SAE 1.04 - Création d'une base de données

GOKCEN Bayram

Introduction :

Nous possédons une base de données. Il s'agit d'un tableau, contenant une liste de jeux vidéo avec leur genre (course, puzzle, aventure), leur date de sortie (l'année), leur plateforme (PS4, PS3), le nom de l'entreprise (Nintendo), leur nombre de ventes global (qui est aussi divisé par continent). Finalement ces jeux sont triés du plus vendu au moins vendu, ce qui leur donne un rang, le jeu le plus vendu est n°1 par exemple.

Nous devons donc utiliser les connaissances acquises dans la ressource R1.05 (Introduction aux bases de données et SQL) pour réaliser une étude qui s'appuie sur cette base de données.

Le travail principal consiste à, dans un premier temps, récupérer des données brutes au format CSV puis utiliser le langage SQL pour les analyser et les stocker dans une nouvelle base de données que nous allons concevoir. Pour dans un dernier temps, créer des requêtes SQL afin de produire des analyses sous forme graphique.

Sommaire :

Partie 1

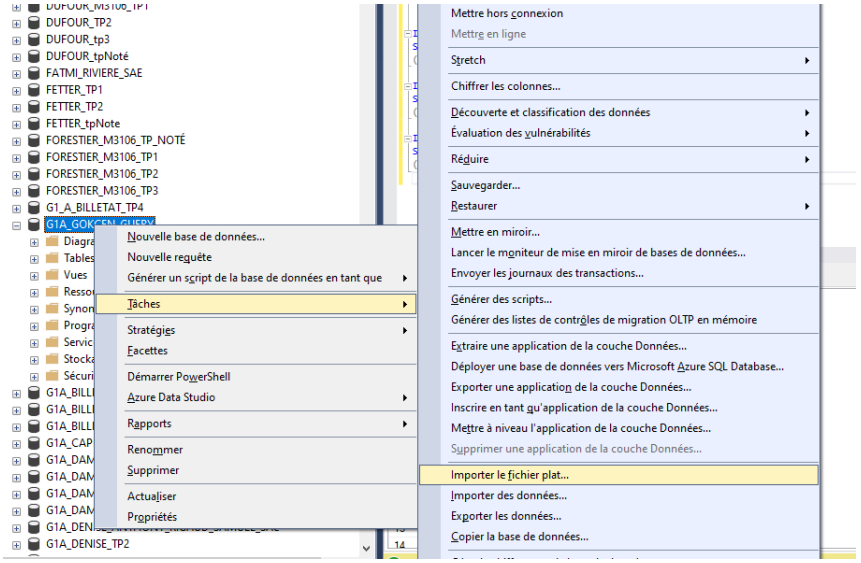
Nous allons dans cette partie, importer les données du tableau afin de les analyser. Ceci nous permettra de découvrir les problèmes et spécificités.

Importer le fichier texte dans une base de données :

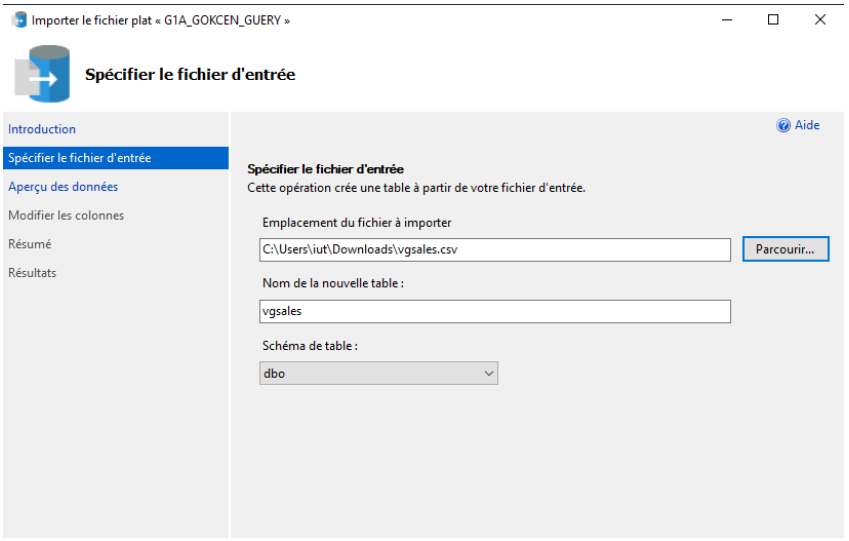
- ***Création d'une base de données :***

"create database G1A_GOKCEN_GUERY"

- ***Importation du fichier :***
  - Sur la base de données, précédemment créée, "Tâches", puis "Importer le fichier plat…"

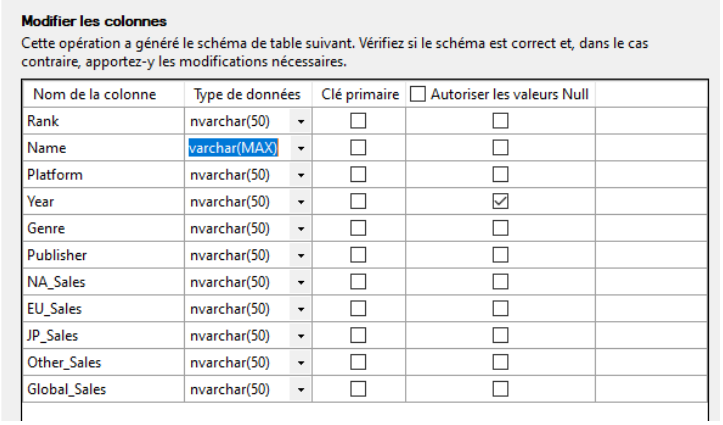


  - Importer le fichier "vgsales.csv"



- **Ce fichier possède quelques spécificités :**

Certains éléments de colonnes (nom du jeu) dépassent la limite de 50 caractères. On change donc nvarchar(50) à varchar(MAX) pour la catégorie “Name” (Ce type est provisoire)



- **Nous allons alors vérifier que les colonnes ne soient pas décalées (à cause d’une virgule dans un nom de jeu ou autre problème) :**
  - Pour ceci, nous allons convertir les colonnes dans des types spécifiques. Si la conversion rencontre des problèmes, on peut en déduire qu’il y a un problème dans la colonne.

