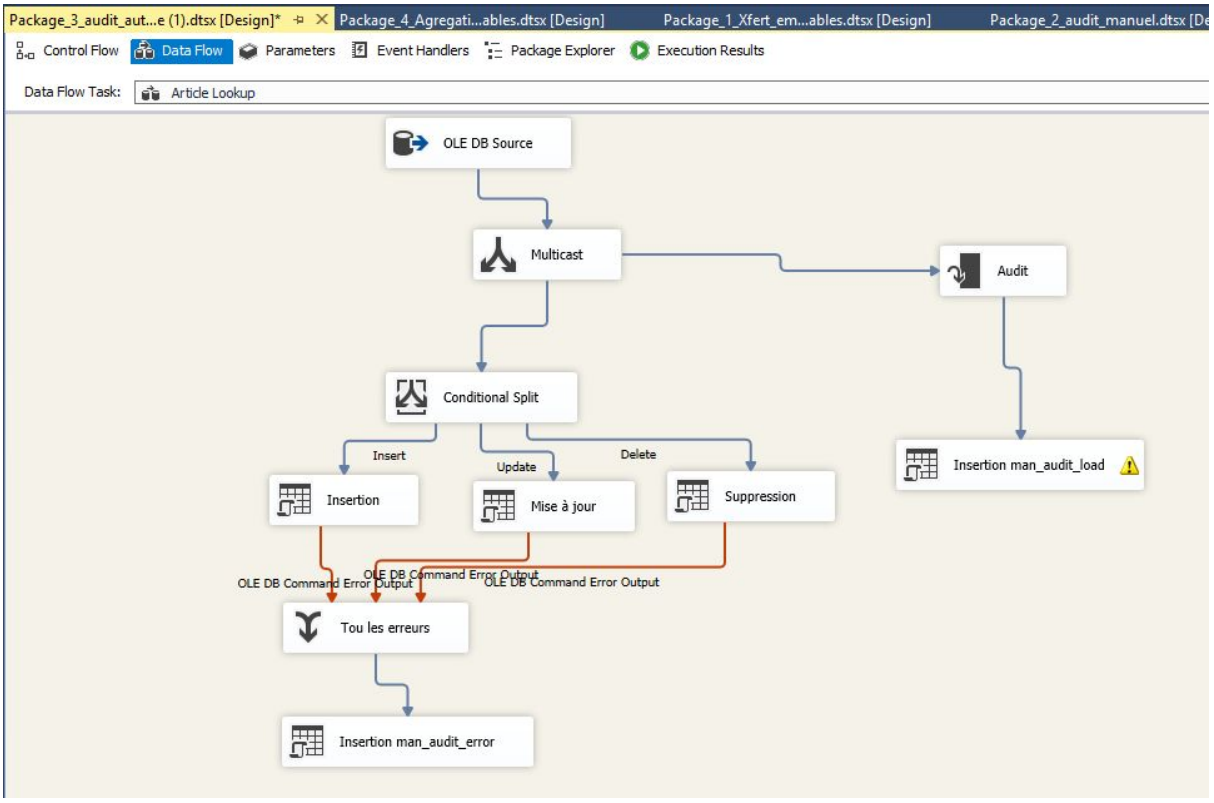


Figure - Control Flow

Le Control Flow est le même que le package précédent, la différence va être au niveau du Data Flow.



Les données sont récupérées de la base Emode_inc, l'outil "Audit" permet d'ajouter dans la table d'audit d'Emode_inc les informations d'audit (numéro d'exécution…). Multicast permet de dupliquer le flux de données vers Audit et vers le Conditional split (qui reprend par la suite les mêmes parties que le package 2).

Package 4

Le dernier package est le package la gestion des tables d'agrégation. La mise à jour des tables d'agrégation se fait par une suppression totale des données de la base Emode de SQL Server, puis d'une insertion utilisant les données actualisées dans les tables de référence.

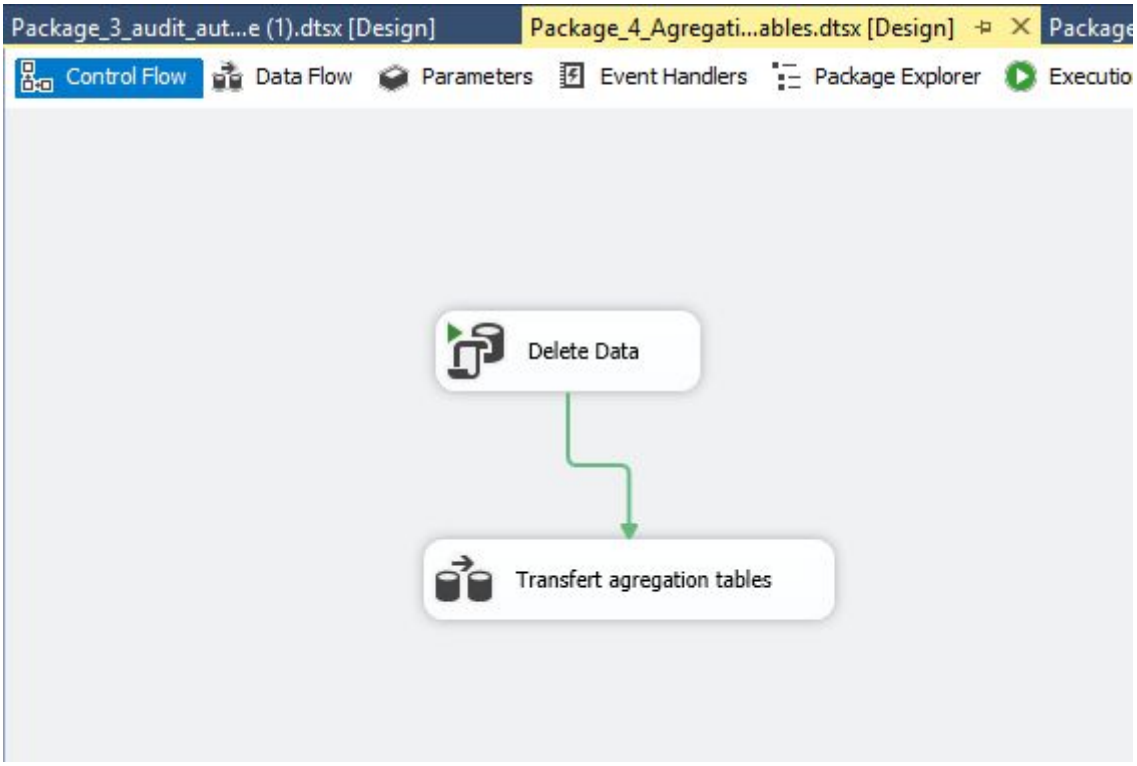


Figure - Control Flow package 4

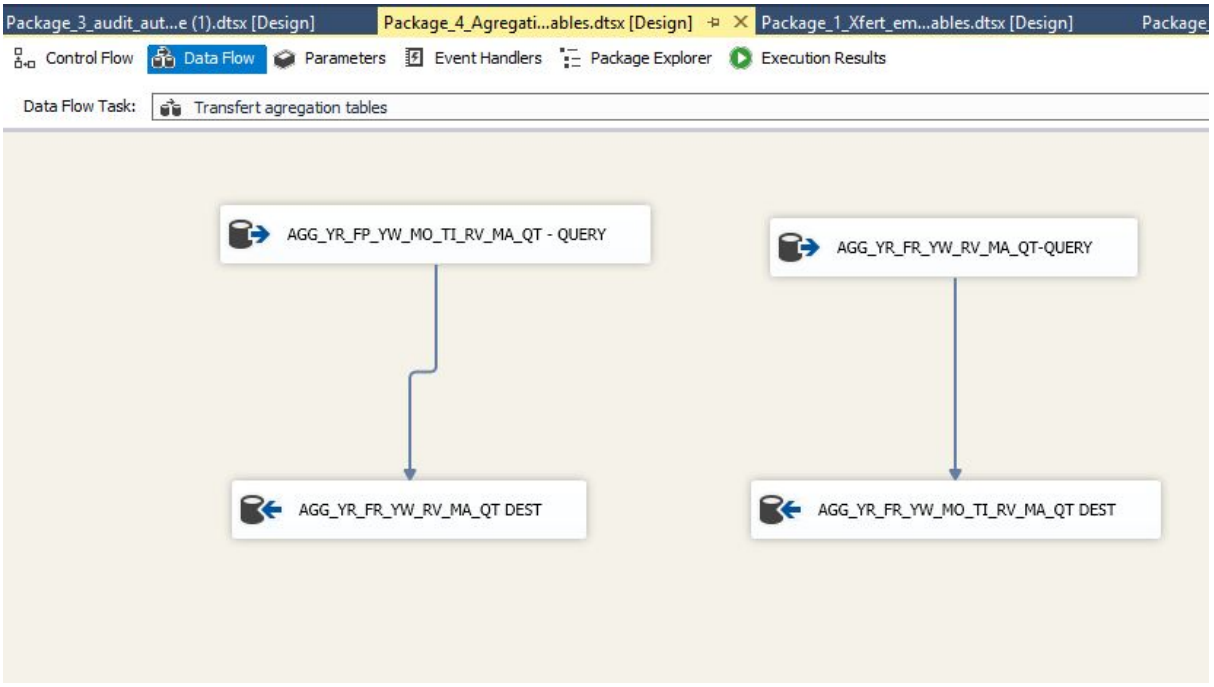


Figure - Data Flow package 4

Part 2: Data Warehouse Optimization

Toute cette partie a été effectué dans la machine virtuelle.

1) Partitionnement de la table SHOP_FACTS

Le script se trouve en Annexe

2) Création d'un projet Analysis Services

Pour créer le cube OLAP, nous avons d'abord créer une nouvelle source de données connectée à la base Emode de SQL Server. On a par la suite créé, un "Data Source Views" pour se connecter à la base pour récupérer les informations pour la création du cube. L'étape suivante est la création du cube et des différentes dimensions.

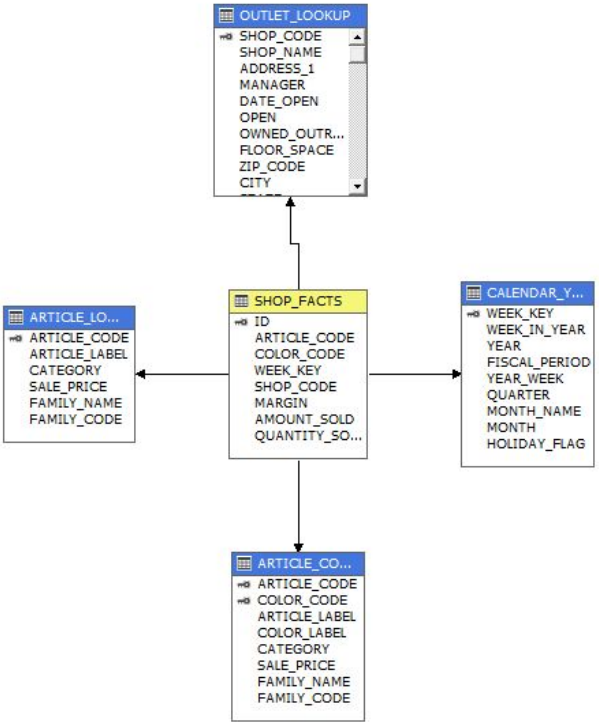


Figure - Modèle en étoile du Cube

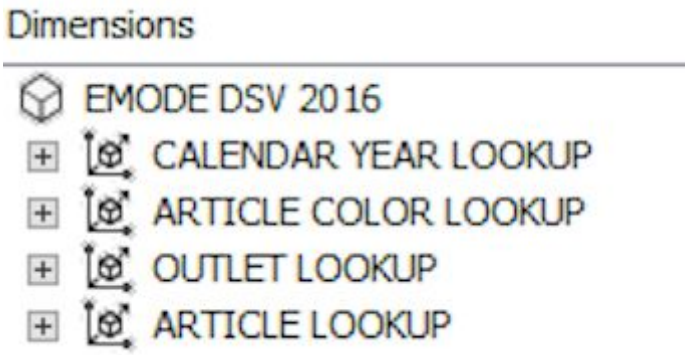


Figure - Dimensions du cube

Pour la dimension temporelle, nous avons utilisé la table : CALENDAR_YEAR_LOOKUP

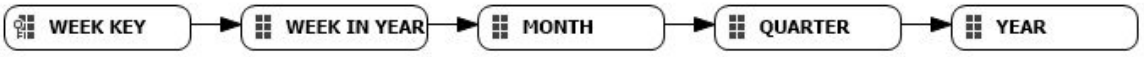


Figure - Relations de la dimension temporelle

Pour la dimension géographique, nous avons utilisé la table : OUTLET_LOOKUP

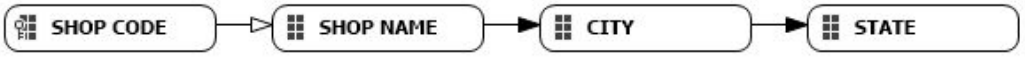


Figure - Relations de la dimension géographique

Puis, nous avons effectué le déploiement du projet :

