



BestMarket

RetailInsight360

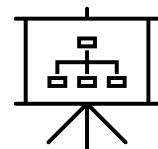
Base de données à l'usage du Service Client

1) Contexte et expression du besoin

BestMarket souhaite se doter d'une base regroupant ses retours clients afin de les exploiter plus facilement



Analyser



Exploiter



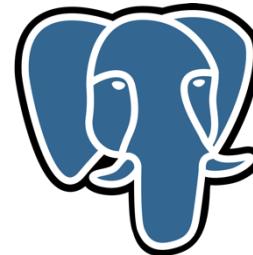
Déployer

2) Sauvegarde et stockage de la BDD

Deux sources de données :

- *Customers_data_Feedbacks*
- *ref_magasin*

Avant tout, il a fallu créer une Base commune afin de pouvoir stocker les données; mais surtout de pouvoir les exploiter plus facilement

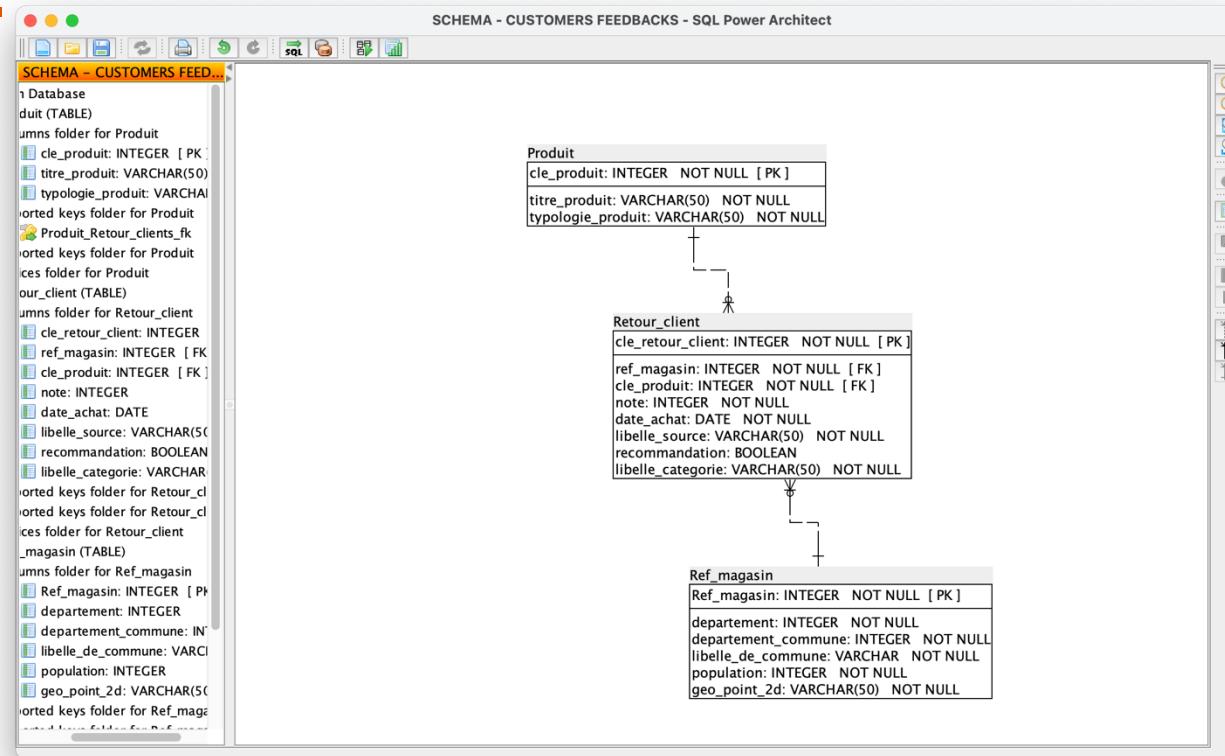


3) Méthodologie suivie

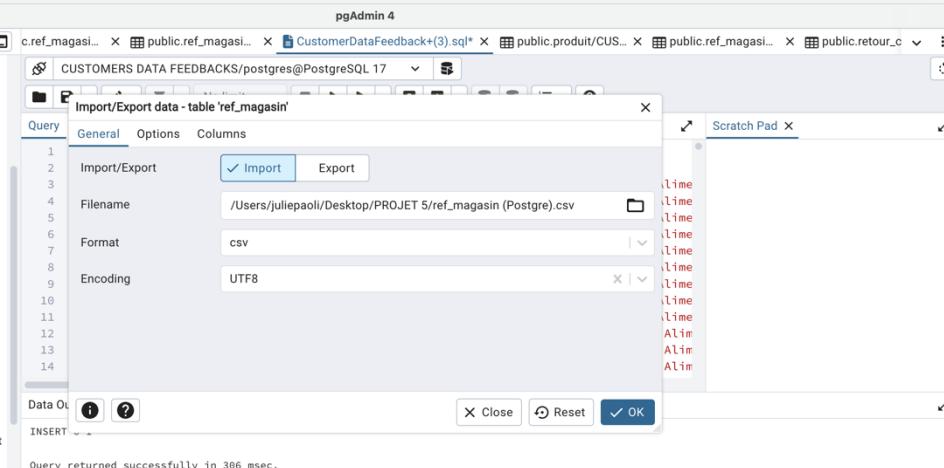
- A. Création du Schéma relationnel
- B. Importation des données

		Nom du champs	Type de données	Taille	Contrainte	Description
1	Table Retour client	cle_retour_client	INT		Clé primaire	ID unique pour les retours clients
2		note	INT			Note donnée par le client, comprise entre 0 et 10, la note est la réponse à la question : "Sur une échelle de 0 à 10 quelle est la probabilité que vous recommandiez notre entreprise à votre entourage ?"
3		Cle_produit	INT		Clé étrangère	ID des produits
4		ref_magasin	INT		Clé étrangère	ID des magasins
5		date_achat	DATE			Date à laquelle l'achat du client a eu lieu
6		libelle_source	CHAR	50		Libellé de la source d'où provient le retour client (Réseaux sociaux, téléphone, email)
7	Table Produit	libelle_categorie	CHAR	50		Libellé de la catégorie du retour client (Drive, service après-vente, qualité produit, expérience en magasin, livraison)
8		recommandation	CHAR			Recommandation laissée par le client à la question 'Recommandez vous l'entreprise?' True / False