Nous allons donc détailler le type et les spécificités de chaque colonne :

Colonnes :

- *Rang* : Il s’agit d’entiers donc le type est “int” et il faut faire attention car 2 rangs sont sautés. (Les rangs 654 et 14200.) Il n’y a finalement pas de redondance car en effet il s’agit de nombres uniques par cases. (contrainte unique).

- *Name* : Il s’agit de chaînes de caractères, qui peuvent aller jusqu’à 132 caractères. En effet, la chaîne la plus longue mesure 132 caractères.

```
SELECT MAX(LEN(Name)) as longName FROM vgsales
```

longName
132

Nous pouvons donc lui donner un type “varchar (132)”. Cette case ne peut pas être vide, en effet il s’agit d’une information indispensable pour identifier un jeu.

- *Platform :* Il s'agit de chaînes de caractères, qui peuvent aller jusqu'à 4 caractères.

```
SELECT MAX(LEN(Platform)) as longPlatform FROM vgsales
```

Nous pouvons donc lui donner un type varchar(4). Cette case ne peut pas être vide et il y a de nombreuses redondances.

longPlatform
4

- *Year* : Cette colonne est spéciale car même si elle doit être de type int, certaines dates sont inconnues et donc la valeur par défaut est N/A. C'est pourquoi le type "int" est inutilisable pour vérifier la case. Nous avons alors, de façon arbitraire, décidé de mettre comme valeur par défaut, la valeur 0 pour les cases null, en remplaçant "N/A" par 0. (Un autre moyen serait pour vérifier, de convertir toutes les valeurs sauf "N/A" en int.)

```
UPDATE vgsales
SET Year = REPLACE(Year, 'N/A', '0')
```

En effet, comme ça nous pouvons convertir la colonne en int. Finalement si nous devons créer une base de données avec cette colonne, nous pouvons utiliser la contrainte "default" avec N/A ou 0.

- *Genre* : Il s'agit de chaînes de caractères qui peuvent aller jusqu'à 12 caractères.

```
SELECT MAX(LEN(Genre)) as longGenre FROM vgsales
```

longGenre
12

Nous pouvons donc lui donner un type varchar(12). Cette case ne peut pas être vide et il y a de nombreuses redondances.

- *Publisher* : Il s'agit de chaînes de caractères pouvant aller jusqu'à 38 caractères.

```
SELECT MAX(LEN(Publisher)) as longPublisher FROM vgsales
```

longPublisher
38

- **_Sales* : Il s'agit d'un réel. Nous donnerons donc aux types relatant les ventes des jeux le type float. Il n'y a pas de redondance car chaque valeur dépend du jeu.

Finalement on peut essayer de convertir toutes les colonnes, dans les types trouvés précédemment :

```
select convert (float,NA_Sales) from vgsales
select convert (float,EU_Sales) from vgsales
select convert (float,JP_Sales) from vgsales
select convert (float,Other_Sales) from vgsales
select convert (float,Global_Sales) from vgsales
select convert (int,Rank) from vgsales
select convert (varchar(132),Name) from vgsales
select convert (varchar(4),Platform) from vgsales
select convert (int,Year) from vgsales
select convert (varchar(12),Genre) from vgsales
select convert (varchar(38),Publisher) from vgsales
```

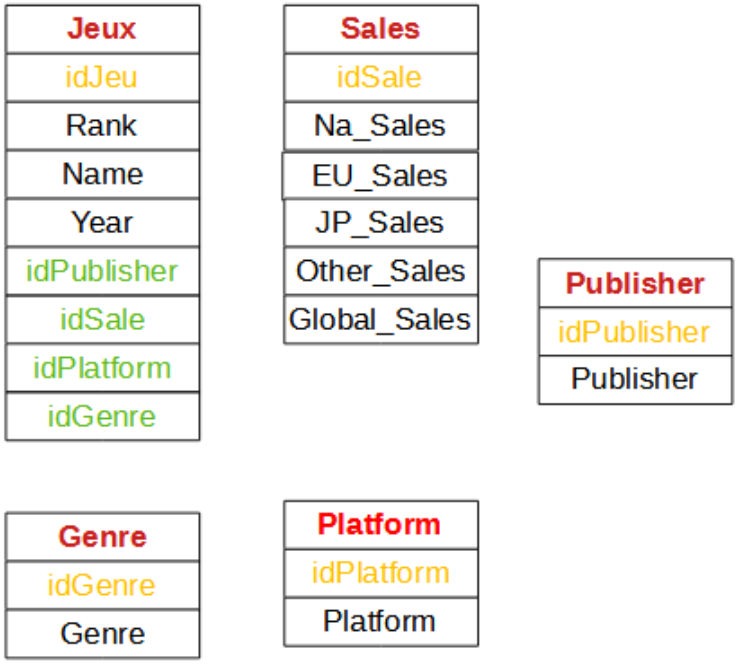
Nous remarquons qu'aucune erreur n'est rencontrée, en effet nous pouvons donc conjecturer qu'on a aucune erreur. Si nous avions des problèmes de décalage dans les colonnes, un message d'erreur serait envoyé pour les colonnes (Par exemple, pour les ventes, si un caractère alphabétique allait dans la colonne, il y aurait un problème de type).