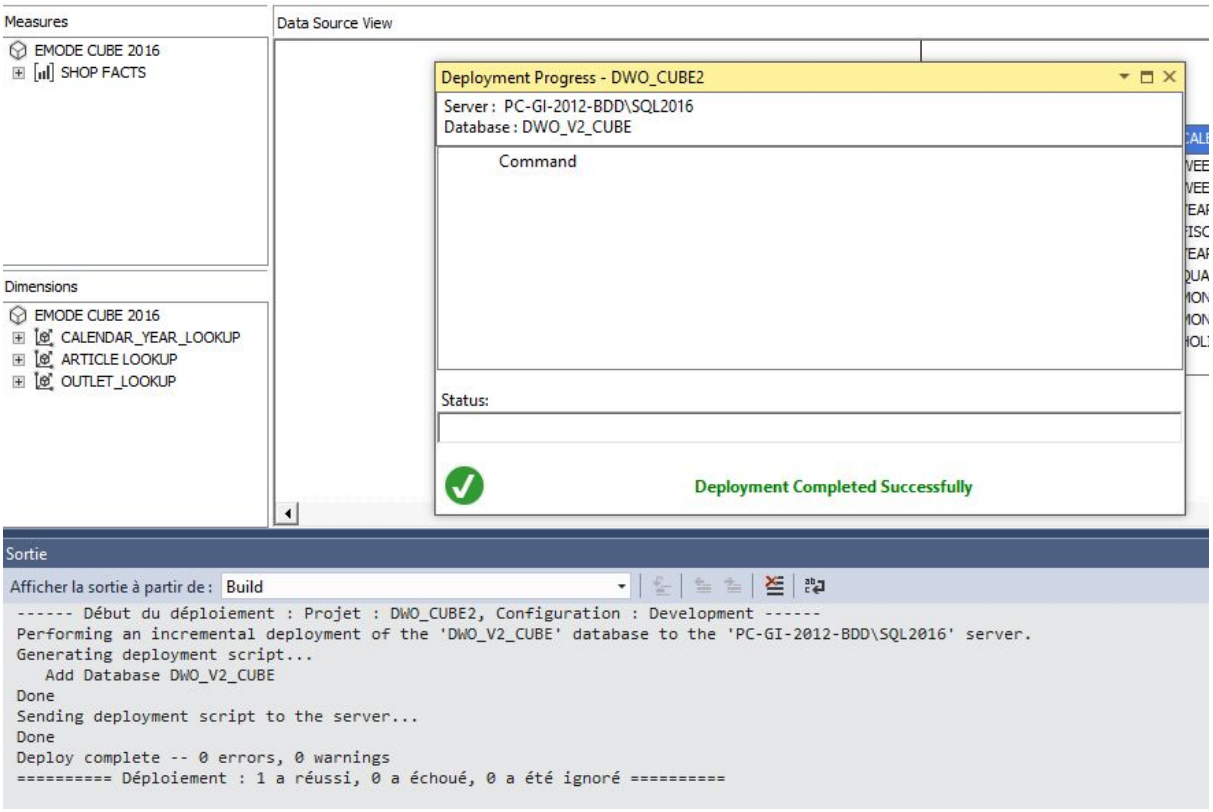


Figure - Déploiement du projet

3) Exploration du cube

Exploration du cube pour la dimension géographique :

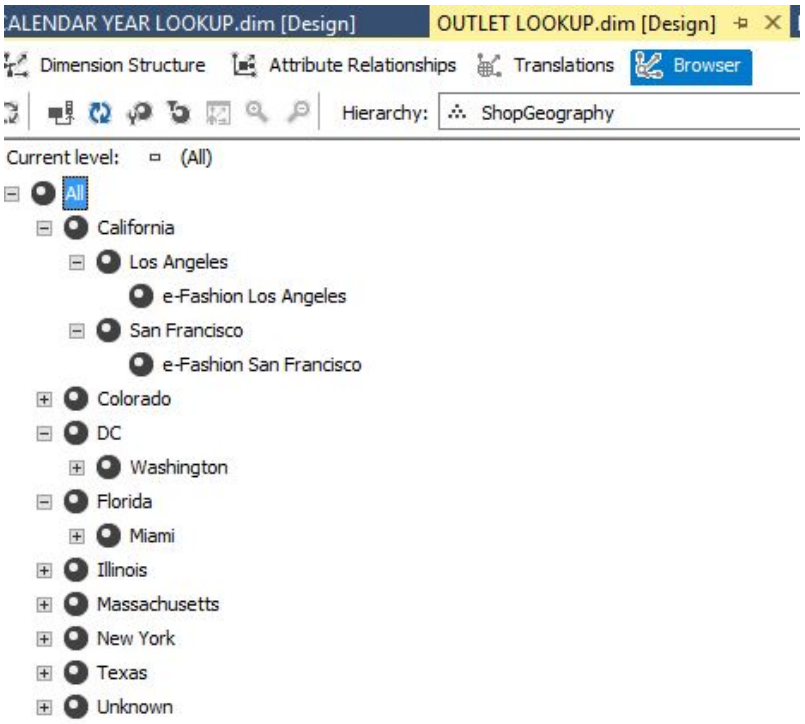


Figure - Exploration dimension géographique

Exploration du cube pour la dimension temporelle :

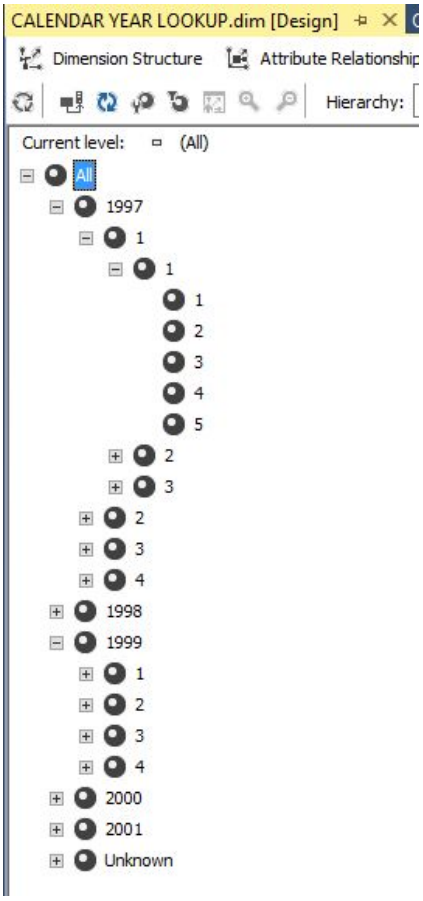
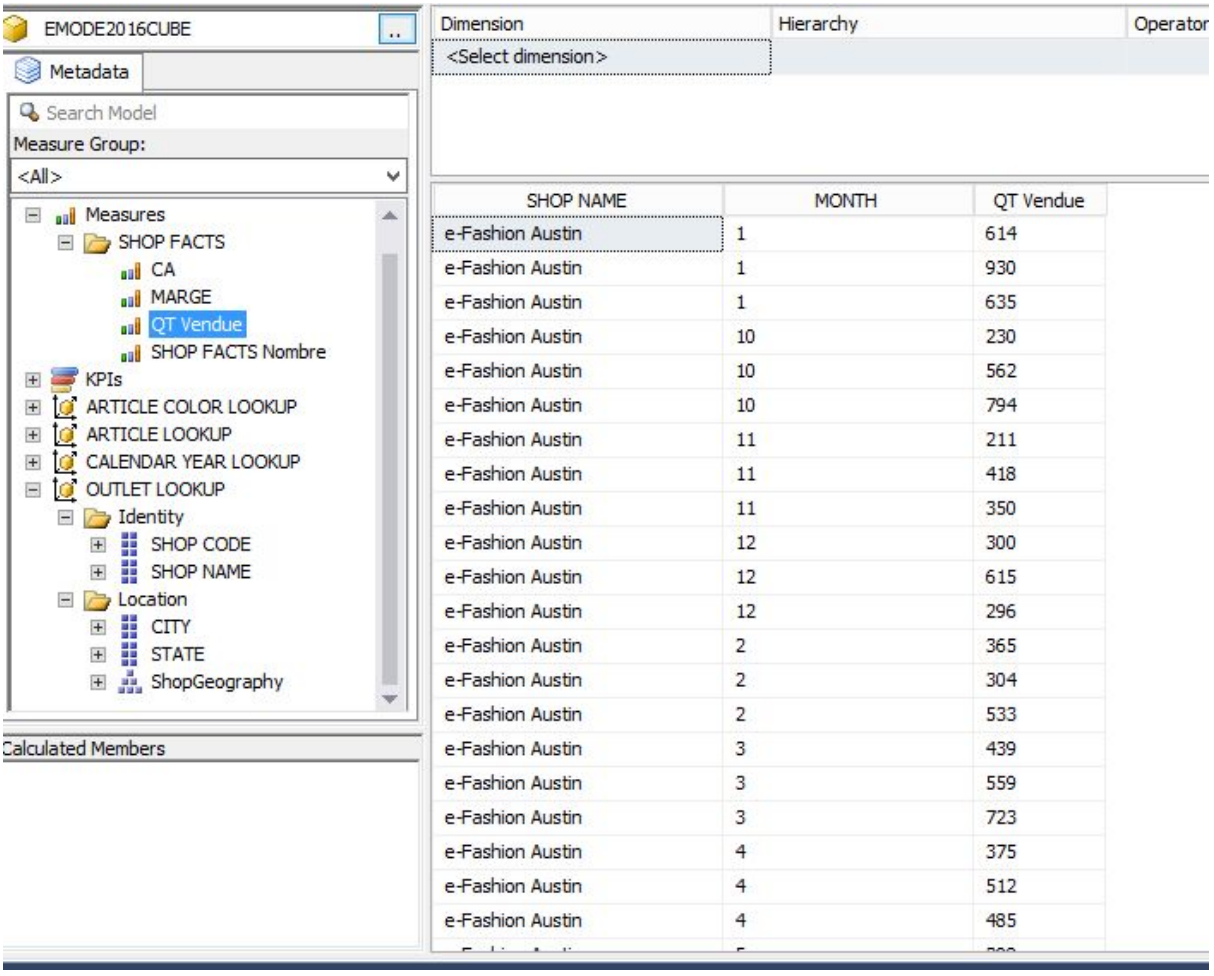


Figure - Exploration dimension temporelle



SHOP NAME	MONTH	QT Vendue
e-Fashion Austin	1	614
e-Fashion Austin	1	930
e-Fashion Austin	1	635
e-Fashion Austin	10	230
e-Fashion Austin	10	562
e-Fashion Austin	10	794
e-Fashion Austin	11	211
e-Fashion Austin	11	418
e-Fashion Austin	11	350
e-Fashion Austin	12	300
e-Fashion Austin	12	615
e-Fashion Austin	12	296
e-Fashion Austin	2	365
e-Fashion Austin	2	304
e-Fashion Austin	2	533
e-Fashion Austin	3	439
e-Fashion Austin	3	559
e-Fashion Austin	3	723
e-Fashion Austin	4	375
e-Fashion Austin	4	512
e-Fashion Austin	4	485

Figure - Exploration du cube

Partie 3 : Implémentation du reporting

La dernière partie de ce projet est la partie reporting c'est à dire la réalisation de différents rapports analysant les ventes des magasins disponible dans EMODE. Pour effectuer ces rapports, nous avons utilisés différents outils étudiés au cours de cette UV.

SSRS :

Le premier outil utilisé est Sql Server Reporting Services (SSRS) permettant la création de tableaux mais aussi de graphiques.

Nous avons créés un nouveau projet, puis les différents rapports en ajoutant pour chacun une connexion à la base Emode de SQL SERVER. L'étape suivante a été de créer des "datasets" : ce sont les données que nous allons utilisés pour les rapports, pour cela nous avons sélectionné les colonnes des tables de Emode nécessaires pour chaque rapport.

1) Création d'un tableau vertical :

Quantity sold by year and by country :

Year	Month	Country	Qty sold
1999	1	California	1185
		Colorado	309
		DC	542
		Florida	378
		Illinois	670
		Massachusetts	236
		New York	1483
		Texas	2094
		Total par mois :	6897
	2	California	921
		Colorado	261
		DC	364
		Florida	230
		Illinois	463
		Massachusetts	196
		New York	918
		Texas	1336
		Total par mois :	4689
	3	California	1403
		Colorado	351
	12	California	947
		Colorado	258
		DC	258
		Florida	231
		Illinois	373
		Massachusetts	246
		New York	789
		Texas	1142
		Total par mois :	4244
		Total par année :	90296
Total général :			223229

Figure - Tableau de quantité de produits vendue dans chaque Etat par année et par mois

Sur ce tableau, nous avons l'affichage de la quantité de produits vendus pour chaque Etat par année et par mois. Ce tableau affiche la quantité total par magasin par mois et par année mais aussi le total par mois de chaque Etat mais aussi le total par année. A la fin du rapport, il y a l'affichage du total de quantité vendue pour toutes les années présentes dans le rapport.