9		cle_produit	INT		Clé primaire	ID unique pour les produits
10		titre_produit	CHAR	50		Libellé des produits
11	Table Ref_magasin	typologie_produit	CHAR	50		Typologie des produits (Alimentaire, High-tech etc...)
12		ref_magasin	INT		Clé primaire	ID des magasins
13		departement	INT			Numéro de département du magasin
14	Table Ref_magasin	departement_commune	INT			Code postal de la commune du magasin
15		libelle_de_commune	CHAR			Nom complet de la commune
16		population	INT			Population de la commune de résidence du magasin
17		geo_point_2d	CHAR	50		coordonnées géographiques du magasin
18						
19						

A. Crédation du Schéma



B. Importation des données



The screenshot shows the pgAdmin 4 interface with multiple tabs open at the top. The main area displays a SQL query in the "Query" tab:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

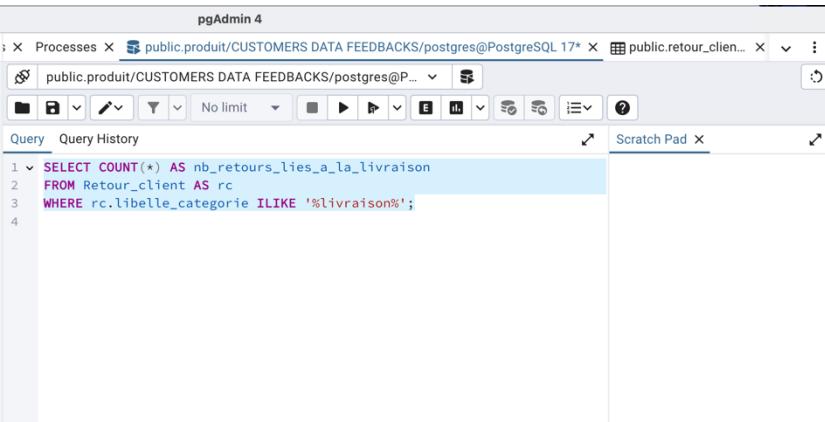
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (1, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (2, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (3, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (4, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (5, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (6, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (7, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (8, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (9, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (10, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (11, 'Alime
INSERT INTO produit (cle_produit, typologie_produit, titre_produit) VALUES (12, 'Alime