Partie 2 :

Nous allons dès à présent concevoir une base de données avec des tables et contraintes adéquates. Dans ce but, nous allons dans un premier temps schématiser la situation, puis créer les tables avec les contraintes mûrement choisies, pour finalement les alimenter avec les données de notre première table.

Conception de la Base de Données

Schéma Base de Données



Nous avons alors créé une base de données, avec une table pour chaque information où il y a soit des redondances soit plusieurs colonnes similaires , on obtient au final 5 tables :

- ***Jeux*** : Il s'agit de la table qui réunit toutes les informations propres au jeu comme : leur nom, leur année de publication, leur rang et leur identifiant. Mais aussi, des clés secondaires, comme l'idPublisher, l'idSale, l'idPlatform ou l'idGenre, en effet il s'agit de liens permettant d'échanger avec les autres tables. Les informations comme, le nom, le rang sont des informations uniques et liées directement au jeu, c'est pourquoi on a décidé de mettre ces colonnes dans la table. L'année de création, quant à elle, aurait pû être dans une table à part cependant, nous avons pensé que celà créerait des

problèmes de lecture, nous aurions des identifiants d'années qui seraient encore d'autres nombres…

- ***Sales*** : Il s'agit de la table qui réunit toutes les informations concernant les ventes. On retrouve donc la clé primaire "idSale" qui est l'identifiant de vente, la colonne "Global_Sales" qui réunit le nombre de ventes global et les autre colonnes qui représentent les ventes dans différentes parties du monde : comme l'Europe "EU_Sales", le Japon "JP_Sales" ou autres "Other_Sales". Nous avons choisi de créer une table pour les ventes car il y avait, initialement, de nombreuses colonnes dans le tableau qui concernaient ces ventes et qu'on utilisait pas forcément. Ceci permet donc, de ne pas avoir dans la table "jeux", à se coltiner 5 colonnes liées à la même idée.

- ***Publisher*** : Cette table contient 2 colonnes, la première qui est la clé primaire est l'identifiant de l'éditeur qui permet, comme son nom l'indique de l'identifier, il est unique pour chaque éditeur, la seconde est le nom de l'éditeur. Cette table permet d'éviter la redondance car en effet, de nombreux jeux proviennent du même éditeur, c'est pourquoi on évite la redondance grâce à cette table.

- ***Platform*** : Cette table contient aussi 2 colonnes, la première (toujours clé primaire) est l'identifiant de la plateforme qui permet, comme son nom l'indique de l'identifier, il est unique pour chaque plateforme, la seconde est le nom de la plateforme. Cette table permet d'éviter la redondance car en effet, de nombreux jeux se jouent sur la même plateforme, c'est pourquoi on évite la redondance grâce à cette table. De plus, cette table nous permet de vérifier si certains jeux sont présents plusieurs fois dans la le tableau, car vendus sur plusieurs plateformes.

- ***Genre*** : Cette table contient tout comme les précédentes tables, 2 colonnes, la première est toujours l'identifiant étant une clé primaire, la seconde est le genre du jeu. Il n'y a pas une infinité de genres de jeux, de ce fait on évite la redondance grâce à cette table.

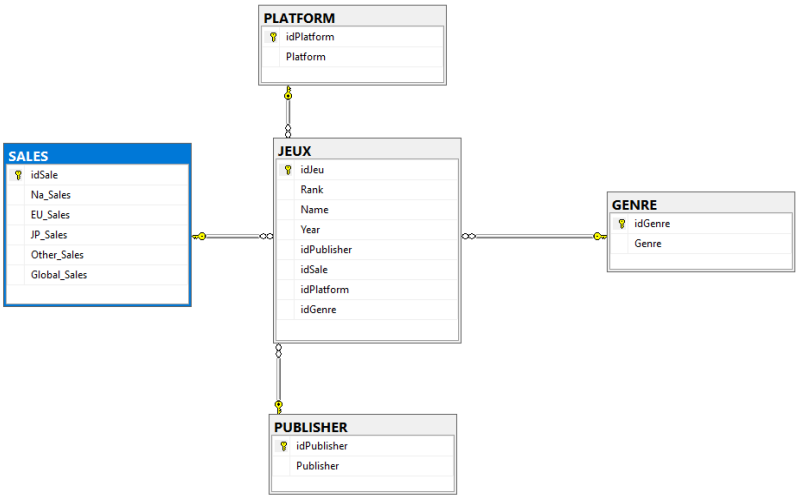
Scripts de création des tables avec les types et contraintes adéquats :

```
CREATE TABLE GENRE(
    idGenre int PRIMARY KEY NOT NULL,
    Genre varchar(12) NOT NULL
    )

CREATE TABLE PUBLISHER(
    idPublisher int PRIMARY KEY NOT NULL,
    Publisher varchar(38) NOT NULL
    )

CREATE TABLE PLATFORM(
    idPlatform int PRIMARY KEY NOT NULL,
    Platform varchar(4) NOT NULL
    )
```

```
CREATE TABLE SALES(
    idSale int PRIMARY KEY NOT NULL,
    Na_Sales float NOT NULL,
    EU_Sales float NOT NULL,
    JP_Sales float NOT NULL,
    Other_Sales float NOT NULL,
    Global_Sales float NOT NULL
)
```

```sql
CREATE TABLE JEUX(
    idJeu int PRIMARY KEY NOT NULL,
    Rank int NOT NULL,
    Name varchar(132) NOT NULL,
    Year int NOT NULL,
    idPublisher int FOREIGN KEY REFERENCES PUBLISHER(idPublisher) NOT NULL,
    idSale int FOREIGN KEY REFERENCES SALES(idSale) NOT NULL,
    idPlatform int FOREIGN KEY REFERENCES PLATFORM(idPlatform) NOT NULL,
    idGenre int FOREIGN KEY REFERENCES GENRE(idGenre) NOT NULL,
)
```