2) Création d'un tableau croisé :

Quantity sold by Category and Shop

Family name	Category	California		Colorado	DC	Florida	Illinois	Massachusetts	New York		Texas				Total
		e-Fashion Los Angeles	e-Fashion San Francisco	e-Fashion Colorado Springs	e-Fashion Washington Tolbooth	e-Fashion Miami Sundance	e-Fashion Chicago 33rd	e-Fashion Boston Newbury	e-Fashion New York 5th	e-Fashion New York Magnolia	e-Fashion Austin	e-Fashion Dallas	e-Fashion Houston	e-Fashion Houston Leighton	
		196	128	156	148	86	132	56	126	300	146	140	126	248	1988
	Belts, bags, wallets	8932	5196	6256	9374	3261	12050	1627	9008	15040	10460	3836	4415	8730	98185
	Belts,bags,wallets						1								1
	Hair accessories	611	268	370	292	398	586	504	621	1067	620	410	552	841	7140
Accessories	Hats, gloves, scarves	7988	4056	3147	5057	2621	6964	9	4432	7686	4750	3340	3418	5070	58538
	Hats,gloves,scarves	214			38	20	12		284	18	2	96	4	6	694
	Jewelry	25796	19781	15763	22339	13351	20884	10	22405	24647	17331	13048	13327	22656	231338
	Lounge wear	879	604	357	545	309	481	137	646	1232	636	399	485	707	7417
	Samples	3265	1485	1457	1208	1912	1556	1129	4035	5996	3536	1061	1503	3480	31623

Figure - Tableau croisé de la quantité de produits vendus par catégorie et magasin

Ce tableau permet d'avoir la quantité de produits vendus par Etat, magasin en fonction de la catégorie de produits (et de la ligne des produits). Ce tableau nous indique aussi le total de ventes par catégorie de produits de tous les magasins.

3) Création du graphe :

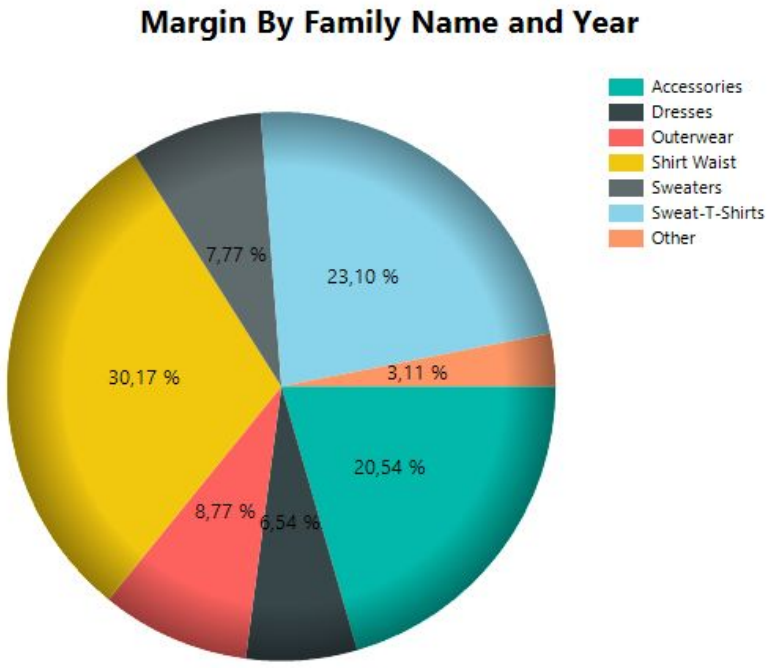


Figure - Graphe sectoriel de la marge par catégorie

Le graphe SSRS est un graphe en secteur indiquant les pourcentages de marge pour chaque ligne de produits. Les lignes de produits ayant une marge faible ont été rassemblés dans la donnée "Other".

La dernière étape a été d'ajouter l'url pour qu'ils soient disponible depuis le navigateur web.

BO :

Création de l'univers et du modèle en étoile. Nous avons utilisé Universe Designer pour créer cet univers.

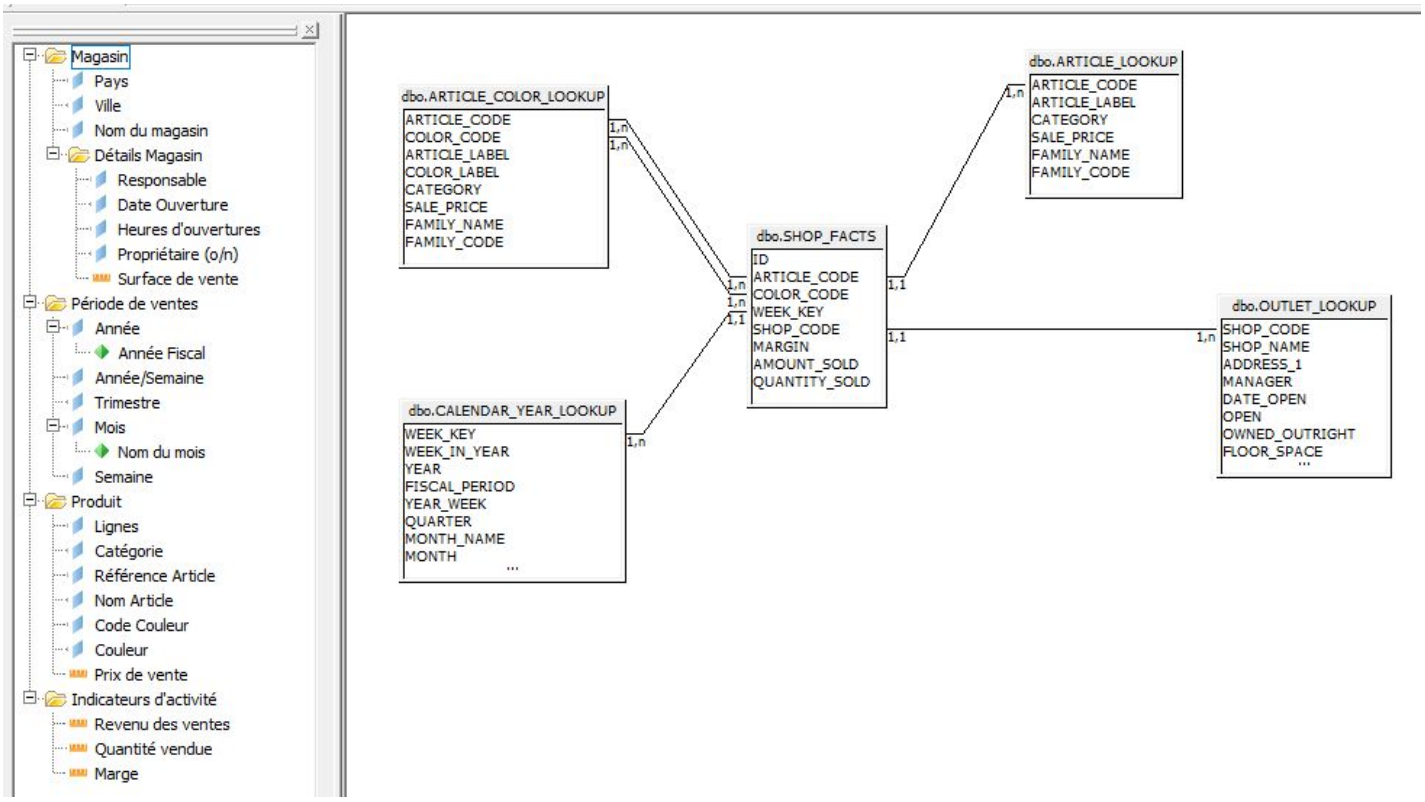


Figure - Univers

Rapport Web Intelligence :
Pour faire les différents rapports sur Web intelligence Rich Client, nous avons utilisé l'univers EFashion

Il a fallu créer 2 rapports différents avec plusieurs onglets dans chacun de ces rapports.

Pour chaque document, les différentes étapes sont :
1ère étape : la requête
2ème étape : la création des rapports : tableaux, graphique
3ème étape : Ajouter les contrôles d'entrées

Premier document : B01SalesRevenueAnalysis.wid

Premier rapport :

Sales Analysis by Geography and Time

2004	2004											
	1	2	3	4	5	6	7	8	9	10	11	12
California	$178 186	$120 205	$220 830	$185 423	$163 346	$92 725	$115 918	$36 011	$242 380	$141 648	$100 114	$107 426
Colorado	$41 687	$35 181	$54 929	$55 378	$43 661	$30 037	$33 140	$9 012	$43 469	$38 581	$27 833	$35 394
DC	$74 120	$48 128	$86 076	$82 401	$60 683	$36 779	$48 804	$14 281	$68 601	$78 923	$40 881	$53 532
Florida	$55 732	$31 099	$50 699	$41 044	$49 190	$30 937	$23 098	$9 698	$18 130	$28 763	$24 240	$43 356
Illinois	$101 985	$63 674	$90 795	$87 756	$95 829	$57 564	$42 214	$11 403	$53 389	$54 457	$37 917	$40 932
Massachusetts	$38 798	$27 823	$25 975	$24 839	$28 985	$17 079	$10 995	$1 071	########	########	$4 582	$58 673
New York	$222 474	$124 300	$209 210	$174 392	$179 541	$126 029	$128 789	$39 453	$88 872	$138 461	$118 595	$117 581
Texas	$290 560	$179 663	$288 573	$244 026	$244 381	$126 670	$122 946	$52 828	$153 339	$174 375	$129 862	$192 456

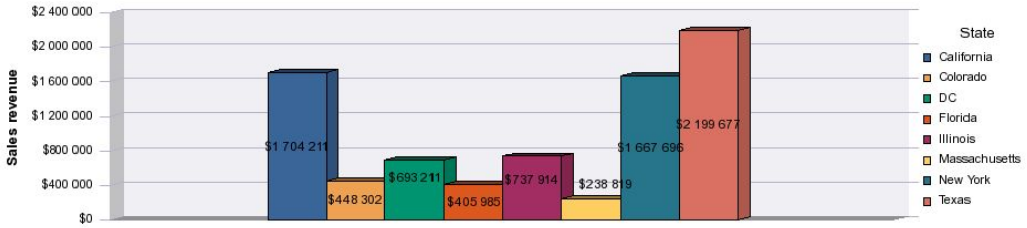
2005	2005											
	1	2	3	4	5	6	7	8	9	10	11	12
California	$226 268	$128 909	$295 538	$220 478	$182 835	$125 943	$149 159	$147 115	$464 168	$327 991	$202 223	$312 054
Colorado	$70 036	$41 301	$77 794	$64 918	$54 592	$37 828	$47 109	$28 042	$117 116	$90 613	$56 225	$82 815
DC	$95 960	$55 961	$127 568	$99 078	$93 788	$70 620	$61 389	$57 886	$169 651	$159 385	$95 601	$128 271
Florida	$73 919	$31 271	$69 086	$53 238	$59 211	$34 909	$39 070	$29 644	$52 601	$77 762	$58 074	$82 465
Illinois	$156 927	$57 663	$119 706	$100 257	$96 983	$57 482	$66 726	$48 572	$115 275	$142 000	$77 616	$111 451
Massachusetts	########	########	########	########	########	########	########	########	########	########	$32 289	$125 430
New York	$292 889	$113 226	$277 856	$227 485	$283 938	$181 089	$210 747	$78 363	$212 111	$314 298	$259 748	$311 753
Texas	$419 403	$180 682	$414 208	$302 855	$310 538	$182 586	$227 756	$191 471	$365 333	$433 822	$300 139	$404 095

2006	2006											
	1	2	3	4	5	6	7	8	9	10	11	12
California	$288 260	$161 506	$279 979	$260 420	$327 341	$201 638	$250 330	$156 525	$368 911	$298 916	$202 128	$196 726
Colorado	$82 188	$49 653	$72 912	$71 550	$81 342	$60 770	$64 338	$49 775	$118 776	$80 313	$59 011	$52 954
DC	$108 675	$73 420	$96 913	$94 272	$98 438	$70 388	$91 387	$47 496	$132 762	$102 744	$84 697	$52 389
Florida	$85 677	$42 192	$76 014	$64 080	$86 731	$70 658	$67 346	$45 600	$102 623	$73 109	$49 372	$48 523
Illinois	$116 260	$53 410	$85 989	$90 183	$170 066	$94 476	$71 733	$54 915	$146 539	$81 115	$91 693	$77 709
Massachusetts	$83 637	$60 773	$75 890	$71 267	$86 464	$62 797	$72 316	$53 530	$111 618	$95 671	$61 474	$51 732
New York	$320 715	$163 063	$263 382	$215 974	$355 162	$284 480	$313 186	$170 350	$430 711	$282 432	$188 334	$163 232
Texas	$415 955	$259 434	$427 091	$354 584	$408 604	$325 034	$316 677	$231 174	$484 778	$416 000	$306 390	$239 377

Deuxième rapport :