Below the query, the "Data Output" tab shows the message "Query returned successfully in 306 msec." with a status code "INSERT @ 1".

4) Requêtes SQL et Analyses

I. Indicateurs de Retour et Volume de Feedback

Mesurer le nombre de retours clients sur la livraison



```
pgAdmin 4
Processes public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* public.retour_clien...
public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... Scratch Pad

Query Query History
1 v SELECT COUNT(*) AS nb_retours_liés_a_la_livraison
2   FROM Retour_client AS rc
3 WHERE rc.libelle_catégorie ILIKE '%livraison%';
4

Data Output Messages Notifications
nb_retours_liés_a_la_livraison
bigint
1 639
```

Analyser le volume de retours clients par source



```
pgAdmin 4
Processes public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* public.retour_clien...
public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... Scratch Pad

Query Query History
1 v SELECT
2   rc.libelle_source AS source_du_retour,
3   COUNT(*) AS nombre_de_retours,
4   ROUND(
5     COUNT(*) * 100.0 / SUM(COUNT(*)) OVER(),
6     2
7   ) AS pourcentage_des_retours
8   FROM Retour_client AS rc
9   GROUP BY rc.libelle_source
10  ORDER BY nombre_de_retours DESC;
11

Data Output Messages Notifications
source_du_retour
character varying (50)
nombre_de_retours
bigint
pourcentage_des_retours
numeric
1 email 1032 34.40
2 réseaux sociaux 998 33.27
3 téléphone 970 32.33
```

Identifier les 5 magasins recevant le plus de feedbacks Repérer les magasins ayant plus de 12 feedbacks sur le drive

pgAdmin 4

```

public.retour_clien... public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* public.ref_magasi... public.ref_magasi... public.retour_clien...
public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... No limit ▶ E ⓘ ? Scratch Pad ×

Query Query History
1 v SELECT
2   rm.ref_magasin AS "Référence du magasin",
3   rm.libelle_de_commune AS magasin,
4   COUNT(*) AS nombre_de_retours,
5   STRING_AGG(DISTINCT rc.libelle_categorie, ', ') AS categories_de_retours
6   FROM retour_client AS rc
7   JOIN ref_magasin AS rm ON rc.ref_magasin = rm.ref_magasin
8   GROUP BY rm.ref_magasin, rm.libelle_de_commune
9   ORDER BY nombre_de_retours DESC
10  LIMIT 5;

```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

	Référence du magasin	magasin	nombre_de_retours	categories_de_retours
1	29	Mareuil-l'Es-Meaux	55	drive, expérience en magasin, livraison, qualité produit, service après-ve...
2	6	Osny	49	drive, expérience en magasin, livraison, qualité produit, service après-ve...
3	80	Lognes	47	drive, expérience en magasin, livraison, qualité produit, service après-ve...
4	5	Villecresnes	45	drive, expérience en magasin, livraison, qualité produit, service après-ve...
5	63	Irvy-sur-Seine	44	drive, expérience en magasin, livraison, qualité produit, service après-ve...

pgAdmin 4

```

public.retour_clien... public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* public.ref_magasi... public.ref_magasi... public.retour_clien...
public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... No limit ▶ E ⓘ ? Scratch Pad ×

Query Query History
1 v SELECT
2   rm.ref_magasin AS "Référence du magasin",
3   rm.libelle_de_commune AS magasin,
4   COUNT(*) AS nb_retours_drive
5   FROM retour_client AS rc
6   JOIN ref_magasin AS rm ON rc.ref_magasin = rm.ref_magasin
7   WHERE rc.libelle_categorie ILIKE '%drive%'
8   GROUP BY rm.ref_magasin, rm.libelle_de_commune
9   HAVING COUNT(*) > 12
10  ORDER BY nb_retours_drive DESC;
11