Schéma final :



Importation des données :

Nous pouvons dès à présent introduire dans nos différentes tables, les valeurs que nous possédons dans la table de la “Partie 1”, nous obtiendrons donc une base de données avec différentes tables cohérentes liées entre-elles et des colonnes possédant le bon type :

```sql
insert into GENRE
select ROW_NUMBER () over (order by Genre) as idGenre, Genre from
(select distinct Genre from vgsales) as Liste_des_genres

insert into PLATFORM
select ROW_NUMBER () over (order by Platform) as idPlatform, Platform from
(select distinct Platform from vgsales) as Liste_des_Platformes

insert into PUBLISHER
select ROW_NUMBER () over (order by Publisher) as idPublisher, Publisher from
(select distinct Publisher from vgsales) as Liste_des_Publishers

insert into SALES (idSale, Global_Sales,NA_Sales,EU_Sales, JP_Sales, Other_Sales)
select ROW_NUMBER () over (order by Global_Sales) as idSales, Global_Sales, NA_Sales, EU_Sales, JP_Sales, Other_Sales from
(select distinct Global_Sales, NA_Sales, EU_Sales, JP_Sales, Other_Sales  from vgsales) as Ventes
```

Nous avons dès à présent importer les données du tableau dans différentes tables, de plus, nous avons créé pour chaque table une clé primaire qui est un entier allant de 1 à n. De ce fait, grâce à ces clés primaires, nous évitons les redondances.

Par exemple, nous pouvons voir ci-contre qu'il existe une table plateforme avec les 31 plateformes et leur clé primaire (leur id). Ceci nous évite donc de répéter la même plateforme plusieurs fois pour tous les jeux.

	idPlatform	Platform
1	1	2600
2	2	3DO
3	3	3DS
4	4	DC
5	5	DS
6	6	GB
7	7	GBA
8	8	GC
9	9	GEN
10	10	GG
11	11	N64
12	12	NES
13	13	NG
14	14	PC
15	15	PCFX
16	16	PS
17	17	PS2
18	18	PS3
19	19	PS4
20	20	PSP
21	21	PSV
22	22	SAT
23	23	SCD
24	24	SNES
25	25	TG16
26	26	Wii
27	27	WiiU
28	28	WS
29	29	X360
30	30	XB
31	31	XOne

Ces clés primaires peuvent donc permettre d'identifier la plateforme qui leur est propre. Nous devons donc maintenant remplir la table principale qui est la table JEUX. C'est elle qui est le lien entre toutes les autres tables.

En somme, dans la table JEUX, les colonnes possèdent des clés secondaires qui sont liées aux clés primaires des autres colonnes qui sont liées aux données.

La dernière étape consiste donc logiquement à importer toutes les données dans la table "Jeux", il faut alors importer le nom, l'année, le rang et surtout toutes les clés secondaires qui la relie aux autres tables.

```
insert into JEUX (idJeu, Rank, Name, Year, idPublisher, idSale, idPlatform, idGenre)
select ROW_NUMBER () over (order by Rank) as idJeu, Rank, Name, Year, idPublisher, idSale, idPlatform, idGenre from
(select distinct Rank, Name, Year, PUBLISHER.idPublisher, SALES.idSale, PLATFORM.idPlatform, GENRE.idGenre from
vgsales, PUBLISHER, SALES, PLATFORM, GENRE
where PUBLISHER.publisher=vgsales.publisher and SALES.Global_Sales=vgsales.Global_Sales and SALES.EU_Sales=vgsales.EU_Sales and
SALES.Na_Sales=vgsales.NA_Sales and SALES.JP_Sales=vgsales.JP_Sales and SALES.Other_Sales=vgsales.Other_Sales and
PLATFORM.platform=vgsales.platform and GENRE.genre=vgsales.genre) as Rang_jeu
```

Pour ce faire nous utilisons la commande insert into, nous insérons, dans un premier temps, les 4 premières colonnes "idJeu, Rank, Name et Year" avec les données de la première table "vgsales". Puis dans un second temps, nous allons chercher dans chaque table précédemment créée les id, des colonnes, adéquats. Il suffit alors de mettre les clés secondaires dans le bon ordre. Chaque clé primaire des tables SALES, PUBLISHER, PLATFORM et GENRE est donc bien reliée à une clé secondaire de la table JEUX.

idJeu	Rank	Name	Year	idPublisher	idSale	idPlatform	idGenre
1	1	Wii Sports	2006	369	6761	26	11
2	2	Super Mario Br...	1985	369	6628	12	5
3	3	Mario Kart Wii	2008	369	6533	26	7
4	4	Wii Sports Resort	2009	369	6532	26	11
5	5	Pokemon Red/...	1996	369	6531	6	8
6	6	Tetris	1989	369	6530	6	6
7	7	New Super Mar...	2006	369	6529	5	5
8	8	Wii Play	2006	369	6358	26	4
9	9	New Super Mar...	2009	369	6357	26	5

Partie 3 :

La base de données est maintenant finie et utilisable, nous pouvons alors désormais produire des requêtes et en obtenir des résultats sous forme de graphiques.