Sales Analysis by Geography and Time

2004	2004												2004
	1	2	3	4	5	6	7	8	9	10	11	12	Somme :
California	$178 186	$120 205	$220 830	$185 423	$163 346	$92 725	$115 918	$36 011	$242 380	$141 648	$100 114	$107 426	$1 704 211
Colorado	$41 687	$35 181	$54 929	$55 378	$43 661	$30 037	$33 140	$9 012	$43 469	$38 581	$27 833	$35 394	$448 302
DC	$74 120	$48 128	$86 076	$82 401	$60 683	$36 779	$48 804	$14 281	$68 601	$78 923	$40 881	$53 532	$693 211
Florida	$55 732	$31 099	$50 699	$41 044	$49 190	$30 937	$23 098	$9 698	$18 130	$28 763	$24 240	$43 356	$405 985
Illinois	$101 985	$63 674	$90 795	$87 756	$95 829	$57 564	$42 214	$11 403	$53 389	$54 457	$37 917	$40 932	$737 914
Massachusetts	$38 798	$27 823	$25 975	$24 839	$28 985	$17 079	$10 995	$1 071	#########	#########	$4 582	$58 673	$238 819
New York	$222 474	$124 300	$209 210	$174 392	$179 541	$126 029	$128 789	$39 453	$88 872	$138 461	$118 595	$117 581	$1 667 696
Texas	$290 560	$179 663	$288 573	$244 026	$244 381	$126 670	$122 946	$52 828	$153 339	$174 375	$129 862	$192 456	$2 199 677
Somme :	$1 003 541	$630 073	$1 027 085	$895 260	$865 615	$517 819	$525 904	$173 756	$668 181	$655 206	$484 024	$649 350	$8 095 814



Troisième rapport : la page de couverture

Analyse multiple des tables

1. Analyse mensuelle des ventes par tableau croisé

2. Analyse mensuelle des ventes par graphique

Deuxième Document : B02QuantityAnalysis.wid

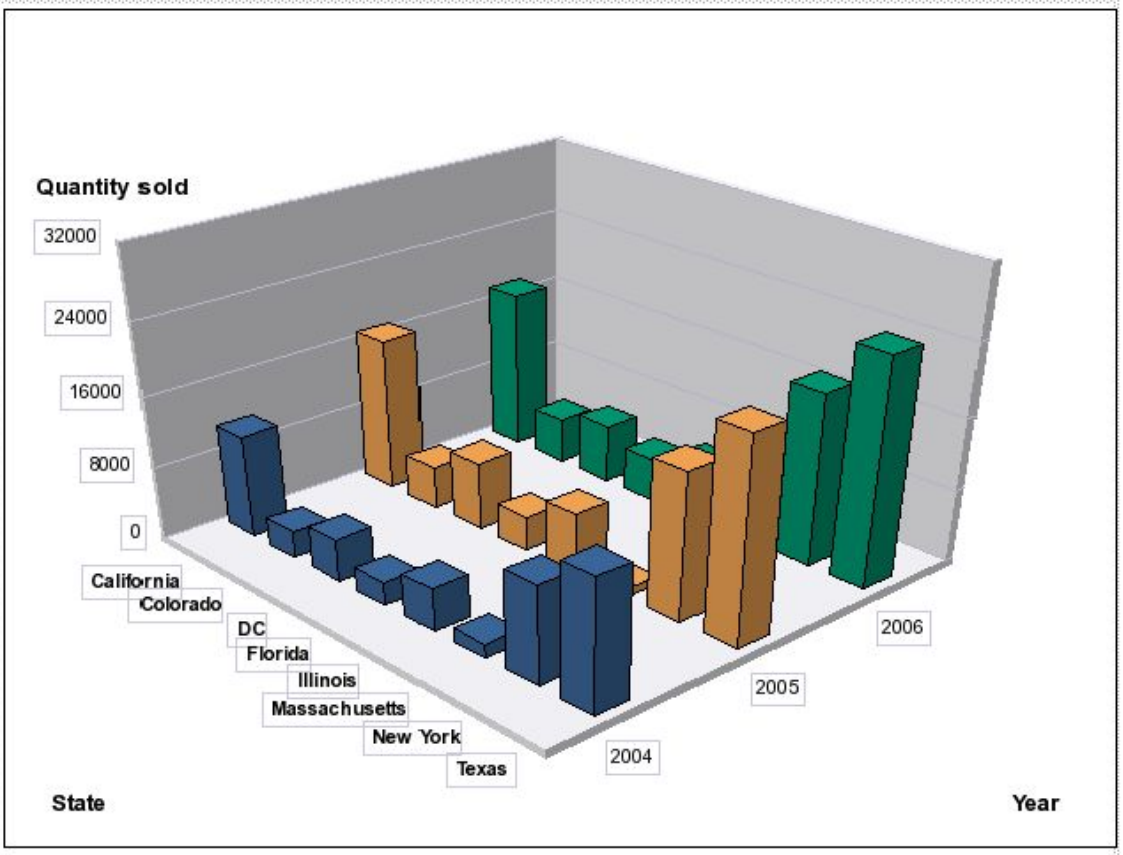
Premier rapport :

Quantity Sold by Year and State

Year	State	Quantity sold
2004	California	11 304
	Colorado	2 971
	DC	4 681
	Florida	2 585
	Illinois	4 713
	Massachusetts	1 505
	New York	10 802
	Texas	14 517
2004	Somme :	53 078

Deuxième rapport :

Quantity Sold by Year and State



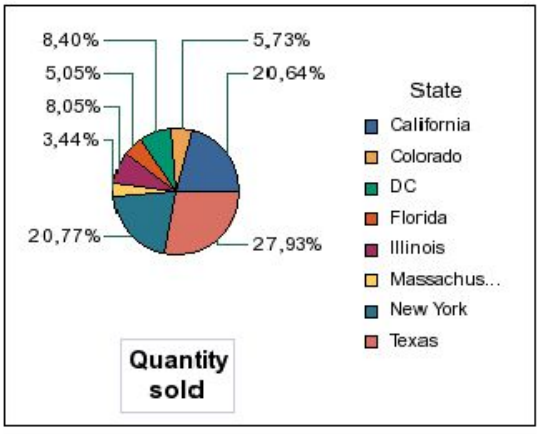
State	Quantity sold
California	46 074
Colorado	12 787
DC	18 744
Florida	11 267
Illinois	17 976
Massachusetts	7 676
New York	46 358
Texas	62 347

Year	Quantity sold
2004	53 078
2005	79 855
2006	90 296

Troisième rapport :

Quantity Sold by Year and State

State	Quantity sold	Percentage
California	46 074	20,64%
Colorado	12 787	5,73%
DC	18 744	8,40%
Florida	11 267	5,05%
Illinois	17 976	8,05%
Massachusetts	7 676	3,44%
New York	46 358	20,77%
Texas	62 347	27,93%
	Percentage :	100,00%

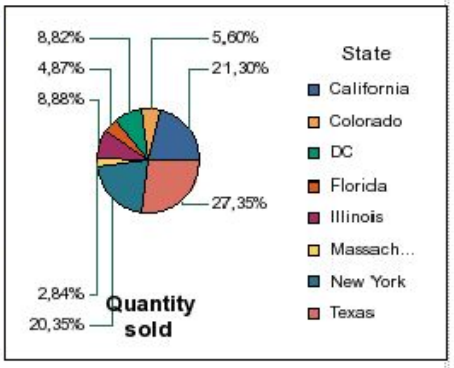


Quatrième rapport :

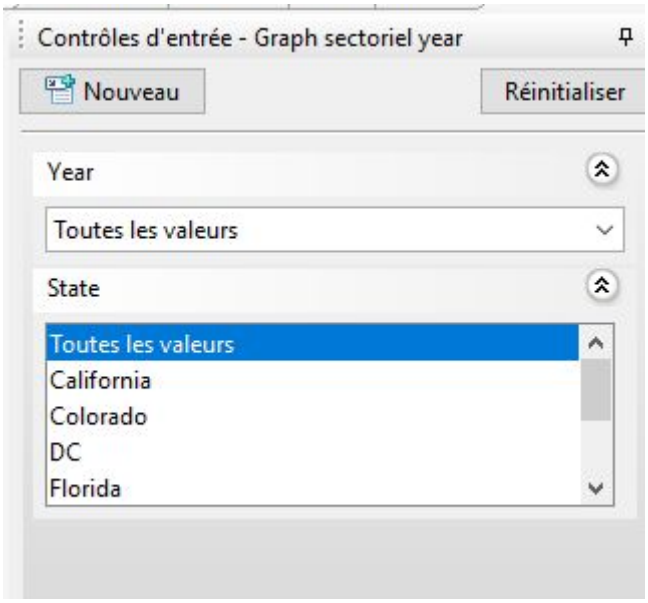
Quantity Sold by Year and State

2004

State	Quantity sold	Pourcentage
California	11 304	21,30%
Colorado	2 971	5,60%
DC	4 681	8,82%
Florida	2 585	4,87%
Illinois	4 713	8,88%
Massachusetts	1 505	2,84%
New York	10 802	20,35%
Texas	14 517	27,35%
	Pourcentage :	100,00%



