FALKNER Florian
Florian.falkner@etu.unistra.fr

TP OLAP
EDID

1. L'objectif était de modifier la table sales en ajoutant les **clés étrangères** et en enlevant les données en trop. J'ai aussi créé un deuxième schéma où la table **Period** est divisée en sous-tables pour que ça corresponde à un schéma en flocon de neiges.

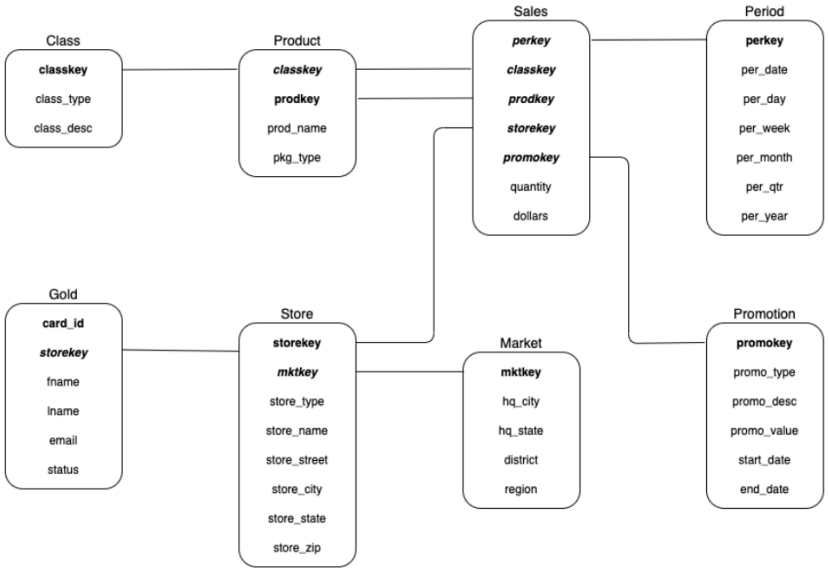

Schéma sans diviser

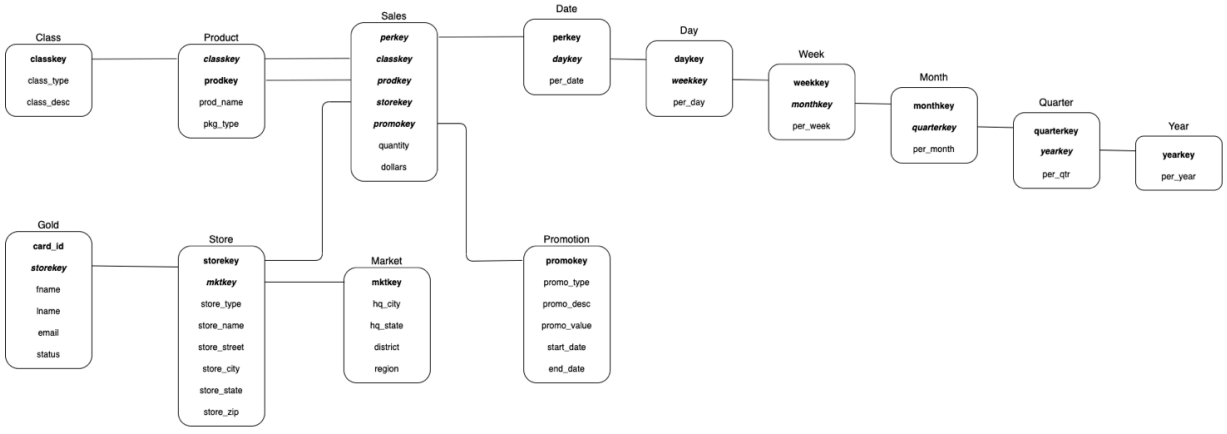

Schéma divisé

2.

```
SELECT s.classkey AS Classe, SUM(s.dollars) AS Total_Montant_Ventes
FROM HAKID.aroma_sales s, HAKID.aroma_class c, HAKID.aroma_product p
WHERE s.prodkey = p.prodkey
   AND s.classkey = p.classkey
   AND c.classkey = p.classkey
GROUP BY s.classkey
ORDER BY Total_Montant_Ventes DESC;
```

3.

```
SELECT DISTINCT s.classkey AS Classe, ROUND((SUM(s.dollars) OVER(PARTITION BY s.classkey) / SUM(s.dollars) OVER()) * 100) AS Pourcentage
FROM HAKID.aroma_sales s, HAKID.aroma_class c, HAKID.aroma_product p
WHERE s.prodkey = p.prodkey
   AND s.classkey = p.classkey
   AND c.classkey = p.classkey
ORDER BY Pourcentage DESC;
```

4.

```
SELECT s.classkey AS Classe, SUM(s.dollars) AS Total_Montant, RANK() OVER (ORDER BY SUM(s.dollars) DESC) as ClassementVentes
FROM HAKID.aroma_sales s, HAKID.aroma_class c, HAKID.aroma_product p
WHERE s.prodkey = p.prodkey
   AND s.classkey = p.classkey
   AND c.classkey = p.classkey
GROUP BY s.classkey;
```

C’est la suite de la 3.

