Projet Entrepôt de données – HAI708I

Antoinette Laurent, Boshkovska Marta, Campillo Romain, Dovi Late Laurencia

L'objectif de ce projet est la conception, l'implémentation et l'interrogation d'un entrepôt de données. En choisissant Spotify comme illustration pour notre projet d'entrepôt de données, nous avons pour objectif d'exploiter les données disponibles afin de satisfaire davantage la clientèle et stimuler la croissance du chiffre d'affaires de cette entreprise.

Analyse

Nous considérons que Spotify nous a demandé d'imaginer un entrepôt de données qui lui permettrait de soutenir son système d'aide à la décision. Nous devons mettre en place une infrastructure solide permettant la collecte, le stockage et l'analyse efficaces des données issues de sa vaste communauté d'utilisateurs. Nous pourrons alors exploiter judicieusement ces données et Spotify serait en mesure d'améliorer la personnalisation de ses recommandations musicales, d'optimiser ses stratégies de marketing, et d'éclairer de manière approfondie ses processus décisionnels.

Spotify est un service de streaming musical. Cette plateforme de distribution numérique permet une écoute quasi instantanée de plus de 40 millions de fichiers musicaux. Avec plus de 500 millions d'utilisateurs actifs mensuels, il s'agit du service de musique à la demande le plus utilisé.

Spotify est financé en partie par la publicité (modèle du freemium). Les utilisateurs peuvent payer un abonnement mensuel, leur donnant ainsi le statut « d'utilisateur premium » qui leur permet d'avoir une interface sans publicité (visuelle et sonore).

Les objectifs du service suédois sont d'améliorer l'expérience client et d'augmenter le chiffre d'affaire. Il faudrait obtenir des informations d'une part sur ce que les gens aiment écouter et d'autre part sur le nombre de publicités regardées ou le nombre d'abonnement pris.

Pour récupérer ces informations nous proposons différentes actions:

- les écoutes (un utilisateur écoute une musique)
- les abonnements (un utilisateur a acheté un abonnement)
- les publicités (un utilisateur a regardé une publicité)

Pour chacune de ces actions nous avons pensé à différents traitements possibles.

Pour les écoutes :
- Le nombre d'écoute par artiste chaque mois.
- Le nombre d'écoute par musique par mois.
- La durée moyenne d'écoute par genre pour chaque utilisateur.

Pour les abonnements :
- Le revenu total de chaque abonnement.
- Le nombre d'abonnés selon le type d'utilisateur.
- Le nombre d'abonné pour chaque type de promotion.

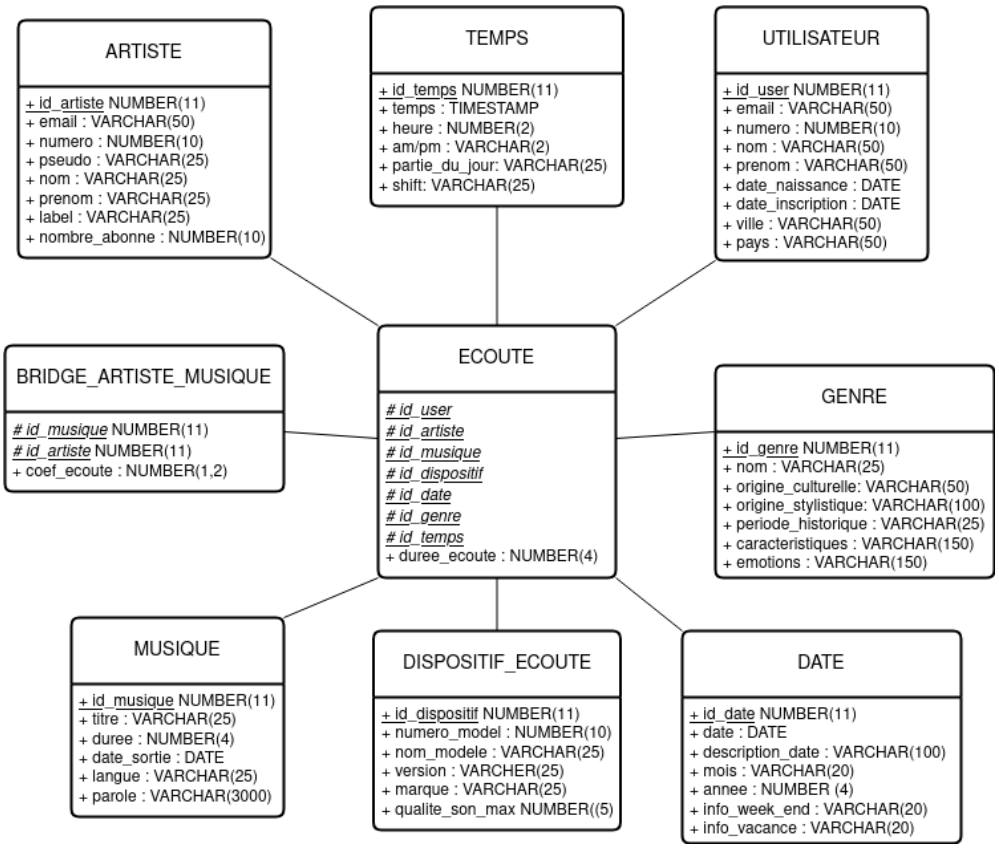
Pour les publicités :
- Le bénéfice généré de chaque pub à Spotify (dans le sens où d'autres marques payent pour pouvoir montrer cette publicité)
- Le nombre de clic par publicité.
- Le nombre de clic par type d'utilisateur.