Dernière étape : les contrôles d'entrées sur chaque rapport



Reporting Excel

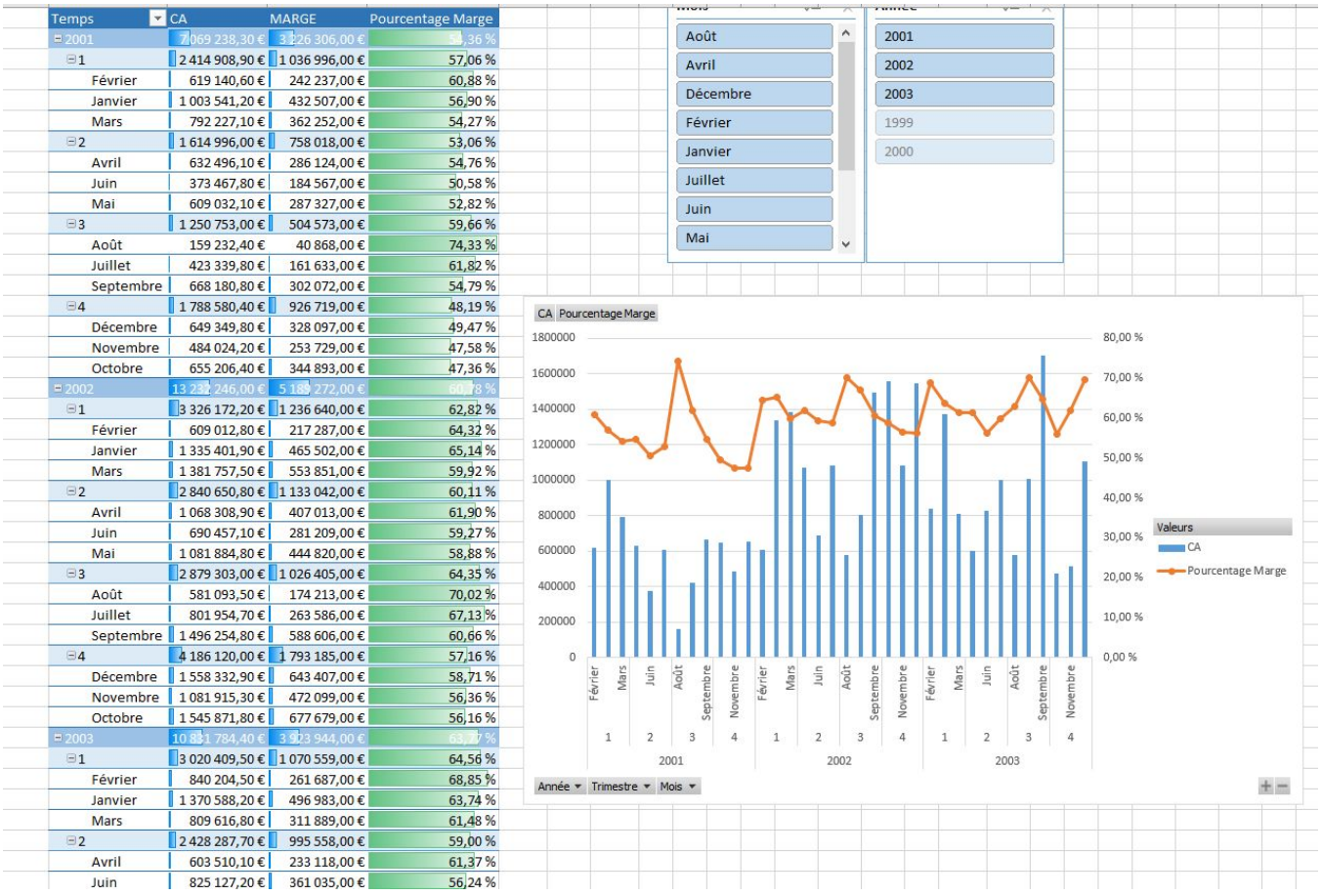


Figure - Tableau de bord Excel

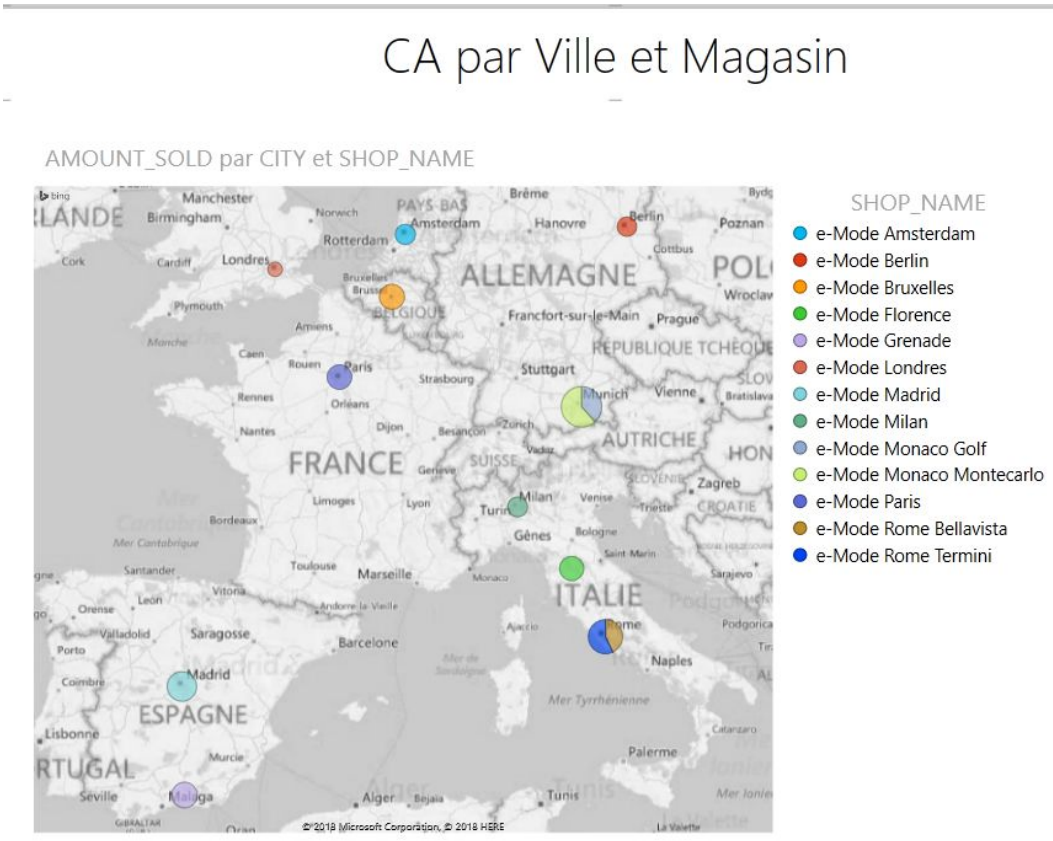


Figure - CA par ville et magasin

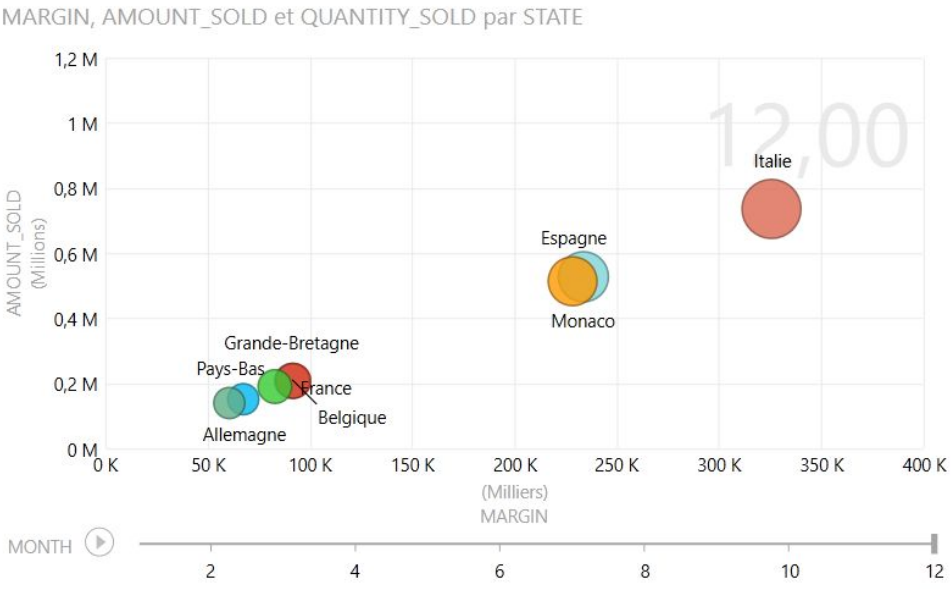


Figure - Evolution du CA et de la marge par pays en fonction du mois

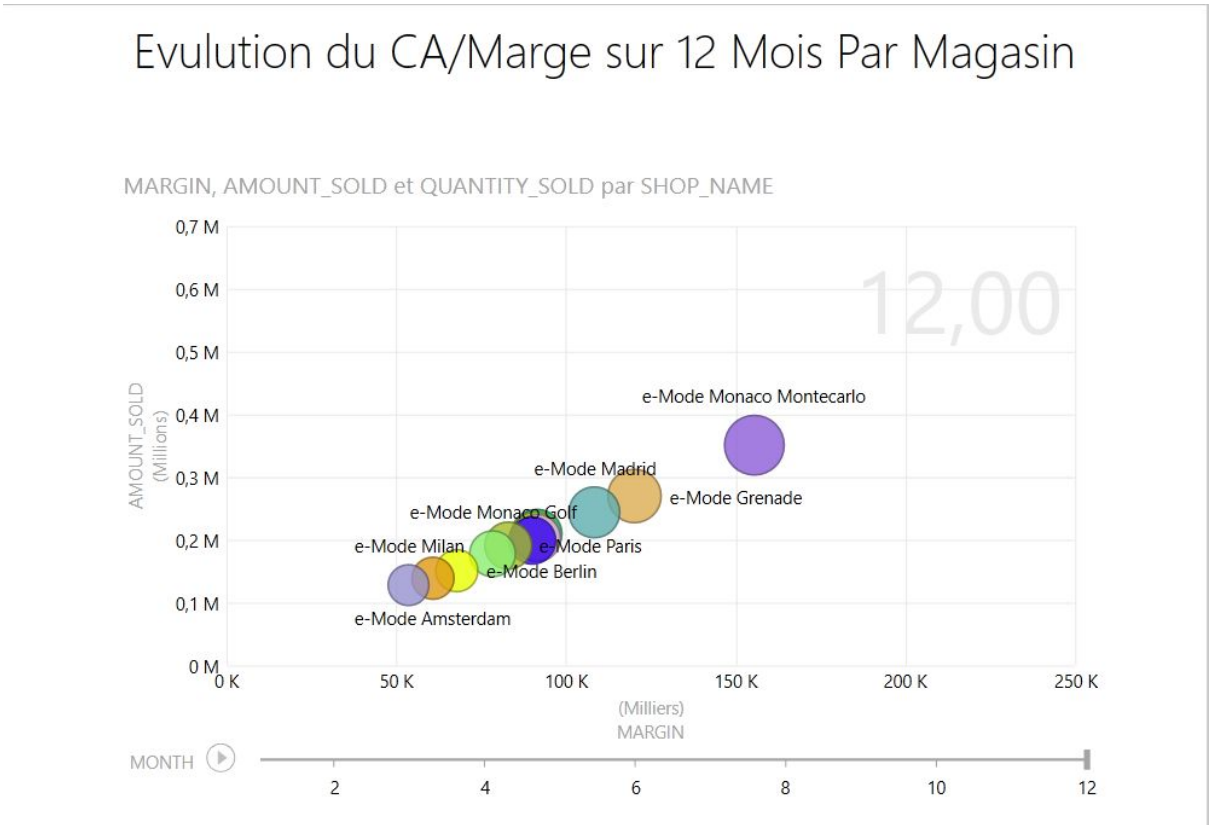


Figure - Evolution du CA et de la marge par magasin en fonction du mois

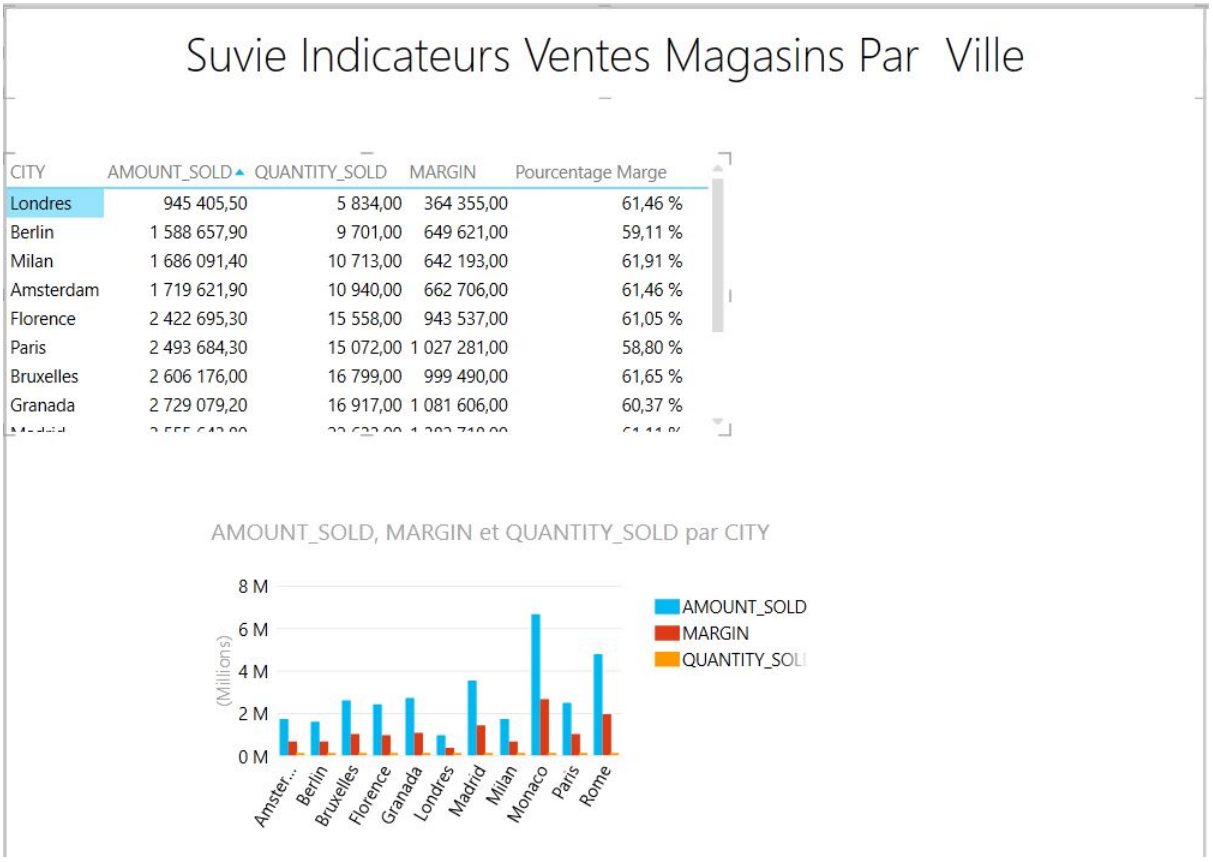


Figure - Suivi des indicateurs de ventes des magasins par ville

Reporting QlikView :

La 1ère étape a été la création du modèle en étoile avec les 5 tables demandées. Ensuite, il a fallu créer un tableau de bord avec un tableau, un tableau croisé et un diagramme.

Notre tableau de bord est composé de plusieurs éléments : une barre de recherche, un bouton “effacer”, des colonnes de tables permettant de sélectionner des valeurs et les tableaux, graphes demandés.