1. *Nous allons, dans un premier temps, afficher la liste des plateformes ayant plus de 20M de jeux vendus en Europe depuis 2010 :*

```
select distinct PLATFORM.Platform from PLATFORM, SALES, JEUX where
SALES.idSale = JEUX.idSale and Platform.idPlatform = JEUX.idPlatform and
JEUX.Year>2010
group by (PLATFORM.platform)
having sum(SALES.EU_Sales)>20
```

Platform
PC
WiiU
X360
Wii
XOne
PS4
3DS
PS3

Nous nous servons donc des tables PLATFORM, SALES et JEUX que l'on lie grâce aux clés secondaires et primaires (SALES.idSale = JEUX.idSale and etc..). Nous provoquons alors une restriction aux jeux vendus seulement après 2010 (where JEUX.Year>2010) et aux plateformes qui possèdent plus de 20 millions de ventes en Europe (having sum(SALES.EU_Sales)>20).
Il suffit alors d'afficher de façon distinctes le nom des plateformes qui respectent ces conditions (select distinct PLATFORM.Platform).

2. Nous devons dès à présent, afficher la liste des éditeurs de jeux PS4 dépassant les 2M de jeux vendus quelle que soit l'année, le résultat sera trié par nombre de jeux vendus du plus grand au plus petit.

```
select distinct PUBLISHER.Publisher, sum(SALES.Global_Sales) as vente from PUBLISHER, SALES, JEUX, PLATFORM where
SALES.idSale = JEUX.idSale and Publisher.idPublisher = JEUX.idPublisher and JEUX.idPlatform=PLATFORM.idPlatform
and PLATFORM.Platform = 'PS4'
group by PUBLISHER.Publisher
having sum(SALES.Global_Sales)>2
order by sum(SALES.Global_Sales) desc
```

Nous nous servons donc des tables PLATFORM, SALES, JEUX et PUBLISHER que l'on lie grâce aux clés secondaires et primaires (SALES.idSale = JEUX.idSale and etc..).
Nous provoquons alors une restriction aux plateformes : nous gardons que la plateforme PS4 (where PLATFORM.Platform='PS4') et aux ventes globales : nous gardons que les éditeurs dont la somme des ventes est supérieure à 2 millions (having sum(SALES.Global_Sales)>2).
Il suffit alors d'afficher de façon distinctes le nom des éditeurs (et leur nombre de ventes) qui respectent ces conditions (select distinct PUBLISHER.Publisher, sum(SALES.Global_Sales)).

Publisher	vente
Electronic Arts	55.31999999999999
Activision	40.25999999999999
Ubisoft	31.66
Sony Computer Entertainment	30.71
Take-Two Interactive	25.310000000000002
Warner Bros. Interactive Entertainm	19.580000000000002
Bethesda Softworks	14.25
Namco Bandai Games	11.110000000000001
Square Enix	10.249999999999998
Sony Computer Entertainment Europe	7.52
Konami Digital Entertainment	6.930000000000001
Tecmo Koei	2.6100000000000003
Capcom	2.5999999999999996

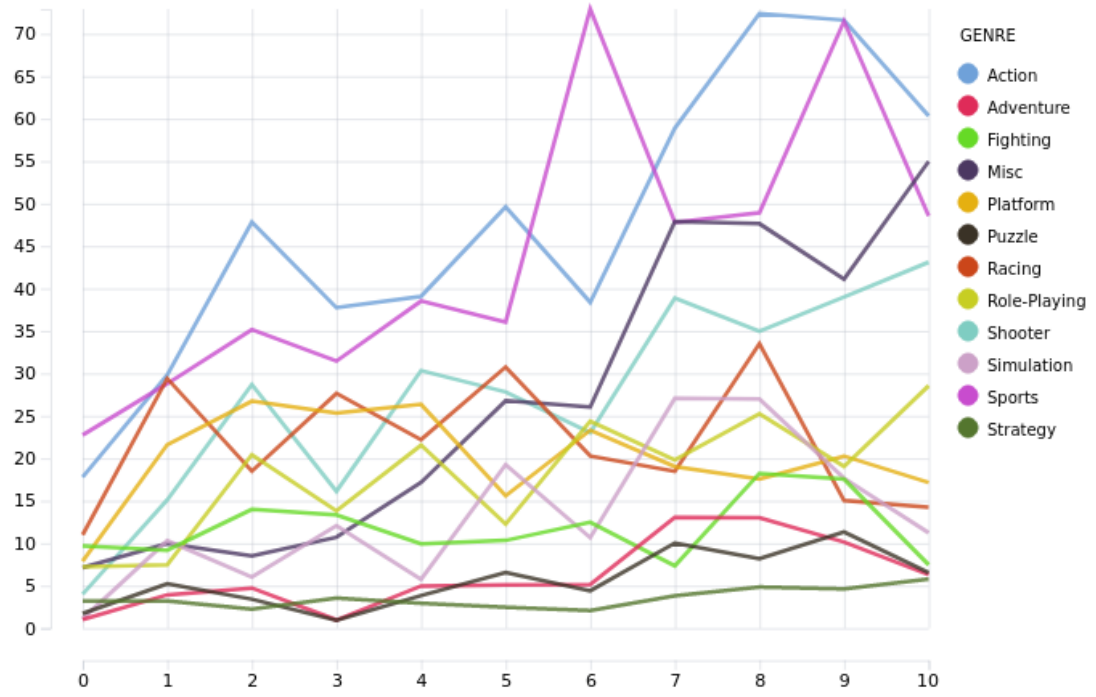
3. Nous pouvons aussi, montrer l'évolution du nombre de ventes de jeux par genre en Amérique du Nord sur la période 2000 à 2010 :

```
select distinct GENRE.Genre, (JEUX.Year-2000) as Année, sum (SALES.Na_Sales) as Vente from GENRE, SALES, JEUX where
GENRE.idGenre = JEUX.idGenre and SALES.idSale = JEUX.idSale
and JEUX.Year between '2000' and '2010'
group by GENRE.Genre, JEUX.Year
```