Nous pensons que l'expérience utilisateur est plus importante que le chiffre d'affaire puisque Spotify possède déjà une position confortable dans le marché.
Les abonnements génèrent plus d'argent que les publicités donc l'ordre d'importance de nos actions est: d'abord les écoutes, ensuite les abonnements et enfin les publicités.

Conception

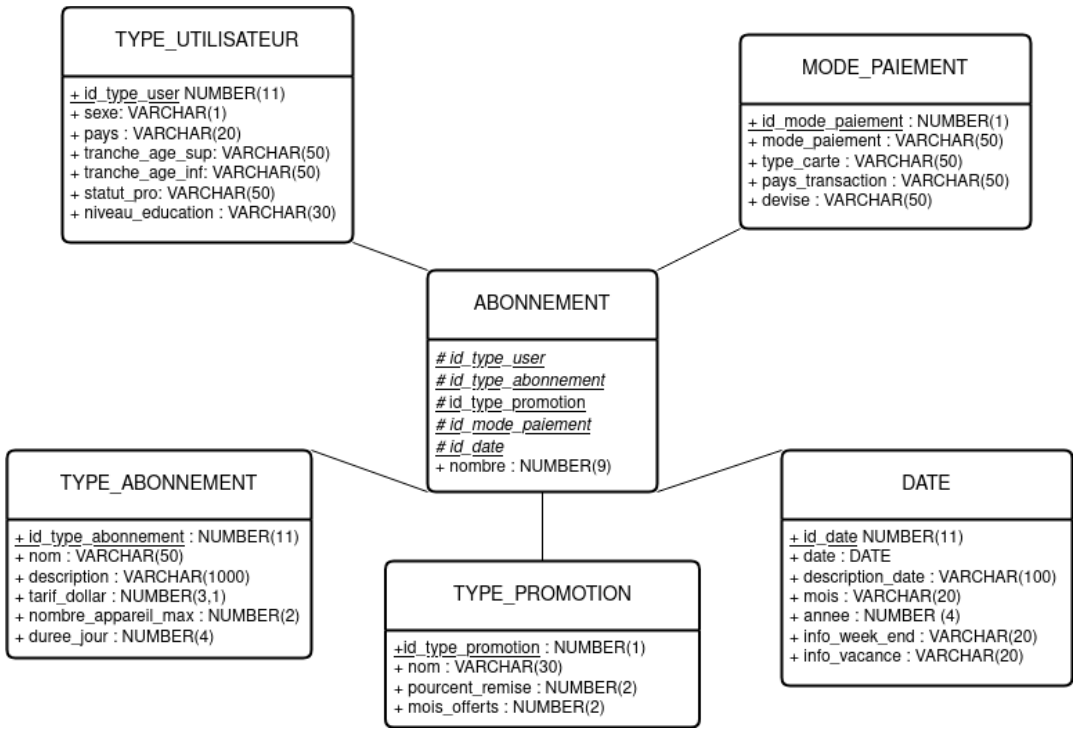
Les écoutes et les abonnements seront les deux actions que nous analyserons. Il s'agira d'une part de comprendre ce qu'attendent les utilisateurs et d'autre part de comprendre ce qui fait que les utilisateurs renouvellent leur abonnement.