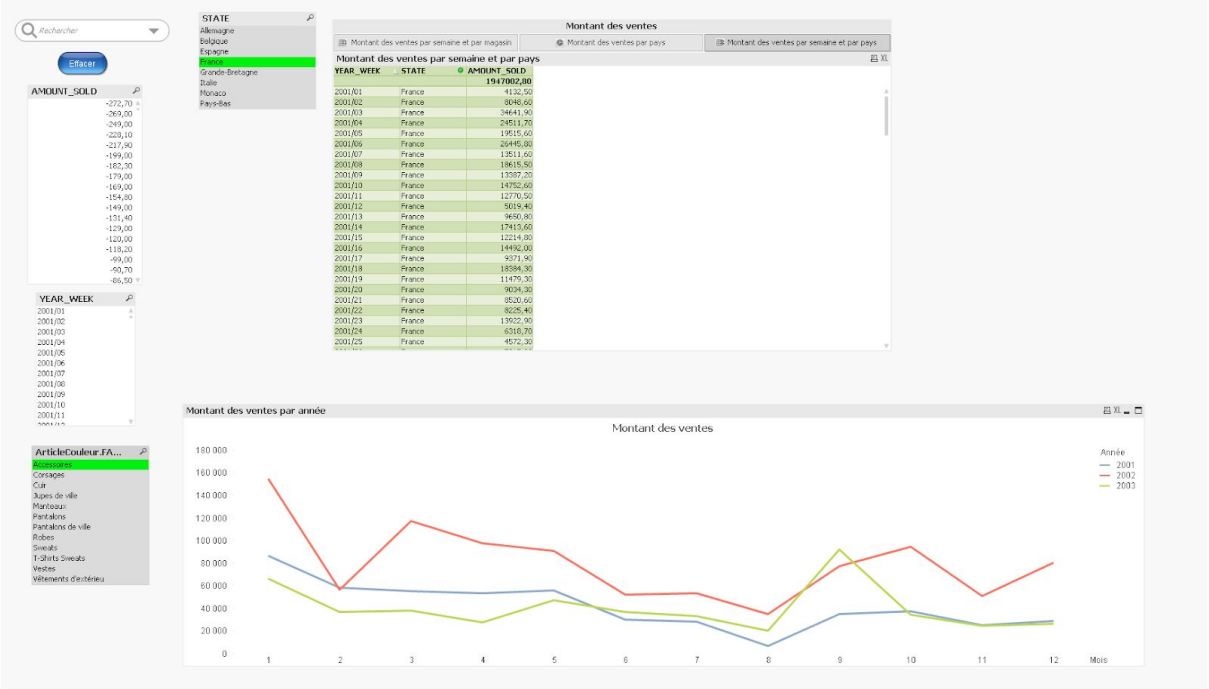
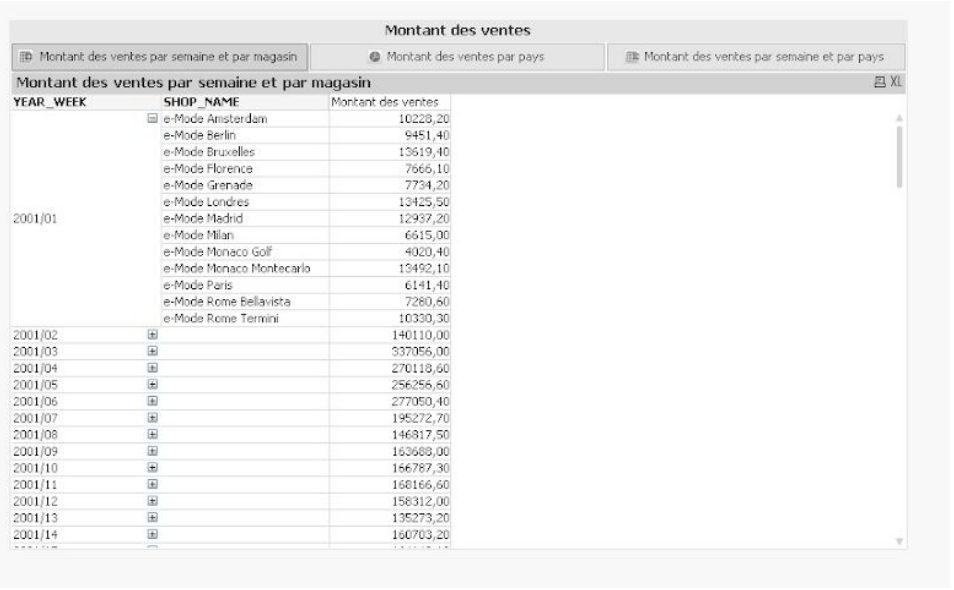


Figure - Tableau de bord

Notre premier tableau est un tableau croisé représentant le montant des ventes par semaine et par magasin.



Montant des ventes

Montant des ventes par semaine et par magasin | Montant des ventes par pays | Montant des ventes par semaine et par pays

Montant des ventes par semaine et par magasin

YEAR_WEEK	SHOP_NAME	Montant des ventes
	e-Mode Amsterdam	10228,20
	e-Mode Berlin	9451,40
	e-Mode Bruxelles	13619,40
	e-Mode Florence	7666,10
	e-Mode Grenade	7734,20
	e-Mode Londres	13425,50
2001/01	e-Mode Madrid	12937,20
	e-Mode Milan	6615,00
	e-Mode Monaco Golf	4020,40
	e-Mode Monaco Montecarlo	13492,10
	e-Mode Paris	6141,40
	e-Mode Rome Bellavista	7280,60
	e-Mode Rome Termini	10330,30
2001/02		140110,00
2001/03		337056,00
2001/04		270118,60
2001/05		256256,60
2001/06		277050,40
2001/07		195272,70
2001/08		146817,50
2001/09		163688,00
2001/10		166787,30
2001/11		168166,60
2001/12		158312,00
2001/13		135273,20
2001/14		160703,20

Figure - Tableau croisé du montant des ventes par semaine et par magasin

Notre deuxième tableau, est le tableau représentant le montant des ventes par semaine/année et par pays.

Montant des ventes

Montant des ventes par semaine et par magasin | Montant des ventes par pays | Montant des ventes par semaine et par pays

Montant des ventes par semaine et par pays

YEAR_WEEK	STATE	AMOUNT_SOLD
		31133268,70
2001/01	France	6141,40
2001/01	Allemagne	9451,40
2001/01	Pays-Bas	10228,20
2001/01	Grande-Bretagne	13425,50
2001/01	Belgique	13619,40
2001/01	Monaco	17512,50
2001/01	Espagne	20671,40
2001/01	Italie	31892,00
2001/02	Pays-Bas	5301,20
2001/02	Allemagne	6040,40
2001/02	France	11437,90
2001/02	Grande-Bretagne	13295,80
2001/02	Belgique	15799,50
2001/02	Espagne	24995,50
2001/02	Italie	30692,10
2001/02	Monaco	32547,60
2001/03	Grande-Bretagne	7398,10
2001/03	Pays-Bas	13867,70
2001/03	Belgique	18742,60
2001/03	Allemagne	21165,70
2001/03	France	38486,60
2001/03	Espagne	57864,30
2001/03	Monaco	79262,70
2001/03	Italie	100268,30
2001/04	Pays-Bas	7822,10

Figure - Tableau du montant des ventes par semaine/année et par pays

Nous avons 2 graphes sur notre tableau de bord : le montant des ventes par pays est représenté avec un graphe sectoriel et le montant des ventes par année est représenté par un graphique.

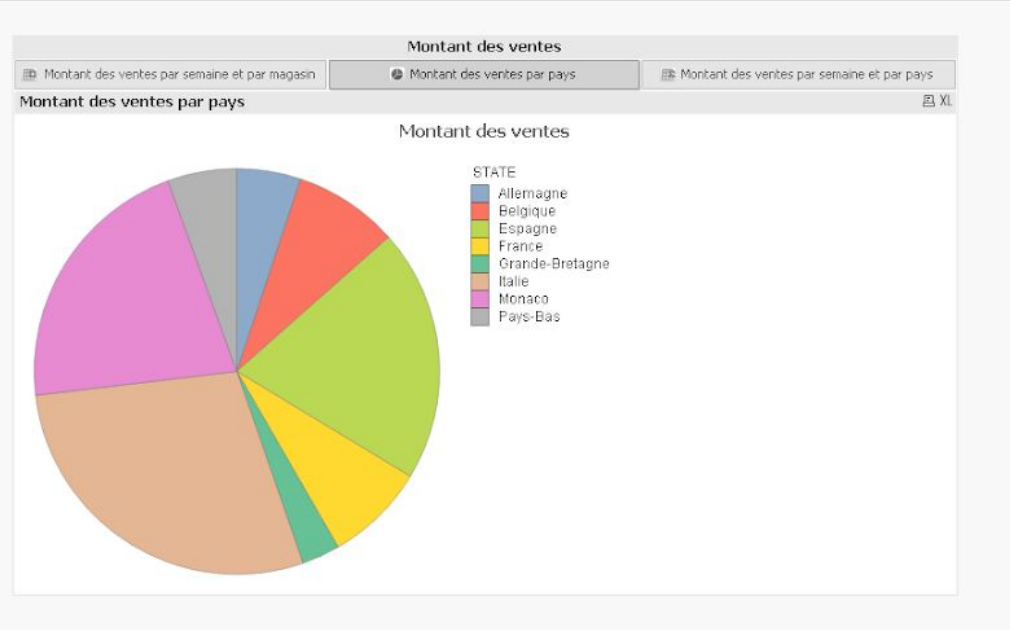


Figure - Graphe sectoriel du montant des ventes par pays

Figure - Graphique représentant le montant des ventes par année

Annexe

Annexe 1 : Qualité des données

Data quality

```
-- Liste des tables, description des tables
SELECT * FROM TAB;

DESC ARTICLE_COLOR_LOOKUP;
DESC ARTICLE_LOOKUP;
DESC ARTICLE_LOOKUP_CRITERIA;
DESC CALENDAR_YEAR_LOOKUP;
DESC OUTLET_LOOKUP;
DESC PRODUCT_PROMOTION_FACTS;
DESC PROMOTION_LOOKUP;
DESC SHOP_FACTS;

-- Nombre d'élément de chaque table
SELECT COUNT(CONCAT(ARTICLE_CODE,COLOR_CODE)) AS "NB ARTICLE CODE/COLOR
CODE" FROM ARTICLE_COLOR_LOOKUP;
SELECT COUNT(*) AS "NB ARTICLE CODE" FROM ARTICLE_LOOKUP;
SELECT COUNT(*) AS "NB ID ARTICLE CRITERIA" FROM ARTICLE_LOOKUP_CRITERIA;
SELECT COUNT(*) AS "NB WEEK_KEY" FROM CALENDAR_YEAR_LOOKUP;
SELECT COUNT(*) AS "NB SHOP CODE"  FROM OUTLET_LOOKUP;
SELECT COUNT(*) AS "NB PROMOTION PRODUCT" FROM PRODUCT_PROMOTION_FACTS;
SELECT COUNT(*) AS "NB PROMOTION_LOOKUP" FROM PROMOTION_LOOKUP;
SELECT COUNT(*) AS "NB SHOP_FACTS" FROM SHOP_FACTS;

-- Clef primaires en double de la table ARTICLE_COLOR_LOOKUP
SELECT
        ACL.ARTICLE_CODE AS "ARTICLE CODE"
        , ACL.COLOR_CODE AS "COLOR CODE"
        , COUNT(CONCAT(ACL.ARTICLE_CODE,ACL.COLOR_CODE)) AS "NB PK ARTICLE
CODE + COLOR CODE"
FROM
        ARTICLE_COLOR_LOOKUP ACL
GROUP BY
        ACL.ARTICLE_CODE
        , ACL.COLOR_CODE
        , CONCAT(ACL.ARTICLE_CODE,ACL.COLOR_CODE)
HAVING
        COUNT(CONCAT(ACL.ARTICLE_CODE,ACL.COLOR_CODE)) > 1
ORDER BY
```

```
        ACL.ARTICLE_CODE,ACL.COLOR_CODE ASC;

-- Clefs étrangères manquantes de la table ARTICLE_COLOR_LOOKUP
SELECT
        ACL.ARTICLE_CODE AS "ARTICLE CODE"
FROM
        ARTICLE_COLOR_LOOKUP ACL
WHERE
        ACL.ARTICLE_CODE NOT IN (SELECT
                                                ALO.ARTICLE_CODE
                                        FROM
                                                ARTICLE_LOOKUP ALO)
ORDER BY
        ACL.ARTICLE_CODE ASC;

-- Clef primaires en double de la table ARTICLE_LOOKUP
SELECT
        A.ARTICLE_CODE
        ,COUNT(A.ARTICLE_CODE) AS "NB PK"
FROM
        ARTICLE_LOOKUP A
HAVING
        COUNT(A.ARTICLE_CODE) > 1
GROUP BY
        A.ARTICLE_CODE
ORDER BY
        A.ARTICLE_CODE ASC;

-- Clef primaires en double de la table ARTICLE_LOOKUP_CRITIERIA
SELECT
        A.ID AS "ID"
        ,COUNT(A.ID) AS "NB ID"
FROM
        ARTICLE_LOOKUP_CRITERIA A
HAVING
        COUNT(A.ID) > 1
GROUP BY
        A.ID;

-- Clef primaires en double de la table CALENDAR_YEAR_LOOKUP
SELECT
        C.WEEK_KEY
FROM
        CALENDAR_YEAR_LOOKUP C
HAVING
        COUNT(C.WEEK_KEY) > 1
```

```
GROUP BY
	C.WEEK_KEY;

-- Clef primaires en double de la table OUTLET_LOOKUP
SELECT
	O.SHOP_CODE AS "SHOP CODE"
	,COUNT(O.SHOP_CODE) AS "NB SHOP CODE"
FROM
	OUTLET_LOOKUP O
HAVING
	COUNT(O.SHOP_CODE) > 1
GROUP BY
	O.SHOP_CODE;

-- Clef primaires en double de la table  PRODUCT_PROMOTION_FACTS
SELECT
	P.ID AS "ID"
	,COUNT(P.ID) AS "NB ID"
FROM
	PRODUCT_PROMOTION_FACTS P
HAVING
		COUNT(P.ID) > 1
GROUP BY
	P.ID;

-- Clef primaires en double de la table PROMOTION_LOOKUP
SELECT
	P.PROMOTION_KEY AS "PROMOTION KEY"
	,COUNT(P.PROMOTION_KEY) AS "NB PROMOTION KEY"
FROM
	PROMOTION_LOOKUP P
HAVING
		COUNT(P.PROMOTION_KEY) > 1
GROUP BY
	P.PROMOTION_KEY;

-- Clef primaires en double de la table SHOP_FACTS
SELECT
	S.ID AS "ID"
	,COUNT(S.ID) AS "NB ID"
FROM
	SHOP_FACTS S
HAVING
	COUNT(S.ID) > 1
GROUP BY
	S.ID;
```

```
-- Clefs étrangères manquantes de la table SHOP_FACTS
-- WEEK_KEY
SELECT
	S.ID "ID SHOP FACTS"
	,S.WEEK_KEY AS "WEEK KEY"
FROM
	SHOP_FACTS S
WHERE
	S.WEEK_KEY NOT IN (SELECT
		DISTINCT C.WEEK_KEY
	FROM
		CALENDAR_YEAR_LOOKUP C)
ORDER BY
	S.ID ASC;

-- ARTICLE_CODE
SELECT
	S.ID "ID SHOP FACTS"
	,S.ARTICLE_CODE AS "ARTICLE CODE"
FROM
	SHOP_FACTS S
WHERE
	S.ARTICLE_CODE NOT IN ( SELECT
		DISTINCT A.ARTICLE_CODE
	FROM
		ARTICLE_LOOKUP A)
ORDER BY
	S.ID ASC;

-- COLOR_CODE
SELECT
	S.ID "ID SHOP FACTS"
	,S.COLOR_CODE AS "COLOR CODE"
FROM
	SHOP_FACTS S
WHERE
	S.COLOR_CODE NOT IN (   SELECT
		DISTINCT A.COLOR_CODE
	FROM
		ARTICLE_COLOR_LOOKUP A)
ORDER BY
	S.ID ASC;
```