Pour faire ceci, nous nous servons donc des table GENRE, SALES et JEUX que l'on lie grâce aux clés secondaires et primaires (*from GENRE, SALES, JEUX where SALES.idSale = JEUX.idSale and* etc..). .
Nous provoquons alors une restriction à l'année de vente :
On veut que le jeu soit vendu entre 2000 et 2010 (*where JEUX.Year between '2000' and '2010'*)..
Il suffit alors de sélectionner et d'afficher :

- de façon distinctes le genre des jeux qui respectent ces conditions (*select distinct GENRE.Genre*)).
- L'année de ventes-2000, étant donné que l'on garde les jeux vendus de 2000 à 2010, si l'on soustrait on obtient donc les ventes de l'année 0 à 10 (avec année 0 = 2000 jusqu'à année 10=2010) (*select (JEUX.Year-2000*)
- Puis finalement le nombre de vente en Europe (celui-ci est la somme de toutes les ventes européenne regroupée par genre et puis année) (*select sum (SALES.Na_Sales) [...] group by GENRE.Genre, JEUX.Year*)

On met alors sur l'axe des abscisses : l'Année, celui des ordonnées : les Ventes et en couleur : le Genre.
On obtient alors ce graphique :



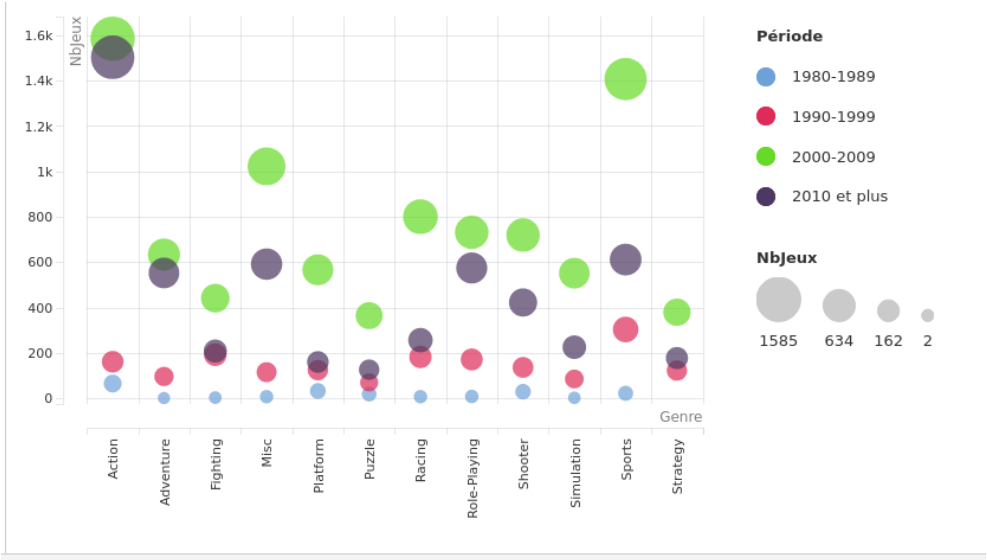
4. Nous allons afficher le nombre de jeux édités en fonction du genre par tranche de 10 ans (4 tranches : de 1980 à 1989, de 1990 à 1999, de 2000 à 2009, 2010 à maintenant) :

```
select distinct GENRE.Genre, count(*) as "NbJeux",case
        when JEUX.Year between '1980' and '1989' then '1980-1989'
        when JEUX.Year between '1990' and '1999' then '1990-1999'
        when JEUX.Year between '2000' and '2009' then '2000-2009'
        else '2010 et plus'
end as Période from GENRE,JEUX,SALES
where GENRE.idGenre = JEUX.idGenre and SALES.idSale = JEUX.idSale
group by case
        when JEUX.Year between '1980' and '1989' then '1980-1989'
        when JEUX.Year between '1990' and '1999' then '1990-1999'
        when JEUX.Year between '2000' and '2009' then '2000-2009'
        else '2010 et plus'
end, GENRE.Genre
```

Pour faire ceci, nous nous servons aussi des table GENRE, SALES et JEUX que l'on lie grâce aux clés secondaires et primaires (*from GENRE, SALES, JEUX where SALES.idSale = JEUX.idSale and* etc..) Il s'agit, en effet, des mêmes tables que pour la requête précédente.
Nous allons alors utiliser la commande case qui permet de créer une colonne avec des données qui dépendent de conditions.
Il faut alors sélectionner et afficher :

- de façon distincte le genre des jeux (*select distinct GENRE.Genre*).
- Le nombre de jeux édités regroupé par genre et par période de 10 ans. (select count(*) as "NbJeux" [...] group by case[...], GENRE.Genre). En effet, on compte alors le nombre de lignes pour chaque regroupement de Genre et de période.
- Puis finalement, on crée une colonne "Période" grâce à la commande case qui vaut selon l'année du jeu, une période. Cela nous permet alors de séparer chaque genre par période. (si le jeu est un puzzle édité en 1992, il est ajouté au Puzzle de la période "1990-1999")

On met alors sur l'axe des abscisses: le genre, celui des ordonnées: le nombre de jeux et en couleur : la période:



Il ne faut, cependant, pas oublier que quand les données ont été recueillies, on était en 2016. La décennie n'était donc pas terminée c'est pourquoi on peut remarquer que pour la plupart des jeux, après 2010, le nombre de ventes est inférieur à la période 2000-2009.

5. On peut, finalement, réaliser deux requêtes au choix et construire les graphiques adéquats :

Première requête :

Nous avons tous, quand nous étions petits, voulu savoir quelle était la plateforme qui vendait le plus de jeux.
Notre premier script aura alors pour objectif de comparer le nombre de ventes global de chaque plateforme, afin de savoir quelle plateforme a réalisé le plus de ventes et laquelle, le moins, sans limite de période.