Ce modèle en étoile représente la conception de l'entrepôt de données pour la table de fait « écoute »



Une écoute d'un utilisateur correspondra à n lignes correspondants aux n artistes qui ont produit la musique. On enregistre pour chaque ligne la durée réelle d'écoute.
Une table bridge permet de stocker un coefficient dans le cas où nous voulons compter les écoutes d'une musique produite par plusieurs artistes (pour ne pas compter plusieurs tuples qui désignent la même écoute)

La durée d'écoute est additive (si on considère la multiplication par le coefficient si par exemple on voudrait le temps total d'écoute d'une musique)



Une ligne de la table abonnement existe pour chaque combinaison de type d'utilisateur, de type d'abonnement, de type de promotion, de mode de paiement et de jour.
Le nombre d'abonnement est une mesure non-additive puisqu'il s'agit du nombre d'abonnement en cours de validité (à l'image d'un stock)

Avec la modélisation des écoutes, on peut savoir :

- **Le nombre d'écoute par artiste chaque mois.**

jointure entre écoute et artiste puis jointure avec date pour group by par mois et ensuite count le nombre de lignes

- **Le nombre d'écoute par musique par mois.**

jointure entre écoute et date pour group by par le mois. Ensuite jointure avec la table bridge pour récupérer le coefficient et ne pas compter plusieurs fois la meme écoute puis count * coeff

- **La durée moyenne d'écoute par genre pour chaque utilisateur.**

jointure entre écoute et utilisateur puis avec genre puis average de toutes les durées d'écoutes

Grâce à la modélisation des abonnements on peut savoir :

- **Le nombre d'abonnés selon le type d'utilisateur**

jointure entre abonnement et type utilisateur puis group by selon des attributs des utilisateurs qui nous intéressent.

- **Le nombre d'abonnés pour chaque type de promotion**

jointure entre abonnement et type promotion puis group by selon les attributs qui nous intéressent.

- **Combien nous rapporte chaque type d'abonnement (abonnement étudiant, familial, solo etc.)**

jointure entre abonnement et type d'abonnement puis on somme sur chaque type d'abonnement

Exemple d'instance de l'entrepôt de données

Table des faits Écoute

Id User	Id Artiste	Id Musique	Id Dispositif	Id Date	Id Temps	Id Genre	Duree Ecoute
1	3	3	1	1	1	3	35
2	2	2	3	1	2	2	240
1	1	1	1	1	4	1	280
1	4	1	1	1	4	1	280
3	3	3	2	2	3	3	14
2	1	1	3	2	3	1	300
2	4	1	3	2	3	1	300
1	1	1	1	3	4	1	5
1	4	1	1	3	4	1	5
3	3	3	2	3	2	3	268

Dimension Utilisateur

Id User	Nom	Prenom	Date de Naissance	Adresse mail	Pays	Ville	Code postal	Numero telephone
1	Antoinetto	Laurent	07/11/02	Laurent.antoinetto@gmail.com	France	Montpellier	34090	123456789
2	Guyard	Luc	30/09/98	Luc.guyard@yahoo.com	France	Lyon	69000	987654321
3	Yang	Xue	19/02/96	Xue.yang@gmail.com	Chine	Canton	510000	561741846

Dimension Date

Id Date	Date	Description date	Mois	Annee	InfoWeekend	Info Vacances
1	09/11/2023	9 novembre 2023	Novembre	2023	Non	Non
2	10/11/2023	9 novembre 2023	Novembre	2023	Non	Non
3	11/11/2023	10 novembre 2023	Novembre	2023	Oui	Non

Dimension dispositif

Id dispositif	Nom modèle	Version	Marque	Qualité son max
1	Iphone 14	Pro	Apple	123
2	Redmi Note	7	Xiaomi	64
3	Soundlink	Neutre	Bose	256
4	Freebuds	4i	Huawei	128

Dimension Musique

Id musique	Titre	Duree	Date de sortie	Langue	parole
1	Inachevé	300	15/02/15	Français	blabla
2	Blank Space	240	30/10/14	Anglais	blabla2
3	All I know	268	08/07/18	Anglais	blabla3

Dimension Artiste

Id artiste	Nom artiste	Nom	Prenom	Date Inscription	Website	Pays	ville
1	Orelsan	Cotentin	Aurélien	12/08/12	Avnier.com	France	Caen
2	Taylor Swift	Swift	Taylor	11/11/08	TaylorSwift.com	Etats-Unis	West Reading
3	The Weeknd	Testaye	Abel	15/12/08	Theweeknd.com	Canada	Toronto
4	Gringe	Guillaume	Tranchant	12/08/12	Avnier.com	France	Poitiers