Create reject table

```
-- Création des différentes tables de rejets
-- Table de rejet des clés primaires non unique de la table ARTICLE_COLOR_LOOKUP
CREATE TABLE R_ARTICLE_COLOR_LOOKUP_PK
AS (SELECT * FROM ARTICLE_COLOR_LOOKUP WHERE 1=2);

-- Table de rejet des clés étrangères manquantes de la table ARTICLE_COLOR_LOOKUP
CREATE TABLE R_ARTICLE_COLOR_LOOKUP_FK
AS (SELECT * FROM ARTICLE_COLOR_LOOKUP WHERE 1=2);

-- Table de rejet des clés primaires non uniques de la table ARTICLE_LOOKUP
CREATE TABLE R_ARTICLE_LOOKUP_PK
AS (SELECT * FROM ARTICLE_LOOKUP WHERE 1=2);

-- Table de rejet des clés primaires non uniques de la table ARTICLE_LOOKUP_CRITERIA
CREATE TABLE R_ARTICLE_LOOKUP_CRITERIA_PK
AS (SELECT * FROM ARTICLE_LOOKUP_CRITERIA WHERE 1=2);

-- Table de rejet clés primaires non uniques de la table CALENDAR_YEAR_LOOKUP
CREATE TABLE R_CALENDAR_YEAR_LOOKUP_PK
AS (SELECT * FROM CALENDAR_YEAR_LOOKUP WHERE 1=2);

-- Table de rejet des clés primaires non uniques de la table OUTLET_LOOKUP
CREATE TABLE R_OUTLET_LOOKUP_PK
AS (SELECT * FROM OUTLET_LOOKUP WHERE 1=2);

-- Table de rejet des clés primaires non uniques de la table PRODUCT_PROMOTION_FACTS
CREATE TABLE R_PRODUCT_PROMOTION_FACTS_PK
AS (SELECT * FROM PRODUCT_PROMOTION_FACTS WHERE 1=2);

-- Table de rejet des clés primaires non uniques de la table PROMOTION_LOOKUP
CREATE TABLE R_PROMOTION_LOOKUP_PK
AS (SELECT * FROM PROMOTION_LOOKUP WHERE 1=2);

-- Table de rejet des clés primaires non unique de la table SHOP_FACTS
CREATE TABLE R_SHOP_FACTS_PK
AS (SELECT * FROM SHOP_FACTS WHERE 1=2);

-- Table de rejet des clés étrangères manquantes de la table SHOP_FACTS
-- Table de rejet de FK_WEEK_KEY
CREATE TABLE R_SHOP_FACTS_FK_WEEK_KEY
AS (SELECT * FROM SHOP_FACTS WHERE 1=2);

-- Table de rejet de FK_COLOR_CODE
CREATE TABLE R_SHOP_FACTS_FK_COLOR_CODE
AS (SELECT * FROM SHOP_FACTS WHERE 1=2);
```

```
-- Table de rejet de FK_ARTICLE_CODE
CREATE TABLE R_SHOP_FACTS_FK_ARTICLE_CODE
AS (SELECT * FROM SHOP_FACTS WHERE 1=2);
```

Fill reject table

```
-- remplissage des tables de rejet

-- Remplissage  Version 1 unique
INSERT INTO R_ARTICLE_COLOR_LOOKUP_PK(
        ARTICLE_CODE
        , COLOR_CODE
        , ARTICLE_LABEL
        , COLOR_LABEL
        , CATEGORY
        , SALE_PRICE
        , FAMILY_NAME
        , FAMILY_CODE)
SELECT
        ACL.ARTICLE_CODE
        , ACL.COLOR_CODE
        , ACL.ARTICLE_LABEL
        , ACL.COLOR_LABEL
        , ACL.CATEGORY
        , ACL.SALE_PRICE
        , ACL.FAMILY_NAME
        , ACL.FAMILY_CODE
FROM
        ARTICLE_COLOR_LOOKUP ACL
WHERE

   ARTICLE_CODE = 189480
      AND
   (COLOR_CODE = 210  OR COLOR_CODE = 902)

;

-- Table de rejet des clés primaires non uniques de la table ARTICLE_COLOR_LOOKUP
INSERT INTO R_ARTICLE_COLOR_LOOKUP_PK(
        ARTICLE_CODE
        , COLOR_CODE
        , ARTICLE_LABEL
        , COLOR_LABEL
        , CATEGORY
        , SALE_PRICE
        , FAMILY_NAME
```

```
        , FAMILY_CODE)
SELECT
        ACL.ARTICLE_CODE
        , ACL.COLOR_CODE
        , ACL.ARTICLE_LABEL
        , ACL.COLOR_LABEL
        , ACL.CATEGORY
        , ACL.SALE_PRICE
        , ACL.FAMILY_NAME
        , ACL.FAMILY_CODE
FROM
        ARTICLE_COLOR_LOOKUP ACL
WHERE
        CONCAT(ACL.ARTICLE_CODE,ACL.COLOR_CODE) IN(
                                                                SELECT

CONCAT(ACL2.ARTICLE_CODE,ACL2.COLOR_CODE)
                                                                FROM

ARTICLE_COLOR_LOOKUP ACL2
                                                                GROUP BY

CONCAT(ACL2.ARTICLE_CODE,ACL2.COLOR_CODE)
                                                                HAVING

COUNT(CONCAT(ACL2.ARTICLE_CODE,ACL2.COLOR_CODE))>1);


-- Table de rejet des clés étrangères manquantes de la table ARTICLE_COLOR_LOOKUP
INSERT INTO R_ARTICLE_COLOR_LOOKUP_FK(
        ARTICLE_CODE
        , COLOR_CODE
        , ARTICLE_LABEL
        , COLOR_LABEL
        , CATEGORY
        , SALE_PRICE
        , FAMILY_NAME
        , FAMILY_CODE)
SELECT
        ACL.ARTICLE_CODE
        , ACL.COLOR_CODE
        , ACL.ARTICLE_LABEL
        , ACL.COLOR_LABEL
        , ACL.CATEGORY
        , ACL.SALE_PRICE
        , ACL.FAMILY_NAME
```

```
        , ACL.FAMILY_CODE
FROM
        ARTICLE_COLOR_LOOKUP ACL
WHERE
        ACL.ARTICLE_CODE NOT IN (SELECT
                                                A.ARTICLE_CODE
                                        FROM
                                                ARTICLE_LOOKUP A);

-- Table de rejet des clés primaires non uniques de la table ARTICLE_LOOKUP
INSERT INTO R_ARTICLE_LOOKUP_PK(
        ARTICLE_CODE
        ,ARTICLE_LABEL
        ,CATEGORY
        ,SALE_PRICE
        ,FAMILY_NAME
        ,FAMILY_CODE)
SELECT
        A.ARTICLE_CODE
        ,A.ARTICLE_LABEL
        ,A.CATEGORY
        ,A.SALE_PRICE
        ,A.FAMILY_NAME
        ,A.FAMILY_CODE
FROM
        ARTICLE_LOOKUP A
WHERE
        A.ARTICLE_CODE IN (     SELECT
                                        ART.ARTICLE_CODE
                                FROM
                                        ARTICLE_LOOKUP ART
                                HAVING
                                        COUNT(ART.ARTICLE_CODE) > 1
                                GROUP BY
                                        ART.ARTICLE_CODE);

-- Table de rejet des clés primaires non uniques de la table ARTICLE_LOOKUP_CRITERIA
INSERT INTO R_ARTICLE_LOOKUP_CRITERIA_PK(
        ID
        ,ARTICLE_CODE
        ,CRITERIA_TYPE
        ,CRITERIA
        ,CRITERIA_TYPE_LABEL
        ,CRITERIA_LABEL)
SELECT
        A.ID
```

```
        ,A.ARTICLE_CODE
        ,A.CRITERIA_TYPE
        ,A.CRITERIA
        ,A.CRITERIA_TYPE_LABEL
        ,A.CRITERIA_LABEL
FROM
        ARTICLE_LOOKUP_CRITERIA A
WHERE
        A.ID IN (SELECT
                        ART.ID
                FROM
                        ARTICLE_LOOKUP_CRITERIA ART
                HAVING
                        COUNT(ART.ID) > 1
                GROUP BY
                        ART.ID);

-- Table de rejet des clés primaires non uniques de la table CALENDAR_YEAR_LOOKUP
INSERT INTO R_CALENDAR_YEAR_LOOKUP_PK(
        WEEK_KEY
        ,WEEK_IN_YEAR
        ,YEAR
        ,FISCAL_PERIOD
        ,YEAR_WEEK
        ,QUARTER
        ,MONTH_NAME
        ,MONTH
        ,HOLIDAY_FLAG)
SELECT
        C.WEEK_KEY
        ,C.WEEK_IN_YEAR
        ,C.YEAR
        ,C.FISCAL_PERIOD
        ,C.YEAR_WEEK
        ,C.QUARTER
        ,C.MONTH_NAME
        ,C.MONTH
        ,C.HOLIDAY_FLAG
FROM
        CALENDAR_YEAR_LOOKUP C
WHERE
        C.WEEK_KEY IN (SELECT
                                CAL.WEEK_KEY
                        FROM
                                CALENDAR_YEAR_LOOKUP CAL
                        HAVING
```

```
                                        COUNT(CAL.WEEK_KEY) > 1
                                GROUP BY
                                        CAL.WEEK_KEY);

-- Table de rejet des clés primaires non uniques de la table OUTLET_LOOKUP
INSERT INTO R_OUTLET_LOOKUP_PK(
        SHOP_NAME
        ,ADDRESS_1
        ,MANAGER
        ,DATE_OPEN
        ,OPEN
        ,OWNED_OUTRIGHT
        ,FLOOR_SPACE
        ,ZIP_CODE
        ,CITY
        ,STATE
        ,SHOP_CODE)
SELECT
        O.SHOP_NAME
        ,O.ADDRESS_1
        ,O.MANAGER
        ,O.DATE_OPEN
        ,O.OPEN
        ,O.OWNED_OUTRIGHT
        ,O.FLOOR_SPACE
        ,O.ZIP_CODE
        ,O.CITY
        ,O.STATE
        ,O.SHOP_CODE
FROM
        OUTLET_LOOKUP O
WHERE
        O.SHOP_CODE IN (SELECT
                                        OUTLET.SHOP_CODE
                                FROM
                                        OUTLET_LOOKUP OUTLET
                                HAVING
                                        COUNT(OUTLET.SHOP_CODE) > 1
                                GROUP BY
                                        OUTLET.SHOP_CODE);

-- Table de rejet des clés primaires non uniques de la table PRODUCT_PROMOTION_FACTS
INSERT INTO R_PRODUCT_PROMOTION_FACTS_PK(
        ID
        ,ARTICLE_CODE
        ,WEEK_KEY
```

```
        ,DURATION
        ,PROMOTION_KEY
        ,PROMOTION_COST)
SELECT
        P.ID
        ,P.ARTICLE_CODE
        ,P.WEEK_KEY
        ,P.DURATION
        ,P.PROMOTION_KEY
        ,P.PROMOTION_COST
FROM
        PRODUCT_PROMOTION_FACTS P
WHERE
        P.ID IN (SELECT
                        PPM.ID
                FROM
                        PRODUCT_PROMOTION_FACTS PPM
                HAVING
                                COUNT(PPM.ID) > 1
                GROUP BY
                        PPM.ID);

-- Table de rejet des clés primaires non uniques de la table PROMOTION_LOOKUP
INSERT INTO R_PROMOTION_LOOKUP_PK(
        PROMOTION_KEY
        ,PROMOTION
        ,PRINT_FLAG
        ,RADIO_FLAG
        ,TELEVISION_FLAG
        ,DIRECT_MAIL_FLAG)
SELECT
        P.PROMOTION_KEY
        ,P.PROMOTION
        ,P.PRINT_FLAG
        ,P.RADIO_FLAG
        ,P.TELEVISION_FLAG
        ,P.DIRECT_MAIL_FLAG
FROM
        PROMOTION_LOOKUP P
WHERE
        P.PROMOTION_KEY IN (SELECT
                                        PL.PROMOTION_KEY
                                FROM
                                        PROMOTION_LOOKUP PL
                                HAVING
                                                COUNT(PL.PROMOTION_KEY) > 1
```