```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1

	Référence du magasin	magasin	nb_retours_drive
1	67	...ragny	14
2	63	Irvy-sur-Seine	13
3	45	Paris 12e Arrondissement	13

Trouver le pourcentage de recommandations clients afin de savoir si les clients sont satisfait et représentent une bonne proportion de promoteurs

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, it lists database objects under "public". It includes tables like "typologie_produ", "ref_magasin", and "retour_client".
- Query Editor:** The main window contains a SQL query to calculate the percentage of recommendations. The result is shown in the Data Output tab.
- Data Output:** The results of the query are displayed in a table:

pourcentage_de_recommandations_clients
70.50
- Status Bar:** At the bottom, it shows "Total rows: 1" and "Query complete 00:00:00.172".

```
SELECT
    ROUND(
        COUNT(CASE WHEN rc.recommandation = '1' THEN 1 END) * 100.0 / COUNT(*),
        2
    ) AS pourcentage_de_recommandations_clients
FROM retour_client AS rc;
```

II. Indicateurs de Satisfaction et Classements

Analyser les notes attribuées par les clients sur les réseaux sociaux concernant les TV

The screenshot shows the pgAdmin 4 interface with a query editor containing the following SQL code:

```
1 v SELECT
2   p.titre_produit AS nom_du_produit,
3   AVG(rc.note) AS note_moyenne,
4   COUNT(*) AS nombre_de_retours_reseaux_sociaux
5 FROM Retour_client AS rc
6 JOIN Produit AS p ON rc.cle_produit = p.cle_produit
7 WHERE rc.libelle_source ILIKE '%réseaux sociaux%'
8   AND p.titre_produit ILIKE 'TV'
9 GROUP BY p.titre_produit
10 ORDER BY note_moyenne DESC;
```

The results pane shows a single row of data:

nom_du_produit	note_moyenne	nombre_de_retours_reseaux_sociaux
TV	9.250000000000000	4

Calculer la note moyenne pour chaque catégorie de produit et établir un classement

The screenshot shows the pgAdmin 4 interface with a query editor containing the following SQL code:

```
1 v SELECT
2   p.typologie_produit AS categorie_de_produit,
3   ROUND(AVG(rc.note), 2) AS note_moyenne,
4   COUNT(*) AS nombre_de_retours
5 FROM Retour_client AS rc
6 JOIN Produit AS p ON rc.cle_produit = p.cle_produit
7 GROUP BY p.typologie_produit
8 ORDER BY note_moyenne DESC;
```

The results pane displays the following table:

categorie_de_produit	note_moyenne	nombre_de_retours
High-Tech	8.16	305
Loisirs	8.09	188
Alimentaire	8.04	2440
Maison	7.85	67

Identifier les 5 magasins ayant les meilleures notes moyennes

```
pgAdmin 4
public.retour_clien... X public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* X public.r...
public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... ▾
Query History
Query
Query History
1 SELECT
2     rm.ref_magasin AS "Référence du magasin",
3     rm.libelle_de_commune AS magasin,
4     ROUND(AVG(rc.note), 2) AS note_moyenne,
5     COUNT(*) AS nombre_de_retours
6 FROM retour_client AS rc
7 JOIN ref_magasin AS rm ON rc.ref_magasin = rm.ref_magasin
8 GROUP BY rm.ref_magasin, rm.libelle_de_commune
9 ORDER BY note_moyenne DESC
10 LIMIT 5;
11
12
```

Data Output Messages Notifications

Showing rows: 1 to 5

	Référence du magasin	magasin	note_moyenne	nombre_de_retours
1		75	Paris 14e Arrondissement	8.73
2		78	Saint-Pierre-du-Perray	8.55
3		62	Paris 19e Arrondissement	8.50
4		23	Paris 11e Arrondissement	8.48
5		19	Coulommiers	8.45