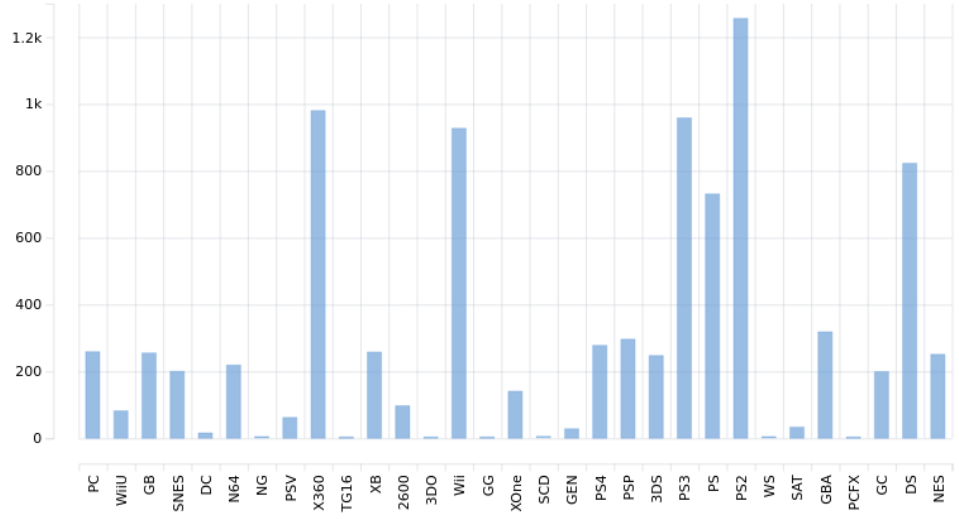
```
1  select PLATFORM.Platform, sum(SALES.Global_Sales) Vente
2  from SALES,PLATFORM,JEUX
3  where SALES.idSale = JEUX.idSale and PLATFORM.idPlatform = JEUX.idPlatform
4  group by PLATFORM.Platform
```

Pour réaliser cette requête, nous utilisons les table PLATFORM, SALES et JEUX que l'on lie grâce aux clés secondaires et primaires *(from GENRE, SALES, JEUX where SALES.idSale = JEUX.idSale and etc..)*
Il suffit alors de sélectionner et d'afficher :

- la colonne plateforme où sont listées toutes les plateformes. (*select PLATFORM.Platform*).
- Ainsi que la colonne des ventes globaux qui seront regroupés par plateforme. Nous obtenons alors la somme des ventes par plateforme. (*select sum(SALES.Global_Sales) group by PLATFORM.Platform*)

Finalement, nous obtenons deux colonnes, l'une avec les plateformes et l'autre avec les ventes réalisées par plateforme.

Quand on met en abscisses les plateformes et en ordonnées les ventes, on obtient ce graphique :



Nous pouvons alors enfin répondre, à cette question, d'après le graphique, la plateforme ayant vendu le plus de jeux est la PS2, suivie par la XBOX 360, puis la Wii.

Deuxième requête :

Nous avons remarqué pendant la création de la table qu'il y avait un grand nombre d'éditeurs différents qui ont été ajoutés (579 plus précisément). Cependant, quand nous regardons les jeux les plus vendus, nous pouvons voir que ce sont souvent les mêmes éditeurs (Nintendo, Activision etc…).
Nous nous sommes alors demandés, quelle part des ventes totales représentent les 7 plus grands éditeurs de 2000 à 2010.
Nous avons choisis les éditeurs connus suivants : Nintendo, Activision, Electronic Arts, Sony Computer Entertainment, Ubisoft, Take-Two Interactive, Microsoft Games Studio et Sega.

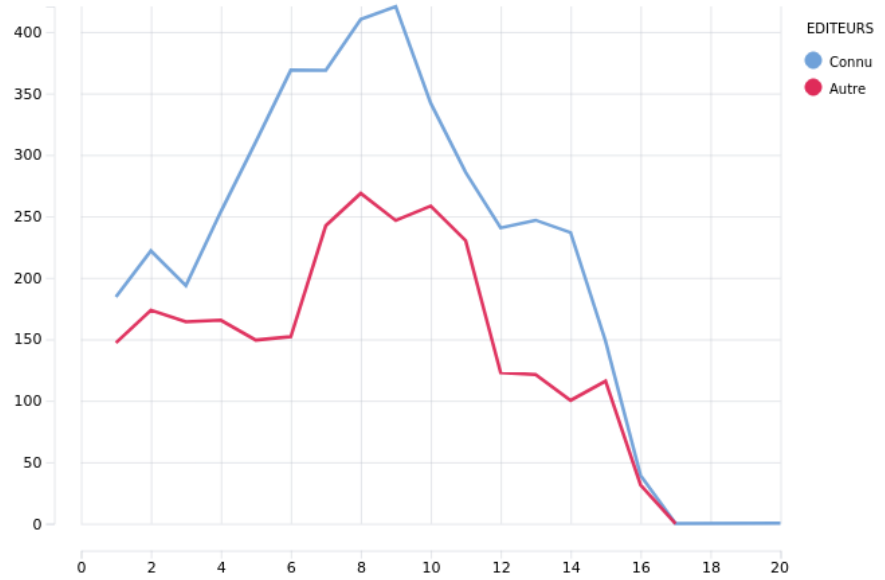
```
select sum(SALES.Global_Sales) as Vente, (JEUX.Year-2000) as Année, case
        when PUBLISHER.Publisher='Nintendo'
        or PUBLISHER.Publisher='Activision'
        or PUBLISHER.Publisher='Electronic Arts'
        or PUBLISHER.Publisher='Sony Computer Entertainment'
        or PUBLISHER.Publisher='Ubisoft'
        or PUBLISHER.Publisher='Take-Two Interactive'
        or PUBLISHER.Publisher='Microsoft Games Studios'
        or PUBLISHER.Publisher='Sega' then 'Connu'
        else 'Autre'
end as Editeurs from PUBLISHER,JEUX,SALES
where PUBLISHER.idPublisher = JEUX.idPublisher and SALES.idSale = JEUX.idSale and Year>2000
group by case
        when PUBLISHER.Publisher='Nintendo'
        or PUBLISHER.Publisher='Activision'
        or PUBLISHER.Publisher='Electronic Arts'
        or PUBLISHER.Publisher='Sony Computer Entertainment'
        or PUBLISHER.Publisher='Ubisoft'
        or PUBLISHER.Publisher='Take-Two Interactive'
        or PUBLISHER.Publisher='Microsoft Games Studios'
        or PUBLISHER.Publisher='Sega' then 'Connu'
        else 'Autre'
end,(JEUX.Year-2000)
```