Dimension Temps

Id temps	Temps	Heure	am/pm	Moment de la journée	shift
1	7h	7	am	matin	oui
2	12h	12	pm	midi	non
3	18h	18	pm	soir	oui
4	22h	22	pm	nuit	non

Dimension Genre Musique

Id genre	Nom	Origine culturelle	Origine stylistique	Periode historique	Caractéristique	émotions
1	Rap	Bronx New York jsp	Poésie parlée	1970	Rimes, rythmes rapides	Colère, tristesse
2	Pop	Musique populaire américaine	Variée (jsp quoi mettre)	1950	Mélodies simples,	Joie, fête, romantique
3	R&B	Etats-Unis	Blues, jazz	1940-1960	Voix puissante	Romantisme, mélancolie

Table pont entre artiste et musique

Id Artiste	Id Musique	Coeff Participation
1	1	0.5
4	1	0.5

Taille de l'entrepôt.

On estime le nombre de tuples pour les deux tables de faits sur 12 mois afin de rendre compte de la différence entre le volume d'une base de données et d'un entrepôt de données.

Pour le nombre d'écoute :

En moyenne, les utilisateurs de Spotify passent plus de 25 heures par mois à écouter de la musique (https://blogpascher.com/ressources/statistiques-spotify)

La durée moyenne d'une musique est de 3 minutes 45 selon KDS WORLD Radio (ce qui fait environ 400 musiques écoutées par mois)

Par ailleurs, plus de la moitié des des musiques ne sont pas écoutées en entier. (https://www.lesechos.fr/tech-medias/medias/spotify-dix-ans-en-dix-chiffres-141771)

Sachant qu'environ 500 millions d'utilisateurs sont actifs par mois, cela fait :
500 000 000 x 12 x 400 x 2 = **4 800 000 000 000 lignes par an (4 800 milliards)**
On imagine que les attributs de la table (donc les clés étrangères) sont stockés sur 8 octets. Avec 8 attributs, une ligne est donc stockée sur 64 octets ce qui donne un total de **307 200 000 000 000 octets par an (= 307 To).**

Ce résultat paraît conséquent mais pas incohérent compte tenu du nombre d'utilisateurs important et des ressources de l'entreprise.

Pour le nombre d'abonnement:

En un an, il y a autant de lignes de combinaisons possibles entre les types d'utilisateurs, les abonnements, les types de promotions, les modes de paiement et de jour de l'année.

La dimension type utilisateur est une mini-dimension qui est une combinaison selon le genre de l'individu, son pays, sa tranche d'âge, son niveau d'étude et sa catégorie professionnelle.
En simplifiant on trouve environ 100 000 types d'utilisateurs différents.

Il y a environ 10 promotions différentes et pour chaque type de promotion on les précise en disant si c'est une remise e 5%, 10%, 15% ... Ou s'il s'agit d'un mois offert, de deux mois offerts...
Il y aura en tout environ 2000 types de promotions différentes

On considère qu'il y a environ 1000 moyens de paiement différents (combinaison entre mode de paiement, type de carte, devise...)

On arrondit le nombre d'abonnement à 10.

On a donc : 100 000 * 2000 * 1000 * 10 * 365 = **730 000 000 000 000 lignes par an**
Comme pour le nombre d'écoute, on considère qu'un attribut est stocké sur 8 octets (6 attributs ici donc une ligne est stockée sur 48 octets).
En un an, cela représente : **35 040 000 000 000 000 octets (=35 040 To).**
Encore une fois le résultat est considérable mais puisqu'il s'agit d'une table de fait snapshot il n'est pas incohérent d'obtenir un volume de données beaucoup plus important.