5.

```
SELECT *
FROM (
        SELECT DISTINCT p.prod_name, ROUND((SUM(s.quantity) OVER(PARTITION BY p.prod_name) / SUM(s.quantity) OVER() * 100), 2) AS Pourcentage
   FROM HAKID.aroma_sales s, HAKID.aroma_class c, HAKID.aroma_product p
   WHERE p.prodkey = s.prodkey
      AND p.classkey = s.classkey
      AND c.classkey = p.classkey
   ORDER BY Pourcentage DESC
   )
WHERE ROWNUM <= 1;
```

Cette requête ci-dessus je l’ai compris de cette façon: Afficher uniquement le produit qui a été vendu le plus de fois avec son pourcentage. Au début je voulais utiliser la commande FETCH FIRST 1 ROW ONLY ou LIMIT mais la version ORACLE du serveur de l’unistra n’est pas à jour pour l’utiliser. C’est pour cela que j’ai utilisé une sous-requête.

6.

```
SELECT p.prod_name as Produit, per.per_week AS Semaine, sum(s.dollars) AS MontantVentesProduits
FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
WHERE s.prodkey = p.prodkey
   AND s.classkey = p.classkey
   AND s.perkey = per.perkey
GROUP BY per.per_week, p.prod_name
ORDER BY per.per_week;
```

Pour cette requête j'avais un doute. Moi j'ai choisi d'afficher pour chaque produit son montant total de ventes par semaine (vous me l'avez confirmé sur BBB). Car l'autre doute que j'avais c'était peut-être d'afficher uniquement le montant total des ventes de tous les produits par semaine. Donc avoir pour chaque semaine le montant total direct sans regarder combien a rapporté le produit X dans cette semaine.

7.

```
SELECT p.prod_name AS Produit, per.per_week AS Semaine, sum(s.dollars) AS MontantVentesProduits, RANK() OVER(PARTITION BY per.per_week ORDER BY SUM(s.dollars) DESC) as ClassementVentes
FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
WHERE s.prodkey = p.prodkey
   AND s.classkey = p.classkey
   AND s.perkey = per.perkey
GROUP BY per.per_week, p.prod_name
ORDER BY per.per_week ASC, MontantVentesProduits DESC, ClassementVentes DESC;
```

Basée sur la requête 6.

8.

```
SELECT Produit, ROUND(AVG(ClassementVentes)) AS ClassementMoyen
FROM (
        SELECT p.prod_name AS Produit, per.per_week AS Semaine, sum(s.dollars) AS MontantVentesProduits, RANK() OVER(PARTITION BY per.per_week ORDER BY SUM(s.dollars) DESC) as ClassementVentes
   FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
   WHERE s.prodkey = p.prodkey
      AND s.classkey = p.classkey
      AND s.perkey = per.perkey
   GROUP BY per.per_week, p.prod_name
   ORDER BY per.per_week ASC, MontantVentesProduits DESC, ClassementVentes DESC
   )
GROUP BY Produit
ORDER BY ClassementMoyen;
```

Basée sur la requête 6.

9.

Pour cette dernière requête, j'avais un doute concernant la phrase. Je n'ai pas bien compris si c'était demandé d'afficher uniquement la moyenne du total des ventes pour chaque jour de la semaine (voir requête sur cette page), je pense que c'est cette requête que vous demandez.