Repérer les magasins ayant une note inférieure à la moyenne (8.05)

```
pgAdmin 4
public.retour_clien... X public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* X public.ref_magasi...
public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... ▾
Query History
Query
Query History
1 WITH moyenne_globale AS (
2     SELECT AVG(note) AS moyenne_generale
3     FROM retour_client
4 )
5
6 SELECT
7     rm.ref_magasin AS "Référence du magasin",
8     rm.libelle_de_commune AS magasin,
9     ROUND(AVG(rc.note), 2) AS note_moyenne_du_magasin,
10    COUNT(*) AS nombre_de_retours
11   FROM retour_client AS rc
12  JOIN ref_magasin AS rm ON rc.ref_magasin = rm.ref_magasin
13  GROUP BY rm.ref_magasin, rm.libelle_de_commune
14  HAVING AVG(rc.note) < (SELECT moyenne_generale FROM moyenne_globale)
15  ORDER BY note_moyenne_du_magasin ASC;
```

Data Output Messages Notifications

Showing rows: 1 to 39 Page No: 1

	Référence du magasin	magasin	note_moyenne_du_magasin	nombre_de_retours
1		60	Buchelay	7.38
2		81	Nanterre	7.44
3		82	Montgeron	7.53
4		46	Paris 15e Arrondissement	7.56
5		55	Rosny-sous-Bois	7.59
6		24	Levallois-Perret	7.62
7		80	Lognes	7.62
8		8	Aubervilliers	7.66
9		44	Fontenay-sous-Bois	7.67

Total rows: 39 Query complete 00:00:00.125

Établir un classement des départements en fonction de leur note moyenne pour comparer les performances régionales.

Data Output		Messages		Notifications	
☰	File	Table	Bin	SQL	Showing rows: 1 to 8
	numero_departement integer	note_moyenne_du_departement numeric		nombre_de_retours bigint	
1		95		8.14	425
2		75		8.11	446
3		94		8.06	194
4		91		8.05	429
5		77		8.04	452
6		92		8.03	324
7		78		8.02	473
8		93		7.94	257

Déterminer la typologie de produit offrant le meilleur service après-vente en fonction des retours clients.

pgAdmin 4

public.retour_clien... X public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* X public.ref_ma

public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... ▾

No limit ▾

Query History

Query

```
1 ✓ SELECT
2     p.typologie_produit,
3     ROUND(AVG(rc.note), 2) AS note_moyenne
4 FROM retour_client AS rc
5 JOIN produit AS p ON rc.cle_produit = p.cle_produit
6 WHERE rc.libelle_catégorie ILIKE '%service après-vente%'
7 GROUP BY p.typologie_produit
8 ORDER BY note_moyenne DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 4

	typologie_produit character varying (50)	note_moyenne numeric
1	Loisirs	8.51
2	High-Tech	8.12
3	Alimentaire	8.03
4	Maison	7.88

Calculer la note moyenne sur l'ensemble des boissons



```
pgAdmin 4
public.retour_clien... X public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* X
public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... X