Implantation

Nous avons implémenter sur Oracle les deux tables de faits écoute et abonnement modélisées plus haut.

```
CREATE TABLE ECOUTE (
    id_user NUMBER(11) REFERENCES UTILISATEUR(id_user),
    id_musique NUMBER(11) REFERENCES MUSIQUE(id_musique),
    id_artiste NUMBER(11) REFERENCES ARTISTE(id_artiste),
    id_genre NUMBER(11) REFERENCES GENRE(id_genre),
    id_dispositif NUMBER(11) REFERENCES DISPOSITIF_ECOUTE(id_dispositif),
    id_date NUMBER(11) REFERENCES UNE_DATE(id_date),
    id_temps NUMBER(11) REFERENCES TEMPS(id_temps),
    duree_ecoute NUMBER(4),
    PRIMARY
KEY(id_user,id_musique,id_artiste,id_genre,id_dispositif,id_date,id_temps)
);
```

```
CREATE TABLE ABONNEMENT(
    id_type_utilisateur NUMBER(11) REFERENCES
TYPE_UTILISATEUR(id_type_utilisateur),
    id_type_abonnement NUMBER(11) REFERENCES
TYPE_ABONNEMENT(id_type_abonnement),
    id_type_promotion NUMBER(11) REFERENCES TYPE_PROMOTION(id_type_promotion),
    id_mode_paiment NUMBER(11) REFERENCES MODE_PAIMENT(id_mode_paiment),
    id_date NUMBER(11) REFERENCES UNE_DATE(id_date),
    nombre_abonnement NUMBER(4),
    PRIMARY
KEY(id_type_utilisateur,id_type_abonnement,id_type_promotion,id_mode_paiment,i
d_date)
);
```

```
CREATE VIEW VUE_UNE_DATE AS
SELECT
  id_date,
  date_col,
  description_date,
  jour,
  mois,
  annee,
  info_week_end,
  info_vacance
FROM UNE_DATE;
```

Après un certain nombre d'insertion, nous pouvons exécuter certaines requêtes utiles à la prise de décision.

Requêtes pour les écoutes :

- **Le nombre d'écoute par genre et par artiste**

(Pour comprendre quels artistes sont les plus écoutés pour un genre de musique spécifique et pouvoir les recommander à ceux qui consomment ce genre de musique)

```sql
-- le nombre total d'écoute par genre par artiste
SELECT G.genre, A.pseudo,COUNT(*) AS duree_moyenne
FROM ECOUTE E
JOIN ARTISTE A ON A.id_artiste=E.id_artiste
JOIN GENRE G ON G.id_genre=E.id_genre
GROUP BY ROLLUP(G.genre, A.pseudo);
```

- **Le nombre d'écoute pour chaque genre de musique par heure et par mois**

(Pour comprendre les habitudes d'écoute des utilisateurs pour adapter les recommandations (qu'est ce qu'ils écoutent le matin en été? Le soir en hiver?))

```sql
-- le nombre d'écoute pour chaque genre de musique par heure par mois par annee
SELECT id_genre, UD.annee, UD.mois, T.heure, SUM(BAM.coef_ecoute) AS nombre_ecoutes
FROM ECOUTE E
JOIN UNE_DATE UD ON E.id_date = UD.id_date
JOIN TEMPS T ON E.id_temps = T.id_temps
JOIN BRIDGE_ARTISTE_MUSIQUE BAM ON BAM.id_artiste=E.id_artiste AND BAM.id_musique=E.id_musique
GROUP BY id_genre, UD.annee, UD.mois, T.heure;
```

- Le nombre d'écoute par musique par mois.

(Repérer les musiques les plus écoutées pour faire des listes de lectures basées sur les musique tendances)

```sql
-- le nombre d'écoute pour chaque genre de musique par heure par mois par annee
SELECT id_genre, UD.annee, UD.mois, T.heure, SUM(BAM.coef_ecoute) AS nombre_ecoutes
FROM ECOUTE E
JOIN UNE_DATE UD ON E.id_date = UD.id_date
JOIN TEMPS T ON E.id_temps = T.id_temps
JOIN BRIDGE_ARTISTE_MUSIQUE BAM ON BAM.id_artiste=E.id_artiste AND BAM.id_musique=E.id_musique
GROUP BY id_genre, UD.annee, UD.mois, T.heure;
```

- La durée moyenne d'écoute des genres pour chaque utilisateur.

(Connaître les préférences des utilisateurs et pouvoir améliorer la personnalisation des recommandations)

```sql
--la durée moyenne d'écoute par genre par utilisateur


SELECT E.id_user, E.id_genre, AVG(E.duree_ecoute*BAM.coef_ecoute) AS duree_moyenne
FROM ECOUTE E
JOIN BRIDGE_ARTISTE_MUSIQUE BAM ON BAM.id_artiste=E.id_artiste AND BAM.id_musique=E.id_musique
GROUP BY E.id_user, E.id_genre;
```

- Le nombre d'écoute par pays des utilisateurs pour chaque genre de musique.