Ou d'afficher pour chaque produit sa moyenne de ventes pour chaque jour de la semaine (voir requête sur la prochaine page). Pour pas me pénaliser je vous ai mis les 2 requêtes :)

```
SELECT
   ROUND(SUM(CASE WHEN per.per_day = 'MO'
      then (s.dollars/53/7) else 0 end), 2) AS Lundi,
   ROUND(SUM(CASE WHEN per.per_day = 'TU'
      then (s.dollars/53/7) else 0 end), 2) AS Mardi,
   ROUND(SUM(CASE WHEN per.per_day = 'WE'
      then (s.dollars/53/7) else 0 end), 2) AS Mercredi,
   ROUND(SUM(CASE WHEN per.per_day = 'TH'
      then (s.dollars/53/7) else 0 end), 2) AS Jeudi,
   ROUND(SUM(CASE WHEN per.per_day = 'FR'
      then (s.dollars/53/7) else 0 end), 2) AS Vendredi,
   ROUND(SUM(CASE WHEN per.per_day = 'SA'
      then (s.dollars/53/7) else 0 end), 2) AS Samedi,
   ROUND(SUM(CASE WHEN per.per_day = 'SU'
      then (s.dollars/53/7) else 0 end), 2) AS Dimanche
FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
WHERE s.prodkey = p.prodkey
      AND s.classkey = p.classkey
      AND s.perkey = per.perkey;
```

```
SELECT t1.prod_name, Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche
FROM (SELECT p.prod_name, per.per_day, SUM(ROUND(SUM(s.dollars)/53/7, 2)) OVER(PARTITION BY p.prod_name, per.per_day ORDER BY p.prod_name ROWS UNBOUNDED PRECEDING) AS Lundi
      FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
      WHERE s.prodkey = p.prodkey
        AND s.classkey = p.classkey
        AND s.perkey = per.perkey
        AND per.per_day = 'MO'
      GROUP BY p.prod_name, per.per_day
      ORDER BY p.prod_name ASC) t1
JOIN
          (SELECT p.prod_name, per.per_day, SUM(ROUND(SUM(s.dollars)/53/7, 2)) OVER(PARTITION BY p.prod_name, per.per_day ORDER BY p.prod_name ROWS UNBOUNDED PRECEDING) AS Mardi
      FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
      WHERE s.prodkey = p.prodkey
        AND s.classkey = p.classkey
        AND s.perkey = per.perkey
        AND per.per_day = 'TU'
      GROUP BY p.prod_name, per.per_day
      ORDER BY p.prod_name ASC) t2
ON t1.prod_name = t2.prod_name
JOIN
          (SELECT p.prod_name, per.per_day, SUM(ROUND(SUM(s.dollars)/53/7, 2)) OVER(PARTITION BY p.prod_name, per.per_day ORDER BY p.prod_name ROWS UNBOUNDED PRECEDING) AS Mercredi
      FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
      WHERE s.prodkey = p.prodkey
        AND s.classkey = p.classkey
        AND s.perkey = per.perkey
        AND per.per_day = 'WE'
      GROUP BY p.prod_name, per.per_day
      ORDER BY p.prod_name ASC) t3
ON t1.prod_name = t3.prod_name
JOIN
          (SELECT p.prod_name, per.per_day, SUM(ROUND(SUM(s.dollars)/53/7, 2)) OVER(PARTITION BY p.prod_name, per.per_day ORDER BY p.prod_name ROWS UNBOUNDED PRECEDING) AS Jeudi
      FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
      WHERE s.prodkey = p.prodkey
        AND s.classkey = p.classkey
        AND s.perkey = per.perkey
        AND per.per_day = 'TH'
      GROUP BY p.prod_name, per.per_day
      ORDER BY p.prod_name ASC) t4
ON t1.prod_name = t4.prod_name
JOIN
          (SELECT p.prod_name, per.per_day, SUM(ROUND(SUM(s.dollars)/53/7, 2)) OVER(PARTITION BY p.prod_name, per.per_day ORDER BY p.prod_name ROWS UNBOUNDED PRECEDING) AS Vendredi
      FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
      WHERE s.prodkey = p.prodkey
```

```
        AND s.classkey = p.classkey
        AND s.perkey = per.perkey
        AND per.per_day = 'FR'
    GROUP BY p.prod_name, per.per_day
    ORDER BY p.prod_name ASC) t5
ON t1.prod_name = t5.prod_name
JOIN
        (SELECT p.prod_name, per.per_day, SUM(ROUND(SUM(s.dollars)/53/7, 2))
OVER(PARTITION BY p.prod_name, per.per_day ORDER BY p.prod_name ROWS
UNBOUNDED PRECEDING) AS Samedi
    FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
    WHERE s.prodkey = p.prodkey
        AND s.classkey = p.classkey
        AND s.perkey = per.perkey
        AND per.per_day = 'SA'
    GROUP BY p.prod_name, per.per_day
    ORDER BY p.prod_name ASC) t6
ON t1.prod_name = t6.prod_name
JOIN
        (SELECT p.prod_name, per.per_day, SUM(ROUND(SUM(s.dollars)/53/7, 2))
OVER(PARTITION BY p.prod_name, per.per_day ORDER BY p.prod_name ROWS
UNBOUNDED PRECEDING) AS Dimanche
    FROM HAKID.aroma_sales s, HAKID.aroma_product p, HAKID.aroma_period per
    WHERE s.prodkey = p.prodkey
        AND s.classkey = p.classkey
        AND s.perkey = per.perkey
        AND per.per_day = 'SU'
    GROUP BY p.prod_name, per.per_day
    ORDER BY p.prod_name ASC) t7
ON t1.prod_name = t7.prod_name;
```