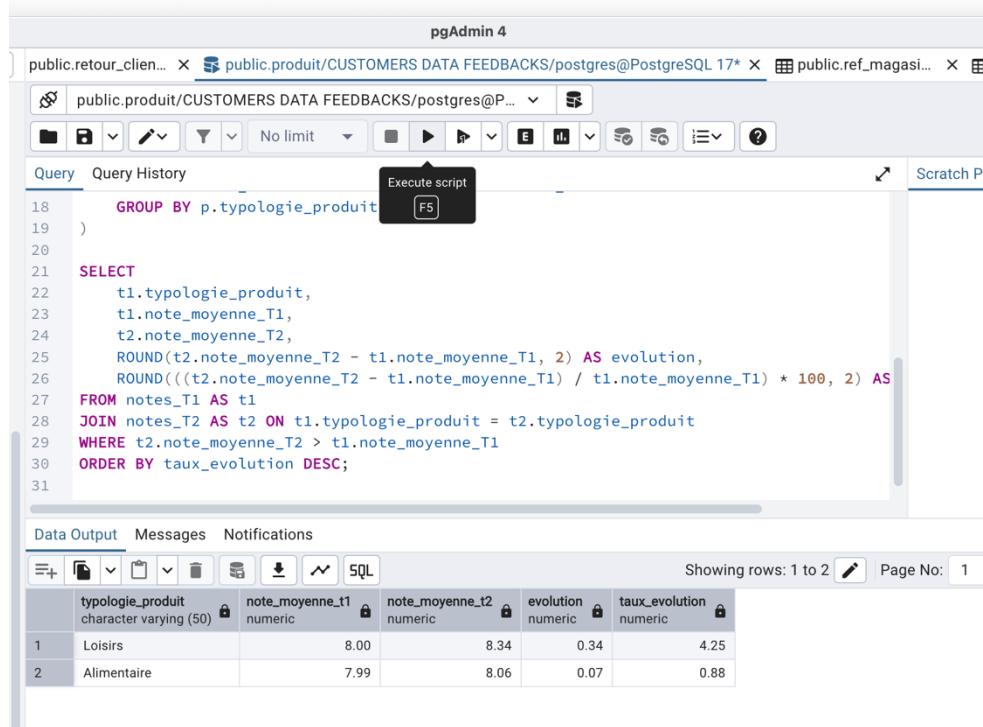
Query History
Query

1 SELECT
2   ROUND(AVG(rc.note), 2) AS note_moyenne_boissons,
3   COUNT(*) AS nombre_de_retours
4 FROM retour_client AS rc
5 JOIN produit AS p ON rc.cle_produit = p.cle_produit
6 WHERE p.titre_produit ILIKE '%boisson%'
7   OR p.titre_produit ILIKE '%café%'
8   OR p.titre_produit ILIKE '%thé%'
9   OR p.titre_produit ILIKE '%bière%'
10  OR p.titre_produit ILIKE '%jus%'
11  OR p.titre_produit ILIKE '%gazeuse%'
12  OR p.titre_produit ILIKE '%soda%';
13
14

Data Output  Messages  Notifications
Showing rows: 1 to 2
SQL

note_moyenne_boissons | nombre_de_retours
numeric              | bigint
1                   | 8.28
2                   | 207
```

Identifier les typologies de produits ayant amélioré leur note moyenne entre le 1er et le 2^e trimestre 2021



```
pgAdmin 4
public.retour_clien... X public.produit/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17* X
public.produit/CUSTOMERS DATA FEEDBACKS/postgres@P... X
public.ref_magasi... X

Query History
Query
Execute script F5

18   GROUP BY p.typologie_produit
19 )
20
21 SELECT
22   t1.typologie_produit,
23   t1.note_moyenne_T1,
24   t2.note_moyenne_T2,
25   ROUND(t2.note_moyenne_T2 - t1.note_moyenne_T1, 2) AS evolution,
26   ROUND(((t2.note_moyenne_T2 - t1.note_moyenne_T1) / t1.note_moyenne_T1) * 100, 2) AS
27   taux_evolution
28 FROM notes_T1 AS t1
29 JOIN notes_T2 AS t2 ON t1.typologie_produit = t2.typologie_produit
30 WHERE t2.note_moyenne_T2 > t1.note_moyenne_T1
31 ORDER BY taux_evolution DESC;
32

Data Output  Messages  Notifications
Showing rows: 1 to 2
Page No: 1
SQL

typologie_produit | note_moyenne_t1 | note_moyenne_t2 | evolution | taux_evolution
character varying(50) | numeric | numeric | numeric | numeric
1 | Loisirs | 8.00 | 8.34 | 0.34 | 4.25
2 | Alimentaire | 7.99 | 8.06 | 0.07 | 0.88
```

III. Indicateurs de Temporalité et Expérience Client

Classer les jours de la semaine où l'expérience client est la meilleure en magasin

The screenshot shows the pgAdmin 4 interface with a query editor and a results table.

Query Editor:

```
1 SELECT
2   TRIM(TO_CHAR(date_achat, 'Day')) AS jour_semaine,
3   ROUND(AVG(note), 2) AS note_moyenne,
4   COUNT(*) AS nombre_de_retours_exp_en_magasin
5 FROM Retour_client
6 WHERE libelle_categorie ILIKE '%expérience en magasin%'
7 GROUP BY jour_semaine
8 ORDER BY note_moyenne DESC;
```

Data Output:

	jour_semaine	note_moyenne	nombre_de_retours_exp_en_magasin
1	Saturday	8.34	73
2	Sunday	8.18	78
3	Friday	8.07	87
4	Thursday	8.04	75
5	Wednesday	7.99	76
6	Tuesday	7.95	86
7	Monday	7.74	81

Déterminer le mois comptabilisant le plus de retours sur le service après-vente

The screenshot shows the pgAdmin 4 interface with a query editor and a results table.

Query Editor:

```
1 SELECT
2   TO_CHAR(date_achat, 'YYYY-MM') AS mois,
3   COUNT(*) AS nombre_de_retours_sav
4 FROM Retour_client
5 WHERE libelle_categorie ILIKE '%service après-vente%'
6 GROUP BY mois
7 ORDER BY nombre_de_retours_sav DESC;
```

Data Output:

mois	nombre_de_retours_sav
2021-10	55
2021-09	53
2021-06	53
2021-05	52
2021-03	52
2021-08	52
2021-04	52
2021-11	52
2021-01	52

IV. Net Promoter Score (NPS)

NPS global : Mesurer la satisfaction client en soustrayant **NPS par source**
le pourcentage de détracteurs au pourcentage de promoteurs.

The image shows two pgAdmin 4 interface windows side-by-side, both connected to the same PostgreSQL database.

Left Window: This window displays a single SQL query to calculate the global NPS. The query uses CASE statements to count promoters (9-10), neutrals (6-8), and detractors (0-5) across all feedback sources. It then calculates the percentage of each category and the global NPS score.

```
1 SELECT
2     ROUND(100.0 * SUM(CASE WHEN note BETWEEN 9 AND 10 THEN 1 ELSE 0 END) / COUNT(*), 2) AS pourcentage_promoteurs,
3     ROUND(100.0 * SUM(CASE WHEN note BETWEEN 0 AND 6 THEN 1 ELSE 0 END) / COUNT(*), 2) AS pourcentage_detracteurs,
4     ROUND(
5         100.0 * SUM(CASE WHEN note BETWEEN 9 AND 10 THEN 1 ELSE 0 END) / COUNT(*) -
6         100.0 * SUM(CASE WHEN note BETWEEN 0 AND 6 THEN 1 ELSE 0 END) / COUNT(*),
7         2
8     ) AS nps_global
9 FROM Retour_client;
```

Right Window: This window displays a more complex SQL query to calculate the NPS by source. It first performs the same calculations as the left query to get the percentage of promoters and detractors for each source. Then, it groups the results by source, counts the number of returns for each source, and finally orders the results by the calculated NPS score in descending order.

```
1 SELECT
2     rc.libelle_source AS source_du_retour,
3     ROUND(100.0 * SUM(CASE WHEN note BETWEEN 9 AND 10 THEN 1 ELSE 0 END) / COUNT(*), 2) AS pourcentage_promoteurs,
4     ROUND(100.0 * SUM(CASE WHEN note BETWEEN 0 AND 6 THEN 1 ELSE 0 END) / COUNT(*), 2) AS pourcentage_detracteurs,
5     ROUND(
6         100.0 * SUM(CASE WHEN note BETWEEN 9 AND 10 THEN 1 ELSE 0 END) / COUNT(*) -
7         100.0 * SUM(CASE WHEN note BETWEEN 0 AND 6 THEN 1 ELSE 0 END) / COUNT(*),
8         2
9     ) AS nps_par_source,
10    COUNT(*) AS nombre_de_retours
11 FROM Retour_client AS rc
12 GROUP BY rc.libelle_source
13 ORDER BY nps_par_source DESC;
```

Data Output: Both windows show the results of their respective queries in a tabular format.

source_du_retour	pourcentage_promoteurs	pourcentage_detracteurs	nps_par_source	nombre_de_retours
1 téléphone	41.55	7.73	33.81	970
2 email	37.50	7.85	29.65	1032
3 réseaux sociaux	41.08	11.52	29.56	998

V. Autres requêtes

Identifier les 10 produits avec les meilleures notes et les 10 plus mauvaises

pgAdmin 4

gasi... x public.ref_magasin/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17*