Annexe 2 : ETL Test

CALENDAR_YEAR_LOOKUP

```
USE EMODE GO

INSERT INTO CALENDAR_YEAR_LOOKUP(WEEK_KEY, WEEK_IN_YEAR, "YEAR", FISCAL_PERIOD, YEAR_WEEK,
QUARTER, MONTH_NAME , "MONTH", HOLIDAY_FLAG)
VALUES(300, 5, 2018, 'FY12', '2018/01', 1, ‘January’, 1, 'y');

INSERT INTO CALENDAR_YEAR_LOOKUP(WEEK_KEY, WEEK_IN_YEAR, "YEAR", FISCAL_PERIOD, YEAR_WEEK,
QUARTER, MONTH_NAME , "MONTH", HOLIDAY_FLAG)
VALUES(301, 5, 2018, 'FY12', '2018/01', 1, 'January', 1, 'y');

INSERT INTO CALENDAR_YEAR_LOOKUP(WEEK_KEY, WEEK_IN_YEAR, "YEAR", FISCAL_PERIOD, YEAR_WEEK,
QUARTER, MONTH_NAME , "MONTH", HOLIDAY_FLAG)
VALUES(302, 5, 2018, 'FY12', '2018/01', 1, 'January', 1, 'y');


INSERT INTO CALENDAR_YEAR_LOOKUP(WEEK_KEY, WEEK_IN_YEAR, "YEAR", FISCAL_PERIOD, YEAR_WEEK,
QUARTER, MONTH_NAME , "MONTH", HOLIDAY_FLAG)
VALUES(302, 5, 2018, 'FY12', '2018/01', 1, 'January', 1, 'y');

COMMIT;
```

OUTLET_LOOKUP

```
USE EMODE GO

INSERT INTO OUTLET_LOOKUP(SHOP_NAME, ADDRESS_1, MANAGER, DATE_OPEN, "OPEN", OWNED_OUTRIGHT,
FLOOR_SPACE, ZIP_CODE, CITY, "STATE", SHOP_CODE)
VALUES ('e-Fashion Belfort', 'Route de lure', 'Jeremy', '15-JAN-00', 'Y', 'N', 4160, '90000', 'Belfort', 'France', 400);

INSERT INTO OUTLET_LOOKUP(SHOP_NAME, ADDRESS_1, MANAGER, DATE_OPEN, "OPEN", OWNED_OUTRIGHT,
FLOOR_SPACE, ZIP_CODE, CITY, "STATE", SHOP_CODE)
VALUES ('e-Fashion L, 'Route de Belfort', 'Jeremy', '15-JAN-00', 'Y', 'N', 4160, '70100', 'Lure', 'France', 401);

INSERT INTO OUTLET_LOOKUP(SHOP_NAME, ADDRESS_1, MANAGER, DATE_OPEN, "OPEN", OWNED_OUTRIGHT,
FLOOR_SPACE, ZIP_CODE, CITY, "STATE", SHOP_CODE)
VALUES ('e-Fashion Belfort', 'Route de L', 'Jeremy', '15-JAN-00', 'Y', 'N', 4160, '25000', 'Belfort', 'France', 402);

INSERT INTO OUTLET_LOOKUP(SHOP_NAME, ADDRESS_1, MANAGER, DATE_OPEN, "OPEN", OWNED_OUTRIGHT,
FLOOR_SPACE, ZIP_CODE, CITY, "STATE", SHOP_CODE)
VALUES ('e-Fashion Belfort', 'Route de L', 'Jeremy', '15-JAN-00', 'Y', 'N', 4160, '25000', 'Belfort', 'France', 402);

COMMIT;
```

SHOP_FACTS

```
USE EMODE GO

INSERT INTO SHOP_FACTS(ID,ARTICLE_CODE,COLOR_CODE,WEEK_KEY,SHOP_CODE,MARGIN,AMOUNT_SOLD,QU
ANTITY_SOLD)
VALUES(90000,200000,997,300,400,20,90,3);

--Erreur clé primaire -> Table de rejet

INSERT INTO SHOP_FACTS(ID,ARTICLE_CODE,COLOR_CODE,WEEK_KEY,SHOP_CODE,MARGIN,AMOUNT_SOLD,QU
ANTITY_SOLD)
VALUES(90000,200000,997,300,400,20,90,3);

--Erreur clé étrangère : ARTICLE_CODE -> Table de rejet

INSERT INTO SHOP_FACTS(ID,ARTICLE_CODE,COLOR_CODE,WEEK_KEY,SHOP_CODE,MARGIN,AMOUNT_SOLD,QU
ANTITY_SOLD)
VALUES(90001,200500,997,300,400,20,90,3);

--Erreur clé étrangère (COLOR_CODE) -> Table de rejet

INSERT INTO SHOP_FACTS(ID,ARTICLE_CODE,COLOR_CODE,WEEK_KEY,SHOP_CODE,MARGIN,AMOUNT_SOLD,QU
ANTITY_SOLD)
VALUES(90002,200000,994,300,400,20,90,3);

--Erreur clé étrangère : WEEK_KEY -> Table de rejet

INSERT INTO SHOP_FACTS(ID,ARTICLE_CODE,COLOR_CODE,WEEK_KEY,SHOP_CODE,MARGIN,AMOUNT_SOLD,QU
ANTITY_SOLD)
VALUES(90003,200000,994,700,400,20,90,3);

--Erreur clé étrangère : SHOP_CODE -> Table de rejet

INSERT INTO SHOP_FACTS(ID,ARTICLE_CODE,COLOR_CODE,WEEK_KEY,SHOP_CODE,MARGIN,AMOUNT_SOLD,QU
ANTITY_SOLD)
VALUES(90004,200000,994,300,700,20,90,3);

COMMIT;
```

Annexe 3 : Partitionnement de SHOP_FACTS

```
/*CrÈation des fichiers de groupes pour le partitionnement*/

ALTER DATABASE EMODE ADD FILEGROUP SF_FG_01;
ALTER DATABASE EMODE ADD FILEGROUP SF_FG_02;
ALTER DATABASE EMODE ADD FILEGROUP SF_FG_03;
ALTER DATABASE EMODE ADD FILEGROUP SF_FG_04;
ALTER DATABASE EMODE ADD FILEGROUP SF_FG_05;
ALTER DATABASE EMODE ADD FILEGROUP SF_FG_06;
```

```
/*CrÈation des fichiers de donnÈes par groupe de partitionnements*/

ALTER DATABASE EMODE
  ADD FILE (
                    NAME = 'SF_FILE_01'
                    ,FILENAME  = 'C:\Program Files\Microsoft SQL
Server\MSSQL13.SQL2016\MSSQL\DATA\SF_FILE_01.ndf'
                    ,SIZE = 1 GB
                    ,FILEGROWTH = 10 MB
                    )
TO FILEGROUP SF_FG_01
;

ALTER DATABASE EMODE
  ADD FILE (
                    NAME = 'SF_FILE_02'
                    ,FILENAME  = 'C:\Program Files\Microsoft SQL
Server\MSSQL13.SQL2016\MSSQL\DATA\SF_FILE_02.ndf'
                    ,SIZE = 1 GB
                    ,FILEGROWTH = 10 MB
                    )
TO FILEGROUP SF_FG_02
;

ALTER DATABASE EMODE
  ADD FILE (
                    NAME = 'SF_FILE_03'
                    ,FILENAME  = 'C:\Program Files\Microsoft SQL
Server\MSSQL13.SQL2016\MSSQL\DATA\SF_FILE_03.ndf'
                    ,SIZE = 1 GB
                    ,FILEGROWTH = 10 MB
                    )
TO FILEGROUP SF_FG_03
;

ALTER DATABASE EMODE
  ADD FILE (
                    NAME = 'SF_FILE_04'
                    ,FILENAME  = 'C:\Program Files\Microsoft SQL
Server\MSSQL13.SQL2016\MSSQL\DATA\SF_FILE_04.ndf'
                    ,SIZE = 1 GB
                    ,FILEGROWTH = 10 MB
                    )
TO FILEGROUP SF_FG_04
;

ALTER DATABASE EMODE
  ADD FILE (
                    NAME = 'SF_FILE_05'
                    ,FILENAME  = 'C:\Program Files\Microsoft SQL
Server\MSSQL13.SQL2016\MSSQL\DATA\SF_FILE_05.ndf'
                    ,SIZE = 1 GB
                    ,FILEGROWTH = 10 MB
```

```
                    )
TO FILEGROUP SF_FG_05
;

ALTER DATABASE EMODE
  ADD FILE (
                    NAME = 'SF_FILE_06'
                    ,FILENAME  = 'C:\Program Files\Microsoft SQL
Server\MSSQL13.SQL2016\MSSQL\DATA\SF_FILE_06.ndf'
                    ,SIZE = 1 GB
                    ,FILEGROWTH = 10 MB
                    )
TO FILEGROUP SF_FG_06
;


/*CrÈation de la fonction de partitionnnement*/

CREATE PARTITION FUNCTION PARTI_FUNCT_SF (NUMERIC(3,0))
AS
RANGE RIGHT FOR VALUES (53,105,157,210,263)
;

/*CrÈation du schÈma de partitionnnement*/

CREATE PARTITION SCHEME PARTI_SCHEM_SF
AS
PARTITION PARTI_FUNCT_SF
TO (SF_FG_01, SF_FG_02, SF_FG_03, SF_FG_04, SF_FG_05, SF_FG_06)
;

--  Sauvegarde et Suppression de SHOP_FACTS
SELECT
          *
INTO
          SHOP_FACTS_BACKUP_TEMP
FROM
          SHOP_FACTS;
GO

DROP TABLE SHOP_FACTS;
GO


CREATE TABLE SHOP_FACTS(
           ID                                NUMERIC (5,0) NOT NULL
          , ARTICLE_CODE                     NUMERIC (6,0)
          , COLOR_CODE              NUMERIC (4,0)
          , WEEK_KEY                         NUMERIC (3,0)
          , SHOP_CODE                        NUMERIC (3,0)
          , MARGIN                           NUMERIC (18,0)
          , AMOUNT_SOLD                      NUMERIC (13,2)
          , QUANTITY_SOLD                    NUMERIC  (13,2)
          ,
```

```
	 CONSTRAINT [PK_SHOP_FACTS] PRIMARY KEY NONCLUSTERED
	(
		[ID] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [emode_ind_g2]

)
ON  PARTI_SCHEM_SF(WEEK_KEY);

-- Referencement des clÈs etrangËres
ALTER TABLE [dbo].[SHOP_FACTS]   ADD  CONSTRAINT [s_f_arti_color_look_fk] FOREIGN KEY([ARTICLE_CODE],
[COLOR_CODE])
REFERENCES [dbo].[ARTICLE_COLOR_LOOKUP] ([ARTICLE_CODE], [COLOR_CODE]) ;


ALTER TABLE [dbo].[SHOP_FACTS]  ADD  CONSTRAINT [shop_facts_article_lookup_fk] FOREIGN
KEY([ARTICLE_CODE])
REFERENCES [dbo].[ARTICLE_LOOKUP] ([ARTICLE_CODE]);


ALTER TABLE [dbo].[SHOP_FACTS]  ADD  CONSTRAINT [shop_facts_calendar_yearlookup_fk] FOREIGN
KEY([WEEK_KEY])
REFERENCES [dbo].[CALENDAR_YEAR_LOOKUP] ([WEEK_KEY]);


ALTER TABLE [dbo].[SHOP_FACTS] ADD  CONSTRAINT [shop_facts_outlet_lookup_fk] FOREIGN KEY([SHOP_CODE])
REFERENCES [dbo].[OUTLET_LOOKUP] ([SHOP_CODE])


INSERT INTO SHOP_FACTS SELECT * FROM SHOP_FACTS_BACKUP_TEMP;
	-- Suppression de la table temporaire
	DROP TABLE SHOP_FACTS_BACKUP_TEMP;
GO
```