Pour faire ceci, nous nous servons donc des table PUBLISHER, SALES et JEUX que l'on lie grâce aux clés secondaires et primaires (*from PUBLISHER, SALES, JEUX where SALES.idSale = JEUX.idSale and* etc..). .
Nous provoquons alors une restriction à l'année de vente :
On veut que le jeu soit vendu après 2000 (*where JEUX.Year > 2000)*. On a donc les jeux vendus entre 2000 et maintenant.
Il faut alors sélectionner et afficher :

- la somme des ventes globales des jeux par année et par Éditeur (connu ou pas connu). Pour ceci, on utilise (*select sum(SALES.Global_Sales as Vente group by case[...], (JEUX.Year-2000)*). Par exemple, le nombre de ventes d'un jeu édité par Activision en 2004 est ajouté au nombre de Vente de l'an 4 (pour 2004) de "éditeur connu".
- L'année de ventes-2000, étant donné que l'on garde les jeux vendus après 2000, si l'on soustrait on obtient donc les ventes de l'année 0 maintenant (avec année 0 = 2000 jusqu'à année 16=2016). En effet, après les résultats sont faussés car il n'y a pas de données. (*select (JEUX.Year-2000*)
- Puis finalement, on crée une colonne "Editeur" grâce à la commande case qui vaut selon l'éditeur : "connu" ou "autre". En effet, cette colonne vaut alors "Connu" si l'éditeur fait partie de la liste ci-dessus et "Autre" si ce n'est pas le cas.

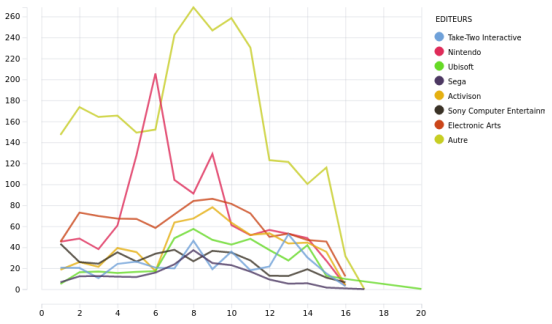
Nous avons donc au final 3 colonnes, avec l'éditeur (connu ou pas connu), l'année (de 0 à 16) et le nombre de ventes (par rapport à l'année et l'éditeur).

Quand on met les années en abscisses et les ventes en ordonnées, on obtient ce graphique-ci :



On peut alors voir que le nombre de ventes des 7 éditeurs les plus connus est plus élevé durant 16 ans que le nombre de ventes de tous les autres éditeurs réunis.
Ce qui prouve qu'au final c'est une poignée d'éditeurs qui ont le contrôle sur le monde des jeux-vidéos.

```sql
select sum(SALES.Global_Sales) as Vente, (JEUX.Year-2000) as Année, case
       when PUBLISHER.Publisher='Nintendo' then 'Nintendo'
       when PUBLISHER.Publisher='Activision' then 'Activison'
       when PUBLISHER.Publisher='Electronic Arts' then 'Electronic Arts'
       when PUBLISHER.Publisher='Sony Computer Entertainment' then 'Sony Computer Entertainment'
       when PUBLISHER.Publisher='Ubisoft' then 'Ubisoft'
       when PUBLISHER.Publisher='Take-Two Interactive' then 'Take-Two Interactive'
       when PUBLISHER.Publisher='Microsoft Games Studios' then 'Microsoft Games Studios'
       when PUBLISHER.Publisher='Sega' then 'Sega'
       else 'Autre'
end as Editeurs from PUBLISHER,JEUX,SALES
where PUBLISHER.idPublisher = JEUX.idPublisher and SALES.idSale = JEUX.idSale and Year>2000
group by case
       when PUBLISHER.Publisher='Nintendo' then 'Nintendo'
       when PUBLISHER.Publisher='Activision' then 'Activison'
       when PUBLISHER.Publisher='Electronic Arts' then 'Electronic Arts'
       when PUBLISHER.Publisher='Sony Computer Entertainment' then 'Sony Computer Entertainment'
       when PUBLISHER.Publisher='Ubisoft' then 'Ubisoft'
       when PUBLISHER.Publisher='Take-Two Interactive' then 'Take-Two Interactive'
       when PUBLISHER.Publisher='Microsoft Games Studios' then 'Microsoft Games Studios'
       when PUBLISHER.Publisher='Sega' then 'Sega'
       else 'Autre'
end,(JEUX.Year-2000)
```



Nous pouvons aussi voir, en changeant un peu la requête que, plus précisément, les éditeurs comme Nintendo ou Electronic Arts possèdent une très grosse part du marché.