(Comprendre les habitudes culturelles et pouvoir affiner la personnalisation des recommandation)

```sql
--le nombre d'écoute par pays des utilisateurs pour chaque genre de musique

SELECT U.pays AS Pays_Utilisateur, G.nom AS Genre_Musical, ROUND(SUM(BAM.coef_ecoute)) AS Nombre_Ecoutes
FROM ECOUTE E
JOIN UTILISATEUR U ON E.id_user = U.id_user
JOIN GENRE G ON E.id_genre = G.id_genre
JOIN BRIDGE_ARTISTE_MUSIQUE BAM on BAM.id_artiste=E.id_artiste AND BAM.id_musique=E.id_musique
GROUP BY U.pays, G.nom, BAM.coef_ecoute
ORDER BY U.pays, G.nom, Nombre_Ecoutes DESC;
```

- La durée totale d'écoute par mois pour chaque utilisateur.

(identifier les utilisateurs qui passent le plus de temps sur la plateforme. Cela peut être utile pour reconnaître les utilisateurs les plus engagés et leur offrir des récompenses, des offres spéciales pour leur fidélité.)

```sql
-- la durée totale d'écoute des utilisateurs par mois

SELECT U.id_user AS ID_Utilisateur, U.nom AS Nom_Utilisateur, UD.mois,
SUM(E.duree_ecoute*BAM.coef_ecoute) AS Duree_Ecoute_Totale
FROM ECOUTE E
JOIN UTILISATEUR U ON E.id_user = U.id_user
JOIN UNE_DATE UD ON E.id_date = UD.id_date
JOIN BRIDGE_ARTISTE_MUSIQUE BAM on BAM.id_artiste=E.id_artiste AND BAM.id_musique=E.id_musique
GROUP BY U.id_user, U.nom, UD.mois, BAM.coef_ecoute
ORDER BY U.id_user,  Mois, Duree_Ecoute_Totale DESC;
```

Requêtes pour les abonnements:

- Le nombre d'abonné pour chaque abonnement selon le type d'utilisateur

(Évaluer le type d'abonnement le plus souscrit par chaque type d'utilisateur (pour adapter des campagnes de pub pour des promotions susceptibles d'intéresser les bons types d'utilisateurs)

```sql
-- le nombre d'abonné pour chaque abonnement selon le type d'utilisateur par mois par annee

SELECT A.id_type_abonnement, A.id_type_utilisateur, UD.annee, UD.mois, SUM(nombre_abonnement) AS
nombre_abonne
FROM ABONNEMENT A
JOIN UNE_DATE UD ON A.id_date=UD.id_date
WHERE UD.jour = 01
GROUP BY A.id_type_abonnement, A.id_type_utilisateur, UD.annee, UD.mois;
```

- Le gain obtenu pour chaque abonnement par mois.

(Évaluer et suivre l'évolution du chiffre d'affaires généré de chaque abonnement)

```sql
-- le revenu total de chaque abonnement par mois sans prendre en compte les réduction dues aux promotions


SELECT A.id_type_abonnement, UD.annee, UD.mois, SUM(nombre_abonnement*TA.tarif_dollar) AS revenu_total
FROM ABONNEMENT A
JOIN UNE_DATE UD ON A.id_date=UD.id_date
JOIN TYPE_ABONNEMENT TA ON TA.id_type_abonnement=A.id_type_abonnement
WHERE UD.jour = 01
GROUP BY A.id_type_abonnement, UD.annee, UD.mois;
```

- Le nombre, par jour, de chaque abonnement pour chaque type de promotion

(Évaluer à quelles périodes de l'année nous avons le + d'abonnés (et savoir lesquels en général sont le plus pris, pendant les périodes de fêtes notamment)

```sql
--Le nombre d'abonnements  par date de chaque abonnement pour chaque type de promotion

SELECT TA.nom , TP.nom, id_date, SUM(A.nombre_abonnement) AS total_abonnements
FROM ABONNEMENT A
JOIN TYPE_PROMOTION TP ON A.id_type_promotion=TP.id_type_promotion
 JOIN TYPE_ABONNEMENT TA ON A.id_type_abonnement=TA.id_type_abonnement
GROUP BY TA.nom , TP.nom, id_date
ORDER BY TA.nom, id_date;
```

- Le nombre d'abonnements par moyen de paiement

(Connaître les moyens de paiement les plus utilisés dans le but, par exemple, de plus sécuriser ceux qui sont le plus utilisés et de rendre plus accessible ceux qui le sont moins)

```sql
--Le nombre d'abonnements par moyen de paiement

SELECT MP.id_mode_paiment, SUM(A.nombre_abonnement) AS nombre_abonnements
FROM ABONNEMENT A
JOIN MODE_PAIMENT  MP ON A.id_mode_paiment = MP.id_mode_paiment
GROUP BY MP.id_mode_paiment
ORDER BY nombre_abonnements DESC;
```