```
1 SELECT
2     p.titre_produit AS nom_du_produit,
3     ROUND(AVG(rc.note), 2) AS note_moyenne,
4     COUNT(*) AS nombre_de_retours
5 FROM Retour_client AS rc
6 JOIN Produit AS p ON rc.cle_produit = p.cle_produit
7 GROUP BY p.titre_produit
8 ORDER BY note_moyenne DESC
9 LIMIT 10;
```

Data Output Messages Notifications

	nom_du_produit	note_moyenne	nombre_de_retours
1	Plantes aromatiques surgelées	9.20	20
2	Sodas	9.00	19
3	Boissons alcoolisées	8.79	19
4	Sauces au soja	8.77	22
5	TV	8.71	21
6	Aliments à base de plantes frais	8.71	21
7	Aliments à base de plantes séchées	8.68	19
8	Concentré de tomate	8.60	20
9	Petit-déjeuners	8.60	10
10	Plats au boeuf	8.57	21

pgAdmin 4

gasi... x public.ref_magasin/CUSTOMERS DATA FEEDBACKS/postgres@PostgreSQL 17*

```
1 SELECT
2     p.titre_produit AS nom_du_produit,
3     ROUND(AVG(rc.note), 2) AS note_moyenne,
4     COUNT(*) AS nombre_de_retours
5 FROM Retour_client AS rc
6 JOIN Produit AS p ON rc.cle_produit = p.cle_produit
7 GROUP BY p.titre_produit
8 ORDER BY note_moyenne ASC
9 LIMIT 10;
```

Data Output Messages Notifications

	nom_du_produit	note_moyenne	nombre_de_retours
1	Vinaigres	7.00	12
2	Produits à tartiner	7.05	20
3	Cuisine	7.06	18
4	Gâteaux et pâtisseries surgelés	7.15	13
5	Mangues au sirop	7.29	17
6	Vinaigres d'alcools	7.33	15
7	Pickles d'origine végétale	7.35	20
8	Haricots préparés	7.36	11
9	Légumineuses	7.38	24
10	Fruits à coques et dérivés	7.50	18

Classement des magasins dans les communes les plus peuplées

The screenshot shows the pgAdmin 4 interface with a query editor and a data output viewer.

Query Editor:

```
SELECT
    rm.ref_magasin AS "Référence du magasin",
    rm.libelle_de_commune AS commune,
    rm.population,
    ROUND(AVG(rc.note), 2) AS note_moyenne,
    COUNT(*) AS nombre_de_retours
FROM retour_client AS rc
JOIN ref_magasin AS rm ON rc.ref_magasin = rm.ref_magasin
GROUP BY rm.ref_magasin, rm.libelle_de_commune, rm.population
ORDER BY note_moyenne DESC;
```

Data Output:

	Référence du magasin integer	commune character varying	population integer	note_moyenne numeric	nombre_de_retours bigint
1	75	Paris 14e Arrondissement	138299	8.73	33
2	78	Saint-Pierre-du-Perray	8350	8.55	31
3	62	Paris 19e Arrondissement	186652	8.50	34
4	23	Paris 11e Arrondissement	153202	8.48	31
5	19	Coulommiers	14544	8.45	42
6	30	Taverny	26144	8.44	39
7	27	Colombes	85398	8.44	36
8	21	Longjumeau	21361	8.42	26
9	42	Sainte-Geneviève-des-Bois	34195	8.40	35

Total rows: 83 Query complete 00:00:00.132

5) Cohérence des données

- ❖ *Aucun résultat ne sort de l'ordinaire*
- ❖ *Les résultats se répondent et montrent une certaine logique*
- ❖ *Les NPS également*



MERCI POUR VOTRE ATTENTION